```
{ Rum.dfn

Definitions for Rum, the Dandelion Smalltalk-80 microcoded virtual machine.

        by P McCullough, J Trow, J Susser, M Udagawa, T Tokunaga

        20-Jan-86 20:57:19

Copyright 1983, 1984, 1985, 1986 by Xerox Corporation.  All rights reserved. }



{+++++ Control store definitions +++++}

        SetTask [0];


        Reserve [0F6F];
        Reserve [0F77];
        Reserve [0F7F, 0FFF];   {Save space for the Kernel}


{ "bytecode" macro utilized to build the bytecode dispatch table}

        MacroDef [bytecode, at [Add [#1, bytecodeBase]]];

        Set [bytecodeBase, 800] {as good a place as any};


{The following macro dispatches to the next bytecode interpreter and sets L3 to zero}

        MacroDef [NextBytecode, (IBDisp, L3 ← 0)];



{ Definitions for setting the control store bank register }

        Set [memoryState0, 0];          {display banks are 0, 10, 80, 90}
        Set [memoryState1, 4];          {display banks are 0, 10}
        Set [memoryState2, 8];          {display bank is 0}
        Set [memoryState3, 0C];         {display banks are 0, 10-1F, 90-9F}

        Set [csBank0, Add [0, memoryState1]];
        Set [csBank1, Add [1, memoryState1]];
        Set [csBank2, Add [2, memoryState1]];



{ Definitions of known Pilot stuff }

        Set [mouseBank, 2];
        Set [mouseX,    3B];    {must be less than 100}
        Set [mouseY,    3C];    {must be less than 100}



        Set [needToStabilize,           1];

        Set [notYetInvented,            1];
        Set [bytecodeFailed,            2];




{+++++ Hardware register definitions +++++}

{register equates for the fourteen R and RH interpreter registers}

        RegDef [ipHigh,         RH, 0];
        RegDef [ipLow,          R,  0];

        RegDef [stackHigh,      RH, 1];
        RegDef [stackLow,       R,  1];

        RegDef [homeHigh,       RH, 2];
        RegDef [homeLow,        R,  2];

        RegDef [otHigh,         RH, 3];
        RegDef [otLow,          R,  3];
```

```
        RegDef [temp1High,      RH, 4];
        RegDef [temp1Low,       R,  4];

        RegDef [temp2High,      RH, 5];
        RegDef [temp2Low,       R,  5];

        RegDef [temp3High,      RH, 6];
        RegDef [temp3Low,       R,  6];


        RegDef [objectSize,     R,  0]; {for compactor}
        RegDef [oopCount,       R,  0]; {for compactor}

        RegDef [sourceHigh,     RH, 1]; {for compactor}
        RegDef [sourceLow,      R,  1]; {for compactor}
        RegDef [otIndex,        R,  1]; {for compactor}
        RegDef [wordCountHigh,  RH, 1]; {for compactor}
        RegDef [wordCountLow,   R,  1]; {for compactor}

        RegDef [rumHigh,        RH, 2]; {for compactor}
        RegDef [rumLow,         R,  2]; {for compactor}

        RegDef [destHigh,       RH, 4]; {for compactor}
        RegDef [destLow,        R,  4]; {for compactor}

        RegDef [objectHigh,     RH, 5]; {for compactor}
        RegDef [objectLow,      R,  5]; {for compactor}

{Register equates for saving the address of the Rum Communications Record  -- these registers are completely unused by Mesa. This is
essential if we are to continue Rum microcode after going to the Mesa debugger (or any other non-smalltalk XDE volume)}

        RegDef [uRumRecordHigh,         U, 49];
        RegDef [uRumRecordLow,          U, 5F];



{U register definitions}

        RegDef [uLargeIntegerValueHigh, U, 14];
        RegDef [uLargeIntegerValueLow,  U, 17];

{       RegDef [u3FFF,                  U, 15]; {** constant initialized by Mesa}}

        RegDef [uAtPutValue,            U, 19];

        RegDef [uZctBaseHigh,           U, 1a];

        RegDef [uZctSweepHigh,          U, 1b]; {used in stabilizer}
        RegDef [uZctSweepLow,           U, 1c]; {used in stabilizer}

        RegDef [uZctBaseLow,            U, 1d]; {used in stabilizer}

        RegDef [uAtPutLow,              U, 1e];

        RegDef [uTimeToStabilize,       U, 1f]; {reserved by stabilizer}

        RegDef [uAtIndex,               U, 24];

{       RegDef [u1FFF,                  U, 36]; {** constant initialized by Mesa}}

{       RegDef [u7FF,                   U, 2F]; {** constant initialized by Mesa}}

{       RegDef [MesaStateL,             U, 30];}        {&&}

        RegDef [uMakeVolatileOop,       U, 31]; {used in makeVolatile}
        RegDef [uNewObject,             U, 31]; {used in createInstance; referenced in activateNewMethod}
        RegDef [uQueueHead,             U, 31]; {used in stabilizer}

{       RegDef [MesaStateRhL,           U, 33];}        {&&}

        RegDef [uClassToInstantiate,    U, 34]; {input parameter for createInstance}
        RegDef [uFieldType,             U, 35]; {used in createInstance}

        RegDef [uReceiverOop,           U, 2E]; {smashed in activateNewMethod}

        RegDef [uDefault,               U, 37]; {used in createInstance}
        RegDef [uRequestedSize,         U, 38]; {used in createInstance}

        RegDef [uNewReceiver,           U, 3a];

        RegDef [uNewMethodOop,          U, 3b]; {referenced in activateNewMethod}

        RegDef [uNewContextOop,         U, 3c]; {smashed in activateNewMethod}
        RegDef [uPredecessor,           U, 3c]; {used in createInstance}
        RegDef [uCurrentObject,         U, 3c]; {used in stabilizer}


        RegDef [uNewClassHigh,          U, 42];
        RegDef [uNewClassLow,           U, 43];

        RegDef [uNewObjectHigh,         U, 42]; {used in makeVolatile}
        RegDef [uNewObjectLow,          U, 43]; {used in createInstance}


        RegDef [uMakeVolatileHigh,      U, 44]; {used in makeVolatile}
```

```
        RegDef [uCurrentObjectBaseLow,  U, 44]; {used in deallocate}

{       RegDef [MesaStateG,             U, 45];}        {&&}
{       RegDef [MesaStateRhG,           U, 46];}        {&&}

        RegDef [uMakeVolatileLow,       U, 47]; {used in makeVolatile}
        RegDef [uSoFar,                 U, 47]; {used in deallocate}


        RegDef [uWrap,                  U, 48];
        RegDef [uSoFarHigh,             U, 48]; {used in deallocate}
        RegDef [uSaveHome,              U, 48];


        RegDef [uArgumentCount,         U, 4a];
        RegDef [uOtHigh,                U, 4a];

        RegDef [uNewReceiversClass,     U, 4b];
        RegDef [uMemLimitHigh,          U, 4b];

        RegDef [uNewReceiverHigh,       U, 4c];
        RegDef [uDestLow,               U, 4c];

        RegDef [uNewMethodHigh,         U, 4d];
        RegDef [uNewMethodLow,          U, 4e];

        RegDef [uNewReceiverLow,        U, 4f];

{       RegDef [MesaStatePC,            U, 50];}        {&&}
{       RegDef [MesaStateRhPC,          U, 51];}        {&&}

{       RegDef [MesaStatePC16,          U, 52];}        {&&}
{       RegDef [MesaStateIBPtr,         U, 53];}        {&&}
{       RegDef [MesaStateIB,            U, 54];}        {&&}

        RegDef [uHash,                  U, 55];

        RegDef [uMethodCacheHigh,       U, 56]; {&&}
        RegDef [uMethodCacheLow,        U, 57]; {&&}

        RegDef [uSelectorsStartInDictionary,    U, 58];

        RegDef [uSmashTos,              U, 59];

        RegDef [uZctLimit,              U, 59]; {used in stabilizer}

        RegDef [uActiveContextHigh,     U, 5a]; {referenced in activateNewMethod}
        RegDef [uActiveContextLow,      U, 5b]; {referenced in activateNewMethod}

        RegDef [uPrimitiveNumber,       U, 5c];

        RegDef [uNewMethodHeader,       U, 5d];  {referenced in activateNewMethod}

        RegDef [uNextFreeChunk,         U, 60]; {used in create instance}
        RegDef [uClass,                 U, 60]; {used in deallocator}

        RegDef [uSelector,              U, 63];

        RegDef [uHomeLow,               U, 64]; {&&}

{       RegDef [MesaStateRhMDS,         U, 65];}        {&&}

        RegDef [uCurrentMethodHigh,     U, 66]; {smashed in activateNewMethod}
        RegDef [uCurrentMethodLow,      U, 67]; {referenced in activateNewMethod}

        RegDef [uReceiverHigh,          U, 68]; {smashed in activateNewMethod}
        RegDef [uReceiverLow,           U, 69]; {smashed in activateNewMethod}

        RegDef [uHashedSelector,        U, 6a];

        RegDef [uLastPointer,           U, 6b]; {used in makeVolatile}

        RegDef [uActiveContextOop,      U, 6c]; {referenced in activateNewMethod}

        RegDef [uReturnValue,           U, 6d];

        RegDef [uCurrentFreeChunkOop,   U, 6d]; {used in create instance}

        RegDef [uMakeVolatileLinkage,   U, 6e]; {used in makeVolatile}

        RegDef [uStartLookup,           U, 6f];


{Float:}
{Register Definition}
        RegDef [uArgValueHigh,          U, 14],
        RegDef [uArgValueLow,           U, 17],
        RegDef [uResultSize,            U, 19],
        RegDef [uLargeReceiverHigh,     U, 1A],
        RegDef [uLargeReceiverLow,      U, 1E],

        RegDef [uResult0,       U, 2 {24}],
        RegDef [uResult1,       U, 3 {3A}],
        RegDef [uResult2,       U, 4 {3B}],
        RegDef [uResult3,       U, 5 {4A}],

{BankPlug return adress U register}
```

```
        RegDef [uReturnAdSave,  U, 6 {4F}],

{qFetchByte subroutine working register}
        RegDef [uSave,          U, 7 {4D}],

{BankPlug return address U register}
        RegDef [uReturnBank0,   U, 8 {4E}],



        RegDef [floatT,         R, 2];  { temporary for float dividing }
        RegDef [floatTemp,      R, 3];
        RegDef [divCount,       R, 3];  { temporary for float dividing  }
        RegDef [divResult,      R, 4];  { temporary for float dividing  }
        RegDef [divisorHigh,    R, 5];  { temporary for float dividing  }
        RegDef [divisorLow,     R, 6];  { temporary for float dividing  }


        RegDef [uArg1Hi,                U, 14]; { used in divide or compare }
        RegDef [uArg1Lo,                U, 17]; { used in devide or compare }

        RegDef [uHighHalf1,             U, 19]; { divide }
        RegDef [uExp1,                  U, 1A]; { divide }
        RegDef [uLowHalf1,              U, 1B]; { divide }
        RegDef [uExp2,                  U, 1C]; { divide }
        RegDef [uHighHalf2,             U, 1D]; { divide }
        RegDef [uLowHalf2,              U, 1E]; { divide }

        RegDef [uStickyReg,             U, 24]; { divide }

        RegDef [uStickyBit,             U, 31]; { divide }
        RegDef [uSign1,                 U, 34]; { divide }
        RegDef [uSign2,                 U, 35]; { divide }
        RegDef [uFloatComResult,        U, 37]; { }
        RegDef [uPPsave,                U, 3C];

        RegDef [uFloat,                 U, 42]; { AsFloat }
        RegDef [uSaveIPLow,             U, 43]; { to save the ipLow during float divide primitive}
        RegDef [uFloatPlus,             U, 44]; { Float Add }
        RegDef [uSaveStackLow,          U, 44]; { to save the stacklow during float divide primitive}

        RegDef [uNewValHi,              U, 4B]; { Result of the floating point operatio nis stored -- High half }
        RegDef [uNewValLo,              U, 4C]; { Result of the floating point operation is stored.-- Low half }

        RegDef [uFloatMinus,            U, 55]; { Float Subtract : A-B}
        RegDef [uSaveHomeLow,           U, 55]; { to save the homeLow during float divide primitive}
        RegDef [uFloatMode,             U, 58]; { }
        RegDef [uArg2Hi,                U, 5C]; { }
        RegDef [uArg2Lo,                U, 5D]; { }

        RegDef [uFloatMultiply,         U, 60];         { Multiply}
        RegDef [uFloatFix,              U, 63]; { }



{Process:}
{Ureg definitions}
        RegDef[uVeryTemp1, U, 25];
        RegDef[uVeryTemp2, U, 27];
        RegDef[ustack, U, 14];
        RegDef[uhome, U, 17];



{+++++ Microcode subroutine return link definitions +++++}


{It would be nice if Mass allowed one to define names for L registers, but it doesn't.}

{
Link registers are used as follows:

        L0 tracks the number of bytecodes to back up the PC when a bytecode fails
                and is return link for fetchContextRegisters and stabilize.

        L1 is the return link for otMap, otMap2, otMapBank0, arithmetic primitives,
                refi, refd, getDeltaWord, returnTopOfStack, transferWords, getTos, getSmashTos,
                addToFreeChunkList, and is used in conditional branch bytecodes.

        L2 is the return link for getClass, getNewMethodHeader, lastPointerOf,
                addToZeroCountTable, nextFreeChunk, primRefi, and fixedFieldsOf.

        L3 is the return link for primitives, makeVolatile, deallocate,
                and is set to zero at each bytecode dispatch.
}

{
It would be nice if Mass macros could appear on the left side of ← clauses, but they can't, so we define some macros to keep track of the
number of bytes to back up when an instruction fails}
```

```
        MacroDef [backupIs0Bytes, L0 ← 0];
        MacroDef [backupIs1Byte,  L0 ← 1];
        MacroDef [backupIs2Bytes, L0 ← 2];
        MacroDef [backupIs3Bytes, L0 ← 3];


{ Definitions for otMap returns.  L1 is the return link. }

        Set [gettingSmalltalkState,             0];
        Set [isOopGettingSmalltalkState,        1];
        Set [isSmallGettingSmalltalkState,      2];
        Set [pushingLiteralVariable,            3];
        Set [gettingClass,                      4];
        Set [gettingMethodBase,                 5];
        Set [makingVolatile,                    6];
        Set [primStringAt,                      7];
        Set [methodSearch,                      8];
        Set [messageDictionary,                 9];
        Set [methodArray,                       0A];
        Set [methodBaseAfterLookup,             0B];
        Set [saveSuperclass,                    0C];
        Set [getMethodClass,                    0D];
        Set [getSuperclass,                     0E];
        Set [storingLiteralVariable,            0F];


{ Definitions for otMap2 returns.  L1 is the return link. }

        Set [gettingNextFreeChunk,                      0];
        Set [receiverDuringFetch,                       1];
        Set [methodDuringFetch,                         2];
        Set [gettingActiveContextDuringInterpreterSwap, 3];
        Set [getSenderBase,                             4];
        Set [chasingContextChain,                       5];
        Set [invalidatingContext,                       6];
        Set [gettingSpecialSelectors,                   7];
        Set [instanceSpec,                              8];
        Set [splicingBigFreeList,                       9];
        Set [stabilizing,                               0A];
        Set [sweepingObject,                            0B];
        Set [largeIntegerResult,                        0C];
        Set [blockCopyOfBlock,                          0D];
        Set [countBitsLargeInt,                         0E];
        Set [nextInstance,                              0F];

{Defs for otMap3-return}
        Set[mappingScheduler, 0];
        Set[mappingProcessor, 1];
        Set[mappingProcessLists, 2];
        Set[mappingHighestPriority, 3];
        Set[removingLink, 4];
        Set[mappingLastLink, 5];
        Set[addingLink, 6];
        Set[mappingActProc, 7];
        Set[mappingNewProc, 8];
        Set[mappingActProcResume, 9];
        Set[mappingResumeProcess, 0a];
        Set[mappingResumeProcLists, 0b];
        Set[mappingSleepList, 0c];
        Set[mappingPostponeList, 0d];


{ Definitions for otMapBank0 returns.  L1 is the return link. }

        Set [gettingClassBank0,         0];
        Set [getFloatReceiver,          1];
        Set [floatReturn,               2];
        Set [getUnaryFloatReceiver,     3];
        Set [getTimesTwoPowerReceiver,  8];
        Set [makeInstLargeInt,          9];


{ Definitions for otMap1Bank0 returns.  L1 is the return link. }

        Set [getParameterAfterMInt,             0];
        Set [getDestMapAfterMInt,               1];
        Set [getHalftoneBitsAfterMInt,          2];
        Set [getHalftoneBitsAfterMInt1,         3];
        Set [getParameter,                      4];
        Set [getSFormWH,                        5];
        Set [getHalftoneBits,                   6];
        Set [getHalftoneBits1,                  7];
        Set [getCursorMap,                      8];
        Set [getMap,                            9];
        Set [getMapBase,                        0A];


{ Definitions for getOtAddress and putOtAddress returns.  L1 is the return link. }

        Set [splittingFreeChunk,        0];
        Set [countingOops,              1];
        Set [compactorCreateFreeChunk,  9];
        Set [countingWords,             0A];
```

```
{ Definitions for getOtFlags and putOtFlags returns.  Temp2High is the return link. }

        {splittingFreeChunk,              0}
    Set [newObjectHeader,                 1];
    Set [doingNormalRefi,                 2];
    Set [addingToZct,                     4];
    Set [volatizing,                      5];
    Set [deallocating,                    6];
    Set [doingSpecialRefd,                7];
    Set [returningToPool,                 8];
        {compactorCreateFreeChunk,        9}
        {countingWords,                   0A}
    Set [doingNormalRefd2,                0D];    {must be odd for skipRefd ??}
    Set [doingNormalRefd,                 0F];    {must be odd for skipRefd}


{ Definitions for convert2SmallIntegers returns. L1 is the return link and the SpecialSelectors index for primitives that are also
bytecodes. }
    Set [smallLess,          2];     {bytecode}
    Set [smallGreater,       3];     {bytecode}
    Set [smallLessOrEqual,   4];     {bytecode}
    Set [smallGreaterEqual,  5];     {bytecode}
    Set [smallMultiply,      8];     {bytecode}
    Set [smallDivide,        9];     {bytecode}
    Set [smallMod,           0A];    {bytecode}
    Set [smallMakePoint,     0B];    {bytecode}
    Set [smallBitShift,      0C];    {bytecode}
    Set [smallDiv,           0D];    {bytecode}
    Set [smallQuo,           0E];


{ Definitions for primitiveTest2SmallIntegers returns.  L1 is the return link and the SpecialSelectors index for primitives that are also
bytecodes. }
    Set [smallAdd,           0];     {bytecode}
    Set [smallSubtract,      1];     {bytecode}
    Set [smallEqual,         6];     {bytecode}
    Set [smallNotEqual,      7];     {bytecode}
    Set [smallBitXor,        8];
    Set [smallBitAnd,        0E];    {bytecode}
    Set [smallBitOr,         0F];    {bytecode}


{definitions for getClass return links. L2 is the return link register}
  {these constants are used in bank 1 }
    Set [primitiveClass,             0];
    Set [pos16Bit,                   1];
    Set [startingDeallocate,         2];
    Set [directBlockCopy,            3];
    Set [directValue,                4];
    Set [primStringAtPut,            5];
    Set [gettingNewReceiversClass,   6];
    Set [superclassReceiver,         7];
    Set [primitivePerform,           8];
    Set [lookingForInstances,        9];


{definition for getClassBank0 return links. L2 is the return link register }
  {these constants are used in bank 0  }
    Set [primNeeds2LargeIntegers1,  0];     { for LargeInteger }
    Set [primNeeds2LargeIntegers2,  1];     { for LargeInteger }
    Set [getFloatClass,             2];     { for Floating Point }
    Set [fetch2WordVal,             3];     { for LargeInteger }
    Set [fetch2WordValPos,          4];     { for LargeInteger }


{definitions for getNewMethodHeader return links. L2 is the return link register}
    Set [foundViaCache,     0];
    Set [foundViaLookup,    1];
    Set [gettingSuperclass, 2];


{definitions for conditional branch bytecodes}
    Set [branchIfTrue,      1];
    Set [branchIfFalse,     0];


{definitions for refi. L1 is the return link register}
    Set [popAndStoreRecVariable,    0];
    Set [inMakeVolatile,            1];
    Set [upClassAtInstantiation,    2];
    Set [primitiveRefi,             3];
    Set [changingActiveContext,     4];
    Set [correcting,                5];
    Set [storingLitVar,             6];
    Set [upNewSmallContext,         7];
    Set [viaPrimitiveAtPut,         0C];

    Set[addFirstNil, 8];
    Set[addLast, 9];
    Set[bumpLink, 0a];
    Set[bumpMyList, 0b];
    {skip 0c}
    Set [newActContext, 0d];
    Set [primResumeRefi, 0e];
```

```
{definitions for refd. L1 is the return link register}
            {popAndStoreRecVariable,            0}
            {inMakeVolatile,                    1}
        Set [pushingActiveContext,              2];
            {unusable because of a BRANCH       3}
            {changingActiveContext,             4}
        Set [downOldContextOnReturn,            5];
            {storingLitVar,                     6}
        Set [nextContext,                       7];
        Set [newLargeContextSmashLeaf,          8];
        Set [downOldLeafContext,                9];
        Set [viaPrimitiveValue,                 0A];
            {unusable because of a BRANCH       0B}
            {viaPrimitiveAtPut,                 0C}

        Set[removeFirstLink, 0d];
        Set[debumpLast, 0e];
        Set[debumpMyList, 0f];

{definitions for refd2Return. L1 is the return linkage register}
        Set[transferingTo, 0];
        Set[oldActContext, 1];
        Set[oldActProc, 2];
        Set[sleepingIt, 3];
        Set[postponingIt, 4];


{definitions for getDeltaWord. L1 is the return link register}
            {popAndStoreRecVariable,    0}


{definitions for returnTopOfStack. L1 is the return link register}
            {popAndStoreRecVariable,    0}
            {storingLitVar,             6}
        Set [storeTemporary,            0F];


{definitions for addToZeroCountTable. L2 is the return link register}
            {inMakeVolatile,                    1}
        Set [creatingAnInstance,                2];
            {unusable because of a BRANCH       4}
        Set [doingRefd,                         5];

        Set[doingRefd2, 7];


{definitions for makeVolatile. uMakeVolatileLinkage is the return link register. odd -> no refd of context fields, even -> refd of
context fields}
        Set [makeNewContextVolatile,    0];
        Set [fetchingContextRegisters,  1];
        Set [blockCopying,              2];
        Set [home,                      3];
        Set [primValue,                 5];
        Set [activeAfterStabilize,      7];
        Set [homeAfterStabilize,        9];
        Set [leafAfterStabilize,        0B];

        Set[switchingProcs, 0d];


{definitions for lastPointerOf. L2 is the return link register}
        Set [stabilizingContext,        1];
        Set [getObjectEndForFreeing,    2];


{definitions for transferWords. L1 is the return link register}
        Set [activatingMove,    0];
        Set [primitiveValue,    1];
        Set [performPrim,       2];


{definitions for fetchContextRegisters. L0 is the return link register}
        Set [newContext,                0];
        Set [returningToAContext,       1];


{definitions for nextFreeChunk. L2 is the return link register}
        Set [creatingInstance,          0];
        Set [consideringBigChunks,      1];


{ definitions for creating instances.  temp3High is the return link register }
  { these constants are for Rum microcode in bank 1 }
        Set [makeBigContext,            0];
        Set [makeSmallContext,          1];
        Set [viaPrimitiveNew,           2];
        Set [viaPrimitiveMakePoint,     3];
        Set [viaBlockCopy,              4];
        Set [twoByteLargeInteger,       5];
        Set [countBitsMakeInt,          6];
        Set [bankPlugCreateInstance,    7];
        Set [primitiveMousePoint,       8];


  { these constants are for Rum microcode in bank 0 -- extra rum microcode }
        Set [creatingFloatInstance,     0];     { for Floating Point }
```

```
        Set [makeObjectLargeInt,        1];     { for LargeInteger }


{definitions for positive16BitValueOf. temp3High is the return link register}
        Set [primitiveObjectAt,         0];
        Set [primitiveNew,              1];
        Set [primitiveAt,               2];
        Set [primitiveAtPut,            3];
        Set [primitiveStringAt,         4];
        Set [primitiveStringAtPut,      5];
        Set [primitiveInstVarAt,        6];
        Set [primitiveAsObject,         7];


{definitions for fixedFieldsOf. L2 is the return link register}
        Set [newPrimitive,      0];
        Set [classInPrimAt,     1];
        Set [forSizePrim,       2];
        Set [forInstVarAtPrim,  3];


{link definitions for getTos. L1 is the return link register}
        Set [makePoint, 0];


{link definitions for getSmashTos. L1 is the return link register}
            {makePoint, 0}


{link definitions for primRefi. L2 is the return link register}
        Set [upY,       0];
        Set [upX,       1];


{link definitions for addToFreeChunkList. L1 is the return link register}
        Set [movoFromBigToSmall,        0];
        Set [freeNonPointerObject,      1];
        Set [nowDoneWithObject,         2];
            {compactorCreateFreeChunk,  9}


{Definitions for newFreeChunk.  L0 is the return link.}
        Set [remainderFree,             0];
        Set [carveFree1,                1];
        Set [carveFree2,                2];
        Set [carveFree3,                3];


{link definitions for deallocate. L3 is the return link register}
        Set [fromStabilize,     0];


{linkge definitions for stabilize. L0 is the return link register}
        Set [mesaRequestedStabilization,        0];
        Set [uCodeRequestedStabilization,       1];


{link definitions for getObjectSize. L3 is the return link register}
        Set [atPrim,            0];
        Set [atPutPrim,         1];
        Set [stringAtPrim,      2];
        Set [stringAtPutPrim,   3];
        Set [sizePrim,          4];
        Set [instVarAtPrim,     5];
        Set [countBitsPrim,     6];


{link definitions for getByteOrAddress.  L2 is the return link register}
            {atPrim,            0}
            {stringAtPrim,      2}
            {stringAtPutPrim,   3}

        {link constant for small integer multiply}
        Set [smallPlusMultiply,         0];
        Set [smallMinusMultiply,        1];

        {link constant for small integer divide}
        Set [smallPlusDivide,   0];
        Set [smallMinusDivide,  1];

        {link constant for small integer mod}
        Set [smallModtemp3LowIsPlusQISPlus,     4];
        Set [smallModtemp3LowIsPlusQISMinus,    5];
        Set [smallModtemp3LowIsMinusQISPlus,    6];
        Set [smallModtemp3LowIsMinusQISMinus,   7];

        {link constant for smallInteger div}
        Set [smallPlusDiv,      2];
        Set [smallMinusDiv,     3];

        {link constant for smallInteger quo:}
        Set [smallQuoResultPlus,        8];
        Set [smallQuoResultMinus,       9];
```

```
{qFetchByte2 subroutine return linkage,  L1 is linkage register}
        Set [wordValueIs4Byte0,         0];
        Set [wordValueIs4Byte1,         1];
        Set [wordValueIs4Byte2,         2];
        Set [wordValueIs4Byte3,         3];
        Set [wordValueIs4Byte0Pos,      4];
        Set [wordValueIs4Byte1Pos,      5];
        Set [wordValueIs4Byte2Pos,      6];
        Set [wordValueIs4Byte3Pos,      7];

{fetchByteLength subroutine return linkage,  L2 is linkage register}
        Set [get2word,          0];
        Set [get2wordPos,       1];

{fetch2WordValuePos subroutine return linkage,  L3 is linkage registor}
        Set [largePosAddArgument,       0];
        Set [largePosSubArgument,       1];
        Set [largePosMulArgument,       2];
        Set [largePosDivArgument,       3];
        Set [largePosModArgument,       4];
        Set [largePosDivideArgument,    5];
        Set [largePosCompareArgument,   6];

{fetch2WordValue subroutine return linkage,  L3 is linkage register}
        Set [largePosAddReceiver,       0];
        Set [largePosSubReceiver,       1];
        Set [largePosMulReceiver,       2];
        Set [largePosDivReceiver,       3];
        Set [largePosModReceiver,       4];
        Set [largePosDivideReceiver,    5];
        Set [largePosAndArgument,       6];
        Set [largePosAndReceiver,       7];
        Set [largePosOrArgument,        8];
        Set [largePosOrReceiver,        9];
        Set [largePosXorArgument,       0A];
        Set [largePosXorReceiver,       0B];
        Set [largePosCompareReceiver,   0C];
        Set [largePosBitShiftReceiver,  0D];

{compareLargeInteger subroutine return linkage,  L3 is linkage register}
        Set [primLargeLess,             0];
        Set [primLargeGreater,          1];
        Set [primLargeLessEqual,        2];
        Set [primLargeGreaterEqual,     3];
        Set [primLargeEqual,            4];
        Set [primLargeNotEqual,         5];


{MultiplySubForLargeInt subroutine return linkage,  L1 is linkage register}
        Set [mulLowLow,         0];
        Set [mulLowHigh,        1];
        Set [mulHighLow,        2];
        Set [mulHighHigh,       3];

{Divide4byte SubForLargeInt subroutine return linkage,  L3 is linkage register}
        Set [prim4ByteDiv,      0];
        Set [prim4ByteMod,      1];
        Set [prim4ByteDivide,   2];




{ floating point constant =Float=}
        Set [unaryMessage,              0];
        Set [binaryMessage,             1];

        Set [asFloat,                   0];
        Set [floatAdd,                  0];
        Set [floatTruncated,            1];
        Set [floatSub,                  1];
        Set [floatFractional,           2];
        Set [floatLessThan,             2];
        Set [floatExponent,             3];
        Set [floatGreaterThan,          3];
        Set [RN.PI.IEEE,                4];
        Set [floatTwoTimes,             4];
        Set [floatLessOrEqual,          4];
        Set [floatGreaterOrEqual,       5];
        Set [floatEqual,                6];
        Set [floatNotEqual,             7];
        Set [floatMultiply,             8];
        Set [floatDivide,               9];

{ for fptDivideLoop }
        Set [L3.divide1,                0]; {float divide }
        Set [L3.divide2,                1]; {float divide }

{ for deNorm }
        Set [L3.rePack1,                0]; {float divide }

{ for checkFloatType  ----}
        Set [L1.checkResultType,        0];     {float compare}
        Set [L1.checkInfinity,          1];     {float Compare}
        Set [L1.checkInfinity1,         2];     {float Compare}
        Set [L1.checkReceiver,          3];     {float Divide}
        Set [L1.checkArgument,          4];     {float Divide}
```

```
{ for floatCompare subroutine }
        Set [L2.floatLessThan,          0];
        Set [L2.floatGreaterThan,       1];
        Set [L2.floatLessOrEqual,       2];
        Set [L2.floatGreaterOrEqual,    3];
        Set [L2.floatEqual,             4];
        Set [L2.floatNotEqual,          5];


        Set [floatZero,         0];     { for floarcomapre,=Float=}
        Set [floatNormal,       1];     { for floatcompare,     =Float=}
        Set [floatInfinity,     2];     { for floatcompare, =Float=}
        Set [floatNan,          3],     { for floatcompare, =Float=}


{Defs for activeProc-ret}
        Set[primWaitAP, 0];
        Set[actProcCheckProc, 1];
        Set[primSuspendAP, 2];
        Set[resumeActProc, 3];
        Set[actProcCheckProc, 4];

{Defs for removeFirst-ret}
        Set[wakingHighest, 0];
        Set[primSignalRemove, 1];

{Defs for addFirstOrLast-ret}
        Set[primWaitAdd, 0];
        Set[sleepAddLast, 1];
        Set[postponeAddFirst, 2];

{Defs for schedPtr-ret}
        Set[suspendActiveSPtr, 0];
        Set[resumeSPtr, 1];
        Set[activeProcSPtr, 2];




{+++++ Debugging definitions +++++}


        Set [debug,     1];
        Set [noDebug,   0];

        Set [debugTraps, debug];


        Set [bank0Error,                0];
        Set [refdZero,                  1];
        Set [specialRefdZero,           2];
        Set [primitive0,                3];
        Set [atPutInstError,            4];
        Set [sizeInstError,             5];
        Set [pushLiteralLambda,         6];
        Set [trapAtLocation0,           7];
        Set [otMapSmallInteger,         8];
        Set [otMap2SmallInteger,        9];
        Set [otMap3SmallInteger,        0A];
        Set [tooManyOops,               0B];
        Set [classIsLambda,             0C];
        Set [superclassIsLambda,        0D];
        Set [impossiblePurpose01,       0E];
        Set [impossiblePurpose10,       0F];
        Set [impossibleClass,           10];
        Set [impossibleSize1,           11];
        Set [impossibleSize2,           12];
        Set [impossibleLink,            13];
        Set [otMapBank0SmallInteger,    14];
        Set [otMap1Bank0SmallInteger,   15];
        Set [trapAtLocation0Bank0,      16];
        Set [refCountZero,              17];
        Set [refd2Zero,                 18];




{+++++ Smalltalk-80 object definitions +++++}


        Set [notMinusOnePointer,                3];
        Set [zeroPointer,                       0];
        Set [onePointer,                        4];
        Set [twoPointer,                        8];

        Set [lambdaPointer,                     1];
        Set [nilPointer,                        3];
        Set [falsePointer,                      5];
```

```
Set [truePointer,                       7];
Set [schedulerAssociationPointer,       9];
Set [classSmallIntegerPointer,          0D];
Set [classStringPointer,                0F];
Set [classArrayPointer,                 11];
Set [classFloatPointer,                 15];
Set [methodContextClassOop,             17];
Set [blockContextClassOop,              19];
Set [classPointPointer,                 1B];
Set [classLargePositiveIntegerPointer,  1D];
Set [classMessagePointer,               21];
Set [classCompiledMethodOop,            23];
Set [classCharacterOop,                 29];
Set [doesNotUnderstandSelector,         2B];
Set [cannotReturnSelector,              2D];
Set [specialSelectorsOop,               31];
Set [characterTableOop,                 33];
Set [mustBeBooleanSelector,             35];



{       Object table entry format:

        even word:      address in bank

        odd word:       rrrr rrzo bppu bbbb

                        r:      FC00    reference count
                        z:       200    in zero count table
                        o:       100    in overflow table (Loom)
                        b:        8F    bank address
                        p:        60    purpose bits (60: free oop, 40: error, 20: error, 0: real object)
                        u:        10    untouched
}

        Set [purposeBits,       60];
        Set [freeOop,           60];
        Set [inUse,             0];

        Set [inZctRot8,         2];     {must be rotated to high byte}
        Set [refCountRot8,      0FC];   {must be rotated to high byte}

        Set [refPlusOneRot8,    4];     {must be rotated to high byte}
        Set [refMinusOneRot8,   0FC];   {must be rotated to high byte}



{       Object delta word format:

        dddd dddd uuuc lvop

        d:      FF00    delta count (Loom)
        u:        E0    unused
        c:        10    clean
        l:         8    contains lambda
        v:         4    volatile
        o:         2    odd bytes
        p:         1    pointers
}

        Set [evenBytes,         0];
        Set [hasWords,          0];
        Set [hasPointers,       1];
        Set [oddBytes,          2];
        Set [volatileBit,       4];



{       Stretch format compiledMethod header:

        ffft tttt llll l100

        f:      E000    flag field
        t:      1F00    temporary count
        l:        FC    literal count
        0:         3    smallInteger tag
}



{       Stretch format compiledMethod header extension:

        baaa aapp pppp pp00

        b:      8000    big context flag
        a:      7C00    argument count
        p:       3FC    primitive index
        0:         3    smallInteger tag
}
```

```
{       Stretch format class instance specification:

            pwif ffff ffff ff00

            P:      8000    pointers flag
            W:      4000    words flag
            I:      2000    indexable flag
            F:      1FFC    fixed field count
            0:         3    smallInteger tag
}



        Set [twoByteInteger,    2];
        Set [threeByteInteger,  3];
        Set [fourByteInteger,   4];


        Set [freeBlockMark,     0];
        Set [notAnObjectRot1,   1];


{definitions for size of the Object Header and fields within objects}



        Set [objectHeaderSize, 3];

        Set [twiceObjectHeaderSize, Mul[2, objectHeaderSize]];
        Set [twiceObjectHeaderSizeLessOne, Sub [twiceObjectHeaderSize, 1]];


        Set [tempFrameStart, 6];


        Set [stackPointerAdjustmentFactor, Sub [Add [objectHeaderSize, tempFrameStart], 1]];


        Set [largeContextSizeLessObjectHeader, Add [32'd, tempFrameStart]];
        Set [smallContextSizeLessObjectHeader, Add [12'd, tempFrameStart]];



        Set [field0,  Add [ 0, objectHeaderSize]];
        Set [field1,  Add [ 1, objectHeaderSize]];
        Set [field2,  Add [ 2, objectHeaderSize]];
        Set [field3,  Add [ 3, objectHeaderSize]];
        Set [field4,  Add [ 4, objectHeaderSize]];
        Set [field5,  Add [ 5, objectHeaderSize]];
        Set [field6,  Add [ 6, objectHeaderSize]];
        Set [field7,  Add [ 7, objectHeaderSize]];
        Set [field8,  Add [ 8, objectHeaderSize]];
        Set [field9,  Add [ 9, objectHeaderSize]];
        Set [field10, Add [0A, objectHeaderSize]];
        Set [field11, Add [0B, objectHeaderSize]];
        Set [field12, Add [0C, objectHeaderSize]];
        Set [field13, Add [0D, objectHeaderSize]];
        Set [field14, Add [0E, objectHeaderSize]];
        Set [field15, Add [0F, objectHeaderSize]];


{instance variable offsets into objects}
        Set[processNextLink,            Add[objectHeaderSize,0]];
        Set[processSuspendedContext,    Add[objectHeaderSize,1]];
        Set[processPriority,            Add[objectHeaderSize,2]];
        Set[processMyList,              Add[objectHeaderSize,3]];

        Set[semaphoreFirstLink,         Add[objectHeaderSize,0]];
        Set[semaphoreLastLink,          Add[objectHeaderSize,1]];
        Set[semaphoreExcessSignals,     Add[objectHeaderSize,2]];

        Set[schedulerProcessLists,      Add[objectHeaderSize,0]];
        Set[schedulerActiveProcess,     Add[objectHeaderSize,1]];

        Set[associationKey,             Add[objectHeaderSize,0]];
        Set[associationValue,           Add[objectHeaderSize,1]];

        Set[schedulerAssoicationPointer, 8];
```

```
{definitions for temporaries  within Contexts}


     Set [temp0,  Add [ 0, tempFrameStart, objectHeaderSize]];
     Set [temp1,  Add [ 1, tempFrameStart, objectHeaderSize]];
     Set [temp2,  Add [ 2, tempFrameStart, objectHeaderSize]];
     Set [temp3,  Add [ 3, tempFrameStart, objectHeaderSize]];
     Set [temp4,  Add [ 4, tempFrameStart, objectHeaderSize]];
     Set [temp5,  Add [ 5, tempFrameStart, objectHeaderSize]];
     Set [temp6,  Add [ 6, tempFrameStart, objectHeaderSize]];
     Set [temp7,  Add [ 7, tempFrameStart, objectHeaderSize]];
     Set [temp8,  Add [ 8, tempFrameStart, objectHeaderSize]];
     Set [temp9,  Add [ 9, tempFrameStart, objectHeaderSize]];
     Set [temp10, Add [0A, tempFrameStart, objectHeaderSize]];
     Set [temp11, Add [0B, tempFrameStart, objectHeaderSize]];
     Set [temp12, Add [0C, tempFrameStart, objectHeaderSize]];
     Set [temp13, Add [0D, tempFrameStart, objectHeaderSize]];
     Set [temp14, Add [0E, tempFrameStart, objectHeaderSize]];
     Set [temp15, Add [0F, tempFrameStart, objectHeaderSize]];


{definitions for Literal  fields in CompiledMethods}

     Set [literalStart, 1];

     Set [literalField0,  Add [ 0, literalStart, objectHeaderSize]];
     Set [literalField1,  Add [ 1, literalStart, objectHeaderSize]];
     Set [literalField2,  Add [ 2, literalStart, objectHeaderSize]];
     Set [literalField3,  Add [ 3, literalStart, objectHeaderSize]];
     Set [literalField4,  Add [ 4, literalStart, objectHeaderSize]];
     Set [literalField5,  Add [ 5, literalStart, objectHeaderSize]];
     Set [literalField6,  Add [ 6, literalStart, objectHeaderSize]];
     Set [literalField7,  Add [ 7, literalStart, objectHeaderSize]];
     Set [literalField8,  Add [ 8, literalStart, objectHeaderSize]];
     Set [literalField9,  Add [ 9, literalStart, objectHeaderSize]];
     Set [literalField10, Add [0A, literalStart, objectHeaderSize]];
     Set [literalField11, Add [0B, literalStart, objectHeaderSize]];
     Set [literalField12, Add [0C, literalStart, objectHeaderSize]];
     Set [literalField13, Add [0D, literalStart, objectHeaderSize]];
     Set [literalField14, Add [0E, literalStart, objectHeaderSize]];
     Set [literalField15, Add [0F, literalStart, objectHeaderSize]];
     Set [literalField16, Add [10, literalStart, objectHeaderSize]];
     Set [literalField17, Add [11, literalStart, objectHeaderSize]];
     Set [literalField18, Add [12, literalStart, objectHeaderSize]];
     Set [literalField19, Add [13, literalStart, objectHeaderSize]];
     Set [literalField20, Add [14, literalStart, objectHeaderSize]];
     Set [literalField21, Add [15, literalStart, objectHeaderSize]];
     Set [literalField22, Add [16, literalStart, objectHeaderSize]];
     Set [literalField23, Add [17, literalStart, objectHeaderSize]];
     Set [literalField24, Add [18, literalStart, objectHeaderSize]];
     Set [literalField25, Add [19, literalStart, objectHeaderSize]];
     Set [literalField26, Add [1A, literalStart, objectHeaderSize]];
     Set [literalField27, Add [1B, literalStart, objectHeaderSize]];
     Set [literalField28, Add [1C, literalStart, objectHeaderSize]];
     Set [literalField29, Add [1D, literalStart, objectHeaderSize]];
     Set [literalField30, Add [1E, literalStart, objectHeaderSize]];
     Set [literalField31, Add [1F, literalStart, objectHeaderSize]];


{Definitions for offsets into objects}

     {Loom:  certain offsets are Loom sensitive}

     Set [deltaWordOffset,  0];
     Set [sizeFieldOffset,  1];
     Set [classFieldOffset, 2];

     Set [firstPointerFieldOfObject, classFieldOffset];
     Set [firstFieldOfObject, Add [1, classFieldOffset]];

     Set [sizeOfLargeIntegerForPositive16BitValueOf, Add [1, objectHeaderSize]];


     Set [associationValueIndex {where values live in associations}, Add [1, objectHeaderSize]];



     Set [messageDictionaryOffset, Add [1, objectHeaderSize]];
     Set [instanceSpecificationFieldOffset, Add [2, objectHeaderSize]];
     Set [selectorStart, 2];
     Set [selectorStartPlusObjectHeaderSize, Add [selectorStart, objectHeaderSize]];

     Set [superclassOffset, Add [0, objectHeaderSize]];

     Set [chunkLinkOffset, classFieldOffset];
```

```
{fields of context objects}
        Set [senderFieldOffset,                 Add [0, objectHeaderSize]];
        Set [instructionPointerFieldOffset,     Add [1, objectHeaderSize]];
        Set [stackPointerFieldOffset,           Add [2, objectHeaderSize]];
        Set [methodFieldOffset,                 Add [3, objectHeaderSize]];
        Set [blockArgumentCountOffset,          Add [3, objectHeaderSize]];
        Set [initialInstructionPointerOffset,   Add [4, objectHeaderSize]];
        Set [receiverFieldOffset,               Add [5, objectHeaderSize]];
        Set [homeFieldOffset, receiverFieldOffset];


        Set [differenceBetweenSenderAndMethodFields, Sub[methodFieldOffset, senderFieldOffset]];
        Set [differenceBetweenSenderAndInstructionPointerFields, Sub[instructionPointerFieldOffset, senderFieldOffset]];


        Set [offsetFromDeltaWordToSizeField,  1];
        Set [offsetFromDeltaWordToClassField, 2];

        Set [offsetFromSizeFieldToClassField,  1];
        Set [offsetFromClassFieldToFirstField, 1];

        Set [offsetFromInstructionPointerToStackPointer,        1];
        Set [offsetFromStackPointerToArgCount,                  1];
        Set [offsetFromArgCountToInitialIP,                     1];
        Set [offsetFromInitialIPToHome,                         1];
        Set [offsetFromSenderToBlockArgCount,                   3];
        Set [offsetFromBlockArgumentCountToFirstTemp,           Sub [temp0, blockArgumentCountOffset]];
        Set [offsetFromInitialIpToIp,                           3];
        Set [offsetFromIpToArgumentCount,                       2];
        Set [offsetFromIpToStackPointer,                        1];
        Set [offsetFromSenderToStackPointer,                    2];



{known displacements within Points}
        Set [yFieldOffsetInPoint, Add [1, objectHeaderSize]];
        Set [offsetFromXFieldToYField, 1];



{+++++ Main memory definitions +++++}


        Set [largestFreeChunkSize, 45'd];       {large contexts with up to four Loom words should fit a small free list}
        Set [largestFreeChunkSizeLessOne, Sub [largestFreeChunkSize, 1]];

        Set [maximumObjectSize, 65484'd];
        Set [objectSizeTestLimit, Sub [65535'd, maximumObjectSize]];

        Set [otSizeRot8, 0C0];



{ Offsets into the Rum/Molasses communications record }

        Set [directiveOffset,                   0];

        Set [activeContextOopOffset,            1];

        Set [homeContextOopOffset,              2];

        Set [receiverOopOffset,                 3];

        Set [currentMethodOopOffset,            4];

        Set [instructionPointerOffset,          5];

        Set [stackPointerLowOffset,             6];
        Set [stackPointerHighOffset,            7];

        Set [newProcessWaitingOffset,           8];

        Set [newProcessOopOffset,               9];

        Set [leafContextOopOffset,              0A];

        Set [zctLowOffset,                      0B];
        Set [zctHighOffset,                     0C];

        Set [zctIndexOffset,                    0D];

        Set [displayBitmapOopOffset,            0E];

        Set [displayScreenOopOffset,            0F];

        Set [cursorBitmapOopOffset,             10];

        Set [objectTableLowOffset,              11]  {never actually read nor written!};
        Set [objectTableHighOffset,             12];
```

```
Set [objectMemoryLowOffset,             13];
Set [objectMemoryHighOffset,            14];

Set [objectMemoryAfterLowOffset,        15];
Set [objectMemoryAfterHighOffset,       16];

Set [methodCacheLowOffset,              17];
Set [methodCacheHighOffset,             18];

Set [stabilizationFlagOffset,           19];

Set [stabilizationLimitOffset,          1A];

Set [oopLevelLowOffset,                 1B];
Set [oopLevelHighOffset,                1C];

Set [oopAlertLevelLowOffset,            1D];
Set [oopAlertLevelHighOffset,           1E];

Set [wordLevelLowOffset,                1F];
Set [wordLevelHighOffset,               20];

Set [wordAlertLevelLowOffset,           21];
Set [wordAlertLevelHighOffset,          22];

Set [alreadyAlertedOffset,              23];

Set [signalAlertOffset,                 24];

Set [inputBufferLocLow,                 25];
Set [inputBufferLocHigh,                26];

Set [inputBufferSize,                   27];

Set [inputBufferIn,                     28];

Set [inputBufferOut,                    29];

Set [primitiveMapLowOffset,             2A];
Set [primitiveMapHighOffset,            2B];

Set [freePointersOopOffset,             2C];

Set [freeListsOffset,                   2D];

Set [bigFreeListOffset,                 Add [freeListsOffset, largestFreeChunkSize]];
```

```
{ Edit history:

30-Sep-85 17:10:47      Trow.pa         convert to stretch format
21-Mar-85 13:36:32      Tokunaga.fx     add link for creatInstance, getClassBank0, otMapBank0, used in bank 0 (large integer, floating
point)
16-Mar-85 18:54:20      Udagawa.fx      add create instance link for split Rum bank and comment out u3FFF, u1FFF, u7FF for bank 0
13-Mar-85 12:59:54      Udagawa.fx      mark used u-reg by "&&".
28-Feb-85 15:09:52      Udagawa.fx      Klamath conversion, exchange U36, U2E
21-Feb-85 14:21:44      Susser.pasa     add InputBuffer stuff at the end of Rum communications record
11-Jan-85 10:01:45      Susser.pasa     add Set macro for large memory
 8-Jan-85 10:53:02      Udagawa.fx      add Set macro for sendArithmeticMessage }
```

```
{ MemoryManglement.mc

Object creation, reference counting, stabilization, and other memory management stuff for Rum, the Daybreak Smalltalk-80 microcoded
virtual machine.

        by P McCullough, J Trow, M Hoshino

        31-Jul-87 15:22:16


Copyright 1983, 1984, 1985, 1986, 1987 by Xerox Corporation.  All rights reserved. }



{for each of these entry points, uClassToInstantiate must be the oop of the class, temp3Low is the size in words or bytes.  temp3High is
the return linkage register}

createInstanceWithPointers:
        temp2Low ← nilPointer,                                                  c1;
        temp1Low ← hasPointers, GOTO [createInstance],                          c2;


createInstanceWithBytes:
        temp2Low ← 0,                                                           c1;
        [] ← temp3Low LRot0, XDisp,                                             c2;
        temp3Low ← temp3Low + 1, BRANCH [$, byteCountIsOdd, 0E],                c3;

byteCountIsEven:
        temp1Low ← evenBytes, GOTO [byteShift],                                 c1;


byteCountIsOdd:
        temp1Low ← oddBytes,                                                    c1;
byteShift:
        temp3Low ← RShift1 temp3Low, SE ← 0, GOTO [createInstance],             c2;


createInstanceWithWords:
        temp2Low ← 0,                                                           c1;
        temp1Low ← hasWords, GOTO [createInstance],                             c2;


createLargePositiveInteger:
        temp1Low ← classLargePositiveIntegerPointer,                            c1;
        uClassToInstantiate ← temp1Low,                                         c2;
        GOTO [createInstanceWithBytes]                                          c3;




        { createInstance

        Create a new instance of a given class.

                input:  temp1Low is the odd byte and pointer bits of the delta word
                        temp2Low is the initial value of the fields of the instance
                        temp3Low is the size of the instance in words not including the header
                        uClassToInstantiate is the class of the new instance
                        temp3High is the return link

                output: uNewObject is the new instance
                        uNewObjectHigh/Low is the address of the new instance
                        uRequestedSize is the size of the new instance

                smash:  otLow, temp1High/Low, temp2High/Low, temp3High/Low, Q, uPredecessor, uFieldType, uDefault, uCurrentFreeChunkOop,
                uNextFreeChunk, L1, L2,  }

createInstance:
        uFieldType ← temp1Low
                {save these for initializing the object and its ot entry},      c3;

        uDefault ← temp2Low,                                                    c1;
        temp3Low ← temp3Low + objectHeaderSize, CarryBr,                        c2;
        Q ← objectSizeTestLimit, BRANCH [$, massiveSenility2],                  c3;

        [] ← temp3Low + Q, CarryBr,                                             c1;
        BRANCH [$, requestedSizeTooBig],                                        c2;
        temp2High ← uRumRecordHigh,                                             c3;

        temp2Low ← uRumRecordLow,                                               c1;
        temp1Low ← largestFreeChunkSize,                                        c2;
        [] ← temp3Low - temp1Low, CarryBr,                                      c3;

        uRequestedSize ← temp3Low, BRANCH [$, useBigFreeList],                  c1;
        temp2Low ← temp2Low + freeListsOffset,   {try specific list}            c2;
        Noop,                                                                   c3;

        MAR ← [temp2High, temp2Low + temp3Low],                                 c1;
        CANCELBR [$, 0],                                                        c2;
        otLow ← MD, L2 ← creatingInstance {for nextFreeChunk},                  c3;

        Noop,                                                                   c1;
        Noop,                                                                   c2;
        [] ← otLow and 3, ZeroBr,                                               c3;     {temp}
```

```
        uNewObject ← otLow, BRANCH [$, tryBigList],                          c1;
        L1 ← gettingNextFreeChunk {for otMap2 inside nextFreeChunk},
                CALL [nextFreeChunk],    {got one}                           c2;

        MAR ← [temp2High, temp2Low + temp3Low] {update free list head}       c1, at [creatingInstance, 10, nextFreeChunk-return];
        MDR ← Q, LOOPHOLE [wok], CANCELBR [$, 0],                            c2;
        Q ← temp1High, {save new object's address}                           c3;

        uNewObjectHigh ← Q,                                                  c1;
        uNewObjectLow ← temp1Low,                                            c2;
        GOTO [allocate],                                                     c3;


tryBigList:
        temp2Low ← uRumRecordLow, GOTO [useBigFreeListA],                    c2;


{upon entry, temp2High/Low contains the rum record address. uRequestedSize is valid. temp3Low is the requested size}

useBigFreeList:
        Noop,                                                                c2;
useBigFreeListA:
        uPredecessor ←  0 {should be nilPointer},
                L1 ← gettingNextFreeChunk {for otmap call in nextFreeChunk}, c3;

        MAR ← [temp2High, temp2Low + bigFreeListOffset],
                L2 ← consideringBigChunks,                                   c1;
        temp3Low ← temp3Low + objectHeaderSize, CarryBr,
                {yields minimum splittable block size} CANCELBR [$, 0],      c2;
        otLow ← MD {current free chunk}, BRANCH [$, massiveSenility4],       c3;

considerNextBigFreeChunk:
        Noop,                                                                c1;
        Noop,                                                                c2;
        [] ← otLow and 3, ZeroBr,                                            c3;     {temp}

        uNewObject ← otLow, BRANCH [$, outOfChunks],                         c1;
        uCurrentFreeChunkOop ← otLow, CALL [nextFreeChunk],                  c2;

        uNextFreeChunk ← Q {remember next free chunk},                       c1, at [consideringBigChunks, 10, nextFreeChunk-return];
        temp1Low ← temp1Low + sizeFieldOffset,                               c2;
        Noop,                                                                c3;

        MAR ← [temp1High, temp1Low + 0],                                     c1;
        Noop,                                                                c2;
        Q ← MD {size of current free chunk},                                 c3;

        [] ← Q xor uRequestedSize, ZeroBr,                                   c1;
        [] ← Q - temp3Low, CarryBr, BRANCH [$, exactFit],                    c2;
        BRANCH [$, canSubdivide],                                            c3;

iterate:
        uPredecessor ← otLow,                                                c1;
        Noop,                                                                c2;
        otLow ← uNextFreeChunk, GOTO [considerNextBigFreeChunk],             c3;


exactFit:
        temp1Low ← temp1Low - sizeFieldOffset, CANCELBR [$, 1],              c3;

        Q ← temp1High,                                                       c1;
        uNewObjectLow ← temp1Low,                                            c2;
        uNewObjectHigh ← Q, GOTO [splice],                                   c3;


canSubdivide:
        temp3Low ← uRequestedSize,                                           c1;
        temp3Low {new size} ← Q {current size} - temp3Low {requested size},  c2;
        Q ← temp1High {part of new object's address},                        c3;

        MAR ← [temp2High, temp2Low + freePointersOopOffset],                 c1;
        uNewObjectHigh ← Q, CANCELBR [$, 0],                                 c2;
        otLow ← MD {first free oop},                                         c3;

        Noop,                                                                c1;
        Noop,                                                                c2;
        [] ← otLow and 3, ZeroBr,                                            c3;     {temp}

        uNewObject ← otLow, BRANCH [$, outOfOops],                           c1;
        Q ← temp1Low - sizeFieldOffset, L1 ← splittingFreeChunk,             c2;
        Q {object address} ← Q {chunk address} + temp3Low {chunk size},      c3;

{write the new size of the current free chunk (temp1High/Low still pointing at its size field)}

        MAR ← [temp1High, temp1Low + 0],                                     c1;
        MDR ← temp3Low {new chunk size},                                     c2;
        uNewObjectLow ← Q,                                                   c3;

        CALL [getOtAddress] {free oop link},                                 c1;

        MAR ← [temp2High, temp2Low + freePointersOopOffset],                 c1, at [splittingFreeChunk, 10, getAddressReturn];
        MDR ← temp1Low {new free oop head}, LOOPHOLE [wok], CANCELBR [$, 0], c2;
        temp2High ← splittingFreeChunk,                                      c3;

        temp1Low ← uNewObjectHigh, CALL [putOtFlags],                        c1;
```

```
        temp1Low ← uNewObjectLow, CALL [putOtAddress],                          c1, at [splittingFreeChunk, 10, putFlagsReturn];

        temp1Low {chunk address} ← temp1Low - temp3Low,                         c1, at [splittingFreeChunk, 10, putAddressReturn];
        temp2Low ← largestFreeChunkSize,                                        c2;
        [] ← temp3Low {chunk size} - temp2Low, CarryBr,                         c3;

{should we move the current free chunk to a small free chunk list?}

        otLow ← uCurrentFreeChunkOop, BRANCH [$, itsFineWhereItIs],             c1;
        L1 ← moveFromBigToSmall,                                                c2;
        CALL [addToFreeChunkList] {this call returns directly to splice},       c3;


splice:
        otLow ← uPredecessor, L1 ← splicingBigFreeList,                         c1, at [moveFromBigToSmall, 10, addFreeChunkReturn];
        Noop,                                                                   c2;
        Noop,                                                                   c3;

        [] ← otLow and 3, ZeroBr,                                               c1;     {temp}
        temp2High ← uRumRecordHigh, BRANCH [didHavePredecessor, $],             c2;
        temp2Low ← uRumRecordLow, {no predecessor}                              c3;

        MAR ← [temp2High, temp2Low + bigFreeListOffset], GOTO [linkNextChunk],  c1;


didHavePredecessor:
        CALL [otMap2],                                                          c3;

        temp1Low ← temp1Low + chunkLinkOffset,                                  c1, at [splicingBigFreeList, 10, otMap2-return];
        Noop,                                                                   c2;
        Noop,                                                                   c3;

        MAR ← [temp1High, temp1Low + 0],                                        c1;
linkNextChunk:
        MDR ← uNextFreeChunk, LOOPHOLE [wok], CANCELBR [bigListWrapup, 0],      c2;


itsFineWhereItIs:
        Noop,                                                                   c2;
bigListWrapup:
        otLow ← uNewObject, GOTO [allocate],                                    c3;


outOfChunks:
        GOTO [massiveSenility],                                                 c2;


outOfOops:
        Noop,                                                                   c2;
massiveSenility:
        Noop,                                                                   c3;

massiveSenility2:
        GOTO [bytecodeFailed],                                                  c1;


massiveSenility4:
        CANCELBR [bytecodeFailed, OF],                                          c1;


requestedSizeTooBig:
        GOTO [massiveSenility2],                                                c3;




{Adjust the memory and oop levels and signal Mesa if either is below its alert level and Mesa has not yet been signalled.

(((wordLevel < wordAlertLevel) or: [oopLevel < oopAlertLevel]) and: [alreadyAlerted = 0]) ifTrue: [signalAlert ← 1. MesaIntRq]}

allocate:
        temp2High ← uRumRecordHigh,                                             c1;
        temp2Low ← uRumRecordLow,                                               c2;
        Noop,                                                                   c3;

        MAR ← [temp2High, temp2Low + oopLevelLowOffset],                        c1;
        CANCELBR [$, 0],                                                        c2;
        temp3Low ← MD,                                                          c3;

        MAR ← [temp2High, temp2Low + oopLevelLowOffset],                        c1;
        MDR ← temp3Low ← temp3Low - 1, LOOPHOLE [wok], CANCELBR [$, 0],         c2;
        Noop,                                                                   c3;

lowOopTest:
        MAR ← [temp2High, temp2Low + oopAlertLevelLowOffset],                   c1;
        CANCELBR [$, 0],                                                        c2;
        Q ← MD,                                                                 c3;

        Noop,                                                                   c1;
        Q ← temp3Low - Q, CarryBr,                                              c2;
        BRANCH [$, decreaseWordLevel],                                          c3;

        MAR ← [temp2High, temp2Low + alreadyAlertedOffset],                     c1;
        CANCELBR [$, 0],                                                        c2;
        Q ← MD,                                                                 c3;
```

```
        [] ← Q, ZeroBr,                                                    c1;
        BRANCH [decreaseWordLevel2, $],                                    c2;
        Noop,                                                              c3;

        MAR ← [temp2High, temp2Low + signalAlertOffset],                   c1;
        MDR ← 1, LOOPHOLE [wok], CANCELBR [$, 0],                          c2;
        MesaIntRq {run mesa before next bytecode}, GOTO [decreaseWordLevel], c3;


decreaseWordLevel2:
        Noop,                                                              c3;

decreaseWordLevel:
        MAR ← [temp2High, temp2Low + wordLevelLowOffset],                  c1;
        temp3Low ← uRequestedSize, CANCELBR [$, 0],                        c2;
        Q ← MD,                                                            c3;

        MAR ← [temp2High, temp2Low + wordLevelLowOffset],                  c1;
        MDR ← temp3Low ← Q - temp3Low, CarryBr, LOOPHOLE [wok],
              CANCELBR [$, 0],                                             c2;
        BRANCH [$, lowMemoryTestHighGetData],                              c3;

wordLevelBorrow:
        MAR ← [temp2High, temp2Low + wordLevelHighOffset],                 c1;
        CANCELBR [$, 0],                                                   c2;
        Q ← MD,                                                            c3;

        MAR ← [temp2High, temp2Low + wordLevelHighOffset],                 c1;
        MDR ← Q - 1, LOOPHOLE [wok], CANCELBR [$, 0],                      c2;
        temp1Low ← Q - 1, GOTO [lowMemoryTestHighHaveData],                c3;


lowMemoryTestHighGetData:
        MAR ← [temp2High, temp2Low + wordLevelHighOffset],                 c1;
        CANCELBR [$, 0],                                                   c2;
        temp1Low ← MD,                                                     c3;

lowMemoryTestHighHaveData:
        MAR ← [temp2High, temp2Low + wordAlertLevelHighOffset],            c1;
        CANCELBR [$, 0],                                                   c2;
        Q ← MD,                                                            c3;

        Q ← Q - temp1Low, CarryBr,                                         c1;
        [] ← Q, ZeroBr, BRANCH [reallyAllocate1, $],                       c2;
        BRANCH [$, lowMemoryTestLow],                                      c3;

        MAR ← [temp2High, temp2Low + alreadyAlertedOffset],                c1;
        CANCELBR [$, 0],                                                   c2;
        Q ← MD,                                                            c3;

        [] ← Q, ZeroBr,                                                    c1;
        BRANCH [lowMemoryTestLow2, $],                                     c2;
        Noop,                                                              c3;

        MAR ← [temp2High, temp2Low + signalAlertOffset],                   c1;
        MDR ← 1, LOOPHOLE [wok], CANCELBR [$, 0],                          c2;
        MesaIntRq {run mesa before next bytecode}, GOTO [reallyAllocate3], c3;


lowMemoryTestLow2:
        Noop,                                                              c3;

lowMemoryTestLow:
        MAR ← [temp2High, temp2Low + wordAlertLevelLowOffset],             c1;
        CANCELBR [$, 0],                                                   c2;
        Q ← MD,                                                            c3;

        Noop,                                                              c1;
        Q ← temp3Low - Q, CarryBr,                                         c2;
        BRANCH [$, reallyAllocate3],                                       c3;

        MAR ← [temp2High, temp2Low + alreadyAlertedOffset],                c1;
        CANCELBR [$, 0],                                                   c2;
        Q ← MD,                                                            c3;

        [] ← Q, ZeroBr,                                                    c1;
        BRANCH [reallyAllocate2, $],                                       c2;
        Noop,                                                              c3;

        MAR ← [temp2High, temp2Low + signalAlertOffset],                   c1;
        MDR ← 1, LOOPHOLE [wok], CANCELBR [$, 0],                          c2;
        MesaIntRq {run mesa before next bytecode}, GOTO [reallyAllocate3], c3;


reallyAllocate1:
        CANCELBR [reallyAllocate3, 1],                                     c3;


reallyAllocate2:
        Noop,                                                              c3;

reallyAllocate3:
        Noop,                                                              c1;
        temp1High ← uNewObjectHigh,                                        c2;
        temp1Low ← uNewObjectLow,                                          c3;

        temp3Low ← uRequestedSize,                                         c1;
```

```
        temp3Low ← temp1Low + temp3Low, {end of the object + 1}                 c2;
        temp1Low ← temp1Low + deltaWordOffset,                                  c3;

{initialize the object header now}

        MAR ← [temp1High, temp1Low + 0],                                        c1;
        MDR ← uFieldType, {set delta word}                                      c2;
        temp1Low ← temp1Low + offsetFromDeltaWordToSizeField,                   c3;

        MAR ← [temp1High, temp1Low + 0],                                        c1;
        MDR ← uRequestedSize,                                                   c2;
        temp1Low ← temp1Low + offsetFromSizeFieldToClassField,                  c3;

        MAR ← [temp1High, temp1Low + 0],                                        c1;
        MDR ← uClassToInstantiate,                                              c2;
        {ok, object header is done, now zap the object body}
        temp1Low ← temp1Low + offsetFromClassFieldToFirstField,                 c3;

        temp2Low ← uDefault,                                                    c1;
        Noop,                                                                   c2;
        [] ← temp1Low - temp3Low, ZeroBr,                                       c3;

initializeObjectBody:
        MAR ← [temp1High, temp1Low + 0], BRANCH [$, zapped],                    c1;
        MDR ← temp2Low, temp1Low ← temp1Low + 1,                                c2;
        [] ← temp1Low - temp3Low, ZeroBr, GOTO [initializeObjectBody],          c3;


zapped:         {the object header and object body are completely initialized. now fix up the object table entry}
        Noop,                                                                   c2;
        temp3Low ← temp3High {return link}, L2 ← creatingAnInstance,            c3;

        temp2High ← newObjectHeader, CALL [getOtFlags],                         c1;

        temp1Low ← temp1Low and 1F, CALL [putOtFlags],                          c1, at [newObjectHeader, 10, getFlagsReturn];

        temp2Low ← temp3Low {save return link}, CALL [addToZeroCountTable],     c1, at [newObjectHeader, 10, putFlagsReturn];

        Noop,                                                                   c1, at [creatingAnInstance, 10,
        addToZeroCountTableReturn];
        L1 ← upClassAtInstantiation,                                            c2;
        otLow ← uClassToInstantiate, XDisp, CALL [refi],                        c3;

        otLow ← uNewObject,                                                     c1, at [upClassAtInstantiation, 10, refiReturn];
        Xbus ← temp2Low LRot0, XDisp,                                           c2;
        RET [createInstance-return],                                            c3;




        { nextFreeChunk

        Return the next object on a free list.  Free objects are linked through their class fields.

                input:  otLow is the current object
                        otHigh is the high part of the object table base address
                        L1 is the return link for otMap2
                        L2 is the return link

                output: Q is the next object
                        temp1High/Low is the address of the current object

                smash:  }

nextFreeChunk:
        CALL [otMap2],                                                          c3;

        temp1Low ← temp1Low + chunkLinkOffset,                                  c1, at [gettingNextFreeChunk, 10, otMap2-return];
        Noop,                                                                   c2;
        Noop,                                                                   c3;

        MAR ← [temp1High, temp1Low + 0],                                        c1;
        temp1Low ← temp1Low - chunkLinkOffset, L2Disp,                          c2;
        Q ← MD, RET [nextFreeChunk-return],                                     c3;




        { addToFreeChunkList

        Add an object to the appropriate free list.

                input:  otLow is the object
                        temp1High/Low is the address of the object
                        temp3Low is the index of the list (size of the object for small objects)
                        uRumRecordHigh/Low is the Rum communications record base address
                        L1 is the return link

                output: temp2High/Low is the Rum communications record base address

                smash:  Q }

addToFreeChunkList:
        temp2High ← uRumRecordHigh,                                             c1;
        temp2Low ← uRumRecordLow,                                               c2;
```

```
        temp2Low ← temp2Low + freeListsOffset,                                  c3;

        MAR ← [temp2High, temp2Low + temp3Low],                                 c1;
        temp1Low ← temp1Low + chunkLinkOffset, CANCELBR [$, 0],                 c2;
        Q ← MD {current free list head},                                        c3;

        MAR ← [temp1High, temp1Low + 0],                                        c1;
        MDR ← Q, {link new object to old list head}                             c2;
        Noop,                                                                   c3;

        MAR ← [temp2High, temp2Low + temp3Low],                                 c1;
        MDR ← otLow {new list head}, LOOPHOLE [wok], L1Disp, CANCELBR [$, 0],   c2;
        temp1Low ← temp1Low - chunkLinkOffset, RET [addFreeChunkReturn],        c3;




        { refi

        Increment the reference count of an object.  Nil, false, and true have permanently stuck counts, so skip them.  SmallIntegers
        don't have reference counts, so skip them too.

                input:  otLow is the object
                        otHigh is the high part of the object table base address
                        L1 is the return link
                        there is a pending XDisp to test for a smallInteger

                output:

                smash:  temp1Low, temp2High, Q }

refi:
        DISP4 [refiTable, 0C],                                                  c1;


refiOop01:
        [] ← falsePointer - otLow, CarryBr, GOTO [doRefi],                      c2, at [0D, 10, refiTable];


refiOop11:
        [] ← truePointer - otLow, CarryBr, GOTO [doRefi],                       c2, at [0F, 10, refiTable];


refiOop10:
        [] ← 1, ZeroBr, GOTO [doRefi],                                          c2, at [0E, 10, refiTable];


refiSmallInteger00:
        L1Disp, GOTO [returnFromRefi],                                          c2, at [0C, 10, refiTable];


doRefi:
        BRANCH [$, skipRefi], {skip nil, false, true},                          c3;

        temp2High ← doingNormalRefi, CALL [getOtFlags],                         c1;

        Q ← temp1Low,                                                           c1, at [doingNormalRefi, 10, getFlagsReturn];
        temp1Low ← refPlusOneRot8 {for incrementing ref count},                 c2;
        [] ← Q LRot0, XHDisp {first part of test for stuck ref count},          c3;

        temp1Low ← temp1Low LRot8, BRANCH [$, maybeStuckRefi, 2],               c1;
        Q ← Q + temp1Low {up ref count}, {sign is positive, thus not
                stuck and cannot become stuck}                                  c2;
        [] ← 1, ZeroBr, {force next BRANCH}                                     c3;

updateOtRefi:
        temp1Low ← Q, BRANCH [{CALL} putOtFlags, justGotStuckRefi],             c1;

        Noop,                                                                   c1, at [doingNormalRefi, 10, putFlagsReturn];
        L1Disp,                                                                 c2;
returnFromRefi:
        RET [refiReturn],                                                       c3;


maybeStuckRefi: {sign is negative, can get stuck, may already be stuck}
        Q ← Q + temp1Low {up ref count}, CarryBr {carry implies already stuck},  c2;
        [] ← Q + temp1Low, CarryBr {carry implies just got stuck},
                BRANCH [updateOtRefi, $],                                       c3;

stuckRefi:
        CANCELBR [$, 1],                                                        c1;
stuckRefiReturn:
        L1Disp, GOTO [returnFromRefi],                                          c2;


justGotStuckRefi: {Loom: need to call Loom here for newly stuck ref count}
        Noop,                                                                   c2;
        Noop,                                                                   c3;

        CALL [putOtFlags],                                                      c1;


skipRefi:
        GOTO [stuckRefiReturn],                                                 c1;
```

```
        { refd

        Decrement the reference count of an object.  Nil, false, and true have permanently stuck counts, so skip them.  SmallIntegers
        don't have reference counts, so skip them too.

                input:  otLow is the object
                        otHigh is the high part of the object table base address
                        L1 is the return link
                        there is a pending XDisp to test for a smallInteger

                output:

                smash:  temp1High/Low, temp2High, temp3High/Low, Q, L2 }

refd:
        temp3Low ← refMinusOneRot8, DISP4 [refdTable, 0C],                      c1;


refdOop01:
        [] ← falsePointer - otLow, CarryBr, GOTO [doRefd],                      c2, at [0D, 10, refdTable];


refdOop11:
        [] ← truePointer - otLow, CarryBr, GOTO [doRefd],                       c2, at [0F, 10, refdTable];


refdOop10:
        [] ← 1, ZeroBr, GOTO [doRefd],                                          c2, at [0E, 10, refdTable];


refdSmallInteger00:
        L1Disp, GOTO [returnFromRefd],                                          c2, at [0C, 10, refdTable];


doRefd:
        temp3Low ← temp3Low LRot8,
                BRANCH [$, skipRefd], {skip nil, false, true},                  c3;

getRefCount:
        temp2High ← doingNormalRefd, CALL [getOtFlags],                         c1;

        [] ← temp1Low LRot0, XHDisp {first part of stuck ref count test},       c1, at [doingNormalRefd, 10, getFlagsReturn];
        Q ← ~temp3Low, BRANCH [$, negativeRefCount, 2],                         c2;
positiveRefCount:       {not stuck but could go to zero}
        temp1Low ← temp1Low + temp3Low {subtract 1}, CarryBr
                {no carry implies already zero, an error}, L2 ← doingRefd,      c3;

updateOtRefd:
        BRANCH [triedToRefdZeroCountObject, $],                                 c1;
        Noop,                                                                   c2;
        Noop,                                                                   c3;

        CALL [putOtFlags],                                                      c1;

        [] ← temp1Low + temp3Low {subtract again}, CarryBr {no carry implies
                just went to zero},                                             c1, at [doingNormalRefd, 10, putFlagsReturn];
        L1Disp, BRANCH [belongsInZct, $],                                       c2;
returnFromRefd:
        RET [refdReturn],                                                       c3;


negativeRefCount:       {could be stuck but cannot go to zero}
        Q ← Q + 1, {refPlusOneRot8 LRot8}                                       c3;

        [] ← temp1Low + Q, CarryBr {carry implies stuck ref count},             c1;
        temp1Low ← temp1Low + temp3Low {subtract one from ref count},
                BRANCH [$, stuckRefd],                                          c2;
        [] ← 0, ZeroBr {force BRANCH}, GOTO [updateOtRefd],                     c3;


belongsInZct:
        CANCELBR [$, 0F],                                                       c3;

        CALL [addToZeroCountTable],                                             c1;

skipRefd:
        GOTO [refdSmallInteger00],                                              c1, at [doingRefd, 10, addToZeroCountTableReturn];


stuckRefd:
        GOTO [skipRefd],                                                        c3;


triedToRefdZeroCountObject:
        Q ← refdZero, CANCELBR [bailout3, 1], {error}                           c2;




        { refd2

        Identical to refd except for return links. }
```

```
refd2:
        temp3Low ← refMinusOneRot8, DISP4 [refd2Table, 0C],                    c1;


refd2Oop01:
        [] ← falsePointer - otLow, CarryBr, GOTO [doRefd2],                    c2, at [0D, 10, refd2Table];


refd2Oop11:
        [] ← truePointer - otLow, CarryBr, GOTO [doRefd2],                     c2, at [0F, 10, refd2Table];


refd2Oop10:
        [] ← 1, ZeroBr, GOTO [doRefd2],                                        c2, at [0E, 10, refd2Table];


refd2SmallInteger00:
        L1Disp, GOTO [returnFromRefd2],                                        c2, at [0C, 10, refd2Table];


doRefd2:
        temp3Low ← temp3Low LRot8,
                BRANCH [$, skipRefd2], {skip nil, false, true},                c3;

getRefCount2:
        temp2High ← doingNormalRefd2, CALL [getOtFlags],                       c1;

        [] ← temp1Low LRot0, XHDisp {first part of stuck ref count test},      c1, at [doingNormalRefd2, 10, getFlagsReturn];
        Q ← ~temp3Low, BRANCH [$, negativeRefCount2, 2],                       c2;
positiveRefCount2:      {not stuck but could go to zero}
        temp1Low ← temp1Low + temp3Low {subtract 1}, CarryBr
                {no carry implies already zero, an error}, L2 ← doingRefd2,    c3;

updateOtRefd2:
        BRANCH [triedToRefd2ZeroCountObject, $],                               c1;
        Noop,                                                                  c2;
        Noop,                                                                  c3;

        CALL [putOtFlags],                                                     c1;

        [] ← temp1Low + temp3Low {subtract again}, CarryBr {no carry implies
                just went to zero},                                            c1, at [doingNormalRefd2, 10, putFlagsReturn];
        L1Disp, BRANCH [belongsInZct2, $],                                     c2;
returnFromRefd2:
        RET [refd2Return],                                                     c3;


negativeRefCount2:      {could be stuck but cannot go to zero}
        Q ← Q + 1, {refPlusOneRot8 LRot8}                                      c3;

        [] ← temp1Low + Q, CarryBr {carry implies stuck ref count},            c1;
        temp1Low ← temp1Low + temp3Low {subtract one from ref count},
                BRANCH [$, stuckRefd2],                                        c2;
        [] ← 0, ZeroBr {force BRANCH}, GOTO [updateOtRefd2],                   c3;


belongsInZct2:
        CANCELBR [$, 0F],                                                      c3;

        CALL [addToZeroCountTable],                                            c1;

skipRefd2:
        GOTO [refd2SmallInteger00],                                            c1, at [doingRefd2, 10, addToZeroCountTableReturn];


stuckRefd2:
        GOTO [skipRefd2],                                                      c3;


triedToRefd2ZeroCountObject:
        Q ← refd2Zero, CANCELBR [bailout3, 1], {error}                         c2;




        { addToZeroCountTable           Loom: Loom may want to get involved here--but I don't think so

        Add an object to the zero count table.  Objects get added to the zero count table when their reference counts go to zero.
        Eventually the zero count table fills up and all the objects in it are recursively freed.  In the meantime, we turn on the inZct
        bit in the OT and write the new OT entry.  Then we look at the object's old inZct bit and add the object to the zct if it's not
        already there.  If this object fills the table, we'll have to stabilize and free soon.  There's room for some overflow so we can
        finish this bytecode.

                input:  otLow is the object to put into the zct
                        otHigh is the high part of the object table base address
                        L2 is the return link

                output: uTimeToStabilize is -1 if the zct is full

                smash:  temp1High/Low, temp2High, temp3High/Low, Q }

addToZeroCountTable:
        temp3Low ← inZctRot8,                                                  c2;
        temp3Low ← temp3Low LRot8,                                             c3;

        temp2High ← addingToZct, CALL [getOtFlags],                            c1;
```

```
        [] ← temp1Low and temp3Low, ZeroBr,                                  c1, at [addingToZct, 10, getFlagsReturn];
        temp1Low ← temp1Low or temp3Low, L2Disp, BRANCH [alreadyInZct, $],    c2;
        Q ← uRumRecordHigh, CANCELBR [$, 0F]                                 c3;

        temp1High ← Q LRot0, CALL [putOtFlags],                              c1;

        temp1Low ← uRumRecordLow,                                            c1, at [addingToZct, 10, putFlagsReturn];
        Noop,                                                                c2;
        Noop,                                                                c3;

        MAR ← [temp1High, temp1Low + zctIndexOffset],                        c1;
        CANCELBR [$, 0],                                                     c2;
        Q ← MD, {read current index}                                         c3;

        MAR ← [temp1High, temp1Low + zctLowOffset],                          c1;
        CANCELBR [$, 0],                                                     c2;
        temp3Low ← MD, {get zct low address}                                 c3;

        MAR ← [temp1High, temp1Low + zctHighOffset],                         c1;
        temp3Low ← temp3Low + Q, CANCELBR [$, 0],                            c2;
        temp3High ← MD, {get zct high address}                               c3;

                { Note: The Molasses zeroCountTable is one-relative, not zero-relative. So, while Molasses bumps the index before putting
                something in the zct, we put it in, then bump. }

        MAR ← [temp3High, temp3Low + 0],                                     c1;
        MDR ← otLow, {add the object}                                        c2;
        Noop,                                                                c3;

        MAR ← [temp1High, temp1Low + zctIndexOffset],                        c1;
        MDR ← Q ← Q + 1 {new index}, LOOPHOLE [wok], CANCELBR [$, 0],        c2;
        Noop,                                                                c3;

zctFullTest:
        MAR ← [temp1High, temp1Low + stabilizationLimitOffset],              c1;
        CANCELBR [$, 0],                                                     c2;
        temp3Low ← MD, {get the zct stabilization limit}                     c3;

        Noop,                                                                c1;
        [] ← temp3Low - Q, NegBr,                                            c2;
        BRANCH [zctIndexOk, $], {need to stabilize if limit exceeded}        c3;

        MAR ← [temp1High, temp1Low + stabilizationFlagOffset],               c1;
        MDR ← needToStabilize, LOOPHOLE [wok], CANCELBR [$, 0],              c2;
        uTimeToStabilize ← ~stackLow xor stackLow {0FFFF},                   c3;

zctIndexOk:
        Noop,                                                                c1;
        L2Disp,                                                              c2;
alreadyInZct:
        RET [addToZeroCountTableReturn],                                     c3;




        { makeVolatile


                input:

                output:

                smash:  temp2High/Low, uZctBaseHigh }

{upon entry, otLow is the oop to make volatile,  uMakeVolatileLinkage is the return linkage register: if it is odd, each object referred
to by the object will be refd'd; it it is even, the object is marked volatile, but no refding occurs. smashes temp3Low, Q, L1, L2.
leaves base of object in uMakeVolatileHigh/Low, and in temp1High/Low. leaves uLastPointer set up}

{see if we're trying to make nil volatile -- this happens when the leaf context oop is nil.  check should probably be moved to the place
where volatilization is done after stabilization...}

makeVolatile:
        [] ← otLow xor nilPointer, ZeroBr,                                   c2;
        BRANCH [$, nilMakeVolatile],                                         c3;

        uMakeVolatileOop ← otLow,                                            c1;
        L1 ← makingVolatile,                                                 c2;
        CALL [otMap] {get address of base of object},                        c3;

        Q ← temp1High {save object base}                                     c1, at [makingVolatile, 10, otMap-return];
        uMakeVolatileHigh ← Q,                                               c2;
        uMakeVolatileLow ← temp1Low,                                         c3;

{       temp1Low ← temp1Low + deltaWordOffset,                               c1;
        Noop,                                                                c2;
        Noop,                                                                c3;     }

        MAR ← [temp1High, temp1Low + 0],                                     c1;
        Noop,                                                                c2;
        Q ← MD {delta word},                                                 c3;

        [] ← Q and volatileBit, ZeroBr {already volatile?},                  c1;
        Ybus ← uMakeVolatileLinkage, XDisp,
                BRANCH [$, doMakeVolatile, 0E],                              c2;
```

```
        temp1Low ← temp1Low - deltaWordOffset, RET [makeVolatile-return],        c3;


doMakeVolatile:
        Q ← Q or volatileBit, CANCELBR [$, Of],                                  c3;

        MAR ← [temp1High, temp1Low + 0],                                         c1;
        MDR ← Q {delta word with volatile bit set},                              c2;
        Ybus ← uMakeVolatileLinkage, XDisp
                {should we refd the referents or not?},                          c3;

        temp1Low ← temp1Low - deltaWordOffset,
                BRANCH [returnFromMakeVolatile, $, 0E],                          c1;
        temp1Low ← temp1Low + sizeFieldOffset,   {refd fields}                   c2;
        Q ← temp1High,                                                           c3;

        MAR ← [temp1High, temp1Low + 0],                                         c1;
        temp1Low ← temp1Low - sizeFieldOffset,                                   c2;
        temp3Low ← MD {size field},                                              c3;

        temp3Low ← temp3Low + temp1Low,                                          c1;
        temp3Low ← temp3Low - 1 {low 16 bits of last pointer
                of context object},                                              c2;
        uLastPointer ← temp3Low,                                                 c3;

{now, sweep the object decrementing reference counts of all pointer fields}

        temp2Low ← temp1Low + firstPointerFieldOfObject,                         c1;
        uZctBaseHigh ← Q,                                                        c2;
        temp2High ← Q LRot0,                                                     c3;

makeVolatileLoop:
        MAR ← [temp2High, temp2Low + 0], L1 ← inMakeVolatile,                    c1;
        temp2Low ← temp2Low + 1,                                                 c2;
        otLow ← MD, XDisp, CALL [refd],                                          c3;

        temp3Low ← uLastPointer,                                                 c1, at [inMakeVolatile, 10, refdReturn];
        [] ←  temp2Low - temp3Low, ZeroBr,                                       c2;
        temp2High ← uZctBaseHigh, BRANCH [makeVolatileLoop, $],                  c3;

        otLow ← uMakeVolatileOop, L2 ← inMakeVolatile,                           c1;
        Noop,                                                                    c2;
        Noop,                                                                    c3;

        CALL [addToZeroCountTable],                                              c1;

        temp1High ← uMakeVolatileHigh,                                           c1, at [inMakeVolatile, 10, addToZeroCountTableReturn];
        temp1Low ← uMakeVolatileLow,                                             c2;
        temp2High ← uZctBaseHigh,                                                c3;

        Noop,                                                                    c1;
returnFromMakeVolatile:
        Ybus ← uMakeVolatileLinkage, XDisp,                                      c2;
returningFromMakeVolatile:
        RET [makeVolatile-return],                                               c3;


nilMakeVolatile:
        GOTO [returnFromMakeVolatile],                                           c1;




        { lastPointerOf

        Return the address of the last pointer field of an object.

                input:  temp1High/Low is the address of the object
                        Q is the object's delta word
                        L2 is the return link

                output: temp1High/Low is the address of the object
                        temp3Low and uLastPointer are the address of the last pointer

                smash:  Q }

lastPointerOf:
        [] ← Q and 1 {pointer bit}, ZeroBr,                                      c1;
        Q ← classCompiledMethodOop,
                BRANCH [doesHavePointers, doesNotHavePointers],                  c2;


{is pure pointer object -- last pointer is is at (base + size - 1)}

doesHavePointers:
        temp1Low ← temp1Low + sizeFieldOffset,                                   c3;

        MAR ← [temp1High, temp1Low + 0] {start read of length field},            c1;
        temp1Low ← temp1Low - sizeFieldOffset {again point at base of object},   c2;
        temp3Low ← MD, GOTO [returnFromLastPointerOf],                           c3;


        {no pointers, might be compiledMethod -- need to check class}
doesNotHavePointers:
        temp1Low ← temp1Low + classFieldOffset,                                  c3;
```

```
        MAR ← [temp1High, temp1Low + 0],                                        c1;
        temp1Low ← temp1Low - classFieldOffset {again point at base of object}, c2;
        temp3Low ← MD {the class of the object},                                c3;

        [] ← temp3Low xor Q {compiledMethodClass oop}, ZeroBr,                  c1;
        BRANCH [$, isCompiledMethod],                                           c2;
        temp3Low ← objectHeaderSize, GOTO [returnFromLastPointerOf],           c3;


        {need to get number of literals from the method header}
isCompiledMethod:
        temp1Low ← temp1Low + objectHeaderSize {point at method header},        c3;

        MAR ← [temp1High, temp1Low + 0],                                        c1;
        temp1Low ← temp1Low - objectHeaderSize, {again point at base of object} c2;
        temp3Low ← MD {the method header},                                      c3;

        temp3Low ← (RShift1 temp3Low and 0FF) {get literal count
                of compiledMethod}, SE ← 0,                                     c1;
        temp3Low ← RShift1 temp3Low, SE ← 0,                                    c2;
        Noop,                                                                   c3;

        Noop,                                                                   c1;
        temp3Low ← temp3Low + literalStart,                                     c2;
        temp3Low ← temp3Low + objectHeaderSize,                                 c3;

returnFromLastPointerOf:
        temp3Low ← temp3Low - 1,                                                c1;
        temp3Low ← temp3Low + temp1Low, L2Disp,                                 c2;
        uLastPointer ← temp3Low, RET [lastPointerOf-return],                    c3;




        { stabilize


                input:

                output:

                smash:   }

{Test the memory and oop levels and reset alreadyAlerted if both are above their alert levels.

((wordLevel >= wordAlertLevel) and: [oopLevel >= oopAlertLevel]) ifTrue: [alreadyAlerted ← 0]}

{linkage register is L0 -- runs only between bytecodes}

stabilize:
        temp1High ← uRumRecordHigh,                                             c1;
        temp1Low ← uRumRecordLow,                                               c2;
        uTimeToStabilize ← 0,                                                   c3;

        MAR ← [temp1High, temp1Low + stabilizationFlagOffset],                  c1;
        CANCELBR [$, 0],                                                        c2;
        temp3Low ← MD, XDisp,                                                   c3;

        {Molasses clears the stabilization flag on the initial call.  It is set to 1 below while stabilization is occurring and cleared
        when stabilization is completed.  If it is 1 now, we are restarting after a Mesa interrupt.}

        BRANCH [$, restoreStabilizeState, 0E],                                  c1;
        Noop,                                                                   c2;
        Noop,                                                                   c3;

testOopLevel:
        MAR ← [temp1High, temp1Low + oopLevelLowOffset],                        c1;
        CANCELBR [$, 0],                                                        c2;
        temp3Low ← MD,                                                          c3;

        MAR ← [temp1High, temp1Low + oopAlertLevelLowOffset],                   c1;
        CANCELBR [$, 0],                                                        c2;
        Q ← MD,                                                                 c3;

        Q ← temp3Low - Q, CarryBr,                                              c1;
        BRANCH [stabilize2, $],                                                 c2;
        Noop,                                                                   c3;

testWordLevelHigh:
        MAR ← [temp1High, temp1Low + wordLevelHighOffset],                      c1;
        CANCELBR [$, 0],                                                        c2;
        temp3Low ← MD,                                                          c3;

        MAR ← [temp1High, temp1Low + wordAlertLevelHighOffset],                 c1;
        CANCELBR [$, 0],                                                        c2;
        Q ← MD,                                                                 c3;

        Q ← temp3Low - Q, CarryBr,                                              c1;
        [] ← Q, ZeroBr, BRANCH [stabilize1, $],                                 c2;
        BRANCH [resetAlreadyAlerted, $],                                        c3;

testWordLevelLow:
        MAR ← [temp1High, temp1Low + wordLevelLowOffset],                       c1;
        CANCELBR [$, 0],                                                        c2;
        temp3Low ← MD,                                                          c3;
```

```
        MAR ← [temp1High, temp1Low + wordAlertLevelLowOffset],                  c1;
        CANCELBR [$, 0],                                                        c2;
        Q ← MD,                                                                 c3;

        Q ← temp3Low - Q, CarryBr,                                              c1;
        BRANCH [stabilize3, $],                                                 c2;
        Noop,                                                                   c3;

resetAlreadyAlerted:
        MAR ← [temp1High, temp1Low + alreadyAlertedOffset],                     c1;
        MDR ← 0, LOOPHOLE [wok], CANCELBR [$, 0],                               c2;
        GOTO [reallyStabilize],                                                 c3;


stabilize1:
        CANCELBR [reallyStabilize, 1],                                          c3;


stabilize2:
        GOTO [reallyStabilize],                                                 c3;


stabilize3:
        GOTO [reallyStabilize],                                                 c3;


reallyStabilize:
        MAR ← [temp1High, temp1Low + zctLowOffset],                             c1;
        CANCELBR [$, 0],                                                        c2;
        temp2Low ← MD,  {get address of the zct}                                c3;

        MAR ← [temp1High, temp1Low + zctHighOffset],                            c1;
        uZctBaseLow ← temp2Low, CANCELBR [$, 0],                                c2;
        Q ← temp2High ← MD,                                                     c3;

        uZctBaseHigh ← Q,                                                       c1;
        Noop,                                                                   c2;
        Noop,                                                                   c3;

        {get, then smash the zct index from the Rum record}
        MAR ← [temp1High, temp1Low + zctIndexOffset],                           c1;
        CANCELBR [$, 0],                                                        c2;
        temp3Low ← MD,                                                          c3;

        MAR ← [temp1High, temp1Low + zctIndexOffset],                           c1;
        MDR ← 0, LOOPHOLE [wok], CANCELBR [$, 0],                               c2;
        Noop,                                                                   c3;

        {reset the stabilization flag}
        MAR ← [temp1High, temp1Low +  stabilizationFlagOffset],                 c1;
        MDR ← 1 {stabilization in progress}, LOOPHOLE [wok], CANCELBR [$, 0],   c2;
        temp3Low ← temp3Low + temp2Low {yields low 16 bits of one word past
                the last valid oop in the zct},                                 c3;

        uZctLimit ← temp3Low,                                                   c1;
        uQueueHead ← 0, {should be nilPointer}                                  c2;
        uCurrentObject ← 0,     {should be nilPointer}                          c3;

{sweep the zct, for each oop marked (in its ot entry) as volatile, reset the isVolatile bit, and increase the reference counts of all of
its referents, recall that the zct index is one greater than the number of valid entries in the zct}

stabilizationLoop:
        [] ← temp2Low xor uZctLimit, ZeroBr {are we there yet?},                c1;
        temp3Low ← 0FF, BRANCH [$, countsAreNowCorrect],                        c2;
        temp3Low ← temp3Low LRot8,                                              c3;

        MAR ← [temp2High, temp2Low + 0],                                        c1;
        temp2Low ← temp2Low + 1, L1 ← stabilizing,                              c2;
        otLow ← MD, CALL [otMap2] {so we can get its delta word},               c3;

        temp1Low ← temp1Low + deltaWordOffset,                                  c1, at [stabilizing, 10, otMap2-return];
        temp3Low ← temp3Low or 0FB {yields FFFB, for turning off
                the isVolatile bit},                                            c2;
        Noop,                                                                   c3;

        MAR ← [temp1High, temp1Low + 0] {read delta word}                       c1;
        Noop,                                                                   c2;
        Q ← MD, XDisp {to test isVolatile bit},                                 c3;

        MAR ← [temp1High, temp1Low + 0], BRANCH [oopIsNotVolatile, $, 0B],      c1;
        Q ← MDR ← Q and temp3Low {not volatile anymore!},
                L2 ← stabilizingContext,                                        c2;
        temp1Low ← temp1Low - deltaWordOffset, CALL [lastPointerOf],            c3;

        {need to move the temp1 regs into temp3 to keep refi from smashing them...}
        Q ← temp1High,                                                          c1, at [stabilizingContext, 10, lastPointerOf-return];
        temp3High ← Q LRot0,                                                    c2;
        temp3Low ← temp1Low + classFieldOffset,                                 c3;

        {sweep over the volatile object, upping the reference counts of its referents}

upReferents:
        MAR ← [temp3High, temp3Low + 0], L1 ← correcting,                       c1;
        Noop,                                                                   c2;
        otLow ← MD, XDisp, CALL [refi],                                         c3;
```

```
        [] ← temp3Low xor uLastPointer, ZeroBr,                          c1, at [correcting, 10, refiReturn];
        temp3Low ← temp3Low + 1, BRANCH [$, thisOneIsStable],            c2;
        GOTO [upReferents],                                              c3;


oopIsNotVolatile:
        Noop,                                                            c2;
thisOneIsStable:
        temp2High ← uZctBaseHigh, GOTO [stabilizationLoop],              c3;


{at this point, all contexts have been stabilized, and all reference counts are correct.  sweep over the zct again: any object in the zct
whose reference count is zero is garbage!}

{temp2High is still valid despite the CALLs. restore temp2Low}

countsAreNowCorrect:
        temp2Low ←  uZctBaseLow,                                         c3;

sweepAndDeallocateLoop:
        [] ← temp2Low xor uZctLimit, ZeroBr {are we there yet?},         c1;
        temp2High ← uZctBaseHigh, BRANCH [$, returnFromStabilize],       c2;
        temp3Low ← inZctRot8,                                            c3;

        MAR ← [temp2High, temp2Low + 0] {get oop from zct}               c1;
        temp3Low ← ~temp3Low LRot8,                                      c2;
        otLow ← MD,                                                      c3;

        temp2High ← deallocating, CALL [getOtFlags],                     c1;

        temp1Low ← temp1Low and temp3Low {clear inZct}, CALL [putOtFlags],  c1, at [deallocating, 10, getFlagsReturn];

        temp3Low ← refCountRot8,                                         c1, at [deallocating, 10, putFlagsReturn];
        temp3Low ← temp3Low LRot8,                                       c2;
        Noop,                                                            c3;

        Noop,                                                            c1;
        [] ← temp1Low and temp3Low, ZeroBr,                              c2;
        temp2Low ← temp2Low + 1, BRANCH [sweepAndDeallocateLoop, $],     c3;

needToDeallocate:
        uZctSweepLow ← temp2Low,                                         c1;
        Noop,                                                            c2;
        GOTO [deallocate],                                               c3;

returnFromDeallocate:
        Noop,                                                            c1;
        Noop,                                                            c2;
        temp2Low ← uZctSweepLow, GOTO [sweepAndDeallocateLoop],          c3;


returnFromStabilize:
        Noop,                                                            c3;

        temp1High ← uRumRecordHigh,                                      c1;
        temp1Low ← uRumRecordLow,                                        c2;
        Noop,                                                            c3;

        MAR ← [temp1High, temp1Low + stabilizationFlagOffset],           c1;
        MDR ← 0 {completed}, LODisp, LOOPHOLE [wok], CANCELBR [$, 0],    c2;
        RET [stabilize-return],                                          c3;




        { deallocate

        Make an object whose reference count is zero reusable.

                input:  otLow is the object to deallocate
                        otHigh is the high part of the object table base address

                output:

                smash:  Q, uClass, L1, L2 }

deallocate:
        L2 ← startingDeallocate,                                         c1;
        Noop,                                                            c2;
        [] ← otLow LRot0, XDisp, CALL [getClass],                        c3;

        temp1Low ← temp1Low + deltaWordOffset,                           c1, at [startingDeallocate, 10, getClass-return];
        Q ← classCompiledMethodOop,                                      c2;
        [] ← temp3Low xor Q, ZeroBr,                                     c3;

        {get delta word to see if object has pointers}
        MAR ← [temp1High, temp1Low + 0],
                BRANCH [$, deallocatingACompiledMethod],                 c1;
        Noop,                                                            c2;
        Q ← MD, XLDisp,                                                  c3;

        BRANCH [deallocateWithNoPointers, deallocateWithPointers, 2],    c1;


deallocateWithNoPointers:
        temp1Low ← temp1Low - deltaWordOffset, L1 ← freeNonPointerObject,  c2;
```

```
        uClass ← temp3Low, CALL [adjustLevelsAndReturnToPool],                  c3;

        temp3Low ← uClass, GOTO [nowDoObjectsClass],                            c1, at [freeNonPointerObject, 10, addFreeChunkReturn];


deallocatingACompiledMethod:
        GOTO [deallocateWithPointersA],                                         c2;


deallocateWithPointers:
        Noop,                                                                   c2;
deallocateWithPointersA:
        temp1Low ← temp1Low + offsetFromDeltaWordToClassField,                  c3;

        {enqueue this object for deallocation}
        MAR ← [temp1High, temp1Low + 0],                                        c1;
        MDR ← uQueueHead,                                                       c2;
        uQueueHead ← otLow,                                                     c3;

        Noop,                                                                   c1;
nowDoObjectsClass:
        Noop,                                                                   c2;
        otLow ← temp3Low LRot0, XDisp, GOTO [specialRefd],                      c3;




{if there is a current object, continue with it.  if not, if there is
        a queued object, start it. otherwise we are done}

more:
        [] ← rInterrupt, ZeroBr, L4 ← 2,                                        c1, at [nowDoneWithObject, 10, addFreeChunkReturn];
        BRANCH [skipInterrupts, {CALL} serviceInterrupt],                       c2;

        BRANCH [saveStabilizeState, $],                                         c2, at [2, 10, interrupt-return];
        rInterrupt ← 20, GOTO [restartMore],                                    c3;


skipInterrupts:
        Noop,                                                                   c3;

restartMore:
        otLow ← uCurrentObject, ZeroBr,                                         c1;
        rInterrupt ← rInterrupt - 1, BRANCH [continueWithCurrentObject, $],     c2;
        otLow ← uQueueHead, ZeroBr,                                             c3;

        BRANCH [startWithQueueHead, $],                                         c1;
        Noop,                                                                   c2;
        GOTO [returnFromDeallocate], {all recursive freeing is now complete}    c3;


startWithQueueHead:
        uCurrentObject ← otLow, L1 ← sweepingObject,                            c2;
        CALL [otMap2],                                                          c3;

        temp1Low ← temp1Low + deltaWordOffset,                                  c1, at [sweepingObject, 10, otMap2-return];
        Q ← temp1High,                                                          c2;
        uSoFarHigh ← Q,                                                         c3;

        MAR ← [temp1High, temp1Low + 0],                                        c1;
        temp1Low ← temp1Low - deltaWordOffset, L2 ← getObjectEndForFreeing,     c2;
        Q ← MD, XDisp, {test pointers bit}                                      c3;

        BRANCH [doingACompiledMethod, notDoingACompiledMethod, OE],             c1;


        {both isCompiledMethod and doesHavePointers live in the lastPointerOf routine}
doingACompiledMethod:
        CALL [isCompiledMethod],                                                c2;


notDoingACompiledMethod:
        CALL [doesHavePointers],                                                c2;

        uCurrentObjectBaseLow ← temp1Low,                                       c1, at [getObjectEndForFreeing, 10,
        lastPointerOf-return];
        temp1Low ← temp1Low + classFieldOffset,                                 c2;
        Noop,                                                                   c3;

        MAR ← [temp1High, temp1Low + 0],                                        c1;
        Noop,                                                                   c2;
        temp2Low ← MD {link to next object on queue},                           c3;

        uQueueHead ← temp2Low,                                                  c1;
        Noop,                                                                   c2;
        Noop,                                                                   c3;

areWeThereYet:
        [] ← temp1Low xor temp3Low, ZeroBr,                                     c1;
        temp1Low ← temp1Low + 1, BRANCH [$, doneWithObject],                    c2;
        uSoFar ← temp1Low,                                                      c3;

        MAR ← [temp1High, temp1Low + 0],                                        c1;
        Noop,                                                                   c2;
        otLow ← MD {next field}, XDisp, GOTO [specialRefd],                     c3;
```

```
continueWithCurrentObject:
        temp1Low ← uSoFar,                                                      c3;

        temp1High ← uSoFarHigh,                                                 c1;
        temp3Low ← uLastPointer,                                                c2;
        GOTO [areWeThereYet],                                                   c3;


doneWithObject:
        Noop,                                                                   c3;

        otLow ← uCurrentObject,                                                 c1;
        temp1Low ← uCurrentObjectBaseLow, L1 ← nowDoneWithObject,               c2;
        uCurrentObject ← 0 {should be nilPointer},
                CALL [adjustLevelsAndReturnToPool],                             c3;




        { specialRefd

        Decrement the reference count of an object.  Nil, false, and true have permanently stuck counts, so skip them.  SmallIntegers
        don't have reference counts, so skip them too.

                input:  otLow is the object
                        otHigh is the high part of the object table base address
                        temp3Low is a negative one reference count mask
                        there is a pending XDisp to test for a smallInteger

                output:

                smash:  temp1High/Low, temp2High, temp3High, Q, L2 }

specialRefd:
        temp3Low ← refMinusOneRot8, DISP4 [specialRefdTable, 0C],               c1;


specialRefdOop01:
        [] ← falsePointer - otLow, CarryBr, GOTO [specialDoRefd],               c2, at [0D, 10, specialRefdTable];


specialRefdOop11:
        [] ← truePointer - otLow, CarryBr, GOTO [specialDoRefd],                c2, at [0F, 10, specialRefdTable];


specialRefdOop10:
        [] ← 1, ZeroBr, GOTO [specialDoRefd],                                   c2, at [0E, 10, specialRefdTable];


specialRefdSmallInteger00:
        GOTO [specialStuckRefd],                                                c2, at [0C, 10, specialRefdTable];


specialDoRefd:
        temp3Low ← temp3Low LRot8,
                BRANCH [$, skipSpecialRefd], {skip nil, false, true}            c3;

getSpecialRefCount:
        temp2High ← doingSpecialRefd, CALL [getOtFlags],                        c1;

        [] ← temp1Low LRot0, XHDisp {first part of stuck ref count test},       c1, at [doingSpecialRefd, 10, getFlagsReturn];
        Q ← ~temp3Low, BRANCH [$, negativeSpecialRefCount, 2],                  c2;
positiveSpecialRefCount:        {not stuck but could go to zero}
        temp1Low ← temp1Low + temp3Low {subtract 1}, CarryBr {no carry implies
                already zero, an error},                                        c3;

updateOtSpecialRefd:
        BRANCH [triedToSpecialRefdZeroCountObject, $],                          c1;
        Noop,                                                                   c2;
        Noop,                                                                   c3;

        CALL [putOtFlags],                                                      c1;

        [] ← temp1Low + temp3Low {subtract again}, CarryBr {no carry implies
                just went to zero},                                             c1, at [doingSpecialRefd, 10, putFlagsReturn];
        temp1Low ← temp1Low LRot8, BRANCH [specialNeedsDeallocation, $],        c2;
        GOTO [more],                                                            c3;


negativeSpecialRefCount:        {could be stuck but cannot go to zero}
        Q ← Q + 1, {refPlusOneRot8 LRot8}                                       c3;

        [] ← temp1Low + Q, CarryBr {carry implies stuck ref count},             c1;
        temp1Low ← temp1Low + temp3Low {subtract one from ref count},
                BRANCH [$, specialStuckRefd],                                   c2;
        [] ← 0, ZeroBr {force BRANCH}, GOTO [updateOtSpecialRefd],              c3;


specialNeedsDeallocation:       {ref count just went to zero}
        [] ← temp1Low and inZctRot8, ZeroBr,                                    c3;

        BRANCH [specialAlreadyInZct, $], {deallocate now if not in zct}         c1;
        Noop,                                                                   c2;
        GOTO [deallocate],                                                      c3;
```

```
specialAlreadyInZct:
        Noop,                                                   c2;
specialStuckRefd:
        GOTO [more],                                            c3;


skipSpecialRefd:
        GOTO [specialAlreadyInZct],                             c1;


triedToSpecialRefdZeroCountObject:
        Q ← specialRefdZero, GOTO [bailout3],                   c2;




        { adjustLevelsAndReturnToPool

        Put an object on the appropriate free list.  Adjust the word count and oop count to include the new free chunk.

                input:  otLow is the object
                        temp1High/Low is the object's address

                output:

                smash:  temp2High/Low, temp3Low, Q,}

adjustLevelsAndReturnToPool:
        temp1Low ← temp1Low + sizeFieldOffset,                  c1;
        temp2High ← uRumRecordHigh,                             c2;
        temp2Low ← uRumRecordLow,                               c3;

incrementOopLevel:
        MAR ← [temp2High, temp2Low + oopLevelLowOffset],        c1;
        CANCELBR [$, 0],                                        c2;
        temp3Low ← MD,                                          c3;

        MAR ← [temp2High, temp2Low + oopLevelLowOffset],        c1;
        MDR ← temp3Low ← temp3Low + 1, CarryBr, LOOPHOLE [wok],
                CANCELBR [$, 0],                                c2;
        BRANCH [increaseWordLevelLow, impossibleOopLevel],      c3;


impossibleOopLevel:
        Q ← tooManyOops, GOTO [bailout2],                       c1;


increaseWordLevelLow:
        MAR ← [temp1High, temp1Low + 0],                        c1;
        temp1Low ← temp1Low - sizeFieldOffset,                  c2;
        temp3Low ← MD,                                          c3;

        MAR ← [temp2High, temp2Low + wordLevelLowOffset],       c1;
        CANCELBR [$, 0],                                        c2;
        Q ← MD,                                                 c3;

        MAR ← [temp2High, temp2Low + wordLevelLowOffset],       c1;
        MDR ← Q + temp3Low, CarryBr, LOOPHOLE [wok], CANCELBR [$, 0],   c2;
        BRANCH [returnToPool, wordLevelCarry],                  c3;


wordLevelCarry:
        MAR ← [temp2High, temp2Low + wordLevelHighOffset],      c1;
        CANCELBR [$, 0],                                        c2;
        temp3Low ← MD,                                          c3;

        MAR ← [temp2High, temp2Low + wordLevelHighOffset],      c1;
        MDR ← temp3Low ← temp3Low + 1, LOOPHOLE [wok], CANCELBR [$, 0],  c2;
        Noop,                                                   c3;

returnToPool:
        {build a mask to set all ref count bits on and to set purpose bits to free (11)}
        temp3Low ← refCountRot8,                                c1;
        temp3Low ← temp3Low LRot8,                              c2;
        temp3Low ← temp3Low or freeOop,                         c3;

        temp2High ← returningToPool, temp2Low ← temp1Low, CALL [getOtFlags],    c1;

        temp1Low ← temp1Low or temp3Low, CALL [putOtFlags],     c1, at [returningToPool, 10, getFlagsReturn];

{upon entry, otLow is the oop of the object to add to the free list,
        temp1High/temp2Low must be that object's base, calls addToFreeChunkList
        thus smashing temp2High/Low and Q, smashes temp3Low}

addToProperFreeChunkList:
        temp1Low ← temp2Low + sizeFieldOffset,                  c1, at [returningToPool, 10, putFlagsReturn];
        Q ← largestFreeChunkSize,                               c2;
        Noop,                                                   c3;

        MAR ← [temp1High, temp1Low + 0],                        c1;
        temp1Low ← temp1Low - sizeFieldOffset,                  c2;
        temp3Low ← MD {object's size},                          c3;

        [] ← temp3Low - Q, CarryBr,                             c1;
        BRANCH [selectRegularList, selectBigFreeList],          c2;
```

```
selectRegularList:
        GOTO [addToFreeChunkList],                                  c3;

selectBigFreeList:
        temp3Low ← Q, GOTO [addToFreeChunkList],                    c3;



{ Edit history:

 }
```

```
{ ST80CoreInitialVariableVM.mc

CoreInitialVariableVM.mc modified for Smalltalk.

        by J Trow

        18-Nov-86 18:16:43

Copyright 1981, 1982, 1983, 1984, 1985, 1986 by Xerox Corporation.  All rights reserved. }



        { Smalltalk only runs on an 1108 workstation with CPE-FP and MCC-3584 boards.  Therefore, all references to Trident and ethernet
        initialization have been removed. }



        Reserve [ProtectStart, ProtectFence], Reserve [0FE0, 0FFF];      {save room for boot kernel}

        SetTask [0]

        StartAddress [go];

        Set [PilotMemoryBanks, 08];      {number of banks of real memory that Pilot knows about}



go:
        rB ← 80, CANCELBR [$, 0F],                                      c1; {+++}
        rB ← rB LRot8,                                                  c2; {+++}
        MCtl ← rB, {MCtl ← 8000}                                        c3; {+++}

        Bank ← 4, {MS[0..1] ← 1, Bank ← 0}                              c1; {+++}
        IOPCtl ← 0,                                                     c2;
        KCtl ← 0 {SAx000},                                              c3;

        DCtl ← 3 {display black, enable task},                          c1;
        PCtl ← 0,                                                       c2;
        EICtl ← 0,                                                      c3;

        EOCtl ← 0,                                                      c1;
        Noop,                                                           c2;
        CALL [mapInit],                                                 c3;

{map initialization goes on during this interval}

mapRet:
{clear all but the first two pages of bank 0}
        acR ← 0,                                                        c2;
        passTraps ← acR,                                                c3;

        rD ← 2,                                                         c1;
        rD ← rD LRot8,                                                  c2;
        rDrh ← 0,                                                       c3;

clear:  MAR ← [rDrh, rD+0],                                             c1;
        MDR ← acR, rD ← rD + 1, ZeroBr,                                 c2;
        BRANCH [clear, $],                                              c3;

        Noop,                                                           c1;
        acR ← OFF + 1,                                                  c2;
        uBootStart ← acR, GOTO [OnceOnlyInit],                          c3;


{OnceOnlyInit transfers control to DoneOnceOnlyInit.  DoneOnceOnlyInit lives in the device specific initial microcode.  When that
finishes, control passes to exitToEmulator.}


exitToEmulator:
        Noop,                                                           c1;
        rErh ← IOPageHigh,                                              c2;
DPYOff:         {make sure the display is off when the germ starts}
        acR ← RShift1 0, SE ← 1, {acR ← 8000}                           c3;

        rE ← uIOPage, {IOPage real address}                             c1;
        rB ← uMapPages,                                                 c2;
        rBrh ← 0,                                                       c3;

        MAR ← [rErh, IOPage.DSCB.syncCmd + 0],                          c1;
        MDR ← acR,                                                      c2;
        rB ← rB LRot8,                                                  c3;

SetVMMSize:
        MAR ← [rErh, IOPage.VMMSize + 0],                               c1;
        MDR ← rB,                                                       c2;
enableIOP:
        rB ← 0, {Set rB to its real value: 0}                           c3;

        MAR ← [rBrh, 0 + 0],                                            c1;
        MDR ← 0FF,                                                      c2;
        Noop,                                                           c3;

ST80CoreInitialVariableVM.mc        18-Nov-86 18:16:44 PST
```

```
        MAR ← [rBrh, 0 + 1],                                    c1;
        MDR ← uBootStart, CANCELBR [$, 0],                      c2;
        IOPCtl ← IOPInMode, GOTOABS [IdleLoc],                  c3;


{subroutines and end matter}

{Map initialization -- Sets up map and leaves next available page in topPage.  Start looking for pages before 768K, i.e., below bank 0C.
Does not map any pages in bank 0.

Beware -- clobbers the first word of each page in bank 0 even though you don't mean it!!!!!!!!!!!!!!!!!!!!!!!!

Write page number in the first word of each page.  Go through memory top down to give lower addresses precedence.

Register usage
acR     page number
rB      memory address register
rC      temporary
rD      temporary
rE      next available page}

mapInit:
        rBrh ← PilotMemoryBanks, {start in last bank}           c1; {+++}
        rB ← 0, {address within bank}                           c2;
        acR ← rB-1,                                             c3;

        passTraps ← acR, {catch faults}                         c1;
        acR ← PilotMemoryBanks,                                 c2; {+++}
        acR ← acR LRot8, {page counter}                         c3;

        Noop,                                                   c1;
        GOTO [mark1],                                           c2;


{mark the first word of each page with its page number}
markPages:
        MAR ← [rBrh, rB + 0],                                   c1;
        MDR ← acR,                                              c2;
mark1:
        acR← acR - 1, NegBr, {written all pages?}               c3;

        rC ← OFF + 1, BRANCH [$, mapBuild],                     c1;
        rC ← rB - rC, CarryBr,                                  c2;
        rB ← rC, BRANCH [$, markPages],                         c3;

        rB ← rBrh,                                              c1;
        rB ← rB - 1,                                            c2;
        rBrh ← rB LRot0,                                        c3;

        rB ← rC,                                                c1;
        Noop,                                                   c2;
        GOTO [markPages],                                       c3;


mapBuild:
        Noop,                                                   c2;
        Noop,                                                   c3;

        rBrh ← 1,                                               c1;
        rC ← rCrh ← 1,                                          c2;
        rB ← 0,                                                 c3;

{make all pages vacant}
        MAR ← [rBrh, rB + 0],                                   c1;
        MDR ← vacant,                                           c2;
        rErh ← 0F0,                                             c3;

{Do a trial Map operation using a large address.  If the Map doesn't trap, our CP board supports large virtual memory, and we set the VM
size to 23-bit.  If the Map traps, we will catch the trap and eventually return to mapOutOfBound, where we set the virtual memory size
to 22 bits for an older CP board.}
        Map ← [rErh, rE],                                       c1;
        acR ← 80,                                               c2;
        GOTO [largeVMOK],                                       c3;


mapOutOfBound: {use 22-bit vm}
        acR ← 40,                                               c3;

largeVMOK:
        uMapPages ← acR,                                        c1;
        rBrh ← 1,                                               c2;
        rC ← rCrh ← 1,                                          c3;

        acR ← acR LRot8, L0 ← 0A,                               c1;
        acR ← acR - 1,                                          c2;
        rB ← 0, GOTO [BLT],                                     c3;


{Set up the rE pair with the real address of the virtual memory map and the rB pair with the real address of the first location to be
mapped.  acR must contain the page number of the first real page to be mapped.}

        rErh ← MapRealAddrHigh,                                 c1, at[0A, 10, subrRet];
        rE ← MapRealAddrLow,                                    c2;
        rBrh ← rB ← FirstRealPageToMapHigh,                     c3;
```

```
        acR ← rB LRot8, {construct page number in acR}                    c1;
        rC ← FirstRealPageToMap,                                          c2;
        rB ← rC LRot8,                                                    c3;

        acR ← acR or rC,                                                  c1;
        Noop,                                                             c2;

{Check if the page has its own page number in the first location of the page.}
mapLoop3:
        Noop,                                                             c3;

mapLoop1:
        MAR ← [rBrh, rB + 0],                                             c1;
        Noop,                                                             c2;
        rC ← MD, {double bit error possible here}                         c3;

        [] ← rC xor acR, ZeroBr,                                          c1;
        rC ← IOPageHigh, BRANCH [nextReal3, $],                           c2;

{Check if this is the page number of the IOPage.  If so, then do not map this real page yet.  Go on to the next real page.}

        rC ← rC LRot8,                                                    c3;

        rC ← rC or IOPage,                                                c1;
        [] ← rC xor acR, ZeroBr,                                          c2;
        rC ← IOPageVirtual, BRANCH [NotIOPageReal, IsIOPageReal],         c3;


IsIOPageReal:
        Noop,                                                             c1;
        GOTO [nextReal3],                                                 c2;


{Check if this is the virtual map entry for the IOPage.  If so, map the IOPage to this map entry.  Do not increment the real page
number.}

NotIOPageReal:
        [] ← rC xor rE, NZeroBr,                                          c1;
        BRANCH [$, NotIOPageVirt],                                        c2;
MapIOPage:
        rC ← IOPage,                                                      c3;

        rC ← IOPage,                                                      c1;
        rC ← rC LRot8,                                                    c2;
        rC ← rC or IOPageHigh,                                            c3;

        MAR ← [rErh, rE + 0],                                             c1;
        MDR ← rC or present,                                              c2;
        rE ← rE + 1, GOTO [mapLoop1],                                     c3;


{Map this page in.}
NotIOPageVirt:
        topPage ← rB,                                                     c3;

        rB← rB or rBrh,                                                   c1;
        rC← rB,                                                           c2;
        rB← topPage,                                                      c3;

        MAR ← [rErh, rE + 0],                                             c1;
        MDR ← rC or present,                                              c2;
        rE ← rE + 1, GOTO [nextReal1],                                    c3;


nextReal3:
        Noop,                                                             c3;
nextReal1:
        acR ← acR + 1, GOTO [IncReal],                                    c1;


nextReal:
        Noop,                                                             c1;
IncReal:
        rC ← OFF + 1,                                                     c2;
        rC ← rB + rC, CarryBr,                                            c3;

        rB ← rBrh, BRANCH [$, nextBank],                                  c1;
        rB ← rC, GOTO [mapLoop3],                                         c2;


nextBank:
        rB ← rB + 1,                                                      c2;
        rD ← PilotMemoryBanks,                                            c3; {+++}

        Noop,                                                             c1; {+++}
        Noop,                                                             c2; {+++}
        [] ← rB xor rD, ZeroBr,                                           c3; {+++}

        rBrh ← rB LRot0, BRANCH [$, clearMem],                            c1;
        rB ← rC, GOTO [mapLoop3],                                         c2;


{clear all mapped pages}
clearMem:
        topPage ← rE, {save away}                                         c2;
        rE ← 0, {word offset}                                             c3;
```

```
        rD ← 0, {source of zero}                                        c1;
        passTraps ← rD, {die on double bit errors}                      c2;
        Noop,                                                           c3;

clearLoop:
        MAR ← [rErh, rE + 0], {read the map}                            c1;
        rC ← 0,                                                         c2;
        acR ← MD,                                                       c3;

        rB ← acR and 0F,                                                c1;
        rBrh ← rB LRot0,                                                c2;
        rB ← acR and ~0FF,                                              c3;

{write into every word of the page}
clearPage:
        MAR ← [rBrh, rC + 0],                                           c1;
        MDR ← rD, rC ← rC + 1, PgCarryBr,                               c2;
        acR ← topPage, BRANCH [clearPage, $],                           c3;

        rE ← rE + 1,                                                    c1;
        [] ← rE xor acR, ZeroBr, {compare to topPage}                   c2;
        BRANCH [clearLoop, $],                                          c3;

{write 0 to locations 20000 through FFFFF}
clearExternal:
        rDrh ← 2, {start at end of map (20000)}                         c1; {+++}
        rD ← 0,                                                         c2; {+++}
        rC ← 0, {source of zero}                                        c3; {+++}

clearLoopExternal:
        MAR ← [rDrh, rD + 0],                                           c1; {+++}
        MDR ← rC, rD ← rD + 1, CarryBr,                                 c2; {+++}
        Q ← rDrh, BRANCH [clearLoopExternal, $],                        c3; {+++}

        rD ← Q + 1,                                                     c1; {+++}
        rDrh ← rD LRot0,                                                c2; {+++}
        [] ← rD xor acR, ZeroBr, {last bank?}                           c3; {+++}

        rD ← 0, BRANCH [$, clearHighMem],                               c1; {+++}
        acR ← 10, {end value}                                           c2; {+++}
        GOTO [clearLoopExternal],                                       c3; {+++}

{write 0 to locations 800000 through 8BFFFF}
clearHighMem:{+++}
        Noop,                                                           c2; {+++}
        acR ← 8C,{end value}                                            c3; {+++}

        rDrh ← 80, {start at end of gap (800000)}                       c1; {+++}
        rD ← 0,                                                         c2; {+++}
        rC ← 0, {source of zero}                                        c3; {+++}

clearLoopHighMem:
        MAR ← [rDrh, rD + 0],                                           c1; {+++}
        MDR ← rC, rD ← rD + 1, CarryBr,                                 c2; {+++}
        Q ← rDrh, BRANCH [clearLoopHighMem, $],                         c3; {+++}

        rD ← Q + 1,                                                     c1; {+++}
        rDrh ← rD LRot0,                                                c2; {+++}
        [] ← rD xor acR, ZeroBr, {last bank?}                           c3; {+++}

        rD ← 0, BRANCH [$, mapRet],                                     c1; {+++}
        Noop,                                                           c2; {+++}
        GOTO [clearLoopHighMem],                                        c3; {+++}


{trap catcher, gets here with rC ← RRot1 ErrnIBnStkp}
error:
        Xbus ← rC LRot0, XwdDisp,                                       c2, at [ErrorHandlerLoc];
        DISP2 [errorType],                                              c3;


        GOTO [death], {control store parity error}                      c1, at [0, 4, errorType];
        Xbus ← MStatus, XLDisp, GOTO [memFault],                        c1, at [1, 4, errorType];
        GOTO [death], {stack error}                                     c1, at [2, 4, errorType];
        GOTO [death], {instruction buffer empty}                        c1, at [3, 4, errorType];


death:
        GOTO [death],                                                   c*;


memFault:
        BRANCH [mapOutOfBound, $, 1], {map out of bound?}               c2;
        GOTO [nextReal], {no, double bit error}                         c3;




{block transfer, takes count in acR, from in rB, and to in rC, returns first word past from block in rE}
BLT2:
        Noop,                                                           c2;
BLT3:
        Noop,                                                           c3;

BLT:    MAR ← [rBrh, rB + 0],                                           c1;
        [] ← acR, ZeroBr,                                               c2;
```

```
        rE ← MD, BRANCH [$, endBLT],                    c3;

        MAR ← [rCrh, rC + 0],                           c1;
        MDR ← rE,                                       c2;
        acR ← acR - 1,                                  c3;

        rB ← rB + 1,                                    c1;
        rC ← rC + 1,                                    c2;
        GOTO [BLT],                                     c3;


endBLT:
        Noop,                                           c1;
endBLT1:
        pRet0,                                          c2;
endBLT2:
        RET [subrRet],                                  c3;
```

```
--
-- Package: Smalltalk80 Object Memory Implementation.
-- Part A: Procedures for initialization, reference counting,
-- garbage collection, and allocating and deallocating objects.
--

-- last edited by Malcolm
-- 19-Jun-87 11:18:02    cosmetic
-- 18-Jun-87 18:33:54    root test
-- 18-Jun-87 18:05:59    LastOffsetOf [LastPointerOf - 1]
-- 18-Jun-87 16:54:12    debugging
-- 18-Jun-87 15:05:10    iterative Deallocate
-- 29-Apr-86 14:48:58    use memoryHoleRealStart
-- 26-Mar-86 14:26:12    simplify AddToZeroCountTable; countDowns statistics
-- 10-Jan-86 14:40:58    check class in Deallocate
-- 15-Oct-85 19:09:19    oop~Level ptr to CARD [not long]
--  7-Oct-85 17:10:26    use RefI, RefD
--  7-Oct-85 11:55:51    speed AddToZeroCountTable
-- 17-Sep-85 14:47:56    cleanup/speed CountDown, Deallocate, ...
--  9-Sep-85 18:10:55    refI, refD speed
--  6-Sep-85 19:16:05    "G
-- 30-Aug-85 11:48:29    startInit in CreateInstance, Allocate
-- 15-Aug-85 18:13:53    use OtIndexOf[oop] > 3 in refI refD
-- 26-Jul-85 16:53:12    minor speed ups - refI, refD
--  8-Jul-85 17:16:28    USING for all
--  2-Jul-85 19:00:14    move memoryAlertSemaphore to D
--  6-Jun-85 14:03:38    oop > truePointer [instead of 6]
--  3-Apr-85 19:32:40    begin Stretch [use OopFromOtIndex]

-- 13-Feb-85 13:47:23    debug flgs in ST80DebugFlags
--  8-Feb-85 18:42:35    explicit init of internal ptrs
--  8-Feb-85 14:51:25    use TRUE instead of debugRefCountFlg
--  8-Feb-85 12:30:53    SetRefCountOf
--  4-Feb-85 17:25:18    Fetch, Smash
--  2-Feb-85 16:44:16    check CountDown past 0
-- 30-Jan-85 17:15:44    move allocation and lists to ST80MemImpl0
-- 29-Jan-85 19:52:29    adjust for memory hole
-- 28-Jan-85 14:02:20    removed some old code in comments
-- 18-Jan-85 11:33:35    check < minSize before try to AddToFreeChunkList
-- 16-Jan-85 18:50:33    size checks for lists; link using Copy*to*
-- 15-Jan-85 16:53:42    currentChunkAddress calculated only when valid
-- 15-Jan-85 16:47:53    tests for SmallInteger oop in AddToFreeChunkList
-- 12-Jan-85 18:01:48    removed use of SmashLONG
-- 17-Dec-84 15:41:51    move Stabilize, Volatilize, MakeVolatile to st80MemImpl0
-- 17-Dec-84 11:12:59    BITAND, BITOR removed
-- 15-Dec-84 13:14:51    SmashOTEntry
-- 15-Dec-84 13:00:49    fix OTEntry refs
-- 11-Dec-84 19:01:13    ptr shufflihg using PMem*
-- 11-Dec-84 17:56:10    lastOop -> lastOopIndex
-- 11-Dec-84 13:26:21    [1..lastOop]
--  7-Dec-84 13:46:58    memory accesses in
--      AddToTemporaryFreeList RemoveFromTemporaryFreeList
--  6-Nov-84 14:30:23

DIRECTORY
   Runtime USING [CallDebugger],
   Inline USING [LowHalf],
   ST80Send USING [literalStart],
   ST80Defs USING [Address, AddressOf, At, classCompiledMethodPointer, DeltaWord, falsePointer,
lastOopIndex, Loc, maxCount, nilPointer, objectFieldBase, objectMemory, objectMemoryPages, Of, Oop,
OopFromOtIndex, OopIsSmallInteger, OTEntry, OtEntry, OtIndexOf, Physical, PhysicalAddress, PMemFetch,
PMemSmash, PMemSmashInvisibly, RealFetch, RealSmash, SetOTAddressOf, SmallInteger, SmashOTEntry,
truePointer, Virtual, VirtualAddress, Word],
   ST80Debug USING [LogRefCountDown, LogAdditionToZCT, watchOopFlg, watchedOop],
   ST80DebugFlags,
   ST80Mem USING [AddToFreePointerList, AddToProperFreeChunkList, Allocate, bigSize, chunkLinkOffset,
FetchClass, IntegerObjectOf, IsClass. IsIntegerObject, minSize, nonAddress, nonPointer,
objectClassOffset, ObjectFieldType, objectSizeOffset, RefD, RefI, RemoveFromFreePointerList,
RemoveFromTemporaryFreeList, StackTop, tempSizeOffset, UnaryPrimReturn, Volatilize, zctIndexLimit,
zctSize],
   ST80Pilot USING[imageSize, memoryHoleSize, memoryHoleRealStart],
   ST80Rum USING [common, Stabilize],
   ST80Stretch USING [StretchHeader];

ST80MemImpl1A: PROGRAM
IMPORTS Runtime, Inline, ST80Pilot,

ST80MemImpl1A.mesa        19-Jun-87 11:21:25 PDT                    1
```

```
    ST80Debug, ST80Rum, ST80Mem, ST80Defs
EXPORTS ST80Mem = BEGIN
OPEN ST80Defs, ST80DebugFlags, ST80Mem,
    DBug: ST80Debug, Pilot: ST80Pilot, Rum: ST80Rum, Send: ST80Send;

    --
    -- Important constants
    --

    -- Controls conditional compilation to preclude actual deallocation
    -- of objects

    noDeallocateFlg: BOOLEAN = FALSE;


    --
    -- Internal state of the Object Memory
    --

    -- Initialize them explicitly in StartMemModule
    -- to insure that @Rum.common is correct

    -- Note that our implementation departs from the book in that there
    -- is no segmentation of real memory.  The standard free lists are
    -- indirect in order to make them efficiently accessible from Rum
    -- (they are part of the Rum/Molasses common area.)

    freePointerListHead: PUBLIC LONG POINTER TO Oop;
    freeChunkListHeads: PUBLIC LONG POINTER TO ARRAY [0..bigSize] OF Oop;
    tempFreeChunkListHead: PUBLIC VirtualAddress ← nonAddress;

    -- Zero count table and associated apparatus

    zctIndex: PUBLIC LONG POINTER TO CARDINAL;
    zeroCountTable: PUBLIC LONG POINTER TO ARRAY (0..zctSize] OF Oop;
    stabilizeFlg: PUBLIC LONG POINTER TO BOOLEAN;

    -- Word and Oop lower limits which, when reached, should cause a
    -- signal to the designated semaphore.  The alert levels are
    -- indirect in order to make them efficiently accessible from Rum
    -- (they are part of the Rum/Molasses common area.)

    wordAlertLevel: PUBLIC LONG POINTER TO LONG CARDINAL;
    oopAlertLevel: PUBLIC LONG POINTER TO CARDINAL;
    alreadyAlerted: PUBLIC LONG POINTER TO Word;

    -- Registers containing the number of Oops and words remaining
    -- in the object memory.  These are indirect in order to make them
    -- efficiently accessible from Rum (they are part of the Rum/Molasses
    -- common area.)

    wordLevel: PUBLIC LONG POINTER TO LONG CARDINAL ← @Rum.common.wordLevel;
    oopLevel: PUBLIC LONG POINTER TO CARDINAL;  -- ← @Rum.common.oopLevel;


    --
    -- Initialization:
    --

    StartMemModule: PUBLIC PROCEDURE =
        BEGIN

        -- Believe it or leave it, this is the best way of getting this
        -- module STARTed.  The "right" way involves importing this PROGRAM
        -- into some other PROGRAM, which causes unwanted binding/compilation
        -- dependencies.

        -- initialize lists, etc., explicitly
        -- since timing of init @Rum.common is uncertain

        freePointerListHead ← @Rum.common.freePointers;
        freeChunkListHeads ← @Rum.common.freeLists;
        zctIndex ← @Rum.common.zctIndex;
        stabilizeFlg ← @Rum.common.stabilizationNeeded;
        wordAlertLevel ← @Rum.common.wordAlertLevel;
        oopAlertLevel ← LOOPHOLE[@Rum.common.oopAlertLevel];
```

```
    alreadyAlerted ← @Rum.common.alreadyAlerted;
    wordLevel ← @Rum.common.wordLevel;
    oopLevel ← LOOPHOLE[@Rum.common.oopLevel];


    END;


InitializeObjectMemory: PUBLIC PROCEDURE =
    BEGIN

    -- This procedure sets up the free pointer list and free chunk
    -- lists.

    newChunkOop: Oop ← NULL;
    chunkSize: CARDINAL ← NULL;
    longChunkSize: LONG CARDINAL ← NULL;
    otEntry: OTEntry;
    virtualHoleStart: VirtualAddress = Virtual[Pilot.memoryHoleRealStart];

    -- The largest possible chunk must leave room in its segment for
    -- at least one other object (otherwise, we would leave holes
    -- between the segments - not nice.)

    maxInitialBigChunkSize: LONG CARDINAL =
        LONG[LAST[CARDINAL]] + 1 - minSize;

    -- Determine the size and location of the primordial free chunk.

    remainderSize: LONG CARDINAL ←
        (LONG[objectMemoryPages] * LONG[256]) - Pilot.imageSize;
    remainderAddress: VirtualAddress ← objectMemory +
        Pilot.imageSize;
    -- adjust for extended memory high bit
    IF remainderAddress > virtualHoleStart THEN
        remainderAddress ← remainderAddress + Pilot.memoryHoleSize;

    -- Initialize the volatile context stabilization apparatus:

    zctIndex↑ ← 0;
    stabilizeFlg↑ ← FALSE;

    -- Initialize the memory counters and alert levels

    wordLevel↑ ← LONG[0];
    wordAlertLevel↑ ← LONG[0];
    Rum.common.oopLevel ← LONG[0];  -- in case Rum looks at high bits
    Rum.common.oopAlertLevel ← LONG[0];  -- ditto

    -- Initialize the free pointer list.

    freePointerListHead↑ ← nonPointer;

    -- Find the free entries in the OT and add them to the free
    -- pointer list.  LOOM Sensitive.

    FOR otIndex: CARDINAL DECREASING IN [1..lastOopIndex] DO

        -- Checking whether the reference count is zero is
        -- superfluous since there should be no free objects
        -- at this point.

        IF OtEntry[At[otIndex]].purpose = free
        THEN AddToFreePointerList[OopFromOtIndex[otIndex]];

        ENDLOOP;

    -- Initialize the free chunk lists to be empty.

    FOR size: CARDINAL IN [0..bigSize] DO
        freeChunkListHeads[size] ← nonPointer;
        ENDLOOP;

    -- There will have been some segment breakage placed on a
    -- temporary free list during the loading process.  Shuffle
    -- this onto the appropriate free lists.  We are guaranteed
```

```
      -- that none of these remainders is of segment size, therefor
      -- no need to futz with LONG quantities.

      WHILE tempFreeChunkListHead # nonAddress DO
         start: VirtualAddress ← RemoveFromTemporaryFreeList[];
         oop: Oop ← RemoveFromFreePointerList[];
         size: CARDINAL ← PMemFetch[start + tempSizeOffset];

         IF size < minSize THEN Runtime.CallDebugger[
            "Attempt to link too small a chunk into freeChunkList."G];

         -- Make the Oop point at the chunk.  Give the Oop an artificial
         -- reference count to distinguish it from a free Oop.

         SetOTAddressOf[oop, start];
         otEntry ← OtEntry[Of[oop]];
         otEntry.referenceCount.count ← maxCount;
         SmashOTEntry[Of[oop], otEntry];

         -- Mark the chunk properly with its size.

         PMemSmash[start + objectSizeOffset, size];

         -- Distribute the chunk onto the proper free list.

         AddToProperFreeChunkList[oop: oop, address: start, size: size];
         ENDLOOP;

      -- Use the primordial free chunk to finish initializing the
      -- free chunk lists.  We want to guarantee that no chunk will
      -- cross a segment boundary.  On the other hand, no chunk can
      -- be of segment size.  Therefore, in each segment, carve out
      -- a very large  chunk, followed by a minimum size chunk.

      WHILE remainderSize > 0
         DO

         -- Determine the size of the next initial chunk and
         -- mark the chunk with its size before cutting it out.

         longChunkSize ← MIN[
            LONG[SegmentRemainder[remainderAddress]] + 1,
            remainderSize,
            maxInitialBigChunkSize];
         chunkSize ← Inline.LowHalf[longChunkSize];
         PMemSmashInvisibly[remainderAddress + objectSizeOffset,
            chunkSize];

         -- Acquire an Oop for the new initial chunk, mark the Oop
         -- not free and set it to point at the chunk.

         newChunkOop ← RemoveFromFreePointerList[];
         SetOTAddressOf[newChunkOop, remainderAddress];

         -- Give the Oop to the free chunk an artificial reference
         -- count so it won't be confused with a free Oop. LOOM
         -- Sensitive.

         otEntry ← OtEntry[Of[newChunkOop]];
         otEntry.referenceCount.count ← maxCount;
         SmashOTEntry[Of[newChunkOop], otEntry];

         -- Add the new chunk to the appropriate free chunk list, and
         -- debit the remainder of the primordial chunk.

         AddToProperFreeChunkList[oop: newChunkOop, address:
            remainderAddress, size: chunkSize];
         remainderAddress ← remainderAddress + longChunkSize;
         -- adjust for real memory address hole
         IF remainderAddress = virtualHoleStart THEN
            remainderAddress ← remainderAddress + Pilot.memoryHoleSize;
         remainderSize ← remainderSize - LONG[chunkSize];
         ENDLOOP;

      -- Determine the number of Oops and words available in the
      -- newly initialized memory.
```

```
    [oopLevel↑, wordLevel↑] ← StorageRemaining[];
    END;


--
-- Initialization support:
--

StorageRemaining: PROCEDURE RETURNS
    [totalOops: CARDINAL, totalWords: LONG CARDINAL] =
    BEGIN

    -- Determine how many unallocated Oops words remain in the object
    -- memory.

    smallChunk: Oop ← NULL;
    smallChunkCount: CARDINAL ← NULL;
    bigChunk: Oop ← NULL;
    bigChunkAddress: VirtualAddress ← NULL;
    freeOop: Oop ← freePointerListHead↑;
    totalOops ← 0;
    totalWords ← 0;

    -- Add up the Oops on the free Oop list.

    WHILE freeOop # nonPointer DO
        totalOops ← totalOops + 1;
        freeOop ← LOOPHOLE[OtEntry[Of[freeOop]].offsetLow, Oop];
        ENDLOOP;

    -- Add up the words and Oops entailed in uniform-size small
    -- chunks.

    FOR smallChunkSize: CARDINAL IN [minSize..bigSize) DO
        smallChunk ← freeChunkListHeads[smallChunkSize];
        smallChunkCount ← 0;
        WHILE smallChunk # nonPointer DO
            smallChunkCount ← smallChunkCount + 1;
            smallChunk ← LOOPHOLE[PMemFetch[
                AddressOf[smallChunk] + chunkLinkOffset],
                Oop];
            ENDLOOP;
        totalOops ← totalOops + smallChunkCount;
        totalWords ← totalWords + LONG[smallChunkSize] *
            LONG[smallChunkCount];
        ENDLOOP;

    -- Add up the words entailed in non-uniform big chunks.

    bigChunk ← freeChunkListHeads[bigSize];
    WHILE bigChunk # nonPointer DO
        bigChunkAddress ← AddressOf[bigChunk];
        totalOops ← totalOops + 1;
        totalWords ← totalWords +
            LONG[PMemFetch[bigChunkAddress + objectSizeOffset]];
        bigChunk ← LOOPHOLE[PMemFetch[
            bigChunkAddress + chunkLinkOffset], Oop];
        ENDLOOP;

    RETURN[totalOops, totalWords];
    END;


-- Public because it is needed for segment boundary alignment while
-- loading

SegmentRemainder: PUBLIC PROCEDURE [virtual: VirtualAddress] RETURNS
    [CARDINAL] =
    BEGIN

    -- Say how many words remain in virtual's segment, counting from,
    -- but not including, virtual.

    real: PhysicalAddress ← Physical[virtual];
    RETURN[177777B - Inline.LowHalf[real]];
```

```
        END;


    --
    -- Reference counting
    --

    IncreaseReferences: PUBLIC PROCEDURE [oop: Oop] = {RefI[oop]};


    DecreaseReferences: PUBLIC PROCEDURE [oop: Oop] = {RefD[oop]};


    AddToZeroCountTable: PUBLIC PROCEDURE [new: Oop] =
       BEGIN

       -- Add the Oop to the ZCT for later possible deallocation or
       -- processing as a volatile context, but only do this if it
       -- is not already there.

       otPtr: LONG POINTER TO OTEntry = Of[new];

       IF ~otPtr.inZeroCountTable
       THEN BEGIN
          index: CARDINAL = zctIndex↑ + 1;

          -- Inserted for debugging purposes
          IF debugFlg
          AND new = DBug.watchedOop
          AND DBug.watchOopFlg
          THEN DBug.LogAdditionToZCT[];

          IF debugFlg
          AND index > zctSize
          THEN Runtime.CallDebugger["Zero count table overflow"G];

          otPtr.inZeroCountTable ← TRUE;
          zctIndex↑ ← index;
          zeroCountTable[index] ← new;

          IF index > zctIndexLimit
          THEN stabilizeFlg↑ ← TRUE;
          END;
       END;


    --
    -- Deallocation:
    --

-- for debug access
interruptPending: BOOLEAN ← FALSE;

Deallocate: PUBLIC PROCEDURE [freeing: Oop]
 = BEGIN
    -- free this oop and Deallocate any of its fields that refD to 0.
    -- essentially recursive, freeing a tree based at oop.
    -- this is an 'iterative' version; recursion state is hidden in the tree.
    -- last is offset of next field to deal with; held in delta word.
    -- last is descending, so don't have to remember stop index.
    -- when going another level, link back is put in this last field.
    -- (which formerly held reference to the oop for the new level)

    ENABLE ANY => Runtime.CallDebugger["Deallocate problem"L];

    LastOffsetOf: PROC [o: Oop] RETURNS [l: CARDINAL]
     = INLINE {l ← LastPointerOf[o] - 1};
    RefDCount: PROC [o: Oop] RETURNS [c: CARDINAL]
     = INLINE {RefD[o]; c ← oep.referenceCount.count};

    next: Oop ← nilPointer;  -- parent object of freeing tree
    oep: LONG POINTER TO OTEntry ← Of[freeing];  -- for freeing or ref
    loc: PhysicalAddress ← Loc[oep];  -- start of object header
    last: CARDINAL;  -- offset of last reference field; decremented
    ref: Oop;  -- object referenced in 'last' field
```

```
    IF OopIsSmallInteger[freeing]
    OR oep.purpose # inUse
    OR oep.referenceCount.count # 0
    THEN RETURN;

    last ← LastOffsetOf[freeing];

    WHILE TRUE  -- RETURN in nested loop
    DO
       -- at this point freeing is ready to peel off another reference
       -- or suspend for interrupt, with state minimized to freeing and next

       IF interruptPending
       THEN BEGIN
          RealSmash[loc, last];  -- remember where to pick up
          -- SaveFreeingAndNext;
          -- DeferToMesa;
          -- RestoreFreeingAndNext;
          loc ← Loc[Of[freeing]];
          last ← RealFetch[loc];
          END;

       ref ← RealFetch[loc + last];
       oep ← Of[ref];

       IF OopIsSmallInteger[ref]
       OR oep.referenceCount.count = maxCount
       OR RefDCount[ref] > 0
       THEN last ← last - 1
       ELSE BEGIN
          -- this ref must be (recursively) deallocated, ie become freeing
          RealSmash[loc, last];  -- remember field to resume in freeing
          RealSmash[loc + last, next];  -- and how to go back up tree

          -- change levels & regenerate locals
          next ← freeing;
          freeing ← ref;
          loc ← Loc[Of[freeing]];
          last ← LastOffsetOf[freeing];
          END;

       -- only loops in case class gets deallocated
       -- multiple loop probably impossible or at least astronomically rare
       WHILE last < objectClassOffset
       DO
          oep ← Of[freeing];  -- oep no longer needed for ref

          -- adjust OTEntry for free object
          oep.purpose ← free;
          oep.referenceCount.count ← maxCount;

          -- and link its chunk into free list (and update words, oops left)
          AddToProperFreeChunkList[
             oop: freeing,
             address: Address[oep],
             size: RealFetch[loc + objectSizeOffset]];

          -- go back up the tree
          freeing ← next;
          IF freeing = nilPointer THEN RETURN;  -- finished traversing tree

          loc ← Loc[Of[freeing]];
          last ← RealFetch[loc];
          next ← RealFetch[loc + last];
          last ← last - 1;
          ENDLOOP;
       ENDLOOP;
    END;

    <<
    Deallocate: PUBLIC PROCEDURE [freeing: Oop] =
       BEGIN

       -- Free an object, and as well all other objects to which it
       -- refers (concatenatively) uniquely.  This is the recursive
       -- version.
```

```
      otPtr: LONG POINTER TO OTEntry = Of[freeing];
      loc: PhysicalAddress ← Loc[otPtr];

      -- Inserted for debugging purposes
      IF debugFlg
      AND freeing = DBug.watchedOop
      AND DBug.watchOopFlg
      THEN Runtime.CallDebugger["Watched Oop is being deallocated"G];

      IF debugFlg
      AND NOT IsClass[FetchClass[freeing]]
      THEN Runtime.CallDebugger["Bad class field"G];

      IF noDeallocateFlg
      THEN BEGIN
         ChasePointersAdjustingReferences[from: freeing];
         RETURN[];
         END;

      -- Chase all the references from the object being freed.

      FOR offset: CARDINAL
      IN [objectClassOffset..LastPointerOf[object: freeing])
      DO CountDown[RealFetch[loc + offset]] ENDLOOP;

      -- Mark the chunk as free by setting its purpose code and giving
      -- it an artificial reference count (to distinguish it from a free Oop.)

      otPtr.purpose ← free;
      otPtr.referenceCount.count ← maxCount;
      --SmashOTEntry[Of[freeing], freeingOTEntry];

      -- Put the current free chunk where it belongs.

      AddToProperFreeChunkList[oop: freeing, address: Address[otPtr],
         size: RealFetch[loc + objectSizeOffset]];

      END;
   >>


   LastPointerOf: PUBLIC PROCEDURE [object: Oop] RETURNS [CARDINAL] =
      BEGIN

      -- Note that this returns the number of pointer fields counting
      -- from the very start of the object (including the delta word)
      -- rather than the offset of the last pointer field.  The first
      -- edition of the book contains a bug: it asserts that compiled
      -- methods have the pointer bit set.

      loc: PhysicalAddress ← LOOPHOLE[OtEntry[Of[object]]];
      delta: DeltaWord ← RealFetch[loc];

      IF delta.hasPointers
      THEN RETURN[RealFetch[loc + objectSizeOffset]]
      ELSE BEGIN
         class: Oop = RealFetch[loc + objectClassOffset];

         IF class = classCompiledMethodPointer
         THEN BEGIN
            header: ST80Stretch.StretchHeader ← RealFetch[loc + objectFieldBase];

            RETURN[header.lits + Send.literalStart + objectFieldBase];
            END
         ELSE RETURN[objectFieldBase];
         END;
      END;


   -- for gathering statistics
   countDowns: BOOLEAN ← FALSE;
   ints: LONG CARDINAL ← 0;
   nils: LONG CARDINAL ← 0;
   falses: LONG CARDINAL ← 0;
   trues: LONG CARDINAL ← 0;
```

```
    oops: LONG CARDINAL ← 0;

    CountDown: PROCEDURE [oop: Oop] =
       BEGIN
       -- Same as DecreaseReferences,
       -- except actually Deallocate rather than add to zct

       index: CARDINAL ← NULL;

       -- Inserted for debugging purposes
       IF debugFlg
       AND oop = DBug.watchedOop
       AND DBug.watchOopFlg
       THEN DBug.LogRefCountDown[];

       IF countDowns
       THEN BEGIN
          IF OopIsSmallInteger[oop]
          THEN ints ← ints + 1
          ELSE oops ← oops + 1;  -- includes nil, false, true

          IF oop = nilPointer THEN nils ← nils + 1;
          IF oop = falsePointer THEN falses ← falses + 1;
          IF oop = truePointer THEN trues ← trues + 1;
          END;

       IF ~OopIsSmallInteger[oop]
       AND (index ← OtIndexOf[oop]) > 3
       THEN BEGIN
          otLoc1: PhysicalAddress = Physical[At[index]] + 1;
          hiOtWord: CARDINAL ← RealFetch[otLoc1];

          IF hiOtWord < 176000B  -- not stuck
          THEN IF hiOtWord < 2000B
             THEN Runtime.CallDebugger["Decrement of zero reference count"G]
             ELSE BEGIN
                RealSmash[otLoc1, hiOtWord - 2000B];  -- decr by 1

                IF hiOtWord < 4000B  -- ref cnt was 1
                AND hiOtWord < 3000B -- not in zct already
                THEN Deallocate[oop];
                END;
          END;
       END;


    --
    -- Reference counting debug assists:
    --

    ChasePointersAdjustingReferences: PROCEDURE [from: Oop] =
       BEGIN

       -- This is a stand-in for Deallocate, so that when debugging
       -- reference counting problems we can go into a mode in which
       -- nobody is actually deallocated.

       objectAddress: VirtualAddress = AddressOf[from];
       FOR offset: CARDINAL IN
          [objectClassOffset..LastPointerOf[object: from]) DO
          CountDown[LOOPHOLE[PMemFetch[objectAddress + offset], Oop]];
          ENDLOOP;
       END;

    -- In order to allow inspection of reference counts from Smalltalk,
    -- the following primitive is provided.

    PrimitiveRefCount: PUBLIC PROCEDURE =
       BEGIN

       -- Return the reference count of the receiver, subtracting
       -- one to account for the reference from the active context.

       receiver: Oop = StackTop[];
       refCount: SmallInteger ← NULL;
       IF IsIntegerObject[receiver] THEN refCount ← 0
```

```
      ELSE
         BEGIN
         Rum.Stabilize[];
         refCount ← OtEntry[Of[receiver]].referenceCount.count;
         IF refCount # maxCount THEN refCount ← refCount - 1;
         Volatilize[];
         END;
      UnaryPrimReturn[IntegerObjectOf[refCount]];
      END;


   --
   -- Object creation:
   --

   CreateInstance: PUBLIC PROCEDURE [classOop: Oop, fieldType:
      ObjectFieldType, fields: CARDINAL, startInit: CARDINAL ← 0] RETURNS [Oop] =
      BEGIN
      size: CARDINAL ← NULL;
      new: Oop ← NULL;
      SELECT fieldType FROM
         pointer =>
            BEGIN

            -- Note there is no need to insert an extra word at the
            -- end of very large pointer objects, because the microcode
            -- will use Glenn's tail recursive algorithm (which chains
            -- via the class field) and Molasses uses full recursion.

            size ← fields + objectFieldBase;
            new ← Allocate[size: size, odd: 0, pointers: TRUE,
               class: classOop, startInit: startInit];
            END;
         word =>
            BEGIN
            size ← fields + objectFieldBase;
            new ← Allocate[size: size, odd: 0, pointers: FALSE,
               class: classOop, startInit: startInit];
            END;
         byte =>
            BEGIN
            size ← IF fields = LAST[CARDINAL] THEN
                  (fields / 2) + 1 + objectFieldBase
               ELSE (fields + 1)/2 + objectFieldBase;
            new ← Allocate[size: size, odd: fields MOD 2, pointers: FALSE,
               class: classOop, startInit: startInit];
            END;
         ENDCASE;
      RETURN[new];
      END;


   -- For debugging
   Fetch: PROC [loc: LONG UNSPECIFIED] RETURNS [UNSPECIFIED] = BEGIN
      RETURN [RealFetch[loc]]; END;

   Smash: PROC [loc: LONG UNSPECIFIED, value: UNSPECIFIED] = BEGIN
      RealSmash[loc, value]; END;


   SetRefCountOf: PROC [oop: Oop, count: UNSPECIFIED] = BEGIN
      otEntry: OTEntry ← OtEntry[Of[oop]];
      otEntry.referenceCount ← LOOPHOLE[count];
      SmashOTEntry[Of[oop], otEntry];
      END;


   END.
```

```
--
-- Definitions: The procedures implementing Smalltalk80 multi-processing
-- primitives.
--

--  8-Jul-85 15:47:30   Malcolm: USING, semaphoreIndex
-- last edited by Zdybel,    1-Aug-84 13:20:54

DIRECTORY
   ST80Defs USING [Oop];

ST80Proc: DEFINITIONS =
   BEGIN OPEN ST80Defs;

   --
   -- Fields of Process-related classes made public here for the
   -- sake of InitializeInterpreter
   --

   activeProcessIndex: CARDINAL = 1;
   suspendedContextIndex: CARDINAL = 1;


   --
   -- Public for the sake of space layout
   --

   semaphoreTableSize: CARDINAL = 256;
   semaphoreTable: LONG POINTER TO ARRAY [0..semaphoreTableSize) OF Oop;
   semaphoreIndex: READONLY CARDINAL [0 .. semaphoreTableSize);


   --
   -- Initialization
   --

   InitializeSemaphoreTable: PROCEDURE;

   InitializeClock: PROCEDURE;


   --
   -- Process-related primitives
   --

   PrimitiveSignal: PROCEDURE;

   PrimitiveWait: PROCEDURE;

   PrimitiveResume: PROCEDURE;

   PrimitiveSuspend: PROCEDURE;

   PrimitiveFlushCache: PROCEDURE;


   --
   -- Used by the Interpreter for process switching
   --

   CheckProcessSwitch: PROCEDURE;


   --
   -- Used by the Interpreter for signalling semaphores
   --

   AsynchronousSignal: PROCEDURE [semaphore: Oop];

   ReSignal: PROCEDURE;

   SynchronousSignal: PROCEDURE [semaphore: Oop];


   --
   -- Needed for snapshot
```

```
--

ActiveProcess: PROCEDURE RETURNS [Oop];


--
-- Clock primitives
--

PrimitiveTickWordsInto: PROCEDURE;

PrimitiveTimeWordsInto: PROCEDURE;

PrimitiveSignalAtTick: PROCEDURE;

--
-- Clock utility
--

KillTimer: PROCEDURE;

END.
```

```
--
-- Package: Process and Clock Primitives.
--
-- last edited by malcolm,
-- 20-Dec-85 15:33:36   recode without Sleep, Postpone
-- 20-Dec-85 12:28:13   AddLast, RealReplace
-- 19-Dec-85 15:58:30   several INLINEs
-- 19-Dec-85 12:29:44   speedups: RemoveFirst and callers
-- 18-Dec-85 17:00:55   some speedups, incl ListCheck
-- 17-Dec-85 15:24:59   INLINE SchedulerPointer, InlineActiveProcess
--  8-Oct-85 15:24:04   use RefD, RefI
--  7-Oct-85 16:53:32   no warnings
--  6-Sep-85 18:52:43   "G
-- 29-Aug-85 17:55:52   faster time prims
-- 29-Aug-85 14:51:50   common.active, leaf
-- 29-Jul-85 15:48:00   Base[Of
-- 22-Jul-85 15:37:20   PrimitiveFail no arg
--  8-Jul-85 18:10:24   USING for all
-- 21-Jun-85 16:18:06   no return from Store
--  5-Jun-85 11:11:40   import Rum
--  4-Jun-85 18:46:14   newProcessWaiting, newProcess in Rum.common
-- 21-May-85 14:46:08   IsWords in Mem
-- 20-May-85 15:50:43   ST80Stretch
-- 16-May-85 17:47:11   tidy up M..sec..Clock, Timer
-- 18-Apr-85 17:37:55   Stretch compatible IsWords
--  5-Apr-85 16:15:43   usual ListIsFullOfCorrectness
--  4-Apr-85 15:14:10   begin Stretch, no isSmallInteger

-- 27-Mar-85 17:17:01   ListIsFullOfCorrectness = TRUE
-- 12-Mar-85 16:41:58   fail in PrimSiganlAtTick if timerSetting is small int
-- 11-Mar-85 18:01:02   ~WordsInto should stuff the ARGUMENT!
-- 11-Mar-85 17:33:42   look at pulses in MillisecondClock
-- 11-Mar-85 15:28:10   break up long Timer waits
-- 22-Feb-85 14:40:42   fix DeltaWord use, debug flags in ST80DebugFlags
-- last edited by Zdybel, 21-Feb-85 17:40:59

DIRECTORY
   Inline USING [BITROTATE, DBITSHIFT, HighHalf, LowHalf],
   Process USING [Abort, Detach, MsecToTicks, Pause, SetPriority, GetCurrent,
      Milliseconds, Ticks, InvalidProcess, priorityForeground,
      priorityNormal],
   System USING [PulsesToMicroseconds, GetClockPulses, SecondsSinceEpoch,
      GetGreenwichMeanTime, Pulses],
   ST80Debug USING [LogProcessSwitch],
   ST80DebugFlags,
   ST80Defs USING [Base, Bletch, classSemaphorePointer,
      DeltaWordFrom, FetchDeltaWord, Loc, nilPointer,
      objectFieldBase, Of, Oop, OopIsSmallInteger, Physical,
      PhysicalAddress, PMemFetch, PMemSmash, PrimitiveFail, RealFetch, RealSmash,
      schedulerAssociationPointer, SmallInteger, VirtualAddress],
   ST80Mem USING [associationValueIndex, IsWords,
      FetchByteLength, FetchClass, FetchInteger, FetchPointer,
      IntegerObjectOf, IntegerValueOf, objectSizeOffset,
      Pop, QFetchPointer, QSmashPointer, QStoreByte, RefD, RefI,
      SmashPointer, StackAccess, StackTop, TernaryPrimReturn, UnaryPrimReturn],
   ST80Proc USING [AsynchronousSignal, ReSignal, semaphoreIndex, suspendedContextIndex,
      activeProcessIndex],
   ST80Rum USING [common],
   ST80Send USING [InitializeMethodCache, NewActiveContext],
   ST80Stretch USING [];


ST80ProcImpl: PROGRAM
   IMPORTS Inline, Process, System,
      ST80Debug, ST80Defs, ST80Mem, ST80Proc, ST80Rum, ST80Send
   EXPORTS ST80Proc = BEGIN
   OPEN ST80DebugFlags, ST80Defs, ST80Proc, ST80Rum,
      DBug: ST80Debug, Mem: ST80Mem, Send: ST80Send, Stretch: ST80Stretch;

   --
   -- Process and clock related registers of the interpreter
   --

   currentTimerProcess: PROCESS ← NIL;

   -- In the DLion, the clock is a "pulses" clock, NOT a millisecond
```

```
    -- clock.  A pulse being on the order of microseconds, and the clock
    -- being only two words wide, the DLion's "millisecond" clock wraps
    -- approximately once every 34 hours.  To get the most out of this
    -- limited amount, offset the actual pulses clock by its value when
    -- starting up an image.

    initialPulses: System.Pulses ← NULL;
    clockModulus: LONG CARDINAL ← System.PulsesToMicroseconds[
       LOOPHOLE[4294967295/1000, System.Pulses]];


    --
    -- Structure of process-related classes defined here
    --

    -- Class ProcessScheduler

    processListsIndex: CARDINAL = 0;

    -- Class LinkedList

    firstLinkIndex: CARDINAL = 0;
    lastLinkIndex: CARDINAL = 1;

    -- Class Semaphore

    excessSignalsIndex: CARDINAL = 2;

    -- Class Link

    nextLinkIndex: CARDINAL = 0;

    -- Class Process

    priorityIndex: CARDINAL = 2;
    myListIndex: CARDINAL = 3;


    --
    -- Initialization
    --

    InitializeClock: PUBLIC PROCEDURE =
       BEGIN

       -- The DLion "millisecond clock" wraps so frequently it is
       -- best to reset it to zero every time we start a new image.

       initialPulses ← System.GetClockPulses[];
       END;


    --
    -- Process-related primitives:
    --

    -- Reference counting theory is that any argument of Resume has had
    -- its reference count temporarily inflated by one.  This is mainly
    -- done to make sure that a process does not go away between the time
    -- it is removed from some list and the time it actually becomes the
    -- active process.  In most of the motivated cases, the inflation is
    -- done by the routine that removes items from lists.  PrimitiveResume
    -- does it just to be like everybody else.  The temporary inflation
    -- is undone either by Resume or by CheckProcessSwitch.  Because there
    -- are other outstanding references to semaphores, shoving a semaphore
    -- into the semaphore buffer doesn't require similar futzing around
    -- with reference counts.

    PrimitiveSignal: PUBLIC PROCEDURE =
       BEGIN

       -- Used for signalling semaphores from within the (Mesa) process
       -- of the interpreter.

       SynchronousSignal[Mem.StackTop[]];
       END;
```

```
PrimitiveWait: PUBLIC PROCEDURE =
   BEGIN

   -- Associated with the wait message in Semaphore.  If the receiver
   -- has an excess signal count greater than 0, decrements the count.
   -- If the excess signal count is 0, suspends the active Process and
   -- adds it to the receiver's list of Processes.

   receiver: Oop = Mem.StackTop[];
   excessSignalsRefLoc: PhysicalAddress ← Loc[Of[receiver]]
      + excessSignalsIndex + objectFieldBase;
   excessSignals: Oop ← RealFetch[excessSignalsRefLoc];
   n: SmallInteger;

   IF ~OopIsSmallInteger[excessSignals] THEN PrimitiveFail;

   IF (n ← Mem.IntegerValueOf[excessSignals]) > 0
   THEN -- excessSignals remains small integer, so no refcnts
      RealSmash[excessSignalsRefLoc, Mem.IntegerObjectOf[n - 1]]
   ELSE BEGIN
      -- AddLastLinkToList[InlineActiveProcess[], receiver];
      AddLast[receiver, InlineActiveProcess[]];
      TransferTo[WakeHighestPriority[]]; -- SuspendActive;
      END;
   END;


PrimitiveResume: PUBLIC PROCEDURE =
   BEGIN
   receiver: Oop = Mem.StackTop[];

   -- Inflate the receiver's refcount because Resume expects this.

   Mem.RefI[receiver];
   Resume[receiver];
   END;


PrimitiveSuspend: PUBLIC PROCEDURE =
   BEGIN

   -- Suspends the receiver if it is the active Process.  If the
   -- receiver is not the active Process, the primitive fails.

   receiver: Oop = Mem.StackTop[];

   IF receiver # InlineActiveProcess[]
   THEN PrimitiveFail; -- [reason: "Suspend Received by Inactive Process"L];

   TransferTo[WakeHighestPriority[]]; -- SuspendActive;
   Mem.UnaryPrimReturn[nilPointer];
   END;


PrimitiveFlushCache: PUBLIC PROCEDURE =
   BEGIN
   Send.InitializeMethodCache;
   END;


--
-- Interface procedure used by the Interpreter for process switching
--

CheckProcessSwitch: PUBLIC PROCEDURE =
   BEGIN

   -- Called before each bytecode fetch (in the basic interpreter loop)
   -- to perform the actual process switch if one has been called for.
   -- Stores the active context pointer into the old Process, stores
   -- the new Process in the ProcessorScheduler's active process field,
   -- and loads the new active context out of that Process.  Important:
   -- newProcess has had its reference count artificially inflated to
   -- make sure it doesn't go away inopportunely.  Be sure to set things
```

```
      -- to rights when the coast is clear.

      oldActiveProcess: Oop ← NULL;

      IF semaphoreIndex # 0 THEN ReSignal[];

      IF common.newProcessWaiting
      THEN BEGIN
         common.newProcessWaiting ← FALSE;
         oldActiveProcess ← InlineActiveProcess[];

         Mem.SmashPointer[suspendedContextIndex, oldActiveProcess,
            common.active];
         Mem.SmashPointer[activeProcessIndex, SchedulerPointer[],
            common.newProcess];
         Send.NewActiveContext[
            Mem.FetchPointer[suspendedContextIndex, common.newProcess]];

         -- Uninflate the count on newProcess:

         Mem.RefD[common.newProcess];

         -- Inserted for debugging purposes

         IF debugFlg THEN DBug.LogProcessSwitch[common.newProcess];

         END;
      END;


   --
   -- Private process-related procedures
   --

   <<
   SuspendActive: PROCEDURE = INLINE
      BEGIN
      TransferTo[WakeHighestPriority[]];
      END;
      >>


   SynchronousSignal: PUBLIC PROCEDURE [semaphore: Oop] =
      BEGIN
      first: Oop = RemoveFirst[semaphore];

      IF first = nilPointer
      THEN BEGIN
         excessSignalsRefLoc: PhysicalAddress = Loc[Of[semaphore]]
            + excessSignalsIndex + objectFieldBase;
         excessSignals: Oop ← RealFetch[excessSignalsRefLoc];
         n: SmallInteger;

         -- excessSignals must always be SmallInteger, so no refcnts
         IF OopIsSmallInteger[excessSignals]
         AND (n ← Mem.IntegerValueOf[excessSignals]) < LAST[SmallInteger]
         THEN RealSmash[excessSignalsRefLoc, Mem.IntegerObjectOf[n + 1]]
         ELSE PrimitiveFail;
         END
      ELSE Resume[first]; -- note that RemoveFirst inflated first's refcnt
      END;


   Resume: PROCEDURE [process: Oop] = BEGIN

      -- This routine assumes that its argument has had its reference
      -- count artificially inflated by one.

      activeProcess: Oop = InlineActiveProcess[];
      activePriority: CARDINAL = Mem.FetchInteger[priorityIndex, activeProcess];
      newPriority: CARDINAL = Mem.FetchInteger[priorityIndex, process];
      processLists: Oop = Mem.FetchPointer[processListsIndex, SchedulerPointer[]];
      processList: Oop ← NULL;

      IF newPriority > activePriority
      THEN BEGIN
```

```
        -- We are unfairly interrupting the current active process.
        -- Instead of circulating it back to the end of its queue,
        -- push it onto the front.

        -- Postpone[activeProcess];
        processList ← Mem.FetchPointer[activePriority - 1, processLists];
        AddFirstLinkToList[activeProcess, processList];

        -- If newProcessWaiting was TRUE, we must readjust the
        -- reference count on activeProcess, which will have been
        -- inflated to make sure it didn't go away.  It's ok to do
        -- this now because we have just placed it on a queue.

        IF common.newProcessWaiting THEN Mem.RefD[activeProcess];
        TransferTo[process];
        END

     ELSE BEGIN
        -- Add a process onto the back of its quiescent process list.

        --Sleep[process];
        processList ← Mem.FetchPointer[newPriority - 1, processLists];
        AddLast[processList, process];

        Mem.RefD[process]; -- undo inflation
        END;
     END;


  TransferTo: PROCEDURE [process: Oop] = INLINE BEGIN

     -- ANY process being transferred to will have had its reference
     -- count artificially inflated by one.  Note that if the current
     -- activeContext is also the current leafContext, we must make
     -- sure that it does not get recycled.

     common.newProcessWaiting ← TRUE;
     common.newProcess ← process;
     IF common.leafContext = common.active THEN
        BEGIN

        -- Make sure to readjust the reference count of the leafContext,
        -- which will have been inflated in order to keep it around.

        Mem.RefD[common.leafContext];
        common.leafContext ← nilPointer;
        END;
     END;


  WakeHighestPriority: PROCEDURE RETURNS [process: Oop] = BEGIN

     -- Remove a link from the quiescent process list.  The link's
     -- count will have been artificially inflated by the list manip-
     -- ulation routine.

     processLists: Oop =
        Mem.FetchPointer[processListsIndex, SchedulerPointer[]];
     processListsLoc: PhysicalAddress = Loc[Of[processLists]];
     processListRefLoc: PhysicalAddress ← processListsLoc -- high priority first
        + RealFetch[processListsLoc + Mem.objectSizeOffset] - 1;
     processList: Oop ← NULL;

     WHILE (process ← RemoveFirst[processList ← RealFetch[processListRefLoc]])
        = nilPointer
     DO processListRefLoc ← processListRefLoc - 1 ENDLOOP;

     RETURN[process]
     END;

  <<
  Sleep: PROCEDURE [process: Oop] = BEGIN

     -- Add a process onto the back of its quiescent process list.
```

```
    priority: CARDINAL = Mem.FetchInteger[priorityIndex, process];
    processLists: Oop =
       Mem.FetchPointer[processListsIndex, SchedulerPointer[]];
    processList: Oop ← Mem.FetchPointer[priority - 1, processLists];

    -- AddLastLinkToList[process, processList];
    AddLast[processList, process];
    END;


Postpone: PROCEDURE [process: Oop] = BEGIN

    -- We are changing priority levels; add process back onto the
    -- front of its quiescent process list.

    priority: CARDINAL = Mem.FetchInteger[priorityIndex, process];
    processLists: Oop =
       Mem.FetchPointer[processListsIndex, SchedulerPointer[]];
    processList: Oop ← Mem.FetchPointer[priority - 1, processLists];

    AddFirstLinkToList[process, processList];
    END;
>>

--
-- Low Level Support for the Above
--

InlineActiveProcess: PROCEDURE RETURNS [Oop] = INLINE
   BEGIN
   RETURN[IF common.newProcessWaiting
      THEN common.newProcess
      ELSE Mem.FetchPointer[activeProcessIndex, SchedulerPointer[]]];
   END;

ActiveProcess: PUBLIC PROCEDURE RETURNS [Oop] =
   BEGIN
   RETURN[InlineActiveProcess[]];
   END;


SchedulerPointer: PROCEDURE RETURNS [Oop] = INLINE
   BEGIN
   RETURN[Mem.FetchPointer[Mem.associationValueIndex,
      schedulerAssociationPointer]];
   END;


--
-- LinkedList-related procedures
--

<<
RemoveFirstLinkOfList: PROCEDURE [listBase: VirtualAddress]
   RETURNS [link: Oop] = BEGIN

   -- Guaranteed to leave the thing removed with a (hopefully
   -- temporarily) exalted reference count.  This routine is
   -- called only when the list is known not to be empty.

   firstLink: Oop = Mem.QFetchPointer[firstLinkIndex, listBase];
   lastLink: Oop = Mem.QFetchPointer[lastLinkIndex, listBase];
   nextLink: Oop ← NULL;

   -- There is a shortfall in the current Virtual machine design
   -- in that there is no way to guard against the eventuality of
   -- encountering one of these lists while it is in the process
   -- of being modified from the Smalltalk side.  No way to recover
   -- once the condition is detected, so break before the object
   -- memory is smashed.

   LinksCheck[firstLink, lastLink];

   -- Make sure the reference count of the link removed will be
   -- one too many.
```

```
      Mem.RefI[firstLink];

      IF lastLink = firstLink
      THEN BEGIN
         Mem.QSmashPointer[firstLinkIndex, listBase, nilPointer];
         Mem.QSmashPointer[lastLinkIndex, listBase, nilPointer];
         END
      ELSE BEGIN
         nextLink ← Mem.FetchPointer[nextLinkIndex, firstLink];
         Mem.QSmashPointer[firstLinkIndex, listBase, nextLink];
         END;

      Mem.SmashPointer[nextLinkIndex, firstLink, nilPointer];
      RETURN[firstLink];
      END;
      >>

   RemoveFirst: PROCEDURE [list: Oop] RETURNS [first: Oop] = INLINE BEGIN
   -- leaves first with refcnt inflated; also allows for empty list

      firstRefLoc: PhysicalAddress = Loc[Of[list]]
         + objectFieldBase + firstLinkIndex;

      first ← RealFetch[firstRefLoc];

      IF first = nilPointer
      THEN RETURN
      ELSE BEGIN
         lastRefLoc: PhysicalAddress = firstRefLoc
            + lastLinkIndex - firstLinkIndex;
         last: Oop = RealFetch[lastRefLoc];

         LinksCheck[first, last];  -- make sure list is consistent

         -- list refs first, so don't refD (hence first ends up inflated)
         -- next merely moves from a ref by first to a ref by list, refcnt same

         IF first = last  -- so also next = nilPointer
         THEN BEGIN
            RealSmash[firstRefLoc, nilPointer];
            RealSmash[lastRefLoc, nilPointer];
            Mem.RefD[last];  -- mustn't overinflate
            END
         ELSE BEGIN
            nextRefLoc: PhysicalAddress = Loc[Of[first]]
               + objectFieldBase + nextLinkIndex;
            next: Oop = RealFetch[nextRefLoc];

            RealSmash[firstRefLoc, next];
            RealSmash[nextRefLoc, nilPointer];
            END;
         END;
      END;


   AddFirstLinkToList: PROCEDURE [link: Oop, list: Oop] =
      BEGIN
      listBase: VirtualAddress = Base[Of[list]];
      linkBase: VirtualAddress = Base[Of[link]];

      -- Check to see if we interrupted a method that was changing
      -- the process list.

      ListCheck[list];

      IF IsEmptyList[listBase]
      THEN Mem.QSmashPointer[lastLinkIndex, listBase, link];

      Mem.QSmashPointer[nextLinkIndex, linkBase,
         Mem.QFetchPointer[firstLinkIndex, listBase]];
      Mem.QSmashPointer[firstLinkIndex, listBase, link];
      Mem.QSmashPointer[myListIndex, linkBase, list];
      END;

   <<
   AddLastLinkToList: PROCEDURE [link: Oop, list: Oop] =
```

```
        BEGIN
        listBase: VirtualAddress = Base[Of[list]];
        linkBase: VirtualAddress = Base[Of[link]];

        -- Check to see if we interrupted a method that was changing
        -- the process list.

        ListCheck[list];

        IF IsEmptyList[listBase]
        THEN Mem.QSmashPointer[firstLinkIndex, listBase, link]
        ELSE Mem.SmashPointer[nextLinkIndex,
              Mem.QFetchPointer[lastLinkIndex, listBase], link];

        Mem.QSmashPointer[lastLinkIndex, listBase, link];
        Mem.QSmashPointer[nextLinkIndex, linkBase, nilPointer];
        Mem.QSmashPointer[myListIndex, linkBase, list];
        END;
    >>

    AddLast: PROCEDURE [list, link: Oop] = BEGIN

        firstRefLoc: PhysicalAddress = Loc[Of[list]]
          + objectFieldBase + firstLinkIndex;
        lastRefLoc: PhysicalAddress = firstRefLoc + lastLinkIndex - firstLinkIndex;
        first: Oop = RealFetch[firstRefLoc];
        last: Oop = RealFetch[lastRefLoc];
        linkNextRefLoc: PhysicalAddress = Loc[Of[link]]
          + objectFieldBase + nextLinkIndex;

        LinksCheck[first, last];

        IF first = nilPointer
        THEN RealSmash[firstRefLoc, link]
        ELSE RealSmash[Loc[Of[last]] + objectFieldBase + nextLinkIndex, link];

        Mem.RefI[link];
        RealReplace[lastRefLoc, link];
        RealReplace[linkNextRefLoc, nilPointer];
        RealReplace[linkNextRefLoc + myListIndex - nextLinkIndex, list];
        END;

    IsEmptyList: PROCEDURE [listBase: VirtualAddress] RETURNS [BOOLEAN] =
        INLINE BEGIN
        RETURN[Mem.QFetchPointer[firstLinkIndex, listBase] = nilPointer];
        END;

    <<
    IsEmptyListLoc: PROCEDURE [listLoc: PhysicalAddress] RETURNS [BOOLEAN] =
        INLINE BEGIN
        RETURN[RealFetch[listLoc + firstLinkIndex + objectFieldBase] = nilPointer];
        END;
    >>

    <<
    ListIsFullOfCorrectness: PROCEDURE [listBase: VirtualAddress]
        RETURNS [BOOLEAN] =
        BEGIN

        -- Check to make absolutely sure that the argument is a valid
        -- list.

        nextLink: Oop ← Mem.QFetchPointer[firstLinkIndex, listBase];
        previous: Oop ← nilPointer;
        WHILE nextLink # nilPointer DO
          previous ← nextLink;
          nextLink ← Mem.FetchPointer[nextLinkIndex, previous];
          ENDLOOP;
        RETURN[previous = Mem.QFetchPointer[lastLinkIndex, listBase]];
        END;
    >>

    ListCheck: PROCEDURE [list: Oop] = INLINE BEGIN
        loc: PhysicalAddress = Loc[Of[list]] + objectFieldBase;

        LinksCheck[RealFetch[loc + firstLinkIndex], RealFetch[loc + nextLinkIndex]];
```

```
    END;

LinksCheck: PROCEDURE [firstLink, lastLink: Oop] = INLINE
    BEGIN

    -- Check to make absolutely sure that we have a valid list.

    previous: Oop ← nilPointer;

    WHILE firstLink # nilPointer DO
        linkRefLoc: PhysicalAddress ← Loc[Of[firstLink]]
            + nextLinkIndex + objectFieldBase;

        previous ← firstLink;
        firstLink ← RealFetch[linkRefLoc];
        ENDLOOP;

    IF previous # lastLink THEN Bletch["Process list invalid"G];
    END;


--
-- Clock Primitives
--

PrimitiveTickWordsInto: PUBLIC PROCEDURE =
    BEGIN

    -- Stuff the current reading of the millisecond clock into the
    -- receiver, checking first to make sure that the receiver is
    -- roomy enough.

    -- The receiver here is Time class!! ARGUMENT is byteArray

    byteArray: Oop = Mem.StackTop[];
    byteArrayBase: VirtualAddress ← NULL;
    classBase: VirtualAddress ← NULL;
    length: CARDINAL ← NULL;
    time: LONG CARDINAL = MillisecondClock[];
    --currentWord: CARDINAL ← NULL;
    classBase ← Base[Of[Mem.FetchClass[byteArray]]];
    IF Mem.IsWords[classBase] THEN
        ERROR PrimitiveFail; -- [reason: "Bad receiver for PrimitiveTickWordsInto"L];
    length ← Mem.FetchByteLength[byteArray];
    IF length < 4 THEN
        ERROR PrimitiveFail; -- [reason: "Undersized receiver for PrimitiveTickWordsInto"L];
    byteArrayBase ← Base[Of[byteArray]];

    PMemSmash[byteArrayBase, Inline.BITROTATE[Inline.LowHalf[time], 8]];
    PMemSmash[byteArrayBase + 1, Inline.BITROTATE[Inline.HighHalf[time], 8]];
    <<
    currentWord ← Inline.LowHalf[time];
    Mem.QStoreByte[0, byteArrayBase, Inline.LowByte[currentWord]];
    Mem.QStoreByte[1, byteArrayBase, Inline.HighByte[currentWord]];
    currentWord ← Inline.HighHalf[time];
    Mem.QStoreByte[2, byteArrayBase, Inline.LowByte[currentWord]];
    Mem.QStoreByte[3, byteArrayBase, Inline.HighByte[currentWord]];
    >>

    FOR index: CARDINAL IN [4..length) DO
        Mem.QStoreByte[index, byteArrayBase, 0];
        ENDLOOP;
    [] ← Mem.Pop[]; -- return self by popping arg array
    END;


PrimitiveTimeWordsInto: PUBLIC PROCEDURE =
    BEGIN

    -- Stuff the current reading of the GMT clock into the receiver,
    -- checking first to make sure that the receiver is roomy enough.

    byteArray: Oop = Mem.StackTop[];
    byteArrayBase: VirtualAddress ← NULL;
    classBase: VirtualAddress ← NULL;
    length: CARDINAL ← NULL;
```

```
      time: LONG CARDINAL =
         System.SecondsSinceEpoch[gmt: System.GetGreenwichMeanTime[]] +
            2114294400;
      --currentWord: CARDINAL ← NULL;
      classBase ← Base[Of[Mem.FetchClass[byteArray]]];
      IF Mem.IsWords[classBase] THEN
         ERROR PrimitiveFail; -- [reason: "Bad receiver for PrimitiveTickWordsInto"L];
      length ← Mem.FetchByteLength[byteArray];
      IF length < 4 THEN
         ERROR PrimitiveFail; -- [reason: "Undersized receiver for PrimitiveTickWordsInto"L];
      byteArrayBase ← Base[Of[byteArray]];

      PMemSmash[byteArrayBase, Inline.BITROTATE[Inline.LowHalf[time], 8]];
      PMemSmash[byteArrayBase + 1, Inline.BITROTATE[Inline.HighHalf[time], 8]];
      <<
      currentWord ← Inline.LowHalf[time];
      Mem.QStoreByte[0, byteArrayBase, Inline.LowByte[currentWord]];
      Mem.QStoreByte[1, byteArrayBase, Inline.HighByte[currentWord]];
      currentWord ← Inline.HighHalf[time];
      Mem.QStoreByte[2, byteArrayBase, Inline.LowByte[currentWord]];
      Mem.QStoreByte[3, byteArrayBase, Inline.HighByte[currentWord]];
      >>

      FOR index: CARDINAL IN [4..length) DO
         Mem.QStoreByte[index, byteArrayBase, 0];
         ENDLOOP;
      [] ← Mem.Pop[]; -- return self by popping arg
      END;


   PrimitiveSignalAtTick: PUBLIC PROCEDURE =
      BEGIN

      -- Arrange for an AsynchronousSignal to happen at some specified
      -- setting of the millisecond clock.  Be sure that timerSetting is
      -- at least four bytes long.  If semaphore is not a Semaphore,
      -- cancel the current timer setting.

      timerSetting: Oop = Mem.StackTop[];
      timerSettingBase: VirtualAddress = Base[Of[timerSetting]];
      semaphore: Oop = Mem.StackAccess[1];
      messageReceiver: Oop = Mem.StackAccess[2];
      timerValue: LONG CARDINAL ← 0;
      nowValue: LONG CARDINAL ← NULL;

      -- Check to see if we have merely been called to abort the current
      -- timer setting.

      IF Mem.FetchClass[semaphore] # classSemaphorePointer THEN
         BEGIN

         -- Suggest to the current timer process that it should abort.

         KillTimer[];
         Mem.TernaryPrimReturn[messageReceiver];
         RETURN[];
         END;

      -- Check out timersetting and make sure it has the right stuff.

      IF OopIsSmallInteger[timerSetting]
      OR FetchDeltaWord[
            DeltaWordFrom[timerSettingBase - objectFieldBase]].hasPointers
      OR Mem.FetchByteLength[timerSetting] < 4
      THEN ERROR PrimitiveFail; -- [reason: "Bad clock value"L];

      -- Get the current timer process to abort.

      KillTimer[];

      -- Extract the desired clock setting for the next signal.

      timerValue ← Inline.DBITSHIFT[
            LONG[Inline.BITROTATE[PMemFetch[timerSettingBase + 1], 8]],
            16]
       + LONG[Inline.BITROTATE[PMemFetch[timerSettingBase], 8]];
```

```
      <<
      FOR index: CARDINAL DECREASING IN [0..4) DO
         timerValue ← Inline.DBITSHIFT[timerValue, 8];
         timerValue ← Inline.DBITOR[timerValue,
            LONG[Mem.QFetchByte[index, timerSettingBase]]];
         ENDLOOP;
      >>

      -- Some values of the "millisecond clock" are UNOBTAINABLE:  the
      -- System pulses-clock will wrap before these values are achieved.
      -- Wrap the specified timer setting into the actual range of the
      -- clock.

      timerValue ← timerValue MOD clockModulus;

      -- Find out the current setting of the millisecond clock.

      nowValue ← MillisecondClock[];

      IF nowValue >= timerValue THEN

         -- The desired time for the signal has already occurred.
         -- Generate the signal immediately.

         SynchronousSignal[semaphore]
      ELSE
         BEGIN

         -- Launch a process that will signal at the desired value of
         -- the millisecond clock.  Make sure that it runs at foreground
         -- priority.

         Process.SetPriority[Process.priorityForeground];
         currentTimerProcess ← FORK Timer[signal: semaphore, at: timerValue];
         Process.Detach[currentTimerProcess];
         Process.SetPriority[Process.priorityNormal];
         END;
      Mem.TernaryPrimReturn[messageReceiver];
      END;


   --
   -- Time utilities
   --

   MillisecondClock: PROCEDURE RETURNS [LONG CARDINAL] =
      BEGIN
      pulses: LONG CARDINAL ← System.GetClockPulses[].pulses
         - initialPulses.pulses;
      returnValue: LONG CARDINAL ←
          System.PulsesToMicroseconds[LOOPHOLE[pulses/1000, System.Pulses]]
       + (System.PulsesToMicroseconds[LOOPHOLE[pulses MOD 1000, System.Pulses]]
          / 1000);
      RETURN[returnValue];
      END;


   Timer: PROCEDURE [signal: Oop, at: LONG CARDINAL] =
      BEGIN
      nowValue: LONG CARDINAL ← MillisecondClock[];
      WHILE nowValue < at DO
         msec: LONG CARDINAL ← at - nowValue;
         msecToWait: Process.Milliseconds ← LOOPHOLE[Inline.LowHalf[msec]];
         ticksToWait: Process.Ticks ← Process.MsecToTicks[msecToWait];
         Process.Pause[ticksToWait ! ABORTED => GO TO Aborted];

         -- Suspenders and belt.  If this global doesn't point to us,
         -- a signal is definitely not wanted from us ever.

         IF currentTimerProcess # Process.GetCurrent[] THEN RETURN[];

         nowValue ← MillisecondClock[];
         ENDLOOP;
      AsynchronousSignal[signal];
      currentTimerProcess ← NIL;
      RETURN[];
```

```
      EXITS Aborted =>
         BEGIN
         currentTimerProcess ← NIL;
         RETURN[];
         END;
      END;


   KillTimer: PUBLIC PROCEDURE =
      BEGIN
      IF currentTimerProcess # NIL THEN
         Process.Abort[currentTimerProcess !
            Process.InvalidProcess => CONTINUE];
      END;

--
-- local inlines for speedup
--
   <<
   SmashPointer: PROCEDURE [fieldIndex: CARDINAL, oop: Oop, valuePointer: Oop]
    = INLINE {
      loc: PhysicalAddress = Loc[Of[oop]] + (objectFieldBase + fieldIndex);
      Mem.RefD[RealFetch[loc]]; --DecreaseReferences
      RealSmash[loc, valuePointer];
      Mem.RefI[valuePointer] }; --IncreaseReferences

   QSmashPointer: PROCEDURE
      [fieldIndex: CARDINAL, base: VirtualAddress, valuePointer: Oop]
    = INLINE {
      loc: PhysicalAddress = Physical[base + fieldIndex];
      Mem.RefD[RealFetch[loc]]; --DecreaseReferences--
      RealSmash[loc, valuePointer];
      Mem.RefI[valuePointer] }; --IncreaseReferences--
   >>


   RealReplace: PROCEDURE [loc: PhysicalAddress, value: Oop]
    = INLINE BEGIN
      Mem.RefD[RealFetch[loc]];
      RealSmash[loc, value];
      Mem.RefI[value];
      END;


END.
```

```
{ BBT.dfn

Definitions for bitblt in Rum, the Dandelion Smalltalk-80 microcoded virtual machine.

        by T Tokunaga, J Trow

         9-Feb-86 17:33:52

Copyright 1985, 1986 by Xerox Corporation.  All rights reserved. }



{ R, RH register }

        RegDef [sourceIndex,            R, 0];  RegDef [skew,                   RH, 0];
        RegDef [destAddrLow,            R, 1];  RegDef [destAddrHigh,           RH, 1];
        RegDef [sourceAddrLow,          R, 2];  RegDef [sourceAddrHigh,         RH, 2];

        RegDef [halftoneAddrLow,        R, 4];  RegDef [halftoneAddrHigh,       RH, 4];



{ U register }

        { ******** Mesa Stack ********** }
        RegDef [uDestBitMap,            U,  2];
        RegDef [uH,                     U,  3];
        RegDef [uI,                     U,  3];
        RegDef [uDestAddrLow,           U,  4];
        RegDef [uDestWidthA,            U,  4];
        RegDef [uDestAddrHigh,          U,  5];
        RegDef [uDestHeightA,           U,  5];
        RegDef [uDestDelta,             U,  6];
        RegDef [uMask1,                 U,  7];
        RegDef [uMask2,                 U,  8];
        RegDef [uSkewMask,              U,  9];
        RegDef [uCurrentDispBitMap,     U,  9];
        RegDef [uMisc,                  U, 0A];
        RegDef [uHalftoneAddrLow,       U, 0B];
        RegDef [uSourceAddrLow,         U, 0C];
        RegDef [uSaveHighAddr,          U, 0D];
        RegDef [uNWordsM1,              U, 0E];

        {********** U block 1 ************ }
        RegDef [uSaveIPL,               U, 14]; { save smalltalk instruction pointer }
        RegDef [uSaveIPH,               U, 17];
        RegDef [uSaveStackL,            U, 19]; { save smalltalk stack pointer }
        RegDef [uSaveStackH,            U, 1A];
        RegDef [uSaveHomeL,             U, 1B]; { save smalltalk home context pointer }
        RegDef [uSaveHomeH,             U, 1D];
        RegDef [uArgument,              U, 1E]; { combinatin Rule }

        {********** U block 2 ************ }
        RegDef [uSourceDelta,           U, 24];

        {********** U block 3 ************ }
        RegDef [uClipHeight,            U, 31];
        RegDef [uDestMap,               U, 31];
        RegDef [uClipWidth,             U, 34];
        RegDef [uBBTTemp,               U, 34];
        RegDef [uStartBits,             U, 35];
        RegDef [uClipY,                 U, 35];
        RegDef [uW,                     U, 37];
        RegDef [uDX,                    U, 38];
        RegDef [uBooleans,              U, 3C];

        {********** U block 4 ************ }
        RegDef [uCombinationRule,       U, 42];
        RegDef [uDestForm,              U, 43];
        RegDef [uSourceForm,            U, 44];
        RegDef [uSX,                    U, 47];
        RegDef [uHalftoneForm,          U, 48];
        RegDef [uDestX,                 U, 4C];
        RegDef [uDestY,                 U, 4D];
        RegDef [uSkewWord,              U, 4E];
        RegDef [uSourceX,               U, 4F];
        RegDef [uPrevWord,              U, 4F];

        {********** U block 5 ************ }
        RegDef [uDY,                    U, 56];
        RegDef [uVDir,                  U, 58];
        RegDef [uHDir,                  U, 5C];
        RegDef [uSY,                    U, 5D];
        RegDef [uHalftoneWord,          U, 5D];

        {********** U block 6 ************ }
        RegDef [uSourceY,               U, 63];
        RegDef [uMergeMask,             U, 63];
        RegDef [uDestWidth,             U, 6A];
        RegDef [uWord,                  U, 6A];
        RegDef [uDestHeight,            U, 6B];
        RegDef [uNWords,                U, 6B];
```

```
        RegDef [uClipX,                 U, 6D];
        RegDef [uHM1,                   U, 6F];



{************************************************************
 **********                  constant                **********
 ************************************************************ }

        Set [CSBforCursorHigh,  2];
        Set [CSBforCursorLow,   0F0];

        Set [destFormIndex,     firstFieldOfObject];
        Set [diffDFAndHF,       2];

        Set [diffCurrentBitMapAndCursor, Sub [cursorBitmapOopOffset, displayBitmapOopOffset]];
        Set [halftoneFormIndex, Add [firstFieldOfObject, 2]];

        Set [difFieldAndSize, Sub [firstFieldOfObject, sizeFieldOffset]];
        Set [firstFieldOfObjectM1, Sub [firstFieldOfObject, 1]];


{ getSTFormMapBase : L2 }
        Set [getDestBitsAfterMInt,      0];
        Set [getDestBits,               1];
        Set [getSourceBits,             2];

        Set [formMapBaseIndex, firstFieldOfObject];

        Set [formMapBitsNoIndex, Add [1, firstFieldOfObject]];

        Set [formBitsIndex, firstFieldOfObject];
        Set [formWidthIndex, Add [1, formBitsIndex]];
        Set [formHeightIndex, Add [1, formWidthIndex]];;


{ checkSmallInt: L1}
        Set [checkCombinationRule,      0];
        Set [checkDestX,                1];
        Set [checkDestY,                2];
        Set [checkDestWidth,            3];
        Set [checkDestHeight,           4];
        Set [checkSourceX,              5];
        Set [checkSourceY,              6];
        Set [checkClipX,                7];
        Set [checkClipY,                8];
        Set [checkClipWidth,            9];
        Set [checkClipHeight,           0A];
        Set [checkSourceFormHeight,     0B];
        Set [checkSourceFormWidth,      0C];
        Set [cursorWidth,               0D];
        Set [getWidth,                  0E];
        Set [cursorWidth1,              0F];

{ checkSmallInt2: L1}
        Set [checkDestFormWidth,        0];
        Set [checkDestFormHeight,       1];
        Set [getWidth,                  2];
        Set [getHeight,                 3];


{checkWidthAndHeight: L3 }
        Set [checkClip,                 0];
        Set [checkDest,                 1];
        Set [checkSource,               2];


{makeRightMasks}
        Set [getMask1Again,             0];
        Set [getMask2Again,             1];
        Set [getMask1,                  2];
        Set [getMask2,                  3];
        Set [computeSkewMasks,          4];


{bbtMultiply: L3}
        Set [getSourceMul,              0];
        Set [getDestMul,                1];
        Set [widthAndHeight,            2];


{getSeveralMasks  ---  get skew, mask1, mask2}
        Set [getMasks,                  0];
        Set [getMasksAgain,             1];


{checkFormNil}
        Set [checkSourceFormAgain,      0];
        Set [checkHalftoneFormAgain,    1];


{bitBBTShift}
        Set [bitShiftSkewP,             0];
        Set [bitShiftSkewM,             1];
```

```
{checkMInt}
        Set [bbtCheckMInt,              0];


{updateCursor}
        Set [cursorAndDest,             0];
        Set [cursorAndSource,           1];


{restoreStatus}
        Set [bitbltFinished,            0];
        Set [bitbltNotFinished,         1];
        Set [primFail,                  7]; {must be odd}
        Set [noTransfer,                3];


{ checkRange: L2 }
        Set [checkXRange,               0]; {must be even}
        Set [checkYRange,               1]; {must be odd}


{calSkewWord: L3 }
        Set [skewWord,                  0];
        Set [skewWord1,                 1];
        Set [comb0D,                    2];


{write: L2 }
        Set [comb0D01,                  0];
        Set [comb0D02,                  1];
        Set [comb0D03,                  2];
        Set [comb0D04,                  3];
        Set [comb0D06,                  4];
        Set [comb0D07,                  5];
        Set [comb0D08,                  6];
        Set [comb0D09,                  7];
        Set [comb0D0B,                  8];
        Set [comb0D0C,                  9];
        Set [comb0D0D,                 0A];
        Set [comb0D0E,                 0B];


{mergeSrcAndDest: L2 }
        Set [hP1SFormNil,               1];
        Set [hP1NoLast,                 0];
        Set [hP1NoLastSFNonNil,         2];
        Set [hM1SFormNil,               5];
        Set [hM1NoLast,                 4];
        Set [hM1SFormNonNil,            9];
        Set [hM1NoLastSFNonNil,         8];
        Set [hP1SFormNonNil,           0D];



{ Edit history:

22-Jan-86 15:41:19      Tokunaga.fx     add constant for checksmallInt1, checkWidthAndHeight }
```

```
{ BBT.mc

Bitblt primitive for Rum, the Dandelion Smalltalk-80 microcoded virtual machine.

        by T Tokunaga, M Sakakibara, J Trow

         9-Feb-86 16:16:57

Copyright 1985, 1986 by Xerox Corporation.  All rights reserved. }



        { <primitive: 96>       BitBit >> copyBits

        Perform the movement of bits from one Form to another described by the instance variables of the receiver.  Fail if any instance
        variable is not of the right type (Integer or Form) or if combinationRule is not between 0 and 15 inclusive.  Set the variables
        and try again (BitBlt>>copyBitsAgain).

                input:

                output:

                smash:   }

primitiveCopyBits:
        uSaveIPL  ipLow,                                                        c1;
        ipLow  ipHigh ,                                                         c2;
        uSaveIPH  ipLow,                                                        c3; { save Smalltalk Instruction Pointer }

        uSaveStackL  stackLow,                                                  c1;
        stackLow  stackHigh,                                                    c2;
        uSaveStackH  stackLow,                                                  c3; { save Smalltalk Stack Pointer }

        uSaveHomeL  homeLow,                                                    c1;
        homeLow  homeHigh,                                                      c2;
        uSaveHomeH  homeLow,                                                    c3; { save the Pointer to home context }

checkMesaStackP:
        Xbus  ErrnIBnStkp, XDisp,                                               c1; { X [12~15]  ~stackP}
        DISP4 [bitBltHowBigStack, 7],                                           c2; { Only for case of Mesa Int }
        stackLow  uSaveStackL, GOTO [normalEntry],                              c3, at [0F, 10, bitBltHowBigStack];




{ ------------------------------------
  --   After  Mesa Interrupt  -------
  ------------------------------------}


        {At this point, part of the Parameters of Smalltalk BitBlt are stored in following order }

{                               -------------------------
 Oop for bitblt Argument        |  uDestBitMap          |       U02
                                -------------------------
 Loop counter (Vertical)        |  uI                   |       U03
                                ------------- -----------
 Destination Address(low)       |  uDestAddrLow         |       U04
                                -------------------------
 Destination Address(high)      |  *uDestAddrHigh       |       U05
                                -------------------------
 uSource, Dest Delta            |  *uDestDelta          |       U08
                                -------------------------
  uMaks1                        |  Mask1                |       U07
                                -------------------------
  uMask2                        |  Mask2                |       U08
                                -------------------------
  uSkewMask                     |  skewMask             |       U09
                                -------------------------
  uMisc                         |  *Misc                |       U0A
                                -------------------------
  uNWords                       |  NWords               |       U0B
                                -------------------------
  uHalftoneAddrLow              |  HalftoneAddrLow      |       U0C :
                                -------------------------
  uSourceAddrLow                |  SourceAddrLow        |       U0D :
                                -------------------------
  uSaveHighAddr                 |  *SaveHighAddr        |       U0E
                                -------------------------
************uDestAddrHigh
    00 - 03     : unused
    04 - 07     : combinationRule
    08 - 15     : destination Address high
************ uMisc
    00 ~ 03     : DY's offset( LS Nibble)
    04 ~ 07     : skew
    08          : unused
    09          : vDir = -1?
    0A          : hDir = -1?
    0B          : unused
    0C          : skew = 0?
    0D          : halftoneFormNIL -- 1 -> halftoneForm = Nil, 0 -> halftoneForm nonNil
    0E          : sourceFormNIL  -- 1 -> sourceForm = Nil, 0 -> sourceForm nonNil
    0F          : preload -- 1 -> preload True(uPreload=all 1), 0 -> preload False(uPreload=0)
```

```
**************** uSaveHighAddr
   00-07        : High address of Source bitMap
   08-0F        : High address of Halftone bitmap

**************** uDestDelta
   00-07        : source Delta
   08-0F        : dest Delta

  Note :  it means that these parameters without * marking is used for calculating the parameter again used in copyLoop
}



{ Now, We calculate the parameter(mask1, Mask2, skewMask, skew, vDir, hDir, preload) used in copybit Loop, again. }

afterMesaInt:
        temp2Low  uMisc, XwdDisp,                                       c3, at [07, 10, bitBltHowBigStack];

afterMesaInt1:
        Q  1, DISP2 [checkDirMInt],                                     c1;


{vDir = hDir = 1}
        uVDir  Q,                                                       c2, at [0, 4, checkDirMInt];
        uHDir  Q, GOTO [checkSkew],                                     c3;


{vDir = 1, hDir = -1 }
        uHDir  temp2Low xor ~temp2Low,                                  c2, at [1, 4, checkDirMInt];
        uVDir  Q, GOTO [checkSkew],                                     c3;


{vDir = -1, hDir = 1}
        uHDir  Q,                                                       c2, at [2, 4, checkDirMInt];
        uVDir  temp2Low xor ~temp2Low, GOTO [checkSkew],                c3;


{vDir  -1, hDir = -1}
        uVDir  temp2Low xor ~temp2Low,                                  c2, at [3, 4, checkDirMInt];
        uHDir  temp2Low xor ~temp2Low,                                  c3;

checkSkew:
        temp3Low  temp2Low and 0F, YDisp,                               c1; {temp2Low = 00-03:DY, 04-07:skew, 09:hdir, 0A:vDir,
        0c-0F: booleans}
        uBooleans  temp3Low,
                BRANCH [skewNonZeroAfterMInt, skewZeroAfterMInt, 7],    c2;


skewNonZeroAfterMInt:
        sourceIndex  temp2Low LRot8,                                    c3;

        sourceIndex  sourceIndex  and 0F,                               c1;
        skew  sourceIndex LRot0,                                        c2; {restore skew}
        GOTO [checkForm],                                               c3;


skewZeroAfterMInt:
        skew  0,                                                        c3; {restore skew}

checkForm:
        temp3Low  uDestDelta, XLDisp,                                   c1;
        destAddrLow  temp3Low and 0FF, BRANCH [destDeltaP, destDeltaM, 1],  c2;
destDeltaM:
        destAddrLow  destAddrLow xor ~0FF,                              c3; { create minus value for destDelta}

        Noop,                                                           c1;
        Noop,                                                           c2;
destDeltaP:
        uDestDelta  destAddrLow,                                        c3; {DestDelta}

        Xbus  uBooleans, XDisp,                                         c1;
        sourceIndex  uSaveHighAddr, DISP4 [checkFormNil, 9],            c2;


{both nonNil}
bothNonNil:
        temp1High  sourceIndex LRot0,                                   c3, at [9, 10, checkFormNil]; {highaddress of halftone}

        temp1Low  uHalftoneAddrLow,                                     c1; {low address of halftone form}
        temp2Low  temp2Low LRot4,                                       c2;
        uDY  temp2Low,                                                  c3; {save dy's offset}

restoreSAddr:
        otLow  sourceIndex LRot8,                                       c1;
        sourceAddrHigh  otLow LRot0,                                    c2; {highaddress of source}
        sourceAddrLow  uSourceAddrLow,                                  c3; {low address of source form }

restoreSDelta:
        temp3Low  temp3Low LRot8, XLDisp,                               c1; { check sourceDelta < 0}
        temp3Low  temp3Low and 0FF, BRANCH [sourceDeltaP, sourceDeltaM, 1],  c2;


sourceDeltaM:
        temp3Low  temp3Low xor ~0FF, GOTO [sourceDeltaP1],              c3;
```

```
sourceDeltaP:
        Noop,                                                                   c3;
sourceDeltaP1:
        uSourceDelta  temp3Low,                                                 c1; { sourceDelta}
        sourceAddrLow  sourceAddrLow + temp3Low, GOTO [bothNilAfterMInt],       c2;


{SForm=nil, HForm=nonNil}
sNilhNonNil:
        temp1High  sourceIndex LRot0,                                           c3, at [0B, 10, checkFormNil];

        temp1Low  uHalftoneAddrLow,                                             c1; { halftone form address}
        temp2Low  temp2Low LRot4,                                               c2;
        temp2Low  temp2Low and 0F,                                              c3;

        uDY  temp2Low, GOTO [bothNilAfterMInt1],                                c1; {dy's offset}


{SForm=nonNil, HForm=nil}
sNonNilhNil:
        GOTO [restoreSAddr],                                                    c3, at [0D, 10, checkFormNil];


{both nil}
bothNilAfterMInt1:
        Noop,                                                                   c2;
bothNilAfterMInt:
        otLow  uDestAddrHigh,                                                   c3, at [0F, 10, checkFormNil]; {otLow=destdelta}

        destAddrHigh  otLow LRot0,                                              c1;
        otLow  otLow LRot8,                                                     c2;
        uCombinationRule  otLow,                                                c3;

goVLoopMInt:
        otLow  uDestAddrLow                                                     c1; {otLow=destdelta}
        destAddrLow  destAddrLow + otLow, Xbus  uBooleans, XDisp,
                L2  bbtCheckMInt, GOTO [startVLoop1],                           c2; { restore destination address (16 bit)}




{
        ------------------------------------------------
        -----   First Entry for Smalltalk Bitblt   -----
        ------------------------------------------------
}

normalEntry:
        MAR  [stackHigh, stackLow + 0],                                         c1;
        L1  getParameter,                                                       c2;
        otLow  MD, CALL [otMap2Bank0],                                          c3; { get the real address of object }

normalEntry1:
        temp1Low  temp1Low + firstFieldOfObject,                                c1, at [getParameter, 10, otMap2Bank0-return]; { Point
        the destinationForm }
        uArgument  otLow,                                                       c2; { save the original Oop }
        Q  nilPointer,                                                          c3;

{ Now get the BitBlt Argument }

normalEntry2:
        MAR  [temp1High, temp1Low + 0],                                         c1;
        temp1Low  temp1Low + 1,                                                 c2; { point the SourceForm }
        destAddrLow  MD,                                                        c3; { get destination Form ****** }

normalEntry3:
        MAR  [temp1High, temp1Low + 0],                                         c1; {**********}
        temp1Low  temp1Low + 1,                                                 c2; { point the halftoneForm }
        sourceAddrLow  MD,                                                      c3; { get source Form ******** }

normalEntry4:
        MAR  [temp1High, temp1Low + 0],                                         c1; {**********}
        temp1Low  temp1Low + 1, L1  checkCombinationRule,                       c2; { point the combinationRule }
        otLow  MD,                                                              c3; { get halftoneForm and check three Oop of Form
        **********}

        [] otLow and nonPointerMask, ZeroBr,                                    c1;
        [] destAddrLow and nonPointerMask, ZeroBr,
             BRANCH [$, halftoneNonOop],                                        c2;
        [] sourceAddrLow and nonPointerMask, ZeroBr,
             BRANCH [$, destinationNonOop],                                     c3;

normalEntry5:
        MAR  [temp1High, temp1Low + 0], BRANCH [$, sourceNonOop],               c1;
        uDestForm  destAddrLow, CALL [checkSmallInt],                           c2; {save destination Form }

normalEntry55:
        []  temp2Low and ~0F, ZeroBr,                                           c1, at [checkCombinationRule, 10, checkSmallInt-return];
        BRANCH [combinationRuleTooBig, $],                                      c2;
        Noop,                                                                   c3;

normalEntry6:
        MAR  [temp1High, temp1Low + 0],  L1  checkDestX,                        c1;
        uCombinationRule  temp2Low, CALL [checkSmallInt],                       c2;
```

```
normalEntry7:
        MAR  [temp1High, temp1Low + 0], L1  checkDestY,                    c1, at [checkDestX, 10, checkSmallInt-return];
        uDestX  temp2Low, CALL [checkSmallInt],                            c2;

normalEntry8:
        MAR  [temp1High, temp1Low + 0], L1  checkDestWidth,                c1, at [checkDestY, 10, checkSmallInt-return];
        uDestY  temp2Low, CALL [checkSmallInt],                            c2;

normalEntry9:
        MAR  [temp1High, temp1Low + 0], L1  checkDestHeight,               c1, at [checkDestWidth, 10, checkSmallInt-return];
        uDestWidth  temp2Low, CALL [checkSmallInt],                        c2;

normalEntry10:
        MAR  [temp1High, temp1Low + 0], L1  checkSourceX,                  c1, at [checkDestHeight, 10, checkSmallInt-return];
        uDestHeight  temp2Low, CALL [checkSmallInt],                       c2;

normalEntry11:
        MAR  [temp1High, temp1Low + 0], L1  checkSourceY,                  c1, at [checkSourceX, 10, checkSmallInt-return];
        uSourceX  temp2Low, CALL [checkSmallInt],                          c2;

normalEntry12:
        MAR  [temp1High, temp1Low + 0], L1  checkClipX,                    c1, at [checkSourceY, 10, checkSmallInt-return];
        uSourceY  temp2Low, CALL [checkSmallInt],                          c2;

normalEntry13:
        MAR  [temp1High, temp1Low + 0], L1  checkClipY,                    c1, at [checkClipX, 10, checkSmallInt-return];
        uClipX   temp2Low, CALL [checkSmallInt],                           c2;

normalEntry14:
        MAR  [temp1High, temp1Low + 0], L1  checkClipWidth,                c1, at [checkClipY, 10, checkSmallInt-return];
        uClipY  temp2Low, CALL [checkSmallInt],                            c2;

normalEntry15:
        MAR  [temp1High, temp1Low + 0], L1  checkClipHeight,               c1, at [checkClipWidth, 10, checkSmallInt-return];
        uClipWidth  temp2Low, CALL [checkSmallInt],                        c2;

normalEntry16:
        []  sourceAddrLow - Q, ZeroBr,  {Q : nilPointer}                   c1, at [checkClipHeight, 10, checkSmallInt-return];
        []  otLow - Q, ZeroBr, BRANCH [sourceFormNonNilXX, sourceFormNilXX],  c2;


sourceFormNonNilXX:
        destAddrLow  0, BRANCH [halftoneFormNonNilXX1, halftoneFormNilXX1],   c3;


sourceFormNilXX:
        destAddrLow  2, BRANCH [halftoneFormNonNilXX1, halftoneFormNilXX1],   c3; { halftoneform = nil }


halftoneFormNonNilXX1:
        GOTO [saveOop],                                                    c1;


halftoneFormNilXX1:
        destAddrLow  destAddrLow or 4, GOTO [saveOop],                     c1; { sourceForm = nil }


saveOop:
        uBooleans  destAddrLow,                                            c2; {save the boolean, so far sourceformNil,
        halftoneformNil}
        Q  uClipHeight  temp2Low,                                          c3; {save clipHieght}

normalEntry17:
        uSourceForm  sourceAddrLow,                                        c1; {save sourceForm}
        uHalftoneForm  otLow,                                              c2; {save halftoneForm}


{       {****************** debug ***********************************}
        L1  1,                                                             c3;

checkDebug:
        L1Disp,                                                            c1;
        BRANCH [$, normalEntry18, 0E],                                     c2;
        L2  primFail, GOTO [primitiveFailBitBlt1],                         c3;
        {****************** debug ***********************************}              }


{If destWidth<=0 OR destHeight<=0 OR clipWidth<=0 OR clipHeight<=0 THEN immediately Return, Nothing is done }

normalEntry18:
        []  temp2Low - 1, NegBr,                                           c3; { check clipHeight <= 0 }

        temp2Low  uClipWidth, BRANCH [$, clipHeightLess0],                 c1;
        []  temp2Low - 1, NegBr,                                           c2; { check clipWidth <= 0 }
        temp3Low  uDestWidth, BRANCH [$, clipWidthLess0],                  c3;

normalEntry19:
        temp3Low  temp3Low - 1, NegBr, L3  checkClip,                      c1;

{ In Height, we check whether the area size(width*height) is greater than 64640?. If it is true, it cause a primitive fail }
{ Q : height, temp2Low : Width,  NOTE :::: temp2Low, Q, sourceIndex, temp1Low is smashed in the routine }

        otLow  uDestHeight,
               BRANCH [{CALL} checkWidthAndHeight, destWidthLess0],        c2;

        otLow  otLow - 1, NegBr,                                           c3, at [checkClip, 10, checkWidthAndHeight-return];
```

```
normalEntry20:
        BRANCH [$, destHeightLess0],                                    c1;
        otLow  uDestForm, L1  getDestFormData,                          c2;
        CALL [otMap2Bank0],                                             c3;

getDestination:
        MAR  temp1Low  [temp1High, temp1Low + firstFieldOfObject],      c1, at [getDestFormData, 10, otMap2Bank0-return];
getDestinationX:
        temp1Low  temp1Low + 1, BRANCH [noPCGetDest, yesPCGetDest, 1],  c2;


yesPCGetDest:
        temp1Low  temp1Low + OFF,                                       c3;

        MAR  [temp1High, temp1Low + 0], GOTO [getDestinationX],         c1;


noPCGetDest:
        sourceIndex  MD {destination bitmap},                           c3; { get Form BitMap Oop }

getDestination1:
        temp3Low  uRumRecordLow,                                        c1;
        temp3High  uRumRecordHigh,                                      c2;
        temp3Low  temp3Low + displayBitmapOopOffset,                    c3; { point the current display BitMap Oop }

getDestination2:
        MAR  [temp3High, temp3Low + 0],                                 c1;
        []  sourceIndex and nonPointerMask, Zero8r,                     c2; { save the destination BitMap Oop }
        uDestBitMap  sourceIndex, temp3Low  MD {display bitmap},
                BRANCH [$, destinationFormNonOop],                      c3; { get current display BitMap Oop }

getDestination3:
        []  temp3Low xor sourceIndex, Zero8r,                           c1; {dest bitmap = current display bitmap? ********}

        uCurrentDispBitMap  temp3Low,
                BRANCH [noDestAndCurrent, yesDestAndCurrent],           c2;


{ destAddrLow = uBooleans }
yesDestAndCurrent:
        temp3Low  3,                                                    c3;

yesDestAndCurrent1:
        temp3Low  temp3Low LRot8,                                       c1;
        temp3Low  temp3Low or 28,                                       c2; { 808'd }
        uDestHeightA  temp3Low,                                         c3;

yesDestAndCurrent2:
        temp3Low  4,                                                    c1;
        temp3Low  temp3Low LRot8,                                       c2; { 1024'd }
        uDestWidthA  temp3Low,                                          c3;

yesDestAndCurrent3:
        destAddrLow  destAddrLow + 80,                                  c1;
        GOTO [noDestAndCurrent31],                                      c2;


noDestAndCurrent:
        temp1Low  temp1Low + 1,  { Point the Form.Width }               c3;

noDestAndCurrent1:
        MAR  [temp1High, temp1Low + 0], L1  checkDestFormWidth,         c1;
        CALL [checkSmallInt2],                                          c2;

{temp1Low points the Height, since it is incremented by 1 in checkSmallInt2 routine }
noDestAndCurrent2:
        MAR  [temp1High, temp1Low + 0], L1  checkDestFormHeight,        c1, at [checkDestFormWidth, 10, checkSmallInt2-return];
        uDestWidthA  temp2Low, CALL [checkSmallInt2],                   c2;

noDestAndCurrent3:
        uDestHeightA  temp2Low, L3  checkDest,                          c1, at [checkDestFormHeight, 10, checkSmallInt2-return];
        Q  uDestWidthA, CALL [checkWidthAndHeight],                     c2;
noDestAndCurrent31:
        uBooleans  destAddrLow, GOTO [clipRange1],                      c3, at [checkDest, 10, checkWidthAndHeight-return];


destinationFormNonOop:
        L2  primFail, GOTO [primitiveFailBitBlt2],                      c1;




{-------------------------------------------------------------------------
 ----------                    ClipRange                      ----------
 -------------------------------------------------------------------------}

{First of all, we check the X-coordinate }
clipRange1:
        Noop,                                                           c1;
        sourceAddrLow  uSourceX,                                        c2;
        destAddrLow  uDestX,                                            c3;

clipRange:
        sourceIndex  uDestWidth,                                        c1;
        temp2Low  uClipX,  L2  checkXRange,                             c2;
```

```
        temp3Low  uClipWidth, CALL [bbtCheckRange],                    c3;

saveX:
        uSX  sourceAddrLow,                                           c2, at [checkXRange, 10, bbtCheckRange-return];
        uDX  destAddrLow,                                             c3;

        uW  sourceIndex, NegBr,                                       c1; { w < 0? }
checkRangeUpY:
        sourceAddrLow  uSourceY, BRANCH [widthGE0, widthLT0],         c2;


widthGE0:
        destAddrLow  uDestY,                                          c3;

        temp2Low  uClipY,                                             c1;
        sourceIndex  uDestHeight, L2  checkYRange,                    c2;
        temp3Low  uClipHeight, CALL [bbtCheckRange],                  c3;

saveY:
        uH  sourceIndex, NegBr, L1  getSFormWH,                       c2, at [checkYRange, 10, bbtCheckRange-return];
        uDY  destAddrLow, BRANCH [heightGE0, heightLT0],              c3;


heightGE0:
        Xbus  uBooleans, XDisp,                                       c1;
        uSY  sourceAddrLow, BRANCH [sFormNonNil0, sFormNil0, 0D],     c2;


sFormNonNil0:
        otLow  uSourceForm, CALL [otMap2Bank0],                       c3;

sFormNonNil01:
        temp1Low  temp1Low + formHeightIndex,                         c1, at [getSFormWH, 10, otMap2Bank0-return];
        sourceAddrLow  sourceAddrLow + sourceIndex,                   c2; { sy + h }
        sourceIndex  sourceIndex - sourceAddrLow,                     c3; { h  h - (sy + h)}

sFormNonNil02:
        MAR  [temp1High, temp1Low + 0], L1  checkSourceFormHeight,    c1;
        temp1Low  temp1Low - 2, CALL [checkSmallInt],                 c2; { point the form width, -2 since temp1Low incremented
        by 1 in checksmallInt  }

sFormNonNil03:
        []  temp2Low - sourceAddrLow, NegBr,                          c1, at [checkSourceFormHeight, 10, checkSmallInt-return];
        {sy + h > sourceForm height??}
        sourceIndex  sourceIndex +  temp2Low,
                BRANCH [sFormHeightGE, sFormHeightLT],                c2; { h  h + sourceForm Height }


sFormHeightGE:
        GOTO [checkWidth],                                            c3;


sFormHeightLT:
        uH  sourceIndex,                                              c3;

{------ next : check width   --------}
checkWidth:
        sourceAddrLow  uSX,                                           c1;
        sourceIndex  uW,                                              c2;
        sourceAddrLow  sourceAddrLow + sourceIndex,                   c3; { sx + w }

checkWidth1:
        MAR  [temp1High, temp1Low + 0], L1  checkSourceFormWidth,     c1;
        sourceIndex  sourceIndex - sourceAddrLow, CALL [checkSmallInt], c2;  { w  w - (sx + w )}

checkWidth2:
        []  temp2Low - sourceAddrLow, NegBr,                          c1, at [checkSourceFormWidth, 10, checkSmallInt-return];
        { sx + w > sourceForm width ??}
        sourceIndex  sourceIndex + temp2Low,
                        BRANCH [sFormWidthGE, sFormWidthLT],          c2;


sFormWidthGE:
        sourceIndex  uW, GOTO [sFormNil1],                            c3;


sFormWidthLT:
        uW  sourceIndex, GOTO [sFormNil1],                            c3;


sFormNil0:
        sourceIndex  uW,                                              c3;

{ ------ check the h and w, if h <= 0 OR w <=0 then return immediately ------ }
sFormNil1:
        sourceAddrLow  uH,                                            c1;
        sourceIndex  sourceIndex - 1, NegBr,                          c2; { check the width =< 0 ??}
        sourceAddrLow  sourceAddrLow - 1, NegBr, BRANCH [widthGT0, widthLE0],  c3; { check the height =< 0 ???}

widthGT0:
        uHM1  sourceAddrLow, BRANCH [heightGT0, heightLE0],           c1;


heightGT0:
        L1  getMasks, GOTO [computeMask],                             c2;
```

```
widthLEO:
        CANCELBR [$, OF],                                                       c1;
heightLEO:
        L2  noTransfer, GOTO [noTransfer3],                                     c2;  { restore the smalltalk state and execute next byte
        code }




{-------------------------------------------------------------------------
 -------------------- compute the several kind of Mask -------------------
 -------------------------------------------------------------------------}
computeMask:
        Q  1, CALL [getSeveralMasks],                                           c3; { Q for the intial value of  nWords  }

{
At return point :
mask1, mask2, skewMask, skew and startBits have already been calculated, and been stored in uMask1, uMask2, SkewMask, uSkew, and
uStartBits respectively
 }

getSeveralMasksRet:
        otLow  uW,                                                              c1;
getSeveralMasksRet1:
        destAddrLow  uStartBits,                                                c2;
        otLow  otLow - destAddrLow, NegBr,                                      c3; {w < startBits ?? : w - startBits }

        sourceIndex  uMask2, BRANCH [widthNotTooSmall, widthTooSmall],          c1;


widthTooSmall:
        sourceIndex  sourceIndex and uMask1,                                    c2; { mask1  mask1 bitAnd: mask2 }
        uMask1  sourceIndex,                                                    c3;

        uMask2  sourceIndex xor sourceIndex,                                    c1; { mask2  0 }
        uNWords  Q,                                                             c2; { uNWords  0{1}, because of 0 origin }
        otLow  Q, GOTO [checkOverlap],                                          c3;


widthNotTooSmall:
        otLow  otLow - 1, NegBr,                                                c2;
        otLow  otLow and ~0F,    BRANCH [noNegative, yesNegative],              c3;


yesNegative:
        otLow  otLow xor ~otLow, GOTO [noNegative1],                            c1; { make otLow -1}


noNegative:
        otLow  otLow LRot12,                                                    c1; { (w-startBits-1//16}
noNegative1:
        otLow  otLow + 2,                                                       c2; { (w - startBits - 1//16 + 2 }
        uNWords  otLow, GOTO [checkOverlap],                                    c3; { save nWords }




{-------------------------------------------------------------------------
 ----------                    CheckOverlap                         ----------
 -------------------------------------------------------------------------}
checkOverlap:
        Noop,                                                                   c1;
        otLow  otLow - Q,                                                       c2; {decrement}
        uNWordsM1  otLow,                                                       c3;

checkOverlap1:
        uStackF  otLow,                                                         c1;
        uVDir  Q,                                                               c2; { default v-direction = 1}
        uHDir  Q, L2  getDestBits,                                              c3; { default h-direction = 1 }

checkOverlap2:
        temp1Low  uSourceForm,                                                  c1;
        destAddrLow  uDY,                                                       c2;
        sourceAddrLow  uSY,                                                     c3;

checkOverlap3:
        []  temp1Low xor uDestForm, ZeroBr,                                     c1; { sourceForm == destFrom ??}

        []  destAddrLow - sourceAddrLow, NegBr,
                BRANCH [sameOopNo, sameOopYes],                                 c2; { dy >= sy  ?? }


sameOopYes:
        []  sourceAddrLow xor destAddrLow, ZeroBr, BRANCH [dyGE, dyLT],         c3; { dy = sy ??}


dyGE:
        sourceIndex  uHM1, BRANCH [dyGT, dyEQ],                                 c1; { sourceIndex  h - 1}


{sourceForm==destForm and dy>sy}
dyGT:
        uVDir  temp2Low xor ~temp2Low,                                          c2; { vDir  -1 }
        destAddrLow  destAddrLow + sourceIndex,                                 c3; { dy  dy + h - 1}
```

```
dyGT1:
        sourceAddrLow  sourceAddrLow + sourceIndex,                          c1; { sy  sy + h - 1}
        uDY  destAddrLow,                                                    c2;
        uSY  sourceAddrLow, GOTO [dyLT],                                     c3;


{sourceForm==destForm and dy=sy}
dyEQ:
        sourceAddrLow  uSX,                                                  c2;
        destAddrLow  uDX,                                                    c3;

dyEQ1:
        []  sourceAddrLow - destAddrLow, NegBr,                              c1; { dx > sx ??}
        uHDir  temp2Low xor ~temp2Low,   BRANCH [dxLE, dxGT],                c2; { hDir  -1 : So far default}


{ sourceForm==destForm and dy=sy and dx>sx }
dxGT:
        sourceIndex  uW,                                                     c3;

        sourceIndex  sourceIndex - 1,                                        c1;
        sourceAddrLow  sourceAddrLow + sourceIndex,                          c2; { sx  sx + w - 1}
        destAddrLow  destAddrLow + sourceIndex,                              c3; { dx  dx + w - 1}

dxGT1:
        otLow  uSkewMask,                                                    c1;
        uSkewMask  ~otLow,                                                   c2; { skewMask bitInvert }
        temp1Low  uMask1,                                                    c3;

dxGT2:
        temp2Low  uMask2,                                                    c1;
        uMask1  temp2Low,                                                    c2;
        uMask2  temp1Low,                                                    c3; {exchange mask1 and mask2}

dxGT3:
        uDX  destAddrLow,                                                    c1;
        uSX  sourceAddrLow,                                                  c2; { save sx }

sameOopNo:
        CANCELBR [dyLT, 1],                                                  c3;


dxLE:
        uHDir  Q, GOTO [dyLT],                                               c3;


dyLT:
        CANCELBR [$, 1],                                                     c1;




{-------------------------------------------------------------
 ------------------- calculate Offsets ----------------------
 -------------------------------------------------------------}

calculateOffset:
        otLow  uDestBitMap, CALL [getSTFormMapBase1],                        c2;

{ at return
  temp1High, temp1Low : map base
  temp2Low :  word no. per 1 horizontal line ( destRaster )
  otLow : Oop for bitmap
}
calculateOffset1:
        destAddrLow  temp1Low, L2  getDestMul,                               c2, at [getDestBits, 10, getSTFormMapBase-return];
        Q  uDY, CALL [bbtMultiply],                                          c3; { destIndex  dy*destRaster : Q  Q*temp2Low }

{ destAddrLow Points the actual destination Address. Now, we calculate the destDelta }
        destAddrLow  destAddrLow + Q,                                        c2, at [getDestMul, 10, bbtMultiply-return];
        sourceIndex  uDX,                                                    c3;

getDestAddr1:
        sourceIndex  sourceIndex and ~0F,                                    c1;
        sourceIndex  sourceIndex LRot12,                                     c2;
        destAddrLow  destAddrLow + sourceIndex,                              c3;  { destIndex   destRaster*dy +(dx//16)}

getDestAddr2:
        temp1Low  temp1High,                                                 c1;
        destAddrHigh  temp1Low LRot0,                                        c2;{ save the destination High address }
        Xbus  uVDir, XHDisp,                                                 c3;

checkDir:
        Xbus  uHDir, XHDisp, BRANCH [vP1, vM1, 2],                           c1;


vP1:
        sourceIndex  uNWords, BRANCH [vP1hP1, vP1hM1, 2],                    c2;{save the destination Map to check}


vP1hP1:
        temp2Low  temp2Low-sourceIndex, L2  getSourceBits,
                GOTO [saveDestDelta1],                                       c3; { destDelta  destRaster *1 - nWords * 1 }


vP1hM1:
```

```
        temp2Low  temp2Low + sourceIndex, L2  getSourceBits,
                GOTO [saveDestDelta1],                                  c3; { destDealta  destRaster*1 - nWords*(-1)}

vM1:
        sourceIndex  uNWords, BRANCH [vM1hP1, vM1hM1, 2],               c2;

vM1hP1:
        temp2Low  - temp2Low,                                           c3;

        temp2Low  temp2Low - sourceIndex {- 1}, GOTO [saveDestDelta],   c1; { destDelta  destRaster*(-1) - nWords*1}

vM1hM1:
        temp2Low  sourceIndex - temp2Low, GOTO [saveDestDelta1],        c3; {destDelta  destRaster*(-1) - nWords*(-1)}

saveDestDelta1:
        Noop,                                                           c1;
saveDestDelta:
        uDestDelta  temp2Low, L2  getSourceBits,                        c2;
{  Next  we check the sourceForm }

getSFormAddr:
        temp1Low  uBooleans, XDisp,                                     c3;

        otLow  uSourceForm, BRANCH [getSTFormMapBase, saveBooleans, 0D], c1; { dafault preload = False, so uBooleans.15 = 0}

{ at return
        temp1High, Low : source MapBase
        temp2Low :  Word no of 1 horizontal line ( sourceRaster )
        otLow : Oop for BitMap
}

getSFormAddr1:
        sourceAddrLow  temp1Low, L2  getSourceMul,                      c2, at [getSourceBits, 10, getSTFormMapBase-return];
        Q  uSY, CALL [bbtMultiply],                                     c3; { sourceIndex  sy*sourceRaster : Q  Q*temp2Low }

{ sourceAddrLow  points the actual source address }
        sourceAddrLow  sourceAddrLow + Q,                               c2, at [getSourceMul, 10, bbtMultiply-return];
        Q  temp1High,                                                   c3;

getSourceAddr:
        sourceAddrHigh  Q LRot0,                                        c1; { save high source address }
        sourceIndex  uSX,                                               c2;
        temp1Low  sourceIndex and ~0F,                                  c3;

        temp1Low  temp1Low LRot12,                                      c1;
        Q  skew, ZeroBr,                                                c2;
        sourceAddrLow  sourceAddrLow + temp1Low,
                BRANCH [skewNonZero3, skewZero3],                       c3;{ sourceIndex  sourceRaster*sy + (sx//16)}

skewNonZero3:
        sourceIndex  sourceIndex and 0F,                                c1; { sx bitAnd: 15 }
skewNonZero4:
        [] sourceIndex - Q, NegBr,                                      c2; { skew <= (sx bitAnd: 15)}
        temp1Low  uBooleans, BRANCH [skewLE1, skewGT1],                 c3;

skewLE1:
        temp1Low  temp1Low or 1, GOTO [checkhDir],                      c1; { preload = True }

skewGT1:
        GOTO [checkhDir],                                               c1; { preload = False }

skewZero3:
        temp1Low  uBooleans, GOTO [checkhDir],                          c1;

{ Now, temp2Low still has the sourceRaster }
checkhDir:
        Q  uHDir, XHDisp,                                               c2;
        [] temp1Low, YDisp, BRANCH [hP1, hM1, 2],                       c3;

hP1:
        sourceIndex  uNWords,
                BRANCH [hP1PreFalse, hP1PreTrue, 0E],                   c1;

hP1PreFalse:
        sourceIndex  sourceIndex, GOTO [checkVDir1],                    c2; { nWords + 0*1 : preload = False, hDir = 1 }

hP1PreTrue:
        sourceIndex  sourceIndex + 1, GOTO [checkVDir1],                c2; { nWords + 1*1 : preload = True, hDir = 1 }

hM1:
        sourceIndex  uNWords,
                BRANCH [hM1PreFalse, hM1PreTrue, 0E],                   c1;
```

```
{ ------------------------------
  ---- Now,  hDir < 0 ifTrue: [preload  preload == false]
  ----  so,  preloadFalse actually means preload = True, preloadTrue vice versa,
  ------------------------------- }
hM1PreFalse:
        sourceIndex  0 - sourceIndex - 1,                                   c2; {nWords + 1*(-1) : preload = False, hDir = -1 }
        temp1Low  temp1Low or 1 , GOTO [checkVDir2],                        c3; { make preload = True }


hM1PreTrue:
        sourceIndex  0 - sourceIndex ,                                      c2; {nWords + 0*(-1) : preload = Trued, hDir = -1 }
        temp1Low  temp1Low and 0E, GOTO [checkVDir2],                       c3; { make preload = False }


checkVDir1:
        Noop,                                                               c3;

checkVDir2:
        Xbus  uVDir, XHDisp,                                                c1;
        Q  - temp2Low, BRANCH [vDirP1, vDirM1, 2],                          c2; { Q   - sourceRaster }


vDirP1:
{ sourceRaster*1 - (nWords +(preload ifTrue: [1] ifFalse: [0])*hDir)}
        temp2Low  temp2Low - sourceIndex,                                   c3;

        uSourceDelta  temp2Low, GOTO [saveBooleans],                        c1;


vDirM1:
{sourceRaster*(-1) - (nWords +(preload ifTrue: [1] ifFalse: [0])*hDir)}
        Q  Q - sourceIndex,                                                 c3;

        uSourceDelta  Q,                                                    c1;
saveBooleans:
        uBooleans  temp1Low, YDisp, L1  getHalftoneBits,                    c2;
{ now Get the halftone Form address }
        otLow  uHalftoneForm, BRANCH [{CALL} otMap2Bank0, hFormNil21, 0B],  c3;


getHalftone:
        MAR  temp1Low  [temp1High, temp1Low + formMapBaseIndex],            c1, at [getHalftoneBits, 10, otMap2Bank0-return];
getHalftoneAddr:
        BRANCH [noPCGetHalftone, yesPCGetHalftone, 1],                      c2;


yesPCGetHalftone:
        temp1Low  temp1Low + OFF + 1,                                       c3;

        MAR  [temp1High, temp1Low + 0], GOTO [getHalftoneAddr],             c1;


noPCGetHalftone:
        otLow  MD, L1  getHalftoneBits1,                                    c3;

        [] otLow and nonPointerMask, ZeroBr,                                c1;
        BRANCH [$, hBitMapNoOop],                                           c2;
        CALL [otMap2Bank0],                                                 c3;


hFormNil21:
        GOTO [startVLoop],                                                  c1;


hFormNil2:
        temp1Low  temp1Low + firstFieldOfObject,                            c1, at [getHalftoneBits1, 10, otMap2Bank0-return];
```

```
{-------------------------------------------------------------------------------------------------------------
 -- Register Usage:     R0      -->     (sourceIndex)                                                    --
 --                     RH1, R1         -->     Address of destinationBits      (destAddrHigh, sourceAddrLow)          --
 --                                                                                                      --
 --                     RH2, R2         -->     Address of sourceBits           (sourceAddrHigh, sourceAddrLow)        --
 --                     RH3     -->     **** High Address of OT ******                                   --
 --                     R3      -->     DestinationIndex                (destIndex)                      --
 --                     RH5, R5 -->     Address of halftoneBits         (HalftoneAddrHigh, HalftoneAddrLow)    --
 --                                                                                                      --
 ------------------------------------------------------------------------------------------------------------- }
{
 ------------------------------------------------------------------------------
 ------------------------- Start of Vertical Loop -----------------------------
 ------------------------------------------------------------------------------
 }
startVLoop:
        Xbus  uBooleans, XDisp, L2  bbtCheckMInt,                           c2;
startVLoop1:
        L1  0,  Q  uHDir, DISP4 [vLoop],                                    c3;


{-------skew#0, HForm=SForm=~nil, Preload=FALSE----------}
startVLoop00:
        sourceIndex  uVDir, CALL [getHalftoneWord],                         c1, at [00, 10, vLoop];
```

```
startVLoop002:
        temp3Low  uNWordsM1, L2  skewWord,                                  c2, at [00, 10, getHalftoneWord-return];
startVLoop001:
        otLow  uMask1,                                                      c3;

        sourceIndex  sourceIndex xor sourceIndex,                           c1;
        Noop,                                                               c2;
        CALL [calSkewWord],                                                 c3;

        MAR  [destAddrHigh, destAddrLow + 0],                               c1, at [skewWord, 10, calSkewWord-return];
        temp2Low  temp2Low and uHalftoneWord,                               c2;
        sourceIndex  MD,                                                    c3;

        L2  bbtCheckMInt,                                                   c1;
        Xbus  uCombinationRule, XDisp,                                      c2;
        Q  uHDir, DISP4 [comb0D],                                           c3;


{-------skew<>0, HForm=~nil, SForm=~nil, Preload=TRUE--------------}
startVLoop01:
        sourceIndex  uVDir, CALL [getHalftoneWord],                         c1, at [01, 10, vLoop];

startVLoop011:
        temp3Low  uNWordsM1,                                                c2, at [01, 10, getHalftoneWord-return];
        otLow  uMask1,                                                      c3;

        MAR  [sourceAddrHigh, sourceAddrLow + 0],                           c1;
        sourceAddrLow  sourceAddrLow + Q, L2  skewWord,                     c2;
        sourceIndex  MD, CALL [calSkewWord],                                c3;


{-------skew<>0, HForm=~nil, SForm=nil, Preload=FALSE **--------------}
startVLoop02:
        sourceIndex  uVDir, CALL [getHalftoneWord],                         c1, at [02, 10, vLoop];

        temp3Low  uNWordsM1, GOTO [startVLoop0A1],                          c2, at [02, 10, getHalftoneWord-return];


{-------skew<>0, HForm=~nil, SForm=nil, Preload=TRUE----------   INVALID  --------------}
startVLoop03:
        GOTO [iteration],                                                   c1, at [03, 10, vLoop];


{-------skew<>0, HForm=nil, SForm=~nil, Preload=FALSE--------------}
startVLoop04:
        temp2Low  uHalftoneWord  temp2Low xor ~temp2Low,                    c1, at [04, 10, vLoop];
        temp3Low  uNWordsM1, L2  skewWord, GOTO [startVLoop001],            c2;


{-------skew<>0, HForm=nil, SForm=~nil, Preload=TRUE--------------}
startVLoop05:
        uHalftoneWord  temp2Low xor ~temp2Low, GOTO [startVLoop011],        c1, at [05, 10, vLoop];


{-------skew<>0, HForm=nil, SForm=nil, Preload=FALSE **--------------}
startVLoop06:
        temp2Low  uHalftoneWord  temp2Low xor ~temp2Low,
                GOTO [startVLoop0E1],                                       c1, at [06, 10, vLoop];


{-------skew<>0, HForm=nil, SForm=nil, Preload=TRUE----------    INVALID  --------------}
startVLoop07:
        GOTO [iteration],                                                   c1, at [07, 10, vLoop];


{-------skew=0, HForm=~nil, SForm=~nil, Preload=FALSE--------------}
startVLoop08:
        sourceIndex  uVDir, CALL [getHalftoneWord],                         c1, at [08, 10, vLoop];

        otLow  uMask1,                                                      c2, at [08, 10, getHalftoneWord-return];
startVLoop081:
        temp3Low  uNWordsM1,                                                c3;

        MAR  [destAddrHigh, destAddrLow + 0],                               c1;
        Xbus  uCombinationRule, XDisp,                                      c2;
        sourceIndex  MD, L3  0, DISP4 [comb0C],                             c3;


{-------skew=0, HForm=~nil, SForm=~nil, Preload=TRUE----------   INVALID  --------------}
startVLoop09:
        GOTO [iteration],                                                   c1, at [09, 10, vLoop];


{-------skew=0, HForm=~nil, SForm=nil, Preload=FALSE--------------}
startVLoop0A:
        sourceIndex  uVDir, CALL [getHalftoneWord],                         c1, at [0A, 10, vLoop];

        temp3Low  uNWordsM1,                                                c2, at [0A, 10, getHalftoneWord-return];
startVLoop0A1:
        otLow  uMask1,                                                      c3;

        MAR  [destAddrHigh, destAddrLow + 0],                               c1;
        Xbus  uCombinationRule, XDisp,                                      c2;
        sourceIndex  MD, L3  0, DISP4 [comb0E],                             c3;


{-------skew=0, HForm=~nil, SForm=nil, Preload=TRUE----------    INVALID  --------------}
```

```
startVLoop0B:
        GOTO [iteration],                                                       c1, at [0B, 10, vLoop];


{-------skew=0, HForm=nil, SForm=~nil, Preload=FALSE--------------}
startVLoop0C:
        temp2Low  uHalftoneWord  temp2Low xor ~temp2Low,                        c1, at [0C, 10, vLoop];
        otLow  uMask1, GOTO [startVLoop081],                                    c2;


{-------skew=0, HForm=nil, SForm=~nil, Preload=TRUE----------    INVALID  --------------}
startVLoop0D:
        GOTO [iteration],                                                       c1, at [0D, 10, vLoop];


{-------skew=0, HForm=nil, SForm=nil, Preload=FALSE--------------}
startVLoop0E:
        temp2Low  uHalftoneWord  temp2Low xor ~temp2Low,                        c1, at [0E, 10, vLoop];
startVLoop0E1:
        temp3Low  uNWordsM1, GOTO [startVLoop0A1],                              c2;


{-------skew=0, HForm=nil, SForm=nil, Preload=TRUE -------       INVALID --------------}
startVLoop0F:
        Noop,                                                                   c1, at [0F, 10, vLoop];
iteration:
        Noop,                                                                   c2;
        GOTO [startVLoop0F],                                                    c3;




{-------------------------------------------------------------------------------------------
 ------------------------ Start of Horizontal Loop ------------------------------------------
 -------------------------------------------------------------------------------------------
 ---------------------------------------------------------------------
  ----          sourceAddrHigh, sourceAddrLow           (RH2, R2)       --
  ----          destAddrHigh, destAddrLow               (RH1, R1)       --
  ----          temp1High, temp1Low(Halftone Form)      (RH4, R4)       --
  ----          temp2Low : prevWord             (R5)                    --
  ----          sourceIndex : skewWord          (R0)                    --
  ----          skew :                          (RH0)                   --
  ----          temp3Low : loop conter(word)    (R6)                    --
  ----          otLow : mask1(mergeMask)        (R3)                    --
 ---------------------------------------------------------------------}


{-------------------------------------
 ------ no Mesa Interrupt ------------
 -------------------------------------}
reentryBitblt:
        destAddrLow  destAddrLow + temp3Low, L1Disp,
                BRANCH [sFormNonNilMIntCheck, sFormNilMIntCheck, 0D],           c2, at [bbtCheckMInt, 10, noMesaInterrupt-return];


sFormNilMIntCheck:
        Q  uHDir, DISP4 [vLoop],                                                c3; { update destAddr }


sFormNonNilMIntCheck:
        Q  uSourceDelta, CANCELBR [$, 0F],                                      c3;

        sourceAddrLow  sourceAddrLow + Q,                                       c1;
        L1Disp, GOTO [sFormNilMIntCheck],                                       c2;


{-------------------------------------
 ---------- Mesa Interrupt -----------
 -------------------------------------}
 {save the bitBlt parameters into the mesa stack, and restore smalltalk ip, stackp, home

 temp2Low=0F , this value is store in checkMesaInterrupt}
mesaIntInBitblt:
        uDestAddrLow  destAddrLow,                                              c3, at [bbtCheckMInt, 10, mesaInterrupt-return];

mesaIntInBitblt1:
        destAddrLow  destAddrHigh, stackP  temp2Low,                            c1;
        sourceIndex  uCombinationRule,                                          c2;
        sourceIndex  sourceIndex LRot8,                                         c3;

        destAddrLow  destAddrLow or sourceIndex,                                c1;
        uDestAddrHigh  destAddrLow,                                             c2; {save highaddr of dest and combination Rule }
        sourceIndex  uDY,                                                       c3;

mesaIntInBitblt2:
        sourceIndex  sourceIndex and temp2Low,                                  c1; {08-0B: dy's offset}
        sourceIndex  sourceIndex LRot4,                                         c2;
        Q  skew,                                                                c3;

mesaIntInBitblt3:
        sourceIndex  sourceIndex or Q,                                          c1; {00-03: dy's offset, 04-07: skew}
        sourceIndex  sourceIndex LRot8,                                         c2;
        temp2Low  uBooleans, XDisp,                                             c3;

mesaIntInBitblt4:
        temp2Low  temp2Low and 0F, DISP4 [shForm, 9],                           c1;
```

```
{halftone=nil, source~=nil}
        uSourceAddrLow  sourceAddrLow,                                  c2, at [0D, 10, shForm]; { sourceAddrLow }
        sourceAddrLow  sourceAddrHigh,                                  c3;

        sourceAddrLow  sourceAddrLow LRot8,                             c1;
        uSaveHighAddr  sourceAddrLow,                                   c2; {00-07: sourceAddrHigh }
saveDelta:
        temp1Low  uSourceDelta,                                         c3;

        temp1Low  temp1Low and 0FF, GOTO [bothNil],                     c1;


{halftone~=nil, source=nil}
        uHalftoneAddrLow  temp1Low,                                     c2, at [08, 10, shForm];
        temp1Low  temp1High,                                            c3;

        uSaveHighAddr  temp1Low, GOTO [bothNil],                        c1; {08-0F: halftoneAddrHigh}


{halftone~=nil, source~=nil}
        uHalftoneAddrLow  temp1Low,                                     c2, at [9, 10, shForm]; {save halftoneAddrLw }
        Q  temp1High,                                                   c3;

        uSourceAddrLow  sourceAddrLow,                                  c1; {sacve sourceAddrLow}
        sourceAddrLow  sourceAddrHigh,                                  c2;
        sourceAddrLow  sourceAddrLow LRot8,                             c3;

        Q  Q or sourceAddrLow,                                          c1; {00-07:sourceAddrHigh, 08-0F:halftoneAddrHigh}
        uSaveHighAddr  Q, GOTO [saveDelta],                             c2;


{halftone=nil, source=nil}
bothNil:
        temp3Low  uDestDelta,                                           c2, at [0F, 10, shForm];
        temp3Low  temp3Low and 0FF,                                     c3;

        temp1Low  temp1Low LRot8,                                       c1;
        temp1Low  temp1Low or temp3Low,                                 c2;
        uDestDelta  temp1Low,                                           c3; {00-07:sourceDelta, 08-0F: destDelta}

saveDirection:
        Xbus  uHDir, XHDisp,                                            c1;
        Xbus  uVDir, XHDisp, BRANCH [hDirPlusSave, hDirMinusSave, 2],   c2;


hDirPlusSave:
         BRANCH [vDirPlusSave, vDirMinusSave, 2],                       c3;


vDirMinusSave:
        temp2Low  temp2Low or 40, GOTO [saveMisc],                      c1; {vDir = -1}


hDirMinusSave:
        temp2Low  temp2Low or 20, BRANCH [vDirPlusSave, vDirMinusSave, 2],  c3; {hDir = -1}


vDirPlusSave:
        Noop,                                                           c1;
saveMisc:
{00-03: dy's offset, 04-07:skew, 09:vDir=-1, 0A:hDir=-1, 0C-0F: booleans}
        sourceIndex  sourceIndex or temp2Low, L2  bitbltNotFinished,    c2;
        uMisc  sourceIndex, CALL [restoreStatus],                       c3;

        L0Disp,                                                         c3, at [bitbltNotFinished, 10, restoreStatus-return];

        XC2npcDisp, DISP2 [ipAjustInBBT],                               c1;


        CANCELBR [ipAjustedInBBT, 0F],                                  c2, at [0, 4, ipAjustInBBT];
ipAjustedInBBT:
        temp1Low  0, GOTO [saveSmalltalkStateBank0],                    c3;


        temp1Low  0, BRANCH [pc16One, pc16Zero, 0E],                    c2, at [1, 4, ipAjustInBBT];


pc16One:
        Cin  pc16, GOTO [saveSmalltalkStateBank0],                      c3;


pc16Zero:
        ipLow  ipLow - 1, Cin  pc16, GOTO [saveSmalltalkStateBank0],    c3;


        ipLow  ipLow - 1, CANCELBR [ipAjustedInBBT, 0F],                c2, at [2, 4, ipAjustInBBT];


        temp1Low  0, BRANCH [pc16One1, pc16Zero1, 0E],                  c2, at [3, 4, ipAjustInBBT];


pc16One1:
        ipLow  ipLow - 1, Cin  pc16, GOTO [saveSmalltalkStateBank0],    c3;
```

```
pc16Zero1:
        ipLow  ipLow - 2, Cin  pc16, GOTO [saveSmalltalkStateBank0],            c3;


allFinished:
        temp2High  uRumRecordHigh,                                              c2;
allFinished1:
        temp2Low  uRumRecordLow,                                                c3;

allFinished3:
        temp2Low  temp2Low + cursorBitmapOopOffset,                             c1;
        L2  bitbltFinished,                                                     c2;
        stackP  1, CALL [restoreStatus],                                        c3; { restore the Mesa stack Pointer }

        otLow  uDestBitMap,                                                     c3, at [bitbltFinished, 10, restoreStatus-return];

        MAR  [temp2High, temp2Low + 0],                                         c1;
        L1  getCursorMap,                                                       c2;
        Q  MD,                                                                  c3; { get current cursor map }

        []  Q xor otLow, ZeroBr,                                                c1;
        BRANCH [noCurrentCursor, yesCurrentCursor],                             c2;


yesCurrentCursor:
        temp3High  CSBforCursorHigh, CALL [otMap2Bank0],                        c3;

yesCurrentCursor1:
        temp1Low  temp1Low + sizeFieldOffset,                                   c1, at [getCursorMap, 10, otMap2Bank0-return];
        temp3Low  CSBforCursorLow,                                              c2;
        Q  10,                                                                  c3;

        MAR  [temp1High, temp1Low + 0], L1  cursorWidth1,                       c1;
        CALL [checkSmallInt],                                                   c2;

        []  temp2Low {size} - Q, NegBr,                                         c1, at [cursorWidth1, 10, checkSmallInt-return]; { size <
        16?? }
        temp1Low  temp1Low + 1, BRANCH [sizeGE10x, sizeLT10x],                  c2; { point the first (??) field of object }


sizeGE10x:
        temp2Low  11, GOTO [restoreCursorPattern],                              c3;  { loop ocunt = 16'd}


sizeLT10x:
        temp2Low  temp2Low + 1,                                                 c3; { loop count = size of currentcursorbitmap}

restoreCursorPattern:
        temp2Low  temp2Low - 1, ZeroBr,                                         c1;
        BRANCH [continueRestoreCursor, endRestoreCursor],                       c2;


continueRestoreCursor:
        Noop,                                                                   c3;

        MAR  [temp3High, temp3Low + 0],                                         c1;
        temp3Low  temp3Low + 1,                                                 c2;
        Q  MD,                                                                  c3; { get current pattern }

        MAR  [temp1High, temp1Low + 0],                                         c1;
        MDR  Q,                                                                 c2;  { save current pattern to ...}
        temp1Low  temp1Low + 1, GOTO [restoreCursorPattern],                    c3;


endRestoreCursor:
        GOTO [nextByteCodeInBank0],                                             c3;


noCurrentCursor:
        GOTO [nextByteCodeInBank0],                                             c3;

    { the receiver returns itself, so there is no need for smalltalk stack clean up}




{-------------------------------------------------}
destWidthLess0:
        GOTO [noTransfer1],                                                     c3;


destHeightLess0:
        L2  noTransfer, GOTO [noTransfer3],                                     c2;


clipWidthLess0:
        GOTO [noTransfer2],                                                     c1;


clipHeightLess0:
        L2  noTransfer, GOTO [noTransfer3],                                     c2;


widthLT0:
        GOTO [noTransfer1],                                                     c3;
```

```
heightLT0:
        GOTO [noTransfer2],                                             c1;

noTransfer1:
        L2  noTransfer,                                                 c1;

noTransfer2:
        L2  noTransfer,                                                 c2;

noTransfer3:
        CALL [restoreStatus],                                           c3;

        GOTO [nextByteCodeInBank0],                                     c3, at [noTransfer, 10, restoreStatus-return];

hBitMapNoOop:
        L2  primFail, GOTO [primitiveFailBitBlt1],                      c3;

halftoneFormInvalid:
        L2  primFail, GOTO [primitiveFailBitBlt3],                      c2;

destFormInvalid:
        L2  primFail, GOTO [primitiveFailBitBlt3],                      c2;

sourceFormInvalid:
        L2  primFail, GOTO [primitiveFailBitBlt3],                      c2;

primitiveFailBitBlt1:
        Noop,                                                           c1;
primitiveFailBitBlt2:
        Noop,                                                           c2;
primitiveFailBitBlt3:
        stackP  1, CALL [restoreStatus],                                c3;

        temp1Low  2, GOTO [saveSmalltalkStateBank0],                    c3, at [primFail, 10, restoreStatus-return];

destinationNonOop:
        CANCELBR [$, 1],                                                c1;
sourceNonOop:
        CANCELBR [$, 1],                                                c2;
halftoneNonOop:
        L2  primFail, CANCELBR [primitiveFailBitBlt1, 1],               c3;

combinationRuleTooBig:
        L2  primFail, GOTO [primitiveFailBitBlt1],                      c3;



{ Edit history:

22-Jan-86 15:55:22      Tokunaga        for stretched
20-Jan-86 10:31:02      Tokunaga        exchange uDestBitMap and uCombinationRule, and modify the save and restore ConbinationRule value
when MInt occur.
27-Dec-85 14:35:33      Tokunaga        refine the routine concerning with saving and restoring copyBits status when mesa Int occur
 7-Nov-85 18:48:48      Tokunaga        modify the routine for getting the actual address of BitMap Object (calculateOffset)
16-Oct-85 16:59:32      Sakakibara      change usource
16-Oct-85 16:34:22      Sakakibara      change several points (by Tokunaga)
30-Sep-85 15:56:37      Sakakibara      Fix temp1Low value at noPCarryC
27-Sep-85 16:34:57      Sakakibara      Added combinationRule=16 check
27-Sep-85 14:00:35      Sakakibara      Bug fix PrimitiveFail
27-Sep-85 13:59:42      Matsumoto       Add CANCELBR Mask
13-Sep-85 13:28:45      Tokunaga        Add the adjustment for smalltalk instruction pointer when Mesa Int occur during
primitiveCopyBits and remove calling updateCursor.
20-Jun-85  9:53:33      Tokunaga        remove checkSmallInt to bbtsubs.mc }
```

```
{ BBTSubs.mc

Subroutines for BitBlt primitive for Rum, the Dandelion Smalltalk-80 microcoded virtual machine.

        by T Tokunanga, M Sakakibara, J Trow

         9-Feb-86 17:43:46

Copyright 1985, 1986 by Xerox Corporation.  All rights reserved. }



{checkSmallInt: *************** stretched ***************
subroutine for checking smallinteger & convert to normal integer
 At entry : c3
 At Return : c1
  Link register : L1
  to be checked : temp2Low
  result : temp2Low

  Note: temp1Low is incremented.

  }


checkSmallInt:
        temp2Low  MD, XHDisp,                                                   c3;

        []  temp2Low and nonPointerMask, ZeroBr,
                BRANCH [posSmallInteger, negSmallInteger, 2],                   c1;


posSmallInteger:
        temp2Low  RRot1 temp2Low, BRANCH [oops, notOops],                       c2;


negSmallInteger:
        temp2Low  RRot1 (temp2Low or 3), BRANCH [oops, notOops],                c2;


notOops:
        temp2Low  RRot1 temp2Low,                                               c3;

        Noop,                                                                   c1;
        temp1Low  temp1Low + 1, L1Disp,                                         c2;
        RET [checkSmallInt-return],                                             c3;


oops:
        L2  primFail, GOTO [primitiveFailBitBlt1],                              c3;




 {checkSmallInt2: *************** stretched ***************
subroutine for checking smallinteger & convert to normal integer
 At entry : c3
 At Return : c1
  Link register : L1
  to be checked : temp2Low
  result : temp2Low

  Note: temp1Low is incremented.

  }

checkSmallInt2:
        temp2Low  MD, XHDisp,                                                   c3;

checkSmallInt2c1:
        []  temp2Low and nonPointerMask, ZeroBr,
                BRANCH [posSmallInteger2, negSmallInteger2, 2],                 c1;


posSmallInteger2:
        temp2Low  RRot1 temp2Low, BRANCH [oops2, notOops2],                     c2;


negSmallInteger2:
        temp2Low  RRot1 (temp2Low or 3), BRANCH [oops2, notOops2],              c2;


notOops2:
        temp2Low  RRot1 temp2Low,                                               c3;

        Noop,                                                                   c1;
        temp1Low  temp1Low + 1, L1Disp,                                         c2;
        RET [checkSmallInt2-return],                                            c3;


oops2:
        L2  primFail, GOTO [primitiveFailBitBlt1],                              c3;
```

```
{
 ----  ex: mask1  RightMasks at: startBits + 1  ---------

  Entry point : c2, c3  Exit point : c3
  the argument is stored in temp3Low , if it isn't between 0 and 10'X then primitiveFail.
  Return link: L3
  Result : temp2Low }
makeRightMasks:
        Noop,                                                                   c2;
makeRightMasks3:
        temp2Low  temp2Low xor temp2Low,                                        c3;

{ argument is between 0 and 10'X }
argOKInBBT:
        temp3Low  temp3Low - 1, NegBr,                                          c1; {decrement the shift counter }
        L3Disp, BRANCH [shiftLoop, shiftEnd],                                   c2;


shiftLoop:
        temp2Low  temp2Low LShift1, SE  1, CANCELBR [argOKInBBT, OF],           c3;


shiftEnd:
        RET [makeRightMasks-return],                                            c3;




{ This routine is for checking MesaInterrupt , ST80 BitBlt use this routine.  }

{
 return link : L2
 No Interrupt Exit point: return cycle = c1, pending Xdipatch by uBooleans
                temp3Low : DestDelta
 Interrupt Exit point :  c2}

{
        PC : MesaStatePC,
        pc16 : MesaStatePC16 (pc16 = Bit.16)
        rhPC : MesaStateRhPC
        IBptr: MesaStateIBPtr
        IB   : MesaStateIB

        uPPCross:        0 --> no Cross
                        ~0 --> Cross
        UvChigh : high Address bot od code segment
        UvPCpage: virtual page No of current mesa code
}


{ register definition }
        RegDef [MesaStatePC, U, 50];
        RegDef [MesaStateIBPtr, U, 53];
        RegDef [MesaStateIB, U, 54];

        RegDef [MesaStateRhPC, U, 51];

checkMesaInterrupt:
         temp2Low  OF, MesaIntBr,                                               c1;{using if mesa interrupt occur as a stack value}
checkuWP:
        temp3Low  uWakeupPending, ZeroBr,
                BRANCH [bitBltNoInterrupt, maybeInterrupt],                     c2;


bitBltNoInterrupt:
        temp3Low  uDestDelta, L2Disp, CANCELBR [noInterrupt, 1],                c3;


maybeInterrupt:
        temp3Low  uDestDelta, L2Disp, BRANCH [bitBltInterrupt, noInterrupt],    c3;


noInterrupt:
        Xbus  uBooleans, XDisp, RET [noMesaInterrupt-return],                   c1;


bitBltInterrupt:
        temp3Low  MesaStatePC,   CANCELBR [$, OF],                              c1;
        temp3Low  temp3Low -1,                                                  c2;
        temp3High  MesaStateRhPC,                                               c3;

bitBltNoCross:
        MAR  [temp3High, temp3Low + 0], Xbus  MesaStateIBPtr, XDisp,            c1;
        MesaStatePC  temp3Low, DISP4 [bitBltRestoreIB, 0C],                     c2;


{Empty} Noop, GOTO [clearCrossIndicator],                                       c3, at [0C, 10, bitBltRestoreIB];
{Byte}  temp3Low  MD, GOTO [clearCrossIndicator],                               c3, at [0D, 10, bitBltRestoreIB];
{Full}  GOTO [bitBltNoCross],                                                   c3, at [0E, 10, bitBltRestoreIB]; {forever iteration}
{Word}  temp3Low  MD, GOTO [clearCrossIndicator],                               c3, at [0F, 10, bitBltRestoreIB];

{there is no check for pc16, bacause 1 or 2 bytes are stored into IB from MesaStateIB coressponding to the po16 respectively in
InitAndMesaStateSaveAndRestore.mc }
```

```
clearCrossIndicator:
        uPCCross  destAddrLow xor destAddrLow,   L2Disp,                    c1;{uPPCross  0}
        MesaStateIB  temp3Low, RET [mesaInterrupt-return],                  c2;




{
  Subroutine : getSTFormMapBase
   Entry :      otLow = Form Oop (getSTFormMapBase) --- SourceForm
                otLow = Form BitMap Oop (getSTFormMapBase1) --- DestForm
                L2 : returnLink
  Exit  : temp1High, temp1Low has the map base
          otLow : Oop  for bitmap
          temp2Low : the Words No. for 1 horizontal line,

  cycle: Entry c1, Exit c1
 }

getSTFormMapBase1: { for DESTINATION FORM }
        Xbus  uBooleans, XLDisp, L3  checkSource,                           c3; { destBitMap = current BitMap ? }

        Q  uDestHeightA, BRANCH [destNEQCurrent, destEQCurrent, 1],         c1;
destNEQCurrent:
{destWidth, destHeight have been already checked whether they would be greater than 1024, 1010 respectively }
        temp2Low  uDestWidthA, CALL [checkWidthAndHeight],                  c2;


destEQCurrent:
        GOTO [yesCurrent],                                                  c2;


getSTFormMapBase: { for SOURCE FORM }
        temp3Low  uCurrentDispBitMap, L1  getMap,                           c2;
        CALL [otMap2Bank0],                                                 c3;

        MAR  temp1Low  [temp1High, temp1Low + formMapBaseIndex],
             L1  getWidth,                                                  c1, at [getMap, 10, otMap2Bank0-return];
getMapBase2:
        temp1Low  temp1Low + 1,
              BRANCH [noPCInGetMapBase2, yesPCInGetMapBase2, 1],            c2;


yesPCInGetMapBase2:
        temp1Low  temp1Low + 0FF,                                           c3;

        MAR  [temp1High, temp1Low + 0], GOTO [getMapBase2],                 c1;


noPCInGetMapBase2:
        otLow  MD, XDisp,       { get SourceForm BitMap Oop }               c3;

getMapBase3:
        MAR  [temp1High, temp1Low + 0], DISP4 [sourceBitMapOop, 0C],        c1; {********}


        []  temp3Low xor otLow, ZeroBr, L3  checkSource, GOTO [bitMapOop],  c2, at [0D, 10, sourceBitMapOop]; {********}
        []  temp3Low xor otLow, ZeroBr, L3  checkSource, GOTO [bitMapOop],  c2, at [0E, 10, sourceBitMapOop]; {********}
        []  temp3Low xor otLow, ZeroBr, L3  checkSource, GOTO [bitMapOop],  c2, at [0F, 10, sourceBitMapOop]; {********}


bitMapOop:
        temp2Low  MD, XHDisp, BRANCH [{CALL} checkSmallInt2c1, yesCurrent1],  c3;


noCurrent1:
        MAR  [temp1High, temp1Low + 0], L1  getHeight,                      c1, at [getWidth, 10, checkSmallInt2-return];
        temp3Low  temp2Low, CALL [checkSmallInt2],                          c2; { for checkWidthAndHeight routine}

noCurrent2:
        Q  temp2Low, L1  getMapBase,                  {Q = Form.Height}     c1, at [getHeight, 10, checkSmallInt2-return];
        temp2Low  temp3Low, CALL [checkWidthAndHeight],                     c2; {temp2Low = Form.Height}

{  At this point,  Q <== INT(((BitWidth-1)/16) + 1 )*height, temp2Low = Words /horizontal line.
   Also, otLow = Bitmap Oop(Source, Destination)}

        temp3Low  Q + objectHeaderSize, CALL [otMap2Bank0],                 c3, at [checkSource, 10, checkWidthAndHeight-return];
        {otLow = BitMap Oop }

noCurrent4:
        MAR  temp1Low  [temp1High, temp1Low + sizeFieldOffset],             c1, at [getMapBase, 10, otMap2Bank0-return];
getMapBase4:
        BRANCH [noPCGetMapBase4, yesPCGetMapBase4, 1],                      c2;


yesPCGetMapBase4:
        temp1Low  temp1Low + 0FF + 1,                                       c3;

        MAR  [temp1High, temp1Low + 0], GOTO [getMapBase4],                 c1;


noPCGetMapBase4:
        Q  MD,                                                              c3; { get BitMap Size }
```

```
noCurrent6:
        []  temp3Low xor Q, ZeroBr, {BitMapSize = (width * height) ? }          c1;
        temp1Low  temp1Low + 2, { point the actual address of BitMap }
                        BRANCH [funnyBitMapSize, correctBitMapSize],            c2;


correctBitMapSize:
        L2Disp,                                                                 c3;

        RET [getSTFormMapBase-return],                                          c1;


yesCurrent1:
        CANCELBR [$, 0F],                                                       c1;
        Noop,                                                                   c2;
yesCurrent:
        temp2Low  40, L2Disp,                                                   c3; { INT [(1024 + 15)/16] = INT [64.9xx] = 64}

yesCurrent2:
        temp1High  temp1Low  (temp1Low xor temp1Low) LRot0,
                RET [getSTFormMapBase-return],                                  c1; { point the Low real memory }


        L2  primFail, GOTO [primitiveFailBitBlt3],                              c2, at [0C, 10, sourceBitMapOop];


funnyBitMapSize:
        L2  primFail, GOTO [primitiveFailBitBlt1],                              c3;




{
 Subroutine : bbtMultiply
  C  A*B
 Entry :        temp2Low        = A
                Q               = B

                L2              = return link
 Exit:          Q               = result
                temp1Low, temp3Low, sourceIndex are smashed

 entry: c1, c2, c3  exit : c1
 }


bbtMultiply2:
        Noop,                                                                   c2;
bbtMultiply3:
        Noop,                                                                   c3;

bbtMultiply:
        temp1Low  0,                                                            c1; { product }
        sourceIndex  10,                                                        c2; { loop counter for mult-loop }
bbtMulLoop:
        []  Q and 1, NZeroBr,                                                   c3;

        sourceIndex  sourceIndex - 1, ZeroBr,
                BRANCH [bbtMulDigit0, bbtMulDigit1],                            c1;


bbtMulDigit0:
        temp1Low  DARShift1 (temp1Low + 0), BRANCH [bbtMulLoop, bbtMulEnd],     c2;


bbtMulDigit1:
        temp1Low  DARShift1 (temp1Low + temp2Low),
                BRANCH [bbtMulLoop, bbtMulEnd],                                 c2;


bbtMulEnd:
        Q  ~ Q, L2Disp,                                                         c3;

        RET [bbtMultiply-return],                                               c1; {Result is Q}




{ Subroutine: getSeveralMasks
        description :           calculate the several masks ( skew, mask1, mask2 )
        sourceIndex :           uW - 1
        L1 :                    return Link
 Exit:  skew :                  skew
        uMask1 :                mask1
        uMask2(temp2Low) :      mask2

 Note: smashed register -- temp1Low, temp2Low

 Entry: c1, Exit: c1
 }

getSeveralMasks:
        temp1Low  uDX,  L3  getMask1,                                           c1;
        temp3Low  temp1Low and 0F,                                              c2;  { dx and 15 }
```

```
        temp3Low  10 - temp3Low,                                    c3;  {16 - (dx and 15)}

        uStartBits  temp3Low, CALL [makeRightMasks],                c1;

        uMask1  temp2Low,                                           c1, at [getMask1, 10, makeRightMasks-return];
        sourceIndex  temp1Low + sourceIndex, L3  getMask2,          c2; { sourceIndex = uW - 1 }
        temp3Low  sourceIndex and 0F,                               c3; { (dx+W-1) bitAnd: 15}

        temp3Low  0F - temp3Low, CALL [makeRightMasks],             c1; { 15 - ((dx+W-1) bitAnd: 15)}

        uMask2  ~temp2Low ,                                         c1, at [getMask2, 10, makeRightMasks-return];
        temp3Low  uSX, L3  computeSkewMasks,                        c2; { to make skewMask}
        temp3Low  temp3Low - temp1Low,                              c3;

        temp3Low  temp3Low and 0F, ZeroBr, L3  computeSkewMasks,    c1;
        skew  temp3Low LRot0, BRANCH [skewNonZero, skewZero],       c2;


skewNonZero:
        temp3Low  10 - temp3Low,                                    c3;

        sourceIndex  uBooleans, CALL [makeRightMasks],              c1;

        uSkewMask  temp2Low,                                        c1, at [computeSkewMasks, 10, makeRightMasks-return];
        sourceIndex  sourceIndex and 87, GOTO [storeSkewZewoIndi],  c2;


skewZero:
        sourceIndex  uBooleans,                                     c3;

        uSkewMask  sourceIndex xor sourceIndex,                     c1;
        sourceIndex  sourceIndex or 8,                              c2;
storeSkewZewoIndi:
        uBooleans  sourceIndex, GOTO [getSeveralMasksRet],          c3;




{restoreStatus
        Entry : c1, c2, c3, Exit: c2
        return Link: L2
}
restoreStatus:
        ipLow  uSaveIPL,                                            c1;
        temp1Low  uSaveIPH,                                         c2;
        ipHigh  temp1Low LRot0,                                     c3;

        stackLow  uSaveStackL,                                      c1;
        stackHigh  uSaveStackH,                                     c2;
        homeLow  uSaveHomeL,                                        c3;

        temp1Low  uSaveHomeH,   L2Disp,                             c1;
        homeHigh  temp1Low LRot0, RET [restoreStatus-return],       c2;




{
Subroutine: bbtCheckRange
Entry:
        destAddrLow     : destX         or destY
        souuceAddrLow   : sourceX       or sourceY
        sourceIndex     : destWisth     or destHeight
        temp2Low        : clipX         or clipY
        temp3Low        : clipWidth     or clipHeight
Exit:
        sourceAddrLow : sx or sy
        destAddrLow   : dx or dy
        sourceIncex   : w or h
        returnLink : L2
        entry: c1, exit: c1
}

bbtCheckRange:
        []  destAddrLow - temp2Low, NegBr,                          c1;
        Q  temp2Low - destAddrLow, BRANCH [destXGE, destXLT],       c2; {clipX - destX }


destXLT: {no}
        destAddrLow  temp2Low,                                      c3; { dX  clipX }

        sourceAddrLow  sourceAddrLow + Q,                           c1; { sX  sourceX + (clipX - destX)}
        sourceIndex  sourceIndex - Q,                               c2; { w   width - (clipX - destX)}
destXGE: {yes}
        Q  destAddrLow + sourceIndex,                               c3; { dx + w }

checkRightRange:
        temp2Low  temp2Low + temp3Low, L2Disp,                      c1; { clipX + clipWidth }
        []  temp2Low - Q, NegBr, BRANCH [checkX, checkY, 0E],       c2; {(dx+w) > (clipX+clipWidth) ??}


checkX:
        temp1Low  uDestWidthA, BRANCH [clipGEOnRightSide, clipLTOnRightSide],   c3;
```

```
checkY:
        temp1Low  uDestHeightA, BRANCH [clipGEOnRightSide, clipLTOnRightSide],   c3;


clipGEOnRightSide: { clipX+clipWidth >= dx+w }
        Q  temp1Low - Q, NegBr,                                                  c1; { dx+w > destWidth ?}
        BRANCH [rangeLEDestForm1, rangeGTDestForm1],                             c2;


rangeLEDestForm1:
        GOTO [nextStep1],                                                        c3;


rangeGTDestForm1:
        sourceIndex  sourceIndex + Q, GOTO [nextStep1],                          c3;


clipLTOnRightSide: { clipX+clipWidth < dx+w }
        []  temp1Low - temp2Low, NegBr,                                          c1;
        temp2Low  Q - temp2Low, {temp2Low = (dx+w) - (clipx+clipWidth)}
                BRANCH [rangeLEDestForm2, rangeGTDestForm2],                     c2;


rangeLEDestForm2:
        sourceIndex  sourceIndex - temp2Low, GOTO [nextStep1],                   c3;


rangeGTDestForm2:
        Q  Q - temp1Low, { Q = (dx+w) - Form.Width }                             c3;

rangeLTDestForm21:
        sourceIndex  sourceIndex - Q, GOTO [nextStep],                           c1;


nextStep1:
        Noop,                                                                    c1;
nextStep:
        Xbus  uBooleans, XLDisp,                                                 c2; { check DestForm = currentScreen?}
        BRANCH [noCurrentInCheck, yesCurrentInCheck, 1],                         c3; { dx < 0 ?? }


yesCurrentInCheck:
        []  destAddrLow, NegBr,                                                  c1;
        BRANCH [destNoNegative, destYesNegative],                                c2;


{ if destForm = currentScreen, and dx(dy) < 0, we have to make dx(dy) = 0 and ajust the sx(sy), w(h) }
destYesNegative:
        sourceAddrLow  sourceAddrLow - destAddrLow,                              c3;

        sourceIndex  sourceIndex + destAddrLow,                                  c1;
        destAddrLow  0,                                                          c2;
destNoNegative:
        Noop,                                                                    c3;

noCurrentInCheck:
        []  sourceAddrLow, NegBr,                                                c1;
        BRANCH [noSourceNegative, yesSourceNegative],                            c2;


noSourceNegative:
        L2Disp,                                                                  c3;

        RET [bbtCheckRange-return],                                              c1;


yesSourceNegative:
        destAddrLow  destAddrLow - sourceAddrLow,                                c3;

        sourceIndex  sourceIndex + sourceAddrLow,                                c1;
        sourceAddrLow  sourceAddrLow xor sourceAddrLow,
                GOTO [noSourceNegative],                                         c2;




{ Subroutine: calSkewWord       -- 4 click }
{
 Entry  : c1
        sourceIndex : prevWord
        Q : uHDir

 Exit   : c3
        temp2Low : skewWord
        Q :  New prevWord for next using

 otLow : smashed

}

calSkewWord:
        MAR  [sourceAddrHigh, sourceAddrLow + 0],                                c1;
calSkewWord1:
        sourceAddrLow  sourceAddrLow + Q,                                        c2;
        Q  MD,                                                                   c3; { get this word }
```

```
        uPrevWord  Q,                                                   c1;
        Noop,                                                           c2;
        Noop,                                                           c3;

calSkewWord2:
        sourceIndex  sourceIndex and uSkewMask,                         c1; { prevWord  prevWord bitAnd: skewMask }
        Q  ~uSkewMask and Q,                                            c2;{ thisWord bitAnd: skewMaks bitInvewrt }
calSkewWord3:
        Q   sourceIndex or Q, Xbus  skew, XDisp,                        c3;

bbtRotation:
        sourceIndex  LRot1 Q, DISP4 [bbtRot],                           c1;


        L2Disp, temp2Low  Q, GOTO [bbtShift0],                          c2, at [ 0, 10, bbtRot];
        L2Disp, temp2Low  sourceIndex, GOTO [bbtShift0],                c2, at [ 1, 10, bbtRot];
        L2Disp, temp2Low  LRot1 sourceIndex , GOTO [bbtShift0],         c2, at [ 2, 10, bbtRot];
        L2Disp, temp2Low  RRot1 Q, GOTO [bbtShift4],                    c2, at [ 3, 10, bbtRot];

        L2Disp, temp2Low  Q, GOTO [bbtShift4],                          c2, at [ 4, 10, bbtRot];
        L2Disp, temp2Low  LRot1 Q, GOTO [bbtShift4],                    c2, at [ 5, 10, bbtRot];
        L2Disp, temp2Low  LRot1 sourceIndex, GOTO [bbtShift4],          c2, at [ 6, 10, bbtRot];
        L2Disp, temp2Low  RRot1 Q, GOTO [bbtShift8],                    c2, at [ 7, 10, bbtRot];

        L2Disp, temp2Low  Q, GOTO [bbtShift8],                          c2, at [ 8, 10, bbtRot];
        L2Disp, temp2Low  LRot1 Q, GOTO [bbtShift8],                    c2, at [ 9, 10, bbtRot];
        L2Disp, temp2Low  LRot1 sourceIndex, GOTO [bbtShift8],          c2, at [0A, 10, bbtRot];
        L2Disp, temp2Low  RRot1 Q, GOTO [bbtShift12],                   c2, at [0B, 10, bbtRot];

        L2Disp, temp2Low  Q, GOTO [bbtShift12],                         c2, at [0C, 10, bbtRot];
        L2Disp, temp2Low  LRot1 Q, GOTO [bbtShift12],                   c2, at [0D, 10, bbtRot];
        L2Disp, temp2Low  LRot1 sourceIndex, GOTO [bbtShift12],         c2, at [0E, 10, bbtRot];
        L2Disp, temp2Low  RRot1 Q, GOTO [bbtShift0],                    c2, at [0F, 10, bbtRot];


bbtShift0:
        RET [calSkewWord-return],                                       c3;


bbtShift4:
        temp2Low  temp2Low LRot4, RET [calSkewWord-return],             c3;


bbtShift8:
        temp2Low  temp2Low LRot8, RET [calSkewWord-return],             c3;


bbtShift12:
        temp2Low  temp2Low LRot12, RET [calSkewWord-return],            c3;




{Subroutine: getHalftoneWord   ------  2-clicks}
{
 Entry:
        c2,
        sourceIndex : vertical direction
 Exit:
        c1,
        sourceIndex: halftoneWord
}

getHalftoneWord:
        temp2Low  uDY,                                                  c2;
        temp3Low  temp2Low and 0F,                                      c3; { dy bitAnd: 15}

        MAR  [halftoneAddrHigh, halftoneAddrLow + temp3Low],            c1;
        temp2Low  temp2Low + sourceIndex,
                BRANCH [noPCInGetHWord, yesPCInGetHWord, 1],            c2;{ dy  dy + vDir}


yesPCInGetHWord:
        halftoneAddrLow  temp3Low + halftoneAddrLow,                    c3;

        MAR  [halftoneAddrHigh, halftoneAddrLow + 0],                   c1;
        halftoneAddrLow  halftoneAddrLow - temp3Low,                    c2;
noPCInGetHWord:
        temp2Low  MD, uDY  temp2Low, L1Disp,                            c3;

        uHalftoneWord   temp2Low, RET [getHalftoneWord-return],         c1;




{
Subroutine: checkWidthAndHeight

 Entry:         temp2Low        = Form.Width
                Q               = Form.Height
                L3              = return Link

 Exit :         Width * Height > 64840 then primitiveFail, otherwise return

 Note :         temp1Low, otLow, Q, sourceIndex  is smashed by using bbtMultiply routine.
```

```
 }

checkWidthAndHeight:
        temp2Low  temp2Low - 1,                                                 c3;

checkWidthAndHeight1:
        temp2Low  temp2Low and ~0F, L2  widthAndHeight,                         c1;
        temp2Low  temp2Low LRot12 + 1,                                          c2; { width  INT((bitWidth - 1)/16) + 1}
        CALL [bbtMultiply],                                                     c3; { Q = width * height, temp1Low, sourceIndex are
        smashed}

{On Smalltalk-80 on 1108X(kiku-X), the largest size of BitMap is 64640 word, i.e. 1024*1010/16. If user try to create the form of size
with greater than 64640, automatically system modify the size with 64640, and not modify the width, height of the corresponding Form. So
we may have the trouble, since copyBits primitive refers Form.width, height when actually transferring the Bit Block.}
{Why the reason above, we check it describing below}
        [] Q, NegBr,                                                            c2, at [widthAndHeight, 10, bbtMultiply-return];
        temp1Low  0FC,
                BRANCH [checkWidthAndHeight4, maybeLargerThanMaxSize],          c3;


maybeLargerThanMaxSize:
        temp1Low  temp1Low LRot8,                                               c1;
        temp1Low  temp1Low or 80,                                               c2; {64640 = FC80'x = 1024*1010/16}
        [] temp1Low - Q, NegBr,                                                 c3;

        L3Disp, BRANCH [checkWidthAndHeight41, overBitMapMaxSize],              c1;


checkWidthAndHeight4:
        L3Disp,                                                                 c1;
checkWidthAndHeight41:
        RET [checkWidthAndHeight-return],                                       c2;


overBitMapMaxSize:
        temp2Low  0F1, CANCELBR [$, 0F],                                        c2;
        L2  primFail, GOTO [primitiveFailBitBlt1],                              c3;



{ Edit history:

22-Jan-86 16:17:09      Tokunaga.iwafx          modify getSTFormMapBase for stretch
21-Jan-86 16:46:27      Tokunaga.iwafx          modify the checkWidthAndHeight and getSTFormMapBase
17-Dec-86  9:00:16      Tokunaga.iwafx          add the checking routine for dx < 0 or not in bbtCheckRange when DestForm = CurrentScreen
 5-Nov-86  9:12:07      Tokunaga.iwafx          add the checking routine for BitMap.Size = (Width*Height) in getSTFormMapBase when bitMap
 oop is not current display bitMap.
 4-Nov-86 11:31:18      Tokuanga.iwafx          add the checking DestForm.width in bbtCheckRange routine.
 2-Nov-86 20:09:58      Tokunaga.iwafx          add checkSmallInt2 and checkWidthAndHeight routine
27-Sep-86 13:49:07      Sakakibara.iwafx        bug fix CANCELBR
13-Sep-86 11:36:53      Tokuanga.iwafx          remove "updateCursor" routine and RegDefs in checkMesaInterrupt routine.
20-Jun-86  9:52:54      Tokunaga.iwafx          add checkSmallInt from bbt.mc
10-Jun-86 19:38:06      Tokunaga.iwafx          add bitShift subroutine }
```

```
{ MemoryManglement.mc

Object creation, reference counting, stabilization, and other memory management stuff for Rum, the Dandelion Smalltalk-80 microcoded
virtual machine.

        by P McCullough, J Trow

        20-Jan-86 20:46:14

Copyright 1983, 1984, 1985, 1986 by Xerox Corporation.  All rights reserved. }



{for each of these entry points, uClassToInstantiate must be the oop of the class, temp3Low is the size in words or bytes.  temp3High is
the return linkage register}

createInstanceWithPointers:
        temp2Low  nilPointer,                                           c1;
        temp1Low  hasPointers,  GOTO [createInstance],                  c2;


createInstanceWithBytes:
        temp2Low  0,                                                    c1;
        [] temp3Low LRot0, XDisp,                                       c2;
        temp3Low  temp3Low + 1, BRANCH [$, byteCountIsOdd, 0E],         c3;

byteCountIsEven:
        temp1Low  evenBytes, GOTO [byteShift],                          c1;


byteCountIsOdd:
        temp1Low  oddBytes,                                             c1;
byteShift:
        temp3Low  RShift1 temp3Low, SE  0, GOTO [createInstance],       c2;


createInstanceWithWords:
        temp2Low  0,                                                    c1;
        temp1Low  hasWords, GOTO [createInstance],                      c2;


createLargePositiveInteger:
        temp1Low  classLargePositiveIntegerPointer,                     c1;
        uClassToInstantiate  temp1Low,                                  c2;
        GOTO [createInstanceWithBytes]                                  c3;




        { createInstance

        Create a new instance of a given class.

                input:  temp1Low is the odd byte and pointer bits of the delta word
                        temp2Low is the initial value of the fields of the instance
                        temp3Low is the size of the instance in words not including the header
                        uClassToInstantiate is the class of the new instance
                        temp3High is the return link

                output: uNewObject is the new instance
                        uNewObjectHigh/Low is the address of the new instance
                        uRequestedSize is the size of the new instance

                smash:  otLow, temp1High/Low, temp2High/Low, temp3High/Low, Q, uPredecessor, uFieldType, uDefault, uCurrentFreeChunkOop,
                uNextFreeChunk, L1, L2,  }

createInstance:
        uFieldType  temp1Low
                {save these for initializing the object and its ot entry},      c3;

        uDefault  temp2Low,                                             c1;
        temp3Low  temp3Low + objectHeaderSize, CarryBr,                 c2;
        Q  objectSizeTestLimit, BRANCH [$, massiveSenility2],           c3;

        [] temp3Low + Q, CarryBr,                                       c1;
        BRANCH [$, requestedSizeTooBig],                                c2;
        temp2High  uRumRecordHigh,                                      c3;

        temp2Low  uRumRecordLow,                                        c1;
        temp1Low  largestFreeChunkSize,                                 c2;
        [] temp3Low - temp1Low, CarryBr,                                c3;

        uRequestedSize  temp3Low, BRANCH [$, useBigFreeList],           c1;
        temp2Low  temp2Low + freeListsOffset,   {try specific list}     c2;
        Noop,                                                           c3;

        MAR  [temp2High, temp2Low + temp3Low],                          c1;
        CANCELBR [$, 0],                                                c2;
        otLow  MD, L2  creatingInstance {for nextFreeChunk},            c3;

        Noop,                                                           c1;
        Noop,                                                           c2;
        [] otLow and 3, ZeroBr,                                         c3;     {temp}
```

```
        uNewObject  otLow, BRANCH [$, tryBigList],                          c1;
        L1  gettingNextFreeChunk {for otMap2 inside nextFreeChunk},
              CALL [nextFreeChunk],   {got one}                             c2;

        MAR  [temp2High, temp2Low + temp3Low] {update free list head}       c1, at [creatingInstance, 10, nextFreeChunk-return];
        MDR  Q, CANCELBR [$, 0], LOOPHOLE[wok],                             c2;
        Q  temp1High, {save new object's address}                           c3;

        uNewObjectHigh  Q,                                                  c1;
        uNewObjectLow  temp1Low,                                            c2;
        GOTO [allocate],                                                    c3;


tryBigList:
        temp2Low  uRumRecordLow, GOTO [useBigFreeListA],                    c2;


{upon entry, temp2High/Low contains the rum record address. uRequestedSize is valid. temp3Low is the requested size}

useBigFreeList:
        Noop,                                                               c2;
useBigFreeListA:
        uPredecessor   0 {should be nilPointer},
              L1  gettingNextFreeChunk {for otmap call in nextFreeChunk},   c3;

        MAR  [temp2High, temp2Low + bigFreeListOffset],
              L2  consideringBigChunks,                                     c1;
        temp3Low  temp3Low + objectHeaderSize, CarryBr,
              {yields minimum splittable block size} CANCELBR [$, 0],       c2;
        otLow  MD {current free chunk}, BRANCH [$, massiveSenility4],       c3;

considerNextBigFreeChunk:
        Noop,                                                               c1;
        Noop,                                                               c2;
        []  otLow and 3, ZeroBr,                                            c3;     {temp}

        uNewObject  otLow, BRANCH [$, outOfChunks],                         c1;
        uCurrentFreeChunkOop  otLow, CALL [nextFreeChunk],                  c2;

        uNextFreeChunk  Q {remember next free chunk},                       c1, at [consideringBigChunks, 10, nextFreeChunk-return];
        temp1Low  temp1Low + sizeFieldOffset,                               c2;
        Noop,                                                               c3;

        MAR  [temp1High, temp1Low + 0],                                     c1;
        Noop,                                                               c2;
        Q  MD {size of current free chunk},                                 c3;

        []  Q xor uRequestedSize, ZeroBr,                                   c1;
        []  Q - temp3Low, CarryBr, BRANCH [$, exactFit],                    c2;
        BRANCH [$, canSubdivide],                                           c3;

iterate:
        uPredecessor  otLow,                                                c1;
        Noop,                                                               c2;
        otLow  uNextFreeChunk, GOTO [considerNextBigFreeChunk],             c3;


exactFit:
        temp1Low  temp1Low - sizeFieldOffset, CANCELBR [$, 1],              c3;

        Q  temp1High,                                                       c1;
        uNewObjectLow  temp1Low,                                            c2;
        uNewObjectHigh  Q, GOTO [splice],                                   c3;


canSubdivide:
        temp3Low  uRequestedSize,                                           c1;
        temp3Low {new size}  Q {current size} - temp3Low {requested size},  c2;
        Q  temp1High {part of new object's address},                        c3;

        MAR  [temp2High, temp2Low + freePointersOopOffset],                 c1;
        uNewObjectHigh  Q, CANCELBR [$, 0],                                 c2;
        otLow  MD {first free oop},                                         c3;

        Noop,                                                               c1;
        Noop,                                                               c2;
        []  otLow and 3, ZeroBr,                                            c3;     {temp}

        uNewObject  otLow, BRANCH [$, outOfOops],                           c1;
        Q  temp1Low - sizeFieldOffset, L1  splittingFreeChunk,              c2;
        Q {object address}  Q {chunk address} + temp3Low {chunk size},      c3;

{write the new size of the current free chunk (temp1High/Low still pointing at its size field)}

        MAR  [temp1High, temp1Low + 0],                                     c1;
        MDR  temp3Low {new chunk size},                                     c2;
        uNewObjectLow  Q,                                                   c3;

        CALL [getOtAddress] {free oop link},                                c1;

        MAR  [temp2High, temp2Low + freePointersOopOffset],                 c1, at [splittingFreeChunk, 10, getAddressReturn];
        MDR  temp1Low {new free oop head}, LOOPHOLE [wok], CANCELBR [$, 0], c2;
        temp2High  splittingFreeChunk,                                      c3;

        temp1Low  uNewObjectHigh, CALL [putOtFlags],                        c1;
```

```
        temp1Low  uNewObjectLow, CALL [putOtAddress],                    c1, at [splittingFreeChunk, 10, putFlagsReturn];

        temp1Low {chunk address}  temp1Low - temp3Low,                   c1, at [splittingFreeChunk, 10, putAddressReturn];
        temp2Low  largestFreeChunkSizeLessOne,                           c2;
        []  temp3Low {chunk size} - temp2Low, CarryBr,                   c3;

{should we move the current free chunk to a small free chunk list?}

        otLow  uCurrentFreeChunkOop, BRANCH [$, itsFineWhereItIs],       c1;
        L1  moveFromBigToSmall,                                          c2;
        CALL [addToFreeChunkList] {this call returns directly to splice}, c3;


splice:
        otLow  uPredecessor, L1  splicingBigFreeList,                    c1, at [moveFromBigToSmall, 10, addFreeChunkReturn];
        Noop,                                                            c2;
        Noop,                                                            c3;

        []  otLow and 3, ZeroBr,                                         c1;     {temp}
        temp2High  uRumRecordHigh, BRANCH [didHavePredecessor, $],       c2;
        temp2Low  uRumRecordLow, {no predecessor}                        c3;

        MAR  [temp2High, temp2Low + bigFreeListOffset], GOTO [linkNextChunk], c1;


didHavePredecessor:
        CALL [otMap2],                                                   c3;

        temp1Low  temp1Low + chunkLinkOffset,                            c1, at[splicingBigFreeList, 10, otMap2-return];
        Noop,                                                            c2;
        Noop,                                                            c3;

        MAR  [temp1High, temp1Low + 0],                                  c1;
linkNextChunk:
        MDR  uNextFreeChunk, LOOPHOLE [wok], CANCELBR [bigListWrapup, 0], c2;


itsFineWhereItIs:
        Noop,                                                            c2;
bigListWrapup:
        otLow  uNewObject, GOTO [allocate],                              c3;


outOfChunks:
        GOTO [massiveSenility],                                          c2;


outOfOops:
        Noop,                                                            c2;
massiveSenility:
        Noop,                                                            c3;

massiveSenility2:
        GOTO [bytecodeFailed],                                           c1;


massiveSenility4:
        CANCELBR [bytecodeFailed, OF],                                   c1;


requestedSizeTooBig:
        GOTO [massiveSenility2],                                         c3;



{Adjust the memory and oop levels and signal Mesa if either is below its alert level and Mesa has not yet been signalled.

(((wordLevel < wordAlertLevel) or: [oopLevel < oopAlertLevel]) and: [alreadyAlerted = 0]) ifTrue: [signalAlert  1. MesaIntRq]}

allocate:
        temp2High  uRumRecordHigh,                                       c1;
        temp2Low  uRumRecordLow,                                         c2;
        Noop,                                                            c3;

        MAR  [temp2High, temp2Low + oopLevelLowOffset],                  c1;
        CANCELBR [$, 0],                                                 c2;
        temp3Low  MD,                                                    c3;

        MAR  [temp2High, temp2Low + oopLevelLowOffset],                  c1;
        MDR  temp3Low  temp3Low - 1, CANCELBR [$, 0], LOOPHOLE [wok],    c2;
        Noop,                                                            c3;

lowOopTest:
        MAR  [temp2High, temp2Low + oopAlertLevelLowOffset],             c1;
        CANCELBR [$, 0],                                                 c2;
        Q  MD,                                                           c3;

        Noop,                                                            c1;
        Q  temp3Low - Q, CarryBr,                                        c2;
        BRANCH [$, decreaseWordLevel],                                   c3;

        MAR  [temp2High, temp2Low + alreadyAlertedOffset],               c1;
        CANCELBR [$, 0],                                                 c2;
        Q  MD,                                                           c3;
```

```
        [] Q, ZeroBr,                                                   c1;
        BRANCH [decreaseWordLevel2, $],                                 c2;
        Noop,                                                           c3;

        MAR [temp2High, temp2Low + signalAlertOffset],                  c1;
        MDR 1, CANCELBR [$, 0], LOOPHOLE [wok],                         c2;
        MesaIntRq {run mesa before next bytecode}, GOTO [decreaseWordLevel], c3;


decreaseWordLevel2:
        Noop,                                                           c3;

decreaseWordLevel:
        MAR [temp2High, temp2Low + wordLevelLowOffset],                 c1;
        temp3Low uRequestedSize, CANCELBR [$, 0],                       c2;
        Q MD,                                                           c3;

        MAR [temp2High, temp2Low + wordLevelLowOffset],                 c1;
        MDR temp3Low Q - temp3Low, CANCELBR [$, 0],
            LOOPHOLE [wok], CarryBr,                                    c2;
        BRANCH [$, lowMemoryTestHighGetData],                           c3;

wordLevelBorrow:
        MAR [temp2High, temp2Low + wordLevelHighOffset],                c1;
        CANCELBR [$, 0],                                                c2;
        Q MD,                                                           c3;

        MAR [temp2High, temp2Low + wordLevelHighOffset],                c1;
        MDR Q - 1, CANCELBR [$, 0], LOOPHOLE [wok],                     c2;
        temp1Low Q - 1, GOTO [lowMemoryTestHighHaveData],               c3;


lowMemoryTestHighGetData:
        MAR [temp2High, temp2Low + wordLevelHighOffset],                c1;
        CANCELBR [$, 0],                                                c2;
        temp1Low MD,                                                    c3;

lowMemoryTestHighHaveData:
        MAR [temp2High, temp2Low + wordAlertLevelHighOffset],           c1;
        CANCELBR [$, 0],                                                c2;
        Q MD,                                                           c3;

        Q Q - temp1Low, CarryBr,                                        c1;
        [] Q, ZeroBr, BRANCH [reallyAllocate1, $],                      c2;
        BRANCH [$, lowMemoryTestLow],                                   c3;

        MAR [temp2High, temp2Low + alreadyAlertedOffset],               c1;
        CANCELBR [$, 0],                                                c2;
        Q MD,                                                           c3;

        [] Q, ZeroBr,                                                   c1;
        BRANCH [lowMemoryTestLow2, $],                                  c2;
        Noop,                                                           c3;

        MAR [temp2High, temp2Low + signalAlertOffset],                  c1;
        MDR 1, CANCELBR [$, 0], LOOPHOLE [wok],                         c2;
        MesaIntRq {run mesa before next bytecode}, GOTO [reallyAllocate3], c3;


lowMemoryTestLow2:
        Noop,                                                           c3;

lowMemoryTestLow:
        MAR [temp2High, temp2Low + wordAlertLevelLowOffset],            c1;
        CANCELBR [$, 0],                                                c2;
        Q MD,                                                           c3;

        Noop,                                                           c1;
        Q temp3Low - Q, CarryBr,                                        c2;
        BRANCH [$, reallyAllocate3],                                    c3;

        MAR [temp2High, temp2Low + alreadyAlertedOffset],               c1;
        CANCELBR [$, 0],                                                c2;
        Q MD,                                                           c3;

        [] Q, ZeroBr,                                                   c1;
        BRANCH [reallyAllocate2, $],                                    c2;
        Noop,                                                           c3;

        MAR [temp2High, temp2Low + signalAlertOffset],                  c1;
        MDR 1, CANCELBR [$, 0], LOOPHOLE [wok],                         c2;
        MesaIntRq {run mesa before next bytecode}, GOTO [reallyAllocate3], c3;


reallyAllocate1:
        CANCELBR [reallyAllocate3, 1],                                  c3;


reallyAllocate2:
        Noop,                                                           c3;

reallyAllocate3:
        Noop,                                                           c1;
        temp1High uNewObjectHigh,                                       c2;
        temp1Low uNewObjectLow,                                         c3;

        temp3Low uRequestedSize,                                        c1;
```

```
        temp3Low  temp1Low + temp3Low, {end of the object + 1}                c2;
        temp1Low  temp1Low + deltaWordOffset,                                 c3;

{initialize the object header now}

        MAR  [temp1High, temp1Low + 0],                                       c1;
        MDR  uFieldType, {set delta word}                                     c2;
        temp1Low  temp1Low + offsetFromDeltaWordToSizeField,                  c3;

        MAR  [temp1High, temp1Low + 0],                                       c1;
        MDR  uRequestedSize,                                                  c2;
        temp1Low  temp1Low + offsetFromSizeFieldToClassField,                 c3;

        MAR  [temp1High, temp1Low + 0],                                       c1;
        MDR  uClassToInstantiate,                                             c2;
        {ok, object header is done, now zap the object body}
        temp1Low  temp1Low + offsetFromClassFieldToFirstField,                c3;

        temp2Low  uDefault,                                                   c1;
        Noop,                                                                 c2;
        []  temp1Low - temp3Low, ZeroBr,                                      c3;

initializeObjectBody:
        MAR  [temp1High, temp1Low + 0], BRANCH [$, zapped],                   c1;
        MDR  temp2Low, temp1Low  temp1Low + 1,                                c2;
        []  temp1Low - temp3Low, ZeroBr, GOTO [initializeObjectBody],         c3;


zapped:         {the object header and object body are completely initialized. now fix up the object table entry}
        Noop,                                                                 c2;
        temp3Low  temp3High {return link}, L2  creatingAnInstance,            c3;

        temp2High  newObjectHeader, CALL [getOtFlags],                        c1;

        temp1Low  temp1Low and 8F, CALL [putOtFlags],                         c1, at [newObjectHeader, 10, getFlagsReturn];

        temp2Low  temp3Low {save return link}, CALL [addToZeroCountTable],    c1, at [newObjectHeader, 10, putFlagsReturn];

        Noop,                                                                 c1, at [creatingAnInstance, 10,
        addToZeroCountTableReturn];
        L1  upClassAtInstantiation,                                           c2;
        otLow  uClassToInstantiate, XDisp, CALL [refi],                       c3;

        otLow  uNewObject,                                                    c1, at [upClassAtInstantiation, 10, refiReturn];
        Xbus  temp2Low LRot0, XDisp,                                          c2;
        RET [createInstance-return],                                          c3;



        { nextFreeChunk

        Return the next object on a free list.  Free objects are linked through their class fields.

                input:  otLow is the current object
                        otHigh is the high part of the object table base address
                        L1 is the return link for otMap2
                        L2 is the return link

                output: Q is the next object
                        temp1High/Low is the address of the current object

                smash:  }

nextFreeChunk:
        CALL [otMap2],                                                        c3;

        temp1Low  temp1Low + chunkLinkOffset,                                 c1, at [gettingNextFreeChunk, 10, otMap2-return];
        Noop,                                                                 c2;
        Noop,                                                                 c3;

        MAR  [temp1High, temp1Low + 0],                                       c1;
        temp1Low  temp1Low - chunkLinkOffset, L2Disp,                         c2;
        Q  MD, RET [nextFreeChunk-return],                                    c3;



        { addToFreeChunkList

        Add an object to the appropriate free list.

                input:  otLow is the object
                        temp1High/Low is the address of the object
                        temp3Low is the index of the list (size of the object for small objects)
                        uRumRecordHigh/Low is the Rum communications record base address
                        L1 is the return link

                output: temp2High/Low is the Rum communications record base address

                smash:  Q }

addToFreeChunkList:
        temp2High  uRumRecordHigh,                                            c1;
        temp2Low  uRumRecordLow,                                              c2;
```

```
        temp2Low  temp2Low + freeListsOffset,                                  c3;

        MAR  [temp2High, temp2Low + temp3Low],                                 c1;
        temp1Low  temp1Low + chunkLinkOffset, CANCELBR [$, 0],                 c2;
        Q  MD {current free list head},                                        c3;

        MAR  [temp1High, temp1Low + 0],                                        c1;
        MDR  Q, {link new object to old list head}                             c2;
        Noop,                                                                  c3;

        MAR  [temp2High, temp2Low + temp3Low],                                 c1;
        MDR  otLow {new list head}, CANCELBR [$, 0], LOOPHOLE [wok], L1Disp,   c2;
        temp1Low  temp1Low - chunkLinkOffset, RET [addFreeChunkReturn],        c3;



        { refi

        Increment the reference count of an object.  Nil, false, and true have permanently stuck counts, so skip them.  SmallIntegers
        don't have reference counts, so skip them too.

                input:  otLow is the object
                        otHigh is the high part of the object table base address
                        L1 is the return link
                        there is a pending XDisp to test for a smallInteger

                output:

                smash:  temp1Low, temp2High, Q }

refi:
        DISP4 [refiTable, 0C],                                                 c1;


refiOop01:
        []  falsePointer - otLow, CarryBr, GOTO [doRefi],                      c2, at [0D, 10, refiTable];


refiOop11:
        []  truePointer - otLow, CarryBr, GOTO [doRefi],                       c2, at [0F, 10, refiTable];


refiOop10:
        []  1, ZeroBr, GOTO [doRefi],                                          c2, at [0E, 10, refiTable];


refiSmallInteger00:
        L1Disp, GOTO [returnFromRefi],                                         c2, at [0C, 10, refiTable];


doRefi:
        BRANCH [$, skipRefi], {skip nil, false, true},                         c3;

        temp2High  doingNormalRefi, CALL [getOtFlags],                         c1;

        Q  temp1Low,                                                           c1, at [doingNormalRefi, 10, getFlagsReturn];
        temp1Low  refPlusOneRot8 {for incrementing ref count},                 c2;
        []  Q LRot0, XHDisp {first part of test for stuck ref count},          c3;

        temp1Low  temp1Low LRot8, BRANCH [$, maybeStuckRefi, 2],               c1;
        Q  Q + temp1Low {up ref count}, {sign is positive, thus not
                stuck and cannot become stuck}                                 c2;
        []  1, ZeroBr, {force next BRANCH}                                     c3;

updateOtRefi:
        temp1Low  Q, BRANCH [{CALL} putOtFlags, justGotStuckRefi],             c1;

        Noop,                                                                  c1, at [doingNormalRefi, 10, putFlagsReturn];
        L1Disp,                                                                c2;
returnFromRefi:
        RET [refiReturn],                                                      c3;


maybeStuckRefi: {sign is negative, can get stuck, may already be stuck}
        Q  Q + temp1Low {up ref count}, CarryBr {carry implies already stuck},  c2;
        []  Q + temp1Low, CarryBr {carry implies just got stuck},
                BRANCH [updateOtRefi, $],                                      c3;

stuckRefi:
        CANCELBR [$, 1],                                                       c1;
stuckRefiReturn:
        L1Disp, GOTO [returnFromRefi],                                         c2;


justGotStuckRefi: {Loom: need to call Loom here for newly stuck ref count}
        Noop,                                                                  c2;
        Noop,                                                                  c3;

        CALL [putOtFlags],                                                     c1;


skipRefi:
        GOTO [stuckRefiReturn],                                                c1;
```

```
        { refd

        Decrement the reference count of an object   Nil, false, and true have permanently stuck counts, so skip them.  SmallIntegers
        don't have reference counts, so skip them too.

                input:  otLow is the object
                        otHigh is the high part of the object table base address
                        L1 is the return link
                        there is a pending XDisp to test for a smallInteger

                output:

                smash:  temp1High/Low, temp2High, temp3High/Low, Q, L2 }

refd:
        temp3Low  refMinusOneRot8, DISP4 [refdTable, 0C],                       c1;


refdOop01:
        []  falsePointer - otLow, CarryBr, GOTO [doRefd],                       c2, at [0D, 10, refdTable];


refdOop11:
        []  truePointer - otLow, CarryBr, GOTO [doRefd],                        c2, at [0F, 10, refdTable];


refdOop10:
        []  1, ZeroBr, GOTO [doRefd],                                           c2, at [0E, 10, refdTable];


refdSmallInteger00:
        L1Disp, GOTO [returnFromRefd],                                          c2, at [0C, 10, refdTable];


doRefd:
        temp3Low  temp3Low LRot8,
                BRANCH [$, skipRefd], {skip nil, false, true},                  c3;

gotRefCount:
        temp2High  doingNormalRefd, CALL [getOtFlags],                          c1;

        []  temp1Low LRot0, XHDisp {first part of stuck ref count test},        c1, at [doingNormalRefd, 10, getFlagsReturn];
        Q  ~temp3Low, BRANCH [$, negativeRefCount, 2],                          c2;
positiveRefCount:       {not stuck but could go to zero}
        temp1Low  temp1Low + temp3Low {subtract 1}, CarryBr
                {no carry implies already zero, an error}, L2  doingRefd,       c3;

updateOtRefd:
        BRANCH [triedToRefdZeroCountObject, $],                                 c1;
        Noop,                                                                   c2;
        Noop,                                                                   c3;

        CALL [putOtFlags],                                                      c1;

        []  temp1Low + temp3Low {subtract again}, CarryBr {no carry implies
                just went to zero},                                             c1, at [doingNormalRefd, 10, putFlagsReturn];
        L1Disp, BRANCH [belongsInZct, $],                                       c2;
returnFromRefd:
        RET [refdReturn],                                                       c3;


negativeRefCount:       {could be stuck but cannot go to zero}
        Q  Q + 1, {refPlusOneRot8 LRot8}                                        c3;

        []  temp1Low + Q, CarryBr {carry implies stuck ref count},              c1;
        temp1Low  temp1Low + temp3Low {subtract one from ref count},
                BRANCH [$, stuckRefd],                                          c2;
        []  0, ZeroBr {force BRANCH}, GOTO [updateOtRefd],                      c3;


belongsInZct:
        CANCELBR [$, 0F],                                                       c3;

        CALL [addToZeroCountTable],                                             c1;

skipRefd:
        GOTO [refdSmallInteger00],                                              c1, at [doingRefd, 10, addToZeroCountTableReturn];


stuckRefd:
        GOTO [skipRefd],                                                        c3;


triedToRefdZeroCountObject:
        Q  refdZero, CANCELBR [bailout3, 1], {error}                            c2;




        { refd2

        Identical to refd except for return links. }
```

```
refd2:
        temp3Low  refMinusOneRot8, DISP4 [refd2Table, 0C],                    c1;

refd20op01:
        []  falsePointer - otLow, CarryBr, GOTO [doRefd2],                    c2, at [0D, 10, refd2Table];

refd20op11:
        []  truePointer - otLow, CarryBr, GOTO [doRefd2],                     c2, at [0F, 10, refd2Table];

refd20op10:
        []  1, ZeroBr, GOTO [doRefd2],                                        c2, at [0E, 10, refd2Table];

refd2SmallInteger00:
        L1Disp, GOTO [returnFromRefd2],                                       c2, at [0C, 10, refd2Table];

doRefd2:
        temp3Low  temp3Low LRot8,
                BRANCH [$, skipRefd2], {skip nil, false, true},               c3;

getRefCount2:
        temp2High  doingNormalRefd2, CALL [getOtFlags],                       c1;

        []  temp1Low LRot0, XHDisp {first part of stuck ref count test},      c1, at [doingNormalRefd2, 10, getFlagsReturn];
        Q  ~temp3Low, BRANCH [$, negativeRefCount2, 2],                       c2;
positiveRefCount2:      {not stuck but could go to zero}
        temp1Low  temp1Low + temp3Low {subtract 1}, CarryBr
                {no carry implies already zero, an error}, L2  doingRefd2,    c3;

updateOtRefd2:
        BRANCH [triedToRefd2ZeroCountObject, $],                              c1;
        Noop,                                                                 c2;
        Noop,                                                                 c3;

        CALL [putOtFlags],                                                    c1;

        []  temp1Low + temp3Low {subtract again}, CarryBr {no carry implies
                just went to zero},                                           c1, at [doingNormalRefd2, 10, putFlagsReturn];
        L1Disp, BRANCH [belongsInZct2, $],                                    c2;
returnFromRefd2:
        RET [refd2Return],                                                    c3;

negativeRefCount2:      {could be stuck but cannot go to zero}
        Q  Q + 1, {refPlusOneRot8 LRot8}                                      c3;

        []  temp1Low + Q, CarryBr {carry implies stuck ref count},            c1;
        temp1Low  temp1Low + temp3Low {subtract one from ref count},
                BRANCH [$, stuckRefd2],                                       c2;
        []  0, ZeroBr {force BRANCH}, GOTO [updateOtRefd2],                   c3;

belongsInZct2:
        CANCELBR [$, 0F],                                                     c3;

        CALL [addToZeroCountTable],                                           c1;

skipRefd2:
        GOTO [refd2SmallInteger00],                                           c1, at [doingRefd2, 10, addToZeroCountTableReturn];

stuckRefd2:
        GOTO [skipRefd2],                                                     c3;

triedToRefd2ZeroCountObject:
        Q  refd2Zero, CANCELBR [bailout3, 1], {error}                         c2;

dummy2: GOTO [bailout2],                                                      c1, at [0, 10, refd2Return];


        { addToZeroCountTable           Loom: Loom may want to get involved here--but I don't think so

        Add an object to the zero count table.  Objects get added to the zero count table when their reference counts go to zero.
        Eventually the zero count table fills up and all the objects in it are recursively freed.  In the meantime, we turn on the inZct
        bit in the OT and write the new OT entry.  Then we look at the object's old inZct bit and add the object to the zct if it's not
        already there.  If this object fills the table, we'll have to stabilize and free soon.  There's room for some overflow so we can
        finish this bytecode.

                input:  otLow is the object to put into the zct
                        otHigh is the high part of the object table base address
                        L2 is the return link

                output: uTimeToStabilize is -1 if the zct is full

                smash:  temp1High/Low, temp2High, temp3High/Low, Q }

addToZeroCountTable:
        temp3Low  inZctRot8,                                                  c2;
```

```
        temp3Low  temp3Low LRot8,                                          c3;

        temp2High  addingToZct, CALL [getOtFlags],                         c1;

        [] temp1Low and temp3Low, ZeroBr,                                  c1, at [addingToZct, 10, getFlagsReturn];
        temp1Low  temp1Low or temp3Low, L2Disp, BRANCH [alreadyInZct, $],  c2;
        Q  uRumRecordHigh, CANCELBR [$, OF]                                c3;

        temp1High  Q LRot0, CALL [putOtFlags],                             c1;

        temp1Low  uRumRecordLow,                                           c1, at [addingToZct, 10, putFlagsReturn];
        Noop,                                                              c2;
        Noop,                                                              c3;

        MAR  [temp1High, temp1Low + zctIndexOffset],                       c1;
        CANCELBR [$, 0],                                                   c2;
        Q  MD, {read current index}                                        c3;

        MAR  [temp1High, temp1Low + zctLowOffset],                         c1;
        CANCELBR [$, 0],                                                   c2;
        temp3Low  MD, {get zct low address}                                c3;

        MAR  [temp1High, temp1Low + zctHighOffset],                        c1;
        temp3Low  temp3Low + Q, CANCELBR [$, 0],                           c2;
        temp3High  MD, {get zct high address}                              c3;

                { Note: The Molasses zeroCountTable is one-relative, not zero-relative. So, while Molasses bumps the index before putting
                something in the zct, we put it in, then bump. }

        MAR  [temp3High, temp3Low + 0],                                    c1;
        MDR  otLow, {add the object}                                       c2;
        Noop,                                                              c3;

        MAR  [temp1High, temp1Low + zctIndexOffset],                       c1;
        MDR  Q  Q + 1, CANCELBR [$, 0], LOOPHOLE [wok], {write new index}  c2;
        Noop,                                                              c3;

        MAR  [temp1High, temp1Low + stabilizationLimitOffset],             c1;
        CANCELBR [$, 0],                                                   c2;
        temp3Low  MD, {get the zct stabilization limit}                    c3;

        Noop,                                                              c1;
        [] temp3Low - Q, NegBr,                                            c2;
        BRANCH [zctIndexOk, $], {need to stabilize if limit exceeded}      c3;

        MAR  [temp1High, temp1Low + stabilizationFlagOffset],              c1;
        MDR  needToStabilize, CANCELBR[$, 0], LOOPHOLE[wok],               c2;
        uTimeToStabilize  ~stackLow xor stackLow {-1},                     c3;

zctIndexOk:
        Noop,                                                              c1;
        L2Disp,                                                            c2;
alreadyInZct:
        RET [addToZeroCountTableReturn],                                   c3;



        { makeVolatile

                input:

                output:

                smash:  temp2High/Low, uZctBaseHigh }

{upon entry, otLow is the oop to make volatile,  uMakeVolatileLinkage is the return linkage register: if it is odd, each object referred
to by the object will be refd'd; it it is even, the object is marked volatile, but no refding occurs. smashes temp3Low, Q, L1, L2.
leaves base of object in uMakeVolatileHigh/Low, and in temp1High/Low. leaves uLastPointer set up}

{see if we're trying to make nil volatile -- this happens when the leaf context oop is nil.  check should probably be moved to the place
where volatilization is done after stabilization...}

makeVolatile:
        [] otLow xor nilPointer, ZeroBr,                                   c2;
        BRANCH [$, nilMakeVolatile],                                       c3;

        uMakeVolatileOop  otLow,                                           c1;
        L1  makingVolatile,                                                c2;
        CALL [otMap] {get address of base of object},                      c3;

        Q  temp1High {save object base}                                    c1, at[makingVolatile, 10, otMap-return];
        uMakeVolatileHigh  Q,                                              c2;
        uMakeVolatileLow  temp1Low,                                        c3;

        temp1Low  temp1Low + deltaWordOffset,                              c1;
        Noop,                                                              c2;
        Noop,                                                              c3;

        MAR  [temp1High, temp1Low + 0],                                    c1;
        Noop,                                                              c2;
        Q  MD {delta word},                                                c3;

        [] Q and volatileBit, ZeroBr {already volatile?},                  c1;
```

```
        Ybus  uMakeVolatileLinkage, XDisp,
              BRANCH [$, doMakeVolatile, 0E],                          c2;
        temp1Low  temp1Low - deltaWordOffset, RET [makeVolatile-return],  c3;


doMakeVolatile:
        Q  Q or volatileBit, CANCELBR [$, 0f],                         c3;

        MAR  [temp1High, temp1Low + 0],                                c1;
        MDR  Q {delta word with volatile bit set},                     c2;
        Ybus  uMakeVolatileLinkage, XDisp
              {should we refd the referents or not?},                  c3;

        temp1Low  temp1Low - deltaWordOffset,
              BRANCH [returnFromMakeVolatile, $, 0E],                  c1;
        temp1Low  temp1Low + sizeFieldOffset,   {refd fields}          c2;
        Q  temp1High,                                                  c3;

        MAR  [temp1High, temp1Low + 0],                                c1;
        temp1Low  temp1Low - sizeFieldOffset,                          c2;
        temp3Low  MD {size field},                                     c3;

        temp3Low  temp3Low + temp1Low,                                 c1;
        temp3Low  temp3Low - 1 {low 16 bits of last pointer
              of context object},                                      c2;
        uLastPointer  temp3Low,                                        c3;

{now, sweep the object decrementing reference counts of all pointer fields}

        temp2Low  temp1Low + firstPointerFieldOfObject,                c1;
        uZctBaseHigh  Q,                                               c2;
        temp2High  Q LRot0,                                            c3;

makeVolatileLoop:
        MAR  [temp2High, temp2Low + 0], L1  inMakeVolatile,            c1;
        temp2Low  temp2Low + 1,                                        c2;
        otLow  MD, XDisp, CALL [refd],                                 c3;

        temp3Low  uLastPointer,                                        c1, at [inMakeVolatile, 10, refdReturn];
        []   temp2Low - temp3Low, ZeroBr,                              c2;
        temp2High  uZctBaseHigh, BRANCH [makeVolatileLoop, $],         c3;

        otLow  uMakeVolatileOop, L2  inMakeVolatile,                   c1;
        Noop,                                                          c2;
        Noop,                                                          c3;

        CALL [addToZeroCountTable],                                    c1;

        temp1High  uMakeVolatileHigh,                                  c1, at [inMakeVolatile, 10, addToZeroCountTableReturn];
        temp1Low  uMakeVolatileLow,                                    c2;
        temp2High  uZctBaseHigh,                                       c3;

        Noop,                                                          c1;
returnFromMakeVolatile:
        Ybus  uMakeVolatileLinkage, XDisp,                             c2;
returningFromMakeVolatile:
        RET [makeVolatile-return],                                     c3;


nilMakeVolatile:
        GOTO [returnFromMakeVolatile],                                 c1;




        { lastPointerOf

              input:

              output:

              smash:  Q }

{upon entry,  temp1High/Low must point at the base of the object of interest, Q must be the delta word of the object, L2 is the return
linkage register. returns the low 16 bits of the ADDRESS of the last pointer in uLastPointer and in temp3Low. smashes Q}

lastPointerOf:
        []  Q and 1 {pointer bit}, ZeroBr,                             c1;
        Q  classCompiledMethodOop,
              BRANCH [doesHavePointers, doesNotHavePointers],          c2;


{is pure pointer object -- last pointer is size of object}

doesHavePointers:
        temp1Low  temp1Low + sizeFieldOffset,                          c3;

        MAR  [temp1High, temp1Low + 0] {start read of length field},   c1;
        temp1Low  temp1Low - sizeFieldOffset {again point at base of object},  c2;
        temp3Low  MD, GOTO [returnFromLastPointerOf],                  c3;


        {no pointers, might be compiledMethod -- need to check class}
doesNotHavePointers:
```

```
        temp1Low  temp1Low + classFieldOffset,                                  c3;

        MAR  [temp1High, temp1Low + 0],                                        c1;
        temp1Low  temp1Low - classFieldOffset {again point at base of object}, c2;
        temp3Low  MD {the class of the object},                                c3;

        []  temp3Low xor Q {compiledMethodClass oop}, ZeroBr,                  c1;
        BRANCH [$, isCompiledMethod],                                          c2;
        temp3Low  objectHeaderSize, GOTO [returnFromLastPointerOf],            c3;


        {need to get number of literals from the method header}
isCompiledMethod:
        temp1Low  temp1Low + objectHeaderSize {point at method header},        c3;

        MAR  [temp1High, temp1Low + 0],                                        c1;
        temp1Low  temp1Low - objectHeaderSize, {again point at base of object} c2;
        temp3Low  MD {the method header},                                      c3;

        temp3Low  (RShift1 temp3Low and 0FF) {get literal count
              of compiledMethod}, SE  0,                                       c1;
        temp3Low  RShift1 temp3Low, SE  0,                                     c2;
        Noop,                                                                  c3;

        Noop,                                                                  c1;
        temp3Low  temp3Low + literalStart,                                     c2;
        temp3Low  temp3Low + objectHeaderSize,                                 c3;

returnFromLastPointerOf:
        temp3Low  temp3Low - 1,                                                c1;
        temp3Low  temp3Low + temp1Low, L2Disp,                                 c2;
        uLastPointer  temp3Low, RET [lastPointerOf-return],                    c3;



        { stabilize


                input:

                output:

                smash:   }

{Test the memory and oop levels and reset alreadyAlerted if both are above their alert levels.

((wordLevel >= wordAlertLevel) and: [oopLevel >= oopAlertLevel]) ifTrue: [alreadyAlerted  0]}

{linkage register is L0 -- runs only between bytecodes}

stabilize:
        temp1High  uRumRecordHigh,                                             c1;
        temp1Low   uRumRecordLow,                                              c2;
        uTimeToStabilize  0,                                                   c3;

testOopLevel:
        MAR  [temp1High, temp1Low + oopLevelLowOffset],                        c1;
        CANCELBR [$, 0],                                                       c2;
        temp3Low  MD,                                                          c3;

        MAR  [temp1High, temp1Low + oopAlertLevelLowOffset],                   c1;
        CANCELBR [$, 0],                                                       c2;
        Q  MD,                                                                 c3;

        Q  temp3Low - Q, CarryBr,                                              c1;
        BRANCH [stabilize2, $],                                                c2;
        Noop,                                                                  c3;

testWordLevelHigh:
        MAR  [temp1High, temp1Low + wordLevelHighOffset],                      c1;
        CANCELBR [$, 0],                                                       c2;
        temp3Low  MD,                                                          c3;

        MAR  [temp1High, temp1Low + wordAlertLevelHighOffset],                 c1;
        CANCELBR [$, 0],                                                       c2;
        Q  MD,                                                                 c3;

        Q  temp3Low - Q, CarryBr,                                              c1;
        []  Q, ZeroBr, BRANCH [stabilize1, $],                                 c2;
        BRANCH [resetAlreadyAlerted, $],                                       c3;

testWordLevelLow:
        MAR  [temp1High, temp1Low + wordLevelLowOffset],                       c1;
        CANCELBR [$, 0],                                                       c2;
        temp3Low  MD,                                                          c3;

        MAR  [temp1High, temp1Low + wordAlertLevelLowOffset],                  c1;
        CANCELBR [$, 0],                                                       c2;
        Q  MD,                                                                 c3;

        Q  temp3Low - Q, CarryBr,                                              c1;
        BRANCH [stabilize3, $],                                                c2;
        Noop,                                                                  c3;
```

```
resetAlreadyAlerted:
        MAR  [temp1High, temp1Low + alreadyAlertedOffset],                    c1;
        MDR  0, CANCELBR [$, 2], LOOPHOLE [wok],                              c2;
        GOTO [reallyStabilize],                                               c3;


stabilize1:
        CANCELBR [reallyStabilize, 1],                                        c3;


stabilize2:
        GOTO [reallyStabilize],                                               c3;


stabilize3:
        GOTO [reallyStabilize],                                               c3;


reallyStabilize:
        MAR  [temp1High, temp1Low + zctLowOffset],                            c1;
        CANCELBR [$, 0],                                                      c2;
        temp2Low  MD,   {get address of the zct}                              c3;

        MAR  [temp1High, temp1Low + zctHighOffset],                           c1;
        uZctBaseLow  temp2Low, CANCELBR [$, 0],                               c2;
        Q  temp2High  MD,                                                     c3;

        uZctBaseHigh  Q,                                                      c1;
        Noop,                                                                 c2;
        Noop,                                                                 c3;

        {get, then smash the zct index from the Rum record}
        MAR  [temp1High, temp1Low + zctIndexOffset],                          c1;
        MDR  0, CANCELBR [$, 0], LOOPHOLE [wok],                              c2;
        temp3Low  MD,                                                         c3;

        {reset the stabilization flag}
        MAR  [temp1High, temp1Low +  stabilizationFlagOffset],                c1;
        MDR  0, CANCELBR [$, 0], LOOPHOLE [wok],                              c2;
        temp3Low  temp3Low + temp2Low {yields low 16 bits of one word past
                the last valid oop in the zct},                               c3;

        uZctLimit  temp3Low,                                                  c1;
        uQueueHead  0,  {should be nilPointer}                                c2;
        uCurrentObject  0,      {should be nilPointer}                        c3;

{sweep the zct. for each oop marked (in its ot entry) as volatile, reset the isVolatile bit, and increase the reference counts of all of
its referents. recall that the zct index is one greater than the number of valid entries in the zct}

stabilizationLoop:
        []  temp2Low xor uZctLimit, ZeroBr {are we there yet?},               c1;
        temp3Low  OFF, BRANCH [$, countsAreNowCorrect],                       c2;
        temp3Low  temp3Low LRot8,                                             c3;

        MAR  [temp2High, temp2Low + 0],                                       c1;
        temp2Low  temp2Low + 1, L1  stabilizing,                              c2;
        otLow  MD, CALL [otMap2] {so we can get its delta word},              c3;

        temp1Low  temp1Low + deltaWordOffset,                                 c1, at [stabilizing, 10, otMap2-return];
        temp3Low  temp3Low or 0FB {yields FFFB, for turning off
                the isVolatile bit},                                          c2;
        Noop,                                                                 c3;

        MAR  [temp1High, temp1Low + 0] {read delta word}                      c1;
        Noop,                                                                 c2;
        Q  MD, XDisp {to test isVolatile bit},                                c3;

        MAR  [temp1High, temp1Low + 0], BRANCH [oopIsNotVolatile, $, 0B],     c1;
        Q  MDR  Q and temp3Low {not volatile anymore!},
                L2  stabilizingContext,                                       c2;
        temp1Low  temp1Low - deltaWordOffset, CALL [lastPointerOf],           c3;

        {need to move the temp1 regs into temp3 to keep refi from smashing them...}
        Q  temp1High,                                                         c1, at [stabilizingContext, 10, lastPointerOf-return];
        temp3High  Q LRot0,                                                   c2;
        temp3Low  temp1Low + classFieldOffset,                                c3;

        {sweep over the volatile object, upping the reference counts of its referents}

upReferents:
        MAR  [temp3High, temp3Low + 0], L1  correcting,                       c1;
        Noop,                                                                 c2;
        otLow  MD, XDisp, CALL [refi],                                        c3;

        []  temp3Low xor uLastPointer, ZeroBr,                                c1, at [correcting, 10, refiReturn];
        temp3Low  temp3Low + 1, BRANCH [$, thisOneIsStable],                  c2;
        GOTO [upReferents],                                                   c3;


oopIsNotVolatile:
        Noop,                                                                 c2;
thisOneIsStable:
        temp2High  uZctBaseHigh, GOTO [stabilizationLoop],                    c3;


{at this point, all contexts have been stabilized, and all reference counts are correct.  sweep over the zct again: any object in the zct
whose reference count is zero is garbage!}
```

```
{temp2High is still valid despite the CALLs. restore temp2Low}

countsAreNowCorrect:
        temp2Low   uZctBaseLow,                                                 c3;

sweepAndDeallocateLoop:
        [] temp2Low xor uZctLimit, ZeroBr {are we there yet?},                  c1;
        temp2High  uZctBaseHigh, BRANCH [$, returnFromStabilize],               c2;
        temp3Low  inZctRot8,                                                    c3;

        MAR  [temp2High, temp2Low + 0] {get oop from zct}                       c1;
        temp3Low  ~temp3Low LRot8,                                              c2;
        otLow  MD,                                                              c3;

        temp2High  deallocating, CALL [getOtFlags],                             c1;

        temp1Low  temp1Low and temp3Low {clear inZct}, CALL [putOtFlags],       c1, at [deallocating, 10, getFlagsReturn];

        temp3Low  refCountRot8,                                                 c1, at [deallocating, 10, putFlagsReturn];
        temp3Low  temp3Low LRot8,                                               c2;
        Noop,                                                                   c3;

        Noop,                                                                   c1;
        [] temp1Low and temp3Low, ZeroBr,                                       c2;
        temp2Low  temp2Low + 1, BRANCH [sweepAndDeallocateLoop, $],             c3;

needToDeallocate:
        {save our current state}
        Q  temp2High,                                                           c1;
        uZctSweepHigh  Q, L3  fromStabilize,                                    c2;
        uZctSweepLow  temp2Low, CALL [deallocate],                              c3;

        {recover our state}
        temp2High  uZctSweepHigh,                                               c1, at [fromStabilize, 10, deallocate-return];
        temp2Low  uZctSweepLow,                                                 c2;
        GOTO [sweepAndDeallocateLoop],                                          c3;


returnFromStabilize:
        Noop,                                                                   c3;

        Noop,                                                                   c1;
        L0Disp,                                                                 c2;
        RET [stabilize-return],                                                 c3;




        { deallocate


                input:  otLow is the object to deallocate
                        L3 is the return link

                output:

                smash:  Q, uClass, L1, L2 }

{otLow is the oop to deallocate -- it has already been determined that
        its reference count is 0. L3 is the return linkage register}
deallocate:
        L2  startingDeallocate,                                                 c1;
        Noop,                                                                   c2;
        [] otLow LRot0, XDisp, CALL [getClass],                                 c3;

        temp1Low  temp1Low + deltaWordOffset,                                   c1, at [startingDeallocate, 10, getClass-return];
        Q  classCompiledMethodOop,                                              c2;
        [] temp3Low xor Q, ZeroBr,                                              c3;

        {get delta word to see if object has pointers}
        MAR  [temp1High, temp1Low + 0],
                BRANCH [$, deallocatingACompiledMethod],                        c1;
        Noop,                                                                   c2;
        Q  MD, XLDisp,                                                          c3;

        BRANCH [deallocateWithNoPointers, deallocateWithPointers, 2],           c1;


deallocateWithNoPointers:
        temp1Low  temp1Low - deltaWordOffset, L1  freeNonPointerObject,         c2;
        uClass  temp3Low, CALL [adjustLevelsAndReturnToPool],                   c3;

        temp3Low  uClass, GOTO [nowDoObjectsClass],                             c1, at [freeNonPointerObject, 10, addFreeChunkReturn];


deallocatingACompiledMethod:
        GOTO [deallocateWithPointersA],                                         c2;


deallocateWithPointers:
        Noop,                                                                   c2;
deallocateWithPointersA:
        temp1Low  temp1Low + offsetFromDeltaWordToClassField,                   c3;
```

```
        {enqueue this object for deallocation}
        MAR  [temp1High, temp1Low + 0],                                  c1;
        MDR  uQueueHead,                                                 c2;
        uQueueHead  otLow,                                               c3;

        Noop,                                                            c1;
nowDoObjectsClass:
        Noop,                                                            c2;
        otLow  temp3Low LRot0, XDisp, GOTO [specialRefd],                c3;



{if there is a current object, continue with it.  if not, if there is
        a queued object, start it. otherwise we are done}

more:
        otLow  uCurrentObject, ZeroBr,                                   c1, at [nowDoneWithObject, 10, addFreeChunkReturn];
        BRANCH [continueWithCurrentObject, $],                           c2;
        otLow  uQueueHead, ZeroBr,                                       c3;

        BRANCH [startWithQueueHead, $],                                  c1;
        L3Disp,                                                          c2;
        RET [deallocate-return], {all recursive freeing is now complete} c3;


startWithQueueHead:
        uCurrentObject  otLow,  L1  sweepingObject,                      c2;
        CALL [otMap2],                                                   c3;

        temp1Low  temp1Low + deltaWordOffset,                            c1, at [sweepingObject, 10, otMap2-return];
        Q  temp1High,                                                    c2;
        uSoFarHigh  Q,                                                   c3;

        MAR  [temp1High, temp1Low + 0],                                  c1;
        temp1Low  temp1Low - deltaWordOffset, L2  getObjectEndForFreeing, c2;
        Q  MD, XDisp, {test pointers bit}                                c3;

        BRANCH [doingACompiledMethod, notDoingACompiledMethod, 0E],      c1;


        {both isCompiledMethod and doesHavePointers live in the lastPointerOf routine}
doingACompiledMethod:
        CALL [isCompiledMethod],                                         c2;


notDoingACompiledMethod:
        CALL [doesHavePointers],                                         c2;

        uCurrentObjectBaseLow  temp1Low,                                 c1, at [getObjectEndForFreeing, 10,
        lastPointerOf-return];
        temp1Low  temp1Low + classFieldOffset,                           c2;
        Noop,                                                            c3;

        MAR  [temp1High, temp1Low + 0],                                  c1;
        Noop,                                                            c2;
        temp2Low  MD {link to next object on queue},                     c3;

        uQueueHead  temp2Low,                                            c1;
        Noop,                                                            c2;
        GOTO [areWeThereYet],                                            c3;


doAnotherField:
        MAR  [temp1High, temp1Low + 0],                                  c1;
        Noop,                                                            c2;
        otLow  MD, XDisp, GOTO [specialRefd],                            c3;


continueWithCurrentObject:
        temp1Low  uSoFar,                                                c3;

        temp1High  uSoFarHigh,                                           c1;
        temp3Low  uLastPointer,                                          c2;
        Noop,                                                            c3;

areWeThereYet:
        []  temp1Low xor temp3Low, ZeroBr,                               c1;
        temp1Low  temp1Low + 1, BRANCH [$, doneWithObject],              c2;
        uSoFar  temp1Low, GOTO [doAnotherField],                         c3;


doneWithObject:
        Noop,                                                            c3;

        otLow  uCurrentObject,                                           c1;
        temp1Low  uCurrentObjectBaseLow, L1  nowDoneWithObject,          c2;
        uCurrentObject  0 {should be nilPointer},
                CALL [adjustLevelsAndReturnToPool],                      c3;



        { specialRefd
```

```
        Decrement the reference count of an object.  Nil, false, and true have permanently stuck counts, so skip them.  SmallIntegers
        don't have reference counts, so skip them too.

                input:  otLow is the object
                        otHigh is the high part of the object table base address
                        there is a pending XDisp to test for a smallInteger

                output:

                smash:  temp1High/Low, temp2High, temp3High/Low, Q, L2 }

specialRefd:
        temp3Low  refMinusOneRot8, DISP4 [specialRefdTable, 0C],                c1;


specialRefdOop01:
        [] falsePointer - otLow, CarryBr  GOTO [specialDoRefd],                 c2, at [0D, 10, specialRefdTable];


specialRefdOop11:
        [] truePointer - otLow, CarryBr, GOTO [specialDoRefd],                  c2, at [0F, 10, specialRefdTable];


specialRefdOop10:
        [] 1, ZeroBr, GOTO [specialDoRefd],                                     c2, at [0E, 10, specialRefdTable];


specialRefdSmallInteger00:
        GOTO [specialStuckRefd],                                                c2, at [0C, 10, specialRefdTable];


specialDoRefd:
        temp3Low  temp3Low LRot8,
                BRANCH [$, skipSpecialRefd], {skip nil, false, true}            c3;

getSpecialRefCount:
        temp2High  doingSpecialRefd, CALL [getOtFlags],                         c1;

        [] temp1Low LRot0, XHDisp {first part of stuck ref count test},         c1, at [doingSpecialRefd, 10, getFlagsReturn];
        Q  ~temp3Low, BRANCH [$, negativeSpecialRefCount, 2],                   c2;
positiveSpecialRefCount:        {not stuck but could go to zero}
        temp1Low  temp1Low + temp3Low {subtract 1}, CarryBr {no carry implies
                already zero, an error},                                        c3;

updateOtSpecialRefd:
        BRANCH [triedToSpecialRefdZeroCountObject, $],                          c1;
        Noop,                                                                   c2;
        Noop,                                                                   c3;

        CALL [putOtFlags],                                                      c1;

        [] temp1Low + temp3Low {subtract again}, CarryBr {no carry implies
                just went to zero},                                             c1, at [doingSpecialRefd, 10, putFlagsReturn];
        temp1Low  temp1Low LRot8, BRANCH [specialNeedsDeallocation, $],         c2;
        GOTO [more],                                                            c3;


negativeSpecialRefCount:        {could be stuck but cannot go to zero}
        Q  Q + 1, {refPlusOneRot8 LRot8}                                        c3;

        [] temp1Low + Q, CarryBr {carry implies stuck ref count},               c1;
        temp1Low  temp1Low + temp3Low {subtract one from ref count},
                BRANCH [$, specialStuckRefd],                                   c2;
        [] 0, ZeroBr {force BRANCH}, GOTO [updateOtSpecialRefd],                c3;


specialNeedsDeallocation:       {ref count just went to zero}
        [] temp1Low and inZctRot8, ZeroBr,                                      c3;

        BRANCH [specialAlreadyInZct, $], {deallocate now if not in zct}         c1;
        Noop,                                                                   c2;
        GOTO [deallocate],                                                      c3;

specialAlreadyInZct:
        Noop,                                                                   c2;
specialStuckRefd:
        GOTO [more],                                                            c3;


skipSpecialRefd:
        GOTO [specialAlreadyInZct],                                             c1;


triedToSpecialRefdZeroCountObject:
        Q  specialRefdZero, GOTO [bailout3], {error}                            c2;




adjustLevelsAndReturnToPool:
        temp1Low  temp1Low + sizeFieldOffset,                                   c1;
        temp2High  uRumRecordHigh,                                              c2;
        temp2Low  uRumRecordLow,                                                c3;

incrementOopLevel:
        MAR  [temp2High, temp2Low + oopLevelLowOffset],                         c1;
```

```
        CANCELBR [$, 0],                                                    c2;
        temp3Low  MD,                                                       c3;

        MAR  [temp2High, temp2Low + oopLevelLowOffset],                     c1;
        MDR  temp3Low  temp3Low + 1,
               CANCELBR [$, 0], LOOPHOLE [wok], CarryBr,                    c2;
        BRANCH [increaseWordLevelLow, impossibleOopLevel],                  c3;


impossibleOopLevel:
        Q  tooManyOops, GOTO [bailout2],                                    c1;


increaseWordLevelLow:
        MAR  [temp1High, temp1Low + 0],                                     c1;
        temp1Low  temp1Low - sizeFieldOffset,                               c2;
        temp3Low  MD,                                                       c3;

        MAR  [temp2High, temp2Low + wordLevelLowOffset],                    c1;
        CANCELBR [$, 0],                                                    c2;
        Q  MD,                                                              c3;

        MAR  [temp2High, temp2Low + wordLevelLowOffset],                    c1;
        MDR  Q + temp3Low, CANCELBR [$, 0], LOOPHOLE [wok], CarryBr,        c2;
        BRANCH [returnToPool, wordLevelCarry],                              c3;


wordLevelCarry:
        MAR  [temp2High, temp2Low + wordLevelHighOffset],                   c1;
        CANCELBR [$, 2],                                                    c2;
        temp3Low  MD,                                                       c3;

        MAR  [temp2High, temp2Low + wordLevelHighOffset],                   c1;
        MDR  temp3Low  temp3Low + 1, CANCELBR [$, 0], LOOPHOLE [wok],       c2;
        Noop,                                                               c3;

returnToPool:
        {build a mask to set all ref count bits on and to set purpose bits to free (11)}
        temp3Low  refCountRot8,                                             c1;
        temp3Low  temp3Low LRot8,                                           c2;
        temp3Low  temp3Low or freeOop,                                      c3;

        temp2High  returningToPool, temp2Low  temp1Low, CALL [getOtFlags],  c1;

        temp1Low  temp1Low or temp3Low, CALL [putOtFlags],                  c1, at [returningToPool, 10, getFlagsReturn];

{upon entry, otLow is the oop of the object to add to the free list,
        temp1High/temp2Low must be that object's base. calls addToFreeChunkList
        thus smashing temp2High/Low and Q. smashes temp3Low}

addToProperFreeChunkList:
        temp1Low  temp2Low + sizeFieldOffset,                               c1, at [returningToPool, 10, putFlagsReturn];
        Q  largestFreeChunkSize,                                            c2;
        Noop,                                                               c3;

        MAR  [temp1High, temp1Low + 0],                                     c1;
        temp1Low  temp1Low - sizeFieldOffset,                               c2;
        temp3Low  MD {object's size},                                       c3;

        []  temp3Low - Q, CarryBr,                                          c1;
        BRANCH [selectRegularList, selectBigFreeList],                      c2;


selectRegularList:
        GOTO [addToFreeChunkList],                                          c3;


selectBigFreeList:
        temp3Low  Q, GOTO [addToFreeChunkList],                             c3;




{ Edit history:

13-Jan-86 15:47:16      Trow.pa         tighten initializeObjectBody loop
30-Sep-85 17:21:46      Trow.pa         convert to stretch format }
```

```
{ Comb0C.mc
Combination rule subroutines for bitblt primitive for Rum, the Dandelion Smalltalk-80 microcoded virtual machine.
        by T Tokunaga, J Trow
        19-Jan-86 18:07:13
Copyright 1985, 1986 by Xerox Corporation.  All rights reserved. }



        MAR _ [destAddrHigh, destAddrLow + 0], GOTO [comb0E001],              c1, at [0, 10, comb0C];

        MAR _ [destAddrHigh, destAddrLow + 0], GOTO [comb0E0F1],              c1, at [0F, 10, comb0C];

{****************************** combinationRule = 1 ********************************}
{ - - sourceWord bitAnd: destinationWord - - - }
comb0C01:
        MAR _ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource],          c1, at [01, 10, comb0C];



{- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -}
{subroutine : getSource }
getSource:
        uBBTTemp _ sourceIndex and ~otLow, L3Disp,                            c2; { mergeMask bitInvert bitAnd:destBits at:destIndex + 1}
        temp2Low _ MD and temp2Low, RET [getSource-return],                   c3;{ sourceWord _ skewWord bitAnd: HalftoneWord }
{- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -}



        sourceIndex _ sourceIndex and temp2Low,                               c1, at [01, 10, getSource-return]; { mergeWord _
        sourceWord and destinationWord }
        otLow _ sourceIndex and otLow, CALL [store1],                         c2; { mergeMask bitAnd: mergeWord }



{- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -}
{Subroutine: store1}
store1:
        temp2Low _ otLow or uBBTTemp,                                         c3;

        MAR _ [destAddrHigh, destAddrLow + 0],                                c1;
        MDR _ temp2Low, temp3Low _ temp3Low + 0, ZeroBr,                      c2;
        sourceAddrLow _ sourceAddrLow + Q,
                BRANCH [contComb0C01, finishedComb0C01],                      c3;


contComb0C01:
        temp2Low _ uHalftoneWord, L3Disp,                                     c1;
        temp3Low _ temp3Low - 1, ZeroBr, RET [store1-return],                 c2;


finishedComb0C01:
        destAddrLow _ destAddrLow + Q,        GOTO [finishedComb0E00],        c1;
{- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -}



        destAddrLow _ destAddrLow + Q,
                BRANCH [noLastComb0C01, yesLastComb0C01],                     c3, at [01, 10, store1-return];


noLastComb0C01:
        MAR _ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource1],         c1;



{- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -}
getSource1:
        sourceAddrLow _ sourceAddrLow + Q,                                    c2;
        sourceIndex _ MD and temp2Low, L3Disp,                                c3; { sourceWord _ skewWord bitAnd: HalftoneWord }

        MAR _ [destAddrHigh, destAddrLow + 0], RET [getSource1-return],       c1;
{- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -}



        Noop,                                                                 c2, at [01, 10, getSource1-return];
        sourceIndex _ MD and sourceIndex, CALL [store],                       c3; { mergeWord _ sourceWord and destinationWord }
```

```
{------------------------------------------------------------------------------------------}
{subroutine : store }
store:
        MAR _ [destAddrHigh, destAddrLow + 0],        L3Disp,                       c1;
        MDR _ sourceIndex, temp3Low _ temp3Low - 1, ZeroBr, RET [store - return],   c2;
{------------------------------------------------------------------------------------------}


        destAddrLow _ destAddrLow + Q,
                BRANCH [noLastComb0C011, yesLastComb0C011],                 c3, at [01, 10, store - return];


noLastComb0C011:
        MAR _ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource1],       c1;


yesLastComb0C011:
        MAR _ [destAddrHigh, destAddrLow + 0],                              c1;
yesLastComb0C012:
        otLow _ uMask2,                                                     c2;
        sourceIndex _ MD, GOTO [comb0C01],                                  c3;


yesLastComb0C01:
        MAR _ [destAddrHigh, destAddrLow + 0], GOTO [yesLastComb0C012],     c1;



{****************************** combinationRule = 2 ********************************}
{ ---- sourceWord bitAnd: destinationWord bitInvert ---- }
comb0C02:
        MAR _ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource],        c1, at [02, 10, comb0C];

        temp2Low _ temp2Low and ~sourceIndex,                               c1, at [02, 10, getSource - return];{mergeWord_sourceWord
        and destinationWord bitInvert }
        otLow _ temp2Low and otLow, CALL [store1],                          c2; { mergeMask bitAnd: mergeWord }

        destAddrLow _ destAddrLow + Q,
                BRANCH [noLastComb0C02, yesLastComb0C02],                   c3, at [02, 10, store1 - return]; { restore halftoneWord }


noLastComb0C02:
        MAR _ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource1],       c1;

        Noop,                                                               c2, at [02, 10, getSource1 - return];
        sourceIndex _ ~MD and sourceIndex, CALL [store],                    c3; { sourceWord bitAnd: destination bitInvert }

        destAddrLow _ destAddrLow + Q,
                BRANCH [noLastComb0C021, yesLastComb0C021],                 c3, at [02, 10, store - return];


noLastComb0C021:
        MAR _ [destAddrHigh, destAddrLow + 0], CALL [getSource1],           c1;


yesLastComb0C021:
        MAR _ [destAddrHigh, destAddrLow + 0],                              c1;
yesLastComb0C022:
        otLow _ uMask2,                                                     c2;
        sourceIndex _ MD, GOTO [comb0C02],                                  c3;


yesLastComb0C02:
        MAR _ [destAddrHigh, destAddrLow + 0], GOTO [yesLastComb0C022],     c1;



{****************************** combinationRule = 3 ********************************}
{ ---- sourceWord ---- }
comb0C03:
        MAR _ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource],        c1, at [03, 10, comb0C];

        otLow _ temp2Low and otLow,                                         c1, at [03, 10, getSource - return]; { mergeMask bitAnd:
        mergeWord }
        CALL [store1],                                                      c2;

        destAddrLow _ destAddrLow + Q,
                BRANCH [noLastComb0C03, yesLastComb0C03],                   c3, at [03, 10, store1 - return];


noLastComb0C03:
        MAR _ [sourceAddrHigh, sourceAddrLow + 0],                          c1;
noLastComb0C032:
        sourceAddrLow _ sourceAddrLow + Q,                                  c2;
        sourceIndex _ MD and temp2Low, CALL [store],                        c3; { skewWord bitAnd: HalftoneWord }
```

```
        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0C031, yesLastComb0C031],                    c3, at [03, 10, store – return];

noLastComb0C031:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], GOTO [noLastComb0C032],    c1;

yesLastComb0C031:
        MAR __ [destAddrHigh, destAddrLow + 0],                                c1;
yesLastComb0C032:
        otLow __ uMask2,                                                       c2;
        sourceIndex __ MD, GOTO [comb0C03],                                    c3;

yesLastComb0C03:
        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [yesLastComb0C032],       c1; {***** Toku *****}


{****************************** combinationRule = 4 *********************************}
{ – – – – sourceWord bitInvert bitAnd: destinationWord – – – – }
comb0C04:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource],          c1, at [04, 10, comb0C];

        sourceIndex __ ~temp2Low and sourceIndex,                              c1, at [04, 10, getSource – return];{mergeWrod__sourceWord
        bitInvert bitAnd:destinationWord}
        otLow __ sourceIndex and otLow, CALL [store1],                         c2;

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0C04, yesLastComb0C04],                      c3, at [04, 10, store1 – return];

noLastComb0C04:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource1],         c1;

        sourceIndex __ ~sourceIndex,                                           c2, at [04, 10, getSource1 – return]; { sourceWord
        bitInvert }
        sourceIndex __ MD and sourceIndex, CALL [store],                       c3; { get DestinationWord and merging }

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0C041, yesLastComb0C041],                    c3, at [04, 10, store – return];

noLastComb0C041:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource1],         c1;

yesLastComb0C04:
        MAR __ [destAddrHigh, destAddrLow + 0],                                c1;
yesLastComb0C042:
        otLow __ uMask2,                                                       c2;
        sourceIndex __ MD, GOTO [comb0C04],                                    c3;

yesLastComb0C041:
        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [yesLastComb0C042],       c1;


{****************************** combinationRule = 5 *********************************}
comb0C05:
        GOTO [allFinished],                                                    c1, at [05, 10, comb0C];


{****************************** combinationRule = 6 *********************************}
{ – – – – sourceWord bitXor: destinationWord – – – – }
comb0C06:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource],          c1, at [06, 10, comb0C];

        sourceIndex __ sourceIndex xor temp2Low,                               c1, at [06, 10, getSource – return];
        otLow __ sourceIndex and otLow, CALL [store1],                         c2; { masking }

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0C06, yesLastComb0C06],                      c3, at [06 , 10, store1 – return];

noLastComb0C06:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource1],         c1;

        Noop,                                                                  c2, at [06, 10, getSource1 – return];
        sourceIndex __ MD xor sourceIndex, CALL [store],                       c3; { get destinationWord and merging }

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0C061, yesLastComb0C061],                    c3, at [06, 10, store – return];

noLastComb0C061:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource1],         c1;
```

```
yesLastComb0C061:
        MAR __ [destAddrHigh, destAddrLow + 0],                          c1;
yesLastComb0C062:
        otLow __ uMask2,                                                 c2;
        sourceIndex __ MD, GOTO [comb0C06],                              c3;

yesLastComb0C06:
        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [yesLastComb0C062], c1;



{****************************** combinationRule = 7 *********************************}
{ - - - - sourceWord bitOr: destinationWord - - - - }
comb0C07:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource],    c1, at [07, 10, comb0C];

        sourceIndex __ temp2Low or sourceIndex,                          c1, at [07, 10, getSource - return]; { mergeWord __
        sourceWord or destinationWord }
        otLow __ sourceIndex and otLow, CALL [store1],                   c2;

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0C07, yesLastComb0C07],                c3, at [07, 10, store1 - return];


noLastComb0C07:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource1],   c1;

        Noop,                                                            c2, at [07, 10, getSource1 - return];
        sourceIndex __ MD or sourceIndex, CALL [store],                  c3; { SWord or DWord }

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0C071, yesLastComb0C071],              c3, at [07, 10, store - return];


noLastComb0C071:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource1],   c1;


yesLastComb0C071:
        MAR __ [destAddrHigh, destAddrLow + 0],                          c1;
yesLastComb0C072:
        otLow __ uMask2,                                                 c2;
        sourceIndex __ MD, GOTO [comb0C07],                              c3;


yesLastComb0C07:
        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [yesLastComb0C072], c1;



{****************************** combinationRule = 8 *********************************}
{ - - - - sourceWord bitInvert bitAnd: destinationWord bitInvert - - - - }
comb0C08:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource],    c1, at [08, 10, comb0C];

        temp2Low __ ~temp2Low,                                           c1, at [08, 10, getSource - return]; { sourceword bitInvert
        }
        temp2Low __ ~sourceIndex and temp2Low,                           c2; { merging }
        otLow __ temp2Low and otLow,                                     c3;

        Noop,                                                            c1;
        CALL [store1],                                                   c2;

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0C08, yesLastComb0C08],                c3, at [08, 10, store1 - return];


noLastComb0C08:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource1],   c1;

        sourceIndex __ ~sourceIndex,                                     c2, at [08, 10, getSource1 - return];
        sourceIndex __ ~MD and sourceIndex, CALL [store],                c3; { ~sourceWord and ~destinationWord }

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0C081, yesLastComb0C081],              c3, at [08, 10, store - return];


noLastComb0C081:
        MAR __ [destAddrHigh, destAddrLow + 0], CALL [getSource1],       c1;


yesLastComb0C081:
        MAR __ [destAddrHigh, destAddrLow + 0],                          c1;
yesLastComb0C082:
        otLow __ uMask2,                                                 c2;
        sourceIndex __ MD, GOTO [comb0C08],                              c3;


yesLastComb0C08:
        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [yesLastComb0C082], c1;
```

```
{****************************** combinationRule = 9 ********************************}
{ - - - - sourceWord bitInvert bitXor: destinatioWord - - - - }
comb0C09:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource],                   c1, at [09, 10, comb0C];

        sourceIndex __ ~temp2Low xor sourceIndex,                                       c1, at [09, 10, getSource - return]; { merging }
        otLow __ sourceIndex and otLow, CALL [store1],                                  c2;

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0C09, yesLastComb0C09],                               c3, at [09, 10, store1 - return];


noLastComb0C09:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource1],                  c1;

        sourceIndex __ ~sourceIndex,                                                    c2, at [09, 10, getSource1 - return];
        sourceIndex __ MD xor sourceIndex, CALL [store],                                c3; { sourceWord bitInvert bitXor: destinationWord }

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0C091, yesLastComb0C091],                             c3, at [09, 10, store - return];


noLastComb0C091:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource1],                  c1;


yesLastComb0C091:
        MAR __ [destAddrHigh, destAddrLow + 0],                                         c1;
yesLastComb0C092:
        otLow __ uMask2,                                                                c2;
        sourceIndex __ MD, GOTO [comb0C09],                                             c3;


yesLastComb0C09:
        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [yesLastComb0C092],                c1;




{****************************** combinationRule = 0A ********************************}
{ - - - - destinationWord bitInvert - - - - }
comb0C0A:
        MAR __ [destAddrHigh, destAddrLow + 0],                                         c1, at [0A, 10, comb0C];
{ put: ((mergeMask bitAnd: mergeWord) bitOr:(mergeMaks bitrInvert bitAnd: (destBits at; destIndex = 1))}
        MDR __ sourceIndex xor otLow,                                                   c2;
        [] __ temp3Low, ZeroBr,                                                         c3;

        destAddrLow __ destAddrLow + Q, BRANCH [contComb0C0A, finishedComb0C0A],        c1;


contComb0C0A:
        temp3Low __ temp3Low - 1, ZeroBr,                                               c2;
        temp2Low __ uHalftoneWord, BRANCH [noLastComb0C0A, yesLastComb0C0A],            c3;


noLastComb0C0A:
        MAR __ [destAddrHigh, destAddrLow + 0],                                         c1;
noLastComb0C0A2:
        Noop,                                                                           c2;
        sourceIndex __ ~MD, CALL [store],                                               c3; { get ~destinationWord }

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0C0A1, yesLastComb0C0A1],                             c3, at [0A, 10, store - return];


noLastComb0C0A1:
        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [noLastComb0C0A2],                 c1;


yesLastComb0C0A1:
        MAR __ [destAddrHigh, destAddrLow + 0],                                         c1;
yesLastComb0C0A2:
        otLow __ uMask2,                                                                c2;
        sourceIndex __ MD, GOTO [comb0C0A],                                             c3;


yesLastComb0C0A:
        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [yesLastComb0C0A2],                c1;


finishedComb0C0A:
          temp3Low__ uH, GOTO [finishedComb0E001],                                      c2;




{****************************** combinationRule = 0B ********************************}
{ - - - - sourceWord bitOr: destinationWord bitInvert - - - - }
comb0C0B:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource],                   c1, at [0B, 10, comb0C];

        sourceIndex __ ~temp2Low and sourceIndex,                                       c1, at [0B, 10, getSource - return]; { merging }
```

```
        otLow __ ~sourceIndex and otLow,        CALL [store1],                          c2;

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0C0B, yesLastComb0C0B],                               c3, at [0B, 10, store1 - return];


noLastComb0C0B:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource1],                  c1;

        sourceIndex __ ~sourceIndex,                                                    c2, at [0B, 10, getSource1 - return];
        sourceIndex __ MD and sourceIndex,                                              c3;

        MAR __ [destAddrHigh, destAddrLow + 0],                                         c1;
        MDR __ ~sourceIndex,                                                            c2;
        destAddrLow __ destAddrLow + Q,                                                 c3;{ sourceWord bitInvert bitOr: destinationWord }

        Noop,                                                                           c1;
        temp3Low __ temp3Low - 1, ZeroBr,                                               c2;
        BRANCH [noLastComb0C0B1, yesLastComb0C0B1],                                     c3;


noLastComb0C0B1:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource1],                  c1;


yesLastComb0C0B1:
        MAR __ [destAddrHigh, destAddrLow + 0],                                         c1;
yesLastComb0C0B2:
        otLow __ uMask2,                                                                c2;
        sourceIndex __ MD, GOTO [comb0C0B],                                             c3;


yesLastComb0C0B:
        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [yesLastComb0C0B2],                c1;




{****************************** combinationRule = 0C ********************************}
{ - - - -  sourceWord bitInvert - - - - }
comb0C0C:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource],                   c1, at [0C, 10, comb0C];

        otLow __ ~temp2Low and otLow,                                                   c1, at [0C, 10, getSource - return];
        CALL [store1],                                                                  c2;
        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0C0C, yesLastComb0C0C],                               c3, at [0C, 10, store1 - return];


noLastComb0C0C:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource1],                  c1;
noLastComb0C0C2:
        sourceIndex __ ~sourceIndex,                                                    c2, at [0C, 10, getSource1 - return];
        CALL [store],                                                                   c3;

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0C0C1, yesLastComb0C0C1],                             c3, at [0C, 10, store - return];


noLastComb0C0C1:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource1],                  c1;


yesLastComb0C0C1:
        MAR __ [destAddrHigh, destAddrLow + 0],                                         c1;
yesLastComb0C0C2:
        otLow __ uMask2,                                                                c2;
        sourceIndex __ MD, GOTO [comb0C0C],                                             c3;


yesLastComb0C0C:
        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [yesLastComb0C0C2],                c1;




{****************************** combinationRule = 0D ********************************}
{ - - - - sourceWord bitInvert bitOr: destnationWord - - - - }
comb0C0D:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource],                   c1, at [0D, 10, comb0C];

        temp2Low __ temp2Low and ~sourceIndex,                                          c1, at [0D, 10, getSource - return]; { merging }
        otLow __ ~temp2Low and otLow, CALL [store1],                                    c2;
        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0C0D, yesLastComb0C0D],                               c3, at [0D, 10, store1 - return];


noLastComb0C0D:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource1],                  c1;
        sourceIndex __ ~sourceIndex,                                                    c2, at [0D, 10, getSource1 - return]; { sourceWord
        bitInvert }
        sourceIndex __ MD or sourceIndex, CALL [store],                                 c3; { ~SWord or DWord }

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0C0D1, yesLastComb0C0D1],                             c3, at [0D, 10, store - return];
```

```
noLastComb0C0D1:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource1],                  c1;


yesLastComb0C0D1:
        MAR __ [destAddrHigh, destAddrLow + 0],                                         c1;
yesLastComb0C0D2:
        otLow __ uMask2,                                                                c2;
        sourceIndex __ MD, GOTO [comb0C0D],                                             c3;


yesLastComb0C0D:
        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [yesLastComb0C0D2],                c1;




{****************************** combinationRule = 0E ********************************}
{ - - - - sourceWord bitInvert bitOr: destinationWord bitInvert - - - - }
comb0C0E:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource],                   c1, at [0E, 10, comb0C];

        sourceIndex __ sourceIndex and temp2Low,                                        c1, at [0E, 10, getSource - return]; { sourceWord bitInvert
        }
        otLow __ ~sourceIndex and otLow,        CALL [store1],                          c2;
        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0C0E, yesLastComb0C0E],                               c3, at [0E, 10, store1 - return];


noLastComb0C0E:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource1],                  c1;
        Noop,                                                                           c2, at [0E, 10, getSource1 - return];
        sourceIndex __ MD and sourceIndex,                                              c3;

        MAR __ [destAddrHigh, destAddrLow + 0],                                         c1;
        MDR __ ~sourceIndex,                                                            c2;{ sourceWord bitInvert bitOr: destinationWord
        bitInvert }
        destAddrLow __ destAddrLow + Q,                                                 c3;

        Noop,                                                                           c1;
        temp3Low __ temp3Low - 1, ZeroBr,                                               c2;
        BRANCH [noLastComb0C0E1, yesLastComb0C0E1],                                     c3;


noLastComb0C0E1:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0], CALL [getSource1],                  c1;


yesLastComb0C0E1:
        MAR __ [destAddrHigh, destAddrLow + 0],                                         c1;
yesLastComb0C0E2:
        otLow __ uMask2,                                                                c2;
        sourceIndex __ MD, GOTO [comb0C0E],                                             c3;


yesLastComb0C0E:
        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [yesLastComb0C0E2],                c1;




{ Edit history:

13 - Sep - 85 13:29:06          Tokunga.fx          modify when combination Rule = 6 {comb0C06} }
```

```
{ Comb0D.mc

Combination rule subroutines for bitblt primitive for Rum, the Dandelion Smalltalk - 80 microcoded virtual machine.

        by T Tokunaga, J Trow

         2 - Feb - 86 14:25:19

Copyright 1985, 1986 by Xerox Corporation.  All rights reserved. }



        MAR _ [destAddrHigh, destAddrLow + 0], L3 _ 0, GOTO [comb0E001],                    c1, at [0, 10, comb0D];


        MAR _ [destAddrHigh, destAddrLow + 0], L3 _ 0F, GOTO [comb0E0F1],                   c1, at [0F, 10, comb0D];


{********************* combinationRule = 1 *********************}
{ - - - - sourceWord bitAnd: destinationWord - - - - }
comb0D01:
        temp2Low _ sourceIndex and temp2Low,                                                c1, at [1, 10, comb0D]; { merging }
        temp2Low _ temp2Low and otLow,                                                      c2;
        sourceIndex _ sourceIndex and ~otLow,                                               c3;

        temp2Low _ temp2Low or sourceIndex, L3 _ comb0D01,                                  c1;
        otLow _ uMask2, CALL [write3],                                                      c2;


{ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - }
{ Subroutine : write
 entry : c1
 exit  : c2 , pending zeroBr,  Q = hDir }
{ the click number =  7 click in interation except first word and last word }
write3:
        Noop,                                                                               c3;

write:
        MAR _ [destAddrHigh, destAddrLow + 0],          L2 _ bbtCheckMInt,                  c1;
        MDR _ temp2Low, temp3Low _ temp3Low + 0, NZeroBr,                                   c2; { Write and check}
        sourceIndex _ uPrevWord, BRANCH [finishedWrite, contWrite],                         c3;


contWrite:
        MAR _ [sourceAddrHigh, sourceAddrLow  + 0], L2 _ comb0D,
                CALL [calSkewWord1],                                                        c1;

        Q _ uHDir, L3Disp,                                                                  c1, at [comb0D, 10, calSkewWord - return];
        temp3Low _ temp3Low  -  1, ZeroBr, RET [write - return],                            c2;


finishedWrite:
        destAddrLow _ destAddrLow  + Q,         GOTO [finishedComb0E00],                    c1;

{ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - }


        destAddrLow _ destAddrLow  + Q,
                BRANCH [noLastcomb0D01, yesLastComb0D01],                                   c3, at [comb0D01, 10, write - return];


noLastcomb0D01:
        MAR _ [destAddrHigh, destAddrLow  + 0],                                             c1;
        temp2Low _ temp2Low and uHalftoneWord,                                              c2;
        temp2Low _ MD and temp2Low, CALL [write],                                           c3; { get and merging }


yesLastComb0D01:
        MAR _ [destAddrHigh, destAddrLow  + 0],                                             c1;
        temp2Low _ temp2Low and uHalftoneWord,                                              c2;
        sourceIndex _ MD, GOTO [comb0D01],                                                  c3;


{********************* combinationRule = 2 *********************}
{ - - - - sourceWord bitAnd: destinationWord bitInvert - - - - }
comb0D02:
        temp2Low _ ~sourceIndex and temp2Low,                                               c1, at [2, 10, comb0D]; { merging }
        sourceIndex _ sourceIndex and ~otLow,                                               c2;
        temp2Low _ temp2Low and otLow,                                                      c3;

        temp2Low _ sourceIndex or temp2Low, L3 _ comb0D02,                                  c1;
        otLow _ uMask2, CALL [write3],                                                      c2;

        destAddrLow _ destAddrLow  + Q,
                BRANCH [noLastComb0D02, yesLastComb0D02],                                   c3, at [comb0D02, 10, write - return];


noLastComb0D02:
        MAR _ [destAddrHigh, destAddrLow  + 0],                                             c1;
        temp2Low _ temp2Low and uHalftoneWord,                                              c2;
        temp2Low _ ~MD and temp2Low, CALL [write],                                          c3; { get and merging }


yesLastComb0D02:
```

```
        MAR __ [destAddrHigh, destAddrLow + 0],                          c1;
        temp2Low __ temp2Low and uHalftoneWord,                          c2;
        sourceIndex __ MD, GOTO [comb0D02],                              c3;


{********************** combinationRule = 3 **********************}
{ - - - - sourceWord - - - - }
comb0D03:
        temp2Low __ temp2Low and otLow,                                  c1, at [3, 10, comb0D]; { merging }
        sourceIndex __ sourceIndex and ~otLow, L3 __ comb0D03,           c2;
        temp2Low __ temp2Low or sourceIndex, CALL [write],               c3;

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastcomb0D03, yesLastComb0D03],                c3, at [comb0D03, 10, write - return];


noLastcomb0D03:
        MAR __ [destAddrHigh, destAddrLow + 0],                          c1;
        MDR __ temp2Low and uHalftoneWord,                               c2;
        sourceIndex __ uPrevWord, CALL [contWrite],                      c3;


yesLastComb0D03:
        MAR __ [destAddrHigh, destAddrLow + 0],                          c1;
        temp2Low __ temp2Low and uHalftoneWord,                          c2;
        sourceIndex __ MD,                                               c3;

        otLow __ uMask2,                                                 c1;
        uBBTTemp __ sourceIndex and ~otLow,                              c2;
        otLow __ temp2Low and otLow,                                     c3;

        MAR __ [destAddrHigh, destAddrLow + 0],                          c1;
        MDR __ otLow or uBBTTemp,                                        c2;
        L2 __ bbtCheckMInt, GOTO [finishedWrite],                        c3;


{********************** combinationRule = 4 **********************}
{ - - - - sourceWord bitInvert bitAnd: destinationWord - - - - }
comb0D04:
        uBBTTemp __ sourceIndex and ~otLow,                              c1, at [4, 10, comb0D];
        sourceIndex __ ~temp2Low and sourceIndex,                        c2; { merging }
        otLow __ sourceIndex and otLow,                                  c3;

        temp2Low __ otLow or uBBTTemp, L3 __ comb0D04,                   c1;
        otLow __ uMask2, CALL [write3],                                  c2;

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastcomb0D04, yesLastComb0D04],                c3, at [comb0D04, 10, write - return];


noLastcomb0D04:
        MAR __ [destAddrHigh, destAddrLow + 0],                          c1;
        temp2Low __ temp2Low and uHalftoneWord,                          c2;
        sourceIndex __ MD,                                               c3; { get }

        MAR __ [destAddrHigh, destAddrLow + 0],                          c1;
        MDR __ ~temp2Low and sourceIndex,                                c2; { merging }
        sourceIndex __ uPrevWord, CALL [contWrite],                      c3;


yesLastComb0D04:
        MAR __ [destAddrHigh, destAddrLow + 0],                          c1;
        temp2Low __ temp2Low and uHalftoneWord,                          c2;
        sourceIndex __ MD, GOTO [comb0D04],                              c3;


{********************** combinationRule = 5 **********************}
{ - - - - destinationWord - - - - }
comb0D05:
        GOTO [allFinished],                                              c1, at [5, 10, comb0D];


{********************** combinationRule = 6 **********************}
{ - - - - sourceWord bitXor: destinationWord - - - - }
comb0D06:
        temp2Low __ sourceIndex xor temp2Low,                            c1, at [6, 10, comb0D]; { merging }
        sourceIndex __ sourceIndex and ~otLow,                           c2;
        temp2Low __ temp2Low and otLow,                                  c3;

        temp2Low __ temp2Low or sourceIndex, L3 __ comb0D06,             c1;
        otLow __ uMask2, CALL [write3],                                  c2;

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0D06, yesLastComb0D06],                c3, at [comb0D06, 10, write - return];


noLastComb0D06:
        MAR __ [destAddrHigh, destAddrLow + 0],                          c1;
        temp2Low __ temp2Low and uHalftoneWord,                          c2;
        temp2Low __ MD xor temp2Low, CALL [write],                       c3; { get and merging }


yesLastComb0D06:
        MAR __ [destAddrHigh, destAddrLow + 0],                          c1;
        temp2Low __ temp2Low and uHalftoneWord,                          c2;
        sourceIndex __ MD, GOTO [comb0D06],                              c3;
```

```
{********************* combinationRule = 7 *********************}
{ - - - - sourceWord bitOr: destinationWord - - - - }
comb0D07:
        temp2Low __ sourceIndex or temp2Low,                    c1, at [7, 10, comb0D]; { merging }
        sourceIndex __ sourceIndex and ~otLow,                  c2;
        temp2Low __ temp2Low and otLow,                         c3;

        temp2Low __ temp2Low or sourceIndex, L3 __ comb0D07,    c1;
        otLow __ uMask2, CALL [write3],                         c2;

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0D07, yesLastComb0D07],       c3, at [comb0D07, 10, write - return];


noLastComb0D07:
        MAR __ [destAddrHigh, destAddrLow + 0],                 c1;
        temp2Low __ temp2Low and uHalftoneWord,                 c2;
        temp2Low __ MD or temp2Low, CALL [write],               c3; { get and merging }


yesLastComb0D07:
        MAR __ [destAddrHigh, destAddrLow + 0],                 c1;
        temp2Low __ temp2Low and uHalftoneWord,                 c2;
        sourceIndex __ MD, GOTO [comb0D07],                     c3;


{********************* combinationRule = 8 *********************}
{ - - - - sourceWord bitInvert bitAnd: destinationWord bitInvert - - - - }
comb0D08:
        temp2Low __ ~temp2Low,                                  c1, at [8, 10, comb0D]; { source bitInvert }
        temp2Low __ temp2Low and ~sourceIndex,                  c2; { merging }
        temp2Low __ temp2Low and otLow,                         c3; { skewWord and mergeMask}

        sourceIndex __ sourceIndex and ~otLow,                  c1;
        temp2Low __ temp2Low or sourceIndex, L3 __ comb0D08,    c2; { concatination }
        otLow __ uMask2,    CALL [write],                       c3;

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastcomb0D08, yesLastComb0D08],       c3, at [comb0D08, 10, write - return];


noLastcomb0D08:
        MAR __ [destAddrHigh, destAddrLow + 0],                 c1;
        temp2Low __ temp2Low and uHalftoneWord,                 c2;
        sourceIndex __ ~MD,                                     c3; { get inverted destination }

        MAR __ [destAddrHigh, destAddrLow + 0],                 c1;
        MDR __ ~temp2Low and sourceIndex,                       c2;
        sourceIndex __ uPrevWord, CALL [contWrite],             c3;


yesLastComb0D08:
        MAR __ [destAddrHigh, destAddrLow + 0],                 c1;
        temp2Low __ temp2Low and uHalftoneWord,                 c2;
        sourceIndex __ MD, GOTO [comb0D08],                     c3;


{********************* combinationRule = 9 *********************}
{ - - - - sourceWord bitInvert bitXor: destinationWord - - - - }
comb0D09:
        uBBTTemp __ sourceIndex and ~otLow,                     c1, at [9, 10, comb0D];
        sourceIndex __ sourceIndex xor ~temp2Low,               c2; { merging }
        otLow __ sourceIndex and otLow,                         c3;

        temp2Low __ otLow or uBBTTemp, L3 __ comb0D09,          c1;
        otLow __ uMask2, CALL [write3],                         c2;

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastcomb0D09, yesLastComb0D09],       c3, at [comb0D09, 10, write - return];


noLastcomb0D09:
        MAR __ [destAddrHigh, destAddrLow + 0],                 c1;
        temp2Low __ temp2Low and uHalftoneWord,                 c2;
        sourceIndex __ MD,                                      c3; { get and merging }

        MAR __ [destAddrHigh, destAddrLow + 0],                 c1;
        MDR __ sourceIndex xor ~temp2Low,                       c2;
        sourceIndex __ uPrevWord, CALL [contWrite],             c3;


yesLastComb0D09:
        MAR __ [destAddrHigh, destAddrLow + 0],                 c1;
        temp2Low __ temp2Low and uHalftoneWord,                 c2;
        sourceIndex __ MD, GOTO [comb0D09],                     c3;


{********************* combinationRule = 0A *********************}
{ - - - - destinationWord bitInvert - - - - }
comb0D0A:
        MAR __ [destAddrHigh, destAddrLow + 0],                 c1, at [0A, 10, comb0D];
        MDR __ sourceIndex xor otLow,                           c2;
        temp3Low __ temp3Low + 0, ZeroBr,                       c3;

        destAddrLow __ destAddrLow + Q,
                BRANCH [contComb0D0A, finishedComb0D0A],        c1;
```

```
contComb0D0A:
        temp3Low __ temp3Low - 1, ZeroBr,                                   c2;
        BRANCH [noLastComb0D0A, yesLastComb0D0A],                           c3;


noLastComb0D0A:
        MAR __ [destAddrHigh, destAddrLow + 0],                             c1;
noLastComb0D0A2:
        Noop,                                                               c2;
        sourceIndex __ ~MD,                                                 c3; { get and bitInvert }

        MAR __ [destAddrHigh, destAddrLow + 0],                             c1;
        MDR __ sourceIndex, temp3Low __ temp3Low - 1, ZeroBr,               c2;
        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0D0A1, yesLastComb0D0A1],                 c3;


noLastComb0D0A1:
        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [noLastComb0D0A2],     c1;


yesLastComb0D0A1:
        MAR __ [destAddrHigh, destAddrLow + 0],                             c1;
yesLastComb0D0A2:
        otLow __ uMask2,    L2 __ bbtCheckMInt,                             c2;
        sourceIndex __ MD, GOTO [comb0D0A],                                 c3;


yesLastComb0D0A:
        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [yesLastComb0D0A2],    c1;


finishedComb0D0A:
        temp3Low __ uH,     GOTO [finishedComb0E001],                       c2;


{********************** combinationRule = 0B **********************}
{ - - - - sourceWord bitOr: destinationWord bitInvert - - - - }
comb0D0B:
        uBBTTemp __ sourceIndex and ~otLow,                                 c1, at [0B, 10, comb0D];
        sourceIndex __ sourceIndex and ~temp2Low,                           c2; { Dword and ~Sword}
        otLow __ ~sourceIndex and otLow,                                    c3; { skewWord and MergeMask }

        temp2Low __ otLow or uBBTTemp, L3 __ comb0D0B,                      c1;
        otLow __ uMask2,    CALL [write3],                                  c2;

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0D0B, yesLastComb0D0B],                   c3, at [comb0D0B, 10, write - return];


noLastComb0D0B:
        MAR __ [destAddrHigh, destAddrLow + 0],                             c1;
        temp2Low __ temp2Low and uHalftoneWord,                             c2;
        sourceIndex __ ~MD,                                                 c3; { get and merging }

        MAR __ [destAddrHigh, destAddrLow + 0],                             c1;
        MDR __ temp2Low or sourceIndex,                                     c2;
        sourceIndex __ uPrevWord, CALL [contWrite],                         c3;


yesLastComb0D0B:
        MAR __ [destAddrHigh, destAddrLow + 0],                             c1;
        temp2Low __ temp2Low and uHalftoneWord,                             c2;
        sourceIndex __ MD, GOTO [comb0D0B],                                 c3;


{********************** combinationRule = 0C **********************}
{ - - - - sourceWord bitInvert - - - - }
comb0D0C:
        uBBTTemp __ sourceIndex and ~otLow, L3 __ comb0D0C,                 c1, at [0C, 10, comb0D]; { merging }
        otLow __ ~temp2Low and otLow,                                       c2;
        temp2Low __ otLow or uBBTTemp, CALL [write],                        c3;

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastcomb0D0C, yesLastComb0D0C],                   c3, at [comb0D0C, 10, write - return];


noLastcomb0D0C:
        temp2Low __ temp2Low and uHalftoneWord,                             c1;
        temp2Low __ ~temp2Low,                                              c2;
        Noop,                                                               c3;

        MAR __ [destAddrHigh, destAddrLow + 0],                             c1;
        MDR __ temp2Low,                                                    c2;
        sourceIndex __ uPrevWord, CALL [contWrite],                         c3;


yesLastComb0D0C:
        MAR __ [destAddrHigh, destAddrLow + 0],                             c1;
        temp2Low __ temp2Low and uHalftoneWord,                             c2;
        sourceIndex __ MD,                                                  c3;

        otLow __ uMask2,                                                    c1;
        uBBTTemp __ sourceIndex and ~otLow,                                 c2;
        otLow __ ~temp2Low and otLow,                                       c3;
```

```
        MAR __ [destAddrHigh, destAddrLow + 0],                    c1;
        MDR __ otLow or uBBTTemp,                                  c2;
        L2 __ bbtCheckMInt, GOTO [finishedWrite],                  c3;


{********************* combinationRule = 0D **********************}
{ - - - - sourceWord bitInvert bitOr: destinationWord  - - - - }
comb0D0D:
        uBBTTemp __ sourceIndex and ~otLow,                        c1, at [0D, 10, comb0D]; { merging }
        temp2Low __ temp2Low and ~sourceIndex,                     c2; { SWord and ~DWord }
        otLow __ otLow and ~temp2Low,                              c3; { skewWord and Mergemask }

        temp2Low __ otLow or uBBTTemp, L3 __ comb0D0D,             c1;
        otLow __ uMask2, CALL [write3],                            c2;

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastcomb0D0D, yesLastComb0D0D],          c3, at [comb0D0D, 10, write - return];


noLastcomb0D0D:
        MAR __ [destAddrHigh, destAddrLow + 0],                    c1;
        temp2Low __ temp2Low and uHalftoneWord,                    c2;
        sourceIndex __ ~MD and temp2Low,                           c3; { get and merging }

        MAR __ [destAddrHigh, destAddrLow + 0],                    c1;
        MDR __ ~sourceIndex,                                       c2;
        sourceIndex __ uPrevWord, CALL [contWrite],                c3;

yesLastComb0D0D:
        MAR __ [destAddrHigh, destAddrLow + 0],                    c1;
        temp2Low __ temp2Low and uHalftoneWord,                    c2;
        sourceIndex __ MD, GOTO [comb0D0D],                        c3;


{********************* combinationRule = 0E **********************}
{ - - - - sourceWord bitInvert bitOr: destinationWord bitInvert - - - - }
comb0D0E:
        uBBTTemp __ sourceIndex and ~otLow,                        c1, at [0E, 10, comb0D];
        temp2Low __ temp2Low and sourceIndex,                      c2;
        otLow __ otLow and ~temp2Low,                              c3;

        temp2Low __ otLow or uBBTTemp, L3 __ comb0D0E,             c1;
        otLow __ uMask2,    CALL [write3],                         c2;

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastcomb0D0E, yesLastComb0D0E],          c3, at [comb0D0E, 10, write - return];


noLastcomb0D0E:
        MAR __ [destAddrHigh, destAddrLow + 0],                    c1;
        temp2Low __ temp2Low and uHalftoneWord,                    c2;
        sourceIndex __ MD and temp2Low,                            c3; { get inveted destination }

        MAR __ [destAddrHigh, destAddrLow + 0],                    c1;
        MDR __ ~sourceIndex,                                       c2;
        sourceIndex __ uPrevWord, CALL [contWrite],                c3;


yesLastComb0D0E:
        MAR __ [destAddrHigh, destAddrLow + 0],                    c1;
        temp2Low __ temp2Low and uHalftoneWord,                    c2;
        sourceIndex __ MD, GOTO [comb0D0E],                        c3;



{ Edit history:

 }
```

```
{ Comb0E.mc

Combination rule subroutines for bitblt primitive for Rum, the Dandelion Smalltalk-80 microcoded virtual machine.

        by T Tokunaga, J Trow

        14-Feb-86 17:46:02

Copyright 1985, 1986 by Xerox Corporation. All rights reserved. }



{******************************** combinationRule = 0 ******************************}
{ ---- ^0 ---- }
comb0E00:
        MAR _ [destAddrHigh, destAddrLow + 0],                                  c1, at [0, 10, comb0E];
comb0E001:
        MDR _ sourceIndex and ~otLow,                                           c2;
        temp3Low _ temp3Low, ZeroBr,                                            c3;

        destAddrLow _ destAddrLow + Q, BRANCH [contComb0E00, finishedComb0E00], c1;


contComb0E00:
        temp3Low _ temp3Low - 1, ZeroBr,                                        c2;
        temp2Low _ temp2Low xor temp2Low,
                BRANCH [noLastComb0E001, yesLastComb0E001],                     c3;


noLastComb0E001:
        MAR _ [destAddrHigh, destAddrLow + 0],                                  c1;
noLastComb0E0011:
        MDR _ temp2Low, temp3Low _ temp3Low - 1, ZeroBr,                        c2; { put Zero }
        destAddrLow _ destAddrLow + Q,
                BRANCH [noLastComb0E002, yesLastComb0E002],                     c3;


noLastComb0E002:
        MAR _ [destAddrHigh, destAddrLow + 0], GOTO [noLastComb0E0011],         c1;


yesLastComb0E001:
        MAR _ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],         c1;


yesLastComb0E002:
        MAR _ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],         c1;

        sourceIndex _ MD, GOTO [comb0E00],                                      c3, at [00, 10, getDest-return];


finishedComb0E00:
        temp3Low _ uH,     GOTO [finishedComb0E001],                            c2;
finishedComb0E001:
        temp3Low _ temp3Low - 1, ZeroBr,                                        c3;

finishedComb0E002:
        uH _ temp3Low, BRANCH [vLoopContinue, allFinished],                     c1;


vLoopContinue:
        Noop,                                                                   c2;
        CALL [checkMesaInterrupt],                                              c3;




{******************************** combinationRule = 1 ******************************}
{ ---- sourceWord bitAnd: destinationWord ---- }
comb0E01:
        uBBTTemp _ sourceIndex and ~otLow,                                      c1, at [1, 10, comb0E]; { do nothing }
        sourceIndex _ sourceIndex and temp2Low,                                 c2; { merging }
        otLow _ sourceIndex and otLow, CALL [store0E1],                         c3;


{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}
store0E1:
        MAR _ [destAddrHigh, destAddrLow + 0],                                  c1;
        MDR _ otLow or uBBTTemp, L3Disp,                                        c2;
        temp3Low _ temp3Low + 0, ZeroBr, RET [store0E1-return],                 c3;
{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}

        destAddrLow _ destAddrLow + Q, BRANCH [contComb0E01, finishedComb0E01], c1, at [01, 10, store0E1-return];


contComb0E01:
        temp3Low _ temp3Low - 1, ZeroBr,                                        c2;
        BRANCH [noLastComb0E01, yesLastComb0E01],                               c3;


noLastComb0E01:
        MAR _ [destAddrHigh, destAddrLow + 0],                                  c1;
naLastComb0E012:
```

```
	Noop,								c2;
	sourceIndex _ MD and temp2Low, CALL [store0E2],			c3; { get and merging }


{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}
store0E2:
	MAR _ [destAddrHigh, destAddrLow + 0], L3Disp,			c1;
store0E21:
	MDR _ sourceIndex, temp3Low _ temp3Low - 1, ZeroBr,
		RET [store0E2 - return],				c2;
{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}


	destAddrLow _ destAddrLow + Q,
		BRANCH [noLastComb0E011, yesLastComb0E011],		c3, at [01, 10, store0E2 - return];


noLastComb0E011:
	MAR _ [destAddrHigh, destAddrLow + 0], GOTO [naLastComb0E012],	c1;


yesLastComb0E01:
	MAR _ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],	c1;


yesLastComb0E011:
	MAR _ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],	c1;


{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}
getDest:
	MAR _ [destAddrHigh, destAddrLow + 0], L3Disp,			c1;
getDest1:
	otLow _ uMask2,	RET [getDest - return],				c2;
{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}


	sourceIndex _ MD, GOTO [comb0E01],				c3, at [01, 10, getDest - return];


finishedComb0E01:
	temp3Low _ uH, GOTO [finishedComb0E001],			c2;




{********************************* combinationRule = 2 ******************************}
{ - - - - sourceWord bitAnd: destinationWord bitInvert - - - - }
comb0E02:
	uBBTTemp _ sourceIndex and ~otLow,				c1, at [2, 10, comb0E];
	temp2Low _ temp2Low and ~sourceIndex,				c2; { merging }
	otLow _ temp2Low and otLow, CALL [store0E1],			c3;

	destAddrLow _ destAddrLow + Q, BRANCH [contComb0E02, finishedComb0E02],	c1, at [02, 10, store0E1 - return];


contComb0E02:
	temp3Low _ temp3Low - 1, ZeroBr,				c2;
	temp2Low _ uHalftoneWord, BRANCH [noLastComb0E02, yesLastComb0E02],	c3;


noLastComb0E02:
	MAR _ [destAddrHigh, destAddrLow + 0],				c1;
noLastComb0E022:
	Noop,								c2;
	sourceIndex _ ~MD and temp2Low, CALL [store0E2],		c3; { get and merging }

	destAddrLow _ destAddrLow + Q,
		BRANCH [noLastComb0E021, yesLastComb0E021],		c3, at [02, 10, store0E2 - return];


noLastComb0E021:
	MAR _ [destAddrHigh, destAddrLow + 0], GOTO [noLastComb0E022],	c1;


yesLastComb0E02:
	MAR _ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],	c1;


yesLastComb0E021:
	MAR _ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],	c1;

	sourceIndex _ MD, GOTO [comb0E02],				c3, at [02, 10, getDest - return];


finishedComb0E02:
	temp3Low _ uH, GOTO [finishedComb0E001],			c2;




{********************************* combinationRule = 3 ******************************}
{ - - - - sourceWord - - - - }
comb0E03:
	uBBTTemp _ sourceIndex and ~otLow,				c1, at [3, 10, comb0E];
```

```
        otLow __ temp2Low and otLow,                                        c2;
        otLow __ otLow or uBBTTemp,                                         c3;

        MAR __ [destAddrHigh , destAddrLow + 0],                            c1;
        MDR __ otLow, temp3Low __ temp3Low + 0, ZeroBr,                     c2;
        destAddrLow __ destAddrLow + Q, BRANCH [contComb0E03, finishedComb0E03],  c3;


contComb0E03:
        temp3Low __ temp3Low - 1, ZeroBr,                                   c1;
        BRANCH [noLastComb0E03, yesLastComb0E03],                           c2;


noLastComb0E03:
        sourceIndex __ temp2Low,      CALL [store0E2],                      c3;

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0E031, yesLastComb0E031],                 c3, at [03, 10, store0E2 - return];


noLastComb0E031:
        MAR __ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [store0E21],   c1;


yesLastComb0E03:
        CALL [getDest],                                                     c3;


yesLastComb0E031:
        MAR __ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],    c1;

        sourceIndex __ MD, GOTO [comb0E03],                                 c3, at [03, 10, getDest - return];


finishedComb0E03:
        GOTO [finishedComb0E00],                                            c1;




{******************************** combinationRule = 4 ******************************}
{ - - - - sourceWord bitInvert bitAnd: destinationWord - - - - }
comb0E04:
        uBBTTemp __ sourceIndex {destination word} and ~otLow {mask},       c1, at [4, 10, comb0E];
        sourceIndex __ sourceIndex and ~temp2Low {halftone word},           c2;
comb0E041:
        otLow __ sourceIndex and otLow, CALL [store0E1],                    c3;

        destAddrLow __ destAddrLow + Q, BRANCH [contComb0E04, finishedComb0E04],  c1, at [04, 10, store0E1 - return];


contComb0E04:
        temp3Low __ temp3Low - 1, ZeroBr,                                   c2;
        BRANCH [noLastComb0E04, yesLastComb0E04],                           c3;


noLastComb0E04:
        MAR __ [destAddrHigh, destAddrLow + 0],                             c1;
noLastComb0E042:
        temp2Low __ ~uHalftoneWord,                                         c2;
        sourceIndex __ MD and temp2Low, CALL [store0E2],                    c3; { get and merging }

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0E041, yesLastComb0E041],                 c3, at [04, 10, store0E2 - return];


noLastComb0E041:
        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [noLastComb0E042],     c1;


yesLastComb0E04:
        MAR __ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],    c1;


yesLastComb0E041:
        MAR __ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],    c1;

        sourceIndex __ MD,                                                  c3, at [04, 10, getDest - return];

        uBBTTemp __ sourceIndex and ~otLow,                                 c1;
        sourceIndex __ sourceIndex and ~temp2Low, GOTO [comb0E041],         c2;


finishedComb0E04:
        temp3Low __ uH, GOTO [finishedComb0E001],                           c2;




{******************************** combinationRule = 5 ******************************}
{ - - - - destinationWord - - - - }
comb0E05:
        GOTO [allFinished],                                                 c1, at [5, 10, comb0E];
```

```
{******************************** combinationRule = 6 ******************************}
{ - - - - sourceWord bitXor: destinationWord - - - - }
comb0E06:
        uBBTTemp __ sourceIndex and ~otLow,                                  c1, at [6, 10, comb0E]; { do nothing }
        sourceIndex __ sourceIndex xor temp2Low,                             c2; { merging }
        otLow __ sourceIndex and otLow, CALL [store0E1],                     c3;

        destAddrLow __ destAddrLow + Q, BRANCH [contComb0E06, finishedComb0E06],   c1, at [06, 10, store0E1 - return];


contComb0E06:
        temp3Low __ temp3Low - 1, ZeroBr,                                    c2;
        BRANCH [noLastComb0E06, yesLastComb0E06],                            c3;


noLastComb0E06:
        MAR __ [destAddrHigh, destAddrLow + 0],                              c1;
noLastComb0E062:
        Noop,                                                                c2;
        sourceIndex __ MD xor temp2Low, CALL [store0E2],                     c3; { get and merging }

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0E061, yesLastComb0E061],                  c3, at [06, 10, store0E2 - return];


noLastComb0E061:
        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [noLastComb0E062],      c1;


yesLastComb0E06:
        MAR __ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],     c1;


yesLastComb0E061:
        MAR __ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],     c1;

        sourceIndex __ MD, GOTO [comb0E06],                                  c3, at [06, 10, getDest - return];


finishedComb0E06:
        temp3Low __ uH, GOTO [finishedComb0E001],                            c2;



{******************************** combinationRule = 7 ******************************}
{ - - - - sourceWord bitOr: destinationWord - - - - }
comb0E07:
        uBBTTemp __ sourceIndex and ~otLow,                                  c1, at [7, 10, comb0E]; { do nothing }
        sourceIndex __ sourceIndex or temp2Low,                              c2; { merging }
        otLow __ sourceIndex and otLow, CALL [store0E1],                     c3;

        destAddrLow __ destAddrLow + Q, BRANCH [contComb0E07, finishedComb0E07],   c1, at [07, 10, store0E1 - return];


contComb0E07:
        temp3Low __ temp3Low - 1, ZeroBr,                                    c2;
        BRANCH [noLastComb0E07, yesLastComb0E07],                            c3;


noLastComb0E07:
        MAR __ [destAddrHigh, destAddrLow + 0],                              c1;
noLastComb0E072:
        Noop,                                                                c2;
        sourceIndex __ MD or temp2Low, CALL [store0E2],                      c3; { get and merging }

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0E071, yesLastComb0E071],                  c3, at [07, 10, store0E2 - return];


noLastComb0E071:
        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [noLastComb0E072],      c1;


yesLastComb0E07:
        MAR __ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],     c1;


yesLastComb0E071:
        MAR __ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],     c1;

        sourceIndex __ MD, GOTO [comb0E07],                                  c3, at [07, 10, getDest - return];


finishedComb0E07:
        temp3Low __ uH, GOTO [finishedComb0E001],                            c2;



{******************************** combinationRule = 8 ******************************}
{ - - - - sourceWord bitInvert bitAnd: destinationWord bitInvert - - - - }
```

```
comb0E08:
        uBBTTemp __ sourceIndex and ~otLow,                                  c1, at [8, 10, comb0E];
        temp2Low __ ~temp2Low,                                               c2; { sourceWord bitInvert }
comb0E081:
        temp2Low __ ~sourceIndex and temp2Low,                               c3; { merging }

        otLow __ temp2Low and otLow,                                         c1;
        otLow __ otLow or uBBTTemp,                                          c2;
        Noop,                                                                c3;

        MAR __ [destAddrHigh, destAddrLow + 0],                              c1;
        MDR __ otLow, temp3Low __ temp3Low + 0, ZeroBr,                      c2;
        destAddrLow __ destAddrLow + Q, BRANCH [contComb0E08, finishedComb0E08],  c3;


contComb0E08:
        temp3Low __ temp3Low - 1, ZeroBr,                                    c1;
        temp2Low __ ~uHalftoneWord, BRANCH [noLastComb0E08, yesLastComb0E08],  c2;


noLastComb0E08:
        Noop,                                                                c3;

noLastComb0E081:
        MAR __ [destAddrHigh, destAddrLow + 0],                              c1;
        Noop,                                                                c2;
        sourceIndex __ ~MD and temp2Low,      CALL [store0E2],               c3; { get and merging }

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0E081, yesLastComb0E081],                  c3, at [08, 10, store0E2 - return];


yesLastComb0E08:
        CALL [getDest],                                                      c3;


yesLastComb0E081:
        MAR __ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],     c1;

        sourceIndex __ MD,                                                   c3, at [08, 10, getDest - return];

        uBBTTemp __ sourceIndex and ~otLow,                                  c1;
        GOTO [comb0E081],                                                    c2;


finishedComb0E08:
        GOTO [finishedComb0E00],                                             c1;




{********************************* combinationRule = 9 ********************************}
{ - - - - sourceword bitInvert: bitXor: destinationWord  - - - - }
comb0E09:
        uBBTTemp __ sourceIndex and ~otLow,                                  c1, at [9, 10, comb0E];
        sourceIndex __ sourceIndex xor ~temp2Low,                            c2; { merging }
comb0E091:
        otLow __ sourceIndex and otLow, CALL [store0E1],                     c3;

        destAddrLow __ destAddrLow + Q, BRANCH [contComb0E09, finishedComb0E09],  c1, at [09, 10, store0E1 - return];


contComb0E09:
        temp3Low __ temp3Low - 1, ZeroBr,                                    c2;
        temp2Low __ ~temp2Low, BRANCH [noLastComb0E09, yesLastComb0E09],     c3;


noLastComb0E09:
        MAR __ [destAddrHigh, destAddrLow + 0],                              c1;
noLastComb0E092:
        Noop,                                                                c2;
        sourceIndex __ MD xor temp2Low, CALL [store0E2],                     c3;

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0E091, yesLastComb0E091],                  c3, at [09, 10, store0E2 - return];


noLastComb0E091:
        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [noLastComb0E092],      c1;


yesLastComb0E09:
        MAR __ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],     c1;


yesLastComb0E091:
        MAR __ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],     c1;

        sourceIndex __ MD,                                                   c3, at [09, 10, getDest - return];

        uBBTTemp __ sourceIndex and ~otLow,                                  c1;
        sourceIndex __ sourceIndex xor temp2Low, GOTO [comb0E091],           c2;


finishedComb0E09:
        temp3Low __ uH, GOTO [finishedComb0E001],                            c2;
```

```
{******************************** combinationRule = 0A ******************************}
{ - - - - desinationWord bitInvert - - - - }
comb0E0A:
        uBBTTemp __ sourceIndex and ~otLow,                                  c1, at [0A, 10, comb0E];
        otLow __ ~sourceIndex and otLow,                                     c2;
        otLow __ otLow or uBBTTemp,                                          c3;

        MAR __ [destAddrHigh, destAddrLow + 0],                              c1;
        MDR __ otLow, temp3Low __ temp3Low + 0, ZeroBr,                      c2;
        destAddrLow __ destAddrLow + Q, BRANCH [contComb0E0A, finishedComb0E0A],  c3;


contComb0E0A:
        temp3Low __ temp3Low - 1, ZeroBr,                                    c1;
        BRANCH [noLastComb0E0A, yesLastComb0E0A],                            c2;


noLastComb0E0A:
        Noop,                                                                c3;

noLastComb0E0A1:
        MAR __ [destAddrHigh, destAddrLow + 0],                              c1;
        Noop,                                                                c2;
        sourceIndex __ ~MD, CALL [store0E2],                                 c3; { get and merging }

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0E0A1, yesLastComb0E0A1],                  c3, at [0A, 10, store0E2 - return];


yesLastComb0E0A:
        CALL [getDest],                                                      c3;


yesLastComb0E0A1:
        MAR __ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],     c1;

        sourceIndex __ MD, GOTO [comb0E0A],                                  c3, at [0A, 10, getDest - return];


finishedComb0E0A:
        GOTO [finishedComb0E00],                                             c1;



{******************************** combinationRule = 0B ******************************}
{ - - - - sourceword bitOr: destinationWord bitInvert - - - - }
comb0E0B:
        uBBTTemp __ sourceIndex and ~otLow,                                  c1, at [0B, 10, comb0E];
        sourceIndex __ ~temp2Low and sourceIndex,                            c2; { merging }
        otLow __ ~sourceIndex and otLow,        CALL [store0E1],             c3;

        destAddrLow __ destAddrLow + Q, BRANCH [contComb0E0B, finishedComb0E0B],  c1, at [0B, 10, store0E1 - return];


contComb0E0B:
        temp3Low __ temp3Low - 1, ZeroBr,                                    c2;
        BRANCH [noLastComb0E0B, yesLastComb0E0B],                            c3;


noLastComb0E0B:
        MAR __ [destAddrHigh, destAddrLow + 0],                              c1;
noLastComb0E0B2:
        Noop,                                                                c2;
        sourceIndex __ ~MD,                                                  c3; { get }

        sourceIndex __ sourceIndex or temp2Low,                              c1;        { merging }
        Noop,                                                                c2;
        CALL [store0E2],                                                     c3;


noLastComb0E0B1:
        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [noLastComb0E0B2],      c1;


        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0E0B1, yesLastComb0E0B1],                  c3, at [0B, 10, store0E2 - return];


yesLastComb0E0B:
        MAR __ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],     c1;


yesLastComb0E0B1:
        MAR __ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],     c1;

        sourceIndex __ MD, GOTO [comb0E0B],                                  c3, at [0B, 10, getDest - return];


finishedComb0E0B:
        temp3Low __ uH, GOTO [finishedComb0E001],                            c2;
```

```
{******************************** combinationRule = 0C ******************************}
{ - - - - sourceWord bitInvert - - - - }
comb0E0C:
        uBBTTemp __ sourceIndex and ~otLow,                                  c1, at [0C, 10, comb0E];
        otLow __ ~temp2Low and otLow,                                        c2;
        otLow __ otLow or uBBTTemp,                                          c3;

        MAR __ [destAddrHigh , destAddrLow + 0],                             c1;
        MDR __ otLow, temp3Low __ temp3Low + 0, ZeroBr,                      c2;
        destAddrLow __ destAddrLow + Q, BRANCH [contComb0E0C, finishedComb0E0C],  c3;


contComb0E0C:
        temp3Low __ temp3Low - 1, ZeroBr,                                    c1;
        BRANCH [noLastComb0E0C, yesLastComb0E0C],                            c2;


noLastComb0E0C:
        sourceIndex __ ~temp2Low, CALL [store0E2],                           c3;


noLastComb0E0C1:
        MAR __ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [store0E21],    c1;


        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0E0C1, yesLastComb0E0C1],                  c3, at [0C, 10, store0E2 - return];


yesLastComb0E0C:
        CALL [getDest],                                                      c3;


yesLastComb0E0C1:
        MAR __ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],     c1;

        sourceIndex __ MD, GOTO [comb0E0C],                                  c3, at [0C, 10, getDest - return];


finishedComb0E0C:
        GOTO [finishedComb0E00],                                             c1;



{******************************** combinationRule = 0D ******************************}
{ - - - - sourceWord bitInvert bitOr: destinationWord - - - - }
comb0E0D:
        uBBTTemp __ sourceIndex and ~otLow,                                  c1, at [0D, 10, comb0E];
        temp2Low __ temp2Low and ~sourceIndex,                               c2; { merging }
        otLow __ ~temp2Low and otLow, CALL [store0E1],                       c3;

        destAddrLow __ destAddrLow + Q, BRANCH [contComb0E0D, finishedComb0E0D],  c1, at [0D, 10, store0E1 - return];


contComb0E0D:
        temp3Low __ temp3Low - 1, ZeroBr,                                    c2;
        BRANCH [noLastComb0E0D, yesLastComb0E0D],                            c3;


noLastComb0E0D:
        MAR __ [destAddrHigh, destAddrLow + 0],                              c1;
noLastComb0E0D2:
        temp2Low __ ~temp2Low,                                               c2;
        sourceIndex __ MD,                                                   c3; { get and merging }

        sourceIndex __ sourceIndex or temp2Low,                              c1;
        temp2Low __ ~temp2Low,                                               c2;
        CALL [store0E2],                                                     c3;


noLastComb0E0D1:
        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [noLastComb0E0D2],      c1;

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0E0D1, yesLastComb0E0D1],                  c3, at [0D, 10, store0E2 - return];


yesLastComb0E0D:
        MAR __ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],     c1;


yesLastComb0E0D1:
        MAR __ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],     c1;

        sourceIndex __ MD, GOTO [comb0E0D],                                  c3, at [0D, 10, getDest - return];


finishedComb0E0D:
        temp3Low __ uH,     GOTO [finishedComb0E001],                        c2;
```

```
{******************************** combinationRule = 0E ******************************}
{ - - - - sourceWord bitInvert bitOr: destinationWord bitInvert - - - - }
comb0E0E:
        uBBTTemp __ sourceIndex and ~otLow,                                   c1, at [0E, 10, comb0E];
        sourceIndex __ ~sourceIndex,                                          c2; { destinationWord bitInvert }
        sourceIndex __ sourceIndex and ~temp2Low,                             c3; { merging }

        otLow __ sourceIndex and otLow,                                       c1;
        otLow __ otLow or uBBTTemp,                                           c2;
        Noop,                                                                 c3;

        MAR __ [destAddrHigh, destAddrLow + 0],                               c1;
        MDR __ otLow, temp3Low __ temp3Low + 0, ZeroBr,                       c2;
        destAddrLow __ destAddrLow + Q, BRANCH [contComb0E0E, finishedComb0E0E],  c3;


contComb0E0E:
        temp3Low __ temp3Low - 1, ZeroBr,                                     c1;
        BRANCH [noLastComb0E0E, yesLastComb0E0E],                             c2;


noLastComb0E0E:
        Noop,                                                                 c3;


noLastComb0E0E1:
        MAR __ [destAddrHigh, destAddrLow + 0],                               c1;
        temp2Low __ ~temp2Low,                                                c2;
        sourceIndex __ ~MD,                                                   c3; { get and merging }

        sourceIndex __ sourceIndex or temp2Low,                               c1;
        temp2Low __ ~temp2Low,                                                c2;
        CALL [store0E2],                                                      c3;

        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0E0E1, yesLastComb0E0E1],                   c3, at [0E, 10, store0E2 - return];


yesLastComb0E0E:
        CALL [getDest],                                                       c3;


yesLastComb0E0E1:
        MAR __ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],      c1;

        sourceIndex __ MD, GOTO [comb0E0E],                                   c3, at [0E, 10, getDest - return];


finishedComb0E0E:
        GOTO [finishedComb0E00],                                              c1;



{******************************** combinationRule = 0F ******************************}
{ - - - - sourceWord - - - - }
comb0E0F:
        MAR__ [destAddrHigh, destAddrLow + 0],                                c1, at [0F, 10, comb0E];
comb0E0F1:
        MDR __ sourceIndex or otLow,                                          c2;
        temp3Low __ temp3Low, ZeroBr,                                         c3;

        destAddrLow __ destAddrLow + Q, BRANCH [contComb0E0F, finishedComb0E0F],  c1;
contComb0E0F:
        temp3Low __ temp3Low - 1, ZeroBr,                                     c2;
        sourceIndex __ sourceIndex xor ~ sourceIndex,
                BRANCH [noLastComb0E0F1, yesLastComb0E0F1],                   c3;


noLastComb0E0F1:
        MAR __ [destAddrHigh, destAddrLow + 0],                               c1;
noLastComb0E0F11:
        MDR __ sourceIndex, temp3Low __ temp3Low - 1, ZeroBr,                 c2; { put Zero }
        destAddrLow __ destAddrLow + Q,
                BRANCH [noLastComb0E0F2, yesLastComb0E0F2],                   c3;


noLastComb0E0F2:
        MAR __ [destAddrHigh, destAddrLow + 0], GOTO [noLastComb0E0F11],      c1;


yesLastComb0E0F1:
        MAR __ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],      c1;


yesLastComb0E0F2:
        MAR __ [destAddrHigh, destAddrLow + 0], L3Disp, CALL [getDest1],      c1;

        sourceIndex __ MD, GOTO [comb0E0F],                                   c3, at [0F, 10, getDest - return];


finishedComb0E0F:
```

```
        temp3Low __ uH,     GOTO [finishedComb0E001],                    c2;


{ Edit history:
 }
```

{ BBT.mc

Bitblt primitive for Rum, the Dandelion Smalltalk – 80 microcoded virtual machine.

by T Tokunaga, M Sakakibara, J Trow

14 – Feb – 86 16:50:25

{ <primitive: 96> BitBit > > copyBits

Perform the movement of bits from one Form to another described by the instance variables of the receiver. Fail if any instance variable is not of the right type (Integer or Form) or if combinationRule is not between 0 and 15 inclusive. Set the variables and try again (BitBlt > > copyBitsAgain).

input:

output:

smash: }

```
primitiveCopyBits:
        uSaveIPL __ ipLow,                                          c1;
        ipLow __ ipHigh ,                                           c2;
        uSaveIPH __ ipLow.                                          c3; { save Smalltalk Instruction Pointer }

        uSaveStackL __ stackLow,                                    c1;
        stackLow __ stackHigh,                                      c2;
        uSaveStackH __ stackLow,                                    c3; { save Smalltalk Stack Pointer }

        uSaveHomeL __ homeLow,                                      c1;
        homeLow __ homeHigh,                                        c2;
        uSaveHomeH __ homeLow,                                      c3; { save the Pointer to home context }

checkMesaStackP:
        Xbus __ ErrnlBnStkp, XDisp,                                 c1; { X [12~15] __ ~stackP}
        DISP4 [bitBltHowBigStack, 7],                               c2; { Only for case of Mesa Int }
        stackLow __ uSaveStackL, GOTO [normalEntry],                c3, at [0F, 10, bitBltHowBigStack];
```

```
{ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  - -  After Mesa Interrupt  - - - - - - -
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -}
```

{At this point, part of the Parameters of Smalltalk BitBlt are stored in following order }

```
{
 Oop for bitblt Argument        | uDestBitMap          |      U02
 Loop counter (Vertical)        | ul                   |      U03
 Destination Address(low)       | uDestAddrLow         |      U04
 Destination Address(high)      | *uDestAddrHigh       |      U05
 uSource, Dest Delta            | *uDestDelta          |      U06
  uMaks1                        | Mask1                |      U07
  uMask2                        | Mask2                |      U08
  uSkewMask                     | skewMask             |      U09
  uMisc                         | *Misc                |      U0A
  uNWords                       | NWords               |      U0B
  uHalftoneAddrLow              | HalftoneAddrLow      |      U0C :
  uSourceAddrLow                | SourceAddrLow        |      U0D :
  uSaveHighAddr                 | *SaveHighAddr        |      U0E
```

```
**********uDestAddrHigh
    00 – 03      : unused
    04 – 07      : combinationRule
    08 – 15      : destination Address high
********** uMisc
    00 ~ 03      : DY's offset( LS Nibble)
    04 ~ 07      : skew
    08           : unused
    09           : vDir = – 1?
    0A           : hDir = – 1?
    0B           : unused
    0C           : skew = 0?
    0D           : halftoneFormNIL – – 1 – > halftoneForm = Nil, 0 – > halftoneForm nonNil
    0E           : sourceFormNIL – – 1 – > sourceForm = Nil, 0 – > sourceForm nonNil
    0F           : preload – – 1 – > preload True(uPreload = all 1), 0 – > preload False(uPreload = 0)
```

```
*************** uSaveHighAddr
    00 - 07          : High address of Source bitMap
    08 - 0F          : High address of Halftone bitmap

*************** uDestDelta
    00 - 07          : source Delta
    08 - 0F          : dest Delta

  Note :  it means that these parameters without * marking is used for calculating the parameter again used in copyLoop
}



{ Now, We calculate the parameter(mask1, Mask2, skewMask, skew, vDir, hDir, preload) used in copybit Loop, again. }

afterMesaInt:
        temp2Low __ uMisc, XwdDisp,                                   c3, at [07, 10, bitBltHowBigStack];

afterMesaInt1:
        Q __ 1, DISP2 [checkDirMInt],                                 c1;


{vDir = hDir = 1}
        uVDir __ Q,                                                   c2, at [0, 4, checkDirMInt];
        uHDir __ Q, GOTO [checkSkew],                                 c3;


{vDir = 1, hDir = -1}
        uHDir __ temp2Low xor ~temp2Low,                              c2, at [1, 4, checkDirMInt];
        uVDir __ Q, GOTO [checkSkew],                                 c3;


{vDir = -1, hDir = 1}
        uHDir __ Q,                                                   c2, at [2, 4, checkDirMInt];
        uVDir __ temp2Low xor ~temp2Low, GOTO [checkSkew],            c3;


{vDir __ -1, hDir = -1}
        uVDir __ temp2Low xor ~temp2Low,                              c2, at [3, 4, checkDirMInt];
        uHDir __ temp2Low xor ~temp2Low,                              c3;

checkSkew:
        temp3Low __ temp2Low and 0F, YDisp,                           c1; {temp2Low = 00 - 03:DY, 04 - 07:skew, 09:hdir, 0A:vDir,
        0c - 0F: booleans}
        uBooleans __ temp3Low,
                BRANCH [skewNonZeroAfterMInt, skewZeroAfterMInt, 7],  c2;


skewNonZeroAfterMInt:
        sourceIndex __ temp2Low LRot8,                                c3;

        sourceIndex __ sourceIndex  and 0F,                           c1;
        skew __ sourceIndex LRot0,                                    c2; {restore skew}
        GOTO [checkForm],                                             c3;


skewZeroAfterMInt:
        skew __ 0,                                                    c3; {restore skew}

checkForm:
        temp3Low __ uDestDelta, XLDisp,                               c1;
        destAddrLow __ temp3Low and 0FF, BRANCH [destDeltaP, destDeltaM, 1],  c2;
destDeltaM:
        destAddrLow __ destAddrLow xor ~0FF,                          c3; { create minus value for destDelta}

        Noop,                                                         c1;
        Noop,                                                         c2;
destDeltaP:
        uDestDelta __ destAddrLow,                                    c3; {DestDelta}

        Xbus __ uBooleans, XDisp,                                     c1;
        sourceIndex __ uSaveHighAddr, DISP4 [checkFormNil, 9],        c2;


{both nonNil}
bothNonNil:
        temp1High __ sourceIndex LRot0,                               c3, at [9, 10, checkFormNil]; {highaddress of halftone}

        temp1Low __ uHalftoneAddrLow,                                 c1; {low address of halftone form}
        temp2Low __ temp2Low LRot4,                                   c2;
        uDY __ temp2Low,                                              c3; {save dy's offset}

restoreSAddr:
        otLow __ sourceIndex LRot8,                                   c1;
        sourceAddrHigh __ otLow LRot0,                                c2; {highaddress of source}
        sourceAddrLow __ uSourceAddrLow,                              c3; {low address of source Form }

restoreSDelta:
        temp3Low __ temp3Low LRot8, XLDisp,                           c1; { check sourceDelta < 0}
        temp3Low __ temp3Low and 0FF, BRANCH [sourceDeltaP, sourceDeltaM, 1],  c2;


sourceDeltaM:
        temp3Low __ temp3Low xor ~0FF, GOTO [sourceDeltaP1],          c3;
```

```
sourceDeltaP:
        Noop,                                                                   c3;
sourceDeltaP1:
        uSourceDelta __ temp3Low,                                               c1; { sourceDelta}
        sourceAddrLow __ sourceAddrLow + temp3Low, GOTO [bothNilAfterMInt],     c2;


{SForm = nil, HForm = nonNil}
sNilhNonNil:
        temp1High __ sourceIndex LRot0,                                         c3, at [0B, 10, checkFormNil];

        temp1Low __ uHalftoneAddrLow,                                           c1; { halftone form address}
        temp2Low __ temp2Low LRot4,                                             c2;
        temp2Low __ temp2Low and 0F,                                            c3;

        uDY __ temp2Low, GOTO [bothNilAfterMInt1],                              c1; {dy's offset}


{SForm = nonNil, HForm = nil}
sNonNilhNil:
        GOTO [restoreSAddr],                                                    c3, at [0D, 10, checkFormNil];


{both nil}
bothNilAfterMInt1:
        Noop,                                                                   c2;
bothNilAfterMInt:
        otLow __ uDestAddrHigh,                                                 c3, at [0F, 10, checkFormNil]; {otLow = destdelta}

        destAddrHigh __ otLow LRot0,                                            c1;
        otLow __ otLow LRot8,                                                   c2;
        uCombinationRule __ otLow,                                              c3;

goVLoopMInt:
        otLow __ uDestAddrLow                                                   c1; {otLow = destdelta}
        destAddrLow __ destAddrLow + otLow, Xbus __ uBooleans, XDisp,
                L2 __ bbtCheckMInt, GOTO [startVLoop1],                         c2; { restore destination address (16 bit)}




{
        - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
        - - - - -  First Entry for Smalltalk BitBlt  - - - - -
        - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
}

normalEntry:
        MAR __ [stackHigh, stackLow + 0],                                       c1;
        L1 __ getParameter,                                                     c2;
        otLow __ MD, CALL [otMap2Bank0],                                        c3; { get the real address of object }

normalEntry1:
        temp1Low __ temp1Low + firstFieldOfObject,                              c1, at [getParameter, 10, otMap2Bank0 - return]; { Point
        the destinationForm }
        uArgument __ otLow,                                                     c2; { save the original Oop }
        Q __ nilPointer,                                                        c3;

{ Now get the BilBlt Argument }

normalEntry2:
        MAR __ [temp1High, temp1Low + 0],                                       c1;
        temp1Low __ temp1Low + 1,                                               c2; { point the SourceForm }
        destAddrLow __ MD,                                                      c3; { get destination Form ****** }

normalEntry3:
        MAR __ [temp1High, temp1Low + 0],                                       c1; {**********}
        temp1Low __ temp1Low + 1,                                               c2; { point the halftoneForm }
        sourceAddrLow __ MD,                                                    c3; { get source Form ******** }

normalEntry4:
        MAR __ [temp1High, temp1Low + 0],                                       c1; {**********}
        temp1Low __ temp1Low + 1, L1 __ checkCombinationRule,                   c2; { point the combinationRule }
        otLow __ MD,                                                            c3; { get halftoneForm and check three Oop of Form
        **********}

        [] __ otLow and nonPointerMask, ZeroBr,                                 c1;
        [] __ destAddrLow and nonPointerMask, ZeroBr,
                BRANCH [$, halftoneNonOop],                                     c2;
        [] __ sourceAddrLow and nonPointerMask, ZeroBr,
                BRANCH [$, destinationNonOop],                                  c3;

normalEntry5:
        MAR __ [temp1High, temp1Low + 0], BRANCH [$, sourceNonOop],             c1;
        uDestForm __ destAddrLow, CALL [checkSmallInt],                         c2; {save destination Form }

normalEntry55:
        [] __ temp2Low and ~0F, ZeroBr,                                         c1, at [checkCombinationRule, 10, checkSmallInt - return];
        BRANCH [combinationRuleTooBig, $],                                      c2;
        Noop,                                                                   c3;

normalEntry6:
        MAR __ [temp1High, temp1Low + 0], L1 __ checkDestX,                     c1;
        uCombinationRule __ temp2Low, CALL [checkSmallInt],                     c2;
```

```
normalEntry7:
        MAR _ [temp1High, temp1Low + 0], L1 _ checkDestY,                    c1, at [checkDestX, 10, checkSmallInt - return];
        uDestX _ temp2Low, CALL [checkSmallInt],                             c2;

normalEntry8:
        MAR _ [temp1High, temp1Low + 0], L1 _ checkDestWidth,                c1, at [checkDestY, 10, checkSmallInt - return];
        uDestY _ temp2Low, CALL [checkSmallInt],                             c2;

normalEntry9:
        MAR _ [temp1High, temp1Low + 0], L1 _ checkDestHeight,               c1, at [checkDestWidth, 10, checkSmallInt - return];
        uDestWidth _ temp2Low, CALL [checkSmallInt],                         c2;

normalEntry10:
        MAR _ [temp1High, temp1Low + 0], L1 _ checkSourceX,                  c1, at [checkDestHeight, 10, checkSmallInt - return];
        uDestHeight _ temp2Low, CALL [checkSmallInt],                        c2;

normalEntry11:
        MAR _ [temp1High, temp1Low + 0], L1 _ checkSourceY,                  c1, at [checkSourceX, 10, checkSmallInt - return];
        uSourceX _ temp2Low, CALL [checkSmallInt],                           c2;

normalEntry12:
        MAR _ [temp1High, temp1Low + 0], L1 _ checkClipX,                    c1, at [checkSourceY, 10, checkSmallInt - return];
        uSourceY _ temp2Low, CALL [checkSmallInt],                           c2;

normalEntry13:
        MAR _ [temp1High, temp1Low + 0], L1 _ checkClipY,                    c1, at [checkClipX, 10, checkSmallInt - return];
        uClipX _ temp2Low, CALL [checkSmallInt],                             c2;

normalEntry14:
        MAR _ [temp1High, temp1Low + 0], L1 _ checkClipWidth,                c1, at [checkClipY, 10, checkSmallInt - return];
        uClipY _ temp2Low, CALL [checkSmallInt],                             c2;

normalEntry15:
        MAR _ [temp1High, temp1Low + 0], L1 _ checkClipHeight,               c1, at [checkClipWidth, 10, checkSmallInt - return];
        uClipWidth _ temp2Low, CALL [checkSmallInt],                         c2;

normalEntry16:
        [] _ sourceAddrLow - Q, ZeroBr,          {Q : nilPointer}            c1, at [checkClipHeight, 10, checkSmallInt - return];
        [] _ otLow - Q, ZeroBr, BRANCH [sourceFormNonNilXX, sourceFormNilXX], c2;


sourceFormNonNilXX:
        destAddrLow _ 0, BRANCH [halftoneFormNonNilXX1, halftoneFormNilXX1], c3;


sourceFormNilXX:
        destAddrLow _ 2, BRANCH [halftoneFormNonNilXX1, halftoneFormNilXX1], c3; { halftoneform = nil }


halftoneFormNonNilXX1:
        GOTO [saveOop],                                                      c1;


halftoneFormNilXX1:
        destAddrLow _ destAddrLow or 4, GOTO [saveOop],                      c1; { sourceForm = nil }


saveOop:
        uBooleans _ destAddrLow,                                             c2; {save the boolean. so far sourceformNil,
        halftoneformNil}
        Q _ uClipHeight _ temp2Low,                                          c3; {save clipHieght}

normalEntry17:
        uSourceForm _ sourceAddrLow,                                         c1; {save sourceForm}
        uHalftoneForm _ otLow,                                               c2; {save halftoneForm}


{       {****************** debug ***********************************}
        L1 _ 1,                                                              c3;

checkDebug:
        L1Disp,                                                              c1;
        BRANCH [$, normalEntry18, 0E],                                       c2;
        L2 _ primFail, GOTO [primitiveFailBitBlt1],                          c3;
        {****************** debug ***********************************}                    }

{If destWidth < = 0 OR destHeight < = 0 OR clipWidth < = 0 OR clipHeight < = 0 THEN immediately Return, Nothing is done }

normalEntry18:
        [] _ temp2Low - 1, NegBr,                                            c3; { check clipHeight < = 0 }

        temp2Low _ uClipWidth, BRANCH [$, clipHeightLess0],                  c1;
        [] _ temp2Low - 1, NegBr,                                            c2; { check clipWidth < = 0 }
        temp3Low _ uDestWidth, BRANCH [$, clipWidthLess0],                   c3;

normalEntry19:
        temp3Low _ temp3Low - 1, NegBr, L3 _ checkClip,                      c1;

{ In Height, we check whether the area size(width*height) is greater than 64640?. If it is true, it cause a primitive fail }
{ Q : height, temp2Low : Width,  NOTE ::: temp2Low, Q, sourceIndex, temp1Low is smashed in the routine }

        otLow _ uDestHeight,
                BRANCH [{CALL} checkWidthAndHeight, destWidthLess0],         c2;

        otLow _ otLow - 1, NegBr,                                            c3, at [checkClip, 10, checkWidthAndHeight - return];
```

```
normalEntry20:
        BRANCH [$, destHeightLess0],                                    c1;
        otLow _ uDestForm, L1 _ getDestFormData,                        c2;
        CALL [otMap2Bank0],                                             c3;

getDestination:
        MAR _ temp1Low _ [temp1High, temp1Low + firstFieldOfObject],    c1, at [getDestFormData, 10, otMap2Bank0 - return];
getDestinationX:
        temp1Low _ temp1Low + 1, BRANCH [noPCGetDest, yesPCGetDest, 1], c2;


yesPCGetDest:
        temp1Low _ temp1Low + 0FF,                                      c3;

        MAR _ [temp1High, temp1Low + 0], GOTO [getDestinationX],        c1;


noPCGetDest:
        sourceIndex _ MD {destination bitmap},                          c3; { get Form BitMap Oop }

getDestination1:
        temp3Low _ uRumRecordLow,                                       c1;
        temp3High _ uRumRecordHigh,                                     c2;
        temp3Low _ temp3Low + displayBitmapOopOffset,                   c3; { point the current display BitMap Oop }

getDestination2:
        MAR _ [temp3High, temp3Low + 0],                                c1;
        [] _ sourceIndex and nonPointerMask, ZeroBr,                    c2; { save the destination BitMap Oop }
        uDestBitMap _ sourceIndex, temp3Low _ MD {display bitmap},
                BRANCH [$, destinationFormNonOop],                      c3; { get current display BitMap Oop }

getDestination3:
        [] _ temp3Low xor sourceIndex, ZeroBr,                          c1; {dest bitmap = current display bitmap? ********}

        uCurrentDispBitMap _ temp3Low,
                BRANCH [noDestAndCurrent, yesDestAndCurrent],           c2;


{ destAddrLow = uBooleans }
yesDestAndCurrent:
        temp3Low _ 3,                                                   c3;

yesDestAndCurrent1:
        temp3Low _ temp3Low LRot8,                                      c1;
        temp3Low _ temp3Low or 28,                                      c2; { 808'd }
        uDestHeightA _ temp3Low,                                        c3;

yesDestAndCurrent2:
        temp3Low _ 4,                                                   c1;
        temp3Low _ temp3Low LRot8,                                      c2; { 1024'd }
        uDestWidthA _ temp3Low,                                         c3;

yesDestAndCurrent3:
        destAddrLow _ destAddrLow + 80,                                 c1;
        GOTO [noDestAndCurrent31],                                      c2;


noDestAndCurrent:
        temp1Low _ temp1Low + 1, { Point the Form.Width }               c3;

noDestAndCurrent1:
        MAR _ [temp1High, temp1Low + 0], L1 _ checkDestFormWidth,       c1;
        CALL [checkSmallInt2],                                          c2;

{temp1Low points the Height, since it is incremented by 1 in checkSmallInt2 routine }
noDestAndCurrent2:
        MAR _ [temp1High, temp1Low + 0], L1 _ checkDestFormHeight,      c1, at [checkDestFormWidth, 10, checkSmallInt2 - return];
        uDestWidthA _ temp2Low, CALL [checkSmallInt2],                  c2;

noDestAndCurrent3:
        uDestHeightA _ temp2Low, L3 _ checkDest,                        c1, at [checkDestFormHeight, 10, checkSmallInt2 - return];
        Q _ uDestWidthA, CALL [checkWidthAndHeight],                    c2;
noDestAndCurrent31:
        uBooleans _ destAddrLow, GOTO [clipRange1],                     c3, at [checkDest, 10, checkWidthAndHeight - return];


destinationFormNonOop:
        L2 _ primFail, GOTO [primitiveFailBitBlt2],                     c1;




{- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  - - - - - - - - - -                   ClipRange                       - - - - - - - - - -
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -}

{First of all, we check the X - coordinate }
clipRange1:
        Noop,                                                           c1;
        sourceAddrLow _ uSourceX,                                       c2;
        destAddrLow _ uDestX,                                           c3;

clipRange:
        sourceIndex _ uDestWidth,                                       c1;
        temp2Low _ uClipX, L2 _ checkXRange,                            c2;
```

```
        temp3Low __ uClipWidth, CALL [bbtCheckRange],                          c3;

saveX:
        uSX __ sourceAddrLow,                                                  c2, at [checkXRange, 10, bbtCheckRange - return];
        uDX __ destAddrLow,                                                    c3;

        uW __ sourceIndex, NegBr,                                              c1; { w < 0? }
checkRangeUpY:
        sourceAddrLow __ uSourceY, BRANCH [widthGE0, widthLT0],                c2;


widthGE0:
        destAddrLow __ uDestY,                                                 c3;

        temp2Low __ uClipY,                                                    c1;
        sourceIndex __ uDestHeight, L2 __ checkYRange,                         c2;
        temp3Low __ uClipHeight, CALL [bbtCheckRange],                         c3;

saveY:
        uH __ sourceIndex, NegBr, L1 __ getSFormWH,                            c2, at [checkYRange, 10, bbtCheckRange - return];
        uDY __ destAddrLow, BRANCH [heightGE0, heightLT0],                     c3;


heightGE0:
        Xbus __ uBooleans, XDisp,                                              c1;
        uSY __ sourceAddrLow, BRANCH [sFormNonNil0, sFormNil0, 0D],            c2;


sFormNonNil0:
        otLow __ uSourceForm, CALL [otMap2Bank0],                              c3;

sFormNonNil01:
        temp1Low __ temp1Low + formHeightIndex,                                c1, at [getSFormWH, 10, otMap2Bank0 - return];
        sourceAddrLow __ sourceAddrLow + sourceIndex,                          c2; { sy + h }
        sourceIndex __ sourceIndex - sourceAddrLow,                            c3; { h __ h - (sy + h)}

sFormNonNil02:
        MAR __ [temp1High, temp1Low + 0], L1 __ checkSourceFormHeight,         c1;
        temp1Low __ temp1Low - 2, CALL [checkSmallInt],                        c2; { point the form width, -2 since temp1Low incremented
        by 1 in checksmallInt }

sFormNonNil03:
        [] __ temp2Low - sourceAddrLow, NegBr,                                 c1, at [checkSourceFormHeight, 10, checkSmallInt - return];
        {sy + h > sourceForm height??}
        sourceIndex __ sourceIndex + temp2Low,
                BRANCH [sFormHeightGE, sFormHeightLT],                         c2; { h __ h + sourceForm Height }


sFormHeightGE:
        GOTO [checkWidth],                                                     c3;


sFormHeightLT:
        uH __ sourceIndex,                                                     c3;

{ - - - - - - next : check width  - - - - - - - - }
checkWidth:
        sourceAddrLow __ uSX,                                                  c1;
        sourceIndex __ uW,                                                     c2;
        sourceAddrLow __ sourceAddrLow + sourceIndex,                          c3; { sx + w }

checkWidth1:
        MAR __ [temp1High, temp1Low + 0], L1 __ checkSourceFormWidth,          c1;
        sourceIndex __ sourceIndex - sourceAddrLow, CALL [checkSmallInt],      c2; { w __ w - (sx + w )}

checkWidth2:
        [] __ temp2Low - sourceAddrLow, NegBr,                                 c1, at [checkSourceFormWidth, 10, checkSmallInt - return];
        { sx + w > sourceForm width ??}
        sourceIndex __ sourceIndex + temp2Low,
                        BRANCH [sFormWidthGE, sFormWidthLT],                   c2;


sFormWidthGE:
        sourceIndex __ uW, GOTO [sFormNil1],                                   c3;


sFormWidthLT:
        uW __ sourceIndex, GOTO [sFormNil1],                                   c3;


sFormNil0:
        sourceIndex __ uW,                                                     c3;

{ - - - - - - check the h and w, if h < = 0 OR w < = 0 then return immediately - - - - - - }
sFormNil1:
        sourceAddrLow __ uH,                                                   c1;
        sourceIndex __ sourceIndex - 1, NegBr,                                 c2; { check the width = < 0 ??}
        sourceAddrLow __ sourceAddrLow - 1, NegBr, BRANCH [widthGT0, widthLE0], c3; { check the height = < 0 ???}

widthGT0:
        uHM1 __ sourceAddrLow, BRANCH [heightGT0, heightLE0],                  c1;


heightGT0:
        L1 __ getMasks, GOTO [computeMask],                                    c2;
```

```
widthLE0:
        CANCELBR [$, 0F],                                               c1;
heightLE0:
        L2 _ noTransfer, GOTO [noTransfer3],                            c2; { restore the smalltalk state and execute next byte
        code }



{------------------------------------------------------------------------------------------
 ---------------------- compute the several kind of Mask -------------------
 -------------------------------------------------------------------------------------------}
computeMask:
        Q _ 1, CALL [getSeveralMasks],                                  c3; { Q for the intial value of nWords }

{
At return point :
mask1, mask2, skewMask, skew and startBits have already been calculated, and been stored in uMask1, uMask2, SkewMask, uSkew, and
uStartBits respectively
 }

getSeveralMasksRet:
        otLow _ uW,                                                     c1;
getSeveralMasksRet1:
        destAddrLow _ uStartBits,                                       c2;
        otLow _ otLow - destAddrLow, NegBr,                             c3; {w < startBits ?? : w - startBits }

        sourceIndex _ uMask2, BRANCH [widthNotTooSmall, widthTooSmall], c1;


widthTooSmall:
        sourceIndex _ sourceIndex and uMask1,                           c2; { mask1 _ mask1 bitAnd: mask2 }
        uMask1 _ sourceIndex,                                           c3;

        uMask2 _ sourceIndex xor sourceIndex,                           c1; { mask2 _ 0 }
        uNWords _ Q,                                                    c2; { uNWords _ 0{1}, because of 0 origin }
        otLow _ Q, GOTO [checkOverlap],                                 c3;


widthNotTooSmall:
        otLow _ otLow - 1, NegBr,                                       c2;
        otLow _ otLow and ~0F,      BRANCH [noNegative, yesNegative],   c3;


yesNegative:
        otLow _ otLow xor ~otLow, GOTO [noNegative1],                   c1; { make otLow - 1}


noNegative:
        otLow _ otLow LRot12,                                           c1; { (w - startBits - 1//16}
noNegative1:
        otLow _ otLow + 2,                                              c2; { (w - startBits - 1//16 + 2 }
        uNWords _ otLow, GOTO [checkOverlap],                           c3; { save nWords }



{------------------------------------------------------------------------------------------------
 ----------                    CheckOverlap                    ----------
 ------------------------------------------------------------------------------------------------}
checkOverlap:
        Noop,                                                           c1;
        otLow _ otLow - Q,                                              c2; {decrement}
        uNWordsM1 _ otLow,                                              c3;

checkOverlap1:
        uStackF _ otLow,                                                c1;
        uVDir _ Q,                                                      c2; { default v-direction = 1}
        uHDir _ Q, L2 _ getDestBits,                                    c3; { default h-direction = 1 }

checkOverlap2:
        temp1Low _ uSourceForm,                                         c1;
        destAddrLow _ uDY,                                              c2;
        sourceAddrLow _ uSY,                                            c3;

checkOverlap3:
        [] _ temp1Low xor uDestForm, ZeroBr,                            c1; { sourceForm = = destFrom ??}

        [] _ destAddrLow - sourceAddrLow, NegBr,
                BRANCH [sameOopNo, sameOopYes],                         c2; { dy > = sy ?? }


sameOopYes:
        [] _ sourceAddrLow xor destAddrLow, ZeroBr, BRANCH [dyGE, dyLT], c3; { dy = sy ??}


dyGE:
        sourceIndex _ uHM1, BRANCH [dyGT, dyEQ],                        c1; { sourceIndex _ h - 1}


{sourceForm = = destForm and dy > sy}
dyGT:
        uVDir _ temp2Low xor ~temp2Low,                                 c2; { vDir _ - 1 }
        destAddrLow _ destAddrLow + sourceIndex,                        c3; { dy _ dy + h - 1}
```

```
dyGT1:
        sourceAddrLow __ sourceAddrLow + sourceIndex,                   c1; { sy __ sy + h - 1}
        uDY __ destAddrLow,                                             c2;
        uSY __ sourceAddrLow, GOTO [dyLT],                              c3;


{sourceForm = =destForm and dy = sy}
dyEQ:
        sourceAddrLow __ uSX,                                           c2;
        destAddrLow __ uDX,                                             c3;

dyEQ1:
        [] __ sourceAddrLow - destAddrLow, NegBr,                       c1; { dx > sx ??}
        uHDir __ temp2Low xor ~temp2Low,    BRANCH [dxLE, dxGT],        c2; { hDir __ -1 : So far default}


{ sourceForm = =destForm and dy = sy and dx > sx }
dxGT:
        sourceIndex __ uW,                                              c3;

        sourceIndex __ sourceIndex - 1,                                 c1;
        sourceAddrLow __ sourceAddrLow + sourceIndex,                   c2; { sx __ sx + w - 1}
        destAddrLow __ destAddrLow + sourceIndex,                       c3; { dx __ dx + w - 1}

dxGT1:
        otLow __ uSkewMask,                                             c1;
        uSkewMask __ ~otLow,                                            c2; { skewMask bitInvert }
        temp1Low __ uMask1,                                             c3;

dxGT2:
        temp2Low __ uMask2,                                             c1;
        uMask1 __ temp2Low,                                             c2;
        uMask2 __ temp1Low,                                             c3; {exchange mask1 and mask2}

dxGT3:
        uDX __ destAddrLow,                                             c1;
        uSX __ sourceAddrLow,                                           c2; { save sx }

sameOopNo:
        CANCELBR [dyLT, 1],                                             c3;


dxLE:
        uHDir __ Q, GOTO [dyLT],                                        c3;


dyLT:
        CANCELBR [$, 1],                                                c1;




{- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  - - - - - - - - - - - - - - - - - - - - calculate Offsets - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -}

calculateOffset:
        otLow __ uDestBitMap, CALL [getSTFormMapBase1],                 c2;

{ at return
   temp1High, temp1Low : map base
   temp2Low :  word no. per 1 horizontal line ( destRaster )
   otLow : Oop for bitmap
}
calculateOffset1:
        destAddrLow __ temp1Low, L2 __ getDestMul,                      c2, at [getDestBits, 10, getSTFormMapBase - return];
        Q __ uDY, CALL [bbtMultiply],                                   c3; { destIndex __ dy*destRaster : Q __ Q*temp2Low }

{ destAddrLow Points the actual destination Address. Now, we calculate the destDelta }
        destAddrLow __ destAddrLow + Q,                                 c2, at [getDestMul, 10, bbtMultiply - return];
        sourceIndex __ uDX,                                             c3;

getDestAddr1:
        sourceIndex __ sourceIndex and ~0F,                             c1;
        sourceIndex __ sourceIndex LRot12,                              c2;
        destAddrLow __ destAddrLow + sourceIndex,                       c3; { destIndex __ destRaster*dy + (dx//16)}

getDestAddr2:
        temp1Low __ temp1High,                                          c1;
        destAddrHigh __ temp1Low LRot0,                                 c2;{ save the destination High address }
        Xbus __ uVDir, XHDisp,                                          c3;

checkDir:
        Xbus __ uHDir, XHDisp, BRANCH [vP1, vM1, 2],                    c1;


vP1:
        sourceIndex __ uNWords, BRANCH [vP1hP1, vP1hM1, 2],             c2;{save the destination Map to check}


vP1hP1:
        temp2Low __ temp2Low - sourceIndex, L2 __ getSourceBits,
                GOTO [saveDestDelta1],                                  c3; { destDelta __ destRaster *1 - nWords * 1 }


vP1hM1:
```

```
        temp2Low __ temp2Low + sourceIndex, L2 __ getSourceBits,
                GOTO [saveDestDelta1],                                  c3; { destDealta __ destRaster*1 - nWords*(-1)}

vM1:
        sourceIndex __ uNWords, BRANCH [vM1hP1, vM1hM1, 2],             c2;

vM1hP1:
        temp2Low __ - temp2Low,                                         c3;

        temp2Low __ temp2Low - sourceIndex { - 1}, GOTO [saveDestDelta], c1; { destDelta __ destRaster*(-1) - nWords*1}

vM1hM1:
        temp2Low __ sourceIndex - temp2Low, GOTO [saveDestDelta1],      c3; {destDelta __ destRaster*(-1) - nWords*(-1)}

saveDestDelta1:
        Noop,                                                           c1;
saveDestDelta:
        uDestDelta __ temp2Low, L2 __ getSourceBits,                    c2;
{ Next we check the sourceForm }

getSFormAddr:
        temp1Low __ uBooleans, XDisp,                                   c3;

        otLow __ uSourceForm,
                BRANCH [{CALL} getSTFormMapBase, saveBooleans, 0D],     c1; { default preload = False, so uBooleans.15 = 0}

{ at return
        temp1High, Low : source MapBase
        temp2Low : Word no of 1 horizontal line ( sourceRaster )
        otLow : Oop for BitMap
}

getSFormAddr1:
        sourceAddrLow __ temp1Low, L2 __ getSourceMul,                  c2, at [getSourceBits, 10, getSTFormMapBase - return];
        Q __ uSY, CALL [bbtMultiply],                                   c3; { sourceIndex __ sy*sourceRaster : Q __ Q*temp2Low }

{ sourceAddrLow points the actual source address }
        sourceAddrLow __ sourceAddrLow + Q,                             c2, at [getSourceMul, 10, bbtMultiply - return];
        Q __ temp1High,                                                 c3;

getSourceAddr:
        sourceAddrHigh __ Q LRot0,                                      c1; { save high source address }
        sourceIndex __ uSX,                                             c2;
        temp1Low __ sourceIndex and ~0F,                                c3;

        temp1Low __ temp1Low LRot12,                                    c1;
        Q __ skew, ZeroBr,                                              c2;
        sourceAddrLow __ sourceAddrLow + temp1Low,
                BRANCH [skewNonZero3, skewZero3],                       c3;{ sourceIndex __ sourceRaster*sy + (sx//16)}

skewNonZero3:
        sourceIndex __ sourceIndex and 0F,                              c1; { sx bitAnd: 15 }
skewNonZero4:
        [] __ sourceIndex - Q, NegBr,                                   c2; { skew < = (sx bitAnd: 15)}
        temp1Low __ uBooleans, BRANCH [skewLE1, skewGT1],               c3;

skewLE1:
        temp1Low __ temp1Low or 1, GOTO [checkhDir],                    c1; { preload = True }

skewGT1:
        GOTO [checkhDir],                                               c1; { preload = False }

skewZero3:
        temp1Low __ uBooleans, GOTO [checkhDir],                        c1;

{ Now, temp2Low still has the sourceRaster }
checkhDir:
        Q __ uHDir, XHDisp,                                             c2;
        [] __ temp1Low, YDisp, BRANCH [hP1, hM1, 2],                    c3;

hP1:
        sourceIndex __ uNWords,
                BRANCH [hP1PreFalse, hP1PreTrue, 0E],                   c1;

hP1PreFalse:
        sourceIndex __ sourceIndex, GOTO [checkVDir1],                  c2; { nWords + 0*1 : preload = False, hDir = 1 }

hP1PreTrue:
        sourceIndex __ sourceIndex + 1, GOTO [checkVDir1],              c2; { nWords + 1*1 : preload = True, hDir = 1 }

hM1:
        sourceIndex __ uNWords,
                BRANCH [hM1PreFalse, hM1PreTrue, 0E],                   c1;
```

```
{ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  - - - - Now, hDir < 0 ifTrue: [preload __ preload = = false]
  - - - - so, preloadFalse actually means preload = True, preloadTrue vice versa,
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - }
hM1PreFalse:
        sourceIndex __ 0 - sourceIndex - 1,                             c2; {nWords + 1*(-1) : preload = False, hDir = -1 }
        temp1Low __ temp1Low or 1 , GOTO [checkVDir2],                  c3; { make preload = True }


hM1PreTrue:
        sourceIndex __ 0 - sourceIndex ,                                c2; {nWords + 0*(-1) : preload = Trued, hDir = -1 }
        temp1Low __ temp1Low and 0E, GOTO [checkVDir2],                 c3; { make preload = False }


checkVDir1:
        Noop,                                                           c3;

checkVDir2:
        Xbus __ uVDir, XHDisp,                                          c1;
        Q __ - temp2Low, BRANCH [vDirP1, vDirM1, 2],                    c2; { Q __ - sourceRaster }


vDirP1:
{ sourceRaster*1 - (nWords +(preload ifTrue: [1] ifFalse: [0])*hDir)}
        temp2Low __ temp2Low - sourceIndex,                             c3;

        uSourceDelta __ temp2Low, GOTO [saveBooleans],                  c1;


vDirM1:
{sourceRaster*(-1) - (nWords +(preload ifTrue: [1] ifFalse: [0])*hDir)}
        Q __ Q - sourceIndex,                                           c3;

        uSourceDelta __ Q,                                              c1;
saveBooleans:
        uBooleans __ temp1Low, YDisp, L1 __ getHalftoneBits,            c2;
{ now Get the halftone Form address }
        otLow __ uHalftoneForm, BRANCH [{CALL} otMap2Bank0, hFormNil21, 0B],   c3;


getHalftone:
        MAR __ temp1Low __ [temp1High, temp1Low + formMapBaseIndex],    c1, at [getHalftoneBits, 10, otMap2Bank0 - return];
getHalftoneAddr:
        BRANCH [noPCGetHalftone, yesPCGetHalftone, 1],                  c2;


yesPCGetHalftone:
        temp1Low __ temp1Low + 0FF + 1,                                 c3;

        MAR __ [temp1High, temp1Low + 0], GOTO [getHalftoneAddr],       c1;


noPCGetHalftone:
        otLow __ MD, L1 __ getHalftoneBits1,                            c3;

        [] __ otLow and nonPointerMask, ZeroBr,                         c1;
        BRANCH [$, hBitMapNoOop],                                       c2;
        CALL [otMap2Bank0],                                             c3;


hFormNil21:
        GOTO [startVLoop],                                              c1;


hFormNil2:
        temp1Low __ temp1Low + firstFieldOfObject,                      c1, at [getHalftoneBits1, 10, otMap2Bank0 - return];



{ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 - - Register Usage:     R0         - - >    (sourceIndex)                                                              - -
 - -                     RH1, R1              - - >   Address of destinationBits        (destAddrHigh, sourceAddrLow)          - -
 - -                                                                                                                    - -
 - -                     RH2, R2              - - >   Address of sourceBits             (sourceAddrHigh, sourceAddrLow)        - -
 - -                     RH3        - - >    **** High Address of OT     ******                                         - -
 - -                     R3         - - >    DestinationIndex                 (destIndex)                                 - -
 - -                     RH5, R5    - - >    Address of halftoneBits          (HalftoneAddrHigh, HalftoneAddrLow)         - -
 - -                                                                                                                    - -
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
{
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 - - - - - - - - - - - - - - - - - - - - - - - - - - - Start of Vertical Loop - - - - - - - - - - - - - - - - - - - - - - - - - - -
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 }
startVLoop:
        Xbus __ uBooleans, XDisp, L2 __ bbtCheckMInt,                   c2;
startVLoop1:
        L1 __ 0, Q __ uHDir, DISP4 [vLoop],                             c3;


{ - - - - - - - skew#0, HForm = SForm = ~nil, Preload = FALSE - - - - - - - - - - - }
startVLoop00:
        sourceIndex __ uVDir, CALL [getHalftoneWord],                   c1, at [00, 10, vLoop];
```

```
startVLoop002:
        temp3Low __ uNWordsM1, L2 __ skewWord,                          c2, at [00, 10, getHalftoneWord - return];
startVLoop001:
        otLow __ uMask1,                                                c3;

        sourceIndex __ sourceIndex xor sourceIndex,                     c1;
        Noop,                                                           c2;
        CALL [calSkewWord],                                             c3;

        MAR __ [destAddrHigh, destAddrLow + 0],                         c1, at [skewWord, 10, calSkewWord - return];
        temp2Low __ temp2Low and uHalftoneWord,                         c2;
        sourceIndex __ MD,                                              c3;

        L2 __ bbtCheckMInt,                                             c1;
        Xbus __ uCombinationRule, XDisp,                                c2;
        Q __ uHDir, DISP4 [comb0D],                                     c3;


{ - - - - - - - skew < >0, HForm = ~nil, SForm = ~nil, Preload = TRUE - - - - - - - - - - - - - - - }
startVLoop01:
        sourceIndex __ uVDir, CALL [getHalftoneWord],                   c1, at [01, 10, vLoop];

startVLoop011:
        temp3Low __ uNWordsM1,                                          c2, at [01, 10, getHalftoneWord - return];
        otLow __ uMask1,                                                c3;

        MAR __ [sourceAddrHigh, sourceAddrLow + 0],                     c1;
        sourceAddrLow __ sourceAddrLow + Q, L2 __ skewWord,             c2;
        sourceIndex __ MD, CALL [calSkewWord],                          c3;


{ - - - - - - - skew < >0, HForm = ~nil, SForm = nil, Preload = FALSE ** - - - - - - - - - - - - - - }
startVLoop02:
        sourceIndex __ uVDir, CALL [getHalftoneWord],                   c1, at [02, 10, vLoop];

        temp3Low __ uNWordsM1, GOTO [startVLoop0A1],                    c2, at [02, 10, getHalftoneWord - return];


{ - - - - - - - skew < >0, HForm = ~nil, SForm = nil, Preload = TRUE - - - - - - -INVALID    - - - - - - - - - - - - - - }
startVLoop03:
        GOTO [iteration],                                               c1, at [03, 10, vLoop];


{ - - - - - - - skew < >0, HForm = nil, SForm = ~nil, Preload = FALSE - - - - - - - - - - - - - - - }
startVLoop04:
        temp2Low __ uHalftoneWord __ temp2Low xor ~temp2Low,            c1, at [04, 10, vLoop];
        temp3Low __ uNWordsM1, L2 __ skewWord, GOTO [startVLoop001],    c2;


{ - - - - - - - skew < >0, HForm = nil, SForm = ~nil, Preload = TRUE - - - - - - - - - - - - - - - }
startVLoop05:
        uHalftoneWord __ temp2Low xor ~temp2Low, GOTO [startVLoop011],  c1, at [05, 10, vLoop];


{ - - - - - - - skew < >0, HForm = nil, SForm = nil, Preload = FALSE ** - - - - - - - - - - - - - - }
startVLoop06:
        temp2Low __ uHalftoneWord __ temp2Low xor ~temp2Low,
                GOTO [startVLoop0E1],                                   c1, at [06, 10, vLoop];


{ - - - - - - - skew < >0, HForm = nil, SForm = nil, Preload = TRUE - - - - - - -INVALID    - - - - - - - - - - - - - - }
startVLoop07:
        GOTO [iteration],                                               c1, at [07, 10, vLoop];


{ - - - - - - - skew = 0, HForm = ~nil, SForm = ~nil, Preload = FALSE - - - - - - - - - - - - - - - }
startVLoop08:
        sourceIndex __ uVDir, CALL [getHalftoneWord],                   c1, at [08, 10, vLoop];

        otLow __ uMask1,                                                c2, at [08, 10, getHalftoneWord - return];
startVLoop081:
        temp3Low __ uNWordsM1,                                          c3;

        MAR __ [destAddrHigh, destAddrLow + 0],                         c1;
        Xbus __ uCombinationRule, XDisp,                                c2;
        sourceIndex __ MD, L3 __ 0, DISP4 [comb0C],                     c3;


{ - - - - - - - skew = 0, HForm = ~nil, SForm = ~nil, Preload = TRUE - - - - - - -INVALID    - - - - - - - - - - - - - - }
startVLoop09:
        GOTO [iteration],                                               c1, at [09, 10, vLoop];


{ - - - - - - - skew = 0, HForm = ~nil, SForm = nil, Preload = FALSE - - - - - - - - - - - - - - - }
startVLoop0A:
        sourceIndex __ uVDir, CALL [getHalftoneWord],                   c1, at [0A, 10, vLoop];

        temp3Low __ uNWordsM1,                                          c2, at [0A, 10, getHalftoneWord - return];
startVLoop0A1:
        otLow __ uMask1,                                                c3;

        MAR __ [destAddrHigh, destAddrLow + 0],                         c1;
        Xbus __ uCombinationRule, XDisp,                                c2;
        sourceIndex __ MD, L3 __ 0, DISP4 [comb0E],                     c3;
```

```
{ - - - - - - - skew = 0, HForm = ~nil, SForm = nil, Preload = TRUE - - - - - - - - INVALID   - - - - - - - - - - - - - - }
startVLoop0B:
        GOTO [iteration],                                                   c1, at [0B, 10, vLoop];


{ - - - - - - - skew = 0, HForm = nil, SForm = ~nil, Preload = FALSE - - - - - - - - - - - - - - }
startVLoop0C:
        temp2Low _ uHalftoneWord _ temp2Low xor ~temp2Low,                  c1, at [0C, 10, vLoop];
        otLow _ uMask1, GOTO [startVLoop081],                               c2;


{ - - - - - - - skew = 0, HForm = nil, SForm = ~nil, Preload = TRUE - - - - - - - - INVALID   - - - - - - - - - - - - - - }
startVLoop0D:
        GOTO [iteration],                                                   c1, at [0D, 10, vLoop];


{ - - - - - - - skew = 0, HForm = nil, SForm = nil, Preload = FALSE - - - - - - - - - - - - - - }
startVLoop0E:
        temp2Low _ uHalftoneWord _ temp2Low xor ~temp2Low,                  c1, at [0E, 10, vLoop];
startVLoop0E1:
        temp3Low _ uNWordsM1, GOTO [startVLoop0A1],                         c2;


{ - - - - - - - skew = 0, HForm = nil, SForm = nil, Preload = TRUE - - - - - - -   INVALID   - - - - - - - - - - - - - - }
startVLoop0F:
        Noop,                                                               c1, at [0F, 10, vLoop];
iteration:
        Noop,                                                               c2;
        GOTO [startVLoop0F],                                                c3;




{ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  - - - - - - - - - - - - - - - - - - - - - - - - - - - Start of Horizontal Loop - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  - - - -         sourceAddrHigh, sourceAddrLow                (RH2, R2)        - -
  - - - -         destAddrHigh, destAddrLow                    (RH1, R1)        - -
  - - - -         temp1High, temp1Low(Halftone Form)           (RH4, R4)        - -
  - - - -         temp2Low : prevWord                 (R5)                      - -
  - - - -         sourceIndex : skewWord              (R0)                      - -
  - - - -         skew :                              (RH0)                     - -
  - - - -         temp3Low : loop conter(word)        (R6)                      - -
  - - - -         otLow : mask1(mergeMask)            (R3)                      - -
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - }


{ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  - - - - - - no Mesa Interrupt - - - - - - - - - - - - -
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - }
reentryBitblt:
        destAddrLow _ destAddrLow + temp3Low, L1Disp,
                BRANCH [sFormNonNilMIntCheck, sFormNilMIntCheck, 0D],       c2, at [bbtCheckMInt, 10, noMesaInterrupt - return];


sFormNilMIntCheck:
        Q _ uHDir, DISP4 [vLoop],                                           c3; { update destAddr }


sFormNonNilMIntCheck:
        Q _ uSourceDelta, CANCELBR [$, 0F],                                 c3;

        sourceAddrLow _ sourceAddrLow + Q,                                  c1;
        L1Disp,    GOTO [sFormNilMIntCheck],                                c2;


{ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  - - - - - - - - - - Mesa Interrupt - - - - - - - - - - -
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - }
 {save the bitBlt parameters into the mesa stack, and restore smalltalk ip, stackp, home

  temp2Low = 0F , this value is store in checkMesaInterrupt}
mesaIntInBitblt:
        uDestAddrLow _ destAddrLow,                                         c3, at [bbtCheckMInt, 10, mesaInterrupt - return];

mesaIntInBitblt1:
        destAddrLow _ destAddrHigh, stackP _ temp2Low,                      c1;
        sourceIndex _ uCombinationRule,                                     c2;
        sourceIndex _ sourceIndex LRot8,                                    c3;

        destAddrLow _ destAddrLow or sourceIndex,                           c1;
        uDestAddrHigh _ destAddrLow,                                        c2; {save highaddr of dest and combination Rule }
        sourceIndex _ uDY,                                                  c3;

mesaIntInBitblt2:
        sourceIndex _ sourceIndex and temp2Low,                             c1; {08 - 0B: dy's offset}
        sourceIndex _ sourceIndex LRot4,                                    c2;
        Q _ skew,                                                           c3;

mesaIntInBitblt3:
        sourceIndex _ sourceIndex or Q,                                     c1; {00 - 03: dy's offset, 04 - 07: skew}
        sourceIndex _ sourceIndex LRot8,                                    c2;
        temp2Low _ uBooleans, XDisp,                                        c3;

mesaIntInBitblt4:
```

er }

-atus – return];

Bank0 – return];

«llint – return]; { size <

ect }

ursorbitmap}

```
        temp2Low __ temp2Low and 0F, DISP4 [shForm, 9],                          c1;


{halftone = nil, source~ = nil}
        uSourceAddrLow __ sourceAddrLow,                                         c2, at [0D, 10, shForm]; { sourceAddrLow }
        sourceAddrLow __ sourceAddrHigh,                                         c3;

        sourceAddrLow __ sourceAddrLow LRot8,                                    c1;
        uSaveHighAddr __ sourceAddrLow,                                          c2; {00 - 07: sourceAddrHigh }
saveDelta:
        temp1Low __ uSourceDelta,                                                c3;

        temp1Low __ temp1Low and 0FF, GOTO [bothNil],                            c1;


{halftone~ = nil, source = nil}
        uHalftoneAddrLow __ temp1Low,                                            c2, at [0B, 10, shForm];
        temp1Low __ temp1High,                                                   c3;

        uSaveHighAddr __ temp1Low, GOTO [bothNil],                               c1; {08 - 0F: halftoneAddrHigh}


{halftone~ = nil, source~ = nil}
        uHalftoneAddrLow __ temp1Low,                                            c2, at [9, 10, shForm]; {save halftoneAddrLw }
        Q __ temp1High,                                                          c3;

        uSourceAddrLow __ sourceAddrLow,                                         c1; {sacve sourceAddrLow}
        sourceAddrLow __ sourceAddrHigh,                                         c2;
        sourceAddrLow __ sourceAddrLow LRot8,                                    c3;

        Q __ Q or sourceAddrLow,                                                 c1; {00 - 07:sourceAddrHigh, 08 - 0F:halftoneAddrHigh}
        uSaveHighAddr __ Q, GOTO [saveDelta],                                    c2;


{halftone = nil, source = nil}
bothNil:
        temp3Low __ uDestDelta,                                                  c2, at [0F, 10, shForm];
        temp3Low __ temp3Low and 0FF,                                            c3;

        temp1Low __ temp1Low LRot8,                                              c1;
        temp1Low __ temp1Low or temp3Low,                                        c2;
        uDestDelta __ temp1Low,                                                  c3; {00 - 07:sourceDelta, 08 - 0F: destDelta}

saveDirection:
        Xbus __ uHDir, XHDisp,                                                   c1;
        Xbus __ uVDir, XHDisp, BRANCH [hDirPlusSave, hDirMinusSave, 2],          c2;


hDirPlusSave:
         BRANCH [vDirPlusSave, vDirMinusSave, 2],                                c3;


vDirMinusSave:
        temp2Low __ temp2Low or 40, GOTO [saveMisc],                             c1; {vDir = - 1}


hDirMinusSave:
        temp2Low __ temp2Low or 20, BRANCH [vDirPlusSave, vDirMinusSave, 2],     c3; {hDir = - 1}


vDirPlusSave:
        Noop,                                                                    c1;
saveMisc:
{00 - 03: dy's offset, 04 - 07:skew, 09:vDir = - 1, 0A:hDir = - 1, 0C - 0F: booleans}
        sourceIndex __ sourceIndex or temp2Low, L2 __ bitbltNotFinished,         c2;
        uMisc __ sourceIndex, CALL [restoreStatus],                              c3;

        L0Disp,                                                                  c3, at [bitbltNotFinished, 10, restoreStatus - return];

        XC2npcDisp, DISP2 [ipAjustInBBT],                                        c1;


        CANCELBR [ipAjustedInBBT, 0F],                                           c2, at [0, 4, ipAjustInBBT];
ipAjustedInBBT:
        temp1Low __ 0, GOTO [saveSmalltalkStateBank0],                           c3;


        temp1Low __ 0, BRANCH [pc16One, pc16Zero, 0E],                           c2, at [1, 4, ipAjustInBBT];


pc16One:
        Cin __ pc16, GOTO [saveSmalltalkStateBank0],                             c3;


pc16Zero:
        ipLow __ ipLow - 1, Cin __ pc16, GOTO [saveSmalltalkStateBank0],         c3;


        ipLow __ ipLow - 1, CANCELBR [ipAjustedInBBT, 0F],                       c2, at [2, 4, ipAjustInBBT];


        temp1Low __ 0, BRANCH [pc16One1, pc16Zero1, 0E],                         c2, at [3, 4, ipAjustInBBT];


pc16One1:
        ipLow __ ipLow - 1, Cin __ pc16, GOTO [saveSmalltalkStateBank0],         c3;
```

```
pc16Zero1:
        ipLow _ ipLow - 2, Cin _ pc16, GOTO [saveSmalltalkStateBank0],          c3;


allFinished:
        temp2High _ uRumRecordHigh,                                             c2;
allFinished1:
        temp2Low _ uRumRecordLow,                                               c3;

allFinished3:
        temp2Low _ temp2Low + cursorBitmapOopOffset,                            c1;
        L2 _ bitbltFinished,                                                    c2;
        stackP _ 1, CALL [restoreStatus],                                       c3; { restore the Mesa stack Pointer }

        otLow _ uDestBitMap,                                                    c3, at [bitbltFinished, 10, restoreStatus - return];

        MAR _ [temp2High, temp2Low + 0],                                        c1;
        L1 _ getCursorMap,                                                      c2;
        Q _ MD,                                                                 c3; { get current cursor map }

        [] _ Q xor otLow, ZeroBr,                                               c1;
        BRANCH [noCurrentCursor, yesCurrentCursor],                             c2;


yesCurrentCursor:
        temp3High _ CSBforCursorHigh, CALL [otMap2Bank0],                       c3;

yesCurrentCursor1:
        temp1Low _ temp1Low + sizeFieldOffset,                                  c1, at [getCursorMap, 10, otMap2Bank0 - return];
        temp3Low _ CSBforCursorLow,                                             c2;
        Q _ 10,                                                                 c3;

        MAR _ [temp1High, temp1Low + 0], L1 _ cursorWidth1,                     c1;
        CALL [checkSmallInt],                                                   c2;

        [] _ temp2Low {size} - Q, NegBr,                                        c1, at [cursorWidth1, 10, checkSmallInt - return]; { size <
        16?? }
        temp1Low _ temp1Low + 1, BRANCH [sizeGE10x, sizeLT10x],                 c2; { point the first (??) field of object }


sizeGE10x:
        temp2Low _ 11, GOTO [restoreCursorPattern],                             c3; { loop ocunt = 16'd}


sizeLT10x:
        temp2Low _ temp2Low + 1,                                                c3; { loop count = size of currentcursorbitmap}

restoreCursorPattern:
        temp2Low _ temp2Low - 1, ZeroBr,                                        c1;
        BRANCH [continueRestoreCursor, endRestoreCursor],                       c2;


continueRestoreCursor:
        Noop,                                                                   c3;

        MAR _ [temp3High, temp3Low + 0],                                        c1;
        temp3Low _ temp3Low + 1,                                                c2;
        Q _ MD,                                                                 c3; { get current pattern }

        MAR _ [temp1High, temp1Low + 0],                                        c1;
        MDR _ Q,                                                                c2; { save current pattern to ...}
        temp1Low _ temp1Low + 1, GOTO [restoreCursorPattern],                   c3;


endRestoreCursor:
        GOTO [nextByteCodeInBank0],                                             c3;


noCurrentCursor:
        GOTO [nextByteCodeInBank0],                                             c3;

    { the receiver returns itself, so there is no need for smalltalk stack clean up}




{- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -}
destWidthLess0:
        GOTO [noTransfer1],                                                     c3;


destHeightLess0:
        L2 _ noTransfer, GOTO [noTransfer3],                                    c2;


clipWidthLess0:
        GOTO [noTransfer2],                                                     c1;


clipHeightLess0:
        L2 _ noTransfer, GOTO [noTransfer3],                                    c2;


widthLT0:
        GOTO [noTransfer1],                                                     c3;
```

```
heightLT0:
        GOTO [noTransfer2],                                          c1;

noTransfer1:
        L2 __ noTransfer,                                            c1;

noTransfer2:
        L2 __ noTransfer,                                            c2;

noTransfer3:
        CALL [restoreStatus],                                        c3;

        GOTO [nextByteCodeInBank0],                                  c3, at [noTransfer, 10, restoreStatus - return];

hBitMapNoOop:
        L2 __ primFail, GOTO [primitiveFailBitBlt1],                 c3;

halftoneFormInvalid:
        L2 __ primFail, GOTO [primitiveFailBitBlt3],                 c2;

destFormInvalid:
        L2 __ primFail, GOTO [primitiveFailBitBlt3],                 c2;

sourceFormInvalid:
        L2 __ primFail, GOTO [primitiveFailBitBlt3],                 c2;

primitiveFailBitBlt1:
        Noop,                                                        c1;
primitiveFailBitBlt2:
        Noop,                                                        c2;
primitiveFailBitBlt3:
        stackP __ 1, CALL [restoreStatus],                           c3;

        temp1Low __ 2, GOTO [saveSmalltalkStateBank0],               c3, at [primFail, 10, restoreStatus - return];

destinationNonOop:
        CANCELBR [$, 1],                                             c1;
sourceNonOop:
        CANCELBR [$, 1],                                             c2;
halftoneNonOop:
        L2 __ primFail, CANCELBR [primitiveFailBitBlt1, 1],          c3;

combinationRuleTooBig:
        L2 __ primFail, GOTO [primitiveFailBitBlt1],                 c3;
```

{ Edit history:

22 - Jan - 86 15:55:22	Tokunaga	for stretched
20 - Jan - 86 10:31:02	Tokunaga	exchange uDestBitMap and uCombinationRule, and modify the save and restore ConbinationRule value when MInt occur.
27 - Dec - 85 14:35:33	Tokunaga	refine the routine concerning with saving and restoring copyBits status when mesa Int occur
7 - Nov - 85 18:48:48	Tokunaga	modify the routine for getting the actual address of BitMap Object (calculateOffset)
16 - Oct - 85 16:59:32	Sakakibara	change usource
16 - Oct - 85 16:34:22	Sakakibara	change several points (by Tokunaga)
30 - Sep - 85 15:56:37	Sakakibara	Fix temp1Low value at noPCarryC
27 - Sep - 85 16:34:57	Sakakibara	Added combinationRule = 16 check
27 - Sep - 85 14:00:35	Sakakibara	Bug fix PrimitiveFail
27 - Sep - 85 13:59:42	Matsumoto	Add CANCELBR Mask
13 - Sep - 85 13:28:45	Tokunaga	Add the adjustment for smalltalk instruction pointer when Mesa Int occur during primitiveCopyBits and remove calling updateCursor.
20 - Jun - 85 9:53:33	Tokunaga	remove checkSmallInt to bbtsubs.mc }

{ BBTSubs.mc

Subroutines for BitBlt primitive for Rum, the Dandelion Smalltalk – 80 microcoded virtual machine.

by T Tokunanga, M Sakakibara, J Trow

10 – Feb – 86 20:07:45

{ checkSmallInt

Test MD and convert it to a binary number if it is a SmallInteger. Fail if it is not a SmallInteger.

input: MD is object to be tested
L1 is the return link

output: temp2Low is the converted binary value
temp1Low is incremented

smash: }

```
checkSmallInt:
        temp2Low __ MD, XHDisp,                                                 c3;

        [] __ temp2Low and nonPointerMask, ZeroBr,
                BRANCH [posSmallInteger, negSmallInteger, 2],                   c1;


posSmallInteger:
        temp2Low __ RRot1 temp2Low, BRANCH [oops, notOops],                     c2;


negSmallInteger:
        temp2Low __ RRot1 (temp2Low or 3), BRANCH [oops, notOops],              c2;


notOops:
        temp2Low __ RRot1 temp2Low,                                             c3;

        Noop,                                                                   c1;
        temp1Low __ temp1Low + 1, L1Disp,                                       c2;
        RET [checkSmallInt – return],                                           c3;


oops:
        L2 __ primFail, GOTO [primitiveFailBitBlt1],                            c3;
```

{checkSmallInt2: ********** stretched **********
subroutine for checking smallinteger & convert to normal integer
At entry : c3
At Return : c1
Link register : L1
to be checked : temp2Low
result : temp2Low

Note: temp1Low is incremented.

}

```
checkSmallInt2:
        temp2Low __ MD, XHDisp,                                                 c3;

checkSmallInt2c1:
        [] __ temp2Low and nonPointerMask, ZeroBr,
                BRANCH [posSmallInteger2, negSmallInteger2, 2],                 c1;


posSmallInteger2:
        temp2Low __ RRot1 temp2Low, BRANCH [oops2, notOops2],                   c2;


negSmallInteger2:
        temp2Low __ RRot1 (temp2Low or 3), BRANCH [oops2, notOops2],            c2;


notOops2:
        temp2Low __ RRot1 temp2Low,                                             c3;

        Noop,                                                                   c1;
        temp1Low __ temp1Low + 1, L1Disp,                                       c2;
        RET [checkSmallInt2 – return],                                          c3;


oops2:
        L2 __ primFail, GOTO [primitiveFailBitBlt1],                            c3;
```

```
{
  - - - - ex: mask1 __ RightMasks at: startBits + 1 - - - - - - - - -

   Entry point : c2, c3  Exit point : c3
   the argument is stored in temp3Low , if it isn't between 0 and 10'X then primitiveFail.
   Return link: L3
   Result : temp2Low }
makeRightMasks:
        Noop,                                                          c2;
makeRightMasks3:
        temp2Low __ temp2Low xor temp2Low,                             c3;

{ argument is between 0 and 10'X }
argOKInBBT:
        temp3Low __ temp3Low - 1, NegBr,                               c1; {decrement the shift counter }
        L3Disp, BRANCH [shiftLoop, shiftEnd],                          c2;


shiftLoop:
        temp2Low __ temp2Low LShift1, SE __ 1, CANCELBR [argOKInBBT, 0F],   c3;


shiftEnd:
        RET [makeRightMasks - return],                                 c3;




{ This routine is for checking MesaInterrupt , ST80 BitBlt use this routine.  }

{
 return link : L2
 No Interrupt Exit point: return cycle = c1, pending Xdipatch by uBooleans
                temp3Low : DestDelta
 Interrupt Exit point :  c2}

{
        PC : MesaStatePC,
        pc16 : MesaStatePC16 (pc16 = Bit.15)
        rhPC : MesaStateRhPC
        IBptr: MesaStateIBPtr
        IB   : MesaStateIB

        uPPCross:           0 - - > no Cross
                            ~0 - - > Cross
        UvChigh   : high Address bot od code segment
        UvPCpage: virtual page No of current mesa code
}


{ register definition }
        RegDef [MesaStatePC, U, 50];
        RegDef [MesaStateIBPtr, U, 53];
        RegDef [MesaStateIB, U, 54];

        RegDef [MesaStateRhPC, U, 51];

checkMesaInterrupt:
          temp2Low __ 0F, MesaIntBr,                                   c1;{using if mesa interrupt occur as a stack value}
checkuWP:
        temp3Low __ uWakeupPending, ZeroBr,
                BRANCH [bitBltNoInterrupt, maybeInterrupt],            c2;


bitBltNoInterrupt:
        temp3Low __ uDestDelta, L2Disp, CANCELBR [noInterrupt, 1],     c3;


maybeInterrupt:
        temp3Low __ uDestDelta, L2Disp, BRANCH [bitBltInterrupt, noInterrupt],   c3;


noInterrupt:
        Xbus __ uBooleans, XDisp, RET [noMesaInterrupt - return],      c1;


bitBltInterrupt:
        temp3Low __ MesaStatePC,     CANCELBR [$, 0F],                 c1;
        temp3Low __ temp3Low - 1,                                      c2;
        temp3High __ MesaStateRhPC,                                    c3;

bitBltNoCross:
        MAR __ [temp3High, temp3Low + 0], Xbus __ MesaStateIBPtr, XDisp,   c1;
        MesaStatePC __ temp3Low, DISP4 [bitBltRestoreIB, 0C],          c2;


{Empty}  Noop, GOTO [clearCrossIndicator],                             c3, at [0C, 10, bitBltRestoreIB];
{Byte}   temp3Low __ MD, GOTO [clearCrossIndicator],                   c3, at [0D, 10, bitBltRestoreIB];
{Full} GOTO [bitBltNoCross],                                           c3, at [0E, 10, bitBltRestoreIB]; {forever iteration}
{Word}   temp3Low __ MD, GOTO [clearCrossIndicator],                   c3, at [0F, 10, bitBltRestoreIB];

{there is no check for pc16, bacause 1 or 2 bytes are stored into IB from MesaStateIB coressponding to the po16 respectively in
InitAndMesaStateSaveAndRestore.mc }
```

```
clearCrossIndicator:
        uPCCross __ destAddrLow xor destAddrLow,        L2Disp,                    c1;{uPPCross __ 0}
        MesaStateIB __ temp3Low, RET [mesaInterrupt - return],                     c2;




{
  Subroutine : getSTFormMapBase
   Entry :          otLow = Form Oop (getSTFormMapBase) - - - SourceForm
                    otLow = Form BitMap Oop (getSTFormMapBase1) - - - DestForm
                    L2 : returnLink
  Exit  : temp1High, temp1Low has the map base
            otLow : Oop  for bitmap
            temp2Low : the Words No. for 1 horizontal line,

  cycle: Entry c1, Exit c1
 }

getSTFormMapBase1: { for DESTINATION FORM }
        Xbus __ uBooleans, XLDisp, L3 __ checkSource,                              c3; { destBitMap = current BitMap ? }

        Q __ uDestHeightA, BRANCH [destNEQCurrent, destEQCurrent, 1],              c1;
destNEQCurrent:
{destWidth, destHeight have been already checked whether they would be greater than 1024, 1010 respectively }
        temp2Low __ uDestWidthA, CALL [checkWidthAndHeight],                       c2;


destEQCurrent:
        GOTO [yesCurrent],                                                         c2;


getSTFormMapBase: { for SOURCE FORM }
        temp3Low __ uCurrentDispBitMap, L1 __ getMap,                              c2;
        CALL [otMap2Bank0],                                                        c3;

        MAR __ temp1Low __ [temp1High, temp1Low + formMapBaseIndex],
                L1 __ getWidth,                                                    c1, at [getMap, 10, otMap2Bank0 - return];
getMapBase2:
        temp1Low __ temp1Low + 1,
                BRANCH [noPCInGetMapBase2, yesPCInGetMapBase2, 1],                 c2;


yesPCInGetMapBase2:
        temp1Low __ temp1Low + 0FF,                                                c3;

        MAR __ [temp1High, temp1Low + 0], GOTO [getMapBase2],                      c1;


noPCInGetMapBase2:
        otLow __ MD, XDisp,             { get SourceForm BitMap Oop }              c3;

getMapBase3:
        MAR __ [temp1High, temp1Low + 0], DISP4 [sourceBitMapOop, 0C],             c1; {********}


        [] __ temp3Low xor otLow, ZeroBr, L3 __ checkSource, GOTO [bitMapOop],     c2, at [0D, 10, sourceBitMapOop]; {********}
        [] __ temp3Low xor otLow, ZeroBr, L3 __ checkSource, GOTO [bitMapOop],     c2, at [0E, 10, sourceBitMapOop]; {********}
        [] __ temp3Low xor otLow, ZeroBr, L3 __ checkSource, GOTO [bitMapOop],     c2, at [0F, 10, sourceBitMapOop]; {********}


bitMapOop:
        temp2Low __ MD, XHDisp, BRANCH [{CALL} checkSmallInt2c1, yesCurrent1],    c3;


noCurrent1:
        MAR __ [temp1High, temp1Low + 0], L1 __ getHeight,                         c1, at [getWidth, 10, checkSmallInt2 - return];
        temp3Low __ temp2Low, CALL [checkSmallInt2],                               c2; { for checkWidthAndHeight routine}

noCurrent2:
        Q __ temp2Low, L1 __ getMapBase,                {Q = Form.Height}          c1, at [getHeight, 10, checkSmallInt2 - return];
        temp2Low __ temp3Low, CALL [checkWidthAndHeight],                          c2; {temp2Low = Form.Height}

{ At this point, Q < = = INT(((BitWidth - 1)/16) + 1 )*height, temp2Low = Words /horizontal line.
    Also, otLow = Bitmap Oop(Source, Destination)}

        temp3Low __ Q + objectHeaderSize, CALL [otMap2Bank0],                      c3, at [checkSource, 10, checkWidthAndHeight - return];
        {otLow = BitMap Oop }

noCurrent4:
        MAR __ temp1Low __ [temp1High, temp1Low + sizeFieldOffset],                c1, at [getMapBase, 10, otMap2Bank0 - return];
getMapBase4:
        BRANCH [noPCGetMapBase4, yesPCGetMapBase4, 1],                             c2;


yesPCGetMapBase4:
        temp1Low __ temp1Low + 0FF + 1,                                            c3;

        MAR __ [temp1High, temp1Low + 0], GOTO [getMapBase4],                      c1;


noPCGetMapBase4:
        Q __ MD,                                                                   c3; { get BitMap Size }
```

```
noCurrent5:
        [] __ temp3Low xor Q, ZeroBr, {BitMapSize = (width * height) ? }          c1;
        temp1Low __ temp1Low + 2, { point the actual address of BitMap }
                        BRANCH [funnyBitMapSize, correctBitMapSize],              c2;


correctBitMapSize:
        L2Disp,                                                                   c3;

        RET [getSTFormMapBase – return],                                          c1;


yesCurrent1:
        CANCELBR [$, 0F],                                                         c1;
        Noop,                                                                     c2;
yesCurrent:
        temp2Low __ 40, L2Disp,                                                   c3; { INT [(1024 + 15)/16] = INT [64.9xx] = 64}

yesCurrent2:
        temp1High __ temp1Low __ (temp1Low xor temp1Low) LRot0,
                RET [getSTFormMapBase – return],                                  c1; { point the Low real memory }


        L2 __ primFail, GOTO [primitiveFailBitBlt3],                              c2, at [0C, 10, sourceBitMapOop];


funnyBitMapSize:
        L2 __ primFail, GOTO [primitiveFailBitBlt1],                              c3;




{
 Subroutine : bbtMultiply
   C __ A*B
 Entry :        temp2Low        = A
                Q               = B

                L2              = return link
 Exit:          Q               = result
                temp1Low, temp3Low, sourceIndex are smashed

 entry: c1, c2, c3  exit : c1
 }


bbtMultiply2:
        Noop,                                                                     c2;
bbtMultiply3:
        Noop,                                                                     c3;

bbtMultiply:
        temp1Low __ 0,                                                            c1; { product }
        sourceIndex __ 10,                                                        c2; { loop counter for mult – loop }
bbtMulLoop:
        [] __ Q and 1, NZeroBr,                                                   c3;

        sourceIndex __ sourceIndex – 1, ZeroBr,
                BRANCH [bbtMulDigit0, bbtMulDigit1],                              c1;


bbtMulDigit0:
        temp1Low __ DARShift1 (temp1Low + 0), BRANCH [bbtMulLoop, bbtMulEnd],     c2;


bbtMulDigit1:
        temp1Low __ DARShift1 (temp1Low + temp2Low),
                BRANCH [bbtMulLoop, bbtMulEnd],                                   c2;


bbtMulEnd:
        Q __ ~ Q, L2Disp,                                                         c3;

        RET [bbtMultiply – return],                                               c1; {Result is Q}




{ Subroutine: getSeveralMasks
        description :           calculate the several masks ( skew, mask1, mask2 )
        sourceIndex :           uW – 1
        L1 :                    return Link
 Exit:  skew :                  skew
        uMask1 :                mask1
        uMask2(temp2Low) :      mask2

 Note: smashed register – – temp1Low, temp2Low

 Entry: c1, Exit: c1
 }

getSeveralMasks:
        temp1Low __ uDX,   L3 __ getMask1,                                        c1;
        temp3Low __ temp1Low and 0F,                                              c2; { dx and 15 }
        temp3Low __ 10 – temp3Low,                                                c3; {16 – (dx and 15)}
```

```
        uStartBits __ temp3Low, CALL [makeRightMasks],                          c1;

        uMask1 __ temp2Low,                                                     c1, at [getMask1, 10, makeRightMasks - return];
        sourceIndex __ temp1Low + sourceIndex, L3 __ getMask2,                  c2; { sourceIndex = uW - 1 }
        temp3Low __ sourceIndex and 0F,                                         c3; { (dx + W - 1) bitAnd: 15}

        temp3Low __ 0F - temp3Low, CALL [makeRightMasks],                       c1; { 15 - ((dx + W - 1) bitAnd: 15)}

        uMask2 __ ~temp2Low ,                                                   c1, at [getMask2, 10, makeRightMasks - return];
        temp3Low __ uSX, L3 __ computeSkewMasks,                                c2; { to make skewMask}
        temp3Low __ temp3Low - temp1Low,                                        c3;

        temp3Low __ temp3Low and 0F, ZeroBr, L3 __ computeSkewMasks,            c1;
        skew __ temp3Low LRot0, BRANCH [skewNonZero, skewZero],                 c2;


skewNonZero:
        temp3Low __ 10 - temp3Low,                                              c3;

        sourceIndex __ uBooleans, CALL [makeRightMasks],                        c1;

        uSkewMask __ temp2Low,                                                  c1, at [computeSkewMasks, 10, makeRightMasks - return];
        sourceIndex __ sourceIndex and 87, GOTO [storeSkewZewoIndi],            c2;


skewZero:
        sourceIndex __ uBooleans,                                               c3;

        uSkewMask __ sourceIndex xor sourceIndex,                               c1;
        sourceIndex __ sourceIndex or 8,                                        c2;
storeSkewZewoIndi:
        uBooleans __ sourceIndex, GOTO [getSeveralMasksRet],                    c3;




{restoreStatus
        Entry : c1, c2, c3, Exit: c2
        return Link: L2
}
restoreStatus:
        ipLow __ uSaveIPL,                                                      c1;
        temp1Low __ uSaveIPH,                                                   c2;
        ipHigh __ temp1Low LRot0,                                               c3;

        stackLow __ uSaveStackL,                                                c1;
        stackHigh __ uSaveStackH,                                               c2;
        homeLow __ uSaveHomeL,                                                  c3;

        temp1Low __ uSaveHomeH,    L2Disp,                                      c1;
        homeHigh __ temp1Low LRot0, RET [restoreStatus - return],               c2;




{
Subroutine: bbtCheckRange
Entry:
        destAddrLow       : destX          or destY
        souuceAddrLow     : sourceX        or sourceY
        sourceIndex       : destWisth      or destHeight
        temp2Low          : clipX          or clipY
        temp3Low          : clipWidth      or clipHeight
Exit:
        sourceAddrLow : sx or sy
        destAddrLow  : dx or dy
        sourccelncex : w or h
        returnLink : L2
        entry: c1, exit: c1
}

bbtCheckRange:
        [] __ destAddrLow - temp2Low, NegBr,                                    c1;
        Q __ temp2Low - destAddrLow, BRANCH [destXGE, destXLT],                 c2; {clipX - destX }


destXLT: {no}
        destAddrLow __ temp2Low,                                                c3; { dX __ clipX }

        sourceAddrLow __ sourceAddrLow + Q,                                     c1; { sX __ sourceX + (clipX - destX)}
        sourceIndex __ sourceIndex - Q,                                         c2; { w    __ width - (clipX - destX)}
destXGE: {yes}
        Q __ destAddrLow + sourceIndex,                                         c3; { dx + w }

checkRightRange:
        temp2Low __ temp2Low + temp3Low, L2Disp,                                c1; { clipX + clipWidth }
        [] __ temp2Low - Q, NegBr, BRANCH [checkX, checkY, 0E],                 c2; {(dx + w) > (clipX + clipWidth) ??}


checkX:
        temp1Low __ uDestWidthA, BRANCH [clipGEOnRightSide, clipLTOnRightSide], c3;


checkY:
```

```
        temp1Low __ uDestHeightA, BRANCH [clipGEOnRightSide, clipLTOnRightSide],        c3;

clipGEOnRightSide: { clipX + clipWidth > = dx + w }
        Q __ temp1Low - Q, NegBr,                                                      c1; { dx + w > destWIdth ?}
        BRANCH [rangeLEDestForm1, rangeGTDestForm1],                                   c2;

rangeLEDestForm1:
        GOTO [nextStep1],                                                              c3;

rangeGTDestForm1:
        sourceIndex __ sourceIndex + Q, GOTO [nextStep1],                              c3;

clipLTOnRightSide: { clipX + clipWidth < dx + w }
        [] __ temp1Low - temp2Low, NegBr,                                              c1;
        temp2Low __ Q - temp2Low, {temp2Low = (dx + w) - (clipx + clipWidth)}
                BRANCH [rangeLEDestForm2, rangeGTDestForm2],                           c2;

rangeLEDestForm2:
        sourceIndex __ sourceIndex - temp2Low, GOTO [nextStep1],                       c3;

rangeGTDestForm2:
        Q __ Q - temp1Low, { Q = (dx + w) - Form.Width }                               c3;

rangeLTDestForm21:
        sourceIndex __ sourceIndex - Q, GOTO [nextStep],                               c1;

nextStep1:
        Noop,                                                                          c1;
nextStep:
        Xbus __ uBooleans, XLDisp,                                                     c2; { check DestForm = currentScreen?}
        BRANCH [noCurrentInCheck, yesCurrentInCheck, 1],                               c3; { dx < 0 ?? }

yesCurrentInCheck:
        [] __ destAddrLow, NegBr,                                                      c1;
        BRANCH [destNoNegative, destYesNegative],                                      c2;

{ if destForm = currentScreen, and dx(dy) < 0, we have to make dx(dy) = 0 and ajust the sx(sy), w(h) }
destYesNegative:
        sourceAddrLow __ sourceAddrLow - destAddrLow,                                  c3;

        sourceIndex __ sourceIndex + destAddrLow,                                      c1;
        destAddrLow __ 0,                                                              c2;
destNoNegative:
        Noop,                                                                          c3;

noCurrentInCheck:
        [] __ sourceAddrLow, NegBr,                                                    c1;
        BRANCH [noSourceNegative, yesSourceNegative],                                  c2;

noSourceNegative:
        L2Disp,                                                                        c3;

        RET [bbtCheckRange - return],                                                  c1;

yesSourceNegative:
        destAddrLow __ destAddrLow - sourceAddrLow,                                    c3;

        sourceIndex __ sourceIndex + sourceAddrLow,                                    c1;
        sourceAddrLow __ sourceAddrLow xor sourceAddrLow,
                GOTO [noSourceNegative],                                               c2;



{ Subroutine: calSkewWord                - - 4 click }
{
 Entry    : c1
        sourceIndex : prevWord
        Q : uHDir

 Exit     : c3
        temp2Low : skewWord
        Q :  New prevWord for next using

 otLow : smashed

}

calSkewWord:
        MAR __ [sourceAddrHigh, sourceAddrLow + 0],                                    c1;
calSkewWord1:
        sourceAddrLow __ sourceAddrLow + Q,                                            c2;
        Q __ MD,                                                                       c3; { get this word }

        uPrevWord __ Q,                                                                c1;
```

```
        Noop,                                                   c2;
        Noop,                                                   c3;

calSkewWord2:
        sourceIndex _ sourceIndex and uSkewMask,                c1; { prevWord _ prevWord bitAnd: skewMask }
        Q _ ~uSkewMask and Q,                                   c2;{ thisWord bitAnd: skewMaks bitInvewrt }
calSkewWord3:
        Q _ sourceIndex or Q, Xbus _ skew, XDisp,               c3;

bbtRotation:
        sourceIndex _ LRot1 Q, DISP4 [bbtRot],                  c1;


        L2Disp, temp2Low _ Q, GOTO [bbtShift0],                 c2, at [ 0, 10, bbtRot];
        L2Disp, temp2Low _ sourceIndex, GOTO [bbtShift0],       c2, at [ 1, 10, bbtRot];
        L2Disp, temp2Low _ LRot1 sourceIndex , GOTO [bbtShift0], c2, at [ 2, 10, bbtRot];
        L2Disp, temp2Low _ RRot1 Q, GOTO [bbtShift4],           c2, at [ 3, 10, bbtRot];

        L2Disp, temp2Low _ Q, GOTO [bbtShift4],                 c2, at [ 4, 10, bbtRot];
        L2Disp, temp2Low _ LRot1 Q, GOTO [bbtShift4],           c2, at [ 5, 10, bbtRot];
        L2Disp, temp2Low _ LRot1 sourceIndex, GOTO [bbtShift4], c2, at [ 6, 10, bbtRot];
        L2Disp, temp2Low _ RRot1 Q, GOTO [bbtShift8],           c2, at [ 7, 10, bbtRot];

        L2Disp, temp2Low _ Q, GOTO [bbtShift8],                 c2, at [ 8, 10, bbtRot];
        L2Disp, temp2Low _ LRot1 Q, GOTO [bbtShift8],           c2, at [ 9, 10, bbtRot];
        L2Disp, temp2Low _ LRot1 sourceIndex, GOTO [bbtShift8], c2, at [0A, 10, bbtRot];
        L2Disp, temp2Low _ RRot1 Q, GOTO [bbtShift12],          c2, at [0B, 10, bbtRot];

        L2Disp, temp2Low _ Q, GOTO [bbtShift12],                c2, at [0C, 10, bbtRot];
        L2Disp, temp2Low _ LRot1 Q, GOTO [bbtShift12],          c2, at [0D, 10, bbtRot];
        L2Disp, temp2Low _ LRot1 sourceIndex, GOTO [bbtShift12], c2, at [0E, 10, bbtRot];
        L2Disp, temp2Low _ RRot1 Q, GOTO [bbtShift0],           c2, at [0F, 10, bbtRot];


bbtShift0:
        RET [calSkewWord - return],                             c3;


bbtShift4:
        temp2Low _ temp2Low LRot4, RET [calSkewWord - return],  c3;


bbtShift8:
        temp2Low _ temp2Low LRot8, RET [calSkewWord - return],  c3;


bbtShift12:
        temp2Low _ temp2Low LRot12, RET [calSkewWord - return], c3;




{Subroutine: getHalftoneWord  - - - - - - 2 - clicks}
{
 Entry:
        c2,
        sourceIndex : vertical direction
 Exit:
        c1,
        sourceIndex: halftoneWord
}

getHalftoneWord:
        temp2Low _ uDY,                                         c2;
        temp3Low _ temp2Low and 0F,                             c3; { dy bitAnd: 15}

        MAR _ [halftoneAddrHigh, halftoneAddrLow + temp3Low],   c1;
        temp2Low _ temp2Low + sourceIndex,
                BRANCH [noPCInGetHWord, yesPCInGetHWord, 1],    c2;{ dy _ dy + vDir}


yesPCInGetHWord:
        halftoneAddrLow _ temp3Low + halftoneAddrLow,           c3;

        MAR _ [halftoneAddrHigh, halftoneAddrLow + 0],          c1;
        halftoneAddrLow _ halftoneAddrLow - temp3Low,           c2;
noPCInGetHWord:
        temp2Low _ MD, uDY _ temp2Low, L1Disp,                  c3;

        uHalftoneWord _ temp2Low, RET [getHalftoneWord - return], c1;




{
Subroutine: checkWidthAndHeight

 Entry:         temp2Low        = Form.Width
                Q               = Form.Height
                L3              = return Link

 Exit :         Width * Height > 64640 then primitiveFail, otherwise return

 Note :         temp1Low, otLow, Q, sourceIndex  is smashed by using bbtMultiply routine.
 }
```

```
checkWidthAndHeight:
        temp2Low __ temp2Low - 1,                                   c3;

checkWidthAndHeight1:
        temp2Low __ temp2Low and ~0F, L2 __ widthAndHeight,         c1;
        temp2Low __ temp2Low LRot12 + 1,                            c2; { width __ INT((bitWidth - 1)/16) + 1}
        CALL [bbtMultiply],                                         c3; { Q = width * height, temp1Low, sourceIndex are
        smashed}

{On Smalltalk - 80 on 1108X(klku - X), the largest size of BitMap is 64640 word, i.e. 1024*1010/16. If user try to create the form of size
with greater than 64640, automativally system modify the size with 64640, and not modify the width, height of the corresponding Form. So
we may have the trouble, since copyBits primitive refers Form.width, height when actually transferring the Bit Block.}
{Why the reason above, we check it describing below}
        [] __ Q, NegBr,                                             c2, at [widthAndHeight, 10, bbtMultiply - return];
        temp1Low __ 0FC,
                BRANCH [checkWidthAndHeight4, maybeLargerThanMaxSize],   c3;


maybeLargerThanMaxSize:
        temp1Low __ temp1Low LRot8,                                 c1;
        temp1Low __ temp1Low or 80,                                 c2; {64640 = FC80'x = 1024*1010/16}
        [] __ temp1Low - Q, NegBr,                                  c3;

        L3Disp, BRANCH [checkWidthAndHeight41, overBitMapMaxSize],  c1;


checkWidthAndHeight4:
        L3Disp,                                                     c1;
checkWidthAndHeight41:
        RET [checkWidthAndHeight - return],                         c2;


overBitMapMaxSize:
        temp2Low __ 0F1, CANCELBR [$, 0F],                          c2;
        L2 __ primFail, GOTO [primitiveFailBitBlt1],                c3;
```

{ Edit history:

22 - Jan - 86 16:17:09	Tokunaga.iwafx	modify getSTFormMapBase for stretch
21 - Jan - 86 16:46:27	Tokunaga.iwafx	modify the checkWidthAndHeight and getSTFormMapBase
17 - Dec - 85 9:00:16	Tokunaga.iwafx	add the checking routine for dx < 0 or not in bbtCheckRange when DestForm = CurrentScreen
5 - Nov - 85 9:12:07	Tokunaga.iwafx	add the checking routine for BitMap.Size = (Width*Height) in getSTFormMapBase when bitMap oop is not current display bitMap.
4 - Nov - 85 11:31:18	Tokuanga.iwafx	add the checking DestForm.width in bbtCheckRange routine.
2 - Nov - 85 20:09:58	Tokunaga.iwafx	add checkSmallInt2 and checkWidthAndHeight routine
27 - Sep - 85 13:49:07	Sakakibara.iwafx	bug fix CANCELBR
13 - Sep - 85 11:36:53	Tokuanga.iwafx	remove "updateCursor" routine and RegDefs in checkMesaInterrupt routine.
20 - Jun - 85 9:52:54	Tokunaga.iwafx	add checkSmallInt from bbt.mc
10 - Jun - 85 19:38:06	Tokunaga.iwafx	add bitShift subroutine }

```
{ BBT.dfn

Definitions for bitblt in Rum, the Dandelion Smalltalk – 80 microcoded virtual machine.

        by T Tokunaga, J Trow

        9 – Feb – 86 17:33:52

Copyright 1985, 1986 by Xerox Corporation.  All rights reserved. }



{ R, RH register }

        RegDef [sourceIndex,            R, 0];    RegDef [skew,               RH, 0];
        RegDef [destAddrLow,            R, 1];    RegDef [destAddrHigh,       RH, 1];
        RegDef [sourceAddrLow,          R, 2];    RegDef [sourceAddrHigh,     RH, 2];

        RegDef [halftoneAddrLow,        R, 4];    RegDef [halftoneAddrHigh,   RH, 4];



{ U register }

        { ******** Mesa Stack ********** }
        RegDef [uDestBitMap,            U, 2];
        RegDef [uH,                     U, 3];
        RegDef [ul,                     U, 3];
        RegDef [uDestAddrLow,           U, 4];
        RegDef [uDestWidthA,            U, 4];
        RegDef [uDestAddrHigh,          U, 5];
        RegDef [uDestHeightA,           U, 5];
        RegDef [uDestDelta,             U, 6];
        RegDef [uMask1,                 U, 7];
        RegDef [uMask2,                 U, 8];
        RegDef [uSkewMask,              U, 9];
        RegDef [uCurrentDispBitMap,     U, 9];
        RegDef [uMisc,                  U, 0A];
        RegDef [uHalftoneAddrLow,       U, 0B];
        RegDef [uSourceAddrLow,         U, 0C];
        RegDef [uSaveHighAddr,          U, 0D];
        RegDef [uNWordsM1,              U, 0E];

        {********** U block 1 ************ }
        RegDef [uSaveIPL,               U, 14]; { save smalltalk instruction pointer }
        RegDef [uSaveIPH,               U, 17];
        RegDef [uSaveStackL,            U, 19]; { save smalltalk stack pointer }
        RegDef [uSaveStackH,            U, 1A];
        RegDef [uSaveHomeL,             U, 1B]; { save smalltalk home context pointer }
        RegDef [uSaveHomeH,             U, 1D];
        RegDef [uArgument,              U, 1E]; { combinatin Rule }

        {********** U block 2 ************ }
        RegDef [uSourceDelta,           U, 24];

        {********** U block 3 ************ }
        RegDef [uClipHeight,            U, 31];
        RegDef [uDestMap,               U, 31];
        RegDef [uClipWidth,             U, 34];
        RegDef [uBBTTemp,               U, 34];
        RegDef [uStartBits,             U, 35];
        RegDef [uClipY,                 U, 35];
        RegDef [uW,                     U, 37];
        RegDef [uDX,                    U, 38];
        RegDef [uBooleans,              U, 3C];

        {********** U block 4 ************ }
        RegDef [uCombinationRule,       U, 42];
        RegDef [uDestForm,              U, 43];
        RegDef [uSourceForm,            U, 44];
        RegDef [uSX,                    U, 47];
        RegDef [uHalftoneForm,          U, 4B];
        RegDef [uDestX,                 U, 4C];
        RegDef [uDestY,                 U, 4D];
        RegDef [uSkewWord,              U, 4E];
        RegDef [uSourceX,               U, 4F];
        RegDef [uPrevWord,              U, 4F];

        {********** U block 5 ************ }
        RegDef [uDY,                    U, 55];
        RegDef [uVDir,                  U, 58];
        RegDef [uHDir,                  U, 5C];
        RegDef [uSY,                    U, 5D];
        RegDef [uHalftoneWord,          U, 5D];

        {********** U block 6 ************ }
        RegDef [uSourceY,               U, 63];
        RegDef [uMergeMask,             U, 63];
        RegDef [uDestWidth,             U, 6A];
        RegDef [uWord,                  U, 6A];
        RegDef [uDestHeight,            U, 6B];
        RegDef [uNWords,                U, 6B];
```

```
        RegDef [uClipX,                 U, 6D];
        RegDef [uHM1,                   U, 6F];



{************************************************************
 **********                constant                **********
 ************************************************************ }

        Set [CSBforCursorHigh,          2];
        Set [CSBforCursorLow,           0F0];

        Set [destFormIndex,             firstFieldOfObject];
        Set [diffDFAndHF,               2];

        Set [diffCurrentBitMapAndCursor, Sub [cursorBitmapOopOffset, displayBitmapOopOffset]];
        Set [halftoneFormIndex, Add [firstFieldOfObject, 2]];

        Set [difFieldAndSize, Sub [firstFieldOfObject, sizeFieldOffset]];
        Set [firstFieldOfObjectM1, Sub [firstFieldOfObject, 1]];


{ getSTFormMapBase : L2     }
        Set [getDestBitsAfterMInt,      0];
        Set [getDestBits,               1];
        Set [getSourceBits,             2];

        Set [formMapBaseIndex, firstFieldOfObject];

        Set [formMapBitsNoIndex, Add [1, firstFieldOfObject]];

        Set [formBitsIndex, firstFieldOfObject];
        Set [formWidthIndex, Add [1, formBitsIndex]];
        Set [formHeightIndex, Add [1, formWidthIndex]];;


{ checkSmallInt: L1}
        Set [checkCombinationRule,      0];
        Set [checkDestX,                1];
        Set [checkDestY,                2];
        Set [checkDestWidth,            3];
        Set [checkDestHeight,           4];
        Set [checkSourceX,              5];
        Set [checkSourceY,              6];
        Set [checkClipX,                7];
        Set [checkClipY,                8];
        Set [checkClipWidth,            9];
        Set [checkClipHeight,           0A];
        Set [checkSourceFormHeight,     0B];
        Set [checkSourceFormWidth,      0C];
        Set [cursorWidth,               0D];
        Set [getWidth,                  0E];
        Set [cursorWidth1,              0F];

{ checkSmallInt2: L1}
        Set [checkDestFormWidth,        0];
        Set [checkDestFormHeight,       1];
        Set [getWidth,                  2];
        Set [getHeight,                 3];


{checkWidthAndHeight: L3 }
        Set [checkClip,                 0];
        Set [checkDest,                 1];
        Set [checkSource,               2];


{makeRightMasks}
        Set [getMask1Again,             0];
        Set [getMask2Again,             1];
        Set [getMask1,                  2];
        Set [getMask2,                  3];
        Set [computeSkewMasks,          4];


{bbtMultiply: L3}
        Set [getSourceMul,              0];
        Set [getDestMul,                1];
        Set [widthAndHeight,            2];


{getSeveralMasks  - - -  get skew, mask1, mask2}
        Set [getMasks,                  0];
        Set [getMasksAgain,             1];


{checkFormNil}
        Set [checkSourceFormAgain,      0];
        Set [checkHalftoneFormAgain,    1];


{bitBBTShift}
        Set [bitShiftSkewP,             0];
        Set [bitShiftSkewM,             1];
```

```
{checkMInt}
        Set [bbtCheckMInt,              0];

{updateCursor}
        Set [cursorAndDest,             0];
        Set [cursorAndSource,           1];

{restoreStatus}
        Set [bitbltFinished,            0];
        Set [bitbltNotFinished,         1];
        Set [primFail,                  7]; {must be odd}
        Set [noTransfer,                3];

{ checkRange: L2 }
        Set [checkXRange,               0]; {must be even}
        Set [checkYRange,               1]; {must be odd}

{calSkewWord: L3 }
        Set [skewWord,                  0];
        Set [skewWord1,                 1];
        Set [comb0D,                    2];

{write: L2 }
        Set [comb0D01,                  0];
        Set [comb0D02,                  1];
        Set [comb0D03,                  2];
        Set [comb0D04,                  3];
        Set [comb0D06,                  4];
        Set [comb0D07,                  5];
        Set [comb0D08,                  6];
        Set [comb0D09,                  7];
        Set [comb0D0B,                  8];
        Set [comb0D0C,                  9];
        Set [comb0D0D,                  0A];
        Set [comb0D0E,                  0B];

{mergeSrcAndDest: L2 }
        Set [hP1SFormNil,               1];
        Set [hP1NoLast,                 0];
        Set [hP1NoLastSFNonNil,         2];
        Set [hM1SFormNil,               5];
        Set [hM1NoLast,                 4];
        Set [hM1SFormNonNil,            9];
        Set [hM1NoLastSFNonNil,         8];
        Set [hP1SFormNonNil,            0D];



{ Edit history:

22 - Jan - 86 15:41:19      Tokunaga.fx     add constant for checksmallInt1, checkWidthAndHeight }
```

```
*-----------------------------------------------------------------------
Title[AltoBitBlt.mc...July 3, 1981  4:54 PM...P. Deutsch];
*          26 September 1984 1:15:50 pm PDT (Wednesday) gk mode 5/10. fix
* Modified from version April 17, 1981  7:32 PM  Taft,
* to include Smalltalk mode and remove pre – touching of pages
* Bit – boundary block – transfer
* This version emulates Alto and Smalltalk BitBlt,
* including the use of long pointers. (9 December 1982 1:27:04 pm )
*-----------------------------------------------------------------------

%
Refer to the Alto Hardware Manual for primary documentation.
All numbers are octal except timings, which are decimal.

BBTable format:
        0       Function (see below)
        1       unused
        2       DBCA    destination base core address
        3       DBMR    destination bit map raster
        4       DLX     destination left X
        5       DTY     destination top Y
        6       DW      destination width (also source width)
        7       DH      destination height (also source height)
        10      SBCA    source base core address
        11      SBMR    source bit map raster
        12      SLX     source left X
        13      STY     source top Y
        14      Gray0   (for Alto BitBlt; ignored for Smalltalk BitBlt)
        15      Gray1
        16      Gray2
        17      Gray3
        20      LSBCAlo long pointer to source bit map
        21      LSBCAhi
        22      LDBCAlo long pointer to destination bit map
        23      LDBCAhi
        24      LGBCAlo long pointer to gray bit map (Smalltalk only)
        25      LGBCAhi

Function:
        B0 = 1  Use the long pointers in words 20 – 23 and ignore DBCA, SBCA
        B1 = 1  Use Smalltalk operation codes, and gray pointer in words 24 – 25

                Following 2 bits defined only if NOT using long pointers:
        B10 = 1 Source block is in the alternate bank (XM)
        B11 = 1 Destination block is in the alternate bank (XM)

                Following bits are defined only in Alto mode:
        B12:13  SourceType
        B14:15  Operation
                Following bits are defined only in Smalltalk mode:
        B8:9    SourceType:
                B8      Source is supplied (otherwise all ones)
                B9      Gray is supplied (otherwise all ones)
        B12:15  Rule for combining dest __ f(dest, source AND gray):
                B12     dest __ 1 if (NOT dest) AND NOT (source AND gray)
                B13     dest __ 1 if dest AND NOT (source AND gray)
                B14     dest __ 1 if (NOT dest) AND (source AND gray)
                B15     dest __ 1 if dest AND (source AND gray)


The 20 BitBlt Functions (combinations of SourceType and Operation) are
divided into 6 classes:
        A       dest __ gray
        B       dest __ f(gray, dest)
        C       dest __ f(source)
        D       dest __ f(source, gray)
        E       dest __ f(source, dest)
        F       dest __ f(source, gray, dest)

The distribution of functions into classes for Alto mode is:
    Function (class)        SourceType              Operation
        0 (C)               0 source                0 dest __ source
        1 (E)               0 source                1 dest __ source OR dest
        2 (E)               0 source                2 dest __ source XOR dest
        3 (E)               0 source                3 dest __ NOT source AND dest
        4 (C)               1 NOT source            0 dest __ source
        5 (E)               1 NOT source            1 dest __ source OR dest
        6 (E)               1 NOT source            2 dest __ source XOR dest
        7 (E)               1 NOT source            3 dest __ NOT source AND dest
        10 (D)              2 source AND gray       0 dest __ source
        11 (F)              2 source AND gray       1 dest __ source OR dest
        12 (F)              2 source AND gray       2 dest __ source XOR dest
        13 (F)              2 source AND gray       3 dest __ NOT source AND dest
        14 (A)              3 gray                  0 dest __ source
        15 (B)              3 gray                  1 dest __ source OR dest
        16 (B)              3 gray                  2 dest __ source XOR dest
        17 (B)              3 gray                  3 dest __ NOT source AND dest

In Smalltalk mode, rules 0 (all zeroes), 17 (all ones), 3/14 (dest/not dest),
and 5/12 (source/not source) require special treatment; all other rules imply
a function that uses both the dest and source operands.
The degenerate case of neither source nor gray is handled by
constructing a fake 1 – word gray block containing all ones.
Thus the distribution of functions into classes for Smalltalk mode is:
    Rule (class)            SourceType
        0, 17 (A)           0 all ones
        others (B)          0 all ones
```

0, 17 (A)	1 gray
3, 14 (B)	1 gray
5, 12 (A)	1 gray
others (B)	1 gray
0, 17 (A)	2 source
3, 14 (B)	2 source
5, 12 (C)	2 source
others (E)	2 source
0, 17 (A)	3 source AND gray
3, 14 (B)	3 source AND gray
5, 12 (D)	3 source AND gray
others (F)	3 source AND gray

If DLX < = SLX, the BitBlt horizontal loop works left – to – right; if DLX > SLX, right – to – left. Similarly, if DTY < STY, the vertical loop works top – to – bottom; if DTY > = STY, bottom – to – top. This is so that the correct thing will happen if source and destination blocks overlap. (Note that this test depends on the assumption that if the blocks overlap, they belong to the same bit map, i.e., DBCA = SBCA and DBMR = SBMR. No check is made for this.) The " = " cases could be handled either way; however, the DTY = STY case must be handled bottom – to – top because the documented algorithm for proper phasing of the gray block depends on this.

Terminology: when referring to words in a scan line, "left" and "right" refer to the words with lower and higher addresses, respectively, independent of the direction of processing the scan line; "first" and "last" refer to the first and last words encountered in the direction of processing.

The destination block is considered to consist of a left partial word, some number of full (body) words, and a right partial word. If the block begins or ends on a word boundary, the left or right word is still considered to be a partial word.

The destination bits preserved in the left and right partial words are determined by the SHC register's LMask and RMask, respectively, where LMask = DLX mod 20, RMask = 17 – ((DLX + DW – 1) mod 20).

The destination width in words, including the first and last partial words, is computed by DWidth = (DW + (DLX mod 20) + 17)/20. Similarly, the source width in words, including the first and last partial words, is computed by SWidth = (DW + (SLX mod 20) + 17)/20.
DWidth and SWidth differ by at most +/ – 1.

If DWidth > 2 then the destination consists of a left partial word, DWidth – 2 full body words, and a right partial word.
If DWidth = 2 then the block consists of a left partial word immediately followed by a right partial word.
If DWidth = 1 then the entire destination block lies within a single word, not crossing a word boundary. Effectively, the left and right partial words are one and the same. This case (called the "thin" case) requires special handling, as both LMask and RMask must be applied simultaneously.

Processing of the scan line is controlled by two counters: the Cnt register (loaded from lCnt), which counts the inner loop (20 words or fewer), and MCount, which counts the outer loop. Roughly speaking,
Cnt is loaded initially with DWidth mod 20 and MCount with DWidth/20.
The inner loop is executed Cnt times; then, until MCount is exhausted, Cnt is reloaded with 20 and the inner loop is executed 20 more times.
The reason for this arrangement is to permit a PreFetch to be executed every 20 words.

More precisely: Before each horizontal loop, the Cnt register is loaded with
if DWidth < 23 then DWidth – 1 else (DWidth – 3) mod 20 + 2, so:
Cnt > 1 if the body contains one or more words.
Cnt = 1 if there are no body words.
Cnt = 0 in the thin case.
Cnt is decremented twice before the main loop is reached, so upon entry to the main loop it contains DWidth – 3 (assuming DWidth < 23), which is precisely the correct value for going around the loop DWidth – 2 times. (The extra 2 in DWidth are the first and last partial words, which are handled outside the word loop.)

MCount is loaded with (DWidth – 3)/20 – 1, so:
MCount < 0 if DWidth < 23
MCount = 0 if 23 < = DWidth < 42, etc.
Each time the inner loop terminates, if MCount<0 the scan line is finished, but if MCount > = 0, MCount is decremented, Cnt is loaded with 17, and the main loop is reentered for another 20 iterations.

When working left – to – right, the source may be thought of as T,,SrcWd – – that is, with T on the left and SrcWd on the right. In any given operation, T contains the previous source word and SrcWd contains the current one, and the shift operation consists of left – shifting T,,SrcWd by 0 to 17 bits (to align it with the destination) and storing the leftmost 20 bits of the result. Since the shifter is actually a cycler and produces the rightmost 20 bits (rather than the leftmost) of the cycled result, the shifter must be set up to left – cycle an additional 20 bits (by exchanging SHA and SHB). Hence SHA = R and SHB = T.

When working right – to – left, the source may be thought of as SrcWd,,T, where T contains the previous source word and SrcWd the current, as before. This is exactly symmetric with the left – to – right case and requires only that SHA and SHB be exchanged. Hence SHA = T and SHB = R.

Regardless of the direction, if there are at least as many data bits in the first word of the source block as in the first word of the destination block, then all the destination bits come from SrcWd during the first shift and T need not be loaded at all. On the other hand, if there are fewer bits in the first word of the source block, the destination bits come from both SrcWd and T. In this case, T must contain the first source word and SrcWd the second. These two situations must be distinguished, since one requires fetching only a single word whereas the other requires fetching two words before operating the shifter for the first time.

A similar situation arises at the end of a block; that is, all the bits in the last partial destination block may come from the leftover source word in T, or it may be necessary to fetch an additional word into SrcWd. These situations must be distinguished also. An earlier version of this microcode omitted the test and simply fetched the "extra" word always; unfortunately, this sometimes resulted in touching a word outside the bit map, which caused problems in a paged environment when the bit map happened to be page – aligned.

Two bits in BBFlags control the fetching of the "extra" word in the above two cases.

All the horizontal loops work in both directions by use of a trick: ALUF[15] is redefined to be A + 1 if moving left – to – right but A – 1 if right – to – left. This ALU function is invoked by the operation "A +/– 1", and is used to advance source and destination pointers along the scan line. A consequence of this is that ALUF[15] (normally A AND NOT B) cannot be used by I/O tasks.

BBDst and BBSrc are 16 – bit displacements relative to base registers BBDstBR and BBSrcBR. If moving top – to – bottom they start near zero and count up; if bottom – to – top they start at ~2^15 and count down. If more than ~2^15 words of bit map are processed, one of these displacements becomes negative. This is detected and causes BitBlt to restart, recomputing the base registers and displacements. BR displacement overflow is handled this way because it takes too much code to recompute the BRs and displacements in mid – stream. The cost of restarting BitBlt from scratch in this case is unimportant, since such a large amount of data is involved. (Indeed, in normal programs an interrupt will occur before 2^15 words are processed, so BitBlt will be restarted anyway.)

```
*---------------------------------------------------------------------
* BitBlt Calling Sequence
*---------------------------------------------------------------------

The BitBlt subroutine is called by:
        SCall[BitBltSub]
with the Top-of-Stack (TOS) containing the following arguments:
        TOS-1:  pointer to BBTable (in current space)
        TOS:    0

Assumes RBase[AEmRegs] when called and leaves it that way upon return.

Assumes that the BBTable is relative to the MemBase in effect at the time
of the call.  The caller shouldn't assume anything about MemBase upon return.

BitBlt exits in one of two ways:

1. Normal completion: BitBlt sets TOS to zero and returns to caller + 2.

2. Interrupt request pending: BitBlt sets TOS to the number of scan lines
already processed and returns to caller + 1.  The calling emulator processes
the interrupt and then restarts the BitBlt with the value left in TOS.
Note: if the BitBlt source or destination block is larger than 2^15 words,
BitBlt stops after processing at most 2^15 words and returns + 1, exactly
as if an interrupt had occurred.

Local stack usage:
        TOS-1:  BBTable
        TOS:    number of scan lines completed
        TOS+1:  not used (preserved for Alto Emulator's benefit)
        TOS+2:  return Link
        TOS+3:  saved ALUFM[17]
        TOS+4:  saved ALUFM[15]

Approximate timing for initialization and cleanup (excluding main loops):
          69 cycles (minimum)
          + 4 if working bottom-to-top
        + 21 if DTY # 0 or working bottom-to-top
          + 1 if DWidth = 1 or DWidth > 22B
Add the following if a source bit map is required:
        + 31
          + 5 if working right-to-left
          + 4 if working bottom-to-top
        + 21 if STY # 0 or working bottom-to-top
          + 5 if DWidth = SWidth
          + 1 if SWidth = 1
          + 2 if STY = DTY and SBMR = DBMR

          95 cycles maximum in destination-only cases
        164 cycles maximum in source-destination cases

See comments above main loops for main loop times.

These timings are for Alto mode; Smalltalk mode is a little slower
for the initial setup.
%
```

```
*-----------------------------------------------------------------------
* R-register assignments:
*-----------------------------------------------------------------------

SetRMRegion[BBRegs];        * The RMRegion itself is defined in RegisterDefs.mc

RVN[BBDst];                 * Address of next word to process
RVN[BBSrc];

RVN[DstInc];                * Address increment between scan lines - -
RVN[SrcInc];                *  negative if working bottom - to - top

RVN[DRast];                 * Raster length (words) - -
RVN[SRast];                 *  negative if working bottom - to - top

RVN[PrefDst];               * Address of next munch to PreFetch
RVN[PrefSrc];

RVN[DPrefOffset];           * Offset of leftmost word of next scan line relative
RVN[SPrefOffset];           *  to first word of current scan line

RVN[VCount];                * Vertical line count
RVN[MCount];                * Horizontal munch count
RVN[ICnt];                  * Initial value of Cnt register for word loops

RVN[BBDispl;                * Control flags and horizontal loop dispatch
                            * B0 = 1 = > work right - to - left
                            * B8:15 = dispatch value relative to HorizontalDisp
                            * (see below for details)

RVN[BBFlags];               * Control flags:
                            * B0 = 1 = > 2 source words required for first dest word
                            * B15 = 1 = > 2 source words required for last dest word

RVN[SrcWd];                 * Leftover source word - - must be RVREL 17

* Additional registers, overlaid with emulator temps in the AEmRegs block.
RME[BBRule, ETemp0];        * Low 4 bits of function word (during initialization only)
RME[GrayDirection, ETemp0];         * 0 if working top - to - bottom, - 1 if bottom - to - top
RME[GrayAddrMask, ETemp1];          * Mask for low bits of gray offset:
                            * 3 if Alto mode, 17 if Smalltalk mode and SourceType
                            * not zero, 0 if Smalltalk mode and SourceType = 0
RME[GBCAlo, ETemp2];        * long pointer to gray block
RME[GBCAhi, ETemp3];
RME[GrayOffset, ETemp4];            * Offset within gray of first scan line

* Aliases used during initialization
RME[Width, DPrefOffset]; * Width of block in bits
RME[DWidth, ICnt];          * Width of destination block in words
RME[SWidth, PrefSrc];       * Width of source block in words
RME[Skew, DstInc];          * Destination - source skew, mod 20
RME[BBMasks, PrefDst];      * LMask and RMask values to be loaded into SHC
RME[BBFunc, SrcInc];        * BitBlt function word
RME[DstX, BBDst];           * Destination starting X in bits, later in words
RME[SrcX, BBSrc];           * Source starting X in bits, later in words
RME[DstY, SPrefOffset];     * Destination starting Y in scan lines
RME[SrcY, SRast];           * Source starting Y in scan lines
RME[BBTemp, SrcWd];         * Must be RVRel 17 because it is an arg to MulSub.
```

```
*--------------------------------------------------------------------
* Other definitions
*--------------------------------------------------------------------

* Base -- register assignments

% -- Actually defined in ADefs.mc.  First two must be an even - odd pair.
BR[BBDstBR, ?];                 * BitBlt destination base
BR[BBSrcBR, ?];                 * BitBlt source base
BR[ScratchBR, ?];               * Scratch (emulator use only)
%


* ALU functions defined by BitBlt.
* The ALUF Ram is loaded by BitBlt with the desired operations.
XALUOP[,BBOp,,17,E];            * A BBOp B -- logical operation invoked with shifter

XAOP[, +/-1,15,E];              * A +/-1 -- A + 1 or A - 1 depending on horizontal
                                * direction.  This value of ALUF is normally
                                * A AND NOT B and is restored by BitBlt when done.
                                * This means, however, that A AND NOT B cannot be used
                                * by other tasks.


* Layout of BBDisp register:
* B0 = 0 if working left - to - right, 1 if right - to - left
* B8 - 15: BigBDispatch value for setup and body dispatches.
* The following addressing constraints apply:
* (1) B9 = 1 and B15 = 1 if a source bit map is used.
* (2) B12 = 1 and B13 = 0 if gray is used; B12 = 0 and B13 = 1 otherwise.
* (3) B11 = 1 if the destination is an operand.
* (4) Certain targets are tied together by Call constraints.
* These bits are rather carefully selected to permit the same BBDisp to be
* used in three different dispatches.

Set[SrcFlg, 10];
Set[NoGrayFlg, 4];
Set[GrayFlg, 10];
Set[DstFlg, 20];

* BBDispX defines the first of two pages used for dispatches on BBDisp.
* BBDispX must refer to an even page.
Set[BBDispX, 1200];
M[BBAt, At[BBDispX, Add[#1, #2]]];

* Page -- relative entry points for setup routines.
* All must have B11 = 1, to neutralize body dispatch (DstFlag).
Set[GrayDstSetupLoc, 30];
Set[SrcDstSetupLoc, 125];
Set[SrcGrayDstSetupLoc, 131];

Set[HorizontalDispLoc, 20];             * Target of setup dispatch

* Page -- relative entry points for body routines.
* They must neutralize all bits used for the setup dispatch.
Set[GrayBodyLoc, 16];
Set[GrayDstBodyLoc, 36];
Set[SrcGrayBodyLoc, 117];               * These must be xxxx1111 because they contain
Set[SrcGrayDstBodyLoc, 137];            * conditional Call instructions
```

```
*-----------------------------------------------------------------------
BitBltSub:              * Entry point
* Preliminaries: read BBTable, decode function
*-----------------------------------------------------------------------
KnowRBase[AEmRegs];
Subroutine;
        T__ Link, StkP - 1;
TopLevel;

        Fetch__ Stack&+3;                       * Fetch Function from BBTable
        Stack&+1__ T, MemBase__ ScratchBR;

* Set up ScratchBR to point to base of BBTable.
        T__ VAHi;
        BRHi__ T;
        T__ VALo;
        BRLo__ T;
        BBRule__ (17S) AND MD, T __ MD; * BBRule__ function dispatch
        BBFunc__ MD;                            * Save function word
                                                * (only to test XM flags below)

* Dispatch on BitBlt function to compute BBOP and BBDisp.
* The dispatch is quite different in the Alto and Smalltalk cases.
* Note that MD still contains the full BitBlt function word.
        PD __ T and (40000c);
        DblBranch[STBBDisp, AltoBBDisp, ALU # 0];

* Smalltalk mode dispatch
STBBDisp:
        GrayAddrMask __ 17C;                    * 16 - word gray
        T __ LDF[T, 2, 6];                      * Source type
        GBCAlo __ 24C;
        GBCAlo __ (Fetch __ GBCAlo) + 1; * Fetch long gray pointer
        GBCAlo __ MD, Fetch __ GBCAlo;
        BBRule __T, BigBDispatch__ BBRule;              * BBRule __ source type
        GBCAhi __ MD, Branch[STBBFunctionTable];
*-----------------------------------------------------------------------
STBBFunctionTable: DispTable[20];               * Rule
        * Note that the rule specifies dest __ g(dest, maskedSource),
        * but the loops all do dest __ f(NOT maskedSource, dest).
        T __ A0, Branch[STBBFZ];        * 0  all zeros
        T __ 21C, Branch[STBBF];        * 1  A AND B
        T __ 11C, Branch[STBBF];        * 2  NOT A AND B
        T __ 1C, Branch[STBBFS];                * 3  B
        T __ 35C, Branch[STBBF];        * 4  A AND NOT B
        T __ 25C, Branch[STBBFD];               * 5  A
        T __ 15C, Branch[STBBF];        * 6  A XOR B
        T __ 5C, Branch[STBBF];         * 7  A OR B
        T __ 33C, Branch[STBBF];        * 10  NOT A AND NOT B
        T __ 23C, Branch[STBBF];        * 11  A EQV B
        T __ 13C, Branch[STBBFD];               * 12  NOT A
        T __ 3C, Branch[STBBF];         * 13  NOT A OR B
        T __ 37C, Branch[STBBFS];               * 14  NOT B
        T __ 27C, Branch[STBBF];        * 15  A OR NOT B
        T __ 17C, Branch[STBBF];        * 16  NOT A OR NOT B
        T __ T - T - 1, Branch[STBBFZ];         * 17  all ones
*-----------------------------------------------------------------------

* Re - dispatch on the source type

* Rule uses neither source nor dest
KnowRBase[AEmRegs];
STBBFZ:
        Store __ 17S, Dbuf __ T, Call[SetFakeGray];  * Construct one - word gray
        T __ 1C;                        * ALU function to select maskedSource
        RBase __ RBase[BBRegs], Branch[SBBFA];  *do this way for placement

* Rule uses dest but not source
KnowRBase[AEmRegs];
STBBFD:
        Call[SetFakeGray];  * 9/26/84 Construct one - word gray
        RBase __ RBase[BBRegs], Branch[SBBFB];  *do this way for placement

* Rule uses source but not dest
KnowRBase[AEmRegs];
STBBFS:
        BDispatch __ BBRule;
        RBase __ RBase[BBRegs], Branch[. + 1];
                DispTable[4],
        RBase __ RBase[AEmRegs], Branch[UseAllOnes];
        PD __ T - 1, Branch[TestCompSource];
        Branch[BBFC];
        Branch[BBFD];
TestCompSource:                         * Test if using gray or gray'
        DblBranch[SBBFA, SBBFB, ALU = 0];               * Only use case A if gray straight through
SBBFA:  Branch[BBFA];           *placement
SBBFB:  Branch[BBFB];           *placement

* Rule uses both source and dest
KnowRBase[AEmRegs];
STBBF:
        BDispatch __ BBRule;
        RBase __ RBase[BBRegs], Branch[. + 1];
                DispTable[4],
        RBase __ RBase[AEmRegs], Branch[UseAllOnes];
        Branch[BBFB];
```

```
        Branch[BBFE];
        Branch[BBFF];

UseAllOnes:
        GrayAddrMask _ T - T - 1;
        Store _ 17S, Dbuf _ GrayAddrMask, Call[SetFakeGray];
        RBase _ RBase[BBRegs], Branch[SBBFB];

* Subroutine to set address of fake gray
Subroutine;
KnowRBase[AEmRegs];
SetFakeGray:
        GBCAhi _ VAHi;              * Set gray pointer
        GBCAlo _ VALo;
        GrayAddrMask _ A0, Return;          * 1 - word gray
Top level;
**********

* Alto mode dispatch
KnowRBase[AEmRegs];
AltoBBDisp:
        GrayAddrMask _ 3C;                  * 4 - word gray
        T _ 14C;
        DummyRef _ T;               * Compute pointer to gray
        GBCAhi _ VAHi;
        GBCAlo _ VALo;
        BigBDispatch_ BBRule;
        Rbase _ Rbase[BBRegs], Branch[AltoBBFunctionTable];

*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
AltoBBFunctionTable: DispTable[20];         * SourceType, Operation
        T_ 1C, Branch[BBFC];                * 0, 0      NOT A
        T_ 5C, Branch[BBFE];                * 0, 1      NOT A OR B
        T_ 15C, Branch[BBFE];               * 0, 2      A EQV B
        T_ 35C, Branch[BBFE];               * 0, 3      A AND B
        T_ 37C, Branch[BBFC];               * 1, 0      A
        T_ 27C, Branch[BBFE];               * 1, 1      A OR B
        T_ 23C, Branch[BBFE];               * 1, 2      A XOR B
        T_ 21C, Branch[BBFE];               * 1, 3      NOT A AND B
        T_ 1C, Branch[BBFD];                * 2, 0      NOT A
        T_ 5C, Branch[BBFF];                * 2, 1      NOT A OR B
        T_ 15C, Branch[BBFF];               * 2, 2      A EQV B
        T_ 35C, Branch[BBFF];               * 2, 3      A AND B
        T_ 1C, Branch[BBFA];                * 3, 0      NOT A
        T_ 5C, Branch[BBFB];                * 3, 1      NOT A OR B
        T_ 15C, Branch[BBFB];               * 3, 2      A EQV B
        T_ 35C, Branch[BBFB];               * 3, 3      A AND B
*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

* Select the appropriate dispatch word for the function.
BBFA:   BBDisp_ Add[GrayFlg]C, Branch[SetBBF];
BBFB:   BBDisp_ Add[GrayFlg, DstFlg]C, Branch[SetBBF];
BBFC:   BBDisp_ Add[SrcFlg, NoGrayFlg]C, Branch[SetBBF];
BBFD:   BBDisp_ Add[SrcFlg, GrayFlg]C, Branch[SetBBF];
BBFE:   BBDisp_ Add[SrcFlg, NoGrayFlg, DstFlg]C, Branch[SetBBF];
BBFF:   BBDisp_ Add[SrcFlg, GrayFlg, DstFlg]C, Branch[SetBBF];

SetBBF: Stack& + 1_ ALUFMRW_ T, ALUF[17];   * Set ALU function and save old
```

```
*-----------------------------------------------------------------
* X--coordinate setup: widths, margins, skew, masks, etc.
*-----------------------------------------------------------------

* Fetch the X-coordinate information.
        MCount_ NOT (Fetch_ 6S);                * MCount_ small negative, fetch DW
        Width_ MD, Fetch_ 12S;                  * Fetch SLX
        SrcX_ MD, T_ A0, Fetch_ 4S;             * Fetch DLX -- must be fetched last

* Compute LMask and RMask.
* LMask = DLX mod 20, RMask = 17-((DLX+DW-1) mod 20) = (-DLX-DW) mod 20.
* For reference, SHC fields are:
* B2: SHA=T, B3: SHB=T, B4-7: count, B8-11: RMask, B12-15: LMask
        DstX_ MD, T_ T-(Width);         * T_ -DW
        T_ T-(DstX);                            * T_ -DLX-DW
        BBMasks_ DPF[T, 4, 4, MD];              * B8-11_ ((-DLX-DW) mod 20) lsh 4,
                                                * B12-15_ DLX mod 20

* Compute destination width in words, including first and last partial words.
* DWidth_ (Width + (DLX mod 20) + 17)/20.
        T_ (DstX) AND (17C);
        T_ (Width)+T;
        T_ T+(17C);
        DWidth_ RSH[T, 4];

        BBDisp, Branch[.+2, R odd]; * Source block required?

* Source block not required. Set shift count to send R straight thru shifter,
* and handle as left-to-right case.
        Skew_ A0, Branch[SetupLtoR];            * SHA=R, SHB=R

* Compute source width in words, including first and last partial words.
* SWidth_ (Width + (SLX mod 20) + 17)/20.
        T_ (SrcX) AND (17C);
        T_ (Width)+T;
        T_ T+(17C);
        SWidth_ T_ RSH[T, 4];

* Set flags to control fetching of "extra" first and last words.
* Except in the "thin" case (DWidth=1), the setup/finish routines for the
* horizontal loops nominally fetch 1 word and store 2 words; an extra fetch
* may be required at the beginning, the end, or both, depending on the number
* of words in the source and destination blocks (see introductory comments).
* DWidth and SWidth differ by at most +/-1.
* If SWidth = DWidth+1, an extra source word must be fetched at both ends.
* If SWidth = DWidth-1 or SWidth = 1, no extra source words need be fetched.
* If SWidth = DWidth # 1, an extra source word must be fetched at one end:
*  if SLX mod 20 > DLX mod 20 then left else right.
* Set BBFlags[0]_ extraLeft, BBFlags[1:15]_ extraRight.
* (these flags are exchanged later if working right-to-left.)
* T still has SWidth; MD still has DLX.
* The following 2 instructions set BBFlags_ -1 if SWidth>DWidth, 0 otherwise.
        PD_ (DWidth)-T;
        BBFlags_ T-T-1, XorSavedCarry, Branch[SetupSkew, ALU#0];

* Set BBFlags_ 100000 if SLX mod 20 > DLX mod 20, 77777 otherwise.
        T_ (SrcX) AND (17C);
        BBFlags_ (17S) AND MD;                  * DLX mod 20
        PD_ T-(BBFlags)-1;                      * Carry iff SLX mod 20 > DLX mod 20
        BBFlags_ 100000C;
        BBFlags_ (BBFlags)-1, XorSavedCarry;
```

```
* X - coordinate setup (cont'd)

* Compute skew = (SLX - DLX) mod 20, and decide on horizontal direction.
SetupSkew:
        T_ (SrcX) - MD;
        Skew_ T AND (17C), DblBranch[SetupRtoL, SetupLtoR, ALU<0];

* SLX >= DLX: work from left to right.  Set SHA = R, SHB = T, ALUF[15] = "A + 1".
* Note: if skew = 0, set SHA = R, SHB = R.
SetupLtoR:
        T_ 200C, Branch[NoSrcThinChk, ALU=0]; * ALUFM control for "A + 1"
        Skew_ (Skew) OR (20C), Branch[SetupALU&ShC]; * SHB = T

* SLX < DLX: work from right to left.
* Set SHA = T, SHB = R, ALUF[15] = "A - 1".
* Advance starting X coordinates to rightmost ends of blocks.
* Note: if skew = 0, set SHA = R, SHB = R, and do not exchange extra - word flags.
SetupRtoL:
        BBDisp_ (BBDisp) OR (100000C), Branch[. + 3, ALU=0];
        Skew_ (Skew) OR (40C);                  * SHA = T
        BBFlags_ (BBFlags) LCY 1;               * Exchange source extra - word flags
        T_ (Width) - 1;
        DstX_ (DstX) + T;                       * Advance to rightmost X - coordinates
        SrcX_ (SrcX) + T;
        T_ 36C;                                 * ALUFM control for "A - 1"

* Have ALUFM control in T for "A +/- 1" operation.
SetupALU&ShC:
        PD_ (SWidth) - 1;                       * Thin source check (see below)
NoSrcThinChk:
        Stack& - 4_ ALUFMRW_ T, A +/- 1,        * Set ALU function, save old value.
                Branch[SetupShC, ALU#0]; * Leave StkP -> TOS (scan line count)

* If we would have fetched an extra source word, but there is only one source
* word to fetch, then reset source extra - word flags and set SHA = R, SHB = R.
        BBFlags_ A0, Branch[. + 2, R>=0];
        Skew_ (Skew) AND (17C);
        T_ LSH[BBMasks, 10], Branch[. + 2]; * Placement

* Merge skew with masks and load SHC.
SetupShC:
        T_ LSH[BBMasks, 10];                    * Shift masks to B0 - 7
        T_ LCY[T, Skew, 10];                    * Concatenate SHA, SHB, count, masks
        PD_ (Width) - 1, ShC_ T;

* Convert DstX and SrcX to X word displacements relative to start of
* first scan line.  Note: BBDst is the same register as DstX, and
* BBSrc is the same register as SrcX.
        BBDst_ RSH[DstX, 4], Branch[. + 2, ALU>=0]; * BBDst_ DstX/20
        Branch[BitBltDone];                     * Width<=0 -- nothing to do
        BBSrc_ RSH[SrcX, 4];                    * BBSrc_ SrcX/20

*-----------------------------------------------------------------------
* Fetch and set up Y - coordinate information
*-----------------------------------------------------------------------
        Fetch_ 7S;                              * Fetch DH
        T_ MD, Fetch_ 5S;                       * Fetch DTY
        DstY_ MD, Fetch_ 13S;                   * Fetch STY -- must be fetched last
        Skew_ (DstY) - MD;                      * Test and remember vertical direction
* VCount_ (scan lines left to do) - 1.  Stack has scan lines already done.
        VCount_ T - (Stack) - 1, Branch[YTopToBottom, ALU<0]; * DH - (scan lines done) - 1

* DTY >= STY, work from bottom to top.  Start with lowest line not yet done.
        GrayDirection _ T - T - 1;
        T_ (DstY) + 1; WAS: (2c);
        GrayOffset _ T;                 * Offset into gray
        T_ VCount, Branch[YFinish];

* DTY < STY, work from top to bottom.  Start with highest line not yet done.
YTopToBottom:
        GrayDirection _ A0;
        T_ (DstY) + T;
        GrayOffset _ T - 1;                     * Offset into gray
        T_ Stack;

YFinish:
        DstY_ (DstY) + T;                       * Compute starting Y coordinates
        SrcY_ T + MD;                           * MD still has STY
```

```
*-----------------------------------------------------------------------
* Set up destination base address and increments
*-----------------------------------------------------------------------

* Compute starting destination scan line offset relative to base of bit map.
* DstY has scan line number.
        T__ DstY, Fetch__ 3S;                   * Fetch DBMR
        DRast__ MD, T__ A0, Q__ T, Branch[. + 2, ALU = 0]; * Don't multiply if DstY = 0
        T__ DRast, Call[MulSub];                * T,,Q __ Q*T, clobbers BBTemp

* Now have 32 - bit offset in T,,Q.  Add it to bit map base address.
**** Program around MicroD problem.  Desired statement is:
****    PD__ (BBFunc) AND (20C), DblCall[DFetchBBLong, DFetchBBShort, R < 0];
**** but we must actually write:
        PD__ (BBFunc) AND (20C), BRGO@[0] RETCL@[3] JCN[44] GPW0@[11400];
                Set[DFetchBBShortLoc, 1420];
****

* Now T,,BBTemp = adjusted base value.
* Note: Skew < 0 here iff working top - to - bottom.
        MemBase__ BBDstBR, Skew, Branch[DstLoadBR, R < 0];

* Processing bottom - to - top.
* Decrease base by 2^15, and increase initial displacement by 2^15.
        BBTemp__ (BBTemp) - (100000C);
        T__ T - 1, XorSavedCarry;
        BBDst__ (BBDst) + (100000C);
        DRast__ (0S) - (DRast);                 * DRast__ - DBMR

* Now finally ready to load the base register!
DstLoadBR:
        T__ DWidth, BRHi__ T;                   * DWidth for code below
        BRLo__ BBTemp;

* Compute destination inter - scan - line word address increment.
* DstInc__ DRast + (if left - to - right then - DWidth else DWidth).
* Also compute first PreFetch offset (from BBDst).
* = leftmost word on next scan line, even if processing right - to - left.
* DPrefOffset__ if L - to - R then DRast else DRast - DWidth.
* Note that DRast already has DBMR or - DBMR as appropriate.  T = DWidth.
        BBDisp, Branch[DstIncRtoL, R < 0]; * Test direction

DstIncLtoR:
        DstInc__ (DRast) - T;                   * L to R: DRast - DWidth
        DPrefOffset__ T__ DRast, Branch[. + 3];

DstIncRtoL:
        DstInc__ (DRast) + T;                   * R to L: DRast + DWidth
        DPrefOffset__ T__ (DRast) - T;

        PrefDst__ (BBDst) + T;                  * Initial PreFetch address
```

```
*-----------------------------------------------------------------------
* Set up source base address and increments
*-----------------------------------------------------------------------

* Is a source block required? If not, skip source setup.
        BBDisp, Branch[BBNoSource, R even];

        MemBase__ ScratchBR;
        Fetch__ 11S, T __ SrcY;                 * Fetch SBMR

* Compute starting source scan line offset relative to base of bit map.
* T = ALU = SrcY = starting source scan line number.
        SRast__ MD, T__ A0, Q__ T, Branch[. + 2, ALU = 0]; * Don't multiply if SrcY = 0
        T__ SRast, Call[MulSub];                * T,,Q __ Q*T, clobbers BBTemp

* Now have 32 - bit offset in T,,Q. Add it to bit map base address.
**** Program around MicroD problem. Desired statement is:
****    PD__ (BBFunc) AND (40C), DblCall[SFetchBBLong, SFetchBBShort, R < 0];
**** but we must actually write:
        PD__ (BBFunc) AND (40C), BRGO@[0] RETCL@[3] JCN[104] GPW0@[11400];
                Set[SFetchBBShortLoc, 1440];
****

* Now T,,BBTemp = adjusted base value.
* Note: DRast > = 0 here iff working top - to - bottom.
        MemBase__ BBSrcBR, DRast, Branch[SrcLoadBR, R > = 0];

* Processing bottom - to - top.
* Decrease base by 2^15, and increase initial displacement by 2^15.
        BBTemp__ (BBTemp) - (100000C);
        T__ T - 1, XorSavedCarry;
        BBSrc__ (BBSrc) + (100000C);
        SRast__ (0S) - (SRast);                 * SRast__ - SBMR

* Now finally ready to load the base register!
SrcLoadBR:
        T__ SWidth, BRHi__ T;                   * SWidth for code below
        BRLo__ BBTemp;

* Compute source inter - scan - line word address increment.
* SrcInc__ SRast + (if left - to - right then - SWidth else SWidth).
* Also compute first PreFetch offset (from BBSrc).
* = leftmost word on next scan line, even if processing right - to - left.
* SPrefOffset__ if L - to - R then SRast else SRast - SWidth.
* Note that SRast already has SBMR or - SBMR as appropriate. T = SWidth.
        BBDisp, Branch[SrcIncRtoL, R < 0]; * Test direction

SrcIncLtoR:
        SrcInc__ (SRast) - T;                   * L to R: SRast - SWidth
        SPrefOffset__ T__ SRast, Branch[. + 3];

SrcIncRtoL:
        SrcInc__ (SRast) + T;                   * R to L: SRast + SWidth
        SPrefOffset__ T__ (SRast) - T;

        PrefSrc__ (BBSrc) + T;                  * Initial PreFetch address
```

```
*--------------------------------------------------------------------------
* Final adjustments prior to entering vertical loop
*--------------------------------------------------------------------------

* Compute ICnt, the initial value of the Cnt register for each loop.
* ICnt_ if DWidth < = 22B then DWidth - 1 else NOT (DWidth - 3) [= 2 - DWidth].
* MCount starts out negative (was initialized long ago).
BBNoSource:
        PD_ (DWidth) - (23C);
        ICnt_ T_ (DWidth) - 1, Branch[. + 2, ALU<0];
        ICnt_ (1S) - T;                         * = (2S) - (DWidth)
        Stack_ (Stack) - 1;

* Reset ScratchBR to point to gray
        T _ Pointers;
        MemBase _ ScratchBR;
        RBase _ RBase[GBCAlo];
        BRLo _ GBCAlo;
        BRHi _ GBCAhi;
        MemBase _ T;

* Enter vertical loop with (scan lines done) - 1 on TOS and ALU > = 0.
        RBase _ RBase[BBRegs], T_ A0;
        PD_ Q_ T, Branch[BBVerticalLoop]; * Init gray value to 0


*--------------------------------------------------------------------------
* xFetchBBShort:
* Fetch BitBlt short pointer for destination or source
* Entry: ALU#0 iff alternate bank bit set
*        T,,Q = 32 - bit displacement
* Exit: T,,BBTemp = adjusted base address
*--------------------------------------------------------------------------
Subroutine;

**** Note: flush absolute placement when MicroD is fixed ****
DFetchBBShort: At[DFetchBBShortLoc],            * Fetch DBMR
        BBTemp_ Q, Fetch_ 2S, DblBranch[BBNormal, BBAlternate, ALU = 0];
SFetchBBShort: At[SFetchBBShortLoc],            * Fetch SBMR
        BBTemp_ Q, Fetch_ 10S, DblBranch[BBNormal, BBAlternate, ALU = 0];

BBNormal:
        BBTemp_ (BBTemp) + MD, RBase_ RBase[AEmRegs];
        T_ T + (EmuBRHiReg), XorSavedCarry, Branch[FetchBBRet];

KnowRBase[BBRegs];
BBAlternate:
        BBTemp_ (BBTemp) + MD, RBase_ RBase[AEmRegs];
        T_ T + (EmuXMBRHiReg), XorSavedCarry;
FetchBBRet:
        RBase_ RBase[BBRegs], Return;


*--------------------------------------------------------------------------
* xFetchBBLong:
* Fetch BitBlt long pointer for destination or source
* Entry: T,,Q = 32 - bit displacement
* Exit: T,,BBTemp = adjusted base address
*--------------------------------------------------------------------------
Subroutine;

**** Note: flush absolute placement when MicroD is fixed ****
DFetchBBLong: At[DFetchBBShortLoc, 1],
        BBTemp_ 22C, Branch[. + 2];             * Destination long pointer low word
SFetchBBLong: At[SFetchBBShortLoc, 1],
        BBTemp_ 20C;                            * Source long pointer low word
        BBTemp_ (Fetch_ BBTemp) + 1;
        BBTemp_ MD, Fetch_ BBTemp;
        BBTemp_ (BBTemp) + Q;
        T_ T + MD, XorSavedCarry, Return;

TopLevel;
```

```
*-----------------------------------------------------------------
* BitBlt vertical loop (per-scan-line)
* At top of loop, the following invariants hold:
*         VCount = (number of scan lines remaining) - 1
*         Stack = (number of scan lines done) - 1
*         MemBase = BBDstBR in destination-only cases, BBSrcBR otherwise
*         ALU < 0 iff destination BR displacement has overflowed
* Vertical loop overhead:
*          6 cycles for loop control and destination pointer update
*          +4 cycles for source pointer update if source block is used
*          +4 cycles if block is greater than 20B words wide
*-----------------------------------------------------------------

BBVerticalLoop:
        VCount, Branch[BBDoneOrOverflow, ALU<0, R<0];
        VCount__(VCount) - 1;
* If positive, ICnt has the desired initial value of Cnt (DWidth - 1).
        T__ NOT (Cnt__ ICnt), Branch[SmallBlock, R>=0];

* Block greater than one munch wide.  Set up separate munch and word counts.
* T now has DWidth - 3, where DWidth is the number of words in the destination
* block, including first and last partial words.
* MCount __ (DWidth - 3)/20 - 1, Cnt __ (DWidth - 3) mod 20 + 2.
        MCount__ RSH[T, 4];
        T__ T AND (17C);
        T__ T + (2C);
        MCount__ (MCount) - 1, Cnt__ T;

* This dispatch goes to one of: GrayDstSetup,
* SrcDstSetup, or SrcGrayDstSetup.
SmallBlock:
        BigBDispatch__ BBDisp;
        Stack__ (Stack) + 1, Branch[HorizontalDisp]; * Advance scan lines done


*-----------------------------------------------------------------
AdvanceSrcDst:
* Control returns here at the end of individual horizontal loops that
* involve both source and destination blocks.
* BBSrc, BBDst point one beyond last word processed.
*-----------------------------------------------------------------
        T__ SrcInc;
        BBSrc__ T__ (BBSrc) + T;
        PrefSrc__ T + (SPrefOffset);
        T__ BBDst, Branch[SrcBROverflow, ALU<0];

*-----------------------------------------------------------------
AdvanceDst:
* Control returns here at the end of individual horizontal loops that
* involve only a destination block.
* BBDst and T point one beyond last word processed.
*-----------------------------------------------------------------
        BBDst__ T__ T + (DstInc);
        PrefDst__ T__ T + (DPrefOffset),
                DblBranch[BBReschedPending, BBVerticalLoop, Reschedule];

SrcBROverflow:
        Branch[BBDoneOrOverflow];
```

```
*-----------------------------------------------------------------------
* Reschedule pending.  See if interrupt is really being requested,
* and if so, terminate processing and return +1 from BitBltSub.
* The calling emulator will process the interrupt and then
* call BitBltSub again.
* T = PrefDst here.
*-----------------------------------------------------------------------
BBReschedPending:
        VCount, RBase__ RBase[NWW], Branch[.+2, R>=0];

* We just processed the last scan line, so return normally.
        Branch[BitBltDone];

* Test NWW to see whether an interrupt is really pending.
        PD__ NWW, NoReschedule;
        Branch[.+2, ALU>0];

* No interrupt pending.  Restore RBase and ALU and continue vertical loop.
        PD__ T, RBase__ RBase[BBRegs], Branch[BBVerticalLoop];

* Interrupt really pending.  Restore clobbered ALUFM locations before
* taking interrupt.  Note: scan line count at TOS is one behind, so fix it.
BBInitiateInterrupt:
        T__ Stack&+3__(Stack&+3)+1, RescheduleNow, * Force immediate trap
                Branch[BitBltDone1];


*-----------------------------------------------------------------------
* Either VCount is exhausted or one of the BR displacements overflowed
* (or conceivably both events occurred at the same time).
* If VCount is exhausted then return normally; otherwise restart BitBlt.
* The easiest way to restart is to pretend an interrupt occurred.
* Note: scan line count at TOS is one behind.
*-----------------------------------------------------------------------
BBDoneOrOverflow:
        VCount, Branch[BitBltDone, R<0];
        RBase__ RBase[AEmRegs], Branch[BBInitiateInterrupt];


*-----------------------------------------------------------------------
* Really done.  Set TOS=0, restore clobbered ALUFM, and return.
*-----------------------------------------------------------------------
BitBltDone:
        T__ Stack&+3__ A0, RBase__ RBase[AEmRegs];
BitBltDone1:
        ALUFMRW__ Stack&+1, ALUF[17];       * Restore ALUF[17]
        ALUFMRW__ Stack&-2, A+/-1;          * Restore ALUF borrowed for A+/-1
        PD__ T, Link__ Stack&-2;
Subroutine;
        Return[ALU=0];                      * Return +2 if done, +1 otherwise
TopLevel;
```

```
*-----------------------------------------------------------------
* Horizontal loops
*-----------------------------------------------------------------

%
Organization of horizontal (per - word) loops:

There are a number of variations, depending on the following:
        1. Whether or not a source block is used;
        2. Whether or not a gray block is used;
        3. Whether or not the destination is an operand;
        4. Whether or not the destination is "thin" (one word per scan line);

There are fewer than 32 total cases because some of these combinations
cannot occur (for example, no source block and no gray block).
Each case has a "setup" routine, a "body" routine, and a "finish" routine.
Many of these routines are shared among cases, and the flow of control
is determined by a complicated network of dispatches on BBDisp.
Finish routines exit to the vertical loop by a branch to AdvanceDst or
AdvanceSrcDst.

Q is used exclusively to hold the complement of the gray value for the
current scan line, or zero if gray is not used.

To reduce miss wait and increase performance, while processing each
scan line we fall out of the main loop once per munch and PreFetch
one munch for the next scan line.  This strategy depends on the
assumption that each scan line is less than 100 munches long (for a 100 - row
cache, which is what the Dorado has at present).  Note that PreFetches
are done left - to - right even if transfers are done right - to - left.

Note that the current implementation is imperfect in that the last munch
of the next scan line may not be prefetched, or an extra munch prefetched
unnecessarily, because the main loop does not terminate at munch boundaries
but rather at multiples of 20 words from the end of the scan line.
Also, the first scan line is not prefetched, and a line past the end of
the last scan line is prefetched unnecessarily.  These defects should not
affect performance noticeably in normal use.
%

TopLevel;
KnowRBase[BBRegs];
```

```
%*----------------------------------------------------------------------------
Case A:  dest_gray

ShC = R straight through.
BBOp = NOT A.

Entry point to this code is at GrayDestSetup.
It dispatches here after setup.

Timing, per scan line, including vertical loop overhead:
          20 cycles minimum in the normal case
          + 1 cycle per full word (excluding first and last partial words)
          + 4 + (2 * # of munches) cycles if block is wider than 20B words

          13 cycles total in the "thin" case
%*----------------------------------------------------------------------------

* T has first partial word to be stored, and SrcWd has (uncomplemented)
* gray word.
GrayBody: BBAt[GrayBodyLoc],
          Q_ SrcWd;
          BBDst_ (Store_ BBDst) +/- 1, DBuf_ T, Branch[GrayEnd, Cnt=0& - 1];

GrayLoop:
          BBDst_ (Store_ BBDst) +/- 1, DBuf_ Q, Branch[GrayLoop, Cnt#0& - 1];

GrayEnd:
          MCount_ (MCount) - 1, Cnt_ 17S, Branch[GrayLast, R<0];
          PrefDst_ (PreFetch_ PrefDst), Carry20, Branch[GrayLoop];

GrayLast:
          Fetch_ BBDst, Branch[DstFinish];
```

%* -
Case B: dest _ f(gray, dest)

ShC = R straight through.
BBOp is taken from the following table:

Operation	SourceType	BBOP
source OR dest	gray	NOT A OR B
source XOR dest	gray	A EQV B
NOT source AND dest	gray	A AND B

The gray word is kept in SrcWd and put directly into the shifter.

Case A also dispatches to one of case B's entry points, GrayDstSetup.
The setup code dispatches to the correct body routine.

Timing, per scan line, including vertical loop overhead:
 18 cycles minimum in the normal case
 +3 cycles per full word (excluding first and last partial words)
 +6 + (2 * # of pages) cycles if data needs to be "touched"
 +4 + (2 * # of munches) cycles if block is wider than 208 words

 13 cycles total in the "thin" case
%* -

```
GrayDstSetup: BBAt[GrayDstSetupLoc],
        T _ NOT (VCount), RBase _ RBase[GrayAddrMask], Call[GetGray];
KnowRBase[BBRegs];
        MemBase _ BBDstBR;
        SrcWd _ NOT T, Fetch _ BBDst;

* Label the target for the horizontal dispatch. It is never actually reached
* by the dispatch, since the branch address is modified by BigBDispatch.
* The instruction so labelled should be on the same page as the actual
* targets, and it should not otherwise be constrained.
HorizontalDisp: BBAt[HorizontalDispLoc],
        PrefDst _ (PreFetch _ PrefDst), Carry20, Branch[DstThin, Cnt=0& - 1];
* The following dispatch may modify GrayBody to GrayDstBody.
        BigBDispatch _ BBDisp, Branch[. + 2, R<0];
        T _ XShMDLMask[SrcWd], B _ MD, Branch[GrayBody];
        T _ XShMDRMask[SrcWd], B _ MD, Branch[GrayBody];

* Body routine for case B only.
GrayDstBody: BBAt[GrayDstBodyLoc],
        BBDst _ T _ (Store _ BBDst) +/- 1, DBuf _ T;
        BBSrc _ (Fetch _ T) +/- 1, Branch[GrayDstEnd, Cnt=0& - 1];

* Inner loop runs with one word fetched ahead (now in MD).
* BBSrc runs one word ahead of BBDst.
GrayDstLoop:
        BBSrc _ (Fetch _ BBSrc) +/- 1, T _ MD;
        T _ XShiftNoMask[SrcWd], B _ T;
        BBDst _ (Store _ BBDst) +/- 1, DBuf _ T, Branch[GrayDstLoop, Cnt#0& - 1];

GrayDstEnd:
        MCount _ (MCount) - 1, Cnt _ 17S, Branch[DstFinish, R<0];
        PrefDst _ (PreFetch _ PrefDst), Carry20, Branch[GrayDstLoop];

* Store last partial word. This is the tail of case A also.
* Have already fetched the last word (at BBDst).
DstFinish:
        BBDisp, Branch[. + 2, R<0];
        T _ XShMDRMask[SrcWd], B _ MD, Branch[StoreLastDst];
        T _ XShMDLMask[SrcWd], B _ MD, Branch[StoreLastDst];

* Thin destination slice, for cases A and B.
DstThin:
        T _ XShMDBothMasks[SrcWd], B _ MD;
StoreLastDst:
        BBDst _ T _ (Store _ BBDst) +/- 1, DBuf _ T, Branch[AdvanceDst];
```

```
%*------------------------------------------------------------------------
Case C:  dest _ f(source)
Case D:  dest _ f(source, gray)

BBOp is taken from the following table:

Operation              SourceType             BBOP
source                 source                 NOT A        (Q = 0)
source                 NOT source             A            (Q = 0)
source                 source AND gray        NOT A        (Q = NOT gray)

Q is ORed with shifter output on the AMux.

Entry points to this code are at SrcDstSetup, SrcDstSetup&Touch,
SrcGrayDstSetup, and SrcGrayDstSetup&Touch.  They dispatch here after setup.

Timing, per scan line, including vertical loop overhead:
        25 cycles minimum in the normal case
        +4 cycles per full word (excluding first and last partial words)
        +12 + (4 * # of pages) cycles if data needs to be "touched"
        +1 cycle if 2 source words are required for the first destination word
        +3 cycles if gray is required
        +4 + (5 * # of munches) cycles if block is wider than 20B words

        17 cycles minimum in the "thin" case
        +1 cycle if 2 source words are required
        +3 cycles if gray is required
%*------------------------------------------------------------------------


* Body routine for cases C and D only.
SrcGrayBody: BBAt[SrcGrayBodyLoc],
        T_BBDst_(Store_BBDst)+/-1, DBuf_T, FlipMemBase,
**** Program around MicroD problem.  Desired branch clause is:
****            Call[SrcGrayEnd, Cnt=0&-1];
**** but we must actually write:
                BRGO@[0] RETCL@[2] JCN[43];
        BBSrc_(Fetch_BBSrc)+/-1, FlipMemBase,
                BBAt[SrcGrayBodyLoc, 1]; **** MicroD problem
        SrcWd_MD, T_SrcWd;
        T_XShiftNoMask[SrcWd], A_Q, Branch[SrcGrayBody];

* This is called as a subroutine at the end of each munch.
* It either exits the horizontal loop or returns to do another munch.
SrcGrayEnd:
Subroutine;
        MCount_(MCount)-1, Cnt_17S,
                DblBranch[SrcDstFinish, SrcDstMore, R<0],
                BBAt[SrcGrayBodyLoc, 2]; **** MicroD problem
TopLevel;
```

```
%* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Case E:  dest _ f(source, dest)
Case F:  dest _ f(source, gray, dest)

BBOp is taken from the following table:

Operation               SourceType          BBOP
source OR dest          source              NOT A OR B      (Q = 0)
source XOR dest         source              A EQV B         (Q = 0)
NOT source AND dest     source              A AND B         (Q = 0)
source OR dest          NOT source          A OR B          (Q = 0)
source XOR dest         NOT source          A XOR B         (Q = 0)
NOT source AND dest     NOT source          NOT A AND B     (Q = 0)
source OR dest          source AND gray     NOT A OR B      (Q = NOT gray)
source XOR dest         source AND gray     A EQV B         (Q = NOT gray)
NOT source AND dest     source AND gray     A AND B         (Q = NOT gray)

Q is ORed with shifter output on the AMux.

Cases C and D also dispatch these entry points.
The setup code dispatches to the correct body routines.

Timing, per scan line, including vertical loop overhead:
          25 cycles minimum in the normal case
        + 5 cycles per full word (excluding first and last partial words)
        + 12 + (4 * # of pages) cycles if data needs to be "touched"
        + 1 cycle if 2 source words required for the first destination word
        + 3 cycles if gray is required
        + 4 + (5 * # of munches) cycles if block is wider than 20B words

          17 cycles minimum in the "thin" case
        + 1 cycle if 2 source words are required
        + 3 cycles if gray is required
%* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

SrcGrayDstSetup: BBAt[SrcGrayDstSetupLoc],
        T _ NOT (VCount), RBase _ RBase[GrayAddrMask], Call[GetGray];
KnowRBase[BBRegs];
        MemBase _ BBSrcBR;
        Q _ T, BBFlags, DblBranch[SrcDstSetup2, SrcDstSetup1, R<0];

SrcDstSetup: BBAt[SrcDstSetupLoc],
        BBFlags, DblBranch[SrcDstSetup2, SrcDstSetup1, R<0];

SrcDstSetup2:
        BBSrc _ (Fetch _ BBSrc) +/- 1;
SrcDstSetup1:
        PrefSrc _ (PreFetch _ PrefSrc), Carry20;
        T _ MD, BBSrc _ (Fetch _ BBSrc) +/- 1, FlipMemBase;
        SrcWd _ MD, Fetch _ BBDst;
        PrefDst _ (PreFetch _ PrefDst), Carry20, Branch[SrcDstThin, Cnt=0& - 1];
* The following dispatch may modify SrcGrayBody to SrcGrayDstBody.
        BigBDispatch _ BBDisp, Branch[. + 2, R<0];
        T _ XShMDLMask[SrcWd], A _ Q, B _ MD, Branch[SrcGrayBody];
        T _ XShMDRMask[SrcWd], A _ Q, B _ MD, Branch[SrcGrayBody];
```

```
* Cases E and F (cont'd)

* Body routine for cases E and F only.
SrcGrayDstBody: BBAt[SrcGrayDstBodyLoc],
        T__ BBDst__ (Store__ BBDst) + / – 1, DBuf__ T, FlipMemBase,
**** Program around MicroD problem.  Desired branch clause is:
****                Call[SrcGrayDstEnd, Cnt = 0& – 1];
**** but we must actually write:
                    BRGO@[0] RETCL@[2] JCN[103];
        BBSrc__ (Fetch__ BBSrc) + / – 1, FlipMemBase,
                    BBAt[SrcGrayDstBodyLoc, 1]; **** MicroD problem
        SrcWd__ MD, T__ SrcWd, Fetch__ T;
        T__ XShiftNoMask[SrcWd], A__ Q, B__ MD, Branch[SrcGrayDstBody];

* This is called as a subroutine at the end of each munch.
* It either exits the horizontal loop or returns to do another munch.
* Note: code following SrcGrayDstMore is used by cases C and D also.
SrcGrayDstEnd:
Subroutine;
        MCount__ (MCount) – 1, Cnt__ 17S,
                    DblBranch[SrcDstFinish, SrcDstMore, R < 0],
                    BBAt[SrcGrayDstBodyLoc, 2]; **** MicroD problem
SrcDstMore:
        PrefSrc__ (PreFetch__ PrefSrc), Carry20;
        FlipMemBase;
        PrefDst__ (PreFetch__ PrefDst), Carry20;
        FlipMemBase, Return;
TopLevel;

* Store last partial word, for cases C, D, E, and F.
SrcDstFinish:
        BBFlags, Branch[. + 2, R even];          * Fetch extra word at end?
        BBSrc__ (Fetch__ BBSrc) + / – 1, FlipMemBase, Branch[. + 2];
        FlipMemBase;
        SrcWd__ MD, T__ SrcWd, Fetch__ T;
        BBDisp, Branch[. + 2, R < 0];
        T__ XShMDRMask[SrcWd], A__ Q, B__ MD, Branch[StoreLastSrcDst];
        T__ XShMDLMask[SrcWd], A__ Q, B__ MD, Branch[StoreLastSrcDst];

* Thin destination slice, for cases C, D, E, and F.
SrcDstThin:
        T__ XShMDBothMasks[SrcWd], A__ Q, B__ MD;
StoreLastSrcDst:
        BBDst__ (Store__ BBDst) + / – 1, DBuf__ T, FlipMemBase,
                    Branch[AdvanceSrcDst];
```

```
*--------------------------------------------------------------------------
* Other subroutines
*--------------------------------------------------------------------------

*--------------------------------------------------------------------------
GetGray:
* Get the gray word for the current scan line.
* Calling sequence:
*       T _ NOT (VCount), RBase_ RBase[GrayAddrMask], Call[GetGray];
* Exit:  T = complement of gray word
*       RBase = BBRegs
* Note: returns Gray(n mod 4 or 16), where n is the number of scan lines
* remaining after the current scan line.  Note that VCount = n - 1.
*--------------------------------------------------------------------------
Subroutine;
KnowRBase[GrayAddrMask];
                Global,
        T _ T XOR (GrayDirection);              * Invert if working bottom - to - top
        T _ T + (GrayOffset);                   * Adjust for initial Y
        T _ T AND (GrayAddrMask), MemBase _ ScratchBR;
        Fetch _ T, T _ A0;
        T _ T - (MD) - 1, Rbase _ Rbase[BBRegs], Return;

END;
```

```
-- File: CvSTC.mesa
-- 11-Sep-89 12:22:22

-- Last Revised by: Erickson                    24-Nov-16:53:30
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (c) 1984, 1985, 1986, 1987, 1989 by Xerox Corporation.  All rights reserved.

DIRECTORY
  Converter
    USING [ConvertProc, CvData, DestroyProc, DependentOptionProc, MenuItemProc],
  NSFile
    USING [Reference],
  Window
    USING [Handle],
  XString
    USING [Reader, ReaderBody, Writer];

<<
--=============================================================================
-- OVERVIEW:

Private definitions interface for the ascii conversion.
--=============================================================================
>>

CvSTC: DEFINITIONS =
BEGIN

--====================
-- CONSTANTS
--====================

stc: CARDINAL = 0;
ctc: CARDINAL = 1;

modern: CARDINAL = 0;
classic: CARDINAL = 1;

twelve: CARDINAL = 0;
twentyFour: CARDINAL = 1;

unlimited: CARDINAL = 0;
limited: CARDINAL = 1;

dfltCodeScheme: CARDINAL = stc;
dfltFont: CARDINAL = modern;
dfltFontSize: CARDINAL = twelve;
dfltTrailing: BOOLEAN = FALSE;
dfltTelCodes: BOOLEAN = FALSE;
dfltLineLen: CARDINAL = unlimited;
dfltChars: CARDINAL = 80;
dfltWordWrap: BOOLEAN = TRUE;

leadingMargin: CARDINAL = 2;
pointsBetweenItems: CARDINAL = 10;

--====================
-- TYPES
--====================

Boolean: TYPE = MACHINE DEPENDENT RECORD[
  zeros(0:0..14): [0..7777B], value(0:15..15): BOOLEAN];

Common: TYPE = LONG POINTER TO CommonData;
CommonData: TYPE = RECORD [
  cvData: Converter.CvData,
  options: BOOLEAN,
  window: Window.Handle,
  owner: Owners,
  ref: NSFile.Reference,
  f: CommonObj,
  textRb: FiledXStrings,
  text: EncodedText,
  z: UNCOUNTED ZONE];
<<
The same data strcture is used by all the client formwindows/details sections.
>>

Filed: TYPE = LONG POINTER TO CommonObj;
CommonObj: TYPE = MACHINE DEPENDENT RECORD [
  avCodeScheme: CARDINAL ← dfltCodeScheme,
  font: CARDINAL ← dfltFont,
  fontSize: CARDINAL ← dfltFontSize,
  ignoreTrailing: Boolean ← [0, dfltTrailing],
  includeTelCodes: Boolean ← [0, dfltTelCodes],
  vaCodeScheme: CARDINAL ← dfltCodeScheme,
  lineLen: CARDINAL ← dfltLineLen,
  charsSuffix: CARDINAL ← dfltChars,
  wordWrap: Boolean ← [0, dfltWordWrap]];
<<
This data structure is the filed data object, along with the var  s strings/text items that come from the formwindows.
>>

EncodedText: TYPE = ARRAY TextIDs OF LONG STRING;
<<
Use long strings internally, since they are better suited to ASCII text.
```

```
>>

FiledXStrings: TYPE = ARRAY TextIDs OF XString.ReaderBody;
<<
Filed strings are kept here.
>>

Owners: TYPE = {AtoVsrc, AtoVdst, VtoAdst, backstop};

TextIDs: TYPE = {
  paraEndsWith,
  atovReplaceUnknown,
  endLine,
  endPara,
  vtoaReplaceUnknown
  };


--====================
-- SIGNALS
--====================

Problem: SIGNAL [err: ProblemType];

ProblemType: TYPE = {obsoleteDataFile, fatalError, doDflts, other};

--====================
-- PROCEDURES
--====================

AsciiToVP: Converter.ConvertProc;
<< = PROCEDURE [source: NSFile.Handle, cvData: Converter.CvData, session: NSFile.Session, srcInstance: LONG POINTER ← NIL, dstInstance:
LONG POINTER ← NIL, background: BOOLEAN ← FALSE] RETURNS [dest: NSFile.Handle ← LOOPHOLE[0]];

Exported by CvSTCToVPImpl.
>>


AsciiToVPSrcOps: Converter.DependentOptionProc;
<< = PROCEDURE [options: BOOLEAN ← TRUE, cvData: Converter.CvData, which: Converter.FormatToUse, srcFormat: XString.Reader, destFormat:
XString.Reader, window: Window.Handle, oldInstance: LONG POINTER ← NIL] RETURNS [menuItemProc: Converter.MenuItemProc, destroy:
Converter.DestroyProc, instance: LONG POINTER];

Exported by CvSTCToVPImpl.
>>


AsciiToVPDstOps: Converter.DependentOptionProc;
<< = PROCEDURE [options: BOOLEAN ← TRUE, cvData: Converter.CvData, which: Converter.FormatToUse, srcFormat: XString.Reader, destFormat:
XString.Reader, window: Window.Handle, oldInstance: LONG POINTER ← NIL] RETURNS [menuItemProc: Converter.MenuItemProc, destroy:
Converter.DestroyProc, instance: LONG POINTER];

Exported by CvSTCToVPImpl.
>>


CommonMenu: Converter.MenuItemProc;
<< = PROCEDURE [instance: LONG POINTER, menuItem: PropertySheet.MenuItemType] RETURNS [ok: BOOLEAN ← TRUE];

Exported by CvSTCFWImpl.
>>


CreateCommon: PROC [cvData: Converter.CvData, options: BOOLEAN, window: Window.Handle, owner: Owners] RETURNS [my: Common];  --!
NSFile.Error
<<
Exported by CvSTCDataImpl.
>>


CreateFW: PROC [my: Common, window: Window.Handle, owner: Owners];
<<
Exported by CvSTCFWImpl.
>>


DataFromWindow: PROC [w: Window.Handle] RETURNS [my: Common];
<<
Exported by CvSTCMainImpl
>>


DataToWindow: PROC [my: Common, w: Window.Handle];
<<
Exported by CvSTCMainImpl
>>


DestroyCommon: Converter.DestroyProc;
<< = PROCEDURE [instance: LONG POINTER];

Exported by CvSTCDataImpl.
>>

GetPreMargin: PROC [item: MessageKey] RETURNS [leads: CARDINAL];
<<
```

```
Exported by CvSTCMainImpl.
>>

GetMessage: PROC [msg: MessageKey] RETURNS [msgRb: XString.ReaderBody];
<<
Exported by CvSTCMsgFileImpl.
>>

InitFiledData: PROC [my: Common];  -- ! NSFile.Error
<<
Create and initialize client file.  Exported by CvSTCDataImpl.
>>

LoadFiledData: PROC [my: Common];  -- ! NSFile.Error, Problem
<<
Read filed data.  Exported by CvSTCDataImpl.
>>

ParseItem: PROC [my: Common, r: XString.Reader, item: MessageKey, buf: XString.Writer ← NIL] RETURNS [ok: BOOLEAN, ls: LONG STRING];
<<
Exported by CvSTCParseImpl.   If ok is FALSE, error during parse.  ls is NIL if item has null text.  buf is a temporary buffer that will
be created and destroyed each time the proc is called if defaulted, otherwise it will just be used.
>>

StoreFiledData: PROC [my: Common];   -- ! NSFile.Error
<<
Write filed data.  Exported by CvSTCDataImpl.
>>

VPToAscii: Converter.ConvertProc;
<< = PROCEDURE [source: NSFile.Handle, cvData: Converter.CvData, session: NSFile.Session, srcInstance: LONG POINTER ← NIL, dstInstance:
LONG POINTER ← NIL, background: BOOLEAN ← FALSE] RETURNS [dest: NSFile.Handle ← LOOPHOLE[0]];

Exported by CvSTCFromVPImpl.
>>

VPToAsciiDstOps: Converter.DependentOptionProc;
<< = PROCEDURE [options: BOOLEAN ← TRUE, cvData: Converter.CvData, which: Converter.FormatToUse, srcFormat: XString.Reader, destFormat:
XString.Reader, window: Window.Handle, oldInstance: LONG POINTER ← NIL] RETURNS [menuItemProc: Converter.MenuItemProc, destroy:
Converter.DestroyProc, instance: LONG POINTER];

Exported by CvSTCFromVPImpl.
>>

--====================
-- MESSAGES
--====================

MessageKey: TYPE = {
    asciiDoc,
    paraEndsWith,
    codeScheme,
    codeSchemeChoices,
    font,
    fontChoices,
    fontSize,
    fontSizeChoices,
    ignoreTrailing,
    includeTelCodes,
    lineLen,
    lineLenChoices,
    charsSuffix,
    wordWrap,
    endLine,
    endPara,
    replaceUnknown,
    lastPsheetItem,
    left,
    right,
    cr,
    lf,
    nl,
    ff,
    tab,
    createError,
    notPF,
    paginating,
    skippedTableData,
    dfltAVEndParagraph,
    dfltAVReplaceCharacter,
    prefix,
    doneFailed,
    backstop,
    metaError,
    charsOutOfBounds,
    fatalError,
    extraErr0,
    extraErr1,
    dfltVAEndLine,
    dfltVAEndParagraph,
```

```
    dfltVAReplaceCharacter};

END.

LOG
 5-Dec-84 15:01:26 - MSchneider.pa - CREATED
19-Dec-84 15:31:39 - MSchneider - update to BWS 4.0
16-Apr-85 10:40:52 - MSchneider - added some comments and owner statement
28-May-85  9:23:59 - MSchneider - took out messages now in common interface
26-Feb-87 16:17:12 - Caro - Added paginating and spares
18-Mar-87 14:02:39 - Caro - Completely rewritten for Enhancements I
24-Nov-87 16:51:13 - Erickson - added aToVDfltMeta to change A to V paraEndsWith default
                        from <CR><LF> to <CR>
```

```
-- File: CvSTCDataImpl.mesa
-- Trow 11-Sep-89 12:42:03

-- Last Revised by: Erickson                    24-Nov-87 16:54:21
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (c) 1987, 1989 by Xerox Corporation.  All rights reserved.

DIRECTORY
  Courier
    USING [Description, DeserializeParameters, Error, Free,
    Parameters, SerializeParameters],
  Converter
    USING [CreateClientFile, CvData, DestroyProc, FindClientFile],
  ConverterMsg
    USING [Get, kvpDocument],
  CvSTC
    USING [Common, CommonData, CommonObj,
    GetMessage, Owners, Problem, TextIDs],
  Environment
    USING [bytesPerPage],
  Heap
    USING [Create, Delete],
  NSFile
    USING [Delete, Error, Handle, nullReference, OpenByReference],
  NSFileStream
    USING [Create, GetLength, Handle, SetLength],
  Stream
    USING [Delete, InvalidOperation],
  Window
    USING [Handle],
  XString
    USING [CopyToNewReaderBody, DescribeReaderBody, nullReaderBody, ReaderBody];

<<
--=============================================================================
-- OVERVIEW:

Data and filed data procedures
--=============================================================================
>>

CvSTCDataImpl: PROGRAM
IMPORTS
  Converter, ConverterMsg, Courier, CvSTC, Heap,
  NSFile, NSFileStream, Stream, XString
EXPORTS
  CvSTC =
BEGIN

--====================
-- CONSTANTS
--====================

keyBits: Key = 2707974433;   --/* never change this value! */

currentVersion: Version = 3;  --/* change this value if you alter the filed data format */
--=============================================================================
-- History of Versions (update each time version number changes)
-- 18-Mar-87 11:48:29 -  1 - First version
-- 12-Feb-89 19:15:55 -  2 - STC version
-- 11-Sep-89 12:23:12 -  3 - STC version
--=============================================================================

--====================
-- TYPES
--====================

Key: TYPE = LONG CARDINAL;

Version: TYPE = INTEGER;

--====================
-- PUBLIC PROCEDURES
--====================

CreateCommon: PUBLIC PROC [cvData: Converter.CvData, options: BOOLEAN, window: Window.Handle, owner: CvSTC.Owners] RETURNS [my:
CvSTC.Common] = {
    z: UNCOUNTED ZONE ← Heap.Create[initial: 16, increment: 28];

    my ← z.NEW[CvSTC.CommonData ← [
      cvData: cvData,
      options: options,
      window: window,
      owner: owner,
      ref: NSFile.nullReference,
      textRb: ALL[XString.nullReaderBody],
      text: ALL[NIL],
      z: z]];

    --/* find client file */
    BEGIN
      ENABLE UNWIND => Heap.Delete[z];
    prefix: XString.ReaderBody ← CvSTC.GetMessage[prefix];
    src: XString.ReaderBody ← CvSTC.GetMessage[asciiDoc];
    dst: XString.ReaderBody ← ConverterMsg.Get[ConverterMsg.kvpDocument];
```

```
    my.ref ← Converter.FindClientFile[
      cvData: cvData,
      srcFormat: @src,
      destFormat: @dst,
      prefix: @prefix];

    IF my.ref = NSFile.nullReference THEN
        {
        --/* file never created, so initialize */
        InitFiledData[my];  --/* fills in my.ref */
        };

    --/* read data */
    BEGIN
      ENABLE CvSTC.Problem =>
        {
        file: NSFile.Handle ← NSFile.OpenByReference[my.ref];
        --/* get rid of old file, reinitialize */
        NSFile.Delete[file];
        InitFiledData[my];
        CONTINUE;
        };
    LoadFiledData[my];
    END;

    END;
};


DestroyCommon: PUBLIC Converter.DestroyProc = {
<< = PROCEDURE [instance: LONG POINTER];
>>
    my: CvSTC.Common ← instance;
    z: UNCOUNTED ZONE;

    IF my = NIL THEN RETURN;
    z ← my.z;
    Heap.Delete[z];
};


InitFiledData: PUBLIC PROC [my: CvSTC.Common] = {
    myObj: CvSTC.CommonData;
    avPara: XString.ReaderBody ← CvSTC.GetMessage[dfltAVEndParagraph];
    avChar: XString.ReaderBody ← CvSTC.GetMessage[dfltAVReplaceCharacter];
    vaLine: XString.ReaderBody ← CvSTC.GetMessage[dfltVAEndLine];
    vaPara: XString.ReaderBody ← CvSTC.GetMessage[dfltVAEndParagraph];
    vaChar: XString.ReaderBody ← CvSTC.GetMessage[dfltVAReplaceCharacter];

    --/* make dummy filed data */
    myObj.textRb ← my.textRb ← [
      paraEndsWith: avPara,
      atovReplaceUnknown: avChar,
      endLine: vaLine,
      endPara: vaPara,
      vtoaReplaceUnknown: vaChar];
    myObj.text ← ALL[NIL];

    --/* create client file */
    BEGIN
    prefix: XString.ReaderBody ← CvSTC.GetMessage[prefix];
    src: XString.ReaderBody ← CvSTC.GetMessage[asciiDoc];
    dst: XString.ReaderBody ← ConverterMsg.Get[ConverterMsg.kvpDocument];

    my.ref ← Converter.CreateClientFile[
      cvData: my.cvData,
      srcFormat: @src,
      destFormat: @dst,
      prefix: @prefix];
    END;

    myObj.ref ← my.ref;
    myObj.z ← my.z;
    myObj.owner ← backstop;  --/* let StoreFiledData know we are initializing */
    --/* store */
    StoreFiledData[@myObj];
};


LoadFiledData: PUBLIC PROC [my: CvSTC.Common] = {
    sh: NSFileStream.Handle ← [NIL];
    file: NSFile.Handle;
    parms: Courier.Parameters;
    tz: UNCOUNTED ZONE ← NIL;

    --/* read filed data */
    BEGIN
      ENABLE
        {
        Courier.Error, Stream.InvalidOperation => NSFile.Error[[access[fileDamaged]]];
        UNWIND =>
            {
            IF sh # NSFileStream.Handle[NIL] THEN Stream.Delete[sh];
            IF tz # NIL THEN Heap.Delete[tz];
            };
        };
```

```
    --/* open data file */
    file ← NSFile.OpenByReference[my.ref];

    --/* open read stream on data file */
    sh ← NSFileStream.Create[file: file, closeOnDelete: TRUE];

    --/* create temporary zone for disjoint data */
    tz ← Heap.Create[(NSFileStream.GetLength[sh]/Environment.bytesPerPage) + 2];

    --/* read key */
    BEGIN
    key: Key;

    parms ← [location: @key, description: DescribeKey];
    Courier.DeserializeParameters[parms, sh, tz];
    IF key # keyBits THEN
        {
        --/* quit */
        Courier.Free[parms, tz];
        Stream.Delete[sh];
        sh ← [NIL];
        SIGNAL CvSTC.Problem[obsoleteDataFile];
        };
    Courier.Free[parms, tz];
    END;

    --/* read version */
    BEGIN
    ver: Version;

    parms ← [location: @ver, description: DescribeVersion];
    Courier.DeserializeParameters[parms, sh, tz];
    IF ver # currentVersion THEN
        {
        --/* quit */
        Courier.Free[parms, tz];
        Stream.Delete[sh];
        sh ← [NIL];
        SIGNAL CvSTC.Problem[obsoleteDataFile];
        };
    Courier.Free[parms, tz];
    END;

    --/* read commonObj */
    parms ← [location: @my.f, description: DescribeCommonObj];
    Courier.DeserializeParameters[parms, sh, tz];

    --/* read paraEndsWith */
    BEGIN
    rb: XString.ReaderBody;

    parms ← [location: @rb, description: XString.DescribeReaderBody];
    Courier.DeserializeParameters[parms, sh, tz];
    my.textRb[paraEndsWith] ← XString.CopyToNewReaderBody[@rb, my.z];
    Courier.Free[parms, tz];
    END;

    --/* read atovReplaceUnknown */
    BEGIN
    rb: XString.ReaderBody;

    parms ← [location: @rb, description: XString.DescribeReaderBody];
    Courier.DeserializeParameters[parms, sh, tz];
    my.textRb[atovReplaceUnknown] ← XString.CopyToNewReaderBody[@rb, my.z];
    Courier.Free[parms, tz];
    END;

    --/* read endLine */
    BEGIN
    rb: XString.ReaderBody;

    parms ← [location: @rb, description: XString.DescribeReaderBody];
    Courier.DeserializeParameters[parms, sh, tz];
    my.textRb[endLine] ← XString.CopyToNewReaderBody[@rb, my.z];
    Courier.Free[parms, tz];
    END;

    --/* read endPara */
    BEGIN
    rb: XString.ReaderBody;

    parms ← [location: @rb, description: XString.DescribeReaderBody];
    Courier.DeserializeParameters[parms, sh, tz];
    my.textRb[endPara] ← XString.CopyToNewReaderBody[@rb, my.z];
    Courier.Free[parms, tz];
    END;

    --/* read vtoaReplaceUnknown */
    BEGIN
    rb: XString.ReaderBody;

    parms ← [location: @rb, description: XString.DescribeReaderBody];
    Courier.DeserializeParameters[parms, sh, tz];
    my.textRb[vtoaReplaceUnknown] ← XString.CopyToNewReaderBody[@rb, my.z];
    Courier.Free[parms, tz];
    END;
```

```
    END;

    --/* clean up */
    Stream.Delete[sh];
    Heap.Delete[tz];
};


--=============================================================================
-- StoreFiledData
-- * This is tricky, since common data is used.  This routine could be called
-- * three different times, with different subsets of data, but the whole
-- * file must be written each time.
--=============================================================================

StoreFiledData: PUBLIC PROC [my: CvSTC.Common] = {
    dataFile: NSFile.Handle;
    sh: NSFileStream.Handle;
    parms: Courier.Parameters;
    tmpMy: CvSTC.CommonData;

    --/* fill out dummy */
    tmpMy ← my↑;
    IF my.owner # backstop THEN
        LoadFiledData[@tmpMy];

    --/* open data file */
    dataFile ← NSFile.OpenByReference[my.ref];

    --/* open stream on file */
    sh ← NSFileStream.Create[file: dataFile, closeOnDelete: TRUE];
    NSFileStream.SetLength[fileStream: sh, lengthInBytes: 0];

    --/* write data */
    BEGIN
      ENABLE
        {
        Courier.Error, Stream.InvalidOperation => NSFile.Error[[access[fileDamaged]]];
        UNWIND => Stream.Delete[sh];
        };

    --/* write key */
    BEGIN
    key: Key ← keyBits;

    parms ← [location: @key, description: DescribeKey];
    Courier.SerializeParameters[parms, sh];
    END;

    --/* write version */
    BEGIN
    ver: Version ← currentVersion;

    parms ← [location: @ver, description: DescribeVersion];
    Courier.SerializeParameters[parms, sh];
    END;

    --/* update portions of data record */
    SELECT my.owner FROM
        AtoVsrc =>
            {
            tmpMy.textRb[paraEndsWith] ← my.textRb[paraEndsWith];
            tmpMy.f.avCodeScheme ← my.f.avCodeScheme;
            };
        AtoVdst =>
            {
            tmpMy.f.font ← my.f.font;
            tmpMy.f.fontSize ← my.f.fontSize;
            tmpMy.textRb[atovReplaceUnknown] ← my.textRb[atovReplaceUnknown];
            tmpMy.f.ignoreTrailing ← my.f.ignoreTrailing;
            tmpMy.f.includeTelCodes ← my.f.includeTelCodes;
            };
        VtoAdst =>
            {
            tmpMy.f.vaCodeScheme ← my.f.vaCodeScheme;
            tmpMy.f.lineLen ← my.f.lineLen;
            tmpMy.f.charsSuffix ← my.f.charsSuffix;
            tmpMy.f.wordWrap ← my.f.wordWrap;
            tmpMy.textRb[endLine] ← my.textRb[endLine];
            tmpMy.textRb[endPara] ← my.textRb[endPara];
            tmpMy.textRb[vtoaReplaceUnknown] ← my.textRb[vtoaReplaceUnknown];
            };
        ENDCASE;

    --/* write filed data record */
    parms ← [location: @tmpMy.f, description: DescribeCommonObj];
    Courier.SerializeParameters[parms, sh];

    --/* write paraEndsWith string */
    parms ← [location: @tmpMy.textRb[paraEndsWith], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write atovReplaceUnknown string */
    parms ← [location: @tmpMy.textRb[atovReplaceUnknown], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];
```

```
    --/* write endLine string */
    parms ← [location: @tmpMy.textRb[endLine], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write endPara string */
    parms ← [location: @tmpMy.textRb[endPara], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write vtoaReplaceUnknown string */
    parms ← [location: @tmpMy.textRb[vtoaReplaceUnknown], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    END;

    Stream.Delete[sh];
};

--====================
-- PROCEDURES
--====================

DescribeKey: Courier.Description = {
    p: LONG POINTER TO Key = notes.noteSize[SIZE[Key]];
    notes.noteLongCardinal[p];
};

DescribeVersion: Courier.Description = {
    p: LONG POINTER TO Version = notes.noteSize[SIZE[Version]];
};

DescribeCommonObj: Courier.Description = {
    p: LONG POINTER TO CvSTC.CommonObj = notes.noteSize[
      SIZE[CvSTC.CommonObj]];
};

END...

LOG
16-Mar-87 14:06:16 - Caro - Created
24-Nov-87 16:55:56 - Erickson - Changed default setting of paraEndsWith
                      to <CR> instead of <CR><LF>
```

```
-- File: CvSTCFromVPImpl.mesa
-- 11-Sep-89 12:32:11

-- Last Revised by: Caro                          16-Sep-87 12:21:45
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (c) 1987, 1989 by Xerox Corporation.  All rights reserved.

DIRECTORY
  Ascii
    USING [CR, FF, LF, SP, TAB],
  BackgroundProcess
    USING [UserAbort],
  BWSZone
    USING [logonSession, Permanent],
  Catalog
    USING [GetFile],
  Converter
    USING [ConvertProc, CvData, DependentOptionProc, DestroyProc, MenuItemProc,
    PostMessage],
  ConverterMsg,
  CvSTC
    USING [Common, CommonMenu, CreateCommon, CreateFW,
    DestroyCommon, GetMessage, limited, MessageKey, Owners,
    ParseItem, Problem, stc, unlimited],
  DocInterchangeDefs
    USING [Close, Doc, Enumerate, EnumProcsRecord, Error, NewParagraphProc,
    Open, OpenStatus, PFCProc, PageBreakProc, TextProc],
  Environment
    USING [wordsPerPage],
  NSAssignedTypes
    USING [tUnspecified],
  NSFile
    USING [Attribute, Create, Close, Delete, Error, Filter, Find, GetReference,
    Handle, nullHandle, nullReference, OpenByReference, Reference, Session],
  NSFileStream
    USING [Create, Handle],
  NSString
    USING [FreeString, String],
  Space
    USING [ScratchMap],
  StarFileTypes
    USING [text],
  Stream
    USING [EndOfStream, GetWord, PutChar, Delete],
  String
    USING [AppendDecimal, MakeString],
  TIP
    USING [UserAbort],
  XChar
    USING [Character, Code, not, Set],
  XMessage
    USING [MsgKey],
  XString
    USING [FromSTRING, InvalidEncoding, NSStringFromReader, Reader, ReaderBody,
    Map, MapCharProc];

<<
--*************************************************************************
-- OVERVIEW:

VP to Telegraph Code conversion.
--*************************************************************************
>>

CvSTCFromVPImpl: PROGRAM
IMPORTS
  BackgroundProcess, BWSZone, Catalog, Converter, ConverterMsg, CvSTC,
  DocInterchangeDefs, NSFile, NSFileStream, NSString, Space, Stream, String,
  TIP, XChar, XString
EXPORTS
  CvSTC =
BEGIN

--====================
-- CONSTANTS
--====================

tabInterval: CARDINAL = 8;

setMapSize: CARDINAL = SIZE[VPToAsciiSetMap];
charMapSize: CARDINAL = SIZE[VPToAsciiCharMap];

--====================
-- TYPES
--====================

VAData: TYPE = LONG POINTER TO VADataObj;
VADataObj: TYPE = RECORD [
    source: NSFile.Handle,
    output: NSFileStream.Handle,  --/* created from dest */
    cvData: Converter.CvData,
    session: NSFile.Session,
    dst: CvSTC.Common,
    background: BOOLEAN,
    doc: DocInterchangeDefs.Doc,
    putc: PutCProc,
```

```
    firstPara: BOOLEAN,
    line: LONG STRING,          --/* line buffer */
    n: CARDINAL,                --/* index of next char in line buffer */
    pos: CARDINAL,              --/* current position on virtual line */
    max: CARDINAL,              --/* last column in line */
    lastWhite: CARDINAL,        --/* last white */
    wordWrap: BOOLEAN,
    after: CARDINAL,            --/* number of eop strings to output */
                                --/*   for previous paragraph */
    z: UNCOUNTED ZONE];

PutCProc: TYPE = PROC [va: VAData, c: CHARACTER];

VPToAsciiSetMap: TYPE = ARRAY [0..256) OF LONG POINTER TO VPToAsciiCharMap;
VPToAsciiCharMap: TYPE = ARRAY [0..256) OF CARDINAL;

--====================
-- GLOBALS
--====================

Global: TYPE = RECORD [
    prc: LONG POINTER TO VPToAsciiSetMap,
    roc: LONG POINTER TO VPToAsciiSetMap,
    gz: UNCOUNTED ZONE];

g: Global;

tct: LONG POINTER TO VPToAsciiSetMap;

--====================
-- PUBLIC PROCEDURES
--====================

VPToAscii: PUBLIC Converter.ConvertProc = {
<< = PROCEDURE [source: NSFile.Handle, cvData: Converter.CvData, session: NSFile.Session, srcInstance: LONG POINTER ← NIL, dstInstance:
LONG POINTER ← NIL, background: BOOLEAN ← FALSE] RETURNS [dest: NSFile.Handle ← LOOPHOLE[0]];
>>
  ENABLE CvSTC.Problem, NSFile.Error, XString.InvalidEncoding =>
      {
      msgRb: XString.ReaderBody ← CvSTC.GetMessage[fatalError];

      Post[msgRb, cvData];
      CONTINUE;
      };

    IF source = NSFile.nullHandle THEN RETURN;
    dest ← VtoA[source, cvData, session, srcInstance, dstInstance, background];
};


<<
--=============================================================================
This DependentOptionProc creates instance data with CreateCommon.  The data is distinguished by the owner variable.  The CommonObj within
CvSTC.CommonData is the data structure written to the client file stored as the icon properties.  Only those fields pertaining to the
owner are used.
--=============================================================================
>>


VPToAsciiDstOps: PUBLIC Converter.DependentOptionProc = {
<< = PROCEDURE [options: BOOLEAN ← TRUE, cvData: Converter.CvData, which: Converter.FormatToUse, srcFormat: XString.Reader, destFormat:
XString.Reader, window: Window.Handle, oldInstance: LONG POINTER ← NIL] RETURNS [menuItemProc: Converter.MenuItemProc, destroy:
Converter.DestroyProc, instance: LONG POINTER];
>>
    owner: CvSTC.Owners ← VtoAdst;

    menuItemProc ← CvSTC.CommonMenu;
    destroy ← CvSTC.DestroyCommon;
    IF oldInstance = NIL THEN
        instance ← CvSTC.CreateCommon[cvData, options, window, owner ! NSFile.Error, CvSTC.Problem => {owner ← backstop; instance ← NIL;
        CONTINUE}]
    ELSE
        {
        my: CvSTC.Common ← oldInstance;
        my.window ← window;  --/* AR 13535: update window handle */
        instance ← my;
        };


    --/* make formwindow */
    CvSTC.CreateFW[instance, window, owner];
};

--====================
-- PROCEDURES
--====================

VtoA: Converter.ConvertProc = {
    aborted: BOOLEAN ← FALSE;
    dataSkipped: BOOLEAN ← FALSE;
    attr: ARRAY [0..1) OF NSFile.Attribute ← [[type[StarFileTypes.text]]];
    enumProcs: DocInterchangeDefs.EnumProcsRecord ← [
      newParagraphProc: EndPrevAsciiPara,
      pageBreakProc: AddAsciiPage,
      textProc: AddAsciiText,
      pfcProc: AddAsciiPFC];
    openStatus: DocInterchangeDefs.OpenStatus;
```

```
    vaData: VADataObj;  --/* only works if Enumerate doesn't FORK */
    dst: CvSTC.Common ← NIL;

    --/* initialize instance data */
    IF dstInstance = NIL THEN
        {
        ENABLE NSFile.Error, CvSTC.Problem =>
            {
            msgRb: XString.ReaderBody ← CvSTC.GetMessage[extraErr0];  --" Unrecoverable ASCII conversion error: damaged converter icon. "
            Converter.PostMessage[
              msg: @msgRb,
              cvData: cvData,
              cr: FALSE,
              clear: FALSE];
            GOTO terminate;
            };
        key: CvSTC.MessageKey ← CvSTC.MessageKey.FIRST;  --/* dummy */

        --/* we only care about dst */
        dst ← CvSTC.CreateCommon[cvData, FALSE, NIL, VtoAdst];

        dst.text[endLine] ← CvSTC.ParseItem[
          my: dst,
          r: @dst.textRb[endLine],
          item: key].ls;

        dst.text[endPara] ← CvSTC.ParseItem[
          my: dst,
          r: @dst.textRb[endPara],
          item: key].ls;

        dst.text[vtoaReplaceUnknown] ← CvSTC.ParseItem[
          my: dst,
          r: @dst.textRb[vtoaReplaceUnknown],
          item: key].ls;
        EXITS terminate => RETURN;
        }
    ELSE
        {
        dst ← dstInstance;
        };

    vaData ← [
         source: source,
         output: [NIL],
         cvData: cvData,
         session: session,
         dst: dst,
         background: background,
         doc: TRASH,
         putc: IF dst.f.lineLen = CvSTC.unlimited THEN UnbufferedPutC ELSE BufferedPutC,
         firstPara: TRUE,
         line: NIL,
         n: 0,
         pos: 0,
         max: 0,
         lastWhite: CARDINAL.LAST,
         wordWrap: dst.f.wordWrap.value,
         after: 0,
         z: dst.z];

    IF dst.f.lineLen = CvSTC.limited THEN
        {
        --/* ASSERT: charsSuffix IN [10..256] */
        --/* create line buffer */
        vaData.line ← String.MakeString[z: vaData.z, maxlength: dst.f.charsSuffix];
        vaData.line.length ← vaData.line.maxlength;
        vaData.max ← dst.f.charsSuffix - 1;
        IF dst.text[endLine] # NIL AND dst.text[endLine].length < vaData.max THEN
            {
            --/* max column is limit less visible end-of-line characters */
            FOR i: CARDINAL IN [0..dst.text[endLine].length) DO
                SELECT dst.text[endLine][i] FROM
                    Ascii.CR, Ascii.LF, Ascii.FF => NULL;
                    ENDCASE => vaData.max ← vaData.max - 1;
            ENDLOOP;
            };
        };

    IF dst.f.vaCodeScheme = CvSTC.stc THEN tct ← g.prc ELSE tct ← g.roc;

    BEGIN
      ENABLE
        {
        NSFile.Error => GOTO nsErr;
        DocInterchangeDefs.Error => GOTO docErr;
        UNWIND => IF dstInstance = NIL THEN
            {
            CvSTC.DestroyCommon[dst];
            dst ← NIL;
            };
        };

    [vaData.doc, openStatus] ← DocInterchangeDefs.Open[
      docFileRef: NSFile.GetReference[source, vaData.session],
      session: vaData.session];
    IF openStatus # ok THEN GOTO docErr;
```

```
    dest ← NSFile.Create[
      directory: NSFile.nullHandle,
      attributes: DESCRIPTOR[attr],
      session: session ! NSFile.Error => {
          IF error = [space[mediumFull]] THEN
              Post[ConverterMsg.Get[ConverterMsg.koutOfSpace], vaData.cvData]
          ELSE
              Post[CvSTC.GetMessage[createError], vaData.cvData];
          GOTO nsErr}];

    vaData.output ← NSFileStream.Create[
      file: dest,
      closeOnDelete: FALSE,
      session: vaData.session];

    dataSkipped ← DocInterchangeDefs.Enumerate[
      textContainer: [doc[vaData.doc]],
      procs: @enumProcs,
      clientData: @vaData ! ABORTED => {
        dataSkipped ← TRUE;
        aborted ← TRUE;
        Post[ConverterMsg.Get[ConverterMsg.kuserAbort], vaData.cvData];
        CONTINUE}];

    --/* AR 13705: flush any remaining text */
    --/* ASSERT: n = 0 IF dst.f.lineLen # CvSTC.limited */
    IF NOT aborted AND vaData.n > 0 THEN
        {
        RawPuts[@vaData, vaData.line, vaData.n];
        --/* AR 14393: terminate last paragraph */
        RawPuts[@vaData, vaData.dst.text[endPara]];
        };

    Stream.Delete[vaData.output ! NSFile.Error => {
      IF error = [space[mediumFull]] THEN
          Post[ConverterMsg.Get[ConverterMsg.koutOfSpace], vaData.cvData]
      ELSE
          Post[ConverterMsg.Get[ConverterMsg.kunknownProblem], vaData.cvData];
      NSFile.Delete[dest, vaData.session];
      dest ← NSFile.nullHandle;
      GOTO nsErr}];
    IF dataSkipped THEN
          Post[ConverterMsg.Get[ConverterMsg.kdataSkipped], vaData.cvData];

    DocInterchangeDefs.Close[@vaData.doc];

    EXITS
      nsErr => {
        IF vaData.doc # NIL THEN
            DocInterchangeDefs.Close[@vaData.doc ! DocInterchangeDefs.Error => CONTINUE]};
      docErr => {
        key: XMessage.MsgKey ←
          SELECT openStatus FROM
            malFormed, incompatible => ConverterMsg.kincompatible,
            outOfDiskSpace, outOfVM => ConverterMsg.koutOfSpace,
            ENDCASE => ConverterMsg.kcantOpen;

        Post[ConverterMsg.Get[key], vaData.cvData];
        IF vaData.doc # NIL THEN
            DocInterchangeDefs.Close[@vaData.doc ! DocInterchangeDefs.Error => CONTINUE];
        dest ← NSFile.nullHandle};
    END;
    IF vaData.line # NIL THEN vaData.z.FREE[@vaData.line];
    --/* destroy instance data if created by this proc */
    IF dstInstance = NIL AND dst # NIL THEN CvSTC.DestroyCommon[dst];
};


Post: PROC [msgRb: XString.ReaderBody, cvData: Converter.CvData] = {
    Converter.PostMessage[
      msg: @msgRb,
      cvData: cvData,
      cr: TRUE,
      clear: FALSE];
};


CheckAbort: PROC [background: BOOLEAN] RETURNS [yes: BOOLEAN] = INLINE {
    yes ← (background AND BackgroundProcess.UserAbort[]) OR
      (NOT background AND TIP.UserAbort[NIL]);
};


--/* Enumeration Procs */

AddAsciiPage: DocInterchangeDefs.PageBreakProc = {
<< = PROCEDURE [clientData: LONG POINTER, fontProps: DocInterchangePropsDefs.ReadonlyFontProps] RETURNS [stop: BOOL ← FALSE];
>>
    va: VAData ← clientData;
    -- form feed appended for a new page
    va.putc[va, Ascii.FF];
};


EndPrevAsciiPara:  DocInterchangeDefs.NewParagraphProc = {
<< = PROCEDURE [clientData: LONG POINTER, fontProps: DocInterchangePropsDefs.ReadonlyFontProps, paraProps:
```

```
DocInterchangePropsDefs.ReadonlyParaProps] RETURNS [stop: BOOL ← FALSE];
>>
    va: VAData ← clientData;

    --/* ASSERT: n = 0 IF dst.f.lineLen # CvSTC.limited */
    IF va.n > 0 THEN RawPuts[va, va.line, va.n];  --/* flush any pending text */

    --/* a new para char means we terminate the previous ASCII paragraph */
    IF va.firstPara AND paraProps.basicProps.preLeading < paraProps.basicProps.lineHeight THEN
        va.firstPara ← FALSE
    ELSE
        {
        --/* ceiling to next highest line */
        newlines: CARDINAL ←
          (paraProps.basicProps.preLeading + paraProps.basicProps.lineHeight - 1)  /
          paraProps.basicProps.lineHeight;

        IF NOT va.firstPara THEN
            {
            --/* end previous paragraph */
            RawPuts[va, va.dst.text[endPara]];

            --/* append endLine strings for AFTER paragraph spacing */
            THROUGH [1..va.after] DO
                RawPuts[va, va.dst.text[endLine]];
            ENDLOOP;
            };

        --/* this newPara character contains properties for the FOLLOWING *
        --/*    paragraph, therefore output BEFORE line spacing first */
        THROUGH [1..newlines] DO
            RawPuts[va, va.dst.text[endLine]];
        ENDLOOP;
        va.firstPara ← FALSE;
        };
    va.n ← 0;  --/* reset line index */
    va.pos ← 0;  --/* reset line position */
    va.lastWhite ← CARDINAL.LAST;  --/* reset last white */
    --/* save AFTER line spacing */
    va.after ←
      (paraProps.basicProps.postLeading + paraProps.basicProps.lineHeight - 1) / paraProps.basicProps.lineHeight;
};


AddAsciiPFC: DocInterchangeDefs.PFCProc = {};

<<
--=============================================================================
AddAsciiText
This procedure does the bulk of the text handling.  Its main purpose is to translate VP characters into ASCII characters, according to
the user's encoding selection.
--=============================================================================
>>

AddAsciiText: DocInterchangeDefs.TextProc = {
<< = PROCEDURE [clientData: LONG POINTER, fontProps: DocInterchangePropsDefs.ReadonlyFontProps, text: XString.Reader, textEndContext:
XString.Context] RETURNS [stop: BOOL ← FALSE];
>>
    va: VAData = clientData;

    --/* local procs */
    XnToXO: XString.MapCharProc = {
        xset: [0..256) ← XChar.Set[c];
        xcode: [0..256) ← XChar.Code[c];
        mapc: CARDINAL;
        putc: PutCProc = va.putc;
        zString: LONG STRING ← "0000"L;
        tString: LONG STRING ← [5];

        IF tct[xset] # NIL THEN {
            mapc ← tct[xset][xcode];
            SELECT mapc FROM
                IN [10000..19999] => {
                    String.AppendDecimal[tString, mapc - 10000];
                    zString.length ← 4 - tString.length;
                    Puts[va, zString];
                    Puts[va, tString];
                    putc[va, Ascii.SP];
                    };
                12B, 15B => {
                    Puts[va, va.dst.text[endLine]];
                    IF va.n > 0 THEN FlushLine[va];
                    };
                IN [1..377B] => {
                    putc[va, VAL[mapc]];
                    };
                ENDCASE => {
                    Puts[va, va.dst.text[vtoaReplaceUnknown]];
                    };
            }
        ELSE Puts[va, va.dst.text[vtoaReplaceUnknown]];
        stop ← FALSE;
    };

    --/* begin code */
    IF CheckAbort[va.background] THEN ERROR ABORTED;
    [] ← XString.Map[r: text, proc: XnToXO];
```

408

```
};

--/* put procs */

UnbufferedPutC: PutCProc = {
    Stream.PutChar[va.output, c];
};


BufferedPutC: PutCProc = {
    line: LONG STRING ← va.line;
    output: NSFileStream.Handle ← va.output;

    IF va.pos > va.max THEN
        {
        IF va.wordWrap THEN
            {
            offset: CARDINAL;

            --/* determine offset to new text */
            IF va.lastWhite = CARDINAL.LAST THEN
                {
                IF va.n > 0 THEN
                    {
                    va.lastWhite ← va.n - 1;
                    offset ← va.n;
                    }
                ELSE
                    offset ← va.lastWhite ← 0;
                }
            ELSE
                offset ← va.lastWhite + 1;

            --/* flush to mark */
            FOR i: CARDINAL IN [0..va.lastWhite] DO
                Stream.PutChar[output, line[i]];
            ENDLOOP;

            --/* end line */
            RawPuts[va, va.dst.text[endLine]];

            --/* restore line */
            FOR i: CARDINAL IN [offset..va.n) DO
                line[i-offset] ← line[i];
            ENDLOOP;
            va.n ← va.n - offset;
            va.lastWhite ← CARDINAL.LAST;

            --/* reset pos */
            va.pos ← 0;
            FOR i: CARDINAL IN [0..va.n) DO
                va.pos ← IF line[i] = Ascii.TAB THEN
                    ((va.pos / tabInterval) + 1) * tabInterval
                ELSE IF (c = Ascii.CR OR c = Ascii.LF) THEN
                    va.pos
                ELSE
                    va.pos + 1;
            ENDLOOP;
            }
        ELSE
            {
            RawPuts[va, line, va.n];
            --/* end line */
            RawPuts[va, va.dst.text[endLine]];
            va.n ← 0;
            va.pos ← 0;
            };
        };

    IF va.n >= line.length THEN
        {
        RawPuts[va, line];
        va.n ← va.pos ← 0;
        va.lastWhite ← CARDINAL.LAST;
        };

    --/* append character */
    line[va.n] ← c;

    IF c = Ascii.SP THEN va.lastWhite ← va.n;
    va.pos ← IF c = Ascii.TAB THEN
        ((va.pos / tabInterval) + 1) * tabInterval
    ELSE IF (c = Ascii.CR OR c = Ascii.LF) THEN
        va.pos
    ELSE
        va.pos + 1;

    va.n ← va.n + 1;
};


--/* put a string */
Puts: PROC [va: VAData, s: LONG STRING] = {
    IF s = NIL THEN RETURN;
    IF s.length = 0 THEN RETURN;
```

```
    FOR i: CARDINAL IN [0..s.length) DO
        va.putc[va, s[i]];
    ENDLOOP;
};


--/* raw put string */
RawPuts: PROC [va: VAData, s: LONG STRING, limit: CARDINAL ← CARDINAL.LAST] = {
    IF s = NIL THEN RETURN;
    IF s.length = 0 THEN RETURN;

    IF limit = CARDINAL.LAST THEN limit ← s.length;
    FOR i: CARDINAL IN [0..limit) DO
        Stream.PutChar[va.output, s[i]];
    ENDLOOP;
};


FlushLine: PROC [va: VAData] = {
    RawPuts[va, va.line, va.n];
    va.n ← va.pos ← 0;
    va.lastWhite ← CARDINAL.LAST;
};


--/* Table Procs */

InvertTCFile: PROC [tableName: XString.Reader, vat: LONG POINTER TO VPToAsciiSetMap] = {
    tf: NSFile.Handle;
    ts: NSFileStream.Handle;
    xchar: XChar.Character;
    charMap: LONG POINTER TO VPToAsciiCharMap;
    xset: [0..256);
    telcode: CARDINAL;

    tf ← GetTableFile[tableName];
    ts ← NSFileStream.Create[file: tf];
    telcode ← 0;
    DO
        ENABLE {Stream.EndOfStream => EXIT};
        xchar ← LOOPHOLE[Stream.GetWord[ts]];
        IF xchar # XChar.not THEN {
            xset ← XChar.Set[xchar];
            charMap ← vat[xset];
            IF charMap = NIL THEN {
                vat[xset] ← Space.ScratchMap[(charMapSize + Environment.wordsPerPage-1) / Environment.wordsPerPage];
                FOR c: CARDINAL IN [0..256) DO
                    vat[xset][c] ← 0;
                ENDLOOP;
                charMap ← vat[xset];
            };
            charMap[XChar.Code[xchar]] ← telcode + 10000;
        };
        telcode ← telcode + 1;
    ENDLOOP;
    Stream.Delete[ts];
    ts ← [NIL];
};


GetTableFile: PROC [tableName: XString.Reader] RETURNS [file: NSFile.Handle] = {
    -- assume folder is in System catalog
    folderName: XString.ReaderBody ← XString.FromSTRING["Transliteration Tables"L];
    ref: NSFile.Reference ← TRASH;
    ref ← GetFile[@folderName, tableName];
    file ← NSFile.OpenByReference[ref];
};


GetFile: PROC [folderName, fileName: XString.Reader]
    RETURNS [file: NSFile.Reference ← NSFile.nullReference] = {
    directory: NSFile.Handle ← TRASH;

    FileFromName: PROC [name: XString.Reader] = {
        nsName: NSString.String ← XString.NSStringFromReader[
          r: name, z: BWSZone.logonSession];
        handle: NSFile.Handle ← NSFile.nullHandle;
        filters: ARRAY [0..2) OF NSFile.Filter ← [
          [matches[[name[nsName]]]],
          [equal[[type[NSAssignedTypes.tUnspecified]]]]];

        handle ← NSFile.Find[
          directory: directory,
          scope: [filter: [and[DESCRIPTOR [filters]]]]
            ! NSFile.Error => {handle ← NSFile.nullHandle; CONTINUE}];

        IF handle # NSFile.nullHandle THEN {
          file ← NSFile.GetReference[handle];
          NSFile.Close[handle]};
        NSString.FreeString[z: BWSZone.logonSession, s: nsName];
    };

    directory ← Catalog.GetFile[name: folderName, readonly: TRUE];
    FileFromName[fileName];
    NSFile.Close[directory];
};
```

```
ZzInit: PROC = {
    gz: UNCOUNTED ZONE = BWSZone.Permanent[];
    fileName: XString.ReaderBody;

    --/* these Spaces should not be unmapped while this application is loaded */
    g ← [
      prc: Space.ScratchMap[(setMapSize + Environment.wordsPerPage-1) / Environment.wordsPerPage],
      roc: Space.ScratchMap[(setMapSize + Environment.wordsPerPage-1) / Environment.wordsPerPage],
      gz: gz];

    FOR s: CARDINAL IN [0..256) DO
        g.prc[s] ← NIL;
        g.roc[s] ← NIL;
    ENDLOOP;

    g.prc[0] ← Space.ScratchMap[(charMapSize + Environment.wordsPerPage-1) / Environment.wordsPerPage];
    g.prc[41B] ← Space.ScratchMap[(charMapSize + Environment.wordsPerPage-1) / Environment.wordsPerPage];
    g.prc[357B] ← Space.ScratchMap[(charMapSize + Environment.wordsPerPage-1) / Environment.wordsPerPage];

    g.roc[0] ← Space.ScratchMap[(charMapSize + Environment.wordsPerPage-1) / Environment.wordsPerPage];
    g.roc[41B] ← Space.ScratchMap[(charMapSize + Environment.wordsPerPage-1) / Environment.wordsPerPage];
    g.roc[357B] ← Space.ScratchMap[(charMapSize + Environment.wordsPerPage-1) / Environment.wordsPerPage];

    FOR c: CARDINAL IN [0..256) DO
        g.prc[0][c] ← c;
        g.prc[41B][c] ← 0;
        g.prc[357B][c] ← 0;
        g.roc[0][c] ← c;
        g.roc[41B][c] ← 0;
        g.roc[357B][c] ← 0;
    ENDLOOP;

    g.prc[0][211B] ← 11B;       -- 211B -> tab
    g.prc[0][244B] ← 44B;       -- dollar -> $
    g.prc[0][252B] ← 42B;       -- leftDoubleQuote -> "
    g.prc[0][272B] ← 42B;       -- rightDoubleQuote -> "
    g.prc[0][251B] ← 47B;       -- leftSingleQuote -> '
    g.prc[0][271B] ← 47B;       -- rightSingleQuote -> '
    g.prc[41B][76B] ← 55B;      -- hyphen -> minus
    g.prc[357B][42B] ← 55B;     -- nonBreakingHyphen -> minus
    g.prc[357B][41B] ← 40B;     -- nonBreakingSpace -> space

    g.roc[0][211B] ← 11B;       -- 211B -> tab
    g.roc[0][244B] ← 44B;       -- dollar -> $
    g.roc[0][252B] ← 42B;       -- leftDoubleQuote -> "
    g.roc[0][272B] ← 42B;       -- rightDoubleQuote -> "
    g.roc[0][251B] ← 47B;       -- leftSingleQuote -> '
    g.roc[0][271B] ← 47B;       -- rightSingleQuote -> '
    g.roc[41B][76B] ← 55B;      -- hyphen -> minus
    g.roc[357B][42B] ← 55B;     -- nonBreakingHyphen -> minus
    g.roc[357B][41B] ← 40B;     -- nonBreakingSpace -> space

    fileName ← XString.FromSTRING["PRC.tcTable"L];
    InvertTCFile[@fileName, g.prc];

    fileName ← XString.FromSTRING["ROC.tcTable"L];
    InvertTCFile[@fileName, g.roc];
};


--/* main line code */

ZzInit[];


END...

LOG
16-Mar-87 14:06:16 - Caro - Created
26-Jun-87 11:30:20 - Caro - Added error catcher in ConvertProc over CreateCommon,
                            ISO8 now has correct ENDCASE
10-Jul-87 11:31:10 - Caro - Added before/after line spacing
19-Aug-87 11:03:02 - Caro - Fixed AR 13535 by updating oldInstance window
                            Fixed AR 13705 by flushing remaining text
16-Sep-87 12:21:09 - Caro - Fixed AR 14393 by terminating with endPara
```

```
-- File: CvSTCFWImpl.mesa
-- Trow  11-Sep-89 12:36:36

-- Last Revised by: Erickson                    17-Dec-87 16:03:15
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (c) 1987, 1989 by Xerox Corporation.  All rights reserved.

DIRECTORY
  Attention
    USING [Post],
  Converter
    USING [MenuItemProc, ResizeDetailWindow],
  CvSTC,
  FormWindow
    USING [AppendItem, AppendLine, ChoiceChangeProc, ChoiceItems, Create,
    GetBooleanItemValue, GetChoiceItemValue,
    GetIntegerItemValue, GetTextItemValue,
    HasBeenChanged, HasAnyBeenChanged, LayoutProc, Line, MakeItemsProc,
    MakeBooleanItem, MakeChoiceItem, MakeIntegerItem, MakeTextItem,
    MinDimsChangeProc,
    SetBooleanItemValue, SetChoiceItemValue, SetIntegerItemValue,
    SetTabStops, SetTextItemValue,
    SetVisibility, TabStops, SetSelection, SetInputFocus],
  FormWindowMessageParse
    USING [FreeChoiceItems, ParseChoiceItemMessage],
  NSFile
    USING [Error],
  Window
    USING [Handle],
  XString
    USING [FreeReaderBytes, FreeWriterBytes, NewWriterBody, nullReaderBody,
    ReaderBody, WriterBody, InvalidNumber, Overflow];

<<
--============================================================================
-- OVERVIEW:

Formwindow procedures
--============================================================================
>>

CvSTCFWImpl: PROGRAM
IMPORTS
  Attention, Converter, CvSTC,
  FormWindow, FormWindowMessageParse, NSFile, XString
EXPORTS
  CvSTC =
BEGIN

--====================
-- CONSTANTS
--====================

textWidth: CARDINAL = 320;
tabStopInterval: CARDINAL = CvSTC.pointsBetweenItems/2;

--====================
-- TYPES
--====================

--====================
-- PUBLIC PROCEDURES
--====================


CommonMenu: PUBLIC Converter.MenuItemProc = {
<< = PROCEDURE [instance: LONG POINTER, menuItem: PropertySheet.MenuItemType] RETURNS [ok: BOOLEAN ← TRUE];>>
    my: CvSTC.Common = instance;
    avPara: XString.ReaderBody ← CvSTC.GetMessage[dfltAVEndParagraph];
    avChar: XString.ReaderBody ← CvSTC.GetMessage[dfltAVReplaceCharacter],
    vaLine: XString.ReaderBody ← CvSTC.GetMessage[dfltVAEndLine];
    vaPara: XString.ReaderBody ← CvSTC.GetMessage[dfltVAEndParagraph];
    vaChar: XString.ReaderBody ← CvSTC.GetMessage[dfltVAReplaceCharacter];

    IF my = NIL THEN RETURN[ok: TRUE];

    SELECT menuItem FROM
        defaults =>
            {
            SELECT my.owner FROM
                AtoVsrc =>
                    {
                    FormWindow.SetTextItemValue[
                      window: my.window,
                      item: CvSTC.MessageKey.paraEndsWith.ORD,
                      newValue: @avPara,
                      repaint: FALSE];
                    FormWindow.SetChoiceItemValue[
                      window: my.window,
                      item: CvSTC.MessageKey.codeScheme.ORD,
                      newValue: CvSTC.dfltCodeScheme,
                      repaint: FALSE];
                    };
                AtoVdst =>
                    {
                    FormWindow.SetChoiceItemValue[
```

```
                    window: my.window,
                    item: CvSTC.MessageKey.font.ORD,
                    newValue: CvSTC.dfltFont,
                    repaint: FALSE];
                  FormWindow.SetChoiceItemValue[
                    window: my.window,
                    item: CvSTC.MessageKey.fontSize.ORD,
                    newValue: CvSTC.dfltFontSize,
                    repaint: FALSE];
                  FormWindow.SetTextItemValue[
                    window: my.window,
                    item: CvSTC.MessageKey.replaceUnknown.ORD,
                    newValue: @avChar,
                    repaint: FALSE];
                  FormWindow.SetBooleanItemValue[
                    window: my.window,
                    item: CvSTC.MessageKey.ignoreTrailing.ORD,
                    newValue: CvSTC.dfltTrailing,
                    repaint: FALSE];
                  FormWindow.SetBooleanItemValue[
                    window: my.window,
                    item: CvSTC.MessageKey.includeTelCodes.ORD,
                    newValue: CvSTC.dfltTelCodes,
                    repaint: TRUE];
                  };
              VtoAdst =>
                  {
                  FormWindow.SetChoiceItemValue[
                    window: my.window,
                    item: CvSTC.MessageKey.codeScheme.ORD,
                    newValue: CvSTC.dfltCodeScheme,
                    repaint: FALSE];
                  FormWindow.SetChoiceItemValue[
                    window: my.window,
                    item: CvSTC.MessageKey.lineLen.ORD,
                    newValue: CvSTC.dfltLineLen,
                    repaint: FALSE];
                  FormWindow.SetIntegerItemValue[
                    window: my.window,
                    item: CvSTC.MessageKey.charsSuffix.ORD,
                    newValue: CvSTC.dfltChars,
                    repaint: FALSE];
                  FormWindow.SetBooleanItemValue[
                    window: my.window,
                    item: CvSTC.MessageKey.wordWrap.ORD,
                    newValue: CvSTC.dfltWordWrap,
                    repaint: FALSE];
                  FormWindow.SetTextItemValue[
                    window: my.window,
                    item: CvSTC.MessageKey.endLine.ORD,
                    newValue: @vaLine,
                    repaint: FALSE];
                  FormWindow.SetTextItemValue[
                    window: my.window,
                    item: CvSTC.MessageKey.endPara.ORD,
                    newValue: @vaPara,
                    repaint: FALSE];
                  FormWindow.SetTextItemValue[
                    window: my.window,
                    item: CvSTC.MessageKey.replaceUnknown.ORD,
                    newValue: @vaChar,
                    repaint: TRUE];
                  };
              ENDCASE;
          };
          done =>
              {
              ENABLE NSFile.Error, CvSTC.Problem =>
                  {
                  msgRb: XString.ReaderBody ← CvSTC.GetMessage[doneFailed];
                  Attention.Post[@msgRb];
                  GOTO notOK;
                  };
              IF FormWindow.HasAnyBeenChanged[my.window] THEN
                  {
                  ok ← ApplyChanges[my];
                  IF NOT ok THEN GOTO notOK;
                  CvSTC.StoreFiledData[my];
                  };
              EXITS notOK => RETURN[ok: FALSE];
              };
          start =>
              {
              ok ← ApplyChanges[my];
              };
      ENDCASE;
};

CreateFW: PUBLIC PROC [my: CvSTC.Common, window: Window.Handle, owner: CvSTC.Owners] = {
    SELECT owner FROM
        AtoVsrc =>
            {
            FormWindow.Create[
              window: window,
              makeItemsProc: MakeAtoVSrc,
              layoutProc: LayoutAtoVSrc,
```

```
            minDimsChangeProc: GrowParent,
            clientData: my];
        CvSTC.DataToWindow[my, window];
        };
    AtoVdst =>
        {
        FormWindow.Create[
          window: window,
          makeItemsProc: MakeAtoVDst,
          layoutProc: LayoutAtoVDst,
          clientData: my];
        };
    VtoAdst =>
        {
        FormWindow.Create[
          window: window,
          makeItemsProc: MakeVtoADst,
          layoutProc: LayoutVtoADst,
          minDimsChangeProc: GrowParent,
          clientData: my];
        CvSTC.DataToWindow[my, window];
        };
    backstop =>
        {
        FormWindow.Create[
          window: window,
          makeItemsProc: MakeBackstop];
        };
    ENDCASE;
};

--====================
-- PROCEDURES
--====================

ApplyChanges: PROC [my: CvSTC.Common] RETURNS [ok: BOOLEAN ← TRUE] = {
    bufWb: XString.WriterBody ← XString.NewWriterBody[maxLength: 30, z: my.z];
    SELECT my.owner FROM
        AtoVsrc =>
        {
        IF FormWindow.HasBeenChanged[window: my.window, item: CvSTC.MessageKey.paraEndsWith.ORD] THEN
            {
            IF my.textRb[paraEndsWith] # XString.nullReaderBody THEN
                XString.FreeReaderBytes[@my.textRb[paraEndsWith], my.z];
            my.textRb[paraEndsWith] ← FormWindow.GetTextItemValue[
              window: my.window,
              item: CvSTC.MessageKey.paraEndsWith.ORD,
              zone: my.z];
            };
        [ok: ok, ls: my.text[paraEndsWith]] ← CvSTC.ParseItem[
          my: my,
          r: @my.textRb[paraEndsWith],
          item: CvSTC.MessageKey.paraEndsWith,
          buf: @bufWb];
        IF NOT ok THEN RETURN;

        IF FormWindow.HasBeenChanged[window: my.window, item: CvSTC.MessageKey.codeScheme.ORD] THEN
            {
            my.f.avCodeScheme ← FormWindow.GetChoiceItemValue[
              window: my.window,
              item: CvSTC.MessageKey.codeScheme.ORD];
            };
        };
    AtoVdst =>
        {
        IF FormWindow.HasBeenChanged[my.window, CvSTC.MessageKey.font.ORD] THEN
            {
            my.f.font ← FormWindow.GetChoiceItemValue[
              window: my.window,
              item: CvSTC.MessageKey.font.ORD];
            };
        IF FormWindow.HasBeenChanged[my.window, CvSTC.MessageKey.fontSize.ORD] THEN
            {
            my.f.fontSize ← FormWindow.GetChoiceItemValue[
              window: my.window,
              item: CvSTC.MessageKey.fontSize.ORD];
            };
        IF FormWindow.HasBeenChanged[my.window, CvSTC.MessageKey.replaceUnknown.ORD] THEN
            {
            IF my.textRb[atovReplaceUnknown] # XString.nullReaderBody THEN
                XString.FreeReaderBytes[@my.textRb[atovReplaceUnknown], my.z];
            my.textRb[atovReplaceUnknown] ← FormWindow.GetTextItemValue[
              window: my.window,
              item: CvSTC.MessageKey.replaceUnknown.ORD,
              zone: my.z];
            };
        [ok: ok, ls: my.text[atovReplaceUnknown]] ← CvSTC.ParseItem[
          my: my,
          r: @my.textRb[atovReplaceUnknown],
          item: CvSTC.MessageKey.replaceUnknown,
          buf: @bufWb];
        IF NOT ok THEN RETURN;

        IF FormWindow.HasBeenChanged[my.window, CvSTC.MessageKey.ignoreTrailing.ORD] THEN
            {
            my.f.ignoreTrailing.value ← FormWindow.GetBooleanItemValue[
              window: my.window,
```

```
            item: CvSTC.MessageKey.ignoreTrailing.ORD];
          };

      IF FormWindow.HasBeenChanged[my.window, CvSTC.MessageKey.includeTelCodes.ORD] THEN
          {
          my.f.includeTelCodes.value ← FormWindow.GetBooleanItemValue[
            window: my.window,
            item: CvSTC.MessageKey.includeTelCodes.ORD];
          };
      };
    VtoAdst =>
      {
      IF FormWindow.HasBeenChanged[my.window, CvSTC.MessageKey.codeScheme.ORD] THEN
          {
          my.f.vaCodeScheme ← FormWindow.GetChoiceItemValue[
            window: my.window,
            item: CvSTC.MessageKey.codeScheme.ORD];
          };
      IF FormWindow.HasBeenChanged[my.window, CvSTC.MessageKey.lineLen.ORD] THEN
          {
          my.f.lineLen ← FormWindow.GetChoiceItemValue[
            window: my.window,
            item: CvSTC.MessageKey.lineLen.ORD];
          };
      IF FormWindow.HasBeenChanged[my.window, CvSTC.MessageKey.charsSuffix.ORD] THEN
          {
          my.f.charsSuffix ← CARDINAL[FormWindow.GetIntegerItemValue[window: my.window,
            item: CvSTC.MessageKey.charsSuffix.ORD !
              XString.InvalidNumber => {
                msgRb: XString.ReaderBody ← CvSTC.GetMessage[extraErr1];
                Attention.Post[@msgRb];
                GOTO Badnum;
                };
              XString.Overflow => {
                my.f.charsSuffix ← 0;
                CONTINUE;
                }]];
          IF my.f.charsSuffix NOT IN [10..256] THEN
              {
              msgRb: XString.ReaderBody ← CvSTC.GetMessage[charsOutOfBounds];
              Attention.Post[@msgRb];
              GOTO Badnum;
              };
          EXITS
            Badnum => {
              FormWindow.SetSelection[window: my.window,
                                      item: CvSTC.MessageKey.charsSuffix.ORD,
                                      firstChar: 0, lastChar: CARDINAL.LAST];
              FormWindow.SetInputFocus[window: my.window,
                                       item: CvSTC.MessageKey.charsSuffix.ORD,
                                       beforeChar: CARDINAL.LAST];
              RETURN[ok: FALSE];
            };
          };
      IF FormWindow.HasBeenChanged[my.window, CvSTC.MessageKey.wordWrap.ORD] THEN
          {
          my.f.wordWrap.value ← FormWindow.GetBooleanItemValue[
            window: my.window,
            item: CvSTC.MessageKey.wordWrap.ORD];
          };
      IF FormWindow.HasBeenChanged[my.window, CvSTC.MessageKey.endLine.ORD] THEN
          {
          IF my.textRb[endLine] # XString.nullReaderBody THEN
              XString.FreeReaderBytes[@my.textRb[endLine], my.z];
          my.textRb[endLine] ← FormWindow.GetTextItemValue[
            window: my.window,
            item: CvSTC.MessageKey.endLine.ORD,
            zone: my.z];
          };
      [ok: ok, ls: my.text[endLine]] ← CvSTC.ParseItem[
        my: my,
        r: @my.textRb[endLine],
        item: CvSTC.MessageKey.endLine,
        buf: @bufWb];
      IF NOT ok THEN RETURN;

      IF FormWindow.HasBeenChanged[my.window, CvSTC.MessageKey.endPara.ORD] THEN
          {
          IF my.textRb[endPara] # XString.nullReaderBody THEN
              XString.FreeReaderBytes[@my.textRb[endPara], my.z];
          my.textRb[endPara] ← FormWindow.GetTextItemValue[
            window: my.window,
            item: CvSTC.MessageKey.endPara.ORD,
            zone: my.z];
          };
      [ok: ok, ls: my.text[endPara]] ← CvSTC.ParseItem[
        my: my,
        r: @my.textRb[endPara],
        item: CvSTC.MessageKey.endPara,
        buf: @bufWb];
      IF NOT ok THEN RETURN;

      IF FormWindow.HasBeenChanged[my.window, CvSTC.MessageKey.replaceUnknown.ORD] THEN
          {
          IF my.textRb[vtoaReplaceUnknown] # XString.nullReaderBody THEN
              XString.FreeReaderBytes[@my.textRb[vtoaReplaceUnknown], my.z];
          my.textRb[vtoaReplaceUnknown] ← FormWindow.GetTextItemValue[
```

```
                    window: my.window,
                    item: CvSTC.MessageKey.replaceUnknown.ORD,
                    zone: my.z];
                  };
              [ok: ok, ls: my.text[vtoaReplaceUnknown]] ← CvSTC.ParseItem[
                my: my,
                r: @my.textRb[vtoaReplaceUnknown],
                item: CvSTC.MessageKey.replaceUnknown,
                buf: @bufWb];
              IF NOT ok THEN RETURN;

              };
          ENDCASE;
      XString.FreeWriterBytes[@bufWb];
};


GrowParent: FormWindow.MinDimsChangeProc = {
<< = PROCEDURE [window: Window.Handle, old: Window.Dims, new: Window.Dims];
>>
    my: CvSTC.Common = CvSTC.DataFromWindow[window];
    oldHeight: INTEGER;

    --/* don't adjust the first time window is viewed */
    IF my = NIL THEN RETURN;
    IF old = new THEN RETURN;

    --/* defaulting newHeight returns oldHeight without resizing */
    oldHeight ← Converter.ResizeDetailWindow[
      cvData: my.cvData,
      window: window,
      which: IF my.owner = AtoVsrc THEN source ELSE destination];

    --/* now resize window */
    [] ← Converter.ResizeDetailWindow[
      cvData: my.cvData,
      window: window,
      which: IF my.owner = AtoVsrc THEN source ELSE destination,
      newHeight: oldHeight + (new.h - old.h)];
};


MakeBackstop: FormWindow.MakeItemsProc = {
    tag: XString.ReaderBody ← CvSTC.GetMessage[backstop];

    FormWindow.MakeTextItem[
      window: window,
      myKey: CvSTC.MessageKey.backstop.ORD,
      boxed: FALSE,
      readOnly: TRUE,
      width: 400,
      initString: @tag];
};



MakeAtoVDst: FormWindow.MakeItemsProc = {
<< = PROCEDURE [window: Window.Handle, clientData: LONG POINTER];
>>
    my: CvSTC.Common = clientData;
    tag: XString.ReaderBody;
    tmp: XString.ReaderBody;

    tag ← CvSTC.GetMessage[font];
    tmp ← CvSTC.GetMessage[fontChoices];
    BEGIN
    values: FormWindow.ChoiceItems ← FormWindowMessageParse.ParseChoiceItemMessage[choiceItemMessage: @tmp, zone: my.z];
    FormWindow.MakeChoiceItem[
      window: window,
      myKey: CvSTC.MessageKey.font.ORD,
      tag: @tag,
      values: values,
      initChoice: my.f.font,
      fullyDisplayed: TRUE];
    FormWindowMessageParse.FreeChoiceItems[choiceItems: values, zone: my.z];
    END;

    tag ← CvSTC.GetMessage[fontSize];
    tmp ← CvSTC.GetMessage[fontSizeChoices];
    BEGIN
    values: FormWindow.ChoiceItems ← FormWindowMessageParse.ParseChoiceItemMessage[choiceItemMessage: @tmp, zone: my.z];
    FormWindow.MakeChoiceItem[
      window: window,
      myKey: CvSTC.MessageKey.fontSize.ORD,
      tag: @tag,
      values: values,
      initChoice: my.f.fontSize,
      fullyDisplayed: TRUE];
    FormWindowMessageParse.FreeChoiceItems[choiceItems: values, zone: my.z];
    END;

    tag ← CvSTC.GetMessage[replaceUnknown];
    FormWindow.MakeTextItem[
      window: window,
      myKey: CvSTC.MessageKey.replaceUnknown.ORD,
      tag: @tag,
      width: textWidth,
```

```
        initString: @my.textRb[atovReplaceUnknown]];

      tag ← CvSTC.GetMessage[ignoreTrailing];
      FormWindow.MakeBooleanItem[
        window: window,
        myKey: CvSTC.MessageKey.ignoreTrailing.ORD,
        label: [string[tag]],
        initBoolean: my.f.ignoreTrailing.value];

      tag ← CvSTC.GetMessage[includeTelCodes];
      FormWindow.MakeBooleanItem[
        window: window,
        myKey: CvSTC.MessageKey.includeTelCodes.ORD,
        label: [string[tag]],
        initBoolean: my.f.includeTelCodes.value];
};


MakeAtoVSrc: FormWindow.MakeItemsProc = {
<< = PROCEDURE [window: Window.Handle, clientData: LONG POINTER];
>>
    my: CvSTC.Common = clientData;
    tag: XString.ReaderBody;
    tmp: XString.ReaderBody;

    tag ← CvSTC.GetMessage[paraEndsWith];
    FormWindow.MakeTextItem[
      window: window,
      myKey: CvSTC.MessageKey.paraEndsWith.ORD,
      tag: @tag,
      width: textWidth,
      initString: @my.textRb[paraEndsWith]];

    tag ← CvSTC.GetMessage[codeScheme];
    tmp ← CvSTC.GetMessage[codeSchemeChoices];
    BEGIN
    values: FormWindow.ChoiceItems ← FormWindowMessageParse.ParseChoiceItemMessage[choiceItemMessage: @tmp, zone: my.z];
    FormWindow.MakeChoiceItem[
      window: window,
      myKey: CvSTC.MessageKey.codeScheme.ORD,
      tag: @tag,
      values: values,
      initChoice: my.f.avCodeScheme,
      fullyDisplayed: TRUE];
    FormWindowMessageParse.FreeChoiceItems[choiceItems: values, zone: my.z];
    END;
};


MakeVtoADst: FormWindow.MakeItemsProc = {
<< = PROCEDURE [window: Window.Handle, clientData: LONG POINTER];>>
    my: CvSTC.Common = clientData;
    tag: XString.ReaderBody;
    tmp: XString.ReaderBody;

    tag ← CvSTC.GetMessage[codeScheme];
    tmp ← CvSTC.GetMessage[codeSchemeChoices];
    BEGIN
    values: FormWindow.ChoiceItems ← FormWindowMessageParse.ParseChoiceItemMessage[choiceItemMessage: @tmp, zone: my.z];
    FormWindow.MakeChoiceItem[
      window: window,
      myKey: CvSTC.MessageKey.codeScheme.ORD,
      tag: @tag,
      values: values,
      initChoice: my.f.vaCodeScheme,
      fullyDisplayed: TRUE];
    FormWindowMessageParse.FreeChoiceItems[choiceItems: values, zone: my.z];
    END;

    tag ← CvSTC.GetMessage[lineLen];
    tmp ← CvSTC.GetMessage[lineLenChoices];
    BEGIN
    values: FormWindow.ChoiceItems ← FormWindowMessageParse.ParseChoiceItemMessage[choiceItemMessage: @tmp, zone: my.z];
    FormWindow.MakeChoiceItem[
      window: window,
      myKey: CvSTC.MessageKey.lineLen.ORD,
      tag: @tag,
      values: values,
      initChoice: my.f.lineLen,
      changeProc: LineLenXProc,
      fullyDisplayed: TRUE];
    FormWindowMessageParse.FreeChoiceItems[choiceItems: values, zone: my.z];
    END;

    tag ← CvSTC.GetMessage[charsSuffix];
    FormWindow.MakeIntegerItem[
      window: window,
      myKey: CvSTC.MessageKey.charsSuffix.ORD,
      suffix: @tag,
      visibility: IF my.f.lineLen = CvSTC.limited THEN visible ELSE invisible,
      signed: FALSE,
      width: 30,
      initInteger: INTEGER[my.f.charsSuffix]];

    tag ← CvSTC.GetMessage[wordWrap];
    FormWindow.MakeBooleanItem[
      window: window,
```

```
        myKey: CvSTC.MessageKey.wordWrap.ORD,
        visibility: IF my.f.lineLen = CvSTC.limited THEN visible ELSE invisible,
        label: [string[tag]],
        initBoolean: my.f.wordWrap.value];

      tag ← CvSTC.GetMessage[endLine];
      FormWindow.MakeTextItem[
        window: window,
        myKey: CvSTC.MessageKey.endLine.ORD,
        tag: @tag,
        width: textWidth,
        initString: @my.textRb[endLine]];

      tag ← CvSTC.GetMessage[endPara];
      FormWindow.MakeTextItem[
        window: window,
        myKey: CvSTC.MessageKey.endPara.ORD,
        tag: @tag,
        width: textWidth,
        initString: @my.textRb[endPara]];

      tag ← CvSTC.GetMessage[replaceUnknown];
      FormWindow.MakeTextItem[
        window: window,
        myKey: CvSTC.MessageKey.replaceUnknown.ORD,
        tag: @tag,
        width: textWidth,
        initString: @my.textRb[vtoaReplaceUnknown]];
};


LayoutAtoVDst: FormWindow.LayoutProc = {
<< = PROCEDURE [window: Window.Handle, clientData: LONG POINTER];
>>
    leadingMargin: CARDINAL = CvSTC.leadingMargin;
    spaceAboveLine: CARDINAL = 5;
    line: FormWindow.Line;
    tabChoice: fixed FormWindow.TabStops = [fixed[tabStopInterval]];

    FormWindow.SetTabStops[window, tabChoice];

    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvSTC.MessageKey.font.ORD,
      line: line,
      preMargin: CvSTC.GetPreMargin[font] MOD tabStopInterval,
      tabStop: CvSTC.GetPreMargin[font] / tabStopInterval,
      repaint: FALSE];

    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvSTC.MessageKey.fontSize.ORD,
      line: line,
      preMargin: CvSTC.GetPreMargin[fontSize] MOD tabStopInterval,
      tabStop: CvSTC.GetPreMargin[fontSize] / tabStopInterval,
      repaint: FALSE];

    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvSTC.MessageKey.replaceUnknown.ORD,
      line: line,
      preMargin: CvSTC.GetPreMargin[replaceUnknown] MOD tabStopInterval,
      tabStop: CvSTC.GetPreMargin[replaceUnknown] / tabStopInterval,
      repaint: FALSE];

    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvSTC.MessageKey.ignoreTrailing.ORD,
      line: line,
      preMargin: CvSTC.GetPreMargin[ignoreTrailing] MOD tabStopInterval,
      tabStop: CvSTC.GetPreMargin[ignoreTrailing] / tabStopInterval,
      repaint: FALSE];

    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvSTC.MessageKey.includeTelCodes.ORD,
      line: line,
      preMargin: CvSTC.GetPreMargin[includeTelCodes] MOD tabStopInterval,
      tabStop: CvSTC.GetPreMargin[includeTelCodes] / tabStopInterval,
      repaint: FALSE];
};


LayoutAtoVSrc: FormWindow.LayoutProc = {
<< = PROCEDURE [window: Window.Handle, clientData: LONG POINTER];
>>
    leadingMargin: CARDINAL = CvSTC.leadingMargin;
    spaceAboveLine: CARDINAL = 5;
    line: FormWindow.Line;
    tabChoice: fixed FormWindow.TabStops = [fixed[tabStopInterval]];

    FormWindow.SetTabStops[window, tabChoice];
```

```
    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvSTC.MessageKey.paraEndsWith.ORD,
      line: line,
      preMargin: CvSTC.GetPreMargin[paraEndsWith] MOD tabStopInterval,
      tabStop: CvSTC.GetPreMargin[paraEndsWith] / tabStopInterval,
      repaint: FALSE];

    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvSTC.MessageKey.codeScheme.ORD,
      line: line,
      preMargin: CvSTC.GetPreMargin[codeScheme] MOD tabStopInterval,
      tabStop: CvSTC.GetPreMargin[codeScheme] / tabStopInterval,
      repaint: FALSE];

    line ← FormWindow.AppendLine[window, spaceAboveLine];
};


LayoutVtoADst: FormWindow.LayoutProc = {
<< = PROCEDURE [window: Window.Handle, clientData: LONG POINTER];>>
    leadingMargin: CARDINAL = CvSTC.leadingMargin;
    spaceAboveLine: CARDINAL = 5;
    line: FormWindow.Line;
    tabChoice: fixed FormWindow.TabStops = [fixed[tabStopInterval]];

    FormWindow.SetTabStops[window, tabChoice];

    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvSTC.MessageKey.codeScheme.ORD,
      line: line,
      preMargin: CvSTC.GetPreMargin[codeScheme] MOD tabStopInterval,
      tabStop: CvSTC.GetPreMargin[codeScheme] / tabStopInterval,
      repaint: FALSE];

    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvSTC.MessageKey.lineLen.ORD,
      line: line,
      preMargin: CvSTC.GetPreMargin[lineLen] MOD tabStopInterval,
      tabStop: CvSTC.GetPreMargin[lineLen] / tabStopInterval,
      repaint: FALSE];
    FormWindow.AppendItem[
      window: window,
      item: CvSTC.MessageKey.charsSuffix.ORD,
      line: line,
      preMargin: CvSTC.GetPreMargin[charsSuffix],
      tabStop: ,
      repaint: FALSE];
    FormWindow.AppendItem[
      window: window,
      item: CvSTC.MessageKey.wordWrap.ORD,
      line: line,
      preMargin: CvSTC.GetPreMargin[wordWrap],
      tabStop: ,
      repaint: FALSE];

    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvSTC.MessageKey.endLine.ORD,
      line: line,
      preMargin: CvSTC.GetPreMargin[endLine] MOD tabStopInterval,
      tabStop: CvSTC.GetPreMargin[endLine] / tabStopInterval,
      repaint: FALSE];

    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvSTC.MessageKey.endPara.ORD,
      line: line,
      preMargin: CvSTC.GetPreMargin[endPara] MOD tabStopInterval,
      tabStop: CvSTC.GetPreMargin[endPara] / tabStopInterval,
      repaint: FALSE];

    line ← FormWindow.AppendLine[window, spaceAboveLine];
    FormWindow.AppendItem[
      window: window,
      item: CvSTC.MessageKey.replaceUnknown.ORD,
      line: line,
      preMargin: CvSTC.GetPreMargin[replaceUnknown] MOD tabStopInterval,
      tabStop: CvSTC.GetPreMargin[replaceUnknown] / tabStopInterval,
      repaint: FALSE];
};


--/* Change Procs */

LineLenXProc: FormWindow.ChoiceChangeProc = {
<< = PROCEDURE [window: Window.Handle, item: FormWindow.ItemKey, calledBecauseOf: FormWindow.ChangeReason, oldValue:
```

```
FormWindow.ChoiceIndex, newValue: FormWindow.ChoiceIndex];
>>
    IF newValue = oldValue THEN RETURN;
    IF newValue = CvSTC.limited THEN
        {
        FormWindow.SetVisibility[
          window: window,
          item: CvSTC.MessageKey.charsSuffix.ORD,
          visibility: visible,
          repaint: FALSE];
        FormWindow.SetVisibility[
          window: window,
          item: CvSTC.MessageKey.wordWrap.ORD,
          visibility: visible,
          repaint: TRUE];
        }
    ELSE
        {
        FormWindow.SetVisibility[
          window: window,
          item: CvSTC.MessageKey.charsSuffix.ORD,
          visibility: invisible,
          repaint: FALSE];
        FormWindow.SetVisibility[
          window: window,
          item: CvSTC.MessageKey.wordWrap.ORD,
          visibility: invisible,
          repaint: TRUE];
        };
};

END...

LOG
16-Mar-87 14:06:16 - Caro - Created
24-Nov-87 16:58:56 - Erickson - Changed paraEndsWith default to <CR> instead of <CR><LF>
17-Dec-87 15:48:52 - Erickson - AR 16414 - Added to ApplyChanges in the CvSTC.MessageKey.charsSuffix section.  The value read from the
prop sheet was expected to be a valid number.  If text was entered, the converter crashed the system.  I added signal checking for
InvalidNumber and Overflow.  If text is entered, the InvalidNumber signal is raised by FormWindow.GetIntegerItemValue, and is caught
here.  The user's input is then highlighted, that field of the propsheet is made the input focus, and a message is posted indicating the
problem.  This message was placed in the extraErr1 position in CvSTC
MsgFileImpl.mesa.  While I was here, I added a catch phrase for the Overflow signal also, this simply sets the input value to zero and
allows the already existing code to treat this as input out of range.
```

```
-- File: CvSTCMainImpl.mesa
-- Trow 10-Sep-89 19:53:16

-- Last Revised by: Caro                          30-Jun-87 12:39:53
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (c) 1987, 1989 by Xerox Corporation.  All rights reserved.

DIRECTORY
  Atom
    USING [MakeAtom],
  Attention
    USING [Post],
  BWSZone
    USING [Permanent],
  Context
    USING [Create, Error, Find, NopDestroyProc, Type, UniqueType],
  Converter
    USING [DestinationOptions, GetEventType, Register, Status, SourceOptions],
  ConverterMsg
    USING [Get, kvpDocument],
  ConverterPFOptions
    USING [conASCII],
  CvSTC
    USING [AsciiToVP, AsciiToVPDstOps, AsciiToVPSrcOps, Common,
    GetMessage, leadingMargin, MessageKey,
    pointsBetweenItems, ProblemType, VPToAscii, VPToAsciiDstOps],
  Event
    USING [AddDependency, AgentProcedure, EventType],
  Process
    USING [Detach, Pause, SecondsToTicks],
  ProductFactoring
    USING [Enabled],
  SimpleTextDisplay
    USING [MeasureString],
  StarFileTypes
    USING [document, text, unspecified],
  Window
    USING [Handle],
  XString
    USING [ReaderBody];
<<
--=============================================================================
-- OVERVIEW:

Main code for ascii conversion.  Registations done here.
--=============================================================================
>>

CvSTCMainImpl: PROGRAM
IMPORTS
  Atom, Attention, BWSZone, Context, Converter, ConverterMsg,
  CvSTC, Event, Process, ProductFactoring, SimpleTextDisplay
EXPORTS
  CvSTC =
BEGIN

--====================
-- CONSTANTS
--====================

--====================
-- TYPES
--====================

Globals: TYPE = RECORD [
    leads: ItemLeads,
    ctype: Context.Type,
    z: UNCOUNTED ZONE];

ItemLeads: TYPE = ARRAY CvSTC.MessageKey[paraEndsWith..lastPsheetItem] OF CARDINAL;

--====================
-- GLOBALS
--====================

g: Globals;

--====================
-- PUBLIC SIGNALS
--====================

Problem: PUBLIC SIGNAL [err: CvSTC.ProblemType] = CODE;

--====================
-- PUBLIC PROCEDURES
--====================

DataFromWindow: PUBLIC PROC [w: Window.Handle] RETURNS [my: CvSTC.Common] = {
    my ← Context.Find[type: g.ctype, window: w ! Context.Error => {my ← NIL; CONTINUE}];
};


DataToWindow: PUBLIC PROC [my: CvSTC.Common, w: Window.Handle] = {
    Context.Create[
      type: g.ctype,
      data: my,
```

```
        proc: Context.NopDestroyProc,
        window: w ! Context.Error => CONTINUE];
};


GetPreMargin: PUBLIC PROC [item: CvSTC.MessageKey] RETURNS [leads: CARDINAL] = {
    RETURN[g.leads[item]];
};

--=====================
-- PROCEDURES
--=====================


Init: PROC = {
    z: UNCOUNTED ZONE = BWSZone.Permanent[];

    g ← [
      leads: ALL[CARDINAL.LAST],
      ctype: Context.UniqueType[],
      z: z];

    MeasureTags[];

    --/* register with converter icon */
    Register[];
};


MeasureTags: PROC = {
    lmarg: CARDINAL = CvSTC.leadingMargin;
    max: CARDINAL ← 0;
    --/* local proc */
    Length: PROC [key: CvSTC.MessageKey] RETURNS [width: CARDINAL] =
        {
        rb: XString.ReaderBody ← CvSTC.GetMessage[key];
        [width: width] ← SimpleTextDisplay.MeasureString[string: @rb];
        RETURN [width];
        };

    --/* begin code */
    g.leads ← [
      paraEndsWith: Length[paraEndsWith],
      codeScheme: Length[codeScheme],
      codeSchemeChoices: 0,
      font: Length[font],
      fontChoices: 0,
      fontSize: Length[fontSize],
      fontSizeChoices: 0,
      ignoreTrailing: 1,  -- no tag
      includeTelCodes: 1,  -- no tag
      lineLen: Length[lineLen],
      lineLenChoices: 0,
      charsSuffix: CARDINAL.LAST,
      wordWrap: CARDINAL.LAST,
      endLine: Length[endLine],
      endPara: Length[endPara],
      replaceUnknown: Length[replaceUnknown],
      lastPsheetItem: 0];

    --/* now determine max */
    FOR i: CvSTC.MessageKey IN CvSTC.MessageKey[paraEndsWith..lastPsheetItem] DO
        IF g.leads[i] = CARDINAL.LAST THEN LOOP;
        max ← MAX[max, g.leads[i]];
    ENDLOOP;

    --/* now adjust */
    max ← max + lmarg;
    FOR i: CvSTC.MessageKey IN CvSTC.MessageKey[paraEndsWith..lastPsheetItem] DO
        SELECT g.leads[i] FROM
            0 => LOOP;
            1 => g.leads[i] ← max + 8;  -- compensate for no tag
            CARDINAL.LAST => g.leads[i] ← CvSTC.pointsBetweenItems;
            ENDCASE => g.leads[i] ← max - g.leads[i];
    ENDLOOP;
};


RegisterNow: PROC [first: BOOLEAN] RETURNS [allOk: BOOLEAN ← TRUE] = {
    doc: XString.ReaderBody ← ConverterMsg.Get[ConverterMsg.kvpDocument];
    asciiDoc: XString.ReaderBody ← CvSTC.GetMessage[asciiDoc];
    status: Converter.Status;
    --/* local proc */
    Check: PROC [status: Converter.Status] =
        {
        SELECT status FROM
            registered, alreadyExisted, overridden => NULL;
            busy =>
                {
                IF first THEN
                    {
                    et: Event.EventType ← Converter.GetEventType[];
                    --/* tell user registration will be done later */
                    --+$$$ not implemented

                    [] ← Event.AddDependency[
```

```
                        agent: RetryRegistration,
                        myData: NIL,
                        event: et];
                    first ← FALSE;  --/* only add once! */
                    };
                allOk ← FALSE;
                };
            error => allOk ← FALSE;  --+$$$ should post a message
            ENDCASE;
        };

    --/* begin code */
    status ← Converter.Register[
      srcType: StarFileTypes.text,
      srcFormat: @asciiDoc,
      destFormat: @doc,
      convertProc: CvSTC.AsciiToVP,
      sizeChange: 190,
      forkable: TRUE].status;

    Check[status];

    status ← Converter.Register[
      srcType: StarFileTypes.unspecified,
      srcFormat: @asciiDoc,
      destFormat: @doc,
      convertProc: CvSTC.AsciiToVP,
      sizeChange: 190,
      forkable: TRUE].status;

    Check[status];

    asciiDoc ← CvSTC.GetMessage[asciiDoc];
    status ← Converter.Register[
      srcType: StarFileTypes.document,
      srcFormat: @doc,
      destFormat: @asciiDoc,
      convertProc: CvSTC.VPToAscii,
      sizeChange: 63,
      forkable: TRUE].status;

    Check[status];

    --/* register ops */

    IF NOT allOk THEN RETURN;

    status ← Converter.DestinationOptions[
      srcFormat: @doc,
      destFormat: @asciiDoc,
      dependentOptions: CvSTC.VPToAsciiDstOps,
      override: TRUE].status;

    Check[status];

    asciiDoc ← CvSTC.GetMessage[asciiDoc];
    status ← Converter.SourceOptions[
      srcFormat: @asciiDoc,
      destFormat: @doc,
      dependentOptions: CvSTC.AsciiToVPSrcOps,
      override: TRUE].status;

    Check[status];

    status ← Converter.DestinationOptions[
      srcFormat: @asciiDoc,
      destFormat: @doc,
      dependentOptions: CvSTC.AsciiToVPDstOps,
      override: TRUE].status;

    Check[status];

};


RetryRegistration: Event.AgentProcedure = {
    IF RegisterNow[first: FALSE].allOk THEN remove ← TRUE;
};


RetryProductFactoring: Event.AgentProcedure = {
    IF NOT ProductFactoring.Enabled[option: ConverterPFOptions.conASCII] THEN
        {
        msg: XString.ReaderBody ← CvSTC.GetMessage[notPF];

        Attention.Post[@msg];
        remove ← FALSE;
        }
    ELSE
        {
        Process.Detach[FORK AvoidDeadlock[]];
        remove ← TRUE;
        };
};

<<
```

```
--============================================================================
AvoidDeadlock
* Finish doing registrations in another process, to make sure we don't try to AddDependency from inside of an AgentProcedure.
--============================================================================
>>
AvoidDeadlock: PROC = {
    Process.Pause[Process.SecondsToTicks[2]];  --/* give other process a chance */
    [] ← RegisterNow[first: TRUE];
};


Register: PROCEDURE = {
    IF NOT ProductFactoring.Enabled[option: ConverterPFOptions.conASCII] THEN
        {
        msg: XString.ReaderBody ← CvSTC.GetMessage[notPF];
        logon: Event.EventType ← Atom.MakeAtom["LogonCompleted"L];

        Attention.Post[@msg];
        [] ← Event.AddDependency[
          agent: RetryProductFactoring,
          myData: NIL,
          event: logon];
        }
    ELSE [] ← RegisterNow[first: TRUE]; -- OK
};

--/* MAIN code */

Init[];


END...

LOG
16-Mar-87 14:06:16 - Caro - Created
30-Jun-87 12:39:59 - Caro - MDS relief, RetryProductFactoring
```

```
-- File: CvSTCMsgFileImpl.mesa
-- Trow  8-Mar-89 14:18:00

-- Last Revised by: Erickson                    17-Dec-87 16:06:35
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (c) 1985, 1986, 1987, 1988, 1989 by Xerox Corporation.  All rights reserved.

DIRECTORY
  ApplicationFolderExtra
    USING [InitMessages],
  CvSTC,
  NSFile
    USING [Error],
  Runtime
    USING [UnboundProcedure],
  XMessage
    USING [AllocateMessages, Get, Handle, MsgEntry, RegisterMessages],
  XString
    USING [FromSTRING, nullReaderBody, ReaderBody];

CvSTCMsgFileImpl: PROGRAM
IMPORTS
  ApplicationFolderExtra, NSFile, Runtime, XMessage, XString
EXPORTS CvSTC =
BEGIN

--====================
-- GLOBALS
--====================

h: XMessage.Handle ← NIL;

--====================
-- SIGNALS
--====================

NoMessageFile: ERROR = CODE;

--====================
-- PUBLIC PROCEDURES
--====================

GetMessage: PUBLIC PROCEDURE [msg: CvSTC.MessageKey] RETURNS [msgRb: XString.ReaderBody] = {
    IF h # NIL THEN RETURN[h.Get[msg.ORD]];
    RETURN[XString.nullReaderBody];
};

--====================
-- PROCEDURES
--====================

InitMessages: PROCEDURE = {
    internalName: XString.ReaderBody ← XString.FromSTRING["FC STC Documents"L];
    messageFile: XString.ReaderBody ← XString.FromSTRING["MessageFile"L];

    h ← ApplicationFolderExtra.InitMessages[
      internalName: @internalName,
      label: @messageFile,
      domainIndex: 0 ! ANY => {h ← NIL; CONTINUE}];
    IF h = NIL THEN
        InitFromArray[];
};


InitFromArray: PROC = {
    h ← XMessage.AllocateMessages["STC Conversion"L, CvSTC.MessageKey.LAST.ORD.SUCC, NIL, NIL];

    Init0to17[];
    Init18toLAST[];
};


Init0to17: PROC = {
    msgArray: ARRAY CvSTC.MessageKey[asciiDoc..lastPsheetItem] OF XMessage.MsgEntry ← [
      asciiDoc: [
        msgKey: CvSTC.MessageKey.asciiDoc.ORD,
        msg: XString.FromSTRING["Chinese Telegraph Codes"L],
        type: userMsg,
        translationNote: "Label for source or destination of conversion"L,
        translatable: FALSE,
        id: 0],
      paraEndsWith: [
        msgKey: CvSTC.MessageKey.paraEndsWith.ORD,
        msg: XString.FromSTRING["Paragraph ends with"L],
        type: pSheetItem,
        translationNote: "Tag for text item, should read as if user were filling in the blank/completing the sentence"L,
        translatable: TRUE,
        id: 1],
      codeScheme: [
        msgKey: CvSTC.MessageKey.codeScheme.ORD,
        msg: XString.FromSTRING["Telegraph code scheme"L],
        type: pSheetItem,
        translationNote: "Choice item tag"L,
        translatable: TRUE,
        id: 2],
      codeSchemeChoices: [
```

```
    msgKey: CvSTC.MessageKey.codeSchemeChoices.ORD,
    msg: XString.FromSTRING["PRC/stc:0@ROC/ctc:1"L],
    type: argList,
    translationNote: "Choices that go with id#2"L,
    translatable: FALSE,
    id: 3],
  font: [
    msgKey: CvSTC.MessageKey.font.ORD,
    msg: XString.FromSTRING["Font"L],
    type: pSheetItem,
    translationNote: "Choice item tag"L,
    translatable: TRUE,
    id: 4],
  fontChoices: [
    msgKey: CvSTC.MessageKey.fontChoices.ORD,
    msg: XString.FromSTRING["Modern:0@Classic:1"L],
    type: argList,
    translationNote: "Choices that go with id#4"L,
    translatable: TRUE,
    id: 5],
  fontSize: [
    msgKey: CvSTC.MessageKey.fontSize.ORD,
    msg: XString.FromSTRING["Font size"L],
    type: pSheetItem,
    translationNote: "Choice item tag"L,
    translatable: TRUE,
    id: 6],
  fontSizeChoices: [
    msgKey: CvSTC.MessageKey.fontSizeChoices.ORD,
    msg: XString.FromSTRING["12:0@24:1"L],
    type: argList,
    translationNote: "Choices that go with id#6"L,
    translatable: TRUE,
    id: 7],
  ignoreTrailing: [
    msgKey: CvSTC.MessageKey.ignoreTrailing.ORD,
    msg: XString.FromSTRING["IGNORE TRAILING WHITE SPACE"L],
    type: pSheetItem,
    translationNote: "Boolean item"L,
    translatable: TRUE,
    id: 8],
  includeTelCodes: [
    msgKey: CvSTC.MessageKey.includeTelCodes.ORD,
    msg: XString.FromSTRING["INCLUDE TELEGRAPH CODES"L],
    type: pSheetItem,
    translationNote: "Boolean item"L,
    translatable: TRUE,
    id: 9],
  lineLen: [
    msgKey: CvSTC.MessageKey.lineLen.ORD,
    msg: XString.FromSTRING["Line length"L],
    type: pSheetItem,
    translationNote: "Choice item tag"L,
    translatable: TRUE,
    id: 10],
  lineLenChoices: [
    msgKey: CvSTC.MessageKey.lineLenChoices.ORD,
    msg: XString.FromSTRING["Unlimited:0@Limited:1"L],
    type: argList,
    translationNote: "Choices that go with id#10"L,
    translatable: TRUE,
    id: 11],
  charsSuffix: [
    msgKey: CvSTC.MessageKey.charsSuffix.ORD,
    msg: XString.FromSTRING["characters"L],
    type: pSheetItem,
    translationNote: "Suffix for number item -- to be read e.g. '[80] characters'"L,
    translatable: TRUE,
    id: 12],
  wordWrap: [
    msgKey: CvSTC.MessageKey.wordWrap.ORD,
    msg: XString.FromSTRING["WORD WRAP"L],
    type: pSheetItem,
    translationNote: "Boolean item, indicating that text lines should break only on the white space between words"L,
    translatable: TRUE,
    id: 13],
  endLine: [
    msgKey: CvSTC.MessageKey.endLine.ORD,
    msg: XString.FromSTRING["End line with"L],
    type: pSheetItem,
    translationNote: "Text item tag, should read as if user is filling in the blank/completing sentence"L,
    translatable: TRUE,
    id: 14],
  endPara: [
    msgKey: CvSTC.MessageKey.endPara.ORD,
    msg: XString.FromSTRING["End paragraph with"L],
    type: pSheetItem,
    translationNote: "Text item tag, should read as if user is filling in the blank/completing sentence"L,
    translatable: TRUE,
    id: 15],
  replaceUnknown: [
    msgKey: CvSTC.MessageKey.replaceUnknown.ORD,
    msg: XString.FromSTRING["Replace unknown character with"L],
    type: pSheetItem,
    translationNote: "Text item tag, should read as if user is filling in the blank/completing sentence"L,
    translatable: TRUE,
    id: 16],
```

```
      lastPsheetItem: [
        msgKey: CvSTC.MessageKey.lastPsheetItem.ORD,
        msg: XString.FromSTRING[" "L],
        type: others,
        translationNote: "DO NOT TRANSLATE -- spare key"L,
        translatable: TRUE,
        id: 17]
      ];

    XMessage.RegisterMessages[h, LOOPHOLE[LONG[DESCRIPTOR[msgArray]]], FALSE];
};


Init18toLAST: PROC = {
    msgArray: ARRAY CvSTC.MessageKey[left..CvSTC.MessageKey.LAST] OF XMessage.MsgEntry ← [
      left: [
        msgKey: CvSTC.MessageKey.left.ORD,
        msg: XString.FromSTRING["<"L],
        type: others,
        translationNote: "do not translate"L,
        translatable: FALSE,
        id: 18],
      right: [
        msgKey: CvSTC.MessageKey.right.ORD,
        msg: XString.FromSTRING[">"L],
        type: others,
        translationNote: "do not translate"L,
        translatable: FALSE,
        id: 19],
      cr: [
        msgKey: CvSTC.MessageKey.cr.ORD,
        msg: XString.FromSTRING["CR"L],
        type: others,
        translationNote: "do not translate"L,
        translatable: FALSE,
        id: 20],
      lf: [
        msgKey: CvSTC.MessageKey.lf.ORD,
        msg: XString.FromSTRING["LF"L],
        type: others,
        translationNote: "do not translate"L,
        translatable: FALSE,
        id: 21],
      nl: [
        msgKey: CvSTC.MessageKey.nl.ORD,
        msg: XString.FromSTRING["NL"L],
        type: others,
        translationNote: "do not translate"L,
        translatable: FALSE,
        id: 22],
      ff: [
        msgKey: CvSTC.MessageKey.ff.ORD,
        msg: XString.FromSTRING["FF"L],
        type: others,
        translationNote: "do not translate"L,
        translatable: FALSE,
        id: 23],
      tab: [
        msgKey: CvSTC.MessageKey.tab.ORD,
        msg: XString.FromSTRING["TAB"L],
        type: others,
        translationNote: "do not translate"L,
        translatable: FALSE,
        id: 24],
      createError: [
        msgKey: CvSTC.MessageKey.createError.ORD,
        msg: XString.FromSTRING["The source object was not converted due to an error while creating the output file."L],
        type: errorMsg,
        translationNote: "Posted to attention window"L,
        translatable: TRUE,
        id: 25],
      notPF: [
        msgKey: CvSTC.MessageKey.notPF.ORD,
        msg: XString.FromSTRING["Chinese Telegraph Code Conversion cannot be activated because required Software Option not enabled.
        Please enable Software Option, End Session, then Logon again."L],
        type: errorMsg,
        translationNote: "posted to attention window"L,
        translatable: TRUE,
        id: 26],
      paginating: [
        msgKey: CvSTC.MessageKey.paginating.ORD,
        msg: XString.FromSTRING[" paginating ... "L],
        type: userMsg,
        translationNote: "posted to attention window following 'Converting xyz ... ' converter icon message.  The leading and trailing
        spaces are REQUIRED"L,
        translatable: TRUE,
        id: 27],
      skippedTableData: [
        msgKey: CvSTC.MessageKey.skippedTableData.ORD,
        msg: XString.FromSTRING[" Some data in '<>' was skipped ... "L],
        type: template,
        translationNote: "Some table data skipped.  Leading and trailing blanks REQUIRED."L,
        translatable: TRUE,
        id: 28],
      dfltAVEndParagraph: [
        msgKey: CvSTC.MessageKey.dfltAVEndParagraph.ORD,
        msg: XString.FromSTRING["<CR><CR>"L],
```

```
      type: others,
      translationNote: "do not translate, default value for text items"L,
      translatable: FALSE,
      id: 29],
    dfltAVReplaceCharacter: [
      msgKey: CvSTC.MessageKey.dfltAVReplaceCharacter.ORD,
      msg: XString.FromSTRING["?"L],
      type: others,
      translationNote: "do not translate, default value for text items"L,
      translatable: FALSE,
      id: 30],
    prefix: [
      msgKey: CvSTC.MessageKey.prefix.ORD,
      msg: XString.FromSTRING["CvSTC"L],
      type: others,
      translationNote: "do not translate, internal file name prefix"L,
      translatable: FALSE,
      id: 31],
    doneFailed: [
      msgKey: CvSTC.MessageKey.doneFailed.ORD,
      msg: XString.FromSTRING["Unrecoverable error writing Chinese Telegraph Code conversion properties.  Cancel the property sheet and
      use a new converter icon."L],
      type: errorMsg,
      translationNote: "Posted when user selects Done on property sheet, if there is an NSFile or other error"L,
      translatable: TRUE,
      id: 32],
    backstop: [
      msgKey: CvSTC.MessageKey.backstop.ORD,
      msg: XString.FromSTRING["Problem: the details section could not be created."L],
      type: pSheetItem,
      translationNote: "For some reason, creation of the client details window failed.  This string is put in the formwindow
      instead."L,
      translatable: TRUE,
      id: 33],
    metaError: [
      msgKey: CvSTC.MessageKey.metaError.ORD,
      msg: XString.FromSTRING["The selected text item contains an error.  Please correct it."L],
      type: errorMsg,
      translationNote: "This message is posted to the Attention window when the user tries to Done or Start a sheet with a text/syntax
      error.  Text syntax is described in the Reference Library documentation for ASCII."L,
      translatable: TRUE,
      id: 34],
    charsOutOfBounds: [
      msgKey: CvSTC.MessageKey.charsOutOfBounds.ORD,
      msg: XString.FromSTRING["The line length limit must be between 10 and 256 characters, inclusive. Please reenter."L],
      type: errorMsg,
      translationNote: "Posted when user tries to Done or Start a sheet with an invalid numeric value."L,
      translatable: TRUE,
      id: 35],
    fatalError: [
      msgKey: CvSTC.MessageKey.fatalError.ORD,
      msg: XString.FromSTRING[" conversion failed with an unrecoverable error "L],
      type: errorMsg,
      translationNote: "Posted if NSFile or other error in conversion.  Note that leading and trailing blanks are required."L,
      translatable: TRUE,
      id: 36],
    extraErr0: [
      msgKey: CvSTC.MessageKey.extraErr0.ORD,
      msg: XString.FromSTRING[" Unrecoverable Chinese Telegraph Code conversion error: damaged converter icon. "L],
      type: errorMsg,
      translationNote: "Blanks are required.  Posted if the conversion cannot read properties from the converter icon."L,
      translatable: TRUE,
      id: 37],
    extraErr1: [
      msgKey: CvSTC.MessageKey.extraErr1.ORD,
      msg: XString.FromSTRING["The number in the highlighted field is invalid. Please reenter."L],
      type: errorMsg,
      translationNote: "Posted when the user tries to Done or Start a sheet with text in a numeric field."L,
      translatable: TRUE,
      id: 38],
    dfltVAEndLine: [
      msgKey: CvSTC.MessageKey.dfltVAEndLine.ORD,
      msg: XString.FromSTRING["<CR>"L],
      type: others,
      translationNote: "do not translate, default value for text items"L,
      translatable: FALSE,
      id: 39],
    dfltVAEndParagraph: [
      msgKey: CvSTC.MessageKey.dfltVAEndParagraph.ORD,
      msg: XString.FromSTRING["<CR>"L],
      type: others,
      translationNote: "do not translate, default value for text items"L,
      translatable: FALSE,
      id: 40],
    dfltVAReplaceCharacter: [
      msgKey: CvSTC.MessageKey.dfltVAReplaceCharacter.ORD,
      msg: XString.FromSTRING["?"L],
      type: others,
      translationNote: "do not translate, default value for text items"L,
      translatable: FALSE,
      id: 41]

<<
    1: [
      msgKey: CvSTC.MessageKey.USEAGAINTOREPLACETHISSTRING.ORD,
      msg: XString.FromSTRING["«»"L],
      type: «»,
```

```
        translationNote: "«»"L,
        translatable: TRUE,
        id: «»],
>>
      ];

    XMessage.RegisterMessages[h, LOOPHOLE[LONG[DESCRIPTOR[msgArray]]], FALSE];
};



--/* MAIN line code */

InitMessages[! NSFile.Error, Runtime.UnboundProcedure => NoMessageFile];

END...


LOG
24-Apr-85 12:12:27 - MSchneider - CREATED from SampleBWSApplicationMsgFileImpl
10-May-85 10:56:18 - MSchneider - used correct ApplicationFolder name
28-May-85  9:28:54 - MSchneider - moved localZone into procedure, added use of BWSZone
24-Jun-85 14:33:55 - MSchneider - made "MessageFile" be "MessageFile" in entry name
 9-Jul-85 11:12:31 - MSchneider - added ERROR NoMessageFile
26-Feb-87 14:59:12 - Caro - Upgraded to VP 2.0 (delete 90% of code)
 8-Apr-87 11:43:56 - Caro - Catch ANY error raised from InitMessages
26-Jun-87 11:10:51 - Caro - Made #44 a real error
19-Aug-87 10:51:37 - Caro - Reworded several messages and transNotes
24-Nov-87 17:01:04 - Erickson - added aToVDfltMeta (ID = 46) to change default for ascii to ViewPoint treatment of paraEndsWith.
17-Dec-87 16:04:02 - Erickson - AR 16414 - made #45 a real error, bad number input.
```

```
-- File: CvSTCParseImpl.mesa
-- 10-Feb-89 22:18:01

-- Last Revised by: Caro                        29-Jun-87 11:31:40
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (c) 1987, 1989 by Xerox Corporation.  All rights reserved.

DIRECTORY
  Ascii
    USING [CR, FF, LF, TAB],
  Attention
    USING [Post],
  CvSTC
    USING [Common, GetMessage, MessageKey],
  FormWindow
    USING [SetSelection, SetInputFocus],
  String
    USING [AppendChar, CopyToNewString, MakeString, StringBoundsFault],
  XChar
    USING [Character, Code, not],
  XString
    USING [AppendChar, ClearWriter, CopyToNewReaderBody,
    Empty, Equal, First, FromSTRING, FreeReaderBytes, FreeWriterBytes,
    InvalidEncoding, Lop, NewWriterBody,
    Reader, ReaderBody, ReaderFromWriter, ValidateReader, Writer, WriterBody];

<<
--============================================================================
-- OVERVIEW:

Parse text items containing meta characters into strings.
--============================================================================
>>

CvSTCParseImpl: PROGRAM
IMPORTS
  Attention, CvSTC, FormWindow, String, XChar, XString
EXPORTS
  CvSTC =
BEGIN

--====================
-- CONSTANTS
--====================

max: CARDINAL = 10;
maxAbbr: CARDINAL = 3;  --/* abbreviations only up to 3 characters */
maxOctals: CARDINAL = 3;  --/* need exactly 3 octal digits */

--====================
-- TYPES
--====================

ParseStates: TYPE = {
    entry,
    beginMeta,
    doOctal,
    doAbbrev
    };
--====================
-- SIGNALS
--====================

ParseError: SIGNAL [err: ErrType ← syntaxError, start, pos: CARDINAL] = CODE;

ErrType: TYPE =
    {
    syntaxError,
    invalidMeta,
    unknownAbbr,
    invalidOctal,
    invalidEncoding
    };

--====================
-- PUBLIC PROCEDURES
--====================

ParseItem: PUBLIC PROC [my: CvSTC.Common, r: XString.Reader, item: CvSTC.MessageKey, buf: XString.Writer ← NIL] RETURNS [ok: BOOLEAN, ls:
LONG STRING] = {
    bufRb: XString.WriterBody;
    tmpRb: XString.ReaderBody;
    msgRb: XString.ReaderBody;
    clientBuf: BOOLEAN;

    IF buf = NIL THEN
        {
        bufRb ← XString.NewWriterBody[maxLength: 30, z: my.z];
        buf ← @bufRb;
        clientBuf ← FALSE;
        }
    ELSE
        clientBuf ← TRUE;

    BEGIN
      ENABLE ParseError =>
```

```
        {
        msgRb ← CvSTC.GetMessage[metaError];
        IF my.window = NIL OR item = CvSTC.MessageKey.FIRST THEN GOTO notOK;

        FormWindow.SetSelection[
          window: my.window,
          item: item.ORD,
          firstChar: start,
          lastChar: pos];
        FormWindow.SetInputFocus[
          window: my.window,
          item: item.ORD,
          beforeChar: pos];
        Attention.Post[@msgRb];
        ls ← NIL;
        GOTO notOK;
        };

    tmpRb ← XString.CopyToNewReaderBody[r: r, z: my.z];
    ls ← ParseToLS[text: @tmpRb, z: my.z, buf: buf];

    --/* test for invalid encoding */
    IF my.owner = AtoVdst THEN
        {
        msgRb ← XString.FromSTRING[ls];
        XString.ValidateReader[@msgRb ! XString.InvalidEncoding =>
           SIGNAL ParseError[
             err: invalidEncoding,
             start: 0,
             pos: CARDINAL.LAST]];
         };

    ok ← TRUE;
    EXITS notOK => ok ← FALSE;
    END;

    IF NOT clientBuf THEN
        XString.FreeWriterBytes[buf];
    XString.FreeReaderBytes[r: @tmpRb, z: my.z];
};

--====================
-- PROCEDURES
--====================


ParseToLS: PROC [text: XString.Reader, z: UNCOUNTED ZONE, buf: XString.Writer] RETURNS [ls: LONG STRING ← NIL] = {
    rb: XString.ReaderBody ← CvSTC.GetMessage[left];
    state: ParseStates ← entry;
    start,
    pos: CARDINAL ← 0;
    octals,
    abbrs: CARDINAL ← 0;
    cr: XString.ReaderBody;
    lf: XString.ReaderBody;
    nl: XString.ReaderBody;
    ff: XString.ReaderBody;
    tab: XString.ReaderBody;
    left: XChar.Character;
    right: XChar.Character;
    xc: XChar.Character;
    c: CHARACTER;
    octalValue: CARDINAL[0..255];

    --/* get < and > */
    left ← XString.First[@rb];
    rb ← CvSTC.GetMessage[right];
    right ← XString.First[@rb];

    --/* initialize strings */
    IF XString.Empty[text] THEN
        RETURN[ls: NIL]
    ELSE
        ls ← String.MakeString[z: z, maxlength: max];
    cr ← CvSTC.GetMessage[cr];
    lf ← CvSTC.GetMessage[lf];
    nl ← CvSTC.GetMessage[nl];
    ff ← CvSTC.GetMessage[ff];
    tab ← CvSTC.GetMessage[tab];

    --/* lop through string */
    DO
      ENABLE
        {
        String.StringBoundsFault =>
          {
          ns ← String.CopyToNewString[s: ls, z: z, longer: max];
          z.FREE[@ls];
          ls ← ns;
          RESUME[ns];
          };
        UNWIND =>
          {
          IF ls # NIL THEN z.FREE[@ls];
          };
        };
```

```
        xc ← XString.Lop[text];
        IF xc = XChar.not THEN
            {
            IF state = entry THEN
                EXIT
            ELSE
                SIGNAL ParseError[err: syntaxError, start: start, pos: pos];
            };

        SELECT state FROM
            entry =>
                {
                IF xc = left THEN
                    state ← beginMeta
                ELSE
                    {
                    c ← LOOPHOLE[XChar.Code[xc], CHARACTER];  --/* only Charset 0 */
                    String.AppendChar[s: ls, c: c];
                    state ← entry;
                    };
                pos ← pos + 1;
                };
            beginMeta =>
                {
                start ← pos;
                c ← LOOPHOLE[XChar.Code[xc], CHARACTER];  --/* only Charset 0 */
                SELECT c FROM
                    IN ['0..'3] =>
                        {
                        state ← doOctal;
                        octals ← 1;
                        octalValue ← c - '0;
                        };
                    'C, 'F, 'L, 'N, 'T, '< =>
                        {
                        state ← doAbbrev;
                        XString.ClearWriter[buf];  --/* collect abbreviation here */
                        XString.AppendChar[to: buf, c: xc];
                        abbrs ← 1;
                        };
                    ENDCASE =>
                      SIGNAL ParseError[err: invalidMeta, start: start, pos: pos];
                pos ← pos + 1;
                };
            doOctal =>
                {
                c ← LOOPHOLE[XChar.Code[xc], CHARACTER];  --/* only Charset 0 */
                IF xc = right THEN
                    {
                    IF start = pos THEN
                        SIGNAL ParseError[err: invalidMeta, start: start, pos: pos + 1];
                    IF octals < maxOctals OR octalValue > 377B THEN
                        SIGNAL ParseError[err: invalidOctal, start: start, pos: pos];
                    c ← LOOPHOLE[octalValue, CHARACTER];
                    String.AppendChar[s: ls, c: c];
                    state ← entry;
                    }
                ELSE IF octals >= maxOctals THEN
                    SIGNAL ParseError[err: invalidOctal, start: start, pos: pos]
                ELSE IF NOT c IN ['0..'7] THEN
                    SIGNAL ParseError[err: invalidOctal, start: start, pos: pos]
                ELSE
                    {
                    octalValue ← (octalValue * 8) + (c - '0);
                    octals ← octals + 1;
                    state ← doOctal;
                    };
                pos ← pos + 1;
                };
            doAbbrev =>
                {
                IF xc = right THEN
                    {
                    tmp: XString.Reader ← XString.ReaderFromWriter[buf];

                    IF start = pos THEN
                        SIGNAL ParseError[err: invalidMeta, start: start, pos: pos + 1];
                    IF abbrs > maxAbbr THEN
                        SIGNAL ParseError[err: unknownAbbr, start: start, pos: pos];
                    SELECT TRUE FROM
                        XString.Equal[r1: tmp, r2: @cr] =>
                            String.AppendChar[s: ls, c: Ascii.CR];
                        XString.Equal[r1: tmp, r2: @lf] =>
                            String.AppendChar[s: ls, c: Ascii.LF];
                        XString.Equal[r1: tmp, r2: @nl] =>
                            {
                            String.AppendChar[s: ls, c: Ascii.CR];
                            String.AppendChar[s: ls, c: Ascii.LF];
                            };
                        XString.Equal[r1: tmp, r2: @tab] =>
                            String.AppendChar[s: ls, c: Ascii.TAB];
                        XString.Equal[r1: tmp, r2: @ff] =>
                            String.AppendChar[s: ls, c: Ascii.FF];
                        abbrs = 1 AND c = '< =>
                            String.AppendChar[s: ls, c: '<];
                        ENDCASE =>
                            SIGNAL ParseError[err: unknownAbbr, start: start, pos: pos];
```

```
                    state ← entry;
                    }
                ELSE
                    {
                    XString.AppendChar[to: buf, c: xc];
                    abbrs ← abbrs + 1;
                    state ← doAbbrev;
                    };
                pos ← pos + 1;
                };
            ENDCASE;
    ENDLOOP;
};


END...

LOG
16-Mar-87 14:06:16 - Caro - Created
26-Jun-87 11:28:54 - Caro - Added test for MessageKey.FIRST to ParseItem
29-Jun-87 11:33:00 - Caro - Added validation to ParseItem
```

```
-- File: CvSTCToVPImpl.mesa
-- Conversion of Chinese Telegraph Code (either STC or CTC) to Hanzi
-- Trow 11-Sep-89 12:30:26

-- Last Revised by: Shinsato                    12-Feb-88 13:00:11
-- Owner: Workstation Applications - Foreign Conversion Team



DIRECTORY
  Ascii
    USING [CR, FF, LF, NUL, SP, TAB],
  BackgroundProcess
    USING [ResetUserAbort, UserAbort],
  BWSZone
    USING [shortLifetime, Permanent],
  Catalog
    USING [beforeLogonSession, GetFile],
  Converter
    USING [ConvertProc, CvData, DependentOptionProc, GetPOption, PostMessage],
  ConverterMsg,
  CvSTC,
  DocInterchangeDefs
    USING [AppendNewParagraph, AppendPageBreak, AppendText, CheckAbortProc,
    Doc, Error, FinishCreation, FinishCreationStatus,
    PaginateOption, StartCreation, StartCreationStatus],
  DocInterchangePropsDefs
    USING [Family, FontPropsRecord, GetFontPropsDefaults, GetPagePropsDefaults,
    GetParaPropsDefaults, PagePropsRecord, ParaPropsRecord, modern, classic],
  Environment
    USING [Block, Byte, bytesPerPage, wordsPerPage],
  NSFile
    USING [Close, Error, Find, GetReference, Handle, Logoff,
    nullHandle, nullReference, OpenByReference, Reference, Session],
  NSFileStream
    USING [Create, Handle],
  NSSegment
    USING [Map],
  NSString
    USING [FreeString, String],
  Space
    USING [ScratchMap, Unmap],
  Stream
    USING [CompletionCode, Delete, GetBlock],
  String
    USING [AppendChar, AppendString, FreeString, MakeString, StringToDecimal],
  TIP
    USING [ResetUserAbort, UserAbort],
  XChar
    USING [Make, not],
  XCharSet0
    USING [Make],
  XCharSet41Extra
    USING [Make],
<<
  XCharSet357
    USING [Make],
>>
  XMessage
    USING [Get, Handle],
  XString
    USING [AppendChar, AppendSTRING, ByteLength,
    Character, CharacterLength, ClearWriter, FreeWriterBytes,
    FromSTRING, InvalidEncoding, NewWriterBody, NSStringFromReader, Reader, ReaderBody, ReaderFromWriter,
    Writer, WriterBody, WriterInfo];

<<
--============================================================================
-- OVERVIEW:

Chinese telegrahph code (in ASCII) to VP conversion.
--============================================================================
>>

CvSTCToVPImpl: PROGRAM
IMPORTS
  Catalog,
  NSSegment, XMessage,
  BackgroundProcess, BWSZone, Converter, ConverterMsg, CvSTC,
  DocInterchangeDefs, DocInterchangePropsDefs,
  NSFile, NSFileStream, NSString, Space, Stream, String,
  TIP, XChar, XCharSet0, XCharSet41Extra, XString
EXPORTS
  CvSTC =
BEGIN

--====================
-- CONSTANTS
--====================

maxPara: CARDINAL = 8 * 1024;
bufPages: CARDINAL = (maxPara + Environment.bytesPerPage - 1) / Environment.bytesPerPage;
paraLen: CARDINAL = maxPara/4;

words: CARDINAL = SIZE[CtoVPCharMap];

stopsAt: CARDINAL = 5;  --/* tab stops every five characters */
```

```
tabStopCount: CARDINAL = (132/stopsAt)+1;  --/* 132 columns max */

aHyphen: CHARACTER = 055C;

xNewLine: XString.Character = XCharSet0.Make[newLine];
xSpace: XString.Character = XCharSet0.Make[space];
xLeftDoubleGuillemet: XString.Character = XCharSet0.Make[leftDoubleGuillemet];
xRightDoubleGuillemet: XString.Character = XCharSet0.Make[rightDoubleGuillemet];

--====================
-- TYPES
--====================

AVData: TYPE = LONG POINTER TO AVDataObj;
AVDataObj: TYPE = RECORD [
    source: NSFile.Handle,
    input: NSFileStream.Handle,  --/* created from source */
    cvData: Converter.CvData,
    session: NSFile.Session,
    src: CvSTC.Common,  --/* common data distinguished by owning formwindow */
    dst: CvSTC.Common,
    background: BOOLEAN,
    fontProps: DocInterchangePropsDefs.FontPropsRecord,
    paraProps: DocInterchangePropsDefs.ParaPropsRecord,
    pageProps: DocInterchangePropsDefs.PagePropsRecord,
    doc: DocInterchangeDefs.Doc,
    blk: Environment.Block,   --/* primary input buffer */
    state: AVState,
    z: UNCOUNTED ZONE];

--/* the various states of the StateMachine */
AVState: TYPE =
    {
    entry,
    append,
    ignoreTrailing,
    maxExceeded,
    endPara
    };

CtoVPCharMap: TYPE = ARRAY CHARACTER OF XString.Character;

TelegraphCode: TYPE = CARDINAL [0..9999];

TelegraphCodeTable: TYPE = LONG POINTER TO ARRAY TelegraphCode OF XString.Character;

--====================
-- GLOBALS
--====================

Global: TYPE = RECORD [
    isomap: LONG POINTER TO CtoVPCharMap,
    pz: UNCOUNTED ZONE];

g: Global;

prc: TelegraphCodeTable ← NIL;
roc: TelegraphCodeTable ← NIL;

--====================
-- PUBLIC PROCEDURES
--====================

AsciiToVP: PUBLIC Converter.ConvertProc = {
<< = PROCEDURE [source: NSFile.Handle, cvData: Converter.CvData, session: NSFile.Session, srcInstance: LONG POINTER ← NIL, dstInstance:
LONG POINTER ← NIL, background: BOOLEAN ← FALSE] RETURNS [dest: NSFile.Handle ← LOOPHOLE[0]];
>>
  ENABLE CvSTC.Problem, NSFile.Error, XString.InvalidEncoding =>
      {
      msgRb: XString.ReaderBody ← CvSTC.GetMessage[fatalError];

      Post[msgRb, cvData];
      CONTINUE;
      };

    IF source = NSFile.nullHandle THEN RETURN;
    dest ← AtoV[source, cvData, session, srcInstance, dstInstance, background];
};

<<
--============================================================================
Both DependentOptionProcs create instance data with CreateCommon.  The data is distinguished by the owner variable.  The CommonObj within
CvSTC.CommonData is the data structure written to the client file stored as the icon properties.  Only those fields pertaining to the
owner are used.
--============================================================================
>>

AsciiToVPSrcOps: PUBLIC Converter.DependentOptionProc = {
<< = PROCEDURE [options: BOOLEAN ← TRUE, cvData: Converter.CvData, which: Converter.FormatToUse, srcFormat: XString.Reader, destFormat:
XString.Reader, window: Window.Handle, oldInstance: LONG POINTER ← NIL] RETURNS [menuItemProc: Converter.MenuItemProc, destroy:
Converter.DestroyProc, instance: LONG POINTER];
>>
    owner: CvSTC.Owners ← AtoVsrc;

    menuItemProc ← CvSTC.CommonMenu;
    destroy ← CvSTC.DestroyCommon;
    IF oldInstance = NIL THEN
```

```
        instance ← CvSTC.CreateCommon[cvData, options, window, owner ! NSFile.Error, CvSTC.Problem => {owner ← backstop; instance ← NIL;
        CONTINUE}]
    ELSE
        {
        my: CvSTC.Common ← oldInstance;
        my.window ← window;  --/* AR 13535: update window handle */
        instance ← my;
        };

    --/* make formwindow */
    CvSTC.CreateFW[instance, window, owner];
};


AsciiToVPDstOps: PUBLIC Converter.DependentOptionProc = {
<< = PROCEDURE [options: BOOLEAN ← TRUE, cvData: Converter.CvData, which: Converter.FormatToUse, srcFormat: XString.Reader, destFormat:
XString.Reader, window: Window.Handle, oldInstance: LONG POINTER ← NIL] RETURNS [menuItemProc: Converter.MenuItemProc, destroy:
Converter.DestroyProc, instance: LONG POINTER];
>>
    owner: CvSTC.Owners ← AtoVdst;

    menuItemProc ← CvSTC.CommonMenu;
    destroy ← CvSTC.DestroyCommon;
    IF oldInstance = NIL THEN
        instance ← CvSTC.CreateCommon[cvData, options, window, owner ! NSFile.Error, CvSTC.Problem => {owner ← backstop; instance ← NIL;
        CONTINUE}]
    ELSE
        {
        my: CvSTC.Common ← oldInstance;
        my.window ← window;  --/* AR 13535: update window handle */
        instance ← my;
        };

    --/* make formwindow */
    CvSTC.CreateFW[instance, window,  owner];
};

--====================
-- PROCEDURES
--====================

AtoV: Converter.ConvertProc = {
    aborted: BOOLEAN ← FALSE;
    start: DocInterchangeDefs.StartCreationStatus ← lastAvailable;
    finish: DocInterchangeDefs.FinishCreationStatus ← lastAvailable;
    avData: AVDataObj;
    pOption: DocInterchangeDefs.PaginateOption;
    docSession: NSFile.Session;
    dst,
    src: CvSTC.Common ← NIL;
    lineHtInPoints: CARDINAL ← 18;
    --/* local proc */
    POption: PROCEDURE RETURNS [DocInterchangeDefs.PaginateOption] = INLINE
        {
        SELECT Converter.GetPOption[] FROM
          compress => RETURN[compress];
          simple => RETURN[simple];
          none => RETURN[none];
        ENDCASE => ERROR;
        };

    --/* begin code */
    --/* initialize instance data */
    IF dstInstance = NIL THEN  --/* ASSERT: srcInstance also NIL */
        {
        ENABLE NSFile.Error, CvSTC.Problem =>
            {
            msgRb: XString.ReaderBody ← CvSTC.GetMessage[extraErr0];  --" Unrecoverable ASCII conversion error: damaged converter icon. "
            Converter.PostMessage[
              msg: @msgRb,
              cvData: cvData,
              cr: FALSE,
              clear: FALSE];
            IF src # NIL THEN CvSTC.DestroyCommon[src];
            GOTO terminate;
            };
        key: CvSTC.MessageKey ← CvSTC.MessageKey.FIRST;

        --/* assume both are NIL */
        src ← CvSTC.CreateCommon[cvData, FALSE, NIL, AtoVsrc];
        dst ← CvSTC.CreateCommon[cvData, FALSE, NIL, AtoVdst];

        src.text[paraEndsWith] ← CvSTC.ParseItem[
          my: src,
          r: @src.textRb[paraEndsWith],
          item: key].ls;

        dst.text[atovReplaceUnknown] ← CvSTC.ParseItem[
          my: dst,
          r: @dst.textRb[atovReplaceUnknown],
          item: key].ls;
        EXITS terminate => RETURN;
        }
    ELSE
        {
        src ← srcInstance;
        dst ← dstInstance;
```

```
        };

    avData ← [
         source: source,
         input: [NIL],
         cvData: cvData,
         session: session,
         src: src,
         dst: dst,
         background: background,
         fontProps: TRASH,
         paraProps: TRASH,
         pageProps: TRASH,
         doc: TRASH,
         blk: [Space.ScratchMap[count: bufPages], 0, maxPara],
         state: entry,
         z: dst.z];

    BEGIN
      ENABLE
        {
        DocInterchangeDefs.Error => GOTO err;
        UNWIND =>
            {
            avData.blk.blockPointer ← Space.Unmap[pointer: avData.blk.blockPointer];
            IF srcInstance = NIL THEN CvSTC.DestroyCommon[src];
            IF dstInstance = NIL THEN CvSTC.DestroyCommon[dst];
            src ← dst ← NIL;
            };
        };

    --/* open stream on source */
    avData.input ← NSFileStream.Create[
      file: avData.source,
      closeOnDelete: FALSE,
      session: avData.session ! NSFile.Error => {avData.input ← [NIL]; GOTO err}];

    --/* initialize */
    pOption ← POption[];
    DocInterchangePropsDefs.GetFontPropsDefaults[@avData.fontProps];
    DocInterchangePropsDefs.GetParaPropsDefaults[@avData.paraProps];
    DocInterchangePropsDefs.GetPagePropsDefaults[@avData.pageProps];

    --/* apply initial parms */
    SELECT avData.dst.f.font FROM
        CvSTC.modern => avData.fontProps.fontDesc.family ← DocInterchangePropsDefs.modern;
        CvSTC.classic => avData.fontProps.fontDesc.family ← DocInterchangePropsDefs.classic;
        ENDCASE;

    SELECT avData.dst.f.fontSize FROM
        CvSTC.twelve =>
            {
            avData.fontProps.fontDesc.pointSize ← 12;
            avData.paraProps.basicProps.lineHeight ← 18;
            lineHtInPoints ← 18;
            avData.paraProps.basicProps.defaultTabStopSpacing ← (stopsAt * 12);
            };
        CvSTC.twentyFour =>
            {
            avData.fontProps.fontDesc.pointSize ← 24;
            avData.paraProps.basicProps.lineHeight ← 30;
            lineHtInPoints ← 30;
            avData.paraProps.basicProps.defaultTabStopSpacing ← (stopsAt * 24);
            };
        ENDCASE;

    --/* set paragraph leading by counting CRs in paraEndsWith string */
    BEGIN
    lcount: CARDINAL ← 0;
    eop: LONG STRING ← avData.src.text[paraEndsWith];

    IF eop # NIL THEN
      FOR i: CARDINAL IN [0..eop.length) DO
        IF eop[i] = Ascii.CR THEN lcount ← lcount + 1;
      ENDLOOP;
    IF lcount > 1 THEN
      avData.paraProps.basicProps.preLeading ← lineHtInPoints * (lcount - 1) / 2
    ELSE
      avData.paraProps.basicProps.preLeading ← 0;
    avData.paraProps.basicProps.postLeading ← 0;
    END;

    --/* StartCreation checks process priority to determine forkedness */
    [doc: avData.doc, status: start] ← DocInterchangeDefs.StartCreation[
        paginateOption: pOption,
        initialFontProps: @avData.fontProps,
        initialParaProps: @avData.paraProps,
        initialPageProps: @avData.pageProps ! NSFile.Error => {
          IF error = [space[mediumFull]] THEN
            start ← notEnoughDiskSpace
          ELSE
            start ← lastAvailable;
          CONTINUE}];

    SELECT start FROM
        ok => NULL;
        notEnoughDiskSpace =>
```

```
            {
            Post[ConverterMsg.Get[ConverterMsg.koutOfSpace], avData.cvData];
            GOTO err;
            };
        ENDCASE =>
            {
            Post[ConverterMsg.Get[ConverterMsg.kunknownProblem], avData.cvData];
            GOTO err;
            };

    --/* enter state graph */
    BEGIN
      ENABLE ABORTED => {aborted ← TRUE; CONTINUE};
       StateMachine[@avData];
    END;

    --/* paginating */
    IF option # none THEN
        {
        mrb: XString.ReaderBody ← CvSTC.GetMessage[paginating];
        Converter.PostMessage[
          msg: @mrb,
          cvData: cvData,
          cr: FALSE,
          clear: FALSE];
        };

    --/* user may have partial doc after an abort, so allow paginate/finish */
    --/* reset abort tests.  User must abort paginate separately. */
    IF aborted THEN
        {
        IF avData.background THEN
            BackgroundProcess.ResetUserAbort[]
        ELSE
            TIP.ResetUserAbort[NIL];
        };

    --/* paginate and finish */
    [docFile: dest, session: docSession, status: finish] ← DocInterchangeDefs.FinishCreation[
      docPtr: @avData.doc,
      checkAbortProc: UserAbortsPaginate,
      checkAbortClientData: @avData];

    IF finish = aborted THEN
        {
        aborted ← TRUE;
        Post[ConverterMsg.Get[ConverterMsg.kuserAbort], cvData];
        };

    --/* re-open dest in session */
    IF dest # NSFile.nullHandle THEN
        {
          ENABLE NSFile.Error =>
            {
            NSFile.Close[dest, docSession ! NSFile.Error => CONTINUE];
            dest ← NSFile.nullHandle;
            CONTINUE;
            };
        tmpRef: NSFile.Reference;
        tmp: NSFile.Handle ← dest;

        tmpRef ← NSFile.GetReference[file: dest, session: docSession];
        dest ← NSFile.OpenByReference[reference: tmpRef, session: avData.session];
        NSFile.Close[tmp, docSession];
        --/* if this process is clientBackground, docSession must be logged off */
        IF background THEN NSFile.Logoff[docSession ! NSFile.Error => CONTINUE];
        };

    EXITS err => NULL;
    END;
    IF avData.input # NIL THEN Stream.Delete[avData.input];
    IF avData.blk.blockPointer # NIL THEN
        avData.blk.blockPointer ← Space.Unmap[avData.blk.blockPointer];
    --/* destroy instance data if created by this proc call */
    IF srcInstance = NIL AND src # NIL THEN CvSTC.DestroyCommon[src];
    IF dstInstance = NIL AND dst # NIL THEN CvSTC.DestroyCommon[dst];
    IF finish # ok OR aborted THEN
       Post[ConverterMsg.Get[ConverterMsg.kdataSkipped], cvData];
};


CheckAbort: PROC [background: BOOLEAN] RETURNS [yes: BOOLEAN] = INLINE {
    yes ← (background AND BackgroundProcess.UserAbort[]) OR
      (NOT background AND TIP.UserAbort[NIL]);
};


FlushText: PROC [av: AVData, para: XString.Writer] = {
    r: XString.Reader ← XString.ReaderFromWriter[para];

    IF CheckAbort[av.background] THEN ERROR ABORTED;
    IF XString.ByteLength[r] > 0 THEN
        {
        DocInterchangeDefs.AppendText[
          to: [doc[av.doc]],
          text: r,
          textEndContext: XString.WriterInfo[para].endContext,
```

```
          fontProps: @av.fontProps];
        XString.ClearWriter[para];
        };
};


Post: PROC [msgRb: XString.ReaderBody, cvData: Converter.CvData] = {
    Converter.PostMessage[
      msg: @msgRb,
      cvData: cvData,
      cr: TRUE,
      clear: FALSE];
};


<<
--=============================================================================
StateMachine
This procedure implements a state graph, which is depicted in auxiliary documentation.  The state machine handles the input data
character by character, although the i/o is optimized using block buffers.  Note that the XString.Writer "para" is the output buffer
that gets appended to the document every time text is flushed (see FlushText).  Hereafter are described, briefly, the states, the entry
conditions, exit conditions, and special circumstances:

- entry
The state machine is always entered here.  The entry conditions are that the index "n" references the next character to be handled.  The
next state is determined by the value of the character "c".  The mode "ignore" determines whether white space is treated as standard
text, or as should be handled by the special ignoreTrailing state.  If the character "c" matches the first character of the end of
paragraph string "eop0", then the next state is endPara.  Otherwise, the next state is "append".  Note that the variable "nextState"
does NOT refer to the state executed after entry, but rather the state that the next state RETURNS TO.  Although this violates strict
state machine implementation algorithms, it saves logic.

- append
The state is entered with the character "c", and a valid nextState.  It translates the character "c" to a VP character, and appends it to
the output buffer "para".  Certain special cases are handled.  The exit condition is a valid nextState, which becomes "state".

- ignoreTrailing
The purpose of this state is to implement deletion of white space that precedes an end of line sequence, if the user so desires.  The
state is entered either from entry with "c" being whitespace, or from ignoreTrailing, with "n" indicating the next character to handle.
Variables are initialized to indicate the beginning of whitespace characters.  The state is exited if eop0 is found, or a nonwhitespace
character is found before the end of line.

- maxExceeded
This state handles an overflow exception.  It is entered if "para" is about to exceeded its limits.  A new paragraph is forced if this
state is entered.  It returns to entry.

- endPara
This state tries to determine if the end of a paragraph has been found.  It is entered if the character "c" matched eop0, or (from
endPara itself) if the input text continues to match the string "s".  If a paragraph ending is found, the paragraph is flushed.  The
state returns to entry either if there is a complete match, or of there is a mismatch.  Several special cases are handled.

The state machine loops until input is exhausted.
--=============================================================================
>>


StateMachine: PROC [av: AVData] = {
    lastBlock: BOOLEAN ← FALSE;
    flushed: BOOLEAN ← FALSE;  --/* controls appending text to doc */
    ignore: BOOLEAN ← av.dst.f.ignoreTrailing.value;
    includeTelCodes: BOOLEAN ← av.dst.f.includeTelCodes.value;
    eop: CARDINAL ← 0;  --/* index into paraEndsWith string */
    para: XString.WriterBody ← XString.NewWriterBody[maxLength: paraLen, z: av.z];
    digits: LONG STRING ← String.MakeString[maxlength: 100, z: av.z];
    telCodes: LONG STRING ← String.MakeString[maxlength: (5 * paraLen), z: av.z];
    state: AVState ← entry;
    blankCount: CARDINAL ← 0;  --/* count of "white" characters in buffer */
    blkCount: CARDINAL ← 0;  --/* number of blocks read */
    lastBlkCount: CARDINAL;  --/* for saving "blkCount" */
    nextState: AVState;  --/* the state a state goes back to */
    getNextBlock: BOOLEAN;
    bytes: CARDINAL;
    why: Stream.CompletionCode;
    eop0: CHARACTER;   --/* first character of end-of-paragraph text */
    unknown: LONG STRING;  --/* copy of user defined replacement text */
    blanksStart: CARDINAL;  --/* index into buffer for beginning of blanks */
    map: LONG POINTER TO CtoVPCharMap;
    blk: LONG POINTER TO PACKED ARRAY INTEGER[0..0) OF Environment.Byte;
    n: CARDINAL;  --/* current character in blk */
    last,
    lastFlush,
    c: CHARACTER;
    stc: TelegraphCode ← 0;  -- current STC value
    xc: XString.Character;  -- converted STC
    convertSTC: BOOLEAN ← TRUE;
    goodStc: BOOLEAN ← TRUE;
    codes: CARDINAL;
    tct: TelegraphCodeTable;

    FlushSTC: PROCEDURE = {
      Period: TYPE = {month, hour, day, none};
      period: Period;
      decade: CARDINAL [0..3];
      digit: CARDINAL [0..9];
      IF digits.length = 0 THEN RETURN;
      IF (digits.length = 4 AND convertSTC) THEN {
```

```
        goodStc ← TRUE;
        String.AppendString[from: digits, to: telCodes];
        codes ← codes + 1;
        String.AppendChar[s: telCodes, c: Ascii.SP];
        IF codes MOD 5 = 0 THEN String.AppendChar[s: telCodes, c: Ascii.SP];
        IF codes MOD 10 = 0 THEN String.AppendChar[s: telCodes, c: Ascii.SP];
        stc ← String.StringToDecimal[digits];
        IF (xc ← tct[stc]) = XChar.not THEN {
          IF av.src.f.avCodeScheme = CvSTC.stc THEN {
            period ← none;
            SELECT stc FROM
              IN [9701..9712] => {
                period ← month;
                digit ← stc MOD 10;
                decade ← (stc / 10) MOD 10;
                };
              IN [9800..9824] => {
                period ← hour;
                digit ← stc MOD 10;
                decade ← (stc / 10) MOD 10;
                IF stc = 9800 THEN XString.AppendChar[to: @para, c: XChar.Make[41B, 73B], extra: paraLen];  -- kanji zero
                };
              IN [9901..9931] => {
                period ← day;
                digit ← stc MOD 10;
                decade ← (stc / 10) MOD 10;
                };
              9999 => {
                IF last # Ascii.CR AND nextState # endPara THEN
                  XString.AppendChar[to: @para, c: xNewLine, extra: paraLen];
                };
              ENDCASE => {
                FOR i: CARDINAL IN [0..unknown.length) DO
                  XString.AppendChar[to: @para, c: map[unknown[i]], extra: paraLen];
                ENDLOOP
                };
            IF period # none THEN {
              SELECT decade FROM
                1 => XString.AppendChar[to: @para, c: XChar.Make[241B, 101B], extra: paraLen];
                2 => XString.AppendChar[to: @para, c: XChar.Make[322B, 323B], extra: paraLen];
                3 => XString.AppendChar[to: @para, c: XChar.Make[256B, 324B], extra: paraLen];
                ENDCASE;
              SELECT digit FROM
                1 => XString.AppendChar[to: @para, c: XChar.Make[241B, 42B], extra: paraLen];
                2 => XString.AppendChar[to: @para, c: XChar.Make[241B, 72B], extra: paraLen];
                3 => XString.AppendChar[to: @para, c: XChar.Make[241B, 113B], extra: paraLen];
                4 => XString.AppendChar[to: @para, c: XChar.Make[241B, 136B], extra: paraLen];
                5 => XString.AppendChar[to: @para, c: XChar.Make[241B, 146B], extra: paraLen];
                6 => XString.AppendChar[to: @para, c: XChar.Make[241B, 302B], extra: paraLen];
                7 => XString.AppendChar[to: @para, c: XChar.Make[241B, 345B], extra: paraLen];
                8 => XString.AppendChar[to: @para, c: XChar.Make[241B, 252B], extra: paraLen];
                9 => XString.AppendChar[to: @para, c: XChar.Make[241B, 360B], extra: paraLen];
                ENDCASE;
              SELECT period FROM
                month => XString.AppendChar[to: @para, c: XChar.Make[241B, 275B], extra: paraLen];
                hour => XString.AppendChar[to: @para, c: XChar.Make[241B, 373B], extra: paraLen];
                day => XString.AppendChar[to: @para, c: XChar.Make[241B, 132B], extra: paraLen];
                ENDCASE;
              };
            }
            ELSE {
              FOR i: CARDINAL IN [0..unknown.length) DO
                XString.AppendChar[to: @para, c: map[unknown[i]], extra: paraLen];
              ENDLOOP
              }
            }
          ELSE
            XString.AppendChar[to: @para, c: xc, extra: paraLen];
          }
        ELSE {
          IF lastFlush = Ascii.SP AND ~goodStc THEN XString.AppendChar[to: @para, c: map[Ascii.SP], extra: paraLen];
          XString.AppendSTRING[to: @para, from: digits, extra: paraLen];
          goodStc ← FALSE;
          };
        digits.length ← 0;
        lastFlush ← c;
        };  -- FlushSTC

      FlushTelCodes: PROCEDURE = {
        family: DocInterchangePropsDefs.Family ← av.fontProps.fontDesc.family;
        pointSize: CARDINAL ← av.fontProps.fontDesc.pointSize;
        lineHeight: CARDINAL ← av.paraProps.basicProps.lineHeight;
        preLeading: CARDINAL ← av.paraProps.basicProps.preLeading;
        postLeading: CARDINAL ← av.paraProps.basicProps.postLeading;

        IF telCodes.length = 0 THEN RETURN;
        IF includeTelCodes THEN {
          av.fontProps.fontDesc.family ← helvetica;
          av.fontProps.fontDesc.pointSize ← 9;
          av.paraProps.basicProps.lineHeight ← 12;
          av.paraProps.basicProps.preLeading ← 2;
          IF av.dst.f.fontSize = CvSTC.twelve THEN
            av.paraProps.basicProps.postLeading ← 8
          ELSE
            av.paraProps.basicProps.postLeading ←  8;

          FlushText[av, @para];
```

```
        XString.AppendSTRING[to: @para, from: telCodes, extra: paraLen];
        DocInterchangeDefs.AppendNewParagraph[
          to: [doc[av.doc]], paraProps: @av.paraProps, fontProps: @av.fontProps, nToAppend: 1];
        FlushText[av, @para];

        av.fontProps.fontDesc.family ← family;
        av.fontProps.fontDesc.pointSize ← pointSize;
        av.paraProps.basicProps.lineHeight ← lineHeight;
        av.paraProps.basicProps.preLeading ← preLeading;
        av.paraProps.basicProps.postLeading ← postLeading;
        };
      telCodes.length ← 0;
      codes ← 0;
      };  -- FlushTelCodes

    --/* initialize */
    map ← g.isomap;
    IF av.src.f.avCodeScheme = CvSTC.stc THEN tct ← prc ELSE tct ← roc;

    --/* para is a buffer of VP characters that gets appended to the doc */
    XString.ClearWriter[@para];
    digits.length ← 0;
    telCodes.length ← 0;
    codes ← 0;
    eop0 ← IF av.src.text[paraEndsWith] # NIL THEN
        av.src.text[paraEndsWith][0]
    ELSE
        Ascii.NUL;
    last ← Ascii.NUL;
    unknown ← av.dst.text[atovReplaceUnknown];
    IF unknown = NIL THEN
        {
        --/* so we don't have to test for NIL again */
        unknown ← "?"L;
        unknown.length ← 0;
        };
    --/* make sure getNextBlock is TRUE first time */
    n ← av.blk.stopIndexPlusOne;
    blk ← av.blk.blockPointer;

    --/* enter state graph */
    DO
        getNextBlock ← n >= av.blk.stopIndexPlusOne;

        IF getNextBlock THEN
            {
            IF lastBlock THEN
                {
                --/* might have one last character pending */
                IF state = append THEN
                    {
                    nextState ← entry;
                    GOTO oneLastLoop;
                    };
                FlushSTC[];
                FlushText[av, @para];
                FlushTelCodes[];
                EXIT;  --/* state graph */
                };
            IF CheckAbort[av.background] THEN ERROR ABORTED;
            av.blk.stopIndexPlusOne ← maxPara;
            [bytesTransferred: bytes, why: why] ← Stream.GetBlock[
              sH: av.input,
              block: av.blk];
            lastBlock ← why # normal;
            av.blk.stopIndexPlusOne ← bytes;
            blk ← av.blk.blockPointer;
            n ← 0;
            --/* guard against blkCount overflow */
            blkCount ← IF blkCount = CARDINAL.LAST THEN 0 ELSE blkCount + 1;
            EXITS oneLastLoop => NULL;
            };

        SELECT state FROM
            entry =>
                {
                --/* get next character */
                c ← LOOPHOLE[blk[n], CHAR];
                --/* set up next state */
                SELECT c FROM
                    Ascii.SP, Ascii.TAB => IF ignore THEN
                        {
                        state ← ignoreTrailing;
                        blanksStart ← n;
                        blankCount ← 0;
                        lastBlkCount ← blkCount;
                        }
                      ELSE
                        {
                        state ← append;
                        nextState ← entry;
                        n ← n + 1;
                        };
                    eop0 =>
                        {
                        FlushSTC[];
                        state ← endPara;
```

```
            };
        ENDCASE =>
            {
            state ← append;
            nextState ← entry;
            n ← n + 1;
            };
    };
append =>
    {
    --/* ASSERT: order of select arms is critically important */
    SELECT c FROM
        IN ['0..'9] =>      -- [60C..71C]
            {
            --/* digits 0 through 9 */
            IF convertSTC THEN
              String.AppendChar[s: digits, c: c]
            ELSE
              XString.AppendChar[to: @para, c: map[c], extra: paraLen];
            state ← nextState;
            };
        Ascii.SP =>         -- 40C
            {
            IF (last IN [60C..71C] AND convertSTC) THEN FlushSTC[]
            ELSE IF NOT (last = ', OR last = '.) THEN
              XString.AppendChar[to: @para, c: map[c], extra: paraLen];
            state ← nextState;
            };
        '# =>       -- 43C, toggle convert switch
            {
            FlushSTC[];
            convertSTC ← ~convertSTC;
            state ← nextState;
            };
        IN [40C..176C], 251C, 262C, 271C, 272C =>
            {
            --/* standard characters */
            FlushSTC[];
            XString.AppendChar[to: @para, c: map[c], extra: paraLen];
            state ← nextState;
            };
        Ascii.CR =>
            {
            FlushSTC[];
            IF CheckAbort[av.background] THEN ERROR ABORTED;
            IF nextState = entry THEN
                {
                --/* smart white space */
                SELECT last FROM
                    Ascii.SP,
                    Ascii.TAB,
                    Ascii.CR,
                    Ascii.LF,
                    aHyphen => NULL; --/* just drop CR */
                    ENDCASE =>
                        {
                        XString.AppendChar[
                          to: @para,
                          c: map[Ascii.SP],
                          extra: paraLen];
                        };
                };
            --/* CR is skipped if we came from endPara */
            state ← nextState;
            };
        Ascii.LF =>
            {
            FlushSTC[];
            IF last # Ascii.CR AND nextState # endPara THEN
                {
                --/* append newline */
                XString.AppendChar[
                  to: @para,
                  c: XCharSet0.Make[newLine],
                  extra: paraLen];
                };
            --/* LF is skipped if we came from endPara */
            --/* or if last = CR */
            state ← nextState;
            };
        Ascii.TAB =>
            {
            --/* tab */
            FlushSTC[];
            XString.AppendChar[to: @para, c: map[c], extra: paraLen];
            state ← nextState;
            };
        Ascii.FF =>
            {
            --/* flush page */
            FlushSTC[];
            FlushText[av, @para];
            FlushTelCodes[];
            DocInterchangeDefs.AppendPageBreak[
              to: av.doc,
              fontProps: @av.fontProps];
            state ← nextState;
```

40..377

```
                };
            Ascii.NUL =>
                {
                --/* skip */
                state ← nextState;
                };
            ENDCASE =>
                {
                FlushSTC[];
                FOR i: CARDINAL IN [0..unknown.length) DO
                    XString.AppendChar[to: @para, c: map[unknown[i]], extra: paraLen];
                ENDLOOP;
                state ← nextState;
                };
        last ← c;
        <<
        * XString.CharacterLength is an expensive operation.
        * We make the observation that
        * ByteLength >= CharacterLength ALWAYS. Therefore
        * use faster ByteLength to determine if CharacterLength should
        * be called
        >>
        IF XString.ByteLength[XString.ReaderFromWriter[@para]] > maxPara THEN
            {
            IF XString.CharacterLength[XString.ReaderFromWriter[@para]] > maxPara
              AND nextState # endPara THEN
                state ← maxExceeded;
            };
        };
    ignoreTrailing =>
        {
        --/* get next char if other than first entry */
        IF blanksStart # n THEN
            {
            last ← c;
            c ← LOOPHOLE[blk[n], CHAR];
            };

        SELECT c FROM
            Ascii.SP, Ascii.TAB =>
                {
                state ← ignoreTrailing;
                n ← n + 1;
                blankCount ← blankCount + 1;
                };
            eop0 =>
                {
                --/* end found, so skip all trailing blanks */
                state ← endPara;
                };
            Ascii.CR =>
                {
                --/* NOTE: this arm must follow the eop0 arm */
                --/* ASSERT: eop0 # Ascii.CR by order of execution */
                --/* replace CR with space, and skip blanks */
                XString.AppendChar[
                  to: @para,
                  c: map[Ascii.SP],
                  extra: paraLen];
                state ← entry;
                blankCount ← 0;
                n ← n + 1;
                };
            ENDCASE =>
                {
                IF CheckAbort[av.background] THEN ERROR ABORTED;
                --/* whoops!  Not eol, so append */
                IF lastBlkCount # blkCount THEN
                    {
                    --/* blanks straddle blocks */
                    THROUGH [1..blankCount] DO
                        XString.AppendChar[
                          to: @para,
                          c: map[Ascii.SP],
                          extra: paraLen];
                    ENDLOOP;
                    }
                ELSE
                    FOR i: CARDINAL IN [blanksStart..n) DO
                        XString.AppendChar[
                          to: @para,
                          c: map[LOOPHOLE[blk[i], CHAR]],
                          extra: paraLen];
                    ENDLOOP;
                blankCount ← 0;
                state ← entry;
                };
        };
    maxExceeded =>
        {
        FlushText[av, @para];
        FlushTelCodes[];
        DocInterchangeDefs.AppendNewParagraph[
          to: [doc[av.doc]],
          paraProps: @av.paraProps,
          fontProps: @av.fontProps,
          nToAppend: 1];
```

```
        state ← entry;
        };
    endPara =>
        {
        s: LONG STRING ← av.src.text[paraEndsWith];

        IF s = NIL THEN
            {
            state ← entry;
            nextState ← entry;
            n ← n + 1;
            flushed ← FALSE;
            GOTO restart;
            };

        IF eop # 0 THEN
            {
            last ← c;
            c ← LOOPHOLE[blk[n], CHAR];
            };

        --/* if we are at the end of s, then match! */
        IF eop >= s.length THEN
            {
            IF NOT flushed THEN
                --/* flush all text */
                FlushText[av, @para];

            FlushTelCodes[];
            DocInterchangeDefs.AppendNewParagraph[
              to: [doc[av.doc]],
              paraProps: @av.paraProps,
              fontProps: @av.fontProps,
              nToAppend: 1];

            IF flushed THEN
                --/* flush following text */
                FlushText[av, @para];

            eop ← 0;
            state ← entry;
            nextState ← entry;
            flushed ← FALSE;
            GOTO restart;
            };

        --/* c match with end-of-paragraph? */
        IF s[eop] = c THEN
            {
            eop ← eop + 1;
            n ← n + 1;
            }
        ELSE
            {
            --/* false alarm */
            IF ignore THEN
                --/* ouch, we interrupted ignoreTrailing */
                FOR j: CARDINAL IN [0..eop) DO
                    IF s[j] = Ascii.CR THEN GOTO oneCR;
                    IF s[j] # Ascii.SP OR s[j] # Ascii.TAB THEN
                      GOTO notWhite;
                REPEAT
                    oneCR =>
                        {
                        --/* replace CR with one blank */
                        XString.AppendChar[
                          to: @para,
                          c: map[Ascii.SP],
                          extra: paraLen];
                        --/* other blanks ignored */
                        blankCount ← 0;
                        };
                    notWhite =>
                        {
                        --/* flush blankCount characters */
                        IF lastBlkCount # blkCount THEN
                            {
                            --/* blanks straddle blocks */
                            THROUGH [1..blankCount] DO
                                XString.AppendChar[
                                  to: @para,
                                  c: map[Ascii.SP],
                                  extra: paraLen];
                            ENDLOOP;
                            }
                        ELSE
                            FOR i: CARDINAL IN [blanksStart..blanksStart+blankCount) DO
                                XString.AppendChar[
                                  to: @para,
                                  c: map[LOOPHOLE[blk[i], CHAR]],
                                  extra: paraLen];
                            ENDLOOP;
                        blankCount ← 0;
                        };
                    FINISHED =>
                        {
                        --/* include current chars in blankCount */
```

```
                        blankCount ← blankCount + (MAX[eop,1] - 1);
                        state ← ignoreTrailing;
                        };
                ENDLOOP;

            --/* set up for next state */
            IF (c = Ascii.SP OR c = Ascii.TAB)
              AND ignore
              AND state # ignoreTrailing THEN
                {
                state ← ignoreTrailing;
                blanksStart ← n;
                blankCount ← 0;
                lastBlkCount ← blkCount;
                }
            ELSE
                {
                state ← append;
                n ← n + 1;
                --/* account for any CRs */
                --/* IF last = CR, then kludge handled it */
                IF last # Ascii.CR THEN
                    FOR j: CARDINAL IN [0..eop) DO
                        IF s[j] = Ascii.CR THEN GOTO foundCR;
                    REPEAT
                        foundCR =>
                            {
                            --/* replace one or more CRs with one blank */
                            XString.AppendChar[
                              to: @para,
                              c: map[Ascii.SP],
                              extra: paraLen];
                            };
                        FINISHED => NULL;
                    ENDLOOP;
                };
            eop ← 0;
            nextState ← entry;
            flushed ← FALSE;
            GOTO restart;
            --/* end of false alarm */
            };

        --/* continue looking for eop */
        IF c = Ascii.CR THEN
            {
            --/* flush preceding text, clear buffer */
            FlushText[av, @para];
            flushed ← TRUE;
            };

        --/* translate character */
        state ← append;
        nextState ← endPara;

        --/* special look-ahead kludge to make naked CR's work */
        IF c = Ascii.CR
          AND NOT ignore
          AND eop < s.length
          AND n < av.blk.stopIndexPlusOne
          AND s[eop] # LOOPHOLE[blk[n], CHAR] THEN
            {
            --/* smart white space */
            SELECT last FROM
                Ascii.SP,
                Ascii.TAB,
                Ascii.CR,
                Ascii.LF,
                aHyphen => NULL; --/* just drop CR */
                ENDCASE =>
                    {
                    XString.AppendChar[
                      to: @para,
                      c: map[Ascii.SP],
                      extra: paraLen];
                    };
            };
        EXITS restart => NULL;
        };
    ENDCASE;
ENDLOOP;

--/* clean up */
XString.FreeWriterBytes[@para];
String.FreeString[s: digits, z: av.z];
String.FreeString[s: telCodes, z: av.z];
};

UserAbortsPaginate: DocInterchangeDefs.CheckAbortProc = {
<< = PROCEDURE [clientData: LONG POINTER] RETURNS [abort: BOOL];
>>
    data: AVData = clientData;

    abort ← CheckAbort[data.background];
};
```

```
ZzInit: PROC = {
    pz: UNCOUNTED ZONE = BWSZone.Permanent[];
    tableName: XString.ReaderBody;
    fh: NSFile.Handle;

    tableName ← XString.FromSTRING["PRC.tcTable"L];
    fh ← GetTableFile[@tableName];
    prc ← CreateTable[fh];

    tableName ← XString.FromSTRING["ROC.tcTable"L];
    fh ← GetTableFile[@tableName];
    roc ← CreateTable[fh];

    --/* these Spaces should not be unmapped while this application is loaded */
    g ← [
      isomap: Space.ScratchMap[(words + Environment.wordsPerPage-1) / Environment.wordsPerPage],
      pz: pz];

    FOR c: CHARACTER IN CHARACTER DO
        g.isomap[c] ← XCharSet0.Make[LOOPHOLE[c]];
    ENDLOOP;

    g.isomap[VAL[XCharSet0.Make[comma]]] ← XCharSet41Extra.Make[japaneseComma];  -- 42B
    g.isomap[VAL[XCharSet0.Make[period]]] ← XCharSet41Extra.Make[japanesePeriod];  -- 43B
    g.isomap[VAL[XCharSet0.Make[leftSingleQuote]]] ← XCharSet41Extra.Make[leftJapaneseQuote];  -- 126B
    g.isomap[VAL[XCharSet0.Make[rightSingleQuote]]] ← XCharSet41Extra.Make[rightJapaneseQuote];  -- 130B
    g.isomap[VAL[XCharSet0.Make[leftDoubleQuote]]] ← XCharSet41Extra.Make[leftJapaneseDoubleQuote];  -- 127B
    g.isomap[VAL[XCharSet0.Make[rightDoubleQuote]]] ← XCharSet41Extra.Make[rightJapaneseDoubleQuote];  -- 131B

<<
    g.isomap[VAL[XCharSet0.Make[openBrace]]] ← XCharSet357.Make[leftWhiteLenticularBracket];  -- 72B
    g.isomap[VAL[XCharSet0.Make[closeBrace]]] ← XCharSet357.Make[rightWhiteLenticularBracket];  -- 73B
    g.isomap[VAL[XCharSet0.Make[comma]]] ← XCharSet41Extra.Make[squareComma];  -- 44B
    g.isomap[VAL[XCharSet0.Make[period]]] ← XCharSet41Extra.Make[squarePeriod];  -- 46B
>>

};


GetTableFile: PROC [tableName: XString.Reader] RETURNS [file: NSFile.Handle] = {
    -- assume folder is in System catalog
    folderName: XString.ReaderBody ← XString.FromSTRING["Transliteration Tables"L];
    ref: NSFile.Reference ← TRASH;
    ref ← GetFile[@folderName, tableName];
    file ← NSFile.OpenByReference[reference: ref, session: Catalog.beforeLogonSession];
};


GetFile: PROC [folderName, fileName: XString.Reader]
    RETURNS [file: NSFile.Reference ← NSFile.nullReference] = {
    directory: NSFile.Handle ← TRASH;

    FileFromName: PROC [value: XString.Reader] = {
        nsName: NSString.String ← XString.NSStringFromReader[
          r: value, z: BWSZone.shortLifetime];
        handle: NSFile.Handle ← NSFile.nullHandle;

        handle ← NSFile.Find[
          directory: directory,
          scope: [filter: [matches[attribute: [name[nsName]]]]],
          session: Catalog.beforeLogonSession
            ! NSFile.Error => {handle ← NSFile.nullHandle; CONTINUE}];

        IF handle # NSFile.nullHandle THEN {
          file ← NSFile.GetReference[file: handle, session: Catalog.beforeLogonSession];
          NSFile.Close[file: handle, session: Catalog.beforeLogonSession]};
        NSString.FreeString[z: BWSZone.shortLifetime, s: nsName];
    };

    directory ← Catalog.GetFile[name: folderName, readonly: TRUE, session: Catalog.beforeLogonSession];
    FileFromName[fileName];
    NSFile.Close[file: directory, session: Catalog.beforeLogonSession];
};


CreateTable: PROC [table: NSFile.Handle]
    RETURNS [tct: TelegraphCodeTable] = {
    count: CARDINAL = ((TelegraphCode.LAST+1)+(Environment.wordsPerPage-1))/Environment.wordsPerPage;
    tct ← NSSegment.Map [
    origin: [file: table, base: 0, count: count],
    swapUnits: [uniform[1]],
    session: Catalog.beforeLogonSession].pointer;
};


--/* main line code */

ZzInit[];


END...

LOG
16-Mar-87 14:06:16 - Caro - Created
26-Jun-87 11:21:47 - Caro - Added error catcher in ConvertProc over CreateCommon,
                            Caught NSFile.Error in Logoff
```

typeØ

```
29-Jun-87 13:13:00 - Caro - Added lineHtInPoints, AFTER setting
10-Jul-87 10:55:05 - Caro - Added aHyphen testing for smart spacing
19-Aug-87 11:01:32 - Caro - Fixed AR 13535 by updating oldInstance window
16-Sep-87 13:48:21 - Caro - isomap accentFirst from 241C to 301C
                            isomap lowGraphFirst from 0 to 241C
                            isomap lowGraphLast from 0 to 277C
                            pcmap accentLast from 257C to 245C
                            pcmap hiGraphFirst from 260C to 246C
12-Feb-88 12:58:57 - Shinsato - In AtoV, made sure eop # NIL before counting CR
                                in eop.
```

```
-- CacheLexiconDefs.mesa
-- Revised by   Walden:    3-Jul-84 12:26:19

DIRECTORY
  TxtScanDefs USING [ReadonlyWordFlags],
  XString USING [Reader];

CacheLexiconDefs: DEFINITIONS = {

  CacheLexiconHandle: TYPE = LONG POINTER TO CacheLexiconObject;

  CacheLexiconObject: TYPE;

-- PROCS

  --added = FALSE if entry already there OR word is empty (offset = limit = 0)
  AddWord: PROC [l: CacheLexiconHandle, word: XString.Reader, wordFlags: TxtScanDefs.ReadonlyWordFlags] RETURNS [added: BOOL];

  Close: PROC [l: CacheLexiconHandle];

  LookUpWord: PROC [l: CacheLexiconHandle, word: XString.Reader, wordFlags: TxtScanDefs.ReadonlyWordFlags] RETURNS [found: BOOL];

  Open: PROC [nEntriesMax: CARDINAL, z: UNCOUNTED ZONE ← NIL] RETURNS [CacheLexiconHandle];
  -- NIL if can't open (due to insufficient space/VM)

  }.
```

```
-- CacheLexiconPack.mesa
-- Last edit: Walden      3-Jul-84 17:16:14

-- hash table lru cache lexicon implementation
-- NOTE: should be an object-monitor

DIRECTORY
  CacheLexiconDefs USING [],
  Environment USING [bytesPerWord, wordsPerPage],
  File USING [nullFile],
  HashTableDefs USING [
    CreateHashTable, DestroyHashTable, EntryFreeProc, FindEntry, HashTable, KeyObject, RelKey, RelKeyObject, MakeEntry, MakeEntryStatus],
  Inline USING [LongCOPY],
  Space USING [InsufficientSpace, Interval, Map, Unmap],
  TxtScanDefs USING [ReadonlyWordFlags],
  XString USING [Reader],
  ZoneMgrDefs USING [GetPredefinedZone];

CacheLexiconPack: PROGRAM
  IMPORTS HashTableDefs, Inline, Space, ZoneMgrDefs
  EXPORTS CacheLexiconDefs
  SHARES XString = {
  OPEN Environment, HashTableDefs, CacheLexiconDefs, XS: XString;

  CacheLexiconHandle: PUBLIC TYPE = LONG POINTER TO CacheLexiconObject;
  CacheLexiconObject: PUBLIC TYPE = MACHINE DEPENDENT RECORD [
    zone: UNCOUNTED ZONE,
    space: Space.Interval,
    nextFreeOffset: CARDINAL,
    hashTable: HashTable,
    relKeyLastFreed: RelKeyObject
    ];

  Word: TYPE = LONG POINTER TO WordObject;
  WordObject: TYPE = ARRAY[0..nWORDSPerWordMax) OF WORD;

  nBytesPerWordMax: CARDINAL = 20;
  nWORDSPerWordMax: CARDINAL = (nBytesPerWordMax+bytesPerWord-1)/bytesPerWord; -- longer words aren't cached

  zSession: UNCOUNTED ZONE = ZoneMgrDefs.GetPredefinedZone[session];

  Bug: SIGNAL [Bugtype] = CODE;
  Bugtype: TYPE = {impossible, unimplemented};

  AddWord: PUBLIC PROC [l: CacheLexiconHandle, word: XS.Reader, wordFlags: TxtScanDefs.ReadonlyWordFlags] RETURNS [added: BOOL] = {
    relKey: RelKey;
    status: MakeEntryStatus;
    key: KeyObject = KeyFromWord[word];

    IF ~ (key.nWords IN (0..SIZE[WordObject])) THEN
      RETURN [FALSE];
      --don't cache anything larger than SIZE[WordObject]

    l.relKeyLastFreed.nWords ← 0;
    [status: status, relKey: relKey] ← HashTableDefs.MakeEntry[l.hashTable, key];

      << if cache is full, MakeEntry will delete the LRU entry, calling FreeEntry, which will set l.relKeyLastFreed; we COULD assume that
      relKey↑ is not changed by hashTable when an entry is recycled [is this a good idea, even though currently true? I think not - how
      to tell reused values from new ones; can't without mods to HashTablePack, and documented guarantee...] >>

    IF status # ok THEN SIGNAL Bug[impossible];
    IF l.relKeyLastFreed.nWords = 0 THEN {  --new entry
      relKey.wordOffset ← l.nextFreeOffset;
      l.nextFreeOffset ← l.nextFreeOffset+SIZE[WordObject];
      }
    ELSE -- old entry was "freed"
      relKey.wordOffset ← l.relKeyLastFreed.wordOffset;

    relKey.nWords ← key.nWords;
    Inline.LongCOPY[from: LOOPHOLE[word.bytes], nwords: key.nWords, to: l.space.pointer+relKey.wordOffset];

    RETURN [TRUE];
    };

  Close: PUBLIC PROC [l: CacheLexiconHandle] = {
    z: UNCOUNTED ZONE = l.zone;

    [] ← Space.Unmap[l.space.pointer];
    HashTableDefs.DestroyHashTable[l.hashTable];
    z.FREE[@l];
    };

  LookUpWord: PUBLIC PROC [l: CacheLexiconHandle, word: XString.Reader, wordFlags: TxtScanDefs.ReadonlyWordFlags] RETURNS [found: BOOL] =
  {
    RETURN[HashTableDefs.FindEntry[l.hashTable, KeyFromWord[word]].found];
    };

  Open: PUBLIC PROC [nEntriesMax: CARDINAL, z: UNCOUNTED ZONE ← NIL] RETURNS [l: CacheLexiconHandle] = {
    nPages: CARDINAL = PagesForWords[nEntriesMax*SIZE[WordObject]];

    IF z = NIL THEN z ← zSession;

    l ← z.NEW[CacheLexiconObject];

    l.zone ← z;
    l.space ← Space.Map[
      window: [file: File.nullFile,
```

```
        base: TRASH,
        count: nPages],
      class: data  ! Space.InsufficientSpace => GOTO abort];

    l.nextFreeOffset ← 0;

    l.hashTable ← HashTableDefs.CreateHashTable[
      nEntriesMax: nEntriesMax,
      extraEntriesOK: FALSE,
      relKeyBase: l.space.pointer,
      clientCtxt: l,
      type: lruCache,
      entryFreeProc: FreeProc].hashTable;

    IF l.hashTable = NIL THEN {
      [] ← Space.Unmap[l.space.pointer];
      GOTO abort;
      };

    EXITS
      abort => z.FREE[@l];  --sets it to NIL
    };

-- PRIVATE PROCS

  FreeProc: HashTableDefs.EntryFreeProc = {
    LOOPHOLE[clientCtxt, CacheLexiconHandle].relKeyLastFreed ← relKey↑;
    };

  -- nWords = whole # of WORDs to contain word's bytes
  KeyFromWord: PROC [word: XS.Reader] RETURNS [KeyObject] = INLINE {
    RETURN [[LOOPHOLE[word.bytes], (word.limit+bytesPerWord-1)/bytesPerWord]]};

  PagesForWords: PROC[words: CARDINAL] RETURNS [pages: CARDINAL] = INLINE {
    RETURN[(words + wordsPerPage-1)/ wordsPerPage]};

  }...

LOG
Walden   2-Jul-84 17:37:04      Created
```

```
-- File: PARCLexicon.mesa - last edit:
-- Walden               21-Apr-87 11:36:15

-- Copyright (C) 1987 by Xerox Corporation. All rights reserved.

DIRECTORY
  NSFile USING [Reference, Type],
  XChar USING [Character],
  XString USING [Reader];

PARCLexicon: DEFINITIONS = {

  --TYPES
  EntryProc: TYPE = PROC [entry: XString.Reader] RETURNS [stop: BOOL ← FALSE];

  Handle: TYPE = LONG POINTER TO Object;
  Object: TYPE;

  Case: TYPE = {allLower, capitalized, allUpper, mixed};

  CharArray: TYPE = ARRAY CARDINAL OF XChar.Character;
  CharDescriptor: TYPE = LONG DESCRIPTOR FOR READONLY CharArray;

  -- CONSTANTS
  fileType: NSFile.Type = 4474;  -- see NSAssignedTypes.

  -- PROCS
  Close: PROC [handle: Handle];

  EnumerateEntries: PROCEDURE [handle: Handle, entryProc: EntryProc];

  GetCodeVersion: PROC RETURNS [CARDINAL];

  GetNEntries: PROC [handle: Handle] RETURNS [LONG CARDINAL];

  -- 'entry' should be all lower-case, with 'case' indicating original state
  -- except for 'mixed' case: entry should have original capitalization
  LookUpEntry: PROC [
    handle: Handle, entry: CharDescriptor, entryCase: Case,
    lookUpAsAbbreviation: BOOL]
    RETURNS [found: BOOL, lexiconCase: Case];

  Open: PROC [fileRef: NSFile.Reference, zone: UNCOUNTED ZONE, activate: BOOL]
    RETURNS [Handle];
    -- return some sort of status here???

  }.
```

```
-- File: PARCLexiconImpl.mesa - last edit:
-- Mader                29-Dec-87 10:45:17
-- Walden               20-Jul-87 10:47:42

-- Copyright (C) 1987 by Xerox Corporation. All rights reserved.

  DIRECTORY
    Inline USING [BytePair, HighByte, LongCOPY, LowByte, LowHalf],
    InlineExtra USING [SwapWords],
    NSFile USING [Close, Handle, Reference, OpenByReference],
    NSSegment USING [GetSizeInPages, Map],
    PARCLexicon,
    Space USING [InsufficientSpace, Unmap],
    XChar USING [LowerCase],
    XCharProps USING [IsUpperCase],
    XCharSet0 USING [Make],
    XString USING [Character];

  PARCLexiconImpl: PROGRAM
    IMPORTS Inline, InlineExtra, NSFile, NSSegment, Space, XChar, XCharProps, XCharSet0
    EXPORTS PARCLexicon =
BEGIN OPEN PARCLexicon;

  -- Types

  AltCharAction: TYPE = RECORD [
    counterIncrement: INTEGER,
    locationIncrement: INTEGER];

  AltCharActionTable: TYPE = ARRAY Byte OF AltCharAction;

  Byte: TYPE = [0..256);

  CodedChar: TYPE = Byte;

  FourBytes: TYPE = MACHINE DEPENDENT RECORD [
    lowWord(0:0..15): Inline.BytePair,
    highWord(1:0..15): Inline.BytePair];

  Handle: TYPE = LONG POINTER TO Object;

  Header: TYPE = LONG POINTER TO HeaderRecord;
  HeaderRecord: TYPE = MACHINE DEPENDENT RECORD[
    bytesInFileDiv256(0): CARDINAL,
    bytesInFileMod256(1:0..7): Byte,
    freeFlag(1:8..8): BOOLEAN,
    hasEowPCounts(1:9..9): BOOLEAN,
    loopFree(1:10..10): BOOLEAN,
    isTransducer(1:11..11): BOOLEAN,
    notMinimized(1:12..12): BOOLEAN,
    hasEpsilon(1:13..13): BOOLEAN,
    nonDeterministic(1:14..14): BOOLEAN,
    startFinalp(1:15..15): BOOLEAN,
    nDictCodes(2:0..7): Byte,
    pointerEntrySize(2:8..15): Byte,
    offset1(3:0..7): Byte,
    offset2(3:8..15): Byte,
    offsetLong(4:0..7): Byte,
    alphabetBase(4:8..15): Byte,
    alphabetTable(5): CARDINAL,
    table1(6): CARDINAL,
    table2(7): CARDINAL,
    startStateIndex(8): CARDINAL,
    fsmBase(9): CARDINAL,
    altCounter(10:0..7): Byte,
    decrementCounter(10:8..15): Byte,
    changeCounter(11:0..7): Byte,
    escapeCode(11:8..15): Byte,
    idate(12): LONG CARDINAL,
    maxStateCost(14): CARDINAL,
    firstFinalChar(15:0..7): Byte,
    firstFinalAltChar(15:8..15): Byte,
    table1Base(16): LONG POINTER,
    totalEowPCount(18): LONG CARDINAL,
    startStateTableBase(20): LONG POINTER,
    firstNSChar(22): CARDINAL,
    alphabetTableSize(23:0..7): Byte,
    nStartTables(23:8..15): Byte,
    startStateTableTableBase(24): CARDINAL,
    startStateFirstTransitionLoc(25): CARDINAL,
    epsilonCode(26:0..7): Byte,
    longestMinimalPathLength(26:8..15): Byte,
    maxEscapes(27:0..7): Byte,
    longPointerLength(27:8..15): Byte,
    dictCodeToNSTableBase(28): CARDINAL,
    table1Counts(29): CARDINAL,
    table2Counts(30): CARDINAL,
    startStateCountTableBase(31): CARDINAL,
    startStateZeroCounts(32): CARDINAL,
    freeByte1(33:0..7): Byte,
    freeByte2(33:8..15): Byte,
    table1EowpCountBase(34): LONG POINTER,
    table2EowpCountBase(36): LONG POINTER];

  Location: TYPE = LONG CARDINAL;
  LocationTable: TYPE = LONG POINTER TO LocationSequence;
  LocationSequence: TYPE = RECORD [SEQUENCE COMPUTED CARDINAL OF Location];
```

```
  LocationTableTable: TYPE = LONG POINTER TO LocationTableSequence;
  LocationTableSequence: TYPE = RECORD [SEQUENCE COMPUTED CARDINAL OF LocationTable];

  Object: PUBLIC TYPE = RECORD [
    altCharActionTable: LONG POINTER TO AltCharActionTable,
    rangeTable: LONG POINTER TO RangeTable,
    offsetTable: LONG POINTER TO OffsetTable,
    fsm: Table,
    table1: Table,
    table2: Table,
    wordChars: WordChars,
    nWordChars: CARDINAL,
    maxWordChars: CARDINAL,
    alphabetTable: Table,
    firstNSChar: XString.Character,
    lastNSChar: XString.Character,
    shift: CodedChar,
    apostrophe: CodedChar,
    startStateTableTable: LocationTableTable,
    secondLevelTable: LocationTableTable,
    nStartStates: CARDINAL,
    nDictCodes: CARDINAL,
    pointer: LONG POINTER,
    file: NSFile.Handle,
    zone: UNCOUNTED ZONE];

  OffsetTable: TYPE = ARRAY Byte OF Byte;

  Range: TYPE = {offset1, offset2, offsetLong, alphabetBase, firstFinalChar, altCounter, changeCounter, firstFinalAltChar,
  decrementCounter};

  RangeTable: TYPE = ARRAY Byte OF Range;

  Table: TYPE = LONG POINTER TO PACKED ARRAY OF Byte;

  TwoByteTable: TYPE = LONG POINTER TO ARRAY OF CARDINAL;

  WordChars: TYPE = LONG POINTER TO WordCharSequence;
  WordCharSequence: TYPE = RECORD [SEQUENCE COMPUTED CARDINAL OF CARDINAL];

  nonExistantCode: CodedChar = 255;

  -- Local Procedures

  AltCharLoc: PROCEDURE [handle: Handle, loc: Location, byte: Byte]
    RETURNS [Location] = INLINE
  BEGIN
  altCharActionTable: LONG POINTER TO AltCharActionTable = handle.altCharActionTable;
  locMod: CARDINAL = Inline.LowHalf [loc] MOD 2;
  base: Table ← handle.fsm + (loc ← loc - locMod)/2;
  locOffset: CARDINAL ← locMod + 1;

    SELECT handle.rangeTable[byte] FROM
    changeCounter,
    firstFinalAltChar,
    decrementCounter =>
    BEGIN -- need to search for alternative
    counter: CARDINAL ← 1;
    altCharAction: AltCharAction ← TRASH;

      DO
        altCharAction ← altCharActionTable[base[locOffset]];

        IF altCharAction.locationIncrement < 0 THEN
          IF base[locOffset+3] = 0 THEN
            locOffset ← locOffset + 6
          ELSE locOffset ← locOffset + 4
        ELSE locOffset ← locOffset + altCharAction.locationIncrement;

        IF (counter ← counter + altCharAction.counterIncrement) = 0 THEN EXIT;
      ENDLOOP;
    END;

    ENDCASE;

    -- DecodePointer:
    SELECT handle.rangeTable[byte ← base[locOffset]] FROM
    alphabetBase,
    firstFinalChar,
    altCounter,
    changeCounter,
    firstFinalAltChar,
    decrementCounter => -- simple character byte
      RETURN [loc + locOffset];

    offsetLong => -- long pointer at current location
      RETURN [
        ThreeByteLocation [
          byte1: handle.offsetTable[byte],
          byte2: base[locOffset+1],
          byte3: base[locOffset+2]
        ]
      ];

    offset2 => -- 2 byte indirect pointer
      RETURN GetPointerTableEntry [
```

```
        handle.table2,
        TwoByteIndex [byte1: handle.offsetTable[byte], byte2: base[locOffset+1]]
      ];

  offset1 => -- single byte indirect table
    RETURN GetPointerTableEntry [handle.table1, handle.offsetTable[byte]];

  ENDCASE => RETURN [0]; -- **** can't get here ****

END;

ByteIsEndOfWord: PROCEDURE [range: Range] RETURNS [BOOLEAN] = INLINE
BEGIN
  RETURN [
    SELECT range FROM
    firstFinalChar,
    altCounter,
    firstFinalAltChar,
    decrementCounter => TRUE,

    offset1,
    offset2,
    offsetLong,
    alphabetBase,
    changeCounter => FALSE,

    ENDCASE => FALSE];
END;

ByteHasAlt: PROCEDURE [range: Range] RETURNS [BOOLEAN] = INLINE
BEGIN
  RETURN [
    SELECT range FROM
    altCounter,
    changeCounter,
    firstFinalAltChar => TRUE,

    offset1,
    offset2,
    offsetLong,
    alphabetBase,
    firstFinalChar,
    decrementCounter => FALSE,

    ENDCASE => FALSE];
END;

ByteHasNext: PROCEDURE [range: Range] RETURNS [BOOLEAN] = INLINE
BEGIN
  RETURN [
    SELECT range FROM
    offset1,
    offset2,
    offsetLong,
    alphabetBase,
    firstFinalChar,
    changeCounter,
    firstFinalAltChar => TRUE,

    altCounter,
    decrementCounter => FALSE,

    ENDCASE => FALSE];
END;

ComputeWordChars: PROCEDURE [handle: Handle, entry: CharDescriptor, case: Case]
  RETURNS [BOOLEAN] = INLINE
BEGIN
length: CARDINAL = entry.LENGTH;
wordChars: WordChars = handle.wordChars;
maxWordChars: CARDINAL = handle.maxWordChars;
shift: CodedChar = handle.shift;
apostrophe: XString.Character = XCharSet0.Make[apostrophe];
wc: CARDINAL ← 0;
increment: CARDINAL ← 1;

  IF length > maxWordChars THEN RETURN [FALSE];

  SELECT case FROM
  capitalized =>
  BEGIN
    wordChars[0] ← shift;
    wc ← 1;
    handle.nWordChars ← length + 1;
  END;

  allUpper =>
  BEGIN
    wordChars[0] ← shift;
    Inline.LongCOPY [from: @wordChars[0], to: @wordChars[1], nwords: length*2 - 1];
    wc ← 1;
    increment ← 2;
  END;

  mixed =>
  BEGIN
    FOR c: CARDINAL IN [0..length) DO
```

```
      char: XString.Character ← entry[c];

        IF XCharProps.IsUpperCase [char] THEN
        BEGIN
          wordChars[wc] ← shift;
          wc ← wc + 1;
          char ← XChar.LowerCase [char];
        END;

        IF (wordChars[wc] ← NStoCodedChar [handle, char]) = nonExistantCode THEN RETURN [FALSE];
        wc ← wc + increment;
      ENDLOOP;

      handle.nWordChars ← wc;
      RETURN [TRUE];
    END;

    ENDCASE => NULL;

    FOR c: CARDINAL IN [0..length) DO
    char: XString.Character = entry[c];

      IF char = apostrophe THEN wc ← wc - (increment - 1); -- point to shift if present
      IF (wordChars[wc] ← NStoCodedChar [handle, char]) = nonExistantCode THEN RETURN [FALSE];
      wc ← wc + increment;
    ENDLOOP;
    handle.nWordChars ← wc - (increment - 1);

    RETURN [TRUE];
  END;

  DictChar: PROCEDURE [handle: Handle, byte: Byte]
    RETURNS [CodedChar] = INLINE
  BEGIN
    RETURN [handle.offsetTable[byte]];
  END;

  GetByte: PROCEDURE [table: LONG POINTER, loc: Location] RETURNS [Byte] = INLINE
  BEGIN
  ptr: LONG POINTER TO WORD = table + (loc/2);

    RETURN [
      IF loc MOD 2 = 0 THEN
        Inline.HighByte [ptr↑]
      ELSE Inline.LowByte [ptr↑]];
  END;

  GetPointerTableEntry: PROCEDURE [table: Table, index: CARDINAL]
    RETURNS [Location] = INLINE
  BEGIN
      RETURN [LOOPHOLE [table, TwoByteTable][index]];
  END;

  NextCharLoc: PROCEDURE [handle: Handle,loc: Location] RETURNS [Location] = INLINE
  BEGIN
  base: Table = handle.fsm + (loc/2);
  locOffset: CARDINAL = (Inline.LowHalf [loc] MOD 2) + 1;
  offsetTable: LONG POINTER TO OffsetTable = handle.offsetTable;
  byte: Byte = base[locOffset];

    -- DecodePointer:
    SELECT handle.rangeTable[byte] FROM
    alphabetBase,
    firstFinalChar,
    altCounter,
    changeCounter,
    firstFinalAltChar,
    decrementCounter => -- simple character byte
      RETURN [loc+1];

    offsetLong => -- long pointer at current location
      RETURN ThreeByteLocation [
        byte1: offsetTable[byte],
        byte2: base[locOffset+1],
        byte3: base[locOffset+2]];

    offset2 => -- 2 byte indirect pointer
      RETURN GetPointerTableEntry [
        handle.table2,
        TwoByteIndex [byte1: offsetTable[byte], byte2: base[locOffset+1]]];

    offset1 => -- single byte indirect table
      RETURN GetPointerTableEntry [handle.table1, offsetTable[byte]];

    ENDCASE => RETURN [0]; -- **** can't get here ****
  END;

  NStoCodedChar: PROCEDURE [handle: Handle, ns: XString.Character]
    RETURNS [CodedChar] = INLINE
  BEGIN
    RETURN [
      IF ns IN [handle.firstNSChar..handle.lastNSChar]
      THEN handle.alphabetTable[ns - handle.firstNSChar]
      ELSE nonExistantCode];
  END;

  ThreeByteLocation: PROCEDURE [byte1, byte2, byte3: Byte] RETURNS [Location] = INLINE
```

```
  BEGIN
    RETURN [
      LOOPHOLE [
        FourBytes[highWord: [high: 0, low: byte1], lowWord: [high: byte2, low: byte3]]
      ]
    ];
  END;

  TwoByteIndex: PROCEDURE [byte1, byte2: Byte] RETURNS [CARDINAL] = INLINE
  BEGIN
    RETURN [LOOPHOLE [Inline.BytePair[high: byte1, low: byte2]]];
  END;

  -- Public Procedures

Close: PUBLIC PROCEDURE [handle: Handle] =
BEGIN
z: UNCOUNTED ZONE = handle.zone;

  [] ← Space.Unmap [handle.pointer];
  NSFile.Close [handle.file];
  z.FREE [@handle.altCharActionTable];
  z.FREE [@handle.rangeTable];
  z.FREE [@handle.offsetTable];
  z.FREE [@handle.wordChars];

  FOR c: CARDINAL IN [0..handle.nDictCodes) DO
    z.FREE [@handle.secondLevelTable[c]];
  ENDLOOP;
  z.FREE [@handle.secondLevelTable];

  FOR c: CARDINAL IN [0..handle.nStartStates) DO
    z.FREE [@handle.startStateTableTable[c]];
  ENDLOOP;
  z.FREE [@handle.startStateTableTable];

  z.FREE [@handle];

END;

GetCodeVersion: PUBLIC PROCEDURE RETURNS [CARDINAL] =
BEGIN RETURN [1] END;

 GetNEntries: PUBLIC PROCEDURE [handle: Handle] RETURNS [LONG CARDINAL] =
 BEGIN
 header: Header = handle.pointer;
   RETURN [InlineExtra.SwapWords [header.totalEowPCount]];
 END;

 EnumerateEntries: PUBLIC PROCEDURE [handle: Handle, entryProc: EntryProc] =
 BEGIN END;

 LookUpEntry: PUBLIC PROCEDURE [
   handle: Handle,
   entry: CharDescriptor,
   entryCase: Case,
   lookUpAsAbbreviation: BOOLEAN]
   RETURNS [found: BOOLEAN, lexiconCase: Case] =
 BEGIN
   IF ComputeWordChars [handle, entry, entryCase] THEN
   BEGIN
     -- try in the main dictionary
     SELECT entryCase FROM
     allUpper =>
     BEGIN
       -- try as lower case
       IF Lookup [handle: handle, firstChar: 1, skipShift: TRUE]
       THEN RETURN [TRUE, allLower];

       -- try capitalized
       IF Lookup [handle: handle, firstChar: 0, skipShift: TRUE]
       THEN RETURN [TRUE, capitalized];
     END;

     capitalized =>
       IF Lookup [handle: handle, firstChar: 1] THEN RETURN [TRUE, allLower];

     ENDCASE;

     IF Lookup [handle: handle] THEN RETURN [TRUE, entryCase];

     IF lookUpAsAbbreviation THEN
     BEGIN
       SELECT entryCase FROM
       allUpper =>
       BEGIN
         -- try as lower case
         IF Lookup [handle: handle, firstChar: 1, skipShift: TRUE, startState: 1]
         THEN RETURN [TRUE, allLower];

         -- try capitalized
         IF Lookup [handle: handle, firstChar: 0, skipShift: TRUE, startState: 1]
         THEN RETURN [TRUE, capitalized];
       END;

       capitalized =>
         IF Lookup [handle: handle, firstChar: 1, startState: 1]
```

```
          THEN RETURN [TRUE, allLower];

        ENDCASE;

        IF Lookup [handle: handle, startState: 1] THEN RETURN [TRUE, entryCase];
      END;

    END;
    RETURN [FALSE, allLower];
  END;

  Lookup: PROCEDURE [
    handle: Handle,
    firstChar: CARDINAL ← 0,
    startState: CARDINAL ← 0,
    skipShift: BOOLEAN ← FALSE]
  RETURNS [found: BOOLEAN] =
  BEGIN
  rangeTable: LONG POINTER TO RangeTable = handle.rangeTable;
  wordChars: WordChars = handle.wordChars;
  wchar: CodedChar ← wordChars[firstChar];
  shift: CodedChar = handle.shift;
  wc: CARDINAL ← firstChar + 1;
  nWordChars: CARDINAL = handle.nWordChars;
  loc: Location ← TRASH;
  fsm: Table = handle.fsm;
  range: Range ← TRASH;
  byte: Byte ← TRASH;

    NextWordChar: PROCEDURE RETURNS [wchar: CodedChar] = INLINE
    BEGIN
      DO
        wchar ← wordChars[wc];
        wc ← wc + 1;
        IF ~skipShift OR wchar # shift THEN RETURN;
      ENDLOOP;
    END;

    SELECT TRUE FROM
    startState = 0 AND wc < nWordChars =>
    BEGIN
    slt: LocationTable = handle.secondLevelTable[wchar];

      wchar ← NextWordChar[];
      loc ← slt[wchar];
    END;

    startState >= handle.nStartStates => RETURN [FALSE];

    ENDCASE => loc ← handle.startStateTableTable[startState][wchar];

    IF loc = 0 THEN RETURN [FALSE];

    DO
      range ← rangeTable[byte ← GetByte [fsm, loc]];
      SELECT TRUE FROM
      wchar = DictChar [handle, byte] =>
      BEGIN
        SELECT TRUE FROM
        wc >= nWordChars =>
          RETURN [ByteIsEndOfWord [range]];

        ByteHasNext [range] =>
        BEGIN
          wchar ← NextWordChar [];
          loc ← NextCharLoc [handle, loc];
        END;

        ENDCASE =>
          RETURN [FALSE];
      END;

      ByteHasAlt [range] =>
        loc ← AltCharLoc [handle, loc, byte];

      ENDCASE =>
        RETURN [FALSE];
    ENDLOOP;
  END;

  Open: PUBLIC PROCEDURE [
    fileRef: NSFile.Reference,
    zone: UNCOUNTED ZONE,
    activate: BOOLEAN] RETURNS [handle: Handle] =
  BEGIN
  file: NSFile.Handle ← NSFile.OpenByReference [fileRef];
  altCharActionTable: LONG POINTER TO AltCharActionTable ← zone.NEW [AltCharActionTable];
  rangeTable: LONG POINTER TO RangeTable ← zone.NEW [RangeTable];
  offsetTable: LONG POINTER TO OffsetTable ← zone.NEW [OffsetTable];
  header: Header ← TRASH;
  offset1: CARDINAL ← TRASH;
  offset2: CARDINAL ← TRASH;
  offsetLong: CARDINAL ← TRASH;
  alphabetBase: CARDINAL ← TRASH;
  firstFinalChar: CARDINAL ← TRASH;
  altCounter: CARDINAL ← TRASH;
  changeCounter: CARDINAL ← TRASH;
```

```
    firstFinalAltChar: CARDINAL ← TRASH;
    decrementCounter: CARDINAL ← TRASH;
    nDictCodes: CARDINAL ← TRASH;
    nStartStates: CARDINAL ← TRASH;
    startStateTableOffsets: TwoByteTable ← TRASH;
    sstt: LocationTableTable ← TRASH;
    slt: LocationTableTable ← TRASH;
    fsm: Table ← TRASH;

      handle ← zone.NEW [Object];
      handle.zone ← zone;
      handle.file ← file;

      handle.altCharActionTable ← altCharActionTable;
      handle.rangeTable ← rangeTable;
      handle.offsetTable ← offsetTable;

      header ← handle.pointer ← NSSegment.Map [
        origin: [file: file, base: 0, count: NSSegment.GetSizeInPages [file]],
        swapUnits: [unitary[]]
        !Space.InsufficientSpace => {
          zone.FREE[@handle];  -- return value = NIL
          zone.FREE[@offsetTable];
          zone.FREE[@rangeTable];
          zone.FREE[@altCharActionTable];
          NSFile.Close[file];
          GOTO OutOfVM}].pointer;

      handle.fsm ← handle.pointer + header.fsmBase/2;
      handle.alphabetTable ← handle.pointer + header.alphabetTable/2;
      handle.table1 ← handle.pointer + header.table1/2;
      handle.table2 ← handle.pointer + header.table2/2;

      offset1 ← header.offset1;
      offset2 ← header.offset2;
      offsetLong ← header.offsetLong;
      alphabetBase ← header.alphabetBase;
      firstFinalChar ← header.firstFinalChar;
      altCounter ← header.altCounter;
      changeCounter ← header.changeCounter;
      firstFinalAltChar ← header.firstFinalAltChar;
      decrementCounter ← header.decrementCounter;

      handle.firstNSChar ← header.firstNSChar;
      handle.lastNSChar ← header.firstNSChar + header.alphabetTableSize   1;
      handle.shift ← NStoCodedChar [handle, XCharSet0.Make [grave]];
      handle.apostrophe ← NStoCodedChar [handle, XCharSet0.Make [apostrophe]];

      handle.nWordChars ← 0;
      handle.maxWordChars ← header.longestMinimalPathLength;
      handle.wordChars ← zone.NEW [WordCharSequence[handle.maxWordChars*2]]; -- *2 for shifts

      FOR b: Byte IN [offset1..offset2) DO
        rangeTable[b] ← offset1;
        offsetTable[b] ← b - offset1;
        altCharActionTable[b] ← [counterIncrement: -1, locationIncrement: 1];
      ENDLOOP;

      FOR b: Byte IN [offset2..offsetLong) DO
       rangeTable[b] ← offset2;
       offsetTable[b] ← b - offset2;
       altCharActionTable[b] ← [counterIncrement: -1, locationIncrement: 2];
      ENDLOOP;

      IF header.hasEowPCounts THEN
       FOR b: Byte IN [offsetLong..alphabetBase) DO
         rangeTable[b] ← offsetLong;
         offsetTable[b] ← b - offsetLong;
         altCharActionTable[b] ← [counterIncrement: -1, locationIncrement: -1];
       ENDLOOP
      ELSE
      FOR b: Byte IN [offsetLong..alphabetBase) DO
       rangeTable[b] ← offsetLong;
       offsetTable[b] ← b - offsetLong;
       altCharActionTable[b] ← [counterIncrement: -1, locationIncrement: 3];
      ENDLOOP;

      FOR b: Byte IN [alphabetBase..firstFinalChar) DO
        rangeTable[b] ← alphabetBase;
        offsetTable[b] ← b - alphabetBase;
        altCharActionTable[b] ← [counterIncrement: 0, locationIncrement: 1];
      ENDLOOP;

      FOR b: Byte IN [firstFinalChar..altCounter) DO
        rangeTable[b] ← firstFinalChar;
        offsetTable[b] ← b - firstFinalChar;
        altCharActionTable[b] ← [counterIncrement: 0, locationIncrement: 1];
      ENDLOOP;

      FOR b: Byte IN [altCounter..changeCounter) DO
        rangeTable[b] ← altCounter;
        offsetTable[b] ← b - altCounter;
        altCharActionTable[b] ← [counterIncrement: 0, locationIncrement: 1];
      ENDLOOP;

      FOR b: Byte IN [changeCounter..firstFinalAltChar) DO
        rangeTable[b] ← changeCounter;
```

```
        offsetTable[b] ← b - changeCounter;
        altCharActionTable[b] ← [counterIncrement: +1, locationIncrement: 1];
      ENDLOOP;

      FOR b: Byte IN [firstFinalAltChar..decrementCounter) DO
        rangeTable[b] ← firstFinalAltChar;
        offsetTable[b] ← b - firstFinalAltChar;
        altCharActionTable[b] ← [counterIncrement: +1, locationIncrement: 1];
      ENDLOOP;

      FOR b: Byte IN [decrementCounter..256) DO
        rangeTable[b] ← decrementCounter;
        offsetTable[b] ← b - decrementCounter;
        altCharActionTable[b] ← [counterIncrement: -1, locationIncrement: 1];
      ENDLOOP;

      handle.nStartStates ← nStartStates ← header.nStartTables;
      handle.nDictCodes ← nDictCodes ← header.nDictCodes;
      sstt ← handle.startStateTableTable ← zone.NEW [LocationTableSequence[nStartStates]];
      startStateTableOffsets ← handle.pointer + header.startStateTableTableBase/2;

      -- Pre-compute the starting Location's for each character in each start state.
      FOR ss: CARDINAL IN [0..nStartStates) DO
      sst: LocationTable ← zone.NEW [LocationSequence[nDictCodes]];
      table: Table ← handle.pointer + startStateTableOffsets[ss]/2;
      offset: CARDINAL ← 0;

        sstt[ss] ← sst;
        FOR b: Byte IN [0..nDictCodes) DO
          sst[b] ← ThreeByteLocation [
            byte1: table[offset],
            byte2: table[offset + 1],
            byte3: table[offset + 2]];
          offset ← offset + 3;
        ENDLOOP;
      ENDLOOP;

      -- Pre-compute the second level starting Location's for start state 0
      -- Using this table we can quicly get to the Location for the second character

      handle.secondLevelTable ← slt ← zone.NEW [LocationTableSequence [nDictCodes]];
      fsm ← handle.fsm;

      FOR wc1: CodedChar IN [0..nDictCodes) DO
      lt: LocationTable ← zone.NEW [LocationSequence [nDictCodes]];
      loc: Location ← sstt[0][wc1]; -- FirstCharLoc [wc1]
      wc2: CodedChar ← TRASH;
      byte: Byte ← TRASH;

        slt[wc1] ← lt;
        loc ← NextCharLoc [handle, loc];

        FOR c: CodedChar IN [0..nDictCodes) DO
          lt [c] ← 0;
        ENDLOOP;

        DO
          wc2 ← DictChar [handle, byte ← GetByte [fsm, loc]];
          lt[wc2] ← loc;
          IF ~ByteHasAlt[rangeTable[byte]] THEN EXIT;
          loc ← AltCharLoc [handle, loc, byte];
        ENDLOOP;
      ENDLOOP;

    EXITS OutOfVM => NULL;
    END;  -- Open

END...

20-Jul-87       Walden  Add catch to Space.InsufficientSpace to NSSegment.Map call
11-Dec-87 17:37:06 - Mader - AltCharLoc must do byte ← base[locOffset] even if it doesn't go around the loop.
18-Dec-87 15:22:43 - Mader - Get word count from header.totalEowPCount.
21-Dec-87 16:37:40 - Mader - Do InlineExtra.SwapWords on above.
23-Dec-87 10:52:03 - Mader - Lookup: skip shift if followed by apostrophe; add NStoCodedChar and use it in ComputeWordChars
28-Dec-87 16:44:53 - Mader - ComputeWordChars: don't shift apostrophe; Lookup: always advance wc by 1 after apostrophe.
29-Dec-87 10:45:16 - Mader - Lookup: replace increment with skipShift BOOLEAN.
```

```
-- File: PARCLexiconPluginImpl.mesa - last edit:
-- Walden              21-Apr-87 12:54:34

-- Copyright (C) 1987 by Xerox Corporation. All rights reserved.

DIRECTORY
   -- temp:
   BWSZone USING [semiPermanent],
   CacheLexiconDefs USING [AddWord, CacheLexiconHandle, Close, LookUpWord, Open],
   LexiconDefs USING [
      CloseProc, CreateProc, EnumerateProc,
      GetNWordsProc, IsCurrentProc, LexiconHandle,
      LexiconObject, LexiconType, LookUpWordProc, OpenProc, SetOps],
   NSFile USING [GetReference],
   PARCLexicon USING [
     Case, Close, EnumerateEntries, GetCodeVersion, GetNEntries,
     Handle, LookUpEntry, Open],
   Runtime USING [NarrowFault],
   XChar USING [Character],
   XString USING [Map, MapCharProc];

PARCLexiconPluginImpl: PROGRAM
   IMPORTS BWSZone, CacheLexiconDefs, LexiconDefs, NSFile, PARCLexicon, Runtime, XString
   SHARES LexiconDefs = {

   Handle: TYPE = LONG POINTER TO Object;
   Object: TYPE = MACHINE DEPENDENT RECORD [
      lexicon: LexiconDefs.LexiconObject,
      zone: UNCOUNTED ZONE,
      cache: CacheLexiconDefs.CacheLexiconHandle,
      parcLexiconHandle: PARCLexicon.Handle];

   -- CONSTANTS
   parcLexicon: LexiconDefs.LexiconType = LexiconDefs.LexiconType.firstSpare;

   -- VARIABLES

   -- PROCS
   Close: LexiconDefs.CloseProc
      <<PROC [lexicon: LexiconHandle]>>  = {
      handle: Handle ← Narrow[lexicon];
      zone: UNCOUNTED ZONE = handle.zone;
      CacheLexiconDefs.Close[handle.cache];
      PARCLexicon.Close[handle.parcLexiconHandle];
      zone.FREE[@handle];
      };  --Close

   EnumerateWords: LexiconDefs.EnumerateProc
      <<PROC [lexicon: LexiconHandle, wordProc: WordProc]>>  = {
      handle: Handle = Narrow[lexicon];
      PARCLexicon.EnumerateEntries[handle.parcLexiconHandle, wordProc];
      };

   GetNWords: LexiconDefs.GetNWordsProc
      <<PROC [lexicon: LexiconHandle] RETURNS [nWordsCur, nWordsMax: LONG CARDINAL]>>
       = {
      handle: Handle = Narrow[lexicon];
      nWordsCur ← nWordsMax ← PARCLexicon.GetNEntries[handle.parcLexiconHandle];
      };  --GetNWords

   IsCurrent: LexiconDefs.IsCurrentProc = {RETURN[
     fileAttrs.type = parcLexicon AND
       fileAttrs.version = PARCLexicon.GetCodeVersion[]]};

  LookUpWord: LexiconDefs.LookUpWordProc = {
    <<PROC [lexicon: LexiconHandle, word: XS.Reader, flags: TxtScanDefs.ReadonlyWordFlags] RETURNS [found: BOOL]>>
    handle: Handle = Narrow[lexicon];
    nCharsMax: CARDINAL = 50;
    i: CARDINAL ← 0;
    charArray: ARRAY [0..nCharsMax) OF XChar.Character;
    useCache: BOOL = ~forCorrections;
    lexiconCase: PARCLexicon.Case;

    AddChar: XString.MapCharProc = {
      charArray[i] ← c;
      i ← i+1;
      RETURN [stop: (i = nCharsMax)];
      };

    -- this guy knows that the cache lexicon only contains lower-case, or equivalent, words
    IF useCache AND (wordFlags.case # mixedLowerUpper) AND CacheLexiconDefs.LookUpWord[
      handle.cache, word, wordFlags].found THEN RETURN [TRUE];

    [] ← XString.Map[word, AddChar];

    [found, lexiconCase] ← PARCLexicon.LookUpEntry[
      handle: handle.parcLexiconHandle, entry: DESCRIPTOR[charArray.BASE, i],
      entryCase: (SELECT wordFlags.case FROM
        allLower => allLower,
        firstCharUpperOnly => capitalized,
        mixedLowerUpper => mixed,
        ENDCASE => allUpper),
      lookUpAsAbbreviation: useCache];

    IF found THEN {
      IF useCache AND (lexiconCase = allLower) THEN  -- only add lower-case words to cache
        [] ← CacheLexiconDefs.AddWord [handle.cache, word, wordFlags];
```

```
      RETURN [TRUE];
      }
    ELSE RETURN [FALSE];
    };  -- LookUpWord

  Open: LexiconDefs.OpenProc
    <<PROC [lexiconFile: NSFile.Handle, readonly: BOOL, z: UNCOUNTED ZONE ← NIL,
    session: NSFile.Session]
    RETURNS [lexicon: LexiconHandle]>>
      = {
      handle: Handle;
      parcLexiconHandle: PARCLexicon.Handle;

      IF z = NIL THEN z ← BWSZone.semiPermanent;

      parcLexiconHandle ← PARCLexicon.Open[
        fileRef: NSFile.GetReference[lexiconFile, session], zone: z, activate: TRUE];

      IF parcLexiconHandle = NIL THEN RETURN [NIL];

      handle ← z.NEW[Object ← [
        lexicon: [
          type: parcLexicon, zone: z, add: NIL, close: Close, delete: NIL,
          enumerate: EnumerateWords, getNWords: GetNWords, lookUp: LookUpWord],
        zone: z,
        cache: CacheLexiconDefs.Open [nEntriesMax: 500, z: z],
        parcLexiconHandle: parcLexiconHandle]];

      RETURN [Widen[handle]];
      };  --Open

    -- PRIVATE PROCS
    Narrow: PROC [l: LexiconDefs.LexiconHandle] RETURNS [Handle] = INLINE {
      IF l.type # parcLexicon THEN ERROR Runtime.NarrowFault;
      RETURN[LOOPHOLE[l]]};

    Widen: PROC [handle: Handle] RETURNS [LexiconDefs.LexiconHandle] = INLINE {
      RETURN[@handle.lexicon]};

    -- MAINLINE
    LexiconDefs.SetOps[
      type: parcLexicon, ops: [create: NIL, isCurrent: IsCurrent, open: Open]];

    }...

LOG
 2-Apr-87  9:07:25 - Walden - Created
```

```
-- File: SpellingCheckerDefs.mesa - last edit:
-- Walden.ES           26-Mar-87 16:22:16
-- Maybury.ES           9-Apr-86 10:35:32
-- Marks.ES             9-Mar-85 14:56:30

-- Copyright (C) 1986, 1987 by Xerox Corporation. All rights reserved.


DIRECTORY
  AreaDefs USING [Rs],
  DocumentDefs USING [Handle],
  FormWindow USING [TextHintsProc],
  LexiconDefs USING [LexiconHandle, Lexicons],
  NSAssignedTypes USING [firstStarType],
  NSFile USING [Handle, nullHandle, nullSession, Reference, Session, Type],
  RgnDefs USING [Sc],
  SchemaDefs USING [Lschema, lschemaNil],
  StarWindowShell USING [Handle, nullHandle],
  System USING [GreenwichMeanTime],
  ToolUtilitiesDefs USING [toolPstype],
  TxtDefs USING [Block],
  TxtScanDefs USING [EnumScope, ReadonlyWordFlags, WordFlagsObject, WordScanCtxtObject],
  Window USING [Handle],
  XString USING [nullReaderBody, nullWriterBody, Reader, ReaderBody, WriterBody];

SpellingCheckerDefs: DEFINITIONS = {
  OPEN RgnDefs, SchemaDefs, TxtDefs, TxtScanDefs;

  AddOrDelete: TYPE = {add, delete};

  CheckCtxt: TYPE = LONG POINTER TO CheckCtxtObject;
  CheckCtxtObject: TYPE = RECORD[  --global context
    zone: UNCOUNTED ZONE,

    fw: Window.Handle ← NIL,      -- is the following the same?  I think it is.
    sws: StarWindowShell.Handle ← StarWindowShell.nullHandle,
    windowOpen: BOOLEAN ← FALSE,
    doc: DocumentDefs.Handle ← NIL,
    body: Lschema ← lschemaNil,
    nWords: CARDINAL ← 0,
    scratchBlock: TxtDefs.Block,

    -- miStartContinue: MiSchemaDefs.Mihandle ← TRASH,
    continueShowing: BOOL ← FALSE,

    scScanCtxtObject: SCScanCtxtObject ← TRASH,

    --must be initialized to be long enough to hold longest desireable list of alternatives
    alternatives: XString.WriterBody ← XString.nullWriterBody,
    rsAlternatives: AreaDefs.Rs ← [0, 0],

    cGenerateAlternatives: CONDITION ← [0],
    cDoneGeneratingAlternatives: CONDITION ← [0],
    generatingAlternatives,
    abortAlternatives: BOOL ← FALSE,
    pAlternativesGeneration: PROCESS ← TRASH,

    --alternativesWord must be initialized to be long enough to hold longest word, plus 3 bytes for normalization
    alternativesWord: XString.WriterBody ← XString.nullWriterBody,  -- in normalized form
    alternativesWordFlags: WordFlagsObject ← TRASH,
    alternativesLookUp: LexiconDefs.Lexicons ← NIL,

    correctionLexicon: NSFile.Handle ← TRASH,  --for autocorrect
    correctionList: LexiconDefs.LexiconHandle ← TRASH, --for autocorrect
    lastCorrectionValue: XString.ReaderBody ← XString.nullReaderBody,
    ignoreLexicon: NSFile.Handle ← TRASH,
    ignoreList: LexiconDefs.LexiconHandle ← TRASH,

    userLexiconCtnr: NSFile.Handle ← NSFile.nullHandle,
    lexicons: LptLexiconDataList ← NIL,
    listCreationTime: System.GreenwichMeanTime ← [0],
    lastUserSession: NSFile.Session ← NSFile.nullSession,
    refreshActiveLists,
    invalidAltList: BOOL ← TRUE,

    nLookUp, nEdit: CARDINAL ← 0,
    lexiconsLookUp,
    lexiconsEdit: LptLexiconList ← NIL
    ];

  LexiconData: TYPE = RECORD [
    ref: NSFile.Reference,
    file: NSFile.Handle,
    name: XString.Reader,
    lexicon: LexiconDefs.LexiconHandle,
    nWords: LONG CARDINAL,
    systemLex,
    edit,
    lookUp: BOOL
    ];

  LexiconDataList: TYPE = RECORD [
    seq: SEQUENCE nLexicons: CARDINAL OF LexiconData];

  currentVersion: CARDINAL = 1;
  DisplayAttrsType: TYPE = {lexicon, wn};
  DisplayAttrs: TYPE = MACHINE DEPENDENT RECORD[
```

```
    version(0): CARDINAL ← currentVersion,
    var(1): SELECT type(1): DisplayAttrsType FROM
      lexicon => [
        selectedForLookUp (2: 15..15): BOOL,
        selectedForEdit (2: 14..14): BOOL,
        unused (2:0..13): PACKED ARRAY [0..13] OF [0..1] ← ALL[0]],
      wn => [
        scWn(2): RgnDefs.Sc,
        rsBody(4): AreaDefs.Rs,
        scope(6): TxtScanDefs.EnumScope,
        autoCorrect(7: 15..15): BOOL,
        includeAnchoredFrames(7: 14..14): BOOL,
        sysLexSelectedForLookUp(7: 13..13): BOOL,
        unused (7: 0..12): PACKED ARRAY [0..12] OF [0..1] ← ALL[0]],
      ENDCASE
    ];

  Items: TYPE = {scope, checkFrames, autoCorrect, misspelling, correction};

  LexiconDisplayAttrs: TYPE = LONG POINTER TO DisplayAttrs.lexicon;
  WnDisplayAttrs: TYPE = LONG POINTER TO DisplayAttrs.wn;

  LexiconList: TYPE = RECORD [
    seq: SEQUENCE COMPUTED CARDINAL OF LexiconDefs.LexiconHandle];

  LptLexiconData: TYPE = LONG POINTER TO LexiconData;
  LptLexiconDataList: TYPE = LONG POINTER TO LexiconDataList;
  LptLexiconList: TYPE = LONG POINTER TO LexiconList;

  SCScanCtxt: TYPE = LONG POINTER TO SCScanCtxtObject;
  SCScanCtxtObject: TYPE = MACHINE DEPENDENT RECORD [
    wordScanCtxtObject(0): TxtScanDefs.WordScanCtxtObject,
    --scScanCtxt data
    normalizedWord(28): XString.WriterBody,
    nProcessed(37): CARDINAL ← 0,
    autoCorrect(38): BOOL ← FALSE
    ];

--CONSTANTS
  -- *** HACK FIX.  NEED TO HAVE A UNIQUE StarAttributeType.
  displayAttrType: NSFile.Type = NSAssignedTypes.firstStarType + 39;
  nBytesLongestWord: CARDINAL = 40;  --for scanning and alternatives source word
  nBytesLongestCorrection: CARDINAL = 100;  --for Correction text when used with Correct cmd

-- range choice parm constants
  allText: CARDINAL = 0;
  remainingText: CARDINAL = 1;
  selectedText: CARDINAL = 2;

  checkCtxt: CheckCtxt;

  firstLexiconPstype: CARDINAL = ToolUtilitiesDefs.toolPstype+1;
-- PROCEDURES

  AddMenuCommands: PROC [sws: StarWindowShell.Handle, itemData: LONG UNSPECIFIED];

  AddOrDeleteWords: PROC [addOrDelete: AddOrDelete];

  ChoiceFromScope: PROC [scope: TxtScanDefs.EnumScope] RETURNS [choice: CARDINAL] = INLINE {
    -- if choice parm arrangement changes, this won't be valid
    RETURN [(SELECT scope FROM
      all => allText,
      remainder => remainingText,
      selection => selectedText,
      ENDCASE => allText)]};

  ClearHostDocCs: PROC;

  ClearFdbkParms: PROC;

  CloseUtil: PROC [fw: Window.Handle, itemData: LONG UNSPECIFIED];

  CloseLexicons: PROC RETURNS [sysLexSelectedForLookUp: BOOL];

  ContinueChecking: PROC [lookUp: LexiconDefs.Lexicons];

  CreateAlternativesProcess: PROC RETURNS [PROCESS];

  CreateLexiconLists: PROC [systemLexSelectedForLookUp: BOOL, c: CheckCtxt];

  CreatePairList: PROC [nPairsMax: CARDINAL, z: UNCOUNTED ZONE] RETURNS [LexiconDefs.LexiconHandle];

  AddPopupMenu: PUBLIC PROC [shell: StarWindowShell.Handle];

  DestroyAlternativesProcess: PROC [p: PROCESS];

  DestroyPairList: PROC [list: LexiconDefs.LexiconHandle];

  GetActiveLexicons: PROC RETURNS [lookUp, edit: LexiconDefs.Lexicons];

  -- word should be in lexicon "normalized" form
  LookUpInPairList: PROC [list: LexiconDefs.LexiconHandle, word: XString.Reader, wordFlags: TxtScanDefs.ReadonlyWordFlags] RETURNS
  [found: BOOL, associate: XString.ReaderBody, associateFlags: TxtScanDefs.ReadonlyWordFlags];

  MakeDoc: PROC [lexicon: LexiconDefs.LexiconHandle, name: XString.Reader] RETURNS [ok: BOOL];

  --if none, RETURNS [nullHandle, System.gmtEpoch, default attrs];
```

```
    OpenUserLexiconCtnr: PROC RETURNS [file: NSFile.Handle, lastModifiedOn: System.GreenwichMeanTime, dispAttrs: DisplayAttrs.wn];

    PopUpMenuMakeStringProc: FormWindow.TextHintsProc;

    ReplaceContinueWithStart: PROC;

    ReplaceStartWithContinue: PROC;

    ScanBatchCheckAndAdd: PROC [lookUp, edit: LexiconDefs.Lexicons];

    ScopeFromChoice: PROC [scopeChoice: CARDINAL] RETURNS [TxtScanDefs.EnumScope] = INLINE {
      RETURN[
        SELECT scopeChoice FROM
          allText => all,
          remainingText => remainder,
          selectedText => selection,
          ENDCASE => all]};

    SetDocContext: PROC [doc: DocumentDefs.Handle];

    SetDocNotEdited: PROC [doc: DocumentDefs.Handle];

    -- word should be in lexicon "normalized" form
    SetPair: PROC [list: LexiconDefs.LexiconHandle, word: XString.Reader, wordFlags: TxtScanDefs.ReadonlyWordFlags, associate:
    XString.Reader, associateFlags: TxtScanDefs.ReadonlyWordFlags] RETURNS [ok: BOOL ← TRUE];  --not ok if list is full

    StartAlternativesGeneration: PROC [lookUp: LexiconDefs.Lexicons, word: XString.Reader, wordFlags: TxtScanDefs.ReadonlyWordFlags];

    StopAlternativesGeneration: PROC;

    StartChecking: PROC [lookUp: LexiconDefs.Lexicons];

    UpdateWordCountDisplay: PROC [checkCtxt: CheckCtxt];

    UserEditedDoc: PROC [doc: DocumentDefs.Handle] RETURNS [BOOL];
    }. -- of SpellingCheckerDefs

LOG (date - person - action)
 5-Jun-84 16:34:14 - Walden - OS5 release version
16-Jan-85 11:13:58 - Marks - Removed reference to StarAttributeTypeForgot1Defs by assigning displayAttrType the value
NSAssignedTypes.firstStarType + 39 rather than StarAttributeTypeForgot1Defs.spellingChecker2.  This may need to be fixed at a later
time.
 9-Mar-85 10:15:25 - Marks - Update for OS6.
 9-Apr-86  9:59:05 - Maybury - Changed SCScanCtxtObject to include WordScanCtxtObject as a component (vs. being a replica).
```

```
-- File: SpellingCheckerMenuPack.mesa - last edit:
-- Bartlett:OSBU South:Xerox  2-Dec-87 13:28:08
-- Marks.ES            16-Nov-87 18:12:36
-- Walden              24-Jul-87 16:37:57
-- Lewis:OSBU South:Xerox 11-Dec-86 19:50:30
-- Maybury.ES           9-Apr-86 13:13:47
-- Goodell.ES          15-Mar-85 15:08:32

-- Copyright (C) 1986, 1987 by Xerox Corporation. All rights reserved.


DIRECTORY
  Attention USING [Post, PostAndConfirm],
  BlockFriendsDefs USING [InRedliningMode],
  CharDefs USING [Char, Chset, Code, Roman],
  Cursor USING [Set],
  DocumentDefs USING [GetAccess, SetEdited],
  DocUtilDefs USING [FreeSystemDoc, LockSystemDoc],
  Environment USING [Byte],
  FormWindow USING [FreeTextHintsProc, GetBooleanItemValue, GetTextItemValue,
    TextHintsProc],
  HashTableLexiconDefs USING [fileType],
  LexiconDefs USING [
    AddWord, GetNormalizedWord, LexiconHandle, LexiconIsFull, Lexicons,
    LookUpWord, OpenLexicon],
  MenuData USING [AddItem, CreateItem, ItemHandle, MenuHandle, MenuProc, SwapItem],
  NSFile USING [
    AttributesRecord, ClearAttributes, Close, GetAttributes, GetType, Handle,
    OpenByReference, Reference],
  Process USING [
    Abort, DisableTimeout, EnableAborts, InitializeCondition,
    priorityBackground, SetPriority],
  ProductFactoring USING [Enabled],
  SchemaDefs USING [Lschema],
  Selection USING [CanYouConvert, Convert, Enumerate, EnumerationProc, Free, Value],
  Space USING [ScratchMap],
  SpellingCheckerDefs USING [
    AddOrDeleteWords, checkCtxt, CheckCtxt, ClearHostDocCs, CloseUtil,
    ContinueChecking, GetActiveLexicons, Items, LexiconData, MakeDoc,
    nBytesLongestWord, ScanBatchCheckAndAdd, SCScanCtxt, SetDocContext,
    SetDocNotEdited, SetPair, StartChecking, UserEditedDoc],

  SpellingCheckerMessageDefs USING [
    GetHandle, keySCClose, keySCStart, keySCContinue,
    keySCAddWords, keySCIgnoreWord, keySCCorrect, keySCBatchCheckAndAdd,
    keySCDeleteWord, keySCMakeDocument, keySCMsgMakeDocInProgress,
    keySCMsgMakeDocComplete, keySCMsgUnknownError,
    keySCMsgNoEditLexiconsSpecified, keySCMsgUserCancelled,
    keySCMsgNoLookUpLexiconsSpecified, keySCMsgConfirmCorrectionWasntApplied,
    keySCMsgNoWordToCorrect, keySCMsgUserEditedDocCantCorrect,
    keySCMsgCorrectionListIsFull, keySCMsgIgnoreListIsFull,
    keySCMsgSelectALexicon, keySCMsgAlternativesWordTooLong,
    keySCNoAlternatives, keySCMsgCantOpenLexicon, keySCMsgReadOnlyAccess],

  StarPFOptions USING [starSpelling],
  StarWindowShell USING [
    EnumeratePopupMenus, GetRegularCommands, GetZone, Handle, MenuEnumProc],
  TextUtilDefs USING [
    AppendNumber, DeleteChar, InsertChar, Rdr, ReplaceChar,
    ResetWriter, TextSegmentFromXString, WLength],
  TIP USING [UserAbort],
  TxtBlockDefs USING [BumpBlockaddr, Order, SetBlockaddr],
  TxtDefs USING [TextSegment, textSegmentNil],
  TxtEditDefs USING [AqTxtCtxt, BeginEdit, DestroyTextSegment, EndEdit, ReplaceWithString],
  TxtEditExtraIDefs USING [ReplaceWithStringForRedlining],
  TxtScanDefs USING [
    GetWordFlags, ReadonlyWordFlags, ScanWords, WordFlags, WordFlagsObject,
    WordProc],
  XChar USING [Character, CharRep, Code, Make, not, Set],
  XCharProps USING [Case, GetLetterRange, IsLetter, IsUpperCase],
  XCharSet0 USING [Codes0, Make],
  XCharSets USING [Sets],
  XFormat USING [DecimalFormat],
  XMessage USING [Get, Handle],
  XString USING [
    AppendChar, AppendReader, ByteLength, Bytes, Character, CopyReader, Equal,
    Equivalent, First, FreeReaderBytes, FreeWriterBytes, FromNSString,
    InsufficientRoom, NewWriterBody, NthCharacter,
    nullReaderBody, Reader, ReaderBody, ReaderFromWriter,
    ScanForCharacter, Writer, WriterBody, WriterBodyFromSTRING];

SpellingCheckerMenuPack: MONITOR
  IMPORTS
    Attention, BlockFriendsDefs, CharDefs, Cursor, DocumentDefs, DocUtilDefs,
    FormWindow, LexiconDefs, MenuData, NSFile, Process,
    ProductFactoring, Selection, Space, SpellingCheckerDefs,
    SpellingCheckerMessageDefs, StarWindowShell, TIP,
    TxtBlockDefs, TxtEditDefs, TxtEditExtraIDefs, TxtScanDefs, TextUtilDefs,
    XChar, XCharProps, XCharSet0, XMessage, XString
  EXPORTS SpellingCheckerDefs
  SHARES SchemaDefs, XString =
  BEGIN
  OPEN FW: FormWindow, SpellingCheckerDefs, SpellingCheckerMessageDefs, TextUtilDefs,
    XM: XMessage, XS: XString;

  Bug: SIGNAL [Bugtype] = CODE;
  Bugtype: TYPE = {impossible, badValue, error};
```

```
ChangeStatus: TYPE = {changed, caseChanged, noChange};

<< CONSTANTS >>

h: XMessage.Handle = SpellingCheckerMessageDefs.GetHandle[];
lengthAlternativesList: CARDINAL = 1024;

<< VARIABLES >>

-- the 50 ReaderBody's will fit on one page.  By allocating them using Space.ScratchMap they will be out of the global frame.  If the
range of altWordArray is ever changed, the number of pages requested must be checked.
AqWordArray: TYPE = ARRAY [0..50) OF XS.ReaderBody;
AltWordArray: TYPE = LONG POINTER TO AqWordArray;
altWordArray: AltWordArray ← Space.ScratchMap[1];
startMenuItem, continueMenuItem: MenuData.ItemHandle ← NIL;
myZone: UNCOUNTED ZONE ← NIL;

-- PROCS

-- PUBLIC

AddPopupMenu: PUBLIC PROC [shell: StarWindowShell.Handle] = {
   OPEN MenuData;
   batchCheckAndAdd: XS.ReaderBody ← XM.Get[h, keySCBatchCheckAndAdd];
   delete: XS.ReaderBody ← XM.Get[h, keySCDeleteWord];
   makeDocument: XS.ReaderBody ← XM.Get[h, keySCMakeDocument];
   z: UNCOUNTED ZONE ← StarWindowShell.GetZone[shell];
   menuEnumProc: StarWindowShell.MenuEnumProc = {
      -- PROC [menu: MenuData.Handle] RETURNS [stop: BOOL ← FALSE];
      AddItem[menu, CreateItem[zone: z, name: @batchCheckAndAdd, proc: BatchCheckAndAdd]];
      AddItem[menu, CreateItem[zone: z, name: @delete, proc: DeleteWords]];
      AddItem[menu, CreateItem[zone: z, name: @makeDocument, proc: MakeDocument]];
      stop ← TRUE;
      };
   -- NOTE: This is an icky way to add items to the aux menu, but BWS won't support a direct mechanism to do this.  Hopefully, one day
   soon, they will see the error in their ways and amend this.  (Marks, 3/85)
   StarWindowShell.EnumeratePopupMenus[shell, menuEnumProc];
   };  -- AddPopupMenu

AddMenuCommands: PUBLIC PROC [sws: StarWindowShell.Handle, itemData: LONG UNSPECIFIED] = {
   OPEN MenuData;
   menuHandle: MenuHandle ← StarWindowShell.GetRegularCommands[sws];
   close: XS.ReaderBody ← XM.Get[h, keySCClose];
   continue: XS.ReaderBody ← XM.Get[h, keySCContinue];
   start: XS.ReaderBody ← XM.Get[h, keySCStart];
   addWords: XS.ReaderBody ← XM.Get[h, keySCAddWords];
   ignoreWord: XS.ReaderBody ← XM.Get[h, keySCIgnoreWord];
   correct: XS.ReaderBody ← XM.Get[h, keySCCorrect];
   z: UNCOUNTED ZONE ← StarWindowShell.GetZone[sws];

   startMenuItem ← CreateItem[zone: z, name: @start, proc: Start, itemData: itemData];
   continueMenuItem ← CreateItem[zone: z, name: @continue, proc: Continue, itemData: itemData];

   -- non-standard property sheet commands
   AddItem[menuHandle, CreateItem[zone: z, name: @close, proc: Close, itemData: itemData]];
   AddItem[menuHandle, startMenuItem];
   AddItem[menuHandle, CreateItem[zone: z, name: @addWords, proc: AddWords, itemData: itemData]];
   AddItem[menuHandle, CreateItem[zone: z, name: @ignoreWord, proc: IgnoreWord, itemData: itemData]];
   AddItem[menuHandle, CreateItem[zone: z, name: @correct, proc: Correct, itemData: itemData]];

   };  --AddMenuCommands

CreateAlternativesProcess: PUBLIC PROC RETURNS [PROCESS] = {
   RETURN[FORK GenerateAlternativesInBkgd]; };


DestroyAlternativesProcess: PUBLIC PROC [p: PROCESS] = {
   StopAlternativesGeneration[];
   Process.Abort[p];
   JOIN p; };


-- PRIVATE


AddWords: MenuData.MenuProc = {SpellingCheckerDefs.AddOrDeleteWords[add]};


BatchCheckAndAdd: MenuData.MenuProc = {
   edit, lookUp: LexiconDefs.Lexicons;

   [lookUp: lookUp, edit: edit] ← GetActiveLexicons[];

   IF edit = NIL THEN  { --lookUp=NIL is ok, means don't look up
      msgNoEditLexiconsSpecified: XS.ReaderBody ← XM.Get[h, keySCMsgNoEditLexiconsSpecified];
      Attention.Post[@msgNoEditLexiconsSpecified]}
   ELSE ScanBatchCheckAndAdd[lookUp: lookUp, edit: edit];
   };  -- BatchCheckAndAdd

Close: MenuData.MenuProc = {
   SpellingCheckerDefs.CloseUtil[window, itemData];
   };

Continue: MenuData.MenuProc = {
```

```
    checkCtxt: CheckCtxt ← LOOPHOLE[itemData, CheckCtxt];
    lookUp: LexiconDefs.Lexicons ← GetActiveLexicons[].lookUp;

    IF TIP.UserAbort[checkCtxt.sws] THEN {
       SetDocContext[NIL];  -- pops enumerators, replaces Continue
         {wb: XS.WriterBody ← XS.NewWriterBody[40, checkCtxt.zone];
          r: XS.Reader;
          msgUserCancelled: XS.ReaderBody ← XM.Get[h, keySCMsgUserCancelled];
          XS.AppendReader[@wb, @msgUserCancelled];
          TextUtilDefs.AppendNumber[
             @wb, LONG[INTEGER[checkCtxt.scScanCtxtObject.nProcessed]],
             XFormat.DecimalFormat];
          r ← XS.ReaderFromWriter[@wb];
          Attention.Post[r];
          XS.FreeWriterBytes[@wb];
          };
       RETURN;
       };

    IF lookUp = NIL THEN {
       msgNoLookUpLexiconsSpecified: XS.ReaderBody ← XM.Get[h, keySCMsgNoLookUpLexiconsSpecified];
       Attention.Post[@msgNoLookUpLexiconsSpecified];  }
    ELSE {
       msgConfirmCorrectionWasntApplied: XS.ReaderBody ← XM.Get[
          h, keySCMsgConfirmCorrectionWasntApplied];
       IF checkCtxt.scScanCtxtObject.wordScanCtxtObject.scanCtxtObject.ec # NIL  --in the middle of checking words
          AND CorrectionParmChanged[].status # noChange   --user edited the parm since we first presented a misspelling
          AND ~Attention.PostAndConfirm[@msgConfirmCorrectionWasntApplied].confirmed
          THEN RETURN;

       SpellingCheckerDefs.ContinueChecking[lookUp];
       };
    };  -- Continue


  --is the current value of the correction parm different from misspelling parm (latter value is in 'word' string)
  -- font changes don't count, case changes do...
  Correct: MenuData.MenuProc = {
     c: CheckCtxt ← LOOPHOLE[itemData, CheckCtxt];
     aqTxtCtxt: TxtEditDefs.AqTxtCtxt;
     bumpedOut: BOOL;
     scSC: SCScanCtxt = @c.scScanCtxtObject;
     correctionR: XS.ReaderBody;

     IF scSC.wordScanCtxtObject.ts = TxtDefs.textSegmentNil THEN {
        msgNoWordToCorrect: XS.ReaderBody ← XM.Get[h, keySCMsgNoWordToCorrect];
        Attention.Post[@msgNoWordToCorrect];
        RETURN; };

     IF UserEditedDoc[c.doc] THEN {
        msgUserEditedDocCantCorrect: XS.ReaderBody ← XM.Get[h, keySCMsgUserEditedDocCantCorrect];
        SetDocContext[NIL];  -- pops enumerators, replaces Continue
        Attention.Post[@msgUserEditedDocCantCorrect];
        RETURN;
        };

     IF DocumentDefs.GetAccess[c.doc] = read THEN {
        msgReadOnlyAccess: XS.ReaderBody ← XM.Get[h, keySCMsgReadOnlyAccess];
        SetDocContext[NIL];
        Attention.Post[@msgReadOnlyAccess];
        RETURN;
        };


     correctionR ← FormWindow.GetTextItemValue[
        c.fw, Items.correction.ORD, c.zone];


     --don't apply empty/unchanged correction
     IF (XS.ByteLength[@correctionR] = 0)
        OR (CorrectionChanged[@correctionR].status = noChange) THEN {
        XS.FreeReaderBytes[@correctionR, c.zone]; RETURN; };

     <<If ts.blockaddrLast points to same char as range.blockaddrLast THEN bump range.blockaddrLast +1, bump -1 after replacing.  We
     don't have to worry about range.blockaddrFirst though, because we are setting it to the first char of the new word anyway.  Also
     note that only ts.blockaddrLast can coincide with range.blockaddrLast>>

     WITH e: scSC.wordScanCtxtObject.scanCtxtObject.ec SELECT FROM
        text =>
           IF TxtBlockDefs.Order[scSC.wordScanCtxtObject.ts.blockaddrLast, e.range.blockaddrLast] =
              equal THEN {
              [] ← TxtBlockDefs.BumpBlockaddr[e.range.blockaddrLast, 1, TRUE];
              bumpedOut ← TRUE;
              }
           ELSE bumpedOut ← FALSE;
        ENDCASE;

     ClearHostDocCs[];

     TxtEditDefs.BeginEdit[@aqTxtCtxt];
     IF BlockFriendsDefs.InRedliningMode[] THEN
       TxtEditExtraDefs.ReplaceWithStringForRedlining[
       @aqTxtCtxt, @scSC.wordScanCtxtObject.ts, @correctionR]
     ELSE TxtEditDefs.ReplaceWithString[
       @aqTxtCtxt, @scSC.wordScanCtxtObject.ts, @correctionR];
     TxtEditDefs.EndEdit[@aqTxtCtxt];
```

```
    -- For old style documents need to set edited bit so will update time-stamp.
    [] ← DocumentDefs.SetEdited[c.doc];
    SetDocNotEdited[c.doc];

    IF c.lastCorrectionValue  # XS.nullReaderBody THEN
       XS.FreeReaderBytes[@c.lastCorrectionValue, c.zone];
    c.lastCorrectionValue ← correctionR;

    WITH e: scSC.wordScanCtxtObject.scanCtxtObject.ec SELECT FROM
       text => {
          TxtBlockDefs.SetBlockaddr[
             blockaddr: e.range.blockaddrFirst,
             blockaddrSource: scSC.wordScanCtxtObject.ts.blockaddrFirst];
          IF bumpedOut THEN
             [] ← TxtBlockDefs.BumpBlockaddr[e.range.blockaddrLast, -1, FALSE];
          e.endOfRange ← FALSE;
          };
       ENDCASE;

    --note: repeated invocation of Correct will simply update the association with the original word; thus the pair-list will always
    reflect the most recent correction of the ORIGINAL
    IF FW.GetBooleanItemValue[
       c.fw, Items.autoCorrect.ORD] THEN {
       correctionFlags: TxtScanDefs.WordFlagsObject;
       TxtScanDefs.GetWordFlags[@correctionR, @correctionFlags];
       IF ~SetPair[
          list: c.correctionList, word: Rdr[@scSC.normalizedWord],
          wordFlags: @scSC.wordScanCtxtObject.wordFlags, associate: @correctionR,
          associateFlags: @correctionFlags].ok
       THEN {
          msgCorrectionListIsFull: XS.ReaderBody ← XM.Get[h, keySCMsgCorrectionListIsFull];
          Attention.Post[@msgCorrectionListIsFull];
          RETURN; };
       };

    Continue[window, menu, itemData];
    };  -- Correct


 << conventions for the "Correction" text parm and checkCtxt.lastCorrectionValue: to detect if the Correction parm has been changed
  since initially set or since the last "Correct" operation
parm initially empty, checkCtxt.lastCorrectionValue = nullReaderBody
when a word is found in interactive mode, it is copied into the parm, and checkCtxt.lastCorrectionValue is set to the string (ie,
whenever a new value is set into the Correction parm programmatically the same value is remembered in checkCtxt.lastCorrectionValue)

when a source text word is changed via Correct, the current value in checkCtxt.lastCorrectionValue is destroyed, and it is then set to
the newly applied value

if the user has edited the parm since the last programmatic setting or the last Correct, the parm value will not match the remembered
value

so, lastCorrectionValue remembers the last value in the parm (which matches the state of the source text), and whenever that is
different from the current textparm value, it can be inferred that the parm has been edited.  Font changes don't count, case changes
do  [XS.Equal considers case])

if there is a value for sc.word, there should always also be one (maybe 0 length) for c.lastCorrectionValue
>>

 CorrectionChanged: PROC [currentValue: XS.Reader]
    RETURNS [status: ChangeStatus] = {
    OPEN c: checkCtxt;
    RETURN[
       SELECT TRUE FROM
          c.lastCorrectionValue = XS.nullReaderBody =>
             IF XS.ByteLength[currentValue] > 0 THEN changed ELSE noChange,
          XS.Equal[currentValue, @c.lastCorrectionValue] => noChange,
          XS.Equivalent[currentValue, @c.lastCorrectionValue] => caseChanged,
          ENDCASE => changed];
    }; -- CorrectionChanged


 CorrectionParmChanged: PROC RETURNS [status: ChangeStatus] = {
    correctionR: XS.ReaderBody ← FormWindow.GetTextItemValue[
       checkCtxt.fw, Items.correction.ORD, myZone];
    -- font style changes don't count
    status ← CorrectionChanged[@correctionR];
    }; -- CorrectionParmChanged


 DeleteWords: MenuData.MenuProc = {SpellingCheckerDefs.AddOrDeleteWords[delete]};


 IgnoreWord: MenuData.MenuProc = {
    checkCtxt: CheckCtxt ← LOOPHOLE[itemData, CheckCtxt];
    {OPEN scSC: checkCtxt.scScanCtxtObject;
       IF scSC.wordScanCtxtObject.scanCtxtObject.ec # NIL THEN {  --in the middle of checking words
          a: ARRAY [0..1) OF LexiconDefs.LexiconHandle ← [checkCtxt.ignoreList];
          isFull: BOOLEAN ← FALSE;

          [] ← LexiconDefs.AddWord[
             DESCRIPTOR[a], Rdr[@scSC.normalizedWord], @scSC.wordScanCtxtObject.wordFlags !
             LexiconDefs.LexiconIsFull => {
                msgIgnoreListIsFull: XS.ReaderBody ← XM.Get[h, keySCMsgIgnoreListIsFull];
                Attention.Post[@msgIgnoreListIsFull];
                isFull ← TRUE;
                CONTINUE}];
          IF ~isFull THEN Continue[window, menu, itemData];
```

```
            };
        };
    }; -- IgnoreWord


MakeDocument: MenuData.MenuProc = {
    OPEN c: checkCtxt;
    -- *** THIS IS JUST A HACK BECAUSE StarFileTypeForgotDefs DOES NOT CURRENTLY EXIST.  PREVIOUSLYm INSTEAD OF DEFINING lexiconType,
    THE VALUE StarFileTypeForgotDefs.lexicon WAS USED.
    msgSelectALexicon: XS.ReaderBody ← XM.Get[h, keySCMsgSelectALexicon];

    MakeDocForLexicon: PROC [
        lexicon: LexiconDefs.LexiconHandle, name: XS.Reader] = {
        IF lexicon # NIL THEN {
            msgMakeDocInProgress: XS.ReaderBody ← XM.Get[h, keySCMsgMakeDocInProgress];
            msgMakeDocComplete: XS.ReaderBody ← XM.Get[h, keySCMsgMakeDocComplete];
            msgUnknownError: XS.ReaderBody ← XM.Get[h, keySCMsgUnknownError];
            ok: BOOL;
            Attention.Post[@msgMakeDocInProgress];
            ok ← MakeDoc[lexicon, name];
            Attention.Post[
               (IF ok THEN @msgMakeDocComplete ELSE @msgUnknownError)];
            }
        ELSE {
            msgCantOpenLexicon: XS.ReaderBody ← XM.Get[h, keySCMsgCantOpenLexicon];
            Attention.Post[@msgCantOpenLexicon]; };
        }; -- MakeDocForLexicon

    EnumerationProc: Selection.EnumerationProc = {
        ConvertToDoc[LOOPHOLE[element.value, LONG POINTER TO NSFile.Reference]];
        Selection.Free[@element];
        };

    ConvertToDoc: PROC [value: LONG POINTER TO NSFile.Reference] = {
      FOR i: CARDINAL IN [0..c.lexicons.nLexicons) DO
         lex: LONG POINTER TO LexiconData ← @c.lexicons[i];

         IF lex.ref = value↑ THEN {
            MakeDocForLexicon[lex.lexicon, lex.name]; EXIT};
         REPEAT
            FINISHED => {
               lexiconFile: NSFile.Handle =
                  NSFile.OpenByReference[value↑];
               IF NSFile.GetType[lexiconFile] = HashTableLexiconDefs.fileType THEN {
               -- DO I NEED TO CHECK THAT fileType IS DOC HERE? (StarFileTypes, NSFile.GetRef)
                  lexicon: LexiconDefs.LexiconHandle = LexiconDefs.OpenLexicon[
                  lexiconFile: lexiconFile, readonly: TRUE, z: c.zone];
                  ar: NSFile.AttributesRecord ← TRASH;
                  rb: XString.ReaderBody;
                  NSFile.GetAttributes[lexiconFile, [interpreted: [name: TRUE]], @ar];
                  rb ← XString.FromNSString[ar.name];
                  MakeDocForLexicon[lexicon, @rb];
                  NSFile.ClearAttributes[@ar];
                  IF lexicon # NIL THEN lexicon.close[lexicon];
                  }
               ELSE Attention.Post[@msgSelectALexicon];
               NSFile.Close[lexiconFile];
               };
         ENDLOOP;
         }; --ConvertToDoc

    IF Selection.CanYouConvert[file] THEN {
        element: Selection.Value ← Selection.Convert[file, myZone];
        ConvertToDoc[LOOPHOLE[element.value, LONG POINTER TO NSFile.Reference]];
        Selection.Free[@element];
        }
    ELSE IF Selection.CanYouConvert[file, TRUE]
        THEN [] ← Selection.Enumerate[proc: EnumerationProc, target: file]
        ELSE Attention.Post[@msgSelectALexicon];

    }; -- MakeDocument


PopUpMenuMakeStringProc: PUBLIC FormWindow.TextHintsProc = {
    OPEN c: checkCtxt;

    freeHintProc: FormWindow.FreeTextHintsProc ← NIL;

    InvalidAltList: PROC RETURNS [BOOL] = {
        RETURN[c.invalidAltList OR NewAlternativesWord[]]};

    NewAlternativesWord: PROC RETURNS [new: BOOL] = {
        lsWord: LONG STRING = [nBytesLongestWord];
        word: XS.WriterBody ← XS.WriterBodyFromSTRING[
            s: lsWord, homogeneous: TRUE];

        AppendAltWordfromCorrectionParm[@word, NIL];
        -- note that this test covers the cases where the new alt word is 0-length or the old alt-word was 0-length or both or neither
        new ← ~XS.Equal[Rdr[@word], Rdr[@c.alternativesWord]];
        };

    WaitAlternativesDone: ENTRY PROC = {
        ENABLE UNWIND => NULL;
        WHILE checkCtxt.generatingAlternatives DO
            WAIT checkCtxt.cDoneGeneratingAlternatives; ENDLOOP;
        };
```

```
        IF GeneratingAlternatives[] THEN {  --generation in progress
           --see if the user has changed anything that would invalidate the current list
           IF InvalidAltList[] THEN {
              StopAlternativesGeneration[]; GetAlternatives[]; }
           ELSE {
              Cursor.Set[hourGlass];
              WaitAlternativesDone[];
              Cursor.Set[textPointer];
              };
           }
        ELSE  --either done, or never started
           IF InvalidAltList[] THEN GetAlternatives[];

        IF c.nWords <= 0 THEN {
           altWordArray[0] ← XM.Get[h, keySCNoAlternatives];
           RETURN[DESCRIPTOR[altWordArray, 1], freeHintProc]}
        ELSE {
           string: XS.Reader ← XS.CopyReader[r: Rdr[@c.alternatives], z: c.zone];
           FOR i: CARDINAL IN [0..c.nWords) DO
              altWordArray[i] ← XS.ScanForCharacter[
                 string,
                 XCharSet0.Make[XCharSet0.Codes0.newLine],
                 ignore].front;
              ENDLOOP;
           RETURN[DESCRIPTOR[altWordArray, c.nWords], freeHintProc]
           };

        };  -- PopUpMenuMakeStringProc


     GeneratingAlternatives: ENTRY PROC RETURNS [BOOL] = {
        ENABLE UNWIND => NULL; RETURN[checkCtxt.generatingAlternatives]};


     -- gets a normalized word
     AppendAltWordFromCorrectionParm: PROC [
        altWord: XS.Writer, flags: TxtScanDefs.WordFlags] = {
        tsBlock: TxtDefs.TextSegment;
        string: XString.ReaderBody;

        Show: TxtScanDefs.WordProc = {
           XS.AppendReader[to: altWord, from: word];
           IF flags # NIL THEN flags↑ ← wordFlags↑;
           status ← abort;  --only do the first word
           };  --Show

        string ← FW.GetTextItemValue[
           checkCtxt.fw, Items.correction.ORD, myZone];
        << The scratchBlock was created in the SystemDoc by
           SpellingCheckerWnPack.Init.  Since the block resides in the SystemDoc,
           we must prohibit every other editing action in that doc until we're
           through. >>
        DocUtilDefs.LockSystemDoc[];
        {ENABLE UNWIND => DocUtilDefs.FreeSystemDoc[];
        tsBlock ← TextUtilDefs.TextSegmentFromXString[@string, checkCtxt.scratchBlock];
        IF flags # NIL THEN flags.nChars ← 0;
        IF tsBlock # TxtDefs.textSegmentNil THEN {
           [] ← TxtScanDefs.ScanWords[
              range: tsBlock, wordProc: Show, nBytesLongestWord: nBytesLongestWord];
           TxtEditDefs.DestroyTextSegment[@tsBlock];
           };  --else leave altWord empty
        };  -- ENABLE
        DocUtilDefs.FreeSystemDoc[];
        };  -- AppendAltWordFromCorrectionParm


     GetAlternatives: PROC [] = {  --call from foreground...
        OPEN c: checkCtxt;
        lookUp: LexiconDefs.Lexicons = GetActiveLexicons[].lookUp;

        Cursor.Set[hourGlass];
        c.rsAlternatives ← [0, 0];
        ResetWriter[@c.alternativesWord];
        AppendAltWordFromCorrectionParm[
           @c.alternativesWord, @c.alternativesWordFlags];
        c.invalidAltList ← c.abortAlternatives ← FALSE;
        GenerateAlternatives[
           lookUp, Rdr[@c.alternativesWord], @c.alternativesWordFlags,
           @c.alternatives];
        Cursor.Set[textPointer];
        };


     StopAlternativesGeneration: PUBLIC ENTRY PROC = {
        ENABLE UNWIND => NULL;
        checkCtxt.abortAlternatives ← TRUE;
        WHILE checkCtxt.generatingAlternatives DO
           WAIT checkCtxt.cDoneGeneratingAlternatives; ENDLOOP;
        checkCtxt.abortAlternatives ← FALSE;
        checkCtxt.invalidAltList ← TRUE;  --current list is invalid
        };


     -- word must be normalized
     StartAlternativesGeneration: PUBLIC ENTRY PROC [
        lookUp: LexiconDefs.Lexicons, word: XS.Reader,
        wordFlags: TxtScanDefs.ReadonlyWordFlags] = {
```

```
        ENABLE UNWIND => NULL;
        IF ~checkCtxt.generatingAlternatives THEN {
           OPEN c: checkCtxt;
           ResetWriter[@c.alternativesWord];
           XS.AppendReader[to: @c.alternativesWord, from: word];
           c.alternativesWordFlags ← wordFlags↑;
           c.invalidAltList ← FALSE;
           c.abortAlternatives ← FALSE;
           c.alternativesLookUp ← lookUp;
           c.generatingAlternatives ← TRUE;
           NOTIFY c.cGenerateAlternatives;
           };
        };

     AlternativesStopped: ENTRY PROC RETURNS [BOOL] = {
        ENABLE UNWIND => NULL; RETURN[checkCtxt.abortAlternatives]};


     -- word must be normalized
     GenerateAlternatives: PROC [
        lookUp: LexiconDefs.Lexicons, word: XS.Reader,
        wordFlags: TxtScanDefs.ReadonlyWordFlags, altList: XS.Writer] = {
        OPEN c: checkCtxt;
        {
        ENABLE XS.InsufficientRoom => CONTINUE;

        c.nWords ← 0;

        ResetWriter[altList];
        IF lookUp = NIL THEN {
           msgNoLookUpLexiconsSpecified: XS.ReaderBody ← XM.Get[h, keySCMsgNoLookUpLexiconsSpecified];
           XS.AppendReader[to: altList, from: @msgNoLookUpLexiconsSpecified];
           c.nWords ← 1;
           RETURN;
           };

        IF XS.ByteLength[word] > nBytesLongestWord THEN {
           msgAlternativesWordTooLong: XS.ReaderBody ← XM.Get[h, keySCMsgAlternativesWordTooLong];
           XS.AppendReader[to: altList, from: @msgAlternativesWordTooLong];
           c.nWords ← 1;
           RETURN;
           };

        IF wordFlags.nChars > 0 THEN {
           lsNorm: LONG STRING = [nBytesLongestWord + 10];
           lsTemp: LONG STRING = [nBytesLongestWord + 10];
           wbNorm: XS.WriterBody ← XS.WriterBodyFromSTRING[
              s: lsNorm, homogeneous: TRUE];
           wNorm: XS.Writer = @wbNorm;  -- buffer for guys below to normalize string into before lookup
           wbTemp: XS.WriterBody ← XS.WriterBodyFromSTRING[
              s: lsTemp, homogeneous: TRUE];
           wTemp: XS.Writer = @wbTemp;  -- buffer for guys below to copy string into before editing it

           -- wNorm must be zero-offset
           wbNorm.offset ← wbNorm.limit ← 0;  --if not already.  Claim this is ok, since wbNorm.bytes is the start of the allocation unit,
           and we can use them if we want.  This is prompted by XString's returning bytes = string pointer, offset = 4 in current
           implementation (6-20-84); we will use the length and maxlength part of the string as bytes also

           CheckWord[lookUp, word, wordFlags, wNorm, altList, @c.nWords];  -- 1

           IF AlternativesStopped[] THEN RETURN;
           Deletions[lookUp, word, wordFlags, wTemp, wNorm, altList, @c.nWords];  -- n (for n>1)

           IF AlternativesStopped[] THEN RETURN;
           Substitutions[lookUp, word, wordFlags, wTemp, wNorm, altList, @c.nWords];  -- 25n

           IF AlternativesStopped[] THEN RETURN;
           Transpositions[lookUp, word, wordFlags, wTemp, wNorm, altList, @c.nWords];  -- n-1 - (# of same-letter pairs), for n>1

           IF AlternativesStopped[] THEN RETURN;
           Insertions[lookUp, word, wordFlags, wTemp, wNorm, altList, @c.nWords];  -- 26*(n+1)

           IF AlternativesStopped[] THEN RETURN;
           };

        IF WLength[altList] = 0 THEN {
           msgNoAlternatives: XS.ReaderBody ← XM.Get[h, keySCNoAlternatives];
           XS.AppendReader[to: altList, from: @msgNoAlternatives]; };
        };  --ENABLE block

        };  --GenerateAlternatives


     GenerateAlternativesInBkgd: PROC = {
        ENABLE ABORTED => CONTINUE;

        NotifyAlternativesDone: ENTRY PROC = {
           ENABLE UNWIND => NULL;
           checkCtxt.generatingAlternatives ← FALSE;
           NOTIFY checkCtxt.cDoneGeneratingAlternatives;
           };

        WaitAlternativesStart: ENTRY PROC = {
           ENABLE UNWIND => NULL;
           UNTIL checkCtxt.generatingAlternatives DO
              WAIT checkCtxt.cGenerateAlternatives; ENDLOOP;
```

```
        };

    Process.SetPriority[Process.priorityBackground];
    DO
       OPEN c: checkCtxt;
       WaitAlternativesStart[];
       c.rsAlternatives ← [0, 0];
       GenerateAlternatives[
          c.alternativesLookUp, Rdr[@c.alternativesWord],
          @c.alternativesWordFlags, @c.alternatives];
       NotifyAlternativesDone[];
       ENDLOOP;
    };


 Insertions: PROC [
    lookUp: LexiconDefs.Lexicons, word: XS.Reader,
    wordFlags: TxtScanDefs.ReadonlyWordFlags, wTemp, wNorm, strList: XS.Writer,
    nWords: LONG POINTER TO CARDINAL] =
    {
    wordFlagsLocal: TxtScanDefs.WordFlagsObject ← wordFlags↑;
    c, cPrev: XChar.Character;
    i: CARDINAL ← 0;
    cFirst, cLast: XChar.Character;

    IF wordFlags.nChars = 0 THEN RETURN;

    wordFlagsLocal.nChars ← wordFlagsLocal.nChars+1;  -- doing insertions!

    [cFirst, cLast] ← GetLetterRange[
       chset: XChar.Set[XS.First[word]],
       case: (IF wordFlags.case = allUpper THEN upper ELSE lower)];

    IF cFirst = XChar.not THEN RETURN;

    cPrev ← XChar.not;
    FOR i: CARDINAL IN [0..wordFlags.nChars] DO  -- go to nChars+1 here
       ResetWriter[wTemp];
       XS.AppendReader[to: wTemp, from: word];
       InsertChar[wTemp, i, LOOPHOLE[cFirst]];  --if i > last string position, put it at the end...
       --don't insert a char same as previous one, to suppress duplicate lookups
       IF cFirst # cPrev THEN
          CheckWord[lookUp, Rdr[wTemp], @wordFlagsLocal, wNorm, strList, nWords];

       IF AlternativesStopped[] THEN RETURN;
       LOOPHOLE[c, XChar.CharRep].set ← XChar.Set[cFirst];
       FOR code: Environment.Byte IN (XChar.Code[cFirst]..XChar.Code[cLast]] DO
          LOOPHOLE[c, XChar.CharRep].code ← code;
          IF ~XCharProps.IsLetter[c] THEN LOOP;  -- may be sparse range
          IF c # cPrev THEN {
             --case-sensitive compare will get duplicate spellings but only if case differs
             IF AlternativesStopped[] THEN RETURN;
             [] ← ReplaceChar[wTemp, i, LOOPHOLE[c]];
             CheckWord[lookUp, Rdr[wTemp], @wordFlagsLocal, wNorm, strList, nWords];
             };
          ENDLOOP;
       cPrev ← XS.NthCharacter[word, i];
       ENDLOOP;
    };  --Insertions


 << Deletions: word is in normalized form
leave case flags alone after deletions.  This is ok, since:
  1) if case = allLower, or allUpper, clearly ok
  2) if case = firstUpperOnly, probably doesn't matter; at worst, someone will think that the new word should have it's first char
  upper, which is ok anyway...
  3) if case = mixed, 2 subcases
    only 1 char is upper => deleting it makes it all lower; but if lexicons just insist on an exact match with mixed case, that will
    still be ok since now all lower case
    > 1 uppercase => deleting any 1 char, word is still mixed >>

 Deletions: PROC [
    lookUp: LexiconDefs.Lexicons, word: XS.Reader,
    wordFlags: TxtScanDefs.ReadonlyWordFlags, wTemp, wNorm, strList: XS.Writer,
    nWords: LONG POINTER TO CARDINAL] =
    {
    c, cPrev: CharDefs.Char;
    wordFlagsLocal: TxtScanDefs.WordFlagsObject ← wordFlags↑;

    IF wordFlags.nChars < 2 THEN RETURN;
    wordFlagsLocal.nChars ← wordFlagsLocal.nChars-1;  -- for deletions!
    cPrev ← LOOPHOLE[XChar.not];

    FOR i: CARDINAL IN [0..wordFlags.nChars) DO
       IF AlternativesStopped[] THEN RETURN;
       ResetWriter[wTemp];
       XS.AppendReader[to: wTemp, from: word];
       c ← DeleteChar[wTemp, i];

       IF c # cPrev THEN  --case-sensitive compare, to allow duplicate spellings with different case
          CheckWord[lookUp, Rdr[wTemp], @wordFlagsLocal, wNorm, strList, nWords];
       cPrev ← c;
       ENDLOOP;
    };  --Deletions


 Substitutions: PROC [
```

```
        lookUp: LexiconDefs.Lexicons, word: XS.Reader,
        wordFlags: TxtScanDefs.ReadonlyWordFlags, wTemp, wNorm, strList: XS.Writer,
        nWords: LONG POINTER TO CARDINAL] =
        {
        bytes: XS.Bytes;
        limit, byteN: CARDINAL;
        chset: CharDefs.Chset;
        code: CharDefs.Code;

        IF wordFlags.nChars = 0 THEN RETURN;
        ResetWriter[wTemp];
        XS.AppendReader[to: wTemp, from: word];

        bytes ← wTemp.bytes;
        limit ← wTemp.limit;
        byteN ← wTemp.offset;
        chset ← wTemp.context.prefix;

        -- note: substitutions are done in-place, pad byte remains ok
        UNTIL byteN = limit DO
           IF bytes[byteN] = 377B THEN {
              chset ← bytes[byteN + 1]; byteN ← byteN + 2; };

           --substitute at byteN
           IF AlternativesStopped[] THEN RETURN;
           code ← bytes[byteN];

           -- returns char in same chset as input char
           BEGIN
           xchar: XChar.Character ← XChar.Make[chset, code];  -- scratch char
           cFirst, cLast: XChar.Character;
           IF XCharProps.IsLetter[xchar] THEN {   --substitute
              [cFirst, cLast] ← GetLetterRange [
                  chset: chset,
                  case: (IF XCharProps.IsUpperCase[xchar] THEN upper ELSE lower)];
              IF cFirst # XChar.not THEN -- paranoid check...
                FOR codeT: Environment.Byte IN [XChar.Code[cFirst]..XChar.Code[cLast]] DO
                   IF AlternativesStopped[] THEN RETURN;
                   LOOPHOLE[xchar, XChar.CharRep].code ← codeT;
                   IF ~XCharProps.IsLetter[xchar] THEN LOOP;  -- may be sparse range
                   IF codeT # code THEN {
                      bytes[byteN] ← codeT;
                      CheckWord[lookUp, Rdr[wTemp], wordFlags, wNorm, strList, nWords];
                      bytes[byteN] ← code;  --replace original
                      };
                   ENDLOOP;
              };
           END;

           byteN ← byteN + 1;
           ENDLOOP;
        }; -- Substitutions

     -- Transpositions: word must be normalized, case flag never changes
     Transpositions: PROC [
        lookUp: LexiconDefs.Lexicons, word: XS.Reader,
        wordFlags: TxtScanDefs.ReadonlyWordFlags, wTemp, wNorm, strList: XS.Writer,
        nWords: LONG POINTER TO CARDINAL] =
        {
        c, cPrev: CharDefs.Char;

        IF wordFlags.nChars < 2 THEN RETURN;
        cPrev ← LOOPHOLE[XChar.not];

        FOR i: CARDINAL IN [1..wordFlags.nChars) DO
           IF AlternativesStopped[] THEN RETURN;
           ResetWriter[wTemp];
           XS.AppendReader[to: wTemp, from: word];
           c ← DeleteChar[wTemp, i];
           IF c # cPrev THEN {  --don't transpose same chars
              -- case-sensitive compare, to get duplicate spellings with different case
              InsertChar[wTemp, i - 1, c];  --if insertPos > last string position, then puts it at the end...
              CheckWord[lookUp, Rdr[wTemp], wordFlags, wNorm, strList, nWords];
              };
           cPrev ← c;
           ENDLOOP;
        };  --Transpositions


     -- if the normalized form is there, append the original form...
     -- this is only to be used for Corrections generation
     CheckWord: PROC [
        lookUp: LexiconDefs.Lexicons, word: XS.Reader,
        wordFlags: TxtScanDefs.ReadonlyWordFlags, wNorm, strList: XS.Writer,
        nWords: LONG POINTER TO CARDINAL] = {
        IF wordFlags.nChars > 0 THEN {
           LexiconDefs.GetNormalizedWord[word, wordFlags, wNorm];
           IF LexiconDefs.LookUpWord[lookUp, Rdr[wNorm], wordFlags, TRUE].found THEN {
              IF nWords↑ < 50 THEN {
                 IF WLength[strList] > 0 THEN  --add CR separator
                    XS.AppendChar[to: strList, c: LOOPHOLE[CharDefs.Roman[newLine]]];
                 XS.AppendReader[to: strList, from: word];
                 nWords↑ ← nWords↑ + 1;
                 };
              };
           };
        }; -- CheckWord
```

```
    ReplaceContinueWithStart: PUBLIC PROC = {
       OPEN c: checkCtxt;
       menuHandle: MenuData.MenuHandle ← StarWindowShell.GetRegularCommands[checkCtxt.sws];
     IF c.continueShowing THEN
          MenuData.SwapItem[menu: menuHandle, old: continueMenuItem, new: startMenuItem];
          c.continueShowing ← FALSE;
       }; -- ReplaceContinueWithStart


    ReplaceStartWithContinue: PUBLIC PROC = {
       OPEN c: checkCtxt;
       menuHandle: MenuData.MenuHandle ← StarWindowShell.GetRegularCommands[checkCtxt.sws];

       IF ~c.continueShowing THEN
          MenuData.SwapItem[menu: menuHandle, old: startMenuItem, new: continueMenuItem];

         c.continueShowing ← TRUE;
       }; -- ReplaceStartWithContinue


    Start: MenuData.MenuProc = {
       lookUp: LexiconDefs.Lexicons = GetActiveLexicons[].lookUp;

       IF lookUp = NIL THEN {
          msgNoLookUpLexiconsSpecified: XS.ReaderBody ← XM.Get[h,          keySCMsgNoLookUpLexiconsSpecified];
          Attention.Post[@msgNoLookUpLexiconsSpecified] }
       ELSE StartChecking[lookUp];
       }; -- Start


    GetLetterRange: PROC [chset: Environment.Byte, case: XCharProps.Case]
      RETURNS [cFirst, cLast: XChar.Character] = INLINE {
      RETURN (IF chset = XCharSets.Sets.latin.ORD THEN
          IF case = upper THEN [XCharSet0.Codes0.upperA.ORD, XCharSet0.Codes0.upperZ.ORD]
          ELSE [XCharSet0.Codes0.lowerA.ORD, XCharSet0.Codes0.lowerZ.ORD]
        ELSE XCharProps.GetLetterRange[chset, case])};

   Init: PROC = {
       << Initialize SC tool context if the SC is installed in the product >>
       IF ProductFactoring.Enabled[StarPFOptions.starSpelling] THEN
          BEGIN

          myZone ← checkCtxt.zone;

          --allocate the string for alternatives list
          checkCtxt.alternatives ← XS.NewWriterBody[
             lengthAlternativesList + 1, checkCtxt.zone];

          checkCtxt.alternativesWord ← XS.NewWriterBody[
             nBytesLongestWord, checkCtxt.zone];

          Process.InitializeCondition[@checkCtxt.cGenerateAlternatives, 1];
          Process.DisableTimeout[@checkCtxt.cGenerateAlternatives];
          Process.EnableAborts[@checkCtxt.cGenerateAlternatives];

          Process.InitializeCondition[@checkCtxt.cDoneGeneratingAlternatives, 1];
          Process.DisableTimeout[@checkCtxt.cDoneGeneratingAlternatives];

          END;
          }; -- Init


    -- MAINLINE

    Init[];


    END.  --  SpellingCheckerMenuPack

LOG     (date - person - action)
 5-Jun-84 16:50:40 - Walden - OS5.0 release version 1
 5-Jul-84 11:54:59 - Walden
17-Jul-84 18:12:42 - D.J. Lewis - Tie initialization to FeatureDefs product factoring flag.
 9-Mar-85 10:22:13 - Marks - Update to OS6.
12-Mar-85  8:45:04 - Marks - Replace RealToReaderBody with TextUtilDefs.AppendNumber.
15-Mar-85 15:08:09 - Goodell - ReplaceWithString now comes from TxtEditDefs
28-Mar-85 11:16:44 - Marks - Correctly add SC items to property sheet Aux menu.   1-Apr-85 14:42:50 - Marks - Use BWS product factoring.
12-Apr-85  9:31:17 - Marks - allocate altWordArray using Space.ScratchMap.
15-Apr-85 15:05:50 - Marks - corret use of Selection.CanYouConvert.
22-May-85 20:02:27 - Marks - check filetype before convert doc
26-Jun-85 15:46:12 - Marks - Use TIP.UserAbort.
31-Jul-85 15:14:34 - Marks - AR15416: keySCMsgConfirmCorrectionWasntApplied changed to be understandable.
11-Sep-85 13:13:32 - Marks - AR19939: Correct - set doc edited then set S.C. doc edited bit off.
27-Sep-85 17:38:00 - Marks - AR209555: CreateAlternatives process is forked and later a Process.Abort is called on it.  A JOIN is never
called, but needs to because an Abort does not kill the process.
 3-Feb-86 13:57:22 - Maybury - Adopted HashTableLexiconDefs.fileType (vs. hackLexicon).
21-Feb-86 12:57:24 - Bartlett - use LockSystemDoc, FreeSystemDoc
 3-Feb-86 13:57:22 - Maybury - Continue, Correct, IgnoreWord: conform to new SCScanCtxtObject
 5-Dec-86 11:02:40 - Bartlett - support Redlining in Correct
11-Dec-86 - Lewis - Get IsLetter, IsUpperCase, GetLetterRange from XCharProps.
26-Mar-87 - Walden - remove nsName from lex info passed to MakeDoc; just create that string on demand inside MakeDoc
 1-May-87 10:15:02 - Marks - use BlockFriendsDefs instead of BlockPrivExtraIDefs.
28-May-87 11:19:55 - Walden - Insertions, Substitutions: check for XChar.not from GetLetterRange, no-op when encountered
24-Jul-87 16:35:58 - Walden - Correct: set endOfRange to FALSE when correcting, make sure word is rechecked even if end of range
encountered when getting the word
```

7-Aug-87 15:46:38 - Marks - Don't use ToolUtilitiesDefs.GetFileName. In ConverToDoc make calls myself and fix AR13660 by creating own reader.
16-Nov-87 17:41:26 - Marks - AR 14272 - Continue after IgnoreWord or CorrectWord.
2-Dec-87 13:26:32 - Bartlett - added UNWIND call to FreeSystemDoc for AR 16204

```
-- File: SpellingCheckerMessageDefs.mesa - last edit:
-- Marks.ES            31-Jul-85 15:21:54
-- Owner: Document Group


DIRECTORY
   XMessage USING [Handle, MsgKey];

SpellingCheckerMessageDefs: DEFINITIONS = BEGIN
   OPEN XM: XMessage;

   MsgKey: TYPE = MACHINE DEPENDENT {
     scTitle, scLookUp, scEdit, scScope, scAllText, scRemainingText, scSelectedText, scMisspelling, scCorrection, scCheckFrames,
     scAutoCorrect, scAmerEngLexicon, scEmptyLexicon, scClose, scStart, scContinue, scCorrect, scAddWords, scIgnoreWord,
     scBatchCheckAndAdd, scDeleteWord, scMakeDocument, scMoveWindow, scWordsListedInDoc, scNoAlternatives, scLexiconFolder,
     scConvertLexicons, scMsgSpellingCheckerAlreadyActive, scMsgNeedAppropriateSelection, scMsgCheckingSpelling, scMsgCheckingComplete,
     scMsgUserCancelled, scMsgMustLogIn, scMsgBatchAddingWords, scMsgConfirmBatchAddWords, scMsgConfirmDeleteAllWords,
     scMsgNoEditLexiconsSpecified, scMsgNoLookUpLexiconsSpecified, scMsgConfirmEditLexiconIsFull, scMsgConfirmLexiconCantBeOpened,
     scMsgCorrectionListIsFull, scMsgIgnoreListIsFull, scMsgCantIgnoreMultipleWords, scMsgUserEditedDocCantCorrect,
     scMsgUserEditedDocCantContinue, scMsgNoWordToCorrect, scMsgCorrectionTooLong, scMsgConfirmCorrectionWasntApplied,
     scMsgCantAutoCorrectCaseChange, scMsgCorrectionWasntChanged, scMsgNAdded, scMsgAllDeleted, scMsgNDeleted, scMsgMakeDocInProgress,
     scMsgMakeDocComplete, scMsgSelectALexicon, scMsgLexiconIsEmpty, scMsgCantOpenLexicon, scMsgAlternativesWordTooLong,
     scMsgUnknownError, scMsgUnimplemented, scLexicon, scMsgReadOnlyAccess, scMsgConfirm, scMsgCancel};

   keySCTitle: XM.MsgKey = 0;
   keySCLookUp: XM.MsgKey = keySCTitle + 1;
   keySCEdit: XM.MsgKey = keySCLookUp + 1;
   keySCScope: XM.MsgKey = keySCEdit + 1;
   keySCAllText: XM.MsgKey = keySCScope + 1;
   keySCRemainingText: XM.MsgKey = keySCAllText + 1;
   keySCSelectedText: XM.MsgKey = keySCRemainingText + 1;
   keySCMisspelling: XM.MsgKey = keySCSelectedText + 1;
   keySCCorrection: XM.MsgKey = keySCMisspelling + 1;
   keySCCheckFrames: XM.MsgKey = keySCCorrection + 1;
   keySCAutoCorrect: XM.MsgKey = keySCCheckFrames + 1;
   keySCAmerEngLexicon: XM.MsgKey = keySCAutoCorrect + 1;
   keySCEmptyLexicon: XM.MsgKey = keySCAmerEngLexicon + 1;
   keySCClose: XM.MsgKey = keySCEmptyLexicon + 1;
   keySCStart: XM.MsgKey = keySCClose + 1;
   keySCContinue: XM.MsgKey = keySCStart + 1;
   keySCCorrect: XM.MsgKey = keySCContinue + 1;
   keySCAddWords: XM.MsgKey = keySCCorrect + 1;
   keySCIgnoreWord: XM.MsgKey = keySCAddWords + 1;
   keySCBatchCheckAndAdd: XM.MsgKey = keySCIgnoreWord + 1;
   keySCDeleteWord: XM.MsgKey = keySCBatchCheckAndAdd + 1;
   keySCMakeDocument: XM.MsgKey = keySCDeleteWord + 1;
   keySCMoveWindow: XM.MsgKey = keySCMakeDocument + 1;

   keySCWordsListedInDoc: XM.MsgKey = keySCMoveWindow + 1;

   keySCNoAlternatives: XM.MsgKey = keySCWordsListedInDoc + 1;
   keySCLexiconFolder: XM.MsgKey = keySCNoAlternatives + 1;
   keySCConvertLexicons: XM.MsgKey = keySCLexiconFolder + 1;

   keySCMsgSpellingCheckerAlreadyActive: XM.MsgKey = keySCConvertLexicons + 1;
   keySCMsgNeedAppropriateSelection: XM.MsgKey = keySCMsgSpellingCheckerAlreadyActive + 1;
   keySCMsgCheckingSpelling: XM.MsgKey = keySCMsgNeedAppropriateSelection + 1;
   keySCMsgCheckingComplete: XM.MsgKey = keySCMsgCheckingSpelling + 1;
   keySCMsgUserCancelled: XM.MsgKey = keySCMsgCheckingComplete + 1;
   keySCMsgMustLogIn: XM.MsgKey = keySCMsgUserCancelled + 1;
   keySCMsgBatchAddingWords: XM.MsgKey = keySCMsgMustLogIn + 1;
   keySCMsgConfirmBatchAddWords: XM.MsgKey = keySCMsgBatchAddingWords + 1;
   keySCMsgConfirmDeleteAllWords: XM.MsgKey = keySCMsgConfirmBatchAddWords + 1;
   keySCMsgNoEditLexiconsSpecified: XM.MsgKey = keySCMsgConfirmDeleteAllWords + 1;
   keySCMsgNoLookUpLexiconsSpecified: XM.MsgKey = keySCMsgNoEditLexiconsSpecified + 1;
   keySCMsgConfirmEditLexiconIsFull: XM.MsgKey = keySCMsgNoLookUpLexiconsSpecified + 1;
   keySCMsgConfirmLexiconCantBeOpened: XM.MsgKey = keySCMsgConfirmEditLexiconIsFull + 1;
   keySCMsgCorrectionListIsFull: XM.MsgKey = keySCMsgConfirmLexiconCantBeOpened + 1;
   keySCMsgIgnoreListIsFull: XM.MsgKey = keySCMsgCorrectionListIsFull + 1;
   keySCMsgCantIgnoreMultipleWords: XM.MsgKey = keySCMsgIgnoreListIsFull + 1;
   keySCMsgUserEditedDocCantCorrect: XM.MsgKey = keySCMsgCantIgnoreMultipleWords + 1;
   keySCMsgUserEditedDocCantContinue: XM.MsgKey = keySCMsgUserEditedDocCantCorrect + 1;
   keySCMsgNoWordToCorrect: XM.MsgKey = keySCMsgUserEditedDocCantContinue + 1;
   keySCMsgCorrectionTooLong: XM.MsgKey = keySCMsgNoWordToCorrect + 1;
   keySCMsgConfirmCorrectionWasntApplied: XM.MsgKey = keySCMsgCorrectionTooLong + 1;
   keySCMsgCantAutoCorrectCaseChange: XM.MsgKey = keySCMsgConfirmCorrectionWasntApplied + 1;
   keySCMsgCorrectionWasntChanged: XM.MsgKey = keySCMsgConfirmCorrectionWasntApplied + 1;
   keySCMsgNAdded: XM.MsgKey = keySCMsgCorrectionWasntChanged + 1;
   keySCMsgAllDeleted: XM.MsgKey = keySCMsgNAdded + 1;
   keySCMsgNDeleted: XM.MsgKey = keySCMsgAllDeleted + 1;
   keySCMsgMakeDocInProgress: XM.MsgKey = keySCMsgNDeleted + 1;
   keySCMsgMakeDocComplete: XM.MsgKey = keySCMsgMakeDocInProgress + 1;
   keySCMsgSelectALexicon: XM.MsgKey = keySCMsgMakeDocComplete + 1;
   keySCMsgLexiconIsEmpty: XM.MsgKey = keySCMsgSelectALexicon + 1;
   keySCMsgCantOpenLexicon: XM.MsgKey = keySCMsgLexiconIsEmpty + 1;
   keySCMsgAlternativesWordTooLong: XM.MsgKey = keySCMsgCantOpenLexicon + 1;
   keySCMsgUnknownError: XM.MsgKey = keySCMsgAlternativesWordTooLong + 1;
   keySCMsgUnimplemented: XM.MsgKey = keySCMsgUnknownError + 1;
   keySCLexicon: XM.MsgKey = keySCMsgUnimplemented + 1;
   keySCMsgReadOnlyAccess: XM.MsgKey = keySCLexicon + 1;
   keySCMsgConfirm: XM.MsgKey = keySCMsgReadOnlyAccess + 1;
   keySCMsgCancel: XM.MsgKey = keySCMsgConfirm + 1;

   keyLastMsgKey: CARDINAL = keySCMsgCancel;

   GetHandle: PROCEDURE RETURNS[h: XM.Handle];

SpellingCheckerMessageDefs.mesa        31-Jul-85 15:21:54 PDT                1
```

```
    END...

LOG
  7-Dec-84  - Marks - Create.
 9-Mar-85 10:16:01 - Marks - Updates.
31-Jul-85 15:04:50 - Marks - AR17732: Add keySCMsgConfirm and keySCMsgCancel.
```

```
-- File: SpellingCheckerMessageImpl.mesa - last edit:
-- Marks.ES            31-Jul-85 15:21:34
-- Owner: Document Group


DIRECTORY
   SpellingCheckerMessageDefs,
   XMessage USING [AllocateMessages, DestroyMsgsProc, Handle, MsgEntry, RegisterMessages],
   XString USING [FromSTRING];

SpellingCheckerMessageImpl: PROGRAM
   IMPORTS XMessage, XString
   EXPORTS SpellingCheckerMessageDefs = BEGIN
   OPEN SpellingCheckerMessageDefs;


   h: XMessage.Handle ← NIL;

   GetHandle: PUBLIC PROCEDURE RETURNS [XMessage.Handle] = {RETURN[h]};

   DeleteMessages: XMessage.DestroyMsgsProc = {};

   -- keyFindAlreadyOpen and keyNeedDocTextSelection are not used in OS5.
   Init: PROCEDURE = {
      msgArray: ARRAY MsgKey OF XMessage.MsgEntry ← [
      -- Messages with 'key' are posted in herald.  Other messages appear in the property sheet.
         scTitle: [
            msgKey: keySCTitle,
            msg: XString.FromSTRING["Spelling Checker"L],
            translatable: TRUE,
            type: pSheetItem,
            id: 0],
         scLookUp: [
            msgKey: keySCLookUp,
            msg: XString.FromSTRING["Look Up"L],
            translatable: TRUE,
            type: pSheetItem,
            id: 1],
         scEdit: [
            msgKey: keySCEdit,
            msg: XString.FromSTRING["Edit"L],
            translatable: TRUE,
            type: pSheetItem,
            id: 2],
         scScope: [
            msgKey: keySCScope,
            msg: XString.FromSTRING["Check"L],
            translatable: TRUE,
            type: pSheetItem,
            id: 3],
         scAllText: [
            msgKey: keySCAllText,
            msg: XString.FromSTRING["All text"L],
            translatable: TRUE,
            type: pSheetItem,
            id: 4],
         scRemainingText: [
            msgKey: keySCRemainingText,
            msg: XString.FromSTRING["Remaining text"L],
            translatable: TRUE,
            type: pSheetItem,
            id: 5],
          scSelectedText: [
            msgKey: keySCSelectedText,
            msg: XString.FromSTRING["Selected text"L],
            translatable: TRUE,
            type: pSheetItem,
            id: 6],
         scMisspelling: [
            msgKey: keySCMisspelling,
            msg: XString.FromSTRING["Misspelling:"L],
            translatable: TRUE,
            type: pSheetItem,
            id: 7],
         scCorrection: [
            msgKey: keySCCorrection,
            msg: XString.FromSTRING["Correction"L],
            translatable: TRUE,
            type: pSheetItem,
            id: 8],
         scCheckFrames: [
            msgKey: keySCCheckFrames,
            msg: XString.FromSTRING["Include Frames"L],
            translatable: TRUE,
            type: pSheetItem,
            id: 9],
         scAutoCorrect: [
            msgKey: keySCAutoCorrect,
            msg: XString.FromSTRING["Auto-Correct"L],
            translatable: TRUE,
            type: pSheetItem,
            id: 10],
         scAmerEngLexicon: [
            msgKey: keySCAmerEngLexicon,
            msg: XString.FromSTRING["American English"L],
            translatable: TRUE,
            id: 11],
```

```
    scEmptyLexicon: [
       msgKey: keySCEmptyLexicon,
       msg: XString.FromSTRING["Empty Dictionary"L],
       translatable: TRUE,
       id: 12],
    scClose: [
       msgKey: keySCClose,
       msg: XString.FromSTRING["Close"L],
       translatable: TRUE,
       id: 13],
    scStart: [
       msgKey: keySCStart,
       msg: XString.FromSTRING["Start"L],
       translatable: TRUE,
       id: 14],
    scContinue: [
       msgKey: keySCContinue,
       msg: XString.FromSTRING["Continue"L],
       translatable: TRUE,
       id: 15],
    scCorrect: [
       msgKey: keySCCorrect,
       msg: XString.FromSTRING["Correct"L],
       translatable: TRUE,
       id: 16],
    -- The following are option sheet specific
    scAddWords: [
       msgKey: keySCAddWords,
       msg: XString.FromSTRING["Add"L],
       translatable: TRUE,
       type: pSheetItem,
       id: 17],
    scIgnoreWord: [
       msgKey: keySCIgnoreWord,
       msg: XString.FromSTRING["Ignore"L],
       translatable: TRUE,
       type: pSheetItem,
       id: 18],
    scBatchCheckAndAdd: [
       msgKey: keySCBatchCheckAndAdd,
       msg: XString.FromSTRING["Batch Check & Add"L],
       translatable: TRUE,
       type: pSheetItem,
       id: 19],
    scDeleteWord: [
       msgKey: keySCDeleteWord,
       msg: XString.FromSTRING["Delete"L],
       translatable: TRUE,
       type: pSheetItem,
       id: 20],
    scMakeDocument: [
       msgKey: keySCMakeDocument,
       msg: XString.FromSTRING["Make Document"L],
       translatable: TRUE,
       type: pSheetItem,
       id: 21],
    scMoveWindow: [
       msgKey: keySCMoveWindow,
       msg: XString.FromSTRING["Move Window"L],
       translatable: TRUE,
       type: pSheetItem,
       id: 22],

    scWordsListedInDoc: [
       msgKey: keySCWordsListedInDoc,
       msg: XString.FromSTRING["words"L],
       translatable: TRUE,
       type: pSheetItem,
       id: 23],

    scNoAlternatives: [
       msgKey: keySCNoAlternatives,
       msg: XString.FromSTRING["???"L],
       translatable: TRUE,
       id: 24],
    scLexiconFolder: [
       msgKey: keySCLexiconFolder,
       msg: XString.FromSTRING["Spelling Checker Dictionaries"L],
       translatable: TRUE,
       type: pSheetItem,
       id: 25],
    scConvertLexicons: [
       msgKey: keySCConvertLexicons,
       msg: XString.FromSTRING["Convert Dictionaries"L],
       translatable: TRUE,
       type: pSheetItem,
       id: 26],

    scMsgSpellingCheckerAlreadyActive: [
       msgKey: keySCMsgSpellingCheckerAlreadyActive,
       msg: XString.FromSTRING["The Spelling Checker is already open."L],
       translatable: TRUE,
       id: 27],
    scMsgNeedAppropriateSelection: [
       msgKey: keySCMsgNeedAppropriateSelection,
       msg: XString.FromSTRING["Can't Start: select document text first."L],
       translatable: TRUE,
```

```
      id: 28],
    scMsgCheckingSpelling: [
      msgKey: keySCMsgCheckingSpelling,
      msg: XString.FromSTRING["Checking spelling ..."L],
      translatable: TRUE,
      id: 29],
    scMsgCheckingComplete: [
      msgKey: keySCMsgCheckingComplete,
      msg: XString.FromSTRING["Done.  Words processed: "L],
      translatable: TRUE,
      id: 30],
    scMsgUserCancelled: [
      msgKey: keySCMsgUserCancelled,
      msg: XString.FromSTRING["Cancelled.   Words processed: "L],
      translatable: TRUE,
      id: 31],
    scMsgMustLogIn: [
      msgKey: keySCMsgMustLogIn,
      msg: XString.FromSTRING["Please log in first."L],
      translatable: TRUE,
      id: 32],
    scMsgBatchAddingWords: [
      msgKey: keySCMsgBatchAddingWords,
      msg: XString.FromSTRING["Checking/adding words to Edit dictionaries..."L],
      translatable: TRUE,
      id: 33],
    scMsgConfirmBatchAddWords: [
      msgKey: keySCMsgConfirmBatchAddWords,
      msg: XString.FromSTRING["Please confirm batch Add operation: "L],
      translatable: TRUE,
      id: 34],
    scMsgConfirmDeleteAllWords: [
      msgKey: keySCMsgConfirmDeleteAllWords,
      msg: XString.FromSTRING["Please confirm deleting all words in Edit dictionaries: "L],
      translatable: TRUE,
      id: 35],
    scMsgNoEditLexiconsSpecified: [
      msgKey: keySCMsgNoEditLexiconsSpecified,
      msg: XString.FromSTRING["Can't edit: No Edit dictionaries specified."L],
      translatable: TRUE,
      id: 36],
    scMsgNoLookUpLexiconsSpecified: [
      msgKey: keySCMsgNoLookUpLexiconsSpecified,
      msg: XString.FromSTRING["No Look Up dictionaries specified."L],
      translatable: TRUE,
      id: 37],
    scMsgConfirmEditLexiconIsFull: [
      msgKey: keySCMsgConfirmEditLexiconIsFull,
      msg: XString.FromSTRING["An Edit dictionary is full.  Do you wish to continue?"L],
      translatable: TRUE,
      id: 38],
    scMsgConfirmLexiconCantBeOpened: [
      msgKey: keySCMsgConfirmLexiconCantBeOpened,
      msg: XString.FromSTRING["A dictionary can't be opened.  Do you wish to continue?"L],
      translatable: TRUE,
      id: 39],
    scMsgCorrectionListIsFull: [
      msgKey: keySCMsgCorrectionListIsFull,
      msg: XString.FromSTRING["Can't Auto-Correct that word: "L],
      translatable: TRUE,
      id: 40],
    scMsgIgnoreListIsFull: [
      msgKey: keySCMsgIgnoreListIsFull,
      msg: XString.FromSTRING["Can't Ignore that word: "L],
      translatable: TRUE,
      id: 41],
    scMsgCantIgnoreMultipleWords: [
      msgKey: keySCMsgCantIgnoreMultipleWords,
      msg: XString.FromSTRING["Can't Ignore those words: "L],
      translatable: TRUE,
      id: 42],
    scMsgUserEditedDocCantCorrect: [
      msgKey: keySCMsgUserEditedDocCantCorrect,
      msg: XString.FromSTRING["Can't Correct: the document has been edited."L],
      translatable: TRUE,
      id: 43],
    scMsgUserEditedDocCantContinue: [
      msgKey: keySCMsgUserEditedDocCantContinue,
      msg: XString.FromSTRING["Can't Continue: the document has been edited."L],
      translatable: TRUE,
      id: 44],
    scMsgNoWordToCorrect: [
      msgKey: keySCMsgNoWordToCorrect,
      msg: XString.FromSTRING["Can't Correct: no source word to change."L],
      translatable: TRUE,
      id: 45],
    scMsgCorrectionTooLong: [
      msgKey: keySCMsgCorrectionTooLong,
      msg: XString.FromSTRING["Can't Correct: Correction field is too long."L],
      translatable: TRUE,
      id: 46],
    scMsgConfirmCorrectionWasntApplied: [
      msgKey: keySCMsgConfirmCorrectionWasntApplied,
      msg: XString.FromSTRING["The Correction field was changed but not the document.  Do you wish to continue: "L],
      translatable: TRUE,
      id: 47],
    scMsgCantAutoCorrectCaseChange: [
```

```
            msgKey: keySCMsgCantAutoCorrectCaseChange,
            msg: XString.FromSTRING["Can't Auto-Correct that word: the case change is insignificant."L],
            translatable: TRUE,
            id: 48],
         scMsgCorrectionWasntChanged: [
            msgKey: keySCMsgCorrectionWasntChanged,
            msg: XString.FromSTRING["Can't Correct: The spelling or case of the Correction wasn't changed."L],
            translatable: TRUE,
            id: 49],
         scMsgNAdded: [
            msgKey: keySCMsgNAdded,
            msg: XString.FromSTRING["Done.  Words added: "L],
            translatable: TRUE,
            id: 50],
         scMsgAllDeleted: [
            msgKey: keySCMsgAllDeleted,
            msg: XString.FromSTRING["Done.  Deleted all words in Edit dictionaries."L],
            translatable: TRUE,
            id: 51],
         scMsgNDeleted: [
            msgKey: keySCMsgNDeleted,
            msg: XString.FromSTRING["Done.  Words deleted: "L],
            translatable: TRUE,
            id: 52],
         scMsgMakeDocInProgress: [
            msgKey: keySCMsgMakeDocInProgress,
            msg: XString.FromSTRING["Creating document for user-dictionary ..."L],
            translatable: TRUE,
            id: 53],
         scMsgMakeDocComplete: [
            msgKey: keySCMsgMakeDocComplete,
            msg: XString.FromSTRING["Make Document operation completed."L],
            translatable: TRUE,
            id: 54],
         scMsgSelectALexicon: [
            msgKey: keySCMsgSelectALexicon,
            msg: XString.FromSTRING["Can't Make Document: First select a private dictionary."L],
            translatable: TRUE,
            id: 55],
         scMsgLexiconIsEmpty: [
            msgKey: keySCMsgLexiconIsEmpty,
            msg: XString.FromSTRING["That dictionary is empty; Make Document operation cancelled."L],
            translatable: TRUE,
            id: 56],
         scMsgCantOpenLexicon: [
            msgKey: keySCMsgCantOpenLexicon,
            msg: XString.FromSTRING["Can't open that dictionary; Make Document operation cancelled."L],
            translatable: TRUE,
            id: 57],
         scMsgAlternativesWordTooLong: [
            msgKey: keySCMsgAlternativesWordTooLong,
            msg: XString.FromSTRING["That word is too long."L],
            translatable: TRUE,
            id: 58],
         scMsgUnknownError: [
            msgKey: keySCMsgUnknownError,
            msg: XString.FromSTRING["Unknown Spelling Checker error..."L],
            translatable: TRUE,
            id: 59],
         scMsgUnimplemented: [
            msgKey: keySCMsgUnimplemented,
            msg: XString.FromSTRING["That function is not currently available."L],
            translatable: TRUE,
            id: 60],
          scLexicon: [
             msgKey: keySCLexicon,
             msg: XString.FromSTRING["Spelling Checker Lexicon"L],
             id: 61],
          scMsgReadOnlyAccess: [
             msgKey: keySCMsgReadOnlyAccess,
             msg: XString.FromSTRING["Must be in edit mode when making corrections."L],
             id: 62],
          scMsgConfirm: [
             msgKey: keySCMsgConfirm,
             msg: XString.FromSTRING["Confirm"],
             id: 63],
          scMsgCancel: [
             msgKey: keySCMsgCancel,
             msg: XString.FromSTRING["Cancel"],
             id: 64]
          ];

      h ← XMessage.AllocateMessages[
         applicationName: "Spelling Checker"L,
         maxMessages: MsgKey.LAST.ORD.SUCC,
         clientData: NIL,
         proc: DeleteMessages];
      XMessage.RegisterMessages[
         h: h,
         messages: LOOPHOLE[LONG[DESCRIPTOR[msgArray]]],
         stringBodiesAreReal: FALSE]};

   Init[];

   END...
```

```
LOG  (date - person - action)
 9-Mar-85 10:23:56 - Marks - Create
31-Jul-85 15:08:15 - Marks - AR17732: Add keySCMsgConfirm and keySCMsgCancel: AR15416: keySCMsgConfirmCorrectionWasntApplied changed to
be understandable.
```

```
-- File: SpellingCheckerMsgFileImpl.mesa - last edit:
-- Marks.ES                1-May-85 10:41:25

-- Copyright (C) 1985 by Xerox Corporation. All rights reserved.

DIRECTORY
  ApplicationFolder USING [FindDescriptionFile, FromName],
  BWSFileTypes USING [systemFileCatalog],
  Catalog USING [Open],
  Heap USING [systemZone],
  NSFile USING [Close, Error, Find, GetReference,  Handle, nullHandle, nullReference, OpenByReference, Reference],
  NSString USING [FreeString, String],
  OptionFile USING [GetStringValue, GetWorkstationProfile],
  SpellingCheckerMessageDefs,
  XMessage USING [ClientData, FreeMsgDomainsStorage, Handle, MessagesFromReference, MsgDomains],
  XString USING [FromSTRING, NSStringFromReader, Reader, ReaderBody];

SpellingCheckerMsgFileImpl: PROGRAM
  IMPORTS ApplicationFolder, Catalog, Heap, NSFile, NSString, OptionFile, XMessage, XString
  EXPORTS SpellingCheckerMessageDefs = {

  -- Data

  h: XMessage.Handle ← NIL;

  localZone: UNCOUNTED ZONE ← Heap.systemZone;

  -- Procedures

  DeleteMessages: PROCEDURE [clientData: XMessage.ClientData] = {};

  GetHandle: PUBLIC PROCEDURE RETURNS [XMessage.Handle] = {RETURN[h]};

  InitMessages: PROCEDURE = {
    internalName: XString.ReaderBody ← XString.FromSTRING ["Spelling Checker"L];
    msgDomains: XMessage.MsgDomains ← NIL;
    msgDomains ← XMessage.MessagesFromReference [
      file: GetMessageFileRef [ApplicationFolder.FromName [@internalName]],
      clientData: NIL,
      proc: DeleteMessages ];
    h ← msgDomains[0].handle;
    XMessage.FreeMsgDomainsStorage [msgDomains];
    };

  GetMessageFileRef: PROCEDURE [folder: NSFile.Reference]
    RETURNS [msgFile: NSFile.Reference ← NSFile.nullReference] = {
    folderHandle: NSFile.Handle ← NSFile.nullHandle;
    adf: NSFile.Reference ← NSFile.nullReference;
    internalName: XString.ReaderBody ← XString.FromSTRING ["Spelling Checker"L];
    messageFile: XString.ReaderBody ← XString.FromSTRING ["MessageFile"L];

    FindMessageFileFromName: PROCEDURE [value: XString.Reader] = {
      nssName: NSString.String ← XString.NSStringFromReader [r: value, z: localZone];
      msgFileHandle: NSFile.Handle ← NSFile.nullHandle;
      -- We do NSFile.Find here in case the name has an asterisk in it.
          msgFileHandle ← NSFile.Find [directory: folderHandle,
        scope: [filter: [matches[attribute: [name[nssName]]]]] !
          NSFile.Error => {msgFileHandle ← NSFile.nullHandle; CONTINUE}];
      IF msgFileHandle = NSFile.nullHandle THEN ERROR; -- no message file!
      msgFile ← NSFile.GetReference [msgFileHandle];
      NSFile.Close [msgFileHandle];
      NSString.FreeString [z: localZone, s: nssName];
      };

    IF folder = NSFile.nullReference THEN {
      -- No application folder, so use the system catalog and the WorkstationProfile
      folderHandle ← Catalog.Open [BWSFileTypes.systemFileCatalog];
      adf ← OptionFile.GetWorkstationProfile []}
    ELSE {
      -- There was an application folder, so use the folder and the adf inside it.
      folderHandle ← NSFile.OpenByReference [folder];
      adf ← ApplicationFolder.FindDescriptionFile [folderHandle]};
    OptionFile.GetStringValue [section: @internalName,
      entry: @messageFile,
      callBack: FindMessageFileFromName,
      file: adf];
    NSFile.Close [folderHandle];
    };

  -- Mainline code

  InitMessages[];

  }...
LOG
1-May-85 10:40:54 - Marks - Create.
```

```
-- File: SpellingCheckerScanPack.mesa - last edit:
-- Bartlett:OSBU South:Xerox  2-Dec-87 13:27:55
-- Marks.ES            17-Nov-87  9:42:44
-- Lewis:OSBU South:Xerox 18-Mar-87 15:17:59
-- Maybury.ES           9-Apr-86 13:39:15
-- Walden.ES           28-Sep-84  9:13:33



DIRECTORY
   Attention USING [Clear, Post, PostAndConfirm],
   BlockFriendsDefs USING [InRedliningMode],
   CharDefs USING [Char, Chset, Code, Roman],
   Cursor USING [Set],
   DocumentDefs USING [DocFromIzn, Handle],
   DocUtilDefs USING [GetMainTextChainBlock, FreeSystemDoc, LockSystemDoc],
   DocWindowDefs USING [MakeCaretVisible],
   FormWindow USING [GetBooleanItemValue, GetChoiceItemValue, GetTextItemValue, SetTextItemValue],
   InstanceDefs USING [iznNil],
   LexiconDefs USING [
      AddWord, DeleteWord, LexiconHandle, LexiconIsFull, Lexicons, LookUpWord,
      GetNormalizedWord],
   ProductFactoring USING [Enabled],
   SchemaDefs USING [Lschema, lschemaNil],

   SpellingCheckerDefs USING [
      AddOrDelete, checkCtxt, ClearHostDocCs, GetActiveLexicons, Items,
      LookUpInPairList, nBytesLongestWord, ReplaceStartWithContinue,
      ScopeFromChoice, SCScanCtxt, SetDocContext, SetDocNotEdited,
      StartAlternativesGeneration, StopAlternativesGeneration,
      UpdateWordCountDisplay, UserEditedDoc],

   SpellingCheckerMessageDefs USING [
      GetHandle, keySCMsgConfirmDeleteAllWords, keySCMsgAllDeleted,
      keySCMsgNAdded, keySCMsgNDeleted, keySCMsgConfirmEditLexiconIsFull,
      keySCMsgConfirm, keySCMsgCancel, keySCMsgNoEditLexiconsSpecified,
      keySCMsgNeedAppropriateSelection, keySCMsgConfirmBatchAddWords,
      keySCMsgBatchAddingWords, keySCMsgUserCancelled, keySCMsgUnknownError,
      keySCMsgUserEditedDocCantContinue, keySCMsgCheckingSpelling,
      keySCMsgCheckingComplete, keySCMsgCorrectionTooLong],

   StarPFOptions USING [starSpelling],
   TextUtilDefs USING [
      AppendNumber, LptTextSegment, Rdr,TextSegmentFromXString],
   TIP USING [UserAbort],
   TxtBlockDefs USING [CopyBlockaddr, NewBlockaddr],
   TxtDefs USING [Blockaddr, Seldesc, TextSegment, textSegmentNil],
   TxtEditDefs USING [
      AlterCurrentSelection, AqTxtCtxt, BeginEdit, CopyTextSegment, CreateSeldesc,
      DestroyTextSegment, EndEdit, ReplaceWithString, SelDescription],
   TxtEditExtraIDefs USING [ReplaceWithStringForRedlining],
   TxtScanDefs USING [
      EnumScope, EnumerateStack, EnumStatus, EnumerateWords, PopAllEnum,
      PushEnumFromSelection, ReadonlyWordFlags, ScanCtxt, ScanWords, TokenProc,
      WIDEN, WordFlagsObject, WordProc, WordScanCtxtObject],
   UserTerminal USING [Beep],
   XChar USING [Character, Code, LowerCase, Make, UpperCase],
   XFormat USING [DecimalFormat],
   XMessage USING [Get, Handle],
   XString USING [
      AppendReader, Bytes, Character, CharacterLength, CopyToNewWriterBody,
      Dereference, emptyContext, First, FreeReaderBytes, FreeWriterBytes,
      InsufficientRoom, NewWriterBody, nullReaderBody,
      Reader, ReaderBody, ReaderFromWriter, Writer, WriterBody,
      WriterBodyFromSTRING];

SpellingCheckerScanPack: PROGRAM
   IMPORTS
      Attention, BlockFriendsDefs, CharDefs, Cursor, DocumentDefs, DocUtilDefs,
      DocWindowDefs, FormWindow, LexiconDefs, ProductFactoring,
      SpellingCheckerDefs, SpellingCheckerMessageDefs, TIP, TxtBlockDefs, TxtEditDefs,
      TxtEditExtraIDefs, TxtScanDefs, TextUtilDefs, UserTerminal, XChar,
      XMessage, XString
   EXPORTS SpellingCheckerDefs
   SHARES SchemaDefs, XString =
   BEGIN
   OPEN FW: FormWindow, SchemaDefs, SpellingCheckerDefs,
      SpellingCheckerMessageDefs, TIP, TxtDefs, TxtScanDefs,
      TextUtilDefs, XM: XMessage, XS: XString;

   Bug: SIGNAL [Bugtype] = CODE;
   Bugtype: TYPE = {impossible, unimplemented};

   --TYPES

   -- CONSTANTS
   h: XM.Handle = SpellingCheckerMessageDefs.GetHandle[];

   -- VARIABLES
   myZone: UNCOUNTED ZONE ← NIL;

   -- PROCEDURES

   --temp, to avoid 'from' being normalized
   AppendReader: PROC [to: XS.Writer, from: XS.Reader, extra: CARDINAL ← 0] =
      INLINE {
      rb: XS.ReaderBody ← XS.Dereference[from];
```

```
      XS.AppendReader[to: to, from: @rb, extra: extra];
      };

   ResetWriter: PROC [w: XS.Writer] = {
      IF w # NIL THEN {
         w.context ← w.endContext ← XS.emptyContext; w.offset ← w.limit ← 0; };
      };

   AddOrDeleteWords: PUBLIC PROC [addOrDelete: AddOrDelete] = {
      OPEN c: checkCtxt;
      nAddedOrDeleted: CARDINAL ← 0;
      edit: LexiconDefs.Lexicons = GetActiveLexicons[].edit;
      tsBlock: TextSegment;
      string: XString.ReaderBody;
      lsTempNorm: LONG STRING = [nBytesLongestWord];
      normalizedWord: XS.WriterBody ← XS.WriterBodyFromSTRING[
         s: lsTempNorm, homogeneous: TRUE];
      nWord: XS.Reader = Rdr[@normalizedWord];
      tooLong: BOOL ← FALSE;

      AddOrDeleteProc: TxtScanDefs.WordProc = {
         scMsgConfirmEditLexiconIsFull: XS.ReaderBody ← XM.Get[h, keySCMsgConfirmEditLexiconIsFull];
         status ← done;
         LexiconDefs.GetNormalizedWord[
            word, wordFlags, @normalizedWord !
            XS.InsufficientRoom => GOTO tooLong];
         IF addOrDelete = add THEN {
            IF LexiconDefs.AddWord[
               edit, nWord, wordFlags !
               LexiconDefs.LexiconIsFull =>
                  IF Attention.PostAndConfirm[@scMsgConfirmEditLexiconIsFull].confirmed THEN
                     RESUME
                  ELSE {status ← abort; CONTINUE}].addedToAny
               THEN
               nAddedOrDeleted ← nAddedOrDeleted + 1;
            }
         ELSE {
            IF LexiconDefs.DeleteWord[edit, nWord, wordFlags].deletedFromAny THEN
               nAddedOrDeleted ← nAddedOrDeleted + 1;
            };
         EXITS tooLong => tooLong ← TRUE;
         };  --AddOrDeleteProc

      IF edit = NIL THEN {
         scMsgNoEditLexiconsSpecified: XS.ReaderBody ← XM.Get[
            h, keySCMsgNoEditLexiconsSpecified];
         Attention.Post[@scMsgNoEditLexiconsSpecified]; RETURN; };

      IF addOrDelete = delete THEN {  --first check for '*
         rb: XS.ReaderBody ← FormWindow.GetTextItemValue[
            c.fw, Items.correction.ORD, myZone];

         IF (XS.CharacterLength[@rb] = 1)
            AND (XS.First[@rb] = LOOPHOLE[CharDefs.Roman[asterisk], XS.Character])
            THEN {
            scMsgConfirmDeleteAllWords: XS.ReaderBody ← XM.Get[
               h, keySCMsgConfirmDeleteAllWords];
            scMsgConfirm: XS.ReaderBody ← XM.Get[h, keySCMsgConfirm];
            scMsgCancel: XS.ReaderBody ← XM.Get[h, keySCMsgCancel];
                --it's an *
            IF Attention.PostAndConfirm[s: @scMsgConfirmDeleteAllWords, confirmChoices: [@scMsgConfirm, @scMsgCancel]].confirmed
               THEN {
               scMsgAllDeleted: XS.ReaderBody ← XM.Get[h, keySCMsgAllDeleted];
               wordFlags: TxtScanDefs.WordFlagsObject;
               wordFlags.nChars ← LAST[CARDINAL];
               [] ← LexiconDefs.DeleteWord[edit, NIL, @wordFlags];
               c.invalidAltList ← TRUE;
               UpdateWordCountDisplay[checkCtxt];
               Attention.Post[@scMsgAllDeleted];
               };
            XS.FreeReaderBytes[@rb, myZone];
            RETURN;
            }  --its an *
         ELSE XS.FreeReaderBytes[@rb, myZone];
         };  --addOrDelete = delete

      string ← FormWindow.GetTextItemValue[c.fw, Items.correction.ORD, myZone];
      << The scratchBlock was created in the SystemDoc by
         SpellingCheckerWnPack.Init.  Since the block resides in the SystemDoc,
         we must prohibit every other editing action in that doc until we're
         through. >>
      DocUtilDefs.LockSystemDoc[];
      {ENABLE UNWIND => DocUtilDefs.FreeSystemDoc[];
      tsBlock ← TextUtilDefs.TextSegmentFromXString[@string, c.scratchBlock];
      IF tsBlock # textSegmentNil THEN {
         normalizedWord.offset ← normalizedWord.limit ← 0;
         [] ← TxtScanDefs.ScanWords[
            range: tsBlock, wordProc: AddOrDeleteProc,
            nBytesLongestWord: nBytesLongestWord + 3,
            --allow extra 3 here so that we can detect when a word is present which is too long (and reject it)
            zone: c.zone];
         TxtEditDefs.DestroyTextSegment[@tsBlock];
         };
      };  -- ENABLE
      DocUtilDefs.FreeSystemDoc[];
      IF nAddedOrDeleted > 0 THEN {
         UpdateWordCountDisplay[checkCtxt]; c.invalidAltList ← TRUE; };
```

```
    IF (tooLong AND (nAddedOrDeleted = 0)) THEN {
       scMsgCorrectionTooLong: XS.ReaderBody ← XM.Get[h, keySCMsgCorrectionTooLong];
       Attention.Post[@scMsgCorrectionTooLong]
       }
    ELSE {
       wb: XS.WriterBody ← XS.NewWriterBody[40, c.zone];
       r: XS.Reader;
       message: XS.ReaderBody ← XM.Get[h, (IF addOrDelete = add THEN keySCMsgNAdded ELSE keySCMsgNDeleted)];
       XS.AppendReader[@wb, @message];
       TextUtilDefs.AppendNumber[
          @wb, LONG[INTEGER[nAddedOrDeleted]], XFormat.DecimalFormat];
       r ← XS.ReaderFromWriter[@wb];
       Attention.Post[r];
       XS.FreeWriterBytes[@wb];
       };
    };  -- AddOrDeleteWords

ScanBatchCheckAndAdd: PUBLIC PROC [lookUp, edit: LexiconDefs.Lexicons] = {
   OPEN c: checkCtxt;
   nProcessed, nAdded: CARDINAL ← 0;
   enumStatus: EnumStatus;
   sc: TxtScanDefs.WordScanCtxtObject;
   lsTemp: LONG STRING = [nBytesLongestWord];
   lsTempNorm: LONG STRING = [nBytesLongestWord + 3];
   normalizedWord: XS.WriterBody ← XS.WriterBodyFromSTRING[
      s: lsTempNorm, homogeneous: TRUE];
   nWord: XS.Reader = Rdr[@normalizedWord];
   includeFrames: BOOLEAN ← FW.GetBooleanItemValue[c.fw, Items.checkFrames.ORD];

   tokenProc: TokenProc = {
      scMsgConfirmEditLexiconIsFull: XS.ReaderBody ← XM.Get[
         h, keySCMsgConfirmEditLexiconIsFull];
      IF TIP.UserAbort[c.sws] THEN GOTO aborted;
      WITH tk: token SELECT FROM
         word => {
            LexiconDefs.GetNormalizedWord[tk.sWord, tk.wordFlags, @normalizedWord];
            nProcessed ← nProcessed + 1;
            IF lookUp = NIL
            OR ~LexiconDefs.LookUpWord[lookUp, nWord, tk.wordFlags].found
               THEN {
               [] ← LexiconDefs.AddWord[edit, nWord, tk.wordFlags !LexiconDefs.LexiconIsFull =>
                     IF Attention.PostAndConfirm[@scMsgConfirmEditLexiconIsFull].confirmed THEN
                        RESUME
                     ELSE GOTO aborted;
                     ];
               nAdded ← nAdded + 1;
               };
            };
         ENDCASE;

      EXITS aborted => status ← abort;
      };  --tokenProc

   sc ← [
      scanCtxtObject: [
        textEnum: TxtScanDefs.EnumerateWords,
        tokenProc: tokenProc, z: c.zone,
        includeAnchoredFrames: includeFrames,
        includeCaptions: includeFrames],
      word: XS.WriterBodyFromSTRING[s: lsTemp, homogeneous: TRUE]];

   IF TxtScanDefs.PushEnumFromSelection[
      sc: WIDEN[@sc],
      scope: ScopeFromChoice[FW.GetChoiceItemValue[c.fw, Items.scope.ORD]]].doc
      = NIL THEN {
         scMsgNeedAppropriateSelection: XS.ReaderBody ← XM.Get[h, keySCMsgNeedAppropriateSelection];
         Attention.Post[@scMsgNeedAppropriateSelection];
         RETURN; };
   {
   scMsgConfirmBatchAddWords: XS.ReaderBody ← XM.Get[h, keySCMsgConfirmBatchAddWords];
   IF ~Attention.PostAndConfirm[@scMsgConfirmBatchAddWords].confirmed
      THEN {PopAllEnum[WIDEN[@sc]]; RETURN; };
   };

   --ok, nothing else to go wrong...
   Cursor.Set[hourGlass];
   {scMsgBatchAddingWords: XS.ReaderBody ← XM.Get[h, keySCMsgBatchAddingWords];
   Attention.Post[@scMsgBatchAddingWords];
   };
   StopAlternativesGeneration[];
   normalizedWord.offset ← normalizedWord.limit ← 0;
   enumStatus ← TxtScanDefs.EnumerateStack[WIDEN[@sc]];

   IF nAdded > 0 THEN UpdateWordCountDisplay[checkCtxt];

   {
   wb: XS.WriterBody ← XS.NewWriterBody[40, c.zone];
   r: XS.Reader;
   SELECT enumStatus FROM
      done => {
         scMsgCheckingComplete: XS.ReaderBody ← XM.Get[h, keySCMsgCheckingComplete];
         XS.AppendReader[@wb, @scMsgCheckingComplete];
         };
      abort => {
         scMsgUserCancelled: XS.ReaderBody ← XM.Get[h, keySCMsgUserCancelled];
         XS.AppendReader[@wb, @scMsgUserCancelled];
         };
```

```
            ENDCASE;
         TextUtilDefs.AppendNumber[
            @wb, LONG[INTEGER[nProcessed]], XFormat.DecimalFormat];
         r ← XS.ReaderFromWriter[@wb];
         Attention.Post[r];
         XS.FreeWriterBytes[@wb];
         };
         Cursor.Set[textPointer];
         };  --ScanBatchCheckAndAdd

      StartChecking: PUBLIC PROC [lookUp: LexiconDefs.Lexicons] = {
         OPEN c: checkCtxt;
         doc: DocumentDefs.Handle;
         scSC: SCScanCtxt = @c.scScanCtxtObject;

         -- doc context must be nil already
         IF
            (doc ← PushEnumFromSelection[
               sc: WIDEN[scSC],
               scope: ScopeFromChoice[
               FW.GetChoiceItemValue[c.fw, Items.scope.ORD]]].doc) = NIL THEN {
                  scMsgNeedAppropriateSelection: XS.ReaderBody ← XM.Get[
                     h, keySCMsgNeedAppropriateSelection];
                  Attention.Post[@scMsgNeedAppropriateSelection];
                  RETURN;
                  };

         SetDocContext[doc];
         scSC.nProcessed ← 0;

         WITH e: scSC.wordScanCtxtObject.scanCtxtObject.ec SELECT FROM
            text => scSC.wordScanCtxtObject.ts ← TxtEditDefs.CopyTextSegment[@e.range];
            ENDCASE => {
               ba, ba2: TxtDefs.Blockaddr;
               doc: DocumentDefs.Handle ← DocumentDefs.DocFromIzn[
               WITH e: scSC.wordScanCtxtObject.scanCtxtObject.ec SELECT FROM
                  graphicsFrame => e.frame.izn,
                  table => e.tableSchema.izn,
                  illustratorFrame => e.frame.izn,
                  frameCaptions => e.frame.izn,
                  ENDCASE => InstanceDefs.iznNil  -- shouldn't actually be using this -- ];
               ba ← TxtBlockDefs.NewBlockaddr[DocUtilDefs.GetMainTextChainBlock[doc, first], first];
               ba2 ← TxtBlockDefs.CopyBlockaddr[ba];
               scSC.wordScanCtxtObject.ts ← [ba, ba2];
               };

         ReplaceStartWithContinue[];
         ScanUtil[lookUp];
         };

      ContinueChecking: PUBLIC PROC [lookUp: LexiconDefs.Lexicons] = {
         OPEN c: checkCtxt;
         scSC: SCScanCtxt = @c.scScanCtxtObject;

         StopAlternativesGeneration[];
         SELECT TRUE FROM
            (scSC.wordScanCtxtObject.scanCtxtObject.ec = NIL) => {
               scMsgUnknownError: XS.ReaderBody ← XM.Get[h, keySCMsgUnknownError];
               Attention.Post[@scMsgUnknownError];
               RETURN};
            UserEditedDoc[c.doc] => {
               scMsgUserEditedDocCantContinue: XS.ReaderBody ← XM.Get[
                  h, keySCMsgUserEditedDocCantContinue];
               SetDocContext[NIL];  --replaces Continue cmd
               Attention.Post[@scMsgUserEditedDocCantContinue];
               RETURN;
               };
            ENDCASE => ScanUtil[lookUp];  --ok to proceed
         };

      -- interactive checking scan:
      ScanUtil: PROC [lookUp: LexiconDefs.Lexicons] = {
         OPEN c: checkCtxt;
         enumStatus: EnumStatus;
         scSC: SCScanCtxt = @c.scScanCtxtObject;
         correction: XS.WriterBody;
         replaceWord: BOOL ← FALSE;
         txtCtxt: TxtEditDefs.AqTxtCtxt;
         blockLastWordReplaced: SchemaDefs.Lschema ← SchemaDefs.lschemaNil;
         editsPending: BOOL ← FALSE;
         nWord: XS.Reader = Rdr[@scSC.normalizedWord];
         scMsgCheckingSpelling: XS.ReaderBody ← XM.Get[h, keySCMsgCheckingSpelling];
         includeFrames: BOOLEAN ← FW.GetBooleanItemValue[c.fw, Items.checkFrames.ORD];


         HandleMisspelling: PROC = {
            StartAlternativesGeneration[
              lookUp, Rdr[@scSC.wordScanCtxtObject.word],
              @scSC.wordScanCtxtObject.wordFlags];
            Attention.Clear[];
            SetDocNotEdited[c.doc];
            SelectAndScroll[@scSC.wordScanCtxtObject.ts];
            FW.SetTextItemValue[
               c.fw, Items.misspelling.ORD,
               XS.ReaderFromWriter[@scSC.wordScanCtxtObject.word]];
            FormWindow.SetTextItemValue[
               c.fw, Items.correction.ORD,
```

```
            XS.ReaderFromWriter[@scSC.wordScanCtxtObject.word]];
         UserTerminal.Beep[frequency: 300, duration: 200];

         --remember current value to detect changes
         IF c.lastCorrectionValue # XS.nullReaderBody THEN
            XS.FreeReaderBytes[@c.lastCorrectionValue, myZone];
         c.lastCorrectionValue ← FormWindow.GetTextItemValue[
            c.fw, Items.correction.ORD, myZone];

         };

      ReplaceWord: PROC = {
         IF ~editsPending THEN {
            TxtEditDefs.BeginEdit[@txtCtxt]; editsPending ← TRUE; }
         ELSE  --see if block has changed; if so, EndEdit/BeginEdit to prevent editing too many chains; it is more efficient to EndEdit
         after each block anyway
            IF scSC.wordScanCtxtObject.block # blockLastWordReplaced THEN {
               TxtEditDefs.EndEdit[@txtCtxt]; TxtEditDefs.BeginEdit[@txtCtxt]; };

         blockLastWordReplaced ← scSC.wordScanCtxtObject.block;

         IF BlockFriendsDefs.InRedliningMode[] THEN
            TxtEditExtra1Defs.ReplaceWithStringForRedlining[
            @txtCtxt, @scSC.wordScanCtxtObject.ts, XS.ReaderFromWriter[@correction]]
         ELSE TxtEditDefs.ReplaceWithString[
            @txtCtxt, @scSC.wordScanCtxtObject.ts,
            XS.ReaderFromWriter[@correction]];
         XS.FreeWriterBytes[@correction];
         };

      tokenProc: TokenProc = {
         WITH tk: token SELECT FROM
            word => {
               found: BOOL;

               IF TIP.UserAbort[c.sws] THEN RETURN[status: abort];

               LexiconDefs.GetNormalizedWord[
                  tk.sWord, tk.wordFlags, @scSC.normalizedWord];
               scSC.nProcessed ← scSC.nProcessed + 1;
               found ← LexiconDefs.LookUpWord[lookUp, nWord, tk.wordFlags].found;
               IF ~found THEN {  --not found
                  ignored: ARRAY [0..1) OF LexiconDefs.LexiconHandle ← [
                     c.ignoreList];
                  IF LexiconDefs.LookUpWord[
                     DESCRIPTOR[ignored], nWord, tk.wordFlags].found THEN
                     RETURN[status: done];

                  status ← pause;
                  IF scSC.autoCorrect THEN {
                     temp: XS.ReaderBody;
                     tempFlags: TxtScanDefs.ReadonlyWordFlags;

                     [found, temp, tempFlags] ← LookUpInPairList[
                        list: c.correctionList, word: nWord,
                        wordFlags: tk.wordFlags];
                     IF found THEN {
                        correction ← XS.CopyToNewWriterBody[@temp, c.zone];
                        SetNewStringCaseFromOldWord[
                           oldWord: tk.sWord, oldWordFlags: tk.wordFlags,
                           newString: @correction, newStringFlags: tempFlags];
                        replaceWord ← TRUE;
                        }
                     ELSE replaceWord ← FALSE;
                     };  --IF scSC.autoCorrect
                  };  -- IF ~found
               };
            ENDCASE;
         };  -- tokenProc

      --check these every time so they can be turned off and on while a range is being checked
      scSC.autoCorrect ← FW.GetBooleanItemValue[c.fw, Items.autoCorrect.ORD];
      scSC.wordScanCtxtObject.scanCtxtObject.includeAnchoredFrames ← includeFrames;
      scSC.wordScanCtxtObject.scanCtxtObject.includeCaptions ← includeFrames;

      --assign each time since it's a local proc
      scSC.wordScanCtxtObject.scanCtxtObject.tokenProc ← tokenProc;

      ClearHostDocCs[];
      -- unconditionally clear the cs if in the host doc
      -- MUST do this if auto-correcting, so that cs isn't invalidated by edit
      -- if status below is pause, this will happen anyway and won't be wasted effort; if abort or done, we want to do this to avoid
      surprise operation if Start is immediately bugged again (this has happened)

      Attention.Post[@scMsgCheckingSpelling];
      DO
         enumStatus ← TxtScanDefs.EnumerateStack[WIDEN[scSC]];
         IF (enumStatus = pause) AND replaceWord THEN ReplaceWord[] ELSE EXIT;
         ENDLOOP;

      IF editsPending THEN TxtEditDefs.EndEdit[@txtCtxt];
      IF enumStatus = pause THEN HandleMisspelling[]
      ELSE {
         wb: XS.WriterBody ← XS.NewWriterBody[40, c.zone];
         r: XS.Reader;
         --all enumerators have been popped
         SetDocContext[NIL];  --replaces Continue cmd
```

```
            IF enumStatus = abort THEN {
               scMsgUserCancelled: XS.ReaderBody ← XM.Get[h, keySCMsgUserCancelled];
               XS.AppendReader[@wb, @scMsgUserCancelled];
               }
            ELSE {
               scMsgCheckingComplete: XS.ReaderBody ← XM.Get[h, keySCMsgCheckingComplete];
               XS.AppendReader[@wb, @scMsgCheckingComplete];
               };
            TextUtilDefs.AppendNumber[
               @wb, LONG[INTEGER[scSC.nProcessed]], XFormat.DecimalFormat];
            r ← XS.ReaderFromWriter[@wb];
            Attention.Post[r];
            XS.FreeWriterBytes[@wb];
            };
         }; --ScanUtil

      SelectAndScroll: PROC [ts: LptTextSegment] = {
         seldescNew: TxtDefs.Seldesc;

         -- Set the selection and display properly.
         seldescNew ← TxtEditDefs.CreateSeldesc[
            TxtEditDefs.SelDescription[
            segment, segment[TxtEditDefs.CopyTextSegment[ts]]]];
         TxtEditDefs.AlterCurrentSelection[seldescNew];

         DocWindowDefs.MakeCaretVisible[percentFromLeft: 10, percentFromTop: 10];
         };

      SetNewStringCaseFromOldWord: PROC [
         oldWord: XS.Reader, oldWordFlags: TxtScanDefs.ReadonlyWordFlags,
         newString: XS.Writer, newStringFlags: TxtScanDefs.ReadonlyWordFlags] = {
         newOffset: CARDINAL = newString.offset;
         newBytes: XS.Bytes = newString.bytes;

         SetCaseOnNChars: PROC [
            start: CARDINAL ← 0, n: CARDINAL, case: {upper, lower}] = {
            chset: CharDefs.Chset ← newString.context.prefix;
            code: CharDefs.Code;
            xchar: XChar.Character;
            offset: CARDINAL ← newOffset;

            THROUGH [0..start) DO
               code ← newBytes[offset];
               IF code = 377B THEN {
                  chset ← newBytes[offset + 1]; offset ← offset + 3; }
               ELSE offset ← offset + 1;
               ENDLOOP;

            THROUGH [0..n) DO
               code ← newBytes[offset];
               IF code = 377B THEN {
                  chset ← newBytes[offset + 1];
                  offset ← offset + 2;
                  code ← newBytes[offset];
                  };
               xchar ← XChar.Make [set: chset, code: code];
               newBytes[offset] ← XChar.Code [
                  (IF case = upper THEN XChar.UpperCase[xchar]
                   ELSE XChar.LowerCase[xchar]) ];
               offset ← offset + 1;
               ENDLOOP;
            };

         SELECT oldWordFlags.case FROM
            allLower =>
               SELECT newStringFlags.case FROM
                  allLower => NULL;
                  firstCharUpperOnly => SetCaseOnNChars[n: 1, case: lower];
                  allUpper => SetCaseOnNChars[n: newStringFlags.nChars, case: lower];
                  ENDCASE;  --mixed case, leave new string as is

            firstCharUpperOnly =>
               SELECT newStringFlags.case FROM
                  allLower => SetCaseOnNChars[n: 1, case: upper];
                  firstCharUpperOnly => NULL;
                  allUpper =>
                     SetCaseOnNChars[
                        start: 1, n: newStringFlags.nChars - 1, case: lower];
                  ENDCASE;  --mixed case, leave new string as is

            allUpper =>
               SELECT newStringFlags.case FROM
                  allLower => SetCaseOnNChars[n: newStringFlags.nChars, case: upper];
                  firstCharUpperOnly =>
                     SetCaseOnNChars[
                        start: 1, n: newStringFlags.nChars - 1, case: upper];
                  allUpper => NULL;
                  ENDCASE;  --mixed case, leave new string as is

            ENDCASE;  --mixed case, leave new string as is
         };


   -- MAINLINE

   << Initialize SC tool context if the SC is installed in the product >>
   IF ProductFactoring.Enabled[StarPFOptions.starSpelling] THEN
```

```
        BEGIN

        myZone ← checkCtxt.zone;

        checkCtxt.scScanCtxtObject ← [
           wordScanCtxtObject: [
             scanCtxtObject: [
               textEnum: TxtScanDefs.EnumerateWords,  --never changes
               tokenProc: NIL,  --set in ScanInteractive
               z: checkCtxt.zone,
               callTokenProcOnTiles: FALSE],  --never changes
             word: XS.NewWriterBody[nBytesLongestWord, checkCtxt.zone],  --never changes
             setTs: TRUE],  --never changes
           normalizedWord: XS.NewWriterBody[nBytesLongestWord + 3, checkCtxt.zone]  --never changes
           ];

        checkCtxt.scScanCtxtObject.wordScanCtxtObject.word.zone ← NIL;  -- so that it never automatically grows
        END;


    END.  --  SpellingCheckerScanPack

LOG     (date - person - action)
 5-Jun-84 16:59:38 - Walden - OS5.0 release version 1
27-Jun-84 18:11:46
17-Jul-84 18:47:58 - D.J. Lewis - Tie initialization to FeatureDefs product factoring flag.
28-Sep-84  9:12:52 - Walden - Fix ScanUtil for AR 11447
 9-Mar-85 10:21:43 - Marks - Update to OS6.
12-Mar-85  9:48:52 - Marks - Replace RealToReaderBody with TextUtilDefs.AppendReader
21-Mar-85 15:53:43 - Marks - ReplaceWithString now imported from TxtEditDefs, not TextUtilDefs; MakeCaretVisible now gets 10%.
 1-Apr-85 14:43:31 - Marks - Use BWS product factoring.
22-May-85 15:44:35 - Marks - Change cursor to hourglass when doing Batch Check & Add.
15-Jul-85 14:49:01 - Marks - Correctly process the abort key.
31-Jul-85 15:21:05 - Marks - AR17732: When deleting all elements of lexicon and asking for confirmation choices should be Confirm/Cancel,
not Yes/No.
21-Feb-86 12:58:35 - Bartlett - use LockSystemDoc, FreeSystemDoc
 9-Apr-86 13:17:30 - Maybury - MAIN BODY, ScanBatchCheckAndAdd, StartChecking, ContinueChecking, ScanUtil, HandleMisspelling,
 ReplaceWord, ScanBatchCheckAndAdd: conform to new SCScanCtxtObject
11-Dec-86 - Lewis - Get Upper/LowerCase operators from XCharProps.
18-Mar-87 - Lewis - XCharPropsImpl$Upper/LowerCase now accessed through XChar.
 1-May-87 10:16:01 - Marks - use BlockFriendsDefs instead of BlockPrivExtralDefs.
 7-Aug-87 13:38:30 - Marks - AR10720: set includeCaptions = includeAnchoredFrames
16-Nov-87 15:59:05 - Marks - AR15704: in order to allow init selection to be a frame must use alternative method to allocate
scSC.wordScanCtxtObject.ts in StartChecking
 2-Dec-87 13:27:46 - Bartlett - add UNWIND call to FreeSystemDoc for AR 16204
```

```
-- File: SpellingCheckerUtilPack.mesa - last edit:
-- Walden              27-Mar-87 17:36:42
-- Maybury.ES           9-Apr-86 13:43:26
-- Marks.ES            12-Sep-85 14:43:11

-- Copyright (C) 1985, 1986, 1987 by Xerox Corporation. All rights reserved.

DIRECTORY
  ApplicationFolder USING [FindDescriptionFile, FromName],
  BWSZone USING [shortLifetime],
  CharDefs USING [chsetRoman, Roman],
  DocInterchangeDefs USING [AppendNewParagraph, AppendText, Doc, DocObject, FinishCreation, StartCreation],
  DocSpecialDefs USING [ClearClientsEditedFlag, EditedFlagID, GetClientsEditedFlag],
  DocumentDefs USING [DocFromIzn, Handle ],
  DocUtilDefs USING [subtypeBlankDoc],
  HashTableLexiconDefs USING [fileType, SpaceInadequateCannotEnumerate],
  HashTableTempLexiconDefs USING [
    OpenTemp, CloseTemp, AddTempWord, LookUpTempWord, Value, ValueFreeProc],
  LexiconDefs USING [CreateLexicon, fileAttrType, GetFileAttrs, IsCurrent, LexiconFileAttrs, LexiconHandle, LexiconIsFull, LexiconType,
  WordProc],
  NSFile USING [Attribute, Attributes, AttributesProc, AttributesRecord, ChangeAttributes, ClearAttributes, Close, Delete, Error,
  ExtendedAttributeType, Filter, GetAttributesByName, GetReference, Handle, List, Move, nullHandle, nullReference, OpenByReference,
  Reference, Type, Words],
  NSString USING [FreeString, String, StringFromMesaString],
  OptionFile USING [EntryEnumProc, EnumerateEntries, GetStringValue],
  <<PCLexiconFileTypeDefs USING [USEnglishSystemLexicon],
  PCUtilityDefs USING [CheckLexiconFiles].>>
  ProductFactoring USING [Enabled],
  Prototype USING [Find],
  PrototypeExtra USING [Add],
  SchemaDefs USING [GetRootCs, Lschema, lschemaNil],
  SelectionDefs USING [DeselectCs],
  SpellingCheckerDefs USING [checkCtxt, CheckCtxt, ClearFdbkParms, DisplayAttrs, displayAttrType, LexiconDataList, LexiconDisplayAttrs,
  LexiconList, LptLexiconData, LptLexiconDataList, ReplaceContinueWithStart, SCScanCtxt, StopAlternativesGeneration, WnDisplayAttrs],
  SpellingCheckerMessageDefs USING [GetHandle, keySCEmptyLexicon, keySCLexiconFolder, keySCWordsListedInDoc],
  StarDesktop USING [AddReferenceToDesktop, GetCurrentDesktopFile],
  StarFileTypeDefs USING [folder],
  StarPFOptions USING [starSpelling],
  System USING [gmtEpoch, GreenwichMeanTime],
  ToolUtilitiesDefs USING [LegalScWnFromSrt],
  TxtDefs USING [textSegmentNil],
  TxtEditDefs USING [ClearTextSegment],
  TxtScanDefs USING [PopAllEnum, ReadonlyWordFlags, ScanCtxt, WIDEN, WordFlagsObject],
  TextUtilDefs USING [BytesSize, CopyToBytes, Rdr, ResetWriter],
  XMessage USING [Get, Handle],
  XFormat USING [Number, Object, UnsignedDecimalFormat, WriterObject],
  XString USING [Bytes, ByteSequence, Character, CopyReader, emptyContext,
    FreeReaderBytes, FromChar, FromNSString, FromSTRING, NSStringFromReader, Reader,
    ReaderBody, unknownContext, vanillaContext, WriterBody, WriterBodyFromSTRING],
  XTime USING [Append, Current];

SpellingCheckerUtilPack: PROGRAM
  IMPORTS ApplicationFolder, BWSZone, CharDefs, DocInterchangeDefs, DocSpecialDefs, DocumentDefs, HashTableLexiconDefs,
  HashTableTempLexiconDefs, LexiconDefs, NSFile, NSString, <<PCUtilityDefs, >>OptionFile, ProductFactoring, Prototype, PrototypeExtra,
  SchemaDefs, SelectionDefs, SpellingCheckerDefs, SpellingCheckerMessageDefs, StarDesktop, ToolUtilitiesDefs, TxtEditDefs, TxtScanDefs,
  TextUtilDefs, XFormat, XMessage, XString, XTime
  EXPORTS SpellingCheckerDefs
  SHARES SelectionDefs, XString =
  BEGIN OPEN SpellingCheckerDefs, SpellingCheckerMessageDefs, TxtDefs, TxtScanDefs, TextUtilDefs, XF: XFormat, XS: XString;

  Bug: SIGNAL[Bugtype] = CODE;
  Bugtype: TYPE = {impossible};

--TYPES
  MakeDocCtxt: TYPE = LONG POINTER TO MakeDocCtxtObject;
  MakeDocCtxtObject: PUBLIC TYPE = RECORD [
    fileDoc: NSFile.Handle,
    doc: DocInterchangeDefs.Doc,
    unused: ARRAY[0..16) OF WORD
    ]; -- concrete type cloned from TextUtilPack

  WordItem: TYPE = LONG POINTER TO WordItemObject;
  WordItemObject: TYPE = RECORD[
    bytes: XS.Bytes,
    length: CARDINAL,
    flagsObject: TxtScanDefs.WordFlagsObject];

- CONSTANTS
  amerEngSysLexName: NSString.String = NSString.StringFromMesaString["AmericanEnglish"L];
  h: XMessage.Handle = SpellingCheckerMessageDefs.GetHandle[];
  newLineChar: XS.Character ← LOOPHOLE[CharDefs.Roman[newLine]];
  rbNewLine: XS.ReaderBody ← XS.FromChar[@newLineChar];
  scEditedFlagID: DocSpecialDefs.EditedFlagID = 0;
  shortZone: UNCOUNTED ZONE = BWSZone.shortLifetime;

-- VARIABLES
-- PROCEDURES

  ClearHostDocCs: PUBLIC PROC = {
    OPEN c: checkCtxt;
    -- Get lschema from doc
    IF c.doc # NIL THEN {
       rootCs: SchemaDefs.Lschema ← SchemaDefs.GetRootCs[].lschema;
       IF rootCs # SchemaDefs.lschemaNil
          AND DocumentDefs.DocFromIzn[rootCs.izn] = c.doc THEN
       SelectionDefs.DeselectCs[];
       };
```

```
    };

  CloseLexicons: PUBLIC PROC RETURNS [sysLexSelectedForLookup: BOOL ← FALSE] = {
    OPEN c: checkCtxt;
    FOR i: CARDINAL IN [0..c.lexicons.nLexicons) DO
      IF c.lexicons[i].systemLex AND c.lexicons[i].lookUp THEN
        --***TTT
        sysLexSelectedForLookup ← TRUE;

      IF c.lexicons[i].lexicon # NIL THEN
        CloseLexicon[@c.lexicons[i]];
      ENDLOOP;
    };

  CloseLexicon: PROC [lexData: LptLexiconData] = {
    displayAttributes: DisplayAttrs.lexicon;
    attrList: ARRAY [0..1) OF NSFile.Attribute ← [
      [extended[type: SpellingCheckerDefs.displayAttrType,
       value: DESCRIPTOR[@displayAttributes,
         SIZE [DisplayAttrs.lexicon], WORD]]]];

    lexData.lexicon.close[lexData.lexicon];
    lexData.lexicon ← NIL;

    IF ~lexData.systemLex THEN {
      --set display attributes here
      displayAttributes ← [var: lexicon[
        selectedForLookUp: lexData.lookUp,
        selectedForEdit: lexData.edit]];
      NSFile.ChangeAttributes[lexData.file, DESCRIPTOR[attrList]];
      };
    NSFile.Close[lexData.file];
    lexData.file ← NSFile.nullHandle;
    };

  --assumes all lexicons are properly stamped...
  CreateLexiconLists: PUBLIC PROC [systemLexSelectedForLookUp: BOOL, c: SpellingCheckerDefs.CheckCtxt] = {
    internalName: XString.ReaderBody ← XString.FromSTRING["Spelling Checker"L];
    applFolderRef: NSFile.Reference = ApplicationFolder.FromName [@internalName];
    applFolder: NSFile.Handle;
    nAllocated: CARDINAL ← 10;
    nLexicons: CARDINAL ← 0;
    lexicons: LptLexiconDataList ← c.zone.NEW[LexiconDataList[nAllocated]];
    system: BOOL ← FALSE;
    extendedSelections: ARRAY[0..2) OF NSFile.ExtendedAttributeType ← [lexiconDefs.fileAttrType, SpellingCheckerDefs.displayAttrType];
    adf: NSFile.Reference;
    sectionName: XString.ReaderBody ← XString.FromSTRING ["System Lexicons"L];

    AddSystemLexiconFromEntryName: OptionFile.EntryEnumProc = {
      -- internal name = entry name
      internalNameNS: NSString.String = XString.NSStringFromReader[entry, BWSZone.shortLifetime];

      AddSystemLexiconFromUserName: PROCEDURE [value: XString.Reader] = {
        attributes: NSFile.AttributesRecord ← TRASH;
        NSFile.GetAttributesByName[
          directory: applFolder,
          path: internalNameNS,
          selections: [interpreted: [fileID: TRUE, service: TRUE],
            extended: DESCRIPTOR[extendedSelections]],
          attributes: @attributes
          ! NSFile.Error => GOTO SkipThisOne];

        AddLexicon[attributes: @attributes, userName: value, system: TRUE];

        NSFile.ClearAttributes[@attributes];

        EXITS
          SkipThisOne => NULL;
        };  --AddSystemLexiconFromUserName

      OptionFile.GetStringValue[
        section: @sectionName, entry: entry, callBack: AddSystemLexiconFromUserName,
          index: 0, file: adf];
      NSString.FreeString[BWSZone.shortLifetime, internalNameNS];
      };  --AddSystemLexiconFromEntryName

    -- needs attributes needed by AddLexicon, plus 'name'
    userCtnrListProc: NSFile.AttributesProc = {
      userName: XString.ReaderBody ← XString.FromNSString[attributes.name];
      AddLexicon[attributes: attributes, userName: @userName, system: FALSE]};

    -- needs fileID, service, and extended selections:
    AddLexicon: PROC [attributes: NSFile.Attributes, userName: XString.Reader, system: BOOL] = {
      fileAttrs: LONG POINTER TO LexiconDefs.LexiconFileAttrs;
      displayAttrs: LexiconDisplayAttrs;

      IF attributes.extended = NIL OR attributes.extended[0].value = NIL THEN RETURN;

      fileAttrs ← LOOPHOLE[BASE[attributes.extended[0].value]];
      displayAttrs ← IF (attributes.extended[1].value = NIL)
        OR LENGTH [attributes.extended[1].value] # SIZE[DisplayAttrs.lexicon] THEN NIL
      ELSE LOOPHOLE[BASE[attributes.extended[1].value]];

      IF nLexicons = nAllocated THEN {
        tempLexicons: LptLexiconDataList ← c.zone.NEW[LexiconDataList[(nAllocated ← (nAllocated+5))]];
        FOR i: CARDINAL IN [0..nLexicons) DO
          tempLexicons[i] ← lexicons[i];
```

```
            ENDLOOP;
          c.zone.FREE[@lexicons];
          lexicons ← tempLexicons;
          };

        lexicons[nLexicons] ← [
          ref: [attributes.fileID, attributes.service],
          file: NSFile.nullHandle,
          name: XS.CopyReader[userName, c.zone],
          lexicon: NIL,
          nWords: 0,
          systemLex: system,
          lookUp: SELECT TRUE FROM
            system => systemLexSelectedForLookUp,
            (displayAttrs = NIL) => FALSE,
            ENDCASE => displayAttrs.selectedForLookUp,
          edit: SELECT TRUE FROM
            system, (displayAttrs = NIL) => FALSE,
            ENDCASE => displayAttrs.selectedForEdit
          ];

        nLexicons ← nLexicons + 1;
        }; --AddLexicon

      IF c.userLexiconCtnr # NSFile.nullHandle THEN
        NSFile.List[directory: c.userLexiconCtnr,
          proc: userCtnrListProc,
          scope: [filter: [equal[[type[value: HashTableLexiconDefs.fileType]]]]],
          selections: [
            interpreted: [fileID: TRUE, service: TRUE, name: TRUE],
            extended: DESCRIPTOR[extendedSelections]]];

      IF applFolderRef # NSFile.nullReference THEN {
        IF (applFolder ← NSFile.OpenByReference [applFolderRef]) # NSFile.nullHandle THEN {
          adf ← ApplicationFolder.FindDescriptionFile[applFolder];
          IF adf # NSFile.nullReference THEN
            OptionFile.EnumerateEntries[
              @sectionName, AddSystemLexiconFromEntryName, adf];

          NSFile.Close[applFolder];
          };
        };

      IF c.lexicons = NIL
      OR (c.lexicons.nLexicons # nLexicons) THEN {
        IF c.lexicons # NIL THEN DestroyLexiconLists[];
        c.lexicons ← c.zone.NEW[LexiconDataList[nLexicons]];
        c.lexiconsEdit ← c.zone.NEW[LexiconList[nLexicons]];
        c.lexiconsLookUp ← c.zone.NEW[LexiconList[nLexicons]];
        };

      FOR i: CARDINAL IN [0..nLexicons) DO
        c.lexicons[i] ← lexicons[i];
        ENDLOOP;

      c.zone.FREE[@lexicons];
      };

    CreatePairList: PUBLIC PROC [nPairsMax: CARDINAL, z: UNCOUNTED ZONE] RETURNS [LexiconDefs.LexiconHandle] = {
      RETURN [HashTableTempLexiconDefs.OpenTemp[
        nEntriesMax: nPairsMax,
        hashTableType: ordinary,
        sizeValues: SIZE[WordItemObject],
        valueFreeProc: PairListFreeProc,
        z: z]];
      };

    PairListFreeProc: HashTableTempLexiconDefs.ValueFreeProc = {
      wItem: WordItem = LOOPHOLE[value];
      --need clientCtxt for temp lexicons too
      checkCtxt.zone.FREE[@wItem.bytes];
      };

    DestroyPairList: PUBLIC PROC [list: LexiconDefs.LexiconHandle] = {
      HashTableTempLexiconDefs.CloseTemp[list];
      };

    LookUpInPairList: PUBLIC PROC [list: LexiconDefs.LexiconHandle, word: XS.Reader, wordFlags: TxtScanDefs.ReadonlyWordFlags] RETURNS
    [found: BOOL, associate: XS.ReaderBody, associateFlags: TxtScanDefs.ReadonlyWordFlags] = {
      wItem: WordItem;

      [found, wItem] ← HashTableTempLexiconDefs.LookUpTempWord[list, word, wordFlags];

      IF found THEN {
        associate ← [
          context: XS.vanillaContext,
          limit: wItem.length,
          offset: 0,
          bytes: wItem.bytes];
        associateFlags ← @wItem.flagsObject;
        };
      };

    AppendNWordsAndTime: PROC [doc: DocInterchangeDefs.Doc, nWords: LONG CARDINAL] = {
      tabChar: XS.Character ← LOOPHOLE[CharDefs.Roman[tab]];
      rb: XS.ReaderBody ← XS.FromChar[@tabChar];
      msgWordsListed: XS.ReaderBody ← XMessage.Get[h, keySCWordsListedInDoc];
```

```
        lsTemp: LONG STRING ← [100];
        wb: XS.WriterBody ← XS.WriterBodyFromSTRING[s: lsTemp, homogeneous: TRUE];
        time: System.GreenwichMeanTime ← XTime.Current[];

        -- put the current date/time in the document's header
        XTime.Append[@wb, time];
        DocInterchangeDefs.AppendText[[doc[doc]], TextUtilDefs.Rdr[@wb], XS.emptyContext];

        DocInterchangeDefs.AppendText[[doc[doc]], @rb, XS.emptyContext];
        DocInterchangeDefs.AppendText[[doc[doc]], @rb, XS.emptyContext];

        TextUtilDefs.ResetWriter[@wb];
        {o: XF.Object ← XF.WriterObject[@wb];
        XF.Number[@o, nWords, XF.UnsignedDecimalFormat];
        };

        DocInterchangeDefs.AppendText[[doc[doc]], Rdr[@wb], XS.emptyContext];
        DocInterchangeDefs.AppendText[[doc[doc]], @msgWordsListed, XS.emptyContext];

        };  -- AppendNWordsAndTime

      MakeDoc: PUBLIC PROC [lexicon: LexiconDefs.LexiconHandle, name: XString.Reader] RETURNS [ok: BOOL] = {
        fileDoc: NSFile.Handle;
        doc: DocInterchangeDefs.Doc;

      <<propose: sort TStrings (small, fixed size); the prefix is known to not change and be chsetRoman: the bytes are known to be constant,
      need to fix relationship with lexicon to keep this knowledge private to lexicon

        TString: TYPE = RECORD [offset, length: CARDINAL];
        >>

        IF (doc ← DocInterchangeDefs.StartCreation[].doc) = NIL THEN
           RETURN [ok: FALSE];

        AppendNWordsAndTime[doc, lexicon.getNWords[lexicon].nWordsCur];

        DocInterchangeDefs.AppendText[[doc[doc]], @rbNewLine, XS.emptyContext];

        AddWordsToDoc[lexicon, doc];
        fileDoc ← DocInterchangeDefs.FinishCreation[@doc].docFile;

        {nsName: NSString.String = XString.NSStringFromReader[name, shortZone];
        attrList: ARRAY [0..1) OF NSFile.Attribute ← [[name[value: nsName]]];
        refDoc: NSFile.Reference ← NSFile.GetReference[fileDoc];
        dtReference: NSFile.Reference = StarDesktop.GetCurrentDesktopFile[];
        dtHandle: NSFile.Handle ← NSFile.OpenByReference[dtReference];

        NSFile.Move[file: fileDoc, destination: dtHandle, attributes: DESCRIPTOR[attrList]];
        NSFile.Close[fileDoc];
        NSFile.Close[dtHandle];
        StarDesktop.AddReferenceToDesktop[reference: refDoc];
        NSString.FreeString[shortZone, nsName];
        };

        RETURN [ok: TRUE];
        };  --MakeDoc

      AddWordsToDoc: PROC [lexicon: LexiconDefs.LexiconHandle, doc: DocInterchangeDefs.Doc] = {
        -- we will sort ReaderBodies (more efficient to sort TStrings?)

        AddWordToDoc: LexiconDefs.WordProc = {
          DocInterchangeDefs.AppendNewParagraph[[doc[doc]]];
          DocInterchangeDefs.AppendText[[doc[doc]], word, XS.unknownContext];
          };

        lexicon.enumerate[lexicon, AddWordToDoc  -- enum, is supposed be in sort-order
          <<Someday we should inform the user if this happens:>>
          ! HashTableLexiconDefs.SpaceInadequateCannotEnumerate => RESUME];
        };

      SetDocContext: PUBLIC PROC [doc: DocumentDefs.Handle] = {
        OPEN c: checkCtxt;
        IF doc = c.doc THEN RETURN;

        StopAlternativesGeneration[];

        IF c.doc # NIL THEN {  -- clear old text context
          scSC: SCScanCtxt = @c.scScanCtxtObject;
          IF scSC.wordScanCtxtObject.scanCtxtObject.ec # NIL THEN {  --in the middle of scan
            TxtScanDefs.PopAllEnum[WIDEN[scSC]];
            };

          ReplaceContinueWithStart[];
          IF scSC.wordScanCtxtObject.ts # textSegmentNil THEN
            TxtEditDefs.ClearTextSegment[@scSC.wordScanCtxtObject.ts];

          ClearFdbkParms[];
          };

        c.doc ← doc;
        IF c.doc # NIL THEN SetDocNotEdited[c.doc];
        };

      SetDocNotEdited: PUBLIC PROC [doc: DocumentDefs.Handle] = {
        DocSpecialDefs.ClearClientsEditedFlag[doc, scEditedFlagID] };

      SetPair: PUBLIC PROC [list: LexiconDefs.LexiconHandle, word: XS.Reader, wordFlags: TxtScanDefs.ReadonlyWordFlags, associate: XS.Reader,
```

```
    associateFlags: TxtScanDefs.ReadonlyWordFlags] RETURNS [ok: BOOL ← TRUE] = {
      added: BOOL;
      wItem: WordItem;
      associateLength: CARDINAL = BytesSize[associate];
        --includes initial chset change if not Roman

      [added, LOOPHOLE[wItem, HashTableTempLexiconDefs.Value]] ← HashTableTempLexiconDefs.AddTempWord[list, word, wordFlags
      !LexiconDefs.LexiconIsFull => GOTO notOK];

      IF added THEN
        wItem.bytes ← checkCtxt.zone.NEW[XS.ByteSequence[associateLength]]
      ELSE {  --already there
        wordsAvail: CARDINAL = SIZE[XS.ByteSequence[wItem.length]];
        wordsNeeded: CARDINAL = SIZE[XS.ByteSequence[associateLength]];

        IF wordsAvail # wordsNeeded THEN {
          checkCtxt.zone.FREE[@wItem.bytes];
          wItem.bytes ← checkCtxt.zone.NEW[XS.ByteSequence[associateLength]];
          };
        };

      CopyToBytes[to: wItem.bytes, from: associate, n: associateLength];

      wItem.length ← associateLength;
      wItem.flagsObject ← associateFlags↑;
      ok ← TRUE;

      EXITS
        notOK => RETURN [ok: FALSE];
      };

    UserEditedDoc: PUBLIC PROC [doc: DocumentDefs.Handle] RETURNS [BOOL] = {
      RETURN[DocSpecialDefs.GetClientsEditedFlag[doc, scEditedFlagID]] };

  -- PRIVATE

    --all lexicons and files had better be closed
    DestroyLexiconLists: PROC = {
     OPEN c: checkCtxt;
      FOR i: CARDINAL IN [0..c.lexicons.nLexicons) DO
        XString.FreeReaderBytes[c.lexicons[i].name, c.zone];
        ENDLOOP;
      c.zone.FREE[@c.lexicons];
      c.zone.FREE[@c.lexiconsEdit];
      c.zone.FREE[@c.lexiconsLookUp];
      };

    OpenUserLexiconCtnr: PUBLIC PROC RETURNS [file: NSFile.Handle, lastModifiedOn: System.GreenwichMeanTime, dispAttrs: DisplayAttrs.wn] =
    {
      defaultAttrs: DisplayAttrs.wn = [
        var: wn[
          scWn: ToolUtilitiesDefs.LegalScWnFromSrt[[sc: [641, 22], rs: [0, 0]]],
          rsBody: [350, 200],
          scope: all,
          autoCorrect: FALSE,
          includeAnchoredFrames: FALSE,
          sysLexSelectedForLookUp: TRUE]];

      extendedAttrs: ARRAY [0..1) OF NSFile.ExtendedAttributeType ← [
        displayAttrType];

      found: BOOL ← FALSE;
      rbLexiconFolder: XS.ReaderBody ← XMessage.Get[h, keySCLexiconFolder];
      lexiconCtnrName: NSString.String = XS.NSStringFromReader[@rbLexiconFolder, checkCtxt.zone];
      filterList: ARRAY [0..2) OF NSFile.Filter ← [
        [equal[[name[lexiconCtnrName]]]],
        [equal[[type[StarFileTypeDefs.folder <<lexiconFolder>>]]]]
        ];
      dtReference: NSFile.Reference = StarDesktop.GetCurrentDesktopFile[];
      dtHandle: NSFile.Handle;

      ListProc: NSFile.AttributesProc = {
        file ← NSFile.OpenByReference[[attributes.fileID, attributes.service]];
        lastModifiedOn ← attributes.modifiedOn;
        dispAttrs ← IF (attributes.extended = NIL)
          OR (attributes.extended[0].value = NIL)
          OR (LENGTH[attributes.extended[0].value] # SIZE[DisplayAttrs.wn])  THEN defaultAttrs
        ELSE
          LOOPHOLE[BASE[attributes.extended[0].value], WnDisplayAttrs]↑;
        found ← TRUE;
        RETURN [continue: FALSE];
        };  --ListProc

      dtHandle ← NSFile.OpenByReference[dtReference];
      NSFile.List[directory: dtHandle,
        proc: ListProc,
        selections: [interpreted: [fileID: TRUE, service: TRUE, modifiedOn: TRUE], extended: DESCRIPTOR[extendedAttrs]],
        scope: [filter: [and[DESCRIPTOR[filterList]]]] ];
      NSFile.Close[dtHandle];

      IF ~found THEN {
        file ← NSFile.nullHandle;
        lastModifiedOn ← System.gmtEpoch;
        dispAttrs ← defaultAttrs;
        };

      NSString.FreeString[checkCtxt.zone, lexiconCtnrName];
```

```
    };

  Init: PROC = {
    IF ProductFactoring.Enabled[StarPFOptions.starSpelling] THEN
      BEGIN
        reference: NSFile.Reference;
        lex: NSFile.Handle;
        rbLexiconName: XS.ReaderBody ← XMessage.Get[h, keySCEmptyLexicon];
        emptyLexiconName: NSString.String = XS.NSStringFromReader[@rbLexiconName, shortZone];

        <<-- init system lexicons
        PCUtilityDefs.CheckLexiconFiles[amerEngSysLexName];
          --generalize for multiple system proximity-format lexicons>>

        --init prototype private lexicon
        reference ← Prototype.Find[
          type: HashTableLexiconDefs.fileType, version: 1, subtype: DocUtilDefs.subtypeBlankDoc];
        lex ←
          IF reference = NSFile.nullReference THEN NSFile.nullHandle
          ELSE NSFile.OpenByReference[reference: reference];
        IF lex = NSFile.nullHandle THEN {
          lex ← LexiconDefs.CreateLexicon[
            hashTable, NSFile.nullHandle, emptyLexiconName, 1000, TRUE];
          PrototypeExtra.Add[
            file: lex, version: 1, subtype: DocUtilDefs.subtypeBlankDoc];
          }
        ELSE
          IF ~LexiconDefs.IsCurrent[LexiconDefs.GetFileAttrs[lex]] THEN {
            NSFile.Delete[lex];
            lex ← LexiconDefs.CreateLexicon[
              hashTable, NSFile.nullHandle, emptyLexiconName, 1000, TRUE];
            PrototypeExtra.Add[
              file: lex, version: 1, subtype: DocUtilDefs.subtypeBlankDoc];
            };
        NSString.FreeString[shortZone, emptyLexiconName];
        IF lex # NSFile.nullHandle THEN NSFile.Close[lex];

      END;

    };



-- MAINLINE

  Init[];

  END.  --  SpellingCheckerUtilPack
```

```
LOG     (date - person - action)
 5-Jun-84 16:45:48 - Walden - OS5.0 release version 1
29-Jun-84 10:51:49 - Walden
17-Jul-84 19:14:49 - D.J. Lewis - Tie initialization to FeatureDefs product factoring flag.
26-Jul-84 12:11:08 - D.J. Lewis - Replace XTime with Star time service (MessageDefs.GetTimeMsg).
17-Aug-84 15:34:23 - Walden - Stop alternatives generation in SetDocContext - this SHOULD be done anyway, and if it isn't, causes process
interference in the unMONITORed PCMain lexicon.  This fixes lurking bug when document is closed or paginated or something moved/deleted
when in the middle of checking.
 9-Mar-85 10:22:45 - Marks - Update to OS6.
 9-Mar-85 10:22:45 - Marks - Fix UserEditedDoc, SetDocNotEdited with calls to DocSpecialDefs.
13-Mar-85 15:49:20 - Marks - Make sure that only one Empty Lex is added to prototype folder when needed and previous versions are
deleted.
30-Apr-85 11:14:38 - Marks - Get Dictionaries from Application Folder.
 7-May-85 14:49:38 - Marks - DocInterchangeDefs returns two values now.
 7-Aug-85 11:20:15 - Marks - AR 17975: CreateLexiconLists - when allocate larger data structure copy over data contained in it.
11-Sep-85 13:12:29 - Marks - AR 19933: MakeDoc - use default place.
18-Dec-85 16:35:40 - Maybury - AddWordToDoc: Using para. per word to improve MakeDoc performance.
19-Dec-85 16:08:47 - Maybury - AddWordsToDoc: Cease using GSortDefs (due to bugs in GSortPack; also for perf. gain); rely on
lexicon.enumerate to produce a SORTED enumeration.
30-Jan-86 11:44:07 - Maybury - Init: moved lexicon icon initialization to HashTableLexiconPack.
 3-Feb-86 13:57:22 - Maybury - Adopted HashTableLexiconDefs.fileType (vs. hackLexicon).  AddWordsToDoc: catch <<and RESUME without
comment>> SpaceInadequateCannotEnumerate.
 9-Apr-86 13:42:20 - Maybury - SetDocContext: conform to new SCScanCtxtObject.
16-Apr-86 16:31:21 - Walden - use PCLexiconFileTypeDefs.USEnglishSystemLexicon for file type of system USEnglish lexicon
```

```
-- File: SpellingCheckerWnPack.mesa - last edit:
-- Bartlett:OSBU South:Xerox  2-Dec-87 13:29:06
-- Marks.ES              5-Aug-87 14:14:28
-- Bartlett.ES          21-Feb-86 12:10:09
-- Walden.ES            16-Aug-84 14:42:02
-- Lewis                17-Jul-84 18:55:29

-- Copyright (C) 1986, 1987 by Xerox Corporation. All rights reserved.

-- items flagged with ***TTT will go away when the folder UI is done "right"

DIRECTORY
   Atom USING [ATOM, MakeAtom, null],
   Attention USING [AddMenuItem, Post, RemoveMenuItem],
   Context USING [Create, Find, NopDestroyProc, Type, UniqueType],
   DocEventDefs USING [
      AddDependency, AgentProcedure, Dependency, RemoveDependency],
   DocumentDefs USING [GetIzn, Handle],
   DocUtilDefs USING [FreeSystemDoc, GetSystemDoc, LockSystemDoc],
   Event USING [AddDependency, AgentProcedure],
   FormWindow USING [
      AppendItem, AppendLine, BooleanChangeProc, ChoiceChangeProc, ChoiceItem,
      DestroyItem, FreeTextHintsProc, GetBooleanItemValue, GetChoiceItemValue,
      GetTextItemValue, HasAnyBeenChanged, HasBeenChanged, ItemKey,
      LayoutProc, Line, MakeBooleanItem, MakeChoiceItem, MakeIntegerItem,
      MakeItemsProc, MakeTextItem, Repaint, ResetAllChanged, SetBooleanItemValue,
      SetChoiceItemValue, SetIntegerItemValue, SetTextItemValue, SetVisibility,
      TabStops, TextHintsProc, Visibility],
   InstanceDefs USING [rrefNil],
   Heap USING [Create, NWords],
   LexiconDefs USING [CreateLexicon, Lexicons, OpenLexicon],
   MenuData USING [CreateItem, ItemHandle, MenuProc],
   NSFile USING [
      Attribute, Close, Delete, GetDefaultSession, nullHandle, nullReference,
      OpenByReference, Session],
   NSFileExtra USING [ChangeAttributesPrivileged],
   NSString USING [String, StringFromMesaString],
   Process USING [Detach],
   ProductFactoring USING [Enabled],
   PropertySheet USING [Create, GetFormWindows, MenuItemProc],
   SpellingCheckerDefs USING [
      AddMenuCommands, AddPopupMenu, CheckCtxt, CheckCtxtObject, ChoicefromScope,
      CloseLexicons, CreateAlternativesProcess, DestroyAlternativesProcess,
      CreateLexiconLists, CreatePairList, DestroyPairList, DisplayAttrs,
      displayAttrType, LptLexiconData, OpenUserLexiconCtnr,
      PopUpMenuMakeStringProc, ScopeFromChoice, SetDocContext, WnDisplayAttrs],
   SpellingCheckerMessageDefs,
   StarDesktop USING [GetCurrentDesktopFile],
   StarPFOptions USING [starSpelling],
   StarWindowShell USING [GetAvailableBodyWindowDims, Handle, Pop,
      PoppedProc, Push, ShellFromChild, SleepOrDestroy],
   StarWindowShellExtra USING [SetSleeps],
   System USING [
      GetGreenwichMeanTime, gmtEpoch, GreenwichMeanTime, SecondsSinceEpoch],
   TxtBlockDefs USING [Create, CreateParms],
   Window USING [BitmapPlace, Dims, Handle, Place],
   XMessage USING [Get, Handle],
   XString USING [FreeReaderBytes, nullReaderBody, Reader, ReaderBody];

SpellingCheckerWnPack: MONITOR
   IMPORTS
      Atom, Attention, Context, DocEventDefs, DocumentDefs, DocUtilDefs, Event,
      FormWindow, Heap, LexiconDefs, MenuData, NSFile, NSFileExtra, NSString,
      Process, ProductFactoring, PropertySheet, SpellingCheckerDefs,
      SpellingCheckerMessageDefs, StarDesktop, StarWindowShell, StarWindowShellExtra, System,
      TxtBlockDefs, Window, XMessage, XString
   EXPORTS SpellingCheckerDefs
   SHARES MenuData =
   BEGIN
   OPEN FW: FormWindow, SpellingCheckerDefs,
      SpellingCheckerMessageDefs, XM: XMessage,
      XS: XString;

   ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
   -- TYPES
   ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----

   ScopeType: TYPE = {allText, remainingText, selectedText};

   ItemData: TYPE = LONG POINTER TO AqItemData;
   AqItemData: TYPE = RECORD [
      scope: ScopeType ← allText,
      checkFrames: BOOLEAN ← FALSE,
      autoCorrect: BOOLEAN ← FALSE,
      misspelling: XS.ReaderBody ← XS.nullReaderBody,
      correction: XS.ReaderBody ← XS.nullReaderBody];

   Items: TYPE = {scope, checkFrames, autoCorrect, misspelling, correction};


   ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
   -- CONSTANTS
   ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----

   SpellingCheckerCtxtType: Context.Type = Context.UniqueType[];
   h: XM.Handle = GetHandle[];
   initialStringLength: CARDINAL = 40;
```

```
    -- THIS IS A DEFINITE HACK!!! SHOULD BE IN A DEF; DEFINES NUMBER OF ITEMS IN PS BEFORE LEXICON INFO
    lastItem: CARDINAL = 5;
    size: Window.Dims = [480, 300];
    tabStopInterval: CARDINAL = 100;
    spaceAboveLine: CARDINAL = 5;


    ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
    -- GLOBAL VARIABLES
    ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----

    -- Only one instance of any option sheet can be on the screen at any given time, so it is safe to not put monitors around the data
    shared by the procedures used in this module.  The permanent data record is kept in the global frame.  It is borderline in size for
    what is recommended to be kept in a global frame, but I don't know how to store permanent data in hyperspace.

    checkCtxt: PUBLIC CheckCtxt ← NIL;
    paginationDependency: DocEventDefs.Dependency ← LOOPHOLE[LONG[NIL]];
    docDeleteDependency: DocEventDefs.Dependency ← LOOPHOLE[LONG[NIL]];
    docCloseDependency: DocEventDefs.Dependency ← LOOPHOLE[LONG[NIL]];
    fw: Window.Handle ← NIL;
    shell: StarWindowShell.Handle;
    myZone: UNCOUNTED ZONE ← NIL;

    ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
    -- SIGNALS
    ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
    Bug: SIGNAL [Bugtype] = CODE;
    Bugtype: TYPE = {impossible, badValue, error};

    ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
    -- PROCEDURES
    ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----

    BuildLexiconRow: PROC [
       -- builds and appends lexicon row.
       window: Window.Handle, wthRow: INTEGER, lex: LptLexiconData, currentLexicon: CARDINAL] = {
       lsvNWords: LONG STRING = [50];
       xcMinusNWordsInfo: INTEGER = wthRow - 50;  --allow space for nWords display
       notOpen: BOOL ← lex.lexicon = NIL;
       visibility: FW.Visibility ← IF notOpen THEN invisible ELSE visible;
       editVisibility: FW.Visibility ← IF lex.systemLex THEN invisibleGhost ELSE visible;
       -- kludge multiplication - because four entries in line
       nextItem: CARDINAL ← lastItem + (4*currentLexicon);


       leadingMargin: CARDINAL = 5;
       line: FW.Line;
       tabChoice: FW.TabStops = [fixed[tabStopInterval]];
       -- FW.SetTabStops[window: window, tabStops: tabChoice];
       FW.MakeBooleanItem[
          window: window, myKey: nextItem,
          initBoolean: lex.lookUp, label: [string[XM.Get[h, keySCLookUp]]],
          visibility: visibility, changeProc: LookUpStateChanged ];
       FW.MakeBooleanItem[
          window: window, myKey: nextItem + 1,
          initBoolean: lex.edit, label: [string[XM.Get[h, keySCEdit]]],
          visibility: editVisibility, changeProc: EditStateChanged];
       FW.MakeTextItem[
          window: window, myKey: nextItem + 2,
          initString: lex.name,
          readOnly: TRUE,  boxed: FALSE, width: 260,
          visibility: visibility];
       FW.MakeIntegerItem[
          window: window, myKey: nextItem + 3,
          initInteger: LOOPHOLE[lex.nWords],
          readOnly: TRUE, boxed: FALSE, width: 42,
          visibility: visibility];

       line ← FW.AppendLine[window: window, spaceAboveLine: spaceAboveLine];
       FW.AppendItem[
          window: window, item: nextItem, line: line, preMargin: 10,
          repaint: FALSE];
       FW.AppendItem[
          window: window, item: nextItem + 1, line: line, preMargin: 10,
          repaint: FALSE];
       FW.AppendItem[
          window: window, item: nextItem + 2, line: line, preMargin: 10,
          repaint: FALSE];
       FW.AppendItem[
          window: window, item: nextItem + 3, line: line, preMargin: 10,
          repaint: FALSE];


       };  -- BuildLexiconRow

    CleanUpPermanentData: PROCEDURE = {};


    ClearFdbkParms: PUBLIC PROC  = {
       c: CheckCtxt ← Context.Find[SpellingCheckerCtxtType, fw];
       emptyReaderBody: XS.ReaderBody ← XS.nullReaderBody;
       IF c.lastCorrectionValue # XS.nullReaderBody THEN {
          FW.SetTextItemValue[c.fw, Items.misspelling.ORD, NIL];
          XS.FreeReaderBytes[@c.lastCorrectionValue, c.zone];
          c.lastCorrectionValue ← XS.nullReaderBody};
       };  -- ClearFdbkParms
```

```
    CloseInBkgd: ENTRY PROC [scWn: Window.Dims, rsBody: Window.Place, c: CheckCtxt] = {  --for fast close
      ENABLE UNWIND => NULL;
        sysLexSelectedForLookUp: BOOL;

        DestroyAlternativesProcess[c.pAlternativesGeneration];
        SetDocContext[NIL];  --Clears info parm, nils lastCorrectionValue, pops all enumerators

        IF c.ignoreList # NIL THEN
          c.ignoreList.close[c.ignoreList];
        IF c.ignoreLexicon # NSFile.nullHandle THEN
          NSFile.Delete[c.ignoreLexicon];
        DestroyPairList[c.correctionList];

        FW.SetTextItemValue[c.fw, Items.correction.ORD, NIL];
        FW.SetTextItemValue[c.fw, Items.misspelling.ORD, NIL];
        sysLexSelectedForLookUp ← CloseLexicons[];  -- close all lexicons, files
        save IncludeAnchors and auto-correct with user lex ctnr
        IF c.userLexiconCtnr # NSFile.nullHandle THEN {
          attrList: ARRAY [0..1) OF NSFile.Attribute ← [
            [extended[type: SpellingCheckerDefs.displayAttrType,
              value: DESCRIPTOR[@dispAttrs,
                SIZE [wn DisplayAttrs], WORD]]]];

          dispAttrs: wn DisplayAttrs ← [
            var: wn[scWn: LOOPHOLE[scWn],
              rsBody: LOOPHOLE[rsBody],
              scope: ScopeFromChoice[FW.GetChoiceItemValue[c.fw, Items.scope.ORD]],
              autoCorrect: FW.GetBooleanItemValue[c.fw, Items.autoCorrect.ORD],
              includeAnchoredFrames: FW.GetBooleanItemValue[c.fw, Items.autoCorrect.ORD],
              sysLexSelectedForLookUp: sysLexSelectedForLookUp
              ]
            ];

          --so modifiedOn date won't be updated when these attrs are changes
          NSFileExtra.ChangeAttributesPrivileged[c.userLexiconCtnr, DESCRIPTOR[attrList]];
          NSFile.Close[c.userLexiconCtnr];
          };
      };

    CloseUtil: PUBLIC PROC [fw: Window.Handle, itemData: LONG UNSPECIFIED] = {
        shell: StarWindowShell.Handle ← StarWindowShell.ShellFromChild[fw];

        [] ← StarWindowShell.Pop[shell];

        };  -- CloseUtil


    CollectValues: PROCEDURE [fw: Window.Handle] = {
        itemData: ItemData = Context.Find[SpellingCheckerCtxtType, fw];
        IF ~FW.HasAnyBeenChanged[fw] THEN RETURN;
        FOR myItem: Items IN Items DO
            itemKey: FW.ItemKey = myItem.ORD;
            IF ~FW.HasBeenChanged[fw, itemKey] THEN LOOP;
            SELECT myItem FROM
              scope =>
                itemData.scope ← VAL[FW.GetChoiceItemValue[fw, itemKey]];
              checkFrames =>
                itemData.checkFrames ← FW.GetBooleanItemValue[fw, itemKey];
              autoCorrect =>
                itemData.autoCorrect ← FW.GetBooleanItemValue[fw, itemKey];
              misspelling => SIGNAL Bug[impossible];  -- Shouldn't be user editable.
              correction =>
                itemData.correction ← FormWindow.GetTextItemValue[fw, itemKey, myZone];
              ENDCASE;
            ENDLOOP;

        FW.ResetAllChanged[fw];
        };  -- CollectValues

    DestroyOldLexRows: PROC [fw: Window.Handle] = {
        c: CheckCtxt ← Context.Find[SpellingCheckerCtxtType, fw];
        currentItem: CARDINAL ← lastItem;
        IF c.lexicons # NIL THEN {
            numberEntries: CARDINAL = c.lexicons.nLexicons*4;
            FOR i: CARDINAL IN [0..numberEntries) DO
              FW.DestroyItem[window: fw, item: currentItem, repaint: FALSE];
              currentItem ← currentItem + 1;
              ENDLOOP;
            FW.Repaint[c.fw];
            };
        };  -- DestroyOldLexRows

    DoLayout: FW.LayoutProc = {
        -- PROCEDURE [window: Window.Handle, clientData: LONG POINTER];
        leadingMargin: CARDINAL = 5;
        line: FW.Line;
        -- set the tabs for FormWindow
        tabChoice: FW.TabStops = [fixed[tabStopInterval]];
        -- FW.SetTabStops[window: window, tabStops: tabChoice];
        -- Line 1
        line ← FW.AppendLine[window: window, spaceAboveLine: spaceAboveLine];
        FW.AppendItem[
            window: window, item: Items.scope.ORD, line: line,
            tabStop: leadingMargin / tabStopInterval, preMargin: 10];
        -- Line 2
        line ← FW.AppendLine[window: window, spaceAboveLine: spaceAboveLine];
```

```
        FW.AppendItem[
           window: window, item: Items.checkFrames.ORD, line: line,
            preMargin: 10];
        FW.AppendItem[
           window: window, item: Items.autoCorrect.ORD, line: line,
           preMargin: 10];
        -- Line 3
        line ← FW.AppendLine[window: window, spaceAboveLine: spaceAboveLine];
        FW.AppendItem[
           window: window, item: Items.misspelling.ORD, line: line,
           tabStop: leadingMargin / tabStopInterval, preMargin: 10];
        -- Line 4
        line ← FW.AppendLine[window: window, spaceAboveLine: spaceAboveLine];
        FW.AppendItem[
           window: window, item: Items.correction.ORD, line: line,
           tabStop: leadingMargin / tabStopInterval, preMargin: 10];
        -- Line 5
        -- Add a little space before lexicons

        };  -- DoLayout


    EditStateChanged: FW.BooleanChangeProc = {
    -- [window: Window.Handle, item: ItemKey, calledBecauseOf: ChangeReason, newValue: Boolean
       c: CheckCtxt ← Context.Find[SpellingCheckerCtxtType, window];
       index: CARDINAL = (item - lastItem)/4;

       c.lexicons[index].edit ← newValue;
       c.refreshActiveLists ← TRUE;
       };  -- EditStateChanged


    FreeData: PROCEDURE = {
       --
       -- Call this procedure when deactivating the tool.
       -- It frees the string data from the system zone.
       --
       };  -- FreeData


    GetActiveLexicons: PUBLIC PROC RETURNS [lookUp, edit: LexiconDefs.Lexicons] = {

       c: CheckCtxt ← Context.Find[SpellingCheckerCtxtType, fw];
       IF c.refreshActiveLists THEN {
          c.nLookUp ← c.nEdit ← 0;
          FOR i: CARDINAL IN [0..c.lexicons.nLexicons) DO
             lex: SpellingCheckerDefs.LptLexiconData = @c.lexicons[i];
             IF lex.edit THEN {
                c.lexiconsEdit[c.nEdit] ← lex.lexicon; c.nEdit ← c.nEdit + 1; };
             IF lex.lookUp THEN {
                c.lexiconsLookUp[c.nLookUp] ← lex.lexicon;
                c.nLookUp ← c.nLookUp + 1;
                };
             ENDLOOP;
          c.refreshActiveLists ← FALSE;
          };

       lookUp ←
          IF c.nLookUp > 0 THEN DESCRIPTOR[@c.lexiconsLookUp[0], c.nLookUp]
          ELSE NIL;
       edit ← IF c.nEdit > 0 THEN DESCRIPTOR[@c.lexiconsEdit[0], c.nEdit] ELSE NIL;
       };  -- GetActiveLexicons


    LookUpStateChanged: FW.BooleanChangeProc = {
    -- [window: Window.Handle, item: ItemKey, calledBecauseOf: ChangeReason, newValue: Boolean
       c: CheckCtxt ← Context.Find[SpellingCheckerCtxtType, window];
       index: CARDINAL = (item - lastItem)/4;

       c.lexicons[index].lookUp ← newValue;
       c.refreshActiveLists ← c.invalidAltList ← TRUE;
       };  -- LookUpStateChanged


    MakeItems: FW.MakeItemsProc = {
       -- PROC [window: Window.Handle, clientData: LONG POINTER]
       itemData: ItemData ← myZone.NEW[AqItemData];
       -- set values so that they are equivalent to when the option sheet was last closed.
       Context.Create[
          SpellingCheckerCtxtType, clientData, Context.NopDestroyProc, --CloseUtil,-- window];

       {
       allText: XS.ReaderBody ← XM.Get[h, keySCAllText];
       remainingText: XS.ReaderBody ← XM.Get[h, keySCRemainingText];
       selectedText: XS.ReaderBody ← XM.Get[h, keySCSelectedText];
       scope: XS.ReaderBody ← XM.Get[h, keySCScope];
       choices: ARRAY [0..3) OF FW.ChoiceItem ← [
          [string[0, allText]], [string[1, remainingText]], [
          string[2, selectedText]]];
       FW.MakeChoiceItem[
          window: window, myKey: Items.scope.ORD, tag: @scope,
          values: DESCRIPTOR[choices], initChoice: itemData.scope.ORD,
          changeProc: ScopeChanged]};

       {
       checkFrames: XS.ReaderBody ← XM.Get[h, keySCCheckFrames];
       FW.MakeBooleanItem[
```

```
            window: window, myKey: Items.checkFrames.ORD,
            initBoolean: itemData.checkFrames, label: [string[checkFrames]]];
         {
         autoCorrect: XS.ReaderBody ← XM.Get[h, keySCAutoCorrect];
         FW.MakeBooleanItem[
            window: window, myKey: Items.autoCorrect.ORD,
            initBoolean: itemData.autoCorrect, label: [string[autoCorrect]]];
         {
         misspelling: XS.ReaderBody ← XM.Get[h, keySCMisspelling];
         FW.MakeTextItem[
            window: window, myKey: Items.misspelling.ORD, tag: @misspelling,
            boxed: FALSE, width: 180]];
         {
         correction: XS.ReaderBody ← XM.Get[h, keySCCorrection];
         FW.MakeTextItem[
            window: window, myKey: Items.correction.ORD, tag: @correction,
            hintsProc: SpellingCheckerDefs.PopUpMenuMakeStringProc, width: 180]}
         };  -- MakeItems

      MakePropertySheet: Event.AgentProcedure = {
         title: XS.ReaderBody ← XM.Get[h, keySCTitle];
         c: CheckCtxt ← LOOPHOLE[myData];
         c.sws ← shell ← PropertySheet.Create[
            formWindowItems: MakeItems, menuItemProc: MenuItemProc,
            menuItems: [
            done: FALSE, apply: FALSE, cancel: FALSE, defaults: FALSE,
            start: FALSE, reset: FALSE], size: size, title: @title,
            formWindowItemsLayout: DoLayout,display: FALSE, clientData: myData];
         SpellingCheckerDefs.AddPopupMenu[shell];
         SpellingCheckerDefs.AddMenuCommands[shell, c];
         c.fw ← fw ← PropertySheet.GetFormWindows[shell].form;
         [] ← StarWindowShellExtra.SetSleeps[shell, TRUE];
         };  -- MakePropertySheet

      MenuItemProc: PropertySheet.MenuItemProc = {
         -- PROC [shell: StarWindowShell.Handle, formWindow: Window.Handle,
         --       menuItem: PropertySheet.MenuItemType, clientData: LONG POINTER]
         -- RETURNS [destroy: BOOLEAN ← FALSE];
         SELECT menuItem FROM
      <<    start => {CollectValues[formWindow]; CleanUpPermanentData[]};
            close => {CloseUtil[]; CleanUpPermanentData[]; RETURN[ok: TRUE]};
            Add => {};
            ignore => {};
            correct => {};
      >>
            ENDCASE;
         RETURN[ok: FALSE];
         };  -- MenuItemProc

      MyNotifyDocCloseProc: DocEventDefs.AgentProcedure = {
         c: CheckCtxt ← myData;
         IF doc = c.doc THEN SetDocContext[NIL]; };

      MyNotifyDocDeleteProc: DocEventDefs.AgentProcedure = {
         c: CheckCtxt ← myData;
         IF doc = c.doc THEN SetDocContext[NIL]; };

      MyNotifyDocPaginateProc: DocEventDefs.AgentProcedure = {
         c: CheckCtxt ← myData;
         IF doc = c.doc THEN SetDocContext[NIL]; };

      myPoppedProc: StarWindowShell.PoppedProc = {
         OPEN c: checkCtxt;
         ResetHackOverrides: PROC = {
            DocEventDefs.RemoveDependency[paginationDependency];
            DocEventDefs.RemoveDependency[docDeleteDependency];
            DocEventDefs.RemoveDependency[docCloseDependency];
            };  --ResetHackOverrides

         ResetHackOverrides[];
         c.windowOpen ← FALSE;
         Process.Detach[ FORK
            CloseInBkgd[
               StarWindowShell.GetAvailableBodyWindowDims[shell],
               Window.BitmapPlace[c.fw],
               checkCtxt] ];
         [] ← StarWindowShell.SleepOrDestroy[shell];
         CleanUpPermanentData[];
         };

   Open: MenuData.MenuProc = {OpenUtil[window, LOOPHOLE[itemData]]; };

   -- merge this with list creation perhaps ...
   OpenLexicons: PROC [eraseParmsIfLexCantOpen: BOOL, fw: Window.Handle] = {
   --assumes all files can open
      c: CheckCtxt ← Context.Find[SpellingCheckerCtxtType, fw];

      FOR i: CARDINAL IN [0..c.lexicons.nLexicons) DO
```

```
            lex: SpellingCheckerDefs.LptLexiconData = @c.lexicons[i];
            lex.file ← NSFile.OpenByReference[lex.ref];
            lex.lexicon ← LexiconDefs.OpenLexicon[
               lexiconFile: lex.file, readonly: lex.systemLex, z: c.zone];
            IF lex.lexicon = NIL THEN {  --can't be opened
               NSFile.Close[lex.file];
               lex.file ← NSFile.nullHandle;
               IF eraseParmsIfLexCantOpen THEN {
                  parm: CARDINAL ← lastItem + 4*i;
                  FW.SetVisibility[c.fw, parm, invisibleGhost, FALSE];
                  lex.lookUp ← lex.edit ← FALSE;  --must be false, we've closed the file
                  IF ~lex.systemLex THEN  --erase Edit boolean item
                     FW.SetVisibility[c.fw, parm+1, invisibleGhost, FALSE];
                  FW.Repaint[c.fw];
                  };
               }  --can't be opened
            ELSE lex.nWords ← lex.lexicon.getNWords[lex.lexicon].nWordsCur;
            ENDLOOP;
         };  -- OpenLexicons


      OpenUtil: ENTRY PROC [fw: Window.Handle, myCtxt: LONG UNSPECIFIED] = {
         ENABLE UNWIND => NULL;
         {
         c: CheckCtxt ← myCtxt;
         p: PROCESS;
         dispAttrsObj: wn DisplayAttrs;
         isNewSession: BOOL;
         currentSession: NSFile.Session;
         ctnrLastModifiedOn: System.GreenwichMeanTime;

         SetHackOverrides: PROC = {
            DocEventPaginate: Atom.ATOM ← Atom.MakeAtom["DocEventPaginate"L];
            SomethingMaybeDeletedFromDoc: Atom.ATOM ← Atom.MakeAtom["SomethingMaybeDeletedFromDoc"L];
            DocEventClose: Atom.ATOM ← Atom.MakeAtom["DocEventClose"L];
            paginationDependency ← DocEventDefs.AddDependency[
               agent: MyNotifyDocPaginateProc, myData: c, event: DocEventPaginate,
               remove: NIL];
            docDeleteDependency ← DocEventDefs.AddDependency[
               agent: MyNotifyDocDeleteProc, myData: c,
               event: SomethingMaybeDeletedFromDoc, remove: NIL];
            docCloseDependency ← DocEventDefs.AddDependency[
               agent: MyNotifyDocCloseProc, myData: c, event: DocEventClose,
               remove: NIL];
            -- override overlayWn destroy op was
            -- replaced with Context.DestroyProcType

            };  -- SetHackOverrides

         CreatePrivateObjects: PROC = {
            c.pAlternativesGeneration ← CreateAlternativesProcess[];
            c.correctionList ← CreatePairList[nPairsMax: 250, z: c.zone];
            c.ignoreLexicon ← LexiconDefs.CreateLexicon[
               type: hashTable,
               parent: NSFile.nullHandle,  --temp file
               name: NSString.StringFromMesaString["IgnoreList"L],
               nWordsMax: 10000,  --ignored
               enumerable: FALSE];
            c.ignoreList ← IF c.ignoreLexicon # NSFile.nullHandle THEN
               LexiconDefs.OpenLexicon[c.ignoreLexicon, FALSE, c.zone] ELSE NIL;
            c.refreshActiveLists ← TRUE;  --previous handles are invalid
            c.continueShowing ← FALSE;

            };  --CreatePrivateObjects

         -- DO I WANT THIS OR DO I WANT THE SC COMMAND TO BE DEPENDENT ON WHETHER OR NOT A SC WINDOW IS ALREADY UP?
         IF c.windowOpen THEN {
            scMsgSpellingCheckerAlreadyActive: XS.ReaderBody ← XM.Get[h, keySCMsgSpellingCheckerAlreadyActive];
            Attention.Post[@scMsgSpellingCheckerAlreadyActive];
            RETURN; };

         SetHackOverrides[];
         p ← FORK CreatePrivateObjects[];
         [c.userLexiconCtnr, ctnrLastModifiedOn, dispAttrsObj] ← OpenUserLexiconCtnr[
];
         --if none, RETURNS [nullHandle, System.gmtEpoch, default attrs]

         currentSession ← NSFile.GetDefaultSession[];  --***TTT
         isNewSession ← (currentSession # c.lastUserSession);  --***TTT

         -- Determine lexicons with RebuildLexiconRows, then OpenLexicons when are creating the option sheet.

         IF isNewSession THEN {  --***TTT
            c.lastUserSession ← currentSession;
            FW.SetBooleanItemValue[
               c.fw, Items.autoCorrect.ORD, dispAttrsObj.autoCorrect];
            FW.SetBooleanItemValue[
               c.fw, Items.checkFrames.ORD, dispAttrsObj.includeAnchoredFrames];
            FW.SetChoiceItemValue[
               c.fw, Items.scope.ORD, ChoiceFromScope[dispAttrsObj.scope]];
            };

         --***TTT
         IF (isNewSession OR (System.SecondsSinceEpoch[ctnrLastModifiedOn] >
         System.SecondsSinceEpoch[c.listCreationTime]))
         OR (c.userLexiconCtnr = NSFile.nullHandle) THEN {
            DestroyOldLexRows[c.fw];
```

```
            StarWindowShell.Push[newShell: shell, poppedProc: myPoppedProc];
            JOIN p;
            RebuildLexiconRows[dispAttrs: @dispAttrsObj, fw: c.fw];
            }
         ELSE {
            StarWindowShell.Push[newShell: shell, poppedProc: myPoppedProc];
            JOIN p;
            OpenLexicons[eraseParmsIfLexCantOpen: TRUE, fw: c.fw];
            };

         c.windowOpen ← TRUE;

         };
         };  -- OpenUtil


      --called only when window opens
      --assumes the user can't update the system lexicon ctnr during a session and continue to use the spelling checker in that session
      --lexicons are closed here
      RebuildLexiconRows: PROC [dispAttrs: WnDisplayAttrs, fw: Window.Handle] = {
         c: CheckCtxt ← Context.Find[SpellingCheckerCtxtType, fw];
         wthRow: INTEGER = dispAttrs.rsBody.wth - 10;  -- 10 dots extra/right

         c.listCreationTime ←
            IF (c.userLexiconCtnr = NSFile.nullHandle) THEN [System.gmtEpoch]
            ELSE System.GetGreenwichMeanTime[];

         -- use System.gmtEpoch as a list create time to flag a list created when no user lexicon ctnr was available

         CreateLexiconLists[      --destroy old strings and lists, create new set
            systemLexSelectedForLookUp: dispAttrs.sysLexSelectedForLookUp, c: c];
         OpenLexicons[eraseParmsIfLexCantOpen: FALSE, fw: fw];

         -- add lexicon rows to property sheet
         FOR i: CARDINAL IN [0..c.lexicons.nLexicons) DO
            BuildLexiconRow[
               window: c.fw, wthRow: wthRow, lex: @c.lexicons[i], currentLexicon:i];
            FW.Repaint[c.fw];
            ENDLOOP;
         };  -- RebuildLexiconRows


      ScopeChanged: FormWindow.ChoiceChangeProc = {
         SetDocContext[NIL]};


      --to be called immediately after editing; assumes lexiconData.edit indicates the possibility of having been edited
      UpdateWordCountDisplay: PUBLIC PROC [checkCtxt: CheckCtxt] = {
         OPEN c: checkCtxt;
         hasBeenChanged: BOOLEAN ← FALSE;

         FOR i: CARDINAL IN [0..c.lexicons.nLexicons) DO
            lex: LptLexiconData = @c.lexicons[i];

            IF lex.edit THEN {
               nWords: LONG CARDINAL = lex.lexicon.getNWords[lex.lexicon].nWordsCur;
               IF nWords # lex.nWords THEN {  --update display
                  FW.SetIntegerItemValue[
                     window: c.fw, item: (lastItem + (4*i) + 3),
                     newValue: LOOPHOLE[nWords], repaint: FALSE];
                  lex.nWords ← nWords;
                  hasBeenChanged ← TRUE;
                  }

               };
            ENDLOOP;
            IF hasBeenChanged THEN FW.Repaint[c.fw];
         };

      scTitle: XS.ReaderBody ← XM.Get[h, keySCTitle];
      item: MenuData.ItemHandle;

      AddSCCommand: Event.AgentProcedure = {
         item ← MenuData.CreateItem[
            zone: myZone, name: @scTitle, proc: Open, itemData: myData];
         Attention.AddMenuItem[item]};

      RemoveSCCommand: Event.AgentProcedure = {
         Attention.RemoveMenuItem[item];
         dummyProcToForceWait[];
         };


      dummyProcToForceWait: ENTRY PROC = {
       ENABLE UNWIND => NULL;
         };


      Init: PROC = {
         << Initialize SC tool context if the SC is installed in the product >>
         IF ProductFactoring.Enabled[StarPFOptions.starSpelling] THEN {

            {
            z: UNCOUNTED ZONE = Heap.Create[
               initial: 6, increment: 2, largeNodeThreshold: LAST[Heap.NWords]];
            createParms: TxtBlockDefs.CreateParms;
            createParms.lschemaParent ← [
```

```
      DocumentDefs.GetIzn[DocUtilDefs.GetSystemDoc[]],
      InstanceDefs.rrefNil];
    checkCtxt ← z.NEW[CheckCtxtObject];
    myZone ← checkCtxt.zone ← z;
    DocUtilDefs.LockSystemDoc[];
    checkCtxt.scratchBlock ← TxtBlockDefs.Create[
      @createParms ! UNWIND => DocUtilDefs.FreeSystemDoc[]];
    DocUtilDefs.FreeSystemDoc[];
    };

    {
    logon: Atom.ATOM ← Atom.MakeAtom["Logon"L];
    logoff: Atom.ATOM ← Atom.MakeAtom["Logoff"L];
    -- KLUDGE!!!!  TEMPORARY FIX FOR SWS PROBLEM WHEN TRY TO ACCESS DESKTOP DURING INIT CODE.  WHEN FIXED, SIMPLY REMOVE
    AddDependency for desktopWindowAvailable CALL WITH DIRECT CALL TO MakePropertySheet.
    desktopWindowAvailable: Atom.ATOM ← Atom.MakeAtom[
      "DesktopWindowAvailable"L];
    [] ← Event.AddDependency[agent: MakePropertySheet, myData: checkCtxt, event: desktopWindowAvailable];
    [] ← Event.AddDependency[agent: AddSCCommand, myData: checkCtxt, event: logon];
    [] ← Event.AddDependency[agent: RemoveSCCommand, myData: NIL, event: logoff];
    -- might I want an Event.FreeDataProc here?
    };

    -- If starting with Loader want to make SC command available
    IF StarDesktop.GetCurrentDesktopFile[] # NSFile.nullReference THEN {
      [] ← AddSCCommand[event: Atom.null, eventData:NIL, myData: checkCtxt];
      -- another KLUDGE!!!!! Because MakePropertySheet is dependent on the event desktopWindowAvailable, must explicitly call it
      here.
      [] ← MakePropertySheet[event: Atom.null, eventData: NIL, myData: checkCtxt];
      };
    }};


-- Mainline Code
Init[];


END.  --  SpellingCheckerWnPack
```

```
LOG     (date - person - action)
 8-Jun-84 15:31:10 - Walden - OS5.0 release version 1
 3-Jul-84 14:46:19 - Walden
17-Jul-84 18:55:29 - D.J. Lewis - Tie initialization to FeatureDefs product factoring flag.
15-Aug-84 11:53:05 - Walden - Add MyNotifyDocDeleteProc (= MyNotifyDocPaginateProc) [AR 10543] to clear doc context when Document object
might be deleted
 9-Mar-85 10:23:25 - Marks - Update to OS6.
13-Mar-85 17:00:39 - Marks - Change property sheet window size: check if userLexiconCtnr has been deleted between close/open of property
sheet; clear misspelling parm when close.
15-Mar-85 13:39:39 - Marks - Add space between property sheet lines.
 1-Apr-85 14:47:31 - Marks - Use BWS product factoring.
12-Apr-85 16:18:58 - Marks - AR 14335 Post SC command when run with Loader;
 1-May-85 10:42:53 - Marks - Default dependencies so don't get compiler warnings.
24-Jul-85 14:36:11 - Marks - AR17463 Increase size of private dictionary name and correction text parms.
24-Aug-85 10:22:52 - Marks - AR 18933: Let Property Sheet mechanisim determine initial placement of SC P.S. so will look o.k. on
Daybreaks.
26-Aug-85 10:11:58 - Marks - AR18999: Let PoppedProc call CloseUtil to handle non-standard close on p.s.
21-Feb-86 12:10:08 - Bartlett - use LockSystemDoc, FreeSystemDoc
29-May-87 16:54:19 - Marks - change width of integer item (word count for lexicon) from 35 to 42.
 5-Aug-87 14:14:12 - Marks - widen correction text parm
 2-Dec-87 13:29:06 - Bartlett - add UNWIND call to FreeSystemDoc for AR 16204
```

```
-- File: ToolUtilitiesDefs.mesa - last edit:
-- Marks.ES              7-Aug-87 16:37:59

-- Copyright (C) 1987 by Xerox Corporation. All rights reserved.


DIRECTORY
  NfrDefs USING [InterruptRequestOn],
  NSFile USING [Handle, Type],
  NSString USING [String],
  RgnDefs USING [Sc, Srt],
  StandardDefs USING [String],
  StarFileTypeDefs USING [unspecified];

ToolUtilitiesDefs: DEFINITIONS IMPORTS NfrDefs = {
  OPEN RgnDefs;

-- TYPES

-- CONSTANTS
  -- *** HACK BECAUSE ParameterDefs NO LONGER EXISTS.
  -- *** I REPLACED toolPstype: ParameterDefs.Pstype WITH toolPstype: CARDINAL.
  toolPstype: CARDINAL = 1;  --all clients should use this for keying parms

-- PROCEDURES

  FindFile: PROC [directory: NSFile.Handle,
    fileName: NSString.String ← [NIL, 0, 0],
    type: NSFile.Type ← StarFileTypeDefs.unspecified]
    RETURNS [NSFile.Handle];

  LegalScWnFromSrt: PROC [srt: Srt] RETURNS [Sc];

  StopPressed: PROC RETURNS [BOOL] = INLINE {
    RETURN [NfrDefs.InterruptRequestOn[]]};

  Str: PROC [ls: LONG STRING] RETURNS [s: StandardDefs.String] = INLINE {
    RETURN[[LOOPHOLE[@ls.text], ls.length, ls.maxlength]]};

  }. -- of ToolUtilitiesDefs

LOG (date - person - action)
 8-Jun-84 15:54:12 - Walden - OS5.0 release version 1
 9-Mar-85 10:14:29 - Marks - Update for OS6.
 7-Aug-87 15:44:39 - Marks - Remove GetFileName
 a
```

```
-- File: ToolUtilitiesPack.mesa - last edit:
-- Marks.ES              7-Aug-87 16:37:18
-- ToolUtilitiesPack.mesa

-- Copyright (C) 1987 by Xerox Corporation. All rights reserved.

DIRECTORY
  Atom USING [ATOM, GetProp, MakeAtom, null],
  NSFile USING [Error, Filter, Find, Handle, nullHandle, Type],
  NSString USING [String],
  RgnDefs USING [Rs, Sc, Srt],
  StarFileTypeDefs USING [unspecified],
  ToolUtilitiesDefs,
  UserTerminal USING [screenHeight, screenWidth];

ToolUtilitiesPack: PROGRAM
  IMPORTS Atom, NSFile, UserTerminal
  EXPORTS ToolUtilitiesDefs =
  BEGIN

  Bug: SIGNAL[Bugtype] = CODE;
  Bugtype: TYPE = {impossible};

--TYPES

-- CONSTANTS
  screenWidth: CARDINAL = UserTerminal.screenWidth;
  screenHeight: CARDINAL = UserTerminal.screenHeight;
  multiNational: Atom.ATOM ← Atom.null;
  extendedLanguage: Atom.ATOM ← Atom.null;

-- VARIABLES
  ycMin: INTEGER ← 0;  -- value set in init code.  Set to 0 here only to avoid uninitialized variable warnings.

-- PROCEDURES

  FindFile: PUBLIC PROC [
    directory: NSFile.Handle,
    fileName: NSString.String ← [NIL, 0, 0],
    type: NSFile.Type ← StarFileTypeDefs.unspecified]
    RETURNS [file: NSFile.Handle] = {

    filterList: ARRAY [0..2) OF NSFile.Filter ← [
      [equal[[name[fileName]]]],
      [equal[[type[type]]]]
      ];
    filter: NSFile.Filter;

    SELECT TRUE FROM
      (type = StarFileTypeDefs.unspecified) AND (fileName = [NIL, 0, 0]) => RETURN [NSFile.nullHandle];
      (type # StarFileTypeDefs.unspecified) AND fileName # [NIL, 0, 0] => filter ← [and[DESCRIPTOR[filterList]]];
      type # StarFileTypeDefs.unspecified => filter ← filterList[1];
      ENDCASE =>  filter ← filterList[0];

    file ← NSFile.nullHandle;
    file ← NSFile.Find[
      directory: directory,
      scope: [filter: filter] !
        NSFile.Error => IF error = [access[fileNotFound]] THEN CONTINUE ];
    };


<<GetFileName: PUBLIC PROC [file: NSFile.Handle, zone: UNCOUNTED ZONE] RETURNS [s: XString.Reader] = {
    ar: NSFile.AttributesRecord ← TRASH;
    rb: XString.ReaderBody;

    NSFile.GetAttributes[file, [interpreted: [name: TRUE]], @ar];
    rb ← XString.FromNSString[ar.name];
    s ← @rb;
    NSFile.ClearAttributes[@ar];
    };
    >>

  GetycMin: PROC RETURNS [min: INTEGER] = {

    jstar: LONG POINTER TO BOOLEAN ← Atom.GetProp[
       onto: multiNational, prop: extendedLanguage].value;
    min ← (IF jstar↑ THEN 56 ELSE 18)+2;
    };

  LegalScWnFromSrt: PUBLIC PROC [srt: RgnDefs.Srt] RETURNS [RgnDefs.Sc] = {
    OPEN srt.sc;

    xc ← MIN[xc, screenWidth - srt.rs.wth];
    xc ← MAX[0, xc - (xc MOD 2)];  --even

    yc ← MIN[yc, screenHeight - srt.rs.ht];
    yc ← MAX[ycMin, yc - (yc MOD 2)];  --even
    RETURN [srt.sc];
    };

  Init: PROC = {
     multiNational ← Atom.MakeAtom ["MultiNational"L];
     extendedLanguage ← Atom.MakeAtom ["ExtendedLanguage"L];
     ycMin ← GetycMin[];
     };
```

```
    Init[];

  END.  --ToolUtilitiesPack

LOG     (date - person - action)
 8-Jun-84 15:58:17 - Walden - OS5.0 release version 1
 9-Mar-85 10:24:40 - Marks - Update to OS6.
27-Mar-85 16:26:14 - Marks - Use MultiNational instead of MultiNatlDefs.
 7-Aug-87 16:35:35 - Marks - remove GetFileName
```

```
-- File: SpellingCheckerUtilPack.mesa - last edit:
-- Walden              27-Mar-87 17:36:42
-- Maybury.ES           9-Apr-86 13:43:26
-- Marks.ES            12-Sep-85 14:43:11

-- Copyright (C) 1985, 1986, 1987 by Xerox Corporation. All rights reserved.

DIRECTORY
  ApplicationFolder USING [FindDescriptionFile, FromName],
  BWSZone USING [shortLifetime],
  CharDefs USING [chsetRoman, Roman],
  DocInterchangeDefs USING [AppendNewParagraph, AppendText, Doc, DocObject, FinishCreation, StartCreation],
  DocSpecialDefs USING [ClearClientsEditedFlag, EditedFlagID, GetClientsEditedFlag],
  DocumentDefs USING [DocFromIzn, Handle ],
  DocUtilDefs USING [subtypeBlankDoc],
  HashTableLexiconDefs USING [fileType, SpaceInadequateCannotEnumerate],
  HashTableTempLexiconDefs USING [
    OpenTemp, CloseTemp, AddTempWord, LookUpTempWord, Value, ValueFreeProc],
  LexiconDefs USING [CreateLexicon, fileAttrType, GetFileAttrs, IsCurrent, LexiconFileAttrs, LexiconHandle, LexiconIsFull, LexiconType,
  WordProc],
  NSFile USING [Attribute, Attributes, AttributesProc, AttributesRecord, ChangeAttributes, ClearAttributes, Close, Delete, Error,
  ExtendedAttributeType, Filter, GetAttributesByName, GetReference, Handle, List, Move, nullHandle, nullReference, OpenByReference,
  Reference, Type, Words],
  NSString USING [FreeString, String, StringFromMesaString],
  OptionFile USING [EntryEnumProc, EnumerateEntries, GetStringValue],
  <<PCLexiconFileTypeDefs USING [USEnglishSystemLexicon],
  PCUtilityDefs USING [CheckLexiconFiles],>>
  ProductFactoring USING [Enabled],
  Prototype USING [Find],
  PrototypeExtra USING [Add],
  SchemaDefs USING [GetRootCs, Lschema, lschemaNil],
  SelectionDefs USING [DeselectCs],
  SpellingCheckerDefs USING [checkCtxt, CheckCtxt, ClearFdbkParms, DisplayAttrs, displayAttrType, LexiconDataList, LexiconDisplayAttrs,
  LexiconList, LptLexiconData, LptLexiconDataList, ReplaceContinueWithStart, SCScanCtxt, StopAlternativesGeneration, WnDisplayAttrs],
  SpellingCheckerMessageDefs USING [GetHandle, keySCEmptyLexicon, keySCLexiconFolder, keySCWordsListedInDoc],
  StarDesktop USING [AddReferenceToDesktop, GetCurrentDesktopFile],
  StarFileTypeDefs USING [folder],
  StarPFOptions USING [starSpelling],
  System USING [gmtEpoch, GreenwichMeanTime],
  ToolUtilitiesDefs USING [LegalScWnFromSrt],
  TxtDefs USING [textSegmentNil],
  TxtEditDefs USING [ClearTextSegment],
  TxtScanDefs USING [PopAllEnum, ReadonlyWordFlags, ScanCtxt, WIDEN, WordFlagsObject],
  TextUtilDefs USING [BytesSize, CopyToBytes, Rdr, ResetWriter],
  XMessage USING [Get, Handle],
  XFormat USING [Number, Object, UnsignedDecimalFormat, WriterObject],
  XString USING [Bytes, ByteSequence, Character, CopyReader, emptyContext,
    FreeReaderBytes, FromChar, FromNSString, FromSTRING, NSStringFromReader, Reader,
    ReaderBody, unknownContext, vanillaContext, WriterBody, WriterBodyFromSTRING],
  XTime USING [Append, Current];

SpellingCheckerUtilPack: PROGRAM
  IMPORTS ApplicationFolder, BWSZone, CharDefs, DocInterchangeDefs, DocSpecialDefs, DocumentDefs, HashTableLexiconDefs,
  HashTableTempLexiconDefs, LexiconDefs, NSFile, NSString, <<PCUtilityDefs, >>OptionFile, ProductFactoring, Prototype, PrototypeExtra,
  SchemaDefs, SelectionDefs, SpellingCheckerDefs, SpellingCheckerMessageDefs, StarDesktop, ToolUtilitiesDefs, TxtEditDefs, TxtScanDefs,
  TextUtilDefs, XFormat, XMessage, XString, XTime
  EXPORTS SpellingCheckerDefs
  SHARES SelectionDefs, XString =
  BEGIN OPEN SpellingCheckerDefs, SpellingCheckerMessageDefs, TxtDefs, TxtScanDefs, TextUtilDefs, XF: XFormat, XS: XString;

  Bug: SIGNAL[Bugtype] = CODE;
  Bugtype: TYPE = {impossible};

--TYPES
  MakeDocCtxt: TYPE = LONG POINTER TO MakeDocCtxtObject;
  MakeDocCtxtObject: PUBLIC TYPE = RECORD [
    fileDoc: NSFile.Handle,
    doc: DocInterchangeDefs.Doc,
    unused: ARRAY[0..16) OF WORD
    ]; -- concrete type cloned from TextUtilPack

  WordItem: TYPE = LONG POINTER TO WordItemObject;
  WordItemObject: TYPE = RECORD[
    bytes: XS.Bytes,
    length: CARDINAL,
    flagsObject: TxtScanDefs.WordFlagsObject];

-- CONSTANTS
  amerEngSysLexName: NSString.String = NSString.StringFromMesaString["AmericanEnglish"L];
  h: XMessage.Handle = SpellingCheckerMessageDefs.GetHandle[];
  newLineChar: XS.Character ← LOOPHOLE[CharDefs.Roman[newLine]];
  rbNewLine: XS.ReaderBody ← XS.FromChar[@newLineChar];
  scEditedFlagID: DocSpecialDefs.EditedFlagID = 0;
  shortZone: UNCOUNTED ZONE = BWSZone.shortLifetime;

-- VARIABLES
-- PROCEDURES

  ClearHostDocCs: PUBLIC PROC = {
    OPEN c: checkCtxt;
    -- Get lschema from doc
    IF c.doc # NIL THEN {
       rootCs: SchemaDefs.Lschema ← SchemaDefs.GetRootCs[].lschema;
       IF rootCs # SchemaDefs.lschemaNil
          AND DocumentDefs.DocFromIzn[rootCs.izn] = c.doc THEN
       SelectionDefs.DeselectCs[];
       };
```

```
  };

CloseLexicons: PUBLIC PROC RETURNS [sysLexSelectedForLookup: BOOL ← FALSE] = {
  OPEN c: checkCtxt;
  FOR i: CARDINAL IN [0..c.lexicons.nLexicons) DO
    IF c.lexicons[i].systemLex AND c.lexicons[i].lookUp THEN
      --***TTT
      sysLexSelectedForLookup ← TRUE;

    IF c.lexicons[i].lexicon # NIL THEN
      CloseLexicon[@c.lexicons[i]];
    ENDLOOP;
  };

CloseLexicon: PROC [lexData: LptLexiconData] = {
  displayAttributes: DisplayAttrs.lexicon;
  attrList: ARRAY [0..1) OF NSFile.Attribute ← [
    [extended[type: SpellingCheckerDefs.displayAttrType,
     value: DESCRIPTOR[@displayAttributes,
       SIZE [DisplayAttrs.lexicon], WORD]]]];

  lexData.lexicon.close[lexData.lexicon];
  lexData.lexicon ← NIL;

  IF ~lexData.systemLex THEN {
    --set display attributes here
    displayAttributes ← [var: lexicon[
      selectedForLookUp: lexData.lookUp,
      selectedForEdit: lexData.edit]];
    NSFile.ChangeAttributes[lexData.file, DESCRIPTOR[attrList]];
    };
  NSFile.Close[lexData.file];
  lexData.file ← NSFile.nullHandle;
  };

--assumes all lexicons are properly stamped...
CreateLexiconLists: PUBLIC PROC [systemLexSelectedForLookUp: BOOL, c: SpellingCheckerDefs.CheckCtxt] = {
  internalName: XString.ReaderBody ← XString.FromSTRING["Spelling Checker"L];
  applFolderRef: NSFile.Reference = ApplicationFolder.FromName [@internalName];
  applFolder: NSFile.Handle;
  nAllocated: CARDINAL ← 10;
  nLexicons: CARDINAL ← 0;
  lexicons: LptLexiconDataList ← c.zone.NEW[LexiconDataList[nAllocated]];
  system: BOOL ← FALSE;
  extendedSelections: ARRAY[0..2) OF NSFile.ExtendedAttributeType ← [LexiconDefs.fileAttrType, SpellingCheckerDefs.displayAttrType];
  adf: NSFile.Reference;
  sectionName: XString.ReaderBody ← XString.FromSTRING ["System Lexicons"L];

  AddSystemLexiconFromEntryName: OptionFile.EntryEnumProc = {
    -- internal name = entry name
    internalNameNS: NSString.String = XString.NSStringFromReader[entry, BWSZone.shortLifetime];

    AddSystemLexiconFromUserName: PROCEDURE [value: XString.Reader] = {
      attributes: NSFile.AttributesRecord ← TRASH;
      NSFile.GetAttributesByName[
        directory: applFolder,
        path: internalNameNS,
        selections: [interpreted: [fileID: TRUE, service: TRUE],
          extended: DESCRIPTOR[extendedSelections]],
        attributes: @attributes
        ! NSFile.Error => GOTO SkipThisOne];

      AddLexicon[attributes: @attributes, userName: value, system: TRUE];

      NSFile.ClearAttributes[@attributes];

      EXITS
        SkipThisOne => NULL;
      };  --AddSystemLexiconFromUserName

    OptionFile.GetStringValue[
      section: @sectionName, entry: entry, callBack: AddSystemLexiconFromUserName,
        index: 0, file: adf];
    NSString.FreeString[BWSZone.shortLifetime, internalNameNS];
    };  --AddSystemLexiconFromEntryName

  -- needs attributes needed by AddLexicon, plus 'name'
  userCtnrListProc: NSFile.AttributesProc = {
    userName: XString.ReaderBody ← XString.FromNSString[attributes.name];
    AddLexicon[attributes: attributes, userName: @userName, system: FALSE]};

  -- needs fileID, service, and extended selections:
  AddLexicon: PROC [attributes: NSFile.Attributes, userName: XString.Reader, system: BOOL] = {
    fileAttrs: LONG POINTER TO LexiconDefs.LexiconFileAttrs;
    displayAttrs: LexiconDisplayAttrs;

    IF attributes.extended = NIL OR attributes.extended[0].value = NIL THEN RETURN;

    fileAttrs ← LOOPHOLE[BASE[attributes.extended[0].value]];
    displayAttrs ← IF (attributes.extended[1].value = NIL)
      OR LENGTH [attributes.extended[1].value] # SIZE[DisplayAttrs.lexicon] THEN NIL
    ELSE LOOPHOLE[BASE[attributes.extended[1].value]];

    IF nLexicons = nAllocated THEN {
      tempLexicons: LptLexiconDataList ← c.zone.NEW[LexiconDataList[(nAllocated ← (nAllocated+5))]];
      FOR i: CARDINAL IN [0..nLexicons) DO
        tempLexicons[i] ← lexicons[i];
```

```
        ENDLOOP;
      c.zone.FREE[@lexicons];
      lexicons ← tempLexicons;
      };

    lexicons[nLexicons] ← [
      ref: [attributes.fileID, attributes.service],
      file: NSFile.nullHandle,
      name: XS.CopyReader[userName, c.zone],
      lexicon: NIL,
      nWords: 0,
      systemLex: system,
      lookUp: SELECT TRUE FROM
        system => systemLexSelectedForLookUp,
        (displayAttrs = NIL) => FALSE,
        ENDCASE => displayAttrs.selectedForLookUp,
      edit: SELECT TRUE FROM
        system, (displayAttrs = NIL) => FALSE,
        ENDCASE => displayAttrs.selectedForEdit
      ];

    nLexicons ← nLexicons + 1;
    };  --AddLexicon

  IF  c.userLexiconCtnr # NSFile.nullHandle THEN
    NSFile.List[directory: c.userLexiconCtnr,
      proc: userCtnrListProc,
      scope: [filter: [equal[[type[value: HashTableLexiconDefs.fileType]]]]],
      selections: [
        interpreted: [fileID: TRUE, service: TRUE, name: TRUE],
        extended: DESCRIPTOR[extendedSelections]]];

  IF applFolderRef # NSFile.nullReference THEN {
    IF (applFolder ← NSFile.OpenByReference [applFolderRef]) # NSFile.nullHandle THEN {
      adf ← ApplicationFolder.FindDescriptionFile[applFolder];
      IF adf # NSFile.nullReference THEN
        OptionFile.EnumerateEntries[
          @sectionName, AddSystemLexiconFromEntryName, adf];

      NSFile.Close[applFolder];
      };
    };

  IF c.lexicons = NIL
  OR (c.lexicons.nLexicons # nLexicons) THEN {
    IF c.lexicons # NIL THEN DestroyLexiconLists[];
    c.lexicons ← c.zone.NEW[LexiconDataList[nLexicons]];
    c.lexiconsEdit ← c.zone.NEW[LexiconList[nLexicons]];
    c.lexiconsLookUp ← c.zone.NEW[LexiconList[nLexicons]];
    };

  FOR i: CARDINAL IN [0..nLexicons) DO
    c.lexicons[i] ← lexicons[i];
    ENDLOOP;

  c.zone.FREE[@lexicons];
  };

CreatePairList: PUBLIC PROC [nPairsMax: CARDINAL, z: UNCOUNTED ZONE] RETURNS [LexiconDefs.LexiconHandle] = {
  RETURN [HashTableTempLexiconDefs.OpenTemp[
    nEntriesMax: nPairsMax,
    hashTableType: ordinary,
    sizeValues: SIZE[WordItemObject],
    valueFreeProc: PairListFreeProc,
    z: z]];
  };

PairListFreeProc: HashTableTempLexiconDefs.ValueFreeProc = {
  wItem: WordItem = LOOPHOLE[value];
  --need clientCtxt for temp lexicons too
  checkCtxt.zone.FREE[@wItem.bytes];
  };

DestroyPairList: PUBLIC PROC [list: LexiconDefs.LexiconHandle] = {
  HashTableTempLexiconDefs.CloseTemp[list];
  };

LookUpInPairList: PUBLIC PROC [list: LexiconDefs.LexiconHandle, word: XS.Reader, wordFlags: TxtScanDefs.ReadonlyWordFlags] RETURNS
[found: BOOL, associate: XS.ReaderBody, associateFlags: TxtScanDefs.ReadonlyWordFlags] = {
  wItem: WordItem;

  [found, wItem] ← HashTableTempLexiconDefs.LookUpTempWord[list, word, wordFlags];

  IF found THEN {
    associate ← [
      context: XS.vanillaContext,
      limit: wItem.length,
      offset: 0,
      bytes: wItem.bytes];
    associateFlags ← @wItem.flagsObject;
    };
  };

AppendNWordsAndTime: PROC [doc: DocInterchangeDefs.Doc, nWords: LONG CARDINAL] = {
  tabChar: XS.Character ← LOOPHOLE[CharDefs.Roman[tab]];
  rb: XS.ReaderBody ← XS.FromChar[@tabChar];
  msgWordsListed: XS.ReaderBody ← XMessage.Get[h, keySCWordsListedInDoc];
```

```
    lsTemp: LONG STRING = [100];
    wb: XS.WriterBody ← XS.WriterBodyFromSTRING[s: lsTemp, homogeneous: TRUE];
    time: System.GreenwichMeanTime ← XTime.Current[];

    -- put the current date/time in the document's header
    XTime.Append[@wb, time];
    DocInterchangeDefs.AppendText[[doc[doc]], TextUtilDefs.Rdr[@wb], XS.emptyContext];

    DocInterchangeDefs.AppendText[[doc[doc]], @rb, XS.emptyContext];
    DocInterchangeDefs.AppendText[[doc[doc]], @rb, XS.emptyContext];

    TextUtilDefs.ResetWriter[@wb];
    {o: XF.Object ← XF.WriterObject[@wb];
    XF.Number[@o, nWords, XF.UnsignedDecimalFormat];
    };

    DocInterchangeDefs.AppendText[[doc[doc]], Rdr[@wb], XS.emptyContext];
    DocInterchangeDefs.AppendText[[doc[doc]], @msgWordsListed, XS.emptyContext];

    };  -- AppendNWordsAndTime

  MakeDoc: PUBLIC PROC [lexicon: LexiconDefs.LexiconHandle, name: XString.Reader] RETURNS [ok: BOOL] = {
    fileDoc: NSFile.Handle;
    doc: DocInterchangeDefs.Doc;

  <<propose: sort TStrings (small, fixed size); the prefix is known to not change and be chsetRoman; the bytes are known to be constant,
  need to fix relationship with lexicon to keep this knowledge private to lexicon

    TString: TYPE = RECORD [offset, length: CARDINAL];
    >>

    IF (doc ← DocInterchangeDefs.StartCreation[].doc) = NIL THEN
       RETURN [ok: FALSE];

    AppendNWordsAndTime[doc, lexicon.getNWords[lexicon].nWordsCur];

    DocInterchangeDefs.AppendText[[doc[doc]], @rbNewLine, XS.emptyContext];

    AddWordsToDoc[lexicon, doc];
    fileDoc ← DocInterchangeDefs.FinishCreation[@doc].docFile;

    {nsName: NSString.String = XString.NSStringFromReader[name, shortZone];
    attrList: ARRAY [0..1) OF NSFile.Attribute ← [[name[value: nsName]]];
    refDoc: NSFile.Reference ← NSFile.GetReference[fileDoc];
    dtReference: NSFile.Reference = StarDesktop.GetCurrentDesktopFile[];
    dtHandle: NSFile.Handle ← NSFile.OpenByReference[dtReference];

    NSFile.Move[file: fileDoc, destination: dtHandle, attributes: DESCRIPTOR[attrList]];
    NSFile.Close[fileDoc];
    NSFile.Close[dtHandle];
    StarDesktop.AddReferenceToDesktop[reference: refDoc];
    NSString.FreeString[shortZone, nsName];
    };

    RETURN [ok: TRUE];
    };  --MakeDoc

  AddWordsToDoc: PROC [lexicon: LexiconDefs.LexiconHandle, doc: DocInterchangeDefs.Doc] = {
    -- we will sort ReaderBodies (more efficient to sort TStrings?)

    AddWordToDoc: LexiconDefs.WordProc = {
      DocInterchangeDefs.AppendNewParagraph[[doc[doc]]];
      DocInterchangeDefs.AppendText[[doc[doc]], word, XS.unknownContext];
      };

    lexicon.enumerate[lexicon, AddWordToDoc  -- enum. is supposed be in sort-order
      <<Someday we should inform the user if this happens:>>
      ! HashTableLexiconDefs.SpaceInadequateCannotEnumerate => RESUME];
    };

  SetDocContext: PUBLIC PROC [doc: DocumentDefs.Handle] = {
    OPEN c: checkCtxt;
    IF doc = c.doc THEN RETURN;

    StopAlternativesGeneration[];

    IF c.doc # NIL THEN {  -- clear old text context
      scSC: SCScanCtxt = @c.scScanCtxtObject;
      IF scSC.wordScanCtxtObject.scanCtxtObject.ec # NIL THEN {  --in the middle of scan
        TxtScanDefs.PopAllEnum[WIDEN[scSC]];
        };

      ReplaceContinueWithStart[];
      IF scSC.wordScanCtxtObject.ts # textSegmentNil THEN
        TxtEditDefs.ClearTextSegment[@scSC.wordScanCtxtObject.ts];

      ClearFdbkParms[];
      };

    c.doc ← doc;
    IF c.doc # NIL THEN SetDocNotEdited[c.doc];
    };

  SetDocNotEdited: PUBLIC PROC [doc: DocumentDefs.Handle] = {
    DocSpecialDefs.ClearClientsEditedFlag[doc, scEditedFlagID] };

  SetPair: PUBLIC PROC [list: LexiconDefs.LexiconHandle, word: XS.Reader, wordFlags: TxtScanDefs.ReadonlyWordFlags, associate: XS.Reader,
```

```
    associateFlags: TxtScanDefs.ReadonlyWordFlags] RETURNS [ok: BOOL ← TRUE] = {
      added: BOOL;
      wItem: WordItem;
      associateLength: CARDINAL = BytesSize[associate];
        --includes initial chset change if not Roman

      [added, LOOPHOLE[wItem, HashTableTempLexiconDefs.Value]] ← HashTableTempLexiconDefs.AddTempWord[list, word, wordFlags
      !LexiconDefs.LexiconIsFull => GOTO notOK];

      IF added THEN
        wItem.bytes ← checkCtxt.zone.NEW[XS.ByteSequence[associateLength]]
      ELSE {  --already there
        wordsAvail: CARDINAL = SIZE[XS.ByteSequence[wItem.length]];
        wordsNeeded: CARDINAL = SIZE[XS.ByteSequence[associateLength]];

        IF wordsAvail # wordsNeeded THEN {
          checkCtxt.zone.FREE[@wItem.bytes];
          wItem.bytes ← checkCtxt.zone.NEW[XS.ByteSequence[associateLength]];
          };
        };

      CopyToBytes[to: wItem.bytes, from: associate, n: associateLength];

      wItem.length ← associateLength;
      wItem.flagsObject ← associateFlags↑;
      ok ← TRUE;

      EXITS
        notOK => RETURN [ok: FALSE];
      };

    UserEditedDoc: PUBLIC PROC [doc: DocumentDefs.Handle] RETURNS [BOOL] = {
      RETURN[DocSpecialDefs.GetClientsEditedFlag[doc, scEditedFlagID]] };

  -- PRIVATE

    --all lexicons and files had better be closed
    DestroyLexiconLists: PROC = {
     OPEN c: checkCtxt;
      FOR i: CARDINAL IN [0..c.lexicons.nLexicons) DO
        XString.FreeReaderBytes[c.lexicons[i].name, c.zone];
        ENDLOOP;
      c.zone.FREE[@c.lexicons];
      c.zone.FREE[@c.lexiconsEdit];
      c.zone.FREE[@c.lexiconsLookUp];
      };

    OpenUserLexiconCtnr: PUBLIC PROC RETURNS [file: NSFile.Handle, lastModifiedOn: System.GreenwichMeanTime, dispAttrs: DisplayAttrs.wn] =
    {
      defaultAttrs: DisplayAttrs.wn = [
        var: wn[
          scWn: ToolUtilitiesDefs.LegalScWnFromSrt[[sc: [641, 22], rs: [0, 0]]],
          rsBody: [350, 200],
          scope: all,
          autoCorrect: FALSE,
          includeAnchoredFrames: FALSE,
          sysLexSelectedForLookUp: TRUE]];

      extendedAttrs: ARRAY [0..1) OF NSFile.ExtendedAttributeType ← [
        displayAttrType];

      found: BOOL ← FALSE;
      rbLexiconFolder: XS.ReaderBody ← XMessage.Get[h, keySCLexiconFolder];
      lexiconCtnrName: NSString.String = XS.NSStringFromReader[@rbLexiconFolder, checkCtxt.zone];
      filterList: ARRAY [0..2) OF NSFile.Filter ← [
        [equal[[name[lexiconCtnrName]]]],
        [equal[[type[StarFileTypeDefs.folder <<lexiconFolder>>]]]]
        ];
      dtReference: NSFile.Reference = StarDesktop.GetCurrentDesktopFile[];
      dtHandle: NSFile.Handle;

      ListProc: NSFile.AttributesProc = {
        file ← NSFile.OpenByReference[[attributes.fileID, attributes.service]];
        lastModifiedOn ← attributes.modifiedOn;
        dispAttrs ← IF (attributes.extended = NIL)
          OR (attributes.extended[0].value = NIL)
          OR (LENGTH[attributes.extended[0].value] # SIZE[DisplayAttrs.wn])  THEN defaultAttrs
        ELSE
          LOOPHOLE[BASE[attributes.extended[0].value], WnDisplayAttrs]↑;
        found ← TRUE;
        RETURN [continue: FALSE];
        };  --ListProc

      dtHandle ← NSFile.OpenByReference[dtReference];
      NSFile.List[directory: dtHandle,
        proc: ListProc,
        selections: [interpreted: [fileID: TRUE, service: TRUE, modifiedOn: TRUE], extended: DESCRIPTOR[extendedAttrs]],
        scope: [filter: [and[DESCRIPTOR[filterList]]]] ];
      NSFile.Close[dtHandle];

      IF ~found THEN {
        file ← NSFile.nullHandle;
        lastModifiedOn ← System.gmtEpoch;
        dispAttrs ← defaultAttrs;
        };

      NSString.FreeString[checkCtxt.zone, lexiconCtnrName];
```

```
    };

  Init: PROC = {
    IF ProductFactoring.Enabled[StarPFOptions.starSpelling] THEN
      BEGIN
        reference: NSFile.Reference;
        lex: NSFile.Handle;
        rbLexiconName: XS.ReaderBody ← XMessage.Get[h, keySCEmptyLexicon];
        emptyLexiconName: NSString.String = XS.NSStringFromReader[@rbLexiconName, shortZone];

        <<-- init system lexicons
        PCUtilityDefs.CheckLexiconFiles[amerEngSysLexName];
          --generalize for multiple system proximity-format lexicons>>

        --init prototype private lexicon
        reference ← Prototype.Find[
           type: HashTableLexiconDefs.fileType, version: 1, subtype: DocUtilDefs.subtypeBlankDoc];
        lex ←
           IF reference = NSFile.nullReference THEN NSFile.nullHandle
           ELSE NSFile.OpenByReference[reference: reference];
        IF lex = NSFile.nullHandle THEN {
          lex ← LexiconDefs.CreateLexicon[
            hashTable, NSFile.nullHandle, emptyLexiconName, 1000, TRUE];
           PrototypeExtra.Add[
              file: lex, version: 1, subtype: DocUtilDefs.subtypeBlankDoc];
           }
        ELSE
          IF ~LexiconDefs.IsCurrent[LexiconDefs.GetFileAttrs[lex]] THEN {
            NSFile.Delete[lex];
            lex ← LexiconDefs.CreateLexicon[
              hashTable, NSFile.nullHandle, emptyLexiconName, 1000, TRUE];
            PrototypeExtra.Add[
               file: lex, version: 1, subtype: DocUtilDefs.subtypeBlankDoc];
            };
        NSString.FreeString[shortZone, emptyLexiconName];
        IF lex # NSFile.nullHandle THEN NSFile.Close[lex];

       END;

        };



-- MAINLINE

  Init[];

  END.  --  SpellingCheckerUtilPack

LOG     (date - person - action)
 5-Jun-84 16:45:48 - Walden - OS5.0 release version 1
29-Jun-84 10:51:49 - Walden
17-Jul-84 19:14:49 - D.J. Lewis - Tie initialization to FeatureDefs product factoring flag.
26-Jul-84 12:11:08 - D.J. Lewis - Replace XTime with Star time service (MessageDefs.GetTimeMsg).
17-Aug-84 15:34:23 - Walden - Stop alternatives generation in SetDocContext - this SHOULD be done anyway, and if it isn't, causes process
interference in the unMONITORed PCMain lexicon.  This fixes lurking bug when document is closed or paginated or something moved/deleted
when in the middle of checking.
 9-Mar-85 10:22:45 - Marks - Update to OS6.
 9-Mar-85 10:22:45 - Marks - Fix UserEditedDoc, SetDocNotEdited with calls to DocSpecialDefs.
13-Mar-85 15:49:20 - Marks - Make sure that only one Empty Lex is added to prototype folder when needed and previous versions are
deleted.
30-Apr-85 11:14:38 - Marks - Get Dictionaries from Application Folder.
 7-May-85 14:49:38 - Marks - DocInterchangeDefs returns two values now.
 7-Aug-85 11:20:15 - Marks - AR 17975: CreateLexiconLists - when allocate larger data structure copy over data contained in it.
11-Sep-85 13:12:29 - Marks - AR 19933: MakeDoc - use default place.
18-Dec-85 16:35:40 - Maybury - AddWordToDoc: Using para. per word to improve MakeDoc performance.
19-Dec-85 16:08:47 - Maybury - AddWordsToDoc: Cease using GSortDefs (due to bugs in GSortPack; also for perf. gain); rely on
lexicon.enumerate to produce a SORTED enumeration.
30-Jan-86 11:44:07 - Maybury - Init: moved lexicon icon initialization to HashTableLexiconPack.
 3-Feb-86 13:57:22 - Maybury - Adopted HashTableLexiconDefs.fileType (vs. hackLexicon).  AddWordsToDoc: catch <<and RESUME without
 comment>> SpaceInadequateCannotEnumerate.
 9-Apr-86 13:42:20 - Maybury - SetDocContext: conform to new SCScanCtxtObject.
16-Apr-86 16:31:21 - Walden - use PCLexiconFileTypeDefs.USEnglishSystemLexicon for file type of system USEnglish lexicon
```

NewFetch.mc
1-Feb-86 18:01:26

I have an idea in order to decrease the microcode clicks when IB not empty refill occur in rum. I made the new rum that had new bytecode refill routine, but it was not stable. I'm not sure where is bad, since I've modified all place where IBDisp occurred. I show the new refill routine below, so, could you tell me where is bad.

Thank you.

//Toru

New fetching routine: (in refillandtraps.mc)

```
stEmpty:
{ when we arrive here, the buffer is empty, and the ip is pointing at the word to be fetched}
        MAR ← [ipHigh, ipLow+0], Xbus ← uTimeToStabilize, XDisp,            c1, at [400];
        temp1Low ← 1, BRANCH [noStabilize, yesStabilize, 0E],               c2;


yesStabilize:
        IB ← MD, ipLow ← ipLow + 1, GOTO [stabilizeNow], {adjust ip}        c3;


noStabilize:
        IB ← MD, GOTO [stNotEmpty],


stNotEmpty:
        MAR ← ipLow ← [ipHigh, ipLow + temp1Low], Xbus ← uTimeToStabilize,
                XDisp, L3 ← 0,                                              c1, at [500];
stNotEmptyc2:
        AlwaysIBDisp, DISP2 [stabPaCarr],                                   c2;


{1}{no stabilization is needed, page carry does not occur}
        IB ← MD, ipLow ← ipLow - 1, DISPNI [bytecodes],                     c3, at [0, 4, stabPaCarr];


{2}{ this case is special }
   { stabilization is needed, page carry does not occur}
        ipLow ← ipLow - 1, DISPNI [bytecodes],                              c3, at [1, 4, stabPaCarr];


{3}{ no stabilizaion is neede, page carry occurs }
        ipLow ← ipLow + 0FF + 1,                                            c3, at [2, 4, stabPaCarr];

        MAR ← [ipHigh, ipLow + 0], Xbus ← 0, XDisp, GOTO [stNotEmptyc2],    c1;


{4}{ stabilization is needed, page carry occurs }
        ipLow ← ipLow + 0FF + 1, CANCELBR [stabilizeNow, 0F],               c3, at [3, 4, stabPaCarr];


Where : temp1Low = 1 whenever IBDisp occurs.
        uTimeToStabilize = U45 because temp1Low = R4, so, MesaStateG(former U46) = U00.
        The stabilization is needed when uTimeToStabilize = 1 rather than 0FFFF.


{1} : it's normal case, so refill the bytecodes and IBDisp.
{2} : most special case, do not refill the bytecode and IBDisp, since IBDisp can not be canceled. The stabiliztion is executed when
IBEmpty Refill occurs. In this case, we should think about the possibility that IBEmpty Trap occurs when the bytecode in IB0 or IB1
needs more 1 or 2 bytes, so I modified, see the routine below.
{3} : just page carry occurs, adjust the instruction pointer and try to fetch the bytecodes again.
{4} : In this case, IBDisp is canceled. so the stabilization is executed first, and after it, refill the bytecodes.


FatalError:
        temp1Low ← ErrnIBnStkp, ClrIntErr, CANCELBR [$, 0F],                c1, at [0];
FatalErrorSpin:
        temp1Low ← temp1Low  LRot12,                                        c2;
        [] ← temp1Low and 0C, Zero8r,                                       c3;

        Xbus ← uTimeToStabilize, XDisp, BRANCH [ibTrap, otherTrap],         c1;


otherTrap:
        CANCELBR [bailout3, 0F],                                            c2;


ibTrap:
        LODisp, BRANCH [noStabIBErr, yesStabIBErr, 0E],                     c2;


noStabIBErr:
        CANCELBR [bailout1, 0F],                                            c3;


yesStabIBErr:
        ipLow ← ipLow - 1, DISP2 [chkByteLen],                              c3; { point the word to be fetched }


{ it's strange that IB empty Error occured when byte length = 1 }
        GOTO [bailout2],                                                    c1, at [0, 4, chkByteLen];
```

```
{----- byte length = 2 ------}

ibErrByte2:
        MAR ← [ipHigh, ipLow + 0],                                      c1, at [1, 4, chkByteLen];
        ipLow ← ipLow + 1, Cin ← pc16, CANCELBR [$, 0],                 c2; { point theword to be fetched next }
        IB ← MD, IBPtr ← 1, GOTO [stabilizeNow],                        c3;


{----- byte leongth = 3 ------}

ibErrByte3:
        MAR ← [ipHigh, ipLow + 0], XC2npcDisp,                          c1, at [2, 4, chkByteLen];
        ipLow ← ipLow + 1, BRANCH [pcOneInByte3, pcZeroInByte3, OE],    c2; { point the word to be fetched next }

pcZeroInByte3:
        IB ← MD, GOTO [stabilizeNow],                                   c3;


pcOneInByte3:
        IB ← MD, IBPtr ← 1, GOTO [stabilizeNow],                        c3;


{------- byte length = 4 ------}
{ So far, we have no such byte code that it needs another 3 byte to execute, so this case does not implemeted yet.---- bailout ----}
ibErrByte4:
        GOTO [bailout2],                                                c1, at [3, 4, chkByteLen];
```

```
{ refillAndTraps.mc

Instruction buffer refill and other trap handling code for Rum, the Dandelion Smalltalk-80 microcoded virtual machine.

        by P McCullough, J Trow, T Tokunaga

         1-Feb-86 17:16:19

Copyright 1983, 1984, 1985, 1986 by Xerox Corporation.  All rights reserved. }



{Refill and Trap microcode for the Smalltalk Virtual Machine

Introduction to the Refill microcode:



As each Smalltalk bytecode implementation nears completion, an IBDisp is executed.  The action that then occurs is summarized in the
following table:

                Mesa Interrupt Pending                  No Mesa Interrupt Pending
                ----------------------                  -------------------------

IB state
--------

full            trap to location 600                    branch to next macroinstruction interpreter

empty           trap to location 600                    trap to location 400

not empty       trap to location 700                    trap to location 500



If a Mesa interrupt is pending (locations 600 and 700), we pack up the Smalltalk interpreter state and punt to Molasses, otherwise, we
refill the Instruction Buffer and continue interpreting bytecodes.  The code at location 400 (empty Instruction Buffer) reads a word and
loads it into the Instruction Buffer, then starts a read of the next word while simultaneously starting a dispatch on the first byte of
the Instruction Buffer.   If we have trapped to location 500 (non empty Instruction Buffer) we must read one additional word and load it
into the Instruction Buffer.

The reader should note that we unconditionally refill the Instruction Buffer, even if the subsequent bytes are not needed.  Thus, if the
buffer is empty, we read two words even though we may never execute the last 3 bytes.  We take this approach for simplicity and speed,
but it does mean that we can potentially read a word beyond the end of the Object Space.  This is not a problem if the Object Space is
not at the end of the Smalltalk memory (e.g., you could place the Object Table after the Object Space, or simply make the last word of
the Object Space unavailable for allocation).

}



MacroDef [AlwaysIBDisp, (IBDisp, IBPtr ← 1)] {same definition as in Mesa.df};




stEmpty:
        { when we arrive here, the buffer is empty, and the ip is pointing at the word to be fetched }
        MAR ← [ipHigh, ipLow+0],                                        c1, at [400];
        ipLow ← ipLow + 1,                                              c2;
        IB ← MD, GOTO [nextWord-stNotEmpty],                            c3;


stNotEmpty:
        { when we arrive here, the buffer is not empty, and the ip is one word short of pointing at the word to be fetched }
        ipLow ← ipLow + 1,                                              c1, at [500];
        Ybus ← uTimeToStabilize, ZeroBr,                                c2;
        BRANCH [$, nextWord-stNotEmpty],                                c3;

stabilizeNow:
        L0 ← uCodeRequestedStabilization,                               c1;
        Noop,                                                           c2;
        CALL [stabilize],                                               c3;

        otLow ← uActiveContextOop,                                      c1, at [uCodeRequestedStabilization, 10,
        stabilize-return];
        temp3Low ← activeAfterStabilize,                                c2;
        Noop,                                                           c3;

        uMakeVolatileLinkage ← temp3Low, CALL [makeVolatile],           c1;

        temp2High ← uRumRecordHigh,                                     c1, at [activeAfterStabilize, 10, makeVolatile-return];
        temp2Low ← uRumRecordLow,                                       c2;
```

```
        Noop,                                                                   c3;

        MAR ← [temp2High, temp2Low + homeContextOopOffset],                     c1;
        temp3Low ← homeAfterStabilize,  CANCELBR [$, 0],                        c2;
        otLow ← MD,                                                             c3;

        uMakeVolatileLinkage ← temp3Low, CALL [makeVolatile],                   c1;

        temp2High ← uRumRecordHigh,                                             c1, at [homeAfterStabilize, 10, makeVolatile-return];
        temp2Low ← uRumRecordLow,                                               c2;
        Noop,                                                                   c3;

        MAR ← [temp2High, temp2Low + leafContextOopOffset],                     c1;
        temp3Low ← leafAfterStabilize, CANCELBR [$, 0],                         c2;
        otLow ← MD,                                                             c3;

        uMakeVolatileLinkage ← temp3Low, CALL [makeVolatile],                   c1;

        MAR ← [ipHigh, ipLow + 0], GOTO [nextWord-stNotEmpty-c2],               c1, at [leafAfterStabilize, 10, makeVolatile-return];


nextWord-stNotEmpty:
        MAR ← [ipHigh, ipLow + 0],                                              c1;
nextWord-stNotEmpty-c2:
        ipLow ← ipLow - 1 {adjust to proper word}, AlwaysIBDisp, L3 ← 0,        c2;
        IB ← MD, DISPNI [bytecodes],                                            c3;


stEmptyOrFullAndMesaInterrupt:
        Q ← uWP, ZeroBr,                                                        c1, at [600];
        Q ← ~ErrnIBnStkp, BRANCH [yesRealMInt1, noRealMInt1],                   c2;
yesRealMInt1:
        temp1Low ← 0 {mark Mesa interrupt}, GOTO [saveSmalltalkState],          c3;

noRealMInt1:
        [] ← Q LRot12, XDisp,                                                   c3; { check ib = full }

        MAR ← [ipHigh, ipLow + 0], DISP4 [yesIBFull, 0D],                       c1;


yesIBFull:
        AlwaysIBDisp,                                                           c2, at [0F, 10, yesIBFull];
        DISPNI[bytecodes],                                                      c3;


yesIBEmpty:
        ipLow ← ipLow + 1,                                                      c2, at [0D, 10, yesIBFull];
        IB ← MD, GOTO [nextWord-stNotEmpty],                                    c3;


stNotEmptyAndMesaInterrupt:
        Q ← uWP, ZeroBr,                                                        c1, at [700];
mesaInterrupt:
        BRANCH [yesRealMInt, noRealMInt],                                       c2;


yesRealMInt:
        temp1Low ← 0 {mark Mesa interrupt}, GOTO [saveSmalltalkState],          c3;


noRealMInt:
        GOTO[stNotEmpty],                                                       c3;




{

Introduction to the Trap microcode:

Certain conditions (Control Store parity errors, emulator memory errors, Mesa stackPointer overflow or underflow, and IB-empty errors)
cause traps to location 0.  Currently we just hang for any of these errors; future implementations probably want to take a more official
action.  The IB-empty error is useful in the Mesa emulator (where it is utilized to detect and handle page crossings), but in Smalltalk
land it means that a coding error has been made and too many bytes have been fetched from the Instruction Buffer.


}

FatalError:
        temp1Low ← ErrnIBnStkp, ClrIntErr, CANCELBR [$, 0F],                    c1, at [0];
        Q ← trapAtLocation0, GOTO [bailout3],                                   c2;




{ Edit history:

 2-Oct-85 18:31:14      Tokunaga.fx     modify the routine for Mesa interrupt
30-Sep-85 17:09:40      Trow.pa         convert to stretch format }
```

```
{ SmalltalkStateSaveAndRestore.mc

Emulator swapping code for Rum, the Dandelion Smalltalk-80 microcoded virtual machine.

        by P McCullough, J Trow, T Tokunaga

         1-Feb-86 16:57:17

Copyright 1983, 1984, 1985, 1986 by Xerox Corporation.  All rights reserved. }



getSmalltalkState:
        Q ← uRumRecordHigh {retrieve the Rum Record address},                   c1;
        temp3High ← Q LRot0,                                                    c2;
        temp3Low ← uRumRecordLow,                                               c3;

                        {get the method cache address}
        MAR ← [temp3High, temp3Low + methodCacheLowOffset],                     c1;
        CANCELBR [$, 0],                                                        c2;
        Q ← MD,                                                                 c3;

        MAR ← [temp3High, temp3Low + methodCacheHighOffset],                    c1;
        uMethodCacheLow ← Q, CANCELBR [$, 0],                                   c2;
        Q ← MD,                                                                 c3;

        {get the Object Table address}
        MAR ← [temp3High, temp3Low + objectTableHighOffset],                    c1;
        uMethodCacheHigh ← Q, CANCELBR [$, 0],                                  c2;
        otHigh ← MD,                                                            c3;

        {get the oop of the active context, otmap it, save oop and base address}
        MAR ← [temp3High, temp3Low + activeContextOopOffset],
                L1 ← gettingActiveContextDuringInterpreterSwap,                 c1;
        CANCELBR [$, 0],                                                        c2;
        otLow ← MD, CALL [otMap2],                                              c3;

        Q ← temp1High,                                                          c1, at [gettingActiveContextDuringInterpreterSwap, 10,
        otMap2-return];
        uActiveContextHigh ← Q,                                                 c2;
        Q ← temp1Low,                                                           c3;

        uActiveContextLow ← Q,                                                  c1;
        Q ← otLow,                                                              c2;
        uActiveContextOop ← Q,                                                  c3;

        {get current Stack Pointer}
        MAR ← [temp3High, temp3Low + stackPointerLowOffset],                    c1;
        CANCELBR [$, 0],                                                        c2;
        stackLow ← MD,                                                          c3;

        MAR ← [temp3High, temp3Low + stackPointerHighOffset],                   c1;
        CANCELBR [$, 0],                                                        c2;
        stackHigh ← MD,                                                         c3;

        {get Home Context Oop, otMap it, save its address}
        MAR ← [temp3High, temp3Low + homeContextOopOffset],
                L1 ← gettingSmalltalkState,                                     c1;
        CANCELBR [$, 0],                                                        c2;
        otLow ← MD, CALL [otMap],                                               c3;

        uHomeLow ← temp1Low,                                                    c1, at [gettingSmalltalkState, 10, otMap-return];
        Q ← temp1High,                                                          c2;
        homeHigh  ← Q LRot0,                                                    c3;

        {get the oop of current receiver, save it and if it is not a SmallInteger, otMap it and save its address }
        MAR ← [temp3High, temp3Low + receiverOopOffset],                        c1;
        homeLow ← uHomeLow, CANCELBR [$, 0],                                    c2;
        otLow ← MD,                                                             c3;

        uTimeToStabilize ← 0,                                                   c1;
        uReceiverOop ← otLow, YDisp,                                            c2;
        DISP4 [stateTable, 0C],                                                 c3;     {stretch3}


getSmalltalkStateSmallInteger00:
        MAR ← [temp3High, temp3Low + currentMethodOopOffset],
                L1 ← isSmallGettingSmalltalkState,
                GOTO [getSmalltalkStateForSmallInteger],                        c1, at [0C, 10, stateTable];    {stretch3}


getSmalltalkStateOop01:
        L1 ← isOopGettingSmalltalkState, GOTO [getSmalltalkStateForOop],        c1, at [0D, 10, stateTable];    {stretch3}


getSmalltalkStateOop11:
        L1 ← isOopGettingSmalltalkState, GOTO [getSmalltalkStateForOop],        c1, at [0F, 10, stateTable];    {stretch3}


getSmalltalkStateOop10:
        L1 ← isOopGettingSmalltalkState, GOTO [getSmalltalkStateForOop],        c1, at [0E, 10, stateTable];    {stretch3}


getSmalltalkStateForOop:
```

```
        Noop,                                                           c2;
        CALL [otMap],                                                   c3;

        Q ← temp1High,                                                  c1, at [isOopGettingSmalltalkState, 10, otMap-return];
        uReceiverHigh ← Q,                                              c2;
        Q ← temp1Low,                                                   c3;

   {get the oop of the current method, otMap it, set up the machine's instruction pointer registers}
        MAR ← [temp3High, temp3Low + currentMethodOopOffset],
              L1 ← isSmallGettingSmalltalkState,                        c1;
getSmalltalkStateForSmallInteger:
        uReceiverLow ← Q, CANCELBR [$, 0],                              c2;
        otLow ← MD, CALL [otMap],                                       c3;


        Q ← temp1High,                                                  c1, at [isSmallGettingSmalltalkState , 10, otMap-return];
        ipHigh ← Q LRot0,                                               c2;
        ipLow ← temp1Low + objectHeaderSize,                            c3;

        uCurrentMethodHigh ← Q,                                         c1;
        Q ← temp1Low,                                                   c2;
        uCurrentMethodLow ← Q,                                          c3;

        MAR ← [temp3High, temp3Low + instructionPointerOffset],         c1;
        CANCELBR [$, 0],                                                c2;
        temp3Low ← MD,                                                  c3;


fixupInstructionPointer:
        {Upon entry, temp3Low must contain the number of bytes by which to adjust the instruction pointer. ipLow must be the base of the
        current compiled method. Either ipLow must be bumped by objectHeaderSize or temp3Low must account for the object header by being
        overstated by the amount objectHeaderSize*2}

        temp2Low ← RShift1 temp3Low {get word offset}, SE ← 0,
              XC2npcDisp {see what state pc16 is--we want it zero},     c1;
        ipLow ← ipLow + temp2Low {add in word offset},
              BRANCH [flip, noFlip, 0E],                                c2;

flip:
        Cin ← pc16 {make it zero}, GOTO [pc16isZeroNow],                c3;
noFlip:
        GOTO [pc16isZeroNow],                                           c3;

pc16isZeroNow:
        MAR ← [ipHigh, ipLow+ 0] {read a word of bytecodes},            c1;
        [] ← temp3Low {determine desired state of pc16}, YDisp,         c2;
        IB ← MD {load up instruction buffer}, BRANCH [leaveItBe, makeIt1, 0E],  c3;

makeIt1:
        Cin ← pc16, IBPtr ← 1, GOTO [offToSeeTheWizard],                c1;

leaveItBe:
        GOTO [offToSeeTheWizard],                                       c1;

offToSeeTheWizard:
        {because the saving of the Mesa state left the Instruction Buffer empty and we then put in one or two bytes, the following IBDisp
        will cause a trap to the refill code at stNotEmpty which will refill the Instruction Buffer and execute another IBDisp that will
        take us to the interpreter for the current bytecode}

        ipLow ← ipLow + 1,                                              c2;
        GOTO [nextWord-stNotEmpty],                                     c3;


saveSmalltalkState:

        {we come here when Rum finds an unpleasant bytecode, or when a Mesa interrupt has been set.  Upon entry temp1Low should be:  1
        for a notYetInvented bytecode, 0 for a Mesa interrupt has occurred, and 2 for bytecode failure}

        Q ← uRumRecordHigh {retrieve the Rum Record address},           c1;
        temp3High ← Q LRot0,                                            c2;
        temp3Low ← uRumRecordLow,                                       c3;

              {write the current stack pointer}
        MAR ← [temp3High, temp3Low + stackPointerLowOffset],            c1;
        MDR ← stackLow, LOOPHOLE [wok], CANCELBR [$, 0],                c2;
        Noop,                                                           c3;
```

```
        MAR ← [temp3High, temp3Low + stackPointerHighOffset],                   c1;
        MDR ← stackHigh, LOOPHOLE [wok], CANCELBR [$, 0],                       c2;


        {adjust the instruction pointer to make Molasses happy, then write the instructionPointer}

        ipLow ← ipLow - objectHeaderSize,                                       c3;

        temp2Low ← uCurrentMethodLow,                                           c1;
        ipLow ← ipLow - temp2Low,                                               c2;
        ipLow ← LShift1 ipLow,   SE ← pc16,                                     c3;

        [] ← temp1Low, YDisp,                                                   c1;
        DISP4 [smalltalkState],  LODisp {in case of bytecode failure},          c2;

        CANCELBR [writeIp, 0F],                                                 c3, at [0, 10, smalltalkState];

        CANCELBR [writeIp, 0F],                                                 c3, at [1, 10, smalltalkState];

        temp1Low ← 1 {tell Molasses to execute this bytecode},
                DISP2 [ipAdjustment] {bytecode failed -- need to fix up inst ptr},
                                                                                c3, at [2, 10, smalltalkState];

        GOTO [ipAdjusted],                                                      c1, at [0, 4, ipAdjustment];

        ipLow ← ipLow - 1, GOTO [ipAdjusted]                                    c1, at [1, 4, ipAdjustment];

        ipLow ← ipLow - 2, GOTO [ipAdjusted]                                    c1, at [2, 4, ipAdjustment];

        ipLow ← ipLow - 3, GOTO [ipAdjusted]                                    c1, at [3, 4, ipAdjustment];
ipAdjusted:
        Noop,                                                                   c2;
        Noop,                                                                   c3;

writeIp:
        MAR ← [temp3High , temp3Low + instructionPointerOffset],                c1;
        MDR ← ipLow, LOOPHOLE [wok], CANCELBR [$, 0],                           c2;
        Noop,                                                                   c3;

        MAR ← [temp3High, temp3Low + directiveOffset],                          c1;
        MDR ← temp1Low, LOOPHOLE [wok], CANCELBR [$, 0],                        c2;
        GOTO [restoreMesaState],                                                c3;


{todo --- save current method oop}



{ Edit history:

 2-Oct-85 18:22:40      Tokunaga.fx     no IBDisp is used when refill Smalltalk bytecode
30-Sep-85 17:13:45      Trow.pa         convert to stretch format }
```

source (('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '0'))
dest ((''0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '0'))
sim ((''0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '0'))
actual (('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '1F' 'FFFF' 'FFFF') ('0' '0' '0' '0' '0' '0' '0' '1F' 'FFFF' 'FFFF'))
sourceX = 21 sourceY = 0 destX = 123 destY = 1 width = 112 height = 2 combinationRule = 12

source (('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '0'))
dest ((''0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '0'))
sim ((''0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '0'))
actual (('0' '0' '0' '0' '0' '0' '0' 'FF' 'FFFF' 'FFFF') ('0' '0' '0' '0' '0' '0' '0' 'FF' 'FFFF' 'FFFF') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '0'))
sourceX = 99 sourceY = 1 destX = 120 destY = 0 width = 65 height = 2 combinationRule = 13

source (('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '0'))
dest ((''0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '0'))
sim ((''0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '0'))
actual (('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF'))
sourceX = 128 sourceY = 0 destX = 151 destY = 1 width = 49 height = 2 combinationRule = 15

source (('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF'))
dest ((''0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF'))
sim ((''0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF'))
actual (('0' '0' '0' '0' '0' '0' '7FF' 'FFFF' 'FE00' '0') ('0' '0' '0' '0' '0' '0' '7FF' 'FFFF' 'FE00' '1FF') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF'))
sourceX = 121 sourceY = 0 destX = 101 destY = 0 width = 34 height = 2 combinationRule = 15

source (('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF'))
dest ((''0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF'))
sim ((''0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF'))
actual (('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF') ('0' '0' '1FF' 'FFFF' 'E000' '0' '0' '0' '0' '1FF'))
sourceX = 27 sourceY = 0 destX = 39 destY = 2 width = 28 height = 1 combinationRule = 14

source (('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF'))
dest ((''0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF') ('0' '0' '1FF' 'FFFF' 'E000' '0' '0' '0' '0' '1FF'))
sim ((''0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF') ('0' '0' '1FF' 'FFFF' 'E000' '0' '0' '0' '0' '1FF'))
actual (('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FE00'))
sourceX = 35 sourceY = 1 destX = 57 destY = 1 width = 143 height = 2 combinationRule = 8

source (('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF') ('0' '0' '1FF' 'FFFF' 'E000' '0' '0' '0' '0' '1FF'))
dest ((''0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF') ('0' '0' '1FF' 'FFFF' 'E000' '0' '0' '0' '0' '1FF'))
sim ((''0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF') ('0' '0' '1FF' 'FFFF' 'E000' '0' '0' '0' '0' '1FF'))
actual (('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' 'FFFF' 'FFFF' 'FFFF' 'FFFF') ('0' '0' '1FF' 'FFFF' 'E000' '0' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
sourceX = 97 sourceY = 0 destX = 96 destY = 1 width = 63 height = 2 combinationRule = 15

source (('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '0' '0' '0' '0' '0' '0' '1FF') ('0' '0' '1FF' 'FFFF' 'E000' '0' '0' '0' '0' '1FF'))
dest ((''0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FE00'))
sim ((''0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FE00'))
actual (('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '3FF' 'FFFF' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FE00'))
sourceX = 11 sourceY = 1 destX = 22 destY = 1 width = 26 height = 1 combinationRule = 10

source (('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '0' '0' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FE00'))
dest ((''0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '3FF' 'FFFF' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FE00'))
sim ((''0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '3FF' 'FFFF' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FE00'))
actual (('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '3FF' 'FFFF' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
sourceX = 113 sourceY = 0 destX = 95 destY = 2 width = 155 height = 2 combinationRule = 3

source (('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '3FF' 'FFFF' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FE00'))
dest ((''0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '3FF' 'FFFF' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
sim ((''0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '3FF' 'FFFF' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
actual (('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '3FF' 'FFFF' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
sourceX = 24 sourceY = 2 destX = 115 destY = 1 width = 53 height = 2 combinationRule = 4

source (('0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '3FF' 'FFFF' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
dest ((''0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '3FF' 'FFFF' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
sim ((''0' '0' '0' '0' '0' '0' '0' '0' '0' '0') ('0' '3FF' 'FFFF' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
actual (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
sourceX = 62 sourceY = 1 destX = 82 destY = 0 width = 37 height = 1 combinationRule = 13

source (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
dest ((''0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
sim ((''0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
actual (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('1F' '0' '0' '0' '0' '0' '7FFF' 'FFFF' 'DDDD' 'DE00') ('16' '16' '0' '0' '0' '0' '0' '3' 'FE00'))
sourceX = 68 sourceY = 1 destX = 5 destY = 1 width = 109 height = 2 combinationRule = 2

source (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))

dest (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
sim (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
actual (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF0' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
sourceX = 40 sourceY = 0 destX = 20 destY = 1 width = 110 height = 1 combinationRule = 6

source (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
dest (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
sim (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7F' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
actual (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7F' 'FFFF' 'FFFF' 'FC00' '3FFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
sourceX = 44 sourceY = 2 destX = 102 destY = 1 width = 12 height = 1 combinationRule = 8

source (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF0' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
dest (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF0' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
sim (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF0' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
actual (('0' '0' '0' '0' '7FFF' 'CCF6' 'C06' 'DFF' 'FFFF' 'F000') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF0' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
sourceX = 69 sourceY = 1 destX = 65 destY = 0 width = 83 height = 1 combinationRule = 14

source (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF0' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
dest (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF0' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
sim (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF0' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
actual (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '1C' 'FFF0' 'FFF0' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
sourceX = 140 sourceY = 1 destX = 57 destY = 1 width = 2 height = 1 combinationRule = 3

source (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF0' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
dest (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF0' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
sim (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF0' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
actual (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'FFFF') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
sourceX = 84 sourceY = 0 destX = 93 destY = 1 width = 154 height = 2 combinationRule = 13

source (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF0' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
dest (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF0' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
sim (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF0' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFE' '0' '0' '3' 'FE00'))
actual (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF0' 'FFFF' 'FFFF' 'DDDD' 'DE00') ('0' '0' '1FF' 'FF80' '1FF0' '1' 'FFFF' 'FFE9' 'FF03' 'FE00'))
sourceX = 100 sourceY = 2 destX = 76 destY = 2 width = 88 height = 1 combinationRule = 9

source (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'FFFF') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'FFFF') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'FFFF') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' '3FFF' 'FFFF') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' '1FFF' 'FFFF'))
sourceX = 106 sourceY = 1 destX = 122 destY = 1 width = 22 height = 2 combinationRule = 14

source (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'FFFF') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'FFFF') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'FFFF') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'C180') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
sourceX = 57 sourceY = 0 destX = 146 destY = 0 width = 141 height = 2 combinationRule = 4

source (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'C180') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'C180') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'C180') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FF80' '3E7F') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
sourceX = 102 sourceY = 0 destX = 137 destY = 1 width = 71 height = 1 combinationRule = 8

source (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'C180') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'C180') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'C180') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
sourceX = 61 sourceY = 2 destX = 148 destY = 1 width = 158 height = 1 combinationRule = 9

source (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'C180') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'C180') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'C180') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'C180') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFC0' '0' '7FF' 'FFFF'))
sourceX = 91 sourceY = 2 destX = 106 destY = 2 width = 27 height = 2 combinationRule = 12

source (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FE00' '0' '1FF' 'CE7F') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
sourceX = 128 sourceY = 0 destX = 103 destY = 1 width = 151 height = 1 combinationRule = 0

source (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F3F9' 'F200' '0' '0') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))

actual (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FFF' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'
))
sourceX = 94 sourceY = 0 destX = 101 destY = 0 width = 45 height = 1 combinationRule = 14

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FFF' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'
))
dest (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FFF' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'
))
sim (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FFF' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'
))
actual (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FFF' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '0' 'D00' 'F7F' 'E800' 'FF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
sourceX = 60 sourceY = 0 destX = 36 destY = 1 width = 52 height = 2 combinationRule = 6

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FFF' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '7C' 'FFF0' 'FFF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '0' '1FF' 'FF80' '1FFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'
))
dest (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FFF' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '0' 'D00' 'F7F' 'E800' 'FF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FFF' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '0' 'D00' 'F7F' 'E800' 'FF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FFF' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'FF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
sourceX = 91 sourceY = 2 destX = 17 destY = 2 width = 70 height = 2 combinationRule = 3

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FFF' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '0' 'D00' 'F7F' 'E800' 'FF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FFF' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '0' 'D00' 'F7F' 'E800' 'FF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FFF' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '0' 'D00' 'F7F' 'E800' 'FF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FFF' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '0' '8008' '8080' '88' '8800' '3FFF' 'FFFF' 'FFFF' 'FFFF')
)
sourceX = 64 sourceY = 1 destX = 17 destY = 2 width = 81 height = 1 combinationRule = 2

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FFF' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'FF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FFF' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'FF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FFF' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'FF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'FF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
sourceX = 147 sourceY = 0 destX = 122 destY = 0 width = 6 height = 1 combinationRule = 9

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FFF' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'FF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FFF' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'FF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FFF' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'FF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FFF' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'FF' 'FE00' '0' '0' '0'))
sourceX = 51 sourceY = 1 destX = 103 destY = 2 width = 129 height = 1 combinationRule = 9

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'FF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'FF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'FF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
sourceX = 75 sourceY = 2 destX = 90 destY = 2 width = 29 height = 1 combinationRule = 9

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'FF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'FF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'FF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'FFFF') ('0' '16' '16' '16' '16' 'FF' 'FFFF' 'FFFF' 'FFFF' 'FFFF'))
sourceX = 67 sourceY = 2 destX = 139 destY = 1 width = 39 height = 2 combinationRule = 13

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '437F' '7FFF' 'FFFF'))
sourceX = 146 sourceY = 0 destX = 118 destY = 1 width = 85 height = 2 combinationRule = 14

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '7FB0' 'F9F8' '7FF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
sourceX = 34 sourceY = 2 destX = 7 destY = 1 width = 81 height = 2 combinationRule = 8

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' '0' '0') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
sourceX = 113 sourceY = 1 destX = 124 destY = 0 width = 138 height = 1 combinationRule = 0

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('0' '3FF' 'FFFF' '4F' '607' '80F7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '7FB0' 'F9F8' '7FF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '7FB0' 'F9F8' '7FF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '1FF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
sourceX = 124 sourceY = 0 destX = 47 destY = 1 width = 158 height = 1 combinationRule = 3

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '7FB0' 'F9F8' '7FF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '7FB0' 'F9F8' '7FF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '7FB0' 'F9F8' '7FF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '7FB0' 'F9F8' '7FF7' 'FFFF' 'FFFF' 'FFFF' 'CE7E') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
sourceX = 127 sourceY = 1 destX = 158 destY = 1 width = 39 height = 1 combinationRule = 1

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '0' '0' '1209' '1209' '209') ('0' '16' '16' '0' '0' '0' '0' '9' '209'))
sourceX = 29 sourceY = 1 destX = 59 destY = 1 width = 108 height = 2 combinationRule = 1

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'FFFF' 'CE7F') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C000' '0') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
sourceX = 48 sourceY = 0 destX = 130 destY = 1 width = 95 height = 1 combinationRule = 1

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C000' '0') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C000' '0') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C000' '0') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F780' '8000' '7FFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C000' '0') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
sourceX = 135 sourceY = 0 destX = 104 destY = 0 width = 108 height = 1 combinationRule = 2

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C000' '0') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C000' '0') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C000' '0') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1F00' '0' '1' 'C000' '0') ('0' '16' '16' '16' '16' '0' '3000' '3001' 'FFFF' 'FFFF'))
sourceX = 19 sourceY = 1 destX = 88 destY = 1 width = 39 height = 2 combinationRule = 2

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C000' '0') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C000' '0') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C000' '0') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C000' '0') ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF'))
sourceX = 89 sourceY = 1 destX = 101 destY = 2 width = 97 height = 2 combinationRule = 15

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C000' '0') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C000' '0') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C000' '0') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'F000' '0' '3E00' '0') ('0' '16' '16' '16' '16' 'C0' '2' '41FF' 'FFFF' 'FFFF'))
sourceX = 111 sourceY = 0 destX = 100 destY = 1 width = 137 height = 1 combinationRule = 2

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C000' '0') ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C000' '0') ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFFF' 'C000') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C000' '0') ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFC0' '3FFF') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C03F' 'FFFF') ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF'))
sourceX = 7 sourceY = 0 destX = 138 destY = 0 width = 109 height = 2 combinationRule = 10

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFC0' '3FFF') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C03F' 'FFFF') ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFC0' '3FFF') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C03F' 'FFFF') ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFC0' '3FFF') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C03F' 'FFFF') ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFDF' 'FFFF') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C03F' 'FFFF') ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF'))
sourceX = 31 sourceY = 0 destX = 139 destY = 0 width = 158 height = 2 combinationRule = 15

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFC0' '3FFF') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C03F' 'FFFF') ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFC0' '3FFF') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C03F' 'FFFF') ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFC0' '3FFF') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C03F' 'FFFF') ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFC0' '3FFF') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFE' '2F' 'F222') ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF'))
sourceX = 116 sourceY = 1 destX = 126 destY = 1 width = 151 height = 1 combinationRule = 4

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFC0' '3FFF') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C03F' 'FFFF') ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFC0' '3FFF') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C03F' 'FFFF') ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFC0' '3FFF') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFF' 'C03F' 'FFFF') ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F88F' '7FC0' 'FFC0' '3FFF') ('1FF' '7C00' '0' '0' '0' '1FF7' 'F00F' 'FFFF' 'C03F' 'FFFF') ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF'))
sourceX = 152 sourceY = 0 destX = 100 destY = 0 width = 126 height = 2 combinationRule = 14

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFC0' '3FFF') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFE' '2F' 'F222') ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF'))
dest (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFC0' '3FFF') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFE' '2F' 'F222') ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF'))
sim (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFC0' '3FFF') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFE' '2F' 'F222') ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF'))
actual (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFC0' '3FFF') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFE' '2F' 'F222') ('0' '16' '16' '16' '8236' '20CF' 'C7FF' 'FFFF' 'FFFF' 'FFFF'))
sourceX = 94 sourceY = 0 destX = 64 destY = 2 width = 63 height = 1 combinationRule = 13

source (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFC0' '3FFF') ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFE' '2F' 'F222') ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF'))

```
dest  (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFC0' '3FFF' ) ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFE' '2F' 'F222' ) ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF' ) )
sim   (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFC0' '3FFF' ) ('1FF' '7C00' '0' '0' '0' '1FF7' 'FFFF' 'FFFE' '2F' 'F222' ) ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF' ) )
actual (('0' '0' '0' '0' '0' '33F9' 'F77F' '7FC0' 'FFC0' '3FFF' ) ('1FF' '7C00' '0' '0' '0' '1FF7' 'FF80' '1' 'FFF2' '2FFF' ) ('0' '16' '16' '16' '16' 'C0' '7FF' 'FFFF' 'FFFF' 'FFFF' ) )
  sourceX = 58 sourceY = 1 destX = 105 destY = 1 width = 86 height = 1 combinationRule = 11
```

```
-- Copyright (C) 1983  by Xerox Corporation. All rights reserved.
-- EtherBooter.mesa, RXJ      , 11-Jul-83 16:35:04
-- NFS          10-Jun-86 10:42:13  adapted for OthelloTool

DIRECTORY
  Boot USING [EthernetBootFileNumber, EthernetRequest],
  Heap USING [systemZone],
  HostNumbers USING [HostNumber],
  NSConstants USING [bootServerSocket],
  OthelloDefs USING [
    AbortingCommand, CommandProcessor, Confirm, GetName,
    IndexTooLarge, MyNameIs, RegisterCommandProc],
  SpecialBooting USING [BootFromEthernet],
  String USING [CopyToNewString],
  System USING [
    broadcastHostNumber, defaultSwitches,
    NetworkAddress, nullNetworkNumber, Switches],
  Unformat USING [Error, HostNumber];

EtherBooter: PROGRAM
  IMPORTS Heap, OthelloDefs, SpecialBooting, String, Unformat =
  BEGIN

  HostNumber: TYPE = HostNumbers.HostNumber;

  bootFileNumber: LONG STRING;

  EtherBoot: PROCEDURE =
    BEGIN
    request: Boot.EthernetRequest;
    switches: System.Switches ← System.defaultSwitches;
    OthelloDefs.GetName["Ether Boot from boot file number: "L, @bootFileNumber];
    GetAddress[@request.bfn, bootFileNumber !
      Unformat.Error => OthelloDefs.AbortingCommand["Can't parse that one (No CH)"L]];
    OthelloDefs.Confirm[];
    request.address ← [
      net: System.nullNetworkNumber,
      host: System.broadcastHostNumber,
      socket: NSConstants.bootServerSocket];
    SpecialBooting.BootFromEthernet[
      ethernetRequest: request, deviceOrdinal: 0, switches: switches];
    END;

  GetAddress: PROCEDURE [host: POINTER TO Boot.EthernetBootFileNumber, s: LONG STRING] =
    BEGIN
    host↑ ← [LOOPHOLE[Unformat.HostNumber[s, octal]]];
    END;

  Commands: PROCEDURE [index: CARDINAL] =
    BEGIN
    SELECT index FROM
      0 =>
        BEGIN
        OthelloDefs.MyNameIs[
          myNameIs: "Ether Boot"L,
          myHelpIs: "Load another program over the Ethernet"L];
        EtherBoot[];
        END;
      ENDCASE => OthelloDefs.IndexTooLarge;
    END;

  commandProcessor: OthelloDefs.CommandProcessor ← [Commands];
  bootFileNumber ← String.CopyToNewString["25200001000", Heap.systemZone];  -- Setup Default
  OthelloDefs.RegisterCommandProc[@commandProcessor];

  END.....
```

```
-- File: OthelloDefs.mesa - last edit:
-- Riggle.PA          12-Jan-87 16:06:45
-- OthelloDefs.mesa (last edited by: RXJ      19-Apr-83 10:54:58)

-- Copyright (C) 1987 by Xerox Corporation. All rights reserved.

DIRECTORY
  Device         USING [Type],
  Environment    USING [bytesPerWord, wordsPerPage],
  PhysicalVolume USING [Handle, ID],
  System         USING [GreenwichMeanTime],
  Volume         USING [ID, Type];

OthelloDefs: DEFINITIONS =
  BEGIN

  CommandProcessor: TYPE = RECORD [
    proc: PROC [index: CARDINAL], next: LONG POINTER TO CommandProcessor ← NULL];
  MyNameIs:         SIGNAL [myNameIs: LONG STRING, myHelpIs: LONG STRING];
    -- resuming executes command
  AbortingCommand: ERROR [reason: LONG STRING, reasonOne: LONG STRING ← NIL];
  IndexTooLarge:   ERROR;
  Question:        SIGNAL;

  RegisterCommandProc: PROC [commandProc: LONG POINTER TO CommandProcessor];

  ConfirmType: TYPE = {once, twice, thrice};
  EchoNoEcho:  TYPE = {echo, stars};

  -- Utility Io
  Confirm:                PROC [how: ConfirmType ← once];
  DebugAsk:               PROC;
  GetName:                PROC [
    prompt: LONG STRING, dest: LONG POINTER TO LONG STRING,
    how: EchoNoEcho ← echo, signalQuestion: BOOLEAN ← FALSE];
  ReadNumber:             PROC [
    prompt: LONG STRING, min, max: LONG CARDINAL,
    default: LONG CARDINAL ← LAST[LONG CARDINAL]]
    RETURNS [ans: LONG CARDINAL];
  ReadShortNumber:        PROC [
    prompt: LONG STRING, min, max, default: LONG CARDINAL]
    RETURNS [CARDINAL];
  WriteFixedWidthNumber: PROC [
    x: LONG CARDINAL, count: CARDINAL, base: CARDINAL ← 10];
  WriteLongNumber:        PROC [num: LONG CARDINAL];
  WriteOctal:             PROC [CARDINAL];
  Yes:                    PROC [LONG STRING] RETURNS [BOOLEAN];

  -- Basic IO
  Cursor:            TYPE = {pointer, ftp};
  SetCursor:         PROC [Cursor];
  FlipCursor:        PROC;

  SetCommandString:  PROC [LONG STRING];  -- string will be freed to Storage.
  BlinkDisplay:      PROC;
  CheckUserAbort:    PROC;  -- clients should prepare UNWIND in case of abort.
  NewLine:           PROC;
  ReadChar:          PROC RETURNS [CHARACTER];
  WriteChar:         PROC [CHARACTER];
  WriteLine:         PROC [LONG STRING];
  WriteString:       PROC [LONG STRING];

  PackedTimeFromString: PROC [s: LONG STRING, justDate: BOOLEAN]
    RETURNS [System.GreenwichMeanTime];
    --  string format must be:
    --    IF justDate=FALSE THEN bDD-MMM-YYbbHH:MM:SSbbZZTb
    --    IF justDate=TRUE  THEN bDD-MMM-YYb
    --  return System.gmtEpoch for bogus time

  -- Exported by VolumeInitImplA
  GetLvIDFromUser: PROC [
    prompt: LONG STRING ← NIL, calledFromSetDebuggerPtrs: BOOLEAN ← FALSE]
    RETURNS [
      pvID: PhysicalVolume.ID, lvID: Volume.ID,
      drive: PhysicalVolume.Handle];
  GetLvTypeFromUser: PROC [prompt: LONG STRING, defaultType: Volume.Type]
    RETURNS [t: Volume.Type];
  GetDriveFromUser:  PROC RETURNS [h: PhysicalVolume.Handle];
  GetDriveNumber:    PROC [h: PhysicalVolume.Handle] RETURNS [CARDINAL];
  GetDriveType:      PROC [h: PhysicalVolume.Handle] RETURNS [Device.Type];

  -- Get bits interface for Initial ucode fetch
  FetchInitialMicrocode: PROC [
    InstallProc: PROC [getPage: PROC RETURNS [LONG POINTER]]];

  -- Clean up any outstanding ftp/stp/?? like connections
  CloseFetch: PROC;

  -- leader pages on boot files
  leaderPages: CARDINAL = 1;

  lpVersion: CARDINAL = 04193;

  lpNoteOffset: PRIVATE CARDINAL = 2;

  lpNoteLength: CARDINAL =
    (Environment.wordsPerPage-lpNoteOffset)*Environment.bytesPerWord;
```

```
LeaderPage: TYPE = MACHINE DEPENDENT RECORD [
  version(0): CARDINAL ← lpVersion,
  length(1): CARDINAL, -- count of characters in note
  note(lpNoteOffset): PACKED ARRAY [0..lpNoteLength) OF CHARACTER];

-- test for special commands enabled
Wizard: PROC RETURNS [BOOLEAN];

-- Crock to make >@[foo]baz work
AlternateGetCMFile: PROC [LONG STRING];

-- aids for othello varients implementing canned scripts (prometheus)
GetCannedScript: PROC;
SuppressOutput:  PROC RETURNS [BOOLEAN];
ThereIsAnError:  PROC;

END....
```

```
-- File: OthelloFetch.mesa - last edit:
-- Riggle.PA            12-Jan-87 16:10:06
-- OthelloFetch.mesa
--  RXJ          22-Feb-84 16:46:26

-- Copyright (C) 1987 by Xerox Corporation. All rights reserved.
DIRECTORY
  File USING [File],
  Stream USING [Handle];

OthelloFetch: DEFINITIONS =
  BEGIN

  Destination: TYPE = RECORD [
    SELECT type: * FROM
      pilotFileSystemWrite => [localFile: File.File],
      string => [stringProc: PROC [LONG STRING]],
      rawWrite => [linkProc: PROC [getPage: PROC RETURNS [LONG POINTER]]],
      ENDCASE];

  Object: TYPE = RECORD [
    next: Handle ← NIL,
    Retrieve: PROC [fileName: LONG STRING, destination: Destination],
    DoIndirect: PROC [cmFile: LONG STRING] RETURNS [mine: BOOLEAN],
    List: PROC [pattern: LONG STRING],
    Close: PROC];

  Handle: TYPE = LONG POINTER TO Object;

  Register: PROC [h: Handle];

  Select: PROC [h: Handle];  -- makes h current; closes previous

  SetLeaderPage: PUBLIC PROCEDURE [file: File.File, note: LONG STRING];

  StartFeedback: SIGNAL;

  GrabBitsFromStream: PROC [rs: Stream.Handle, rsSizePages: LONG CARDINAL,
    destination: Destination, note: LONG STRING ← NIL];

  userName, userPassword: LONG STRING;

  directory: LONG STRING;

  END.
```

```
-- File: OthelloFetchImpl.mesa - last edit:
-- NFS           3-Jul-86 15:02:29
-- bjd          22-Jun-85 12:17:33
--  rkj 26-Feb-84 15:24:23

-- Copyright (C) 1986 by Xerox Corporation. All rights reserved.

DIRECTORY
  Environment,
  File USING [
    Create, Delete, File, MakePermanent, nullFile, PageNumber, SetSize, Unknown],
  FileTypes USING [tUntypedFile],
  Heap USING [systemZone],
  MFile,
  MStream,
  NSName,
  NSString,
  OthelloDefs USING [
    AbortingCommand, CommandProcessor, Confirm, FlipCursor,
    GetLvIDFromUser, GetName,
    IndexTooLarge, LeaderPage, leaderPages, lpNoteLength, lpVersion,
    MyNameIs, Question, RegisterCommandProc, WriteLine, WriteString, Yes],
  OthelloFetch USING [Destination, Handle, Object, StartFeedback],
  OthelloOps USING [
    BootFileType, GetVolumeBootFile, MakeBootable, MakeUnbootable,
    SetPhysicalVolumeBootFile, SetVolumeBootFile],
  OthelloToolDefs USING [CloseVolume],
  Process ,
  Profile ,
  Space,
  SpecialFile USING [MakeTemporary],
  Stream,
  String ,
  TemporaryBooting USING [InvalidParameters],
  Time,
  Volume USING [ID, InsufficientSpace, Open, systemID];

OthelloFetchImpl: MONITOR
  IMPORTS
    File, Heap, MFile, MStream, NSString, OthelloDefs, OthelloFetch,
    OthelloOps, OthelloToolDefs,
    Process, Profile, Space, SpecialFile, Stream, String,
    TemporaryBooting, Time, Volume
  EXPORTS OthelloDefs, OthelloFetch =
  BEGIN

  Object: TYPE = OthelloFetch.Object;

  Handle: TYPE = OthelloFetch.Handle;

  list: Handle ← NIL;

  current: Handle ← NIL;

  cmFile: LONG STRING ← NIL;
  fileName: LONG STRING ← NIL;

  z: UNCOUNTED ZONE = Heap.systemZone;

  S: PROC [s: LONG STRING] RETURNS [NSString.String] = INLINE {
    RETURN[NSString.StringFromMesaString[s]]};

  -- Fetcher registration

  Register: PUBLIC PROC [h: Handle] =
    BEGIN
    h.next ← list;
    list← h;
    END;

  Select: PUBLIC PROC [h: Handle] =
    BEGIN
    IF current # NIL THEN current.Close[];
    current← h;
    END;


  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- String/Credentials Commands
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  ClearinghouseCmd: PROC = {
    domain, organization: Profile.String ← NIL;
    CopyDomain: PROCEDURE[s: LONG STRING] = {
      domain ← String.CopyToNewString[s, z];};
    CopyOrganization: PROCEDURE[s: LONG STRING] = {
      organization ← String.CopyToNewString[s, z];};
    {ENABLE UNWIND => {
      IF domain # NIL THEN z.FREE[@domain];
      IF organization # NIL THEN z.FREE[@organization];};
    OthelloDefs.MyNameIs[
          myNameIs: "Clearinghouse"L,
          myHelpIs: "Set defaults for Clearinghouse"L];
    Profile.GetDefaultDomain[CopyDomain];
    Profile.GetDefaultOrganization[CopyOrganization];
    OthelloDefs.GetName["Domain: "L, @domain];
    OthelloDefs.GetName["Organization: "L, @organization];

OthelloFetchImpl.mesa        22-Jan-87 13:55:45 PST                    1
```

```
    Profile.SetDefaultDomain[domain];
    Profile.SetDefaultOrganization[organization];
    };
    z.FREE[@domain]; z.FREE[@organization];
  };

  LoginCmd: PROC = {
    userName, userPassword: Profile.String ← NIL;
    CopyNameAndPassword: PROCEDURE[name, password: LONG STRING] = {
      userName ← String.CopyToNewString[name, z];
      userPassword ← String.CopyToNewString[password, z];};
    {ENABLE UNWIND => {
      IF userName # NIL THEN z.FREE[@userName];
      IF userPassword # NIL THEN z.FREE[@userPassword];};
    OthelloDefs.MyNameIs[
      myNameIs: "Login"L, myHelpIs: "Set user name & password"L];
    Profile.GetUser[CopyNameAndPassword, none];
    OthelloDefs.GetName["User: "L, @userName];
    OthelloDefs.GetName["Password: "L, @userPassword, stars];
    Profile.SetUser[userName, userPassword];
    };
    z.FREE[@userName]; z.FREE[@userPassword];
  };

  directory: PUBLIC LONG STRING ← NIL;

  Directory: PROC = {
    OthelloDefs.MyNameIs[
      myNameIs: "Directory"L,
      myHelpIs: "Set Default FTP directory"L];
    OthelloDefs.GetName["Directory: "L, @directory]};


  -- Simple Fetches

  FetchBoot: PROC = {
    Fetch[pilot, "Boot file name: "L, "Fetch Boot File"L, "Fetch Boot File"L, "boot"L, TRUE]};

  FetchGerm: PROC = {
    Fetch[germ,  "Germ file name: "L, "Germ Fetch"L, "Fetch Germ"L, "germ"L]};

  FetchPilotMicrocode: PROC = {
    Fetch[
      softMicrocode,
      "Pilot microcode file name: "L,
      "Pilot Microcode Fetch"L,
      "Fetch and Install Pilot Microcode"L,
      "db"L]};

  FetchDiagnosticMicrocode: PROC = {
    Fetch[
      hardMicrocode,
      "Diagnostic microcode file name: "L,
      "Diagnostic Microcode Fetch"L,
      "Fetch and Install Diagnostic Microcode"L,
      "db"L]};

  Fetch: PROC [
    type: OthelloOps.BootFileType, prompt, name, helpMsg, extension: STRING,
    bootFile: BOOLEAN ← FALSE] = {
    created: BOOLEAN;
    file: File.File;
    firstPage: File.PageNumber;
    lvID: Volume.ID;
    local: BOOLEAN ← current = NIL; -- if no connection open, assume local file.
    OthelloDefs.MyNameIs[myNameIs: name, myHelpIs: helpMsg];
    lvID ← OthelloDefs.GetLvIDFromUser[].lvID;
    OthelloDefs.GetName[prompt, @fileName];
    IF lvID = Volume.systemID AND bootFile THEN {
      -- fetching boot file for system volume.
      oldFile: File.File;
      IF local THEN file ← GetLocalFile[lvID, File.nullFile
        !NoLocalFile => GOTO NoFetch]
      ELSE {
        file ← File.Create[lvID, 1, FileTypes.tUntypedFile];
        current.Retrieve[fileName, [pilotFileSystemWrite[file]]
          !UNWIND => File.Delete[file];
          Volume.InsufficientSpace, Space.InsufficientSpace => {
            OthelloDefs.WriteLine["Insufficient space for new boot file."L];
            File.Delete[file];
            GOTO NoFetch;}];};
      OthelloOps.MakeUnbootable[file, type, firstPage !
        File.Unknown => CONTINUE;
        TemporaryBooting.InvalidParameters => CONTINUE];
      [oldFile, firstPage] ← OthelloOps.GetVolumeBootFile[lvID, type];
      OthelloOps.MakeUnbootable[oldFile, type, firstPage !
        File.Unknown => CONTINUE;
        TemporaryBooting.InvalidParameters => {
          OthelloDefs.WriteLine["Warning, trouble making unbootable"L];
          CONTINUE}];
      SpecialFile.MakeTemporary[oldFile];
      created ← TRUE;}
    ELSE { -- not system volume
      Volume.Open[lvID];
      [file, firstPage] ← OthelloOps.GetVolumeBootFile[lvID, type];
      IF local THEN {
        newBootFile: File.File;
```

```
        created ← TRUE;
        newBootFile ← GetLocalFile[lvID, file !NoLocalFile => GOTO NoFetch];
        file ← newBootFile;}
      ELSE {
        IF (created ← file = File.nullFile) THEN
          file ← File.Create[lvID, 1, FileTypes.tUntypedFile]
        ELSE OthelloOps.MakeUnbootable[file, type, firstPage !
          File.Unknown => CONTINUE;
          TemporaryBooting.InvalidParameters => {
            OthelloDefs.WriteLine["Warning, trouble making unbootable"L];
            CONTINUE}];
        current.Retrieve[fileName, [pilotFileSystemWrite[file]]
          ! UNWIND => {
            IF created THEN File.Delete[file];OthelloToolDefs.CloseVolume[lvID]}];
        };
    };
    OthelloDefs.WriteString["Installing..."L];
    OthelloOps.SetVolumeBootFile[file, type, OthelloDefs.leaderPages];
    IF created THEN File.MakePermanent[file];
    OthelloOps.MakeBootable[file, type, OthelloDefs.leaderPages
    ! TemporaryBooting.InvalidParameters => {
      OthelloDefs.WriteLine["Warning, trouble making bootable"L]; CONTINUE}];
    OthelloDefs.WriteLine["done"L];
    IF type IN [hardMicrocode..germ] AND
       OthelloDefs.Yes["Shall I also use this for the Physical Volume? "L
       ! UNWIND => OthelloToolDefs.CloseVolume[lvID]] THEN
      OthelloOps.SetPhysicalVolumeBootFile[file, type, OthelloDefs.leaderPages];
    OthelloToolDefs.CloseVolume[lvID];
    EXITS NoFetch => NULL;
  };

  NoLocalFile: ERROR = CODE;

  GetLocalFile: PROCEDURE [
    lvID: Volume.ID, oldFile: File.File] RETURNS [file: File.File] = {
    -- fileName already has name of local file.
    mStream: MStream.Handle;
    filePages: LONG CARDINAL;
    note: LONG STRING;
    mStream ← MStream.ReadOnly[fileName, [] !
      MStream.Error => {
        OthelloDefs.WriteLine["Unable to acquire local file"L];
        ERROR NoLocalFile;}];
    IF oldFile # File.nullFile THEN File.Delete[oldFile];
    filePages ← (MStream.GetLength[mStream] + Environment.bytesPerPage -1) /
        Environment.bytesPerPage;
    file ← File.Create[
      volume: lvID, initialSize: filePages + OthelloDefs.leaderPages,
      type: FileTypes.tUntypedFile !
        Volume.InsufficientSpace => OthelloDefs.AbortingCommand["Volume Full"L]];
    note ← MakeNote[MStream.GetFile[mStream]];
    GrabBitsFromStream[
      mStream, filePages, [pilotFileSystemWrite[file]], note!
      OthelloFetch.StartFeedback => {
        OthelloDefs.WriteString["Copying local file..."L];
        RESUME}];
    OthelloDefs.WriteLine["done"L];
    Stream.Delete[mStream];
    z.FREE[@note];
  };

  bufPages: CARDINAL = 8;

  StartFeedback: PUBLIC SIGNAL = CODE;

  MakeNote: PROCEDURE[file: MFile.Handle] RETURNS[note: LONG STRING] = {
    time: LONG STRING ← [20];
    note ← z.NEW[StringBody[MFile.maxNameLength]];
    note.length ← 0;
    MFile.GetFullName[file, note];
    String.AppendStringAndGrow[@note, " ("L, z];
    Time.Append[time, Time.Unpack[MFile.GetCreateDate[file]]];
    String.AppendStringAndGrow[@note, time, z];
    String.AppendCharAndGrow[@note, '), z];
  };

  GrabBitsFromStream: PUBLIC PROC [
    rs: Stream.Handle, rsSizePages: LONG CARDINAL,
    destination: OthelloFetch.Destination, note: LONG STRING ← NIL] = {
    WITH destination SELECT FROM
      pilotFileSystemWrite => {
        buffer: LONG POINTER ← NIL;
        base: File.PageNumber ← 0;
        got: CARDINAL;
        File.SetSize[localFile, rsSizePages + OthelloDefs.leaderPages
        ! Volume.InsufficientSpace => OthelloDefs.AbortingCommand["Volume Full"L]];
        SetLeaderPage[localFile, note];
        SIGNAL StartFeedback;
        WHILE base < rsSizePages DO
          thisPages: CARDINAL = CARDINAL[MIN[rsSizePages-base, bufPages]];
          size:      CARDINAL = thisPages*Environment.bytesPerPage;
          start:     CARDINAL ← 0;
          nProcesses ← 0;
          buffer ← Space.Map[
            window:[localFile, base+OthelloDefs.leaderPages, thisPages],
            life: dead].pointer;
          DO
```

```
            [bytesTransferred: got] ← rs.GetBlock[[
              blockPointer: buffer, startIndex: start, stopIndexPlusOne: size] !
                Stream.EndOfStream => {
                  got ← 0; start ← start + nextIndex; CONTINUE};
                UNWIND => [] ← Space.Unmap[buffer]];
            IF got = 0 THEN {[] ← Space.Unmap[buffer]; RETURN};
            IF (start ← start + got) = size THEN EXIT;
            ENDLOOP;
          ForkUnmap[buffer];
          OthelloDefs.FlipCursor[];
          base ← base + thisPages;
          ENDLOOP;
        buffer ← Space.ScratchMap[1]; -- check for any leftover stuff
        [bytesTransferred: got] ← rs.GetBlock[[
          blockPointer: buffer, startIndex: 0,
          stopIndexPlusOne: Environment.bytesPerPage] !
            Stream.EndOfStream => {got ← nextIndex; CONTINUE};
            UNWIND => [] ← Space.Unmap[buffer]];
        [] ← Space.Unmap[buffer];
        IF got # 0 THEN OthelloDefs.AbortingCommand[
          "File longer than advertised length"L]};
      string => {
        SIGNAL StartFeedback;
        DO
          stringOverhead: CARDINAL = SIZE[StringBody]*Environment.bytesPerWord;
          string: LONG STRING = Space.ScratchMap[bufPages];
          string↑ ← [
            length: 0,
            maxlength: bufPages*Environment.bytesPerPage - stringOverhead,
            text: ];
          WHILE string.length < string.maxlength DO
            got: CARDINAL;
            [bytesTransferred: got] ← rs.get[
              rs,
              [blockPointer: LOOPHOLE[@string.text],
               startIndex: string.length, stopIndexPlusOne: string.maxlength],
              rs.options
              ! Stream.EndOfStream => {
                  got ← 0; string.length ← string.length + nextIndex; CONTINUE};
                UNWIND => [] ← Space.Unmap[string]];
            IF got = 0 THEN {
              stringProc[string! UNWIND => [] ← Space.Unmap[string]];
              [] ← Space.Unmap[string]; RETURN};
            string.length ← string.length + got;
            ENDLOOP;
          [] ← Space.Unmap[string];
          OthelloDefs.AbortingCommand["Command file too long!"L];
          ENDLOOP};
      rawWrite =>{
        buffer: LONG POINTER = Space.ScratchMap[1];
        done:  BOOLEAN ← FALSE;
        first: BOOLEAN ← TRUE;
        options: Stream.InputOptions ← rs.options;
        GetPage: PROC RETURNS [LONG POINTER] = {
          got:   CARDINAL; index: CARDINAL ← 0;
          IF first THEN {SIGNAL StartFeedback; first ← FALSE};
          WHILE ~done DO
            [bytesTransferred: got] ← rs.get[
              sH: rs,
              block: [blockPointer: buffer, startIndex: index,
               stopIndexPlusOne: Environment.bytesPerPage],
              options: options
              ! Stream.EndOfStream => {got ← nextIndex; done ← TRUE; CONTINUE}];
            IF (index ← index + got) = Environment.bytesPerPage
               OR done THEN {OthelloDefs.FlipCursor[]; EXIT}
            ENDLOOP;
          RETURN[IF done AND index = 0 THEN NIL ELSE buffer]};
        options.signalEndOfStream ← TRUE;
        linkProc[GetPage ! UNWIND => [] ← Space.Unmap[buffer]];
        WHILE ~done DO [] ← GetPage[! UNWIND => [] ← Space.Unmap[buffer]] ENDLOOP;
        [] ← Space.Unmap[buffer]};
      ENDCASE => ERROR};

  -- Initial Ucode Fetch Command

  FetchInitialMicrocode: PUBLIC PROC [
    InstallProc: PROC [getPage: PROC RETURNS [LONG POINTER]]] = {
    CheckOpen[];
    OthelloDefs.GetName["File name: "L, @fileName];
    OthelloDefs.Confirm[];
    current.Retrieve[fileName, [rawWrite[InstallProc]]]};

  -- Command Files

  AlternateGetCMFile: PUBLIC PROC [s: LONG STRING] = {
    z.FREE[@cmFile];
    cmFile ← z.NEW[StringBody[s.length+8]];
    FOR i: CARDINAL IN [1..s.length) DO
      String.AppendChar[cmFile, s[i]] ENDLOOP;
    DoIndirect[]};

  Indirect: PROC = {
    OthelloDefs.MyNameIs[
      myNameIs: "@", myHelpIs: "Run command file"L];
    OthelloDefs.GetName["Command file: "L, @cmFile
    ! OthelloDefs.Question => {
      OthelloDefs.WriteLine["[Host]<Dir>Filename"L]; RESUME}];
```

```
    DoIndirect[]};

  DoIndirect: PROC = {
    [] ← String.AppendExtensionIfNeeded[@cmFile, "othello"L, z];
    FOR h: Handle ← list, h.next UNTIL h = NIL DO
      IF h.DoIndirect[cmFile] THEN RETURN;
      ENDLOOP;
    OthelloDefs.AbortingCommand["Unrecognizable command file name"L]};


  -- Misc. commands

   nProcesses, maxProcesses: NATURAL ← LAST[NATURAL];
   finished: CONDITION;

   ForkUnmap: ENTRY PROCEDURE [buffer: LONG POINTER] =
     BEGIN
     BEGIN ENABLE Process.TooManyProcesses => {maxProcesses ← nProcesses; RETRY};
     WHILE nProcesses >= maxProcesses DO
       WAIT finished; ENDLOOP;
     Process.Detach[LOOPHOLE[FORK DoUnmap[buffer]]];
     nProcesses ← nProcesses+1;
     END;
     END;

   DoUnmap: ENTRY PROCEDURE [buffer: LONG POINTER] =
   --buffer ← Space.Unmap[buffer, return];
     BEGIN
     buffer ← Space.Unmap[buffer];
     nProcesses ← MAX[nProcesses-1, 0];
     NOTIFY finished;
     END;

  CloseCmd: PROC = {
    OthelloDefs.MyNameIs[
      myNameIs: "Close"L, myHelpIs: "Close currently open connection"L];
    CloseFetch[]};

  CloseFetch: PUBLIC PROC = {
    Select[NIL]};

  ListCmd: PROC = {
    OthelloDefs.MyNameIs[
      myNameIs: "List Files"L, myHelpIs: "Enumerate files matching pattern"L];
    CheckOpen[];
    OthelloDefs.GetName["Pattern: "L, @fileName
    ! OthelloDefs.Question => {
      OthelloDefs.WriteLine["pattern to match"L]; RESUME}];
    current.List[fileName]};

  CheckOpen: PROC = {
    IF current = NIL THEN
      OthelloDefs.AbortingCommand["You must execute an Open command first"L]};

  SetLeaderPage: PUBLIC PROCEDURE [file: File.File, note: LONG STRING] =
    BEGIN
    lp: LONG POINTER TO OthelloDefs.LeaderPage ← Space.Map[[file, 0, OthelloDefs.leaderPages]].pointer;
    lp.version ← OthelloDefs.lpVersion;
    lp.length ← MIN[note.length, OthelloDefs.lpNoteLength];
    FOR i: CARDINAL IN [0..lp.length) DO
      lp.note[i] ← note[i];
      ENDLOOP;
    [] ← Space.Unmap[lp];
    END;


  -- command processor

  commandProcessor: OthelloDefs.CommandProcessor ← [FetchCommands];

  FetchCommands: PROC [index: CARDINAL] = {
    SELECT index FROM
      0 => Indirect[];
      1 => ClearinghouseCmd[];
      2 => CloseCmd[];
      3 => Directory[];
      4 => FetchBoot[];
      5 => FetchDiagnosticMicrocode[];
      6 => FetchGerm[];
      7 => FetchPilotMicrocode[];
      8 => ListCmd[];
      9 => LoginCmd[];
      ENDCASE => OthelloDefs.IndexTooLarge};

  -- init

  OthelloDefs.RegisterCommandProc[@commandProcessor];

  END.

Log
 5-Jun-86 12:25:59 NFS  Adapted for OthelloTool
```

```
-- Copyright (C) 1984  by Xerox Corporation. All rights reserved.
-- OthelloFloppy.mesa
--   LXR         10-Feb-84 16:25:51
--   RXJ         27-Feb-84 17:17:08

DIRECTORY
  AccessFloppy USING [
    Attributes, AttributesRecord, Close, Error, ErrorType, GetAttributes,
    leaderLength, LookUp, maxDataSize, Open, tFloppyLeaderPage, Time],
  Ascii USING [CR, SP],
  Environment USING [bytesPerPage, bytesPerWord],
  Heap USING [systemZone],
  File USING [File, nullFile, PageNumber, SetSize],
  Floppy USING [
    Error, ErrorType, FileHandle, GetFileAttributes, GetNextFile, nullFileID,
    nullVolumeHandle, PageNumber, Read, VolumeHandle],
  Format USING [Char, Date, Decimal, StringProc],
  NSFile USING [String],
  OthelloDefs USING [
    AbortingCommand, CheckUserAbort, CommandProcessor, FlipCursor,
    IndexTooLarge, leaderPages, MyNameIs, RegisterCommandProc,
    SetCommandString, SetCursor, WriteLine, WriteString],
  OthelloFetch USING [Destination, Object, Register, Select, SetLeaderPage],
  Process USING [Detach],
  Space USING [Map, ScratchMap, Unmap],
  String USING [
    AppendCharAndGrow, AppendLongDecimal, AppendStringAndGrow,
    CopyToNewString, Length, LowerCase],
  Time USING [Append, Unpack],
  Volume USING [ID, InsufficientSpace];

OthelloFloppy: PROGRAM
  IMPORTS
    AccessFloppy, File, Floppy, Format, Heap, OthelloDefs, OthelloFetch,
    Process, Space, String, Time, Volume =
  BEGIN

  DoIndirect: PROC [cmFile: LONG STRING] RETURNS [mine: BOOLEAN] =
    BEGIN
    s: LONG STRING ← NIL;
    GetString: PROC [c: LONG STRING] = {s ← String.CopyToNewString[c, Heap.systemZone]};
    IF cmFile[0] = '[ THEN RETURN [FALSE];
    OthelloFetch.Select[@fetcher]; OpenFloppy[];
    Retrieve[cmFile, [string[GetString]]
      ! UNWIND => Heap.systemZone.FREE[@s]];
    OthelloDefs.WriteLine["done"L];
    OthelloDefs.SetCommandString[s];
    RETURN[TRUE]
    END;

  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- MISC Stuff/Commands
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  floppy: Floppy.VolumeHandle ← Floppy.nullVolumeHandle;

  FloppyOpen: PROC RETURNS [BOOLEAN] = INLINE {RETURN[floppy # Floppy.nullVolumeHandle]};

  OpenCmd: PROC = {
    OthelloDefs.MyNameIs[myNameIs: "Floppy Open"L, myHelpIs: "Prepare to read files from floppy"L];
    OthelloFetch.Select[@fetcher];
    OpenFloppy[]};

  OpenFloppy: PROC = {
    floppy ← AccessFloppy.Open[
      ! AccessFloppy.Error => OthelloDefs.AbortingCommand["Can't open floppy"L];
        Floppy.Error => {FloppyError[error]; RETRY}]};

  CloseFloppy: PROC = {
    AccessFloppy.Close[
      ! AccessFloppy.Error, Floppy.Error => CONTINUE];
    floppy ← Floppy.nullVolumeHandle};

  FloppyList: PROC [pattern: LONG STRING] = {
    ListFiles[IF String.Length[pattern] = 0 THEN NIL ELSE pattern]};

  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- Central commands
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  commandProcessor: OthelloDefs.CommandProcessor ← [FloppyCommands];

  FloppyCommands: PROC [index: CARDINAL] = {
    SELECT index FROM
      0 => OpenCmd[];
      ENDCASE => OthelloDefs.IndexTooLarge};

  fetcher: OthelloFetch.Object ← [
    Retrieve: Retrieve,
    DoIndirect: DoIndirect,
    List: FloppyList,
    Close: CloseFloppy];


  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- file retrieval Stuff/Commands
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  EnumProc: TYPE = PROCEDURE [
```

```
    attributes: AccessFloppy.Attributes, fH: Floppy.FileHandle, name: LONG STRING]
    RETURNS [stop: BOOLEAN ← FALSE];

  ListFiles: PROCEDURE [pattern: LONG STRING] =
    BEGIN
    Write: Format.StringProc = {OthelloDefs.WriteString[s]};
    ListOne: EnumProc =
      BEGIN
      Write[name];
      FOR i: CARDINAL IN [name.length + WritePartial[Write, attributes]..40) DO
        Format.Char[Write, Ascii.SP]; ENDLOOP;
      Format.Date[Write, attributes.createDate, full];
      Format.Char[Write, Ascii.CR];
      END;
    EnumerateFloppyFiles[ListOne, pattern ];
    END;

  WritePartial: PROCEDURE [Write: Format.StringProc, attributes: AccessFloppy.Attributes]
    RETURNS [chars: CARDINAL ← 0] =
    BEGIN
    CountedNumber: PROCEDURE [n: LONG CARDINAL] RETURNS [CARDINAL] = {
      s: STRING = [12];
      String.AppendLongDecimal[s, n];
      Write[s];
      RETURN[s.length]};
    IF attributes.offset # 0 OR attributes.size # attributes.totalSize THEN {
      chars ← 4;
      Format.Char[Write, '[];
      chars ← chars + CountedNumber[attributes.offset];
      Write[".."L];
      chars ← chars + CountedNumber[attributes.offset+attributes.size-1];
      Format.Char[Write, ']]};
    END;

  EnumerateFloppyFiles: PROCEDURE [
    proc: EnumProc, pattern: LONG STRING ← NIL] =
    BEGIN
    nullFile: Floppy.FileHandle ← [volume: floppy, file: Floppy.nullFileID];
    attributes: AccessFloppy.Attributes ← Heap.systemZone.NEW[
      AccessFloppy.AttributesRecord[AccessFloppy.maxDataSize]];
    name: LONG STRING = LOOPHOLE[@attributes.length];
    BEGIN ENABLE Floppy.Error => {
      FloppyError[error]; nullFile.volume ← floppy; RETRY};
    FOR current: Floppy.FileHandle ←
      Floppy.GetNextFile[nullFile].nextFile,
      Floppy.GetNextFile[current].nextFile
      WHILE current#nullFile DO
        ENABLE UNWIND => Heap.systemZone.FREE[@attributes];
        OthelloDefs.CheckUserAbort[];
        IF Floppy.GetFileAttributes[current].type # AccessFloppy.tFloppyLeaderPage THEN LOOP;
        AccessFloppy.GetAttributes[current, attributes];
        IF (pattern = NIL OR MaskFilename[file: name, mask: pattern])
          AND proc[attributes, current, name] THEN EXIT;
        ENDLOOP;
    END; -- ENABLE
    Heap.systemZone.FREE[@attributes];
    END;

  MaskFilename: PROCEDURE [
    file: LONG STRING, fileIndex: CARDINAL ← 0, mask: LONG STRING,
    maskIndex: CARDINAL ← 0]
    RETURNS [BOOLEAN] =
    BEGIN
    -- local variables
    i, j: CARDINAL;
    wildString: CHARACTER = '*;
    wildChar: CHARACTER = '#;
    -- process each character in mask
    FOR i IN [maskIndex..mask.length) DO
      SELECT mask[i] FROM
        wildString =>  -- matches any string of zero or more characters
          BEGIN
          FOR j IN [fileIndex..file.length] DO
            IF MaskFilename[file, j, mask, i + 1] THEN
              RETURN[TRUE];
            ENDLOOP;
          RETURN[FALSE];
          END;
        wildChar =>  -- matches any single character
          IF fileIndex = file.length THEN RETURN[FALSE]
          ELSE fileIndex ← fileIndex + 1;
        ENDCASE =>
          IF fileIndex = file.length
            OR String.LowerCase[file[fileIndex]] # String.LowerCase[mask[i]] THEN
            RETURN[FALSE]
          ELSE fileIndex ← fileIndex + 1;
      ENDLOOP;
    -- filename passes mask if entire filename has been consumed
    RETURN[fileIndex = file.length];
    END;

  StartFeedback: SIGNAL = CODE;

  -- must fix Retrieve to deal with boot files that is in pieces

  Retrieve: PROC [fileName: LONG STRING, destination: OthelloFetch.Destination] = {
    segmentPages, totalPages, bytes, offset: LONG CARDINAL;
```

```
    name: LONG STRING ← NIL;
    BEGIN
    ENABLE Floppy.Error => {FloppyError[error]; RETRY};
    fH: Floppy.FileHandle ← [floppy, Floppy.nullFileID];
    [fh: fH, offset: offset, segmentPages: segmentPages, totalPages: totalPages, bytes: bytes, name: name] ← GetFile[fileName];
    GrabBits[
      fH: fH, offset: offset, segmentPages: segmentPages,
      totalPages: totalPages, sizeBytes: bytes, destination: destination,
      note: name !
        StartFeedback => {
          OthelloDefs.WriteString["Fetching..."L];
          OthelloDefs.SetCursor[ftp];
          RESUME};
        UNWIND => {OthelloDefs.SetCursor[pointer]; Heap.systemZone.FREE[@name]}];
    Heap.systemZone.FREE[@name];
    OthelloDefs.SetCursor[pointer];
    END};

  GetFile: PROC [fileName: LONG STRING] RETURNS [
    fh: Floppy.FileHandle, offset, segmentPages, totalPages, bytes: LONG CARDINAL, name: LONG STRING] =
    BEGIN
    time: LONG STRING ← [20];
    attributes: AccessFloppy.Attributes ← Heap.systemZone.NEW
      [AccessFloppy.AttributesRecord[AccessFloppy.maxDataSize]];
    name ← Heap.systemZone.NEW[StringBody[60]];
    {ENABLE UNWIND =>
      {Heap.systemZone.FREE[@attributes]; Heap.systemZone.FREE[@name]};
    fh ← AccessFloppy.LookUp[
      MakeNSString[fileName], attributes
        ! AccessFloppy.Error =>
          SELECT type FROM
            fileNotFound => OthelloDefs.AbortingCommand["No such file"L];
            volumeNotOpen => {
              CloseFloppy[];
              OpenFloppy[];
              RETRY};
            ENDCASE => OthelloDefs.AbortingCommand[ "Unexpected access floppy problem"L];
          Floppy.Error => {FloppyError[error]; RETRY}];
    String.AppendStringAndGrow[
      @name, LOOPHOLE[@attributes.length], Heap.systemZone];
    String.AppendStringAndGrow[@name, " ("L, Heap.systemZone];
    Time.Append[time, Time.Unpack[attributes.createDate]];
    String.AppendStringAndGrow[@name, time, Heap.systemZone];
    String.AppendCharAndGrow[@name, '), Heap.systemZone];
    offset ← attributes.offset;
    segmentPages ← attributes.size;
    totalPages ← attributes.totalSize;
    bytes ← attributes.totalSizeInBytes;
    Heap.systemZone.FREE[@attributes]};
    END;

  MakeNSString: PROCEDURE [s: LONG STRING] RETURNS [NSFile.String] = {
    IF s = NIL THEN RETURN[[bytes: NIL, length: 0, maxlength: 0]];
    RETURN[[bytes: LOOPHOLE[@s.text], length: s.length, maxlength: s.maxlength]]};

  bufPages: CARDINAL = 8;

  GrabBits: PROC [
    fH: Floppy.FileHandle, offset, segmentPages, totalPages: LONG CARDINAL,
    sizeBytes: LONG CARDINAL, destination: OthelloFetch.Destination,
    note: LONG STRING ← NIL] = {
    base: File.PageNumber ← 0;
    WITH destination SELECT FROM
      pilotFileSystemWrite => {
        buffer: LONG POINTER ← NIL;
        File.SetSize[localFile, totalPages + OthelloDefs.leaderPages
        ! Volume.InsufficientSpace => OthelloDefs.AbortingCommand["Volume Full"L]];
        OthelloFetch.SetLeaderPage[localFile, note];
        SIGNAL StartFeedback;
        WHILE base < segmentPages DO
          thisPages: CARDINAL = CARDINAL[MIN[segmentPages-base, bufPages]];
          buffer ← Space.Map[
            window:[localFile, offset+base+OthelloDefs.leaderPages, thisPages],
            life: dead].pointer;
          Floppy.Read[fH, base+AccessFloppy.leaderLength, thisPages, buffer !
            Floppy.Error => FloppyError[error];
            UNWIND => [] ← Space.Unmap[buffer]];
          --buffer ← Space.Unmap[buffer, return];
          Process.Detach[LOOPHOLE[FORK Space.Unmap[buffer]]]; buffer ← NIL;
          OthelloDefs.FlipCursor[];
          base ← base + thisPages;
          ENDLOOP};
      string => {
        thisPages: CARDINAL = CARDINAL[MIN[segmentPages-base, bufPages]];
        stringOverhead: CARDINAL = SIZE[StringBody]*Environment.bytesPerWord;
        string: LONG STRING ← NIL;
        IF segmentPages-base > thisPages THEN
          OthelloDefs.AbortingCommand["Command file too long!"L];
        SIGNAL StartFeedback;
        string ← Space.ScratchMap[thisPages+1];
        string↑ ← [
          length: CARDINAL[sizeBytes],
          maxlength: bufPages*Environment.bytesPerPage - stringOverhead,
          text: ];
        Floppy.Read[fH, base+AccessFloppy.leaderLength, thisPages, @string.text !
          Floppy.Error => FloppyError[error];
          UNWIND => [] ← Space.Unmap[string]];
```

```
        stringProc[string];
        [] ← Space.Unmap[string]};
      rawWrite =>{
        buffer: LONG POINTER = Space.ScratchMap[1];
        count:  CARDINAL ← 0;
        GetPage: PROC RETURNS [LONG POINTER] = {
          IF count = 0 THEN SIGNAL StartFeedback;
          IF count = segmentPages THEN RETURN[NIL];
          Floppy.Read[fH, base+count+AccessFloppy.leaderLength, 1, buffer !
                Floppy.Error => FloppyError[error];
                UNWIND => [] ← Space.Unmap[buffer]];
          count ← count + 1;
          OthelloDefs.FlipCursor[];
          RETURN[buffer]};
        linkProc[GetPage ! UNWIND => [] ← Space.Unmap[buffer]];
        [] ← Space.Unmap[buffer]};
      ENDCASE => ERROR};

  FloppyError: PROC [error: Floppy.ErrorType] =
    BEGIN
    myProc: Format.StringProc = {OthelloDefs.WriteString[s]};
    SELECT error FROM
      invalidVolumeHandle => {
        AccessFloppy.Close[! AccessFloppy.Error, Floppy.Error => CONTINUE];
        OpenFloppy[]; RETURN};
      notReady => OthelloDefs.AbortingCommand["Can't open floppy"L];
      badDisk, badSectors, hardwareError =>
        OthelloDefs.AbortingCommand["Floppy hardware problem"L];
      invalidFormat, invalidPageNumber, needsScavenging  =>
        OthelloDefs.AbortingCommand["Floppy not readable"L];
      ENDCASE => {
        OthelloDefs.WriteString["Floppy error "L];
        Format.Decimal[myProc, error.ORD]};
    OthelloDefs.AbortingCommand[NIL]
    END;


  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- initialization
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  OthelloDefs.RegisterCommandProc[@commandProcessor];
  OthelloFetch.Register[@fetcher];

  END.....
```

```
-- File: OthelloNS.mesa - last edit:
-- BJD    .PA          15-Feb-85 16:10:58
-- AOF          25-Jan-85 10:33:26

-- Copyright (C) 1983 , 1985 by Xerox Corporation. All rights reserved.

DIRECTORY
  AddressTranslation USING [Error, PrintError, StringToNetworkAddress],
  Auth USING [IdentityHandle],
  NSBuffer USING [Body, Buffer, ReturnBuffer],
  ExtendedString USING [AppendNumber],
  Format USING [StringProc],
  Inline USING [HighByte, LowByte],
  NSConstants USING [echoerSocket],
  NSTypes USING [maxDataBytesPerEcho, wordsPerIDPHeader],
  OthelloDefs,
  Process USING [Detach, Pause, SecondsToTicks, Yield],
  Profile USING [GetID],
  Router USING [
    FillRoutingTable, GetDelayToNet, infinity, endEnumeration, startEnumeration,
    EnumerateRoutingTable],
  Socket USING [
    AssignNetworkAddress, Create, Delete, GetPacket, ChannelHandle,
    PutPacket, GetSendBuffer, SetPacketBytes, GetPacketBytes,
    SetWaitTime, TimeOut],
  String USING [AppendString, StringBoundsFault],
  System USING [NetworkAddress, SocketNumber, NetworkNumber, HostNumber];

OthelloNS: PROGRAM
  IMPORTS AddressTranslation, NSBuffer, ExtendedString, Inline, Profile, Process, Router,
    String, Socket, OthelloDefs =
  BEGIN OPEN OthelloDefs;

  EchoUser: PROC =
    BEGIN
    bytesPerBuffer: CARDINAL;
    funny, late:    LONG CARDINAL ← 0;
    recv, sent:     LONG CARDINAL ← 0;
    wrong:          LONG CARDINAL ← 0;
    me, where:      System.NetworkAddress;
    mySoc:          Socket.ChannelHandle;
    packetNumber:   CARDINAL ← 0;
    pleaseStop:     BOOLEAN ← FALSE;
    routing:        CARDINAL;
    Watch:          PROC = {[] ← ReadChar[]; pleaseStop ← TRUE};
    PrintErrorNS:  PROC [b: NSBuffer.Buffer] = {
      body: NSBuffer.Body ← b.ns;
      source: System.NetworkAddress ← body.source;
      NewLine[];
      IF body.packetType = error THEN {
        len: CARDINAL = body.pktLength;
        WriteString["[Error packet, code="L];
        WriteOctal[LOOPHOLE[body.errorType]];
        WriteString[", from: "L];
        PrintNSAddress[@source];
        WriteString["] "L];
        FOR i: CARDINAL IN [0..len - NSTypes.wordsPerIDPHeader) DO
          WriteChar[Inline.LowByte[body.errorBody[i]]];
          WriteChar[Inline.HighByte[body.errorBody[i]]];
          ENDLOOP}
      ELSE {
        WriteString[" ***** "L];
        WriteString["Funny packet type = "L];
        WriteOctal[LOOPHOLE[body.packetType]];
        WriteString[" ***** "L]};
      NewLine[]};

    identity: Auth.IdentityHandle ← NIL;
    getID: PROC [id: Auth.IdentityHandle] = {identity ← id};

    GetName["Echo to: "L, @echoName];
    Profile.GetID[simple, getID];
    [where, ] ← AddressTranslation.StringToNetworkAddress[echoName, identity !
      AddressTranslation.Error => {
        msg: LONG STRING ← [100];
        appendErrorMsg: Format.StringProc = {
          String.AppendString[msg, s ! String.StringBoundsFault => RESUME[NIL]]};
        AddressTranslation.PrintError[error: errorRecord, proc: appendErrorMsg];
        OthelloDefs.AbortingCommand[msg]}];
    where.socket ← NSConstants.echoerSocket;
    routing ← Router.GetDelayToNet[where.net];
    IF routing = Router.infinity THEN
      AbortingCommand["Can't reach that network"L];

    me ← Socket.AssignNetworkAddress[];
    mySoc ← Socket.Create[me.socket];
    Socket.SetWaitTime[mySoc, 2000];  --two second timeout
    WriteString[".   ["L]; PrintNSAddress[@me]; WriteString["] => ["L];

    PrintNSAddress[@where];
    WriteChar[']]; NewLine[];
    Process.Detach[FORK Watch[]];
    bytesPerBuffer ← NSTypes.maxDataBytesPerEcho;

    UNTIL pleaseStop DO

      FOR len: CARDINAL IN [4..bytesPerBuffer] UNTIL pleaseStop DO

OthelloNS.mesa        23-Jan-87 12:15:58 PST
```

```
        b: NSBuffer.Buffer ← Socket.GetSendBuffer[mySoc];
        body: NSBuffer.Body ← b.ns;
        body.destination ← where;
        body.packetType ← echo; body.echoType ← echoRequest;
        Socket.SetPacketBytes[b, len];
        FOR i: CARDINAL IN [4..len - 4) DO body.echoBytes[i] ← i; ENDLOOP;
        body.echoWords[0] ← body.echoWords[1] ← (packetNumber ← packetNumber + 1);
        Socket.PutPacket[mySoc, b]; sent ← sent + 1;

        Process.Yield[];  -- be sure we don't hog machine

        BEGIN
        b ← Socket.GetPacket[mySoc ! Socket.TimeOut => GOTO late];
        SELECT TRUE FROM
          (body.packetType # echo) =>
            {funny ← funny + 1; PrintErrorNS[b]};
          (body.echoWords[0] # packetNumber) => {WriteChar['#]; late ← late + 1};
          (body.echoWords[1] # packetNumber) => {WriteChar['#]; late ← late + 1};
          (len # Socket.GetPacketBytes[b]) => {WriteChar['#]; late ← late + 1};
          ENDCASE =>
            FOR i: CARDINAL IN [4..len - 4) DO
              IF body.echoBytes[i] # (i MOD 400B) THEN
                {wrong ← wrong + 1; WriteChar['~]; EXIT};
              REPEAT FINISHED => {WriteChar['!]; recv ← recv + 1};
              ENDLOOP;
        NSBuffer.ReturnBuffer[b];
        EXITS late => {WriteChar['?]; late ← late + 1};
        END;

        ENDLOOP;

      NewLine[];
      ENDLOOP;

    Socket.Delete[mySoc];
    WriteString["Out: "L];
    WriteLongNumber[sent];
    WriteString[", In: "L];
    WriteLongNumber[recv];
    WriteString[" ("L];
    WriteLongNumber[(recv*100)/sent];
    WriteLine["%)"L];
    IF late # 0 THEN {
      WriteString["Late: "L]; WriteLongNumber[late];
      WriteString[" ("L]; WriteLongNumber[(late*100)/sent]; WriteLine["%)"L]};
    IF funny # 0 THEN {WriteLongNumber[funny]; WriteLine[" funny"L]};
    IF wrong # 0 THEN {WriteLongNumber[wrong]; WriteLine[" wrong data"L]};
    END;

  PrintLocalRoutingTable: PROC =
    BEGIN
    string: STRING ← [20];
    net: System.NetworkNumber;
    Router.FillRoutingTable[Router.infinity];  --load 'em up
    Process.Pause[Process.SecondsToTicks[2]];
    FOR hop: CARDINAL IN[0..Router.infinity] DO
      net ← Router.EnumerateRoutingTable[Router.startEnumeration, hop];
      IF net = Router.endEnumeration THEN LOOP;  --don't print empties
      WriteString["Networks "L];
      WriteLongNumber[LONG[hop]];
      WriteString[" hops away = {"L];
      UNTIL net = Router.endEnumeration DO
        ExtendedString.AppendNumber[@net, SIZE[System.NetworkNumber], 8, string];
        WriteString[string]; string.length ← 0; WriteChar['B];
        net ← Router.EnumerateRoutingTable[net, hop];
        IF net # Router.endEnumeration THEN WriteString[", "L];
        ENDLOOP;
      WriteChar['}];
      NewLine[];
      ENDLOOP;
    Router.FillRoutingTable[0];  --shut down the table
    END;  --PrintLocalRoutingTable

  PrintNSAddress: PROC [a: POINTER TO System.NetworkAddress] =
    BEGIN
    buffer: STRING ← [50];
    ExtendedString.AppendNumber[@a.net, SIZE[System.NetworkNumber], 8, buffer];
    buffer[buffer.length] ← '.; buffer.length ← buffer.length + 1;
    ExtendedString.AppendNumber[@a.host, SIZE[System.HostNumber], 8, buffer];
    buffer[buffer.length] ← '.; buffer.length ← buffer.length + 1;
    ExtendedString.AppendNumber[@a.socket, SIZE[System.SocketNumber], 8, buffer];
    WriteString[buffer];
    END;

  echoName: LONG STRING ← NIL;
  Commands: PROC [index: CARDINAL] = {
    SELECT index FROM
      0 => {
        MyNameIs[
          myNameIs: "Echo User"L,
          myHelpIs: "Echo user"L];
        EchoUser[]};
      1 => {
        MyNameIs[
          myNameIs: "Routing Tables"L,
          myHelpIs: "Show NS network routing tables"L];
        PrintLocalRoutingTable[]};
```

```
    ENDCASE => IndexTooLarge};

commandProcessor: CommandProcessor ← [Commands];
-- initialization
RegisterCommandProc[@commandProcessor];

END.....
```

```
-- File: OthelloNSFTP.mesa - last edit:
-- WFS           5-Jun-86 14:20:34
-- bjd          23-Aug-85 16:18:16
--   lgr         13-Feb-84 15:31:26
--   rkj         24-Feb-84 18:36:31

-- Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.

DIRECTORY
  AddressTranslation USING [Error, PrintError, StringToNetworkAddress],
  Auth USING [FreeIdentity, IdentityHandle, MakeIdentity],
  --CH USING [MakeRhs],
  Courier USING [ErrorCode, Error],
  NSErrorMsg USING [PostCourierError, PostNSFileError],
  Heap USING [systemZone],
  Format USING [StringProc],
  NSDataStream USING [Abort, Aborted, Handle, SourceStream],
  NSFile USING [
    AttributesProc, AttributesRecord, Close, Error, ErrorRecord, Find, GetAttributes,
    Handle, List, Logoff, LogonDirect, maxStringLength, nullHandle, nullSession, Open,
    Retrieve, Scope, Selections, ServiceRecord, Session, String, Time],
  NSName USING [
    AppendNameToString, Error, FreeNameFields, maxDomainLength, maxFullNameLength,
    maxOrgLength, Name, NameFieldsFromString, NameRecord, String],
  OthelloDefs USING [
    AbortingCommand, CheckUserAbort, CommandProcessor, GetName,
    IndexTooLarge, MyNameIs, RegisterCommandProc, SetCommandString,
    SetCursor, WriteChar, WriteLine, WriteString],
  OthelloFetch USING [
    Destination, directory, Handle, GrabBitsFromStream, Object, Register, Select,
    StartFeedback],
  Profile USING [GetDefaultDomain, GetDefaultOrganization, GetUser, String],
  Stream USING [Delete, Handle],
  String USING [
    AppendChar, AppendCharAndGrow, AppendNumber, AppendString, AppendStringAndGrow,
    CopyToNewString, Empty, Length, StringBoundsFault, SubString, SubStringDescriptor],
  Time USING [Append, Unpack];

OthelloNSFTP: PROGRAM
  IMPORTS
    AddressTranslation, Auth, --CH,-- Courier, NSErrorMsg, Heap,
    NSDataStream, NSFile,
    NSName, OthelloDefs, OthelloFetch, Profile, Stream, String, Time =
  BEGIN

  host: LONG STRING ← NIL;

  nsFileSession: NSFile.Session ← NSFile.nullSession;

  z: UNCOUNTED ZONE = Heap.systemZone;

  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- String/Credentials Commands
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
--  I don't believe we need this proc anymore; It no longer make sense to allow network addresses; must get to Auth server anyway;

  Qualify: PROC [token: LONG STRING] RETURNS [newToken: LONG STRING] = {
    octalAddress: BOOLEAN ← TRUE; -- only '0..'7 and '# allowed
    chChar: CHARACTER = ':;
    defaultDomain, defaultOrganization: LONG STRING ← NIL;
    GetDomain: PROC[domain: LONG STRING] =
      {IF domain # NIL THEN
        String.AppendStringAndGrow[@defaultDomain, domain, z]};
    GetOrg: PROC[org: LONG STRING] =
      {IF org # NIL THEN
        String.AppendStringAndGrow[@defaultOrganization, org, z]};
    IF String.Length[token] = 0 THEN RETURN[NIL];
    FOR i: CARDINAL IN [0..token.length) DO
       SELECT token[i] FROM
         chChar => {RETURN[String.CopyToNewString[token, z]]}; -- already qualified
         IN['0..'7], '# => NULL;
         ENDCASE => octalAddress ← FALSE;
       ENDLOOP;
    newToken ← String.CopyToNewString[token, z];
    IF octalAddress THEN RETURN;
    Profile.GetDefaultDomain[GetDomain];
    Profile.GetDefaultOrganization[GetOrg];
    IF String.Length[defaultDomain] > 0 OR
      String.Length[defaultOrganization] > 0 THEN {
          String.AppendCharAndGrow[@newToken, chChar, z];
          String.AppendStringAndGrow[@newToken, defaultDomain, z];
          String.AppendCharAndGrow[@newToken, chChar, z];
          String.AppendStringAndGrow[@newToken, defaultOrganization, z]};
    z.FREE[@defaultDomain];
    z.FREE[@defaultOrganization]};

  DoIndirect: PROC [cmFile: LONG STRING] RETURNS [mine: BOOLEAN] =
    BEGIN
    fileName: LONG STRING ← NIL;
    ParseCmFileName: PROC = {
      hostEnd: CARDINAL;
      IF cmFile.length = 0 THEN RETURN;
      FOR i: CARDINAL IN [0..cmFile.length) DO
        c: CHARACTER = cmFile[i];
        SELECT c FROM
          '[ => LOOP; '] => {hostEnd ← i; EXIT};
          ENDCASE => String.AppendCharAndGrow[@host, c, z];
```

```
        REPEAT FINISHED => {z.FREE[@host]; RETURN}
        ENDLOOP;
      -- hostEnd points at ']
      FOR i: CARDINAL IN (hostEnd..cmFile.length) DO
        IF cmFile[i] = '< AND OthelloFetch.directory#NIL THEN
          OthelloFetch.directory.length ← 0;
        String.AppendCharAndGrow[@fileName, cmFile[i], z];
        IF cmFile[i] = '> THEN {
          String.AppendStringAndGrow[
            @OthelloFetch.directory, fileName, z];
          fileName.length ← 0};
        ENDLOOP};
    s:          LONG STRING ← NIL;
    GetString: PROC [c: LONG STRING] = {s ← String.CopyToNewString[c, z]};

    IF cmFile[0] # '[ THEN RETURN [FALSE];
    z.FREE[@host];
    z.FREE[@OthelloFetch.directory];
    ParseCmFileName[];
    OthelloFetch.Select[@fetcher]; Open[];
    Retrieve[fileName, [string[GetString]]
      ! UNWIND => {z.FREE[@s]; z.FREE[@fileName]}];
    OthelloDefs.WriteLine["done"L];
    OthelloDefs.SetCommandString[s];
    z.FREE[@fileName];
    RETURN[TRUE]
    END;

  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- MISC Stuff/Commands
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  userOpened: BOOLEAN ← FALSE;
  OpenCmd: PROC = {
    OthelloDefs.MyNameIs[
      myNameIs: "Open Connection"L,
      myHelpIs: "Open connection to file service"L];
    OthelloFetch.Select[@fetcher];
    OthelloDefs.GetName["Open connection to "L, @host];
    Open[]; userOpened ← TRUE};

  ReOpen: PROC RETURNS [BOOLEAN] = {
    IF userOpened=FALSE THEN RETURN[FALSE];
    Open[]; RETURN[TRUE]};

  RemoteList: PROC [fileName: LONG STRING] = {
    IF ~ConnectionOpen[] AND ~ReOpen[] THEN
      OthelloDefs.AbortingCommand["Please open a connection"L];
    ListFiles[IF String.Length[fileName] = 0 THEN "*"L ELSE fileName]};

  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- Central commands
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  commandProcessor: OthelloDefs.CommandProcessor ← [FtpCommands];

  FtpCommands: PROC [index: CARDINAL] = {
    SELECT index FROM
      0 => OpenCmd[];
     ENDCASE => OthelloDefs.IndexTooLarge};

  fetcher: OthelloFetch.Object ← [
    Retrieve: Retrieve,
    DoIndirect: DoIndirect,
    List: RemoteList,
    Close: Close];


  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- file retrieval Stuff/Commands
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  ConnectionOpen: PROC RETURNS [BOOLEAN] = {
    RETURN[nsFileSession # NSFile.nullSession]};

  -- all callers close the connection first
  Open: PROC = {
    clientDefaultsRecord: NSName.NameRecord;
    defaultCHOrg: LONG STRING = [NSName.maxOrgLength];
    defaultCHDomain: LONG STRING = [NSName.maxDomainLength];
    serviceName: LONG STRING ← [NSName.maxFullNameLength];
    serviceRec: NSFile.ServiceRecord ← [];
    nameRecord: NSName.NameRecord ← [];
    id: Auth.IdentityHandle ← NIL;

    GetDomain: PROC[domain: LONG STRING] = {
      String.AppendString[defaultCHDomain, domain !
        String.StringBoundsFault => RESUME[NIL]]};
    GetOrg: PROC[org: LONG STRING] = {
      String.AppendString[defaultCHOrg, org !
        String.StringBoundsFault => RESUME[NIL]]};
    AppendNameToString: PROCEDURE [s: LONG STRING, name: NSName.Name] = {
      newS: NSName.String ← NSName.AppendNameToString[s: S[s], name: name];
      s.length ← newS.length};
    CopyUserAndPassword: PROCEDURE[name, password: LONG STRING] = {
      userName ← String.CopyToNewString[name, z];
      userPassword ← String.CopyToNewString[password, z];
     };
    Cleanup: PROCEDURE = {
      Auth.FreeIdentity[@id, z];
```

```
      z.FREE[@userName]; z.FREE[@userPassword];
      NSName.FreeNameFields[z, @serviceRec.name];
      NSName.FreeNameFields[z, @nameRecord]};

    userName, userPassword: Profile.String ← NIL;

    IF nsFileSession # NSFile.nullSession THEN RETURN;
    Profile.GetDefaultDomain[GetDomain];
    Profile.GetDefaultOrganization[GetOrg];
    clientDefaultsRecord ← [domain: S[defaultCHDomain], org: S[defaultCHOrg]];
    NSName.NameFieldsFromString[
      z: z, s: S[host], destination: @serviceRec.name,
      clientDefaults: @clientDefaultsRecord !
        NSName.Error => OthelloDefs.AbortingCommand["Illegal host name"L]];
    AppendNameToString[serviceName, @serviceRec.name];
    Profile.GetUser[CopyUserAndPassword, clearinghouse];
    NSName.NameFieldsFromString[
      z: z, s: S[userName], destination: @nameRecord,
      clientDefaults: @clientDefaultsRecord !
        UNWIND => NSName.FreeNameFields[z, @serviceRec.name];
        NSName.Error => OthelloDefs.AbortingCommand["Illegal login name"L]];
    id ← MakeIdentity[name: @nameRecord, password: userPassword];
    {ENABLE UNWIND => Cleanup[];
    serviceRec.systemElement ← AddressTranslation.StringToNetworkAddress[
      s: serviceName, id: id !
      AddressTranslation.Error => {
        msg: LONG STRING ← [100];
        appendErrorMsg: Format.StringProc = {
          String.AppendString[msg, s ! String.StringBoundsFault => RESUME[NIL]]};
        AddressTranslation.PrintError[error: errorRecord, proc: appendErrorMsg];
        OthelloDefs.AbortingCommand[msg]}].addr;
    nsFileSession ← NSFile.LogonDirect[
      identity: id, service: @serviceRec
        ! NSFile.Error => NSError[error];
          Courier.Error => CourierError[errorCode]];
     Cleanup[]}};

  MakeIdentity: PROC [name: NSName.Name, password: LONG STRING]
    RETURNS [ident: Auth.IdentityHandle] = {
    ident ← Auth.MakeIdentity[
      myName: name, password: S[password],
      z: z, style: simple, dontCheck: TRUE]};


  Close: PROC = {
    IF ~ConnectionOpen[] THEN RETURN;
    NSFile.Logoff[nsFileSession
      ! NSFile.Error => NSError[error];
        Courier.Error => CourierError[errorCode];
        OthelloDefs.AbortingCommand => {
          OthelloDefs.WriteString[reason];
          OthelloDefs.WriteLine[reasonOne];
          CONTINUE}];
    nsFileSession ← NSFile.nullSession;
    OthelloDefs.WriteLine["connection closed"L]};

  -- could mess with directories.
  -- who cares
  ListFiles: PROC [pattern: LONG STRING]= {
    scope: NSFile.Scope ← [];
    selections: NSFile.Selections ← [];
    fh: NSFile.Handle;
    dir: LONG STRING ← NIL;
    wildCardInFileName: BOOLEAN ← FALSE;
    name: LONG STRING ← NIL;
    ss: String.SubStringDescriptor ← [base: NIL, offset: 0, length: 0];
    ListOne: NSFile.AttributesProc = {
      version: LONG STRING ← [20];
      time: LONG STRING ← [20];
      Time.Append[time, Time.Unpack[attributes.createdOn]];
      String.AppendChar[version, '!];
      String.AppendNumber[version, attributes.version];
    <<  FOR i: CARDINAL IN [0..attributes.pathname.length) DO
        OthelloDefs.WriteChar[VAL[attributes.pathname.bytes[i]]];
        ENDLOOP;>>
      OthelloDefs.WriteString[dir];
      FOR i: CARDINAL IN [0..attributes.name.length) DO
        OthelloDefs.WriteChar[VAL[attributes.name.bytes[i]]];
        ENDLOOP;
      OthelloDefs.WriteString[version];
      THROUGH [dir.length + attributes.name.length + version.length..80-time.length)
       DO OthelloDefs.WriteChar[' ] ENDLOOP;
      OthelloDefs.WriteLine[time];
      --OthelloDefs.WriteChar[' ];
      --OthelloDefs.WriteLine[info.author];
      --OthelloDefs.WriteChar[' ];
      --OthelloDefs.WriteLongNumber[info.size];
      --OthelloDefs.WriteLine[" bytes]"L];
      OthelloDefs.CheckUserAbort[];
      RETURN};
    dir ← z.NEW[StringBody[60]];
    IF pattern[0] # '< AND String.Length[OthelloFetch.directory] # 0 THEN {
      String.AppendStringAndGrow[@dir, OthelloFetch.directory, z];
      IF dir[dir.length - 1] # '> THEN
        String.AppendCharAndGrow[@dir, '>, z]};
    String.AppendStringAndGrow[@dir, pattern, z];
    ss.base ← dir;
```

```
    FOR i: CARDINAL DECREASING IN [0..dir.length) DO
      SELECT dir[i] FROM
        '* => wildCardInFileName ← TRUE;
        '> => {ss.length ← i + 1; EXIT};
        ENDCASE;
      ENDLOOP;
    name ← z.NEW[StringBody[dir.length - ss.length]];
    FOR i: CARDINAL IN [ss.length .. dir.length) DO
      String.AppendChar[name, dir[i]];
      ENDLOOP;
    dir.length ← ss.length;
    IF HasWildCard[dir] THEN {
      z.FREE[@dir];
      z.FREE[@name];
      OthelloDefs.AbortingCommand["No wild cards in directories."L]};
    fh ← GetFileFromSS[ss];
    IF fh = NSFile.nullHandle THEN {
      z.FREE[@name]; z.FREE[@dir]; RETURN};
    scope.filter ← IF wildCardInFileName THEN
      [matches[[name [S[name]]]]]
    ELSE [equal[[name [S[name]]]]];
    selections.interpreted[name] ← TRUE;
    selections.interpreted[version] ← TRUE;
    selections.interpreted[createdOn] ← TRUE;
    selections.interpreted[pathname] ← TRUE; -- not yet implemented
    NSFile.List[
      directory: fh, proc: ListOne, selections: selections, scope: scope,
      session: nsFileSession
        ! NSFile.Error => NSError[error];
          Courier.Error => CourierError[errorCode];
          UNWIND => {z.FREE[@name]; z.FREE[@dir]}];
    z.FREE[@dir];
    z.FREE[@name];
    };

  Retrieve: PROC [
    fileName: LONG STRING, destination: OthelloFetch.Destination] = {
    size: LONG CARDINAL;
    name: LONG STRING ← NIL;
    fH: NSFile.Handle ← NSFile.nullHandle;
    Sink: PROC [source: NSDataStream.SourceStream] =
      BEGIN ENABLE {
        NSDataStream.Aborted => {Stream.Delete[source]; CONTINUE};
        UNWIND => Stream.Delete[source]};
      OthelloFetch.GrabBitsFromStream[source, size, destination, name !
          OthelloFetch.StartFeedback => {
            OthelloDefs.WriteString["Fetching..."L];
            OthelloDefs.SetCursor[ftp];
            RESUME};
          UNWIND => {
            OthelloDefs.SetCursor[pointer];
            NSDataStream.Abort[source ! NSDataStream.Aborted => CONTINUE]}];
      Stream.Delete[source ! NSDataStream.Aborted => CONTINUE]
      END;
    [fH, size, name] ← GetFile[fileName];
    NSFile.Retrieve[fH, [proc [Sink]], nsFileSession
      ! UNWIND =>
          {NSFile.Close[
              fH, nsFileSession ! NSFile.Error, Courier.Error => CONTINUE];
           z.FREE[@name]}];
    NSFile.Close[fH, nsFileSession
      ! NSFile.Error, Courier.Error => CONTINUE;
      UNWIND => z.FREE[@name]];
    z.FREE[@name];
    OthelloDefs.SetCursor[pointer]};

  GetFile: PROC [fileName: LONG STRING] RETURNS [fh: NSFile.Handle, size: LONG CARDINAL, name: LONG STRING] =
    BEGIN
    time: LONG STRING ← [20];
    attributes: NSFile.AttributesRecord;
    ss: String.SubStringDescriptor ← [base: NIL, offset: 0, length: 0];
    name ← z.NEW[StringBody[60]];
    String.AppendChar[name, '[];
    String.AppendStringAndGrow[@name, host, z];
    String.AppendCharAndGrow[@name, '], z];
    ss.offset ← name.length;
    IF fileName[0] # '< AND ~String.Empty[OthelloFetch.directory] THEN {
      String.AppendStringAndGrow[@name, OthelloFetch.directory, z];
      IF name[name.length - 1] # '> THEN
        String.AppendCharAndGrow[@name, '>, z]};
    String.AppendStringAndGrow[@name, fileName, z];
    ss.base ← name;
    ss.length ← name.length - ss.offset;
    fh ← GetFileFromSS[ss];
    NSFile.GetAttributes[
      fh, [[createdOn: TRUE, sizeInPages: TRUE]],
      @attributes, nsFileSession
        ! NSFile.Error => NSError[error];
          Courier.Error => CourierError[errorCode]];
    String.AppendStringAndGrow[@name, " ("L, z];
    Time.Append[time, Time.Unpack[attributes.createdOn]];
    String.AppendStringAndGrow[@name, time, z];
    String.AppendCharAndGrow[@name, '), z];
    size ← attributes.sizeInPages;
    END;

  GetFileFromSS: PROCEDURE [ss: String.SubStringDescriptor] RETURNS [fh: NSFile.Handle ← NSFile.nullHandle] = {
```

```
    ENABLE {NSFile.Error => NSError[error];
            Courier.Error => CourierError[errorCode]};
     tempName: STRING = [NSFile.maxStringLength];
     parent: NSFile.Handle ← NSFile.Open[attributes: NIL, session: nsFileSession];
     DO
       GetRoot[@ss, tempName];
       fh ← NSFile.Find[
         directory: parent,
         scope: [
           direction: backward, filter: [equal[[name[S[tempName]]]]]],
         controls: [timeout: 4], session: nsFileSession !
           UNWIND => NSFile.Close[parent, nsFileSession !
             NSFile.Error, Courier.Error => CONTINUE]];
       NSFile.Close[parent, nsFileSession];
       IF ss.length = 0 THEN EXIT;
       parent ← fh
       ENDLOOP;
     };

   GetRoot: PROC [fileName: String.SubString, root: LONG STRING] = {
     OPEN fileName;
     StripChar: PROC = INLINE {offset ← offset + 1;  length ← length - 1};
     N: CARDINAL ← offset + length;
     quote: CHARACTER = '';
     i: CARDINAL ← offset;
     root.length ← 0;
     WHILE i < N DO
       SELECT base[i] FROM
         quote =>  {
           i ← i+1; StripChar[]; IF i = N THEN EXIT;
           String.AppendChar[root, base[i]]};
         '>, '/ =>  EXIT;
         '< =>  root.length ← 0;
         ENDCASE => String.AppendChar[root, base[i]];
       i ← i+1;
       StripChar[];
       ENDLOOP;
     IF fileName.length = 0 THEN RETURN;
     StripChar[]};

   S: PROCEDURE [s: LONG STRING] RETURNS [NSFile.String] = {
     IF s = NIL THEN RETURN[[bytes: NIL, length: 0, maxlength: 0]];
     RETURN[[bytes: LOOPHOLE[@s.text], length: s.length, maxlength: s.maxlength]]};

   HasWildCard: PROC [s: LONG STRING] RETURNS [BOOLEAN] = {
     IF s#NIL THEN FOR i: CARDINAL IN [0..s.length) DO
       IF s[i] = '* THEN RETURN[TRUE] ENDLOOP;
     RETURN[FALSE]};


   NSError: PROC [error: NSFile.ErrorRecord] =
     BEGIN
     post: Format.StringProc = {OthelloDefs.WriteString[s]};
     NSErrorMsg.PostNSFileError[error, post];
     OthelloDefs.AbortingCommand[NIL];
     END;

   CourierError: PROC [error: Courier.ErrorCode] =
     BEGIN
     post: Format.StringProc = {OthelloDefs.WriteString[s]};
     NSErrorMsg.PostCourierError[error, post];
     OthelloDefs.AbortingCommand[NIL];
     END;

   <<StartCH: PROCEDURE = {
     frame: PROGRAM ← Runtime.GlobalFrame[LOOPHOLE[CH.MakeRhs]];
     START frame};>>

   --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   -- initialization
   --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   OthelloDefs.RegisterCommandProc[@commandProcessor];
   OthelloFetch.Register[@fetcher];
   --StartCH[];

   END.....


Log
NFS      4-Jun-86 13:04:55      Adapted for OthelloTool.
```

```
-- Copyright (C) 1986 by Xerox Corporation. All rights reserved.
-- OthelloToolDefs.mesa
-- Created by NFS          4-Jun-86 10:49:17

DIRECTORY
  TTY USING [Handle],
  Volume USING [ID];
OthelloToolDefs:DEFINITIONS = {

  tty: TTY.Handle;

  Run: PROCEDURE;  -- instead of PilotClient.Run

  CloseVolume: PROCEDURE[volume: Volume.ID];

}.
```

```
-- Copyright (C) 1986 by Xerox Corporation. All rights reserved.
-- OthelloToolImpl.mesa
-- Created by NFS        4-Jun-86 10:26:30

DIRECTORY
  Exec USING [AddCommand, ExecProc, RemoveCommand],
  OthelloToolDefs USING [Run],
  Process USING [Abort],
  Runtime USING [GetBcdTime],
  String USING [AppendString],
  Time USING [Append, Unpack],
  Tool USING [Create, Destroy, MakeSWsProc, MakeTTYSW, UnusedLogName],
  ToolWindow USING [Activate, TransitionProcType],
  TTY USING [Handle],
  TTYSW USING [GetTTYHandle],
  Version USING [Append],
  Volume USING [Close, ID, systemID],
  Window USING [Handle];
OthelloToolImpl: PROGRAM
  IMPORTS
    Exec, Runtime, String, OthelloToolDefs, Process, Time,
    Tool, ToolWindow, TTYSW, Version, Volume
  EXPORTS OthelloToolDefs = {

  tty: PUBLIC TTY.Handle;

  toolWindow: Window.Handle;

  CommandInterpreter: PROCESS;

  Init: PROCEDURE = {
    name: LONG STRING ← [75];
    name.length ← 0;
    String.AppendString[name, "OthelloTool "L];
    Version.Append[name];
    String.AppendString[name," of "L];
    Time.Append[name,Time.Unpack[Runtime.GetBcdTime[]]];
    toolWindow ← Tool.Create[
      name: name, makeSWsProc: MakeTTYSW, clientTransition: Stop,
      cmSection: "OthelloTool"L, tinyName1: "Othello"L, tinyName2: "Tool"L];
    Exec.AddCommand[name: "OthelloTool.~"L, proc: Activate, unload: DestroyTool];
  };

  Stop: ToolWindow.TransitionProcType = {
    IF new = inactive THEN {
      Process.Abort[CommandInterpreter];
      JOIN CommandInterpreter;};
  };

  Activate: Exec.ExecProc = {ToolWindow.Activate[toolWindow];};

  DestroyTool: Exec.ExecProc = {
    Exec.RemoveCommand[h, "OthelloTool.~"L];
    Tool.Destroy[toolWindow];
  };

  MakeTTYSW: Tool.MakeSWsProc = {
    logName: LONG STRING ← [20];
    ttySW: Window.Handle;
    logName.length ← 0;
    Tool.UnusedLogName[unused:logName, root: "OthelloTool.log"L];
    ttySW ← Tool.MakeTTYSW[window:window, name:logName];
    tty ← TTYSW.GetTTYHandle[ttySW];
    CommandInterpreter ← FORK OthelloToolDefs.Run[];
  };

  CloseVolume: PUBLIC PROCEDURE[volume: Volume.ID] = {
    IF volume # Volume.systemID THEN Volume.Close[volume];};

  Init[];

}.
```

```
-- File: VolumeInitCommandImpl.mesa - last edit:
-- NFS           4-Jun-86 11:23:33
-- BJD    .PA          15-Feb-85 16:22:59
--   RXJ        12-Sep-83 22:47:20
--   RSF        14-Dec-83 15:20:28
--   DWE        13-Jan-84 11:27:06
--   LXR        31-Jan-84 16:44:47

-- Copyright (C) 1981, 1982, 1983, 1984 , 1985 by Xerox Corporation. All rights reserved.

-- This file is the command Processor.

DIRECTORY
  File  USING [Error, ErrorType, Unknown],
  Format USING [HostNumber, StringProc],
  Frame USING [Free, ReadLocalWord],
  Heap USING [systemZone],
  Inline USING [BITNOT, HighHalf, LowHalf],
  OthelloDefs,
  OthelloOps USING [
    GetTimeFromTimeServer, IsTimeValid,
    SetProcessorTime, TimeServerError],
  OthelloToolDefs USING [tty],
  PhysicalVolume USING [Error, ErrorType, NeedsScavenging],
  PilotClient USING [],
  PrincOps USING [frameSizeMap, LocalFrameHandle, LocalOverhead],
  Process USING [Pause, SecondsToTicks],
  Runtime USING [GetBcdTime, IsBound],
  SpecialRuntime USING [GetCurrentSignal],
  SpecialSpace USING [realMemorySize],
  SpecialSystem USING [GetProcessorID],
  Scavenger USING [Error, ErrorType],
  String USING [
    AppendChar, AppendCharAndGrow, AppendDecimal, AppendLongNumber,
    EquivalentSubString, InvalidNumber, StringBoundsFault, StringToNumber,
    SubStringDescriptor, UpperCase],
  System USING [
    GetGreenwichMeanTime, GreenwichMeanTime, gmtEpoch,
    LocalTimeParameters, GetLocalTimeParameters, SetLocalTimeParameters],
  TTY USING [
    BlinkDisplay, CharsAvailable, GetChar, Handle,
    PutChar, PutString, RemoveCharacter, ResetUserAbort, UserAbort],
  Time USING [
    Append, defaultTime, Invalid, Pack, Unpack, Unpacked, useGMT, useSystem],
  UserTerminal USING [
    CursorArray, GetCursorPattern, SetCursorPattern],
  Version USING [Append],
  VersionExtras USING [AppendCopyright],
  Volume USING [
    InsufficientSpace, NeedsScavenging, NotOpen, ReadOnly, Unknown],
  VolumeConversion USING [Error, ErrorType];

UtilityPilotClientImpl: PROGRAM
  IMPORTS
    File, Format, Frame, Heap, Inline, OthelloDefs, OthelloOps, OthelloToolDefs,
    PhysicalVolume, Process, Runtime, SpecialRuntime, SpecialSpace, SpecialSystem,
    Scavenger, String, System, Time, TTY, UserTerminal,
    Version, VersionExtras, Volume, VolumeConversion
  EXPORTS OthelloDefs, OthelloToolDefs =
  BEGIN

  MyNameIs:         PUBLIC SIGNAL [
    myNameIs: LONG STRING, myHelpIs: LONG STRING] = CODE;
  AbortingCommand: PUBLIC ERROR [
    reason: LONG STRING, reasonOne: LONG STRING ← NIL] = CODE;
  IndexTooLarge:    PUBLIC ERROR  = CODE;
  Question:         PUBLIC SIGNAL = CODE;
  TryAgain:         PUBLIC SIGNAL = CODE;

  BS:       CHARACTER = 10C;
  ControlA: CHARACTER = 'A - 100B;
  ControlP: CHARACTER = 'P - 100B;
  ControlW: CHARACTER = 'W - 100B;
  CR:       CHARACTER = 15C;
  DEL:      CHARACTER = 177C;
  ESC:      CHARACTER = 33C;
  SP:       CHARACTER = ' ;
  NUL:      CHARACTER = 0C;

  CommandProcessor: TYPE = OthelloDefs.CommandProcessor;

  -- ~~~~~~~~~~~~~~~~~~~~~~~~
  -- Commands
  -- ~~~~~~~~~~~~~~~~~~~~~~~~
  CurrentComand: SIGNAL RETURNS [
    proc: PROC [index: CARDINAL], index: CARDINAL] = CODE;

  ForAllCommandProcs: PROC [P: PROC[LONG STRING]] = {
    FOR c: LONG POINTER TO CommandProcessor ← commands, c.next WHILE c # NIL DO
      FOR i: CARDINAL IN [0..LAST[CARDINAL]) DO
        ENABLE CurrentComand => RESUME[c.proc, i];
        c.proc[i
        ! MyNameIs => {P[myNameIs]; CONTINUE};
          IndexTooLarge => EXIT];
        ENDLOOP ENDLOOP};

  Help: PROC = {
```

```
    WidthProc: PROC [s: LONG STRING] = {tabWidth ← MAX[tabWidth, s.length]};
    tabWidth: CARDINAL ← 0;
    SIGNAL MyNameIs[myNameIs: "Help"L, myHelpIs: "Type this table"L];
    ForAllCommandProcs[WidthProc];
    tabWidth ← tabWidth + 4;
    FOR c: LONG POINTER TO CommandProcessor ← commands, c.next WHILE c # NIL DO
      FOR i: CARDINAL IN [0..LAST[CARDINAL]) DO
        c.proc[i
        ! MyNameIs => {
            WriteString[myNameIs];
            THROUGH [myNameIs.length..tabWidth) DO WriteChar[' ] ENDLOOP;
            WriteLine[myHelpIs];
            CONTINUE};
          IndexTooLarge => EXIT];
        ENDLOOP ENDLOOP;
    WriteLine[
      "In General, Del will abort current command, ? will explain options"L]};

  TimeUser: PROC [index: CARDINAL] = {
    SELECT index FROM
      0 => {
        SIGNAL MyNameIs[myNameIs: "Time"L, myHelpIs: "Time of day"L];
        WriteString["Current time"L]; WriteTime[Time.defaultTime, TRUE]};
      1 =>
        Help[];
      ENDCASE =>
        ERROR IndexTooLarge};

  RegisterCommandProc: PUBLIC PROC [
    commandProc: LONG POINTER TO CommandProcessor] = {
    commandProc.next ← commands; commands ← commandProc};

  commands: LONG POINTER TO CommandProcessor ← @helpCommandProcessor;
  helpCommandProcessor: CommandProcessor ← [TimeUser, NIL];

  -- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- Basic command processing
  -- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  CollectCommand: PROC RETURNS [
    p: PROC [index: CARDINAL], index: CARDINAL] = {
    ExplainOptions: PROC = {
      first: BOOLEAN ← TRUE;
      WriteChar['?];
      IF userString.length # 0 THEN {
        P: PROC [s: LONG STRING] = {
          IF HeadMatch[s, userString.length] THEN {
            WriteString[IF first THEN "\rCurrent Options Are: "L ELSE ", "L];
            WriteString[s]; first ← FALSE}};
        ForAllCommandProcs[P]};
      IF first THEN {  -- Didn't match... tell all
        P: PROC [s: LONG STRING] = {
          IF ~first THEN WriteString[", "L]; WriteString[s]; first ← FALSE};
        WriteString["\rValid Commands Are: "L];
        ForAllCommandProcs[P]};
      WriteString["\r> "L]; WriteString[userString]};
    FindAnswer: TYPE = RECORD [
      SELECT how: * FROM  none => NULL, many => NULL,
      one => [proc: PROC [index: CARDINAL], index: CARDINAL],
      ENDCASE];
    FindPossibles: PROC RETURNS [ans: FindAnswer ← [none[]]] = {
      P: PROC [matchString: LONG STRING] = {
        IF HeadMatch[matchString, head] THEN
          WITH ans SELECT FROM
            none => {
              ans ← [one[CurrentComand[].proc, CurrentComand[].index]];
              UNTIL userString.length = matchString.length DO
                userString[userString.length] ← matchString[userString.length];
                IF (userString.length ← userString.length + 1) = userString.maxlength THEN {
                  WriteLine[" Command too long!"L]; ERROR TryAgain}
                ENDLOOP};
            ENDCASE => {
              --ASSERT[head#0]
              FOR i : CARDINAL IN [head - 1..LAST[CARDINAL]) DO
                IF LowerCase[userString[i]] # LowerCase[matchString[i]] THEN {
                  userString.length ← i; EXIT};
                ENDLOOP;
              ans ← [many[]]}};
      head: CARDINAL ← userString.length;
      IF head = 0 THEN RETURN;
      ForAllCommandProcs[P];
      WHILE head # userString.length DO
        WriteChar[userString[head]]; head ← head + 1 ENDLOOP};
    HeadMatch: PROC [matchString: LONG STRING, head: CARDINAL]
      RETURNS [BOOLEAN] = {
      IF head > matchString.length THEN RETURN[FALSE];
      FOR i: CARDINAL IN [0..head) DO
        IF LowerCase[userString[i]] # LowerCase[matchString[i]] THEN
          RETURN[FALSE]
        ENDLOOP;
      RETURN[TRUE]};
    LowerCase: PROC [c: CHARACTER] RETURNS [CHARACTER] = {
      RETURN[IF c IN ['A..'Z] THEN c + ('a - 'A) ELSE c]};

    userString: STRING = [100];

    userString.length ← 0;
    WriteString["> "L];
```

```
    DO
      c: CHARACTER = ReadChar[];
      SELECT c FROM
        DEL => {WriteLine[" XXX"L]; ERROR TryAgain};
        BS, ControlA => IF userString.length # 0 THEN
          EraseTTYChar[userString[userString.length ← userString.length - 1]];
        ControlW =>
          IF userString.length # 0 THEN DO
            EraseTTYChar[userString[userString.length ← userString.length - 1]];
            IF userString.length=0 OR userString[userString.length - 1] = SP THEN EXIT
            ENDLOOP;
        '? => ExplainOptions[];
        CR, SP => {
          ans: FindAnswer = FindPossibles[];
          WITH theAns: ans SELECT FROM
            none => {
              IF Runtime.IsBound[LOOPHOLE[OthelloDefs.AlternateGetCMFile]]
                 AND userString.length > 1 AND userString[0] = '@ THEN {
                 NewLine[];
                 OthelloDefs.AlternateGetCMFile[userString
                 ! OthelloDefs.MyNameIs => RESUME];
                 ERROR TryAgain};
              IF prometheusBound THEN AbortingCommand["Script Error"L]
              ELSE BlinkDisplay[]};
            many => NULL;
            one  => RETURN[theAns.proc, theAns.index];
            ENDCASE => ERROR};
        ENDCASE =>
          IF (userString.length ← userString.length + 1) = userString.maxlength THEN {
            WriteLine[" Command too long!"L]; ERROR TryAgain}
          ELSE WriteChar[userString[userString.length - 1] ← c];
      ENDLOOP};

  -- ~~~~~~~~~~~~~~~~~~~~~~~~~
  -- Utility-Type Functions
  -- ~~~~~~~~~~~~~~~~~~~~~~~~~
  Confirm: PUBLIC PROC [how: OthelloDefs.ConfirmType ← once] = {
    IF CommandFileActive[] THEN RETURN;
    WriteString["Are you "L];
    IF how = thrice THEN WriteString["still "L];
    WriteString["sure? [y or n]: "L];
    DO
      c: CHARACTER = ReadChar[];
      SELECT c FROM
        'y, 'Y, CR  => {WriteLine["Yes"L]; EXIT};
        'n, 'N, DEL => {WriteLine["No"L]; ERROR TryAgain};
        ENDCASE =>     BlinkDisplay[];
      ENDLOOP;
    IF how = twice THEN {
      Process.Pause[Process.SecondsToTicks[3]]; FlushInput[]; Confirm[thrice]}};

  DebugAsk: PUBLIC PROC = {
    WriteString["\rType ControlP to muddle on........"L];
    WHILE ReadChar[] # ControlP DO ENDLOOP; NewLine[]};

  spacesInStringOK: BOOLEAN ← FALSE;
  GetName: PUBLIC PROC [
    prompt: LONG STRING ← NIL, dest: LONG POINTER TO LONG STRING,
    how: OthelloDefs.EchoNoEcho ← echo, signalQuestion: BOOLEAN ← FALSE] =
    BEGIN
    first: BOOLEAN ← TRUE;
    EraseChar: PROC = {
      IF dest.length = 0 THEN RETURN;
      dest.length ← dest.length - 1;
      EraseTTYChar[IF how = echo THEN dest[dest.length] ELSE '*];
      IF dest.length = 0 AND dest.maxlength > 20 THEN {
        Heap.systemZone.FREE[dest]; dest↑ ← Heap.systemZone.NEW[StringBody[10]]}};
    CWriteC: PROC [c: CHARACTER] = {WriteChar[IF how = echo THEN c ELSE '*]};
    CWriteString: PROC = {
      FOR i: CARDINAL IN [0..dest.length) DO CWriteC[dest[i]] ENDLOOP};

    IF dest↑ = NIL THEN dest↑ ← Heap.systemZone.NEW[StringBody[10]];
    WriteString[prompt]; CWriteString[];
    DO
      c: CHARACTER = ReadChar[];
      SELECT TRUE FROM
        c = BS, c = ControlA => EraseChar[];
       << (c = SP AND ~spacesInStringOK),>> c = CR => {NewLine[]; RETURN};
        c = DEL => {WriteLine[" XXX"L]; ERROR TryAgain};
        c = ControlW =>
          DO
            EraseChar[];
            IF dest.length=0 THEN EXIT;
            SELECT dest[dest.length-1] FROM
              IN ['a..'z], IN ['A..'Z], IN ['0..'9] => LOOP;
              ENDCASE => EXIT;
            ENDLOOP;
        c = '? AND signalQuestion => {
          SIGNAL Question; WriteString[prompt]; CWriteString[]; LOOP};
        c >= SP => {
          IF first THEN WHILE dest.length#0 DO EraseChar[] ENDLOOP;
          String.AppendCharAndGrow[dest, c, Heap.systemZone]; CWriteC[dest[dest.length-1]]};
        ENDCASE => BlinkDisplay[];
      first ← FALSE;
      ENDLOOP;
    END;
```

```
numberString: LONG STRING ← NIL;
ReadNumber: PUBLIC PROC [
  prompt: LONG STRING, min, max, default: LONG CARDINAL ← LAST[LONG CARDINAL]]
  RETURNS [ans: LONG CARDINAL] = {
  DO
    IF default # LAST[LONG CARDINAL] THEN {
      IF numberString=NIL THEN numberString ← Heap.systemZone.NEW[StringBody[15]];
      numberString.length ← 0; String.AppendLongNumber[numberString, default, 10]};
    WriteString[prompt]; WriteChar['[]; WriteLongNumber[min];
    WriteString[".."L]; WriteLongNumber[max]; WriteString["]: "L];
    GetName[dest: @numberString];
    ans ← 0;
    FOR i: CARDINAL IN [0..numberString.length) DO
      IF numberString[i] NOT IN ['0..'9] THEN EXIT;
      ans ← 10*ans + numberString[i] - '0;
      REPEAT FINISHED => IF ans IN [min..max] THEN {
        Heap.systemZone.FREE[@numberString]; RETURN};
      ENDLOOP;
    WriteLine["Bad Number !"L];
    ENDLOOP};

ReadShortNumber: PUBLIC PROC [
  prompt: LONG STRING, min, max, default: LONG CARDINAL]
  RETURNS [CARDINAL] = {
  RETURN[Inline.LowHalf[
    ReadNumber[prompt, min, MIN[max, LONG[LAST[CARDINAL]]], default]]]};

WriteFixedWidthNumber: PUBLIC PROC [
  x: LONG CARDINAL, count: CARDINAL, base: CARDINAL ← 10] = {
  WFD: PROC [x: LONG CARDINAL, c: CARDINAL] = {
    IF c = count THEN RETURN;
    WFD[x/base, c + 1];
    WriteChar[IF c = 0 OR x # 0 THEN Inline.LowHalf[x MOD base] + '0 ELSE ' ]};
  WFD[x, 0]};

WriteLongNumber: PUBLIC PROC [num: LONG CARDINAL] = {
  s: STRING ← [40];
  s.length ← 0;
  String.AppendLongNumber[s, num, 10];
  WriteString[s]};

WriteOctal: PUBLIC PROC [num: CARDINAL] = {
  IF num # 0 THEN WriteOctal[num/8]; WriteChar[(num MOD 8) + '0]};

Yes: PUBLIC PROC [s: LONG STRING] RETURNS [BOOLEAN] = {
  WriteString[s];
  DO
    SELECT ReadChar[] FROM
      'Y, 'y, CR  => {WriteLine["yes"L]; RETURN[TRUE]};
      'N, 'n, DEL => {WriteLine["no"L];  RETURN[FALSE]};
      ENDCASE     => WriteChar['?];
    ENDLOOP};

-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- Time munging
-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

--  string format must be: bDD-MMM-YYbbHH:MM:SSbbZZTb
PackedTimeFromString: PUBLIC PROC [
  s: LONG STRING, justDate: BOOLEAN]
  RETURNS [t: System.GreenwichMeanTime] = {
  Empty: PROC [s: LONG STRING] RETURNS [BOOLEAN] = {
    RETURN[s = NIL OR s.length = 0]};
  EquivalentChar: PUBLIC PROC [c1, c2: CHARACTER] RETURNS [BOOLEAN] = {
    RETURN[String.UpperCase[c1] = String.UpperCase[c2]]};
  GetToken: PROC [storage: LONG STRING, s: LONG STRING, c: CARDINAL]
    RETURNS [is: CARDINAL] = {
    FOR is ← c, is + 1 UNTIL is >= s.length DO
      ch: CHARACTER = s[is];
      SELECT ch FROM
        IN ['a..'z], IN ['A..'Z], IN ['0..'9] =>
          String.AppendChar[storage, ch];
        ':, '- => EXIT; -- terminator
        '      => IF ~Empty[storage] THEN EXIT; --terminating blank
        ENDCASE;
      ENDLOOP;
    RETURN[is + 1]};

  DoIt: PROC [s: LONG STRING] RETURNS [t: System.GreenwichMeanTime] = {
    Get: PROC RETURNS [CARDINAL] = {
      s1.length ← 0; nextChar ← GetToken[s1, s, nextChar];
      RETURN[s1.length]};
    GetNumber: PROC RETURNS [CARDINAL] = {
      [] ← Get[]; RETURN[String.StringToNumber[s1, 10]]};
    m: String.SubStringDescriptor ← [
      base:   "JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC"L,
      offset: NULL, length: 3];
    s1: STRING = [3];
    month: String.SubStringDescriptor ← [
      base: s1, offset: 0, length: NULL];
    time:     Time.Unpacked ← [
      0, 0, 0, 0, 0, 0, 0, FALSE, System.GetLocalTimeParameters[]];
    nextChar: CARDINAL ← 0;
    packIt:   BOOLEAN ← TRUE;
    IF Empty[s] THEN RETURN[System.gmtEpoch];
    time.day ← GetNumber[];
    month.length ← Get[];
```

```
      FOR i: CARDINAL IN [0..12) DO
        m.offset ← i*3;
        IF String.EquivalentSubString[@month, @m] THEN {
          time.month ← i; EXIT};
        ENDLOOP;
      time.year ← GetNumber[];
      time.year ← time.year + (IF time.year>68 THEN 1900 ELSE 2000);
      IF justDate THEN {
        time.hour ← 23; time.minute ← 59; time.second ← 59}
      ELSE {
        time.hour   ← GetNumber[];
        time.minute ← GetNumber[];
        time.second ← GetNumber[];
        IF Get[] # 0 THEN {
          zones: PACKED ARRAY [5..8] OF CHARACTER = ['E, 'C, 'M, 'P];
          FOR i: CARDINAL IN [5..8] DO
            IF EquivalentChar[s1[0], zones[i]] THEN {time.zone.zone ← i; EXIT};
            REPEAT FINISHED => time.zone.zone ← 0; -- GMT
            ENDLOOP;
          time.dst ← EquivalentChar[s1[1], 'D];
          packIt ← FALSE}};
      t ← Time.Pack[time, packIt]};
    t ← DoIt[s
    ! String.InvalidNumber, String.StringBoundsFault, Time.Invalid => {
      t ← System.gmtEpoch; CONTINUE}]};

  WriteTime: PROC [
    t: System.GreenwichMeanTime, showDay: BOOLEAN ← TRUE,
    type: {system, gmt, pacific} ← system] = {
    days: ARRAY [0..7) OF STRING = [
      "Monday"L, "Tuesday"L, "Wednesday"L, "Thursday"L,
      "Friday"L, "Saturday"L, "Sunday"L];
    temps: STRING = [40];
    Time.Append[temps,
      Time.Unpack[t, SELECT type FROM
        pacific => [useThese[[west, 8, 0, 121, 305]]],
        gmt => Time.useGMT,
        ENDCASE => Time.useSystem]];
    IF showDay THEN {
      WriteChar[' ]; WriteString[days[Time.Unpack[t].unpacked.weekday]]};
    IF temps[0] # ' THEN WriteChar[' ];
    WriteLine[temps]};

  -- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- The Big Loop
  -- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  prometheusBound: BOOLEAN =
    Runtime.IsBound[LOOPHOLE[OthelloDefs.GetCannedScript]];

  Run: PUBLIC PROC =
    BEGIN
    ENABLE ABORTED => CONTINUE; -- when deactivated
    ttyHandle ← OthelloToolDefs.tty;
    ResetAbort[];
    PrintHerald[];
    PrintPIDs[];
    PrintMemorySize[];
    GetTime[];
    DO
      TellError: PROC [s: LONG STRING] = {
        IF prometheusBound THEN OthelloDefs.ThereIsAnError[];
        commandIndex ← LAST[CARDINAL]; NewLine[]; WriteString[s]};
      p: PROC [index: CARDINAL]; i: CARDINAL;
      IF (~CommandFileActive[]) AND prometheusBound THEN {
        ResetAbort[]; OthelloDefs.GetCannedScript[]};
      IF CommandFileActive[] THEN
        CheckUserAbort[
        !MyAborted  => {TellError["Command File Aborted\r"L]; LOOP}]
      ELSE ResetAbort[];
      [p, i]  ← CollectCommand[
      ! TryAgain => RETRY;
        AbortingCommand => {TellError[reason]; WriteLine[reasonOne]; LOOP};
        MyAborted => {TellError["Command File Aborted\r"L]; LOOP}];
      NewLine[];
      p[i !
        MyNameIs => RESUME;
        MyAborted => {TellError["ABORTED\r"L]; CONTINUE};
        AbortingCommand => {
          TellError[reason]; WriteLine[reasonOne]; CONTINUE};
        File.Unknown => {
          TellError["File.Unknown"L]; DebugAsk[]; CONTINUE};
        File.Error => {
          PrintNames: PROC [x: File.ErrorType] = {
            e: ARRAY File.ErrorType OF STRING = [
              invalidParameters: "invalidParameters"L,
              reservedType: "reservedType"L];
            WriteString[e[x]]};
          TellError["File.Error["L];
          PrintNames[type];
          WriteChar[']];
          DebugAsk[];
          CONTINUE};
        PhysicalVolume.Error => {
          PrintNames: PROC [x: PhysicalVolume.ErrorType] = {
            e: ARRAY PhysicalVolume.ErrorType OF STRING = [
              badDisk: "badDisk"L,
              badSpotTableFull: "badSpotTableFull"L,
```

```
            containsOpenVolumes: "containsOpenVolumes"L,
            diskReadError: "diskReadError"L,
            hardwareError: "hardwareError"L,
            hasPilotVolume: "hasPilotVolume"L,
            alreadyAsserted: "alreadyAsserted"L,
            insufficientSpace: "insufficientSpace"L,
            invalidHandle: "invalidHandle"L,
            nameRequired: "nameRequired"L,
            needsConversion: "needsConversion"L,
            notReady: "notReady"L,
            noSuchDrive: "noSuchDrive"L,
            noSuchLogicalVolume: "noSuchLogicalVolume"L,
            physicalVolumeUnknown: "physicalVolumeUnknown"L,
            writeProtected: "writeProtected"L,
            wrongFormat: "wrongFormat"L];
          WriteString[e[x]]};
        TellError["PhysicalVolume.Error["L];  PrintNames[error];
        WriteChar[']];
        DebugAsk[];
        CONTINUE};
      PhysicalVolume.NeedsScavenging => {
        TellError["PhysicalVolume.NeedsScavenging"L];
        DebugAsk[]; CONTINUE};
      Scavenger.Error =>{
        PrintNames: PROC [x: Scavenger.ErrorType] = {
          e: ARRAY Scavenger.ErrorType OF STRING = [
            cannotWriteLog: "cannotWriteLog"L,
            noSuchPage: "noSuchPage"L,
            orphanNotFound: "orphanNotFound"L,
            volumeOpen: "volumeOpen"L,
            diskHardwareError: "diskHardwareError"L,
            diskNotReady: "diskNotReady"L,
            needsConversion: "needsConversion"L,
            needsRiskyRepair: "needsRiskyRepair"L];
          WriteString[e[x]]};
        TellError["Scavenger.Error["L]; PrintNames[error]; WriteChar[']];
        DebugAsk[];
        CONTINUE};
      VolumeConversion.Error =>{
        PrintNames: PROC [x: VolumeConversion.ErrorType] = {
          e: ARRAY VolumeConversion.ErrorType OF STRING = [
            hardwareBroken: "hardwareBroken"L,
            lostLog: "lostLog"L,
            runPreviousScavenger: "runPreviousScavenger"L,
            volumeVersionTooNew: "volumeVersionTooNew"L,
            volumeVersionTooOld: "volumeVersionTooOld"L];
          WriteString[e[x]]};
        TellError["VolumeConversion.Error["L]; PrintNames[error]; WriteChar[']];
        DebugAsk[];
        CONTINUE};
      Volume.InsufficientSpace => {
        TellError["Volume.InsufficientSpace"L];
        DebugAsk[]; CONTINUE};
      Volume.NotOpen => {
        TellError["Volume.NotOpen"L]; DebugAsk[]; CONTINUE};
      Volume.NeedsScavenging => {
        TellError["Please Scavenge the volume first"L]; CONTINUE};
      Volume.Unknown => {
        TellError["Volume.Unknown"L]; DebugAsk[]; CONTINUE};
      Volume.ReadOnly => {
        TellError["Volume.ReadOnly"L]; DebugAsk[]; CONTINUE};
      String.StringBoundsFault => {
        TellError["String.StringBoundsFault"L]; DebugAsk[]; CONTINUE};
      TryAgain => CONTINUE;
      ABORTED => REJECT;
      ANY => {
        signal: SIGNAL;
        args:  PrincOps.LocalFrameHandle;
        TellError["Uncaught Signal = ["L];
        [signal: signal, signalArgs: args] ←
           SIGNAL SpecialRuntime.GetCurrentSignal;
        WriteOctal[Inline.LowHalf[LOOPHOLE[signal]]];
        WriteChar[',];
        WriteOctal[Inline.HighHalf[LOOPHOLE[signal]]];
        WriteChar[']];
        IF args # NIL THEN {
          size: CARDINAL ← PrincOps.frameSizeMap[Frame.ReadLocalWord[args].fsi]
            - SIZE[PrincOps.LocalOverhead];
          WriteString[", msg = ["L];
          FOR i: CARDINAL IN [0..size-1) DO
            WriteOctal[args[i]]; WriteString[", "L] ENDLOOP;
          WriteOctal[args[size-1]]; WriteChar[']];
          Frame.Free[args]};
        DebugAsk[]; CONTINUE}];
    ENDLOOP;
  END;

-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- TTY Interface Stuff
-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

useADM: BOOLEAN = FALSE;

ttyHandle: TTY.Handle ← OthelloToolDefs.tty;

BlinkDisplay: PUBLIC PROC = {TTY.BlinkDisplay[ttyHandle]};
```

```
MyAborted: ERROR = CODE;

CheckUserAbort: PUBLIC PROC = {
  IF TTY.UserAbort[ttyHandle] THEN {ResetAbort[]; ERROR MyAborted}};

EraseTTYChar: PROC [c: CHARACTER] = {
  SELECT c FROM IN [' ..'~] => NULL; CR => RETURN; ENDCASE => EraseTTYChar[' ];
  TTY.RemoveCharacter[ttyHandle];};

ReadChar: PUBLIC PROC RETURNS [c: CHARACTER] = {
  gotIt: BOOLEAN;
  [gotIt, c] ← GetCommandFileCharacter[];
  IF gotIt THEN RETURN;
  RETURN[TTY.GetChar[ttyHandle]]};

SetCursor: PUBLIC PROC [c: OthelloDefs.Cursor] = {
  cursor: ARRAY OthelloDefs.Cursor OF UserTerminal.CursorArray = [
    pointer: [
      100000B, 140000B, 160000B, 170000B, 174000B, 176000B, 177000B, 170000B,
      154000B, 114000B, 006000B, 006000B, 003000B, 003000B, 001400B, 001400B],
    ftp: [
      000177B, 076077B, 040037B, 040017B, 070007B, 043703B, 040401B, 040400B,
      000400B, 100436B, 140421B, 160421B, 170036B, 174020B, 176020B, 177020B]];
    IF ~useADM THEN UserTerminal.SetCursorPattern[cursor[c]];
    cursorFlipped ← FALSE};

cursorFlipped: BOOLEAN;

FlipCursor: PUBLIC PROC = {
  IF ~useADM THEN  {
    c: UserTerminal.CursorArray ← UserTerminal.GetCursorPattern[];
    FOR i: CARDINAL IN [0..LENGTH[c]) DO c[i] ← Inline.BITNOT[c[i]] ENDLOOP;
    UserTerminal.SetCursorPattern[c]}
  ELSE {
    IF cursorFlipped THEN WriteChar[BS] ELSE WriteChar[SP];
    cursorFlipped ← ~cursorFlipped}};

FlushInput: PROC = {
  UNTIL TTY.CharsAvailable[ttyHandle] = 0 DO
    [] ← TTY.GetChar[ttyHandle] ENDLOOP};

NewLine: PUBLIC PROC = {WriteChar[CR]};

ResetAbort: PROC = {TTY.ResetUserAbort[ttyHandle]};

WriteChar: PUBLIC PROC [c: CHARACTER] = {
  IF prometheusBound AND OthelloDefs.SuppressOutput[] THEN RETURN;
  TTY.PutChar[ttyHandle, c]};

WriteLine: PUBLIC PROC [s: LONG STRING] = {WriteString[s]; NewLine[]};

WriteString: PUBLIC PROC [s: LONG STRING] = {
  IF prometheusBound AND OthelloDefs.SuppressOutput[] THEN RETURN;
  IF s # NIL THEN TTY.PutString[ttyHandle, s]};

command:      LONG STRING ← NIL;
commandIndex: CARDINAL ← 0;

CommandFileActive: PROC RETURNS [BOOLEAN] = INLINE {RETURN[command#NIL]};

GetCommandFileCharacter: PROC RETURNS [
  isThere: BOOLEAN, c: CHARACTER] = INLINE {
  IF command # NIL THEN {
    IF commandIndex >= command.length THEN  {
      Heap.systemZone.FREE[@command]; command ← NIL}
    ELSE {
     commandIndex ← commandIndex + 1;
     RETURN[TRUE, command[commandIndex-1]]}};
  RETURN[FALSE, 0C]};

SetCommandString: PUBLIC PROC [s: LONG STRING] = {
  IF command # NIL THEN Heap.systemZone.FREE[@command];
  command ← s; commandIndex ← 0};

-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- Initialization Stuff
-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
GetTime: PROC = {
  timeTrys:       CARDINAL ← 3;
  time:           System.GreenwichMeanTime;
  LTPs:           System.LocalTimeParameters;
  timeFromServer: BOOLEAN ← TRUE;
  getTimeString:  LONG STRING ← NIL;
  [time, LTPs] ← OthelloOps.GetTimeFromTimeServer[
    ! OthelloOps.TimeServerError => IF error=noResponse THEN {
        IF (timeTrys ← timeTrys-1)=0 THEN {timeFromServer ← FALSE; CONTINUE}
        ELSE {IF timeTrys=2 THEN WriteString["Locating Time Server..."L]; RETRY}}
      ELSE IF error=noCommunicationFacilities THEN {
        WriteLine["not Communication Facilities to find time"L];
        timeFromServer ← FALSE; CONTINUE}
      ELSE ERROR];
  IF timeFromServer THEN {
    IF timeTrys#3 THEN WriteLine["success"L];
    System.SetLocalTimeParameters[LTPs];
    OthelloOps.SetProcessorTime[time];
    RETURN};
```

```
    WriteLine["failed.\rPlease enter time information (type ? for help)"L];
    getTimeString ← Heap.systemZone.NEW[StringBody[10]];
    LTPs ← GetTimeZoneFromUser[@getTimeString ! TryAgain => RETRY];
    System.SetLocalTimeParameters[LTPs];
    spacesInStringOK ← TRUE;
    GetTimeFromUser[@getTimeString ! TryAgain => RETRY];
    spacesInStringOK ← FALSE;
    Heap.systemZone.FREE[@getTimeString]};

  GetTimeFromUser: PROC [p: LONG POINTER TO LONG STRING] = {
    timePrompt: STRING = "Please Enter the date and 24 hour time in form
      DD-MMM-YY HH:MM:SS
Time: "L;
    IF OthelloOps.IsTimeValid[] THEN {
      WriteString["Current time"L]; WriteTime[System.GetGreenwichMeanTime[]];
      IF ~Yes["Do you wish to change the time?: "L] THEN RETURN};
    IF p#NIL THEN p.length ← 0;
    DO
      time: System.GreenwichMeanTime;
      GetName[timePrompt, p];
      time ← PackedTimeFromString[p↑, FALSE];
      IF time=System.gmtEpoch THEN {
        WriteLine["Invalid date/time -- please try again."L]; LOOP};
      WriteString["Set time to"L]; WriteTime[time];
      IF Yes["Okay?: "L] THEN {
        OthelloOps.SetProcessorTime[time]; EXIT}
      ELSE LOOP
      ENDLOOP};

  GetTimeZoneFromUser: PROC [string: LONG POINTER TO LONG STRING]
    RETURNS [ltp: System.LocalTimeParameters] = {
    GetNum: PROC [
      prompt: STRING, min, max, default: INTEGER]
      RETURNS [ans: INTEGER] = {
      string.length ← 0;
      String.AppendDecimal[string↑, default];
      DO
        isNeg: BOOLEAN ← FALSE;
        WriteString[prompt];
        WriteChar['[]; IF ans<0 THEN WriteChar['-]; WriteLongNumber[ABS[min]];
        WriteString[".."L]; WriteLongNumber[max]; WriteString["]: "L];
        GetName[dest: string, signalQuestion: TRUE];
        ans ← 0;
        FOR i: CARDINAL IN [0..string.length) DO
          IF i=0 AND string[i]='- THEN {isNeg ← TRUE; LOOP};
          IF string[i] NOT IN ['0..'9] THEN EXIT;
          ans ← 10*ans + string[i] - '0;
          REPEAT FINISHED => {
            IF isNeg THEN ans ← -ans; IF ans IN [min..max] THEN RETURN};
          ENDLOOP;
        WriteLine["Bad Number !"L];
        ENDLOOP};
    dstSpiel: STRING = "
The ""First day of DST"" is the day of the year on or before which
Daylight Savings Time takes effect, where:
        1 => January  1
      366 => December 31.
(The correspondence  between numbers and days is based on a leap
year.  Similarly, ""Last day of DST"" is the day of the year on or
before which Daylight Savings Time ends.  Note that in any given
year,  Daylight Savings Time actually begins and ends at 2 AM on
the last Sunday not  following  the  specified date.  The system
makes this adjustment  for you automatically.  The normal values
are
      121   (April 30) for the first day of DST
      305 (October 31) for the last  day of DST.
If Daylight  Savings  Time is not  observed locally, both values
should be set to zero."L;
    ZoneSpiel: STRING = "
Number  of  hours  between  Greenwich  and local time.  For time
zones west of  Greenwich, the offset is negative; for time zones
east of Greenwich, the offset is positive. Examples:
      San Francisco  -8 hours    (Pacific  time zone)
      Denver         -7 hours    (Mountain time zone)
      Chicago        -6 hours    (Central  time zone)
      Boston         -5 hours    (Eastern  time zone)"L;
    n: INTEGER;

    n ← GetNum[prompt: "Time zone offset from Greenwich "L, min: -12, max: 12, default: -8
    ! Question => {WriteLine[ZoneSpiel]; RETRY}];
    ltp.direction ← IF n<0 THEN west ELSE east; ltp.zone ← ABS[n];
    ltp.zoneMinutes ← GetNum[prompt: "Minute offset "L, min: 0, max: 59, default: 0
    ! Question => {WriteLine["\rAlmost always zero"L]; RETRY}];
    ltp.beginDST ← GetNum[prompt: "First day of DST "L, min: 0, max: 366, default: 121
    ! Question => {WriteLine[dstSpiel]; RETRY}];
    ltp.endDST ← GetNum[prompt: "Last day of DST "L,  min: 0, max: 366, default: 305
    ! Question => {WriteLine[dstSpiel]; RETRY}]};

  PrintHerald: PROC = {
    copyright: STRING = [100];
    version: STRING = [10];
    IF useADM THEN WriteChar['\032]; -- clear screen
    VersionExtras.AppendCopyright[copyright];
    Version.Append[version];
    WriteString[copyright];  WriteString["\n\n"L];
    WriteString["Othello "L]; WriteString[version]; WriteString[" of "L];
    WriteTime[Runtime.GetBcdTime[], FALSE, pacific]};
```

```
  PrintPIDs: PROC = {
    w: Format.StringProc = {WriteString[s]};
    WriteString["Processor = "L];
    Format.HostNumber[proc: w,
      hostNumber: LOOPHOLE[SpecialSystem.GetProcessorID[]], format: hex];
    WriteString[" = "L];
    Format.HostNumber[proc: w,
      hostNumber: LOOPHOLE[SpecialSystem.GetProcessorID[]], format: octal];
    WriteString["B = "L];
    Format.HostNumber[proc: w, hostNumber:
      LOOPHOLE[SpecialSystem.GetProcessorID[]], format: productSoftware];
    NewLine[];
    };

  PrintMemorySize: PROC = {
  size: LONG CARDINAL ← ((SpecialSpace.realMemorySize+255)/256)*64;
    WriteString["Memory size = "L];
    WriteLongNumber[size*2];
    WriteString["K bytes"L];
    NewLine[];
    };

  << The following move to proc. Run
  SetCursor[pointer];
  PrintHerald[];
  PrintPIDs[];
  PrintMemorySize[];
  GetTime[];>>

  END..
```

```
LOG
Time:  1-Oct-81 18:44:29  By: FXH      Action: Re-do module,
                                        add Time Stuff & Proc ID
Time:  13-Nov-81 16:27:44  By: FXH     Action: 8.0e build
Time:  19-Nov-81  9:26:07  By: FXH     Make PackedTimeFromString public for
                                        implementing SetBootFileExpirationDate
Time:  17-Dec-81 17:52:16  By: CRF     Action: 8.0f build -- changed herald.
Time:  29-Dec-81 14:29:14  By: CRF     Action: 8.0g build -- changed herald.
Time:  29-Dec-81 14:29:14  By: FXH     Action: 8.0h build -- changed herald.
Time:   1-Feb-82 16:11:37  By: CRF     Action: 8.0i build -- changed herald.
Time:   3-Feb-82 15:02:19  By: CAJ     Action: Print processor ID all 3
                                        ways using Format.
Time:   8-Feb-82 17:20:50  By: CRF     Action: 8.0j build -- changed herald.
Time:   1-Mar-82 13:55:31  By: CAJ     Action: final 8.0 build -- changed herald.
Time:   20-Aug-82 16:49:26 By: AEF     Action: Change to 9.0b.
Time:   16-Sep-82 11:47:54 By: AEF     Action: Change to 9.0c.
Time:   24-Sep-82 17:24:53 By: AEF     Action: Change to 9.0d.
Time:   30-Sep-82 13:48:48 By: AEF     Action: Change to 9.0.
Time:   12-Dec-82 12:50:23 By: RXJ     Action: 10.0c; remove Storage.
Time:    4-Jun-86 12:03:59 By: NFS      Adapted for OthelloTool
```

```
-- Copyright (C) 1983  by Xerox Corporation. All rights reserved.
-- VolumeInitImplA.mesa edited by:
--   RXJ          2-Dec-83 18:21:20
--   RES         17-Oct-83 14:53:35
--   NFS          4-Jun-86 14:08:44

DIRECTORY
  Device USING [PilotDisk, Type],
  DeviceTypes USING [
    q2000, q2010, q2020, q2030, q2040, q2080, sa1000, sa1004, sa4000,
    sa4008, t300, t80],
  Environment USING [wordsPerPage],
  File USING [
    Delete, File, GetAttributes, ID, nullFile, PageNumber, Type, Unknown],
  Heap USING [systemZone],
  Inline USING [BITROTATE],
  OthelloDefs USING [
    AbortingCommand, CloseFetch, CommandProcessor, Confirm, GetName,
    IndexTooLarge, LeaderPage, leaderPages, lpVersion, MyNameIs, NewLine,
    PackedTimeFromString, Question, ReadNumber, RegisterCommandProc,
    SetCommandString, WriteChar, WriteFixedWidthNumber, WriteLine,
    WriteLongNumber, WriteOctal, WriteString, Yes],
  OthelloOps USING [
    BadSwitches, BootFileType, DecodeSwitches, DeleteTempFiles, GetDriveSize,
    GetNextSubVolume, GetPhysicalVolumeBootFile, GetSwitches, GetVolumeBootFile,
    nullSubVolume, SetDebugger, SetDebuggerSuccess, SetExpirationDate,
    SetExpirationDateSuccess, SetGetSwitchesSuccess, SetPhysicalVolumeBootFile,
    SetSwitches, SubVolume, VoidPhysicalVolumeBootFile, VoidVolumeBootFile],
  OthelloToolDefs USING [CloseVolume],
  PhysicalVolume USING [
    AssertPilotVolume, DamageStatus, Error, GetAttributes, GetHandle, GetNext,
    GetNextBadPage, GetNextDrive, GetNextLogicalVolume, Handle, ID,
    InterpretHandle, MarkPageBad, maxNameLength, noProblems, nullBadPage,
    nullDeviceIndex, nullID, Offline, PageNumber, RepairType, Scavenge,
    ScavengerStatus],
  Process USING [MsecToTicks],
  Runtime USING [IsBound],
  Scavenger USING [
    BootFileType, Error, FileEntry, Header, Problem, RepairType, Scavenge],
  Space USING [CopyIn, Map, ScratchMap, Unmap],
  SpecialVolume USING [OpenVolume],
  String USING [
    AppendCharAndGrow, AppendLongNumber, AppendString, CopyToNewString, Equivalent,
    Length, Replace],
  System USING [
    defaultSwitches, GetLocalTimeParameters, gmtEpoch, GreenwichMeanTime,
    PowerOff, Switches],
  TemporaryBooting USING [BootButton, BootFromVolume],
  Volume USING [
    Erase, GetAttributes, GetLabelString, GetType, ID,
    NeedsScavenging, NotOnline, nullID, Open, systemID, Type],
  VolumeVersion USING [Examine];

VolumeInitImplA: PROGRAM
  IMPORTS
    File, Heap, Inline, OthelloDefs, OthelloOps, OthelloToolDefs, PhysicalVolume, Process, Runtime,
    Scavenger, Space, SpecialVolume, System, String, TemporaryBooting, Volume,
    VolumeVersion
  EXPORTS OthelloDefs
  SHARES File =
  BEGIN OPEN OthelloOps, OthelloDefs;

  commandProcessor: CommandProcessor ← [CommonCommands];

  CommonCommands: PROC [index: CARDINAL] = {
    SELECT index FROM
      0 => BootBoot[];
      1 => DeleteBootFiles[];
      2 => DeleteTempFilesUser[];
      3 => DescribePhysicalVolumes[];
      4 => Erase[];
      5 => ListBadPages[];
      6 => ListBootFiles[];
      7 => ListDrives[];
      8 => ListLogicalVolumes[];
      9 => ListPhysicalVolumes[];
     10 => MakeBad[];
     11 => Offline[];
     12 => Online[];
     13 => PowerOff[];
     14 => PVScavenge[];
     15 => Quit[];
     16 => Scavenge[];
     17 => SetBootFileSwitches[];
     18 => SetDebuggerUser[];
     19 => SetExpirationDateUser[];
     20 => SetPvBoot[];
     21 => WizardMode[];
    ENDCASE => IndexTooLarge};

  logicalVolumeTypeString: ARRAY Volume.Type OF LONG STRING ← [
    "normal", "debugger", "debuggerDebugger", "nonPilot"];

  inputDriveString:      LONG STRING ← NIL;
  inputLogicalString:    LONG STRING ← NIL;
  debuggerLogicalString: LONG STRING ← NIL;
  inputPhysString:       LONG STRING ← NIL;
```

```
  switches:              LONG STRING ← NIL;
  lvTypeString:          LONG STRING ← NIL;
  expirationString:      LONG STRING ← NIL;

  maxNameLength: CARDINAL = PhysicalVolume.maxNameLength;

  BootBoot: PROC =
    BEGIN
    lvID: Volume.ID;
    ts:   System.Switches;

    MyNameIs[
      myNameIs: "Boot"L, myHelpIs: "Boot From Logical Volume"L];
    lvID ← GetLvIDFromUser[].lvID;
    GetSetBootFileSwitches[get, lvID
    ! Volume.NeedsScavenging, File.Unknown => {
        WriteLine["(can't get default switches)"L];
        CONTINUE};
      AbortingCommand => {
        WriteLine[reason];
        WriteLine["(can't get default switches)"L];
        CONTINUE}];
    DO
      GetName["switches: "L, @switches, echo, TRUE
      ! Question => {
        WriteLine[
          "See Pilot Users Handbook for list of valid switches."L];
        RESUME}];
      ts ← DecodeSwitches[switches
      ! BadSwitches => {WriteLine["bad switches"L]; LOOP}];
      EXIT;
      ENDLOOP;
    IF Runtime.IsBound[LOOPHOLE[CloseFetch]] THEN CloseFetch[];
    TemporaryBooting.BootFromVolume[lvID, ts];
    END;

  DeleteBootFiles: PROC =
    BEGIN
    lvID: Volume.ID;
    pvID: PhysicalVolume.ID;
    MyNameIs[
      myNameIs: "Delete Boot Files"L,
      myHelpIs: "Delete all boot files from volume"L];
    [pvID: pvID, lvID: lvID] ← GetLvIDFromUser[];
    IF lvID = Volume.systemID THEN {
      WriteLine["Can not delete boot file of current system volume."L];
      RETURN;};
    FOR t: BootFileType IN [hardMicrocode..pilot] DO
      file: File.File = GetVolumeBootFile[lvID, t].file;
      IF file = File.nullFile THEN LOOP;
      Volume.Open[file.volumeID];
      BEGIN ENABLE File.Unknown => CONTINUE;
      File.Delete[file];
      END;
      VoidVolumeBootFile[lvID, t];
      IF GetPhysicalVolumeBootFile[pvID, t].file = file THEN
        VoidPhysicalVolumeBootFile[pvID, t];
      OthelloToolDefs.CloseVolume[lvID];
      ENDLOOP;
    END;

  DeleteTempFilesUser: PROC = {
    lv: Volume.ID;
    MyNameIs[
      myNameIs: "Delete Temporary Files"L,
      myHelpIs: "Delete Temporary Files"L];
    lv ← GetLvIDFromUser[].lvID;
    IF lv = Volume.systemID THEN {
      WriteLine["Can not delete temp files on current system volume."L];
      RETURN;};
    DeleteTempFiles[lv]};

  DescribePhysicalVolumes: PROC =
    BEGIN
    pvID:    PhysicalVolume.ID ← PhysicalVolume.nullID;
    pvFound: BOOLEAN           ← FALSE;

    MyNameIs[
      myNameIs: "Describe Physical Volumes"L,
      myHelpIs: "Describe online physical volumes"L];
    DO
      h:       PhysicalVolume.Handle;
      s:       STRING    ← [maxNameLength];
      sV:      SubVolume ← nullSubVolume;
      sVFound: BOOLEAN   ← FALSE;
      IF (pvID ← PhysicalVolume.GetNext[pvID]) = PhysicalVolume.nullID THEN EXIT;
      pvFound ← TRUE;
      h ← PhysicalVolume.GetAttributes[pvID, s].instance;
      WriteString["Physical Volume "L];
      WriteString[s]; WriteString[" on drive "L];
      WriteString[GetDriveStringName[h]];
      WriteString[" ("L];
      WriteString[
        SELECT GetDriveType[h] FROM
          DeviceTypes.sa1004, DeviceTypes.sa1000  => "Shugart 1000"L,
          DeviceTypes.sa4000, DeviceTypes.sa4008  => "Shugart 4000"L,
          DeviceTypes.q2000,  DeviceTypes.q2010, DeviceTypes.q2020,
```

```
            DeviceTypes.q2030, DeviceTypes.q2040, DeviceTypes.q2080    => "Quantum 2000"L,
            DeviceTypes.t80  =>   "T80"L,
            DeviceTypes.t300 =>   "T300"L,
            ENDCASE =>            "unknown type"L];
        DO
          needsScavenging: BOOLEAN ← FALSE;
          freePages, volumeSize: LONG CARDINAL;
          sV ← GetNextSubVolume[pvID, sV];
          IF sV = nullSubVolume THEN EXIT;
          IF ~sVFound THEN WriteLine[") contains:"L];
          sVFound ← TRUE;
          WriteString["Volume "L];
          [volumeSize: volumeSize, freePageCount: freePages] ← Volume.GetAttributes[sV.lvID
            ! Volume.NeedsScavenging => {
              needsScavenging ← TRUE;
              volumeSize ← 0; -- don't really know
              CONTINUE }];
          IF volumeSize # sV.subVolumeSize AND volumeSize # 0 THEN
            WriteString["piece "L];
          GetLogicalVolumeName[sV.lvID, s];
          WriteString[s]; WriteString[" (type =  "L];
          WriteString[GetLogicalVolumeTypeName[sV.lvID]];  WriteString[") "L];
          IF volumeSize = sV.subVolumeSize THEN {
            WriteLongNumber[freePages]; WriteString[" of "L];
            WriteLongNumber[volumeSize]; WriteString[" pages free"L]}
          ELSE {WriteLongNumber[sV.subVolumeSize]; WriteString[" pages"L]};
          IF needsScavenging THEN WriteString["    *** Needs Scavenging ***"L];
          NewLine[];
          WriteString["   starting at physical address "L];
          WriteLongNumber[sV.firstPVPageNumber];
          NewLine[];
          IF ~needsScavenging THEN ShowBootFiles[pvID, sV.lvID];
          ENDLOOP;
        IF ~sVFound THEN WriteLine[") no subvolumes"L];
        ENDLOOP;
      IF ~pvFound THEN WriteLine["No physical Volumes found"L];
      END;

    Erase: PROC = {
      lvID: Volume.ID;
      pvID: PhysicalVolume.ID;
      MyNameIs[
        myNameIs: "Erase"L, myHelpIs: "Erase Logical Volume"L];
      [pvID: pvID, lvID: lvID] ← GetLvIDFromUser[];
      IF lvID = Volume.systemID THEN {
        WriteLine["Can not erase current system volume."L];
        RETURN;};
      Confirm[];
      OthelloToolDefs.CloseVolume[lvID];
      SELECT VolumeVersion.Examine[lvID] FROM
        otherVersion =>
          IF Yes["That volume is not in the current format. Do you want to convert it? "L]
            THEN Confirm[twice]
          ELSE RETURN;
        ENDCASE;
      WriteString["Erasing..."L];
      Volume.Erase[lvID];
      FOR t: BootFileType IN [hardMicrocode..pilot] DO
        IF GetPhysicalVolumeBootFile[pvID, t].file.volumeID = lvID THEN
          VoidPhysicalVolumeBootFile[pvID, t];
        ENDLOOP;
      WriteLine["complete"L]};

    ListBadPages: PROC =
      BEGIN
      id: PhysicalVolume.ID;
      page: PhysicalVolume.PageNumber ← PhysicalVolume.nullBadPage;
      badSpots: BOOLEAN ← FALSE;
      col: CARDINAL ← 0;
      MyNameIs[
        myNameIs: "List Bad Pages"L,
        myHelpIs: "List known bad pages on Pilot volume"L];
      id ← GetPvIDFromUser[].id;
      WHILE (page ← PhysicalVolume.GetNextBadPage[id, page]) #
        PhysicalVolume.nullBadPage DO
        IF col = 6 THEN BEGIN NewLine[]; col ← 0; END;
        WriteFixedWidthNumber[page, 11];
        col ← col + 1; badSpots ← TRUE;
        ENDLOOP;
      WriteLine[IF badSpots THEN NIL ELSE "No known bad spots"L];
      END;

    ListBootFiles: PROC =
      BEGIN
      lvID: Volume.ID;
      pvID: PhysicalVolume.ID;
      MyNameIs[
        myNameIs: "List Boot Files"L,
        myHelpIs: "List boot files on Pilot volume"L];
      [pvID: pvID, lvID: lvID] ← GetLvIDFromUser[];
      ShowBootFiles[pvID, lvID];
      END;

    ListDrives: PROC = {
      index: CARDINAL ← PhysicalVolume.nullDeviceIndex;
      first: BOOLEAN  ← TRUE;
      MyNameIs[myNameIs: "List Drives"L, myHelpIs: "List Drives"L];
```

```
    DO
      index ← PhysicalVolume.GetNextDrive[index];
      IF index = PhysicalVolume.nullDeviceIndex THEN EXIT;
      IF ~first THEN WriteString[", "L];
      first ← FALSE;
      WriteString[GetDriveStringName[PhysicalVolume.GetHandle[index]]];
      ENDLOOP;
    NewLine[]};

  ListLogicalVolumes: PROC =
    BEGIN
    first: BOOLEAN ← TRUE;
    pID: PhysicalVolume.ID ← PhysicalVolume.nullID;
    MyNameIs[
      myNameIs: "List Logical Volumes"L, myHelpIs: "List Logical Volumes"L];
    DO
      sV: SubVolume ← nullSubVolume;
      pID ← PhysicalVolume.GetNext[pID];
      IF pID = PhysicalVolume.nullID THEN EXIT;
      DO
        s: STRING ← [maxNameLength];
        sV ← GetNextSubVolume[pID, sV];
        IF sV = nullSubVolume THEN EXIT;
        IF sV.firstLVPageNumber # 0 THEN LOOP;
        IF ~first THEN WriteString[", "L];
        WriteString[GetDriveStringName[
          PhysicalVolume.GetAttributes[pID].instance]];
        WriteChar[':];
        GetLogicalVolumeName[sV.lvID, s];
        WriteString[s];
        first ← FALSE;
        ENDLOOP;
      ENDLOOP;
    WriteLine[IF first THEN "No logical volumes found"L ELSE NIL];
    END;

  ListPhysicalVolumes: PROC =
    BEGIN
    s:           STRING = [maxNameLength];
    driveString: LONG STRING;
    first:       BOOLEAN          ← TRUE;
    pID:         PhysicalVolume.ID ← PhysicalVolume.nullID;
    MyNameIs[
      myNameIs: "List Physical Volumes"L,
      myHelpIs: "List Physical Volumes"L];
    DO
      pID ← PhysicalVolume.GetNext[pID];
      IF pID = PhysicalVolume.nullID THEN EXIT;
      driveString ← GetDriveStringName[
        PhysicalVolume.GetAttributes[pID, s].instance];
      IF ~first THEN WriteString[", "L];
      WriteString[driveString];
      WriteChar[':];
      WriteString[s];
      first ← FALSE;
      ENDLOOP;
    WriteLine[IF ~first THEN NIL ELSE "No physical volumes found"L];
    END;

  MakeBad: PROC =
    BEGIN
    h:    PhysicalVolume.Handle;
    id:   PhysicalVolume.ID;
    page: PhysicalVolume.PageNumber;
    IF ~Wizard[] THEN RETURN;
    MyNameIs[
      myNameIs: "Make Page Bad"L,
      myHelpIs: "Enter page into bad page table"L];
    [id, h] ← GetPvIDFromUser[];
    page ← ReadNumber["Decimal Page Number: "L, 0, GetDriveSize[h] - 1];
    PhysicalVolume.MarkPageBad[id, page];
    WriteLine["Consider scavenging some logical volumes."L];
    END;

  Offline: PROC = {
    MyNameIs[
      myNameIs: "Offline"L, myHelpIs: "Bring physical volume offline"L];
    PhysicalVolume.Offline[GetPvIDFromUser[].id]};

  Online: PROC = {
    pvID: PhysicalVolume.ID;
    MyNameIs[
      myNameIs: "Online"L, myHelpIs: "Bring drive online"L];
    pvID ← PhysicalVolume.AssertPilotVolume[GetDriveFromUser[] !
      PhysicalVolume.Error => IF error = alreadyAsserted THEN CONTINUE];
    -- (maybe) update time parameters on disk
    [] ← System.GetLocalTimeParameters[pvID]};

  PowerOff: PROC = {
    MyNameIs[
      myNameIs: "Power Off", myHelpIs: "Execute System.PowerOff"L];
    Confirm[]; CloseFetch[]; System.PowerOff[]};

  PVScavenge: PROC =
    BEGIN OPEN PV: PhysicalVolume;
    convert: BOOLEAN ← FALSE;
    s:       PV.ScavengerStatus;
```

```
    h:      PV.Handle;
    repair: PV.RepairType;
    p:      PV.ID ← PV.nullID;
    PrintDamageStatus: PROC [s: STRING, d: PV.DamageStatus] = {
      IF d=okay THEN RETURN;
      WriteString[s];
      WriteLine[IF d=damaged THEN " damaged"L ELSE " lost"L]};

    MyNameIs[
      myNameIs: "Physical Volume Scavenge"L,
      myHelpIs: "Scavenge physical volume"L];
    h ← GetDriveFromUser[];
    repair ←
      IF ~Yes["Repair? "L] THEN checkOnly
      ELSE IF Wizard[] AND Yes["Risky repair? "L] THEN riskyRepair ELSE safeRepair;
    Confirm[];
    DO
      IF (p←PhysicalVolume.GetNext[p]) = PhysicalVolume.nullID THEN EXIT;
      IF h = PhysicalVolume.GetAttributes[p].instance THEN {
        PhysicalVolume.Offline[p]; EXIT};
      ENDLOOP;
    BEGIN ENABLE PhysicalVolume.Error =>
      IF error = needsConversion AND ~convert THEN
        IF (convert ← Yes["That volume is not in the current format. Do you want to convert it? "L])
          THEN {Confirm[]; RETRY}
        ELSE AbortingCommand["Volume cannot be scavenged"L];
      WriteString["Scavenging..."L];
      s ← PV.Scavenge[h, repair, convert];
      WriteLine["Complete"L];
      END; -- ENABLE
    IF s = PV.noProblems THEN {WriteLine["No problems detected"L]; RETURN};
    WriteString["Damage detected:  "L];
    IF s.internalStructures # okay THEN {
      WriteString["Internal structures "L];
      WriteLine[
        IF s.internalStructures=damaged THEN
          IF repair=safeRepair THEN "damaged -- contact hardware support for risky repair"L
          ELSE "damaged"L
        ELSE "repaired"L]};
    PrintDamageStatus["Bad page list"L,        s.badPageList];
    PrintDamageStatus["Boot file"L,            s.bootFile];
    PrintDamageStatus["Germ"L,                 s.germ];
    PrintDamageStatus["Pilot microcode"L,      s.softMicrocode];
    PrintDamageStatus["Diagnostic microcode"L, s.hardMicrocode];
    END;

  Quit: PROC = {
    MyNameIs[
      myNameIs: "Quit"L, myHelpIs: "Push the boot button"L];
    Confirm[]; CloseFetch[]; TemporaryBooting.BootButton[]};

  Scavenge: PROC = {
    convert: BOOLEAN ← FALSE;
    lvID:    Volume.ID;
    logFile: File.File;
    logPage: File.PageNumber ← 0;
    logWd:   CARDINAL ← Environment.wordsPerPage;
    buffer: LONG POINTER TO ARRAY [0..Environment.wordsPerPage) OF UNSPECIFIED ← NIL;

    GetWds: PROC [p: POINTER, c: CARDINAL] = {
      WHILE c#0 DO
        IF logWd=Environment.wordsPerPage THEN {
          [] ← Space.CopyIn[buffer, [logFile, logPage, 1]];
          logPage ← logPage + 1; logWd ← 0};
        p↑ ← buffer[logWd];
        p ← p+1; c ← c-1; logWd ← logWd+1;
        ENDLOOP};
    DisplayScavLog: PROC = {
      fileCount: LONG CARDINAL; problems: BOOLEAN ← FALSE;
      BEGIN
      hd: Scavenger.Header;
      GetWds[@hd, SIZE[Scavenger.Header]];
      WriteString["volume"L]; IF ~hd.repaired THEN WriteString[" not"L];
      WriteString[" repaired, log file"L];
      IF hd.incomplete THEN WriteString[" not"L]; WriteLine[" complete "L];
      WriteLongNumber[fileCount ← hd.numberOfFiles];
      WriteLine[" files on volume"L];
      END;
      WHILE fileCount#0 DO
        OpenID: TYPE = ARRAY [0..SIZE[File.ID]) OF CARDINAL;
        fe:     Scavenger.FileEntry;
        GetWds[@fe, SIZE[Scavenger.FileEntry]];
        THROUGH [0..fe.numberOfProblems) DO
          fp: Scavenger.Problem;
          GetWds[@fp, SIZE[Scavenger.Problem]];
          WriteChar['[];
          FOR i: CARDINAL IN [0..SIZE[File.ID]-1) DO
            WriteOctal[LOOPHOLE[fe.file, OpenID][i]]; WriteString[", "L] ENDLOOP;
          WriteOctal[LOOPHOLE[fe.file, OpenID][SIZE[File.ID]-1]];
          WriteString["] type = "L];
          BEGIN ENABLE File.Unknown => GOTO noType;
          f: File.Type = File.GetAttributes[[fe.file, lvID]].type;
          WriteLongNumber[LONG[LOOPHOLE[f, CARDINAL]]];
          EXITS noType => WriteString["unknown"L];
          END;
          WriteString["; "L];
          WITH fp SELECT FROM
```

```
            unreadable => {
              WriteString["unreadable"L];
              WriteString[" pages ["L]; WriteLongNumber[first];
              WriteString[".."L]; WriteLongNumber[first+count]; WriteLine[")"L]};
            missing => {
              WriteString["missing"L];
              WriteString[" pages ["L]; WriteLongNumber[first];
              WriteString[".."L]; WriteLongNumber[first+count]; WriteLine[")"L]};
            duplicate => {
              WriteString["duplicate"L]; WriteLine[" page found"L]};
            orphan => {
              WriteString["orphan"L]; WriteLine[" page found"L]};
            ENDCASE => WriteLine["unknown problem"L];
          problems ← TRUE;
          ENDLOOP;
        fileCount ← fileCount-1;
        ENDLOOP;
      WriteLine[IF ~problems THEN "No problems found"L ELSE NIL]};
    MyNameIs[
      myNameIs: "Scavenge"L, myHelpIs: "Scavenge Logical Volume"L];
    lvID ← GetLvIDFromUser[].lvID;
    IF lvID = Volume.systemID THEN {
      WriteLine["Can not scavenge current system volume."L];
      RETURN;};
    Confirm[];
    OthelloToolDefs.CloseVolume[lvID ! ANY => CONTINUE];
    BEGIN ENABLE Scavenger.Error =>
      IF error = needsConversion AND ~convert THEN
        IF (convert ← Yes["That volume is not in the current format. Do you want to convert it? "L])
          THEN {Confirm[twice]; RETRY}
        ELSE AbortingCommand["Volume cannot be scavenged"L];
      WriteString["Scavenging..."L];
      logFile ← Scavenger.Scavenge[lvID, lvID, safeRepair, convert];
      WriteLine["Complete"L];
      END; -- ENABLE
    SpecialVolume.OpenVolume[lvID, read];
    buffer ← Space.ScratchMap[1];
    DisplayScavLog[
      ! UNWIND => {[] ← Space.Unmap[buffer]; OthelloToolDefs.CloseVolume[lvID]}];
    [] ← Space.Unmap[buffer]; OthelloToolDefs.CloseVolume[lvID]};

  SetBootFileSwitches: PROC =
    BEGIN
    ts:   System.Switches;
    lvID: Volume.ID;

    MyNameIs[
      myNameIs: "Set Boot File Default Switches"L,
      myHelpIs: "Set default switches for boot file on volume"L];
    lvID ← GetLvIDFromUser[].lvID;
    GetSetBootFileSwitches[get, lvID];  -- volume.needsScav (caught higher up)
    DO
      GetName["switches: "L, @switches, echo, TRUE
      ! Question => {WriteLine[
        "See Pilot Users Handbook for list of valid switches."L];
        RESUME}];
      ts ← DecodeSwitches[switches
      ! BadSwitches => {WriteLine["bad switches"L]; LOOP}];
      EXIT;
      ENDLOOP;
    Confirm[];
    GetSetBootFileSwitches[set, lvID, ts];
    END;

  SetDebuggerUser: PROC =
    BEGIN
    file:       File.File;
    firstPage: File.PageNumber;
    lvID:      Volume.ID;
    dLvID:     Volume.ID;
    dH:        PhysicalVolume.Handle;
    dT:        Device.Type;
    dO:        CARDINAL;
    outcome:   SetDebuggerSuccess;

    MyNameIs[
      myNameIs: "Set Debugger Pointers"L,
      myHelpIs: "Set up pointers to debugger for volume"L];
    lvID ← GetLvIDFromUser["for debuggee Logical Volume: "L].lvID;
    [file, firstPage] ← GetVolumeBootFile[lvID, pilot];
    IF file = File.nullFile THEN
      AbortingCommand["No boot file found."L];
    [, dLvID, dH] ← GetLvIDFromUser["for debugger Logical Volume: "L, TRUE];
    IF dLvID=Volume.nullID THEN WriteLine["(Clear existing pointers)"L]
    ELSE {dT ← GetDriveType[dH]; dO ← GetDriveNumber[dH]};
    Confirm[];
    Volume.Open[lvID];
    outcome ← SetDebugger[
      debuggeeFile: file, debuggeeFirstPage: firstPage, debugger: dLvID,
      debuggerType: dT, debuggerOrdinal: dO];
    OthelloToolDefs.CloseVolume[lvID];
    WriteSetDebuggerSuccess[outcome];
    END;

  SetExpirationDateUser: PROC =
    BEGIN
    file:       File.File;
```

```
    firstPage: File.PageNumber;
    time:      System.GreenwichMeanTime;
    lvID:      Volume.ID;
    outcome:   SetExpirationDateSuccess;

    MyNameIs[
      myNameIs: "Set Hardware Clock Upper Limit"L,
      myHelpIs: "Set last believable hardware clock date for boot file on logical volume"L];
    lvID ← GetLvIDFromUser[].lvID;
    [file, firstPage] ← GetVolumeBootFile[lvID, pilot];
    IF file = File.nullFile THEN
      AbortingCommand["No boot file found."L];
    DO
      GetName["Date (DD-MMM-YY): "L, @expirationString];
      IF expirationString.length=0 THEN {
        WriteLine["(setting no upper limit on hardware clock)"L];
        time ← System.gmtEpoch;
        EXIT}
      ELSE {
        time ← PackedTimeFromString[s: expirationString, justDate: TRUE];
        IF time=System.gmtEpoch THEN WriteLine["invalid date"L]
        ELSE EXIT};
      ENDLOOP;
    Confirm[];
    Volume.Open[lvID];
    outcome ← SetExpirationDate[file, firstPage, time];
    OthelloToolDefs.CloseVolume[lvID];
    WriteSetDebuggerSuccess[outcome];
    END;

  SetPvBoot: PROC =
    BEGIN
    lvID: Volume.ID;
    set: ARRAY BootFileType[hardMicrocode..pilot] OF BOOLEAN ← ALL[FALSE];
    found, changed: BOOLEAN ← FALSE;
    Smash: PROC [s: STRING, t: BootFileType] = {
      IF GetVolumeBootFile[lvID, t].file = File.nullFile THEN RETURN;
      found ← TRUE;
      WriteString["Set physical volume "L];
      WriteString[s];
      IF (set[t] ← Yes[" from this logical volume? "L]) THEN changed ← TRUE};

    MyNameIs[
      myNameIs: "Set Physical Volume Boot Files"L,
      myHelpIs: "Set Physical Volume Boot Files"L];
    lvID ← GetLvIDFromUser[].lvID;
    Smash["boot file"L, pilot];
    Smash["pilot microcode"L, softMicrocode];
    Smash["germ file"L, germ];
    Smash["diagnostic microcode"L, hardMicrocode];
    IF ~found THEN AbortingCommand["Logical volume has null boot files"L];
    IF ~changed THEN RETURN;
    Confirm[];
    SpecialVolume.OpenVolume[lvID, read];
    FOR t: BootFileType IN [hardMicrocode..pilot] DO
      IF set[t] THEN {
        file: File.File;
        firstPage: File.PageNumber;
        [file, firstPage] ← GetVolumeBootFile[lvID, t];
        SetPhysicalVolumeBootFile[file, t, firstPage]};
      ENDLOOP;
    OthelloToolDefs.CloseVolume[lvID];
    END;


  ShowBootFiles: PROC [pv: PhysicalVolume.ID, lv: Volume.ID] = {
    bootNames: ARRAY BootFileType[hardMicrocode..pilot] OF STRING ← [
      hardMicrocode: "Diagnostic microcode"L,
      softMicrocode: "Pilot microcode"L,
      germ: "Germ"L,
      pilot: "Pilot bootfile"L];
    SpecialVolume.OpenVolume[lv, read ! Volume.NeedsScavenging => GOTO scavenge];
    FOR t: BootFileType IN BootFileType[hardMicrocode..pilot] DO
      ENABLE UNWIND => OthelloToolDefs.CloseVolume[lv];
      file: File.File;
      firstPage: File.PageNumber;
      [file: file, firstPage: firstPage] ← GetVolumeBootFile[lv, t];
      IF file = File.nullFile THEN LOOP;
      WriteString["   "L];
      IF GetPhysicalVolumeBootFile[pv, t].file = file THEN
        WriteString["(PV) "L];
      WriteString[bootNames[t]];
      WriteString[": "L];
      IF firstPage = OthelloDefs.leaderPages THEN ShowLeaderNote[file]
      ELSE WriteLine["(no information available)"L];
      ENDLOOP;
    OthelloToolDefs.CloseVolume[lv];
    EXITS
    scavenge => NULL};

  ShowLeaderNote: PROC [file: File.File] = {
    lp: LONG POINTER TO OthelloDefs.LeaderPage;
    lp ← Space.Map[window:[file, 0, OthelloDefs.leaderPages], access: readOnly].pointer;
    IF lp.version = OthelloDefs.lpVersion THEN
      FOR i: CARDINAL IN [0..lp.length) DO WriteChar[lp.note[i]] ENDLOOP
    ELSE WriteString["(no information available)"L];
    NewLine[];
```

```
    [] ← Space.Unmap[lp]};

  WizardMode: PROC = {
    password: LONG STRING ← NIL;
    IF wizardMode THEN RETURN;
    MyNameIs[
      myNameIs: "Wizard Mode"L, myHelpIs: "Enable special commands"L];
    GetName["Password: "L, @password, stars];
    IF Hash[password] = wizardPassword THEN {wizardMode ← TRUE; WriteLine[" ok"L]}
    ELSE WriteLine[" incorrect password"L]};

  -- Wizard Supporting Procedures
  wizardMode: BOOLEAN ← Process.MsecToTicks[2000] # 39; --FALSE for DLion only.
  wizardPassword: CARDINAL = 18939;

  Hash: PROCEDURE [s: LONG STRING] RETURNS [h: CARDINAL] =
    BEGIN
    h ← 17777;
    FOR i: CARDINAL IN [0..String.Length[s]) DO
      c: CHARACTER;
      IF (c ← s[i]) IN ['A..'Z] THEN c ← c + ('a-'A);
      h ← Inline.BITROTATE[h, 1] + (c-0C);
      ENDLOOP;
    END;

  Wizard: PUBLIC PROC RETURNS [BOOLEAN] = {RETURN[wizardMode]};

  -- Volume Init Supporting Procedures
  unknown: LONG STRING = "Unknown";

  GetSetBootFileSwitches: PROC [
    getSet: {get, set}, lvID: Volume.ID,
    ts: System.Switches ← System.defaultSwitches] = {
    outcome:   SetGetSwitchesSuccess;
    file:       File.File;
    firstPage: File.PageNumber;

    Heap.systemZone.FREE[@switches];
    IF getSet=get THEN SpecialVolume.OpenVolume[lvID, read]
    ELSE Volume.Open[lvID];
    [file, firstPage] ← GetVolumeBootFile[lvID, pilot];
    IF file = File.nullFile THEN AbortingCommand["No boot file found."L];
    IF getSet=get THEN [outcome, ts] ← GetSwitches[file, firstPage]
    ELSE outcome ← SetSwitches[file, firstPage, ts];
    OthelloToolDefs.CloseVolume[lvID];
    WriteSetDebuggerSuccess[outcome];
    IF getSet=set THEN RETURN;
    FOR c: CHARACTER IN [0C..377C] DO
      IF ts[c]=up THEN LOOP;
      SELECT c FROM
        '~, '-, '\\, '', '" => NULL;
        IN ['a..'z], IN ['A..'Z], IN (' ..'?] => {
          String.AppendCharAndGrow[@switches, c, Heap.systemZone]; LOOP};
        ENDCASE => NULL;
      String.AppendCharAndGrow[@switches, '\\, Heap.systemZone];
      String.AppendCharAndGrow[@switches, (c-0C)/64 + '0, Heap.systemZone];
      String.AppendCharAndGrow[@switches, ((c-0C)/8 MOD 8) + '0, Heap.systemZone];
      String.AppendCharAndGrow[@switches, ((c-0C) MOD 8) + '0, Heap.systemZone];
      ENDLOOP};

  WriteSetDebuggerSuccess: PROC [outcome: SetDebuggerSuccess] = {
    SELECT outcome FROM
      success => NULL;
      nullBootFile, cantWriteBootFile, notInitialBootFile =>
        AbortingCommand["Boot file broken."L];
      cantFindStartListHeader, startListHeaderHasBadVersion =>
        AbortingCommand["file built by incompatible version of StartPilot"L];
      noDebugger => AbortingCommand["No debugger installed."L];
      ENDCASE => ERROR};

  GetDriveFromUser: PUBLIC PROC RETURNS [h: PhysicalVolume.Handle] = {
    DO
      index: CARDINAL ← PhysicalVolume.nullDeviceIndex;
      GetName[
        "Drive Name: "L, @inputDriveString, echo, TRUE
        ! Question => {ListDrives[]; RESUME}];
      IF inputDriveString[inputDriveString.length - 1] = ': THEN
        inputDriveString.length ← inputDriveString.length - 1;
      DO
        index ← PhysicalVolume.GetNextDrive[index];
        IF index = PhysicalVolume.nullDeviceIndex THEN EXIT;
        h ← PhysicalVolume.GetHandle[index];
        IF String.Equivalent[GetDriveStringName[h], inputDriveString] THEN RETURN;
        ENDLOOP;
      WriteLine["Drive not found!"L]
      ENDLOOP};

  GetDriveNumber: PUBLIC PROC [h: PhysicalVolume.Handle] RETURNS [CARDINAL] = {
    RETURN[PhysicalVolume.InterpretHandle[h].index]};

  GetDriveStringName: PROC [h: PhysicalVolume.Handle] RETURNS [s: LONG STRING] =
    BEGIN
    s ← SELECT TRUE FROM
      -- damn compiler won't allow t IN Device.PilotDisk
      LOOPHOLE[GetDriveType[h], CARDINAL] IN Device.PilotDisk
            => "Rd?",
      ENDCASE => "UnknownType?";
```

```
    s[s.length - 1] ← GetDriveNumber[h] + '0;
    END;

  GetDriveType: PUBLIC PROC [h: PhysicalVolume.Handle] RETURNS [Device.Type] = {
    RETURN[PhysicalVolume.InterpretHandle[h].type]};

  GetLogicalVolumeName: PROC [vid: Volume.ID, s: STRING] = {
    s.length ← 0;
    Volume.GetLabelString[vid, s ! Volume.NeedsScavenging => GOTO bad];
    EXITS bad => {
      IDRep: TYPE = RECORD [p: ARRAY [0..3) OF CARDINAL, n: LONG CARDINAL];
      String.AppendString[s, "NeedsScavenging"L];
      String.AppendLongNumber[s, LOOPHOLE[vid, IDRep].n, 8]}};

  GetLogicalVolumeTypeName: PROC [vid: Volume.ID] RETURNS [LONG STRING] = {
    RETURN[logicalVolumeTypeString[Volume.GetType[vid ! ANY => GOTO signal]]];
    EXITS signal => RETURN[unknown]};

  -- Accept string of form LogicalVolumeName OR
  --    Drive:LogicalVolumeName
  GetLvIDFromUser: PUBLIC PROC [
    prompt: LONG STRING ← NIL,
    calledFromSetDebuggerPtrs: BOOLEAN ← FALSE]
    RETURNS [
      pvID: PhysicalVolume.ID, lvID: Volume.ID,
      drive: PhysicalVolume.Handle] =
    BEGIN
    IF prompt = NIL THEN prompt ← "Logical Volume Name: "L;
    DO
     ptmpID:      PhysicalVolume.ID ← PhysicalVolume.nullID;
     inputString: LONG STRING;
     matches:     CARDINAL ← 0;
     GetName[
        prompt: prompt, how: echo, signalQuestion: TRUE,
        dest: IF calledFromSetDebuggerPtrs THEN @debuggerLogicalString
              ELSE @inputLogicalString
        ! Question => {ListLogicalVolumes[]; RESUME}];
      IF calledFromSetDebuggerPtrs THEN {
        IF debuggerLogicalString.length=0 THEN {lvID ← Volume.nullID; RETURN}
        ELSE inputString ← debuggerLogicalString}
      ELSE {inputString ← inputLogicalString};
      DO
        driveTemp: PhysicalVolume.Handle;
        ltmpID:    Volume.ID ← Volume.nullID;
        IF (ptmpID ← PhysicalVolume.GetNext[ptmpID]) = PhysicalVolume.nullID THEN EXIT;
        driveTemp ← PhysicalVolume.GetAttributes[ptmpID].instance;
        DO
          s: STRING = [maxNameLength];
          IF (ltmpID ← PhysicalVolume.GetNextLogicalVolume[ptmpID, ltmpID])
             = Volume.nullID THEN EXIT;
          GetLogicalVolumeName[ltmpID, s ! Volume.NotOnline => LOOP];
          IF FunnyEqual[driveTemp, s, inputString] THEN {
            matches ← matches + 1; lvID ← ltmpID; pvID ← ptmpID; drive ← driveTemp};
          ENDLOOP;
        ENDLOOP;
      SELECT matches FROM
        0 => WriteString["Not found\r"L];
        1 => RETURN;
        ENDCASE => WriteLine["Ambigous; please specify Device:LogicalName"L];
      ENDLOOP;
    END;

  FunnyEqual: PROC [
    h: PhysicalVolume.Handle, name: STRING, userName: LONG STRING,
    mode: {checkNakedPName, dontCheckNakedPName} ← dontCheckNakedPName]
    RETURNS[BOOLEAN] = {
    driveName: LONG STRING;
    SameChar: PROC [a, b: CHARACTER]
      RETURNS [BOOLEAN] = {
      IF a=b THEN RETURN[TRUE]
      ELSE IF a IN ['a..'z] AND b IN ['A..'Z] AND (a-'a+'A)=b THEN RETURN[TRUE]
      ELSE IF a IN ['A..'Z] AND b IN ['a..'z] AND (a-'A+'a)=b THEN RETURN[TRUE]
      ELSE RETURN[FALSE]};
    IF String.Equivalent[name, userName] THEN RETURN[TRUE];
    driveName ← GetDriveStringName[h];
    IF userName.length < driveName.length THEN RETURN [FALSE];
    FOR i: CARDINAL IN [0..driveName.length) DO
      IF ~SameChar[driveName[i], userName[i]] THEN RETURN[FALSE] ENDLOOP;
    IF mode=checkNakedPName THEN {
      IF (userName.length=driveName.length)
         OR (userName.length=driveName.length+1
             AND userName[driveName.length] = ':) THEN RETURN[TRUE]};
    IF driveName.length+name.length+1 # userName.length THEN RETURN[FALSE];
    IF userName[driveName.length] # ': THEN RETURN[FALSE];
    FOR i: CARDINAL IN [0..name.length) DO
      IF ~SameChar[name[i], userName[driveName.length+1+i]] THEN RETURN[FALSE]
      ENDLOOP;
    RETURN[TRUE]};

  GetLvTypefromUser: PUBLIC PROC [
    prompt: LONG STRING, defaultType: Volume.Type] RETURNS [Volume.Type] =
    BEGIN
    ListTypes: PROC = {
      FOR t: Volume.Type IN [normal..nonPilot] DO
        WriteString[logicalVolumeTypeString[t]];
        WriteString[IF t = nonPilot THEN "\r"L ELSE ", "L];
        ENDLOOP};
```

```
    String.Replace[@lvTypeString, logicalVolumeTypeString[defaultType], Heap.systemZone];
    DO
      GetName[prompt, @lvTypeString, echo, TRUE
      ! Question => {ListTypes[]; RESUME}];
      FOR t: Volume.Type IN [normal..nonPilot] DO
        IF String.Equivalent[logicalVolumeTypeString[t], lvTypeString] THEN
          RETURN[t]
        ENDLOOP;
      WriteLine["Illegal type"L];
      ENDLOOP;
    END;

  -- Accept string of Form PhysicalVolumeName OR
  --   Drive:PhysicalVolumeName OR Drive
  GetPvIDFromUser: PROC
    RETURNS [id: PhysicalVolume.ID, drive: PhysicalVolume.Handle] =
    BEGIN
    DO
      tmpID:   PhysicalVolume.ID ← PhysicalVolume.nullID;
      matches: CARDINAL ← 0;
      GetName["Physical Volume Name: "L, @inputPhysString, , TRUE
      ! Question => {ListPhysicalVolumes[]; RESUME}];
      DO
        s:          STRING = [maxNameLength];
        match:      BOOLEAN;
        driveTemp: PhysicalVolume.Handle;
        IF (tmpID ← PhysicalVolume.GetNext[tmpID]) = PhysicalVolume.nullID THEN
          EXIT;
        driveTemp ← PhysicalVolume.GetAttributes[tmpID, s].instance;
        match ← FunnyEqual[driveTemp, s, inputPhysString, checkNakedPName];
        IF match THEN {matches ← matches + 1; id ← tmpID; drive ← driveTemp};
        ENDLOOP;
      SELECT matches FROM
        0 => WriteLine["Not Found"L];
        1 => RETURN;
        ENDCASE => WriteLine["Ambigous; please specify Device:PhysicalName"L];
      ENDLOOP;
    END;

  StringInit: PROC = {
    SetCommandString[String.CopyToNewString["Online RDO"L, Heap.systemZone]];
    debuggerLogicalString ← String.CopyToNewString["CoPilot"L, Heap.systemZone];

  RegisterCommandProc[@commandProcessor];

  StringInit[];

  END.....
```

```
March 19, 1980  3:47 PM FXH     Delete newly created temporary files when fetch fails; ome indentation changing
April 16, 1980  12:16 PM        RXG     Addd diagnostic microcode fetch
May 31, 1980  11:49 PM  FXH     Shuffle around VolumeInitImplA and B
July 30, 1980  4:33 PM  AWL             Permit Online'ing an already online volume
September 18, 1980  12:04 PM    PXM     Don't bother to open volume to boot from
September 19, 1980  11:24 AM    AWL             physicalVolumeOverhead ← 2 for new physical volume format.
September 29, 1980  2:07 PM     CAJ     Add SA800 format and scan, USING clauses.
October 10, 1980  3:17 PM       FXH     Version 5.0.
January 5, 1981  10:14 PM       FXH     Made use String for appendChar, equivilantString, appendLongNumber.  Add TemporaryBooting.invalid
paramater catch.
January 31, 1981  9:19 PM       CAJ     Fix format prompt.
March 1, 1981  12:59 PM AWL             Version => 6.0b.
March 13, 1981  7:22 PM SXY     Version => 6.0c, trouple => trouble (correction), "Boot file header broken" => "Error: Debuggee built by
incompatible version of StartPilot".
March 25, 1981  8:28 PM CRF     Version => 6.0.
April 14, 1981  11:38 AM        BXM     @ added.
11-Jun-81 10:53:01      Taft    Remove all machine- and device-dependent code to separate module OthelloDeviceImplD*.mesa
17-Jul-81 15:34:33      SCG             Merged OthelloDevice into OthelloDefs
12-Aug-81 12:33:54      SXY     Added a catch phrase for Volume.GetAttributes and commented it out for Volume.GetLabelString
 5-Dec-81 17:30:28      CRF     Converted from PhysicalVolumeExtras to PhysicalVolume for PV scavenger.
11-Dec-82 15:10:21      RXJ             Removed Storage.
13-Apr-83 12:27:04      RXJ             Klamath conversion
 4-Jun-86 14:09:04      NFS             Adapted for OthelloTool
```

```
-- VolumeVersion.mesa    17-Aug-83 12:16:18 by WDK

-- TEMPORARY HACK for Klamath <> Trinity cross volume problems.

DIRECTORY
  Volume USING [ID];

VolumeVersion: DEFINITIONS =
  BEGIN

  Examine: PROCEDURE [volume: Volume.ID]
    RETURNS [result: Result];

  Result: TYPE = {
    currentVersion, badRootPageLabel, ioError, trashedRootPage,
    otherVersion, volumeUnknown};

  END.
```

```
-- Copyright (C) 1983, 1984  by Xerox Corporation, All rights reserved.
-- VolumeVersionImpl.mesa       17-Aug-83 11:20:47 by WDK
-- BJD           28-Feb-84 14:10:16

-- TEMPORARY HACK for Klamath <> Trinity cross volume problems.

DIRECTORY
  Device,
  DiskChannel,
  Environment,
  File,
  LogicalVolumeFormat,
  OthelloOps,
  PhysicalVolume,
  PilotDisk,
  PilotFileTypes,
  Space,
  VM,
  Volume,
  VolumeVersion;

VolumeVersionImpl: PROGRAM
  IMPORTS
    DiskChannel, Environment, OthelloOps, PhysicalVolume, PilotDisk, Space, VM
  EXPORTS VolumeVersion =
  BEGIN

  Bug: SIGNAL [b: BugType] = CODE;
  BugType: TYPE = {
    impossibleEndcase, invalidChannel, invalidDriveState};

  Examine: PUBLIC PROCEDURE [volume: Volume.ID]
    RETURNS [result: VolumeVersion.Result] =
    BEGIN
    lvRoot: LONG POINTER TO LogicalVolumeFormat.Descriptor;
    physicalVol: PhysicalVolume.ID =
      PhysicalVolume.GetContainingPhysicalVolume[volume];
    pvHandle: PhysicalVolume.Handle =
      PhysicalVolume.GetAttributes[physicalVol].instance;
    deviceType: Device.Type;
    index: CARDINAL;
    subVolume: OthelloOps.SubVolume ← OthelloOps.nullSubVolume;
    drive: DiskChannel.Drive;
    DO
      subVolume ← OthelloOps.GetNextSubVolume[physicalVol, subVolume];
      IF subVolume = OthelloOps.nullSubVolume THEN RETURN[volumeUnknown];
      IF subVolume.lvID = volume AND subVolume.firstLVPageNumber = 0 THEN EXIT;
      ENDLOOP;

    lvRoot ← Space.ScratchMap[count: 1].pointer;
    [type: deviceType, index: index] ← PhysicalVolume.InterpretHandle[pvHandle];
    FOR drive ← DiskChannel.GetNextDrive[prev: DiskChannel.nullDrive],
       DiskChannel.GetNextDrive[prev: drive]
      UNTIL drive = DiskChannel.nullDrive DO
        dType: Device.Type;
        dOrdinal: CARDINAL;
        [deviceType: dType, deviceOrdinal: dOrdinal] ←
          DiskChannel.GetDriveAttributes[drive];
        IF dType=deviceType AND dOrdinal=index THEN EXIT
        ENDLOOP;

    BEGIN  --scope of Exit--
    pageBuffer: Environment.PageNumber = Environment.PageFromLongPointer[lvRoot];
    channel: DiskChannel.Handle = DiskChannel.Create[drive];
    request: DiskChannel.IORequest ← [
      diskPage: subVolume.firstPVPageNumber, memoryPage: pageBuffer,
      tries: DiskChannel.defaultTries, label: @label,
      count: 1, useSamePage: TRUE, command: [verify, read, read]];
    status: DiskChannel.IOStatus;
    countValid: File.PageCount;
    label: PilotDisk.Label;

    VM.MakeResident[[page: pageBuffer, count: 1], wait];
    [status, countValid] ← DiskChannel.DoIO[channel, @request];
    WITH boundStatus: status SELECT FROM
      invalidChannel => Bug[invalidChannel];
      invalidDriveState => Bug[invalidDriveState];
      disk =>
        IF boundStatus.status # goodCompletion THEN {result ← ioError; GOTO Exit};
      ENDCASE => Bug[impossibleEndcase];

    -- Check Logical Root Page Label:
    IF volume = Volume.ID[label.fileID.id]
      AND PilotDisk.GetLabelFilePage[@label] = LogicalVolumeFormat.rootPageNumber
      AND ~label.temporary AND label.pad1 = 0 AND label.pad2 = 0
      AND label.type = PilotFileTypes.tLogicalVolumeRootPage
      THEN NULL ELSE {result ← badRootPageLabel; GOTO Exit};

    IF lvRoot.seal # LogicalVolumeFormat.lvRootSeal THEN
      {result ← trashedRootPage; GOTO Exit};
    IF lvRoot.version # LogicalVolumeFormat.currentVersion THEN
      {result ← otherVersion; GOTO Exit};
    result ← currentVersion;

    EXITS Exit => NULL
    END;
```

```
    lvRoot ← Space.Unmap[lvRoot];
    END;

END.
```

```
-- Copyright (C) 1982, 1983  by Xerox Corporation. All rights reserved.
-- OthelloOps.mesa      16-Aug-83 13:33:02 by JXP

DIRECTORY
  Device         USING [Type],
  File           USING [File, PageNumber],
  PhysicalVolume USING [Handle, ID, PageNumber],
  System         USING [GreenwichMeanTime, LocalTimeParameters, Switches],
  Volume         USING [ID, nullID, PageCount];


-- Operations useful to volume utilities.
OthelloOps: DEFINITIONS =

  BEGIN

  DecodeSwitches: PROC [switchString: LONG STRING]
    RETURNS [switches: System.Switches];
  BadSwitches: ERROR;

  DeleteTempFiles: PROCEDURE [Volume.ID];
  VolumeNotClosed: ERROR;

  GetDriveSize: PROCEDURE [h: PhysicalVolume.Handle]
    RETURNS [nPages: LONG CARDINAL];
  -- returns total size, ignoring reserved spaces, etc.

  BootFileType: TYPE = {hardMicrocode, softMicrocode, germ, pilot};

  MakeBootable, MakeUnbootable: PROCEDURE [
    file: File.File, type: BootFileType, firstPage: File.PageNumber];
  InvalidVersion: ERROR;
  -- Prepare (or undo) chains for making a file bootable.
  --  There is no need to remove bootlinks merely to delete
  -- a file.  Boot links should be removed before any increase or decrease in
  -- the size of a file, and reinstalled after the operation.

  SetVolumeBootFile, SetPhysicalVolumeBootFile: PROCEDURE [
    file: File.File, type: BootFileType, firstPage: File.PageNumber];

  GetVolumeBootFile: PROCEDURE [lvID: Volume.ID, type: BootFileType]
    RETURNS [file: File.File, firstPage: File.PageNumber];

  GetPhysicalVolumeBootFile:
    PROCEDURE [pvID: PhysicalVolume.ID, type: BootFileType]
    RETURNS [file: File.File, firstPage: File.PageNumber];

  VoidVolumeBootFile: PROCEDURE [lvID: Volume.ID, type: BootFileType];
  VoidPhysicalVolumeBootFile:
    PROCEDURE [pvID: PhysicalVolume.ID, type: BootFileType];

  SetDebuggerSuccess: TYPE = {
    success, nullBootFile, cantWriteBootFile, notInitialBootFile,
    cantFindStartListHeader, startListHeaderHasBadVersion, other, noDebugger};
  SetGetSwitchesSuccess:    TYPE = SetDebuggerSuccess[success..other];
  SetExpirationDateSuccess: TYPE = SetDebuggerSuccess[success..other];
  GetExpirationDateSuccess: TYPE = SetDebuggerSuccess[success..other];

  SetDebugger: PROCEDURE [
    debuggeeFile: File.File, debuggeeFirstPage: File.PageNumber,
    debugger: Volume.ID, debuggerType: Device.Type, debuggerOrdinal: CARDINAL]
    RETURNS [SetDebuggerSuccess];

  SetExpirationDate: PROCEDURE [
    file: File.File, firstPage: File.PageNumber,
    expirationDate: System.GreenwichMeanTime]
    RETURNS [SetExpirationDateSuccess];

  GetExpirationDate: PROCEDURE [file: File.File, firstPage: File.PageNumber]
    RETURNS [GetExpirationDateSuccess, System.GreenwichMeanTime];

  SetSwitches: PROCEDURE [
    file: File.File, firstPage: File.PageNumber, switches: System.Switches]
    RETURNS [SetGetSwitchesSuccess];

  GetSwitches: PROCEDURE [file: File.File, firstPage: File.PageNumber]
    RETURNS [SetGetSwitchesSuccess, System.Switches];

  -- Physical layout of subvolumes on a physical Volume:

  LogicalVolumePageNumber: TYPE = LONG CARDINAL;
  SubVolume: TYPE = RECORD [
    lvID:              Volume.ID,
    subVolumeSize:     Volume.PageCount,
    firstLVPageNumber: LogicalVolumePageNumber,
    firstPVPageNumber: PhysicalVolume.PageNumber];

  nullSubVolume: SubVolume = [Volume.nullID, 0, 0, 0];

  GetNextSubVolume: PROCEDURE [pvID: PhysicalVolume.ID, thisSv: SubVolume]
    RETURNS [nextSV: SubVolume];
  SubVolumeUnknown: ERROR [sv: SubVolume];

  -- Time related functions:

  IsTimeValid: PROCEDURE RETURNS [valid: BOOLEAN];
  -- Returns TRUE if the processor clock appears to be set correctly.

OthelloOps.mesa        16-Aug-83 13:34:09 PDT
```

```
  SetProcessorTime: PROCEDURE [time: System.GreenwichMeanTime];
  -- Sets the processor clock to the specified value.  In general, the
  -- clock should NOT be set backwards from that returned by
  -- System.GetGreenwhichMeanTime.

  GetTimeFromTimeServer: PROCEDURE []
     RETURNS [
       serverTime: System.GreenwichMeanTime,
       serverLTPs: System.LocalTimeParameters];
  -- Attempts to access a time server to determine what time it currently is
  -- and the local time parameters that the server is using.  Will
  -- raise TimeServerError if this operation fails.

  TimeServerError: ERROR [error: TimeServerErrorType];
  TimeServerErrorType: TYPE = {noCommunicationFacilities, noResponse};

  END.


--LOG

July 13, 80 7:17 PM  FXH
   Create from old OthelloOps and Old OthelloInternalInterface
11-Aug-81 15:14:43   AWL
   DecodeSwitches takes a LONG STRING.  Switches now in System.  Added SetProcessorTime,
   GetTimeFromTimeServer and TimeServerError.
10 Nov 81 13:46:00   JXP      Added InvalidVersion ERROR.
12-Nov-81  9:46:49   FXH       Add SetExpirationDate.
24-Nov-82  9:12:20   AWL
   Add GetExpirationDate, GetExpirationDateSuccess.  File.Capability => File.File.  Renamed "cap" arguments to "file".
27-Jun-83 17:40:34   WDK       Added pilotSnapshot.
Time: 11-Jul-83 14:30:56        By: JXP
   The parm. debuggeeCap => debuggeeFile in SetDebugger.
Time: 16-Aug-83 13:33:12        By: JXP
   Decommision the pilotSnapshot BootFileType for the time being.
```

```
-- Copyright (C) 1983  by Xerox Corporation. All rights reserved.
-- SpecialVolume.mesa      2-Jul-83 18:18:31 by WDK

DIRECTORY
  Boot USING [LVBootFiles, PVBootFiles],
  PhysicalVolume USING [ID, PageNumber],
  System USING [GreenwichMeanTime],
  Volume USING [ID, nullID, PageCount, Type];

SpecialVolume: DEFINITIONS =
  BEGIN

  LogicalVolumePageNumber: TYPE = LONG CARDINAL;

  SubVolume: TYPE = RECORD [
    lvID: Volume.ID,
    subVolumeSize: Volume.PageCount,
    firstLVPageNumber: LogicalVolumePageNumber,
    firstPVPageNumber: PhysicalVolume.PageNumber];

  nullSubVolume: SubVolume = [
    Volume.nullID, LAST[Volume.PageCount], LAST[LogicalVolumePageNumber], LAST[
    PhysicalVolume.PageNumber]];

  GetNextSubVolume: PROCEDURE [pvID: PhysicalVolume.ID, this: SubVolume]
    RETURNS [next: SubVolume];
  -- Stateless enumeration of subVolumes.
  -- Enumerations starts and ends with nullSubVolume.

  SubVolumeUnknown: ERROR;


  OpenVolume: PROCEDURE [volume: Volume.ID, access: Access];
  -- Opens a volume while explicitly controlling its writability
  -- (independent of its type).

  Access: TYPE = {read, readWrite, default};


  ChangeVolumeType: PROCEDURE [lvID: Volume.ID, newType: Volume.Type];
   << Changes the type of a volume. Note that changing a volume's type affects
  the selection of its debugger. The safest procedure is to change a volume's
  type from UtilityPilot with no other logical volumes open
  and then reinstall all debuggers on that physical volume.
  If the volume is not open for read/Write, InvalidParameters is raised. >>

  InvalidParameters: ERROR;

  LastOpenedForWrite: PROCEDURE [lvID: Volume.ID] RETURNS [System.GreenwichMeanTime];


  -- Get/set boot file pointers:

  GetLogicalVolumeBootFiles: PROCEDURE [
    lvID: Volume.ID, pBootFiles: LONG POINTER TO Boot.LVBootFiles];

  GetPhysicalVolumeBootFiles: PROCEDURE [
    pvID: PhysicalVolume.ID, pBootFiles: LONG POINTER TO Boot.PVBootFiles];

  SetLogicalVolumeBootFiles: PROCEDURE [
    lvID: Volume.ID, pBootFiles: LONG POINTER TO Boot.LVBootFiles];

  SetPhysicalVolumeBootFiles: PROCEDURE [
    pvID: PhysicalVolume.ID, pBootFiles: LONG POINTER TO Boot.PVBootFiles];

  END.

LOG

May 15, 80 4:40 PM   PXM
   Equate PhysicalVolume.ID to same type in System
Jun 28, 80 5:23 PM   FXH
   Eliminate procedures now defined in other interfaces, and ReGroup adding some comments about moving Procedures.  Move
   [Get|Set][Logical|Physiccal]VolumeBootFiles to KernelFile,
   GetContaining PhysicalVolume to PhysicalVolume.
Jul 13, 80 9:15 PM   FXH         Move most procedures elsewhere.
Jul 28, 80 2:33 PM   AWL         Added nullSubVolume, SubVolumeUnknown.
11-Aug-81 11:29:26   AWL         Added OpenVolume from VolumeExtras.
 2-Nov-81 11:28:26   AWL
   Changed "readOnly: BOOLEAN" arg to OpenVolume to "access: Access".
 5-Oct-82 16:15:15   AWL
   Added ChangeVolumeType procedure and InvalidParameters error.
 3-Feb-83 16:40:20   LXD
   Redefined error conditions for ChangeVolumeType.
 2-Jul-83 18:18:36   WDK         Improved documentation, Added LastOpenedForWrite.
```

```
-- Copyright (C) 1982, 1983  by Xerox Corporation. All rights reserved.
-- TemporaryBooting.mesa (last edited by: JXP    on:   6-Jul-83 15:23:15)


DIRECTORY
  Environment USING [PageCount],
  File USING [File, PageNumber, Type],
  PhysicalVolume USING [ID],
  System USING [defaultSwitches, Switches],
  Volume USING [ID];

TemporaryBooting: DEFINITIONS =

  BEGIN

  tBootFile: READONLY File.Type;
     -- Only files of this type may be made bootable.

  MakeBootable: PROCEDURE [file: File.File, firstPage: File.PageNumber ← 0];
     -- Install the chaining links in the labels of the file pages starting with the specified
     -- first page and for the length of the contained Pilot boot file.
     -- MakeBootable must be performed after contents are written and before file is passed to
     -- BootFrom.
     -- InvalidParameters is raised if file is of wrong type or doesn't contain Pilot boot file
     -- starting at firstPage.

  MakeUnbootable: PROCEDURE [
    file: File.File, firstPage: File.PageNumber ← 0];
     -- Remove the chaining links.  May speed subsequent bulk access to the file.

  InstallVolumeBootFile: PROCEDURE [
    file: File.File, firstPage: File.PageNumber ← 0];
     -- Set up the file as the one gaining control when the containing (logical) volume is booted.

  InstallPhysicalVolumeBootFile: PROCEDURE [
    file: File.File, firstPage: File.PageNumber ← 0];
     -- Set up the (logical) volume as the one gaining control when the containing physical
     -- volume is booted.

  BootFromFile: PROCEDURE [
    file: File.File, firstPage: File.PageNumber ← 0,
    switches: System.Switches ← System.defaultSwitches];
     -- Restart the system from the specified boot file.  Improper arguments will likely result
     -- in an MP code and crash.

  BootFromVolume: PROCEDURE [
    volume: Volume.ID, switches: System.Switches ← System.defaultSwitches];
     -- Restart the system from the Pilot boot file installed on the specified volume.

  BootFromPhysicalVolume: PROCEDURE [
    volume: Volume.ID, switches: System.Switches ← System.defaultSwitches];
     -- Restart the system from the Pilot boot file installed on the physical volume
     -- containing the specified (logical) volume.

  BootButton: PROCEDURE [switches: System.Switches ← System.defaultSwitches];
  -- Restart the system as if its boot button had been pressed (may ignore switches).

  BootLocation: TYPE = RECORD [
    body: SELECT bootLocation: * FROM
      bootButton, none => NULL,
      physicalVolume => [pvLocation: PVLocation],
      logicalVolume => [volumeLocation: VolumeLocation],
      file => [fileLocation: FileLocation],
      ENDCASE];
  -- Describes a place that the state of a running Pilot may be saved in, or
  -- restored from.
  -- Currently, it is only possible to save state in a file BootLocation.
  -- A bootButton BootLocation is always valid.  The other variants are only valid
  -- for limited periods of time, as denoted below.  The conservative approach is
  -- to never store these other variants in a permanent location and to always
  -- recreate them just before calling OutLoadInLoad.

  PVLocation: TYPE [11];
  VolumeLocation: TYPE [11];
  FileLocation: TYPE [11];

  -- The following procedures return a BootLocation for the specified location.
  -- For each operation, the circumstances under which the returned information
  -- becomes invalid is noted.

  GetFileLocation: PROCEDURE [
    file: File.File, firstPage: File.PageNumber ← 0]
    RETURNS [bootLocation: file BootLocation];
  -- The returned BootLocation is valid so long as the specified file is
  -- neither deleted or has any of its attributes changed (including size
  -- and permanency).  Scavenging may invalidate the returned BootLocation
  -- if the file was damaged and the client scavenger repaired the damage.
  -- The returned BootLocation is also only valid if the specified file has
  -- been made bootable (via TemporaryBooting.MakeBootable) and is not
  -- subsequently made unbootable.
  -- May raise File.Unknown, Volume.Unknown.
  -- Raises TemporaryBooting.InvalidParameters if the specified file page is
  -- beyond the end of the file.

  GetVolumeLocation: PROCEDURE [volume: Volume.ID]
    RETURNS [bootLocation: logicalVolume BootLocation];
  -- The returned BootLocation is only valid so long as the boot files are
```

```
  -- not changed on the specified volume.  The comments for the validity of
  -- returned BootLocations in GetFileLocation also apply here.
  -- Raises TemporaryBooting.InvalidParameters if the specified volume does
  -- not have a Pilot boot file installed on it.
  -- May raise Volume.Unknown, Volume.NotOpen.

  GetPVLocation: PROCEDURE [volume: PhysicalVolume.ID]
    RETURNS [bootLocation: physicalVolume BootLocation];
  -- Valid so long as the boot files are not changed on the specified
  -- physical volume.  The comments for the validity of returned BootLocations
  -- in GetFileLocation also apply here.
  -- Raises TemporaryBooting.InvalidParameters if the specified volume does
  -- not have a Pilot boot file installed on it.
  -- May raise PhysicalVolume.Error[physicalVolumeUnknown].

  OutLoadInLoad: PROCEDURE [
    outloadLocation: file BootLocation,  -- loaciton to save current system
    -- The following describe the instance of Pilot to be restored
    inloadLocation: BootLocation,
    pMicrocode, pGerm: LONG POINTER ← NIL, countGerm: Environment.PageCount ← 0,
    switches: System.Switches ← System.defaultSwitches];
  -- The state of the currently running Pilot is saved on outloadLocation.
  -- The Pilot represented by inloadLocation is restored to a running state.
  -- The microcode and/or germ may be changed by passing the appropriate
  -- information in pMicrocode, pGerm and countGerm.  If these pMicrocode is
  -- defaulted, the microcode is not changed.  If pGerm is defaulted, the germ
  -- is not changed.
  -- The switches are available to the inloaded Pilot.  These are typically
  -- only examined when the Pilot being booted is not an outload file (e.g., it
  -- was made by MakeBoot).  Note that the switches may be ignored if
  -- inloadLocation is a bootButton BootLocation.
  -- Upon return, the client has successfully performed the outload and someone
  -- else later has requested that THIS instance of Pilot be inloaded.

  InvalidParameters: ERROR;
  InvalidVersion: ERROR; -- Note that this error is raised by MakeBootable AFTER the
  -- the file is made bootable.

  END.



--LOG

Time: September 14, 1979  2:45 PM       By: PXM
   Create file
Time: October 3, 1979  6:17 PM  By: PXM
   Add InstallAsBoot*
Time: January 25, 1980  10:58 AM        By: PXM
   InstallAsBootFile=>InstallVolumeBootFile;
   InstallAsBootVolume[vol]=>InstallPhysicalBootVolume[file]; add BootFromPhysicalVolume,
   Boot; add Switches to Boot*
Time: April 17, 1980  10:34 AM  By: PXM
   Default for firstPage to BootFrmFile missing
Time:  9-Aug-81 15:03:38        By: AWL
   Switches now in System.
Time: 10-Nov-81  13:45:00       By: JXP
   Added InvalidVersion ERROR.
Time: 22-Nov-82 16:03:03        By: AWL
   File.Capability => File.File.
Time: 30-Mar-83 17:25:52        By: AWL
   Added BootLocation, PVLocation, VolumeLocation, FileLocation, GetFileLocation, GetVolumeLocation, GetPVLocation, OutLoadInLoad.
Time:  6-Jul-83 15:25:51        By: JXP
   Added "none" variant to BootLocation
```

```
-- File: QuickRestartClientImpl.mesa - last edit:
-- Mita                 27-Jan-88 14:15:09
-- Breisacher           21-Oct-87  9:40:58


-- Copyright (C) 1987, 1988 by Xerox Corporation. All rights reserved.

DIRECTORY
  Boot USING [LVBootFiles],
  File USING [Delete, File, nullFile, Unknown],
  LevelIVKeys USING [KeyBits, KeyName],
  NSConstants,
  Space USING [CopyIn, ScratchMap, Unmap],
  String USING [Equivalent],
  SpecialVolume USING [GetLogicalVolumeBootFiles, LastOpenedForWrite, OpenVolume],
  OthelloOps USING [GetTimeFromTimeServer, SetProcessorTime, TimeServerError],
  PhysicalVolume USING [
    AssertPilotVolume, Error, GetHandle, GetNextDrive, Handle, ID,
    NeedsScavenging, nullDeviceIndex, nullID],
  PilotMP USING [cScavenging, cDiskHardwareError, cTimeNotAvailable, cClient],
  ProcessorFace USING [SetMP],
  PowerOffRestartInternal USING [CheckHandle, outloadFileType, firstPage],
  Router USING [FindDestinationRelativeNetID],
  Scavenger USING [Scavenge],
  System USING [GreenwichMeanTime, LocalTimeParameters, NetworkNumber, nullNetworkNumber, SetLocalTimeParameters],
  TemporaryBooting USING [BootFromFile, BootFromVolume, InstallPhysicalVolumeBootFile, InvalidParameters],
  UserTerminal USING [WaitForScanLine, keyboard],
  Volume USING [GetNext, GetLabelString, ID, maxNameLength, NeedsScavenging, nullID,
    LookUpRootFile, Open, RootDirectoryError, Unknown];

QuickRestartClientImpl: PROGRAM
  IMPORTS
    File, OthelloOps, PhysicalVolume, ProcessorFace, Router, Scavenger, Space, SpecialVolume, String,
    System, TemporaryBooting, UserTerminal, Volume =
  BEGIN

 userVolume: Volume.ID ← Volume.nullID;

 CheckOutloadOK: PROC [volumeName: LONG STRING]
    RETURNS [inload: File.File, proceed: BOOLEAN] =
    BEGIN
    outloadFile: File.File ← File.nullFile;
    userVolume ← GetVolumeID[volumeName];
    proceed ← FALSE;
    IF userVolume # Volume.nullID THEN {
      SpecialVolume.OpenVolume[userVolume, read!Volume.NeedsScavenging=>  {LogicalVolumeScavenge[userVolume]; CONTINUE}]; -- delete
      because this is permanent.
      outloadFile ← Volume.LookUpRootFile[
        PowerOffRestartInternal.outloadFileType, userVolume!Volume.RootDirectoryError => GOTO Errors];
      IF outloadFile # File.nullFile THEN {
        check: PowerOffRestartInternal.CheckHandle = Space.ScratchMap[1];
        proceed ← CheckOutloaded[outloadFile, check!File.Unknown=> GOTO Errors];
        IF proceed THEN proceed ←  (check.lastNetworkNumber = System.nullNetworkNumber) OR (LocalNet[] = check.lastNetworkNumber);
        inload ← outloadFile;
        [] ← Space.Unmap[check]};
      };
    EXITS Errors => {proceed ← FALSE; RETURN};
    END;

  LogicalVolumeScavenge: PROC [volume: Volume.ID] =
    BEGIN
    scavengerlog: File.File;
    ProcessorFace.SetMP[PilotMP.cScavenging];
    scavengerlog ← Scavenger.Scavenge[volume, volume, safeRepair, FALSE];
    Volume.Open[volume];
    File.Delete[scavengerlog];
    ProcessorFace.SetMP[PilotMP.cClient];
    END;


  CheckOutloaded: PROC [file: File.File, check: PowerOffRestartInternal.CheckHandle]
    RETURNS [out: BOOLEAN] = {
    scavengerVolume: Volume.ID ← GetVolumeID["Scavenger"L];
    [] ← Space.CopyIn[check, [file, 0, 1]];
    out ← check.valid AND
        (check.lastWriteDateinUserVolume = SpecialVolume.LastOpenedForWrite[
          userVolume])
      AND
        (check.lastWriteDateinScavengeVolume = SpecialVolume.LastOpenedForWrite[
          scavengerVolume])};

  GetVolumeID: PROC [svVolume: LONG STRING] RETURNS [vl: Volume.ID] =
    BEGIN
    svLabel: STRING = [Volume.maxNameLength];
    vl ← Volume.nullID;
    UNTIL (vl ← Volume.GetNext[vl]) = Volume.nullID DO
      Volume.GetLabelString[vl, svLabel];
      IF String.Equivalent[svVolume, svLabel] THEN EXIT;
      ENDLOOP;
    END;  -- of GetVolumeID

  LocalNet: PROC RETURNS [System.NetworkNumber] = {
    RETURN[Router.FindDestinationRelativeNetID[System.nullNetworkNumber]];
    };

  Run: PROCEDURE =
    BEGIN
```

```
    userVolName: LONG STRING ← "User"L;
    timeFromServer: BOOLEAN ← TRUE;
    doInload: BOOLEAN ← TRUE;
    time: System.GreenwichMeanTime;
    ltps: System.LocalTimeParameters;
    inloadFile: File.File;
    ProcessorFace.SetMP[PilotMP.cTimeNotAvailable];

    [time, ltps] ← OthelloOps.GetTimeFromTimeServer[
      ! OthelloOps.TimeServerError => RETRY];
    IF timeFromServer THEN {
      System.SetLocalTimeParameters[ltps]; OthelloOps.SetProcessorTime[time]};
    -- If there is no Time server boot from scratch.
    doInload ← timeFromServer;
    IF doInload THEN doInload ← BringFirstReadyPhysicalVolumeOnline[];
    IF ~doInload THEN  -- Can not Online Volume
      {ProcessorFace.SetMP[PilotMP.cDiskHardwareError]; RETURN}
    ELSE {FOR i:CARDINAL IN[0..5) DO UserTerminal.WaitForScanLine[0];
    doInload ← ~KeysAreDown[key1: N, key2: B];
    IF ~doInload THEN EXIT;
    ENDLOOP;
    };
    IF doInload THEN [inloadFile, doInload] ← CheckOutloadOK[userVolName];
    IF doInload THEN [] ← TemporaryBooting.BootFromFile[
      inloadFile, PowerOffRestartInternal.firstPage!TemporaryBooting.InvalidParameters, Volume.Unknown, File.Unknown,
      Volume.NeedsScavenging => {doInload ← FALSE; CONTINUE}];
    -- "only returns if couldn't inload" WDK
    IF userVolume=Volume.nullID THEN userVolume ← GetVolumeID["User"L];
    SetPhysicalBoot[userVolume];
    TemporaryBooting.BootFromVolume[userVolume]
    END;  -- Run

  KeysAreDown: PROCEDURE [key1, key2: LevelIVKeys.KeyName] RETURNS [BOOLEAN] = {
    keys: LONG POINTER TO LevelIVKeys.KeyBits = LOOPHOLE[UserTerminal.keyboard];
    RETURN[(keys[key1] = down) AND (keys[key2] = down)];
    };

  BringFirstReadyPhysicalVolumeOnline: PROC RETURNS [BOOLEAN] =
    BEGIN
    driveIndex: CARDINAL ← PhysicalVolume.nullDeviceIndex;
    driveHandle: PhysicalVolume.Handle;
    pvID: PhysicalVolume.ID;

    DO  -- loop looking for drives connected to this machine
      pvID ← PhysicalVolume.nullID;
      driveIndex ← PhysicalVolume.GetNextDrive[driveIndex];
      IF driveIndex = PhysicalVolume.nullDeviceIndex THEN EXIT;
      driveHandle ← PhysicalVolume.GetHandle[driveIndex];
      IF TRUE <<PhysicalVolume.IsReady[driveHandle]>> THEN
        BEGIN
        <<IsReady doesn't work properly. Thus the following code is necessary.>>
        pvID ← PhysicalVolume.AssertPilotVolume[
          driveHandle !
          PhysicalVolume.Error, PhysicalVolume.NeedsScavenging => CONTINUE];
        IF pvID = PhysicalVolume.nullID THEN EXIT;
        RETURN[TRUE];
        END;
      ENDLOOP;  -- end of loop until a drive is selected
    RETURN[FALSE];
    END;  -- of BringFirstReadyPhysicalVolumeOnline

  SetPhysicalBoot: PROCEDURE [lvID: Volume.ID] = {
    bootFiles: Boot.LVBootFiles;
    SpecialVolume.GetLogicalVolumeBootFiles[lvID, @bootFiles];
    Volume.Open[lvID!Volume.NeedsScavenging=> {LogicalVolumeScavenge[lvID]; CONTINUE;}];
    TemporaryBooting.InstallPhysicalVolumeBootFile [
      file: [bootFiles[pilot].fID.fileID, lvID],
      firstPage: bootFiles[pilot].firstPage];
    };

  --Main
  Run[];

  END.
LOG
22-Jan-87 10:47:55 - Mita - Create from QuickBoot hack.
26-Jan-87 12:36:40 - Mita - Support Time service
 4-Feb-87 19:13:30 - Mita - Incorporate suggestion from D KNUTSEN(various clean up)
10-Feb-87  9:30:13 - Mita - N B = Normal Boot
12-Feb-87 16:15:27 - Mita - Catch signal File.Unknown fo r checkoutloadOK
25-Feb-87 15:36:36 - Mita - Do not crash when user select NB
 4-Sep-87 10:35:14 - Mita - AR14143 Fix.
13-Oct-87 11:59:39 - Mita - If workstation is moved to a different net while powered off, we now detect that and do a normal boot.
21-Oct-87  9:28:08 - LFB - Set physical volume boot back to User volume if we don't do the inload.
26-Jan-88 11:48:16 - Mita - AR17622: The workstation is not booted as normal by "N" and "B" keys. To fix this. I fix the 915 problem I
will do scavenge User volume. see LogicalVolumeScavenge .
```

```
-- File: QuickRestart.bootmesa - last edit:
-- Mita.ES           15-Oct-87 20:23:35
-- Copyright (C) 1987 by Xerox Corporation. All rights reserved.

-- Make all Unpackaged and Packaged code and global frames resident.
RESIDENT:
  GLOBALFRAME[ALL],
  CODE[ALL],
  FRAME[ALL],
  CODEPACK[ALL];

-- LOG [Time - Person - Action]
-- 15-Oct-87 20:23:48 - Mita - Renamed
```

--QuickBootDesignNote.txt
--Makoto Mita 5-Feb-87 16:17:18

As QuickBoot is so tricky, I would like to leave some note.

0. This application use File Type to make Outloaf file

0 page has information on Last opened date of User and scavenger volume.Also this outload file is valid or not. From 1 page Outload image will be stored. Although this is File.File this file is acctually Temporaly file. so should be deleted at boot time. I made a double check to Volume.LookupFil and Delete file which should be deleted at boot time(MP960). We need this because we used to have the problem in the scavenger volume filled up by the Extrabackingfile which should be deleted because originally created for the temporaly. This file will not be deleted even by the pilot scavenge. Ask Lee Breisacher for more details.

1. AT creating and deleting outload file, BWS set Priority 3 and fork another Process to prevent all the othe process running.

This is necessary since, While writing snapshot and change some outload file, a process may change current disk status. which will mess up outload file.

2. Wait 1 second before set to process 3.

It seems Idle control look at the WorkstationProfile and decide wheter he displays bouncing keyboard or not. which will take at least 500 seconds.(I found)

So never set this 1 second to less.

3. Kernel.makeboot is very important, Otherwise boot will end up 921.

4. We call System.PowerOff because SpecialVolume.LastOpenedForWrite is the date which volume last opened for write not the last write operations' date.

```
-- File: PowerOffRestartImpl.mesa - last edit:
-- Mita.ES                 18-Feb-88 14:43:46

-- Copyright (C) 1987, 1988 by Xerox Corporation. All rights reserved.
-- OVERVIEW: PowerOffRestartImpl which actually manage a OutLoad file in USer volume. Throught the PowerOffRestart interface you can
access anything.
--Note Outload file will be deleted automatically when the logonCompleted.
<< OVERALL DESIGN COMMENTS:  wdk
Make the swap check file and the outload file both be ***temporary** files. This has the wonderful property that, if some other agent
opens the volume for write, the files that we don't want to use anyway will be deleted automatically! Nice. [Picture of Dale patting
self on back.]

o  Simplifying idea. Make swap check file and inload file all be one file. The swap check data is page zero, the inload part starts at
page one. This means less code to manage files.

o   When creating and deleting files etc, be very careful of the order of operations so that a crash does not leave orphan files on the
disk. Put comments in the code noting the required order e.g.

    xxx;  -- 1 of 3
    xxx;  -- 2 of 3
    xxx;  -- 3 of 3

o  This program should not know the name of the "user" volume. Its main volume is Volume.systemID and the optional otherVolume parameter
I mentioned in my proposed new interface.

o   If QuickRestartClientImpl does a normal boot, it should first delete all files that this module created.

>>


-- This procesdure has the KLUDGE, find by this pattern.


DIRECTORY
  ApplicationFolder USING [FromName],
  Boot USING [LVBootFiles],
  BootFile USING [maxEntriesPerHeader, maxEntriesPerTrailer,
    MemorySizeToFileSize],
  BWSFileTypes USING [systemFileCatalog],
  Catalog USING [Open],
  Courier,
  Environment USING [PageNumber],
  File USING [
    Create, Delete, File, nullID, nullFile, PageNumber, Type,
    Unknown],
  KernelFile USING [MakeBootable],
  NSFile,
  NSSegmentInternal USING [GetID],
  NSString,
  Process USING [GetPriority, Pause, Priority, SecondsToTicks, SetPriority, Yield],
  ProcessorFace USING [PowerOff],
  ProcessorFaceExtras USING [Version],
  ProcessPriorities USING [priorityPageFaultLow],
  Snapshot USING [OutLoad],
  Space USING [CopyOut, Map, Unmap],
  SpecialSpace USING [realMemorySize],
  SpecialVolume USING [LastOpenedForWrite, GetLogicalVolumeBootFiles],
  PowerOffRestart,
  PowerOffRestartInternal,
  ProductFactoring,
  System,
  TemporaryBooting USING [
    InstallPhysicalVolumeBootFile, InvalidParameters, MakeBootable],
  VPPFOptions,
  Volume USING [ GetAttributes, ID, InsufficientSpace, InsertRootFile, LookUpRootFile, nullID, ReadOnly, RemoveRootFile,
  RootDirectoryError, systemID],
  XString;

PowerOffRestartImpl: PROGRAM
  IMPORTS
    ApplicationFolder, BootFile, Catalog, Courier, File, NSSegmentInternal, Process,
    ProcessorFace, ProcessorFaceExtras, ProductFactoring, Snapshot, Space, SpecialSpace, SpecialVolume, System, Volume, KernelFile,
    NSFile, NSString, TemporaryBooting, XString
  EXPORTS PowerOffRestart =
  BEGIN

  specialBootfirstPage: File.PageNumber ← 1; -- We use page 0 for special info.

  dlionname: NSString.String ← NSString.StringFromMesaString["QuickRestartDlion.boot"G];
  dovename: NSString.String ← NSString.StringFromMesaString["QuickRestartDove.boot"G];
  kikuname: NSString.String ← NSString.StringFromMesaString["QuickRestartKiku.boot"G];

  SpaceError: PUBLIC ERROR [requestedPage: Environment.PageNumber, availablePage: Environment.PageNumber] = CODE;
  Error: PUBLIC ERROR [type: PowerOffRestart.ErrorType] = CODE;


  outloadFile: File.File ← File.nullFile;
  bootFile: File.File;
  fileSize: LONG CARDINAL ← BootFile.MemorySizeToFileSize[
    SpecialSpace.realMemorySize];

  DeleteOutLoad: PROC =
    BEGIN
    outloadFile ← Volume.LookUpRootFile[
        PowerOffRestartInternal.outloadFileType, Volume.systemID !
        Volume.RootDirectoryError => CONTINUE];
    IF outloadFile # File.nullFile THEN {
```

```
    --need to delete file first, then remove from directory. Otherwise, crash would orphan the file. [insert comment explaining why order
    important.] wdk
    File.Delete[outloadFile ! File.Unknown => CONTINUE];
    Volume.RemoveRootFile[
        PowerOffRestartInternal.outloadFileType, Volume.systemID !
        Volume.RootDirectoryError => CONTINUE];
      };
    END;  --DeleteOutLoad

  DeleteBootFile: PROC =
    BEGIN
    --Temporaly Proc for BWS4.3g
    parentFh: NSFile.Handle ← Catalog.Open[BWSFileTypes.systemFileCatalog];
    bootFH: NSFile.Handle ← NSFile.nullHandle;
    filters: ARRAY [0..1) OF NSFile.Filter ← [
      [matches[[name[NSString.StringFromMesaString["QuickRestart*.boot"L]]]]]];
    bootFH ← NSFile.Find[
      directory: parentFh, scope: [filter: [and[DESCRIPTOR[filters]]]] !
      NSFile.Error => CONTINUE];
    IF bootFH = NSFile.nullHandle THEN RETURN;
    NSFile.Delete[bootFH!
      NSFile.Error => CONTINUE];
    END; --DeleteBootFile

  FindQuickRestartBootfile: PROC [] RETURNS [BOOLEAN ← FALSE] =
    BEGIN
    parentFh: NSFile.Handle ← NSFile.nullHandle;
    internalNameRB: XString.ReaderBody ← XString.FromSTRING["Power Off Quick Restart"L];
    applicationfolder: NSFile.Reference ← NSFile.nullReference;
    bootFH: NSFile.Handle ← NSFile.nullHandle;
    filters: ARRAY [0..1) OF NSFile.Filter ← [
      [matches[[name[HardwareDependentBootfile[]]]]]];
    applicationfolder ← ApplicationFolder.FromName[@internalNameRB];
    IF applicationfolder=NSFile.nullReference THEN RETURN;
    parentFh ← NSFile.OpenByReference[applicationfolder!NSFile.Error => CONTINUE];
    IF parentFh=NSFile.nullHandle THEN RETURN;
    bootFH ← NSFile.Find[
      directory: parentFh, scope: [filter: [and[DESCRIPTOR[filters]]]] !
      NSFile.Error => CONTINUE];
    NSFile.Close[parentFh! NSFile.Error =>CONTINUE];
    IF bootFH = NSFile.nullHandle THEN RETURN;
    [bootFile, specialBootfirstPage] ← NSSegmentInternal.GetID[bootFH];
    TemporaryBooting.MakeBootable[
      bootFile, specialBootfirstPage ! TemporaryBooting.InvalidParameters => GOTO invalid];
    RETURN[TRUE];
    EXITS invalid => RETURN;
    END;  -- FindQuickRestartBootfile

  HardwareDependentBootfile: PROCEDURE RETURNS[name: NSString.String] ={
  name ← NSString.nullString;
  name ← SELECT ProcessorFaceExtras.Version[].machineType FROM
      dandelion => dlionname,
      kiku => kikuname,
    ENDCASE => dovename;
  };

  GotResources: PUBLIC PROCEDURE[otherVolume: Volume.ID] RETURNS [BOOLEAN] =
  BEGIN
  mergin: CARDINAL = 150;
  standalone: BOOLEAN ← Standalone[];
  volume: Volume.ID ← IF otherVolume=Volume.nullID THEN Volume.systemID ELSE otherVolume;
  temp: LONG INTEGER;
  available: LONG INTEGER ← Volume.GetAttributes[volume].freePageCount;
  temp ← available - fileSize;
  IF temp <mergin THEN ERROR SpaceError[requestedPage: fileSize+mergin, availablePage: available]; --fix AR18053
  IF ~standalone THEN
    IF ~FindQuickRestartBootfile[] THEN ERROR Error[restartBootFileNotFound];
  RETURN[TRUE];
  END; -- GotResources

  DoIt: PUBLIC PROCEDURE [otherVolume: Volume.ID] =
    BEGIN
    currentPriority: Process.Priority;
    helper: PROCESS;
    helperDone: BOOLEAN ← FALSE;
    standalone: BOOLEAN ← Standalone[];
    file: File.File; firstPage: File.PageNumber;
    Helper: PROC = BEGIN UNTIL helperDone DO Process.Yield; ENDLOOP; END;
    -- Find Bootfile from Volume.systemID.
    --$$$ KLUDGE: To stop other process during Saving, I run the my process Priority=ProcessPriorities.priorityPageFaultLow and Fork
    another process which will stop every other client activity so that any other process(0,1,2) will not write to disk.

    -- wait for the black out display.(unless strage thing happens)
    Process.Pause[ticks: Process.SecondsToTicks[1]];
    currentPriority ← Process.GetPriority[];
    Process.SetPriority[ProcessPriorities.priorityPageFaultLow];
    helper ← FORK Helper[];
    IF ~MakeOutLoadFile[] THEN RETURN;
    -- page 0 will be used for Information
    [] ← KernelFile.MakeBootable[outloadFile, PowerOffRestartInternal.firstPage, fileSize];
    -- We want to get all of the work that we possbily can done BEFORE we do the OutLoad. wdk
    IF Snapshot.OutLoad[outloadFile, PowerOffRestartInternal.firstPage] THEN {
      -- just booted outload
      SetOutloaded[out: FALSE, otherVolume: otherVolume]; -- do this first thing upon return from Outload. wdk
      IF outloadFile # File.nullFile THEN {
      DeleteOutLoad[];
      -- reset physical volume boot from User volume.
```

```
      IF ~standalone THEN {
      [file, firstPage] ← GetVolumeBootFile[Volume.systemID];
      TemporaryBooting.InstallPhysicalVolumeBootFile[ file, firstPage]};
        }}
    ELSE {
      -- just wrote outload
      SetOutloaded[out: TRUE, otherVolume: otherVolume]; -- guess we've got to do this after OutLoad  wdk
      IF ~standalone THEN
        TemporaryBooting.InstallPhysicalVolumeBootFile[bootFile.specialBootfirstPage ];
      ProcessorFace.PowerOff[];
      -- (can't get here)      wdk
      };
    -- join helper process
    helperDone ← TRUE;
    JOIN helper;
    -- set priority to foreground(Note that you ned to change this if the Notifier chage Priority
    Process.SetPriority[currentPriority];
    END;  -- of SaveOutLoad

  MakeOutLoadFile: PROCEDURE RETURNS[BOOLEAN] =
    BEGIN
      outloadFile ← File.Create[
        Volume.systemID, fileSize+1, PowerOffRestartInternal.outloadFileType!Volume.InsufficientSpace, Volume.ReadOnly => GOTO error];
      Volume.InsertRootFile[
        type: PowerOffRestartInternal.outloadFileType, file: outloadFile!Volume.RootDirectoryError => CONTINUE];
    RETURN[TRUE];
    EXITS error => RETURN[FALSE];
    END;


  SetOutloaded: PROC [out: BOOLEAN, otherVolume: Volume.ID] = {
  outLoaded: PowerOffRestartInternal.CheckHandle = Space.Map[[outloadFile, 0, 1]].pointer;

    outLoaded.valid ← out;
        IF out THEN {
      IF otherVolume # Volume.nullID THEN outLoaded.lastWriteDateinScavengeVolume ← SpecialVolume.LastOpenedForWrite[
        otherVolume];
      outLoaded.lastNetworkNumber ← Courier.LocalSystemElement[].net;
      outLoaded.lastWriteDateinUserVolume ← SpecialVolume.LastOpenedForWrite[
      Volume.systemID];
      [] ← Space.CopyOut[outLoaded, [outloadFile, 0, 1]];
};
 [] ← Space.Unmap[outLoaded]
    };

  GetVolumeBootFile: PROC [lvID: Volume.ID]
    RETURNS [file: File.File, firstPage: File.PageNumber] =
    BEGIN
    bootFiles: Boot.LVBootFiles;
    SpecialVolume.GetLogicalVolumeBootFiles[lvID, @bootFiles];
    IF bootFiles[pilot].fID.fileID = File.nullID THEN RETURN[File.nullFile, 0]
    ELSE {
      file ← [bootFiles[pilot].fID.fileID, lvID];
      RETURN[file, bootFiles[pilot].firstPage]};
    END;

  Standalone: PROC RETURNS[BOOLEAN] ={
  saEnabled: BOOLEAN ← ProductFactoring.Enabled [VPPFOptions.vpStandalone];
    rEnabled: BOOLEAN ← ProductFactoring.Enabled [VPPFOptions.vpRemoteCom];
    nEnabled: BOOLEAN ← ProductFactoring.Enabled [VPPFOptions.vpNetCom];
  IF nEnabled THEN RETURN[FALSE]; -- network is strong.
  IF saEnabled OR rEnabled OR
        ((nEnabled AND rEnabled AND saEnabled) AND System.switches ['U] = down) OR
        ((~nEnabled AND ~rEnabled AND ~saEnabled) AND System.switches ['U] = down) THEN RETURN[TRUE] ELSE RETURN[FALSE]}; -- Standalone

  -- Make sure Outload file deleted at boot time

  DeleteOutLoad[];
  --Temporaly Proc for BWS4.3g should be deleted for BWS4.3h
  DeleteBootFile[];

  END.

 9-Feb-87 19:21:00   Mita       Created.
13-Oct-87 11:55:19   Mita       Fix workstation fook to different network move problem.
13-Oct-87 11:55:19   Mita       Support Application Folder Power Off Quick Restart
11-Nov-87 11:03:54   Mita       Support Device dependent check.
18-Feb-88 14:42:04   Mita       18053: To minimiza user confusion
```

```
-- File: PowerOffRestartInternal.mesa - last edit:
-- Mita.es                17-Feb-87 17:08:58

-- Copyright (C) 1987 by Xerox Corporation. All rights reserved.
-- OVERVIEW:

-- This is containd internal definition and File type which is used for Power off Quick Restart capability


DIRECTORY
  File USING [PageNumber, Type],
  System USING [GreenwichMeanTime, NetworkNumber];

PowerOffRestartInternal: DEFINITIONS =
  BEGIN

  CheckHandle: TYPE = LONG POINTER TO CheckRec;
  CheckRec: TYPE = MACHINE DEPENDENT RECORD [
    valid(0): BOOLEAN,
    lastWriteDateinUserVolume(1): System.GreenwichMeanTime,
    lastWriteDateinScavengeVolume(3): System.GreenwichMeanTime,
    lastNetworkNumber(5): System.NetworkNumber];

  -- IN FileTypes.VersatecFileType
  -- I use this to make compatible with Versatec Workstation

  outloadFileType: File.Type = [10078];

  firstPage: File.PageNumber = 1; -- We use page 0 for special info.


  END.

17-Feb-87 17:08:53   Mita        Created.
13-Oct-87 11:31:48   Mita        Add lastNetworkNumber to fix change Workstation to different net .
```

```
-- File: PowerOffRestartImpl.mesa - last edit:
-- Mita.ES                18-Feb-88 14:43:46

-- Copyright (C) 1987, 1988 by Xerox Corporation. All rights reserved.
-- OVERVIEW: PowerOffRestartImpl which actually manage a OutLoad file in USer volume. Throught the PowerOffRestart interface you can
access anything.
--Note Outload file will be deleted automatically when the logonCompleted.
<< OVERALL DESIGN COMMENTS:  wdk
Make the swap check file and the outload file both be ***temporary** files. This has the wonderful property that, if some other agent
opens the volume for write, the files that we don't want to use anyway will be deleted automatically! Nice. [Picture of Dale patting
self on back.]

o  Simplifying idea. Make swap check file and inload file all be one file. The swap check data is page zero, the inload part starts at
page one. This means less code to manage files.

o   When creating and deleting files etc, be very careful of the order of operations so that a crash does not leave orphan files on the
disk. Put comments in the code noting the required order e.g.

    xxx;  -- 1 of 3
    xxx;  -- 2 of 3
    xxx;  -- 3 of 3

o  This program should not know the name of the "user" volume. Its main volume is Volume.systemID and the optional otherVolume parameter
I mentioned in my proposed new interface.

o   If QuickRestartClientImpl does a normal boot, it should first delete all files that this module created.

>>


-- This procesdure has the KLUDGE, find by this pattern.


DIRECTORY
  ApplicationFolder USING [FromName],
  Boot USING [LVBootFiles],
  BootFile USING [maxEntriesPerHeader, maxEntriesPerTrailer,
    MemorySizeToFileSize],
  BWSFileTypes USING [systemFileCatalog],
  Catalog USING [Open],
  Courier,
  Environment USING [PageNumber],
  File USING [
    Create, Delete, File, nullID, nullFile, PageNumber, Type,
    Unknown],
  KernelFile USING [MakeBootable],
  NSFile,
  NSSegmentInternal USING [GetID],
  NSString,
  Process USING [GetPriority, Pause, Priority, SecondsToTicks, SetPriority, Yield],
  ProcessorFace USING [PowerOff],
  ProcessorFaceExtras USING [Version],
  ProcessPriorities USING [priorityPageFaultLow],
  Snapshot USING [OutLoad],
  Space USING [CopyOut, Map, Unmap],
  SpecialSpace USING [realMemorySize],
  SpecialVolume USING [LastOpenedForWrite, GetLogicalVolumeBootFiles],
  PowerOffRestart,
  PowerOffRestartInternal,
  ProductFactoring,
  System,
  TemporaryBooting USING [
    InstallPhysicalVolumeBootFile, InvalidParameters, MakeBootable],
  VPPFOptions,
  Volume USING [ GetAttributes, ID, InsufficientSpace, InsertRootFile, LookUpRootFile, nullID, ReadOnly, RemoveRootFile,
  RootDirectoryError, systemID],
  XString;

PowerOffRestartImpl: PROGRAM
  IMPORTS
    ApplicationFolder, BootFile, Catalog, Courier, File, NSSegmentInternal, Process,
    ProcessorFace, ProcessorFaceExtras, ProductFactoring, Snapshot, Space, SpecialSpace, SpecialVolume, System, Volume, KernelFile,
    NSFile, NSString, TemporaryBooting, XString
  EXPORTS PowerOffRestart =
  BEGIN

  specialBootfirstPage: File.PageNumber ← 1; -- We use page 0 for special info.

  dlionname: NSString.String ← NSString.StringFromMesaString["QuickRestartDlion.boot"G];
  dovename: NSString.String ← NSString.StringFromMesaString["QuickRestartDove.boot"G];
  kikuname: NSString.String ← NSString.StringFromMesaString["QuickRestartKiku.boot"G];

  SpaceError: PUBLIC ERROR [requestedPage: Environment.PageNumber, availablePage: Environment.PageNumber] = CODE;
  Error: PUBLIC ERROR [type: PowerOffRestart.ErrorType] = CODE;


  outloadFile: File.File ← File.nullFile;
  bootFile: File.File;
  fileSize: LONG CARDINAL ← BootFile.MemorySizeToFileSize[
    SpecialSpace.realMemorySize];

  DeleteOutLoad: PROC =
    BEGIN
    outloadFile ← Volume.LookUpRootFile[
        PowerOffRestartInternal.outloadFileType, Volume.systemID !
        Volume.RootDirectoryError => CONTINUE];
    IF outloadFile # File.nullFile THEN {

PowerOffRestartImpl.mesa        18 Feb 88 14:43:48 PST                                                                          1
```

```
    --need to delete file first, then remove from directory. Otherwise, crash would orphan the file. [insert comment explaining why order
    important.] wdk
    File.Delete[outloadFile ! File.Unknown => CONTINUE];
    Volume.RemoveRootFile[
        PowerOffRestartInternal.outloadFileType, Volume.systemID !
        Volume.RootDirectoryError => CONTINUE];
      };
    END;  --DeleteOutLoad

  DeleteBootFile: PROC =
    BEGIN
    --Temporaly Proc for BWS4.3g
    parentFh: NSFile.Handle ← Catalog.Open[BWSFileTypes.systemFileCatalog];
    bootFH: NSFile.Handle ← NSFile.nullHandle;
    filters: ARRAY [0..1) OF NSFile.Filter ← [
      [matches[[name[NSString.StringFromMesaString["QuickRestart*.boot"L]]]]]];
    bootFH ← NSFile.Find[
      directory: parentFh, scope: [filter: [and[DESCRIPTOR[filters]]]] !
      NSFile.Error => CONTINUE];
    IF bootFH = NSFile.nullHandle THEN RETURN;
    NSFile.Delete[bootFH!
      NSFile.Error => CONTINUE];
    END; --DeleteBootFile

  FindQuickRestartBootfile: PROC [] RETURNS [BOOLEAN ← FALSE] =
    BEGIN
    parentFh: NSFile.Handle ← NSFile.nullHandle;
    internalNameRB: XString.ReaderBody ← XString.FromSTRING["Power Off Quick Restart"L];
    applicationfolder: NSFile.Reference ← NSFile.nullReference;
    bootFH: NSFile.Handle ← NSFile.nullHandle;
    filters: ARRAY [0..1) OF NSFile.Filter ← [
      [matches[[name[HardwareDependentBootfile[]]]]]];
    applicationfolder ← ApplicationFolder.FromName[@internalNameRB];
    IF applicationfolder=NSFile.nullReference THEN RETURN;
    parentFh ← NSFile.OpenByReference[applicationfolder!NSFile.Error => CONTINUE];
    IF parentFh=NSFile.nullHandle THEN RETURN;
    bootFH ← NSFile.Find[
      directory: parentFh, scope: [filter: [and[DESCRIPTOR[filters]]]] !
      NSFile.Error => CONTINUE];
    NSFile.Close[parentFh! NSFile.Error =>CONTINUE];
    IF bootFH = NSFile.nullHandle THEN RETURN;
    [bootFile, specialBootfirstPage] ← NSSegmentInternal.GetID[bootFH];
    TemporaryBooting.MakeBootable[
      bootFile, specialBootfirstPage ! TemporaryBooting.InvalidParameters => GOTO invalid];
    RETURN[TRUE];
    EXITS invalid => RETURN;
    END;  -- FindQuickRestartBootfile

  HardwareDependentBootfile: PROCEDURE RETURNS[name: NSString.String] ={
  name ← NSString.nullString;
  name ← SELECT ProcessorFaceExtras.Version[].machineType FROM
      dandelion => dlionname,
      kiku => kikuname,
    ENDCASE => dovename;
  };

  GotResources: PUBLIC PROCEDURE[otherVolume: Volume.ID] RETURNS [BOOLEAN] =
  BEGIN
  mergin: CARDINAL = 150;
  standalone: BOOLEAN ← Standalone[];
  volume: Volume.ID ← IF otherVolume=Volume.nullID THEN Volume.systemID ELSE otherVolume;
  temp: LONG INTEGER;
  available: LONG INTEGER ← Volume.GetAttributes[volume].freePageCount;
  temp ← available - fileSize;
  IF temp <mergin THEN ERROR SpaceError[requestedPage: fileSize+mergin, availablePage: available]; --fix AR18053
  IF ~standalone THEN
    IF ~FindQuickRestartBootfile[] THEN ERROR Error[restartBootFileNotFound];
  RETURN[TRUE];
  END; -- GotResources

  DoIt: PUBLIC PROCEDURE [otherVolume: Volume.ID] =
    BEGIN
    currentPriority: Process.Priority;
    helper: PROCESS;
    helperDone: BOOLEAN ← FALSE;
    standalone: BOOLEAN ← Standalone[];
    file: File.File; firstPage: File.PageNumber;
    Helper: PROC = BEGIN UNTIL helperDone DO Process.Yield; ENDLOOP; END;
    -- Find Bootfile from Volume.systemID.
    --$$$ KLUDGE: To stop other process during Saving, I run the my process Priority=ProcessPriorities.priorityPageFaultLow and Fork
    another process which will stop every other client activity so that any other process(0,1,2) will not write to disk.

    -- wait for the black out display.(unless strage thing happens)
    Process.Pause[ticks: Process.SecondsToTicks[1]];
    currentPriority ← Process.GetPriority[];
    Process.SetPriority[ProcessPriorities.priorityPageFaultLow];
    helper ← FORK Helper[];
    IF ~MakeOutLoadFile[] THEN RETURN;
    -- page 0 will be used for Information
    [] ← KernelFile.MakeBootable[outloadFile, PowerOffRestartInternal.firstPage, fileSize];
    -- We want to get all of the work that we possbily can done BEFORE we do the OutLoad. wdk
    IF Snapshot.OutLoad[outloadFile, PowerOffRestartInternal.firstPage] THEN {
      -- just booted outload
      SetOutloaded[out: FALSE, otherVolume: otherVolume]; -- do this first thing upon return from Outload. wdk
      IF outloadFile # File.nullFile THEN {
      DeleteOutLoad[];
      -- reset physical volume boot from User volume.
```

```
      IF ~standalone THEN {
      [file, firstPage] ← GetVolumeBootFile[Volume.systemID];
      TemporaryBooting.InstallPhysicalVolumeBootFile[ file, firstPage]};
        }}
    ELSE {
      -- just wrote outload
      SetOutloaded[out: TRUE, otherVolume: otherVolume]; -- guess we've got to do this after OutLoad  wdk
      IF ~standalone THEN
        TemporaryBooting.InstallPhysicalVolumeBootFile[bootFile,specialBootfirstPage ];
      ProcessorFace.PowerOff[];
      -- (can't get here)      wdk
      };
    -- join helper process
    helperDone ← TRUE;
    JOIN helper;
    -- set priority to foreground(Note that you ned to change this if the Notifier chage Priority
    Process.SetPriority[currentPriority];
    END;  -- of SaveOutLoad

  MakeOutLoadFile: PROCEDURE RETURNS[BOOLEAN] =
    BEGIN
      outloadFile ← File.Create[
        Volume.systemID, fileSize+1, PowerOffRestartInternal.outloadFileType!Volume.InsufficientSpace, Volume.ReadOnly => GOTO error];
      Volume.InsertRootFile[
        type: PowerOffRestartInternal.outloadFileType, file: outloadFile!Volume.RootDirectoryError => CONTINUE];
    RETURN[TRUE];
    EXITS error => RETURN[FALSE];
    END;


  SetOutloaded: PROC [out: BOOLEAN, otherVolume: Volume.ID] = {
  outLoaded: PowerOffRestartInternal.CheckHandle = Space.Map[[outloadFile, 0, 1]].pointer;

    outLoaded.valid ← out;
        IF out THEN {
      IF otherVolume # Volume.nullID THEN outLoaded.lastWriteDateinScavengeVolume ← SpecialVolume.LastOpenedForWrite[
        otherVolume];
      outLoaded.lastNetworkNumber ← Courier.LocalSystemElement[].net;
      outLoaded.lastWriteDateinUserVolume ← SpecialVolume.LastOpenedforWrite[
      Volume.systemID];
      [] ← Space.CopyOut[outLoaded, [outloadFile, 0, 1]];
};
 [] ← Space.Unmap[outLoaded]
    };

  GetVolumeBootFile: PROC [lvID: Volume.ID]
    RETURNS [file: File.File, firstPage: File.PageNumber] =
    BEGIN
    bootFiles: Boot.LVBootFiles;
    SpecialVolume.GetLogicalVolumeBootFiles[lvID, @bootFiles];
    IF bootFiles[pilot].fID.fileID = File.nullID THEN RETURN[File.nullFile, 0]
    ELSE {
      file ← [bootFiles[pilot].fID.fileID, lvID];
      RETURN[file, bootFiles[pilot].firstPage]};
    END;

  Standalone: PROC RETURNS[BOOLEAN] ={
  saEnabled: BOOLEAN ← ProductFactoring.Enabled [VPPFOptions.vpStandalone];
    rEnabled: BOOLEAN ← ProductFactoring.Enabled [VPPFOptions.vpRemoteCom];
    nEnabled: BOOLEAN ← ProductFactoring.Enabled [VPPFOptions.vpNetCom];
  IF nEnabled THEN RETURN[FALSE]; -- network is strong.
  IF saEnabled OR rEnabled OR
        ((nEnabled AND rEnabled AND saEnabled) AND System.switches ['U] = down) OR
        ((~nEnabled AND ~rEnabled AND ~saEnabled) AND System.switches ['U] = down) THEN RETURN[TRUE] ELSE RETURN[FALSE]}; -- Standalone

  -- Make sure Outload file deleted at boot time

  DeleteOutLoad[];
  --Temporaly Proc for BWS4.3g should be deleted for BWS4.3h
  DeleteBootFile[];

  END.

 9-Feb-87 19:21:00   Mita       Created.
13-Oct-87 11:55:19   Mita       Fix workstation fook to different network move problem.
13-Oct-87 11:55:19   Mita       Support Application Folder Power Off Quick Restart
11-Nov-87 11:03:54   Mita       Support Device dependent check.
18-Feb-88 14:42:04   Mita       18053: To minimiza user confusion
```

```
-- File: PowerOffRestartInternal.mesa - last edit:
-- Mita.es                17-Feb-87 17:08:58

-- Copyright (C) 1987 by Xerox Corporation. All rights reserved.
-- OVERVIEW:

-- This is containd internal definition and File type which is used for Power off Quick Restart capability


DIRECTORY
  File USING [PageNumber, Type],
  System USING [GreenwichMeanTime, NetworkNumber];

PowerOffRestartInternal: DEFINITIONS =
  BEGIN

  CheckHandle: TYPE = LONG POINTER TO CheckRec;
  CheckRec: TYPE = MACHINE DEPENDENT RECORD [
    valid(0): BOOLEAN,
    lastWriteDateinUserVolume(1): System.GreenwichMeanTime,
    lastWriteDateinScavengeVolume(3): System.GreenwichMeanTime,
    lastNetworkNumber(5): System.NetworkNumber];

  -- IN FileTypes.VersatecFileType
  -- I use this to make compatible with Versatec Workstation

  outloadFileType: File.Type = [10078];

  firstPage: File.PageNumber = 1; -- We use page 0 for special info.


  END.

17-Feb-87 17:08:53   Mita        Created.
13-Oct-87 11:31:48   Mita        Add lastNetworkNumber to fix change Workstation to different net .
```

```
-- File: QuickRestartClientImpl.mesa - last edit:
-- Mita                 27-Jan-88 14:15:09
-- Breisacher           21-Oct-87  9:40:58


-- Copyright (C) 1987, 1988 by Xerox Corporation. All rights reserved.

DIRECTORY
  Boot USING [LVBootFiles],
  File USING [Delete, File, nullFile, Unknown],
  LevelIVKeys USING [KeyBits, KeyName],
  NSConstants,
  Space USING [CopyIn, ScratchMap, Unmap],
  String USING [Equivalent],
  SpecialVolume USING [GetLogicalVolumeBootFiles, LastOpenedForWrite, OpenVolume],
  OthelloOps USING [GetTimeFromTimeServer, SetProcessorTime, TimeServerError],
  PhysicalVolume USING [
    AssertPilotVolume, Error, GetHandle, GetNextDrive, Handle, ID,
    NeedsScavenging, nullDeviceIndex, nullID],
  PilotMP USING [cScavenging, cDiskHardwareError, cTimeNotAvailable, cClient],
  ProcessorFace USING [SetMP],
  PowerOffRestartInternal USING [CheckHandle, outloadFileType, firstPage],
  Router USING [FindDestinationRelativeNetID],
  Scavenger USING [Scavenge],
  System USING [GreenwichMeanTime, LocalTimeParameters, NetworkNumber, nullNetworkNumber, SetLocalTimeParameters],
  TemporaryBooting USING [BootFromFile, BootFromVolume, InstallPhysicalVolumeBootFile, InvalidParameters],
  UserTerminal USING [WaitForScanLine, keyboard],
  Volume USING [GetNext, GetLabelString, ID, maxNameLength, NeedsScavenging, nullID,
    LookUpRootFile, Open, RootDirectoryError, Unknown];

QuickRestartClientImpl: PROGRAM
  IMPORTS
    File, OthelloOps, PhysicalVolume, ProcessorFace, Router, Scavenger, Space, SpecialVolume, String,
    System, TemporaryBooting, UserTerminal, Volume =
  BEGIN

 userVolume: Volume.ID ← Volume.nullID;

 CheckOutloadOK: PROC [volumeName: LONG STRING]
    RETURNS [inload: File.File, proceed: BOOLEAN] =
    BEGIN
    outloadFile: File.File ← File.nullFile;
    userVolume ← GetVolumeID[volumeName];
    proceed ← FALSE;
    IF userVolume # Volume.nullID THEN {
      SpecialVolume.OpenVolume[userVolume, read!Volume.NeedsScavenging=>  {LogicalVolumeScavenge[userVolume]; CONTINUE}]; -- delete
      because this is permanent.
      outloadFile ← Volume.LookUpRootFile[
        PowerOffRestartInternal.outloadFileType, userVolume!Volume.RootDirectoryError => GOTO Errors];
      IF outloadFile # File.nullFile THEN {
        check: PowerOffRestartInternal.CheckHandle = Space.ScratchMap[1];
        proceed ← CheckOutloaded[outloadFile, check!File.Unknown=> GOTO Errors];
        IF proceed THEN proceed ←  (check.lastNetworkNumber = System.nullNetworkNumber) OR (LocalNet[] = check.lastNetworkNumber);
        inload ← outloadFile;
        [] ← Space.Unmap[check]};
      };
    EXITS Errors => {proceed ← FALSE; RETURN};
    END;

  LogicalVolumeScavenge: PROC [volume: Volume.ID] =
    BEGIN
    scavengerlog: File.File;
    ProcessorFace.SetMP[PilotMP.cScavenging];
    scavengerlog ← Scavenger.Scavenge[volume, volume, safeRepair, FALSE];
    Volume.Open[volume];
    File.Delete[scavengerlog];
    ProcessorFace.SetMP[PilotMP.cClient];
    END;


  CheckOutloaded: PROC [file: File.File, check: PowerOffRestartInternal.CheckHandle]
    RETURNS [out: BOOLEAN] = {
    scavengerVolume: Volume.ID ← GetVolumeID["Scavenger"L];
    [] ← Space.CopyIn[check, [file, 0, 1]];
    out ← check.valid AND
        (check.lastWriteDateinUserVolume = SpecialVolume.LastOpenedForWrite[
          userVolume])
      AND
        (check.lastWriteDateinScavengeVolume = SpecialVolume.LastOpenedForWrite[
          scavengerVolume])};

  GetVolumeID: PROC [svVolume: LONG STRING] RETURNS [vl: Volume.ID] =
    BEGIN
    svLabel: STRING = [Volume.maxNameLength];
    vl ← Volume.nullID;
    UNTIL (vl ← Volume.GetNext[vl]) = Volume.nullID DO
      Volume.GetLabelString[vl, svLabel];
      IF String.Equivalent[svVolume, svLabel] THEN EXIT;
      ENDLOOP;
    END;  -- of GetVolumeID

  LocalNet: PROC RETURNS [System.NetworkNumber] = {
    RETURN[Router.FindDestinationRelativeNetID[System.nullNetworkNumber]];
    };

  Run: PROCEDURE =
    BEGIN

QuickRestartClientImpl.mesa        27-Jan-88 14:15:10 PST                                                    1
```

```
    userVolName: LONG STRING ← "User"L;
    timeFromServer: BOOLEAN ← TRUE;
    doInload: BOOLEAN ← TRUE;
    time: System.GreenwichMeanTime;
    ltps: System.LocalTimeParameters;
    inloadFile: File.File;
    ProcessorFace.SetMP[PilotMP.cTimeNotAvailable];

    [time, ltps] ← OthelloOps.GetTimeFromTimeServer[
      ! OthelloOps.TimeServerError => RETRY];
    IF timeFromServer THEN {
      System.SetLocalTimeParameters[ltps]; OthelloOps.SetProcessorTime[time]};
    -- If there is no Time server boot from scratch.
    doInload ← timeFromServer;
    IF doInload THEN doInload ← BringFirstReadyPhysicalVolumeOnline[];
    IF ~doInload THEN  -- Can not Online Volume
      {ProcessorFace.SetMP[PilotMP.cDiskHardwareError]; RETURN}
    ELSE {FOR i:CARDINAL IN[0..5) DO UserTerminal.WaitForScanLine[0];
    doInload ← ~KeysAreDown[key1: N, key2: B];
    IF ~doInload THEN EXIT;
    ENDLOOP;
    };
    IF doInload THEN [inloadFile, doInload] ← CheckOutloadOK[userVolName];
    IF doInload THEN [] ← TemporaryBooting.BootFromFile[
      inloadFile, PowerOffRestartInternal.firstPage!TemporaryBooting.InvalidParameters, Volume.Unknown, File.Unknown,
      Volume.NeedsScavenging => {doInload ← FALSE; CONTINUE}];
    -- "only returns if couldn't inload" WDK
    IF userVolume=Volume.nullID THEN userVolume ← GetVolumeID["User"L];
    SetPhysicalBoot[userVolume];
    TemporaryBooting.BootFromVolume[userVolume]
    END;  -- Run

  KeysAreDown: PROCEDURE [key1, key2: LevelIVKeys.KeyName] RETURNS [BOOLEAN] = {
    keys: LONG POINTER TO LevelIVKeys.KeyBits = LOOPHOLE[UserTerminal.keyboard];
    RETURN[(keys[key1] = down) AND (keys[key2] = down)];
    };

  BringFirstReadyPhysicalVolumeOnline: PROC RETURNS [BOOLEAN] =
    BEGIN
    driveIndex: CARDINAL ← PhysicalVolume.nullDeviceIndex;
    driveHandle: PhysicalVolume.Handle;
    pvID: PhysicalVolume.ID;

    DO  -- loop looking for drives connected to this machine
      pvID ← PhysicalVolume.nullID;
      driveIndex ← PhysicalVolume.GetNextDrive[driveIndex];
      IF driveIndex = PhysicalVolume.nullDeviceIndex THEN EXIT;
      driveHandle ← PhysicalVolume.GetHandle[driveIndex];
      IF TRUE <<PhysicalVolume.IsReady[driveHandle]>> THEN
        BEGIN
        <<IsReady doesn't work properly. Thus the following code is necessary.>>
        pvID ← PhysicalVolume.AssertPilotVolume[
          driveHandle !
          PhysicalVolume.Error, PhysicalVolume.NeedsScavenging => CONTINUE];
        IF pvID = PhysicalVolume.nullID THEN EXIT;
        RETURN[TRUE];
        END;
      ENDLOOP;  -- end of loop until a drive is selected
    RETURN[FALSE];
    END;  -- of BringFirstReadyPhysicalVolumeOnline

  SetPhysicalBoot: PROCEDURE [lvID: Volume.ID] = {
    bootFiles: Boot.LVBootFiles;
    SpecialVolume.GetLogicalVolumeBootFiles[lvID, @bootFiles];
    Volume.Open[lvID!Volume.NeedsScavenging=> {LogicalVolumeScavenge[lvID]; CONTINUE;}];
    TemporaryBooting.InstallPhysicalVolumeBootFile [
      file: [bootFiles[pilot].fID.fileID, lvID],
      firstPage: bootFiles[pilot].firstPage];
    };

  --Main
  Run[];

  END.
LOG
22-Jan-87 10:47:55 - Mita - Create from QuickBoot hack.
26-Jan-87 12:36:40 - Mita - Support Time service
 4-Feb-87 19:13:30 - Mita - Incorporate suggestion from D KNUTSEN(various clean up)
10-Feb-87  9:30:13 - Mita - N B = Normal Boot
12-Feb-87 16:15:27 - Mita - Catch signal File.Unknown fo r checkoutloadOK
25-Feb-87 15:36:36 - Mita - Do not crash when user select NB
 4-Sep-87 10:35:14 - Mita - AR14143 Fix.
13-Oct-87 11:59:39 - Mita - If workstation is moved to a different net while powered off, we now detect that and do a normal boot.
21-Oct-87  9:28:08 - LFB - Set physical volume boot back to User volume if we don't do the inload.
26-Jan-88 11:48:16 - Mita - AR17622: The workstation is not booted as normal by "N" and "B" keys. To fix this. I fix the 915 problem I
will do scavenge User volume. see LogicalVolumeScavenge .
```

```
-- Copyright (C) 1983, 1987  by Xerox Corporation. All rights reserved.
-- Boot.mesa          8-Oct-87 12:22:07 by CAJ

-- This module defines TYPEs and constants used for invoking the Germ and passing parameters to it.  It also has constants used for
building Germs.

-- Since the initial microcode is the primeval invoker of the germ, many of the items defined here are known to the initial microcode;
thus changes to this interfaces will require corresponding changes to the initial microcode.

DIRECTORY
  BootFile USING [InLoadMode, MDSIndex],
  Device USING [Type],
  Environment USING [Long, PageCount, PageNumber],
  HostNumbers USING [HostNumber, nullHostNumber],
  PilotDisk USING [FileID, FilePageNumber, nullFileID],
  PilotDiskFace USING [DiskAddress],
  PrincOps USING [ControlLink],
  SDDefs USING [sBoot, SD, sFirstGermRequest, sGermCount ---,sLastPilot -],
  StartList USING [Base],
  System USING [defaultSwitches, NetworkAddress, Switches],
  Volume USING [Type];

Boot: DEFINITIONS =
  BEGIN

  -- SOME ITEMS DEFINED HERE ARE KNOWN TO THE INITIAL MICROCODE.  CHANGING THEM WILL REQUIRE CORRESPONDING CHANGES TO IT.

  --~~~~~~~~~~ Attributes of the Germ: ~~~~~~~~~~

  mdsiGerm: BootFile.MDSIndex = 0;
  -- the MDS of the Germ, as defned in the PrincOps.  Known to the initial microcode.

  pageGerm: Environment.PageNumber = 1;
  --  Page where Germ's image starts (within Germ's MDS).  Known to the initial microcode.

  countGermVM: Environment.PageCount = 96 - pageGerm;
  -- Amount of virtual memory reserved for Germ and its buffers.


  --~~~~~~~~~~ Arguments for the Germ, and returned results: ~~~~~~~~~~

  -- Version numbers for the format of a Request:

  currentRequestBasicVersion: CARDINAL = 3456B;

  currentRequestExtensionVersion: CARDINAL = 7654B;

  Request: TYPE = MACHINE DEPENDENT RECORD [

    -- Basic portion of Request: (format known by the initial microcode)
    -- IF YOU CHANGE THE FORMAT OF THIS PORTION, YOU MUST INCREMENT currentRequestBasicVersion AND GENERATE NEW INITIAL MICROCODE!
    requestBasicVersion (0B): CARDINAL ← currentRequestBasicVersion,
    action (1B): Action,
    location (2B): Location,
    switches (16B): System.Switches ← System.defaultSwitches,  -- When calling OutLoad, these are the default switches to be used for
    system when InLoaded later.  When calling InLoad, these are switches to be used for system being InLoaded; defaultSwitches means use
    the ones built into the boot file.  When returning from InLoad, these are the switches passed from caller of InLoad or, if he passed
    defaultSwitches, the switches built into the boot file.

    -- Extension portion of Request: (not used by the initial microcode)
    requestExtensionVersion (15B): CARDINAL ← currentRequestExtensionVersion,

     -- Extensions for InLoad - "results":
    pStartListHeader  (36B): StartList.Base,  -  only valid after inload of virgin boot file.

    -- Extensions for OutLoad - "arguments":
    inLoadMode (40B): BootFile.InLoadMode,  -- real page numbers significant?
    -- As a side-effect of the cross-mds call mechanism, the entry point of the system being outLoaded is stored in pInitialLink↑ in the
    caller's mds and the mds of the system being outLoaded is passed to the Germ.

    -- Extensions for OutLoad - "results":
    session (41B): Session ];  -- just finished OutLoad, or InLoaded later?


  Action: TYPE = MACHINE DEPENDENT RECORD [act(0): CARDINAL];

  inLoad: Action = [0];
  -- restore volatile processor state from BootFile-format snapshot.

  outLoad: Action = [1];
  -- save volatile processor state in BootFile-format snapshot.

  bootPhysicalVolume: Action = [2];
  -- do inLoad using Location specified indirectly in pilot entry of PVBootFiles array of physical volume root page of disk specified by
  accompanying Location.  Value known by the initial microcode.

  teledebug: Action = [3];
  -- speak to Ethernet as Teledebug Server.  Return when commanded to.

  noOp: Action = [4];  -- simply enter and exit the Germ.

  Session: TYPE = {continuingAfterOutLoad, newSession};


  Location: TYPE = MACHINE DEPENDENT RECORD [  -- format known by the initial microcode.
    -- Description of boot file location:
    deviceType (0): Device.Type,  -- e.g. sa4000, ethernet

Boot.mesa          8-Oct-87 12:22:24 PDT                                                                                        1
```

```
    deviceOrdinal (1): CARDINAL,  -- position of device within all those of same type
    vp (2): SELECT OVERLAID * FROM
      disk => [diskFileID (2): DiskFileID],
      ethernetOne => [bootFileNumber (2): CARDINAL, net (3), host (4): CARDINAL ← 0],
      ethernet => [ethernetRequest (2): EthernetRequest],
      any => [a (2), b (3), c (4), d (5), e (6), f (7), g (10B), h (11B): UNSPECIFIED],
      ENDCASE];


  -- Disk Location Data:

  DiskFileID: TYPE = MACHINE DEPENDENT RECORD [
    fID (0): PilotDisk.FileID,
    firstPage (5): PilotDisk.FilePageNumber,
    da (7): PilotDiskFace.DiskAddress];

  -- Convention: a DiskFileID is null if its fID is null:

  NullDiskFileID: PROCEDURE [diskFileID: DiskFileID] RETURNS [BOOLEAN] = INLINE
    { RETURN[diskFileID.fID = PilotDisk.nullFileID] };

  bootPhysicalVolumeDiskAddress: PilotDiskFace.DiskAddress = LOOPHOLE[LONG[0]];
  -- (It would be cleaner if this was defined using PilotDiskFace.)

  DiskBootChainLink: TYPE = PilotDiskFace.DiskAddress;

  nullDiskBootChainLink: DiskBootChainLink = LOOPHOLE[LONG[0]];
  -- This is written in the boot chain link field of all
  --   but the last page of each disk run of the bootable portion of a file.
  -- A valid disk address is written in the last page of each interior disk run
  --   of the bootable portion of a file.

  eofDiskBootChainLink: DiskBootChainLink = LOOPHOLE[LAST[LONG CARDINAL]];
  -- "End of file". This must be written in the boot chain link field of the last
  --   page of the bootable portion of a file which is loaded by the microcode
  --   (hard microcode, soft microcode, diagnostic microcode, germ).
  --   It is required to delimit the end of the file to the microcode.

  DiskAddress: TYPE = PilotDiskFace.DiskAddress;  -- for compatibility.


  -- Ethernet Location Data:

  EthernetRequest: TYPE = MACHINE DEPENDENT RECORD [
    bfn(0): EthernetBootFileNumber, address(3): System.NetworkAddress];

  EthernetBootFileNumber: TYPE = RECORD [HostNumbers.HostNumber];
  -- Ethernet boot file numbers are allocated from the same name space as
  -- HostNumbers. Of course, these numbers do not represent hosts.

  nullEthernetBootFileNumber: EthernetBootFileNumber = [HostNumbers.nullHostNumber];


--~~~~~~~~~~ Booting information for the Germ and microcode: ~~~~~~~~~~

  -- Types of boot files pointed to from root pages of physical and logical
  -- volumes, and delivered by boot servers:
  -- The following cannot be changed without invalidating all Pilot volumes.
  -- Known to the initial microcode.
  BootFileType: TYPE = MACHINE DEPENDENT{
    hardMicrocode (0), softMicrocode (1), germ (2), pilot (3), debugger (4),
    debuggee (5)};

  PVBootFiles: TYPE = ARRAY BootFileType [hardMicrocode..pilot] OF DiskFileID;
  -- format known to the initial microcode.

  LVBootFiles: TYPE = ARRAY BootFileType OF DiskFileID;

  VolumeType: TYPE = Volume.Type;  -- for compatibility.


--~~~~~~~~~~ Reserved Memory Locations for the Germ and Pilot: ~~~~~~~~~~

  pInitialLink: indirect PrincOps.ControlLink =
    -- The entry point to the system (the Germ or Pilot) which is rooted
    -- in the MDS containing pInitialLink.  Is automatically set as a
    -- side-effect of the cross-mds linkage mechanism.
    -- pInitialLink↑ itself is a control link within the mds containing it.
    -- NOTE: The value of pInitialLink is defined by the PrincOps.
    [indirect[link[link: LOOPHOLE[LOOPHOLE[SDDefs.SD, CARDINAL] +
      SDDefs.sBoot * SIZE[LONG UNSPECIFIED]], fill: 0]]];

    -- Previous replaces following until compiler, broken in 11.1, is again
    -- able to handle expressions of "@externalConstantAddress[constantOffset]".
    --[indirect[link[link: LOOPHOLE[@SDDefs.SD[SDDefs.sBoot]], fill: 0]]];

  << TEMP omitted until compiler can evaluate constant pointer expressions
     at compile time.
  initialLinkAlignedRight: PRIVATE BOOLEAN[TRUE..TRUE] =
    (LOOPHOLE[pInitialLink, PrincOps.ControlLink].indirect AND
    NOT LOOPHOLE[pInitialLink, PrincOps.ControlLink].proc);>>

  pCountGerm: LONG POINTER TO CARDINAL =
    -- The number of pages occupied by the germ.  Set by MakeBoot.
    -- This data is allocated in the Germ's MDS.
    -- Excludes dynamically allocated pages (buffers, etc).
    -- Location known to initial microcode.
```

```
    LOOPHOLE[ Environment.Long[ any[
      low: LOOPHOLE[SDDefs.SD, CARDINAL]+SDDefs.sGermCount*SIZE[LONG UNSPECIFIED],
      high: mdsiGerm]]];

    -- Previous replaces following for same reason as above.
    --LOOPHOLE[ Environment.Long[ any[
    --  low: @SDDefs.SD[SDDefs.sGermCount], high: mdsiGerm]]];

  pRequest: LONG POINTER TO Request =
    -- The job to be done by the Germ.
    -- This data is allocated in the Germ's MDS.
    -- NOTE: pRequest and the address of pRequest↑.action and pRequest↑.location
    -- are known by the initial microcode, and MesaNetExec.
    LOOPHOLE[ Environment.Long[ any[
      low: LOOPHOLE[SDDefs.SD, CARDINAL] + SDDefs.sFirstGermRequest
        * SIZE[LONG UNSPECIFIED], high: mdsiGerm]]];

    -- Previous replaces following for same reason as above.
    --LOOPHOLE[ Environment.Long[ any[
    --  low: @SDDefs.SD[SDDefs.sFirstGermRequest], high: mdsiGerm]]];

  -- (The allocation of SD[sFirstPilot] .. SD[sLastPilot] is defined in GermOps.)

  END.


LOG

September 13, 1979  6:03 PM   PXM       Create file
January 25, 1980  4:22 PM   PXM
   Add bootPhysicalVolume and noOp Request's and ethernet Location
January 25, 1980  6:37 PM   PXM
   Replace Location.device with Location.deviceType and .deviceOrdinal
April 17, 1980  12:41 AM   FXH       Added teledebug
April 17, 1980  10:39 AM   AWL         Added net and host to Location.ethernet
July 15, 1980  10:09 PM   FXH       Add NoOp; refer to Volume.Type
11-Aug-81 15:26:19   AWL
   Location.ethernet => Location.ethernetOne.  Added Location.ethernet forr 10MB ethernet.
   Made ReadMDS an INLINE.
13-Aug-81 15:28:05 WDK
  Switches prepended to Request; address decremented 20B to 13408.
  Un-loopholed pInitialLink.  Added pCrossMdsFrames and assertions.
  Made more machine dependent.
21-Aug-81  8:51:40   WDK
   pInitialLink changed to be an indirect ControlLink.
22-Oct-81 12:04:31   WDK
   New instruction set and SDDefs. Changed value of pInitialLink and pRequest.  Moved mdsiGerm, countSkip, pCountGerm, here from BootSwap
   and made most LONG.  Added pageGerm, countGermVM, currentVersions, bootPhysicalVolumeDiskAddress.  Improved documentation.
 5-Oct-82 10:05:45   AWL
   Modified DiskFileID to use PilotDisk.FileID and not File.ID.
12-Nov-82 11:59:00    LXD
   DiskAddress changed from opaque type to PilotDisk.Address.
28-Mar-83 15:33:43   WDK
   ..and thence to PilotDiskFace.DiskAddress (which is what it actually is). Added EthernetRequest, BootServerPacket, and
   EthernetBootFileNumber. Made Location.ethernet use them. Added DiskBootChainLink, etc.
 5-Apr-83 18:00:47   WDK        Moved BootServerPacket to BootServerDefs.
 6-Jul-83 13:22:36   WDK
   Added nullEthernetBootFileNumber. Made compatible with new PrincOps, SDDefs. Remove compiler bug workarounds.
 8-Oct-87 12:21:08   CAJ
   Changed constant in coungGermVM from 64 to 96.  As a result, changed currentRequestExtensionVersion from 71238 to 76548.  Altered
   pInitialLink, pCountGerm, and pRequest to form compiler will process.
```

```
-- BootUserImpl.mesa
-- 17-Apr-89 12:37:45

-- Copyright (c) 1989 by Xerox Corporation.  All rights reserved.

DIRECTORY
  Exec USING [AddCommand, ExecProc, RemoveCommand],
  OthelloToolDefs USING [Run],
  Process USING [Abort],
  Runtime USING [GetBcdTime],
  String USING [AppendString],
  Time USING [Append, Unpack],
  Tool USING [Create, Destroy, MakeSWsProc, MakeTTYSW, UnusedLogName],
  ToolWindow USING [Activate, TransitionProcType],
  TTY USING [Handle],
  TTYSW USING [GetTTYHandle],
  Version USING [Append],
  Volume USING [Close, ID, systemID],
  Window USING [Handle];
OthelloToolImpl: PROGRAM
  IMPORTS
    Exec, Runtime, String, OthelloToolDefs, Process, Time,
    Tool, ToolWindow, TTYSW, Version, Volume
  EXPORTS OthelloToolDefs = {

  tty: PUBLIC TTY.Handle;

  toolWindow: Window.Handle;

  CommandInterpreter: PROCESS;

  Init: PROCEDURE = {
    name: LONG STRING ← [75];
    name.length ← 0;
    String.AppendString[name, "OthelloTool "L];
    Version.Append[name];
    String.AppendString[name," of "L];
    Time.Append[name,Time.Unpack[Runtime.GetBcdTime[]]];
    toolWindow ← Tool.Create[
      name: name, makeSWsProc: MakeTTYSW, clientTransition: Stop,
      cmSection: "OthelloTool"L, tinyName1: "Othello"L, tinyName2: "Tool"L];
    Exec.AddCommand[name: "OthelloTool.~"L, proc: Activate, unload: DestroyTool];
  };

  Stop: ToolWindow.TransitionProcType = {
    IF new = inactive THEN {
      Process.Abort[CommandInterpreter];
      JOIN CommandInterpreter;};
  };

  Activate: Exec.ExecProc = {ToolWindow.Activate[toolWindow];};

  DestroyTool: Exec.ExecProc = {
    Exec.RemoveCommand[h, "OthelloTool.~"L];
    Tool.Destroy[toolWindow];
  };

  MakeTTYSW: Tool.MakeSWsProc = {
    logName: LONG STRING ← [20];
    ttySW: Window.Handle;
    logName.length ← 0;
    Tool.UnusedLogName[unused:logName, root: "OthelloTool.log"L];
    ttySW ← Tool.MakeTTYSW[window:window, name:logName];
    tty ← TTYSW.GetTTYHandle[ttySW];
    CommandInterpreter ← FORK OthelloToolDefs.Run[];
  };

  CloseVolume: PUBLIC PROCEDURE[volume: Volume.ID] = {
    IF volume # Volume.systemID THEN Volume.Close[volume];};

  Init[];

}.
```

```
-- BootUserImplA.mesa
-- 17-Apr-89 16:19:55

-- Copyright (c) 1989 by Xerox Corporation.  All rights reserved.

DIRECTORY
  OthelloDefs USING [
    AbortingCommand, CloseFetch, CommandProcessor, Confirm, GetName,
    IndexTooLarge, LeaderPage, leaderPages, lpVersion, MyNameIs, NewLine,
    PackedTimeFromString, Question, ReadNumber, RegisterCommandProc,
    SetCommandString, WriteChar, WriteFixedWidthNumber, WriteLine,
    WriteLongNumber, WriteOctal, WriteString, Yes],
  OthelloOps USING [
    BadSwitches, BootFileType, DecodeSwitches, DeleteTempFiles, GetDriveSize,
    GetNextSubVolume, GetPhysicalVolumeBootFile, GetSwitches, GetVolumeBootFile,
    nullSubVolume, SetDebugger, SetDebuggerSuccess, SetExpirationDate,
    SetExpirationDateSuccess, SetGetSwitchesSuccess, SetPhysicalVolumeBootFile,
    SetSwitches, SubVolume, VoidPhysicalVolumeBootFile, VoidVolumeBootFile],
  OthelloToolDefs USING [CloseVolume],
  SpecialVolume USING [OpenVolume],
  TemporaryBooting USING [BootButton],
  Volume USING [
    Erase, GetAttributes, GetLabelString, GetType, ID,
    NeedsScavenging, NotOnline, nullID, Open, systemID, Type],

VolumeInitImplA: PROGRAM
  IMPORTS
    File, Heap, Inline, OthelloDefs, OthelloOps, OthelloToolDefs, PhysicalVolume, Process, Runtime,
    Scavenger, Space, SpecialVolume, System, String, TemporaryBooting, Volume,
    VolumeVersion
  EXPORTS OthelloDefs
  SHARES File =
  BEGIN OPEN OthelloOps, OthelloDefs;

  Quit: PROC = {TemporaryBooting.BootButton[]};

  SetPvBoot: PROC =
    BEGIN
    lvID: Volume.ID ← GetLvIDFromUser[].lvID;
    SpecialVolume.OpenVolume[lvID, read];
    FOR t: BootFileType IN [softMicrocode..pilot] DO
      IF GetVolumeBootFile[lvID, t].file # File.nullFile THEN {
        file: File.File;
        firstPage: File.PageNumber;
        [file, firstPage] ← GetVolumeBootFile[lvID, t];
        SetPhysicalVolumeBootFile[file, t, firstPage]};
      ENDLOOP;
    OthelloToolDefs.CloseVolume[lvID];
    END;

  GetUserLvID: PROC [] RETURNS [lvID: Volume.ID] =
    BEGIN
    DO
     ptmpID:      PhysicalVolume.ID ← PhysicalVolume.nullID;
     inputString: LONG STRING ← "User";
     matches:     CARDINAL ← 0;
      DO
        driveTemp: PhysicalVolume.Handle;
        ltmpID:    Volume.ID ← Volume.nullID;
        IF (ptmpID ← PhysicalVolume.GetNext[ptmpID]) = PhysicalVolume.nullID THEN EXIT;
        driveTemp ← PhysicalVolume.GetAttributes[ptmpID].instance;
        DO
          s: STRING = [maxNameLength];
          IF (ltmpID ← PhysicalVolume.GetNextLogicalVolume[ptmpID, ltmpID])
             = Volume.nullID THEN EXIT;
          GetLogicalVolumeName[ltmpID, s ! Volume.NotOnline => LOOP];
          IF FunnyEqual[driveTemp, s, inputString] THEN {
            matches ← matches + 1; lvID ← ltmpID; pvID ← ptmpID; drive ← driveTemp};
          ENDLOOP;
        ENDLOOP;
      SELECT matches FROM
        0 => WriteString["Not found\r"L];
        1 => RETURN;
        ENDCASE => WriteLine["Ambigous; please specify Device:LogicalName"L];
      ENDLOOP;
    END;


  GetLogicalVolumeName: PROC [vid: Volume.ID, s: STRING] = {
    s.length ← 0;
    Volume.GetLabelString[vid, s ! Volume.NeedsScavenging => GOTO bad];
    EXITS bad => {
      IDRep: TYPE = RECORD [p: ARRAY [0..3) OF CARDINAL, n: LONG CARDINAL];
      String.AppendString[s, "NeedsScavenging"L];
      String.AppendLongNumber[s, LOOPHOLE[vid, IDRep].n, 8]}};


  GetLvIDFromUser: PUBLIC PROC [
    prompt: LONG STRING ← NIL,
    calledFromSetDebuggerPtrs: BOOLEAN ← FALSE]
    RETURNS [
      pvID: PhysicalVolume.ID, lvID: Volume.ID,
      drive: PhysicalVolume.Handle] =
    BEGIN
    IF prompt = NIL THEN prompt ← "Logical Volume Name: "L;
    DO
     ptmpID:      PhysicalVolume.ID ← PhysicalVolume.nullID;
```

```
    inputString: LONG STRING;
    matches:     CARDINAL ← 0;
    GetName[
       prompt: prompt, how: echo, signalQuestion: TRUE,
       dest: IF calledFromSetDebuggerPtrs THEN @debuggerLogicalString
             ELSE @inputLogicalString
       ! Question => {ListLogicalVolumes[]; RESUME}];
     IF calledFromSetDebuggerPtrs THEN {
       IF debuggerLogicalString.length=0 THEN {lvID ← Volume.nullID; RETURN}
       ELSE inputString ← debuggerLogicalString}
     ELSE {inputString ← inputLogicalString};
     DO
       driveTemp: PhysicalVolume.Handle;
       ltmpID:    Volume.ID ← Volume.nullID;
       IF (ptmpID ← PhysicalVolume.GetNext[ptmpID]) = PhysicalVolume.nullID THEN EXIT;
       driveTemp ← PhysicalVolume.GetAttributes[ptmpID].instance;
       DO
         s: STRING = [maxNameLength];
         IF (ltmpID ← PhysicalVolume.GetNextLogicalVolume[ptmpID, ltmpID])
            = Volume.nullID THEN EXIT;
         GetLogicalVolumeName[ltmpID, s ! Volume.NotOnline => LOOP];
         IF FunnyEqual[driveTemp, s, inputString] THEN {
           matches ← matches + 1; lvID ← ltmpID; pvID ← ptmpID; drive ← driveTemp};
         ENDLOOP;
       ENDLOOP;
     SELECT matches FROM
       0 => WriteString["Not found\r"L];
       1 => RETURN;
       ENDCASE => WriteLine["Ambigous; please specify Device:LogicalName"L];
     ENDLOOP;
   END;


FunnyEqual: PROC [h: PhysicalVolume.Handle, name: STRING, userName: LONG STRING]
  RETURNS[BOOLEAN] = {

  SameChar: PROC [a, b: CHARACTER]
    RETURNS [BOOLEAN] = {
    IF a=b THEN RETURN[TRUE]
    ELSE IF a IN ['a..'z] AND b IN ['A..'Z] AND (a-'a+'A)=b THEN RETURN[TRUE]
    ELSE IF a IN ['A..'Z] AND b IN ['a..'z] AND (a-'A+'a)=b THEN RETURN[TRUE]
    ELSE RETURN[FALSE]};

  IF String.Equivalent[name, userName] THEN RETURN[TRUE];
  FOR i: CARDINAL IN [0..driveName.length) DO
    IF ~SameChar[driveName[i], userName[i]] THEN RETURN[FALSE] ENDLOOP;
  IF driveName.length+name.length+1 # userName.length THEN RETURN[FALSE];
  IF userName[driveName.length] # ': THEN RETURN[FALSE];
  FOR i: CARDINAL IN [0..name.length) DO
    IF ~SameChar[name[i], userName[driveName.length+1+i]] THEN RETURN[FALSE]
    ENDLOOP;
  RETURN[TRUE]};


END.....
```

```
-- File: HexConvertToolImpl.mesa - last edit:
-- Cooper:OSBU North:Xerox 25-Nov-88 13:56:03
-- MManley 18-Feb-86 23:13:57

-- Copyright (C) 1988 by Xerox Corporation. All rights reserved.

DIRECTORY
  Ascii USING [CR, SP],
  Environment USING [Byte],
  Event USING [DoneWithProcess, Handle, StartingProcess, toolWindow],
  EventTypes USING [deactivate],
  Exec USING [AddCommand, ExecProc, RemoveCommand],
  FormSW USING [AllocateItemDescriptor, BooleanItem, ClientItemsProcType,
    CommandItem, ItemHandle, line0, line1, line2, ProcType, StringItem],
  Heap USING [Create, Delete],
  Inline USING [BITAND, BITOR, BITSHIFT],
  MFile USING [Type],
  MStream USING [Error, GetLength, ReadOnly, SetLength, WriteOnly],
  Process USING [Detach],
  Put USING [Line, Text],
  Runtime USING [GetBcdTime],
  Stream USING [Delete, EndOfStream, GetChar, GetByte, Handle, PutByte, PutChar,
    PutString],
  String USING [AppendString],
  Supervisor USING [AddDependency, AgentProcedure, CreateSubsystem,
    EnumerationAborted, RemoveDependency, SubsystemHandle],
  Time USING [Append, AppendCurrent, Unpack],
  Tool USING [Create, Destroy, MakeFileSW, MakeFormSW, MakeMsgSW, MakeSWsProc,
    UnusedLogName],
  ToolWindow USING [Activate, GetState, TransitionProcType],
  Window USING [GetChild, GetParent, Handle, Stack, ValidateTree];

HexConvertToolImpl: PROGRAM
  IMPORTS Event, Exec, FormSW, Heap, Inline, MStream, Process, Put,
    Runtime, Stream, String, Supervisor, Time, Tool, ToolWindow, Window
  =
  BEGIN

  ----------------------
  -- Types
  ----------------------

  FormIndex: TYPE = {protectBin, binFile, protectHex, hexFile, binToHex,
    hexToBin};

  ToolData: TYPE = MACHINE DEPENDENT RECORD [
    msgSW(0): Window.Handle ← NIL,
    formSW(2): Window.Handle ← NIL,
    logSW(4): Window.Handle ← NIL,
    protectBin(6): BOOLEAN ← TRUE,
    protectHex(7): BOOLEAN ← TRUE,
    binFileName(8): LONG STRING ← NIL,
    hexFileName(10): LONG STRING ← NIL,
    commandIsRunning(12): BOOLEAN ← FALSE];

  ----------------------
  -- Constants
  ----------------------

  agent: Supervisor.SubsystemHandle = Supervisor.CreateSubsystem[CheckDeactivate];

  ----------------------
  -- Globals
  ----------------------

  data: LONG POINTER TO ToolData ← NIL;
  wh: Window.Handle ← NIL;
  heap: UNCOUNTED ZONE ← NIL;

  ----------------------
  -- Initialisation
  ----------------------

  Init: PROC =
    BEGIN
    Exec.AddCommand["HexConvertTool.~"L, MakeTool, NIL, Unload];
    END; -- Init

  MakeTool: Exec.ExecProc =
    BEGIN
    IF wh = NIL THEN
      BEGIN
      name: LONG STRING ← [60];
      String.AppendString[to: name, from : "Hex Convert Tool 3.0 of "L];
      Time.Append[s: name, unpacked: Time.Unpack[Runtime.GetBcdTime[]]];
      name.length ← name.length - 3; -- lop the seconds
      wh ← Tool.Create[name: name, makeSWsProc: MakeSWs,
        clientTransition: ClientTransition, cmSection: "HexConvertTool"L,
        tinyName1: "Hex"L, tinyName2: "Convert"L];
      END
    ELSE IF ToolWindow.GetState[wh] = active THEN
      BEGIN
      newSibling: Window.Handle = Window.GetChild[Window.GetParent[wh]];
      IF wh # newSibling THEN {
        Window.Stack[wh, newSibling]; Window.ValidateTree[wh]};
      END
    ELSE
```

```
    ToolWindow.Activate[wh];
  END; -- MakeTool

Unload: Exec.ExecProc =
  BEGIN
  IF wh # NIL THEN {Tool.Destroy[wh]; wh ← NIL};
  Exec.RemoveCommand[h, "HexConvertTool.~"L];
  END; -- Unload

--====================
-- State change
--====================

CheckDeactivate: Supervisor.AgentProcedure =
  BEGIN
  IF event = EventTypes.deactivate AND wh # NIL
   AND wh = eventData AND data.commandIsRunning THEN
    BEGIN
    Put.Line[data.msgSW, "Tool is busy!"L];
    ERROR Supervisor.EnumerationAborted;
    END;
  END; -- CheckDeactivate

ClientTransition: ToolWindow.TransitionProcType =
  BEGIN
  SELECT TRUE FROM
    old = inactive =>
      BEGIN
      IF heap = NIL THEN heap ← Heap.Create[initial: 1, increment: 1];
      IF data = NIL THEN data ← heap.NEW[ToolData ← []];
      END;
    new = inactive =>
      BEGIN
      Supervisor.RemoveDependency[client: agent, implementor: Event.toolWindow];
      IF data # NIL THEN heap.FREE[@data];
      IF heap # NIL THEN {Heap.Delete[heap]; heap ← NIL};
      END;
    ENDCASE;
  END; -- ClientTransition

--====================
-- Tool window
--====================

MakeSWs: Tool.MakeSWsProc =
  BEGIN
  logFileName: STRING = [50];
  Tool.UnusedLogName[unused: logFileName, root: "HexConvertTool.log"L];
  data.msgSW ← Tool.MakeMsgSW[window: window];
  data.formSW ← Tool.MakeFormSW[window: window, formProc: MakeForm];
  data.logSW ← Tool.MakeFileSW[window: window, name: logFileName];
  Supervisor.AddDependency[client: agent, implementor: Event.toolWindow];
  END; -- MakeSWs

MakeForm: FormSW.ClientItemsProcType =
  BEGIN
  OPEN FormSW;
  nItems: CARDINAL = FormIndex.LAST.ORD + 1;
  items ← AllocateItemDescriptor[nItems];
  items[FormIndex.protectBin.ORD] ← BooleanItem[
    tag: "ProtectBinary"L, place: [0, line0], switch: @data.protectBin];
  items[FormIndex.binFile.ORD] ← StringItem[
    tag: "Binary File"L, place: [100, line0], inHeap: TRUE,
    string: @data.binFileName];
  items[FormIndex.protectHex.ORD] ← BooleanItem[
    tag: "ProtectHex"L, place: [0, line1], switch: @data.protectHex];
  items[FormIndex.hexFile.ORD] ← StringItem[
    tag: "Hex File"L, place: [100, line1], inHeap: TRUE,
    string: @data.hexFileName];
  items[FormIndex.binToHex.ORD] ← CommandItem[
    tag: "Convert Binary To Hex"L, place: [0, line2], proc: Convert];
  items[FormIndex.hexToBin.ORD] ← CommandItem[
    tag: "Convert Hex To Binary"L, place: [200, line2], proc: Convert];
  RETURN[items: items, freeDesc: TRUE];
  END; -- MakeForm

--====================
-- Convert procs
--====================

Convert: FormSW.ProcType =
  BEGIN
  IF data.commandIsRunning THEN
    Put.Line[data.msgSW, "Tool is busy!"L]
  ELSE
    BEGIN
    convertProc: PROC ← SELECT index FROM
      FormIndex.binToHex.ORD => BinToHex,
      FormIndex.hexToBin.ORD => HexToBin,
      ENDCASE => ERROR;
    data.commandIsRunning ← TRUE;
    Process.Detach[FORK convertProc[]];
    END;
  END; -- Convert

BinToHex: PROC =
  BEGIN
```

```
    PutByteHex: PROC [byte: Environment.Byte] =
      BEGIN
      NibbleToChar: PROC [n: CARDINAL] RETURNS [c: CHAR] = INLINE
        BEGIN
        c ← SELECT n FROM
          IN [0..9] => VAL[ORD['0] + n],
          IN [10..15] => VAL[ORD['A] + n - 10],
          ENDCASE => ERROR;
        END; -- NibbleToChar
      Stream.PutChar[hexStream,
        NibbleToChar[Inline.BITSHIFT[Inline.BITAND[byte, 0F0H], -4]]];
      Stream.PutChar[hexStream, NibbleToChar[Inline.BITAND[byte, 0FH]]];
      END; -- PutByteHex
    binStream, hexStream: Stream.Handle ← NIL;
    handle: Event.Handle ← Event.StartingProcess["Hex Convert Tool is running"L];
    binStream ← MStream.ReadOnly[data.binFileName, [NIL, NIL] !
      MStream.Error => CONTINUE];
    IF binStream = NIL THEN {
      CleanUp["File Not Available!\n"L, handle];
      RETURN};
    hexStream ← GetWriteStream[data.hexFileName, ~data.protectHex, text];
    IF hexStream = NIL THEN {
      Stream.Delete[binStream];
      CleanUp["Output file is protected!\n"L, handle];
      RETURN};
    PutBoth["Converting binary file """L];
    PutBoth[data.binFileName];
    PutBoth[""" to hex file """L];
    PutBoth[data.hexFileName];
    PutBoth[""" ..."L];
    WriteHeader[hexStream];
    DO
      ENABLE Stream.EndOfStream => EXIT;
      THROUGH [0..8) DO
        THROUGH [0 .. 16) DO
          PutByteHex[Stream.GetByte[binStream]];
          PutByteHex[Stream.GetByte[binStream]];
          Stream.PutChar[hexStream, Ascii.SP];
        ENDLOOP;
        Stream.PutChar[hexStream, Ascii.CR];
      ENDLOOP;
      Stream.PutChar[hexStream, Ascii.CR];
      Stream.PutChar[hexStream, Ascii.CR];
    ENDLOOP;
    Stream.PutChar[hexStream, Ascii.CR];
    Stream.PutChar[hexStream, Ascii.CR];
    Stream.Delete[binStream];
    MStream.SetLength[hexStream, MStream.GetLength[hexStream]];
    Stream.Delete[hexStream];
    CleanUp[".. Done!\n"L, handle];
    END; -- BinToHex

  HexToBin: PROC =
    BEGIN
    PutNibble: PROC [nibble: Environment.Byte] =
      BEGIN
      IF isFirstNibble THEN
        firstNibble ← nibble
      ELSE
        Stream.PutByte[binStream,
          Inline.BITOR[Inline.BITSHIFT[firstNibble, 4], nibble]];
      isFirstNibble ← ~isFirstNibble;
      END; -- PutNibble
    binStream, hexStream: Stream.Handle ← NIL;
    isFirstNibble: BOOLEAN ← TRUE;
    firstNibble: Environment.Byte ← 0;
    char: CHARACTER;
    handle: Event.Handle ← Event.StartingProcess["Hex Convert Tool is running"L];
    hexStream ← MStream.ReadOnly[data.hexFileName, [NIL, NIL] !
      MStream.Error => CONTINUE];
    IF hexStream = NIL THEN {
      CleanUp["File Not Available!\n"L, handle];
      RETURN};
    binStream ← GetWriteStream[data.binFileName, ~data.protectBin, binary];
    IF binStream = NIL THEN {
      Stream.Delete[hexStream];
      CleanUp["Output file is protected!\n"L, handle];
      RETURN};
    PutBoth["Converting hexadecimal file """L];
    PutBoth[data.hexFileName];
    PutBoth[""" to binary file """L];
    PutBoth[data.binFileName];
    PutBoth[""" ..."L];
    DO
      BEGIN
      ENABLE Stream.EndOfStream => EXIT;
      char ← Stream.GetChar[hexStream];
      SELECT char FROM
        IN ['a..'f] => PutNibble[char - 'a + 10];
        IN ['A..'F] => PutNibble[char - 'A + 10];
        IN ['0..'9] => PutNibble[char - '0];
        Ascii.CR, Ascii.SP => NULL;
        '- =>
          BEGIN
          IF Stream.GetChar[hexStream] = '- THEN
            UNTIL Stream.GetChar[hexStream] = Ascii.CR DO ENDLOOP
          ELSE
```

```
            GOTO badHexChar;
          END;
        ENDCASE => GOTO badHexChar;
      EXITS badHexChar =>
        BEGIN
        Stream.Delete[hexStream];
        MStream.SetLength[binStream, 0];
        Stream.Delete[binStream];
        CleanUp["Error: Non-hex character in binary file!\n"L, handle];
        RETURN;
        END;
      END;
    ENDLOOP;
    IF ~isFirstNibble THEN {
      PutBoth["Warning: Odd number of nibbles in hex file!"L]; PutNibble[0]};
    Stream.Delete[hexStream];
    MStream.SetLength[binStream, MStream.GetLength[binStream]];
    Stream.Delete[binStream];
    CleanUp[".. Done!\n"L, handle];
    END; -- HexToBin

  GetWriteStream: PROC [name: LONG STRING, overwrite: BOOLEAN, type: MFile.Type]
   RETURNS [stream: Stream.Handle ← NIL] =
    BEGIN
    stream ← MStream.ReadOnly[name, [NIL, NIL] ! MStream.Error => CONTINUE];
    IF stream # NIL THEN {
      Stream.Delete[stream]; IF ~overwrite THEN RETURN[NIL]};
    stream ← MStream.WriteOnly[name, [NIL, NIL], type];
    END; -- GetWriteStream

  WriteHeader: PROC [stream: Stream.Handle] =
    BEGIN
    dateAndTime: LONG STRING ← [40];
    Time.AppendCurrent[dateAndTime, TRUE];
    Stream.PutString[stream, "-- File: "L];
    Stream.PutString[stream, data.hexFileName];
    Stream.PutString[stream, "\n-- From: "L];
    Stream.PutString[stream, data.binFileName];
    Stream.PutString[stream, "\n-- Date: "L];
    Stream.PutString[stream, dateAndTime];
    Stream.PutString[stream, "\n\n"L];
    END; -- WriteHeader

  CleanUp: PROC [reason: LONG STRING, event: Event.Handle] =
    BEGIN
    PutBoth[reason];
    data.commandIsRunning ← FALSE;
    Event.DoneWithProcess[event];
    END; -- CleanUp

  PutBoth: PROC [s: LONG STRING] = {
    Put.Text[data.msgSW, s]; Put.Text[data.logSW, s]};


  --~~~~~~~~~~~~~~~~~~~~
  -- Mainline code
  --~~~~~~~~~~~~~~~~~~~~

   Init[];

END...

LOG (Editor/Date/Comment):
  Mike Manley/10-Feb-84 20:26:52/Create program.
  Mike Manley/16-Aug-84 10:48:00/Converted to Mesa 11.0.
  Mike Manley/18-Feb-85 23:13:52/Converted to HexEditTool.
  Martin Cooper/13-Nov-88 20:24:58/Converted to 14.0 & changed style.
  Martin Cooper/14-Nov-88 15:09:34/Radical revamp of entire module.
```

```
-- File: HexConvertToolImpl.mesa - last edit:
--  3-Feb-89 22:01:39
-- Cooper:OSBU North:Xerox 25-Nov-88 13:56:03
-- MManley 18-Feb-85 23:13:57

-- Copyright (C) 1988 by Xerox Corporation. All rights reserved.

DIRECTORY
  Ascii USING [CR, SP],
  Environment USING [Byte],
  Event USING [DoneWithProcess, Handle, StartingProcess, toolWindow],
  EventTypes USING [deactivate],
  Exec USING [AddCommand, ExecProc, RemoveCommand],
  FormSW USING [AllocateItemDescriptor, BooleanItem, ClientItemsProcType,
    CommandItem, ItemHandle, line0, line1, line2, ProcType, StringItem],
  Heap USING [Create, Delete],
  Inline USING [BITAND, BITSHIFT],
  MFile USING [Type],
  MStream USING [Error, GetLength, ReadOnly, SetLength, WriteOnly],
  Process USING [Detach],
  Put USING [Line, Text],
  Runtime USING [GetBcdTime],
  Stream USING [Delete, EndOfStream, GetChar, GetByte, Handle, PutByte, PutChar,
    PutString],
  String USING [AppendString, AppendLongDecimal],
  Supervisor USING [AddDependency, AgentProcedure, CreateSubsystem,
    EnumerationAborted, RemoveDependency, SubsystemHandle],
  Time USING [Append, AppendCurrent, Unpack],
  Tool USING [Create, Destroy, MakeFileSW, MakeFormSW, MakeMsgSW, MakeSWsProc,
    UnusedLogName],
  ToolWindow USING [Activate, GetState, TransitionProcType],
  Window USING [GetChild, GetParent, Handle, Stack, ValidateTree];

HexConvertToolImpl: PROGRAM
  IMPORTS Event, Exec, FormSW, Heap, Inline, MStream, Process, Put,
    Runtime, Stream, String, Supervisor, Time, Tool, ToolWindow, Window
  =
  BEGIN

  ----------------------
  -- Types
  ----------------------

  FormIndex: TYPE = {protectBin, binFile, protectHex, hexFile, binToHex,
    hexToBin};

  ToolData: TYPE = MACHINE DEPENDENT RECORD [
    msgSW(0): Window.Handle ← NIL,
    formSW(2): Window.Handle ← NIL,
    logSW(4): Window.Handle ← NIL,
    protectBin(6): BOOLEAN ← TRUE,
    protectHex(7): BOOLEAN ← TRUE,
    binFileName(8): LONG STRING ← NIL,
    hexFileName(10): LONG STRING ← NIL,
    commandIsRunning(12): BOOLEAN ← FALSE];

  ----------------------
  -- Constants
  ----------------------

  agent: Supervisor.SubsystemHandle = Supervisor.CreateSubsystem[CheckDeactivate];

  ----------------------
  -- Globals
  ----------------------

  data: LONG POINTER TO ToolData ← NIL;
  wh: Window.Handle ← NIL;
  heap: UNCOUNTED ZONE ← NIL;

  ----------------------
  -- Initialisation
  ----------------------

  Init: PROC =
    BEGIN
    Exec.AddCommand["HexConvertTool.~"L, MakeTool, NIL, Unload];
    END; -- Init

  MakeTool: Exec.ExecProc =
    BEGIN
    IF wh = NIL THEN
      BEGIN
      name: LONG STRING ← [60];
      String.AppendString[to: name, from : "Hex Convert Tool 3.0 of "L];
      Time.Append[s: name, unpacked: Time.Unpack[Runtime.GetBcdTime[]]];
      name.length ← name.length - 3; -- lop the seconds
      wh ← Tool.Create[name: name, makeSWsProc: MakeSWs,
        clientTransition: ClientTransition, cmSection: "HexConvertTool"L,
        tinyName1: "Hex"L, tinyName2: "Convert"L];
      END
    ELSE IF ToolWindow.GetState[wh] = active THEN
      BEGIN
      newSibling: Window.Handle = Window.GetChild[Window.GetParent[wh]];
      IF wh # newSibling THEN {
        Window.Stack[wh, newSibling]; Window.ValidateTree[wh]};
      END
```

```
    ELSE
      ToolWindow.Activate[wh];
    END; -- MakeTool

  Unload: Exec.ExecProc =
    BEGIN
    IF wh # NIL THEN {Tool.Destroy[wh]; wh ← NIL};
    Exec.RemoveCommand[h, "HexConvertTool.~"L];
    END; -- Unload

  --====================
  -- State change
  --====================

  CheckDeactivate: Supervisor.AgentProcedure =
    BEGIN
    IF event = EventTypes.deactivate AND wh # NIL
     AND wh = eventData AND data.commandIsRunning THEN
      BEGIN
      Put.Line[data.msgSW, "Tool is busy!"L];
      ERROR Supervisor.EnumerationAborted;
      END;
    END; -- CheckDeactivate

  ClientTransition: ToolWindow.TransitionProcType =
    BEGIN
    SELECT TRUE FROM
      old = inactive =>
        BEGIN
        IF heap = NIL THEN heap ← Heap.Create[initial: 1, increment: 1];
        IF data = NIL THEN data ← heap.NEW[ToolData ← []];
        END;
      new = inactive =>
        BEGIN
        Supervisor.RemoveDependency[client: agent, implementor: Event.toolWindow];
        IF data # NIL THEN heap.FREE[@data];
        IF heap # NIL THEN {Heap.Delete[heap]; heap ← NIL};
        END;
      ENDCASE;
    END; -- ClientTransition

  --====================
  -- Tool window
  --====================

  MakeSWs: Tool.MakeSWsProc =
    BEGIN
    logFileName: STRING = [50];
    Tool.UnusedLogName[unused: logFileName, root: "HexConvertTool.log"L];
    data.msgSW ← Tool.MakeMsgSW[window: window];
    data.formSW ← Tool.MakeFormSW[window: window, formProc: MakeForm];
    data.logSW ← Tool.MakeFileSW[window: window, name: logFileName];
    Supervisor.AddDependency[client: agent, implementor: Event.toolWindow];
    END; -- MakeSWs

  MakeForm: FormSW.ClientItemsProcType =
    BEGIN
    OPEN FormSW;
    nItems: CARDINAL = FormIndex.LAST.ORD + 1;
    items ← AllocateItemDescriptor[nItems];
    items[FormIndex.protectBin.ORD] ← BooleanItem[
      tag: "ProtectBinary"L, place: [0, line0], switch: @data.protectBin];
    items[FormIndex.binFile.ORD] ← StringItem[
      tag: "Binary File"L, place: [100, line0], inHeap: TRUE,
      string: @data.binFileName];
    items[FormIndex.protectHex.ORD] ← BooleanItem[
      tag: "ProtectHex"L, place: [0, line1], switch: @data.protectHex];
    items[FormIndex.hexFile.ORD] ← StringItem[
      tag: "Hex File"L, place: [100, line1], inHeap: TRUE,
      string: @data.hexFileName];
    items[FormIndex.binToHex.ORD] ← CommandItem[
      tag: "Convert Binary To Hex"L, place: [0, line2], proc: Convert];
    items[FormIndex.hexToBin.ORD] ← CommandItem[
      tag: "Convert Hex To Binary"L, place: [200, line2], proc: Convert];
    RETURN[items: items, freeDesc: TRUE];
    END; -- MakeForm

  --====================
  -- Convert procs
  --====================

  Convert: FormSW.ProcType =
    BEGIN
    IF data.commandIsRunning THEN
      Put.Line[data.msgSW, "Tool is busy!"L]
    ELSE
      BEGIN
      convertProc: PROC ← SELECT index FROM
        FormIndex.binToHex.ORD => BinToHex,
        FormIndex.hexToBin.ORD => HexToBin,
        ENDCASE => ERROR;
      data.commandIsRunning ← TRUE;
      Process.Detach[FORK convertProc[]];
      END;
    END; -- Convert

  BinToHex: PROC =
```

```
    BEGIN
    PutByteHex: PROC [byte: Environment.Byte] =
      BEGIN
      NibbleToChar: PROC [n: CARDINAL] RETURNS [c: CHAR] = INLINE
        BEGIN
        c ← SELECT n FROM
          IN [0..7] => VAL[ORD['0] + n],
          ENDCASE => ERROR;
        END; -- NibbleToChar
      Stream.PutChar[hexStream, NibbleToChar[Inline.BITSHIFT[Inline.BITAND[byte, 300B], -6]]];
      Stream.PutChar[hexStream, NibbleToChar[Inline.BITSHIFT[Inline.BITAND[byte, 070B], -3]]];
      Stream.PutChar[hexStream, NibbleToChar[Inline.BITAND[byte, 007B]]];
      END; -- PutByteHex
    wordCount: LONG INTEGER ← 0;
    wordString: LONG STRING ← [20];
    binStream, hexStream: Stream.Handle ← NIL;
    handle: Event.Handle ← Event.StartingProcess["Hex Convert Tool is running"L];
    binStream ← MStream.ReadOnly[data.binFileName, [NIL, NIL] !
      MStream.Error => CONTINUE];
    IF binStream = NIL THEN {
      CleanUp["File Not Available!\n"L, handle];
      RETURN};
    hexStream ← GetWriteStream[data.hexFileName, ~data.protectHex, text];
    IF hexStream = NIL THEN {
      Stream.Delete[binStream];
      CleanUp["Output file is protected!\n"L, handle];
      RETURN};
    PutBoth["Converting binary file """L];
    PutBoth[data.binFileName];
    PutBoth[""" to hex file """L];
    PutBoth[data.hexFileName];
    PutBoth[""" ..."L];
    WriteHeader[hexStream];
    Stream.PutString[hexStream, "\n-- "L];
    String.AppendLongDecimal[wordString, wordCount];
    Stream.PutString[hexStream, wordString];
    Stream.PutChar[hexStream, Ascii.CR];
    Stream.PutChar[hexStream, Ascii.CR];
    DO
      ENABLE Stream.EndOfStream => EXIT;
      THROUGH [0..10) DO
        THROUGH [0 .. 10) DO
          PutByteHex[Stream.GetByte[binStream]];
          Stream.PutChar[hexStream, '.];
          PutByteHex[Stream.GetByte[binStream]];
          Stream.PutChar[hexStream, Ascii.SP];
          Stream.PutChar[hexStream, Ascii.SP];
        ENDLOOP;
        Stream.PutChar[hexStream, Ascii.CR];
      ENDLOOP;
      wordCount ← wordCount + 50;
      Stream.PutString[hexStream, "\n\n-- "L];
      wordString.length ← 0;
      String.AppendLongDecimal[wordString, wordCount];
      Stream.PutString[hexStream, wordString];
      Stream.PutChar[hexStream, Ascii.CR];
      Stream.PutChar[hexStream, Ascii.CR];
    ENDLOOP;
    Stream.PutChar[hexStream, Ascii.CR];
    Stream.PutChar[hexStream, Ascii.CR];
    Stream.Delete[binStream];
    MStream.SetLength[hexStream, MStream.GetLength[hexStream]];
    Stream.Delete[hexStream];
    CleanUp[".. Done!\n"L, handle];
    END; -- BinToHex

  HexToBin: PROC =
    BEGIN
    binStream, hexStream: Stream.Handle ← NIL;
    byte: Environment.Byte ← 0;
    count: INTEGER ← 0;
    char: CHARACTER;
    handle: Event.Handle ← Event.StartingProcess["Hex Convert Tool is running"L];
    hexStream ← MStream.ReadOnly[data.hexFileName, [NIL, NIL] !
      MStream.Error => CONTINUE];
    IF hexStream = NIL THEN {
      CleanUp["File Not Available!\n"L, handle];
      RETURN};
    binStream ← GetWriteStream[data.binFileName, ~data.protectBin, binary];
    IF binStream = NIL THEN {
      Stream.Delete[hexStream];
      CleanUp["Output file is protected!\n"L, handle];
      RETURN};
    PutBoth["Converting hexadecimal file """L];
    PutBoth[data.hexFileName];
    PutBoth[""" to binary file """L];
    PutBoth[data.binFileName];
    PutBoth[""" ..."L];
    DO
      BEGIN
      ENABLE Stream.EndOfStream => EXIT;
      char ← Stream.GetChar[hexStream];
      SELECT char FROM
        IN ['0..'7] => {
          byte ← byte * 8 + char - '0;
          IF (count ← count + 1) = 3 THEN {
            Stream.PutByte[binStream, byte];
```

```
            byte ← 0;
            count ← 0;
            };
          };
        Ascii.CR, Ascii.SP, '. => NULL;
        '- =>
          BEGIN
          IF Stream.GetChar[hexStream] = '- THEN
            UNTIL Stream.GetChar[hexStream] = Ascii.CR DO ENDLOOP
          ELSE
            GOTO badHexChar;
          END;
        ENDCASE => GOTO badHexChar;
      EXITS badHexChar =>
        BEGIN
        Stream.Delete[hexStream];
        MStream.SetLength[binStream, 0];
        Stream.Delete[binStream];
        CleanUp["Error: Non-hex character in binary file!\n"L, handle];
        RETURN;
        END;
      END;
    ENDLOOP;
    Stream.Delete[hexStream];
    MStream.SetLength[binStream, MStream.GetLength[binStream]];
    Stream.Delete[binStream];
    CleanUp[".. Done!\n"L, handle];
    END; -- HexToBin

  GetWriteStream: PROC [name: LONG STRING, overwrite: BOOLEAN, type: MFile.Type]
   RETURNS [stream: Stream.Handle ← NIL] =
    BEGIN
    stream ← MStream.ReadOnly[name, [NIL, NIL] ! MStream.Error => CONTINUE];
    IF stream # NIL THEN {
      Stream.Delete[stream]; IF ~overwrite THEN RETURN[NIL]};
    stream ← MStream.WriteOnly[name, [NIL, NIL], type];
    END; -- GetWriteStream

  WriteHeader: PROC [stream: Stream.Handle] =
    BEGIN
    dateAndTime: LONG STRING ← [40];
    Time.AppendCurrent[dateAndTime, TRUE];
    Stream.PutString[stream, "-- File: "L];
    Stream.PutString[stream, data.hexFileName];
    Stream.PutString[stream, "\n-- From: "L];
    Stream.PutString[stream, data.binFileName];
    Stream.PutString[stream, "\n-- Date: "L];
    Stream.PutString[stream, dateAndTime];
    Stream.PutString[stream, "\n\n"L];
    END; -- WriteHeader

  CleanUp: PROC [reason: LONG STRING, event: Event.Handle] =
    BEGIN
    PutBoth[reason];
    data.commandIsRunning ← FALSE;
    Event.DoneWithProcess[event];
    END; -- CleanUp

  PutBoth: PROC [s: LONG STRING] = {
    Put.Text[data.msgSW, s]; Put.Text[data.logSW, s]};


  ----------------------
  -- Mainline code
  ----------------------

   Init[];

END...

LOG (Editor/Date/Comment):
  Mike Manley/10-Feb-84 20:26:52/Create program.
  Mike Manley/16-Aug-84 10:48:00/Converted to Mesa 11.0.
  Mike Manley/18-Feb-85 23:13:52/Converted to HexEditTool.
  Martin Cooper/13-Nov-88 20:24:58/Converted to 14.0 & changed style.
  Martin Cooper/14-Nov-88 15:09:34/Radical revamp of entire module.
  Trow/ 3-Feb-89 10:45:27/10 words/line, 10 lines/block.
```

```
<<File K4.mesa

deLaBeaujardiere:OSBU North:Xerox27-Apr-87 12:57:34


>>

DIRECTORY Ascii, Containee, FormWindow,
          NSFile, NSFileStream,  NSString,
          PropertySheet, Prototype, StarWindowShell, Stream,
          Window, XString;


K4: DEFINITIONS =
BEGIN
OPEN SWS: StarWindowShell;



GeneralItems: TYPE = {iconName, channelSpeed, folderName, others};

TextItems: TYPE = {folderName, docName,
                   justification, lineHeight,
                   preLeading, postLeading,
                   underlining,
                   guessMark, dropKeepMark,
                   pageBreak,
                   others};

MapItems: TYPE = {header, from, to};

FontItems: TYPE = {font0, size0, italics0, bold0,
                   font1, size1, italics1, bold1,
                   font2, size2, italics2, bold2,
                   font3, size3, italics3, bold3,
                   font4, size4, italics4, bold4,
                   font5, size5, italics5, bold5,
                   font6, size6, italics6, bold6,
                   font7, size7, italics7, bold7,
                   font8, size8, italics8, bold8,
                   font9, size9, italics9, bold9};

IconParms:              TYPE = LONG POINTER TO IconParmsRecord;
IconParmsRecord:        TYPE = RECORD[
        heap:           UNCOUNTED ZONE ← NIL,
        propSheet:      SWS.Handle ← SWS.nullHandle,
        iconData:       Containee.DataHandle ← NIL,
        changeProc:     Containee.ChangeProc ← NIL,
        changeProcData: LONG POINTER ← NIL,
        iconFile:       NSFile.Handle ← NSFile.nullHandle,
        signature:      CARDINAL ← 0,
        genProps:       GeneralProperties ← NIL,
        docProps:       DocumentProperties ← NIL,
        mapProps:       MapProperties ← NIL,
        fonProps:       FontProperties ← NIL];

ConvertParms:           TYPE = LONG POINTER TO ConvertParmsRecord;
ConvertParmsRecord:     TYPE = RECORD [
        zone:           UNCOUNTED ZONE ← NIL,
        optionSheet:    SWS.Handle ← SWS.nullHandle,
        logFile:        NSFile.Handle ← NSFile.nullHandle,
        docProps:       DocumentProperties ← NIL,
        mapProps:       MapProperties ← NIL,
        fonProps:       FontProperties ← NIL];

GeneralProperties:      TYPE = LONG POINTER TO GeneralPropertyRecord;
GeneralPropertyRecord:  TYPE = RECORD [
        iconName:       XString.ReaderBody,
        channelSpeed:   ChannelSpeed,
        folderName:     XString.ReaderBody,
        others:         XString.ReaderBody,
        tagSize:        LONG POINTER TO GeneralTagSizes ← NIL];

DocumentProperties:     TYPE = LONG POINTER TO DocumentPropertyRecord;
DocumentPropertyRecord: TYPE = RECORD [

K4.mesa         27-Apr-87 12:57:37 PDT
```

```
        folderName:     XString.ReaderBody,
        docName:        XString.ReaderBody,
        justification:  BOOLEAN,
        lineHeight:     LineSpacing,
        preLeading:     LineSpacing,
        postLeading:    LineSpacing,
        underlining:    Underlining,
        guessMark:      XString.ReaderBody,
        dropKeepMark:   DropKeep,
        pageBreak:      PageBreak,
        others:         XString.ReaderBody,
        tagSize:        LONG POINTER TO TextTagSizes ← NIL];

MapProperties:          TYPE = LONG POINTER TO MapPropertyRecord;
MapPropertyRecord:      TYPE = RECORD [
        header:         XString.ReaderBody,
        number:         CARDINAL ← 0,
        map:            Mapping ← NIL,
        tagSize:        LONG POINTER TO MapTagSizes ← NIL];

FontProperties: TYPE = LONG POINTER TO FontPropertyRecord;
FontPropertyRecord:     TYPE = RECORD [
        fonts:          ARRAY [0..9] OF FontRecord,
        tagSize:        LONG POINTER TO FontTagSizes ← NIL];

Mapping:                TYPE = LONG POINTER TO MappingRecord;
MappingRecord:          TYPE = RECORD [
        next:           Mapping ← NIL,
        from:           XString.ReaderBody,
        to:             XString.ReaderBody];

FontRecord:             TYPE = RECORD [
        font:           FontStyle,
        size:           FontSize,
        italics:        BOOLEAN,
        bold:           BOOLEAN];

ChannelSpeed:   TYPE = {ninetySix, fortyEight, three}; -- hectobauds
FontStyle:      TYPE = {modern, classic,  titan};
FontSize:       TYPE = {eight, ten, twelve, fourteen,
                        eighteen, twentyFour};
Underlining:    TYPE = {underline, italics, plain};
LineSpacing:    TYPE = {single, singleHalf, double, triple};
DropKeep:       TYPE = {drop, keep};
PageBreak:      TYPE = {drop, keep, unfilled};
GeneralTagSizes: TYPE = ARRAY GeneralItems OF CARDINAL ← ALL[0];
TextTagSizes:   TYPE = ARRAY TextItems OF CARDINAL ← ALL[0];
MapTagSizes:    TYPE = ARRAY MapItems OF CARDINAL ← ALL[0];
FontTagSizes:   TYPE = ARRAY FontItems OF CARDINAL ← ALL[0];



-- Procedures implemented in K4WindowImpl

OpenWindow: PROCEDURE [iconData: Containee.DataHandle,
                       changeProc: Containee.ChangeProc,
                       changeProcData: LONG POINTER,
                       tinyIcon: XString.Character]
             RETURNS [shell: SWS.Handle ← SWS.nullHandle];

PutFileInFolder: PROCEDURE [file: NSFile.Handle,
                            folderName: NSString.String];


-- Procedures implemented in K4PSheetImpl

OpenPSheet: PROCEDURE [iconData: Containee.DataHandle,
                       changeProc: Containee.ChangeProc,
                       changeProcData: LONG POINTER]
             RETURNS [SWS.Handle];

OpenDocOptionSheet: PROCEDURE [
                       convertParm: ConvertParms,
                       takeDown: PropertySheet.MenuItemProc];
```

```
-- Procedures implemented in K4FiledDataImpl

TypeAndVersion: PROCEDURE RETURNS [fileType: NSFile.Type,
                                   version:  Prototype.Version];

LoadFiledData:  PROCEDURE [parm: IconParms]
                  RETURNS [mismatch: BOOLEAN ← FALSE];

StoreFiledData: PROCEDURE [parm: IconParms];

FreeIconProps:  PROCEDURE [props: GeneralProperties,
                           z: UNCOUNTED ZONE];

FreeTextProps:  PROCEDURE [props: DocumentProperties,
                           z: UNCOUNTED ZONE];

FreeMapProps:   PROCEDURE [props: MapProperties,
                           z: UNCOUNTED ZONE];

FreeFontProps:  PROCEDURE [props: FontProperties,
                           z: UNCOUNTED ZONE];


-- Procedures implemented in K4DocumentImpl

ConvertToDocument: PROCEDURE[logFile: NSFile.Handle,
                             docProps: DocumentProperties,
                             mapProps: MapProperties,
                             fonProps: FontProperties];



END.  -- of K4
14-Jan-87 14:35:51 created from DestText and KDEM3.
28-Jan-87 16:01:53 changed parms of ConvertToViewPoint.
27-Feb-87 15:56:30 added preliminary ConvertToCanvas.
 3-Mar-87 11:05:03 unified parms for ConvertToViewPoint, ConvertToCanvas.
 4-Mar-87 15:55:03 added PutFileInFolder.
 9-Mar-87 14:08:49 added CanvasName, IconParmRecord, CanvasParmRecord.
 9-Mar-87 16:43:35 elimination of Courier and addition of options from linked property sheet. Changed
 iconFileType because of filed props changes.
13-Mar-87 10:12:24 added options and modified others.
16-Mar-87  9:43:17 version 9; added signature.
17-Mar-87 13:15:28 added character mapping.
18-Mar-87 11:04:54 defined OpenDocOptionSheet, OpenCanvasOptionSheet, and dropped all other option
sheet interfaces.
18-Mar-87 14:22:38 version 11: preset number to 0, not 1.
19-Mar-87 10:51:46 moved application file type and version into proc TypeAndVersion to avoid full
recompilation of all impls at every version change.
31-Mar-87 12:58:48 removed iconName from StoreFiledData.
 2-Apr-87 11:26:48 added parameter saveLog to ConvertTo...
 6-Apr-87 15:06:18 changed parameters of ConvertToCanvas, ConvertToDocument.
14-Apr-87 12:56:02 added paragraph line height and leadings, added drop/keep questionable marks.
24-Apr-87 16:08:12 dropped canvas and ArtScan processing, moved fonts to separate prop sheet.
27-Apr-87 12:56:49 added folder of transmisions in general items, and choice for page breaks.
/
```

```
-- File: K4Config.config - Last edit:
-- deLaBeaujardiere:OSBU North:Xerox       3-Aug-88 10:48:04

-- Copyright (C) 1986 by Xerox Corporation. All rights reserved.

K4Config: CONFIGURATION

IMPORTS Atom, Attention, BackgroundProcess,
        Containee, Display,
        DocInterchangeDefs, DocInterchangePropsDefs,
        FormWindow, Heap,
        MenuData, MessageWindow,
        NSFile, NSFileStream, NSString,
        Process, PropertySheet, Prototype,
        RS232C, Runtime, Selection, SimpleTextDisplay, SimpleTextFont,
        StarDesktop, StarWindowShell, StarWindowShellExtra2,
        Stream, XFormat, XString

CONTROL K4IconImpl =

BEGIN
K4FiledDataImpl;
K4IconImpl;
K4PSheetImpl;
K4WindowImpl;
K4DocumentImpl;
END.
<<
 4-Mar-87 10:08:27 added K4CanvasImpl and Interpress.
10-Mar-87 10:16:52 removed K4FiledDescription, added K4FiledData.
23-Mar-87 15:23:27 added MenuData, Selection.
23-Mar-87 17:00:53 removed CommonConversionImpl (merged in K4DocumentImpl).
 7-Apr-87 13:45:04 removed starting of K4WindowImpl.
24-Apr-87 16:55:21 removed K4CanvasImpl, Interpress, String.
 3-Aug-88 10:47:41 added DocInterchangePropsDefs.
/
>>
```

```
<< File: K4DocumentImpl.mesa      14-Sep-88 16:02:55

        Parses K4 output and produces structured data
        for input to CommonConversionImpl
        (which makes a ViewPoint document from the structured data)

Owner: Advanced Design - User Interfaces  - deLaBeaujardiere  >>


DIRECTORY Ascii, BackgroundProcess,
          DocInterchangeDefs, DocInterchangePropsDefs,
          FormWindow, Heap, K4, NSFile, NSFileStream,
          NSString,
          Process, PropertySheet,
          Runtime, StarFileTypes,
          StarWindowShell, Stream, Window,
          XChar, XCharSet0, XCharSet360, XString;

K4DocumentImpl: PROGRAM
  IMPORTS BackgroundProcess,
          DocInterchangeDefs, DocInterchangePropsDefs, FormWindow, Heap, K4,
          NSFile, NSFileStream, NSString,
          Process, PropertySheet, Runtime,
          Stream,
          XChar, XCharSet0, XCharSet360, XString
  EXPORTS K4
  SHARES  XString =
BEGIN
OPEN DI: DocInterchangeDefs, DIP: DocInterchangePropsDefs,
     FW: FormWindow, XS: XString;



Baseline:               TYPE = {null, super, sub};
QuestionHandling:       TYPE = {keep, dropCharacter, dropString};

Line: TYPE = LONG POINTER TO LineRecord;
LineRecord: TYPE = RECORD [
        next:           Line ← NIL,
        chunk:          Chunk ← NIL,
        interLine:      CARDINAL,       -- empty lines above
        leadingSpaces:  CARDINAL,
        paragraphHere:  BOOLEAN ← FALSE,
        text:           XS.WriterBody];

Chunk: TYPE = LONG POINTER TO ChunkRecord;
ChunkRecord: TYPE = RECORD [
        next:           Chunk ← NIL,
        aspect:         Aspect ← plain,
        firstChar:      CARDINAL ← 0,   -- pointer to beginning position
        nChars:         CARDINAL ← 0];  -- number of characters

Aspect: TYPE = MACHINE DEPENDENT {
                notAnAspect, plain,
                onSubscript, offSubscript,
                onSuperscript, offSuperscript,
                onUnderline, offUnderline,
                onFont0, (9),                   -- avoid TAB
                onFont1, onFont2, (12), (13),   -- avoid FF and CR
                onFont3, onFont4, onFont5, onFont6,
                onFont7, onFont8, onFont9,
                offFont0, offFont1, offFont2, offFont3,
                offFont4, offFont5, (27),       -- avoid ESC
                offFont6, offFont7, offFont8, offFont9};
                -- the above enumeration is to replace in the source
                -- stream the font/underline/subscript/.. markers
                -- with characters that cannot be accessed through
                -- the keyboard or are not K4 encodings.
                -- Thus the mapping options typed by
                -- the user cannot interfere with the markers.

ConversionHandle: TYPE = LONG POINTER TO ConversionData;
ConversionData: TYPE = RECORD [
        docHandle:      DI.Doc,
        fontProps:      DIP.FontPropsRecord,
        paraProps:      DIP.ParaPropsRecord,
```

```
        pageProps:      DIP.PagePropsRecord,
        tabProps:       ARRAY [0..maxTabs) OF DIP.TabStop];

WarningLines: TYPE = [0..10);


tab:            XChar.Character = XCharSet0.Make [tab];
space:          XChar.Character = XCharSet0.Make [space];
lastInSet0:     XChar.Character = XCharSet0.Make [lowerEng];
hyphen:         XChar.Character = XCharSet0.Make [minus];
questionMark:   XChar.Character = XCharSet0.Make [questionMark];
substitute:     XChar.Character = XCharSet360.Make [blackRectGraphic];
paraTab:        XChar.Character = XCharSet0.Make[LOOPHOLE[211C]];
pageEnd:        XChar.Character = XCharSet0.Make[LOOPHOLE[014C]];
lineEnd:        XChar.Character = XCharSet0.Make[LOOPHOLE[015C]];
propsBegin:     XChar.Character = XCharSet0.Make[LOOPHOLE['<.ORD]];
propsEnd:       XChar.Character = XCharSet0.Make[LOOPHOLE['>.ORD]];
  subscript:    XChar.Character = XCharSet0.Make[LOOPHOLE['i.ORD]];
  superscript:  XChar.Character = XCharSet0.Make[LOOPHOLE['s.ORD]];
  underline:    XChar.Character = XCharSet0.Make[LOOPHOLE['u.ORD]];
  fontStyle0:   XChar.Character = XCharSet0.Make[LOOPHOLE['0.ORD]];
  fontStyle1:   XChar.Character = XCharSet0.Make[LOOPHOLE['1.ORD]];
  fontStyle2:   XChar.Character = XCharSet0.Make[LOOPHOLE['2.ORD]];
  fontStyle3:   XChar.Character = XCharSet0.Make[LOOPHOLE['3.ORD]];
  fontStyle4:   XChar.Character = XCharSet0.Make[LOOPHOLE['4.ORD]];
  fontStyle5:   XChar.Character = XCharSet0.Make[LOOPHOLE['5.ORD]];
  fontStyle6:   XChar.Character = XCharSet0.Make[LOOPHOLE['6.ORD]];
  fontStyle7:   XChar.Character = XCharSet0.Make[LOOPHOLE['7.ORD]];
  fontStyle8:   XChar.Character = XCharSet0.Make[LOOPHOLE['8.ORD]];
  fontStyle9:   XChar.Character = XCharSet0.Make[LOOPHOLE['9.ORD]];
escape:         XChar.Character = XCharSet0.Make[LOOPHOLE[033C]];

oneInch:        CARDINAL = 1440;
spaceWidth:     CARDINAL = oneInch / 10;
maxTabs:        CARDINAL = 8;   -- 8 arbitrary tabs
                                -- half an inch apart
fourthInch:     CARDINAL = oneInch / 4;
fiveInches:     CARDINAL = oneInch * 5;
linesFor8Inches:        CARDINAL = 48;  -- 6 lines per inch X 8"
pointsPerLine:  CARDINAL = 12;




ConvertToDocument: PUBLIC PROCEDURE [
                            logFile: NSFile.Handle,
                            docProps: K4.DocumentProperties,
                            mapProps: K4.MapProperties,
                            fonProps: K4.FontProperties] =

  BEGIN
  -- This procedure copies the parameters in a record
  -- of its own, so that the original can be freed by the client.
  -- It then paints either an option sheet
  -- or a warning sheet requesting that Interpress be loaded.
  -- Control returns to client as soon as the sheet is painted.
  -- Zone and nodes acquired here are freed by ReleaseParm.

  ownZone: UNCOUNTED ZONE  ← Heap.Create[1];
  cp:      K4.ConvertParms ← ownZone.NEW[K4.ConvertParmsRecord];
  source, sink, map: K4.Mapping ← NIL;

  cp.zone ← ownZone;
  cp.logFile ← logFile;

  cp.docProps ← ownZone.NEW[K4.DocumentPropertyRecord];
  cp.docProps↑ ← docProps↑;
  cp.docProps.folderName ← XS.CopyToNewReaderBody[
                            @docProps.folderName, cp.zone];
  cp.docProps.docName ← XS.CopyToNewReaderBody[
                            @docProps.docName, cp.zone];
  cp.docProps.guessMark ← XS.CopyToNewReaderBody[
                            @docProps.guessMark, cp.zone];
  cp.docProps.others ← XS.CopyToNewReaderBody[
                            @docProps.others, cp.zone];
```

```
    cp.mapProps ← ownZone.NEW[K4.MapPropertyRecord];
    cp.mapProps.header ← XS.CopyToNewReaderBody[@mapProps.header, cp.zone];
    cp.mapProps.number ← mapProps.number;
    source ← mapProps.map;
    FOR k: CARDINAL IN [1..mapProps.number] DO
      map ← cp.zone.NEW [K4.MappingRecord ← [
                  from: XS.CopyToNewReaderBody[@source.from, cp.zone],
                  to:   XS.CopyToNewReaderBody[@source.to, cp.zone],
                  next: NIL]];
      IF cp.mapProps.map = NIL
          THEN cp.mapProps.map ← map
          ELSE sink.next ← map;
      sink ← map;
      source ← source.next;
      ENDLOOP;

    cp.fonProps ← ownZone.NEW[K4.FontPropertyRecord];
    cp.fonProps↑ ← fonProps↑;

    IF Runtime.IsBound [LOOPHOLE [DI.StartCreation]]
      THEN K4.OpenDocOptionSheet [cp, TakeDownSheet]
      ELSE ShowWarning [cp];
    END;  -- of ConvertToDocument


  TakeDownSheet: PropertySheet.MenuItemProc =
    BEGIN
    cp: K4.ConvertParms ← LOOPHOLE[clientData];
    IF menuItem = start
      THEN Process.Detach [FORK ConvertProcess[cp]]
      ELSE ReleaseParm [cp];
    ok ← TRUE;
    END;  -- of TakeDownSheet


  ShowWarning: PROC [cp: K4.ConvertParms] =
    BEGIN
    pSheetName: XS.ReaderBody ← XS.FromSTRING ["Kurzweil Converter"L];
    pSize: Window.Dims ← [420, 300];
    pPlace: Window.Place ← [550, 100];

    [] ← PropertySheet.Create [formWindowItems: MakeWarning,
                               formWindowItemsLayout: LayoutWarning,
                               menuItemProc: ExitWarning,
                               menuItems: [done: TRUE, cancel: TRUE],
                               title: @pSheetName,
                               size: pSize,
                               placeToDisplay: pPlace,
                               clientData: cp];
    END;  -- of ShowWarning


  MakeWarning: FW.MakeItemsProc =
    BEGIN
    txt: ARRAY WarningLines OF LONG STRING ← [
        "VP Editor is needed to make a document, but is not running."L,
        "Please run the following applications (if they are idle,"L,
        "then press <Done>:"L,
        "      Font Manager"L,
        "      WorkStation Keyboards"L,
        "      Keyboards"L,
        "      Interscript Converter"L,
        "      VP Document Editor"L,
        "  "L,
        "If want to drop the document creation, press <Cancel>."L];
    msg: XString.ReaderBody;

    FOR k: WarningLines IN WarningLines DO
      msg ← XString.FromSTRING[txt[k]];
      FW.MakeTextItem [window: window, myKey: k,
                       tag: NIL, readOnly: TRUE, boxed: FALSE,
                       width: 400, initString: @msg];
      ENDLOOP;
    END;  -- of MakeWarning
```

```
LayoutWarning: FW.LayoutProc =
  BEGIN
  FOR k: WarningLines IN WarningLines DO
    line: FW.Line ← FW.AppendLine [window: window, spaceAboveLine: 0];
    FW.AppendItem [window: window, item: k,
                   line: line, preMargin: 10];
    ENDLOOP;
  END;  -- of LayoutWarning


ExitWarning: PropertySheet.MenuItemProc =
  BEGIN
  cp: K4.ConvertParms ← LOOPHOLE[clientData];
  newMsg: XString.ReaderBody ← XString.FromSTRING[
        "The Document Editor is still not running..."L];

  IF menuItem = cancel THEN
    BEGIN
    ReleaseParm [cp];
    RETURN [TRUE];
    END;
  IF Runtime.IsBound[LOOPHOLE[DI.StartCreation]] THEN
    BEGIN
    K4.OpenDocOptionSheet [cp, TakeDownSheet];
    RETURN [TRUE];
    END;

  FW.SetTextItemValue [formWindow, 0, @newMsg, TRUE];
  RETURN [FALSE];
  END;  -- of ExitWarning


ConvertProcess: PROCEDURE [cp: K4.ConvertParms] =
  BEGIN
  processName: XS.ReaderBody ← XS.FromSTRING["Making Document"L];

  ManagedProcess: BackgroundProcess.CallBackProc =
                  -- ManagedProcess must be within ConvertProcess
                  -- to be supervised by the Background Processor
  BEGIN
  ENABLE UNWIND => {finalStatus ← aborted; CONTINUE};

  zone:                   UNCOUNTED ZONE ← cp.zone;
  rawText:                XS.WriterBody;
  firstLine, currentLine: Line ← NIL;
  line:                   Line;
  map:                    K4.Mapping ← NIL;
  maxMapLength:           CARDINAL ← 0;
  leadingSpaces:          CARDINAL;
  emptyLinesAbove:        CARDINAL ← 0;
  conversionHandle:       ConversionHandle;
  docFile:                NSFile.Handle;
  docSession:             NSFile.Session;
  status:                 DI.FinishCreationStatus;
  docNSName, folderNSName: NSString.String;
  docAttributes:          ARRAY [0..1) OF NSFile.Attribute;
  logStream:              Stream.Handle;
  hasProps, endOfPage, endOfStream: BOOLEAN ← FALSE;
  qHandling:              QuestionHandling;
  qCharacter:             XS.Character;    -- accelerator
  qString:                XS.Reader;       -- accelerator

  conversionHandle ← BeginDocument [zone, cp.docProps, cp.fonProps];
  IF conversionHandle = NIL THEN RETURN;          -- creation failed

  IF  (cp.mapProps # NIL)
  AND (cp.mapProps.map # NIL)
  AND (NOT XS.Empty [@cp.mapProps.map.from]) THEN
        [map, maxMapLength] ← OrderMap [cp.mapProps.map];

  qString ← @cp.docProps.guessMark;
  SELECT TRUE FROM
    cp.docProps.dropKeepMark = keep => qHandling ← keep;
    XS.CharacterLength[qString] = 1 =>
        BEGIN
```

```
        qHandling ← dropCharacter;
        qCharacter ← XS.Lop[qString];
        END;
    ENDCASE => qHandling ← dropString;

  logStream ← NSFileStream.Create [cp.logFile, FALSE];
  Stream.SetInputOptions [logStream, [signalEndOfStream: TRUE]];

  rawText ← XS.NewWriterBody [300, zone];
                        -- large enough to contain a whole line
                        -- in small font. Will be expanded if needed.

  DO
    -- get a raw line of text
    [leadingSpaces, hasProps, endOfPage, endOfStream] ← FillBuffer [
                                logStream, @rawText,
                                qHandling, qCharacter, qString];

    IF XS.Empty [XS.ReaderFromWriter[@rawText]] THEN
      emptyLinesAbove ← emptyLinesAbove + 1
    ELSE
      BEGIN
      line ← zone.NEW [LineRecord];
      IF firstLine = NIL THEN currentLine ← firstLine ← line
      ELSE currentLine ← currentLine.next ← line;
      line.leadingSpaces ← leadingSpaces;
      line.interLine ← emptyLinesAbove;
      emptyLinesAbove ← 0;
      line.text ← XS.CopyToNewWriterBody[
                              XS.ReaderFromWriter[@rawText], zone];
      IF map # NIL THEN Map [XS.ReaderFromWriter[@rawText], @line.text, map];
      IF hasProps THEN Parcel [line, zone];
      END;

    IF endOfPage THEN -- end of page found after current line
      BEGIN
      linesInPage: CARDINAL ← FillDocument [conversionHandle, firstLine,
                                            cp.fonProps,
                                            cp.docProps.underlining];
      SELECT cp.docProps.pageBreak FROM
        drop     => NULL;
        keep     => DI.AppendPageBreak [conversionHandle.docHandle,
                                        @conversionHandle.fontProps];
        unfilled => IF linesInPage < linesFor8Inches THEN
                      DI.AppendPageBreak [conversionHandle.docHandle,
                                          @conversionHandle.fontProps];
        ENDCASE;
      ReleaseLines [firstLine, zone];
      firstLine ← NIL;
      emptyLinesAbove ← 0;
      END;

    IF endOfStream THEN EXIT;    -- end of stream found after current line

    ENDLOOP;                     -- loop to get next raw line

  [] ← FillDocument [conversionHandle, firstLine,
                     cp.fonProps, cp.docProps.underlining];
  ReleaseLines [firstLine, zone];

  [docFile, docSession, status] ← DI.FinishCreation[@conversionHandle.docHandle];

  IF docFile # NSFile.nullHandle THEN
    BEGIN       -- reopen docFile in null session
    ENABLE NSFile.Error =>
            {
            NSFile.Close[docFile, docSession ! NSFile.Error => CONTINUE];
            docFile ← NSFile.nullHandle;
            CONTINUE;
            };
    tmpRef: NSFile.Reference ← NSFile.GetReference[file: docFile,
                                                   session: docSession];
    tmpFile: NSFile.Handle ← docFile;
    docFile ← NSFile.OpenByReference[reference: tmpRef];
    NSFile.Close[tmpFile, docSession];
    NSFile.Logoff[docSession ! NSFile.Error => CONTINUE];
```

```
    END;

  docNSName ← XS.NSStringFromReader [
                        @cp.docProps.docName, zone];
  docAttributes[0] ← [name[docNSName]];
  NSFile.ChangeAttributes [docFile, DESCRIPTOR [docAttributes]];
  NSString.FreeString [zone, docNSName];

  folderNSName ← XS.NSStringFromReader [
                        @cp.docProps.folderName, zone];
  K4.PutFileInFolder [docFile, folderNSName];
  NSString.FreeString [zone, folderNSName];
  NSFile.Close [docFile];

  Stream.Delete [logStream];
  XS.FreeWriterBytes [@rawText];
  ReleaseParm [cp];      -- done last, beause it releases the zone
  END;  -- of ManagedProcess

  Process.SetPriority[Process.priorityBackground];
  [] ← BackgroundProcess.ManageMe[name: @processName,
                                  callBackProc: ManagedProcess,
                                  abortable: FALSE];  --** TRUE later
  END;  -- of ConvertProcess


FillBuffer: PROC [stream:        Stream.Handle,
                  bufferW:       XS.Writer,
                  qHandling:     QuestionHandling,
                  qCharacter:    XS.Character,
                  qString:       XS.Reader]
        RETURNS [leadingSpaces: CARDINAL ← 0,
                  hasProps:      BOOLEAN ← FALSE,
                  endOfPage:     BOOLEAN ← FALSE,
                  endOfStream:   BOOLEAN ← FALSE] =
  BEGIN
  -- Gets a line of text in buffer, up to next CR or FF.
  -- Drops the leading spaces.
  -- Replace strings of 3 or more spaces by a tab.
  -- Replaces props marker by ESC+code so that they cannot be
  -- changed by user mapping options.
  ENABLE Stream.EndOfStream => GOTO EndStream;
  xChar, yChar: XChar.Character;
  aspect: Aspect;
  insideSpaces, k: CARDINAL;

  NextChar: PROC RETURNS [XS.Character] = INLINE
        {RETURN [XCharSet0.Make[LOOPHOLE[Stream.GetByte [stream]]]]};

  XS.ClearWriter [bufferW];
  insideSpaces ← 0;

  DO            -- count and drop leading spaces
    xChar ← NextChar[];
    IF xChar # space THEN EXIT;
    leadingSpaces ← leadingSpaces + 1;
    ENDLOOP;

  DO            -- get remainder of line
    SELECT TRUE FROM
      xChar = lineEnd => RETURN;
      xChar = pageEnd => {endOfPage ← TRUE; RETURN};
      qHandling = dropCharacter
      AND xChar = qCharacter => NULL;
      xChar = propsBegin =>      -- replace begin-props markers
                BEGIN
                yChar ← NextChar[];
                aspect ← SELECT yChar FROM
                  underline => onUnderline,
                  subscript => onSubscript,
                  superscript => onSuperscript,
                  fontStyle0 => onFont0,
                  fontStyle1 => onFont1,
                  fontStyle2 => onFont2,
                  fontStyle3 => onFont3,
                  fontStyle4 => onFont4,
```

```
                    fontStyle5 => onFont5,
                    fontStyle6 => onFont6,
                    fontStyle7 => onFont7,
                    fontStyle8 => onFont8,
                    fontStyle9 => onFont9,
                    ENDCASE => notAnAspect;
                  IF aspect = notAnAspect THEN
                    {XS.AppendChar [bufferW, xChar];
                     XS.AppendChar [bufferW, yChar]}
                  ELSE
                    {XS.AppendChar [bufferW, escape];
                     XS.AppendChar [bufferW,
                              XCharSet0.Make[LOOPHOLE[aspect]]];
                     hasProps ← TRUE};
                  END;
        xChar = propsEnd =>       -- replace end-props markers
                  BEGIN
                  yChar ← NextChar[];
                  aspect ← SELECT yChar FROM
                  underline => offUnderline,
                    subscript => offSubscript,
                    superscript => offSuperscript,
                    fontStyle0 => offFont0,
                    fontStyle1 => offFont1,
                    fontStyle2 => offFont2,
                    fontStyle3 => offFont3,
                    fontStyle4 => offFont4,
                    fontStyle5 => offFont5,
                    fontStyle6 => offFont6,
                    fontStyle7 => offFont7,
                    fontStyle8 => offFont8,
                    fontStyle9 => offFont9,
                    ENDCASE => notAnAspect;
                  IF aspect = notAnAspect THEN
                   {XS.AppendChar [bufferW, xChar];
                    XS.AppendChar [bufferW, yChar]}
                  ELSE
                   {XS.AppendChar [bufferW, escape];
                    XS.AppendChar [bufferW,
                              XCharSet0.Make[LOOPHOLE[aspect]]];
                    hasProps ← TRUE};
                  END;
        xChar = space => insideSpaces ← insideSpaces + 1;
                                -- count embedded spaces
        xChar IN (space..lastInSet0] =>
                  BEGIN
                          -- replace space strings by tab or space
                  SELECT insideSpaces FROM
                    0     => NULL;
                    > 5   => XS.AppendChar [bufferW, tab];
                    ENDCASE =>
                      BEGIN
                      FOR k IN [1..insideSpaces] DO
                        XS.AppendChar [bufferW, space];
                        ENDLOOP;
                      END;
                  XS.AppendChar [bufferW, xChar];
                  insideSpaces ← 0;
                  END;
        xChar = tab =>
                  XS.AppendChar [bufferW, xChar];
        ENDCASE =>
                  XS.AppendChar [bufferW, substitute];
        -- ** handle questionable char "dropString" later

      xChar ← NextChar [];
      ENDLOOP;
    EXITS EndStream => {endOfStream ← TRUE};

    END;  -- of FillBuffer


Parcel: PROC [line: Line, zone: UNCOUNTED ZONE] =
  BEGIN
  -- Font or aspect markers found in line.text generate
  -- a linked list of "chunks".
```

```
  -- There is at least one marker when we enter this proc.

  code:         XChar.Character;
  index:        CARDINAL ← 0;
  source:       XS.Reader ← XS.ReaderFromWriter[@line.text];
  newChunk:     Chunk ← NIL;
  currentChunk: Chunk ← zone.NEW [ChunkRecord];

  line.chunk ← currentChunk;

  WHILE index < XS.ByteLength[source] DO         -- parse source text
    code ← XS.NthCharacter [source, index];
    IF code # escape THEN
      BEGIN
      index ← index + 1;
      END
    ELSE
      BEGIN
      currentChunk.nChars ← index - currentChunk.firstChar;
      -- Make a new chunk, unless the current chunk is at beginning of line.
      IF index > 0 THEN
        BEGIN
        newChunk ← zone.NEW [ChunkRecord];
        currentChunk.next ← newChunk;
        currentChunk ← newChunk;
        END;
      code ← XS.NthCharacter [source, index + 1];
      currentChunk.aspect ← LOOPHOLE[XChar.Code[code], Aspect];
      index ← index + 2;
      currentChunk.firstChar ← index;
      END;
    ENDLOOP;
  currentChunk.nChars ← index - currentChunk.firstChar;
  END;  -- of Parcel


ReleaseLines: PROC [firstLine: Line, z: UNCOUNTED ZONE] =
  BEGIN
  line: Line ← firstLine;
  holdLine: Line ← NIL;
  chunk, holdChunk: Chunk ← NIL;

  DO
    IF line = NIL THEN RETURN;
    chunk ← line.chunk;
    DO
      IF chunk = NIL THEN EXIT;
      holdChunk ← chunk.next;
      z.FREE [@chunk];
      chunk ← holdChunk;
      ENDLOOP;
    XS.FreeWriterBytes [@line.text];
    holdLine ← line.next;
    z.FREE [@line];
    line ← holdLine;
    ENDLOOP;
  END;  -- of ReleaseLines


ReleaseParm: PROC [cp: K4.ConvertParms] =
  BEGIN
  z: UNCOUNTED ZONE ← cp.zone;

  NSFile.Close [cp.logFile];
  K4.FreeTextProps [cp.docProps, z];
  K4.FreeMapProps [cp.mapProps, z];
  K4.FreeFontProps [cp.fonProps, z];
  z.FREE [@cp];
  Heap.Delete [z];
  END;  -- of ReleaseParm


BeginDocument: PROCEDURE [z: UNCOUNTED ZONE,
                          docProps: K4.DocumentProperties,
                          fonProps: K4.FontProperties]
```

```
                RETURNS [document: LONG POINTER] =
  BEGIN
  h: ConversionHandle ← z.NEW [ConversionData];

  DIP.GetPagePropsDefaults[@h.pageProps];

  DIP.GetFontPropsDefaults[@h.fontProps];
  h.fontProps.fontDesc.pointSize ←
                SELECT fonProps.fonts[0].size FROM
                                        twelve  => 12,
                                        ten     => 10,
                                        eight   =>  8,
                                        fourteen => 14,
                                        ENDCASE => 12;
  h.fontProps.fontDesc.weight ← IF fonProps.fonts[0].bold
                THEN bold ELSE medium;
  h.fontProps.fontDesc.designVariant ← IF fonProps.fonts[0].italics
                THEN italic ELSE roman;
  h.fontProps.fontDesc.family ←
                SELECT fonProps.fonts[0].font FROM
                                        classic => century,
                                        modern  => frutiger,
                                        ENDCASE => titan;

  DIP.GetParaPropsDefaults[@h.paraProps];
  h.paraProps.basicProps.preLeading ← SELECT docProps.preLeading FROM
                                        single => 0,
                                        singleHalf => 6,
                                        double => 12,
                                        ENDCASE => 24;
  h.paraProps.basicProps.postLeading ← SELECT docProps.postLeading FROM
                                        single => 0,
                                        singleHalf => 6,
                                        double => 12,
                                        ENDCASE => 24;
  h.paraProps.basicProps.lineHeight ← SELECT docProps.lineHeight FROM
                                        single => 12,
                                        singleHalf => 18,
                                        double => 24,
                                        ENDCASE => 36;
  h.paraProps.basicProps.justified ← docProps.justification;
  h.paraProps.tabStops ← DESCRIPTOR [h.tabProps];

  FOR k: CARDINAL IN [0..maxTabs) DO              -- set 12 tabs at 1/2 inch
    h.tabProps[k].dotLeader ← FALSE;
    h.tabProps[k].tabStopOffset ← 36 * k;         -- 72 points/inch
    h.tabProps[k].tabStopAlignment ← left;
    ENDLOOP;

  [h.docHandle,,,,,] ← DI.StartCreation [
                        simple, -- simple pagination
                        FALSE,  -- no header
                        FALSE,  -- no footer
                        @h.fontProps,
                        @h.paraProps,
                        @h.pageProps];
  document ← h;
  END;  -- of BeginDocument


FillDocument: PROC [h: ConversionHandle,
                    lineList: Line,
                    fontOptions: K4.FontProperties,
                    underOption: K4.Underlining]
                    RETURNS [linesInPage: CARDINAL ← 0] =
  BEGIN
  partialBody:  XS.ReaderBody;
  fullReader:   XS.Reader;
  context:      XS.Context ← XS.vanillaContext;
  line:         Line;
  lastChar:     XChar.Character;

  IF lineList = NIL THEN RETURN;

  MarkParagraphs [lineList, FALSE];
                -- Find the paragraph boundaries
```

```
  FOR line ← lineList, line.next WHILE line # NIL DO

    linesInPage ← linesInPage + 1 + line.interLine;
        -- count to see where is the last line in the page;
        -- if it lies "far up enough" from the bottom
        -- (arbitrarily, 3" from bottom, 8" from top),
        -- we will issue a page break.

    IF line.paragraphHere THEN
      BEGIN
      h.fontProps.nUnderlines ← 0;  -- turn off underline at
                                    -- beginning of a paragraph
      FOR k: CARDINAL IN [2..line.interLine) DO  -- make white space
        DI.AppendNewParagraph [[doc[h.docHandle]],
                               @h.paraProps,
                               @h.fontProps];
        ENDLOOP;
      DI.AppendNewParagraph [[doc[h.docHandle]],
                             @h.paraProps, @h.fontProps];
      END;

    fullReader ← XS.ReaderFromWriter[@line.text];
    IF line.chunk = NIL THEN
      BEGIN
      DI.AppendText [[doc[h.docHandle]], fullReader,
                     XS.vanillaContext, @h.fontProps];
      END
    ELSE
      BEGIN
      FOR chunk: Chunk ← line.chunk, chunk.next WHILE chunk # NIL DO
        SetAspect [chunk, fontOptions, underOption, @h.fontProps];
        IF chunk.nChars > 0 THEN
          BEGIN
          [partialBody, context] ← XS.Piece [fullReader,
                                             chunk.firstChar,
                                             chunk.nChars];
          DI.AppendText [[doc[h.docHandle]],
                         @partialBody, context, @h.fontProps];
          END;
        ENDLOOP;  -- loop to process next chunk in line
      END;
    -- Put a space after the last character of the line
    -- unless there is already a space or an hyphen.
    lastChar ← XS.NthCharacter [fullReader,
                                XS.CharacterLength[fullReader] - 1];
    IF lastChar # space AND lastChar # hyphen THEN
      DI.AppendChar [[doc[h.docHandle]], space, @h.fontProps];
    ENDLOOP;  -- loop to process next line

  END;  -- of FillDocument


SetAspect: PROC [chunk: Chunk,
                 fontOptions: K4.FontProperties,
                 underOption: K4.Underlining,
                 vpFont: DIP.FontProps] =
  BEGIN
  -- Given a chunk of text with an aspect code,
  -- and given the font choices made by the user,
  -- adjust a property of the VP font as required.

  SetFont: PROC [fontChoice: K4.FontRecord] =
        BEGIN
        vpFont.fontDesc.family ← SELECT fontChoice.font FROM
                                 classic => century,
                                 modern  => frutiger,
                                 titan   => titan,
                                 ENDCASE => century;
        vpFont.fontDesc.weight ← IF fontChoice.bold
                                 THEN bold ELSE medium;
        vpFont.fontDesc.designVariant ← IF fontChoice.italics
                                 THEN italic ELSE roman;
        vpFont.fontDesc.pointSize ← SELECT fontChoice.size FROM
                                 twelve   => 12,
                                 ten      => 10,
```

```
                                eight    => 8,
                                fourteen => 14,
                                ENDCASE  => 12;
        IF fontChoice.font = titan THEN
          BEGIN
          vpFont.fontDesc.designVariant ← roman; -- in case it was italics
          IF vpFont.fontDesc.pointSize < 10 THEN
                                vpFont.fontDesc.pointSize ← 10;
          IF vpFont.fontDesc.pointSize > 12 THEN
                                vpFont.fontDesc.pointSize ← 12;
          END;
        END;  -- of SetFont

    SELECT chunk.aspect FROM
      onSubscript    => vpFont.placement ← sub;
      offSubscript   => vpFont.placement ← null;
      onSuperscript  => vpFont.placement ← super;
      offSuperscript => vpFont.placement ← null;
      onUnderline    => SELECT underOption FROM
                          underline => vpFont.nUnderlines ← 1;
                          italics   => {vpFont.nUnderlines ← 0;
                                        vpFont.fontDesc.designVariant ← italic};
                          plain     => vpFont.nUnderlines ← 0;
                          ENDCASE;
      offUnderline   => SELECT underOption FROM
                          underline => vpFont.nUnderlines ← 0;
                          italics   => {vpFont.nUnderlines ← 0;
                                        vpFont.fontDesc.designVariant ← roman};
                          plain     => vpFont.nUnderlines ← 0;
                          ENDCASE;
      onFont0        => SetFont [fontOptions.fonts[0]];
      onFont1        => SetFont [fontOptions.fonts[1]];
      onFont2        => SetFont [fontOptions.fonts[2]];
      onFont3        => SetFont [fontOptions.fonts[3]];
      onFont4        => SetFont [fontOptions.fonts[4]];
      onFont5        => SetFont [fontOptions.fonts[5]];
      onFont6        => SetFont [fontOptions.fonts[6]];
      onFont7        => SetFont [fontOptions.fonts[7]];
      onFont8        => SetFont [fontOptions.fonts[8]];
      onFont9        => SetFont [fontOptions.fonts[9]];
      ENDCASE        => SetFont [fontOptions.fonts[0]];  -- restore font 0

    END;  -- of SetAspect


Map: PROC [fromR: XS.Reader, toW: XS.Writer, mapOptions: K4.Mapping] =
    BEGIN
    fromIndex: CARDINAL ← 0;
    fromMapLg: CARDINAL;
    piece: XS.ReaderBody;
    context: XS.Context;  -- not used...

    XS.ClearWriter [toW];

    WHILE fromIndex < XS.CharacterLength [fromR] DO
      FOR map: K4.Mapping ← mapOptions, map.next WHILE map # NIL DO
        fromMapLg ← XS.CharacterLength [@map.from];
        [piece, context] ← XS.Piece [fromR, fromIndex, fromMapLg];
        IF XS.Equal [@piece, @map.from] THEN
          BEGIN
          XS.AppendReader [toW, @map.to];
          fromIndex ← fromIndex + fromMapLg;
          EXIT;
          END;
        REPEAT
        FINISHED => BEGIN -- no mapping match
                    XS.AppendChar [toW, XS.NthCharacter [fromR, fromIndex]];
                    fromIndex ← fromIndex + 1;
                    END;
        ENDLOOP;
      ENDLOOP;

    END;  -- of Map


OrderMap: PROC [map: K4.Mapping] RETURNS [K4.Mapping, CARDINAL] =
```

```
  BEGIN
  -- This procedure sorts the mapping entries by string length,
  -- placing the longest string first.
  -- The sort technique is a bit crude, because we don't have
  -- back pointers, but the lists are generally short.

  prior, current, next, hold: K4.Mapping ← NIL;
  entryHasMoved: BOOLEAN ← FALSE;

  IF map = NIL THEN RETURN [NIL, 0];

  <<** not working correctly yet
  entryHasMoved ← TRUE;
  WHILE entryHasMoved DO
    entryHasMoved ← FALSE;
    prior ← current ← map;
    next ← map.next;
    DO
      IF next = NIL OR current = NIL THEN EXIT;
      IF XS.ByteLength [@next.from] >
         XS.ByteLength [@current.from] THEN
        BEGIN
        IF current = prior THEN
          BEGIN
          map ← next;
          map.next ← current;
          current.next ← next.next;
          END
        ELSE
          BEGIN
          hold ← prior.next;
          prior.next ← current.next;
          current.next ← next.next;
          next.next ← hold;
          END;
        entryHasMoved ← TRUE;
        EXIT;
        END;
      prior ← prior.next;
      current ← current.next;
      next ← next.next;
      ENDLOOP;
    ENDLOOP;
  **>>
  RETURN [map, 10];      --**10 is temp...
  END;  -- of OrderMap


MarkParagraphs: PROC [firstLine: Line,
                      keepLineBreak: BOOLEAN] =  --** unused for now
  BEGIN
  -- We think there is a paragraph if one of the following is true:
  -- 1) the line has an interval (i.e distance from line above)
  --    greater than the smallest line interval;
  -- 2) the line above is "clearly" shorter than the current line,
  --    ("clearly" arbitrarily defined as 1 inch),
  -- 3) the line is "clearly" indented from the prior line's,
  --    ("clearly" arbitrarily defined as 1/4 inch).

  line: Line;
  priorMargin, smallestMargin: CARDINAL ← LAST[CARDINAL];
  priorLength, lineLength: CARDINAL;
  smallestInterline: CARDINAL ← LAST[CARDINAL];
  LineLength: PROC [r: XS.Reader] RETURNS [length: CARDINAL] =
        BEGIN
        length ← XS.CharacterLength [r] * oneInch / 10;
        -- Grossly approximative. Need better way...
        END;  -- of LineLength

  -- If we must keep line breaks, no point looking further...
  IF keepLineBreak THEN
    BEGIN
    FOR line ← firstLine, line.next WHILE line # NIL DO
      line.paragraphHere ← TRUE;
      ENDLOOP;
    RETURN;
```

```
    END;

  -- Find the smallest line interval, to tell us if
  -- the text is generally single-spaced, or double-spaced, etc..
  -- and find the smallest left margin.
  FOR line ← firstLine, line.next WHILE line # NIL DO
    smallestInterline ← MIN [smallestInterline, line.interLine];
    smallestMargin ← MIN [smallestMargin, line.leadingSpaces];
    ENDLOOP;

  priorLength ← LineLength [XS.ReaderFromWriter[@firstLine.text]];

  FOR line ← firstLine.next, line.next WHILE line # NIL DO
    lineLength ← LineLength [XS.ReaderFromWriter[@line.text]];
    line.paragraphHere ←
                (line.interLine > smallestInterline)
             OR (priorLength + oneInch < lineLength)
             OR (line.leadingSpaces > priorMargin + fourthInch)
             OR (line.paragraphHere);   -- already a paragraph
    priorLength ← lineLength;
    priorMargin ← line.leadingSpaces;
    ENDLOOP;

  END;  -- of MarkParagraphs



END.  -- of K4DocumentImpl
 5-Feb-87 12:48:28 upgraded to new version of CommonConversion, and to new parameter for
 ConvertToDocument.
10-Feb-87 14:05:39 added auto creation of folder, movement of document into folder.
18-Feb-87 17:51:47 added NIL header/footer parameters for BeginDocument.
 4-Mar-87  9:48:28 adapted to new common parameters for ConvertToDocument and ConvertToCanvas.
 4-Mar-87 13:55:18 added presetting of fontChoices to user options.
10-Mar-87 15:06:03 adapted to linked pSheet.
11-Mar-87 17:41:42 moved here from K4WindowImpl the test for Editor loaded.
14-Mar-87 10:25:11 forgot to call SetTextOptions.
18-Mar-87 11:02:41 replaced all option sheet logic by call to OpenDocOptionSheet.
19-Mar-87 10:20:41 released mapProps in ReleaseParm.
23-Mar-87 16:01:01 forgot to test for "cancel" in option sheet.
23-Mar-87 16:14:13 merged in ComonConversionImpl to handle string substitutions.
26-Mar-87 11:29:57 increased width of warning text.  Redesigned entire line/subLines concept, replaced
with line/pieces, dropped String interface and used XString for buffer.
 2-Apr-87 12:43:03 named logFile according to document name.
 3-Apr-87 11:06:45 implemented mapping.
 5-Apr-87 11:23:54 was quitting on blank lines.
 5-Apr-87 11:48:11 characters outside [space..376B] not caught.
 6-Apr-87 10:33:28 upgraded paragraph recognition procedure, removed leading spaces.
 6-Apr-87 15:26:20 creation of Convertparms moved here from K4WindowImpl.
 7-Apr-87 14:30:31 moved out saving of logFile; had problem with holding same handle in two processes.
 7-Apr-87 17:25:45 reduced interline in warning window.
13-Apr-87 14:41:18 checked for empty fist map.from; release bytes of bufferWB.
14-Apr-87 11:27:58 released resources when cancelling document creation.
14-Apr-87 13:13:16 added pre/post/lineHeight.
20-Apr-87 13:53:25 picked up initial font size/style/slant/weight from docProps.
24-Apr-87 16:47:32 new parameter fonProps, fonts moved out of docProps.
27-Apr-87 13:44:28 page break choices.
28-Apr-87 18:22:02 underlining vs italics vs plain choice.
 8-May-87 13:22:14 looks like automatic expansion of bufferW did not work; added code to expand it
 manually.
 8-May-87 13:23:18 when font aspect change occurs after first character, all chars between first and
 the aspect change are dropped; extensive change, including new design of the way to translate the
 input text according to the mapping options.
14-May-87  9:31:20 insured that titan font is not italics and in [10, 12].
23-Mar-88 15:18:13 mapped tab into tab (was changed to substitute before); fixed the algorithm
assigning a paragraph mark on certain lines (it was making a paragraph of the last line of real
paragraphs!); made paragraphs of lines less than 5 inches long.
 5-Apr-88 14:36:41 when a string of spaces are found inside a line, replace it with a tab if there are
 more than 5 spaces (rather than 3 spaces); removed the criteria linelength < 5 inches for paragraph
 decision (caused problems on documents with narrow paragraphs).
 3-Aug-88 10:03:47 upgraded to BWS4.3/VP2.0: numerous changes in DocInterchangeDefs and company; added
 session support because of DocInterchangeDefs.FinishCreation.
/
```

```
<< File: K4FiledDataImpl.mesa            27-Apr-87 13:00:46

deLaBeaujardiere:OSBU North:Xerox (deLaBeaujardiere.PA)

Copyright (C) 1986 by Xerox Corporation. All rights reserved.

>>

DIRECTORY Containee, Environment, K4, NSFile, NSFileStream,
          Prototype, StarWindowShell, Stream, XString;

K4FiledDataImpl: PROGRAM

IMPORTS NSFileStream, Stream, XString
EXPORTS K4 =

BEGIN
OPEN XS: XString;


<<==========  PUBLIC PROCEDURES  ==========>>



TypeAndVersion: PUBLIC PROCEDURE
                RETURNS [fileType: NSFile.Type,
                         version:  Prototype.Version] =
  BEGIN
  RETURN [7389325,       -- randomly-chosen number for file type
          19];           -- current version of the application
  END;                   -- The file type must be assigned an
                         -- official number at productization.


LoadFiledData: PUBLIC PROCEDURE [parm: K4.IconParms]
        RETURNS [mismatch: BOOLEAN ← FALSE] =
  BEGIN
  -- This procedure loads into parm the names and values
  -- recorded in the icon file.  Memory allocated here to hold
  -- the bytes for icon/folder/document names
  -- must be freed by the client.

  z: UNCOUNTED ZONE ← parm.heap;
  map, currentMap: K4.Mapping ← NIL;
  signature: CARDINAL ← Signature [];
  stream: Stream.Handle ← NSFileStream.Create [parm.iconFile, FALSE];

  Stream.SetPosition [stream, 0];

  BEGIN ENABLE Stream.EndOfStream => {mismatch ← TRUE;
                                     GOTO Termination};
                                  -- we should not run out of stream
  -- 1. Signature.
  parm.signature ← Stream.GetWord [stream];
  IF parm.signature # signature THEN {mismatch ← TRUE;
                                     GOTO Termination};

  -- 2. General Properties
  parm.genProps ← parm.heap.NEW [K4.GeneralPropertyRecord];
  parm.genProps.iconName ← LoadReaderBody [stream, z];
  parm.genProps.channelSpeed ← VAL[LoadEnumerated[stream]];
  parm.genProps.folderName ← LoadReaderBody [stream, z];
  parm.genProps.others ← LoadReaderBody [stream, z];

  -- 3. Document Properties
  parm.docProps ← parm.heap.NEW [K4.DocumentPropertyRecord];
  parm.docProps.folderName ← LoadReaderBody [stream, z];
  parm.docProps.docName ← LoadReaderBody [stream, z];
  parm.docProps.justification ← LoadBoolean [stream];
  parm.docProps.lineHeight ← VAL[LoadEnumerated [stream]];
  parm.docProps.preLeading ← VAL[LoadEnumerated [stream]];
  parm.docProps.postLeading ← VAL[LoadEnumerated [stream]];
  parm.docProps.underlining ← VAL[LoadEnumerated [stream]];
  parm.docProps.guessMark ← LoadReaderBody [stream, z];
  parm.docProps.dropKeepMark ← VAL[LoadEnumerated [stream]];
  parm.docProps.pageBreak ← VAL[LoadEnumerated [stream]];

K4FiledDataImpl.mesa        27-Apr-87 13:00:48 PDT                    1
```

```
  parm.docProps.others ← LoadReaderBody [stream, z];

  -- 4. String Mapping Properties
  parm.mapProps ← parm.heap.NEW [K4.MapPropertyRecord ← [
                        header:  LoadReaderBody [stream, z],
                        number:  Stream.GetWord [stream],
                        map:     NIL,
                        tagSize: NIL]];
  FOR k: CARDINAL IN [1..parm.mapProps.number] DO
    map ← parm.heap.NEW [K4.MappingRecord ← [
                        next: NIL,
                        from: LoadReaderBody [stream, z],
                        to:   LoadReaderBody [stream, z] ]];
    IF parm.mapProps.map = NIL
      THEN parm.mapProps.map ← map
      ELSE currentMap.next ← map;
     currentMap ← map;
    ENDLOOP;

  -- 5. Font Properties
  parm.fonProps ← parm.heap.NEW [K4.FontPropertyRecord];
   FOR k: CARDINAL IN [0..9] DO
    parm.fonProps.fonts[k].font ← VAL[LoadEnumerated [stream]];
    parm.fonProps.fonts[k].size ← VAL[LoadEnumerated [stream]];
    parm.fonProps.fonts[k].italics ← LoadBoolean [stream];
    parm.fonProps.fonts[k].bold ← LoadBoolean [stream];
    ENDLOOP;

  EXITS  Termination => {};
  END;  -- of ENABLE

  Stream.Delete [stream];
  END;  -- of LoadFiledData


StoreFiledData: PUBLIC PROCEDURE [parm: K4.IconParms] =
  BEGIN
  -- This procedure writed back into the icon file the values
  -- recorded in the property sheet.

  map:       K4.Mapping ← parm.mapProps.map;
  signature: CARDINAL ← Signature[];
  stream:    Stream.Handle ← NSFileStream.Create [parm.iconFile,
                                                  FALSE];

  NSFileStream.SetLength [[stream], 0];  -- truncate old stream

  -- 1. Signature.
  Stream.PutWord [stream, signature];

  -- 2. General Properties
  StoreReaderBody [stream, @parm.genProps.iconName];
  StoreEnumerated [stream, parm.genProps.channelSpeed.ORD];
  StoreReaderBody [stream, @parm.genProps.folderName];
  StoreReaderBody [stream, @parm.genProps.others];

  -- 3. Document Properties
  StoreReaderBody [stream, @parm.docProps.folderName];
  StoreReaderBody [stream, @parm.docProps.docName];
  StoreBoolean [stream, parm.docProps.justification];
  StoreEnumerated [stream, parm.docProps.lineHeight.ORD];
  StoreEnumerated [stream, parm.docProps.preLeading.ORD];
  StoreEnumerated [stream, parm.docProps.postLeading.ORD];
  StoreEnumerated [stream, parm.docProps.underlining.ORD];
  StoreReaderBody [stream, @parm.docProps.guessMark];
  StoreEnumerated [stream, parm.docProps.dropKeepMark.ORD];
  StoreEnumerated [stream, parm.docProps.pageBreak.ORD];
  StoreReaderBody [stream, @parm.docProps.others];

  -- 4. Mapping Properties
  StoreReaderBody [stream, @parm.mapProps.header];
  Stream.PutWord [stream, parm.mapProps.number];
  FOR k: CARDINAL IN [1..parm.mapProps.number]
  WHILE map # NIL DO     -- should diagnose if map is NIL
    StoreReaderBody [stream, @map.from];
    StoreReaderBody [stream, @map.to];
```

```
    map ← map.next;
    ENDLOOP;

  -- 5. Font Properties
  FOR k: CARDINAL IN [0..9] DO
    StoreEnumerated [stream, parm.fonProps.fonts[k].font.ORD];
    StoreEnumerated [stream, parm.fonProps.fonts[k].size.ORD];
    StoreBoolean [stream, parm.fonProps.fonts[k].italics];
    StoreBoolean [stream, parm.fonProps.fonts[k].bold];
    ENDLOOP;

  Stream.SendNow [stream];
  Stream.Delete [stream];
  END;  -- of StoreFiledData


FreeIconProps: PUBLIC PROC [props: K4.GeneralProperties,
                            z: UNCOUNTED ZONE] =
  BEGIN
  IF props = NIL THEN RETURN;
  XS.FreeReaderBytes [@props.iconName, z];
  XS.FreeReaderBytes [@props.folderName, z];
  XS.FreeReaderBytes [@props.others, z];
  IF props.tagSize # NIL THEN z.FREE [@props.tagSize];
  z.FREE [@props];
  END;  -- of FreeIconProps


FreeTextProps: PUBLIC PROC [props: K4.DocumentProperties,
                            z: UNCOUNTED ZONE] =
  BEGIN
  IF props = NIL THEN RETURN;
  XS.FreeReaderBytes [@props.folderName, z];
  XS.FreeReaderBytes [@props.docName, z];
  XS.FreeReaderBytes [@props.guessMark, z];
  XS.FreeReaderBytes [@props.others, z];
  IF props.tagSize # NIL THEN z.FREE [@props.tagSize];
  z.FREE [@props];
  END;  -- of FreeTextProps


FreeMapProps: PUBLIC PROC [props: K4.MapProperties,
                           z: UNCOUNTED ZONE] =
  BEGIN
  map, hold: K4.Mapping;

  IF props = NIL THEN RETURN;
  XS.FreeReaderBytes [@props.header, z];
  FOR map ← props.map, hold WHILE map # NIL DO
    hold ← map.next;
    XS.FreeReaderBytes [@map.from, z];
    XS.FreeReaderBytes [@map.to, z];
    z.FREE [@map];
    ENDLOOP;
  IF props.tagSize # NIL THEN z.FREE [@props.tagSize];
  z.FREE [@props];
  END;  -- of FreeMapProps


FreeFontProps: PUBLIC PROC [props: K4.FontProperties,
                            z: UNCOUNTED ZONE] =
  BEGIN
  IF props = NIL THEN RETURN;
  IF props.tagSize # NIL THEN z.FREE [@props.tagSize];
  z.FREE [@props];
  END;  -- of FreeFontProps



<<==========  PRIVATE PROCEDURES  ==========>>


LoadBoolean: PROC [stream: Stream.Handle] RETURNS [BOOLEAN] =
```

```
  INLINE {RETURN [Stream.GetWord[stream] # 0]};


StoreBoolean: PROC [stream: Stream.Handle, boolean: BOOLEAN] =
  INLINE {Stream.PutWord [stream, boolean.ORD]};


LoadEnumerated: PROC [stream: Stream.Handle]
            RETURNS [Environment.Word] =
  INLINE {RETURN [Stream.GetWord[stream]]};


StoreEnumerated: PROC [stream: Stream.Handle,
                       value: Environment.Word] =
  INLINE {Stream.PutWord [stream, value]};


LoadReaderBody: PROC [stream: Stream.Handle, z: UNCOUNTED ZONE]
        RETURNS [rb: XS.ReaderBody] =
  BEGIN
  length: CARDINAL ← Stream.GetWord [stream];
  wb: XS.WriterBody ← XS.NewWriterBody [length, z];

  FOR k: CARDINAL IN [0..length) DO
    XS.AppendChar [@wb, Stream.GetWord[stream]];
    ENDLOOP;
  rb ← XS.CopyToNewReaderBody[XS.ReaderFromWriter[@wb], z];
       -- the bytes allocated here must be released by the client

  XS.FreeWriterBytes[@wb];
  END;


StoreReaderBody: PROC [stream: Stream.Handle, r: XS.Reader] =
  BEGIN
  length:  CARDINAL ← XS.CharacterLength[r];
  lambdaB: XS.ReaderBody ← XS.Dereference [r];

  Stream.PutWord [stream, length];
  FOR k: CARDINAL IN [0..length) DO
    Stream.PutWord [stream, XS.Lop[@lambdaB]];
    ENDLOOP;
  END;


Signature: PROC RETURNS [CARDINAL] = INLINE
   -- calculates some number likely to change whenever a change is
   -- made to K4.IconParmsRecord.  This is to try to catch instances
   -- where the user has a Kurzweil icon not matching the current
   -- software version.
  {RETURN [    SIZE [K4.DocumentPropertyRecord] +
           3 * SIZE [K4.MapPropertyRecord]   +
           5 * SIZE [K4.FontPropertyRecord]  +
           7 * SIZE [K4.GeneralPropertyRecord]  +
          11 * TypeAndVersion[].version]};



END.  -- of K4FiledDataImpl

10-Mar-87  9:47:45 created from code moved from K4IconImpl.
13-Mar-87 10:11:40 added options and modified others.
16-Mar-87  9:43:36 added signature.
17-Mar-87 14:12:24 added character mapping.
19-Mar-87 10:55:28 implemented TypeAndVersion.
20-Mar-87 10:25:13 version 14: mapProps.number starts at 0 instead of 1.
31-Mar-87 12:30:25 version 16: called ReduceList before writing map.number and only if not creating
prototype.
31-Mar-87 13:21:36 filed data initialization moved to K4IconImpl.
14-Apr-87 14:02:42 version 17: paragraph properties.
24-Apr-87 16:15:03 version 18: dropped canvas, moved fonts to separate prop sheet.
27-Apr-87 13:00:24 version 19: folder name for transmissions, pageBreak choice.
/
```

```
<< File: K4IconImpl.mesa        27-Apr-87 13:06:14

deLaBeaujardiere:OSBU North:Xerox (deLaBeaujardiere.PA)

Copyright (C) 1986 by Xerox Corporation. All rights reserved.

>>

DIRECTORY Atom, Containee,
          Display, Environment,
          Heap, K4,
          NSFile,
          Prototype,
          SimpleTextDisplay, SimpleTextFont,
          StarWindowShell,
          XChar, XString;

K4IconImpl: PROGRAM

IMPORTS Atom, Containee, Display,
        Heap, K4, NSFile,
        Prototype,
        SimpleTextDisplay, SimpleTextFont,
        XString =

BEGIN
        OPEN XS: XString;

<<=====  TYPE DEFINITIONS AND CONSTANTS =====>>

IconRecord: TYPE = RECORD [
  heap:           UNCOUNTED ZONE,
  oldGenericProc: Containee.GenericProc ← NIL,
  keyOpen:        Atom.ATOM,
  keyProps:       Atom.ATOM,
  smallIcon:      XS.Character ← XChar.not];


<<=====  GLOBAL VARIABLES =====>>

icon: LONG POINTER TO IconRecord ← NIL;


<<=====  PROCEDURES  =====>>


InstallK4Icon: PROCEDURE =
  BEGIN
  heap:         UNCOUNTED ZONE ← Heap.Create[1];
  rows:         CARDINAL = 13;  -- 13 rows in small icon picture
  smallPicture: PACKED ARRAY[0..rows) OF WORD ← [
    177770B, 100010B, 125010B, 100010B, 177770B, 100010B, 137750B,
    110110B, 170170B, 010100B, 010700B, 010600B, 017400B];
  iconName:     XS.ReaderBody ← XS.FromSTRING ["Kurzweil 4000"L];
  impl:         Containee.Implementation;
  iconFileType: NSFile.Type;
  currentVersion: Prototype.Version;

  [iconFileType, currentVersion] ← K4.TypeAndVersion[];
  icon ← heap.NEW[IconRecord];
  icon.heap ← heap;
  icon.keyOpen ← Atom.MakeAtom["Open"L];
  icon.keyProps ← Atom.MakeAtom["Props"L];

  IF Prototype.Find [type: iconFileType,
                     version: currentVersion] = NSFile.nullReference THEN
      MakeNewVersion [@iconName, iconFileType, currentVersion, heap];

  impl ← Containee.GetImplementation [iconFileType];

  icon.oldGenericProc ← impl.genericProc;
  icon.smallIcon ← SimpleTextFont.AddClientDefinedCharacter[
                                width: 13,
                                height: rows,
```

```
                                bitsPerLine: 16,
                                bits: @smallPicture];

  impl.genericProc ← GenericProc;
  impl.name ← iconName;
  impl.smallPictureProc ← PaintSmallIcon;
  impl.pictureProc ← PaintBigIcon;

  [] ← Containee.SetImplementation [iconFileType, impl];
  END;  -- of InstallK4Icon


MakeNewVersion: PROC [iconName: XS.Reader,
                      iconType: NSFile.Type,
                      iconVersion: Prototype.Version,
                      z: UNCOUNTED ZONE] =
  BEGIN
  question: LONG STRING ← "@"L;
  parm: K4.IconParms ← z.NEW[K4.IconParmsRecord];

  parm.heap ← z;
  parm.genProps ← z.NEW [K4.GeneralPropertyRecord ← [
              iconName:      iconName↑,
              channelSpeed:  ninetySix,
              folderName:    XS.FromSTRING ["Kurzweil Transmissions"L],
              others:        XS.FromSTRING [
        "Bits per Char: 8, Stop bits: 1, Parity: None, Asynchronous"L]]];
  parm.docProps ← z.NEW [K4.DocumentPropertyRecord ← [
              folderName:    XS.FromSTRING ["Kurzweil Documents"L],
              docName:       XS.FromSTRING ["Interpreted Document"L],
              justification: TRUE,
              lineHeight:    single,
              preLeading:    single,
              postLeading:   singleHalf,
              underlining:   underline,
              guessMark:     XS.FromSTRING [question],
              dropKeepMark:  drop,
              pageBreak:     drop,
              others:        XS.FromSTRING [
        "End Line: HOD, End Page: HOC, End Column/Para: None, Horizontal: Preserve, Vertical:
        Preserve Breaks"L]]];
  parm.mapProps ← z.NEW [K4.MapPropertyRecord ← [
              header:        XS.FromSTRING ["Character Substitutions"L],
              number:        0,
              map: NIL]];
  parm.fonProps ← z.NEW [K4.FontPropertyRecord ← [
              fonts: [[classic, ten,      FALSE, FALSE],
                      [classic, ten,      FALSE, TRUE ],
                      [classic, ten,      TRUE,  FALSE],
                      [classic, eight,    FALSE, FALSE],
                      [classic, eight,    FALSE, TRUE ],
                      [classic, eight,    TRUE,  FALSE],
                      [classic, twelve,   FALSE, FALSE],
                      [classic, twelve,   FALSE, TRUE ],
                      [classic, twelve,   TRUE,  FALSE],
                      [classic, fourteen, FALSE, FALSE]]]];

  parm.iconFile ← Prototype.Create [name: iconName,
                                    type: iconType,
                                    subtype: 0,
                                    version: iconVersion,
                                    isDirectory: FALSE];
  Prototype.PurgeOldVersions [type: iconType,
                              subtype: 0,
                              current: iconVersion];
  K4.StoreFiledData [parm];
  z.FREE [@parm.genProps];
  z.FREE [@parm.docProps];
  z.FREE [@parm.mapProps] ;
  z.FREE [@parm.fonProps] ;
  NSFile.Close [parm.iconFile];
  z.FREE [@parm];
  END;  -- of MakeNewVersion


GenericProc: Containee.GenericProc =
```

```
  -- defined as PROC [atom: Atom.ATOM,
  --                  data: DataHandle,
  --                  changeProc: ChangeProc ← NIL,
  --                  changeProcData: LONG POINTER ← NIL,
  --          RETURNS [LONG UNSPECIFIED]
  BEGIN
  shell: LONG POINTER;

  shell ← SELECT atom FROM
            icon.keyOpen  => K4.OpenWindow [data, changeProc,
                                            changeProcData,
                                            icon.smallIcon],
            icon.keyProps => K4.OpenPSheet [data, changeProc,
                                            changeProcData],
            ENDCASE => icon.oldGenericProc [atom, data, changeProc,
                                            changeProcData];

  RETURN [shell];
  END;  -- of GenericProc


Err: PROCEDURE[message: LONG STRING] = BEGIN
  msgRB: XString.ReaderBody ← XString.FromSTRING[message];
  Containee.Error [@msgRB];
  END;  -- of Err


PaintSmallIcon: Containee.SmallPictureProc = {RETURN[icon.smallIcon]};


PaintBigIcon: Containee.PictureProc = BEGIN
  widthInPixels: CARDINAL = 65;
  widthInWords: CARDINAL = 5;
  heightInPixels: CARDINAL = 60;
  IF new=garbage THEN RETURN
  ELSE BEGIN
    fileName: XS.ReaderBody;
    cacheTicket: Containee.Ticket;
    mask: ARRAY[0..widthInWords*heightInPixels) OF WORD ← [
      037777B, 177777B, 177777B, 177776B, 000000B,
      077777B, 177777B, 177777B, 177777B, 000000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      077777B, 177777B, 177777B, 177777B, 000000B,
      037777B, 177777B, 177777B, 177776B, 000000B,
      000077B, 177777B, 177777B, 177000B, 000000B,
```

```
    000077B, 177777B, 177777B, 177000B, 000000B,
    000077B, 177777B, 177777B, 177000B, 000000B,
    000077B, 177777B, 177777B, 177000B, 000000B,
    000077B, 177777B, 177777B, 177000B, 000000B,
    000077B, 177777B, 177777B, 177000B, 000000B,
    000077B, 177777B, 177777B, 177000B, 000000B,
    000077B, 177777B, 177777B, 177000B, 000000B,
    000077B, 177777B, 177777B, 177000B, 000000B,
    000077B, 177777B, 177777B, 177000B, 000000B,
    000077B, 177777B, 177777B, 177000B, 000000B,
    000077B, 177777B, 177777B, 177000B, 000000B,
    000077B, 177777B, 177777B, 177000B, 000000B,
    000077B, 177777B, 177777B, 177000B, 000000B,
    000077B, 177777B, 177777B, 177000B, 071145B,
    000077B, 177777B, 177777B, 176000B, 000022B,
    000077B, 177777B, 177777B, 174000B, 007144B,
    000077B, 177777B, 177777B, 170000B, 000022B,
    000077B, 177777B, 177777B, 160000B, 002312B,
    000077B, 177777B, 177777B, 140000B, 000005B,
    000077B, 177777B, 177777B, 100000B, 000005B,
    000077B, 177777B, 177777B, 000000B, 000025B,
    000077B, 177777B, 177776B, 000000B, 064556B];

  IF new=reference OR new=referenceHighlighted THEN new ← normal;
  box.place.x ← box.place.x + 2;
  box.dims.w ← widthInPixels;
  box.place.y ← box.place.y + 8;
  box.dims.h ← heightInPixels;

  Display.Bitmap[
    window: window,
    box: box,
    address: [word: @mask, bit: 0],
    bitmapBitWidth: widthInWords*Environment.bitsPerWord,
    flags: SELECT new FROM
      highlighted, referenceHighlighted => [  --for inverting picture
        disjoint: TRUE,
        srcFunc: null,
        dstFunc: or],
      ENDCASE => [
        disjoint: TRUE,
        srcFunc: complement,
        dstFunc: and]];

  IF new#ghost THEN BEGIN
    picture: ARRAY[0..widthInWords*heightInPixels) OF WORD ← [
      037777B, 177777B, 177777B, 177776B, 000000B,
      077777B, 177777B, 177777B, 177777B, 000000B,
      160000B, 000000B, 000000B, 000003B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      141777B, 177777B, 177777B, 177741B, 100000B,
      141760B, 000000B, 000000B, 003741B, 100000B,
```

```
        140020B, 000000B, 000000B, 002001B, 100000B,
        140020B, 052525B, 052525B, 002001B, 100000B,
        140020B, 025252B, 125252B, 002001B, 100000B,
        140020B, 000000B, 000000B, 002001B, 100000B,
        160020B, 000000B, 000000B, 002003B, 100000B,
        077760B, 052525B, 052525B, 003777B, 000000B,
        037760B, 025252B, 125252B, 003776B, 000000B,
        000060B, 000000B, 000000B, 003000B, 000000B,
        000060B, 000000B, 000000B, 003000B, 000000B,
        000060B, 052525B, 052525B, 003000B, 000000B,
        000060B, 025252B, 125252B, 003000B, 000000B,
        000060B, 000000B, 000000B, 003000B, 000000B,
        000060B, 000000B, 000000B, 003000B, 000000B,
        000060B, 052525B, 052525B, 003000B, 000000B,
        000060B, 025252B, 125252B, 003000B, 000000B,
        000060B, 000000B, 000000B, 003000B, 000000B,
        000060B, 000000B, 000000B, 003000B, 000000B,
        000060B, 052525B, 052525B, 003000B, 000000B,
        000060B, 025252B, 125252B, 003000B, 000000B,
        000060B, 000000B, 000000B, 003000B, 000000B,
        000060B, 000000B, 000000B, 003000B, 000000B,
        000060B, 000000B, 000000B, 003000B, 000000B,
        000060B, 000000B, 000007B, 177000B, 000000B,
        000060B, 000000B, 000004B, 016000B, 000000B,
        000060B, 000000B, 000004B, 034000B, 000000B,
        000060B, 000000B, 000004B, 070000B, 000000B,
        000060B, 000000B, 000004B, 160000B, 000000B,
        000060B, 000000B, 000005B, 140000B, 000000B,
        000060B, 000000B, 000007B, 100000B, 000000B,
        000077B, 177777B, 177777B, 000000B, 000000B,
        000077B, 177777B, 177776B, 000000B, 000000B];

      Display.Bitmap[
        window: window,
        box: box,
        address: [word: @picture, bit: 0],
        bitmapBitWidth: widthInWords*Environment.bitsPerWord,
        flags: SELECT new FROM
          highlighted, referenceHighlighted => [
            disjoint: TRUE,
            srcFunc: complement,
            dstFunc: and],
          ENDCASE => [
            disjoint: TRUE,
            srcFunc: null,
            dstFunc: or]];
      END;  --of drawing in the picture per se (not a ghost)

    [fileName, cacheTicket] ← Containee.GetCachedName[data];
    [] ← SimpleTextDisplay.StringIntoWindow[
      string: @fileName,
      window: window,
      place: [x: box.place.x + 3, y: box.place.y + 3],
      lineWidth: widthInPixels - 6,
      maxNumberOfLines: 1,
      wordBreak: FALSE,
      flags: SELECT new FROM
          highlighted, referenceHighlighted => [
            disjoint: TRUE,
            srcFunc: complement,
            dstFunc: and],
          ENDCASE => Display.paintFlags];
    Containee.ReturnTicket[cacheTicket];
    END;  --of drawing; i.e., not garbage
  END;  -- of PaintBigIcon



<<========== MAIN LINE CODE ==========>>

InstallK4Icon[];

END.  -- of K4IconImpl
14-Jan-87 15:03:40 created from DestTextIconImpl.
```

```
 6-Feb-87 15:35:08 changed initial preset of pSheet.
10-Mar-87  9:50:14 removed Courier usage, moved Load/StoreFiledData to K4FiledDataImpl.
19-Mar-87 11:05:26 used TypeAndVersion to get fileType/version.
31-Mar-87 13:50:53 moved initialization of icon main data store here from K4FiledDataImpl.
14-Apr-87 13:53:19 added paragraph properties.
24-Apr-87 16:22:04 dropped canvas and ArtScan, moved fonts to separate prop sheet.
27-Apr-87 13:05:58 foldername for transmissions, page break.
/
```

```
<< File: K4PSheetImpl.mesa        28-Apr-87 16:59:05

deLaBeaujardiere:OSBU North:Xerox (deLaBeaujardiere.PA)

Copyright (C) 1986 by Xerox Corporation. All rights reserved.

>>

DIRECTORY
        Attention, Containee,
        Environment, FormWindow, Heap,
        K4,
        NSFile, NSString,
        PropertySheet,
        SimpleTextDisplay,
        StarWindowShell, Window, XString;

K4PSheetImpl: PROGRAM

IMPORTS Attention,
        FormWindow, Heap, K4, NSFile, NSString,
        PropertySheet,
        SimpleTextDisplay, XString

EXPORTS K4 =

BEGIN

OPEN FW: FormWindow,
     K4,
     XS: XString;



<<==========  PUBLIC PROCEDURES  ==========>>


OpenPSheet: PUBLIC PROCEDURE [iconData: Containee.DataHandle,
                              changeProc: Containee.ChangeProc,
                              changeProcData: LONG POINTER]
                    RETURNS [StarWindowShell.Handle] =
  BEGIN
  zone:          UNCOUNTED ZONE ← Heap.Create [1];
                              -- deleted by DoPropsCommands
  pSheetPlace:   Window.Place ← PropertySheet.nullPlace;
  pSheetSize:    Window.Dims ← [0, 0];
  pSheetName:    XS.ReaderBody ← XS.FromSTRING [
                        "Kurzweil 4000 Properties"L];
  versionMismatch: BOOLEAN;
  iconParms:     K4.IconParms ← zone.NEW [K4.IconParmsRecord ← [
                        heap: zone,
                        iconData: iconData,
                        changeProc: changeProc,
                        changeProcData: changeProcData,
                        iconFile: NSFile.OpenByReference
                                        [iconData.reference] ]];
                        -- freed by TakeDownPSheet

  versionMismatch ← K4.LoadFiledData [iconParms];
                              -- unloaded by TakeDownPSheet
  IF versionMismatch THEN
    BEGIN
    Msg ["Obsolete icon does not work with new software"L];
    NSFile.Close [iconParms.iconFile];
    zone.FREE [@iconParms];
    Heap.Delete [zone];
    RETURN [StarWindowShell.nullHandle];
    END;

  -- Create the Property Sheet.
  iconParms.propSheet ← PropertySheet.CreateLinked [
                linkWindowItems: MakeLinkProps,
                linkWindowItemsLayout: LayLinkItem,
                formWindowItems: MakeTextProps,        -- first shown
                formWindowItemsLayout: LayTextProps,   -- is text sheet
```

```
                menuItemProc: SetTextProps,
                menuItems: [done: TRUE, cancel: TRUE],
                title: @pSheetName,
                size: pSheetSize,
                placeToDisplay: pSheetPlace,
                afterTakenDownProc: TakeDownPSheet,
                clientData: iconParms];

  RETURN [iconParms.propSheet];
  END;  -- of OpenPSheet


OpenDocOptionSheet: PUBLIC PROC [
                        convertParm: K4.ConvertParms,
                        takeDown: PropertySheet.MenuItemProc] =
  BEGIN
  pSheetName: XS.ReaderBody ← XS.FromSTRING ["Document Options"L];
  pSize: Window.Dims ← [500, 400];
  pPlace: Window.Place ← [500, 50];

  convertParm.optionSheet  ← PropertySheet.CreateLinked [
                linkWindowItems: MakeLinkOptions,
                linkWindowItemsLayout: LayLinkItem,
                formWindowItems: MakeTextOptions,       -- first shown
                formWindowItemsLayout: LayTextOptions, -- is text
                menuItemProc: SetTextOptions,
                menuItems: [start: TRUE, cancel: TRUE],
                title: @pSheetName,
                size: pSize,
                placeToDisplay: pPlace,
                afterTakenDownProc: takeDown,
                clientData: convertParm];
  END;  -- of OpenDocOptionSheet



<<==========  PRIVATE PROCEDURES FOR DOCUMENT SUBWINDOW  ==========>>


MakeTextOptions: FW.MakeItemsProc =
  BEGIN
  parm: K4.ConvertParms ← LOOPHOLE[clientData];
  MakeTextItems [window, parm.docProps, parm.zone];
  END;  -- of MakeTextOptions


MakeTextProps: FW.MakeItemsProc =
  BEGIN
  iconParm: K4.IconParms ← LOOPHOLE[clientData];
  MakeTextItems [window, iconParm.docProps, iconParm.heap];
  END;  -- of MakeTextProps


MakeTextItems: PROC [window: Window.Handle,
                     docProps: K4.DocumentProperties,
                     z: UNCOUNTED ZONE] =
  BEGIN
  label: XS.ReaderBody;

  underliningChoices: FW.ChoiceItems ← DESCRIPTOR [uChoices];
  uChoices: ARRAY [0..3) OF FW.ChoiceItem ← [
                [string [choiceNumber: Underlining.underline.ORD,
                         string: XString.FromSTRING["Underline"L]]],
                [string [choiceNumber: Underlining.italics.ORD,
                         string: XString.FromSTRING["Italics"L]]],
                [string [choiceNumber: Underlining.plain.ORD,
                         string: XString.FromSTRING["Plain"L]]] ];

  lineChoices: FW.ChoiceItems ← DESCRIPTOR [lChoices];
  lChoices: ARRAY [0..4) OF FW.ChoiceItem ← [
                [string [choiceNumber: LineSpacing.single.ORD,
                         string: XString.FromSTRING["Single"L]]],
                [string [choiceNumber: LineSpacing.singleHalf.ORD,
                         string: XString.FromSTRING["1 1/2"L]]],
                [string [choiceNumber: LineSpacing.double.ORD,
```

```
                    string: XString.FromSTRING["Double"L]]],
                [string [choiceNumber: LineSpacing.triple.ORD,
                    string: XString.FromSTRING["Triple"L]]] ];

    guessChoices: FW.ChoiceItems ← DESCRIPTOR [gChoices];
    gChoices: ARRAY [0..2) OF FW.ChoiceItem ← [
                [string [choiceNumber: DropKeep.drop.ORD,
                    string: XString.FromSTRING["Drop"L]]],
                [string [choiceNumber: DropKeep.keep.ORD,
                    string: XString.FromSTRING["Keep"L]]] ];

    pageBreakChoices: FW.ChoiceItems ← DESCRIPTOR [pChoices];
    pChoices: ARRAY [0..3) OF FW.ChoiceItem ← [
                [string [choiceNumber: Underlining.underline.ORD,
                    string: XString.FromSTRING["None"L]]],
                [string [choiceNumber: Underlining.italics.ORD,
                    string: XString.FromSTRING["Every Page"L]]],
                [string [choiceNumber: Underlining.plain.ORD,
                    string: XString.FromSTRING["Unfilled Pages"L]]] ];

    IF docProps.tagSize = NIL THEN
      docProps.tagSize ← z.NEW[TextTagSizes];
                                        -- released by FreeTextProps

    docProps.tagSize[folderName] ← Measure [@label, "Document Folder Name"L];
    FW.MakeTextItem [window: window, myKey: K4.TextItems.folderName.ORD,
                    tag: @label,
                    initString: @docProps.folderName,
                    width: 200];

    docProps.tagSize[docName] ← Measure [@label, "Document Name"L];
    FW.MakeTextItem [window: window, myKey: K4.TextItems.docName.ORD,
                    tag: @label,
                    initString: @docProps.docName,
                    width: 200];

    docProps.tagSize[justification] ← Measure [@label,
                                          "Paragraph Right Edge"L];
    FW.MakeBooleanItem[window: window,
      myKey: K4.TextItems.justification.ORD, tag: @label,
      label: [string [XString.FromSTRING["Justify"L]]],
      initBoolean: docProps.justification];

    docProps.tagSize[lineHeight] ← Measure [@label,
                                        "Paragraph Line Height"L];
    FW.MakeChoiceItem[window: window,
      myKey: K4.TextItems.lineHeight.ORD, tag: @label,
      values: lineChoices,
      initChoice: VAL[docProps.lineHeight]];

    docProps.tagSize[preLeading] ← Measure [@label,
                                        "Lines before Paragraph"L];
    FW.MakeChoiceItem[window: window,
      myKey: K4.TextItems.preLeading.ORD, tag: @label,
      values: lineChoices,
      initChoice: VAL[docProps.preLeading]];

    docProps.tagSize[postLeading] ← Measure [@label,
                                        "Lines after Paragraph"L];
    FW.MakeChoiceItem[window: window,
      myKey: K4.TextItems.postLeading.ORD, tag: @label,
      values: lineChoices,
      initChoice: VAL[docProps.postLeading]];

    docProps.tagSize[underlining] ← Measure [@label, "Underlining"L];
    FW.MakeChoiceItem[window: window,
      myKey: K4.TextItems.underlining.ORD, tag: @label,
      values: underliningChoices,
      initChoice: VAL[docProps.underlining]];

    docProps.tagSize[guessMark] ← Measure [@label,
                                    "Questionable Character"L];
    FW.MakeTextItem [window: window, myKey: K4.TextItems.guessMark.ORD,
                    tag: @label,
                    initString: @docProps.guessMark,
                    width: 40];
```

```
  FW.MakeChoiceItem[window: window,
    myKey: K4.TextItems.dropKeepMark.ORD, tag: NIL,
    values: guessChoices,
    initChoice: VAL[docProps.dropKeepMark]];

  docProps.tagSize[pageBreak] ← Measure [@label, "Page Break"L];
  FW.MakeChoiceItem[window: window,
    myKey: K4.TextItems.pageBreak.ORD, tag: @label,
    values: pageBreakChoices,
    initChoice: VAL[docProps.pageBreak]];

  docProps.tagSize[others] ← Measure [@label,
                                  "Required Tailor Choices"L];
  FW.MakeTextItem [window: window, myKey: K4.TextItems.others.ORD,
                   tag: @label,
                   initString: @docProps.others,
                   width: 300, boxed: FALSE, readOnly: TRUE];
  END;  -- of MakeTextItems


LayTextOptions: FW.LayoutProc =
  BEGIN
  parm: K4.ConvertParms ← LOOPHOLE[clientData];
  LayTextItems [window, parm.docProps, parm.zone];
  END;  -- of LayTextOptions


LayTextProps: FW.LayoutProc =
  BEGIN
  iconParm: K4.IconParms ← LOOPHOLE[clientData];
  LayTextItems [window, iconParm.docProps, iconParm.heap];
  END;  -- of LayTextProps


LayTextItems: PROC [window: Window.Handle,
                    docProps: K4.DocumentProperties,
                    z: UNCOUNTED ZONE] =
  BEGIN
  margin: CARDINAL = 5;
  maxTag: CARDINAL ← 0;
  line: FW.Line;
  ts: LONG POINTER TO TextTagSizes ← docProps.tagSize;  -- accelerator

  LaySingleItem: PROC [itemKey: K4.TextItems,
                       tagSize, interline: CARDINAL] =
        BEGIN
        IF interline > 0 THEN
                line ← FW.AppendLine[window: window,
                          spaceAboveLine: interline];
        FW.AppendItem[window: window, line: line,
                item: itemKey.ORD,
                preMargin: IF interline = 0 THEN 8
                           ELSE Prespace[tagSize, maxTag, margin]];
        END;  -- of LaySingleItem

  FOR k: TextItems IN TextItems DO
    maxTag ← MAX [maxTag, ts[k]];
    ENDLOOP;

  LaySingleItem[folderName, ts[folderName], 6];
  LaySingleItem[docName, ts[docName], 6];
  LaySingleItem[justification, ts[justification], 6];
  LaySingleItem[lineHeight, ts[lineHeight], 6];
  LaySingleItem[preLeading, ts[preLeading], 6];
  LaySingleItem[postLeading, ts[postLeading], 6];
  LaySingleItem[underlining, ts[underlining], 6];
  LaySingleItem[guessMark, ts[guessMark], 6];
  LaySingleItem[dropKeepMark, ts[dropKeepMark], 0];
  LaySingleItem[pageBreak, ts[pageBreak], 6];
  LaySingleItem[others, ts[others], 12];

  FW.Repaint [window];
  END;  -- of LayTextItems


SetTextOptions: PropertySheet.MenuItemProc =
```

```
  BEGIN
  parm: K4.ConvertParms ← LOOPHOLE[clientData];
  RETURN [SetTextItems [formWindow, parm.docProps, parm.zone]];
  END;  -- of SetTextOptions


SetTextProps: PropertySheet.MenuItemProc =
  BEGIN
  iconParm: K4.IconParms ← LOOPHOLE[clientData];
  RETURN [SetTextItems [formWindow, iconParm.docProps, iconParm.heap]];
  END;  -- of SetTextProps


SetTextItems: PROC [window: Window.Handle,
                    docProps: K4.DocumentProperties,
                    z: UNCOUNTED ZONE]
          RETURNS [BOOLEAN] =
  BEGIN

  Boolean: PROCEDURE [item: TextItems] RETURNS [BOOLEAN] = INLINE
        {RETURN [FW.GetBooleanItemValue [window, item.ORD]]};

  Choice: PROCEDURE [item: TextItems] RETURNS [CARDINAL] = INLINE
        {RETURN [FW.GetChoiceItemValue [window, item.ORD] ]};

  IF NOT FW.HasAnyBeenChanged [window] THEN RETURN [TRUE];

  IF FW.HasBeenChanged [window, TextItems.folderName.ORD] THEN
    BEGIN
    docProps.folderName ← UpdateBody [window,
                                      TextItems.folderName.ORD,
                                      @docProps.folderName, z];
    IF XS.Empty [@docProps.folderName] THEN
      BEGIN
      Msg ["Folder name cannot be empty"L];
      RETURN [FALSE];
      END;
    END;

  IF FW.HasBeenChanged [window, TextItems.docName.ORD] THEN
    BEGIN
    docProps.docName ← UpdateBody [
                               window, TextItems.docName.ORD,
                               @docProps.docName, z];
    IF XS.Empty [@docProps.docName] THEN
      BEGIN
      Msg ["Document name cannot be empty"L];
      RETURN [FALSE];
      END;
    END;

    docProps.justification ← Boolean [K4.TextItems.justification];
    docProps.lineHeight    ← VAL [Choice [K4.TextItems.lineHeight]];
    docProps.preLeading    ← VAL [Choice [K4.TextItems.preLeading]];
    docProps.postLeading   ← VAL [Choice [K4.TextItems.postLeading]];
    docProps.underlining   ← VAL [Choice [K4.TextItems.underlining]];
    docProps.dropKeepMark  ← VAL [Choice [K4.TextItems.dropKeepMark]];
    docProps.pageBreak     ← VAL [Choice [K4.TextItems.pageBreak]];

  IF FW.HasBeenChanged [window, TextItems.guessMark.ORD] THEN
    docProps.guessMark ← UpdateBody [
                               window, TextItems.guessMark.ORD,
                               @docProps.guessMark, z];

  RETURN [TRUE];
  END;  -- of SetTextItems


<<==========  PRIVATE PROCEDURES FOR MAPPING SUBWINDOW  ==========>>

<< Road map to their usage:
        MakeMapOptions          => MakeMapItems
                                => MakeNextMapItems
        MakeMapProps            => MakeMapItems
                                => MakeNextMapItems
```

```
        LayMapOptions           => LayMapItems
        LayMapProps             => LayMapItems
        AddNextMapOptions       => MakeNextMapItems
                                => LayNextMapItems
        AddNextMapProps         => MakeNextMapItems
                                => LayNextMapItems              >>


MakeMapOptions: FW.MakeItemsProc =
  BEGIN
  parm: K4.ConvertParms ← LOOPHOLE[clientData];

  MakeMapItems [window, AddNextMapOptions, parm.mapProps, parm.zone];

  -- If no mapping option exists yet, must display an empty pair
  IF parm.mapProps.number = 0 THEN
      MakeNextMapItems [window, AddNextMapOptions,
                        parm.mapProps, parm.zone];
  END;  -- of MakeMapOptions


MakeMapProps: FW.MakeItemsProc =
  BEGIN
  ip: K4.IconParms ← LOOPHOLE[clientData];

  MakeMapItems [window, AddNextMapProps, ip.mapProps, ip.heap];
  IF ip.mapProps.number = 0 THEN
      MakeNextMapItems [window, AddNextMapProps,
                        ip.mapProps, ip.heap];
  END;  -- of MakeMapProps


MakeMapItems: PROC [window: Window.Handle,
                    nextOut: FW.NextOutOfProc,
                    mapProps: K4.MapProperties,
                    z: UNCOUNTED ZONE] =
  BEGIN
  -- Create window items for the header and the
  -- existing from/to pairs.
  label: XS.ReaderBody;
  map: K4.Mapping ← mapProps.map;  -- may be NIL if no items yet.

  IF mapProps.tagSize = NIL THEN
    mapProps.tagSize ← z.NEW[MapTagSizes]; -- released by FreeMapProps

  mapProps.tagSize[header] ← 0;
  FW.MakeTextItem[window: window,
                  myKey: 0,
                  tag: NIL,
                  initString: @mapProps.header,
                  width: 300, boxed: FALSE, readOnly: TRUE];

  mapProps.tagSize[from] ← Measure[@label, "          "L];
  mapProps.tagSize[to] ← 0;

  FOR k: CARDINAL IN [1..mapProps.number] DO
    FW.MakeTextItem[window: window, tag: @label, width: 50,
                    myKey: 2*k - 1, initString: @map.from];
    FW.MakeTextItem[window: window, tag: NIL, width: 50,
                    myKey: 2*k, initString: @map.to,
                    nextOutOfProc: nextOut];
    map ← map.next;
    ENDLOOP;
  END;  -- of MakeMapItems


MakeNextMapItems: PROC [window: Window.Handle,
                        nextOut: FW.NextOutOfProc,
                        mapProps: K4.MapProperties,
                        z: UNCOUNTED ZONE] =
  BEGIN
  -- Add an entry in mapProps, and create
  -- the corresponding pair of window items.

  empty: XS.ReaderBody ← XS.FromSTRING [""L];
  map:   K4.Mapping ← z.NEW [K4.MappingRecord ← [
```

```
                        from: XS.CopyToNewReaderBody [@empty, z],
                        to:   XS.CopyToNewReaderBody [@empty, z],
                        next: NIL]];

  IF mapProps.map = NIL THEN mapProps.map ← map
  ELSE
    BEGIN
    FOR last: K4.Mapping ← mapProps.map, last.next DO
      IF last.next = NIL THEN {last.next ← map; EXIT};
      ENDLOOP;
    END;

  mapProps.number ← mapProps.number + 1;    -- update number of pairs

  FW.MakeTextItem[window: window, tag: NIL, width: 50,
                  myKey: 2*mapProps.number - 1,
                  initString: @map.from];
  FW.MakeTextItem[window: window, tag: NIL, width: 50,
                  myKey: 2*mapProps.number,
                  initString: @map.to,
                  nextOutOfProc: nextOut];
  END;  -- of MakeNextMapItems


LayMapOptions: FW.LayoutProc =
  BEGIN
  parm: K4.ConvertParms ← LOOPHOLE[clientData];
  LayMapItems [window, parm.mapProps, parm.zone];
  END;  -- of LayMapOptions


LayMapProps: FW.LayoutProc =
  BEGIN
  iconParm: K4.IconParms ← LOOPHOLE[clientData];
  LayMapItems [window, iconParm.mapProps, iconParm.heap];
  END;  -- of LayMapProps


LayMapItems: PROC [window: Window.Handle,
                   mapProps: K4.MapProperties,
                   z: UNCOUNTED ZONE] =
  BEGIN
  margin: CARDINAL = 5;
  spaceBetweenLines: CARDINAL = 3;
  preMargin, maxTag: CARDINAL ← 0;
  line: FW.Line;

  FOR k: MapItems IN MapItems DO
    maxTag ← MAX [maxTag, mapProps.tagSize[k]];
    ENDLOOP;

  line ← FW.AppendLine[window: window,
                       spaceAboveLine: 2 * spaceBetweenLines];
  FW.AppendItem[window: window, line: line,
              item: K4.MapItems.header.ORD,
              preMargin: Prespace[mapProps.tagSize[K4.MapItems.header],
                                  maxTag, margin]];

  preMargin ← Prespace[mapProps.tagSize[K4.MapItems.from],
                                  maxTag, margin];
  FOR k: CARDINAL IN [1..mapProps.number] DO
    line ← FW.AppendLine[window: window,
                         spaceAboveLine: 2 * spaceBetweenLines];
    FW.AppendItem[window: window, line: line,
                  item: 2*k - 1, preMargin: preMargin];
    FW.AppendItem[window: window, line: line,
                  item: 2*k, preMargin: 8];
    ENDLOOP;

  FW.Repaint [window];
  END;  -- of LayMapItems


LayNextMapItems: PROC [window: Window.Handle,
                       mapProps: K4.MapProperties] =
  BEGIN
```

```
    margin: CARDINAL = 5;
    spaceBetweenLines: CARDINAL = 3;
    preMargin: CARDINAL;
    maxTag: CARDINAL ← 0;
    line: FW.Line;
    lastTo: FW.ItemKey ← 2*mapProps.number;

    FOR k: MapItems IN MapItems DO
      maxTag ← MAX [maxTag, mapProps.tagSize[k]];
      ENDLOOP;

    line ← FW.AppendLine[window: window,
                         spaceAboveLine: 2 * spaceBetweenLines];
    preMargin ← Prespace[mapProps.tagSize[K4.MapItems.from],
                         maxTag, margin];
    FW.AppendItem[window: window, line: line,
                  item: lastTo - 1, preMargin: preMargin];
    FW.AppendItem[window: window, line: line,
                  item: lastTo, preMargin: 8];
    FW.Repaint [window];
    END;  -- of LayNextMapItems


  AddNextMapOptions: FW.NextOutOfProc =
    BEGIN
    -- If NEXTing out of the last field, make and lay a new pair

    parm: K4.ConvertParms ← LOOPHOLE[FW.GetClientData [window]];

    IF item = FW.NumberOfItems[window] THEN
      BEGIN
      MakeNextMapItems [window, AddNextMapOptions,
                        parm.mapProps, parm.zone];
      LayNextMapItems [window, parm.mapProps];
      END;
    END;  -- of AddNextMapOptions


  AddNextMapProps: FW.NextOutOfProc =
    BEGIN
    -- If NEXTing out of the last field, make and lay a new pair

    iconParm: K4.IconParms ← LOOPHOLE[FW.GetClientData [window]];

    IF item = FW.NumberOfItems[window] THEN
      BEGIN
      MakeNextMapItems [window, AddNextMapProps,
                        iconParm.mapProps, iconParm.heap];
      LayNextMapItems [window, iconParm.mapProps];
      END;
    END;  -- of AddNextMapProps


  SetMapOptions: PropertySheet.MenuItemProc =
    BEGIN
    parm: K4.ConvertParms ← LOOPHOLE[clientData];
    RETURN [SetMapItems [formWindow, parm.mapProps, parm.zone]];
    END;  -- of SetMapOptions


  SetMapProps: PropertySheet.MenuItemProc =
    BEGIN
    iconParm: K4.IconParms ← LOOPHOLE[clientData];
    RETURN [SetMapItems [formWindow, iconParm.mapProps, iconParm.heap]];
    END;  -- of SetMapProps


  SetMapItems: PROC [window: Window.Handle,
                     mapProps: K4.MapProperties,
                     z: UNCOUNTED ZONE]  RETURNS [BOOLEAN]=
    BEGIN
    current, prior: K4.Mapping;
    itemFrom, itemTo: FW.ItemKey;
    emptyFrom, emptyTo: BOOLEAN;

    IF NOT FW.HasAnyBeenChanged [window]
```

```
  OR mapProps = NIL THEN RETURN [TRUE];

  current ← mapProps.map;

  FOR itemFrom ← 1, itemFrom + 2
  WHILE itemFrom < FW.NumberOfItems[window] DO
    itemTo ← itemFrom + 1;

    IF FW.HasBeenChanged [window, itemFrom] THEN
        current.from ← UpdateBody [window, itemFrom, @current.from, z];
    IF FW.HasBeenChanged [window, itemTo] THEN
        current.to ← UpdateBody [window, itemTo, @current.to, z];

    emptyFrom ← XS.Empty [@current.from];
    emptyTo ← XS.Empty [@current.to];
    SELECT TRUE FROM
      emptyFrom AND NOT emptyTo =>
                BEGIN
                Msg ["Please Fill Left Field or clear Right Field"L];
                FW.SetInputFocus [window, itemFrom];
                RETURN [FALSE];
                END;
      emptyFrom AND emptyTo =>
                BEGIN
                -- remove empty pair
                mapProps.number ← mapProps.number - 1;
                XS.FreeReaderBytes [@current.from, z];
                XS.FreeReaderBytes [@current.to, z];
                IF current = mapProps.map THEN -- empty pair is first
                  BEGIN
                  mapProps.map ← current.next;
                  z.FREE [@current];
                  current ← mapProps.map;
                  prior ← mapProps.map;
                  END
                ELSE
                  BEGIN
                  prior.next ← current.next;
                  z.FREE [@current];
                  current ← prior.next
                  END;
                END;
      ENDCASE =>
                BEGIN
                prior ← current;
                current ← current.next;
                END;
    ENDLOOP;

  RETURN [TRUE];
  END;  -- of SetMapItems



<<==========  PRIVATE PROCEDURES FOR FONT SUBWINDOW  ==========>>


MakeFontOptions: FW.MakeItemsProc =
  BEGIN
  parm: K4.ConvertParms ← LOOPHOLE[clientData];
  MakeFontItems [window, parm.fonProps, parm.zone];
  END;  -- of MakeFontOptions


MakeFontProps: FW.MakeItemsProc =
  BEGIN
  iconParm: K4.IconParms ← LOOPHOLE[clientData];
  MakeFontItems [window, iconParm.fonProps, iconParm.heap];
  END;  -- of MakeFontProps


MakeFontItems: PROC [window: Window.Handle,
                             fonProps: K4.FontProperties,
                             z: UNCOUNTED ZONE] =
  BEGIN
```

```
    label: XS.ReaderBody;
    fontChoices: FW.ChoiceItems ← DESCRIPTOR [fChoices];
    fChoices: ARRAY [0..3) OF FW.ChoiceItem ← [
                  [string [choiceNumber: FontStyle.modern.ORD,
                           string: XString.FromSTRING["Modern"L]]],
                  [string [choiceNumber: FontStyle.classic.ORD,
                           string: XString.FromSTRING["Classic"L]]],
                  [string [choiceNumber: FontStyle.titan.ORD,
                           string: XString.FromSTRING["Titan"L]]] ];

    sizeChoices: FW.ChoiceItems ← DESCRIPTOR [sChoices];
    sChoices: ARRAY [0..5) OF FW.ChoiceItem ← [
                  [string [choiceNumber: 0,
                           string: XString.FromSTRING["8"L]]],
                  [string [choiceNumber: 1,
                           string: XString.FromSTRING["10"L]]],
                  [string [choiceNumber: 2,
                           string: XString.FromSTRING["12"L]]],
                  [string [choiceNumber: 3,
                           string: XString.FromSTRING["14"L]]],
                  [string [choiceNumber: 4,
                           string: XString.FromSTRING["18"L]]] ];
    MakeFontItems: PROC [fontNumber: CARDINAL,
                         styleKey: K4.FontItems,
                         sizeKey: K4.FontItems,
                         italicsKey: K4.FontItems,
                         boldKey: K4.FontItems] =
            BEGIN
            tagString: LONG STRING ← "Font x"L;
            tagString[tagString.length - 1] ← '0 + VAL[fontNumber];
            fonProps.tagSize[styleKey] ← Measure [@label, tagString];
            fonProps.tagSize[sizeKey] ← 0;
            fonProps.tagSize[boldKey] ← 0;
            fonProps.tagSize[italicsKey] ← 0;
            FW.MakeChoiceItem[
              window: window, tag: @label,
              myKey: styleKey.ORD,
              fullyDisplayed: FALSE,
              values: fontChoices,
              initChoice: VAL[fonProps.fonts[fontNumber].font]];
            FW.MakeChoiceItem[
              window: window, tag: NIL,
              myKey: sizeKey.ORD,
              fullyDisplayed: FALSE,
              values: sizeChoices,
              initChoice: VAL[fonProps.fonts[fontNumber].size]];
            FW.MakeBooleanItem[
              window: window, tag: NIL,
              myKey: boldKey.ORD,
              label: [string [XString.FromSTRING["Bold"L]]],
              initBoolean: fonProps.fonts[fontNumber].bold];
            FW.MakeBooleanItem[
              window: window, tag: NIL,
              myKey: italicsKey.ORD,
              label: [string [XString.FromSTRING["Italics"L]]],
              initBoolean: fonProps.fonts[fontNumber].italics];
            END;  -- of MakeFontItems


    IF fonProps.tagSize = NIL THEN
      fonProps.tagSize ← z.NEW[FontTagSizes];
                                        -- released by FreeFontProps

    MakeFontItems [0, font0, size0, italics0, bold0];
    MakeFontItems [1, font1, size1, italics1, bold1];
    MakeFontItems [2, font2, size2, italics2, bold2];
    MakeFontItems [3, font3, size3, italics3, bold3];
    MakeFontItems [4, font4, size4, italics4, bold4];
    MakeFontItems [5, font5, size5, italics5, bold5];
    MakeFontItems [6, font6, size6, italics6, bold6];
    MakeFontItems [7, font7, size7, italics7, bold7];
    MakeFontItems [8, font8, size8, italics8, bold8];
    MakeFontItems [9, font9, size9, italics9, bold9];
    END;  -- of MakeFontItems
```

```
LayFontOptions: FW.LayoutProc =
  BEGIN
  parm: K4.ConvertParms ← LOOPHOLE[clientData];
  LayFontItems [window, parm.fonProps, parm.zone];
  END;  -- of LayFontOptions


LayFontProps: FW.LayoutProc =
  BEGIN
  iconParm: K4.IconParms ← LOOPHOLE[clientData];
  LayFontItems [window, iconParm.fonProps, iconParm.heap];
  END;  -- of LayFontProps


LayFontItems: PROC [window: Window.Handle,
                    fonProps: K4.FontProperties,
                    z: UNCOUNTED ZONE] =
  BEGIN
  margin: CARDINAL = 5;
  maxTag: CARDINAL ← 0;
  line: FW.Line;
  ts: LONG POINTER TO FontTagSizes ← fonProps.tagSize;  -- accelerator

  LaySingleItem: PROC [itemKey: K4.FontItems,
                       tagSize, interline: CARDINAL] =
        BEGIN
        IF interline > 0 THEN
                line ← FW.AppendLine[window: window,
                          spaceAboveLine: interline];
        FW.AppendItem[window: window, line: line,
                item: itemKey.ORD,
                preMargin: IF interline = 0 THEN 8
                           ELSE Prespace[tagSize, maxTag, margin]];
        END;  -- of LaySingleItem

  FOR k: FontItems IN FontItems DO
    maxTag ← MAX [maxTag, fonProps.tagSize[k]];
    ENDLOOP;

  LaySingleItem[font0, ts[font0], 6];
  LaySingleItem[size0, ts[size0], 0];
  LaySingleItem[italics0, ts[italics0], 0];
  LaySingleItem[bold0, ts[bold0], 0];
  LaySingleItem[font1, ts[font1], 6];
  LaySingleItem[size1, ts[size1], 0];
  LaySingleItem[italics1, ts[italics1], 0];
  LaySingleItem[bold1, ts[bold1], 0];
  LaySingleItem[font2, ts[font2], 6];
  LaySingleItem[size2, ts[size2], 0];
  LaySingleItem[italics2, ts[italics2], 0];
  LaySingleItem[bold2, ts[bold2], 0];
  LaySingleItem[font3, ts[font3], 6];
  LaySingleItem[size3, ts[size3], 0];
  LaySingleItem[italics3, ts[italics3], 0];
  LaySingleItem[bold3, ts[bold3], 0];
  LaySingleItem[font4, ts[font4], 6];
  LaySingleItem[size4, ts[size4], 0];
  LaySingleItem[italics4, ts[italics4], 0];
  LaySingleItem[bold4, ts[bold4], 0];
  LaySingleItem[font5, ts[font5], 6];
  LaySingleItem[size5, ts[size5], 0];
  LaySingleItem[italics5, ts[italics5], 0];
  LaySingleItem[bold5, ts[bold5], 0];
  LaySingleItem[font6, ts[font6], 6];
  LaySingleItem[size6, ts[size6], 0];
  LaySingleItem[italics6, ts[italics6], 0];
  LaySingleItem[bold6, ts[bold6], 0];
  LaySingleItem[font7, ts[font7], 6];
  LaySingleItem[size7, ts[size7], 0];
  LaySingleItem[italics7, ts[italics7], 0];
  LaySingleItem[bold7, ts[bold7], 0];
  LaySingleItem[font8, ts[font8], 6];
  LaySingleItem[size8, ts[size8], 0];
  LaySingleItem[italics8, ts[italics8], 0];
  LaySingleItem[bold8, ts[bold8], 0];
  LaySingleItem[font9, ts[font9], 6];
```

```
LayFontOptions: FW.LayoutProc =
  BEGIN
  parm: K4.ConvertParms ← LOOPHOLE[clientData];
  LayFontItems [window, parm.fonProps, parm.zone];
  END;  -- of LayFontOptions


LayFontProps: FW.LayoutProc =
  BEGIN
  iconParm: K4.IconParms ← LOOPHOLE[clientData];
  LayFontItems [window, iconParm.fonProps, iconParm.heap];
  END;  -- of LayFontProps


LayFontItems: PROC [window: Window.Handle,
                    fonProps: K4.FontProperties,
                    z: UNCOUNTED ZONE] =
  BEGIN
  margin: CARDINAL = 5;
  maxTag: CARDINAL ← 0;
  line: FW.Line;
  ts: LONG POINTER TO FontTagSizes ← fonProps.tagSize;  -- accelerator

  LaySingleItem: PROC [itemKey: K4.FontItems,
                       tagSize, interline: CARDINAL] =
        BEGIN
        IF interline > 0 THEN
                line ← FW.AppendLine[window: window,
                          spaceAboveLine: interline];
        FW.AppendItem[window: window, line: line,
                item: itemKey.ORD,
                preMargin: IF interline = 0 THEN 8
                           ELSE Prespace[tagSize, maxTag, margin]];
        END;  -- of LaySingleItem

  FOR k: FontItems IN FontItems DO
    maxTag ← MAX [maxTag, fonProps.tagSize[k]];
    ENDLOOP;

  LaySingleItem[font0, ts[font0], 6];
  LaySingleItem[size0, ts[size0], 0];
  LaySingleItem[italics0, ts[italics0], 0];
  LaySingleItem[bold0, ts[bold0], 0];
  LaySingleItem[font1, ts[font1], 6];
  LaySingleItem[size1, ts[size1], 0];
  LaySingleItem[italics1, ts[italics1], 0];
  LaySingleItem[bold1, ts[bold1], 0];
  LaySingleItem[font2, ts[font2], 6];
  LaySingleItem[size2, ts[size2], 0];
  LaySingleItem[italics2, ts[italics2], 0];
  LaySingleItem[bold2, ts[bold2], 0];
  LaySingleItem[font3, ts[font3], 6];
  LaySingleItem[size3, ts[size3], 0];
  LaySingleItem[italics3, ts[italics3], 0];
  LaySingleItem[bold3, ts[bold3], 0];
  LaySingleItem[font4, ts[font4], 6];
  LaySingleItem[size4, ts[size4], 0];
  LaySingleItem[italics4, ts[italics4], 0];
  LaySingleItem[bold4, ts[bold4], 0];
  LaySingleItem[font5, ts[font5], 6];
  LaySingleItem[size5, ts[size5], 0];
  LaySingleItem[italics5, ts[italics5], 0];
  LaySingleItem[bold5, ts[bold5], 0];
  LaySingleItem[font6, ts[font6], 6];
  LaySingleItem[size6, ts[size6], 0];
  LaySingleItem[italics6, ts[italics6], 0];
  LaySingleItem[bold6, ts[bold6], 0];
  LaySingleItem[font7, ts[font7], 6];
  LaySingleItem[size7, ts[size7], 0];
  LaySingleItem[italics7, ts[italics7], 0];
  LaySingleItem[bold7, ts[bold7], 0];
  LaySingleItem[font8, ts[font8], 6];
  LaySingleItem[size8, ts[size8], 0];
  LaySingleItem[italics8, ts[italics8], 0];
  LaySingleItem[bold8, ts[bold8], 0];
  LaySingleItem[font9, ts[font9], 6];
```

```
  LaySingleItem[size9, ts[size9], 0];
  LaySingleItem[italics9, ts[italics9], 0];
  LaySingleItem[bold9, ts[bold9], 0];
  FW.Repaint [window];
  END;  -- of LayFontItems


SetFontOptions: PropertySheet.MenuItemProc =
  BEGIN
  parm: K4.ConvertParms ← LOOPHOLE[clientData];
  RETURN [SetFontItems [formWindow, parm.fonProps, parm.zone]];
  END;  -- of SetFontOptions


SetFontProps: PropertySheet.MenuItemProc =
  BEGIN
  iconParm: K4.IconParms ← LOOPHOLE[clientData];
  RETURN [SetFontItems [formWindow, iconParm.fonProps, iconParm.heap]];
  END;  -- of SetFontProps


SetFontItems: PROC [window: Window.Handle,
                    fonProps: K4.FontProperties,
                    z: UNCOUNTED ZONE]
            RETURNS [BOOLEAN] =
  BEGIN

  Boolean: PROCEDURE [item: FontItems] RETURNS [BOOLEAN] = INLINE
        {RETURN [FW.GetBooleanItemValue [window, item.ORD]]};

  Choice: PROCEDURE [item: FontItems] RETURNS [CARDINAL] = INLINE
        {RETURN [FW.GetChoiceItemValue [window, item.ORD] ]};

  IF NOT FW.HasAnyBeenChanged [window] THEN RETURN [TRUE];

  fonProps.fonts[0].font ← VAL [Choice [K4.FontItems.font0]];
  fonProps.fonts[0].size ← VAL [Choice [K4.FontItems.size0]];
  fonProps.fonts[0].bold ← Boolean [K4.FontItems.bold0];
  fonProps.fonts[0].italics ← Boolean [K4.FontItems.italics0];
  fonProps.fonts[1].font ← VAL [Choice [K4.FontItems.font1]];
  fonProps.fonts[1].size ← VAL [Choice [K4.FontItems.size1]];
  fonProps.fonts[1].bold ← Boolean [K4.FontItems.bold1];
  fonProps.fonts[1].italics ← Boolean [K4.FontItems.italics1];
  fonProps.fonts[2].font ← VAL [Choice [K4.FontItems.font2]];
  fonProps.fonts[2].size ← VAL [Choice [K4.FontItems.size2]];
  fonProps.fonts[2].bold ← Boolean [K4.FontItems.bold2];
  fonProps.fonts[2].italics ← Boolean [K4.FontItems.italics2];
  fonProps.fonts[3].font ← VAL [Choice [K4.FontItems.font3]];
  fonProps.fonts[3].size ← VAL [Choice [K4.FontItems.size3]];
  fonProps.fonts[3].bold ← Boolean [K4.FontItems.bold3];
  fonProps.fonts[3].italics ← Boolean [K4.FontItems.italics3];
  fonProps.fonts[4].font ← VAL [Choice [K4.FontItems.font4]];
  fonProps.fonts[4].size ← VAL [Choice [K4.FontItems.size4]];
  fonProps.fonts[4].bold ← Boolean [K4.FontItems.bold4];
  fonProps.fonts[4].italics ← Boolean [K4.FontItems.italics4];
  fonProps.fonts[5].font ← VAL [Choice [K4.FontItems.font5]];
  fonProps.fonts[5].size ← VAL [Choice [K4.FontItems.size5]];
  fonProps.fonts[5].bold ← Boolean [K4.FontItems.bold5];
  fonProps.fonts[5].italics ← Boolean [K4.FontItems.italics5];
  fonProps.fonts[6].font ← VAL [Choice [K4.FontItems.font6]];
  fonProps.fonts[6].size ← VAL [Choice [K4.FontItems.size6]];
  fonProps.fonts[6].bold ← Boolean [K4.FontItems.bold6];
  fonProps.fonts[6].italics ← Boolean [K4.FontItems.italics6];
  fonProps.fonts[7].font ← VAL [Choice [K4.FontItems.font7]];
  fonProps.fonts[7].size ← VAL [Choice [K4.FontItems.size7]];
  fonProps.fonts[7].bold ← Boolean [K4.FontItems.bold7];
  fonProps.fonts[7].italics ← Boolean [K4.FontItems.italics7];
  fonProps.fonts[8].font ← VAL [Choice [K4.FontItems.font8]];
  fonProps.fonts[8].size ← VAL [Choice [K4.FontItems.size8]];
  fonProps.fonts[8].bold ← Boolean [K4.FontItems.bold8];
  fonProps.fonts[8].italics ← Boolean [K4.FontItems.italics8];
  fonProps.fonts[9].font ← VAL [Choice [K4.FontItems.font9]];
  fonProps.fonts[9].size ← VAL [Choice [K4.FontItems.size9]];
  fonProps.fonts[9].bold ← Boolean [K4.FontItems.bold9];
  fonProps.fonts[9].italics ← Boolean [K4.FontItems.italics9];
```

```
    RETURN [TRUE];
    END;  -- of SetFontItems



<<==========  PRIVATE PROCEDURES FOR GENERAL SUBWINDOW  ==========>>


MakeGeneralItems: FW.MakeItemsProc =
  BEGIN
  iconParm: K4.IconParms ← LOOPHOLE[clientData];
  label: XS.ReaderBody;
  speedChoices: FW.ChoiceItems ← DESCRIPTOR [sChoices];
  sChoices: ARRAY [0..3) OF FW.ChoiceItem ← [
              [string [choiceNumber: 0,
                       string: XString.FromSTRING["9600"L]]],
              [string [choiceNumber: 1,
                       string: XString.FromSTRING["4800"L]]],
              [string [choiceNumber: 2,
                       string: XString.FromSTRING["300"L]]] ];

  IF iconParm.genProps.tagSize = NIL THEN
    iconParm.genProps.tagSize ← iconParm.heap.NEW[GeneralTagSizes];
                      -- released by FreeIconProps

  iconParm.genProps.tagSize[iconName] ← Measure[@label, "Icon Name"L];
  FW.MakeTextItem[window: window,
                  myKey: K4.GeneralItems.iconName.ORD,
                  tag: @label,
                  initString: @iconParm.genProps.iconName,
                  width: 200];

  iconParm.genProps.tagSize[channelSpeed] ← Measure [@label, "RS232C Speed"L];
  FW.MakeChoiceItem[window: window,
    myKey: K4.GeneralItems.channelSpeed.ORD, tag: @label,
    fullyDisplayed: FALSE,
    values: speedChoices,
    initChoice: VAL[iconParm.genProps.channelSpeed]];

  iconParm.genProps.tagSize[folderName] ← Measure[@label, "Transmission Folder Name"L];
  FW.MakeTextItem[window: window,
                  myKey: K4.GeneralItems.folderName.ORD,
                  tag: @label,
                  initString: @iconParm.genProps.folderName,
                  width: 200];

  iconParm.genProps.tagSize[others] ← Measure[@label, "Required Tailor Choices"L];
  FW.MakeTextItem[window: window,
                  myKey: K4.GeneralItems.others.ORD,
                  tag: @label,
                  initString: @iconParm.genProps.others,
                  width: 300, boxed: FALSE, readOnly: TRUE];

  END;  -- of MakeGeneralItems


LayGeneralItems: FW.LayoutProc =
  BEGIN
  iconParm: K4.IconParms ← LOOPHOLE[clientData];
  margin: CARDINAL = 5;
  spaceBetweenLines: CARDINAL = 3;
  maxTag: CARDINAL ← 0;
  line: FW.Line;

  FOR k: K4.GeneralItems IN K4.GeneralItems DO
    maxTag ← MAX [maxTag, iconParm.genProps.tagSize[k]];
    ENDLOOP;

  line ← FW.AppendLine[window: window,
                       spaceAboveLine: 2 * spaceBetweenLines];
  FW.AppendItem[
      window: window, item: K4.GeneralItems.iconName.ORD, line: line,
      preMargin: Prespace[iconParm.genProps.tagSize[iconName], maxTag, margin]];
```

```
    line ← FW.AppendLine[window: window,
                         spaceAboveLine: 2 * spaceBetweenLines];
    FW.AppendItem[
        window: window, item: K4.GeneralItems.channelSpeed.ORD, line: line,
        preMargin: Prespace[iconParm.genProps.tagSize[channelSpeed], maxTag, margin]];

    line ← FW.AppendLine[window: window,
                         spaceAboveLine: 2 * spaceBetweenLines];
    FW.AppendItem[
        window: window, item: K4.GeneralItems.folderName.ORD, line: line,
        preMargin: Prespace[iconParm.genProps.tagSize[folderName], maxTag, margin]];

    line ← FW.AppendLine[window: window,
                         spaceAboveLine: 4 * spaceBetweenLines];
    FW.AppendItem[
        window: window, item: K4.GeneralItems.others.ORD, line: line,
        preMargin: Prespace[iconParm.genProps.tagSize[others], maxTag, margin]];

    FW.Repaint [window];
    END;  -- of LayGeneralItems


SetGeneralItems: PropertySheet.MenuItemProc =
  BEGIN
  nsSelections: NSFile.Selections ← [];
  nsName: NSString.String;
  attributeList: ARRAY[0..1) OF NSFile.Attribute;
  ps: K4.IconParms ← LOOPHOLE[clientData];

  IF FW.HasBeenChanged [formWindow, GeneralItems.iconName.ORD] THEN
    BEGIN
    ps.genProps.iconName ← UpdateBody [formWindow,
                                       GeneralItems.iconName.ORD,
                                       @ps.genProps.iconName,
                                       ps.heap];
    IF XS.Empty [@ps.genProps.iconName] THEN
      BEGIN
      Msg ["Icon name cannot be empty"L];
      RETURN [FALSE];
      END;
    nsName ← XS.NSStringFromReader [@ps.genProps.iconName, ps.heap];
    attributeList[0] ← [name [nsName]];
    NSFile.ChangeAttributes [ps.iconFile, DESCRIPTOR[attributeList]];
    NSString.FreeString [ps.heap, nsName];
    END;

  IF FW.HasBeenChanged [formWindow, GeneralItems.folderName.ORD] THEN
    BEGIN
    ps.genProps.folderName ← UpdateBody [formWindow,
                                         GeneralItems.folderName.ORD,
                                         @ps.genProps.folderName,
                                         ps.heap];
    IF XS.Empty [@ps.genProps.folderName] THEN
      BEGIN
      Msg ["Folder name cannot be empty"L];
      RETURN [FALSE];
      END;
    END;

  ps.genProps.channelSpeed ← VAL [FW.GetChoiceItemValue[
                                  formWindow,
                                  K4.GeneralItems.channelSpeed.ORD]];
  RETURN [TRUE];
  END;  -- of SetGeneralItems




<<==========  PRIVATE PROCEDURES FOR LINK SUBWINDOW  ==========>>


MakeLinkOptions: FW.MakeItemsProc =
  BEGIN
  parm: K4.ConvertParms ← LOOPHOLE[clientData];
  sheetChoices: FW.ChoiceItems ← DESCRIPTOR [sChoices];
```

```
    sChoices: ARRAY [0..3) OF FW.ChoiceItem ← [
                  [string [choiceNumber: 0,
                           string: XString.FromSTRING["Document"L]]],
                  [string [choiceNumber: 1,
                           string: XString.FromSTRING["Mapping"L]]],
                  [string [choiceNumber: 2,
                           string: XString.FromSTRING["Fonts"L]]] ];

    MakeLinkItems [window, sheetChoices, 0, SwapOptions];
    END;  -- of MakeLinkOptions


  MakeLinkProps: FW.MakeItemsProc =
    BEGIN
    iconParm: K4.IconParms ← LOOPHOLE[clientData];
    sheetChoices: FW.ChoiceItems ← DESCRIPTOR [sChoices];
    sChoices: ARRAY [0..4) OF FW.ChoiceItem ← [
                  [string [choiceNumber: 0,
                           string: XString.FromSTRING["Icon"L]]],
                  [string [choiceNumber: 1,
                           string: XString.FromSTRING["Document"L]]],
                  [string [choiceNumber: 2,
                           string: XString.FromSTRING["Mapping"L]]],
                  [string [choiceNumber: 3,
                           string: XString.FromSTRING["Font"L]]] ];

    MakeLinkItems [window, sheetChoices, 1, SwapProps];
    END;  -- of MakeLinkProps


  MakeLinkItems: PROC [window: Window.Handle,
                       choices: FW.ChoiceItems,
                       firstChoice: FW.ItemKey,
                       swapProc: FW.ChoiceChangeProc] =
    BEGIN
    label: XS.ReaderBody ← XS.FromSTRING["Sheet for: "L];

    FW.MakeChoiceItem[window: window,
                      myKey: 0, tag: @label,
                      values: choices,
                      initChoice: firstChoice,
                      changeProc: swapProc];
    END;  -- of MakeLinkItems


  LayLinkItem: FW.LayoutProc =
    BEGIN
    line: FW.Line ← FW.AppendLine[window: window,
                                  spaceAboveLine: 9];
    FW.AppendItem[window: window, item: 0, line: line, preMargin: 5];
    END;  -- of LayLinkItem


  SwapOptions: FW.ChoiceChangeProc =
    BEGIN
    parm: K4.ConvertParms ← LOOPHOLE[FW.GetClientData [window]];
    newMake: FW.MakeItemsProc;
    newLay: FW.LayoutProc;
    newSet: PropertySheet.MenuItemProc;

    SELECT newValue FROM
      0       => {newMake ← MakeTextOptions;
                  newLay ← LayTextOptions;
                  newSet ← SetTextOptions};
      1       => {newMake ← MakeMapOptions;
                  newLay ← LayMapOptions;
                  newSet ← SetMapOptions};
      ENDCASE => {newMake ← MakeFontOptions;
                  newLay ← LayFontOptions;
                  newSet ← SetFontOptions};

    [] ← PropertySheet.SwapFormWindows [
                  shell: parm.optionSheet,
                  apply: TRUE,
                  newFormWindowItems: newMake,
                  newFormWindowItemsLayout: newLay,
```

```
                newMenuItemProc: newSet];
  END;  -- of SwapOptions


SwapProps: FW.ChoiceChangeProc =
  BEGIN
  iconParm: K4.IconParms ← LOOPHOLE[FW.GetClientData [window]];
  newMake: FW.MakeItemsProc;
  newLay: FW.LayoutProc;
  newSet: PropertySheet.MenuItemProc;

  SELECT newValue FROM
    1       => {newMake ← MakeTextProps;
                newLay ← LayTextProps;
                newSet ← SetTextProps};
    2       => {newMake ← MakeMapProps;
                newLay ← LayMapProps;
                newSet ← SetMapProps};
    3       => {newMake ← MakeFontProps;
                newLay ← LayFontProps;
                newSet ← SetFontProps};
    ENDCASE => {newMake ← MakeGeneralItems;
                newLay ← LayGeneralItems;
                newSet ← SetGeneralItems};

  [] ← PropertySheet.SwapFormWindows [
                shell: iconParm.propSheet,
                apply: TRUE,
                newFormWindowItems: newMake,
                newFormWindowItemsLayout: newLay,
                newMenuItemProc: newSet];

  END;  -- of SwapProps


TakeDownPSheet: PropertySheet.MenuItemProc =
  BEGIN
  data: Containee.Data;
  nsSelections: NSFile.Selections ← [];
  iconParm: K4.IconParms ← LOOPHOLE[clientData];
  z: UNCOUNTED ZONE ← iconParm.heap;

  IF menuItem = done THEN
    BEGIN
    IF iconParm.mapProps # NIL AND iconParm.mapProps.number = 1 THEN
      BEGIN     -- check if only entry is empty
      IF  XS.Empty [@iconParm.mapProps.map.from]
      AND XS.Empty [@iconParm.mapProps.map.to]
                THEN iconParm.mapProps.number ← 0;
      END;
    K4.StoreFiledData [iconParm];
    IF iconParm.changeProc # NIL THEN
      BEGIN
      data ← [reference: NSFile.GetReference[iconParm.iconFile]];
      nsSelections.interpreted[name] ← TRUE;
      iconParm.changeProc [changeProcData: iconParm.changeProcData,
                           data: @data,
                           changedAttributes: nsSelections,
                           noChanges: FALSE];
      END;
    END;

  K4.FreeIconProps [iconParm.genProps, z];
  K4.FreeTextProps [iconParm.docProps, z];
  K4.FreeMapProps [iconParm.mapProps, z];
  K4.FreeFontProps [iconParm.fonProps, z];
  NSFile.Close [iconParm.iconFile];
  z.FREE [@iconParm];
  Heap.Delete [z];
  RETURN [TRUE];
  END;  -- of TakeDownPSheet


<<==========  OTHER PRIVATE PROCEDURES  ==========>>
```

```
Measure: PROCEDURE[label: XS.Reader, string: LONG STRING]
          RETURNS [size: CARDINAL] =
  {label↑ ← XS.FromSTRING[string];
   size ← SimpleTextDisplay.MeasureString[label].width};


Prespace: PROCEDURE [size, maxSize, margin: CARDINAL]
             RETURNS [preMargin: CARDINAL] =
  BEGIN
  preMargin ← margin + (maxSize - size) + (IF size = 0
                                            THEN 8 ELSE 0);
  END;


UpdateBody: PROC [window: Window.Handle, item: FW.ItemKey,
                  oldReader: XS.Reader, zone: UNCOUNTED ZONE]
         RETURNS [newReaderBody: XS.ReaderBody] =
  BEGIN
  rb: XS.ReaderBody ← FW.LookAtTextItemValue [window, item];

  XS.FreeReaderBytes [oldReader, zone];
  newReaderBody ← XS.CopyToNewReaderBody [@rb, zone];
  FW.DoneLookingAtTextItemValue [window, item];
  END;  -- of UpdateBody


Msg: PROCEDURE [message: LONG STRING] =
  BEGIN
  msgRB: XS.ReaderBody ← XS.FromSTRING [message];
  Attention.Post [@msgRB];
  END;  -- of Msg



END.  -- of K4PSheetImpl
14-Jan-87 15:42:32 created from DestTextPSheetImpl.
 6-Feb-87 10:12:41 fixed storage leak (folderName/outputName not deallocated).
12-Mar-87 13:48:25 made SetText/FontOptions PUBLIC.
16-Mar-87 10:00:54 added signature.
17-Mar-87 15:37:09 added character mapping.
26-Mar-87 11:27:35 made channel speed singly-displayed.
31-Mar-87 14:12:20 changes to StoreFiledData, no longer releasing memory.
31-Mar-87 16:30:15 tried to fix nexting out of mappings.
 7-Apr-87 17:23:53 Taylor => Tailor.
13-Apr-87 13:13:26 set mapProps.number to 0 if only entry is empty.
14-Apr-87 14:32:18 added paragraph properties.
20-Apr-87 17:45:17 initial value of channel speed not picked up from filed data.
24-Apr-87 16:56:45 removed canvas property/option sheets.
27-Apr-87 13:21:20 transmission folder name, page breaks.
28-Apr-87 16:57:56 checked for empty icon, folder or document names
```

```
<< File: K4WindowImpl.mesa -   15-Sep-88  9:16:10

deLaBeaujardiere:OSBU North:Xerox (deLaBeaujardiere.PA)

Copyright (C) 1986 by Xerox Corporation. All rights reserved.

>>

DIRECTORY Attention, Containee, Environment, Heap,
          K4, MenuData, MessageWindow,
          NSFile, NSFileStream, NSString, Process,
          RS232C, RS232CCorrespondents, RS232CEnvironment,
          Selection, StarDesktop, StarFileTypes,
          StarWindowShell, StarWindowShellExtra2,
          Stream, TIP, Window, XFormat, XString;

K4WindowImpl: MONITOR

IMPORTS   Attention, Heap, K4, MenuData, MessageWindow,
          NSFile, NSFileStream, NSString,
          Process, RS232C, Selection,
          StarDesktop, StarWindowShell, StarWindowShellExtra2,
          Stream, XFormat, XString

EXPORTS   K4 =

BEGIN
        OPEN SWS: StarWindowShell, XS: XString;


Variables: TYPE = LONG POINTER TO VariableObject;
VariableObject: TYPE = RECORD [
  window:                Window.Handle,
  channel:               RS232C.ChannelHandle,
  commParamObject:       RS232C.CommParamObject,
  data:                  RS232C.PhysicalRecord,
  windowClosing:         BOOLEAN ← FALSE,
  listener:              PROCESS ← NIL];


parms:  K4.IconParms;
vars:   Variables;

bufferSize: CARDINAL = 512;


OpenWindow: PUBLIC PROCEDURE [iconData: Containee.DataHandle,
                              changeProc: Containee.ChangeProc,
                              changeProcData: LONG POINTER,
                              tinyIcon: XS.Character]
                    RETURNS [shell: SWS.Handle ← SWS.nullHandle] =
  BEGIN
  wDims:    Window.Dims ← [500, 230];
  wPlace:   Window.Place ← [50, 30];
  wLines:   CARDINAL ← 10;
  mismatch: BOOLEAN;
  zone:     UNCOUNTED ZONE ← Heap.Create [4];  --** why 4?
                       -- deleted by ShellClosing
  reconvert: XString.ReaderBody ← XString.FromSTRING["Make Document"L];
  command: ARRAY[0..1) OF MenuData.ItemHandle ← [
                                MenuData.CreateItem[
                                                zone: zone,
                                                name: @reconvert,
                                                proc: Reconvert]];

               --get memory for IconParms and fill it
               -- (freed by ShellClosing, unless there is
               --   a software/icon mismatch).
  parms ← zone.NEW [K4.IconParmsRecord];
  parms.heap ← zone;
  parms.iconFile ← NSFile.OpenByReference [iconData.reference];
                                -- closed by ShellClosing
  mismatch ← K4.LoadFiledData [parms]; -- unloaded by ShellClosing
  IF mismatch THEN
    BEGIN
    msg: XS.ReaderBody ← XS.FromSTRING [
```

```
                "Obsolete icon does not work with new software"L];
    Attention.Post [@msg];
    NSFile.Close [parms.iconFile];
    zone.FREE [@parms];
    Heap.Delete [zone];
    RETURN [SWS.nullHandle];
    END;
                -- get memory for Variable
                -- (will be freed by ShellClosing).
  vars ← zone.NEW[VariableObject];

                -- create window shell and message window
  shell ← SWS.Create [namePicture: tinyIcon,
                      name: @parms.genProps.iconName,
                      scrollData: SWS.vanillaScrollData,
                      isCloseLegalProc: ShellClosing];
  SWS.SetRegularCommands [sws: shell,
                          commands: MenuData.CreateMenu [
                                zone: zone,
                                title: NIL,
                                array: DESCRIPTOR[command]]];
  vars.window ← SWS.CreateBody [shell];
  StarWindowShellExtra2.SetPreferredInteriorDims[sws: shell,
                                                 dims: wDims];
  MessageWindow.Create [vars.window, zone, wLines];
  MsgDate [];

  ReadyRS232C [vars, parms.genProps.channelSpeed];

  vars.listener ← FORK GetTransmissionAndConvert [vars, parms];
  END;  -- of OpenWindow


PutFileInFolder: PUBLIC PROCEDURE [file: NSFile.Handle,
                                   folderName: NSString.String] =
  BEGIN
  -- This proc puts a file
  -- in the named folder on the desktop.
  -- If the folder is not found, a new one is created.
  -- If the folder creation fails, we leave the file
  -- on the desktop as a last resort...

  dateOrdered: key NSFile.Ordering;
  folderFile: NSFile.Handle ← NSFile.nullHandle;
  folderAttrs: ARRAY [0..4) OF NSFile.Attribute;
  desktopFile: NSFile.Handle ← NSFile.OpenByReference [
                StarDesktop.GetCurrentDesktopFile []];

  BEGIN
  folderAttrs[0] ← [name [folderName]];
  folderFile ← NSFile.Open [
        attributes: DESCRIPTOR [BASE[folderAttrs], 1],
        directory: desktopFile
        ! NSFile.Error =>
              BEGIN
              WITH error SELECT FROM
                access =>
                  SELECT problem FROM
                    fileNotFound =>
                      BEGIN
                      dateOrdered.ascending ← TRUE;
                      dateOrdered.key ← createdOn;
                      folderAttrs[1] ← [type[StarFileTypes.folder]];
                      folderAttrs[2] ← [isDirectory [TRUE]];
                      folderAttrs[3] ← [ordering[dateOrdered]];
                      folderFile ← NSFile.Create [
                              directory:  desktopFile,
                              attributes: DESCRIPTOR[folderAttrs]];
                      StarDesktop.AddReferenceToDesktop [
                              NSFile.GetReference [folderFile]];
                      END;
                    ENDCASE;
                ENDCASE;
              GOTO Done;
              END];
  EXITS Done => {};
```

```
  END;

  IF folderFile = NSFile.nullHandle THEN -- folder creation failed
    StarDesktop.AddReferenceToDesktop[NSFile.GetReference [file]]
  ELSE
    BEGIN
    NSFile.Move [file, folderFile];
    NSFile.Close [folderFile];
    END;
  NSFile.Close [desktopFile];
  END;  -- of PutFileInFolder



-- PRIVATE PROCEDURES


ShellClosing: ENTRY SWS.IsCloseLegalProc =
  BEGIN
  ENABLE UNWIND => NULL;
  z: UNCOUNTED ZONE;
  clearMask: RS232C.DeviceStatus ← [
                statusAborted: FALSE, dataLost: FALSE,
                breakDetected: FALSE, clearToSend: TRUE,
                dataSetReady: TRUE, carrierDetect: TRUE,
                ringHeard: FALSE, ringIndicator: FALSE,
                deviceError: FALSE];

  vars.windowClosing ← TRUE;
  IF vars.listener # NIL THEN
    BEGIN
    RS232C.Suspend [vars.channel, all];
    JOIN vars.listener;
    RS232C.SetParameter [vars.channel, [latchBitClear [clearMask]]];
    RS232C.Delete [vars.channel];
    vars.listener ← NIL;
    END;

  IF parms.changeProc # NIL THEN parms.changeProc[
                        changeProcData: parms.changeProcData,
                        data: parms.iconData,
                        noChanges: TRUE];
  MessageWindow.Destroy [vars.window];  -- clear message resources
  NSFile.Close [parms.iconFile];
  z ← parms.heap;
  z.FREE [@vars];
  K4.FreeIconProps [parms.genProps, z];
  K4.FreeTextProps [parms.docProps, z];
  K4.FreeMapProps [parms.mapProps, z];
  K4.FreeFontProps [parms.fonProps, z];
  z.FREE [@parms];
  Heap.Delete [z];
  RETURN[TRUE];
  END;  -- of ShellClosing


Msg: PROC [message: LONG STRING, startOnNewLine: BOOLEAN] =
  BEGIN
  picture: XFormat.Object ← MessageWindow.XFormatObject [vars.window];
  IF startOnNewLine THEN
    BEGIN
    MessageWindow.PostSTRING [vars.window, " "L, TRUE];
    XFormat.Date [h: @picture, format: timeOnly];
    MessageWindow.PostSTRING [vars.window, " "L, FALSE];
    END;
  MessageWindow.PostSTRING [vars.window, message, FALSE];
  END;  -- of Msg


MsgDate: PROC =
  BEGIN
  <<
  pic: XFormat.Object ← MessageWindow.XFormatObject [vars.window];
  MessageWindow.PostSTRING [vars.window, " "L, TRUE];
  XFormat.Date [h: @pic, format: dateOnly];
```

```
  >>
  END;  -- of MsgDate


MsgDecimal: PROC [number: LONG CARDINAL] =
  BEGIN
  picture: XFormat.Object ← MessageWindow.XFormatObject [vars.window];
  XFormat.Decimal [h: @picture, n: number];
  END;  -- of MsgDecimal


ReadyRS232C: PROCEDURE [vars: Variables,
                        optionSpeed: K4.ChannelSpeed] =
  BEGIN
  speed: RS232C.LineSpeed ← SELECT optionSpeed FROM
                                ninetySix     => bps9600,
                                fortyEight    => bps4800,
                                three         => bps300,
                                ENDCASE       => bps9600;
  clearMask: RS232C.DeviceStatus ← [
               statusAborted: FALSE, dataLost: FALSE,
               breakDetected: FALSE, clearToSend: TRUE,
               dataSetReady: TRUE, carrierDetect: TRUE,
               ringHeard: FALSE, ringIndicator: FALSE,
               deviceError: FALSE];
  vars.commParamObject  ← [duplex: full,
                           lineType: asynchronous,
                           lineSpeed: speed,
                           accessDetail: directConn[]];

  vars.channel ← RS232C.Create [
            lineNumber: RS232C.GetNextLine [RS232C.nullLineNumber],
            commParams: @vars.commParamObject,
            preemptOthers: preemptAlways,
            preemptMe: preemptAlways];

  RS232C.SetParameter [vars.channel, [charLength [8]]];
  RS232C.SetParameter [vars.channel,
    [correspondent [RS232CCorrespondents.ttyHost]]];
  RS232C.SetParameter [vars.channel, [frameTimeout [1000]]];
  RS232C.SetParameter [vars.channel, [lineSpeed [speed]]];
  RS232C.SetParameter [vars.channel, [parity [none]]];
  RS232C.SetParameter [vars.channel, [stopBits [1]]];
  RS232C.SetParameter [vars.channel, [latchBitClear [clearMask]]];
  RS232C.SetParameter [vars.channel, [dataTerminalReady [TRUE]]];
  RS232C.SetParameter[vars.channel, [requestToSend [TRUE]]];
  RS232C.SetParameter[vars.channel,
              [flowControl [[type: xOnXOff,
                            xOn: 17,          -- DC1 (↑q)
                            xOff: 19]]]];     -- DC3 (↑s)
  END;  -- of ReadyRS232C


GetTransmissionAndConvert: PROCEDURE [vars: Variables,
                                      parms: K4.IconParms] =
  BEGIN
  byteArray:     PACKED ARRAY [0..bufferSize) OF Environment.Byte;
  logAttrs:      ARRAY[0..2) OF NSFile.Attribute ← [
                       [name [NSString.StringFromMesaString["K4TEXT.LOG"L]]],
                       [type [2]]];
  logFile:       NSFile.Handle;
  logStream:     NSFileStream.Handle;
  folderName:    NSString.String;
  dots:          CARDINAL;       -- to count dots announcing reception
  bytesLogged:   LONG CARDINAL;
  bytesReceived:         CARDINAL;
  transferStatus:        RS232C.TransferStatus;

  Process.SetPriority [Process.priorityNormal];
  vars.data ← [header: Environment.nullBlock,
               body: [blockPointer: @byteArray,
                      startIndex: 0,
                      stopIndexPlusOne: bufferSize],
               trailer: Environment.nullBlock];

  DO  -- run while window is open
```

```
    dots ← 0;
    bytesLogged ← 0;
    Msg ["Waiting for transmission from Kurzweil 4000"L, TRUE];
    [bytesReceived, transferStatus] ← GetBlock[vars];    -- request 1st block

    IF vars.windowClosing THEN RETURN;

    Msg ["Receiving"L, TRUE];    -- announce initial reception
    logFile ← NSFile.Create [directory:  NSFile.nullHandle,
                             attributes: DESCRIPTOR[logAttrs]];
    logStream ← NSFileStream.Create [logFile, FALSE];


    DO
      IF vars.windowClosing THEN
        BEGIN
        Stream.Delete [logStream];
        NSFile.Close [logFile];
        RETURN;
        END;

      IF transferStatus # success THEN EXIT;
      IF bytesReceived = 0 THEN LOOP;

      vars.data.body.stopIndexPlusOne ← bytesReceived;
      bytesLogged ← bytesLogged + bytesReceived;
      dots ← IF dots > 50 THEN 0 ELSE dots + 1;
      Msg ["."L, (dots = 0)];    -- continue announcing block reception

      Stream.PutBlock [logStream, vars.data.body];
      IF  byteArray[bytesReceived - 1] = 200B THEN EXIT;

      [bytesReceived, transferStatus] ← GetBlock[vars];
      ENDLOOP;

    IF bytesLogged < 5 THEN  -- don't bother saving 4 bytes or less...
      BEGIN
      Msg ["  Less than 5 bytes received. Ignored."L, FALSE];
      Stream.Delete [logStream];
      NSFile.Close [logFile];
      END
    ELSE
      BEGIN
      bytesLogged ← bytesLogged - 1; -- to drop the 200B added by K4000
      MsgDecimal [bytesLogged];
      Msg [" bytes received."L, FALSE];
      Stream.SendNow [logStream];
      NSFileStream.SetLength [logStream, bytesLogged];
      Stream.Delete [logStream];
      folderName ← XS.NSStringFromReader [@parms.genProps.folderName,
                                          parms.heap];
      PutFileInFolder [logFile, folderName];
      NSString.FreeString [parms.heap, folderName];
      Msg ["Spawning document creation."L, TRUE];
      K4.ConvertToDocument [logFile, parms.docProps,
                            parms.mapProps, parms.fonProps];
      END;
    ENDLOOP;

  END;  -- of GetTransmissionAndConvert


GetBlock: PROC [vars: Variables]
       RETURNS [count: CARDINAL, status: RS232C.TransferStatus] =
  BEGIN
  completionHandle: RS232C.CompletionHandle;

  vars.data.body.stopIndexPlusOne ← bufferSize;
  completionHandle ← RS232C.Get[vars.channel, @vars.data];
  [count, status] ← RS232C.TransferWait[vars.channel, completionHandle];
  END;  -- of GetBlock


Reconvert: MenuData.MenuProc =
  BEGIN
  --*** use here selection interpretation in K4ToVPUtility
```

```
  OPEN Selection;
  -- User has selected a log stream in document folder
  -- and wants to make a new document from it.
  fileSelection: Value ← Convert [Target.file];
  typeSelection: Value ← Convert [Target.fileType];
  typeAscii: NSFile.Type = 2;

  IF fileSelection.value = nullValue.value
  OR typeSelection.value = nullValue.value
  OR typeSelection.value↑ # typeAscii THEN
    BEGIN
    Msg ["Reconverting: there is no selection to convert."L,
         TRUE];
    END
  ELSE
    BEGIN
    refLoc: LONG POINTER TO NSFile.Reference ←
                              LOOPHOLE[fileSelection.value];
    logFile: NSFile.Handle ← NSFile.OpenByReference [refLoc↑];
                              -- file closed by conversion job
    Msg ["Spawning document creation."L, TRUE];
    K4.ConvertToDocument [logFile, parms.docProps, parms.mapProps, parms.fonProps];
    END;
  END;  -- of Reconvert


END.  -- of K4WindowImpl
LOG:
23-Jan-87 16:42:05 created from DestTextWindowImpl and K43WindowImpl.
 4-Feb-87 11:47:24 moved pSheet items into body window, removed Options command.
 6-Feb-87 10:13:34 fixed storage leak (folderName/outputName not deallocated).
 6-Feb-87 17:58:04 implemented ReceiveScannerData as background job.
13-Feb-87 10:56:09 redid entire RS232C interaction to prevent hanging in RS232C.Suspend.
16-Feb-87 14:06:07 added warning sheet to run Editor if idle.
18-Feb-87 15:47:10 added inspection of header to determine output type.
26-Feb-87 17:22:49 implemented IntervalTimer facility because text files from K4000 do not always have
etx at the end, and we need to time out after a while.
 3-Mar-87 10:20:32 incorporated call to ConvertToCanvas, request to run VP Free-hand Drawing, etc...
 4-Mar-87 15:58:47 made PutFileInFolder a public proc.
10-Mar-87 14:24:23 adapted to linked property sheet, removed panel window in preparation of installing
history window.
11-Mar-87 14:05:31 turned into a message window.
12-Mar-87 10:58:39 fixed messages/transmission coordination.
16-Mar-87 10:12:10 added signature.
18-Mar-87 14:38:46 cleared channel mask bits after deleting.
19-Mar-87 10:12:40 mapProps not initialized in SpinOffDocument.
23-Mar-87 15:21:14 added Reconvert command.
24-Mar-87 13:30:21 ordering folder by creation date.
 2-Apr-87 10:43:20 saving/naming of logFile moved to K4CanvasImpl or K4DocumentImpl.
 6-Apr-87 15:45:27 creation of ConvertParms moved from here to K4DocumentImpl and K4CanvasImpl.
 7-Apr-87 13:54:40 removed useless start code.
 7-Apr-87 14:28:32 moved back here saving of logFile; had problem with holding same handle in two
 processes.
 9-Apr-87 11:46:09 Looks like the logFile does not have all the data received; so now using
 block/PutBlock instead of string/PutString to save bytes received from channel.
14-Apr-87 14:44:48 removed saveLog.
15-Apr-87 15:07:44 remove flow control on RS232C channel, set timeout to 1000 millisecs.
18-Apr-87 16:27:55 restored flow control (still no clues about lost bytes: tried slower baud rates,
larger timeouts, 7 bits, high priority processes, flow control/no flow control, larger buffers, ..).
20-Apr-87 14:48:27 truncated the log stream by one byte because it seems that we always get one extra
byte (200B) at the end, perhaps generated by the Suspend[channel].
20-Apr-87 14:49:40 picked up channel speed from option sheet.
20-Apr-87 18:09:55 adjusted length of silence to channel speed.
24-Apr-87 16:43:38 dropped canvas and ArtScan processing.
27-Apr-87 10:08:52 "Reconvert" becomes "Make Document".
27-Apr-87 13:25:32 transmission folder name.
14-Sep-88  9:59:55 discovered that in VP2.0, we no longer get the last block; perhaps something with
the 5 seconds wait in lower priority process; also discovered with DLM that the last byte of last
transmission, 80X, is sent by Kurzweil, not added by Suspend or some such; thus, the code to wait a
while to see if transmission is ended is replaced by testing 80X.
15-Sep-88  9:13:53 cleared latch bits before deleting channel: I noticed that the DEST claims that the
port is not ready after Kurzweil application has been used; perhaps clearing the latch bits will help.
/
```

```
-- BootUserImplA.mesa
-- 17-Apr-89 16:19:56

-- Copyright (c) 1989 by Xerox Corporation.  All rights reserved.

DIRECTORY
  OthelloDefs USING [
    AbortingCommand, CloseFetch, CommandProcessor, Confirm, GetName,
    IndexTooLarge, LeaderPage, leaderPages, lpVersion, MyNameIs, NewLine,
    PackedTimeFromString, Question, ReadNumber, RegisterCommandProc,
    SetCommandString, WriteChar, WriteFixedWidthNumber, WriteLine,
    WriteLongNumber, WriteOctal, WriteString, Yes],
  OthelloOps USING [
    BadSwitches, BootFileType, DecodeSwitches, DeleteTempFiles, GetDriveSize,
    GetNextSubVolume, GetPhysicalVolumeBootFile, GetSwitches, GetVolumeBootFile,
    nullSubVolume, SetDebugger, SetDebuggerSuccess, SetExpirationDate,
    SetExpirationDateSuccess, SetGetSwitchesSuccess, SetPhysicalVolumeBootFile,
    SetSwitches, SubVolume, VoidPhysicalVolumeBootFile, VoidVolumeBootFile],
  OthelloToolDefs USING [CloseVolume],
  SpecialVolume USING [OpenVolume],
  TemporaryBooting USING [BootButton],
  Volume USING [
    Erase, GetAttributes, GetLabelString, GetType, ID,
    NeedsScavenging, NotOnline, nullID, Open, systemID, Type],

VolumeInitImplA: PROGRAM
  IMPORTS
    File, Heap, Inline, OthelloDefs, OthelloOps, OthelloToolDefs, PhysicalVolume, Process, Runtime,
    Scavenger, Space, SpecialVolume, System, String, TemporaryBooting, Volume,
    VolumeVersion
  EXPORTS OthelloDefs
  SHARES File =
  BEGIN OPEN OthelloOps, OthelloDefs;

  Quit: PROC = {TemporaryBooting.BootButton[]};

  SetPvBoot: PROC =
    BEGIN
    lvID: Volume.ID ← GetLvIDFromUser[].lvID;
    SpecialVolume.OpenVolume[lvID, read];
    FOR t: BootFileType IN [softMicrocode..pilot] DO
      IF GetVolumeBootFile[lvID, t].file # File.nullFile THEN {
        file: File.File;
        firstPage: File.PageNumber;
        [file, firstPage] ← GetVolumeBootFile[lvID, t];
        SetPhysicalVolumeBootFile[file, t, firstPage]};
      ENDLOOP;
    OthelloToolDefs.CloseVolume[lvID];
    END;

  GetUserLvID: PROC [] RETURNS [lvID: Volume.ID] =
    BEGIN
    DO
     ptmpID:      PhysicalVolume.ID ← PhysicalVolume.nullID;
     inputString: LONG STRING ← "User";
     matches:     CARDINAL ← 0;
      DO
        driveTemp: PhysicalVolume.Handle;
        ltmpID:    Volume.ID ← Volume.nullID;
        IF (ptmpID ← PhysicalVolume.GetNext[ptmpID]) = PhysicalVolume.nullID THEN EXIT;
        driveTemp ← PhysicalVolume.GetAttributes[ptmpID].instance;
        DO
          s: STRING = [maxNameLength];
          IF (ltmpID ← PhysicalVolume.GetNextLogicalVolume[ptmpID, ltmpID])
             = Volume.nullID THEN EXIT;
          GetLogicalVolumeName[ltmpID, s ! Volume.NotOnline => LOOP];
          IF FunnyEqual[driveTemp, s, inputString] THEN {
            matches ← matches + 1; lvID ← ltmpID; pvID ← ptmpID; drive ← driveTemp};
          ENDLOOP;
        ENDLOOP;
      SELECT matches FROM
        0 => WriteString["Not found\r"L];
        1 => RETURN;
        ENDCASE => WriteLine["Ambigous; please specify Device:LogicalName"L];
      ENDLOOP;
    END;


  GetLogicalVolumeName: PROC [vid: Volume.ID, s: STRING] = {
    s.length ← 0;
    Volume.GetLabelString[vid, s ! Volume.NeedsScavenging => GOTO bad];
    EXITS bad => {
      IDRep: TYPE = RECORD [p: ARRAY [0..3) OF CARDINAL, n: LONG CARDINAL];
      String.AppendString[s, "NeedsScavenging"L];
      String.AppendLongNumber[s, LOOPHOLE[vid, IDRep].n, 8]}};


  FunnyEqual: PROC [h: PhysicalVolume.Handle, name: STRING, userName: LONG STRING]
    RETURNS[BOOLEAN] = {

    SameChar: PROC [a, b: CHARACTER]
      RETURNS [BOOLEAN] = {
      IF a=b THEN RETURN[TRUE]
      ELSE IF a IN ['a..'z] AND b IN ['A..'Z] AND (a-'a+'A)=b THEN RETURN[TRUE]
      ELSE IF a IN ['A..'Z] AND b IN ['a..'z] AND (a-'A+'a)=b THEN RETURN[TRUE]
      ELSE RETURN[FALSE]};


```

```
    IF String.Equivalent[name, userName] THEN RETURN[TRUE];
    FOR i: CARDINAL IN [0..driveName.length) DO
      IF ~SameChar[driveName[i], userName[i]] THEN RETURN[FALSE] ENDLOOP;
    IF driveName.length+name.length+1 # userName.length THEN RETURN[FALSE];
    IF userName[driveName.length] # ': THEN RETURN[FALSE];
    FOR i: CARDINAL IN [0..name.length) DO
      IF ~SameChar[name[i], userName[driveName.length+1+i]] THEN RETURN[FALSE]
      ENDLOOP;
    RETURN[TRUE]};

  END.....
```

```
{InitDaybreak.mc
Last Revised:
JPM, 31-Dec-86 12:04:27 Add initialization of U-regs uGFI (for MDS relief) and uIdleCount*
SCJ   7-Aug-86 16:25:19 Removed definition of FPTEnabled and stuck it in DualBank.dfn & SingleBank.dfn with values 2 and 0, to indicate
whether or not Floating point is in uCode
JPM, 29-Jul-86 14:24:02 Remove InitDaybreak from emulator and make into initialization routine
JPM, 16-Jul-86  9:50:43 Make uFactoruCode value conditional (set to FPTEnabled iff two control store banks)
SCJ, 14-May-86  9:57:07 merged EIS InitDaybreak.mc with the existing InitDaybreak.mc to build a MesaDove with floating point microcode
DEG, 25-Jan-86 18:24:53 Initialize uFactoruCode and uTimesWplDisp
RDH, 10-Oct-85 16:03:38 Inserted ClrLOCK to test slow mode.
JPM, 30-Jul-85 10:38:38 Corrected offsetMaintPanel and offsetMesaProc
JPM,  3-Jul-85 11:54:16 Opie redesign conversion
JPM,  4-Jun-85  8:25:28 removed u1F
JPM, 11-Apr-85 17:35:31 changed value of uMaintPanel for Opie 19.
JPM, 13-Feb-85 13:13:55 added uMaintPanel init
JGS, 18-Dec-84 13:10:51 initialized uPPMask to E000}

{       Copyright (C) 1984, 1985, 1986 by Xerox Corporation.  All rights reserved.}

StartAddress[BootTrap];

Set[T0Count, 40];
Set[T1Count, 41];
Set[T2Count, 42];
Set[T012Control, 43];
Set[T0Disable, 48];
Set[T0Enable, 4C];
Set[T12Disable, 50];
Set[T12Enable, 54];

Set[T0Mode2, 34];
Set[T1Mode0, 50];
Set[T1Mode2, 74];
Set[T2Mode2, 0B4];
Set[T0InitialLSB, 035];
Set[T0InitialMSB, 0C]; {0C35 hex = 3125 decimal counts = 50 milliseconds}

Set[offsetMaintPanel, Lshift[8,1]];
Set[offsetMesaProc, Lshift[11,1]];

{Get here when the boot button is pushed, or somebody yanks on INIT/.}
BootTrap: {From trap branch in Refill.mc}
        ClrIntErr, ClrLOCK, CANCELBR[$,0F]  {must be c1},       c1, at[0];
        ClrMPIntIOP,                                            c2;
        G ← 0, uPCCross ← 0,                                    c3;

{set up timer counters}
        rhRx ← T0Disable,                                       c1;
        T ← T0InitialLSB,                                       c2;
        TT ← T0Mode2,                                           c3;

        IO ← [rhRx, 0],                                         c1;
        MDR ← 0, {disable counter 0}                            c2;
        rhRx ← T12Disable,                                      c3;

        IO ← [rhRx, 0],                                         c1;
        MDR ← 0, {disable counters 1 and 2}                     c2;
        rhRx ← T012Control,                                     c3;

        IO ← [rhRx, 0],                                         c1;
        MDR ← TT, {set counter 0 mode},                         c2;
        TT ← T2Mode2,                                           c3;

        IO ← [rhRx, 0],                                         c1;
        MDR ← TT, {set counter 2 mode},                         c2;
        rhRx ← T0Count,                                         c3;

        IO ← [rhRx, 0],                                         c1;
        MDR ← T, {set counter 0 initial count LSB},             c2;
        T ← T0InitialMSB,                                       c3;

        IO ← [rhRx, 0],                                         c1;
        MDR ← T, {set counter 0 initial count MSB},             c2;
        rhRx ← T2Count,                                         c3;

        IO ← [rhRx, 0],                                         c1;
        MDR ← 0, {set counter 2 initial count LSB},             c2;
        TT ← T1Mode0,                                           c3;

        IO ← [rhRx, 0],                                         c1;
        MDR ← 0, {set counter 2 initial count MSB},             c2;
        rhRx ← T0Enable,                                        c3;

        IO ← [rhRx, 0],                                         c1;
        MDR ← 0, {enable counter 0}                             c2;
        rhRx ← T12Enable,                                       c3;

        IO ← [rhRx, 0],                                         c1;
        MDR ← 0, {enable counters 1 and 2}                      c2;
        rhRx ← T012Control,                                     c3;

        IO ← [rhRx, 0],                                         c1;
        MDR ← TT, {set counter 1 setup mode},                   c2;
        TT ← T1Mode2,                                           c3;

        IO ← [rhRx, 0],                                         c1;
        MDR ← TT, {set counter 1 mode},                         c2;
```

```
        rhRx ← T1Count,                                         c3;

        IO ← [rhRx, 0],                                         c1;
        MDR ← 0, {set counter 1 initial count LSB},             c2;
        T ← 1,                                                  c3;

        IO ← [rhRx, 0],                                         c1;
        MDR ← 0, {set counter 1 initial count MSB}, UBrkByte ← 0, c2;
        uWDC ← T, ClrIE, {disable interrupts}                   c3;

SetupConstants:
{       TT ← uFactoruCodeVal,                                   c1;
        uFactoruCode ← TT,                                      c2;
        uTimesWplDisp ← 0,                                      c3;
*** uFactoruCode and uTimesWplDisp were removed since uFactoruCode clashes with uVirtPage and FPTEnabled performs the function done by
uFactoruCode. uTimesWplDisp was removed since it is used only for emulating special Versatec opcodes ***}

        TT ← RShift1 0, SE ← 1,                                 c1;
        u8000 ← TT,                                             c2;
        TT ← LShift1 0FF, SE←1,                                 c3;

        TT ← LShift1 TT, SE←1,                                  c1;
        TT ← TT LShift1, SE←1,                                  c2;
        u7FF ← TT, TT ← TT LShift1, SE ← 1,                     c3;

        TT ← TT LShift1, SE ← 1,                                c1;
        u1FFF ← TT, TT ← TT LShift1, SE ← 1,                    c2;
        u3FFF ← TT,                                             c3;

        TT ← 01F,                                               c1;
        TT ← TT LRot8,                                          c2;
        TT ← TT or 0F8,                                         c3;

        uPMask ← TT, {1FF8}                                     c1;
        uPMask2 ← TT,                                           c2;
        r0100 ← 0FF + 1, {0100}                                 c3;

        TT ← 0E0,                                               c1;
        TT ← TT LRot8, {E000}                                   c2;
        uPPMask ← TT,                                           c3;

        TT ← 64,                                                c1;
        TT ← TT LRot8,                                          c2;
        UtbFlags ← TT, {6400}                                   c3;

        Q ← rhIORgn ← 5,                                        c1;
        rIORgn ← 20,                                            c2;
        rIORgn ← rIORgn LRot8, {IORgn real addr = 52000}        c3;

        MAR ← [rhIORgn, rIORgn + offsetMaintPanel],             c1;
        uIORgnHigh ← Q, CANCELBR[$,0],                          c2;
        TT ← MD,                                                c3;

        TT ← TT LRot12,                                         c1;
        TT ← RShift1 TT, SE ← 0,                                c2;
        uMaintPanel ← TT,                                       c3;

        MAR ← [rhIORgn, rIORgn + offsetMesaProc],               c1;
        MAPA ← 4, CANCELBR[$,0], {VM map real addr = 40000H}    c2;
        TT ← MD,                                                c3;

        TT ← TT LRot12,                                         c1;
        TT ← RShift1 TT, SE ← 0,                                c2;
        uMesaProc ← TT,                                         c3;

SetUpEmulatorRegs:
        rhT ← xtFC0,                                            c1;
        UvMDS ← T ← 0,                                          c2;
        rhMDS ← 0, TOS ← 0,                                     c3;

        uGFI ← 0,                                               c1;
        uIdleCountLow ← 0,                                      c2;
        uIdleCountHigh ← 0,                                     c3;

        UvG ← rInt ← 0,                                         c1;
        uXTS ← stackP ← T,                                      c2;
        uWP ← T, PC ← T + 0 + 1{use carry}, SetMPIntIOP,        c3;

        TT ← 0FF LShift1, SE ← 1, {1FF}                         c1;
        T ← TT + 3{@SD[sBoot]},                                 c2;
        uDestLo ← T, L ← 0, ClrMPIntIOP,                        c3;

{initialization done, now loop until IOP halts the CP}
{emulator will resume from this address}

Linkage:
        GOTOABS[addrLinkage],                                   c1{c*}, at[addrLinkage];
```

```
-- Copyright (C) 1986 by Xerox Corporation. All rights reserved.
-- OthelloToolDefs.mesa
-- Created by NFS          4-Jun-86 10:49:17

DIRECTORY
  TTY USING [Handle],
  Volume USING [ID];
OthelloToolDefs:DEFINITIONS = {

  tty: TTY.Handle;

  Run: PROCEDURE;  -- instead of PilotClient.Run

  CloseVolume: PROCEDURE[volume: Volume.ID];

};
```

```
-- Copyright (C) 1986 by Xerox Corporation. All rights reserved.
-- OthelloToolImpl.mesa
-- Created by NFS        4-Jun-86 10:26:30

DIRECTORY
  Exec USING [AddCommand, ExecProc, RemoveCommand],
  OthelloToolDefs USING [Run],
  Process USING [Abort],
  Runtime USING [GetBcdTime],
  String USING [AppendString],
  Time USING [Append, Unpack],
  Tool USING [Create, Destroy, MakeSWsProc, MakeTTYSW, UnusedLogName],
  ToolWindow USING [Activate, TransitionProcType],
  TTY USING [Handle],
  TTYSW USING [GetTTYHandle],
  Version USING [Append],
  Volume USING [Close, ID, systemID],
  Window USING [Handle];
OthelloToolImpl: PROGRAM
  IMPORTS
    Exec, Runtime, String, OthelloToolDefs, Process, Time,
    Tool, ToolWindow, TTYSW, Version, Volume
  EXPORTS OthelloToolDefs = {

  tty: PUBLIC TTY.Handle;

  toolWindow: Window.Handle;

  CommandInterpreter: PROCESS;

  Init: PROCEDURE = {
    name: LONG STRING ← [75];
    name.length ← 0;
    String.AppendString[name, "OthelloTool "L];
    Version.Append[name];
    String.AppendString[name," of "L];
    Time.Append[name,Time.Unpack[Runtime.GetBcdTime[]]];
    toolWindow ← Tool.Create[
      name: name, makeSWsProc: MakeTTYSW, clientTransition: Stop,
      cmSection: "OthelloTool"L, tinyName1: "Othello"L, tinyName2: "Tool"L];
    Exec.AddCommand[name: "OthelloTool.~"L, proc: Activate, unload: DestroyTool];
  };

  Stop: ToolWindow.TransitionProcType = {
    IF new = inactive THEN {
      Process.Abort[CommandInterpreter];
      JOIN CommandInterpreter;};
  };

  Activate: Exec.ExecProc = {ToolWindow.Activate[toolWindow];};

  DestroyTool: Exec.ExecProc = {
    Exec.RemoveCommand[h, "OthelloTool.~"L];
    Tool.Destroy[toolWindow];
  };

  MakeTTYSW: Tool.MakeSWsProc = {
    logName: LONG STRING ← [20];
    ttySW: Window.Handle;
    logName.length ← 0;
    Tool.UnusedLogName[unused:logName, root: "OthelloTool.log"L];
    ttySW ← Tool.MakeTTYSW[window:window, name:logName];
    tty ← TTYSW.GetTTYHandle[ttySW];
    CommandInterpreter ← FORK OthelloToolDefs.Run[];
  };

  CloseVolume: PUBLIC PROCEDURE[volume: Volume.ID] = {
    IF volume # Volume.systemID THEN Volume.Close[volume];};

  Init[];

}.
```

```
-- File: OthelloDefs.mesa - last edit:
-- Riggle.PA           12-Jan-87 16:06:45
-- OthelloDefs.mesa (last edited by: RXJ      19-Apr-83 10:54:58)

-- Copyright (C) 1987 by Xerox Corporation. All rights reserved.

DIRECTORY
  Device         USING [Type],
  Environment    USING [bytesPerWord, wordsPerPage],
  PhysicalVolume USING [Handle, ID],
  System         USING [GreenwichMeanTime],
  Volume         USING [ID, Type];

OthelloDefs: DEFINITIONS =
  BEGIN

  CommandProcessor: TYPE = RECORD [
    proc: PROC [index: CARDINAL], next: LONG POINTER TO CommandProcessor ← NULL];
  MyNameIs:         SIGNAL [myNameIs: LONG STRING, myHelpIs: LONG STRING];
    -- resuming executes command
  AbortingCommand: ERROR [reason: LONG STRING, reasonOne: LONG STRING ← NIL];
  IndexTooLarge:   ERROR;
  Question:        SIGNAL;

  RegisterCommandProc: PROC [commandProc: LONG POINTER TO CommandProcessor];

  ConfirmType: TYPE = {once, twice, thrice};
  EchoNoEcho:  TYPE = {echo, stars};

  -- Utility Io
  Confirm:               PROC [how: ConfirmType ← once];
  DebugAsk:              PROC;
  GetName:               PROC [
    prompt: LONG STRING, dest: LONG POINTER TO LONG STRING,
    how: EchoNoEcho ← echo, signalQuestion: BOOLEAN ← FALSE];
  ReadNumber:            PROC [
    prompt: LONG STRING, min, max: LONG CARDINAL,
    default: LONG CARDINAL ← LAST[LONG CARDINAL]]
    RETURNS [ans: LONG CARDINAL];
  ReadShortNumber:       PROC [
    prompt: LONG STRING, min, max, default: LONG CARDINAL]
    RETURNS [CARDINAL];
  WriteFixedWidthNumber: PROC [
    x: LONG CARDINAL, count: CARDINAL, base: CARDINAL ← 10];
  WriteLongNumber:       PROC [num: LONG CARDINAL];
  WriteOctal:            PROC [CARDINAL];
  Yes:                   PROC [LONG STRING] RETURNS [BOOLEAN];

  -- Basic IO
  Cursor:            TYPE = {pointer, ftp};
  SetCursor:         PROC [Cursor];
  FlipCursor:        PROC;

  SetCommandString:  PROC [LONG STRING];  -- string will be freed to Storage.
  BlinkDisplay:      PROC;
  CheckUserAbort:    PROC;  -- clients should prepare UNWIND in case of abort.
  NewLine:           PROC;
  ReadChar:          PROC RETURNS [CHARACTER];
  WriteChar:         PROC [CHARACTER];
  WriteLine:         PROC [LONG STRING];
  WriteString:       PROC [LONG STRING];

  PackedTimeFromString: PROC [s: LONG STRING, justDate: BOOLEAN]
    RETURNS [System.GreenwichMeanTime];
    --  string format must be:
    --    IF justDate=FALSE THEN bDD-MMM-YYbbHH:MM:SSbbZZTb
    --    IF justDate=TRUE  THEN bDD-MMM-YYb
    --  return System.gmtEpoch for bogus time

  -- Exported by VolumeInitImplA
  GetLvIDFromUser: PROC [
    prompt: LONG STRING ← NIL, calledFromSetDebuggerPtrs: BOOLEAN ← FALSE]
    RETURNS [
      pvID: PhysicalVolume.ID, lvID: Volume.ID,
      drive: PhysicalVolume.Handle];
  GetLvTypeFromUser: PROC [prompt: LONG STRING, defaultType: Volume.Type]
    RETURNS [t: Volume.Type];
  GetDriveFromUser:  PROC RETURNS [h: PhysicalVolume.Handle];
  GetDriveNumber:    PROC [h: PhysicalVolume.Handle] RETURNS [CARDINAL];
  GetDriveType:      PROC [h: PhysicalVolume.Handle] RETURNS [Device.Type];

  -- Get bits interface for Initial ucode fetch
  FetchInitialMicrocode: PROC [
    InstallProc: PROC [getPage: PROC RETURNS [LONG POINTER]]];

  -- Clean up any outstanding ftp/stp/?? like connections
  CloseFetch: PROC;

  -- leader pages on boot files
  leaderPages: CARDINAL = 1;

  lpVersion: CARDINAL = 04193;

  lpNoteOffset: PRIVATE CARDINAL = 2;

  lpNoteLength: CARDINAL =
    (Environment.wordsPerPage-lpNoteOffset)*Environment.bytesPerWord;
```

```
LeaderPage: TYPE = MACHINE DEPENDENT RECORD [
  version(0): CARDINAL ← lpVersion,
  length(1): CARDINAL, -- count of characters in note
  note(lpNoteOffset): PACKED ARRAY [0..lpNoteLength) OF CHARACTER];

-- test for special commands enabled
Wizard: PROC RETURNS [BOOLEAN];

-- Crock to make >@[foo]baz work
AlternateGetCMFile: PROC [LONG STRING];

-- aids for othello varients implementing canned scripts (prometheus)
GetCannedScript: PROC;
SuppressOutput:  PROC RETURNS [BOOLEAN];
ThereIsAnError:  PROC;

END....
```

```
-- File: OthelloFetch.mesa - last edit:
-- Riggle.PA           12-Jan-87 16:10:06
-- OthelloFetch.mesa
--  RXJ          22-Feb-84 16:46:26

-- Copyright (C) 1987 by Xerox Corporation. All rights reserved.
DIRECTORY
  File USING [File],
  Stream USING [Handle];

OthelloFetch: DEFINITIONS =
  BEGIN

  Destination: TYPE = RECORD [
    SELECT type: * FROM
      pilotFileSystemWrite => [localFile: File.File],
      string => [stringProc: PROC [LONG STRING]],
      rawWrite => [linkProc: PROC [getPage: PROC RETURNS [LONG POINTER]]],
      ENDCASE];

  Object: TYPE = RECORD [
    next: Handle ← NIL,
    Retrieve: PROC [fileName: LONG STRING, destination: Destination],
    DoIndirect: PROC [cmFile: LONG STRING] RETURNS [mine: BOOLEAN],
    List: PROC [pattern: LONG STRING],
    Close: PROC];

  Handle: TYPE = LONG POINTER TO Object;

  Register: PROC [h: Handle];

  Select: PROC [h: Handle];  -- makes h current; closes previous

  SetLeaderPage: PUBLIC PROCEDURE [file: File.File, note: LONG STRING];

  StartFeedback: SIGNAL;

  GrabBitsFromStream: PROC [rs: Stream.Handle, rsSizePages: LONG CARDINAL,
    destination: Destination, note: LONG STRING ← NIL];

  userName, userPassword: LONG STRING;

  directory: LONG STRING;

  END.
```

```
-- Copyright (C) 1983  by Xerox Corporation. All rights reserved.
-- EtherBooter.mesa, RXJ      , 11-Jul-83 16:35:04
-- NFS          10-Jun-86 10:42:13  adapted for OthelloTool

DIRECTORY
  Boot USING [EthernetBootFileNumber, EthernetRequest],
  Heap USING [systemZone],
  HostNumbers USING [HostNumber],
  NSConstants USING [bootServerSocket],
  OthelloDefs USING [
    AbortingCommand, CommandProcessor, Confirm, GetName,
    IndexTooLarge, MyNameIs, RegisterCommandProc],
  SpecialBooting USING [BootFromEthernet],
  String USING [CopyToNewString],
  System USING [
    broadcastHostNumber, defaultSwitches,
    NetworkAddress, nullNetworkNumber, Switches],
  Unformat USING [Error, HostNumber];

EtherBooter: PROGRAM
  IMPORTS Heap, OthelloDefs, SpecialBooting, String, Unformat =
  BEGIN

  HostNumber: TYPE = HostNumbers.HostNumber;

  bootFileNumber: LONG STRING;

  EtherBoot: PROCEDURE =
    BEGIN
    request: Boot.EthernetRequest;
    switches: System.Switches ← System.defaultSwitches;
    OthelloDefs.GetName["Ether Boot from boot file number: "L, @bootFileNumber];
    GetAddress[@request.bfn, bootFileNumber !
      Unformat.Error => OthelloDefs.AbortingCommand["Can't parse that one (No CH)"L]];
    OthelloDefs.Confirm[];
    request.address ← [
      net: System.nullNetworkNumber,
      host: System.broadcastHostNumber,
      socket: NSConstants.bootServerSocket];
    SpecialBooting.BootFromEthernet[
      ethernetRequest: request, deviceOrdinal: 0, switches: switches];
    END;

  GetAddress: PROCEDURE [host: POINTER TO Boot.EthernetBootFileNumber, s: LONG STRING] =
    BEGIN
    host↑ ← [LOOPHOLE[Unformat.HostNumber[s, octal]]];
    END;

  Commands: PROCEDURE [index: CARDINAL] =
    BEGIN
    SELECT index FROM
      0 =>
        BEGIN
        OthelloDefs.MyNameIs[
          myNameIs: "Ether Boot"L,
          myHelpIs: "Load another program over the Ethernet"L];
        EtherBoot[];
        END;
      ENDCASE => OthelloDefs.IndexTooLarge;
    END;

  commandProcessor: OthelloDefs.CommandProcessor ← [Commands];
  bootFileNumber ← String.CopyToNewString["25200001000", Heap.systemZone];  -- Setup Default
  OthelloDefs.RegisterCommandProc[@commandProcessor];

  END.....
```

```
-- File: OthelloFetchImpl.mesa - last edit:
-- NFS           3-Jul-86 15:02:29
-- bjd          22-Jun-86 12:17:33
--  rkj 26-Feb-84 15:24:23

-- Copyright (C) 1986 by Xerox Corporation. All rights reserved.

DIRECTORY
  Environment,
  File USING [
    Create, Delete, File, MakePermanent, nullFile, PageNumber, SetSize, Unknown],
  FileTypes USING [tUntypedFile],
  Heap USING [systemZone],
  MFile,
  MStream,
  NSName,
  NSString,
  OthelloDefs USING [
    AbortingCommand, CommandProcessor, Confirm, FlipCursor,
    GetLvIDFromUser, GetName,
    IndexTooLarge, LeaderPage, leaderPages, lpNoteLength, lpVersion,
    MyNameIs, Question, RegisterCommandProc, WriteLine, WriteString, Yes],
  OthelloFetch USING [Destination, Handle, Object, StartFeedback],
  OthelloOps USING [
    BootFileType, GetVolumeBootFile, MakeBootable, MakeUnbootable,
    SetPhysicalVolumeBootFile, SetVolumeBootFile],
  OthelloToolDefs USING [CloseVolume],
  Process ,
  Profile ,
  Space,
  SpecialFile USING [MakeTemporary],
  Stream,
  String ,
  TemporaryBooting USING [InvalidParameters],
  Time,
  Volume USING [ID, InsufficientSpace, Open, systemID];

OthelloFetchImpl: MONITOR
  IMPORTS
    File, Heap, MFile, MStream, NSString, OthelloDefs, OthelloFetch,
    OthelloOps, OthelloToolDefs,
    Process, Profile, Space, SpecialFile, Stream, String,
    TemporaryBooting, Time, Volume
  EXPORTS OthelloDefs, OthelloFetch =
  BEGIN

  Object: TYPE = OthelloFetch.Object;

  Handle: TYPE = OthelloFetch.Handle;

  list: Handle ← NIL;

  current: Handle ← NIL;

  cmFile: LONG STRING ← NIL;
  fileName: LONG STRING ← NIL;

  z: UNCOUNTED ZONE = Heap.systemZone;

  S: PROC [s: LONG STRING] RETURNS [NSString.String] = INLINE {
    RETURN[NSString.StringFromMesaString[s]]};

  -- Fetcher registration

  Register: PUBLIC PROC [h: Handle] =
    BEGIN
    h.next ← list;
    list← h;
    END;

  Select: PUBLIC PROC [h: Handle] =
    BEGIN
    IF current # NIL THEN current.Close[];
    current← h;
    END;


  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- String/Credentials Commands
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  ClearinghouseCmd: PROC = {
    domain, organization: Profile.String ← NIL;
    CopyDomain: PROCEDURE[s: LONG STRING] = {
      domain ← String.CopyToNewString[s, z];};
    CopyOrganization: PROCEDURE[s: LONG STRING] = {
      organization ← String.CopyToNewString[s, z];};
    {ENABLE UNWIND => {
      IF domain # NIL THEN z.FREE[@domain];
      IF organization # NIL THEN z.FREE[@organization];};
    OthelloDefs.MyNameIs[
          myNameIs: "Clearinghouse"L,
          myHelpIs: "Set defaults for Clearinghouse"L];
    Profile.GetDefaultDomain[CopyDomain];
    Profile.GetDefaultOrganization[CopyOrganization];
    OthelloDefs.GetName["Domain: "L, @domain];
    OthelloDefs.GetName["Organization: "L, @organization];
```

```
    Profile.SetDefaultDomain[domain];
    Profile.SetDefaultOrganization[organization];
    };
    z.FREE[@domain]; z.FREE[@organization];
  };

  LoginCmd: PROC = {
    userName, userPassword: Profile.String ← NIL;
    CopyNameAndPassword: PROCEDURE[name, password: LONG STRING] = {
      userName ← String.CopyToNewString[name, z];
      userPassword ← String.CopyToNewString[password, z];};
    {ENABLE UNWIND => {
      IF userName # NIL THEN z.FREE[@userName];
      IF userPassword # NIL THEN z.FREE[@userPassword];};
    OthelloDefs.MyNameIs[
      myNameIs: "Login"L, myHelpIs: "Set user name & password"L];
    Profile.GetUser[CopyNameAndPassword, none];
    OthelloDefs.GetName["User: "L, @userName];
    OthelloDefs.GetName["Password: "L, @userPassword, stars];
    Profile.SetUser[userName, userPassword];
    };
    z.FREE[@userName]; z.FREE[@userPassword];
  };

  directory: PUBLIC LONG STRING ← NIL;

  Directory: PROC = {
    OthelloDefs.MyNameIs[
      myNameIs: "Directory"L,
      myHelpIs: "Set Default FTP directory"L];
    OthelloDefs.GetName["Directory: "L, @directory]};


  -- Simple Fetches

  FetchBoot: PROC = {
    Fetch[pilot, "Boot file name: "L, "Fetch Boot File"L, "Fetch Boot File"L, "boot"L, TRUE]};

  FetchGerm: PROC = {
    Fetch[germ,  "Germ file name: "L, "Germ Fetch"L, "Fetch Germ"L, "germ"L]};

  FetchPilotMicrocode: PROC = {
    Fetch[
      softMicrocode,
      "Pilot microcode file name: "L,
      "Pilot Microcode Fetch"L,
      "Fetch and Install Pilot Microcode"L,
      "db"L]};

  FetchDiagnosticMicrocode: PROC = {
    Fetch[
      hardMicrocode,
      "Diagnostic microcode file name: "L,
      "Diagnostic Microcode Fetch"L,
      "Fetch and Install Diagnostic Microcode"L,
      "db"L]};

  Fetch: PROC [
    type: OthelloOps.BootFileType, prompt, name, helpMsg, extension: STRING,
    bootFile: BOOLEAN ← FALSE] = {
    created: BOOLEAN;
    file: File.File;
    firstPage: File.PageNumber;
    lvID: Volume.ID;
    local: BOOLEAN ← current = NIL; -- if no connection open, assume local file.
    OthelloDefs.MyNameIs[myNameIs: name, myHelpIs: helpMsg];
    lvID ← OthelloDefs.GetLvIDFromUser[].lvID;
    OthelloDefs.GetName[prompt, @fileName];
    IF lvID = Volume.systemID AND bootFile THEN {
      -- fetching boot file for system volume.
      oldFile: File.File;
      IF local THEN file ← GetLocalFile[lvID, File.nullFile
        !NoLocalFile => GOTO NoFetch]
      ELSE {
        file ← File.Create[lvID, 1, FileTypes.tUntypedFile];
        current.Retrieve[fileName, [pilotFileSystemWrite[file]]
          !UNWIND => File.Delete[file];
          Volume.InsufficientSpace, Space.InsufficientSpace => {
            OthelloDefs.WriteLine["Insufficient space for new boot file."L];
            File.Delete[file];
            GOTO NoFetch;}];};
      OthelloOps.MakeUnbootable[file, type, firstPage !
        File.Unknown => CONTINUE;
        TemporaryBooting.InvalidParameters => CONTINUE];
      [oldFile, firstPage] ← OthelloOps.GetVolumeBootFile[lvID, type];
      OthelloOps.MakeUnbootable[oldFile, type, firstPage !
        File.Unknown => CONTINUE;
        TemporaryBooting.InvalidParameters => {
          OthelloDefs.WriteLine["Warning, trouble making unbootable"L];
          CONTINUE}];
      SpecialFile.MakeTemporary[oldFile];
      created ← TRUE;}
    ELSE { -- not system volume
      Volume.Open[lvID];
      [file, firstPage] ← OthelloOps.GetVolumeBootFile[lvID, type];
      IF local THEN {
        newBootFile: File.File;
```

```
        created ← TRUE;
        newBootFile ← GetLocalFile[lvID, file !NoLocalFile => GOTO NoFetch];
        file ← newBootFile;}
      ELSE {
        IF (created ← file = File.nullFile) THEN
          file ← File.Create[lvID, 1, FileTypes.tUntypedFile]
        ELSE OthelloOps.MakeUnbootable[file, type, firstPage !
          File.Unknown => CONTINUE;
          TemporaryBooting.InvalidParameters => {
            OthelloDefs.WriteLine["Warning, trouble making unbootable"L];
            CONTINUE}];
        current.Retrieve[fileName, [pilotFileSystemWrite[file]]
          ! UNWIND => {
            IF created THEN File.Delete[file];OthelloToolDefs.CloseVolume[lvID]}];
        };
    };
    OthelloDefs.WriteString["Installing..."L];
    OthelloOps.SetVolumeBootFile[file, type, OthelloDefs.leaderPages];
    IF created THEN File.MakePermanent[file];
    OthelloOps.MakeBootable[file, type, OthelloDefs.leaderPages
    ! TemporaryBooting.InvalidParameters => {
      OthelloDefs.WriteLine["Warning, trouble making bootable"L]; CONTINUE}];
    OthelloDefs.WriteLine["done"L];
    IF type IN [hardMicrocode..germ] AND
       OthelloDefs.Yes["Shall I also use this for the Physical Volume? "L
       ! UNWIND => OthelloToolDefs.CloseVolume[lvID]] THEN
      OthelloOps.SetPhysicalVolumeBootFile[file, type, OthelloDefs.leaderPages];
    OthelloToolDefs.CloseVolume[lvID];
    EXITS NoFetch => NULL;
  };

  NoLocalFile: ERROR = CODE;

  GetLocalFile: PROCEDURE [
    lvID: Volume.ID, oldFile: File.File] RETURNS [file: File.File] = {
    -- fileName already has name of local file.
    mStream: MStream.Handle;
    filePages: LONG CARDINAL;
    note: LONG STRING;
    mStream ← MStream.ReadOnly[fileName, [] !
      MStream.Error => {
        OthelloDefs.WriteLine["Unable to acquire local file"L];
        ERROR NoLocalFile;}];
    IF oldFile # File.nullFile THEN File.Delete[oldFile];
    filePages ← (MStream.GetLength[mStream] + Environment.bytesPerPage -1) /
        Environment.bytesPerPage;
    file ← File.Create[
      volume: lvID, initialSize: filePages + OthelloDefs.leaderPages,
      type: FileTypes.tUntypedFile !
        Volume.InsufficientSpace => OthelloDefs.AbortingCommand["Volume Full"L]];
    note ← MakeNote[MStream.GetFile[mStream]];
    GrabBitsFromStream[
      mStream, filePages, [pilotFileSystemWrite[file]], note!
      OthelloFetch.StartFeedback => {
        OthelloDefs.WriteString["Copying local file..."L];
        RESUME}];
    OthelloDefs.WriteLine["done"L];
    Stream.Delete[mStream];
    z.FREE[@note];
  };

  bufPages: CARDINAL = 8;

  StartFeedback: PUBLIC SIGNAL = CODE;

  MakeNote: PROCEDURE[file: MFile.Handle] RETURNS[note: LONG STRING] = {
    time: LONG STRING ← [20];
    note ← z.NEW[StringBody[MFile.maxNameLength]];
    note.length ← 0;
    MFile.GetFullName[file, note];
    String.AppendStringAndGrow[@note, " ("L, z];
    Time.Append[time, Time.Unpack[MFile.GetCreateDate[file]]];
    String.AppendStringAndGrow[@note, time, z];
    String.AppendCharAndGrow[@note, '), z];
  };

  GrabBitsFromStream: PUBLIC PROC [
    rs: Stream.Handle, rsSizePages: LONG CARDINAL,
    destination: OthelloFetch.Destination, note: LONG STRING ← NIL] = {
    WITH destination SELECT FROM
      pilotFileSystemWrite => {
        buffer: LONG POINTER ← NIL;
        base: File.PageNumber ← 0;
        got: CARDINAL;
        File.SetSize[localFile, rsSizePages + OthelloDefs.leaderPages
        ! Volume.InsufficientSpace => OthelloDefs.AbortingCommand["Volume Full"L]];
        SetLeaderPage[localFile, note];
        SIGNAL StartFeedback;
        WHILE base < rsSizePages DO
          thisPages: CARDINAL = CARDINAL[MIN[rsSizePages-base, bufPages]];
          size:      CARDINAL = thisPages*Environment.bytesPerPage;
          start:     CARDINAL ← 0;
          nProcesses ← 0;
          buffer ← Space.Map[
            window:[localFile, base+OthelloDefs.leaderPages, thisPages],
            life: dead].pointer;
          DO
```

```
            [bytesTransferred: got] ← rs.GetBlock[[
              blockPointer: buffer, startIndex: start, stopIndexPlusOne: size] !
                Stream.EndOfStream => {
                  got ← 0; start ← start + nextIndex; CONTINUE};
                UNWIND => [] ← Space.Unmap[buffer]];
            IF got = 0 THEN {[] ← Space.Unmap[buffer]; RETURN};
            IF (start ← start + got) = size THEN EXIT;
            ENDLOOP;
          ForkUnmap[buffer];
          OthelloDefs.FlipCursor[];
          base ← base + thisPages;
          ENDLOOP;
        buffer ← Space.ScratchMap[1]; -- check for any leftover stuff
        [bytesTransferred: got] ← rs.GetBlock[[
          blockPointer: buffer, startIndex: 0,
          stopIndexPlusOne: Environment.bytesPerPage] !
            Stream.EndOfStream => {got ← nextIndex; CONTINUE};
            UNWIND => [] ← Space.Unmap[buffer]];
        [] ← Space.Unmap[buffer];
        IF got # 0 THEN OthelloDefs.AbortingCommand[
          "File longer than advertised length"L]};
      string => {
        SIGNAL StartFeedback;
        DO
          stringOverhead: CARDINAL = SIZE[StringBody]*Environment.bytesPerWord;
          string: LONG STRING = Space.ScratchMap[bufPages];
          string↑ ← [
            length: 0,
            maxlength: bufPages*Environment.bytesPerPage - stringOverhead,
            text: ];
          WHILE string.length < string.maxlength DO
            got: CARDINAL;
            [bytesTransferred: got] ← rs.get[
              rs,
              [blockPointer: LOOPHOLE[@string.text],
               startIndex: string.length, stopIndexPlusOne: string.maxlength],
              rs.options
              ! Stream.EndOfStream => {
                  got ← 0; string.length ← string.length + nextIndex; CONTINUE};
                UNWIND => [] ← Space.Unmap[string]];
            IF got = 0 THEN {
              stringProc[string! UNWIND => [] ← Space.Unmap[string]];
              [] ← Space.Unmap[string]; RETURN};
            string.length ← string.length + got;
            ENDLOOP;
          [] ← Space.Unmap[string];
          OthelloDefs.AbortingCommand["Command file too long!"L];
          ENDLOOP};
      rawWrite =>{
        buffer: LONG POINTER = Space.ScratchMap[1];
        done:  BOOLEAN ← FALSE;
        first: BOOLEAN ← TRUE;
        options: Stream.InputOptions ← rs.options;
        GetPage: PROC RETURNS [LONG POINTER] = {
          got:   CARDINAL; index: CARDINAL ← 0;
          IF first THEN {SIGNAL StartFeedback; first ← FALSE};
          WHILE ~done DO
            [bytesTransferred: got] ← rs.get[
              sH: rs,
              block: [blockPointer: buffer, startIndex: index,
               stopIndexPlusOne: Environment.bytesPerPage],
              options: options
              ! Stream.EndOfStream => {got ← nextIndex; done ← TRUE; CONTINUE}];
            IF (index ← index + got) = Environment.bytesPerPage
              OR done THEN {OthelloDefs.FlipCursor[]; EXIT}
            ENDLOOP;
          RETURN[IF done AND index = 0 THEN NIL ELSE buffer]};
        options.signalEndOfStream ← TRUE;
        linkProc[GetPage ! UNWIND => [] ← Space.Unmap[buffer]];
        WHILE ~done DO [] ← GetPage[! UNWIND => [] ← Space.Unmap[buffer]] ENDLOOP;
        [] ← Space.Unmap[buffer]};
      ENDCASE => ERROR};

  -- Initial Ucode Fetch Command

  FetchInitialMicrocode: PUBLIC PROC [
    InstallProc: PROC [getPage: PROC RETURNS [LONG POINTER]]] = {
    CheckOpen[];
    OthelloDefs.GetName["File name: "L, @fileName];
    OthelloDefs.Confirm[];
    current.Retrieve[fileName, [rawWrite[InstallProc]]]};

  -- Command Files

  AlternateGetCMFile: PUBLIC PROC [s: LONG STRING] = {
    z.FREE[@cmFile];
    cmFile ← z.NEW[StringBody[s.length+8]];
    FOR i: CARDINAL IN [1..s.length) DO
      String.AppendChar[cmFile, s[i]] ENDLOOP;
    DoIndirect[]};

  Indirect: PROC = {
    OthelloDefs.MyNameIs[
      myNameIs: "@", myHelpIs: "Run command file"L];
    OthelloDefs.GetName["Command file: "L, @cmFile
    ! OthelloDefs.Question => {
      OthelloDefs.WriteLine["[Host]<Dir>Filename"L]; RESUME}];
```

```
    DoIndirect[]};

  DoIndirect: PROC = {
    [] ← String.AppendExtensionIfNeeded[@cmFile, "othello"L, z];
    FOR h: Handle ← list, h.next UNTIL h = NIL DO
      IF h.DoIndirect[cmFile] THEN RETURN;
      ENDLOOP;
    OthelloDefs.AbortingCommand["Unrecognizable command file name"L]};


  -- Misc. commands

   nProcesses, maxProcesses: NATURAL ← LAST[NATURAL];
   finished: CONDITION;

   ForkUnmap: ENTRY PROCEDURE [buffer: LONG POINTER] =
     BEGIN
     BEGIN ENABLE Process.TooManyProcesses => {maxProcesses ← nProcesses; RETRY};
     WHILE nProcesses >= maxProcesses DO
       WAIT finished; ENDLOOP;
     Process.Detach[LOOPHOLE[FORK DoUnmap[buffer]]];
     nProcesses ← nProcesses+1;
     END;
     END;

   DoUnmap: ENTRY PROCEDURE [buffer: LONG POINTER] =
   --buffer ← Space.Unmap[buffer, return];
     BEGIN
     buffer ← Space.Unmap[buffer];
     nProcesses ← MAX[nProcesses-1, 0];
     NOTIFY finished;
     END;

  CloseCmd: PROC = {
    OthelloDefs.MyNameIs[
      myNameIs: "Close"L, myHelpIs: "Close currently open connection"L];
    CloseFetch[]};

  CloseFetch: PUBLIC PROC = {
    Select[NIL]};

  ListCmd: PROC = {
    OthelloDefs.MyNameIs[
      myNameIs: "List Files"L, myHelpIs: "Enumerate files matching pattern"L];
    CheckOpen[];
    OthelloDefs.GetName["Pattern: "L, @fileName
    ! OthelloDefs.Question => {
      OthelloDefs.WriteLine["pattern to match"L]; RESUME}];
    current.List[fileName]};

  CheckOpen: PROC = {
    IF current = NIL THEN
      OthelloDefs.AbortingCommand["You must execute an Open command first"L]};

  SetLeaderPage: PUBLIC PROCEDURE [file: File.File, note: LONG STRING] =
    BEGIN
    lp: LONG POINTER TO OthelloDefs.LeaderPage ← Space.Map[[file, 0, OthelloDefs.leaderPages]].pointer;
    lp.version ← OthelloDefs.lpVersion;
    lp.length ← MIN[note.length, OthelloDefs.lpNoteLength];
    FOR i: CARDINAL IN [0..lp.length) DO
      lp.note[i] ← note[i];
      ENDLOOP;
    [] ← Space.Unmap[lp];
    END;


  -- command processor

  commandProcessor: OthelloDefs.CommandProcessor ← [FetchCommands];

  FetchCommands: PROC [index: CARDINAL] = {
    SELECT index FROM
      0 => Indirect[];
      1 => ClearinghouseCmd[];
      2 => CloseCmd[];
      3 => Directory[];
      4 => FetchBoot[];
      5 => FetchDiagnosticMicrocode[];
      6 => FetchGerm[];
      7 => FetchPilotMicrocode[];
      8 => ListCmd[];
      9 => LoginCmd[];
      ENDCASE => OthelloDefs.IndexTooLarge};

  -- init

  OthelloDefs.RegisterCommandProc[@commandProcessor];

  END.

Log
 5-Jun-86 12:25:59 NFS  Adapted for OthelloTool
```

```
-- Copyright (C) 1984  by Xerox Corporation. All rights reserved.
-- OthelloFloppy.mesa
--   LXR         10-Feb-84 16:25:51
--   RXJ         27-Feb-84 17:17:08

DIRECTORY
  AccessFloppy USING [
    Attributes, AttributesRecord, Close, Error, ErrorType, GetAttributes,
    leaderLength, LookUp, maxDataSize, Open, tFloppyLeaderPage, Time],
  Ascii USING [CR, SP],
  Environment USING [bytesPerPage, bytesPerWord],
  Heap USING [systemZone],
  File USING [File, nullFile, PageNumber, SetSize],
  Floppy USING [
    Error, ErrorType, FileHandle, GetFileAttributes, GetNextFile, nullFileID,
    nullVolumeHandle, PageNumber, Read, VolumeHandle],
  Format USING [Char, Date, Decimal, StringProc],
  NSFile USING [String],
  OthelloDefs USING [
    AbortingCommand, CheckUserAbort, CommandProcessor, FlipCursor,
    IndexTooLarge, leaderPages, MyNameIs, RegisterCommandProc,
    SetCommandString, SetCursor, WriteLine, WriteString],
  OthelloFetch USING [Destination, Object, Register, Select, SetLeaderPage],
  Process USING [Detach],
  Space USING [Map, ScratchMap, Unmap],
  String USING [
    AppendCharAndGrow, AppendLongDecimal, AppendStringAndGrow,
    CopyToNewString, Length, LowerCase],
  Time USING [Append, Unpack],
  Volume USING [ID, InsufficientSpace];

OthelloFloppy: PROGRAM
  IMPORTS
    AccessFloppy, File, Floppy, Format, Heap, OthelloDefs, OthelloFetch,
    Process, Space, String, Time, Volume =
  BEGIN

  DoIndirect: PROC [cmFile: LONG STRING] RETURNS [mine: BOOLEAN] =
    BEGIN
    s: LONG STRING ← NIL;
    GetString: PROC [c: LONG STRING] = {s ← String.CopyToNewString[c, Heap.systemZone]};
    IF cmFile[0] = '[ THEN RETURN [FALSE];
    OthelloFetch.Select[@fetcher]; OpenFloppy[];
    Retrieve[cmFile, [string[GetString]]
      ! UNWIND => Heap.systemZone.FREE[@s]];
    OthelloDefs.WriteLine["done"L];
    OthelloDefs.SetCommandString[s];
    RETURN[TRUE]
    END;

  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- MISC Stuff/Commands
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  floppy: Floppy.VolumeHandle ← Floppy.nullVolumeHandle;

  FloppyOpen: PROC RETURNS [BOOLEAN] = INLINE {RETURN[floppy # Floppy.nullVolumeHandle]};

  OpenCmd: PROC = {
    OthelloDefs.MyNameIs[myNameIs: "Floppy Open"L, myHelpIs: "Prepare to read files from floppy"L];
    OthelloFetch.Select[@fetcher];
    OpenFloppy[]};

  OpenFloppy: PROC = {
    floppy ← AccessFloppy.Open[
      ! AccessFloppy.Error => OthelloDefs.AbortingCommand["Can't open floppy"L];
        Floppy.Error => {FloppyError[error]; RETRY}]};

  CloseFloppy: PROC = {
    AccessFloppy.Close[
      ! AccessFloppy.Error, Floppy.Error => CONTINUE];
    floppy ← Floppy.nullVolumeHandle};

  FloppyList: PROC [pattern: LONG STRING] = {
    ListFiles[IF String.Length[pattern] = 0 THEN NIL ELSE pattern]};

  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- Central commands
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  commandProcessor: OthelloDefs.CommandProcessor ← [FloppyCommands];

  FloppyCommands: PROC [index: CARDINAL] = {
    SELECT index FROM
      0 => OpenCmd[];
      ENDCASE => OthelloDefs.IndexTooLarge};

  fetcher: OthelloFetch.Object ← [
    Retrieve: Retrieve,
    DoIndirect: DoIndirect,
    List: FloppyList,
    Close: CloseFloppy];


  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- file retrieval Stuff/Commands
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  EnumProc: TYPE = PROCEDURE [
```

```
    attributes: AccessFloppy.Attributes, fH: Floppy.FileHandle, name: LONG STRING]
    RETURNS [stop: BOOLEAN ← FALSE];

  ListFiles: PROCEDURE [pattern: LONG STRING] =
    BEGIN
    Write: Format.StringProc = {OthelloDefs.WriteString[s]};
    ListOne: EnumProc =
      BEGIN
      Write[name];
      FOR i: CARDINAL IN [name.length + WritePartial[Write, attributes]..40) DO
        Format.Char[Write, Ascii.SP]; ENDLOOP;
      Format.Date[Write, attributes.createDate, full];
      Format.Char[Write, Ascii.CR];
      END;
    EnumerateFloppyFiles[ListOne, pattern ];
    END;

  WritePartial: PROCEDURE [Write: Format.StringProc, attributes: AccessFloppy.Attributes]
    RETURNS [chars: CARDINAL ← 0] =
    BEGIN
    CountedNumber: PROCEDURE [n: LONG CARDINAL] RETURNS [CARDINAL] = {
      s: STRING = [12];
      String.AppendLongDecimal[s, n];
      Write[s];
      RETURN[s.length]};
    IF attributes.offset # 0 OR attributes.size # attributes.totalSize THEN {
      chars ← 4;
      Format.Char[Write, '[];
      chars ← chars + CountedNumber[attributes.offset];
      Write[".."L];
      chars ← chars + CountedNumber[attributes.offset+attributes.size-1];
      Format.Char[Write, ']]};
    END;

  EnumerateFloppyFiles: PROCEDURE [
    proc: EnumProc, pattern: LONG STRING ← NIL] =
    BEGIN
    nullFile: Floppy.FileHandle ← [volume: floppy, file: Floppy.nullFileID];
    attributes: AccessFloppy.Attributes ← Heap.systemZone.NEW[
      AccessFloppy.AttributesRecord[AccessFloppy.maxDataSize]];
    name: LONG STRING = LOOPHOLE[@attributes.length];
    BEGIN ENABLE Floppy.Error => {
      FloppyError[error]; nullFile.volume ← floppy; RETRY};
    FOR current: Floppy.FileHandle ←
      Floppy.GetNextFile[nullFile].nextFile,
      Floppy.GetNextFile[current].nextFile
      WHILE current#nullFile DO
        ENABLE UNWIND => Heap.systemZone.FREE[@attributes];
        OthelloDefs.CheckUserAbort[];
        IF Floppy.GetFileAttributes[current].type # AccessFloppy.tFloppyLeaderPage THEN LOOP;
        AccessFloppy.GetAttributes[current, attributes];
        IF (pattern = NIL OR MaskFilename[file: name, mask: pattern])
          AND proc[attributes, current, name] THEN EXIT;
        ENDLOOP;
    END; -- ENABLE
    Heap.systemZone.FREE[@attributes];
    END;

  MaskFilename: PROCEDURE [
    file: LONG STRING, fileIndex: CARDINAL ← 0, mask: LONG STRING,
    maskIndex: CARDINAL ← 0]
    RETURNS [BOOLEAN] =
    BEGIN
    -- local variables
    i, j: CARDINAL;
    wildString: CHARACTER = '*;
    wildChar: CHARACTER = '#;
    -- process each character in mask
    FOR i IN [maskIndex..mask.length) DO
      SELECT mask[i] FROM
        wildString =>  -- matches any string of zero or more characters
          BEGIN
          FOR j IN [fileIndex..file.length] DO
            IF MaskFilename[file, j, mask, i + 1] THEN
              RETURN[TRUE];
            ENDLOOP;
          RETURN[FALSE];
          END;
        wildChar =>  -- matches any single character
          IF fileIndex = file.length THEN RETURN[FALSE]
          ELSE fileIndex ← fileIndex + 1;
        ENDCASE =>
          IF fileIndex = file.length
            OR String.LowerCase[file[fileIndex]] # String.LowerCase[mask[i]] THEN
            RETURN[FALSE]
          ELSE fileIndex ← fileIndex + 1;
      ENDLOOP;
    -- filename passes mask if entire filename has been consumed
    RETURN[fileIndex = file.length];
    END;

  StartFeedback: SIGNAL = CODE;

  -- must fix Retrieve to deal with boot files that is in pieces

  Retrieve: PROC [fileName: LONG STRING, destination: OthelloFetch.Destination] = {
    segmentPages, totalPages, bytes, offset: LONG CARDINAL;
```

```
    name: LONG STRING ← NIL;
    BEGIN
    ENABLE Floppy.Error => {FloppyError[error]; RETRY};
    fH: Floppy.FileHandle ← [floppy, Floppy.nullFileID];
    [fh: fH, offset: offset, segmentPages: segmentPages, totalPages: totalPages, bytes: bytes, name: name] ← GetFile[fileName];
    GrabBits[
      fH: fH, offset: offset, segmentPages: segmentPages,
      totalPages: totalPages, sizeBytes: bytes, destination: destination,
      note: name !
        StartFeedback => {
          OthelloDefs.WriteString["Fetching..."L];
          OthelloDefs.SetCursor[ftp];
          RESUME};
        UNWIND => {OthelloDefs.SetCursor[pointer]; Heap.systemZone.FREE[@name]}];
    Heap.systemZone.FREE[@name];
    OthelloDefs.SetCursor[pointer];
    END};

  GetFile: PROC [fileName: LONG STRING] RETURNS [
    fh: Floppy.FileHandle, offset, segmentPages, totalPages, bytes: LONG CARDINAL, name: LONG STRING] =
    BEGIN
    time: LONG STRING ← [20];
    attributes: AccessFloppy.Attributes ← Heap.systemZone.NEW
      [AccessFloppy.AttributesRecord[AccessFloppy.maxDataSize]];
    name ← Heap.systemZone.NEW[StringBody[60]];
    {ENABLE UNWIND =>
      {Heap.systemZone.FREE[@attributes]; Heap.systemZone.FREE[@name]};
    fh ← AccessFloppy.LookUp[
      MakeNSString[fileName], attributes
        ! AccessFloppy.Error =>
          SELECT type FROM
            fileNotFound => OthelloDefs.AbortingCommand["No such file"L];
            volumeNotOpen => {
              CloseFloppy[];
              OpenFloppy[];
              RETRY};
            ENDCASE => OthelloDefs.AbortingCommand[ "Unexpected access floppy problem"L];
          Floppy.Error => {FloppyError[error]; RETRY}];
    String.AppendStringAndGrow[
      @name, LOOPHOLE[@attributes.length], Heap.systemZone];
    String.AppendStringAndGrow[@name, " ("L, Heap.systemZone];
    Time.Append[time, Time.Unpack[attributes.createDate]];
    String.AppendStringAndGrow[@name, time, Heap.systemZone];
    String.AppendCharAndGrow[@name, '), Heap.systemZone];
    offset ← attributes.offset;
    segmentPages ← attributes.size;
    totalPages ← attributes.totalSize;
    bytes ← attributes.totalSizeInBytes;
    Heap.systemZone.FREE[@attributes]};
    END;

  MakeNSString: PROCEDURE [s: LONG STRING] RETURNS [NSFile.String] = {
    IF s = NIL THEN RETURN[[bytes: NIL, length: 0, maxlength: 0]];
    RETURN[[bytes: LOOPHOLE[@s.text], length: s.length, maxlength: s.maxlength]]};

  bufPages: CARDINAL = 8;

  GrabBits: PROC [
    fH: Floppy.FileHandle, offset, segmentPages, totalPages: LONG CARDINAL,
    sizeBytes: LONG CARDINAL, destination: OthelloFetch.Destination,
    note: LONG STRING ← NIL] = {
    base: File.PageNumber ← 0;
    WITH destination SELECT FROM
      pilotFileSystemWrite => {
        buffer: LONG POINTER ← NIL;
        File.SetSize[localFile, totalPages + OthelloDefs.leaderPages
        ! Volume.InsufficientSpace => OthelloDefs.AbortingCommand["Volume Full"L]];
        OthelloFetch.SetLeaderPage[localFile, note];
        SIGNAL StartFeedback;
        WHILE base < segmentPages DO
          thisPages: CARDINAL = CARDINAL[MIN[segmentPages-base, bufPages]];
          buffer ← Space.Map[
            window:[localFile, offset+base+OthelloDefs.leaderPages, thisPages],
            life: dead].pointer;
          Floppy.Read[fH, base+AccessFloppy.leaderLength, thisPages, buffer !
            Floppy.Error => FloppyError[error];
            UNWIND => [] ← Space.Unmap[buffer]];
          --buffer ← Space.Unmap[buffer, return];
          Process.Detach[LOOPHOLE[FORK Space.Unmap[buffer]]]; buffer ← NIL;
          OthelloDefs.FlipCursor[];
          base ← base + thisPages;
          ENDLOOP};
      string => {
        thisPages: CARDINAL = CARDINAL[MIN[segmentPages-base, bufPages]];
        stringOverhead: CARDINAL = SIZE[StringBody]*Environment.bytesPerWord;
        string: LONG STRING ← NIL;
        IF segmentPages-base > thisPages THEN
          OthelloDefs.AbortingCommand["Command file too long!"L];
        SIGNAL StartFeedback;
        string ← Space.ScratchMap[thisPages+1];
        string↑ ← [
          length: CARDINAL[sizeBytes],
          maxlength: bufPages*Environment.bytesPerPage - stringOverhead,
          text: ];
        Floppy.Read[fH, base+AccessFloppy.leaderLength, thisPages, @string.text !
          Floppy.Error => FloppyError[error];
          UNWIND => [] ← Space.Unmap[string]];
```

```
        stringProc[string];
        [] ← Space.Unmap[string]};
      rawWrite =>{
        buffer: LONG POINTER = Space.ScratchMap[1];
        count:  CARDINAL ← 0;
        GetPage: PROC RETURNS [LONG POINTER] = {
          IF count = 0 THEN SIGNAL StartFeedback;
          IF count = segmentPages THEN RETURN[NIL];
          Floppy.Read[fH, base+count+AccessFloppy.leaderLength, 1, buffer !
                Floppy.Error => FloppyError[error];
                UNWIND => [] ← Space.Unmap[buffer]];
          count ← count + 1;
          OthelloDefs.FlipCursor[];
          RETURN[buffer]};
        linkProc[GetPage ! UNWIND => [] ← Space.Unmap[buffer]];
        [] ← Space.Unmap[buffer]};
      ENDCASE => ERROR};

  FloppyError: PROC [error: Floppy.ErrorType] =
    BEGIN
    myProc: Format.StringProc = {OthelloDefs.WriteString[s]};
    SELECT error FROM
      invalidVolumeHandle => {
        AccessFloppy.Close[! AccessFloppy.Error, Floppy.Error => CONTINUE];
        OpenFloppy[]; RETURN};
      notReady => OthelloDefs.AbortingCommand["Can't open floppy"L];
      badDisk, badSectors, hardwareError =>
        OthelloDefs.AbortingCommand["Floppy hardware problem"L];
      invalidFormat, invalidPageNumber, needsScavenging  =>
        OthelloDefs.AbortingCommand["Floppy not readable"L];
      ENDCASE => {
        OthelloDefs.WriteString["Floppy error "L];
        Format.Decimal[myProc, error.ORD]};
    OthelloDefs.AbortingCommand[NIL]
    END;


--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- initialization
--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
OthelloDefs.RegisterCommandProc[@commandProcessor];
OthelloFetch.Register[@fetcher];

END.....
```

```
-- File: OthelloNSFTP.mesa - last edit:
-- NFS           5-Jun-86 14:20:34
-- bjd          23-Aug-85 16:18:16
--   lgr         13-Feb-84 15:31:26
--   rkj         24-Feb-84 18:36:31

-- Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.

DIRECTORY
  AddressTranslation USING [Error, PrintError, StringToNetworkAddress],
  Auth USING [FreeIdentity, IdentityHandle, MakeIdentity],
  --CH USING [MakeRhs],
  Courier USING [ErrorCode, Error],
  NSErrorMsg USING [PostCourierError, PostNSFileError],
  Heap USING [systemZone],
  Format USING [StringProc],
  NSDataStream USING [Abort, Aborted, Handle, SourceStream],
  NSFile USING [
    AttributesProc, AttributesRecord, Close, Error, ErrorRecord, Find, GetAttributes,
    Handle, List, Logoff, LogonDirect, maxStringLength, nullHandle, nullSession, Open,
    Retrieve, Scope, Selections, ServiceRecord, Session, String, Time],
  NSName USING [
    AppendNameToString, Error, FreeNameFields, maxDomainLength, maxFullNameLength,
    maxOrgLength, Name, NameFieldsFromString, NameRecord, String],
  OthelloDefs USING [
    AbortingCommand, CheckUserAbort, CommandProcessor, GetName,
    IndexTooLarge, MyNameIs, RegisterCommandProc, SetCommandString,
    SetCursor, WriteChar, WriteLine, WriteString],
  OthelloFetch USING [
    Destination, directory, Handle, GrabBitsFromStream, Object, Register, Select,
    StartFeedback],
  Profile USING [GetDefaultDomain, GetDefaultOrganization, GetUser, String],
  Stream USING [Delete, Handle],
  String USING [
    AppendChar, AppendCharAndGrow, AppendNumber, AppendString, AppendStringAndGrow,
    CopyToNewString, Empty, Length, StringBoundsFault, SubString, SubStringDescriptor],
  Time USING [Append, Unpack];

OthelloNSFTP: PROGRAM
  IMPORTS
    AddressTranslation, Auth, --CH,-- Courier, NSErrorMsg, Heap,
    NSDataStream, NSFile,
    NSName, OthelloDefs, OthelloFetch, Profile, Stream, String, Time =
  BEGIN

  host: LONG STRING ← NIL;

  nsFileSession: NSFile.Session ← NSFile.nullSession;

  z: UNCOUNTED ZONE = Heap.systemZone;

  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- String/Credentials Commands
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
--  I don't believe we need this proc anymore; It no longer make sense to allow network addresses; must get to Auth server anyway;

  Qualify: PROC [token: LONG STRING] RETURNS [newToken: LONG STRING] = {
    octalAddress: BOOLEAN ← TRUE; -- only '0..'7 and '# allowed
    chChar: CHARACTER = ':;
    defaultDomain, defaultOrganization: LONG STRING ← NIL;
    GetDomain: PROC[domain: LONG STRING] =
      {IF domain # NIL THEN
        String.AppendStringAndGrow[@defaultDomain, domain, z]};
    GetOrg: PROC[org: LONG STRING] =
      {IF org # NIL THEN
        String.AppendStringAndGrow[@defaultOrganization, org, z]};
    IF String.Length[token] = 0 THEN RETURN[NIL];
    FOR i: CARDINAL IN [0..token.length) DO
       SELECT token[i] FROM
         chChar => {RETURN[String.CopyToNewString[token, z]]}; -- already qualified
         IN['0..'7], '# => NULL;
         ENDCASE => octalAddress ← FALSE;
       ENDLOOP;
    newToken ← String.CopyToNewString[token, z];
    IF octalAddress THEN RETURN;
    Profile.GetDefaultDomain[GetDomain];
    Profile.GetDefaultOrganization[GetOrg];
    IF String.Length[defaultDomain] > 0 OR
      String.Length[defaultOrganization] > 0 THEN {
          String.AppendCharAndGrow[@newToken, chChar, z];
          String.AppendStringAndGrow[@newToken, defaultDomain, z];
          String.AppendCharAndGrow[@newToken, chChar, z];
          String.AppendStringAndGrow[@newToken, defaultOrganization, z]};
    z.FREE[@defaultDomain];
    z.FREE[@defaultOrganization]};

  DoIndirect: PROC [cmFile: LONG STRING] RETURNS [mine: BOOLEAN] =
    BEGIN
    fileName: LONG STRING ← NIL;
    ParseCmFileName: PROC = {
      hostEnd: CARDINAL;
      IF cmFile.length = 0 THEN RETURN;
      FOR i: CARDINAL IN [0..cmFile.length) DO
        c: CHARACTER = cmFile[i];
        SELECT c FROM
          '[ => LOOP; '] => {hostEnd ← i; EXIT};
          ENDCASE => String.AppendCharAndGrow[@host, c, z];
```

```
        REPEAT FINISHED => {z.FREE[@host]; RETURN}
        ENDLOOP;
      -- hostEnd points at ']
      FOR i: CARDINAL IN (hostEnd..cmFile.length) DO
        IF cmFile[i] = '< AND OthelloFetch.directory#NIL THEN
          OthelloFetch.directory.length ← 0;
        String.AppendCharAndGrow[@fileName, cmFile[i], z];
        IF cmFile[i] = '> THEN {
          String.AppendStringAndGrow[
            @OthelloFetch.directory, fileName, z];
          fileName.length ← 0};
        ENDLOOP};
    s:          LONG STRING ← NIL;
    GetString: PROC [c: LONG STRING] = {s ← String.CopyToNewString[c, z]};

    IF cmFile[0] # '[ THEN RETURN [FALSE];
    z.FREE[@host];
    z.FREE[@OthelloFetch.directory];
    ParseCmFileName[];
    OthelloFetch.Select[@fetcher]; Open[];
    Retrieve[fileName, [string[GetString]]
      ! UNWIND => {z.FREE[@s]; z.FREE[@fileName]}];
    OthelloDefs.WriteLine["done"L];
    OthelloDefs.SetCommandString[s];
    z.FREE[@fileName];
    RETURN[TRUE]
    END;

  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- MISC Stuff/Commands
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  userOpened: BOOLEAN ← FALSE;
  OpenCmd: PROC = {
    OthelloDefs.MyNameIs[
      myNameIs: "Open Connection"L,
      myHelpIs: "Open connection to file service"L];
    OthelloFetch.Select[@fetcher];
    OthelloDefs.GetName["Open connection to "L, @host];
    Open[]; userOpened ← TRUE};

  ReOpen: PROC RETURNS [BOOLEAN] = {
    IF userOpened=FALSE THEN RETURN[FALSE];
    Open[]; RETURN[TRUE]};

  RemoteList: PROC [fileName: LONG STRING] = {
    IF ~ConnectionOpen[] AND ~ReOpen[] THEN
      OthelloDefs.AbortingCommand["Please open a connection"L];
    ListFiles[IF String.Length[fileName] = 0 THEN "*"L ELSE fileName]};

  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- Central commands
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  commandProcessor: OthelloDefs.CommandProcessor ← [FtpCommands];

  FtpCommands: PROC [index: CARDINAL] = {
    SELECT index FROM
      0 => OpenCmd[];
     ENDCASE => OthelloDefs.IndexTooLarge};

  fetcher: OthelloFetch.Object ← [
    Retrieve: Retrieve,
    DoIndirect: DoIndirect,
    List: RemoteList,
    Close: Close];


  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- file retrieval Stuff/Commands
  --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  ConnectionOpen: PROC RETURNS [BOOLEAN] = {
    RETURN[nsFileSession # NSFile.nullSession]};

  -- all callers close the connection first
  Open: PROC = {
    clientDefaultsRecord: NSName.NameRecord;
    defaultCHOrg: LONG STRING = [NSName.maxOrgLength];
    defaultCHDomain: LONG STRING = [NSName.maxDomainLength];
    serviceName: LONG STRING ← [NSName.maxFullNameLength];
    serviceRec: NSFile.ServiceRecord ← [];
    nameRecord: NSName.NameRecord ← [];
    id: Auth.IdentityHandle ← NIL;

    GetDomain: PROC[domain: LONG STRING] = {
      String.AppendString[defaultCHDomain, domain !
        String.StringBoundsFault => RESUME[NIL]]};
    GetOrg: PROC[org: LONG STRING] = {
      String.AppendString[defaultCHOrg, org !
        String.StringBoundsFault => RESUME[NIL]]};
    AppendNameToString: PROCEDURE [s: LONG STRING, name: NSName.Name] = {
      newS: NSName.String ← NSName.AppendNameToString[s: S[s], name: name];
      s.length ← newS.length};
    CopyUserAndPassword: PROCEDURE[name, password: LONG STRING] = {
      userName ← String.CopyToNewString[name, z];
      userPassword ← String.CopyToNewString[password, z];
     };
    Cleanup: PROCEDURE = {
      Auth.FreeIdentity[@id, z];
```

```
      z.FREE[@userName]; z.FREE[@userPassword];
      NSName.FreeNameFields[z, @serviceRec.name];
      NSName.FreeNameFields[z, @nameRecord]};

    userName, userPassword: Profile.String ← NIL;

    IF nsFileSession # NSFile.nullSession THEN RETURN;
    Profile.GetDefaultDomain[GetDomain];
    Profile.GetDefaultOrganization[GetOrg];
    clientDefaultsRecord ← [domain: S[defaultCHDomain], org: S[defaultCHOrg]];
    NSName.NameFieldsFromString[
      z: z, s: S[host], destination: @serviceRec.name,
      clientDefaults: @clientDefaultsRecord !
        NSName.Error => OthelloDefs.AbortingCommand["Illegal host name"L]];
    AppendNameToString[serviceName, @serviceRec.name];
    Profile.GetUser[CopyUserAndPassword, clearinghouse];
    NSName.NameFieldsFromString[
      z: z, s: S[userName], destination: @nameRecord,
      clientDefaults: @clientDefaultsRecord !
        UNWIND => NSName.FreeNameFields[z, @serviceRec.name];
        NSName.Error => OthelloDefs.AbortingCommand["Illegal login name"L]];
    id ← MakeIdentity[name: @nameRecord, password: userPassword];
    {ENABLE UNWIND => Cleanup[];
    serviceRec.systemElement ← AddressTranslation.StringToNetworkAddress[
      s: serviceName, id: id !
      AddressTranslation.Error => {
        msg: LONG STRING ← [100];
        appendErrorMsg: Format.StringProc = {
          String.AppendString[msg, s ! String.StringBoundsFault => RESUME[NIL]]};
        AddressTranslation.PrintError[error: errorRecord, proc: appendErrorMsg];
        OthelloDefs.AbortingCommand[msg]}].addr;
    nsFileSession ← NSFile.LogonDirect[
      identity: id, service: @serviceRec
        ! NSFile.Error => NSError[error];
          Courier.Error => CourierError[errorCode]];
     Cleanup[]}};

  MakeIdentity: PROC [name: NSName.Name, password: LONG STRING]
    RETURNS [ident: Auth.IdentityHandle] = {
    ident ← Auth.MakeIdentity[
      myName: name, password: S[password],
      z: z, style: simple, dontCheck: TRUE]};


  Close: PROC = {
    IF ~ConnectionOpen[] THEN RETURN;
    NSFile.Logoff[nsFileSession
      ! NSFile.Error => NSError[error];
        Courier.Error => CourierError[errorCode];
        OthelloDefs.AbortingCommand => {
          OthelloDefs.WriteString[reason];
          OthelloDefs.WriteLine[reasonOne];
          CONTINUE}];
    nsFileSession ← NSFile.nullSession;
    OthelloDefs.WriteLine["connection closed"L]};

  -- could mess with directories.
  -- who cares
  ListFiles: PROC [pattern: LONG STRING]= {
    scope: NSFile.Scope ← [];
    selections: NSFile.Selections ← [];
    fh: NSFile.Handle;
    dir: LONG STRING ← NIL;
    wildCardInFileName: BOOLEAN ← FALSE;
    name: LONG STRING ← NIL;
    ss: String.SubStringDescriptor ← [base: NIL, offset: 0, length: 0];
    ListOne: NSFile.AttributesProc = {
      version: LONG STRING ← [20];
      time: LONG STRING ← [20];
      Time.Append[time, Time.Unpack[attributes.createdOn]];
      String.AppendChar[version, '!];
      String.AppendNumber[version, attributes.version];
    <<  FOR i: CARDINAL IN [0..attributes.pathname.length) DO
        OthelloDefs.WriteChar[VAL[attributes.pathname.bytes[i]]];
        ENDLOOP;>>
      OthelloDefs.WriteString[dir];
      FOR i: CARDINAL IN [0..attributes.name.length) DO
        OthelloDefs.WriteChar[VAL[attributes.name.bytes[i]]];
        ENDLOOP;
      OthelloDefs.WriteString[version];
      THROUGH [dir.length + attributes.name.length + version.length..80-time.length)
       DO OthelloDefs.WriteChar[' ] ENDLOOP;
      OthelloDefs.WriteLine[time];
      --OthelloDefs.WriteChar[' ];
      --OthelloDefs.WriteLine[info.author];
      --OthelloDefs.WriteChar[' ];
      --OthelloDefs.WriteLongNumber[info.size];
      --OthelloDefs.WriteLine[" bytes]"L];
      OthelloDefs.CheckUserAbort[];
      RETURN};
    dir ← z.NEW[StringBody[60]];
    IF pattern[0] # '< AND String.Length[OthelloFetch.directory] # 0 THEN {
      String.AppendStringAndGrow[@dir, OthelloFetch.directory, z];
      IF dir[dir.length - 1] # '> THEN
        String.AppendCharAndGrow[@dir, '>, z]};
    String.AppendStringAndGrow[@dir, pattern, z];
    ss.base ← dir;
```

```
    FOR i: CARDINAL DECREASING IN [0..dir.length) DO
      SELECT dir[i] FROM
        '* => wildCardInFileName ← TRUE;
        '> => {ss.length ← i + 1; EXIT};
        ENDCASE;
      ENDLOOP;
    name ← z.NEW[StringBody[dir.length - ss.length]];
    FOR i: CARDINAL IN [ss.length .. dir.length) DO
      String.AppendChar[name, dir[i]];
      ENDLOOP;
    dir.length ← ss.length;
    IF HasWildCard[dir] THEN {
      z.FREE[@dir];
      z.FREE[@name];
      OthelloDefs.AbortingCommand["No wild cards in directories."L]};
    fh ← GetFileFromSS[ss];
    IF fh = NSFile.nullHandle THEN {
      z.FREE[@name]; z.FREE[@dir]; RETURN};
    scope.filter ← IF wildCardInFileName THEN
      [matches[[name [S[name]]]]]
    ELSE [equal[[name [S[name]]]]];
    selections.interpreted[name] ← TRUE;
    selections.interpreted[version] ← TRUE;
    selections.interpreted[createdOn] ← TRUE;
    selections.interpreted[pathname] ← TRUE; -- not yet implemented
    NSFile.List[
      directory: fh, proc: ListOne, selections: selections, scope: scope,
      session: nsFileSession
        ! NSFile.Error => NSError[error];
          Courier.Error => CourierError[errorCode];
          UNWIND => {z.FREE[@name]; z.FREE[@dir]}];
    z.FREE[@dir];
    z.FREE[@name];
    };

  Retrieve: PROC [
    fileName: LONG STRING, destination: OthelloFetch.Destination] = {
    size: LONG CARDINAL;
    name: LONG STRING ← NIL;
    fH: NSFile.Handle ← NSFile.nullHandle;
    Sink: PROC [source: NSDataStream.SourceStream] =
      BEGIN ENABLE {
        NSDataStream.Aborted => {Stream.Delete[source]; CONTINUE};
        UNWIND => Stream.Delete[source]};
      OthelloFetch.GrabBitsFromStream[source, size, destination, name !
          OthelloFetch.StartFeedback => {
            OthelloDefs.WriteString["Fetching..."L];
            OthelloDefs.SetCursor[ftp];
            RESUME};
          UNWIND => {
            OthelloDefs.SetCursor[pointer];
            NSDataStream.Abort[source ! NSDataStream.Aborted => CONTINUE]}];
      Stream.Delete[source ! NSDataStream.Aborted => CONTINUE]
      END;
    [fH, size, name] ← GetFile[fileName];
    NSFile.Retrieve[fH, [proc [Sink]], nsFileSession
      ! UNWIND =>
          {NSFile.Close[
              fH, nsFileSession ! NSFile.Error, Courier.Error => CONTINUE];
           z.FREE[@name]}];
    NSFile.Close[fH, nsFileSession
      ! NSFile.Error, Courier.Error => CONTINUE;
      UNWIND => z.FREE[@name]];
    z.FREE[@name];
    OthelloDefs.SetCursor[pointer]};

  GetFile: PROC [fileName: LONG STRING] RETURNS [fh: NSFile.Handle, size: LONG CARDINAL, name: LONG STRING] =
    BEGIN
    time: LONG STRING ← [20];
    attributes: NSFile.AttributesRecord;
    ss: String.SubStringDescriptor ← [base: NIL, offset: 0, length: 0];
    name ← z.NEW[StringBody[60]];
    String.AppendChar[name, '[];
    String.AppendStringAndGrow[@name, host, z];
    String.AppendCharAndGrow[@name, '], z];
    ss.offset ← name.length;
    IF fileName[0] # '< AND ~String.Empty[OthelloFetch.directory] THEN {
      String.AppendStringAndGrow[@name, OthelloFetch.directory, z];
      IF name[name.length - 1] # '> THEN
        String.AppendCharAndGrow[@name, '>, z]};
    String.AppendStringAndGrow[@name, fileName, z];
    ss.base ← name;
    ss.length ← name.length - ss.offset;
    fh ← GetFileFromSS[ss];
    NSFile.GetAttributes[
      fh, [[createdOn: TRUE, sizeInPages: TRUE]],
      @attributes, nsFileSession
        ! NSFile.Error => NSError[error];
          Courier.Error => CourierError[errorCode]];
    String.AppendStringAndGrow[@name, " ("L, z];
    Time.Append[time, Time.Unpack[attributes.createdOn]];
    String.AppendStringAndGrow[@name, time, z];
    String.AppendCharAndGrow[@name, '), z];
    size ← attributes.sizeInPages;
    END;

  GetFileFromSS: PROCEDURE [ss: String.SubStringDescriptor] RETURNS [fh: NSFile.Handle ← NSFile.nullHandle] = {
```

```
    ENABLE {NSFile.Error => NSError[error];
            Courier.Error => CourierError[errorCode]};
     tempName: STRING = [NSFile.maxStringLength];
     parent: NSFile.Handle ← NSFile.Open[attributes: NIL, session: nsFileSession];
     DO
       GetRoot[@ss, tempName];
       fh ← NSFile.Find[
         directory: parent,
         scope: [
           direction: backward, filter: [equal[[name[S[tempName]]]]]],
         controls: [timeout: 4], session: nsFileSession !
           UNWIND => NSFile.Close[parent, nsFileSession !
             NSFile.Error, Courier.Error => CONTINUE]];
       NSFile.Close[parent, nsFileSession];
       IF ss.length = 0 THEN EXIT;
       parent ← fh
       ENDLOOP;
     };

   GetRoot: PROC [fileName: String.SubString, root: LONG STRING] = {
     OPEN fileName;
     StripChar: PROC = INLINE {offset ← offset + 1;  length ← length - 1};
     N: CARDINAL ← offset + length;
     quote: CHARACTER = '';
     i: CARDINAL ← offset;
     root.length ← 0;
     WHILE i < N DO
       SELECT base[i] FROM
         quote =>  {
           i ← i+1; StripChar[]; IF i = N THEN EXIT;
           String.AppendChar[root, base[i]]};
         '>, '/ =>  EXIT;
         '< =>  root.length ← 0;
         ENDCASE => String.AppendChar[root, base[i]];
       i ← i+1;
       StripChar[];
       ENDLOOP;
     IF fileName.length = 0 THEN RETURN;
     StripChar[]};

   S: PROCEDURE [s: LONG STRING] RETURNS [NSFile.String] = {
     IF s = NIL THEN RETURN[[bytes: NIL, length: 0, maxlength: 0]];
     RETURN[[bytes: LOOPHOLE[@s.text], length: s.length, maxlength: s.maxlength]]};

   HasWildCard: PROC [s: LONG STRING] RETURNS [BOOLEAN] = {
     IF s#NIL THEN FOR i: CARDINAL IN [0..s.length) DO
       IF s[i] = '* THEN RETURN[TRUE] ENDLOOP;
     RETURN[FALSE]};


   NSError: PROC [error: NSFile.ErrorRecord] =
     BEGIN
     post: Format.StringProc = {OthelloDefs.WriteString[s]};
     NSErrorMsg.PostNSFileError[error, post];
     OthelloDefs.AbortingCommand[NIL];
     END;

   CourierError: PROC [error: Courier.ErrorCode] =
     BEGIN
     post: Format.StringProc = {OthelloDefs.WriteString[s]};
     NSErrorMsg.PostCourierError[error, post];
     OthelloDefs.AbortingCommand[NIL];
     END;

   <<StartCH: PROCEDURE = {
     frame: PROGRAM ← Runtime.GlobalFrame[LOOPHOLE[CH.MakeRhs]];
     START frame};>>

   --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   -- initialization
   --~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   OthelloDefs.RegisterCommandProc[@commandProcessor];
   OthelloFetch.Register[@fetcher];
   --StartCH[];

   END.....

Log
NFS      4-Jun-86 13:04:55      Adapted for OthelloTool.
```

```
-- File: OthelloNS.mesa - last edit:
-- BJD    .PA          15-Feb-85 16:10:58
-- AOF          25-Jan-85 10:33:26

-- Copyright (C) 1983 , 1985 by Xerox Corporation. All rights reserved.

DIRECTORY
  AddressTranslation USING [Error, PrintError, StringToNetworkAddress],
  Auth USING [IdentityHandle],
  NSBuffer USING [Body, Buffer, ReturnBuffer],
  ExtendedString USING [AppendNumber],
  Format USING [StringProc],
  Inline USING [HighByte, LowByte],
  NSConstants USING [echoerSocket],
  NSTypes USING [maxDataBytesPerEcho, wordsPerIDPHeader],
  OthelloDefs,
  Process USING [Detach, Pause, SecondsToTicks, Yield],
  Profile USING [GetID],
  Router USING [
    FillRoutingTable, GetDelayToNet, infinity, endEnumeration, startEnumeration,
    EnumerateRoutingTable],
  Socket USING [
    AssignNetworkAddress, Create, Delete, GetPacket, ChannelHandle,
    PutPacket, GetSendBuffer, SetPacketBytes, GetPacketBytes,
    SetWaitTime, TimeOut],
  String USING [AppendString, StringBoundsFault],
  System USING [NetworkAddress, SocketNumber, NetworkNumber, HostNumber];

OthelloNS: PROGRAM
  IMPORTS AddressTranslation, NSBuffer, ExtendedString, Inline, Profile, Process, Router,
    String, Socket, OthelloDefs =
  BEGIN OPEN OthelloDefs;

  EchoUser: PROC =
    BEGIN
    bytesPerBuffer: CARDINAL;
    funny, late:    LONG CARDINAL ← 0;
    recv, sent:     LONG CARDINAL ← 0;
    wrong:          LONG CARDINAL ← 0;
    me, where:      System.NetworkAddress;
    mySoc:          Socket.ChannelHandle;
    packetNumber:   CARDINAL ← 0;
    pleaseStop:     BOOLEAN ← FALSE;
    routing:        CARDINAL;
    Watch:          PROC = {[] ← ReadChar[]; pleaseStop ← TRUE};
    PrintErrorNS:  PROC [b: NSBuffer.Buffer] = {
      body: NSBuffer.Body ← b.ns;
      source: System.NetworkAddress ← body.source;
      NewLine[];
      IF body.packetType = error THEN {
        len: CARDINAL = body.pktLength;
        WriteString["[Error packet, code="L];
        WriteOctal[LOOPHOLE[body.errorType]];
        WriteString[", from: "L];
        PrintNSAddress[@source];
        WriteString["] "L];
        FOR i: CARDINAL IN [0..len - NSTypes.wordsPerIDPHeader) DO
          WriteChar[Inline.LowByte[body.errorBody[i]]];
          WriteChar[Inline.HighByte[body.errorBody[i]]];
          ENDLOOP}
      ELSE {
        WriteString[" ***** "L];
        WriteString["Funny packet type = "L];
        WriteOctal[LOOPHOLE[body.packetType]];
        WriteString[" ***** "L]};
      NewLine[]};

    identity: Auth.IdentityHandle ← NIL;
    getID: PROC [id: Auth.IdentityHandle] = {identity ← id};

    GetName["Echo to: "L, @echoName];
    Profile.GetID[simple, getID];
    [where, ] ← AddressTranslation.StringToNetworkAddress[echoName, identity !
      AddressTranslation.Error => {
        msg: LONG STRING ← [100];
        appendErrorMsg: Format.StringProc = {
          String.AppendString[msg, s ! String.StringBoundsFault => RESUME[NIL]]};
        AddressTranslation.PrintError[error: errorRecord, proc: appendErrorMsg];
        OthelloDefs.AbortingCommand[msg]}];
    where.socket ← NSConstants.echoerSocket;
    routing ← Router.GetDelayToNet[where.net];
    IF routing = Router.infinity THEN
      AbortingCommand["Can't reach that network"L];

    me ← Socket.AssignNetworkAddress[];
    mySoc ← Socket.Create[me.socket];
    Socket.SetWaitTime[mySoc, 2000];  --two second timeout
    WriteString[".   ["L]; PrintNSAddress[@me]; WriteString["] => ["L];

    PrintNSAddress[@where];
    WriteChar[']]; NewLine[];
    Process.Detach[FORK Watch[]];
    bytesPerBuffer ← NSTypes.maxDataBytesPerEcho;

    UNTIL pleaseStop DO

      FOR len: CARDINAL IN [4..bytesPerBuffer] UNTIL pleaseStop DO
```

```
        b: NSBuffer.Buffer ← Socket.GetSendBuffer[mySoc];
        body: NSBuffer.Body ← b.ns;
        body.destination ← where;
        body.packetType ← echo; body.echoType ← echoRequest;
        Socket.SetPacketBytes[b, len];
        FOR i: CARDINAL IN [4..len - 4) DO body.echoBytes[i] ← i; ENDLOOP;
        body.echoWords[0] ← body.echoWords[1] ← (packetNumber ← packetNumber + 1);
        Socket.PutPacket[mySoc, b]; sent ← sent + 1;

        Process.Yield[];  -- be sure we don't hog machine

        BEGIN
        b ← Socket.GetPacket[mySoc ! Socket.TimeOut => GOTO late];
        SELECT TRUE FROM
          (body.packetType # echo) =>
            {funny ← funny + 1; PrintErrorNS[b]};
          (body.echoWords[0] # packetNumber) => {WriteChar['#]; late ← late + 1};
          (body.echoWords[1] # packetNumber) => {WriteChar['#]; late ← late + 1};
          (len # Socket.GetPacketBytes[b]) => {WriteChar['#]; late ← late + 1};
          ENDCASE =>
            FOR i: CARDINAL IN [4..len - 4) DO
              IF body.echoBytes[i] # (i MOD 400B) THEN
                {wrong ← wrong + 1; WriteChar['~]; EXIT};
              REPEAT FINISHED => {WriteChar['!]; recv ← recv + 1};
              ENDLOOP;
        NSBuffer.ReturnBuffer[b];
        EXITS late => {WriteChar['?]; late ← late + 1};
        END;

        ENDLOOP;

      NewLine[];
      ENDLOOP;

    Socket.Delete[mySoc];
    WriteString["Out: "L];
    WriteLongNumber[sent];
    WriteString[", In: "L];
    WriteLongNumber[recv];
    WriteString[" ("L];
    WriteLongNumber[(recv*100)/sent];
    WriteLine["%)"L];
    IF late # 0 THEN {
      WriteString["Late: "L]; WriteLongNumber[late];
      WriteString[" ("L]; WriteLongNumber[(late*100)/sent]; WriteLine["%)"L]};
    IF funny # 0 THEN {WriteLongNumber[funny]; WriteLine[" funny"L]};
    IF wrong # 0 THEN {WriteLongNumber[wrong]; WriteLine[" wrong data"L]};
    END;

  PrintLocalRoutingTable: PROC =
    BEGIN
    string: STRING ← [20];
    net: System.NetworkNumber;
    Router.FillRoutingTable[Router.infinity];  --load 'em up
    Process.Pause[Process.SecondsToTicks[2]];
    FOR hop: CARDINAL IN[0..Router.infinity] DO
      net ← Router.EnumerateRoutingTable[Router.startEnumeration, hop];
      IF net = Router.endEnumeration THEN LOOP;  --don't print empties
      WriteString["Networks "L];
      WriteLongNumber[LONG[hop]];
      WriteString[" hops away = {"L];
      UNTIL net = Router.endEnumeration DO
        ExtendedString.AppendNumber[@net, SIZE[System.NetworkNumber], 8, string];
        WriteString[string]; string.length ← 0; WriteChar['B];
        net ← Router.EnumerateRoutingTable[net, hop];
        IF net # Router.endEnumeration THEN WriteString[", "L];
        ENDLOOP;
      WriteChar['}];
      NewLine[];
      ENDLOOP;
    Router.FillRoutingTable[0];  --shut down the table
    END;  --PrintLocalRoutingTable

  PrintNSAddress: PROC [a: POINTER TO System.NetworkAddress] =
    BEGIN
    buffer: STRING ← [50];
    ExtendedString.AppendNumber[@a.net, SIZE[System.NetworkNumber], 8, buffer];
    buffer[buffer.length] ← '.; buffer.length ← buffer.length + 1;
    ExtendedString.AppendNumber[@a.host, SIZE[System.HostNumber], 8, buffer];
    buffer[buffer.length] ← '.; buffer.length ← buffer.length + 1;
    ExtendedString.AppendNumber[@a.socket, SIZE[System.SocketNumber], 8, buffer];
    WriteString[buffer];
    END;

  echoName: LONG STRING ← NIL;
  Commands: PROC [index: CARDINAL] = {
    SELECT index FROM
      0 => {
        MyNameIs[
          myNameIs: "Echo User"L,
          myHelpIs: "Echo user"L];
        EchoUser[]};
      1 => {
        MyNameIs[
          myNameIs: "Routing Tables"L,
          myHelpIs: "Show NS network routing tables"L];
        PrintLocalRoutingTable[]};
```

```
        ENDCASE => IndexTooLarge};

    commandProcessor: CommandProcessor ← [Commands];
    -- initialization
    RegisterCommandProc[@commandProcessor];

    END.....
```

```
-- File: VolumeInitCommandImpl.mesa - last edit:
-- NFS           4-Jun-86 11:23:33
-- BJD    .PA          15-Feb-85 16:22:59
--   RXJ       12-Sep-83 22:47:20
--   RSF       14-Dec-83 15:20:28
--   DWE       13-Jan-84 11:27:06
--   LXR       31-Jan-84 16:44:47

-- Copyright (C) 1981, 1982, 1983, 1984 , 1985 by Xerox Corporation. All rights reserved.

-- This file is the command Processor.

DIRECTORY
  File  USING [Error, ErrorType, Unknown],
  Format USING [HostNumber, StringProc],
  Frame USING [Free, ReadLocalWord],
  Heap USING [systemZone],
  Inline USING [BITNOT, HighHalf, LowHalf],
  OthelloDefs,
  OthelloOps USING [
    GetTimeFromTimeServer, IsTimeValid,
    SetProcessorTime, TimeServerError],
  OthelloToolDefs USING [tty],
  PhysicalVolume USING [Error, ErrorType, NeedsScavenging],
  PilotClient USING [],
  PrincOps USING [frameSizeMap, LocalFrameHandle, LocalOverhead],
  Process USING [Pause, SecondsToTicks],
  Runtime USING [GetBcdTime, IsBound],
  SpecialRuntime USING [GetCurrentSignal],
  SpecialSpace USING [realMemorySize],
  SpecialSystem USING [GetProcessorID],
  Scavenger USING [Error, ErrorType],
  String USING [
    AppendChar, AppendCharAndGrow, AppendDecimal, AppendLongNumber,
    EquivalentSubString, InvalidNumber, StringBoundsFault, StringToNumber,
    SubStringDescriptor, UpperCase],
  System USING [
    GetGreenwichMeanTime, GreenwichMeanTime, gmtEpoch,
    LocalTimeParameters, GetLocalTimeParameters, SetLocalTimeParameters],
  TTY USING [
    BlinkDisplay, CharsAvailable, GetChar, Handle,
    PutChar, PutString, RemoveCharacter, ResetUserAbort, UserAbort],
  Time USING [
    Append, defaultTime, Invalid, Pack, Unpack, Unpacked, useGMT, useSystem],
  UserTerminal USING [
    CursorArray, GetCursorPattern, SetCursorPattern],
  Version USING [Append],
  VersionExtras USING [AppendCopyright],
  Volume USING [
    InsufficientSpace, NeedsScavenging, NotOpen, ReadOnly, Unknown],
  VolumeConversion USING [Error, ErrorType];

UtilityPilotClientImpl: PROGRAM
  IMPORTS
    File, Format, Frame, Heap, Inline, OthelloDefs, OthelloOps, OthelloToolDefs,
    PhysicalVolume, Process, Runtime, SpecialRuntime, SpecialSpace, SpecialSystem,
    Scavenger, String, System, Time, TTY, UserTerminal,
    Version, VersionExtras, Volume, VolumeConversion
  EXPORTS OthelloDefs, OthelloToolDefs  =
  BEGIN

  MyNameIs:         PUBLIC SIGNAL [
    myNameIs: LONG STRING, myHelpIs: LONG STRING] = CODE;
  AbortingCommand: PUBLIC ERROR [
    reason: LONG STRING, reasonOne: LONG STRING ← NIL] = CODE;
  IndexTooLarge:    PUBLIC ERROR  = CODE;
  Question:         PUBLIC SIGNAL = CODE;
  TryAgain:         PUBLIC SIGNAL = CODE;

  BS:       CHARACTER = 10C;
  ControlA: CHARACTER = 'A - 100B;
  ControlP: CHARACTER = 'P - 100B;
  ControlW: CHARACTER = 'W - 100B;
  CR:       CHARACTER = 15C;
  DEL:      CHARACTER = 177C;
  ESC:      CHARACTER = 33C;
  SP:       CHARACTER = ' ;
  NUL:      CHARACTER = 0C;

  CommandProcessor: TYPE = OthelloDefs.CommandProcessor;

  -- ~~~~~~~~~~~~~~~~~~~~~~~~~
  -- Commands
  -- ~~~~~~~~~~~~~~~~~~~~~~~~~
  CurrentComand: SIGNAL RETURNS [
    proc: PROC [index: CARDINAL], index: CARDINAL] = CODE;

  ForAllCommandProcs: PROC [P: PROC[LONG STRING]] = {
    FOR c: LONG POINTER TO CommandProcessor ← commands, c.next WHILE c # NIL DO
      FOR i: CARDINAL IN [0..LAST[CARDINAL]) DO
        ENABLE CurrentComand => RESUME[c.proc, i];
        c.proc[i
        ! MyNameIs => {P[myNameIs]; CONTINUE};
          IndexTooLarge => EXIT];
        ENDLOOP ENDLOOP};

  Help: PROC = {
```

```
    WidthProc: PROC [s: LONG STRING] = {tabWidth ← MAX[tabWidth, s.length]};
    tabWidth: CARDINAL ← 0;
    SIGNAL MyNameIs[myNameIs: "Help"L, myHelpIs: "Type this table"L];
    ForAllCommandProcs[WidthProc];
    tabWidth ← tabWidth + 4;
    FOR c: LONG POINTER TO CommandProcessor ← commands, c.next WHILE c # NIL DO
      FOR i: CARDINAL IN [0..LAST[CARDINAL]) DO
        c.proc[i
        ! MyNameIs => {
            WriteString[myNameIs];
            THROUGH [myNameIs.length..tabWidth) DO WriteChar[' ] ENDLOOP;
            WriteLine[myHelpIs];
            CONTINUE};
          IndexTooLarge => EXIT];
        ENDLOOP ENDLOOP;
    WriteLine[
      "In General, Del will abort current command, ? will explain options"L]};

  TimeUser: PROC [index: CARDINAL] = {
    SELECT index FROM
      0 => {
        SIGNAL MyNameIs[myNameIs: "Time"L, myHelpIs: "Time of day"L];
        WriteString["Current time"L]; WriteTime[Time.defaultTime, TRUE]};
      1 =>
        Help[];
      ENDCASE =>
        ERROR IndexTooLarge};

  RegisterCommandProc: PUBLIC PROC [
    commandProc: LONG POINTER TO CommandProcessor] = {
    commandProc.next ← commands; commands ← commandProc};

  commands: LONG POINTER TO CommandProcessor ← @helpCommandProcessor;
  helpCommandProcessor: CommandProcessor ← [TimeUser, NIL];

  -- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- Basic command processing
  -- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  CollectCommand: PROC RETURNS [
    p: PROC [index: CARDINAL], index: CARDINAL] = {
    ExplainOptions: PROC = {
      first: BOOLEAN ← TRUE;
      WriteChar['?];
      IF userString.length # 0 THEN {
        P: PROC [s: LONG STRING] = {
          IF HeadMatch[s, userString.length] THEN {
            WriteString[IF first THEN "\rCurrent Options Are: "L ELSE ", "L];
            WriteString[s]; first ← FALSE}};
        ForAllCommandProcs[P]};
      IF first THEN {  -- Didn't match... tell all
        P: PROC [s: LONG STRING] = {
          IF ~first THEN WriteString[", "L]; WriteString[s]; first ← FALSE};
        WriteString["\rValid Commands Are: "L];
        ForAllCommandProcs[P]};
      WriteString["\r> "L]; WriteString[userString]};
    FindAnswer: TYPE = RECORD [
      SELECT how: * FROM  none => NULL, many => NULL,
      one => [proc: PROC [index: CARDINAL], index: CARDINAL],
      ENDCASE];
    FindPossibles: PROC RETURNS [ans: FindAnswer ←  [none[]]] = {
      P: PROC [matchString: LONG STRING] = {
        IF HeadMatch[matchString, head] THEN
          WITH ans SELECT FROM
            none => {
              ans ← [one[CurrentComand[].proc, CurrentComand[].index]];
              UNTIL userString.length = matchString.length DO
                userString[userString.length] ← matchString[userString.length];
                IF (userString.length ← userString.length + 1) = userString.maxlength THEN {
                  WriteLine[" Command too long!"L]; ERROR TryAgain}
                ENDLOOP};
            ENDCASE => {
              --ASSERT[head#0]
              FOR i : CARDINAL IN [head - 1..LAST[CARDINAL]) DO
                IF LowerCase[userString[i]] # LowerCase[matchString[i]] THEN {
                  userString.length ← i; EXIT};
                ENDLOOP;
              ans ← [many[]]}};
      head: CARDINAL ← userString.length;
      IF head = 0 THEN RETURN;
      ForAllCommandProcs[P];
      WHILE head # userString.length DO
        WriteChar[userString[head]]; head ← head + 1 ENDLOOP};
    HeadMatch: PROC [matchString: LONG STRING, head: CARDINAL]
      RETURNS [BOOLEAN] = {
      IF head > matchString.length THEN RETURN[FALSE];
      FOR i: CARDINAL IN [0..head) DO
        IF LowerCase[userString[i]] # LowerCase[matchString[i]] THEN
          RETURN[FALSE]
        ENDLOOP;
      RETURN[TRUE]};
    LowerCase: PROC [c: CHARACTER] RETURNS [CHARACTER] = {
      RETURN[IF c IN ['A..'Z] THEN c + ('a - 'A) ELSE c]};

    userString: STRING = [100];

    userString.length ← 0;
    WriteString["> "L];
```

```
    DO
      c: CHARACTER = ReadChar[];
      SELECT c FROM
        DEL => {WriteLine[" XXX"L]; ERROR TryAgain};
        BS, ControlA => IF userString.length # 0 THEN
          EraseTTYChar[userString[userString.length ← userString.length - 1]];
        ControlW =>
          IF userString.length # 0 THEN DO
            EraseTTYChar[userString[userString.length ← userString.length - 1]];
            IF userString.length=0 OR userString[userString.length - 1] = SP THEN EXIT
            ENDLOOP;
        '? => ExplainOptions[];
        CR, SP => {
          ans: FindAnswer = FindPossibles[];
          WITH theAns: ans SELECT FROM
            none => {
              IF Runtime.IsBound[LOOPHOLE[OthelloDefs.AlternateGetCMFile]]
                 AND userString.length > 1 AND userString[0] = '@ THEN {
                 NewLine[];
                 OthelloDefs.AlternateGetCMFile[userString
                 ! OthelloDefs.MyNameIs => RESUME];
                 ERROR TryAgain};
              IF prometheusBound THEN AbortingCommand["Script Error"L]
              ELSE BlinkDisplay[]};
            many => NULL;
            one  => RETURN[theAns.proc, theAns.index];
            ENDCASE => ERROR};
        ENDCASE =>
          IF (userString.length ← userString.length + 1) = userString.maxlength THEN {
            WriteLine[" Command too long!"L]; ERROR TryAgain}
          ELSE WriteChar[userString[userString.length - 1] ← c];
      ENDLOOP};

  -- ~~~~~~~~~~~~~~~~~~~~~~~~
  -- Utility-Type Functions
  -- ~~~~~~~~~~~~~~~~~~~~~~~~
  Confirm: PUBLIC PROC [how: OthelloDefs.ConfirmType ← once] = {
    IF CommandFileActive[] THEN RETURN;
    WriteString["Are you "L];
    IF how = thrice THEN WriteString["still "L];
    WriteString["sure? [y or n]: "L];
    DO
      c: CHARACTER = ReadChar[];
      SELECT c FROM
        'y, 'Y, CR  => {WriteLine["Yes"L]; EXIT};
        'n, 'N, DEL => {WriteLine["No"L]; ERROR TryAgain};
        ENDCASE =>     BlinkDisplay[];
      ENDLOOP;
    IF how = twice THEN {
      Process.Pause[Process.SecondsToTicks[3]]; FlushInput[]; Confirm[thrice]}};

  DebugAsk: PUBLIC PROC = {
    WriteString["\rType ControlP to muddle on......."L];
    WHILE ReadChar[] # ControlP DO ENDLOOP; NewLine[]};

  spacesInStringOK: BOOLEAN ← FALSE;
  GetName: PUBLIC PROC [
    prompt: LONG STRING ← NIL, dest: LONG POINTER TO LONG STRING,
    how: OthelloDefs.EchoNoEcho ← echo, signalQuestion: BOOLEAN ← FALSE] =
    BEGIN
    first: BOOLEAN ← TRUE;
    EraseChar: PROC = {
      IF dest.length = 0 THEN RETURN;
      dest.length ← dest.length - 1;
      EraseTTYChar[IF how = echo THEN dest[dest.length] ELSE '*];
      IF dest.length = 0 AND dest.maxlength > 20 THEN {
        Heap.systemZone.FREE[dest]; dest↑ ← Heap.systemZone.NEW[StringBody[10]]}};
    CWriteC: PROC [c: CHARACTER] = {WriteChar[IF how = echo THEN c ELSE '*]};
    CWriteString: PROC = {
      FOR i: CARDINAL IN [0..dest.length) DO CWriteC[dest[i]] ENDLOOP};

    IF dest↑ = NIL THEN dest↑ ← Heap.systemZone.NEW[StringBody[10]];
    WriteString[prompt]; CWriteString[];
    DO
      c: CHARACTER = ReadChar[];
      SELECT TRUE FROM
        c = BS, c = ControlA => EraseChar[];
       << (c = SP AND ~spacesInStringOK),>> c = CR => {NewLine[]; RETURN};
        c = DEL => {WriteLine[" XXX"L]; ERROR TryAgain};
        c = ControlW =>
          DO
            EraseChar[];
            IF dest.length=0 THEN EXIT;
            SELECT dest[dest.length-1] FROM
              IN ['a..'z], IN ['A..'Z], IN ['0..'9] => LOOP;
              ENDCASE => EXIT;
            ENDLOOP;
        c = '? AND signalQuestion => {
          SIGNAL Question; WriteString[prompt]; CWriteString[]; LOOP};
        c >= SP => {
          IF first THEN WHILE dest.length#0 DO EraseChar[] ENDLOOP;
          String.AppendCharAndGrow[dest, c, Heap.systemZone]; CWriteC[dest[dest.length-1]]};
        ENDCASE => BlinkDisplay[];
      first ← FALSE;
      ENDLOOP;
    END;
```

```
    numberString: LONG STRING ← NIL;
    ReadNumber: PUBLIC PROC [
      prompt: LONG STRING, min, max, default: LONG CARDINAL ← LAST[LONG CARDINAL]]
      RETURNS [ans: LONG CARDINAL] = {
      DO
        IF default # LAST[LONG CARDINAL] THEN {
          IF numberString=NIL THEN numberString ← Heap.systemZone.NEW[StringBody[15]];
          numberString.length ← 0; String.AppendLongNumber[numberString, default, 10]};
        WriteString[prompt]; WriteChar['[]; WriteLongNumber[min];
        WriteString[".."L]; WriteLongNumber[max]; WriteString["]: "L];
        GetName[dest: @numberString];
        ans ← 0;
        FOR i: CARDINAL IN [0..numberString.length) DO
          IF numberString[i] NOT IN ['0..'9] THEN EXIT;
          ans ← 10*ans + numberString[i] - '0;
          REPEAT FINISHED => IF ans IN [min..max] THEN {
            Heap.systemZone.FREE[@numberString]; RETURN};
          ENDLOOP;
        WriteLine["Bad Number !"L];
        ENDLOOP};

    ReadShortNumber: PUBLIC PROC [
      prompt: LONG STRING, min, max, default: LONG CARDINAL]
      RETURNS [CARDINAL] = {
      RETURN[Inline.LowHalf[
        ReadNumber[prompt, min, MIN[max, LONG[LAST[CARDINAL]]], default]]]};

    WriteFixedWidthNumber: PUBLIC PROC [
      x: LONG CARDINAL, count: CARDINAL, base: CARDINAL ← 10] = {
      WFD: PROC [x: LONG CARDINAL, c: CARDINAL] = {
        IF c = count THEN RETURN;
        WFD[x/base, c + 1];
        WriteChar[IF c = 0 OR x # 0 THEN Inline.LowHalf[x MOD base] + '0 ELSE ' ]};
      WFD[x, 0]};

    WriteLongNumber: PUBLIC PROC [num: LONG CARDINAL] = {
      s: STRING ← [40];
      s.length ← 0;
      String.AppendLongNumber[s, num, 10];
      WriteString[s]};

    WriteOctal: PUBLIC PROC [num: CARDINAL] = {
      IF num # 0 THEN WriteOctal[num/8]; WriteChar[(num MOD 8) + '0]};

    Yes: PUBLIC PROC [s: LONG STRING] RETURNS [BOOLEAN] = {
      WriteString[s];
      DO
        SELECT ReadChar[] FROM
          'Y, 'y, CR  => {WriteLine["yes"L]; RETURN[TRUE]};
          'N, 'n, DEL => {WriteLine["no"L];  RETURN[FALSE]};
          ENDCASE     => WriteChar['?];
        ENDLOOP};

    -- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    -- Time munging
    -- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    --  string format must be: bDD-MMM-YYbbHH:MM:SSbbZZTb
    PackedTimeFromString: PUBLIC PROC [
      s: LONG STRING, justDate: BOOLEAN]
      RETURNS [t: System.GreenwichMeanTime] = {
      Empty: PROC [s: LONG STRING] RETURNS [BOOLEAN] = {
        RETURN[s = NIL OR s.length = 0]};
      EquivalentChar: PUBLIC PROC [c1, c2: CHARACTER] RETURNS [BOOLEAN] = {
        RETURN[String.UpperCase[c1] = String.UpperCase[c2]]};
      GetToken: PROC [storage: LONG STRING, s: LONG STRING, c: CARDINAL]
        RETURNS [is: CARDINAL] = {
        FOR is ← c, is + 1 UNTIL is >= s.length DO
          ch: CHARACTER = s[is];
          SELECT ch FROM
            IN ['a..'z], IN ['A..'Z], IN ['0..'9] =>
              String.AppendChar[storage, ch];
            ':, '- => EXIT; -- terminator
            '      => IF ~Empty[storage] THEN EXIT; --terminating blank
            ENDCASE;
          ENDLOOP;
        RETURN[is + 1]};

      DoIt: PROC [s: LONG STRING] RETURNS [t: System.GreenwichMeanTime] = {
        Get: PROC RETURNS [CARDINAL] = {
          s1.length ← 0; nextChar ← GetToken[s1, s, nextChar];
          RETURN[s1.length]};
        GetNumber: PROC RETURNS [CARDINAL] = {
          [] ← Get[]; RETURN[String.StringToNumber[s1, 10]]};
        m: String.SubStringDescriptor ← [
          base:   "JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC"L,
          offset: NULL, length: 3];
        s1: STRING = [3];
        month: String.SubStringDescriptor ← [
          base: s1, offset: 0, length: NULL];
        time:     Time.Unpacked ← [
          0, 0, 0, 0, 0, 0, 0, FALSE, System.GetLocalTimeParameters[]];
        nextChar: CARDINAL ← 0;
        packIt:   BOOLEAN ← TRUE;
        IF Empty[s] THEN RETURN[System.gmtEpoch];
        time.day ← GetNumber[];
        month.length ← Get[];
```

```
      FOR i: CARDINAL IN [0..12) DO
        m.offset ← i*3;
        IF String.EquivalentSubString[@month, @m] THEN {
          time.month ← i; EXIT};
        ENDLOOP;
      time.year ← GetNumber[];
      time.year ← time.year + (IF time.year>68 THEN 1900 ELSE 2000);
      IF justDate THEN {
        time.hour ← 23; time.minute ← 59; time.second ← 59}
      ELSE {
        time.hour   ← GetNumber[];
        time.minute ← GetNumber[];
        time.second ← GetNumber[];
        IF Get[] # 0 THEN {
          zones: PACKED ARRAY [5..8] OF CHARACTER = ['E, 'C, 'M, 'P];
          FOR i: CARDINAL IN [5..8] DO
            IF EquivalentChar[s1[0], zones[i]] THEN {time.zone.zone ← i; EXIT};
            REPEAT FINISHED => time.zone.zone ← 0; -- GMT
            ENDLOOP;
          time.dst ← EquivalentChar[s1[1], 'D];
          packIt ← FALSE}};
      t ← Time.Pack[time, packIt]};
    t ← DoIt[s
    ! String.InvalidNumber, String.StringBoundsFault, Time.Invalid => {
      t ← System.gmtEpoch; CONTINUE}]};

  WriteTime: PROC [
    t: System.GreenwichMeanTime, showDay: BOOLEAN ← TRUE,
    type: {system, gmt, pacific} ← system] = {
    days: ARRAY [0..7) OF STRING = [
      "Monday"L, "Tuesday"L, "Wednesday"L, "Thursday"L,
      "Friday"L, "Saturday"L, "Sunday"L];
    temps: STRING = [40];
    Time.Append[temps,
      Time.Unpack[t, SELECT type FROM
        pacific => [useThese[[west, 8, 0, 121, 305]]],
        gmt => Time.useGMT,
        ENDCASE => Time.useSystem]];
    IF showDay THEN {
      WriteChar[' ]; WriteString[days[Time.Unpack[t].unpacked.weekday]]};
    IF temps[0] # ' THEN WriteChar[' ];
    WriteLine[temps]};

  -- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  -- The Big Loop
  -- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  prometheusBound: BOOLEAN =
    Runtime.IsBound[LOOPHOLE[OthelloDefs.GetCannedScript]];

  Run: PUBLIC PROC =
    BEGIN
    ENABLE ABORTED => CONTINUE; -- when deactivated
    ttyHandle ← OthelloToolDefs.tty;
    ResetAbort[];
    PrintHerald[];
    PrintPIDs[];
    PrintMemorySize[];
    GetTime[];
    DO
      TellError: PROC [s: LONG STRING] = {
        IF prometheusBound THEN OthelloDefs.ThereIsAnError[];
        commandIndex ← LAST[CARDINAL]; NewLine[]; WriteString[s]};
      p: PROC [index: CARDINAL]; i: CARDINAL;
      IF (~CommandFileActive[]) AND prometheusBound THEN {
        ResetAbort[]; OthelloDefs.GetCannedScript[]};
      IF CommandFileActive[] THEN
        CheckUserAbort[
        !MyAborted  => {TellError["Command File Aborted\r"L]; LOOP}]
      ELSE ResetAbort[];
      [p, i]  ← CollectCommand[
      ! TryAgain => RETRY;
        AbortingCommand => {TellError[reason]; WriteLine[reasonOne]; LOOP};
        MyAborted => {TellError["Command File Aborted\r"L]; LOOP}];
      NewLine[];
      p[i !
        MyNameIs => RESUME;
        MyAborted => {TellError["ABORTED\r"L]; CONTINUE};
        AbortingCommand => {
          TellError[reason]; WriteLine[reasonOne]; CONTINUE};
        File.Unknown => {
          TellError["File.Unknown"L]; DebugAsk[]; CONTINUE};
        File.Error => {
          PrintNames: PROC [x: File.ErrorType] = {
            e: ARRAY File.ErrorType OF STRING = [
              invalidParameters: "invalidParameters"L,
              reservedType: "reservedType"L];
            WriteString[e[x]]};
          TellError["File.Error["L];
          PrintNames[type];
          WriteChar[']];
          DebugAsk[];
          CONTINUE};
        PhysicalVolume.Error => {
          PrintNames: PROC [x: PhysicalVolume.ErrorType] = {
            e: ARRAY PhysicalVolume.ErrorType OF STRING = [
              badDisk: "badDisk"L,
              badSpotTableFull: "badSpotTableFull"L,
```

```
            containsOpenVolumes: "containsOpenVolumes"L,
            diskReadError: "diskReadError"L,
            hardwareError: "hardwareError"L,
            hasPilotVolume: "hasPilotVolume"L,
            alreadyAsserted: "alreadyAsserted"L,
            insufficientSpace: "insufficientSpace"L,
            invalidHandle: "invalidHandle"L,
            nameRequired: "nameRequired"L,
            needsConversion: "needsConversion"L,
            notReady: "notReady"L,
            noSuchDrive: "noSuchDrive"L,
            noSuchLogicalVolume: "noSuchLogicalVolume"L,
            physicalVolumeUnknown: "physicalVolumeUnknown"L,
            writeProtected: "writeProtected"L,
            wrongFormat: "wrongFormat"L];
          WriteString[e[x]]};
        TellError["PhysicalVolume.Error["L];  PrintNames[error];
        WriteChar[']];
        DebugAsk[];
        CONTINUE};
      PhysicalVolume.NeedsScavenging => {
        TellError["PhysicalVolume.NeedsScavenging"L];
        DebugAsk[]; CONTINUE};
      Scavenger.Error =>{
        PrintNames: PROC [x: Scavenger.ErrorType] = {
          e: ARRAY Scavenger.ErrorType OF STRING = [
            cannotWriteLog: "cannotWriteLog"L,
            noSuchPage: "noSuchPage"L,
            orphanNotFound: "orphanNotFound"L,
            volumeOpen: "volumeOpen"L,
            diskHardwareError: "diskHardwareError"L,
            diskNotReady: "diskNotReady"L,
            needsConversion: "needsConversion"L,
            needsRiskyRepair: "needsRiskyRepair"L];
          WriteString[e[x]]};
        TellError["Scavenger.Error["L]; PrintNames[error]; WriteChar[']];
        DebugAsk[];
        CONTINUE};
      VolumeConversion.Error =>{
        PrintNames: PROC [x: VolumeConversion.ErrorType] = {
          e: ARRAY VolumeConversion.ErrorType OF STRING = [
            hardwareBroken: "hardwareBroken"L,
            lostLog: "lostLog"L,
            runPreviousScavenger: "runPreviousScavenger"L,
            volumeVersionTooNew: "volumeVersionTooNew"L,
            volumeVersionTooOld: "volumeVersionTooOld"L];
          WriteString[e[x]]};
        TellError["VolumeConversion.Error["L]; PrintNames[error]; WriteChar[']];
        DebugAsk[];
        CONTINUE};
      Volume.InsufficientSpace => {
        TellError["Volume.InsufficientSpace"L];
        DebugAsk[]; CONTINUE};
      Volume.NotOpen => {
        TellError["Volume.NotOpen"L]; DebugAsk[]; CONTINUE};
      Volume.NeedsScavenging => {
        TellError["Please Scavenge the volume first"L]; CONTINUE};
      Volume.Unknown => {
        TellError["Volume.Unknown"L]; DebugAsk[]; CONTINUE};
      Volume.ReadOnly => {
        TellError["Volume.ReadOnly"L]; DebugAsk[]; CONTINUE};
      String.StringBoundsFault => {
        TellError["String.StringBoundsFault"L]; DebugAsk[]; CONTINUE};
      TryAgain => CONTINUE;
      ABORTED => REJECT;
      ANY => {
        signal: SIGNAL;
        args:  PrincOps.LocalFrameHandle;
        TellError["Uncaught Signal = ["L];
        [signal: signal, signalArgs: args] ←
           SIGNAL SpecialRuntime.GetCurrentSignal;
        WriteOctal[Inline.LowHalf[LOOPHOLE[signal]]];
        WriteChar[',];
        WriteOctal[Inline.HighHalf[LOOPHOLE[signal]]];
        WriteChar[']];
        IF args # NIL THEN {
          size: CARDINAL ← PrincOps.frameSizeMap[Frame.ReadLocalWord[args].fsi]
            - SIZE[PrincOps.LocalOverhead];
          WriteString[", msg = ["L];
          FOR i: CARDINAL IN [0..size-1) DO
            WriteOctal[args[i]]; WriteString[", "L] ENDLOOP;
          WriteOctal[args[size-1]]; WriteChar[']];
          Frame.Free[args]};
        DebugAsk[]; CONTINUE}];
    ENDLOOP;
  END;

-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- TTY Interface Stuff
-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

useADM: BOOLEAN = FALSE;

ttyHandle: TTY.Handle ← OthelloToolDefs.tty;

BlinkDisplay: PUBLIC PROC = {TTY.BlinkDisplay[ttyHandle]};
```

```
MyAborted: ERROR = CODE;

CheckUserAbort: PUBLIC PROC = {
  IF TTY.UserAbort[ttyHandle] THEN {ResetAbort[]; ERROR MyAborted}};

EraseTTYChar: PROC [c: CHARACTER] = {
  SELECT c FROM IN [' ..'~] => NULL; CR => RETURN; ENDCASE => EraseTTYChar[' ];
  TTY.RemoveCharacter[ttyHandle];};

ReadChar: PUBLIC PROC RETURNS [c: CHARACTER] = {
  gotIt: BOOLEAN;
  [gotIt, c] ← GetCommandFileCharacter[];
  IF gotIt THEN RETURN;
  RETURN[TTY.GetChar[ttyHandle]]};

SetCursor: PUBLIC PROC [c: OthelloDefs.Cursor] = {
  cursor: ARRAY OthelloDefs.Cursor OF UserTerminal.CursorArray = [
    pointer: [
      100000B, 140000B, 160000B, 170000B, 174000B, 176000B, 177000B, 170000B,
      154000B, 114000B, 006000B, 006000B, 003000B, 003000B, 001400B, 001400B],
    ftp: [
      000177B, 076077B, 040037B, 040017B, 070007B, 043703B, 040401B, 040400B,
      000400B, 100436B, 140421B, 160421B, 170036B, 174020B, 176020B, 177020B]];
    IF ~useADM THEN UserTerminal.SetCursorPattern[cursor[c]];
    cursorFlipped ← FALSE};

cursorFlipped: BOOLEAN;

FlipCursor: PUBLIC PROC = {
  IF ~useADM THEN  {
    c: UserTerminal.CursorArray ← UserTerminal.GetCursorPattern[];
    FOR i: CARDINAL IN [0..LENGTH[c]) DO c[i] ← Inline.BITNOT[c[i]] ENDLOOP;
    UserTerminal.SetCursorPattern[c]}
  ELSE {
    IF cursorFlipped THEN WriteChar[BS] ELSE WriteChar[SP];
    cursorFlipped ← ~cursorFlipped}};

FlushInput: PROC = {
  UNTIL TTY.CharsAvailable[ttyHandle] = 0 DO
    [] ← TTY.GetChar[ttyHandle] ENDLOOP};

NewLine: PUBLIC PROC = {WriteChar[CR]};

ResetAbort: PROC = {TTY.ResetUserAbort[ttyHandle]};

WriteChar: PUBLIC PROC [c: CHARACTER] = {
  IF prometheusBound AND OthelloDefs.SuppressOutput[] THEN RETURN;
  TTY.PutChar[ttyHandle, c]};

WriteLine: PUBLIC PROC [s: LONG STRING] = {WriteString[s]; NewLine[]};

WriteString: PUBLIC PROC [s: LONG STRING] = {
  IF prometheusBound AND OthelloDefs.SuppressOutput[] THEN RETURN;
  IF s # NIL THEN TTY.PutString[ttyHandle, s]};

command:      LONG STRING ← NIL;
commandIndex: CARDINAL ← 0;

CommandFileActive: PROC RETURNS [BOOLEAN] = INLINE {RETURN[command#NIL]};

GetCommandFileCharacter: PROC RETURNS [
  isThere: BOOLEAN, c: CHARACTER] = INLINE {
  IF command # NIL THEN {
    IF commandIndex >= command.length THEN  {
      Heap.systemZone.FREE[@command]; command ← NIL}
    ELSE {
     commandIndex ← commandIndex + 1;
     RETURN[TRUE, command[commandIndex-1]]}};
  RETURN[FALSE, 0C]};

SetCommandString: PUBLIC PROC [s: LONG STRING] = {
  IF command # NIL THEN Heap.systemZone.FREE[@command];
  command ← s; commandIndex ← 0};

-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- Initialization Stuff
-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
GetTime: PROC = {
  timeTrys:       CARDINAL ← 3;
  time:           System.GreenwichMeanTime;
  LTPs:           System.LocalTimeParameters;
  timeFromServer: BOOLEAN ← TRUE;
  getTimeString:  LONG STRING ← NIL;
  [time, LTPs] ← OthelloOps.GetTimeFromTimeServer[
    ! OthelloOps.TimeServerError => IF error=noResponse THEN {
        IF (timeTrys ← timeTrys-1)=0 THEN {timeFromServer ← FALSE; CONTINUE}
        ELSE {IF timeTrys=2 THEN WriteString["Locating Time Server..."L]; RETRY}}
      ELSE IF error=noCommunicationFacilities THEN {
        WriteLine["not Communication Facilities to find time"L];
        timeFromServer ← FALSE; CONTINUE}
      ELSE ERROR];
  IF timeFromServer THEN {
    IF timeTrys#3 THEN WriteLine["success"L];
    System.SetLocalTimeParameters[LTPs];
    OthelloOps.SetProcessorTime[time];
    RETURN};
```

```
      WriteLine["failed.\rPlease enter time information (type ? for help)"L];
      getTimeString ← Heap.systemZone.NEW[StringBody[10]];
      LTPs ← GetTimeZoneFromUser[@getTimeString ! TryAgain => RETRY];
      System.SetLocalTimeParameters[LTPs];
      spacesInStringOK ← TRUE;
      GetTimeFromUser[@getTimeString ! TryAgain => RETRY];
      spacesInStringOK ← FALSE;
      Heap.systemZone.FREE[@getTimeString]};

   GetTimeFromUser: PROC [p: LONG POINTER TO LONG STRING] = {
     timePrompt: STRING = "Please Enter the date and 24 hour time in form
       DD-MMM-YY HH:MM:SS
Time: "L;
     IF OthelloOps.IsTimeValid[] THEN {
       WriteString["Current time"L]; WriteTime[System.GetGreenwichMeanTime[]];
       IF ~Yes["Do you wish to change the time?: "L] THEN RETURN};
     IF p#NIL THEN p.length ← 0;
     DO
       time: System.GreenwichMeanTime;
       GetName[timePrompt, p];
       time ← PackedTimeFromString[p↑, FALSE];
       IF time=System.gmtEpoch THEN {
         WriteLine["Invalid date/time -- please try again."L]; LOOP};
       WriteString["Set time to"L]; WriteTime[time];
       IF Yes["Okay?: "L] THEN {
         OthelloOps.SetProcessorTime[time]; EXIT}
       ELSE LOOP
       ENDLOOP};

  GetTimeZoneFromUser: PROC [string: LONG POINTER TO LONG STRING]
    RETURNS [ltp: System.LocalTimeParameters] = {
    GetNum: PROC [
      prompt: STRING, min, max, default: INTEGER]
      RETURNS [ans: INTEGER] = {
      string.length ← 0;
      String.AppendDecimal[string↑, default];
      DO
        isNeg: BOOLEAN ← FALSE;
        WriteString[prompt];
        WriteChar['[]; IF ans<0 THEN WriteChar['-]; WriteLongNumber[ABS[min]];
        WriteString[".."L]; WriteLongNumber[max]; WriteString["]: "L];
        GetName[dest: string, signalQuestion: TRUE];
        ans ← 0;
        FOR i: CARDINAL IN [0..string.length) DO
          IF i=0 AND string[i]='- THEN {isNeg ← TRUE; LOOP};
          IF string[i] NOT IN ['0..'9] THEN EXIT;
          ans ← 10*ans + string[i] - '0;
          REPEAT FINISHED => {
            IF isNeg THEN ans ← -ans; IF ans IN [min..max] THEN RETURN};
          ENDLOOP;
        WriteLine["Bad Number !"L];
        ENDLOOP};
    dstSpiel: STRING = "
The ""First day of DST"" is the day of the year on or before which
Daylight Savings Time takes effect, where:
        1 =>  January  1
      366 => December 31.
(The correspondence  between numbers and days is based on a leap
year.  Similarly, ""Last day of DST"" is the day of the year on or
before which Daylight Savings Time ends.  Note that in any given
year,  Daylight Savings Time actually begins and ends at 2 AM on
the last Sunday not  following  the  specified date.  The system
makes this adjustment  for you automatically.  The normal values
are
      121   (April 30) for the first day of DST
      305 (October 31) for the last  day of DST.
If Daylight  Savings  Time is not  observed locally, both values
should be set to zero."L;
    ZoneSpiel: STRING = "
Number  of  hours  between  Greenwich  and local time.  For time
zones west of  Greenwich, the offset is negative; for time zones
east of Greenwich, the offset is positive. Examples:
      San Francisco  -8 hours    (Pacific  time zone)
      Denver         -7 hours    (Mountain time zone)
      Chicago        -6 hours    (Central  time zone)
      Boston         -5 hours    (Eastern  time zone)"L;
    n: INTEGER;

    n ← GetNum[prompt: "Time zone offset from Greenwich "L, min: -12, max: 12, default: -8
    ! Question => {WriteLine[ZoneSpiel]; RETRY}];
    ltp.direction ← IF n<0 THEN west ELSE east; ltp.zone ← ABS[n];
    ltp.zoneMinutes ← GetNum[prompt: "Minute offset "L, min: 0, max: 59, default: 0
    ! Question => {WriteLine["\rAlmost always zero"L]; RETRY}];
    ltp.beginDST ← GetNum[prompt: "First day of DST "L, min: 0, max: 366, default: 121
    ! Question => {WriteLine[dstSpiel]; RETRY}];
    ltp.endDST ← GetNum[prompt: "Last day of DST "L,  min: 0, max: 366, default: 305
    ! Question => {WriteLine[dstSpiel]; RETRY}]};

  PrintHerald: PROC = {
    copyright: STRING = [100];
    version: STRING = [10];
    IF useADM THEN WriteChar['\032]; -- clear screen
    VersionExtras.AppendCopyright[copyright];
    Version.Append[version];
    WriteString[copyright];  WriteString["\n\n"L];
    WriteString["Othello "L]; WriteString[version]; WriteString[" of "L];
    WriteTime[Runtime.GetBcdTime[], FALSE, pacific]};
```

```
  PrintPIDs: PROC = {
    w: Format.StringProc = {WriteString[s]};
    WriteString["Processor = "L];
    Format.HostNumber[proc: w,
      hostNumber: LOOPHOLE[SpecialSystem.GetProcessorID[]], format: hex];
    WriteString[" = "L];
    Format.HostNumber[proc: w,
      hostNumber: LOOPHOLE[SpecialSystem.GetProcessorID[]], format: octal];
    WriteString["B = "L];
    Format.HostNumber[proc: w, hostNumber:
      LOOPHOLE[SpecialSystem.GetProcessorID[]], format: productSoftware];
    NewLine[];
    };

  PrintMemorySize: PROC = {
  size: LONG CARDINAL ← ((SpecialSpace.realMemorySize+255)/256)*64;
    WriteString["Memory size = "L];
    WriteLongNumber[size*2];
    WriteString["K bytes"L];
    NewLine[];
    };

  << The following move to proc. Run
  SetCursor[pointer];
  PrintHerald[];
  PrintPIDs[];
  PrintMemorySize[];
  GetTime[];>>

  END..
```

```
LOG
Time:  1-Oct-81 18:44:29  By: FXH      Action: Re-do module,
                                        add Time Stuff & Proc ID
Time:  13-Nov-81 16:27:44  By: FXH     Action: 8.0e build
Time:  19-Nov-81  9:26:07  By: FXH     Make PackedTimeFromString public for
                                        implementing SetBootFileExpirationDate
Time:  17-Dec-81 17:52:16  By: CRF     Action: 8.0f build -- changed herald.
Time:  29-Dec-81 14:29:14  By: CRF     Action: 8.0g build -- changed herald.
Time:  29-Dec-81 14:29:14  By: FXH     Action: 8.0h build -- changed herald.
Time:   1-Feb-82 16:11:37  By: CRF     Action: 8.0i build -- changed herald.
Time:   3-Feb-82 15:02:19  By: CAJ     Action: Print processor ID all 3
                                        ways using Format.
Time:   8-Feb-82 17:20:50  By: CRF     Action: 8.0j build -- changed herald.
Time:   1-Mar-82 13:55:31  By: CAJ     Action: final 8.0 build -- changed herald.
Time:  20-Aug-82 16:49:26 By: AEF      Action: Change to 9.0b.
Time:  16-Sep-82 11:47:54 By: AEF      Action: Change to 9.0c.
Time:  24-Sep-82 17:24:53 By: AEF      Action: Change to 9.0d.
Time:  30-Sep-82 13:48:48 By: AEF      Action: Change to 9.0.
Time:  12-Dec-82 12:50:23 By: RXJ      Action: 10.0c; remove Storage.
Time:   4-Jun-86 12:03:59 By: NFS      Adapted for OthelloTool
```

```
-- Copyright (C) 1983  by Xerox Corporation. All rights reserved.
-- VolumeInitImplA.mesa edited by:
--    RXJ          2-Dec-83 18:21:20
--    RES         17-Oct-83 14:53:35
--    NFS          4-Jun-86 14:08:44

DIRECTORY
  Device USING [PilotDisk, Type],
  DeviceTypes USING [
    q2000, q2010, q2020, q2030, q2040, q2080, sa1000, sa1004, sa4000,
    sa4008, t300, t80],
  Environment USING [wordsPerPage],
  File USING [
    Delete, File, GetAttributes, ID, nullFile, PageNumber, Type, Unknown],
  Heap USING [systemZone],
  Inline USING [BITROTATE],
  OthelloDefs USING [
    AbortingCommand, CloseFetch, CommandProcessor, Confirm, GetName,
    IndexTooLarge, LeaderPage, leaderPages, lpVersion, MyNameIs, NewLine,
    PackedTimeFromString, Question, ReadNumber, RegisterCommandProc,
    SetCommandString, WriteChar, WriteFixedWidthNumber, WriteLine,
    WriteLongNumber, WriteOctal, WriteString, Yes],
  OthelloOps USING [
    BadSwitches, BootFileType, DecodeSwitches, DeleteTempFiles, GetDriveSize,
    GetNextSubVolume, GetPhysicalVolumeBootFile, GetSwitches, GetVolumeBootFile,
    nullSubVolume, SetDebugger, SetDebuggerSuccess, SetExpirationDate,
    SetExpirationDateSuccess, SetGetSwitchesSuccess, SetPhysicalVolumeBootFile,
    SetSwitches, SubVolume, VoidPhysicalVolumeBootFile, VoidVolumeBootFile],
  OthelloToolDefs USING [CloseVolume],
  PhysicalVolume USING [
    AssertPilotVolume, DamageStatus, Error, GetAttributes, GetHandle, GetNext,
    GetNextBadPage, GetNextDrive, GetNextLogicalVolume, Handle, ID,
    InterpretHandle, MarkPageBad, maxNameLength, noProblems, nullBadPage,
    nullDeviceIndex, nullID, Offline, PageNumber, RepairType, Scavenge,
    ScavengerStatus],
  Process USING [MsecToTicks],
  Runtime USING [IsBound],
  Scavenger USING [
    BootFileType, Error, FileEntry, Header, Problem, RepairType, Scavenge],
  Space USING [CopyIn, Map, ScratchMap, Unmap],
  SpecialVolume USING [OpenVolume],
  String USING [
    AppendCharAndGrow, AppendLongNumber, AppendString, CopyToNewString, Equivalent,
    Length, Replace],
  System USING [
    defaultSwitches, GetLocalTimeParameters, gmtEpoch, GreenwichMeanTime,
    PowerOff, Switches],
  TemporaryBooting USING [BootButton, BootFromVolume],
  Volume USING [
    Erase, GetAttributes, GetLabelString, GetType, ID,
    NeedsScavenging, NotOnline, nullID, Open, systemID, Type],
  VolumeVersion USING [Examine];

VolumeInitImplA: PROGRAM
  IMPORTS
    File, Heap, Inline, OthelloDefs, OthelloOps, OthelloToolDefs, PhysicalVolume, Process, Runtime,
    Scavenger, Space, SpecialVolume, System, String, TemporaryBooting, Volume,
    VolumeVersion
  EXPORTS OthelloDefs
  SHARES File =
  BEGIN OPEN OthelloOps, OthelloDefs;

  commandProcessor: CommandProcessor ← [CommonCommands];

  CommonCommands: PROC [index: CARDINAL] = {
    SELECT index FROM
      0 => BootBoot[];
      1 => DeleteBootFiles[];
      2 => DeleteTempFilesUser[];
      3 => DescribePhysicalVolumes[];
      4 => Erase[];
      5 => ListBadPages[];
      6 => ListBootFiles[];
      7 => ListDrives[];
      8 => ListLogicalVolumes[];
      9 => ListPhysicalVolumes[];
     10 => MakeBad[];
     11 => Offline[];
     12 => Online[];
     13 => PowerOff[];
     14 => PVScavenge[];
     15 => Quit[];
     16 => Scavenge[];
     17 => SetBootFileSwitches[];
     18 => SetDebuggerUser[];
     19 => SetExpirationDateUser[];
     20 => SetPvBoot[];
     21 => WizardMode[];
    ENDCASE => IndexTooLarge};

  logicalVolumeTypeString: ARRAY Volume.Type OF LONG STRING ← [
    "normal", "debugger", "debuggerDebugger", "nonPilot"];

  inputDriveString:      LONG STRING ← NIL;
  inputLogicalString:    LONG STRING ← NIL;
  debuggerLogicalString: LONG STRING ← NIL;
  inputPhysString:       LONG STRING ← NIL;
```

```
  switches:              LONG STRING ← NIL;
  lvTypeString:          LONG STRING ← NIL;
  expirationString:      LONG STRING ← NIL;

  maxNameLength: CARDINAL = PhysicalVolume.maxNameLength;

  BootBoot: PROC =
    BEGIN
    lvID: Volume.ID;
    ts:   System.Switches;

    MyNameIs[
      myNameIs: "Boot"L, myHelpIs: "Boot From Logical Volume"L];
    lvID ← GetLvIDFromUser[].lvID;
    GetSetBootFileSwitches[get, lvID
    ! Volume.NeedsScavenging, File.Unknown => {
        WriteLine["(can't get default switches)"L];
        CONTINUE};
      AbortingCommand => {
        WriteLine[reason];
        WriteLine["(can't get default switches)"L];
        CONTINUE}];
    DO
      GetName["switches: "L, @switches, echo, TRUE
      ! Question => {
        WriteLine[
          "See Pilot Users Handbook for list of valid switches."L];
        RESUME}];
      ts ← DecodeSwitches[switches
      ! BadSwitches => {WriteLine["bad switches"L]; LOOP}];
      EXIT;
      ENDLOOP;
    IF Runtime.IsBound[LOOPHOLE[CloseFetch]] THEN CloseFetch[];
    TemporaryBooting.BootFromVolume[lvID, ts];
    END;

  DeleteBootFiles: PROC =
    BEGIN
    lvID: Volume.ID;
    pvID: PhysicalVolume.ID;
    MyNameIs[
      myNameIs: "Delete Boot Files"L,
      myHelpIs: "Delete all boot files from volume"L];
    [pvID: pvID, lvID: lvID] ← GetLvIDFromUser[];
    IF lvID = Volume.systemID THEN {
      WriteLine["Can not delete boot file of current system volume."L];
      RETURN;};
    FOR t: BootFileType IN [hardMicrocode..pilot] DO
      file: File.File = GetVolumeBootFile[lvID, t].file;
      IF file = File.nullFile THEN LOOP;
      Volume.Open[file.volumeID];
      BEGIN ENABLE File.Unknown => CONTINUE;
      File.Delete[file];
      END;
      VoidVolumeBootFile[lvID, t];
      IF GetPhysicalVolumeBootFile[pvID, t].file = file THEN
        VoidPhysicalVolumeBootFile[pvID, t];
      OthelloToolDefs.CloseVolume[lvID];
      ENDLOOP;
    END;

  DeleteTempFilesUser: PROC = {
    lv: Volume.ID;
    MyNameIs[
      myNameIs: "Delete Temporary Files"L,
      myHelpIs: "Delete Temporary Files"L];
    lv ← GetLvIDFromUser[].lvID;
    IF lv = Volume.systemID THEN {
      WriteLine["Can not delete temp files on current system volume."L];
      RETURN;};
    DeleteTempFiles[lv]};

  DescribePhysicalVolumes: PROC =
    BEGIN
    pvID:    PhysicalVolume.ID ← PhysicalVolume.nullID;
    pvFound: BOOLEAN           ← FALSE;

    MyNameIs[
      myNameIs: "Describe Physical Volumes"L,
      myHelpIs: "Describe online physical volumes"L];
    DO
      h:       PhysicalVolume.Handle;
      s:       STRING    ← [maxNameLength];
      sV:      SubVolume ← nullSubVolume;
      sVFound: BOOLEAN   ← FALSE;
      IF (pvID ← PhysicalVolume.GetNext[pvID]) = PhysicalVolume.nullID THEN EXIT;
      pvFound ← TRUE;
      h ← PhysicalVolume.GetAttributes[pvID, s].instance;
      WriteString["Physical Volume "L];
      WriteString[s]; WriteString[" on drive "L];
      WriteString[GetDriveStringName[h]];
      WriteString[" ("L];
      WriteString[
        SELECT GetDriveType[h] FROM
          DeviceTypes.sa1004, DeviceTypes.sa1000  => "Shugart 1000"L,
          DeviceTypes.sa4000, DeviceTypes.sa4008  => "Shugart 4000"L,
          DeviceTypes.q2000,  DeviceTypes.q2010, DeviceTypes.q2020,
```

```
          DeviceTypes.q2030, DeviceTypes.q2040, DeviceTypes.q2080    => "Quantum 2000"L,
          DeviceTypes.t80  =>   "T80"L,
          DeviceTypes.t300 =>   "T300"L,
          ENDCASE =>            "unknown type"L];
      DO
        needsScavenging: BOOLEAN ← FALSE;
        freePages, volumeSize: LONG CARDINAL;
        sV ← GetNextSubVolume[pvID, sV];
        IF sV = nilSubVolume THEN EXIT;
        IF ~sVFound THEN WriteLine[") contains:"L];
        sVFound ← TRUE;
        WriteString["Volume "L];
        [volumeSize: volumeSize, freePageCount: freePages] ← Volume.GetAttributes[sV.lvID
          ! Volume.NeedsScavenging => {
            needsScavenging ← TRUE;
            volumeSize ← 0; -- don't really know
            CONTINUE }];
        IF volumeSize # sV.subVolumeSize AND volumeSize # 0 THEN
          WriteString["piece "L];
        GetLogicalVolumeName[sV.lvID, s];
        WriteString[s]; WriteString[" (type =  "L];
        WriteString[GetLogicalVolumeTypeName[sV.lvID]];  WriteString[") "L];
        IF volumeSize = sV.subVolumeSize THEN {
          WriteLongNumber[freePages]; WriteString[" of "L];
          WriteLongNumber[volumeSize]; WriteString[" pages free"L]}
        ELSE {WriteLongNumber[sV.subVolumeSize]; WriteString[" pages"L]};
        IF needsScavenging THEN WriteString["    *** Needs Scavenging ***"L];
        NewLine[];
        WriteString["   starting at physical address "L];
        WriteLongNumber[sV.firstPVPageNumber];
        NewLine[];
        IF ~needsScavenging THEN ShowBootFiles[pvID, sV.lvID];
        ENDLOOP;
      IF ~sVFound THEN WriteLine[") no subvolumes"L];
      ENDLOOP;
    IF ~pvFound THEN WriteLine["No physical Volumes found"L];
    END;

  Erase: PROC = {
    lvID: Volume.ID;
    pvID: PhysicalVolume.ID;
    MyNameIs[
      myNameIs: "Erase"L, myHelpIs: "Erase Logical Volume"L];
    [pvID: pvID, lvID: lvID] ← GetLvIDFromUser[];
    IF lvID = Volume.systemID THEN {
      WriteLine["Can not erase current system volume."L];
      RETURN;};
    Confirm[];
    OthelloToolDefs.CloseVolume[lvID];
    SELECT VolumeVersion.Examine[lvID] FROM
      otherVersion =>
        IF Yes["That volume is not in the current format. Do you want to convert it? "L]
          THEN Confirm[twice]
        ELSE RETURN;
      ENDCASE;
    WriteString["Erasing..."L];
    Volume.Erase[lvID];
    FOR t: BootFileType IN [hardMicrocode..pilot] DO
      IF GetPhysicalVolumeBootFile[pvID, t].file.volumeID = lvID THEN
        VoidPhysicalVolumeBootFile[pvID, t];
      ENDLOOP;
    WriteLine["complete"L]};

  ListBadPages: PROC =
    BEGIN
    id: PhysicalVolume.ID;
    page: PhysicalVolume.PageNumber ← PhysicalVolume.nullBadPage;
    badSpots: BOOLEAN ← FALSE;
    col: CARDINAL ← 0;
    MyNameIs[
      myNameIs: "List Bad Pages"L,
      myHelpIs: "List known bad pages on Pilot volume"L];
    id ← GetPvIDFromUser[].id;
    WHILE (page ← PhysicalVolume.GetNextBadPage[id, page]) #
      PhysicalVolume.nullBadPage DO
      IF col = 6 THEN BEGIN NewLine[]; col ← 0; END;
      WriteFixedWidthNumber[page, 11];
      col ← col + 1; badSpots ← TRUE;
      ENDLOOP;
    WriteLine[IF badSpots THEN NIL ELSE "No known bad spots"L];
    END;

  ListBootFiles: PROC =
    BEGIN
    lvID: Volume.ID;
    pvID: PhysicalVolume.ID;
    MyNameIs[
      myNameIs: "List Boot Files"L,
      myHelpIs: "List boot files on Pilot volume"L];
    [pvID: pvID, lvID: lvID] ← GetLvIDFromUser[];
    ShowBootFiles[pvID, lvID];
    END;

  ListDrives: PROC = {
    index: CARDINAL ← PhysicalVolume.nullDeviceIndex;
    first: BOOLEAN  ← TRUE;
    MyNameIs[myNameIs: "List Drives"L, myHelpIs: "List Drives"L];
```

```
    DO
      index ← PhysicalVolume.GetNextDrive[index];
      IF index = PhysicalVolume.nullDeviceIndex THEN EXIT;
      IF ~first THEN WriteString[", "L];
      first ← FALSE;
      WriteString[GetDriveStringName[PhysicalVolume.GetHandle[index]]];
      ENDLOOP;
    NewLine[]};

  ListLogicalVolumes: PROC =
    BEGIN
    first: BOOLEAN ← TRUE;
    pID: PhysicalVolume.ID ← PhysicalVolume.nullID;
    MyNameIs[
      myNameIs: "List Logical Volumes"L, myHelpIs: "List Logical Volumes"L];
    DO
      sV: SubVolume ← nullSubVolume;
      pID ← PhysicalVolume.GetNext[pID];
      IF pID = PhysicalVolume.nullID THEN EXIT;
      DO
        s: STRING ← [maxNameLength];
        sV ← GetNextSubVolume[pID, sV];
        IF sV = nullSubVolume THEN EXIT;
        IF sV.firstLVPageNumber # 0 THEN LOOP;
        IF ~first THEN WriteString[", "L];
        WriteString[GetDriveStringName[
          PhysicalVolume.GetAttributes[pID].instance]];
        WriteChar[':];
        GetLogicalVolumeName[sV.lvID, s];
        WriteString[s];
        first ← FALSE;
        ENDLOOP;
      ENDLOOP;
    WriteLine[IF first THEN "No logical volumes found"L ELSE NIL];
    END;

  ListPhysicalVolumes: PROC =
    BEGIN
    s:           STRING = [maxNameLength];
    driveString: LONG STRING;
    first:       BOOLEAN           ← TRUE;
    pID:         PhysicalVolume.ID ← PhysicalVolume.nullID;
    MyNameIs[
      myNameIs: "List Physical Volumes"L,
      myHelpIs: "List Physical Volumes"L];
    DO
      pID ← PhysicalVolume.GetNext[pID];
      IF pID = PhysicalVolume.nullID THEN EXIT;
      driveString ← GetDriveStringName[
        PhysicalVolume.GetAttributes[pID, s].instance];
      IF ~first THEN WriteString[", "L];
      WriteString[driveString];
      WriteChar[':];
      WriteString[s];
      first ← FALSE;
      ENDLOOP;
    WriteLine[IF ~first THEN NIL ELSE "No physical volumes found"L];
    END;

  MakeBad: PROC =
    BEGIN
    h:    PhysicalVolume.Handle;
    id:   PhysicalVolume.ID;
    page: PhysicalVolume.PageNumber;
    IF ~Wizard[] THEN RETURN;
    MyNameIs[
      myNameIs: "Make Page Bad"L,
      myHelpIs: "Enter page into bad page table"L];
    [id, h] ← GetPvIDFromUser[];
    page ← ReadNumber["Decimal Page Number: "L, 0, GetDriveSize[h] - 1];
    PhysicalVolume.MarkPageBad[id, page];
    WriteLine["Consider scavenging some logical volumes."L];
    END;

  Offline: PROC = {
    MyNameIs[
      myNameIs: "Offline"L, myHelpIs: "Bring physical volume offline"L];
    PhysicalVolume.Offline[GetPvIDFromUser[].id]};

  Online: PROC = {
    pvID: PhysicalVolume.ID;
    MyNameIs[
      myNameIs: "Online"L, myHelpIs: "Bring drive online"L];
    pvID ← PhysicalVolume.AssertPilotVolume[GetDriveFromUser[] !
      PhysicalVolume.Error => IF error = alreadyAsserted THEN CONTINUE];
    -- (maybe) update time parameters on disk
    [] ← System.GetLocalTimeParameters[pvID]};

  PowerOff: PROC = {
    MyNameIs[
      myNameIs: "Power Off", myHelpIs: "Execute System.PowerOff"L];
    Confirm[]; CloseFetch[]; System.PowerOff[]};

  PVScavenge: PROC =
    BEGIN OPEN PV: PhysicalVolume;
    convert: BOOLEAN ← FALSE;
    s:       PV.ScavengerStatus;
```

```
    h:      PV.Handle;
    repair: PV.RepairType;
    p:      PV.ID ← PV.nullID;
    PrintDamageStatus: PROC [s: STRING, d: PV.DamageStatus] = {
      IF d=okay THEN RETURN;
      WriteString[s];
      WriteLine[IF d=damaged THEN " damaged"L ELSE " lost"L]};

    MyNameIs[
      myNameIs: "Physical Volume Scavenge"L,
      myHelpIs: "Scavenge physical volume"L];
    h ← GetDriveFromUser[];
    repair ←
      IF ~Yes["Repair? "L] THEN checkOnly
      ELSE IF Wizard[] AND Yes["Risky repair? "L] THEN riskyRepair ELSE safeRepair;
    Confirm[];
    DO
      IF (p←PhysicalVolume.GetNext[p]) = PhysicalVolume.nullID THEN EXIT;
      IF h = PhysicalVolume.GetAttributes[p].instance THEN {
        PhysicalVolume.Offline[p]; EXIT};
      ENDLOOP;
    BEGIN ENABLE PhysicalVolume.Error =>
      IF error = needsConversion AND ~convert THEN
        IF (convert ← Yes["That volume is not in the current format. Do you want to convert it? "L])
          THEN {Confirm[]; RETRY}
        ELSE AbortingCommand["Volume cannot be scavenged"L];
      WriteString["Scavenging..."L];
      s ← PV.Scavenge[h, repair, convert];
      WriteLine["Complete"L];
      END; -- ENABLE
    IF s = PV.noProblems THEN {WriteLine["No problems detected"L]; RETURN};
    WriteString["Damage detected:  "L];
    IF s.internalStructures # okay THEN {
      WriteString["Internal structures "L];
      WriteLine[
        IF s.internalStructures=damaged THEN
          IF repair=safeRepair THEN "damaged -- contact hardware support for risky repair"L
          ELSE "damaged"L
        ELSE "repaired"L]};
    PrintDamageStatus["Bad page list"L,        s.badPageList];
    PrintDamageStatus["Boot file"L,            s.bootFile];
    PrintDamageStatus["Germ"L,                 s.germ];
    PrintDamageStatus["Pilot microcode"L,      s.softMicrocode];
    PrintDamageStatus["Diagnostic microcode"L, s.hardMicrocode];
    END;

  Quit: PROC = {
    MyNameIs[
      myNameIs: "Quit"L, myHelpIs: "Push the boot button"L];
    Confirm[]; CloseFetch[]; TemporaryBooting.BootButton[]};

  Scavenge: PROC = {
    convert: BOOLEAN ← FALSE;
    lvID:    Volume.ID;
    logFile: File.File;
    logPage: File.PageNumber ← 0;
    logWd:   CARDINAL ← Environment.wordsPerPage;
    buffer: LONG POINTER TO ARRAY [0..Environment.wordsPerPage) OF UNSPECIFIED ← NIL;

    GetWds: PROC [p: POINTER, c: CARDINAL] = {
      WHILE c#0 DO
        IF logWd=Environment.wordsPerPage THEN {
          [] ← Space.CopyIn[buffer, [logFile, logPage, 1]];
          logPage ← logPage + 1; logWd ← 0};
        p↑ ← buffer[logWd];
        p ← p+1; c ← c-1; logWd ← logWd+1;
        ENDLOOP};
    DisplayScavLog: PROC = {
      fileCount: LONG CARDINAL; problems: BOOLEAN ← FALSE;
      BEGIN
      hd: Scavenger.Header;
      GetWds[@hd, SIZE[Scavenger.Header]];
      WriteString["volume"L]; IF ~hd.repaired THEN WriteString[" not"L];
      WriteString[" repaired, log file"L];
      IF hd.incomplete THEN WriteString[" not"L]; WriteLine[" complete "L];
      WriteLongNumber[fileCount ← hd.numberOfFiles];
      WriteLine[" files on volume"L];
      END;
      WHILE fileCount#0 DO
        OpenID: TYPE = ARRAY [0..SIZE[File.ID]) OF CARDINAL;
        fe:     Scavenger.FileEntry;
        GetWds[@fe, SIZE[Scavenger.FileEntry]];
        THROUGH [0..fe.numberOfProblems) DO
          fp: Scavenger.Problem;
          GetWds[@fp, SIZE[Scavenger.Problem]];
          WriteChar['[];
          FOR i: CARDINAL IN [0..SIZE[File.ID]-1) DO
            WriteOctal[LOOPHOLE[fe.file, OpenID][i]]; WriteString[", "L] ENDLOOP;
          WriteOctal[LOOPHOLE[fe.file, OpenID][SIZE[File.ID]-1]];
          WriteString["] type = "L];
          BEGIN ENABLE File.Unknown => GOTO noType;
          f: File.Type = File.GetAttributes[[fe.file, lvID]].type;
          WriteLongNumber[LONG[LOOPHOLE[f, CARDINAL]]];
          EXITS noType => WriteString["unknown"L];
          END;
          WriteString["; "L];
          WITH fp SELECT FROM
```

```
            unreadable => {
              WriteString["unreadable"L];
              WriteString[" pages ["L]; WriteLongNumber[first];
              WriteString[".."L]; WriteLongNumber[first+count]; WriteLine[")"L]};
            missing => {
              WriteString["missing"L];
              WriteString[" pages ["L]; WriteLongNumber[first];
              WriteString[".."L]; WriteLongNumber[first+count]; WriteLine[")"L]};
            duplicate => {
              WriteString["duplicate"L]; WriteLine[" page found"L]};
            orphan => {
              WriteString["orphan"L]; WriteLine[" page found"L]};
            ENDCASE => WriteLine["unknown problem"L];
          problems ← TRUE;
          ENDLOOP;
        fileCount ← fileCount-1;
        ENDLOOP;
      WriteLine[IF ~problems THEN "No problems found"L ELSE NIL]};
    MyNameIs[
      myNameIs: "Scavenge"L, myHelpIs: "Scavenge Logical Volume"L];
    lvID ← GetLvIDFromUser[].lvID;
    IF lvID = Volume.systemID THEN {
      WriteLine["Can not scavenge current system volume."L];
      RETURN;};
    Confirm[];
    OthelloToolDefs.CloseVolume[lvID ! ANY => CONTINUE];
    BEGIN ENABLE Scavenger.Error =>
      IF error = needsConversion AND ~convert THEN
        IF (convert ← Yes["That volume is not in the current format. Do you want to convert it? "L])
          THEN {Confirm[twice]; RETRY}
        ELSE AbortingCommand["Volume cannot be scavenged"L];
      WriteString["Scavenging..."L];
      logFile ← Scavenger.Scavenge[lvID, lvID, safeRepair, convert];
      WriteLine["Complete"L];
      END; -- ENABLE
    SpecialVolume.OpenVolume[lvID, read];
    buffer ← Space.ScratchMap[1];
    DisplayScavLog[
      ! UNWIND => {[] ← Space.Unmap[buffer]; OthelloToolDefs.CloseVolume[lvID]}];
    [] ← Space.Unmap[buffer]; OthelloToolDefs.CloseVolume[lvID]};

  SetBootFileSwitches: PROC =
    BEGIN
    ts:   System.Switches;
    lvID: Volume.ID;

    MyNameIs[
      myNameIs: "Set Boot File Default Switches"L,
      myHelpIs: "Set default switches for boot file on volume"L];
    lvID ← GetLvIDFromUser[].lvID;
    GetSetBootFileSwitches[get, lvID];  -- volume.needsScav (caught higher up)
    DO
      GetName["switches: "L, @switches, echo, TRUE
      ! Question => {WriteLine[
        "See Pilot Users Handbook for list of valid switches."L];
        RESUME}];
      ts ← DecodeSwitches[switches
      ! BadSwitches => {WriteLine["bad switches"L]; LOOP}];
      EXIT;
      ENDLOOP;
    Confirm[];
    GetSetBootFileSwitches[set, lvID, ts];
    END;

  SetDebuggerUser: PROC =
    BEGIN
    file:       File.File;
    firstPage: File.PageNumber;
    lvID:      Volume.ID;
    dLvID:     Volume.ID;
    dH:        PhysicalVolume.Handle;
    dT:        Device.Type;
    dO:        CARDINAL;
    outcome:   SetDebuggerSuccess;

    MyNameIs[
      myNameIs: "Set Debugger Pointers"L,
      myHelpIs: "Set up pointers to debugger for volume"L];
    lvID ← GetLvIDFromUser["for debuggee Logical Volume: "L].lvID;
    [file, firstPage] ← GetVolumeBootFile[lvID, pilot];
    IF file = File.nullFile THEN
      AbortingCommand["No boot file found."L];
    [, dLvID, dH] ← GetLvIDFromUser["for debugger Logical Volume: "L, TRUE];
    IF dLvID=Volume.nullID THEN WriteLine["(Clear existing pointers)"L]
    ELSE {dT ← GetDriveType[dH]; dO ← GetDriveNumber[dH]};
    Confirm[];
    Volume.Open[lvID];
    outcome ← SetDebugger[
      debuggeeFile: file, debuggeeFirstPage: firstPage, debugger: dLvID,
      debuggerType: dT, debuggerOrdinal: dO];
    OthelloToolDefs.CloseVolume[lvID];
    WriteSetDebuggerSuccess[outcome];
    END;

  SetExpirationDateUser: PROC =
    BEGIN
    file:       File.File;
```

```
    firstPage: File.PageNumber;
    time:      System.GreenwichMeanTime;
    lvID:      Volume.ID;
    outcome:   SetExpirationDateSuccess;

    MyNameIs[
      myNameIs: "Set Hardware Clock Upper Limit"L,
      myHelpIs: "Set last believable hardware clock date for boot file on logical volume"L];
    lvID ← GetLvIDFromUser[].lvID;
    [file, firstPage] ← GetVolumeBootFile[lvID, pilot];
    IF file = File.nullFile THEN
      AbortingCommand["No boot file found."L];
    DO
      GetName["Date (DD-MMM-YY): "L, @expirationString];
      IF expirationString.length=0 THEN {
        WriteLine["(setting no upper limit on hardware clock)"L];
        time ← System.gmtEpoch;
        EXIT}
      ELSE {
        time ← PackedTimeFromString[s: expirationString, justDate: TRUE];
        IF time=System.gmtEpoch THEN WriteLine["invalid date"L]
        ELSE EXIT};
      ENDLOOP;
    Confirm[];
    Volume.Open[lvID];
    outcome ← SetExpirationDate[file, firstPage, time];
    OthelloToolDefs.CloseVolume[lvID];
    WriteSetDebuggerSuccess[outcome];
    END;

  SetPvBoot: PROC =
    BEGIN
    lvID: Volume.ID;
    set: ARRAY BootFileType[hardMicrocode..pilot] OF BOOLEAN ← ALL[FALSE];
    found, changed: BOOLEAN ← FALSE;
    Smash: PROC [s: STRING, t: BootFileType] = {
      IF GetVolumeBootFile[lvID, t].file = File.nullFile THEN RETURN;
      found ← TRUE;
      WriteString["Set physical volume "L];
      WriteString[s];
      IF (set[t] ← Yes[" from this logical volume? "L]) THEN changed ← TRUE};

    MyNameIs[
      myNameIs: "Set Physical Volume Boot Files"L,
      myHelpIs: "Set Physical Volume Boot Files"L];
    lvID ← GetLvIDFromUser[].lvID;
    Smash["boot file"L, pilot];
    Smash["pilot microcode"L, softMicrocode];
    Smash["germ file"L, germ];
    Smash["diagnostic microcode"L, hardMicrocode];
    IF ~found THEN AbortingCommand["Logical volume has null boot files"L];
    IF ~changed THEN RETURN;
    Confirm[];
    SpecialVolume.OpenVolume[lvID, read];
    FOR t: BootFileType IN [hardMicrocode..pilot] DO
      IF set[t] THEN {
        file: File.File;
        firstPage: File.PageNumber;
        [file, firstPage] ← GetVolumeBootFile[lvID, t];
        SetPhysicalVolumeBootFile[file, t, firstPage]};
      ENDLOOP;
    OthelloToolDefs.CloseVolume[lvID];
    END;

  ShowBootFiles: PROC [pv: PhysicalVolume.ID, lv: Volume.ID] = {
    bootNames: ARRAY BootFileType[hardMicrocode..pilot] OF STRING ← [
      hardMicrocode: "Diagnostic microcode"L,
      softMicrocode: "Pilot microcode"L,
      germ: "Germ"L,
      pilot: "Pilot bootfile"L];
    SpecialVolume.OpenVolume[lv, read ! Volume.NeedsScavenging => GOTO scavenge];
    FOR t: BootFileType IN BootFileType[hardMicrocode..pilot] DO
      ENABLE UNWIND => OthelloToolDefs.CloseVolume[lv];
      file: File.File;
      firstPage: File.PageNumber;
      [file: file, firstPage: firstPage] ← GetVolumeBootFile[lv, t];
      IF file = File.nullFile THEN LOOP;
      WriteString["   "L];
      IF GetPhysicalVolumeBootFile[pv, t].file = file THEN
        WriteString["(PV) "L];
      WriteString[bootNames[t]];
      WriteString[": "L];
      IF firstPage = OthelloDefs.leaderPages THEN ShowLeaderNote[file]
      ELSE WriteLine["(no information available)"L];
      ENDLOOP;
    OthelloToolDefs.CloseVolume[lv];
    EXITS
    scavenge => NULL};

  ShowLeaderNote: PROC [file: File.File] = {
    lp: LONG POINTER TO OthelloDefs.LeaderPage;
    lp ← Space.Map[window:[file, 0, OthelloDefs.leaderPages], access: readOnly].pointer;
    IF lp.version = OthelloDefs.lpVersion THEN
      FOR i: CARDINAL IN [0..lp.length) DO WriteChar[lp.note[i]] ENDLOOP
    ELSE WriteString["(no information available)"L];
    NewLine[];
```

```
    [] ← Space.Unmap[lp]};

  WizardMode: PROC = {
    password: LONG STRING ← NIL;
    IF wizardMode THEN RETURN;
    MyNameIs[
      myNameIs: "Wizard Mode"L, myHelpIs: "Enable special commands"L];
    GetName["Password: "L, @password, stars];
    IF Hash[password] = wizardPassword THEN {wizardMode ← TRUE; WriteLine[" ok"L]}
    ELSE WriteLine[" incorrect password"L]};

  -- Wizard Supporting Procedures
  wizardMode: BOOLEAN ← Process.MsecToTicks[2000] # 39; --FALSE for DLion only.
  wizardPassword: CARDINAL = 18939;

  Hash: PROCEDURE [s: LONG STRING] RETURNS [h: CARDINAL] =
    BEGIN
    h ← 17777;
    FOR i: CARDINAL IN [0..String.Length[s]) DO
      c: CHARACTER;
      IF (c ← s[i]) IN ['A..'Z] THEN c ← c + ('a-'A);
      h ← Inline.BITROTATE[h, 1] + (c-0C);
      ENDLOOP;
    END;

  Wizard: PUBLIC PROC RETURNS [BOOLEAN] = {RETURN[wizardMode]};

  -- Volume Init Supporting Procedures
  unknown: LONG STRING = "Unknown";

  GetSetBootFileSwitches: PROC [
    getSet: {get, set}, lvID: Volume.ID,
    ts: System.Switches ← System.defaultSwitches] = {
    outcome:   SetGetSwitchesSuccess;
    file:       File.File;
    firstPage: File.PageNumber;

    Heap.systemZone.FREE[@switches];
    IF getSet=get THEN SpecialVolume.OpenVolume[lvID, read]
    ELSE Volume.Open[lvID];
    [file, firstPage] ← GetVolumeBootFile[lvID, pilot];
    IF file = File.nullFile THEN AbortingCommand["No boot file found."L];
    IF getSet=get THEN [outcome, ts] ← GetSwitches[file, firstPage]
    ELSE outcome ← SetSwitches[file, firstPage, ts];
    OthelloToolDefs.CloseVolume[lvID];
    WriteSetDebuggerSuccess[outcome];
    IF getSet=set THEN RETURN;
    FOR c: CHARACTER IN [0C..377C] DO
      IF ts[c]=up THEN LOOP;
      SELECT c FROM
        '~, '-, '\\, '', '" => NULL;
        IN ['a..'z], IN ['A..'Z], IN (' ..'?] => {
          String.AppendCharAndGrow[@switches, c, Heap.systemZone]; LOOP};
        ENDCASE => NULL;
      String.AppendCharAndGrow[@switches, '\\, Heap.systemZone];
      String.AppendCharAndGrow[@switches, (c-0C)/64 + '0, Heap.systemZone];
      String.AppendCharAndGrow[@switches, ((c-0C)/8 MOD 8) + '0, Heap.systemZone];
      String.AppendCharAndGrow[@switches, ((c-0C) MOD 8) + '0, Heap.systemZone];
      ENDLOOP};

  WriteSetDebuggerSuccess: PROC [outcome: SetDebuggerSuccess] = {
    SELECT outcome FROM
      success => NULL;
      nullBootFile, cantWriteBootFile, notInitialBootFile =>
        AbortingCommand["Boot file broken."L];
      cantFindStartListHeader, startListHeaderHasBadVersion =>
        AbortingCommand["file built by incompatible version of StartPilot"L];
      noDebugger => AbortingCommand["No debugger installed."L];
      ENDCASE => ERROR};

  GetDriveFromUser: PUBLIC PROC RETURNS [h: PhysicalVolume.Handle] = {
    DO
      index: CARDINAL ← PhysicalVolume.nullDeviceIndex;
      GetName[
        "Drive Name: "L, @inputDriveString, echo, TRUE
        ! Question => {ListDrives[]; RESUME}];
      IF inputDriveString[inputDriveString.length - 1] = ': THEN
        inputDriveString.length ← inputDriveString.length - 1;
      DO
        index ← PhysicalVolume.GetNextDrive[index];
        IF index = PhysicalVolume.nullDeviceIndex THEN EXIT;
        h ← PhysicalVolume.GetHandle[index];
        IF String.Equivalent[GetDriveStringName[h], inputDriveString] THEN RETURN;
        ENDLOOP;
      WriteLine["Drive not found!"L]
      ENDLOOP};

  GetDriveNumber: PUBLIC PROC [h: PhysicalVolume.Handle] RETURNS [CARDINAL] = {
    RETURN[PhysicalVolume.InterpretHandle[h].index]};

  GetDriveStringName: PROC [h: PhysicalVolume.Handle] RETURNS [s: LONG STRING] =
    BEGIN
    s ← SELECT TRUE FROM
      -- damn compiler won't allow t IN Device.PilotDisk
      LOOPHOLE[GetDriveType[h], CARDINAL] IN Device.PilotDisk
        => "Rd?",
      ENDCASE => "UnknownType?";
```

```
    s[s.length - 1] ← GetDriveNumber[h] + '0;
    END;

  GetDriveType: PUBLIC PROC [h: PhysicalVolume.Handle] RETURNS [Device.Type] = {
    RETURN[PhysicalVolume.InterpretHandle[h].type]};

  GetLogicalVolumeName: PROC [vid: Volume.ID, s: STRING] = {
    s.length ← 0;
    Volume.GetLabelString[vid, s ! Volume.NeedsScavenging => GOTO bad];
    EXITS bad => {
      IDRep: TYPE = RECORD [p: ARRAY [0..3) OF CARDINAL, n: LONG CARDINAL];
      String.AppendString[s, "NeedsScavenging"L];
      String.AppendLongNumber[s, LOOPHOLE[vid, IDRep].n, 8]}};

  GetLogicalVolumeTypeName: PROC [vid: Volume.ID] RETURNS [LONG STRING] = {
    RETURN[logicalVolumeTypeString[Volume.GetType[vid ! ANY => GOTO signal]]];
    EXITS signal => RETURN[unknown]};

  -- Accept string of Form LogicalVolumeName OR
  --    Drive:LogicalVolumeName
  GetLvIDFromUser: PUBLIC PROC [
    prompt: LONG STRING ← NIL,
    calledFromSetDebuggerPtrs: BOOLEAN ← FALSE]
    RETURNS [
      pvID: PhysicalVolume.ID, lvID: Volume.ID,
      drive: PhysicalVolume.Handle] =
    BEGIN
    IF prompt = NIL THEN prompt ← "Logical Volume Name: "L;
    DO
     ptmpID:      PhysicalVolume.ID ← PhysicalVolume.nullID;
     inputString: LONG STRING;
     matches:     CARDINAL ← 0;
     GetName[
        prompt: prompt, how: echo, signalQuestion: TRUE,
        dest: IF calledFromSetDebuggerPtrs THEN @debuggerLogicalString
              ELSE @inputLogicalString
        ! Question => {ListLogicalVolumes[]; RESUME}];
      IF calledFromSetDebuggerPtrs THEN {
        IF debuggerLogicalString.length=0 THEN {lvID ← Volume.nullID; RETURN}
        ELSE inputString ← debuggerLogicalString}
      ELSE {inputString ← inputLogicalString};
      DO
        driveTemp: PhysicalVolume.Handle;
        ltmpID:    Volume.ID ← Volume.nullID;
        IF (ptmpID ← PhysicalVolume.GetNext[ptmpID]) = PhysicalVolume.nullID THEN EXIT;
        driveTemp ← PhysicalVolume.GetAttributes[ptmpID].instance;
        DO
          s: STRING = [maxNameLength];
          IF (ltmpID ← PhysicalVolume.GetNextLogicalVolume[ptmpID, ltmpID])
             = Volume.nullID THEN EXIT;
          GetLogicalVolumeName[ltmpID, s ! Volume.NotOnline => LOOP];
          IF FunnyEqual[driveTemp, s, inputString] THEN {
            matches ← matches + 1; lvID ← ltmpID; pvID ← ptmpID; drive ← driveTemp};
          ENDLOOP;
        ENDLOOP;
      SELECT matches FROM
        0 => WriteString["Not found\r"L];
        1 => RETURN;
        ENDCASE => WriteLine["Ambigous; please specify Device:LogicalName"L];
      ENDLOOP;
    END;

  FunnyEqual: PROC [
    h: PhysicalVolume.Handle, name: STRING, userName: LONG STRING,
    mode: {checkNakedPName, dontCheckNakedPName} ← dontCheckNakedPName]
    RETURNS[BOOLEAN] = {
    driveName: LONG STRING;
    SameChar: PROC [a, b: CHARACTER]
      RETURNS [BOOLEAN] = {
      IF a=b THEN RETURN[TRUE]
      ELSE IF a IN ['a..'z] AND b IN ['A..'Z] AND (a-'a+'A)=b THEN RETURN[TRUE]
      ELSE IF a IN ['A..'Z] AND b IN ['a..'z] AND (a-'A+'a)=b THEN RETURN[TRUE]
      ELSE RETURN[FALSE]};
    IF String.Equivalent[name, userName] THEN RETURN[TRUE];
    driveName ← GetDriveStringName[h];
    IF userName.length < driveName.length THEN RETURN [FALSE];
    FOR i: CARDINAL IN [0..driveName.length) DO
      IF ~SameChar[driveName[i], userName[i]] THEN RETURN[FALSE] ENDLOOP;
    IF mode=checkNakedPName THEN {
      IF (userName.length=driveName.length)
         OR (userName.length=driveName.length+1
             AND userName[driveName.length] = ':) THEN RETURN[TRUE]};
    IF driveName.length+name.length+1 # userName.length THEN RETURN[FALSE];
    IF userName[driveName.length] # ': THEN RETURN[FALSE];
    FOR i: CARDINAL IN [0..name.length) DO
      IF ~SameChar[name[i], userName[driveName.length+1+i]] THEN RETURN[FALSE]
      ENDLOOP;
    RETURN[TRUE]};

  GetLvTypeFromUser: PUBLIC PROC [
    prompt: LONG STRING, defaultType: Volume.Type] RETURNS [Volume.Type] =
    BEGIN
    ListTypes: PROC = {
      FOR t: Volume.Type IN [normal..nonPilot] DO
        WriteString[logicalVolumeTypeString[t]];
        WriteString[IF t = nonPilot THEN "\r"L ELSE ", "L];
        ENDLOOP};
```

```
    String.Replace[@lvTypeString, logicalVolumeTypeString[defaultType], Heap.systemZone];
    DO
      GetName[prompt, @lvTypeString, echo, TRUE
      ! Question => {ListTypes[]; RESUME}];
      FOR t: Volume.Type IN [normal..nonPilot] DO
        IF String.Equivalent[logicalVolumeTypeString[t], lvTypeString] THEN
          RETURN[t]
        ENDLOOP;
      WriteLine["Illegal type"L];
      ENDLOOP;
    END;

  -- Accept string of Form PhysicalVolumeName OR
  --   Drive:PhysicalVolumeName OR Drive
  GetPvIDFromUser: PROC
    RETURNS [id: PhysicalVolume.ID, drive: PhysicalVolume.Handle] =
    BEGIN
    DO
      tmpID:   PhysicalVolume.ID ← PhysicalVolume.nullID;
      matches: CARDINAL ← 0;
      GetName["Physical Volume Name: "L, @inputPhysString, , TRUE
      ! Question => {ListPhysicalVolumes[]; RESUME}];
      DO
        s:         STRING = [maxNameLength];
        match:     BOOLEAN;
        driveTemp: PhysicalVolume.Handle;
        IF (tmpID ← PhysicalVolume.GetNext[tmpID]) = PhysicalVolume.nullID THEN
          EXIT;
        driveTemp ← PhysicalVolume.GetAttributes[tmpID, s].instance;
        match ← FunnyEqual[driveTemp, s, inputPhysString, checkNakedPName];
        IF match THEN {matches ← matches + 1; id ← tmpID; drive ← driveTemp};
        ENDLOOP;
      SELECT matches FROM
        0 => WriteLine["Not Found"L];
        1 => RETURN;
        ENDCASE => WriteLine["Ambigous; please specify Device:PhysicalName"L];
      ENDLOOP;
    END;

  StringInit: PROC = {
    SetCommandString[String.CopyToNewString["Online RD0"L, Heap.systemZone]]};
    debuggerLogicalString ← String.CopyToNewString["CoPilot"L, Heap.systemZone];

  RegisterCommandProc[@commandProcessor];

  StringInit[];

  END.....
```

```
March 19, 1980  3:47 PM FXH     Delete newly created temporary files when fetch fails; ome indentation changing
April 16, 1980  12:16 PM        RXG     Addd diagnostic microcode fetch
May 31, 1980  11:49 PM  FXH     Shuffle around VolumeInitImplA and B
July 30, 1980  4:33 PM  AWL             Permit Online'ing an already online volume
September 18, 1980  12:04 PM    PXM     Don't bother to open volume to boot from
September 19, 1980  11:24 AM    AWL             physicalVolumeOverhead ← 2 for new physical volume format.
September 29, 1980  2:07 PM     CAJ     Add SA800 format and scan, USING clauses.
October 10, 1980  3:17 PM       FXH     Version 5.0.
January 5, 1981  10:14 PM       FXH     Made use String for appendChar, equivilantString, appendLongNumber.  Add TemporaryBooting.invalid
paramater catch.
January 31, 1981  9:19 PM       CAJ     Fix format prompt.
March 1, 1981  12:59 PM AWL             Version => 6.0b.
March 13, 1981  7:22 PM SXY     Version => 6.0c, trouple => trouble (correction), "Boot file header broken" => "Error: Debuggee built by
incompatible version of StartPilot".
March 25, 1981  8:28 PM CRF     Version => 6.0.
April 14, 1981  11:38 AM        BXM     @ added.
11-Jun-81 10:53:01      Taft    Remove all machine- and device-dependent code to separate module OthelloDeviceImplD*.mesa
17-Jul-81 15:34:33      SCG             Merged OthelloDevice into OthelloDefs
12-Aug-81 12:33:54      SXY     Added a catch phrase for Volume.GetAttributes and commented it out for Volume.GetLabelString
 5-Dec-81 17:30:28      CRF     Converted from PhysicalVolumeExtras to PhysicalVolume for PV scavenger.
11-Dec-82 15:10:21      RXJ             Removed Storage.
13-Apr-83 12:27:04      RXJ             Klamath conversion
 4-Jun-86 14:09:04      NFS             Adapted for OthelloTool
```

```
-- VolumeVersion.mesa    17-Aug-83 12:16:18 by WDK

-- TEMPORARY HACK for Klamath <> Trinity cross volume problems.

DIRECTORY
  Volume USING [ID];

VolumeVersion: DEFINITIONS =
  BEGIN

  Examine: PROCEDURE [volume: Volume.ID]
    RETURNS [result: Result];

  Result: TYPE = {
    currentVersion, badRootPageLabel, ioError, trashedRootPage,
    otherVersion, volumeUnknown};

  END.
```

```
-- Copyright (C) 1983, 1984  by Xerox Corporation. All rights reserved.
-- VolumeVersionImpl.mesa       17-Aug-83 11:20:47 by WDK
-- BJD           28-Feb-84 14:10:16

-- TEMPORARY HACK for Klamath <> Trinity cross volume problems.

DIRECTORY
  Device,
  DiskChannel,
  Environment,
  File,
  LogicalVolumeFormat,
  OthelloOps,
  PhysicalVolume,
  PilotDisk,
  PilotFileTypes,
  Space,
  VM,
  Volume,
  VolumeVersion;

VolumeVersionImpl: PROGRAM
  IMPORTS
    DiskChannel, Environment, OthelloOps, PhysicalVolume, PilotDisk, Space, VM
  EXPORTS VolumeVersion =
  BEGIN

  Bug: SIGNAL [b: BugType] = CODE;
  BugType: TYPE = {
    impossibleEndcase, invalidChannel, invalidDriveState};

  Examine: PUBLIC PROCEDURE [volume: Volume.ID]
    RETURNS [result: VolumeVersion.Result] =
    BEGIN
    lvRoot: LONG POINTER TO LogicalVolumeFormat.Descriptor;
    physicalVol: PhysicalVolume.ID =
      PhysicalVolume.GetContainingPhysicalVolume[volume];
    pvHandle: PhysicalVolume.Handle =
      PhysicalVolume.GetAttributes[physicalVol].instance;
    deviceType: Device.Type;
    index: CARDINAL;
    subVolume: OthelloOps.SubVolume ← OthelloOps.nullSubVolume;
    drive: DiskChannel.Drive;
    DO
      subVolume ← OthelloOps.GetNextSubVolume[physicalVol, subVolume];
      IF subVolume = OthelloOps.nullSubVolume THEN RETURN[volumeUnknown];
      IF subVolume.lvID = volume AND subVolume.firstLVPageNumber = 0 THEN EXIT;
      ENDLOOP;

    lvRoot ← Space.ScratchMap[count: 1].pointer;
    [type: deviceType, index: index] ← PhysicalVolume.InterpretHandle[pvHandle];
    FOR drive ← DiskChannel.GetNextDrive[prev: DiskChannel.nullDrive],
       DiskChannel.GetNextDrive[prev: drive]
      UNTIL drive = DiskChannel.nullDrive DO
        dType: Device.Type;
        dOrdinal: CARDINAL;
        [deviceType: dType, deviceOrdinal: dOrdinal] ←
          DiskChannel.GetDriveAttributes[drive];
        IF dType=deviceType AND dOrdinal=index THEN EXIT
        ENDLOOP;

    BEGIN  --scope of Exit--
    pageBuffer: Environment.PageNumber = Environment.PageFromLongPointer[lvRoot];
    channel: DiskChannel.Handle = DiskChannel.Create[drive];
    request: DiskChannel.IORequest ← [
      diskPage: subVolume.firstPVPageNumber, memoryPage: pageBuffer,
      tries: DiskChannel.defaultTries, label: @label,
      count: 1, useSamePage: TRUE, command: [verify, read, read]];
    status: DiskChannel.IOStatus;
    countValid: File.PageCount;
    label: PilotDisk.Label;

    VM.MakeResident[[page: pageBuffer, count: 1], wait];
    [status, countValid] ← DiskChannel.DoIO[channel, @request];
    WITH boundStatus: status SELECT FROM
      invalidChannel => Bug[invalidChannel];
      invalidDriveState => Bug[invalidDriveState];
      disk =>
        IF boundStatus.status # goodCompletion THEN {result ← ioError; GOTO Exit};
      ENDCASE => Bug[impossibleEndcase];

    -- Check Logical Root Page Label:
    IF volume = Volume.ID[label.fileID.id]
      AND PilotDisk.GetLabelFilePage[@label] = LogicalVolumeFormat.rootPageNumber
      AND ~label.temporary AND label.pad1 = 0 AND label.pad2 = 0
      AND label.type = PilotFileTypes.tLogicalVolumeRootPage
      THEN NULL ELSE {result ← badRootPageLabel; GOTO Exit};

    IF lvRoot.seal # LogicalVolumeFormat.lvRootSeal THEN
      {result ← trashedRootPage; GOTO Exit};
    IF lvRoot.version # LogicalVolumeFormat.currentVersion THEN
      {result ← otherVersion; GOTO Exit};
    result ← currentVersion;

    EXITS Exit => NULL
    END;
```

```
    lvRoot ← Space.Unmap[lvRoot];
    END;

END.
```

```
-- AuthAImpl.mesa
-- JMaloney, 11-Jul-83 12:24:49.

-- Last modified: JMaloney, 20-Jun-84 11:16:36.


DIRECTORY
  Auth USING [
    AuthenticationProblem, CallProblem,
    CheckOutCredsAndNextVerifier, CheckSimpleCredentials,
    ConversationHandle, CopyCredentials, Credentials, FetchStrongCredentials,
    Flavor, HashedPassword, HashSimplePassword, IdentityHandle, Key,
    nullCredentials, nullHostNumber, nullKey,
    nullVerifier, PasswordStringToKey, Verifier, WhichArg],
  AuthInternal USING [
    asStubHeap, CloneNSName, CloneNSString, CloneVerifier,
    ConversationObject, EquivalentNames, FreeCredentials,
    FreeNSName, FreeNSString, FreeVerifier, IdentityObject,
    IncrementVerifierTicks, InternalAuthenticate,
    InternalExtractCredentialsDetails, MakeEmptyNSName,
    MakeVerifierFromHashedPassword, MakeSimpleCredentials
    NilOrNullName, PackStrongVerifier, Style, UnpackStrongVerifier],
  AuthProtocol USING [StrongVerifier],
  AuthSpecial USING [],
  CH USING [
    ConversationHandle, FreeConversationHandle,
    LookupDistinguishedName, ReturnCode],
  Courier USING [Error],
  DESFace USING [CheckKeyParity, Key, nullKey],
  Heap USING [Create, MakeNode],
  NSName USING [
    CopyNameFields, maxDomainLength, maxLocalLength, maxOrgLength, Name],
  NSString USING [nullString, String],
  NSStringExtras USING [EquivalentNames],
  Process USING [InitializeMonitor, Pause, SecondsToTicks, Ticks],
  Router USING [AssignAddress],
  SharedKeys USING [asName, chsName, msName],
  SpecialCHAuth USING [],
  System USING [
    GetClockPulses, GetGreenwichMeanTime, gmtEpoch,
    GreenwichMeanTime, HostNumber, NetworkAddress,
    nullHostNumber, SecondsSinceEpoch];

AuthAImpl: MONITOR
  LOCKS LOOPHOLE[identity, PrivateIdentityHandle] USING identity: Auth.IdentityHandle
  IMPORTS
    Auth, AuthInternal, CH, Courier, DESFace, Heap, NSName,
    NSStringExtras, Process, Router, SharedKeys, System
  EXPORTS Auth, AuthSpecial, SpecialCHAuth
  SHARES Auth =

BEGIN


-- Globals and constants --


cacheConversations: BOOLEAN _ TRUE;

conversationCacheTimeout: LONG CARDINAL = LONG[12] * LONG[60] * LONG[60],
  -- 12 hours, in seconds (half of the crentials lifetime).

conversationHeap: UNCOUNTED ZONE _ Heap.Create[initial: 4, increment: 4],
  -- All conversations and associated storage (creds, etc.) are allocated
  -- from this heap. We no longer use the clients heap. This was
  -- because conversation caching made it possible for one
  -- client to lose storage owned by another.

asStubHeap: UNCOUNTED ZONE = AuthInternal.asStubHeap;

timesHaveHadToWaitForNextVerifier: CARDINAL _ 0;
  -- This global is the number of times we've tried to create more than one
  -- verifier within a given second and run out of ticks since the last boot.
  -- It should be pretty small.

thisMachinesAddress: System.NetworkAddress _ Router.AssignAddress[];


-- Public errors --

AuthenticationError: PUBLIC ERROR [reason: Auth.AuthenticationProblem] = CODE;

CallError: PUBLIC ERROR [reason: Auth.CallProblem, whichArg: Auth.WhichArg] = CODE;

OrphanConversation: PUBLIC ERROR = CODE;
  -- This error is raised by Refresh only.


-- Private errors --


BadLineClock: PRIVATE ERROR = CODE;
  -- Indicates that the line clock is not advancing.
  -- Should never happen unless your hardware is broken.
```

```
-- Identities --

MakeIdentity: PUBLIC PROC [
  myName: NSName.Name, password: NSString.String, z: UNCOUNTED ZONE,
  style: Auth.Flavor, dontCheck: BOOLEAN]
    RETURNS [identity: Auth.IdentityHandle] =
  BEGIN
    newIdentity: PrivateIdentityHandle;

    newIdentity _ z.NEW[AuthInternal.IdentityObject];
    newIdentity.style _ style;
    newIdentity.myName _ AuthInternal.MakeEmptyNSName[z];
  -- Assume: newIdentity.myName will be big enough to hold myName;
  --   no heap will be needed by CopyNameFields.
    NSName.CopyNameFields[
      z: NIL, source: myName, destination: newIdentity.myName];
    newIdentity.myPassword _ AuthInternal.CloneNSString[password, z],
    newIdentity.myStrongKey _ PrivateKey[Auth.PasswordStringToKey[password]];
    newIdentity.myHashedPassword _ Auth.HashSimplePassword[password];
    newIdentity.nameHasBeenResolved _ FALSE;
    newIdentity.conversationsInUse _ NIL;
    newIdentity.cachedConversations _ NIL;
    newIdentity.owningHeap _ z;
    Process.InitializeMonitor[@newIdentity.LOCK];
    identity _ PublicIdent[newIdentity];
    IF ~dontCheck THEN SelfAuthenticate[
      identity ! UNWIND => FreeIdentity[@identity, NIL]]; -- No heap needed.
  END;

MakeStrongIdentityUsingKey: PUBLIC PROC [
  myName: NSName.Name, myKey: Auth.Key, z: UNCOUNTED ZONE,
  dontCheck: BOOLEAN]
    RETURNS [identity: Auth.IdentityHandle] =
  BEGIN
    IF ~DESFace.CheckKeyParity[LOOPHOLE[LONG[@myKey]]]
      THEN ERROR CallError[badKey, notApplicable];
    identity _ MakeIdentity[myName, NSString.nullString, z, strong, TRUE];
    PrivateIdent[identity].myStrongKey _ PrivateKey[myKey];
    IF ~dontCheck THEN SelfAuthenticate[
      identity ! UNWIND => FreeIdentity[@identity, NIL]]; -- No heap needed.
  END;

FreeIdentity: PUBLIC PROC [
  identityPtr: LONG POINTER TO Auth.IdentityHandle, z: UNCOUNTED ZONE] =
  -- Note: z is no longer used.
  BEGIN
    CleanUpIdentity: ENTRY PROC [identity: Auth.IdentityHandle] =
      BEGIN
        ENABLE UNWIND => NULL;
        thisOne: PrivateConversationHandle;
        -- Clean up currently active conversations (making them orphans):
        -- (These conversations are in use; they should not be freed.)
        thisOne _ PrivateIdent[identity].conversationsInUse;
        WHILE thisOne # NIL DO
          nextOne: PrivateConversationHandle _ thisOne.next;
          thisOne.owner _ NIL;
          thisOne.next _ NIL;
          thisOne _ nextOne;
        ENDLOOP;

        -- Clean up cached conversations:
        -- (These conversations are not in use; they should be freed.)
        thisOne _ PrivateIdent[identity].cachedConversations;
        WHILE thisOne # NIL DO
          nextOne: PrivateConversationHandle _ thisOne.next;
          thisOne.owner _ NIL; -- Prevents monitor lock.
          InternalTerminate[LOOPHOLE[LONG[@thisOne]]];
          thisOne _ nextOne;
        ENDLOOP;

        AuthInternal.FreeNSName[
          @PrivateIdent[identity].myName, PrivateIdent[identity].owningHeap];
        AuthInternal.FreeNSString[
          @PrivateIdent[identity].myPassword, PrivateIdent[identity].owningHeap];
      END;

    IF identityPtr^ = NIL THEN RETURN;
    CleanUpIdentity[identityPtr^];
      -- There is a tiny race here; we free the monitor lock before we
      -- free the identity itself. This shouldn't matter; if the client
      -- is calling FreeIdentity, he'd better not be using it!
    PrivateIdent[identityPtr^].owningHeap.FREE[identityPtr];
      -- Smashes NIL into identityPtr^.
  END;


-- Conversations --


Initiate: PUBLIC PROC [
  identity: Auth.IdentityHandle, recipientsName: NSName.Name,
  recipientsHostNumber: System.HostNumber _ Auth.nullHostNumber,
  z: UNCOUNTED ZONE]
    RETURNS [conversation: Auth.ConversationHandle] =
  -- Note: z is no longer used.
  BEGIN
```

```
      SELECT AuthInternal.Style[identity] FROM
        simple = > ResolveNameIfAlias[identity];
        strong = > CheckForValidName[PrivateIdent[identity].myName, strong];
      ENDCASE = > ERROR;
      conversation __ InternalInitiate[identity, recipientsName, recipientsHostNumber];
    END;

  Terminate: PUBLIC PROC [
    conversationPtr: LONG POINTER TO Auth.ConversationHandle, z: UNCOUNTED ZONE] =
    BEGIN
      - - Note: z is no longer used.
      InternalTerminate[conversationPtr, cacheConversations];
    END;

  Refresh: PUBLIC PROC [conversation: Auth.ConversationHandle] =
    BEGIN
      MonitoredFetchStrongCredentials: ENTRY PROC [identity: Auth.IdentityHandle] =
        - - If successful, sets values of newCreds and newConversationKey.
        - - Assume: PrivateConv[conversation].owner = identity
        BEGIN
          ENABLE UNWIND = > NULL;
          [newCreds, newConversationKey] __
            Auth.FetchStrongCredentials[
              PrivateConv[conversation].owner.myName,
              PrivateConv[conversation].recipient,
              PublicKey[PrivateConv[conversation].owner.myStrongKey],
              PrivateConv[conversation].owningHeap];
        END;

      newConversationKey: Auth.Key __ Auth.nullKey;
      newCreds: Auth.Credentials __ Auth.nullCredentials;

      IF PrivateConv[conversation].owner = NIL THEN ERROR OrphanConversation;
      SELECT AuthInternal.Style[PublicIdent[PrivateConv[conversation].owner]] FROM
        simple = > RETURN; - - Noop
        strong = >
          MonitoredFetchStrongCredentials[PublicIdent[PrivateConv[conversation].owner]];
      ENDCASE = > ERROR;

      - - We fetched new credentials using the conversation's heap. We
      - - may now free the old conversation credentials and replace them
      - - with the one's we just fetched.
      AuthInternal.FreeCredentials[
        @PrivateConv[conversation].creds,
        PrivateConv[conversation].owningHeap];
      PrivateConv[conversation].creds __ newCreds;
      newCreds __ Auth.nullCredentials;
        - - We've saved these credentials, so forget 'em.
      PrivateConv[conversation].conversationKey __ PrivateKey[newConversationKey];
      PrivateConv[conversation].clearLastVerifier
        __ [System.GetGreenwichMeanTime[], GetRandomTicksForVerifier[]];
      PrivateConv[conversation].creationTime __ System.GetGreenwichMeanTime[];
    END;

  CheckOutCredentials: PUBLIC PROC [conversation: Auth.ConversationHandle]
    RETURNS [creds: Auth.Credentials] =
    BEGIN
      RETURN[PrivateConv[conversation].creds];
    END;

  CheckOutNextVerifier: PUBLIC PROC [
    conversation: Auth.ConversationHandle, recipientsHostNumber: System.HostNumber]
      RETURNS [verifier: Auth.Verifier] =
    BEGIN
      recipientsHostNumber __
        IF recipientsHostNumber # System.nullHostNumber
          THEN recipientsHostNumber
          ELSE PrivateConv[conversation].recipientsHostNumber;
      RETURN[
        ComputeNextVerifierForConversation[
          PrivateConv[conversation],
          recipientsHostNumber]
        ];
    END;

  CheckOutCredsAndNextVerifier: PUBLIC PROC [
    conversation: Auth.ConversationHandle,
    recipientsHostNumber: System.HostNumber]
    RETURNS [creds: Auth.Credentials, verifier: Auth.Verifier] =
    BEGIN
      recipientsHostNumber __
        IF recipientsHostNumber # System.nullHostNumber
          THEN recipientsHostNumber
          ELSE PrivateConv[conversation].recipientsHostNumber;
      RETURN[
        PrivateConv[conversation].creds,
        ComputeNextVerifierForConversation[
          PrivateConv[conversation],
          recipientsHostNumber]];
    END;


  - - Serialization - -


    - - NOTE: DescribeCredentials and DescribeVerifier are
    - -  exported by AuthProtocolImpl.
```

```
-- Exports to AuthSpecial and SpecialCHAuth --


-- AuthSpecial. --
MakeNullCHConversation: PUBLIC PROC [z: UNCOUNTED ZONE]
  RETURNS [nullCHConv: CH.ConversationHandle _ [NIL, NIL]] =
  BEGIN
    nullConv: PrivateConversationHandle _
      z.NEW[AuthInternal.ConversationObject _ []];
    nullConv.owningHeap _ z;
    nullCHConv.conversation _ PublicConv[nullConv];
  END;

-- SpecialCHAuth. --
MakeConversationFromCredsAndVerifier: PUBLIC ENTRY PROC [
  identity: Auth.IdentityHandle, creds: Auth.Credentials, verifier: Auth.Verifier,
  z: UNCOUNTED ZONE]
    RETURNS [conversation: Auth.ConversationHandle] =
  -- Note: z is no longer used.
  BEGIN
    ENABLE UNWIND => NULL;
    newConversation: PrivateConversationHandle;
    convKey: Auth.Key _ Auth.nullKey;
    badCreds: BOOLEAN;

    SELECT creds.flavor FROM
      simple => NULL;
      strong =>
        BEGIN
          -- Assume: PrivateIdent[identity].style # simple
          [, convKey, , , badCreds, , ] _
            AuthInternal.InternalExtractCredentialsDetails[
              PublicKey[PrivateIdent[identity].myStrongKey], creds, NIL, FALSE];
          IF badCreds THEN ERROR AuthenticationError[credentialsInvalid];
        END;
    ENDCASE => ERROR AuthenticationError[credentialsInvalid];

    newConversation _ conversationHeap.NEW[AuthInternal.ConversationObject];
    newConversation.recipient _ NIL;
    -- Assume: Noone will be interested in the recipient's name so why copy it?
    -- NOTE: This is an orphan conversation.
    newConversation.creds _ Auth.CopyCredentials[creds, conversationHeap];
    newConversation.lastVerifier _ Auth.nullVerifier; -- Filled in below.
    newConversation.conversationKey _ PrivateKey[convKey];
    newConversation.incrementVerifierByTicks _ TRUE;
    newConversation.clearLastVerifier _ [System.gmtEpoch, 0];
    newConversation.recipientsHostNumber _ System.nullHostNumber;
    newConversation.creationTime _ System.GetGreenwichMeanTime[];
    newConversation.owner _ NIL;
    newConversation.next _ NIL;
    newConversation.owningHeap _ conversationHeap;

    SELECT creds.flavor FROM
      simple =>
        newConversation.lastVerifier _
          AuthInternal.CloneVerifier[verifier, conversationHeap];
      strong =>
        BEGIN
          ENABLE UNWIND =>
            InternalTerminate[LOOPHOLE[LONG[@newConversation]]];
              -- No monitor problems because newConversation is an orphan.

          newConversation.clearLastVerifier _
            AuthInternal.UnpackStrongVerifier[
              verifier, PrivateKey[convKey], thisMachinesAddress.host
              ! Courier.Error => ERROR AuthenticationError[verifierInvalid]];
          newConversation.lastVerifier _
            DESCRIPTOR[
              Heap.MakeNode[conversationHeap, SIZE[AuthProtocol.StrongVerifier]],
              SIZE[AuthProtocol.StrongVerifier]];
        END;
    ENDCASE => ERROR AuthenticationError[credentialsInvalid];

    RETURN[PublicConv[newConversation]];
  END;


-- Private stuff --


CheckForValidName: PROC [name: NSName.Name, style: Auth.Flavor] =
  -- Raises an error (according to style) if the name is NOT ok.
  BEGIN
    IF AuthInternal.NilOrNullName[name] OR
      name.local.length > NSName.maxLocalLength OR
      name.domain.length > NSName.maxDomainLength OR
      name.org.length > NSName.maxOrgLength
        THEN ERROR CallError[
          (IF style = simple THEN simpleKeyDoesNotExist ELSE strongKeyDoesNotExist),
          notApplicable];
  END;

ComputeNextVerifierForConversation: PROC [
  conversation: PrivateConversationHandle,
  recipientsHostNumber: System.HostNumber]
```

```
    RETURNS [nextVerifier: Auth.Verifier] =
  -- This will compute a unique verifier later than last. If necessary,
  -- it will wait one second. If the timeStamp in the verifier is greater than
  -- the current time (i.e. in the future), you're out of luck. It has the
  -- side effect of modifying the lastVerifier and clearLastVerifier fields
  -- of the conversation.
  BEGIN
    SELECT TRUE FROM
      conversation.creds.flavor = simple => RETURN[conversation.lastVerifier];
      conversation.incrementVerifierByTicks => -- strong, strange case --
        BEGIN
          conversation.clearLastVerifier ←
            AuthInternal.IncrementVerifierTicks[conversation.clearLastVerifier];
          AuthInternal.PackStrongVerifier[
            from: @conversation.clearLastVerifier,
            recipientsHostNumber: recipientsHostNumber,
            destVerifier: conversation.lastVerifier,
            key: conversation.conversationKey];
        END;
    ENDCASE => -- strong, normal case --
      BEGIN
        last: AuthProtocol.StrongVerifier ← conversation.clearLastVerifier;
        timeOfLastVerifier, timeOfNextVerifier: LONG CARDINAL;
        oneSecond: Process.Ticks ← Process.SecondsToTicks[1];
        next: AuthProtocol.StrongVerifier ← [
          timeStamp: System.GetGreenwichMeanTime[],
          ticks: GetRandomTicksForVerifier[]];
        timeOfLastVerifier ← System.SecondsSinceEpoch[last.timeStamp];
        timeOfNextVerifier ← System.SecondsSinceEpoch[next.timeStamp];
        IF ~(timeOfNextVerifier > timeOfLastVerifier) THEN
          BEGIN
            IF timeOfNextVerifier < timeOfLastVerifier THEN ERROR BadLineClock;
              -- Time appears to be going backwards!!!
          -- If we get this far, timeOfNextVerifier = timeOfLastVerifier. Since
          -- we can't make duplicate verifiers, we've got to increment the ticks
          -- or, if we've run out of ticks in a LONG CARDINAL, then we've got
          -- to wait until the next second. (This should be VERY rare.)
            IF last.ticks # LAST[LONG CARDINAL]
              THEN next.ticks ← last.ticks + 1
              ELSE
                BEGIN
                  Process.Pause[oneSecond];
                  timesHaveHadToWaitForNextVerifier ←
                    timesHaveHadToWaitForNextVerifier + 1;
                  next ← [
                    timeStamp: System.GetGreenwichMeanTime[],
                    ticks: GetRandomTicksForVerifier[]];
                  IF System.SecondsSinceEpoch[next.timeStamp] <=
                    timeOfLastVerifier
                      THEN ERROR BadLineClock;
                        -- If, after pausing for a second, the clock
                        -- hasn't advanced, something must be wrong
                        -- with the clock...
                END;
          END;
        conversation.clearLastVerifier ← next;
        AuthInternal.PackStrongVerifier[
          from: @conversation.clearLastVerifier,
          recipientsHostNumber: recipientsHostNumber,
          destVerifier: conversation.lastVerifier,
          key: conversation.conversationKey];
      END;
      RETURN[conversation.lastVerifier];
  END;

GetConversationFromCache: INTERNAL PROC [
  id: PrivateIdentityHandle, recipient: NSName.Name]
  RETURNS [conv: PrivateConversationHandle ← NIL] =
  BEGIN
    FindConversationFor: PROC [name: NSName.Name]
      RETURNS [convFound: PrivateConversationHandle] =
      BEGIN
        previous: LONG POINTER TO PrivateConversationHandle;
        IF AuthInternal.NilOrNullName[name] THEN RETURN[NIL];
      -- Find the conversation (if any) preceding the one
      --  we'd like to use.
        previous ← @id.cachedConversations;
        WHILE previous^ # NIL DO -- For entire list, do:
          IF NSStringExtras.EquivalentNames[previous^.recipient, name]
            THEN EXIT; -- Found one!
          previous ← @previous^.next;
        ENDLOOP;
        IF previous^ = NIL THEN RETURN[NIL]; -- Didn't find one.
      -- Found one, splice it out:
        convFound ← previous^;
        previous^ ← convFound.next;
      -- Put it on the front of the list of conversations in use:
        convFound.next ← id.conversationsInUse;
        id.conversationsInUse ← convFound;
        RETURN[convFound];
      END;
    WeedOutOldConversations: PROC =
      BEGIN
        previous: LONG POINTER TO PrivateConversationHandle ←
          @id.cachedConversations;
        convToRemove: PrivateConversationHandle ← NIL;
        -- Seek out and destroy all old conversations:
```

```
            WHILE previous^ # NIL DO -- For entire list, do
              SELECT OldConversation[previous^] FROM
                FALSE =>
                  BEGIN
                    previous _ @previous^.next;
                    LOOP; -- This is a young entry; pass over it.
                  END;
                TRUE =>
                  BEGIN
                    convToRemove _ previous^;
                    -- Splice this conversation out of list:
                    previous^ _ previous^.next;
                    -- Kill it:
                    convToRemove.owner _ NIL;
                    InternalTerminate[LOOPHOLE[LONG[@convToRemove]]];
                  END;
              ENDCASE;
            ENDLOOP;
          END;

      WeedOutOldConversations[];
      RETURN[FindConversationFor[recipient]];
    END;

  GetRandomTicksForVerifier: PROC RETURNS [ticks: LONG CARDINAL] =
    BEGIN
      ticks _ System.GetClockPulses[];
      IF ticks > 10000 THEN ticks _ ticks - 10000;
        -- This makes sure that ticks is within [0..MAX[LONG CARDINAL] - 10000].
        -- That way, we can make up to 10000 verifiers within a given second
        -- by incrementing the ticks field.
    END;

  InternalInitiate: PRIVATE ENTRY PROC [
    identity: Auth.IdentityHandle, recipientsName: NSName.Name,
    recipientsHostNumber: System.HostNumber _ Auth.nullHostNumber]
      RETURNS [conversation: Auth.ConversationHandle] =
    BEGIN
      ENABLE UNWIND => NULL;
      newConversation: PrivateConversationHandle;
      newCreds: Auth.Credentials;
      newConversationKey: Auth.Key _ Auth.nullKey;

      newConversation _
        GetConversationFromCache[PrivateIdent[identity], recipientsName];
      IF newConversation # NIL THEN
        BEGIN -- If there was a conversation in the cache, use it!
          newConversation.recipientsHostNumber _ recipientsHostNumber;
          RETURN[PublicConv[newConversation]];
        END;

      -- No credentials in the cache, so manufacture or fetch some:
      SELECT PrivateIdent[identity].style FROM
        simple =>
          newCreds _
            AuthInternal.MakeSimpleCredentials[
              PrivateIdent[identity].myName, conversationHeap];
        strong =>
          [newCreds, newConversationKey] _
            Auth.FetchStrongCredentials[
              PrivateIdent[identity].myName, recipientsName,
              PublicKey[PrivateIdent[identity].myStrongKey], conversationHeap];
      ENDCASE => ERROR;

      newConversation _ conversationHeap.NEW[AuthInternal.ConversationObject];
      newConversation.recipient _
        AuthInternal.CloneNSName[recipientsName, conversationHeap];
      newConversation.creds _ newCreds;
      newConversation.lastVerifier _ Auth.nullVerifier; -- Filled in below.
      newConversation.conversationKey _ PrivateKey[newConversationKey];
      newConversation.incrementVerifierByTicks _ FALSE;
      newConversation.clearLastVerifier _
        [System.GetGreenwichMeanTime[], GetRandomTicksForVerifier[]];
      newConversation.recipientsHostNumber _ recipientsHostNumber;
      newConversation.creationTime _ System.GetGreenwichMeanTime[];
      newConversation.owner _ NIL;
      newConversation.next _ NIL;
      newConversation.owningHeap _ conversationHeap;

      SELECT TRUE FROM
        newConversation.creds = Auth.nullCredentials =>
          newConversation.lastVerifier _ Auth.nullVerifier;
        PrivateIdent[identity].style = simple =>
          newConversation.lastVerifier _
            AuthInternal.MakeVerifierFromHashedPassword[
              PrivateIdent[identity].myHashedPassword, conversationHeap];
        PrivateIdent[identity].style = strong =>
          newConversation.lastVerifier _
            DESCRIPTOR[
              Heap.MakeNode[conversationHeap, SIZE[AuthProtocol.StrongVerifier]],
              SIZE[AuthProtocol.StrongVerifier]];
      ENDCASE => ERROR;

    -- Put on front of identity's list of conversations in use:
      newConversation.owner _ PrivateIdent[identity];
      newConversation.next _ PrivateIdent[identity].conversationsInUse;
      PrivateIdent[identity].conversationsInUse _ newConversation;
```

```
    RETURN[PublicConv[newConversation]];
  END;

InternalTerminate: PROC [
  conversationPtr: LONG POINTER TO Auth.ConversationHandle,
  okToCache: BOOLEAN _ FALSE] =
  BEGIN
    RemoveConvFromIdentity: ENTRY PROC [identity: Auth.IdentityHandle] =
      -- Removes conversationPtr^ from identity.conversationsInUse.
      BEGIN
        ENABLE UNWIND => NULL;
        -- Assume: identity = PrivateConv[conversationPtr^].owner
        -- Assume: identity # NIL
        previous: LONG POINTER TO PrivateConversationHandle _
          @PrivateIdent[identity].conversationsInUse;
      -- Find previous conversation in chain:
        DO
          IF previous^ = PrivateConv[conversationPtr^] THEN EXIT;
          IF previous^ = NIL THEN ERROR;
            -- A conversation has an owner that doesn't know about
            -- the conversation. This "can't" happen (if it does there
            -- is a bug in this code somewhere.
          previous _ @previous^.next;
        ENDLOOP;
      -- Splice out this conversation:
        previous^ _ PrivateConv[conversationPtr^].next;
        PrivateConv[conversationPtr^].next _ NIL;
      END;
    OfferConvToIdentityCache: ENTRY PROC [
      identity: Auth.IdentityHandle, conversation: PrivateConversationHandle]
        RETURNS [accepted: BOOLEAN _ FALSE] =
      -- This operation may add conversation to identity.cachedConversations.
      -- If it does, accepted will be set to TRUE.
      BEGIN
        ENABLE UNWIND => NULL;

        IF ~okToCache
          OR conversation.incrementVerifierByTicks
          OR OldConversation[conversation]
          OR AuthInternal.NilOrNullName[conversation.recipient]
            THEN RETURN[accepted: FALSE];
        conversation.next _ PrivateIdent[identity].cachedConversations;
        PrivateIdent[identity].cachedConversations _ conversation;
        RETURN[accepted: TRUE];
      END;

    addedToCache: BOOLEAN;
    IF conversationPtr^ = NIL THEN RETURN; -- Nothing to do.
    IF PrivateConv[conversationPtr^].owner # NIL THEN
      BEGIN
        RemoveConvFromIdentity[PublicIdent[PrivateConv[conversationPtr^].owner]];
        addedToCache _
          OfferConvToIdentityCache[
            PublicIdent[PrivateConv[conversationPtr^].owner],
            PrivateConv[conversationPtr^]];
        IF addedToCache
          THEN {conversationPtr^ _ NIL; RETURN};
            -- Do NOT free the conversation; do set it to NIL.
      END;

    -- Free the conversation:
    AuthInternal.FreeNSName[
      @PrivateConv[conversationPtr^].recipient,
      PrivateConv[conversationPtr^].owningHeap];
    AuthInternal.FreeCredentials[
      @PrivateConv[conversationPtr^].creds,
      PrivateConv[conversationPtr^].owningHeap];
    AuthInternal.FreeVerifier[
      @PrivateConv[conversationPtr^].lastVerifier,
      PrivateConv[conversationPtr^].owningHeap];
    PrivateConv[conversationPtr^].owningHeap.FREE[conversationPtr];
      -- Smashes NIL into conversationPtr^.
  END;

IsWellKnownName: PROC [name: NSName.Name]
  RETURNS [isWellKnown: BOOLEAN] =
  BEGIN
    RETURN[
      AuthInternal.EquivalentNames[name, SharedKeys.asName] OR
      AuthInternal.EquivalentNames[name, SharedKeys.chsName] OR
      AuthInternal.EquivalentNames[name, SharedKeys.msName]
      ];
  END;

MakeSimpleCHOrphanConversation: PROC [
  for: NSName.Name, hashedPassword: Auth.HashedPassword]
  RETURNS [chConv: CH.ConversationHandle] =
  BEGIN
    chConv _ MakeNullCHConversation[asStubHeap];
    PrivateConv[chConv.conversation].creds _
      AuthInternal.MakeSimpleCredentials[for, asStubHeap];
    PrivateConv[chConv.conversation].lastVerifier _
      AuthInternal.MakeVerifierFromHashedPassword[hashedPassword, asStubHeap];
  END;

OldConversation: PROC [conv: PrivateConversationHandle]
  RETURNS [isOld: BOOLEAN] =
```

```
    BEGIN
      now: LONG CARDINAL __
        System.SecondsSinceEpoch[System.GetGreenwichMeanTime[]];
      then: LONG CARDINAL __
        System.SecondsSinceEpoch[conv.creationTime];
      IF then > now
        THEN isOld __ TRUE
          -- Clock is running backward?? Consider the conversation "old".
        ELSE isOld __ ((now - then) > conversationCacheTimeout);
    END;

  ResolveNameIfAlias: ENTRY PROC [identity: Auth.IdentityHandle] =
    -- Assume: This operation gets called only with a simple identity.
    BEGIN
      ENABLE {
        AuthenticationError = > ERROR CallError[badKey, initiator];
        UNWIND = > NULL;
        };

      chConversation: CH.ConversationHandle __ [NIL, NIL];
      distingName: NSName.Name __ NIL;
      rc: CH.ReturnCode;

      IF PrivateIdent[identity].nameHasBeenResolved THEN RETURN;
      CheckForValidName[PrivateIdent[identity].myName, PrivateIdent[identity].style];
      IF IsWellKnownName[PrivateIdent[identity].myName] THEN
        BEGIN
          PrivateIdent[identity].nameHasBeenResolved __ TRUE;
          RETURN;
        END;

      chConversation __ MakeSimpleCHOrphanConversation[
        PrivateIdent[identity].myName, PrivateIdent[identity].myHashedPassword];
      distingName __ AuthInternal.MakeEmptyNSName[asStubHeap];
      BEGIN
        ENABLE UNWIND = > {
          CH.FreeConversationHandle[@chConversation, NIL];
          AuthInternal.FreeNSName[@distingName, asStubHeap];
          };

        rc __ CH.LookupDistinguishedName[
          chConversation, PrivateIdent[identity].myName, distingName];
        SELECT rc.code FROM
          done = >
            BEGIN
            -- Assume: PrivateIdent[identity].myName is big enough;
            -- no heap will be needed by CopyNameFields.
              NSName.CopyNameFields[
                z: NIL, source: distingName,
                destination: PrivateIdent[identity].myName];
              PrivateIdent[identity].nameHasBeenResolved __ TRUE;
            END;
          noSuchOrg, noSuchDomain, noSuchLocal,
          illegalOrgName, illegalDomainName, illegalLocalName = >
            ERROR CallError[simpleKeyDoesNotExist, initiator];
          rejectedTooBusy = >
            ERROR CallError[tooBusy, notApplicable];
          allDown = >
            ERROR CallError[keysUnavailable, initiator];
          credentialsInvalid = >
            ERROR CallError[badKey, initiator];
        ENDCASE = > ERROR CallError[simpleKeyDoesNotExist, initiator];
        AuthInternal.FreeNSName[@distingName, asStubHeap];
      END;
      CH.FreeConversationHandle[@chConversation, NIL];
    END;

  SelfAuthenticate: PROC [identity: Auth.IdentityHandle] =
    -- This operation will also resolve the name in the identity handle.
    -- Assume: This procedure is only called from an identity creation procedure!
    BEGIN
      creds: Auth.Credentials __ Auth.nullCredentials;
      verifier: Auth.Verifier __ Auth.nullVerifier;
      conv: Auth.ConversationHandle __ NIL;

      SELECT PrivateIdent[identity].style FROM
        simple = >
          ResolveNameIfAlias[identity];
        strong = >
          CheckForValidName[PrivateIdent[identity].myName, strong];
          -- Note: Since the strong credentials produced by the AS always contain
          -- a distinguished name, we save a Clearinghouse operation by extracting
          -- the disinguished name from the credentials that we are using to check
          -- this identity.
      ENDCASE = > ERROR;

      SELECT PrivateIdent[identity].style FROM
        simple = >
          BEGIN
            ENABLE UNWIND = > InternalTerminate[@conv];

            conv __ InternalInitiate[identity, PrivateIdent[identity].myName];
            [creds, verifier] __Auth.CheckOutCredsAndNextVerifier[conv];
            IF ~Auth.CheckSimpleCredentials[creds, verifier].ok
              THEN ERROR CallError[badKey, notApplicable];
            InternalTerminate[@conv];
          END;
```

```
        strong = >
          BEGIN
            ENABLE UNWIND = > InternalTerminate[@conv];

            distingName: NSName.Name _ NIL;
            conv _ InternalInitiate[identity, PrivateIdent[identity].myName];
            [creds, verifier] _
              CheckOutCredsAndNextVerifier[conv, thisMachinesAddress.host];
          - - To get the distinguished name for strong credentials, do an
          - - Authenticate, which returns the distinguished name (which was
          - - resolved by the server). Replace the identity.myName with this name.
            [distingName, , ] _ AuthInternal.InternalAuthenticate[
              PublicKey[PrivateIdent[identity].myStrongKey], creds, verifier,
              asStubHeap, thisMachinesAddress.host,
              FALSE, FALSE, TRUE, FALSE
              ! AuthenticationError = > ERROR CallError[badKey, notApplicable]];

          - - Assume: PrivateIdent[identity].myName is big enough;
          - -  no heap will be needed by CopyNameFields.
            NSName.CopyNameFields[
              z: NIL, source: distingName, destination: PrivateIdent[identity].myName];
            AuthInternal.FreeNSName[@distingName, asStubHeap];
            PrivateIdent[identity].nameHasBeenResolved _ TRUE;
            InternalTerminate[@conv];
          END;
      ENDCASE = > ERROR;
    END;

- - Public/Private pointer conversions - -


  - - Note: This is a hack to avoid exporting public types.
  - - (Because exported type clashes make it impossible for two
  - - different implementations which export the same type to co - exist.)

PrivateIdentityHandle: TYPE = LONG POINTER TO AuthInternal.IdentityObject;
PrivateConversationHandle: TYPE = LONG POINTER TO AuthInternal.ConversationObject;

PrivateConv: PROC [conversation: Auth.ConversationHandle]
  RETURNS [PrivateConversationHandle] = INLINE
    {RETURN[LOOPHOLE[conversation]]};

PublicConv: PROC [privateConversation: PrivateConversationHandle]
  RETURNS [Auth.ConversationHandle] = INLINE
    {RETURN[LOOPHOLE[privateConversation]]};

PrivateIdent: PROC [identity: Auth.IdentityHandle]
  RETURNS [PrivateIdentityHandle] = INLINE
    {RETURN[LOOPHOLE[identity]]};

PublicIdent: PROC [privateIdentity: PrivateIdentityHandle]
  RETURNS [Auth.IdentityHandle] = INLINE
    {RETURN[LOOPHOLE[privateIdentity]]};

PrivateKey: PROC [key: Auth.Key]
  RETURNS [DESFace.Key] = INLINE
    {RETURN[LOOPHOLE[key]]};

PublicKey: PROC [privateKey: DESFace.Key]
  RETURNS [Auth.Key] = INLINE
    {RETURN[LOOPHOLE[privateKey]]};


END.
```

```
-- AuthCImpl.mesa
-- JMaloney, 11-Jul-83 12:24:49.

-- Last modified: JMaloney, 3-Jul-84 12:00:10.


DIRECTORY
  Auth USING [
    AuthenticationError, CallError, CallProblem,
    CheckOutCredsAndNextVerifier, ConversationHandle,
    Credentials, Flavor, HashedPassword, HashSimplePassword,
    IdentityHandle, Initiate, Key, nullCredentials,
    nullHashedPassword, nullKey, nullVerifier,
    PasswordStringToKey, Terminate, Verifier, WhichArg],
  AuthInternal USING [
    asStubHeap, BadCredentialsPackage, CloneNSString,
    ConversationObject, ExtractStuffFromCredentialsPackage, FreeNSString,
    FreeSimpleCredentials, IdentityObject,
    Style, UnpackSimpleCredentials],
  AuthOps USING [
    ChangeSimpleKey, ChangeStrongKey,
    CreateSimpleKey, CreateStrongKey,
    DeleteSimpleKey, DeleteStrongKey,
    CheckSimpleCredentials, GetStrongCredentials],
  AuthProtocol USING [
    CredentialsPackage, DescribeCredentialsPackage, SimpleCredentials],
  AuthServerCache USING [ASAddress, NextPlease, Refill],
  Courier USING [Description, Error, Free],
  DESFace USING [Block, CheckKeyParity, EncryptBlock, Key, nullKey],
  NSName USING [Name],
  NSString USING [String],
  Process USING [MsecToTicks, Pause, Ticks],
  SharedKeys USING [asName],
  System USING [GetClockPulses, NetworkAddress, nullNetworkAddress];

AuthCImpl: MONITOR
  LOCKS LOOPHOLE[identity, PrivateIdentityHandle]^ USING identity: Auth.IdentityHandle
  IMPORTS
    Auth, AuthInternal, AuthProtocol, AuthOps, AuthServerCache, Courier,
    DESFace, Process, SharedKeys, System
  EXPORTS Auth
  SHARES Auth =

BEGIN


-- Globals and constants --


cacheNeverFilled: BOOLEAN _ TRUE;

pauseForInitialCacheFill: Process.Ticks _ Process.MsecToTicks[15000];
  -- fifteen seconds

timesToRetry: CARDINAL _ 1;
  -- The number of times to grab a new address out of the cache and re-try
  -- an operation which must talk to an AS.


-- Password/Key administration operations --


ChangeMyPasswords: PUBLIC PROC [
  identity: Auth.IdentityHandle, newPassword: NSString.String, z: UNCOUNTED ZONE,
  changeStrong, changeSimple: BOOLEAN] =
  BEGIN
    ComputeOldKeyValues: ENTRY PROC [identity: Auth.IdentityHandle]
      RETURNS [oldStrongValue: Auth.Key, oldSimpleValue: Auth.HashedPassword] =
      -- Computes oldStrongValue and oldSimpleValue from password
      -- in identity handle.
      BEGIN
        ENABLE UNWIND => NULL;
        oldStrongValue _
          Auth.PasswordStringToKey[PrivateIdent[identity].myPassword];
        oldSimpleValue _
          Auth.HashSimplePassword[PrivateIdent[identity].myPassword];
      END;
    ChangeIdentityInfo: ENTRY PROC [identity: Auth.IdentityHandle] =
      -- Fix up the password info in the identity.
      BEGIN
        ENABLE UNWIND => NULL;
        AuthInternal.FreeNSString[
          @PrivateIdent[identity].myPassword, PrivateIdent[identity].owningHeap];
        PrivateIdent[identity].myPassword _
          AuthInternal.CloneNSString[newPassword, PrivateIdent[identity].owningHeap];
        IF changeStrong
          THEN PrivateIdent[identity].myStrongKey _ PrivateKey[newStrongKey];
        IF changeSimple
          THEN PrivateIdent[identity].myHashedPassword _ newHashedPassword;
      END;

    newStrongKey: Auth.Key _ Auth.PasswordStringToKey[newPassword];
    newHashedPassword: Auth.HashedPassword _
      Auth.HashSimplePassword[newPassword];
    IF changeStrong THEN ChangeStrongKey[identity, newStrongKey];
    IF changeSimple THEN
      BEGIN
```

```
            ENABLE UNWIND => {
              -- Undo what we have done so far...
              oldStrongKey: Auth.Key _ Auth.nullKey;
              oldSimpleKey: Auth.HashedPassword _ Auth.nullHashedPassword;
              [oldStrongKey, oldSimpleKey] _ ComputeOldKeyValues[identity];
              IF changeStrong THEN
                ChangeStrongKey[
                  identity, oldStrongKey
                  ! Auth.AuthenticationError, Auth.CallError => CONTINUE];
              ChangeSimpleKey[
                identity, oldSimpleKey
                ! Auth.AuthenticationError, Auth.CallError => CONTINUE];
              };

            ChangeSimpleKey[identity, newHashedPassword];
          END;

        -- If we got this far it worked; we can fix up identityObject.
        ChangeIdentityInfo[identity];
      END;

    CreateStrongKey: PUBLIC PROC [
      identity: Auth.IdentityHandle, name: NSName.Name, newStrongKey: Auth.Key] =
      BEGIN
        DoIt: PROC [
          creds: Auth.Credentials, verifier: Auth.Verifier, conv: Auth.ConversationHandle,
          asAddressPtr: LONG POINTER TO System.NetworkAddress] =
            BEGIN
              encryptedInitialKey: DESFace.Block;
              DESFace.EncryptBlock[
                key: PrivateConv[conv].conversationKey,
                from: LOOPHOLE[LONG[@newStrongKey]], to: @encryptedInitialKey];
              AuthOps.CreateStrongKey[
                creds, verifier, name, encryptedInitialKey, asAddressPtr];
            END;

        SELECT AuthInternal.Style[identity] FROM
          simple => ERROR Auth.AuthenticationError[inappropriateCredentials];
          strong => ConverseWithASAndCallMe[identity, name, DoIt];
        ENDCASE => ERROR;
      END;

    ChangeStrongKey: PUBLIC PROC [
      identity: Auth.IdentityHandle, newStrongKey: Auth.Key] =
      BEGIN
        ChangeIdentityInfo: ENTRY PROC [identity: Auth.IdentityHandle] =
          BEGIN
            ENABLE UNWIND => NULL;
            AuthInternal.FreeNSString[
              @PrivateIdent[identity].myPassword, PrivateIdent[identity].owningHeap];
            -- Leave password a nullString because it is no longer related to
            -- the "truth" in identity.myStrongKey.
            PrivateIdent[identity].myStrongKey _ PrivateKey[newStrongKey];
          END;
        DoIt: PROC [
          creds: Auth.Credentials, verifier: Auth.Verifier, conv: Auth.ConversationHandle,
          asAddressPtr: LONG POINTER TO System.NetworkAddress] =
            BEGIN
              encryptedNewKey: DESFace.Block;
              DESFace.EncryptBlock[
                key: PrivateConv[conv].conversationKey,
                from: LOOPHOLE[LONG[@newStrongKey]], to: @encryptedNewKey];
              AuthOps.ChangeStrongKey[creds, verifier, encryptedNewKey, asAddressPtr];
            END;

        SELECT AuthInternal.Style[identity] FROM
          simple => ERROR Auth.AuthenticationError[inappropriateCredentials];
          strong =>
            ConverseWithASAndCallMe[identity, PrivateIdent[identity].myName, DoIt];
        ENDCASE => ERROR;

        -- If we got this far it worked; we can fix up identityObject.
        ChangeIdentityInfo[identity];
      END;

    DeleteStrongKey: PUBLIC PROC [identity: Auth.IdentityHandle, name: NSName.Name] =
      BEGIN
        DoIt: PROC [
          creds: Auth.Credentials, verifier: Auth.Verifier, conv: Auth.ConversationHandle,
          asAddressPtr: LONG POINTER TO System.NetworkAddress] =
            BEGIN
              AuthOps.DeleteStrongKey[creds, verifier, name, asAddressPtr];
            END;

        SELECT AuthInternal.Style[identity] FROM
          simple => ERROR Auth.AuthenticationError[inappropriateCredentials];
          strong => ConverseWithASAndCallMe[identity, name, DoIt];
        ENDCASE => ERROR;
      END;

    CreateSimpleKey: PUBLIC PROC [
      identity: Auth.IdentityHandle, name: NSName.Name,
      newSimpleKey: Auth.HashedPassword] =
      BEGIN
        DoIt: PROC [
          creds: Auth.Credentials, verifier: Auth.Verifier, conv: Auth.ConversationHandle,
          asAddressPtr: LONG POINTER TO System.NetworkAddress] =
```

```
            BEGIN
              AuthOps.CreateSimpleKey[
                creds, verifier, name, newSimpleKey, asAddressPtr];
            END;

        ConverseWithASAndCallMe[identity, name, DoIt];
      END;

    ChangeSimpleKey: PUBLIC PROC [
      identity: Auth.IdentityHandle, newSimpleKey: Auth.HashedPassword] =
      BEGIN
        ChangeIdentityInfo: ENTRY PROC [identity: Auth.IdentityHandle] =
          BEGIN
            ENABLE UNWIND => NULL;
            AuthInternal.FreeNSString[
              @PrivateIdent[identity].myPassword, PrivateIdent[identity].owningHeap];
            -- Leave password a nullString because it is no longer related to
            -- the "truth" in identity.myHashedPassword.
            PrivateIdent[identity].myHashedPassword _ newSimpleKey,
          END;
        DoIt: PROC [
          creds: Auth.Credentials, verifier: Auth.Verifier, conv: Auth.ConversationHandle,
          asAddressPtr: LONG POINTER TO System.NetworkAddress] =
            BEGIN
              AuthOps.ChangeSimpleKey[creds, verifier, newSimpleKey, asAddressPtr];
            END;

        ConverseWithASAndCallMe[identity, PrivateIdent[identity].myName, DoIt];

        -- If we got this far it worked; we can fix up identityObject.
        ChangeIdentityInfo[identity];
      END;

    DeleteSimpleKey: PUBLIC PROC [identity: Auth.IdentityHandle, name: NSName.Name] =
      BEGIN
        DoIt: PROC [
          creds: Auth.Credentials, verifier: Auth.Verifier, conv: Auth.ConversationHandle,
          asAddressPtr: LONG POINTER TO System.NetworkAddress] =
            BEGIN
              AuthOps.DeleteSimpleKey[creds, verifier, name, asAddressPtr];
            END;

        ConverseWithASAndCallMe[identity, name, DoIt];
      END;


    -- Credentials operations which call the server --


    CheckSimpleCredentials: PUBLIC PROC [
      creds: Auth.Credentials, verifier: Auth.Verifier]
        RETURNS [ok: BOOLEAN _ FALSE] =
      BEGIN
        DoIt: PROC [asAddressPtr: LONG POINTER TO System.NetworkAddress] =
          BEGIN
            ok _ AuthOps.CheckSimpleCredentials[creds, verifier, asAddressPtr];
          END;

        IF creds.flavor # simple
          THEN ERROR Auth.AuthenticationError[credentialsInvalid];

        BEGIN
          simpleCreds: AuthProtocol.SimpleCredentials _
            AuthInternal.UnpackSimpleCredentials[
              creds, AuthInternal.asStubHeap
              ! Courier.Error => ERROR Auth.AuthenticationError[credentialsInvalid]];
          -- The following calls the AS and sets ok:
          ContactASAndCallMe[
            DoIt, simpleCreds.initiator
            ! UNWIND =>
              AuthInternal.FreeSimpleCredentials[@simpleCreds, AuthInternal.asStubHeap]];
          AuthInternal.FreeSimpleCredentials[@simpleCreds, AuthInternal.asStubHeap];
        END;
      END;

    FetchStrongCredentials: PUBLIC PROC [
      initiator, recipient: NSName.Name,
      initiatorsStrongKey: Auth.Key, z: UNCOUNTED ZONE]
        RETURNS [creds: Auth.Credentials, conversationKey: Auth.Key] =
      BEGIN
        DoIt: PROC [asAddressPtr: LONG POINTER TO System.NetworkAddress] =
          BEGIN
            credentialsPackage _
              AuthOps.GetStrongCredentials[initiator, recipient, nonce, asAddressPtr];
          END;

        convKey: DESFace.Key _ DESFace.nullKey;
        credentialsPackage: AuthProtocol.CredentialsPackage;
        nonce: LONG CARDINAL _ System.GetClockPulses[];
        SELECT TRUE FROM
          initiator = NIL => ERROR Auth.CallError[strongKeyDoesNotExist, initiator];
          recipient = NIL => ERROR Auth.CallError[strongKeyDoesNotExist, recipient];
        ENDCASE;
        ContactASAndCallMe[DoIt, initiator]; -- fills in credentialsPackage
        BEGIN
          ENABLE UNWIND => Courier.Free[
            [@credentialsPackage, AuthProtocol.DescribeCredentialsPackage],
```

```
            AuthInternal.asStubHeap];

        [creds, convKey] ←
          AuthInternal.ExtractStuffFromCredentialsPackage[
            credentialsPackage, PrivateKey[initiatorsStrongKey],
            recipient, nonce, z
            ! AuthInternal.BadCredentialsPackage = >
              ERROR Auth.CallError[badKey, notApplicable]];
        IF ~DESFace.CheckKeyParity[@convKey]
          THEN ERROR Auth.CallError[badKey, notApplicable];
      END;
      Courier.Free[
        [@credentialsPackage, AuthProtocol.DescribeCredentialsPackage],
        AuthInternal.asStubHeap];
      RETURN[creds, PublicKey[convKey]];
    END;

- - Private stuff - -

  ConverseWithASAndCallMe: PROC [
    identity: Auth.IdentityHandle, nameHint: NSName.Name,
    procToCall: PROC [
      Auth.Credentials, Auth.Verifier, Auth.ConversationHandle,
      LONG POINTER TO System.NetworkAddress]] =
    BEGIN
      retrys: CARDINAL ← 0;
      asAddress: System.NetworkAddress ← System.nullNetworkAddress;
      creds: Auth.Credentials ← Auth.nullCredentials;
      verifier: Auth.Verifier ← Auth.nullVerifier;
      conv: Auth.ConversationHandle;

      conv ← Auth.Initiate[identity, SharedKeys.asName, , AuthInternal.asStubHeap];
      asAddress ← AuthServerCache.ASAddress[nameHint];
      BEGIN
        ENABLE UNWIND = > Auth.Terminate[@conv, AuthInternal.asStubHeap];

        DO
          [creds, verifier] ← Auth.CheckOutCredsAndNextVerifier[conv, asAddress.host];
          procToCall[
            creds, verifier, conv, @asAddress
            ! Auth.CallError = >
              IF (reason = cannotReachAS OR reason = tooBusy) AND
                retrys < timesToRetry
                  THEN
                    BEGIN
                      addressesLeftInCache: CARDINAL ←
                        AuthServerCache.NextPlease[asAddress];
                      AuthServerCache.Refill[];
                      asAddress ← AuthServerCache.ASAddress[];
                      retrys ← retrys + 1;
                      LOOP;
                    END;
             Auth.AuthenticationError = >
               - - The reason must be credentialsInvalid; this is highly unusual!
               {ERROR Auth.CallError[badKey, notApplicable]};
            ];
          EXIT; - - Successful call
        ENDLOOP;
      END;
      Auth.Terminate[@conv, AuthInternal.asStubHeap];
    END;

  ContactASAndCallMe: PROC [
    procToCall: PROC [LONG POINTER TO System.NetworkAddress],
    nameHint: NSName.Name] =
    BEGIN
      retrys: CARDINAL ← 0;
      asAddress: System.NetworkAddress ← AuthServerCache.ASAddress[nameHint];

      IF cacheNeverFilled THEN
        BEGIN
          AuthServerCache.Refill[];
          cacheNeverFilled ← FALSE;
          IF asAddress = System.nullNetworkAddress THEN
            BEGIN
              Process.Pause[pauseForInitialCacheFill]; - - Wait only if we have to.
              asAddress ← AuthServerCache.ASAddress[];
            END;
        END;

      DO
        procToCall[
          @asAddress
          ! Auth.CallError = >
              IF (reason = cannotReachAS OR reason = tooBusy) AND
                retrys < timesToRetry
                  THEN
                    BEGIN
                      addressesLeftInCache: CARDINAL ←
                        AuthServerCache.NextPlease[asAddress];
                      AuthServerCache.Refill[];
                      asAddress ← AuthServerCache.ASAddress[];
                      retrys ← retrys + 1;
                      LOOP;
                    END;
```

```
            ];
          EXIT; -- Successful call
        ENDLOOP;
    END;


-- -- Public/Private pointer conversions -- -


    -- Note: This is a hack to avoid exporting public types.
    -- (Because exported type clashes make it impossible for two
    -- different implementations which export the same type to co-exist.)

PrivateIdentityHandle: TYPE = LONG POINTER TO AuthInternal.IdentityObject;
PrivateConversationHandle: TYPE = LONG POINTER TO AuthInternal.ConversationObject;

PrivateConv: PROC [conversation: Auth.ConversationHandle]
    RETURNS [PrivateConversationHandle] = INLINE
      {RETURN[LOOPHOLE[conversation]]};

PrivateIdent: PROC [identity: Auth.IdentityHandle]
    RETURNS [PrivateIdentityHandle] = INLINE
      {RETURN[LOOPHOLE[identity]]};

PrivateKey: PROC [key: Auth.Key]
    RETURNS [DESFace.Key] = INLINE
      {RETURN[LOOPHOLE[key]]};

PublicKey: PROC [privateKey: DESFace.Key]
    RETURNS [Auth.Key] = INLINE
      {RETURN[LOOPHOLE[privateKey]]};


END.
```

```
-- AuthAImpl.mesa
-- JMaloney, 11-Jul-83 12:24:49.

-- Last modified: JMaloney, 20-Jun-84 11:16:36.


DIRECTORY
  Auth USING [
    AuthenticationProblem, CallProblem,
    CheckOutCredsAndNextVerifier, CheckSimpleCredentials,
    ConversationHandle, CopyCredentials, Credentials, FetchStrongCredentials,
    Flavor, HashedPassword, HashSimplePassword, IdentityHandle, Key,
    nullCredentials, nullHostNumber, nullKey,
    nullVerifier, PasswordStringToKey, Verifier, WhichArg],
  AuthInternal USING [
    asStubHeap, CloneNSName, CloneNSString, CloneVerifier,
    ConversationObject, EquivalentNames, FreeCredentials,
    FreeNSName, FreeNSString, FreeVerifier, IdentityObject,
    IncrementVerifierTicks, InternalAuthenticate,
    InternalExtractCredentialsDetails, MakeEmptyNSName,
    MakeVerifierFromHashedPassword, MakeSimpleCredentials,
    NilOrNullName, PackStrongVerifier, Style, UnpackStrongVerifier],
  AuthProtocol USING [StrongVerifier],
  AuthSpecial USING [],
  CH USING [
    ConversationHandle, FreeConversationHandle,
    LookupDistinguishedName, ReturnCode],
  Courier USING [Error],
  DESFace USING [CheckKeyParity, Key, nullKey],
  Heap USING [Create, MakeNode],
  NSName USING [
    CopyNameFields, maxDomainLength, maxLocalLength, maxOrgLength, Name],
  NSString USING [nullString, String],
  NSStringExtras USING [EquivalentNames],
  Process USING [InitializeMonitor, Pause, SecondsToTicks, Ticks],
  Router USING [AssignAddress],
  SharedKeys USING [asName, chsName, msName],
  SpecialCHAuth USING [],
  System USING [
    GetClockPulses, GetGreenwichMeanTime, gmtEpoch,
    GreenwichMeanTime, HostNumber, NetworkAddress,
    nullHostNumber, SecondsSinceEpoch];

AuthAImpl: MONITOR
  LOCKS LOOPHOLE[identity, PrivateIdentityHandle]^ USING identity: Auth.IdentityHandle
  IMPORTS
    Auth, AuthInternal, CH, Courier, DESFace, Heap, NSName,
    NSStringExtras, Process, Router, SharedKeys, System
  EXPORTS Auth, AuthSpecial, SpecialCHAuth
  SHARES Auth =

BEGIN


-- Globals and constants --


cacheConversations: BOOLEAN _ TRUE;

conversationCacheTimeout: LONG CARDINAL = LONG[12] * LONG[60] * LONG[60];
  -- 12 hours, in seconds (half of the crentials lifetime).

conversationHeap: UNCOUNTED ZONE _ Heap.Create[initial: 4, increment: 4];
  -- All conversations and associated storage (creds, etc.) are allocated
  -- from this heap. We no longer use the clients heap. This was
  -- because conversation caching made it possible for one
  -- client to lose storage owned by another.

asStubHeap: UNCOUNTED ZONE = AuthInternal.asStubHeap;

timesHaveHadToWaitForNextVerifier: CARDINAL _ 0;
  -- This global is the number of times we've tried to create more than one
  -- verifier within a given second and run out of ticks since the last boot.
  -- It should be pretty small.

thisMachinesAddress: System.NetworkAddress _ Router.AssignAddress[];


-- Public errors --


AuthenticationError: PUBLIC ERROR [reason: Auth.AuthenticationProblem] = CODE;

CallError: PUBLIC ERROR [reason: Auth.CallProblem, whichArg: Auth.WhichArg] = CODE;

OrphanConversation: PUBLIC ERROR = CODE;
  -- This error is raised by Refresh only.


-- Private errors --


BadLineClock: PRIVATE ERROR = CODE;
  -- Indicates that the line clock is not advancing.
  -- Should never happen unless your hardware is broken.
```

```
-- Identities --

MakeIdentity: PUBLIC PROC [
  myName: NSName.Name, password: NSString.String, z: UNCOUNTED ZONE,
  style: Auth.Flavor, dontCheck: BOOLEAN]
    RETURNS [identity: Auth.IdentityHandle] =
  BEGIN
    newIdentity: PrivateIdentityHandle;

    newIdentity _ z.NEW[AuthInternal.IdentityObject];
    newIdentity.style _ style;
    newIdentity.myName _ AuthInternal.MakeEmptyNSName[z];
  -- -- Assume: newIdentity.myName will be big enough to hold myName;
  -- --  no heap will be needed by CopyNameFields.
    NSName.CopyNameFields[
      z: NIL, source: myName, destination: newIdentity.myName];
    newIdentity.myPassword _ AuthInternal.CloneNSString[password, z];
    newIdentity.myStrongKey _ PrivateKey[Auth.PasswordStringToKey[password]];
    newIdentity.myHashedPassword _ Auth.HashSimplePassword[password];
    newIdentity.nameHasBeenResolved _ FALSE;
    newIdentity.conversationsInUse _ NIL;
    newIdentity.cachedConversations _ NIL;
    newIdentity.owningHeap _ z;
    Process.InitializeMonitor[@newIdentity.LOCK];
    identity _ PublicIdent[newIdentity];
    IF ~dontCheck THEN SelfAuthenticate[
      identity ! UNWIND => FreeIdentity[@identity, NIL]]; -- No heap needed.
  END;

MakeStrongIdentityUsingKey: PUBLIC PROC [
  myName: NSName.Name, myKey: Auth.Key, z: UNCOUNTED ZONE,
  dontCheck: BOOLEAN]
    RETURNS [identity: Auth.IdentityHandle] =
  BEGIN
    IF ~DESFace.CheckKeyParity[LOOPHOLE[LONG[@myKey]]]
      THEN ERROR CallError[badKey, notApplicable];
    identity _ MakeIdentity[myName, NSString.nullString, z, strong, TRUE];
    PrivateIdent[identity].myStrongKey _ PrivateKey[myKey];
    IF ~dontCheck THEN SelfAuthenticate[
      identity ! UNWIND => FreeIdentity[@identity, NIL]]; -- No heap needed.
  END;

FreeIdentity: PUBLIC PROC [
  identityPtr: LONG POINTER TO Auth.IdentityHandle, z: UNCOUNTED ZONE] =
  -- Note: z is no longer used.
  BEGIN
    CleanUpIdentity: ENTRY PROC [identity: Auth.IdentityHandle] =
      BEGIN
        ENABLE UNWIND => NULL;
        thisOne: PrivateConversationHandle;
        -- Clean up currently active conversations (making them orphans):
        -- (These conversations are in use; they should not be freed.)
        thisOne _ PrivateIdent[identity].conversationsInUse;
        WHILE thisOne # NIL DO
          nextOne: PrivateConversationHandle _ thisOne.next;
          thisOne.owner _ NIL;
          thisOne.next _ NIL;
          thisOne _ nextOne;
        ENDLOOP;

        -- Clean up cached conversations:
        -- (These conversations are not in use; they should be freed.)
        thisOne _ PrivateIdent[identity].cachedConversations;
        WHILE thisOne # NIL DO
          nextOne: PrivateConversationHandle _ thisOne.next;
          thisOne.owner _ NIL; -- Prevents monitor lock.
          InternalTerminate[LOOPHOLE[LONG[@thisOne]]];
          thisOne _ nextOne;
        ENDLOOP;

        AuthInternal.FreeNSName[
          @PrivateIdent[identity].myName, PrivateIdent[identity].owningHeap];
        AuthInternal.FreeNSString[
          @PrivateIdent[identity].myPassword, PrivateIdent[identity].owningHeap];
      END;

    IF identityPtr^ = NIL THEN RETURN;
    CleanUpIdentity[identityPtr^];
      -- There is a tiny race here; we free the monitor lock before we
      -- free the identity itself. This shouldn't matter; if the client
      -- is calling FreeIdentity, he'd better not be using it!
    PrivateIdent[identityPtr^].owningHeap.FREE[identityPtr];
      -- Smashes NIL into identityPtr^.
  END;


-- Conversations --

Initiate: PUBLIC PROC [
  identity: Auth.IdentityHandle, recipientsName: NSName.Name,
  recipientsHostNumber: System.HostNumber _ Auth.nullHostNumber,
  z: UNCOUNTED ZONE]
    RETURNS [conversation: Auth.ConversationHandle] =
  -- Note: z is no longer used.
  BEGIN
```

```
      SELECT AuthInternal.Style[identity] FROM
        simple => ResolveNameIfAlias[identity];
        strong => CheckForValidName[PrivateIdent[identity].myName, strong];
      ENDCASE => ERROR;
      conversation ← InternalInitiate[identity, recipientsName, recipientsHostNumber];
    END;

  Terminate: PUBLIC PROC [
    conversationPtr: LONG POINTER TO Auth.ConversationHandle, z: UNCOUNTED ZONE] =
    BEGIN
      -- Note: z is no longer used.
      InternalTerminate[conversationPtr, cacheConversations];
    END;

  Refresh: PUBLIC PROC [conversation: Auth.ConversationHandle] =
    BEGIN
      MonitoredFetchStrongCredentials: ENTRY PROC [identity: Auth.IdentityHandle] =
        -- If successful, sets values of newCreds and newConversationKey.
        -- Assume: PrivateConv[conversation].owner = identity
        BEGIN
          ENABLE UNWIND => NULL;
          [newCreds, newConversationKey] ←
            Auth.FetchStrongCredentials[
              PrivateConv[conversation].owner.myName,
              PrivateConv[conversation].recipient,
              PublicKey[PrivateConv[conversation].owner.myStrongKey],
              PrivateConv[conversation].owningHeap];
        END;

      newConversationKey: Auth.Key ← Auth.nullKey;
      newCreds: Auth.Credentials ← Auth.nullCredentials;

      IF PrivateConv[conversation].owner = NIL THEN ERROR OrphanConversation;
      SELECT AuthInternal.Style[PublicIdent[PrivateConv[conversation].owner]] FROM
        simple => RETURN; -- Noop
        strong =>
          MonitoredFetchStrongCredentials[PublicIdent[PrivateConv[conversation].owner]];
      ENDCASE => ERROR;

      -- We fetched new credentials using the conversation's heap. We
      -- may now free the old conversation credentials and replace them
      -- with the one's we just fetched.
      AuthInternal.FreeCredentials[
        @PrivateConv[conversation].creds,
        PrivateConv[conversation].owningHeap];
      PrivateConv[conversation].creds ← newCreds;
      newCreds ← Auth.nullCredentials;
        -- We've saved these credentials, so forget 'em.
      PrivateConv[conversation].conversationKey ← PrivateKey[newConversationKey];
      PrivateConv[conversation].clearLastVerifier
        ← [System.GetGreenwichMeanTime[], GetRandomTicksForVerifier[]];
      PrivateConv[conversation].creationTime ← System.GetGreenwichMeanTime[];
    END;

  CheckOutCredentials: PUBLIC PROC [conversation: Auth.ConversationHandle]
    RETURNS [creds: Auth.Credentials] =
    BEGIN
      RETURN[PrivateConv[conversation].creds];
    END;

  CheckOutNextVerifier: PUBLIC PROC [
    conversation: Auth.ConversationHandle, recipientsHostNumber: System.HostNumber]
      RETURNS [verifier: Auth.Verifier] =
    BEGIN
      recipientsHostNumber ←
        IF recipientsHostNumber # System.nullHostNumber
          THEN recipientsHostNumber
          ELSE PrivateConv[conversation].recipientsHostNumber;
      RETURN[
        ComputeNextVerifierForConversation[
          PrivateConv[conversation],
          recipientsHostNumber]
        ];
    END;

  CheckOutCredsAndNextVerifier: PUBLIC PROC [
    conversation: Auth.ConversationHandle,
    recipientsHostNumber: System.HostNumber]
    RETURNS [creds: Auth.Credentials, verifier: Auth.Verifier] =
    BEGIN
      recipientsHostNumber ←
        IF recipientsHostNumber # System.nullHostNumber
          THEN recipientsHostNumber
          ELSE PrivateConv[conversation].recipientsHostNumber;
      RETURN[
        PrivateConv[conversation].creds,
        ComputeNextVerifierForConversation[
          PrivateConv[conversation],
          recipientsHostNumber]];
    END;


  -- Serialization --


    -- NOTE: DescribeCredentials and DescribeVerifier are
    --  exported by AuthProtocolImpl.
```

```
-- Exports to AuthSpecial and SpecialCHAuth --


-- AuthSpecial. --
MakeNullCHConversation: PUBLIC PROC [z: UNCOUNTED ZONE]
  RETURNS [nullCHConv: CH.ConversationHandle _ [NIL, NIL]] =
  BEGIN
    nullConv: PrivateConversationHandle _
      z.NEW[AuthInternal.ConversationObject _ []];
    nullConv.owningHeap _ z;
    nullCHConv.conversation _ PublicConv[nullConv];
  END;

-- SpecialCHAuth. --
MakeConversationFromCredsAndVerifier: PUBLIC ENTRY PROC [
  identity: Auth.IdentityHandle, creds: Auth.Credentials, verifier: Auth.Verifier,
  z: UNCOUNTED ZONE]
    RETURNS [conversation: Auth.ConversationHandle] =
  -- Note: z is no longer used.
  BEGIN
    ENABLE UNWIND => NULL;
    newConversation: PrivateConversationHandle;
    convKey: Auth.Key _ Auth.nullKey;
    badCreds: BOOLEAN;

    SELECT creds.flavor FROM
      simple => NULL;
      strong =>
        BEGIN
          -- Assume: PrivateIdent[identity].style # simple
          [, convKey, , , badCreds, , ] _
            AuthInternal.InternalExtractCredentialsDetails[
              PublicKey[PrivateIdent[identity].myStrongKey], creds, NIL, FALSE];
          IF badCreds THEN ERROR AuthenticationError[credentialsInvalid];
        END;
    ENDCASE => ERROR AuthenticationError[credentialsInvalid];

    newConversation _ conversationHeap.NEW[AuthInternal.ConversationObject];
    newConversation.recipient _ NIL;
    -- Assume: Noone will be interested in the recipient's name so why copy it?
    -- NOTE: This is an orphan conversation.
    newConversation.creds _ Auth.CopyCredentials[creds, conversationHeap];
    newConversation.lastVerifier _ Auth.nullVerifier; -- Filled in below.
    newConversation.conversationKey _ PrivateKey[convKey];
    newConversation.incrementVerifierByTicks _ TRUE;
    newConversation.clearLastVerifier _ [System.gmtEpoch, 0];
    newConversation.recipientsHostNumber _ System.nullHostNumber;
    newConversation.creationTime _ System.GetGreenwichMeanTime[];
    newConversation.owner _ NIL;
    newConversation.next _ NIL;
    newConversation.owningHeap _ conversationHeap;

    SELECT creds.flavor FROM
      simple =>
        newConversation.lastVerifier _
          AuthInternal.CloneVerifier[verifier, conversationHeap];
      strong =>
        BEGIN
          ENABLE UNWIND =>
            InternalTerminate[LOOPHOLE[LONG[@newConversation]]];
              -- No monitor problems because newConversation is an orphan.

          newConversation.clearLastVerifier _
            AuthInternal.UnpackStrongVerifier[
              verifier, PrivateKey[convKey], thisMachinesAddress.host
              ! Courier.Error => ERROR AuthenticationError[verifierInvalid]];
          newConversation.lastVerifier _
            DESCRIPTOR[
              Heap.MakeNode[conversationHeap, SIZE[AuthProtocol.StrongVerifier]],
              SIZE[AuthProtocol.StrongVerifier]];
        END;
    ENDCASE => ERROR AuthenticationError[credentialsInvalid];

    RETURN[PublicConv[newConversation]];
  END;


-- Private stuff --


CheckForValidName: PROC [name: NSName.Name, style: Auth.Flavor] =
  -- Raises an error (according to style) if the name is NOT ok.
  BEGIN
    IF AuthInternal.NilOrNullName[name] OR
      name.local.length > NSName.maxLocalLength OR
      name.domain.length > NSName.maxDomainLength OR
      name.org.length > NSName.maxOrgLength
        THEN ERROR CallError[
          (IF style = simple THEN simpleKeyDoesNotExist ELSE strongKeyDoesNotExist),
            notApplicable];
  END;

ComputeNextVerifierForConversation: PROC [
  conversation: PrivateConversationHandle,
  recipientsHostNumber: System.HostNumber]
```

```
    RETURNS [nextVerifier: Auth.Verifier] =
  -- This will compute a unique verifier later than last. If necessary,
  -- it will wait one second. If the timeStamp in the verifier is greater than
  -- the current time (i.e. in the future), you're out of luck. It has the
  -- side effect of modifying the lastVerifier and clearLastVerifier fields
  -- of the conversation.
  BEGIN
    SELECT TRUE FROM
      conversation.creds.flavor = simple => RETURN[conversation.lastVerifier];
      conversation.incrementVerifierByTicks => -- strong, strange case --
        BEGIN
          conversation.clearLastVerifier ←
            AuthInternal.IncrementVerifierTicks[conversation.clearLastVerifier];
          AuthInternal.PackStrongVerifier[
            from: @conversation.clearLastVerifier,
            recipientsHostNumber: recipientsHostNumber,
            destVerifier: conversation.lastVerifier,
            key: conversation.conversationKey];
        END;
    ENDCASE => -- strong, normal case --
      BEGIN
        last: AuthProtocol.StrongVerifier ← conversation.clearLastVerifier;
        timeOfLastVerifier, timeOfNextVerifier: LONG CARDINAL;
        oneSecond: Process.Ticks ← Process.SecondsToTicks[1];
        next: AuthProtocol.StrongVerifier ← [
          timeStamp: System.GetGreenwichMeanTime[],
          ticks: GetRandomTicksForVerifier[]];
        timeOfLastVerifier ← System.SecondsSinceEpoch[last.timeStamp];
        timeOfNextVerifier ← System.SecondsSinceEpoch[next.timeStamp];
        IF ~(timeOfNextVerifier > timeOfLastVerifier) THEN
          BEGIN
            IF timeOfNextVerifier < timeOfLastVerifier THEN ERROR BadLineClock;
              -- Time appears to be going backwards!!!
          -- If we get this far, timeOfNextVerifier = timeOfLastVerifier. Since
          -- we can't make duplicate verifiers, we've got to increment the ticks
          -- or, if we've run out of ticks in a LONG CARDINAL, then we've got
          -- to wait until the next second. (This should be VERY rare.)
            IF last.ticks # LAST[LONG CARDINAL]
              THEN next.ticks ← last.ticks + 1
              ELSE
                BEGIN
                  Process.Pause[oneSecond];
                  timesHaveHadToWaitForNextVerifier ←
                    timesHaveHadToWaitForNextVerifier + 1;
                  next ← [
                    timeStamp: System.GetGreenwichMeanTime[],
                    ticks: GetRandomTicksForVerifier[]];
                  IF System.SecondsSinceEpoch[next.timeStamp] <=
                    timeOfLastVerifier
                      THEN ERROR BadLineClock;
                        -- If, after pausing for a second, the clock
                        -- hasn't advanced, something must be wrong
                        -- with the clock...
                END;
          END;
        conversation.clearLastVerifier ← next;
        AuthInternal.PackStrongVerifier[
          from: @conversation.clearLastVerifier,
          recipientsHostNumber: recipientsHostNumber,
          destVerifier: conversation.lastVerifier,
          key: conversation.conversationKey];
      END;
      RETURN[conversation.lastVerifier];
  END;

GetConversationFromCache: INTERNAL PROC [
  id: PrivateIdentityHandle, recipient: NSName.Name]
  RETURNS [conv: PrivateConversationHandle ← NIL] =
  BEGIN
    FindConversationFor: PROC [name: NSName.Name]
      RETURNS [convFound: PrivateConversationHandle] =
      BEGIN
        previous: LONG POINTER TO PrivateConversationHandle;
        IF AuthInternal.NilOrNullName[name] THEN RETURN[NIL];
      -- Find the conversation (if any) preceding the one
      --  we'd like to use.
        previous ← @id.cachedConversations;
        WHILE previous^ # NIL DO -- For entire list, do:
          IF NSStringExtras.EquivalentNames[previous^.recipient, name]
            THEN EXIT; -- Found one!
          previous ← @previous^.next;
        ENDLOOP;
        IF previous^ = NIL THEN RETURN[NIL]; -- Didn't find one.
      -- Found one, splice it out:
        convFound ← previous^;
        previous^ ← convFound.next;
      -- Put it on the front of the list of conversations in use:
        convFound.next ← id.conversationsInUse;
        id.conversationsInUse ← convFound;
        RETURN[convFound];
      END;
    WeedOutOldConversations: PROC =
      BEGIN
        previous: LONG POINTER TO PrivateConversationHandle ←
          @id.cachedConversations;
        convToRemove: PrivateConversationHandle ← NIL;
        -- Seek out and destroy all old conversations:
```

```
            WHILE previous^ # NIL DO -- For entire list, do
              SELECT OldConversation[previous^] FROM
                FALSE =>
                  BEGIN
                    previous _ @previous^.next;
                    LOOP; -- This is a young entry; pass over it.
                  END;
                TRUE =>
                  BEGIN
                    convToRemove _ previous^;
                  -- Splice this conversation out of list:
                    previous^ _ previous^.next;
                  -- Kill it:
                    convToRemove.owner _ NIL;
                    InternalTerminate[LOOPHOLE[LONG[@convToRemove]]];
                  END;
              ENDCASE;
            ENDLOOP;
          END;

        WeedOutOldConversations[];
        RETURN[FindConversationFor[recipient]];
      END;

    GetRandomTicksForVerifier: PROC RETURNS [ticks: LONG CARDINAL] =
      BEGIN
        ticks _ System.GetClockPulses[];
        IF ticks > 10000 THEN ticks _ ticks - 10000;
          -- This makes sure that ticks is within [0..MAX[LONG CARDINAL] - 10000].
          -- That way, we can make up to 10000 verifiers within a given second
          -- by incrementing the ticks field.
      END;

    InternalInitiate: PRIVATE ENTRY PROC [
      identity: Auth.IdentityHandle, recipientsName: NSName.Name,
      recipientsHostNumber: System.HostNumber _ Auth.nullHostNumber]
        RETURNS [conversation: Auth.ConversationHandle] =
      BEGIN
        ENABLE UNWIND => NULL;
        newConversation: PrivateConversationHandle;
        newCreds: Auth.Credentials;
        newConversationKey: Auth.Key _ Auth.nullKey;

        newConversation _
          GetConversationFromCache[PrivateIdent[identity], recipientsName];
        IF newConversation # NIL THEN
          BEGIN -- If there was a conversation in the cache, use it!
            newConversation.recipientsHostNumber _ recipientsHostNumber;
            RETURN[PublicConv[newConversation]];
          END;

      -- No credentials in the cache, so manufacture or fetch some:
        SELECT PrivateIdent[identity].style FROM
          simple =>
            newCreds _
              AuthInternal.MakeSimpleCredentials[
                PrivateIdent[identity].myName, conversationHeap];
          strong =>
            [newCreds, newConversationKey] _
              Auth.FetchStrongCredentials[
                PrivateIdent[identity].myName, recipientsName,
                PublicKey[PrivateIdent[identity].myStrongKey], conversationHeap];
        ENDCASE => ERROR;

        newConversation _ conversationHeap.NEW[AuthInternal.ConversationObject];
        newConversation.recipient _
          AuthInternal.CloneNSName[recipientsName, conversationHeap];
        newConversation.creds _ newCreds;
        newConversation.lastVerifier _ Auth.nullVerifier; -- Filled in below.
        newConversation.conversationKey _ PrivateKey[newConversationKey];
        newConversation.incrementVerifierByTicks _ FALSE;
        newConversation.clearLastVerifier _
          [System.GetGreenwichMeanTime[], GetRandomTicksForVerifier[]];
        newConversation.recipientsHostNumber _ recipientsHostNumber;
        newConversation.creationTime _ System.GetGreenwichMeanTime[];
        newConversation.owner _ NIL;
        newConversation.next _ NIL;
        newConversation.owningHeap _ conversationHeap;

        SELECT TRUE FROM
          newConversation.creds = Auth.nullCredentials =>
            newConversation.lastVerifier _ Auth.nullVerifier;
          PrivateIdent[identity].style = simple =>
            newConversation.lastVerifier _
              AuthInternal.MakeVerifierFromHashedPassword[
                PrivateIdent[identity].myHashedPassword, conversationHeap];
          PrivateIdent[identity].style = strong =>
            newConversation.lastVerifier _
              DESCRIPTOR[
                Heap.MakeNode[conversationHeap, SIZE[AuthProtocol.StrongVerifier]],
                SIZE[AuthProtocol.StrongVerifier]];
        ENDCASE => ERROR;

      -- Put on front of identity's list of conversations in use:
        newConversation.owner _ PrivateIdent[identity];
        newConversation.next _ PrivateIdent[identity].conversationsInUse;
        PrivateIdent[identity].conversationsInUse _ newConversation;
```

```
    RETURN[PublicConv[newConversation]];
  END;

InternalTerminate: PROC [
  conversationPtr: LONG POINTER TO Auth.ConversationHandle,
  okToCache: BOOLEAN _ FALSE] =
  BEGIN
    RemoveConvFromIdentity: ENTRY PROC [identity: Auth.IdentityHandle] =
      -- Removes conversationPtr^ from identity.conversationsInUse.
      BEGIN
        ENABLE UNWIND => NULL;
        -- Assume: identity = PrivateConv[conversationPtr^].owner
        -- Assume: identity # NIL
        previous: LONG POINTER TO PrivateConversationHandle _
          @PrivateIdent[identity].conversationsInUse;
      -- Find previous conversation in chain:
        DO
          IF previous^ = PrivateConv[conversationPtr^] THEN EXIT;
          IF previous^ = NIL THEN ERROR;
            -- A conversation has an owner that doesn't know about
            -- the conversation. This "can't" happen (if it does there
            -- is a bug in this code somewhere.
          previous _ @previous^.next;
        ENDLOOP;
      -- Splice out this conversation:
        previous^ _ PrivateConv[conversationPtr^].next;
        PrivateConv[conversationPtr^].next _ NIL;
      END;
    OfferConvToIdentityCache: ENTRY PROC [
      identity: Auth.IdentityHandle, conversation: PrivateConversationHandle]
        RETURNS [accepted: BOOLEAN _ FALSE] =
      -- This operation may add conversation to identity.cachedConversations.
      -- If it does, accepted will be set to TRUE.
      BEGIN
        ENABLE UNWIND => NULL;

        IF ~okToCache
          OR conversation.incrementVerifierByTicks
          OR OldConversation[conversation]
          OR AuthInternal.NilOrNullName[conversation.recipient]
            THEN RETURN[accepted: FALSE];
        conversation.next _ PrivateIdent[identity].cachedConversations;
        PrivateIdent[identity].cachedConversations _ conversation;
        RETURN[accepted: TRUE];
      END;

    addedToCache: BOOLEAN;
    IF conversationPtr^ = NIL THEN RETURN; -- Nothing to do.
    IF PrivateConv[conversationPtr^].owner # NIL THEN
      BEGIN
        RemoveConvFromIdentity[PublicIdent[PrivateConv[conversationPtr^].owner]];
        addedToCache _
          OfferConvToIdentityCache[
            PublicIdent[PrivateConv[conversationPtr^].owner],
            PrivateConv[conversationPtr^]];
        IF addedToCache
          THEN {conversationPtr^ _ NIL; RETURN};
            -- Do NOT free the conversation; do set it to NIL.
      END;

    -- Free the conversation:
    AuthInternal.FreeNSName[
      @PrivateConv[conversationPtr^].recipient,
      PrivateConv[conversationPtr^].owningHeap];
    AuthInternal.FreeCredentials[
      @PrivateConv[conversationPtr^].creds,
      PrivateConv[conversationPtr^].owningHeap];
    AuthInternal.FreeVerifier[
      @PrivateConv[conversationPtr^].lastVerifier,
      PrivateConv[conversationPtr^].owningHeap];
    PrivateConv[conversationPtr^].owningHeap.FREE[conversationPtr];
      -- Smashes NIL into conversationPtr^.
  END;

IsWellKnownName: PROC [name: NSName.Name]
  RETURNS [isWellKnown: BOOLEAN] =
  BEGIN
    RETURN[
      AuthInternal.EquivalentNames[name, SharedKeys.asName] OR
      AuthInternal.EquivalentNames[name, SharedKeys.chsName] OR
      AuthInternal.EquivalentNames[name, SharedKeys.msName]
      ];
  END;

MakeSimpleCHOrphanConversation: PROC [
  for: NSName.Name, hashedPassword: Auth.HashedPassword]
  RETURNS [chConv: CH.ConversationHandle] =
  BEGIN
    chConv _ MakeNullCHConversation[asStubHeap];
    PrivateConv[chConv.conversation].creds _
      AuthInternal.MakeSimpleCredentials[for, asStubHeap];
    PrivateConv[chConv.conversation].lastVerifier _
      AuthInternal.MakeVerifierFromHashedPassword[hashedPassword, asStubHeap];
  END;

OldConversation: PROC [conv: PrivateConversationHandle]
  RETURNS [isOld: BOOLEAN] =
```

```
    BEGIN
      now: LONG CARDINAL _
        System.SecondsSinceEpoch[System.GetGreenwichMeanTime[]];
      then: LONG CARDINAL _
        System.SecondsSinceEpoch[conv.creationTime];
      IF then > now
        THEN isOld _ TRUE
          -- Clock is running backward?? Consider the conversation "old".
        ELSE isOld _ ((now - then) > conversationCacheTimeout);
    END;

  ResolveNameIfAlias: ENTRY PROC [identity: Auth.IdentityHandle] =
    -- Assume: This operation gets called only with a simple identity.
    BEGIN
      ENABLE {
        AuthenticationError => ERROR CallError[badKey, initiator];
        UNWIND => NULL;
        };

      chConversation: CH.ConversationHandle _ [NIL, NIL];
      distingName: NSName.Name _ NIL;
      rc: CH.ReturnCode;

      IF PrivateIdent[identity].nameHasBeenResolved THEN RETURN;
      CheckForValidName[PrivateIdent[identity].myName, PrivateIdent[identity].style];
      IF IsWellKnownName[PrivateIdent[identity].myName] THEN
        BEGIN
          PrivateIdent[identity].nameHasBeenResolved _ TRUE;
          RETURN;
        END;

      chConversation _ MakeSimpleCHOrphanConversation[
        PrivateIdent[identity].myName, PrivateIdent[identity].myHashedPassword];
      distingName _ AuthInternal.MakeEmptyNSName[asStubHeap];
      BEGIN
        ENABLE UNWIND => {
          CH.FreeConversationHandle[@chConversation, NIL];
          AuthInternal.FreeNSName[@distingName, asStubHeap];
          };

        rc _ CH.LookupDistinguishedName[
          chConversation, PrivateIdent[identity].myName, distingName];
        SELECT rc.code FROM
          done =>
            BEGIN
            -- Assume: PrivateIdent[identity].myName is big enough;
            --  no heap will be needed by CopyNameFields.
              NSName.CopyNameFields[
                z: NIL, source: distingName,
                destination: PrivateIdent[identity].myName];
              PrivateIdent[identity].nameHasBeenResolved _ TRUE;
            END;
          noSuchOrg, noSuchDomain, noSuchLocal,
          illegalOrgName, illegalDomainName, illegalLocalName =>
            ERROR CallError[simpleKeyDoesNotExist, initiator];
          rejectedTooBusy =>
            ERROR CallError[tooBusy, notApplicable];
          allDown =>
            ERROR CallError[keysUnavailable, initiator];
          credentialsInvalid =>
            ERROR CallError[badKey, initiator];
        ENDCASE => ERROR CallError[simpleKeyDoesNotExist, initiator];
        AuthInternal.FreeNSName[@distingName, asStubHeap];
      END;
      CH.FreeConversationHandle[@chConversation, NIL];
    END;

  SelfAuthenticate: PROC [identity: Auth.IdentityHandle] =
    -- This operation will also resolve the name in the identity handle.
    -- Assume: This procedure is only called from an identity creation procedure!
    BEGIN
      creds: Auth.Credentials _ Auth.nullCredentials;
      verifier: Auth.Verifier _ Auth.nullVerifier;
      conv: Auth.ConversationHandle _ NIL;

      SELECT PrivateIdent[identity].style FROM
        simple =>
          ResolveNameIfAlias[identity];
        strong =>
          CheckForValidName[PrivateIdent[identity].myName, strong];
          -- Note: Since the strong credentials produced by the AS always contain
          -- a distinguished name, we save a Clearinghouse operation by extracting
          -- the disinguished name from the credentials that we are using to check
          -- this identity.
      ENDCASE => ERROR;

      SELECT PrivateIdent[identity].style FROM
        simple =>
          BEGIN
            ENABLE UNWIND => InternalTerminate[@conv];

            conv _ InternalInitiate[identity, PrivateIdent[identity].myName];
            [creds, verifier] _ Auth.CheckOutCredsAndNextVerifier[conv];
            IF ~Auth.CheckSimpleCredentials[creds, verifier].ok
              THEN ERROR CallError[badKey, notApplicable];
            InternalTerminate[@conv];
          END;
```

```
        strong = >
          BEGIN
            ENABLE UNWIND = > InternalTerminate[@conv];

            distIngName: NSName.Name __ NIL;
            conv __ InternalInitiate[identity, PrivateIdent[identity].myName];
            [creds, verifier] __
              CheckOutCredsAndNextVerifier[conv, thisMachinesAddress.host];
          - - To get the distinguished name for strong credentials, do an
          - - Authenticate, which returns the distinguished name (which was
          - - resolved by the server). Replace the identity.myName with this name.
            [distingName, , ] __ AuthInternal.InternalAuthenticate[
              PublicKey[PrivateIdent[identity].myStrongKey], creds, verifier,
              asStubHeap, thisMachinesAddress.host,
              FALSE, FALSE, TRUE, FALSE
              ! AuthenticationError = > ERROR CallError[badKey, notApplicable]];

          - - Assume: PrivateIdent[identity].myName is big enough;
          - -  no heap will be needed by CopyNameFields.
            NSName.CopyNameFields[
              z: NIL, source: distingName, destination: PrivateIdent[identity].myName];
            AuthInternal.FreeNSName[@distingName, asStubHeap];
            PrivateIdent[identity].nameHasBeenResolved __ TRUE;
            InternalTerminate[@conv];
          END;
      ENDCASE = > ERROR;
    END;


- - Public/Private pointer conversions - -


    - - Note: This is a hack to avoid exporting public types.
    - - (Because exported type clashes make it impossible for two
    - - different implementations which export the same type to co - exist.)

PrivateIdentityHandle: TYPE = LONG POINTER TO AuthInternal.IdentityObject;
PrivateConversationHandle: TYPE = LONG POINTER TO AuthInternal.ConversationObject;

PrivateConv: PROC [conversation: Auth.ConversationHandle]
    RETURNS [PrivateConversationHandle] = INLINE
      {RETURN[LOOPHOLE[conversation]]};

PublicConv: PROC [privateConversation: PrivateConversationHandle]
    RETURNS [Auth.ConversationHandle] = INLINE
      {RETURN[LOOPHOLE[privateConversation]]};

PrivateIdent: PROC [identity: Auth.IdentityHandle]
    RETURNS [PrivateIdentityHandle] = INLINE
      {RETURN[LOOPHOLE[identity]]};

PublicIdent: PROC [privateIdentity: PrivateIdentityHandle]
    RETURNS [Auth.IdentityHandle] = INLINE
      {RETURN[LOOPHOLE[privateIdentity]]};

PrivateKey: PROC [key: Auth.Key]
    RETURNS [DESFace.Key] = INLINE
      {RETURN[LOOPHOLE[key]]};

PublicKey: PROC [privateKey: DESFace.Key]
    RETURNS [Auth.Key] = INLINE
      {RETURN[LOOPHOLE[privateKey]]};


END.
```

```
-- AuthCImpl.mesa
-- JMaloney, 11-Jul-83 12:24:49.

-- Last modified: JMaloney, 3-Jul-84 12:00:10.


DIRECTORY
  Auth USING [
    AuthenticationError, CallError, CallProblem,
    CheckOutCredsAndNextVerifier, ConversationHandle,
    Credentials, Flavor, HashedPassword, HashSimplePassword,
    IdentityHandle, Initiate, Key, nullCredentials,
    nullHashedPassword, nullKey, nullVerifier,
    PasswordStringToKey, Terminate, Verifier, WhichArg],
  AuthInternal USING [
    asStubHeap, BadCredentialsPackage, CloneNSString,
    ConversationObject, ExtractStuffFromCredentialsPackage, FreeNSString,
    FreeSimpleCredentials, IdentityObject,
    Style, UnpackSimpleCredentials],
  AuthOps USING [
    ChangeSimpleKey, ChangeStrongKey,
    CreateSimpleKey, CreateStrongKey,
    DeleteSimpleKey, DeleteStrongKey,
    CheckSimpleCredentials, GetStrongCredentials],
  AuthProtocol USING [
    CredentialsPackage, DescribeCredentialsPackage, SimpleCredentials],
  AuthServerCache USING [ASAddress, NextPlease, Refill],
  Courier USING [Description, Error, Free],
  DESFace USING [Block, CheckKeyParity, EncryptBlock, Key, nullKey],
  NSName USING [Name],
  NSString USING [String],
  Process USING [MsecToTicks, Pause, Ticks],
  SharedKeys USING [asName],
  System USING [GetClockPulses, NetworkAddress, nullNetworkAddress];

AuthCImpl: MONITOR
  LOCKS LOOPHOLE[identity, PrivateIdentityHandle]^ USING identity: Auth.IdentityHandle
  IMPORTS
    Auth, AuthInternal, AuthProtocol, AuthOps, AuthServerCache, Courier,
    DESFace, Process, SharedKeys, System
  EXPORTS Auth
  SHARES Auth =

BEGIN


-- Globals and constants --


cacheNeverFilled: BOOLEAN _ TRUE;

pauseForInitialCacheFill: Process.Ticks _ Process.MsecToTicks[15000];
  -- fifteen seconds

timesToRetry: CARDINAL _ 1;
  -- The number of times to grab a new address out of the cache and re-try
  -- an operation which must talk to an AS.


-- Password/Key administration operations --


ChangeMyPasswords: PUBLIC PROC [
  identity: Auth.IdentityHandle, newPassword: NSString.String, z: UNCOUNTED ZONE,
  changeStrong, changeSimple: BOOLEAN] =
  BEGIN
    ComputeOldKeyValues: ENTRY PROC [identity: Auth.IdentityHandle]
      RETURNS [oldStrongValue: Auth.Key, oldSimpleValue: Auth.HashedPassword] =
      -- Computes oldStrongValue and oldSimpleValue from password
      -- in identity handle.
      BEGIN
        ENABLE UNWIND => NULL;
        oldStrongValue _
          Auth.PasswordStringToKey[PrivateIdent[identity].myPassword];
        oldSimpleValue _
          Auth.HashSimplePassword[PrivateIdent[identity].myPassword];
      END;
    ChangeIdentityInfo: ENTRY PROC [identity: Auth.IdentityHandle] =
      -- Fix up the password info in the identity.
      BEGIN
        ENABLE UNWIND => NULL;
        AuthInternal.FreeNSString[
          @PrivateIdent[identity].myPassword, PrivateIdent[identity].owningHeap];
        PrivateIdent[identity].myPassword _
          AuthInternal.CloneNSString[newPassword, PrivateIdent[identity].owningHeap];
        IF changeStrong
          THEN PrivateIdent[Identity].myStrongKey _ PrivateKey[newStrongKey];
        IF changeSimple
          THEN PrivateIdent[Identity].myHashedPassword _ newHashedPassword;
      END;

    newStrongKey: Auth.Key _ Auth.PasswordStringToKey[newPassword];
    newHashedPassword: Auth.HashedPassword _
      Auth.HashSimplePassword[newPassword];
    IF changeStrong THEN ChangeStrongKey[identity, newStrongKey];
    IF changeSimple THEN
      BEGIN
```

```
            ENABLE UNWIND => {
              -- Undo what we have done so far...
              oldStrongKey: Auth.Key _ Auth.nullKey;
              oldSimpleKey: Auth.HashedPassword _ Auth.nullHashedPassword;
              [oldStrongKey, oldSimpleKey] _ ComputeOldKeyValues[identity];
              IF changeStrong THEN
                ChangeStrongKey[
                  identity, oldStrongKey
                  ! Auth.AuthenticationError, Auth.CallError => CONTINUE];
              ChangeSimpleKey[
                identity, oldSimpleKey
                ! Auth.AuthenticationError, Auth.CallError => CONTINUE];
              };

            ChangeSimpleKey[identity, newHashedPassword];
          END;

        -- If we got this far it worked; we can fix up identityObject.
        ChangeIdentityInfo[identity];
      END;

  CreateStrongKey: PUBLIC PROC [
      identity: Auth.IdentityHandle, name: NSName.Name, newStrongKey: Auth.Key] =
      BEGIN
        DoIt: PROC [
          creds: Auth.Credentials, verifier: Auth.Verifier, conv: Auth.ConversationHandle,
          asAddressPtr: LONG POINTER TO System.NetworkAddress] =
            BEGIN
              encryptedInitialKey: DESFace.Block;
              DESFace.EncryptBlock[
                key: PrivateConv[conv].conversationKey,
                from: LOOPHOLE[LONG[@newStrongKey]], to: @encryptedInitialKey];
              AuthOps.CreateStrongKey[
                creds, verifier, name, encryptedInitialKey, asAddressPtr];
            END;

        SELECT AuthInternal.Style[identity] FROM
          simple => ERROR Auth.AuthenticationError[inappropriateCredentials];
          strong => ConverseWithASAndCallMe[identity, name, DoIt];
        ENDCASE => ERROR;
      END;

  ChangeStrongKey: PUBLIC PROC [
      identity: Auth.IdentityHandle, newStrongKey: Auth.Key] =
      BEGIN
        ChangeIdentityInfo: ENTRY PROC [identity: Auth.IdentityHandle] =
          BEGIN
            ENABLE UNWIND => NULL;
            AuthInternal.FreeNSString[
              @PrivateIdent[identity].myPassword, PrivateIdent[identity].owningHeap];
            -- Leave password a nullString because it is no longer related to
            -- the "truth" in identity.myStrongKey.
            PrivateIdent[identity].myStrongKey _ PrivateKey[newStrongKey];
          END;
        DoIt: PROC [
          creds: Auth.Credentials, verifier: Auth.Verifier, conv: Auth.ConversationHandle,
          asAddressPtr: LONG POINTER TO System.NetworkAddress] =
            BEGIN
              encryptedNewKey: DESFace.Block;
              DESFace.EncryptBlock[
                key: PrivateConv[conv].conversationKey,
                from: LOOPHOLE[LONG[@newStrongKey]], to: @encryptedNewKey];
            AuthOps.ChangeStrongKey[creds, verifier, encryptedNewKey, asAddressPtr];
            END;

        SELECT AuthInternal.Style[identity] FROM
          simple => ERROR Auth.AuthenticationError[inappropriateCredentials];
          strong =>
            ConverseWithASAndCallMe[identity, PrivateIdent[identity].myName, DoIt];
        ENDCASE => ERROR;

        -- If we got this far it worked; we can fix up identityObject.
        ChangeIdentityInfo[identity];
      END;

  DeleteStrongKey: PUBLIC PROC [identity: Auth.IdentityHandle, name: NSName.Name] =
      BEGIN
        DoIt: PROC [
          creds: Auth.Credentials, verifier: Auth.Verifier, conv: Auth.ConversationHandle,
          asAddressPtr: LONG POINTER TO System.NetworkAddress] =
            BEGIN
              AuthOps.DeleteStrongKey[creds, verifier, name, asAddressPtr];
            END;

        SELECT AuthInternal.Style[identity] FROM
          simple => ERROR Auth.AuthenticationError[inappropriateCredentials];
          strong => ConverseWithASAndCallMe[identity, name, DoIt];
        ENDCASE => ERROR;
      END;

  CreateSimpleKey: PUBLIC PROC [
      identity: Auth.IdentityHandle, name: NSName.Name,
      newSimpleKey: Auth.HashedPassword] =
      BEGIN
        DoIt: PROC [
          creds: Auth.Credentials, verifier: Auth.Verifier, conv: Auth.ConversationHandle,
          asAddressPtr: LONG POINTER TO System.NetworkAddress] =
```

```
        BEGIN
          AuthOps.CreateSimpleKey[
            creds, verifier, name, newSimpleKey, asAddressPtr];
        END;

    ConverseWithASAndCallMe[identity, name, DoIt];
  END;

ChangeSimpleKey: PUBLIC PROC [
  identity: Auth.IdentityHandle, newSimpleKey: Auth.HashedPassword] =
  BEGIN
    ChangeIdentityInfo: ENTRY PROC [identity: Auth.IdentityHandle] =
      BEGIN
        ENABLE UNWIND => NULL;
        AuthInternal.FreeNSString[
          @PrivateIdent[identity].myPassword, PrivateIdent[identity].owningHeap];
        -- Leave password a nullString because it is no longer related to
        -- the "truth" in identity.myHashedPassword.
        PrivateIdent[identity].myHashedPassword ← newSimpleKey;
      END;
    DoIt: PROC [
      creds: Auth.Credentials, verifier: Auth.Verifier, conv: Auth.ConversationHandle,
      asAddressPtr: LONG POINTER TO System.NetworkAddress] =
        BEGIN
          AuthOps.ChangeSimpleKey[creds, verifier, newSimpleKey, asAddressPtr];
        END;

    ConverseWithASAndCallMe[identity, PrivateIdent[identity].myName, DoIt];

    -- If we got this far it worked; we can fix up identityObject.
    ChangeIdentityInfo[identity];
  END;

DeleteSimpleKey: PUBLIC PROC [identity: Auth.IdentityHandle, name: NSName.Name] =
  BEGIN
    DoIt: PROC [
      creds: Auth.Credentials, verifier: Auth.Verifier, conv: Auth.ConversationHandle,
      asAddressPtr: LONG POINTER TO System.NetworkAddress] =
        BEGIN
          AuthOps.DeleteSimpleKey[creds, verifier, name, asAddressPtr];
        END;

    ConverseWithASAndCallMe[identity, name, DoIt];
  END;


-- Credentials operations which call the server --


CheckSimpleCredentials: PUBLIC PROC [
  creds: Auth.Credentials, verifier: Auth.Verifier]
    RETURNS [ok: BOOLEAN ← FALSE] =
  BEGIN
    DoIt: PROC [asAddressPtr: LONG POINTER TO System.NetworkAddress] =
      BEGIN
        ok ← AuthOps.CheckSimpleCredentials[creds, verifier, asAddressPtr];
      END;

    IF creds.flavor # simple
      THEN ERROR Auth.AuthenticationError[credentialsInvalid];

    BEGIN
      simpleCreds: AuthProtocol.SimpleCredentials ←
        AuthInternal.UnpackSimpleCredentials[
          creds, AuthInternal.asStubHeap
          ! Courier.Error => ERROR Auth.AuthenticationError[credentialsInvalid]];
      -- The following calls the AS and sets ok:
      ContactASAndCallMe[
        DoIt, simpleCreds.initiator
        ! UNWIND =>
          AuthInternal.FreeSimpleCredentials[@simpleCreds, AuthInternal.asStubHeap]];
      AuthInternal.FreeSimpleCredentials[@simpleCreds, AuthInternal.asStubHeap];
    END;
  END;

FetchStrongCredentials: PUBLIC PROC [
  initiator, recipient: NSName.Name,
  initiatorsStrongKey: Auth.Key, z: UNCOUNTED ZONE]
    RETURNS [creds: Auth.Credentials, conversationKey: Auth.Key] =
  BEGIN
    DoIt: PROC [asAddressPtr: LONG POINTER TO System.NetworkAddress] =
      BEGIN
        credentialsPackage ←
          AuthOps.GetStrongCredentials[initiator, recipient, nonce, asAddressPtr];
      END;

    convKey: DESFace.Key ← DESFace.nullKey;
    credentialsPackage: AuthProtocol.CredentialsPackage;
    nonce: LONG CARDINAL ← System.GetClockPulses[];
    SELECT TRUE FROM
      initiator = NIL => ERROR Auth.CallError[strongKeyDoesNotExist, initiator];
      recipient = NIL => ERROR Auth.CallError[strongKeyDoesNotExist, recipient];
    ENDCASE;
    ContactASAndCallMe[DoIt, initiator]; -- fills in credentialsPackage
    BEGIN
      ENABLE UNWIND => Courier.Free[
        [@credentialsPackage, AuthProtocol.DescribeCredentialsPackage],
```

```
            AuthInternal.asStubHeap];

        [creds, convKey] __
          AuthInternal.ExtractStuffFromCredentialsPackage[
            credentialsPackage, PrivateKey[initiatorsStrongKey],
            recipient, nonce, z
            ! AuthInternal.BadCredentialsPackage = >
              ERROR Auth.CallError[badKey, notApplicable]];
        IF ~DESFace.CheckKeyParity[@convKey]
          THEN ERROR Auth.CallError[badKey, notApplicable];
      END;
      Courier.Free[
        [@credentialsPackage, AuthProtocol.DescribeCredentialsPackage],
        AuthInternal.asStubHeap];
      RETURN[creds, PublicKey[convKey]];
    END;


  - - Private stuff - -


  ConverseWithASAndCallMe: PROC [
    identity: Auth.IdentityHandle, nameHint: NSName.Name,
    procToCall: PROC [
      Auth.Credentials, Auth.Verifier, Auth.ConversationHandle,
      LONG POINTER TO System.NetworkAddress]] =
    BEGIN
      retrys: CARDINAL __ 0;
      asAddress: System.NetworkAddress __ System.nullNetworkAddress;
      creds: Auth.Credentials __ Auth.nullCredentials;
      verifier: Auth.Verifier __ Auth.nullVerifier;
      conv: Auth.ConversationHandle;

      conv __ Auth.Initiate[identity, SharedKeys.asName, , AuthInternal.asStubHeap];
      asAddress __ AuthServerCache.ASAddress[nameHint];
      BEGIN
        ENABLE UNWIND = > Auth.Terminate[@conv, AuthInternal.asStubHeap];

        DO
          [creds, verifier] __ Auth.CheckOutCredsAndNextVerifier[conv, asAddress.host];
          procToCall[
            creds, verifier, conv, @asAddress
            ! Auth.CallError = >
              IF (reason = cannotReachAS OR reason = tooBusy) AND
                retrys < timesToRetry
                  THEN
                    BEGIN
                      addressesLeftInCache: CARDINAL __
                        AuthServerCache.NextPlease[asAddress];
                      AuthServerCache.Refill[];
                      asAddress __ AuthServerCache.ASAddress[];
                      retrys __ retrys + 1;
                      LOOP;
                    END;
            Auth.AuthenticationError = >
              - - The reason must be credentialsInvalid; this is highly unusual!
              {ERROR Auth.CallError[badKey, notApplicable]};
          ];
          EXIT; - - Successful call
        ENDLOOP;
      END;
      Auth.Terminate[@conv, AuthInternal.asStubHeap];
    END;

  ContactASAndCallMe: PROC [
    procToCall: PROC [LONG POINTER TO System.NetworkAddress],
    nameHint: NSName.Name] =
    BEGIN
      retrys: CARDINAL __ 0;
      asAddress: System.NetworkAddress __ AuthServerCache.ASAddress[nameHint];

      IF cacheNeverFilled THEN
        BEGIN
          AuthServerCache.Refill[];
          cacheNeverFilled __ FALSE;
          IF asAddress = System.nullNetworkAddress THEN
            BEGIN
              Process.Pause[pauseForInitialCacheFill]; - - Wait only if we have to.
              asAddress __ AuthServerCache.ASAddress[];
            END;
        END;

      DO
        procToCall[
          @asAddress
          ! Auth.CallError = >
              IF (reason = cannotReachAS OR reason = tooBusy) AND
                retrys < timesToRetry
                  THEN
                    BEGIN
                      addressesLeftInCache: CARDINAL __
                        AuthServerCache.NextPlease[asAddress];
                      AuthServerCache.Refill[];
                      asAddress __ AuthServerCache.ASAddress[];
                      retrys __ retrys + 1;
                      LOOP;
                    END;
```

```
            ];
          EXIT; -- Successful call
        ENDLOOP;
    END;


-- Public/Private pointer conversions --


    -- Note: This is a hack to avoid exporting public types.
    -- (Because exported type clashes make it impossible for two
    -- different implementations which export the same type to co-exist.)

PrivateIdentityHandle: TYPE = LONG POINTER TO AuthInternal.IdentityObject;
PrivateConversationHandle: TYPE = LONG POINTER TO AuthInternal.ConversationObject;

PrivateConv: PROC [conversation: Auth.ConversationHandle]
    RETURNS [PrivateConversationHandle] = INLINE
      {RETURN[LOOPHOLE[conversation]]};

PrivateIdent: PROC [identity: Auth.IdentityHandle]
    RETURNS [PrivateIdentityHandle] = INLINE
      {RETURN[LOOPHOLE[identity]]};

PrivateKey: PROC [key: Auth.Key]
    RETURNS [DESFace.Key] = INLINE
      {RETURN[LOOPHOLE[key]]};

PublicKey: PROC [privateKey: DESFace.Key]
    RETURNS [Auth.Key] = INLINE
      {RETURN[LOOPHOLE[privateKey]]};


END.
```

```
$MOD186
$PAGELENGTH (72)
$PAGEWIDTH  (136)

;-- stored as MesaVM.asm
;-- created on  19-Jul-84 13:20:18

;-- last edited by:
;--     KEK     30-Apr-87 13:24:48      :zero out WHOLE IOR in case of not debugging with a FatIOR to support a prom image (bug!).
;--     KEK     22-Apr-87 11:32:26      :add use of VMMDefs.dovePROMSize to support expanded IOR during debugging.
;--     kek     14-Apr-87 15:31:38      :add Daybreak-only MDS relief.  For Daisy this is still non-MDS-relieved!
;--     RDH     17-Feb-87 15:06:57      :Use masks from vmmdefs to correctly prevent display memory from being cleared.
;--     JAC     23-Jan-87 12:34:45      :initialize aChipCount for Daybreaks
;--     KEK     26-Sep-86 13:55:42      :another bug in AS conversion (off-by-one)
;--     kek     23-Jun-86 16:48:47      ;add ASCheckPage for multi AChip support.
;--     kek     27-May-86 13:18:20      ;add mem zeroing stuff for parity initing.
;--     kek     12-May-86 12:52:08      ;add new Daisy stuff
;--     JPM     20-Aug-85 12:46:45      ;Fix bugs in Search*Memory.
;--     JPM     12-Aug-85 14:31:53      ;Search for memory limits if EEProm bad.
;--     JPM      5-Aug-85 14:05:18      ;EEProm changes.
;--     JPM     29-Jul-85 11:22:59      ;Opie redesign conversion.
;--     JMM     20-Jun-85 10:16:05      ;Upgrades.
;--     JMM      4-Apr-85 15:33:07      ;Upgrades.
;--     JMM     20-Feb-85 10:24:19      ;First release.

NAME                    MesaVM
;-------------------------------------------------------------------------------
;
$NOLIST
$INCLUDE                (HardDefs.asm)
$INCLUDE                (IOPDefs.asm)
$INCLUDE                (ROMEEP.asm)
$INCLUDE                (RAMEEP.asm)
$INCLUDE                (IOPMacro.asm)
$INCLUDE                (VMMDefs.asm)
$LIST



;-------------------------------------------------------------------------------

;*******************************************************************************

;-------------------------------------------------------------------------------
IOPELocalRAM            SEGMENT AT 0

EXTRN                   VMMFirstPage: WORD, VMMSizeInPages: WORD
EXTRN                   firstRealPageInVM: WORD, lastRealPageInVM: WORD
EXTRN                   countRealPagesInVM: WORD
EXTRN                   firstDisplayBankPage: WORD, countDisplayBankPages: WORD
EXTRN                   IOROpieSegmentAddress: WORD
EXTRN                   aChipCount: BYTE
EXTRN                   prebootSwitches: WORD
needFatIOR              EQU     4000H

IOPELocalRAM            ENDS

;-------------------------------------------------------------------------------

BootStrapIOR            SEGMENT         COMMON

EXTRN                   loaderVirtualMemoryLocation  :DWORD ;from IORRAMBt.asm

BootStrapIOR            ENDS

;-------------------------------------------------------------------------------

OpieIOR                 SEGMENT         COMMON

EXTRN                   mesaPageMapOffset: WORD
EXTRN                   mesaPageMapSegment: WORD

OpieIOR                 ENDS

;-------------------------------------------------------------------------------

DisplayIOR                      SEGMENT         COMMON

EXTRN                   bitMapOrg: WORD

DisplayIOR                      ENDS

;-------------------------------------------------------------------------------

;*******************************************************************************


IOPEInRAM               SEGMENT         PUBLIC

PUBLIC                  MesaVM          ;jmm:84-11-27:debug only.

                        Assume CS:IOPEInRAM
                        Assume DS:BootStrapIOR



MesaVM.asm        30-Apr-87 13:25:22 PDT
```

```
;-------------------------------------------------------------------------------
;Local Constants:
;----------------


;Register equates:

currentRealPage          EQU     BX
IndexToCurrentVirtualPage EQU    DI
currentVirtualPage       EQU     SI
request                  EQU     SI
savedContents            EQU     SI
pagesPerBank             EQU     BP

;Local Variables:
;----------------

systemMemDesc            DW      0               ;From EEPROM
expansionMemDesc         DW      0               ;From EEPROM
sizeOfVMM                DW      0               ;Calculated from EEPROM entry
locationOfVMM            DW      0               ;depends on machine type & mem size
sizeOfIORegion           DW      0               ;fixed (for now)
locationOfIORegion       DW      0               ;depends on machine type
sizeOfIORegionCopy       DW      0               ;these are needed for formatting
locationOfIORegionCopy   DW      0               ; the virtual IORegion
sizeOfDisplayMemory      DW      0               ;From EEPROM
locationOfDisplayMemory  DW      0               ;depends on machine type
firstRealPage            DW      -1              ;found during VMM load
lastRealPage             DW      0               ;found during VMM load
countRealPages           DW      0               ;found during VMM load
sizeOfPageInBytes        DW      pageSizeInBytes
sizeOfPageInWords        DW      pageSizeInWords
localPrebootSwitches     DW      0

;-------------------------------------------------------------------------------



;-- Virtual Memory Initialization:
;--------------  Assume the following upon entry into this procedure:
;--------------
;--------------|  Information:  Pilot uses the virtual memory map (VMM) to      |
;--------------|  figure out how much memory it has got. We therefore need to   |
;--------------|  enter all the memory available for Pilot's use into the VMM.  |
;--------------|  Note that this would therefore preclude display memory and of |
;--------------|  course the memory occupied by the VMM itself. i.e. All memory |
;--------------|  is mapped except display memory and the virtual memory map.   |
;--------------|  Display memory ends up being mapped by "UserTerminalHeadDove" |
;--------------|                                                                |
;--------------|  Further so that the IOP does not have to go through the VMM   |
;--------------|  to access real (this was not even physically possible with    |
;--------------|  the DLion) memory the IORegion (IOPage in DLion) is at a      |
;--------------|  well known location in real memory and also in a well known   |
;--------------|  location in virtual memory.                                   |
;--------------|                                                                |
;--------------|                                                                |
;--------------|                                                                |
;--------------|--------                                                        |
;--------------| Upon exiting this procedure the following will be true:        |
;--------------|                                                                |
;--------------|        mesaVMMMapRegister = beginning of 128KbBank containing  |
;--------------|                             the VMM.                           |
;--------------|        loaderVirtualMemoryLocation = 24-bit Opie address of    |
;--------------|                        type "mesaLogicalPageOpieAddress"       |
;--------------|                        where the Germ is to be loaded.         |
;--------------|                                                                |
;-------------------------------------  ---------------------------------------


MesaVM                   PROC    FAR

EnterMesaVM:

                         %ReadEEProm(eePromLowMem,1)
                         JNC     StoreLowMemDesc                 ;if can't trust EEPROM,
                         CALL    SearchLowMemory                 ; try to figure it out
StoreLowMemDesc:         MOV     CS: systemMemDesc, AX
                         %ReadEEProm(eePromHighMem,1)
                         JNC     MakeHighMemDesc                 ;if can't trust EEPROM,
                         CALL    SearchHighMemory                ; try to figure it out
MakeHighMemDesc:         MOV     CX, 16                          ;each count in AX
                         SUB     CX, AX                          ;becomes one bit
                         MOV     AX, 0FFFFH                      ;in mem desc word
                         JS      StoreHighMemDesc                ;(up to 16)
                         SHR     AX, CL
StoreHighMemDesc:        MOV     CS: expansionMemDesc, AX
                         MOV     AX, IOPELocalRAM
                         MOV     ES, AX
                         ASSUME  ES:IOPELocalRAM
                         %ReadEEProm(eePromMemSize,1)            ;jmm:84-12-26:MakeWordForVMMPageCount
                         JNC     CalcSizeOfVMM                   ;if can't trust EEPROM,
                         MOV     AL, 1                           ;use 22 bit VM
CalcSizeOfVMM:           AND     AL, 3                           ;get encoded VM size
                         MOV     CL, AL                          ;prepare to shift


```

```
                        MOV     AX, 32                          ;calculate number
                        SHL     AX, CL                          ;of VMM pages
                        MOV     CS: sizeOfVMM, AX               ;and store
                        MOV     VMMSizeInPages, AX
                        MOV     AX, prebootSwitches
                        MOV     CS: localPrebootSwitches, AX

;-----------------------

InitializeVMMSetUpVariables:
                        TEST    CS: localPrebootSwitches, needFatIOR
                        JNZ     fattenIOR
        skinnyIOR:      MOV     CS: sizeOfIORegion, doveSlimIORegionSize
                        MOV     CS: sizeOfIORegionCopy, doveSlimIORegionSize
                        JMP     doneWithIOR
        fattenIOR:      MOV     CS: sizeOfIORegion, doveFatIORegionSize
                        MOV     CS: sizeOfIORegionCopy, doveFatIORegionSize
        doneWithIOR:
                                                                ;We need to know what
                        IN      AX, machineIDPort               ;machine we are on because
                        AND     AX, machineIDMask               ;VMM, IORegion and display
                        CMP     AX, Daisy                       ;are in different places for
                        JE      DaisyInitialization             ;different configurations.
                        JMP     DaybreakInitialization
DaisyInitialization:    CMP     CS: expansionMemDesc, 0
                        JNZ     FatDaisyVariables
SlimDaisyVariables:     MOV     CS: locationOfVMM, slimDaisyVMMBasePage
                        MOV     VMMFirstPage, slimDaisyVMMBasePage
                        MOV     CS: locationOfIORegion, slimDaisyIORegionBasePage
                        MOV     CS: locationOfIORegionCopy, slimDaisyIORegionBasePage
                        MOV     CS: locationOfDisplayMemory, slimDaisyDisplayBasePage
                        MOV     firstDisplayBankPage, slimDaisyDisplayBasePage
                        MOV     CS: sizeOfDisplayMemory, slimDaisyDisplayMemSize
                        MOV     countDisplayBankPages, slimDaisyDisplayMemSize
                        AND     CS: systemMemDesc, slimDaisyDisplayDescMask     ;don't put display mem in VM
                        ;the AChip was originally programmed assuming a bitMapOrg of zero,
                        ; as would be the case for fatDaisy.  Since it is slimDaisy,
                        ; it must be reprogrammed.
                        ;Note that I am NOT copying the bitMap contents to the new bitMapOrg.
                        ; As a consequence, the bitmap will display trash after this executes.
                        PUSH    ES
                        MOV     AX, DisplayIOR  ;display IOR bitMapOrg.
                        MOV     ES, AX
                        ASSUME  ES: DisplayIOR
                        MOV     ES: bitMapOrg, 768
                        POP     ES
                        ASSUME  ES: IOPELocalRAM
                        MOV     DX, 08C0H       ;AChip.BaseP bitMapOrg.
                        MOV     AX, 0C000H
                        OUT     DX, AX
                        MOV     aChipCount, 1                   ;set actual aChipCount.
                        MOV     DX, slimDaisyDisplayBasePage    ;(for later display bitmap zeroing)
                        MOV     BP, slimDaisyDisplayEndPage
                        JMP     FindDaisyDisplaySize
FatDaisyVariables:      MOV     CS: locationOfVMM, fatDaisyVMMBasePage
                        MOV     VMMFirstPage, fatDaisyVMMBasePage
                        MOV     CS: locationOfIORegion, fatDaisyIORegionBasePage
                        MOV     CS: locationOfIORegionCopy, fatDaisyIORegionBasePage
                        MOV     CS: locationOfDisplayMemory, fatDaisyDisplayBasePage
                        MOV     firstDisplayBankPage, fatDaisyDisplayBasePage
                        MOV     countDisplayBankPages, fatDaisyDisplayMemSize
                        MOV     CS: sizeOfDisplayMemory, fatDaisyDisplayMemSize
                        AND     CS: systemMemDesc, fatDaisyDisplayDescMask      ;don't put display mem in VM
                        ;the AChip was originally programmed assuming a one-ASID situation.  Since
                        ; there is more than one AID, it they must be reprogrammed for more
                        ; than one ASID.
                        MOV     DX, 082EH       ;AID0 ← ASID1
                        MOV     AX, 0001H
                        OUT     DX, AX
                        INC     DH              ;AID1 ← ASID0
                        DEC     AX
                        OUT     DX, AX
                        INC     DH              ;AID2 ← ASID2
                        MOV     AX, 0002H
                        OUT     DX, AX
                        INC     DH              ;AID3 ← ASID3
                        MOV     AX, 0003H
                        OUT     DX, AX
                        MOV     DX, fatDaisyDisplayBasePage     ;(for later display bitmap zeroing)
                        MOV     BP, fatDaisyDisplayEndPage
                        MOV     AX, CS: expansionMemDesc        ;calculate actual aChipCount.
                        MOV     aChipCount, 2   ; 2 chips!
                        SHR     AX, 4
                        JZ      FindDaisyDisplaySize
                        INC     aChipCount      ; 3 chips!
                        SHR     AX, 4
                        JZ      FindDaisyDisplaySize
                        INC     aChipCount      ; 4 chips!
FindDaisyDisplaySize:   IN      AL, DaisyDisplayTypePort
                        AND     AL, DaisyDisplayTypeMask
                        CMP     AL, DaisynineteenInch
                        JMP     SHORT DisplaySizeFound
DaybreakInitialization: MOV     CS: locationOfVMM, fatDaybreakVMMBasePage
                        MOV     VMMFirstPage, fatDaybreakVMMBasePage
                        MOV     CS: locationOfIORegion, fatDaybreakIORegionBasePage
                        MOV     CS: locationOfIORegionCopy, fatDaybreakIORegionBasePage
                        MOV     CS: locationOfDisplayMemory, fatDaybreakDisplayBasePage
```

```
                        MOV     aChipCount, 0                   ;daybreak has no a chips
                        MOV     firstDisplayBankPage, fatDaybreakDisplayBasePage
                        TEST    CS: systemMemDesc, 0FCH
                        MOV     BYTE PTR CS: systemMemDesc, 0   ;don't put display mem in VM
                        JNZ     FatDaybreakVariables
                        MOV     CS: sizeOfDisplayMemory, slimDaybreakDisplayMemSize
                        MOV     countDisplayBankPages, slimDaybreakDisplayMemSize
                        JMP     SHORT FindDaybreakDisplaySize
FatDaybreakVariables:   MOV     CS: sizeOfDisplayMemory, fatDaybreakDisplayMemSize
                        MOV     countDisplayBankPages, fatDaybreakDisplayMemSize
FindDaybreakDisplaySize:MOV     DX, DisplayTypePort
                        IN      AL, DX
                        MOV     DX, fatDaybreakDisplayBasePage;(for later display bitmap zeroing)
                        MOV     BP, fatDaybreakDisplayEndPage
                        TEST    AL, DisplayTypeMask
;------------------------

; zero memory from end of display bitmap to end of display memory bank.
; enter here with start of display bitmap in DX, and the zero flag set if 19".
; oh, also BP needs to be pointing to the last page of the display memory.
DisplaySizeFound:       JZ      fatDisplay
        slimDisplay:    ADD     DX, pagesFor15InchDisplay
                        JMP     InitDisplayMemory
        fatDisplay:     ADD     DX, pagesFor19InchDisplay
InitDisplayMemory:      MOV     AX, DX
                        CALL    ZeroPage
                        INC     DX
                        CMP     DX, BP
                        JB      InitDisplayMemory

;------------------------

MemorySizeFound:        TEST    CS: localPrebootSwitches, needFatIOR
                        JNZ     setUpFatIOR
        setUpSmallIOR:  MOV     BX, (doveSlimIORegionSize SHL 8)        ;word count
                        JMP     IORSetUp
        setUpFatIOR:    MOV     BX, ((doveFatIORegionSize-dovePROMSize) SHL 8)
; dovePROMSize is subtracted from the IORegionSize above so that any
; debugging prom image present is not erased, in the setUpFatIOR case.
; In the setUpSmallIOR case, of course, there is no prom image present since it is
; not a debugging session by definition...!
IORSetUp:               MOV     DX, CS: locationOfIORegion
                        MOV     CX, (extendedBusPageOpieAddress SHL 8)
                        %EstablishIOPAccess(IORegionMapRegister,CX-DX)
                        ASSUME  ES:NOTHING
ZeroIOR:                MOV     CX, BX                  ;the size of IOR to be  0'ed...
                        XOR     AX, AX
                        CLD
                        REP     STOSW
                        MOV     DX, CS: locationOfVMM
                        MOV     CX, (extendedBusPageOpieAddress SHL 8)
                        %EstablishIOPAccess(mesaVMMapRegister,CX-DX)
                        ASSUME  ES:NOTHING

;------------------------

                        PUSH    ES
                        PUSH    DI
                        MOV     CX, CS: sizeOfVMM
StampAllPagesVacant:    MOV     DX, pageSizeInWords
StampAllPgsThisPg:      MOV     WORD PTR ES: [IndexToCurrentVirtualPage], (pageVacantMask SHL 8)
                        INC     IndexToCurrentVirtualPage
                        INC     IndexToCurrentVirtualPage
                        JNZ     StayInThis64KbBank
                        MOV     AX, ES
                        ADD     AX, crossover64KbBank
                        MOV     ES, AX
StayInThis64KbBank:     DEC     DX
                        JNZ     StampAllPgsThisPg
                        LOOP    StampAllPagesVacant
                        POP     DI
                        POP     ES

;------------------------

                        PUSH    ES
                        PUSH    DI
                        MOV     currentVirtualPage, 0
MapSystemPages:         MOV     currentRealPage, 0
                        MOV     DX, CS: systemMemDesc
                        MOV     pagesPerBank, numberOfPagesIn64KbBank
CheckIfDone:            CMP     DX, 0
                        JNE     CheckIfThisBankPresent
                        JMP     IsVMSetUpDone                           ;no more units
CheckIfThisBankPresent: MOV     CX, pagesPerBank                        ;number of pages in a bank
                        RCR     DX, 1
                        JC      PageMappingLoop                         ;found a bank
                        ADD     currentRealPage, CX                     ;skip over this bank
                        JMP     CheckIfDone

PageMappingLoop:        CMP     currentRealPage, CS: locationOfVMM
                        JNE     IsItDisplayMemRealPage
                        CMP     CX, CS: sizeOfVMM
                        JGE     SkipVMM
                        ADD     CS: locationOfVMM, CX
                        SUB     CS: sizeOfVMM, CX
SkipPartOfArea:         ADD     currentRealPage, CX
```

```
                        JMP     CheckIfDone
SkipVMM:                ADD     currentRealPage, CS: sizeOfVMM
                        SUB     CX, CS: sizeOfVMM
ShouldWeContinue:       JNZ     PageMappingLoop
                        JMP     CheckIfDone
IsItDisplayMemRealPage: CMP     currentRealPage, CS: locationOfDisplayMemory
                        JNE     IsItIORegionRealPage
                        CMP     CX, CS: sizeOfDisplayMemory
                        JGE     SkipDisplay
                        ADD     CS: locationOfDisplayMemory, CX
                        SUB     CS: sizeOfDisplayMemory, CX
                        JMP     SkipPartOfArea
SkipDisplay:            ADD     currentRealPage, CS: sizeOfDisplayMemory
                        SUB     CX, CS: sizeOfDisplayMemory
                        JMP     ShouldWeContinue
IsItIORegionRealPage:   CMP     currentRealPage, CS: locationOfIORegion
                        JNE     IsItIORegionVirtualPage
                        CMP     CX, CS: sizeOfIORegion
                        JGE     SkipIORegion
                        ADD     CS: locationOfIORegion, CX
                        SUB     CS: sizeOfIORegion, CX
                        JMP     SkipPartOfArea
SkipIORegion:           ADD     currentRealPage, CS: sizeOfIORegion
                        SUB     CX, CS: sizeOfIORegion
                        JMP     ShouldWeContinue

;this is here to shorten the backward jumps
GoToNextPage:           INC     currentVirtualPage
                        INC     currentRealPage
                        INC     countRealPages
                        LOOP    PageMappingLoop
                        JMP     CheckIfDone

IsItIORegionVirtualPage:
                        TEST    CS: localPrebootSwitches, needFatIOR
                        JNZ     needFatIORVM
        needSlimIORVM:  CMP     currentVirtualPage, slimIORegionFirstVirtualPage
                        JMP     doneIORVM
        needFatIORVM:   CMP     currentVirtualPage, fatIORegionFirstVirtualPage
        doneIORVM:      JNE     KeepMapping
                        PUSH    currentRealPage
                        PUSH    CX
                        MOV     CX, CS: sizeOfIORegionCopy
                        MOV     currentRealPage, CS:locationOfIORegionCopy

MapIORegion:            MOV     AX, currentRealPage
                        OR      AX, (pagePresentMask SHL 8)     ;the map and a (2) 3 bit mask.
                        CALL    ASCheckPage
                        MOV     ES: [IndexToCurrentVirtualPage], AX ;() for large memory machines.
                        INC     IndexToCurrentVirtualPage       ;Point to the next virtual
                        INC     IndexToCurrentVirtualPage       ;page and also to the next
                        INC     currentRealPage                 ;real page.
                        INC     currentVirtualPage
                        INC     CS: countRealPages
                        LOOP    MapIORegion
                        POP     CX
                        POP     currentRealPage                 ;We had saved this page.

KeepMapping:            MOV     AX, currentRealPage             ;into the VMM and mask them
                        CALL    ZeroPage
                        OR      AX, (pagePresentMask SHL 8)     ;the map and a (2) 3 bit mask.
                        CALL    ASCheckPage
                        MOV     ES: [IndexToCurrentVirtualPage], AX ;() for large memory machines.
                        INC     IndexToCurrentVirtualPage       ;Byte swap was for mesa!
                        INC     IndexToCurrentVirtualPage       ;Point to the next virtual
                        JNZ     StayInCurrent64KbBank           ;page and also to the next
                        MOV     AX, ES                          ;real page. Don't forget to
                        ADD     AX, crossover64KbBank           ;test for end of real memory.
                        MOV     ES, AX
StayInCurrent64KbBank:
                        MOV     CS: lastRealPage, currentRealPage
                        CMP     CS: firstRealPage, -1
                        JNE     GoToNextPage
                        MOV     CS: firstRealPage, currentRealPage
                        JMP     GoToNextPage

IsVMSetUpDone:          CMP     pagesPerBank, numberOfPagesIn256KbBank
                        JE      VMSetUpDone
                        MOV     currentRealPage, numberOfPagesInOneMb
                        MOV     DX, CS: expansionMemDesc
                        MOV     pagesPerBank, numberOfPagesIn256KbBank
                        JMP     CheckIfDone

;------------------------


VMSetUpDone:            MOV     AX, IOPELocalRAM
                        MOV     ES, AX
                        ASSUME  ES:IOPELocalRAM
                        MOV     AX, CS: firstRealPage
                        MOV     firstRealPageInVM, AX
                        MOV     AX, CS: lastRealPage
                        MOV     lastRealPageInVM, AX
                        MOV     AX, CS: countRealPages
                        MOV     countRealPagesInVM, AX
                        IN      AX, machineIDPort               ;the following MDS-relief
                        AND     AX, machineIDMask               ; stuff is for Daybreak only.
```

```
                        CMP     AX, Daisy
                        JE      InitializeDaisyGermVM
InitializeDaybreakGermVM:
                        ;(For MDS relief, we first load a one-page GFT into
                        ;the page before the germ. GermInit will move it later.)
                        MOV     WORD PTR loaderVirtualMemoryLocation[0], loaderVirtualPage
                        JMP     VMSetUpDoneContinued
InitializeDaisyGermVM:
                        MOV     WORD PTR loaderVirtualMemoryLocation[0], germVirtualPage
VMSetUpDoneContinued:
                        MOV     WORD PTR loaderVirtualMemoryLocation[2], (mesaLogicalPageOpieAddress SHL 8)
                                                        ;A virtual address
                                                        ;has the page value appearing in
                                                        ;the low sixteen bits of the
                                                        ;address. Here we forward the
                                                        ;Germ's virtual address.
                        MOV     ES, IOROpieSegmentAddress
                        ASSUME  ES:OpieIOR
                        POP     mesaPageMapOffset
                        POP     mesaPageMapSegment

                        RET

MesaVM                  ENDP
                        ASSUME  ES:NOTHING

;-------------------------------------------------------------------------------

;This zeros out the page of real memory pointed to by AX.

;This proc is used to initialize all non-special memory
; (non-display-bitmap/non-VMM/non-IOR).  This is needed to make sure all memory that
; will be used by Pilot later on has been pre-touched, to ensure that the parity
; bit of all non-special memory is inited to show no parity errors.  IE, this code
; prevents parity errors on these pages if they are ever read-accessed before they
; are written.

ZeroPage:               PUSHA
                        PUSH    ES
                        MOV     DX, AX
                        MOV     CX, (extendedBusPageOpieAddress SHL 8)
                        %EstablishIOPAccess(generalMapRegister,CX-DX)
                        MOV     CX, pageSizeInWords
                        XOR     AX, AX
                        CLD
                        REP     STOSW
                        POP     ES
                        POPA
                        RET
;-------------------------------------------------------------------------------

;This adapts the page number in AX about to be mapped to the Daisy A-S
; interface **if applicable**.

;IF the machine is a Daisy, AND the machine has more than one AChip,
; AND the page address is within the first two megabytes of real memory,
; THEN the rules of the Achip-Schip interface apply.
;If applicable, then if the page address is in the first meg then add 1M to it
; else if the page is in the second meg of address space then subtract 1M from it.
;This accomodation to the AS interface is necessary to make the pages mapped
; in the VMM match the backwards Schip point of view.  The reverse of these reversals
; is done in IOPLMap.ConvertAddress in Opie when it reads pages from the VMM.

ASCheckPage:            PUSH    ES
                        PUSH    AX
                        XOR     AX, AX
                        MOV     ES, AX
                        ASSUME  ES:IOPELocalRAM
                        POP     AX
                        CMP     ES:aChipCount, 1        ;(0=>Daybreak)
                        JBE     ASCheckPageDone         ;it's a mult chip Daisy!
                        ;add 1Mb if real addr in first meg.
ASTestFirstMeg:         CMP     AX, 7FFH        ;1Meg in pages
                        JA      ASTestSecondMeg
                        ADD     AX, 800H
                        JMP     ASCheckPageDone
                        ;sub 1Mb if real addr in second meg.
ASTestSecondMeg:        CMP     AX, 1000H       ;3Meg in pages
                        JAE     ASCheckPageDone
                        SUB     AX, 800H
ASCheckPageDone:        POP     ES
                        ASSUME  ES:NOTHING
                        RET

;-------------------------------------------------------------------------------

SearchLowMemory         PROC    NEAR
; Figure out what banks are present in the first 1MB
; Algorithm:
;       Assume bank 0 is good, since it exists on all configurations
;       Look at first and last page of each bank (128 pages/bank)
;       See if memory will return value stored
;       See if bank 0 clobbered by store (incomplete decoding)
;       Check banks 1 - 15
;       Return good bank mask in AX
                        MOV     SI, 8000H       ;assume bank 0 present
                        MOV     DX, 0
```

```
                        MOV     CX, (extendedBusPageOpieAddress SHL 8)
                        %EstablishIOPAccess(mesaVMMapRegister,CX-DX)     ;use this as temporary
                        MOV     WORD PTR ES:[DI], 0
                        MOV     BP, ES
                        MOV     currentRealPage, numberOfPagesIn64KbBank
CheckThisLowBank:       SHR     SI, 1
                        MOV     DX, currentRealPage
                        MOV     CX, (extendedBusPageOpieAddress SHL 8)
                        %EstablishIOPAccess(generalMapRegister,CX-DX)
                        MOV     WORD PTR ES:[DI], 0
                        CMP     WORD PTR ES:[DI], 0
                        JNE     LowBankNotPresent
                        MOV     WORD PTR ES:[DI], 0FFFFH
                        CMP     WORD PTR ES:[DI], 0FFFFH
                        JNE     LowBankNotPresent
                        MOV     ES, BP
                        CMP     WORD PTR ES:[DI], 0
                        JE      CheckThisLowBankLastPage
                        MOV     WORD PTR ES:[DI], 0
                        JMP     SHORT LowBankNotPresent
CheckThisLowBankLastPage:
                        MOV     DX, currentRealPage
                        ADD     DX, numberOfPagesIn64KbBank-1
                        MOV     CX, (extendedBusPageOpieAddress SHL 8)
                        %EstablishIOPAccess(generalMapRegister,CX-DX)
                        MOV     WORD PTR ES:[DI], 0
                        CMP     WORD PTR ES:[DI], 0
                        JNE     LowBankNotPresent
                        MOV     WORD PTR ES:[DI], 0FFFFH
                        CMP     WORD PTR ES:[DI], 0FFFFH
                        JNE     LowBankNotPresent
LowBankPresent:         OR      SI, 8000H
LowBankNotPresent:      ADD     currentRealPage, numberOfPagesIn64KbBank
                        CMP     currentRealPage, numberOfPagesInOneMb
                        JB      CheckThisLowBank
                        MOV     AX, SI
                        RET

SearchLowMemory         ENDP

;------------------------

SearchHighMemory        PROC    NEAR
; Figure out what banks are present after the first 1MB
; Algorithm:
;       Look at first and last page of each bank (512 pages/bank)
;       See if memory will return value stored
;       Stop at first failure or address = 4MB, whichever comes first
;       Return bank count in AX
                        MOV     currentRealPage, numberOfPagesInOneMb
                        MOV     SI, 0
CheckThisHighBank:      MOV     DX, currentRealPage
                        MOV     CX, (extendedBusPageOpieAddress SHL 8)
                        %EstablishIOPAccess(generalMapRegister,CX-DX)
                        MOV     WORD PTR ES:[DI], 0
                        CMP     WORD PTR ES:[DI], 0
                        JNE     HighBankNotPresent
                        MOV     WORD PTR ES:[DI], 0FFFFH
                        CMP     WORD PTR ES:[DI], 0FFFFH
                        JNE     HighBankNotPresent
                        MOV     DX, currentRealPage
                        ADD     DX, numberOfPagesIn256KbBank-1
                        MOV     CX, (extendedBusPageOpieAddress SHL 8)
                        %EstablishIOPAccess(generalMapRegister,CX-DX)
                        MOV     WORD PTR ES:[DI], 0
                        CMP     WORD PTR ES:[DI], 0
                        JNE     HighBankNotPresent
                        MOV     WORD PTR ES:[DI], 0FFFFH
                        CMP     WORD PTR ES:[DI], 0FFFFH
                        JNE     HighBankNotPresent
                        INC     SI
                        ADD     currentRealPage, numberOfPagesIn256KbBank
                        CMP     currentRealPage, hardwareMaxRealPage
                        JB      CheckThisHighBank
HighBankNotPresent:     MOV     AX, SI
                        RET

SearchHighMemory        ENDP

;------------------------


IOPEInRAM               ENDS

;*****************************************************************************


                        END
```

```
;  stored as [Iris]<WMicro>Dove>VMMDefs.asm
;-- It uses IORegion locations of the bootstrap handler.
;
;-- last edited by:

;--     KEK                     22-Apr-87 11:31:14      ;add dovePROMSize to support expanded IOR during debugging.
;--     kek                     14-Apr-87 15:31:38      ;add Daybreak-only MDS relief.  For Daisy this is still non-MDS-relieved!
;--     RDH                     30-Jan-87 18:07:37      ;Change fatDaisyDisplayMemSize to 1024 from 256, and add fatDaisyDisplayDescMask
and slimDaisyDisplayDescMask to not map Display bank correctly.
;--     KEK                     27-May-86 14:12:41      ;add *displayEndPage defs.
;--     KEK                      6-May-86 21:09:52      ;update Daisydefs to current.
;--     JPM                      3-Aug-85  9:01:07      ;Add numberOfPagesIn256KbBank and numberOfPagesInOneMb.
;--     RDH                      2-Aug-85 12:33:23      ;Added hardwareMaxRealPage to fix MEB prob.
;--     JPM                     29-Jul-85  9:58:31      ;Add labels to floppy germ request STRUC.
;--     RDH                     24-Jul-85 15:03:18      ;Removed magic numbers from floppy germ request.
;--     JMM                     22-Jul-85 15:02:29      ;Fixed floppy germ request.
;--     JMM                     20-Jun-85 15:54:51      ;Alt etherboot support.
;--     JPM                     29-May-85 12:48:13      ;Diagnostic boot changes.
;--     JMM                      4-Apr-85 15:46:44      ;Misc. updates.
;--     JMM                      4-Jan-85 17:24:58      ;First release.

;NAME                   VMMDefs
;-------------------------------------------------------------------------------
;
;For both Daisy and Daybreak, the virtual memory map is hardwired
;at main memory location 0-jmm:84-12-27:FixInfo. Also the IORegion is loaded in a well
;known real and virtual memory location (i.e. if it is decided to
;load it into real memory location "x" and virtual memory location
;"y", then the virtual memory map has to show that virtual memory
;location "y" is mapped to real memory location "x". The Germ as
;stipulated in the PrinceOps gets loaded starting at virtual page
;256 and its request section starts at virtual page 288.



germVirtualPage         EQU     0001H
germRequestVirtualPage  EQU     0003H
ethernetBootFileMask    EQU     000400
gftVirtualPage          EQU     0200H                   ;used by MDS-relief
loaderVirtualPage       EQU     germVirtualPage - 1     ;used by MDS-relief
location                EQU     0
value                   EQU     2


;Ethernet, Disk, Floppy, RS232C boot constants locations:
;--------------------------------------------------------

sFirstGermRequest                       EQU     00A0H   ;{start of germ request in SD}

Request@version                         EQU     (sFirstGermRequest + 00H)*2     ;double CP address
Request@action                          EQU     (sFirstGermRequest + 01H)*2     ;value to get IOP
Request@location@deviceType             EQU     (sFirstGermRequest + 02H)*2     ;address.
Request@location@devOrd                 EQU     (sFirstGermRequest + 03H)*2

ethernetBootFileNumber0@location        EQU     (sFirstGermRequest + 04H)*2
ethernetBootFileNumber1@location        EQU     (sFirstGermRequest + 05H)*2
ethernetBootFileNumber2@location        EQU     (sFirstGermRequest + 06H)*2

ethernetNetworkNumber0@location         EQU     (sFirstGermRequest + 07H)*2
ethernetNetworkNumber1@location         EQU     (sFirstGermRequest + 08H)*2

ethernetHostNumber0@Request@location    EQU     (sFirstGermRequest + 09H)*2
ethernetHostNumber1@Request@location    EQU     (sFirstGermRequest + 0AH)*2
ethernetHostNumber2@Request@location    EQU     (sFirstGermRequest + 0BH)*2

ethernetSocket@Request@location         EQU     (sFirstGermRequest + 0CH)*2

Request@location@cylinder@diskFileID@da EQU     (sFirstGermRequest + 09H)*2
Request@location@headSector@diskFileID@da EQU   (sFirstGermRequest + 0AH)*2


Request@floppyLocation@cylinder EQU     (sFirstGermRequest + 0BH)*2
Request@floppyLocation@headSector EQU   (sFirstGermRequest + 0CH)*2


;Ethernet, Disk, Floppy, RS232C boot constants:
;----------------------------------------------

RequestVersion                          EQU     034560
bootPhysicalVolume                      EQU     2       ;request action for disk
bootFloppyPhysicalVolume                EQU     0       ;request action for floppy
inLoad                                  EQU     0       ;request action for ethernet
germEthernet                            EQU     6       ;boot device for ethernet
germPilotDisk                           EQU     64      ;boot device for disk
DeviceTypes@sa800                       EQU     1       ;boot device for floppy
ethernetDeviceOrdinal                   EQU     0

ethernetBootFileNumberHigh              EQU     0000000
ethernetBootFileNumberMiddle            EQU     1250000
ethernetBootFileNumberLow               EQU     0040400


VMMDefs.asm         23-Apr-87 14:21:37 PDT
```

```
ethernetBootNetworkNumberHigh           EQU     0
ethernetBootNetworkNumberLow            EQU     0

ethernetHostNumberHigh                  EQU     0FFFFH
ethernetHostNumberMiddle                EQU     0FFFFH
ethernetHostNumberLow                   EQU     0FFFFH

ethernetBootSocket                      EQU     0000AH

onlyFloppy                              EQU     0
;----------------------------
;The parameters below are byte-swapped for Mesa.

%*DEFINE        (ByteSwap       (wordToSwap))
                (%wordToSwap SHL 8 OR %wordToSwap SHR 8)
;-----------------------------------------------

DiskGermVariables       STRUC

                        DW      Request@version, %ByteSwap (RequestVersion)
                        DW      Request@action, %ByteSwap (bootPhysicalVolume)
                        DW      Request@location@deviceType, %ByteSwap (germPilotDisk)
                        DW      Request@location@devOrd, %ByteSwap (0)


DiskGermVariables       ENDS

;----------------------------

EthernetGermVariables   STRUC

                        DW      Request@version, %ByteSwap (RequestVersion)
                        DW      Request@action, %ByteSwap (inLoad)
                        DW      Request@location@deviceType, %ByteSwap (germEthernet)
                        DW      Request@location@devOrd, %ByteSwap (ethernetDeviceOrdinal)
bootFileNumberHigh      DW      ethernetBootFileNumber0@location, %ByteSwap (ethernetBootFileNumberHigh)
bootFileNumberMiddle    DW      ethernetBootFileNumber1@location, %ByteSwap (ethernetBootFileNumberMiddle)
bootFileNumberLow       DW      ethernetBootFileNumber2@location, %ByteSwap (ethernetBootFileNumberLow)
                        DW      ethernetNetworkNumber0@location, %ByteSwap (ethernetBootNetworkNumberHigh)
                        DW      ethernetNetworkNumber1@location, %ByteSwap (ethernetBootNetworkNumberLow)
                        DW      ethernetHostNumber0@Request@location, %ByteSwap (ethernetHostNumberHigh)
                        DW      ethernetHostNumber1@Request@location, %ByteSwap (ethernetHostNumberMiddle)
                        DW      ethernetHostNumber2@Request@location, %ByteSwap (ethernetHostNumberLow)
                        DW      ethernetSocket@Request@location, %ByteSwap (ethernetBootSocket)

EthernetGermVariables   ENDS

;----------------------------

FloppyGermVariables     STRUC

                        DW      Request@version, %ByteSwap (RequestVersion)
                        DW      Request@action, %ByteSwap (bootFloppyPhysicalVolume)
                        DW      Request@location@deviceType, %ByteSwap (DeviceTypes@sa800)
                        DW      Request@location@devOrd, %ByteSwap (onlyfloppy)
bootFileCylinder        DW      Request@floppyLocation@cylinder, 0
bootFileHeadAndSector   DW      Request@floppyLocation@headSector, 0

FloppyGermVariables     ENDS

;-------------------------------------------------------------------------------
;
;
;
;------------------------

;BootTimeVariables      STRUC


;device                 DW      ?       ;1-Disk, 2-Ethernet, 3-Floppy, 4-RS232C.
;mode                   DW      ?       ;0-Normal, 1-Fast boot
;showUserInterface      DW      ?       ;0-yes, #0-no.
;bootRetryCount         DW      ?       ;initially 0, incremented per failure

;reincarnationFlag      DW      ?       ;For task reinitialization - 0 => tasks
                                        ;to use local RAM otherwise main memory.

;emulatorID             DW      3 DUP(?) ;Initially 0, updated by boot executive.
;loaderID               DW      3 DUP(?) ;Initially 0, updated by boot executive.

;diagnosticsType        DW      ?       ;Initialized at boot-time. 0 => short
                                        ;diagnostics - user has set EEPROM to
                                        ;indicate desire for a diagnostics boot.
                                        ;#0=> long diagnostics boot - user has
                                        ;manually requested diagnostics at boot
                                        ;time!

;BootTimeVariables      ENDS

;------------------------
;Device types:
;-------------

disk                    EQU     1
floppy                  EQU     2
ethernet                EQU     3
rs232C                  EQU     4
```

```
;-----------------------
;Boot types:
;-----------

normal                  EQU     0
diagnostic              EQU     1

;-----------------------


;Virtual Memory Constants:

maximumPageNumber               EQU     2000H   ;A 4Mbyte machine has this many pages.
pageSizeInBytes                 EQU     0200H   ;That is 512 bytes per page which
pageSizeInWords                 EQU     0100H   ;is 256 words per page.

;firstDovePage                  EQU     0       ;All Dove Main memory start at
firstDovePage                   EQU     1024    ;jmm:85-01-17:tmpFIX!!!
firstDaisyPage                  EQU     0
firstSlimDaybreakPage           EQU     1024

fatDaybreakVMMBasePage          EQU     1024    ;In decimal page numbers.
fatDaisyVMMBasePage             EQU     2048    ;1M from IOPside = 0 from SChipside.
slimDaisyVMMBasePage            EQU     0


fatDaybreakIORegionBasePage     EQU     1312    ;In decimal page numbers.
slimDaybreakIORegionBasePage    EQU     1312
fatDaisyIORegionBasePage        EQU     2336    ;In decimal page numbers.
slimDaisyIORegionBasePage       EQU     288

doveFirstBankCeiling            EQU     256     ;jmm:84-12-10In decimal page
dovePROMSize                    EQU     32      ;In decimal page numbers
doveSlimIORegionSize            EQU     64      ;jmm:84-12-10:numbers.
doveFatIORegionSize             EQU     doveSlimIORegionSize+dovePROMSize
slimIORegionFirstVirtualPage    EQU     doveFirstBankCeiling-doveSlimIORegionSize
fatIORegionFirstVirtualPage     EQU     doveFirstBankCeiling-doveFatIORegionSize
                                        ;jmm:84-12-10:fixAfterDemo.


doveDisplayBasePage             EQU     0       ;In decimal page numbers.
fatDaybreakDisplayBasePage      EQU     0       ;In decimal page numbers.
slimDaybreakDisplayBasePage     EQU     0       ;jmm:84-12-04:UntilGAPmoves.
fatDaisyDisplayBasePage         EQU     0
slimDaisyDisplayBasePage        EQU     768

;formula for the display-pages calculation taken from UserTerminalHeadDove
bitsPerWord                     EQU     16
pagesFor19InchDisplay           EQU     (1152/bitsPerWord*861+pageSizeInWords-1)/pageSizeInWords
pagesFor15InchDisplay           EQU     (832/bitsPerWord*633+pageSizeInWords-1)/pageSizeInWords

fatDaybreakDisplayMemSize       EQU     256     ;In decimal page numbers.
slimDaybreakDisplayMemSize      EQU     256
daisyDisplayMemSize             EQU     256
fatDaisyDisplayMemSize          EQU     1024    ;Half a meg
slimDaisyDisplayMemSize         EQU     256
fatDaisyDisplayDescMask         EQU     0FF00H  ;Half a meg
slimDaisyDisplayDescMask        EQU     0FF3FH  ;same size as Dybrk.

fatDaybreakDisplayEndPage       EQU     fatDaybreakDisplayBasePage+fatDaybreakDisplayMemSize
fatDaisyDisplayEndPage  EQU     fatDaisyDisplayBasePage+fatDaisyDisplayMemSize
slimDaisyDisplayEndPage EQU     slimDaisyDisplayBasePage+slimDaisyDisplayMemSize

fatDaisy                        EQU     0
fatDaybreak                     EQU     0

numberOfPagesIn64KbBank         EQU     128
numberOfPagesIn256KbBank        EQU     512
numberOfPagesInOneMb            EQU     2048
hardwareMaxRealPage             EQU     numberOfPagesInOneMb * 4 - 1    ;4 MB max

pageVacantMask                  EQU  01100000B  ;See "Sirius Microcode Reference"
pagePresentMask                 EQU  00000000B  ;Version 1.0 Pp. 6-1 & 6-3.

numberOfPagesPerMapRegAccess    EQU     100H    ;Given that each map register
numberOfPagesPerIndexRegAccess  EQU     80H     ;points to 128Kb of memory and
                                                ;that an 80186 register can access
                                                ;at most 64Kb of memory.
crossover64KbBank               EQU     01000H  ;


;-------------------------------------------------------------------------------
;Convenience macros:
;-------------------
;
%*Define (MultiplyByTwo (register))
        (ROL    %register, 1)
```

```
$MOD186
$PAGELENGTH (95)
$PAGEWIDTH  (136)

;-- stored as [Iris]<WMicro>Dove>GermInit.asm
;-- created on  28-Jan-85 14:47:51

;-- last edited by:
;--     kek             14-Apr-87 15:31:38      :add Daybreak-only MDS relief.  For Daisy this is still non-MDS-relieved!
;--     JPM             12-Aug-85  9:15:12      :Change IOPEInRAM alignment to WORD.
;--     JPM             26-Jul-85 13:40:25      ;Get floppy boot file address from local RAM.
;--     JMM             26-Jun-85 11:50:44      ;Alt etherboot support.
;--     JMM             16-Apr-85 18:03:17      ;Fixed e'net device type.
;--     JMM              4-Apr-85 15:47:11      ;First release.

NAME                    GermInit
;-------------------------------------------------------------------------------
;
$NOLIST
$INCLUDE                (HardDefs.asm)
$INCLUDE                (IOPDefs.asm)
$INCLUDE                (IOPMacro.asm)
$INCLUDE                (VMMDefs.asm)
$LIST


;-------------------------------------------------------------------------------

OpieIOR                 SEGMENT         COMMON

EXTRN                   mesaPageMapOffset: WORD
EXTRN                   mesaPageMapSegment: WORD

OpieIOR                 ENDS


;-------------------------------------------------------------------------------

IOPELocalRAM            SEGMENT AT 0

EXTRN                   device: WORD
EXTRN                   baseEthernetFileID: WORD
EXTRN                   floppyBootFileAddress: WORD

IOPELocalRAM            ENDS


;-------------------------------------------------------------------------------

;*******************************************************************************

IOPEInRAM               SEGMENT         WORD PUBLIC

PUBLIC                  GermInit

                        Assume CS:IOPEInRAM




;-------------------------------------------------------------------------------
;Local Constants:
;----------------


;Register equates:

request                 EQU     SI

;Local Variables:
;----------------


germRequestInit         DW      unKnownInitRequest, DiskInitRequest, FloppyInitRequest
                        DW      EthernetInitRequest, EthernetInitRequest

diskGermRequest         DiskGermVariables <>
ethernetGermRequest     EthernetGermVariables <>
floppyGermRequest       FloppyGermVariables <>
;-------------------------------------------------------------------------------



;-- Virtual Memory Initialization:
;--------------  Assume the following upon entry into this procedure:
;--------------
;--------------|                                                               |
;--------------|                                                               |
;--------------|                                                               |
;--------------|--------                                                       |
;--------------| Upon exiting this procedure the following will be true:       |
;--------------|                                                               |
;--------------|                                                               |
;-------------------------------------------------------------------------------

GermInit.asm        14-Apr-87 15:48:24 PDT
```

```
GermInit                PROC    NEAR

InitializeGermParamaters:
                        ASSUME  DS:OpieIOR
                        IN      AX, machineIDPort               ;the following MDS-relief stuff
                        AND     AX, machineIDMask               ; is for Daybreak only.
                        CMP     AX, Daisy
                        JE      InitializeCommonGermParamaters
InitializeDaybreakGermParamaters:
                        MOV     ES, mesaPageMapSegment          ;The germ's GFT is originally loaded
                        MOV     DI, mesaPageMapOffset           ;into the virtual page before the germ.
                        MOV     AX, ES:[DI]+2*loaderVirtualPage ;We must exchange its real page entry
                        XCHG    AX, ES:[DI]+2*gftVirtualPage    ;with that of the GFT location
                        MOV     ES:[DI]+2*loaderVirtualPage, AX ;before Mesa starts running.
InitializeCommonGermParamaters:
                        MOV     AX, IOPELocalRAM                ;We want to find out what device
                        MOV     ES, AX                          ;we are booting from and hence we
                        MOV     SI, ES: device                  ;need to look at boot variables
                        %MultiplyByTwo (SI)                     ;to know what device we booted from
                        MOV     DX, germRequestVirtualPage      ;but first let us establish access
                        XOR     CX, CX
                        MOV     CH, mesaLogicalPageOpieAddress  ;to the germ request section before
                        %EstablishIOPAccess(generalMapRegister,CX-DX) ;initializing its variables.
                        MOV     DX, DI
                        CALL    WORD PTR CS: germRequestInit[SI] ;
GermInitFinished:       RET

GermInit                ENDP
;-----------------------


unKnownInitRequest:     RET

DiskInitRequest:        MOV     CX, ((SIZE DiskGermVariables)/4)
                        MOV     request, OFFSET diskGermRequest
                        JMP     GermInitRequest

EthernetInitRequest:
                        MOV     AX, IOPELocalRAM
                        PUSH    ES
                        MOV     ES, AX
                        MOV     AX, ES: baseEthernetFileID
                        MOV     CS: ethernetGermRequest.bootFileNumberHigh+2, AX
                        MOV     AX, ES: baseEthernetFileID+2
                        MOV     CS: ethernetGermRequest.bootFileNumberMiddle+2, AX
                        MOV     AX, ES: baseEthernetFileID+4
                        MOV     CS: ethernetGermRequest.bootFileNumberLow+2, AX
                        OR      CS: ethernetGermRequest.bootFileNumberLow+2, %ByteSwap (ethernetBootFileMask)
                        POP     ES


                        MOV     CX, ((SIZE EthernetGermVariables)/4)
                        MOV     request, OFFSET ethernetGermRequest
                        JMP     GermInitRequest

FloppyInitRequest:      MOV     AX, IOPELocalRAM
                        PUSH    ES
                        MOV     ES, AX
                        MOV     AX, ES: floppyBootFileAddress
                        MOV     CS: floppyGermRequest.bootFileCylinder+2, AX
                        MOV     AX, ES: floppyBootFileAddress+2
                        MOV     CS: floppyGermRequest.bootFileHeadAndSector+2, AX
                        POP     ES


                        MOV     CX, ((SIZE FloppyGermVariables)/4)
                        MOV     request, OFFSET floppyGermRequest
                        JMP     GermInitRequest


GermInitRequest:        MOV     DI, DX                          ;Whenever these values change, we
                        OR      DI, CS: [request][location]     ;should reassemble this module to
                        MOV     AX, CS: [request][value]        ;reflect the changes in MDS 0 which
                        MOV     ES: [DI], AX                    ;is where the Germ variables are.
                        ADD     request, 4
                        LOOP    GermInitRequest                 ;Look at the *GermVariables to see
                                                                ;how this data structure is laid out.
                        RET

;-----------------------




IOPEInRAM               ENDS

;*******************************************************************************


                        END

GermInit.asm        14-Apr-87 15:48:24 PDT
```

```
$MOD186
$PAGELENGTH (95)
$PAGEWIDTH  (136)

;Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.

;-- stored as [Iris]<WMicro>Dove>RAMDskBt.asm
;-- created on  30-Nov-84 13:44:00

;-- last edited by:

;--     JPM             11-May-87 10:26:33      Add new MP codes for self-describing disk errors.
;--     JPM              8-May-87 12:45:13      Add delay for non-self-describing disks (so MP code is visible); reduce code size in
StuffNewDiskParms proc.
;--     JPM              7-May-87 10:00:49      Add code for self-describing disks (read sector and change disk parms).
;--     JPM              8-Jan-87 12:49:28      Fix bug in DiskReadError (needed to set BX before jump to InitIOCB).
;--     JPM             20-Sep-85 10:39:40      Do boot buffer allocation.
;--     kek              4-Sep-85 16:03:18      more mp codes, use public mp code rtn.
;--     JPM              4-Sep-85  8:16:12      Standardize MP codes, remove display change
;--     JPM              3-Aug-85 10:38:34      Change EEPDefs.asm to ROMEEP.asm
;--     JPM             22-Jul-85 12:37:34      Opie redesign conversion
;--     JPM             15-Jul-85  8:53:30      ;Set diskFCB.diskStartHandlerForIOP to FALSE after finished booting.
;--     JPM             12-Jul-85 12:03:29      ;Fixed bug in CalcNextAddr (inadequate check for double-zero).
;--     JMM              9-Jul-85 17:22:59      ;Upgraded to new Disk handler.
;--     JMM             26-Jun-85 21:12:43      ;Restored old EOF protocol and IOPMacro.
;--     JMM             26-Jun-85 16:40:07      ;Moved include files here. Also IOPLRAM.
;--     JPM             17-Jun-85 12:19:59

NAME                    RAMDskBt
;-------------------------------------------------------------------------------
;
$NOLIST
$INCLUDE                (HardDefs.asm)          ;
$INCLUDE                (IOPDefs.asm)           ;
$INCLUDE                (RAMBDefs.asm)          ;
$INCLUDE                (ROMEEP.asm)            ;
$INCLUDE                (DskIOFce.def)          ;
$INCLUDE                (DskBDefs.asm)
$INCLUDE                (IOPMacro.asm)
$LIST

EXTRN                   BootStrapHandlerID: ABS
EXTRN                   DiskHandlerID: ABS
EXTRN                   DisplayMPCode: NEAR

%*DEFINE                (Zero   (start,end))
                           LOCAL Lbl0
                           (    MOV     SI, OFFSET %start
                        %Lbl0:  MOV     BYTE PTR [SI], 0
                                INC     SI
                                CMP     SI, OFFSET %end
                                JL      %Lbl0
                                )

;-------------------------------------------------------------------------------

IOPELocalRAM    SEGMENT AT 0

EXTRN           opieReentry: DWORD
EXTRN           bootType: BYTE
EXTRN           bootRetryCount: WORD
EXTRN           startOfBootBufferSpace: WORD
EXTRN           HandlerInitProcTable: DWORD

IOPELocalRAM    ENDS

;-------------------------------------------------------------------------------

BootStrapIOR            SEGMENT COMMON
                        ASSUME  DS:BootStrapIOR

EXTRN                   bootBufferEmpty: Condition
EXTRN                   bootBufferFull: Condition
EXTRN                   startOfBootBufferPool: WORD
EXTRN                   bootDeviceIORSpace: DiskBootArea
EXTRN                   bootStrapTask: TaskContextBlock
EXTRN                   bootTask: TaskContextBlock
EXTRN                   getBootFile: Condition
EXTRN                   finishedLoaderFileFetch: Condition
dskIOCB                 EQU     bootDeviceIORSpace.diskBootIOCB
diskWorkspace           EQU     bootDeviceIORSpace

BootStrapIOR            ENDS

;-------------------------------------------------------------------------------

DiskIOR                 SEGMENT COMMON

EXTRN                   diskFCB: DiskFCBRecord

DiskIOR                 ENDS

;-------------------------------------------------------------------------------

MaintPanelIOR           SEGMENT COMMON

EXTRN                   maintPanelCode: WORD

RAMDskBt.asm        11-May-87 10:26:37 PDT
```

```
EXTRN                   maintPanelChanged: Condition

MaintPanelIOR           ENDS

;-------------------------------------------------------------------------------

BootStrapSTK            SEGMENT COMMON

EXTRN                   BootStack: WORD

BootStrapSTK            ENDS

;-------------------------------------------------------------------------------

IOPEInRam               SEGMENT PUBLIC
                        ASSUME  CS:IOPEInRam

PUBLIC                  RAMDiskEntryPoint

EXTRN                   BootTaskInit: NEAR
EXTRN                   EndOfInitial: NEAR

; Order of actions:
;       (1)     Load rest of DiskInitialDove.db (from cyl. 0, head 1, sectors 1-n,
;               or as directed by label fields)
;       (2)     Read disk shape page (from cyl. 0, head 0, sector 15)
;       (3)     Read root page (from cyl. 0, head 0, sector 0)
;       (4)     Invoke UI to select boot sequence
;       (5)     Load each file in turn, one buffer (= one sector) at a time
;       (6)     Idle this boot task forever


;-- Disk RAM Boot:
;--------------  This module reads in the rest of DiskInitialDove.db into
;--------------  IOP local RAM. It then proceeds to get the emulator file
;--------------  MesaDove.db and the germ Dove.germ in conjunction with the
;--------------  generic boot code.
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|--------                                                      |
;--------------| Upon exiting this module the following will be true:         |
;--------------|                                                              |
;--------------|                                                              |
;--------------|        bootStrapTask will be waiting forever on the          |
;--------------|        condition variable "idledBootTasks".                  |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;-------------------------------------------------------------------------------

                        ORG     0
RAMDiskEntryPoint:
; step 1
; set up ClientCondition in disk FCB
                        %EstablishHandlerAccess (DiskHandlerID)
                        ASSUME  ES:DiskIOR
                        MOV     diskFCB.rd0.diskIOPClientCondition.handlerID, LOW BootStrapHandlerID
                        MOV     diskFCB.rd0.diskIOPClientCondition.conditionPtr, OFFSET diskWorkspace.diskIOCBDone+nonNilPtr
                        MOV     diskFCB.rd0.diskIOPClientCondition.clientMask, 0
; set up disk IOCB (disk characteristics and data address)
                        MOV     BX, OFFSET dskIOCB.diskOperationBlock
                        MOV     AL, diskFCB.rd0.diskSectorsPerTrack
                        MOV     [BX].diskSecPerTrack, AL
                        MOV     AL, diskFCB.rd0.diskHeadsPerCylinder
                        MOV     [BX].diskHdsPerCyl, AL
                        MOV     AX, diskFCB.rd0.diskCylindersPerDrive
                        MOV     [BX].cylPerDrive, AX
                        MOV     AX, diskFCB.rd0.diskReducedWriteCurrentCylinder
                        MOV     [BX].diskReducedWriteCyl, AX
                        MOV     AX, diskFCB.rd0.diskPreCompensationCylinder
                        MOV     [BX].diskPreCompCyl, AX

; set up ES for access to boot variables
                        MOV     AX, IOPELocalRAM
                        MOV     ES, AX
                        ASSUME  ES:IOPELocalRAM
                        MOV     dskIOCB.diskDataPtrHigh, (IOPLogicalOpieAddress SHL 8)
                        MOV     BX, startOfBootBufferSpace
                        MOV     dskIOCB.diskDataPtrLow, BX
; change startOfBootBufferSpace to point past end of code
                        ADD     startOfBootBufferSpace, OFFSET EndOfInitial
; call DiskRead
GetRestOfInitial:       ADD     dskIOCB.diskDataPtrLow, sectorSize
                        CALL    DiskRead
                        JNC     GetRestOfInitial
                        JMP     RamDiskCont

; put DiskRead here, so will be in first sectorSize bytes

; proc to read from disk

DiskRead                PROC    NEAR
                        ASSUME  ES:IOPELocalRAM


RAMDsk8t.asm        11-May-87 10:26:37 PDT                                  2
```

```
; calculate next disk address from DOB (prev. address and label)
CalcNextAddr:           MOV     BX, OFFSET dskIOCB.diskOperationBlock
                        MOV     AX, WORD PTR [BX].diskLabel.diskDontCare
                        MOV     CX, WORD PTR [BX].diskLabel.diskDontCare+2
; if both words are 0, increment old address
                        MOV     DX, AX
                        OR      DX, CX
                        JZ      IncrPrevAddress
; if both words are FFFF, end-of file has been reached
;  otherwise, the words contain the next disk address
                        CMP     AX, 0FFFFH
                        JNE     UseAddressInAXCX
                        CMP     CX, 0FFFFH
                        JNE     UseAddressInAXCX
; end-of-file: set carry and return
                        STC
                        RET

; secondary entry point for new file -- disk address in AX:CX
DiskReadNewFile:        MOV     BX, OFFSET dskIOCB.diskOperationBlock
; AX:CX contains next sector address: store into DOB
UseAddressInAXCX:       MOV     WORD PTR [BX].diskHeader, AX
                        MOV     WORD PTR [BX].diskHeader+2, CX
                        JMP     SHORT InitIOCB
; label contains nil: increment previous address
IncrPrevAddress:        MOV     CX, WORD PTR [BX].diskHeader+2
                        INC     CL
                        CMP     CL, [BX].diskSecPerTrack
                        JL      StoreNewAddress
                        XOR     CL, CL
                        INC     CH
                        CMP     CH, [BX].diskHdsPerCyl
                        JL      StoreNewAddress
                        XOR     CH, CH
                        INC     [BX].diskHeader.diskCylinder
StoreNewAddress:        MOV     WORD PTR [BX].diskHeader+2, CX
; initialize other IOCB fields
InitIOCB:               MOV     AL, read
                        MOV     [BX].diskOperation, ReadDiskLabelAndData
InitIOCBCont:           %Zero   (dskIOCB.diskDataInfoRec, dskIOCB.diskOperationBlock)
                        MOV     dskIOCB.diskDataXferDirection, AL
                        MOV     dskIOCB.diskPageCount, 1
                        MOV     [BX].diskMinusSectorCount, -1
                        MOV     [BX].diskHeaderError, 0
                        MOV     [BX].diskLabelError, 0
                        MOV     [BX].diskDataError, 0
                        MOV     [BX].diskLastError, 0
; put IOCB onto disk queue
EnqueueIOCB:            PUSH    ES
                        %EstablishHandlerAccess (DiskHandlerID)
                        ASSUME  ES:DiskIOR
                        MOV     CH, IOPIORegionOpieAddress
                        MOV     CL, LOW BootStrapHandlerID
                        MOV     diskFCB.rd0.diskIOPNextLow, OFFSET dskIOCB
                        MOV     diskFCB.rd0.diskIOPNextHigh, CX
                        MOV     diskFCB.diskStartHandlerForIOP, 0FFFFH          ;jmm: Change to TRUE later.
                        POP     ES
                        ASSUME  ES:IOPELocalRAM
; wake up handler and wait for return notify
                        %NotifyHandlerCondition (DiskHandlerID,OFFSET diskFCB.diskConditionWork)
                        %WaitForCondition       (OFFSET diskWorkspace.diskIOCBDone,noTimeout)
; check for good completion
                        TEST    dskIOCB.diskError, 0FFH
                        JNZ     DiskReadError
DiskReadDone:           CLC
                        RET
; determine error type (eventually -- just retry for now):
;  (a) CRC error -- retry
;  (b) wrong cylinder error -- recalibrate, retry
;  (c) other errors -- ?
DiskReadError:          DEC     bootRetryCount
                        JZ      TooManyRetries
                        MOV     BX, OFFSET dskIOCB.diskOperationBlock
                        JMP     InitIOCB
;if too many retries, hang with MP code
TooManyRetries:         MOV     AX, mpInitialError
CallDispMP:             CALL    DisplayMPCode
Done:                   %Jam    (BootStrapHandlerID,OFFSET bootStrapTask)
                        %WaitForSystem          ; never returns!

DiskRead                ENDP

;display mp code for executing initial microcode
                        ASSUME  ES:IOPELocalRAM
RamDiskCont:            MOV     AX, mpRunInitial
                        CALL    DisplayMPCode

; step 2
                        MOV     BX, OFFSET dskIOCB.diskOperationBlock
                        MOV     AX, diskShapeCylinder
                        MOV     CX, diskShapeHeadAndSector
                        MOV     SI, startOfBootBufferSpace
                        ADD     SI, bootDataBegins+sectorSize-1
                        AND     SI, NOT (sectorSize-1)
                        MOV     dskIOCB.diskDataPtrLow, SI
                        PUSH    SI
```

```
                        SUB     SI, bootDataBegins
                        MOV     startOfBootBufferPool, SI
                        MOV     ES:[SI].nextBootBuffer, SI
                        MOV     ES:[SI].bootDataStart, Null
                        MOV     ES:[SI].bootDataEnd, Null
                        MOV     bootRetryCount, 10
                        MOV     WORD PTR TooManyRetries+1, mpSDDReadError
ReadDiskShape:          CALL    DiskReadNewFile
                        POP     SI
                        PUSH    SI
; verify seal, version, and checksum
                        CMP     ES:[SI].DSseal, diskShapeSeal
                        JE      DiskShapeSealOK
DiskShapeSealError:     MOV     AX, mpSDDSealError
                        CALL    DisplayMPCode
; seal mismatch may be due to old formatting
; look at checksum words; if they don't match, continue booting with EEPROM disk parms
                        MOV     AX, ES:[SI].DSchecksum
                        NOT     AX
                        CMP     AX, ES:[SI].DSinvertedChecksum
                        JE      Done
; pause before booting so code is visible
                        %WaitForTime (2000)     ; two seconds
                        JMP     SHORT RamDiskStep3

; other error conditions
DiskShapeCksmError:     MOV     AX, mpSDDCksmError
                        JMP     CallDispMP
DiskShapeVrsnError:     MOV     AX, mpSDDVrsnError
                        JMP     CallDispMP

DiskShapeSealOK:        MOV     AX, ES:[SI].DSchecksum
                        NOT     AX
                        CMP     AX, ES:[SI].DSinvertedChecksum
                        JNE     DiskShapeCksmError
                        CALL    CalcChecksum
                        CMP     AX, ES:[SI].DSchecksum
                        JNE     DiskShapeCksmError
                        CMP     ES:[SI].DSversion, diskShapeVersion
                        JNE     DiskShapeVrsnError
; disk shape is OK -- use parms for booting
                        CALL    StuffNewDiskParms

; step 3
RamDiskStep3:           MOV     BX, OFFSET dskIOCB.diskOperationBlock
                        MOV     AX, rootPageCylinder
                        MOV     CX, rootPageHeadAndSector
                        MOV     WORD PTR TooManyRetries+1, mpMesaDoveError
ReadRootPage:           CALL    DiskReadNewFile
                        POP     SI
                        MOV     AX, SI
                        ADD     AX, sectorSize                  ;add sector size
                        MOV     startOfBootBufferSpace, AX      ; and store for booting
                        ADD     SI, rootPageHeaderSize  ; set up for 1st file ID

; step 4 (for now, skip UI and set up for diagnostic or Mesa boot)
                        CMP     bootType, normal
                        JE      SetUpMesaBoot
; diagnostic boot: load one file from root file 0
SetUpDiagnosticBoot:    MOV     CH, ES:[SI].DFIDcylinderHigh
                        MOV     CL, ES:[SI].DFIDcylinderLow
                        MOV     DH, ES:[SI].DFIDhead
                        MOV     DL, ES:[SI].DFIDsector
                        MOV     diskWorkspace.file1.Cyl, CX
                        MOV     diskWorkspace.file1.HdSct, DX
                        MOV     diskWorkspace.fileCount, 1
                        JMP     SHORT LoadFiles
; Mesa boot: load two files from root files 1 (uCode) and 2 (germ)
SetUpMesaBoot:          ADD     SI, SIZE(DiskRootFileID)
                        MOV     CH, ES:[SI].DFIDcylinderHigh
                        MOV     CL, ES:[SI].DFIDcylinderLow
                        MOV     DH, ES:[SI].DFIDhead
                        MOV     DL, ES:[SI].DFIDsector
                        MOV     diskWorkspace.file1.Cyl, CX
                        MOV     diskWorkspace.file1.HdSct, DX
                        ADD     SI, SIZE(DiskRootFileID)
                        MOV     CH, ES:[SI].DFIDcylinderHigh
                        MOV     CL, ES:[SI].DFIDcylinderLow
                        MOV     DH, ES:[SI].DFIDhead
                        MOV     DL, ES:[SI].DFIDsector
                        MOV     diskWorkspace.file2.Cyl, CX
                        MOV     diskWorkspace.file2.HdSct, DX
                        MOV     diskWorkspace.fileCount, 2

; step 5
LoadFiles:              %Restart        (BootStrapHandlerID,OFFSET bootTask,BootTaskInit,OFFSET BootStack)
                        MOV     SI, OFFSET diskWorkspace.file1
; load one complete file
LoadLoop:               PUSH    SI
                        %WaitForCondition       (OFFSET getBootFile,noTimeout)
                        POP     SI
                        MOV     AX, [SI].Cyl
                        MOV     CX, [SI].HdSct
                        MOV     DI, startOfBootBufferPool
                        MOV     bootRetryCount, 10
                        PUSH    SI
                        PUSH    DI
```

```
        ; read first sector of new file into (page-aligned) buffer
                                CALL    DiskReadNewFile
        ; notify boot task that a buffer is ready
        NotifyBufferFull:       POP     DI
                                MOV     ES:[DI].bootDataStart, bootDataBegins
                                MOV     ES:[DI].bootDataEnd, bootDataBegins+sectorSize-1
                                %NotifyCondition        (OFFSET bootBufferFull)
        ; wait till buffer becomes available
        WaitForBuffer:          PUSH    DI
                                %WaitForCondition       (OFFSET bootBufferEmpty,noTimeout)
                                POP     DI
                                TEST    getBootFile.TCBLinkPtr, preNotifyFlag
                                JNZ     DoneWithFile
                                MOV     BX, ES:[DI].bootDataEnd
                                CMP     BX, ES:[DI].bootDataStart
                                JNE     WaitForBuffer
                                MOV     bootRetryCount, 10
        ; read next sector into local buffer
                                PUSH    DI
                                CALL    DiskRead
                                JNC     NotifyBufferFull
                                POP     DI
        ; done with file - decrement file count
        DoneWithFile:           POP     SI
                                DEC     diskWorkspace.fileCount
                                JZ      DoneWithAllFiles
                                ADD     SI, SIZE (FileSpec)
                                JMP     LoadLoop
        DoneWithAllFiles:       %EstablishHandlerAccess (DiskHandlerID)
                                ASSUME  ES:DiskIOR
                                MOV     diskFCB.diskStartHandlerForIOP, FALSE
                                %NotifyCondition        (OFFSET finishedLoaderFileFetch)

        ; step 6
        ; idle till task is subsumed
                                JMP     Done

        ; utility procs
                                ASSUME  ES:NOTHING

        CalcChecksum            PROC    NEAR

        ;Arguments:             ES:SI   offset of word-aligned buffer
        ;Returns:               AX:     word checksum
        ;Uses (without saving): CX, DX

        unChecksummed           EQU     0FFFFH
        checksumExtent          EQU     254             ; size of page, in words, minus checksums

        ;set up for loop
                                PUSH    SI
                                MOV     CX, checksumExtent
                                XOR     AX, AX
        SumNextWord:            MOV     DX, ES:[SI]
                                XCHG    DH, DL          ;byteswap word
        ;The following sequence is the checksum algorithm
                                ADD     AX, DX
                                ADC     AX, 0
                                ROL     AX, 1
        ;advance to next word
                                INC     SI
                                INC     SI
                                LOOP    SumNextWord

        ;normalize result
                                CMP     AX, unChecksummed
                                JNE     ChecksumExit
                                XOR     AX, AX
        ChecksumExit:           XCHG    AH, AL          ;byteswap checksum
                                POP     SI
                                RET
        CalcChecksum            ENDP


        StuffNewDiskParms       PROC    NEAR
        ;Arguments:             ES:SI   offset of disk shape page descriptor
        ;Uses (without saving): AX, BX
        ; note that all word values in disk shape must be byte-swapped

                                PUSH    ES
        ;fix IOCB first, saving values in stack
                                MOV     BX, OFFSET dskIOCB.diskOperationBlock
                                MOV     AL, ES:[SI].DSsectorsPerTrack
                                MOV     [BX].diskSecPerTrack, AL
                                MOV     AH, ES:[SI].DSheadsPerCylinder
                                MOV     [BX].diskHdsPerCyl, AH
                                PUSH    AX
                                MOV     AX, ES:[SI].DScylinderCount
                                XCHG    AH, AL  ;byte-swap
                                MOV     [BX].cylPerDrive, AX
                                PUSH    AX
                                MOV     AX, ES:[SI].DSreducedWriteCyl
                                XCHG    AH, AL  ;byte-swap
                                MOV     [BX].diskReducedWriteCyl, AX
                                PUSH    AX
                                MOV     AX, ES:[SI].DSpreCompCyl
                                XCHG    AH, AL  ;byte-swap
```

```
                        MOV     [BX].diskPreCompCyl, AX
                        PUSH    AX

; now change values in disk FCB
                        %EstablishHandlerAccess (DiskHandlerID)
                        ASSUME  ES:DiskIOR
                        POP     diskFCB.rd0.diskPreCompensationCylinder
                        POP     diskFCB.rd0.diskReducedWriteCurrentCylinder
                        POP     diskFCB.rd0.diskCylindersPerDrive
                        POP     AX
                        MOV     diskFCB.rd0.diskHeadsPerCylinder, AH
                        MOV     diskFCB.rd0.diskSectorsPerTrack, AL

                        POP     ES
                        ASSUME  ES:NOTHING
                        RET
StuffNewDiskParms       ENDP


IOPEInRam               ENDS


                        END
```

```
;Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.

;-- stored as [Idun]<WMicro>Dove>RAMEEP.asm
;-- created on  14-Feb-84 11:13:22
;-- This file is intended to contain RAM-resident portion of EEProm constants.
;
;-- last edited by:

;--     KEK     20-Mar-86 10:21:11      ;add DiagAutoRun/RDCLandingZone/WriteCount
;--     JPM      2-Aug-85 17:22:24      :Removed eePromLowMem and eePromHighMem (now in ROMEEP.asm).
;--     KEK     31-Jul-85 16:31:19      :created RAMEEP.asm from orig EEPDefs.asm
;--     JPM     14-Nov-84 14:52:53      :Updated for new layout.
;--     JPM     12-Oct-84 16:11:44      :Updated for new layout.
;--     VXS     16-Aug-84 11:04:01      :made EEPromSegment COMMON.
;--     VXS      8-Aug-84 15:43:40      :Made tempEEPromImage so we
                ;know where this thing is. (fixes bug where if wordsInEEProm
                ;wasn't on even 4 word boundary, would be overlap)
;--     JPM     19-Jul-84 11:04:57      :Added eePromLockMode.
;--     VXS     11-Jul-84 16:28:56      :Creation. See IOPDefs.asm for earlier history.


;-------------------------------------------------------------------------------
;
;Definitions for RAM-used offsets within EEProm:
;(for the other eeprom defs, see latest ROMEEP.asm and BadPage.asm)

eePromMemSize           EQU     RAMSegment+byteEEPromOffset+25
;       [0..3] = encoded VM size (in increments of 64 map pages)
;                0 => none
;                1 => 64 VM map pages
;                2 => 128 VM map pages
;                3 => 256 VM map pages
;       [4..7] = encoded control store sizes (in increments of 4K pages)
;                0 => none
;                1 => 4K control store
;                2 => 8K control store

eePromHardwareBuild     EQU     RAMSegment+byteEEPromOffset+26
;       0 => ?none
;       1 => B0/B1
;       2 => B2
;       3-255 => spare incremental encodings

eePromMisc              EQU     RAMSegment+byteEEPromOffset+27
;       [0..0] = Default boot type bit2 (with or without diagnostics)
;       [1..1] = Default diagnostic boot type bit (short or long)
;       [2..7] = spare

eePromRS232DCEtype      EQU     RAMSegment+byteEEPromOffset+28
eePromRS232DCEattr      EQU     RAMSegment+byteEEPromOffset+29
eePromRS232DTEtype      EQU     RAMSegment+byteEEPromOffset+30
eePromRS232DTEattr      EQU     RAMSegment+byteEEPromOffset+31

eePromPCEMemSize        EQU     RAMSegment+wordEEPromOffset+32
eePromPCEConfig EQU     RAMSegment+wordEEPromOffset+34

eePromOption1           EQU     RAMSegment+wordEEPromOffset+36
eePromOption2           EQU     RAMSegment+wordEEPromOffset+38
eePromOption3           EQU     RAMSegment+wordEEPromOffset+40

;this is actually defined in ROMEEP.asm.
;eePromFloppy           EQU     ROMSegment+wordEEPromOffset+42

;eePromSpare2           EQU     RAMSegment+byteEEPromOffset+44
;eePromSpare3           EQU     RAMSegment+byteEEPromOffset+45
;eePromSpare4           EQU     RAMSegment+byteEEPromOffset+46
;eePromSpare5           EQU     RAMSegment+byteEEPromOffset+47
;eePromSpare6           EQU     RAMSegment+byteEEPromOffset+48
;eePromSpare7           EQU     RAMSegment+byteEEPromOffset+49

eePromDiagAutoRun       EQU     RAMSegment+wordEEPromOffset+44
eePromRDCLandingZone    EQU     RAMSegment+wordEEPromOffset+46
eePromWriteCount        EQU     RAMSegment+wordEEPromOffset+48

eePromSpare8            EQU     RAMSegment+byteEEPromOffset+50
eePromSpare9            EQU     RAMSegment+byteEEPromOffset+51
eePromSpare10           EQU     RAMSegment+byteEEPromOffset+52
eePromSpare11           EQU     RAMSegment+byteEEPromOffset+53
eePromSpare12           EQU     RAMSegment+byteEEPromOffset+54
eePromSpare13           EQU     RAMSegment+byteEEPromOffset+55
eePromSpare14           EQU     RAMSegment+byteEEPromOffset+56
eePromSpare15           EQU     RAMSegment+byteEEPromOffset+57
eePromSpare16           EQU     RAMSegment+byteEEPromOffset+58
eePromSpare17           EQU     RAMSegment+byteEEPromOffset+59
eePromSpare18           EQU     RAMSegment+byteEEPromOffset+60
eePromSpare19           EQU     RAMSegment+byteEEPromOffset+61
eePromSpare20           EQU     RAMSegment+byteEEPromOffset+62
eePromSpare21           EQU     RAMSegment+byteEEPromOffset+63
eePromSpare22           EQU     RAMSegment+byteEEPromOffset+64
eePromSpare23           EQU     RAMSegment+byteEEPromOffset+65
eePromSpare24           EQU     RAMSegment+byteEEPromOffset+66
eePromSpare25           EQU     RAMSegment+byteEEPromOffset+67
```

```
;Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.

;-- IORegion locations for the boot handler.
;-- stored as [Iris]<WMicro>Dove>ROMDefs.asm
;
;-- last edited by:


;--     RDH                     17-Sep-85 12:08:50      ;Change maintPanelBkGrdRightPad and maintPanelBkGrdHeight for prettier mp codes.
;--     RDH                     14-Sep-85 10:08:50      ;Change timeoutInterval to seconds instead of millisecs.
;--     RDH                     14-Sep-85  9:04:37      ;Add oneSecond.
;--     RDH                     13-Sep-85 19:21:18      ;Add constants for cursor visibility:  maintPanelBkGrdRightPad,
maintPanelBkGrdHeight.
;--     RDH                     13-Sep-85 17:12:16      ;Changed timeout interval to 35000 (thirty five seconds).
;--     RDH                     12-Sep-85 14:37:06      ;Made icondatastructure smaller since there is no more mouse tracking and ICON
segment was never used.
;--     RDH                     12-Sep-85 14:14:25      ;Add hideCursorDisplayOn.
;--     RDH                     12-Sep-85 12:02:22      ;Add timeOutDisab and timeOutEnab.
;--     RDH                     12-Sep-85 11:34:41      ;Add backToUI and bootSystem.
;--     RDH                     11-Sep-85 16:32:29      ;Add timeoutInterval.
;--     kek                      5-Sep-85 19:40:32      ;mp code definitions
;--     RDH                     17-Aug-85 17:39:22      ;Add niceCursorDisplayOn.
;--     RDH                     14-Aug-85 11:49:32      ;Fold in UI stuff.
;--     JPM                     15-Jul-85 17:04:07      ;Added constants from ROMEthBt.asm.
;--     JPM                     11-Jul-85 10:03:32      ;Took fields out of floppy context (will use floppy IOCB def), added
CallHandlerInitProc.
;--     JMM                     19-Jun-85 15:19:54      ;Added macro.
;--     JPM                     21-May-85 12:54:14      ;Diagnostic boot changes.
;--     JMM                      4-Apr-85 15:01:14      ;
;--     JMM                      6-Feb-85  9:21:27      ;Add boot error constants section.
;--     JMM                     28-Jan-85 18:25:43      ;Add macro section.
;--     JMM                     25-Nov-84 16:04:38      ;Added floppy changes.
;--     JPM                      3-Nov-84 11:21:53      ;Remove EthIOFce from INCLUDEs, add Floppy IOCB skeletal structure.

;--     BKI/JMM                 24-Oct-84 16:14:15      ;First release.

;NAME                   ROMBDefs
;-------------------------------------------------------------------------------
;


;-------------------------------------------------------

;Constants for STRUC definitions.
;--------------------------------

BootJumpTable           STRUC

startRAMOpie            DW      ?       ;An IOP Start block will result in an IOP
startRAMOpieCS          DW      ?       ;address being saved here for later entry.
iopEntry                DW      ?
iopEntryCS              DW      ?
processBootBlock        DW      ?

BootJumpTable           ENDS

;------------------------

;BootTimeVariables      STRUC


;device                 DW      ?       ;1-Disk, 2-Ethernet, 3-Floppy, 4-RS232C.
;mode                   DW      ?       ;0-Normal, 1-Fast boot
;showUserInterface      DW      ?       ;0-yes, #0-no.
;bootRetryCount         DW      ?       ;initially 0, incremented per failure

;reincarnationFlag      DW      ?       ;For task reinitialization - 0 => tasks
                                        ;to use local RAM otherwise main memory.

;emulatorID             DW      3 DUP(?) ;Initially 0, updated by boot executive.
;loaderID               DW      3 DUP(?) ;Initially 0, updated by boot executive.

;diagnosticsType                DW      ?       ;Initialized at boot-time. 0 => short
                                        ;diagnostics - user has set EEPROM to
                                        ;indicate desire for a diagnostics boot.
                                        ;#0=> long diagnostics boot - user has
                                        ;manually requested diagnostics at boot
                                        ;time!

;BootTimeVariables      ENDS

;------------------------
;Device types:
;-------------

disk                    EQU     1
floppy                  EQU     2
ethernet                EQU     3
experimental            EQU     4
rs232C                  EQU     4
;------------------------
;Boot types:
;-----------

normal                  EQU     0
diagnostic              EQU     1

ROMBDefs.asm        17-Sep-85 12:13:15 PDT                                                                  1
```

```
;-----------------------
;Boot ICON STRUC's and constants:
;-------------------------------

ICONDataStructure       STRUC

ICONLeftImagePtr        DW      ?       ;Pointer to image of ICON symbols 0 => no symbol.
ICONRightImagePtr       DW      ?       ;Pointer to image of ICON symbols 0 => no symbol.
ICONOffset              DW      ?       ;This is this ICONs offset in display memory.
;ICONSegment            DW      ?       ;This is this ICON's segment in display memory.
;leftBoundary           DW      ?       ;Left icon edge from the left of the screen in bits.
;rightBoundary          DW      ?       ;Right icon edge from the left of the screen in bits.
bootingProcedure        DW      ?       ;Offset to procedure that does selected boot.
invertedICON            DB      ?       ;0 => not inverted else inverted.
bootDevice              DB      ?       ;0 => no device present here.
typeOfBoot              DB      ?       ;0 => normal boot, otherwise diag. boot.
previousICON            DW      ?       ;0 => this is the first icon.
nextICON                DW      ?       ;0 => this is the last icon.

ICONDataStructure       ENDS


cursorPosition          STRUC

x                       DW      ?
y                       DW      ?

cursorPosition          ENDS

totalLengthOfICON       EQU     672             ;In bits.
lengthOfICON            EQU     64              ;
widthOfICON             EQU     24              ;Lines.
ICONVerticalDiplacement EQU     10              ;Trial and error for visual appeal.
ICONHorizontalDiplacement EQU   5+lengthOfICON  ;Trial and error for visual appeal.
displayON               EQU     11101010B       ;Bit 3 set enables the display.
niceCursorDisplayOn     EQU     10001010B       ;High nibble makes cursor AND with bitmap. Bit 3 set enables the display.
hideCursorDisplayOn     EQU     0CAH
niceMPDisplayOn         EQU     04AH
;----------------
;Misc. constants:
;----------------

numberOfSoftKeys        EQU     10
crossOver64KbBank       EQU     01000H
distanceFromBottom      EQU     30              ;Number of scan lines from bottom of icons to bottom of display.
ICONBottomEdge          EQU     10              ;???jmm:85-03-27
ICONInnerDepth          EQU     24              ;scan lines in icon button
ICONByteLength          EQU     8               ;length and width are the same thing here.
ICONBitWidth            EQU     64              ;Depth is up and down.
vertIconOffsetInButton  EQU     4               ;scan lines
leftIconOffsetInButton  EQU     1               ;bytes
rightIconOffsetInButton EQU     5               ;bytes

threeSeconds            EQU     3000
timeoutInterval         EQU     35      ;Number of seconds to wait before starting default booting  This is 10 seconds plus the 25 where
25 is the approx time it takes for the daybreak display to be clearly visible from a cold start.
maintPanelBkGrdRightPad EQU     0F0H    ;High 4 bits of byte quantity are set.
maintPanelBkGrdHeight   EQU     20      ;mpcodeCursorXPos + Cursor height + mpcodeCursorXPos.
mpcodeCursorXPos        EQU     2       ;Leave space between mpcode and border.
mpcodeCursorYPos        EQU     2       ;Leave space between mpcode and border.
backToUI                EQU     1       ;For communication with diagnostics in finishMode.
bootSystem              EQU     0       ;For communication with diagnostics
timeOutEnab             EQU     1       ;Allow timeouts in boot device selection.
timeOutDisab            EQU     0       ;Wait indefinitely for user.

inverted                EQU     -1
notInverted             EQU     0

numberOfICONs           EQU     4
wordBitSize             EQU     16
byteLength              EQU     8
wordLength              EQU     16

fatDaisy                EQU     0
fatDaybreak             EQU     0

doveDisplayBasePage             EQU     0       ;In decimal page numbers.
fatDaybreakDisplayBasePage      EQU     0       ;In decimal page numbers.
fatDaisyDisplayBasePage         EQU     0
;slimDaybreakDisplayBasePage    EQU     768
slimDaybreakDisplayBasePage     EQU     0       ;jmm:84-12-04:UntilGAPmoves.
slimDaisyDisplayBasePage        EQU     512


;fatDaybreakDisplayMemSize      EQU     1024    ;In decimal page numbers.
daisyDisplayMemSize             EQU     256
fatDaybreakDisplayMemSize       EQU     256
slimDaybreakDisplayMemSize      EQU     256


byteWordAlignMask       EQU     0FEH
wordWordAlignMask       EQU     0FFFEH

dontCare                EQU     0
setAtRunTime            EQU     0
status                  EQU     0
```

```
nullOffset              EQU     0

pageSizeInBytes         EQU     0200H

;------------------------
;Disk ROM boot constants and STRUC's:
;-------------------------------------

DiskBootContext         STRUC

diskIOCBDone            DB      (SIZE Condition) DUP (?)
diskBootIOCB            DB      156 DUP (?)     ;SIZE DiskIOCBRecord in DskIOFce.def

DiskBootContext         ENDS


;------------------------
;Ethernet ROM boot constants and STRUC's:
;----------------------------------------

srcOffset       EQU     6
typeFieldOffset EQU     12
checksumOffset  EQU     14
lengthOffset    EQU     16
srcHostOffset   EQU     36
bfnOffset       EQU     46
seqNumberOffset EQU     52
bootDataOffset  EQU     54
maximumBootBufferSize   EQU     bootDataOffset+pageSizeInBytes

bfnSize         EQU     6                       ;bytes
encapsulation   EQU     checksumOffset

tenSeconds      EQU     10000                   ;(in milliseconds)
oneSecond       EQU     1000                    ;(in milliseconds)

EtherBootContext        STRUC

etherIOCB       DB      40 DUP (?)              ;jpm 84-11-2 >= SIZE(ethernetIOCB)
etherIOCBDone   DB      (SIZE Condition) DUP (?)
noResponseTimer DW      ?
retryCount      DW      ?
romRetryEntry   DD      ?
initialBufPtr   DW      ?                       ;place to load next piece of EtherInitialDove.db

EtherBootContext        ENDS


;------------------------
;Floppy ROM boot constants and STRUC's:
;--------------------------------------

FloppyBootContext       STRUC

floppyIOCBDone          DB      (SIZE Condition) DUP (?)
floppyIOCB              DB      152 DUP (?)

FloppyBootContext       ENDS


;-----------------------------------------------------------------------------
;
;
;-----------------------------------------------------------------------------
;Convenience macros:
;-------------------
;
%*Define (MultiplyByTwo (register))
        (ROL    %register, 1)

%*Define (DivideBy2     (register))
        (SHR    %register, 1)

%*Define (DivideBy4     (register))
        (SHR    %register, 2)

%*Define (ByteSwap      (wordToSwap))
        (%wordToSwap SHL 8 OR %wordToSwap SHR 8)

%*Define (CallHandlerInitProc   (handlerID))    (
                MOV     DI, %handlerID
                SHL     DI, 2
                PUSH    DS
                PUSH    ES
                CALL    DWORD PTR HandlerInitProcTable[DI]
                POP     ES
                POP     DS
)

;-----------------------------------------------------------------------------
;Booting error numbers:
;----------------------
;
bootingError    EQU     -1
noBufferSpace   EQU     0101H
noLoadSpace     EQU     0202H
unKnownBootBlock EQU    0303H
```

```
noRAMStartAddress EQU   0404H
diskROMBootError EQU    0405H

;-------------------------------------------------------------------------------
;Maintenance Panel Codes:
;------------------
;
mpStartBooting  EQU     0100D
mpDeviceUnknown EQU     0113D
mpFetchInitial  EQU     0149D
mpRunInitial    EQU     0150D
mpInitialError  EQU     0151D
mpFetchMesaDove EQU     0199D
mpRunMesaDove   EQU     0200D
mpMesaDoveError EQU     0201D
mpRunGerm       EQU     0501D
mpFloppyCleaning        EQU     0077D
```

```
$MOD186
$PAGELENGTH     (72)
$PAGEWIDTH      (136)

;Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.

;-- stored as [Iris]<WMicro>Dove>ROMBoot.asm
;-- created on  19-Jul-84 13:20:18

;-- last edited by:
;--     JAC         15-Jan-87 10:46:00          ;fix alt enet booting. fix short or long alag bug
;--     KEK         12-May-86 12:44:21          ;change e'net address to 252000043008. (Daybreak is 252000040008). Also, OR into
baseEthernetFileID.
;--     CWM         30-Apr-86 22:37:17          ;ReInstall diagnostic icon
;--                                             ;1671 bytes/241 bytes saved
;--     CWM         25-Apr-86 19:58:10          ;Cut out icon top/bottom calcs
;--                                             ;1633 bytes/279 bytes saved
;--     CWM         25-Apr-86 12:17:25          ;Cut out diagnostic icon for Karey
;--                                             ;1649 bytes/263 bytes saved
;--     CWM         23-Apr-86  1:41:16          ;Integ BumpES and InvertICON calls
;--                                             ;1687 bytes/225 bytes saved
;--     CWM         22-Apr-86 23:29:29          ;Function key and kleenup EQU's
;--                                             ;T4Key fixes
;--                                             ;1695 bytes/217 bytes saved
;--     CWM         21-Apr-86 23:42:53          ;New ICON buttons
;--                                             ;1726 bytes/186 bytes saved
;--     CWM         20-Apr-86 22:03:52          ;Fix EQU/DB+DW, AR shifts, MUL
;--                                             ;DIV, Remove useless XOR's
;--                                             ;1887 bytes/25 bytes saved
;--
;------------------------------1912 BYTES-------------------------------
;--
;--     RDH         17-Sep-85 12:06:15          ;Make timeouts disabled on return to UI after diagnostic boot selected by user.  Make
booting mp codes a little prettier by sending the cursor to 2,2 instead of 0,0 and expanding the mp panel background box by 2.
;--     JAC         16-Sep-85 22:16:37          ;Fix handling of no default boot device
;--     RDH         16-Sep-85 16:49:45          ;Fix error in ES handling when EEProm is bad and there is a timeout.  Fix handling of no
default boot device.  Add code to disable timeouts after running diagnostics.
;--     RDH         14-Sep-85  9:47:30          ;Fix bug in 35 second timeout.
;--     RDH         13-Sep-85 18:11:39          ;Improve cursor readability.
;--     RDH         13-Sep-85 13:34:06          ;Fix up LED's for case where EEProm is bad.
;--     RDH         13-Sep-85 11:20:22          ;Add DEC AL to default booting code to fix off by one error.
;--     RDH         12-Sep-85 13:59:34          ;Comment out mouse code.  Comment out floppy cleaning and arrow cursor icons.  Adjust for
smaller icon data structure.  Move IconSelected code below DeviceKnown code to save on JMP's.
;--     JAC/RDH     12-Sep-85 11:35:11          ;Add setting of FinishMode./Add timeouts and watching for STOP key.
;--     RDH         11-Sep-85 13:42:26          ;Fix bug in space saving. Note that INC does not set or clear the Carry flag.
;--     RDH         10-Sep-85 20:06:28          ;Save space by adding SHORT's, changing ADD reg, 2 to INC reg x2, changed some loops to
byte operations.  Moved procedures to the end to save a few jumps over them.
;--     JAC         10-Sep-85 14:05:55          ;test for short or long diags
                                                ;shorten SelectionLoop
;--     JAC          6-Sep-85 11:39:19          ;fix border pattern
                                                ;display lines aren't a multiple of 4
                                                ;add waiting for function key 9
;--     JAC          5-Sep-85 20:11:15          ;fix black line at bottom of display
                                                ;unknown device hangs instead of returning to Selection
;--     JAC          5-Sep-85 11:09:37          ;fix mp codes and arrow cursor again
                                                ;check for function keys 9 and 10 in loop
                                                ;remove floppy head cleaning temporarily
;--     JAC          4-Sep-85 22:48:39          ;clear LEDs
;--     kek          4-Sep-85 19:29:07          ;mp codes.
;--     JAC          4-Sep-85 17:15:56          ;put in function key 9 checks at beginning
                                                ;check prebootSwitches before reading EEPROM
                                                ;put cursor at upper left corner while booting
;--     JAC          3-Sep-85 21:01:59          ;fix code at NextDevice
;--     JAC          3-Sep-85 18:09:34          ;first cut at reading EEPROM for available devices
;--     JAC          2-Sep-85 17:51:17          ;fixed arrow cursor and line at icon border
                                                ;fix unknown device and FloppyHeadClean
;--     JAC          2-Sep-85 17:00:04          ;fix up starting user interface and diagnostics
;--     JAC          2-Sep-85 16:00:21          ;set flag for display instead of calling proc
;--     JAC          1-Sep-85 14:38:24          ;fix floppy icon; alt eth boot is now function key 3
                                                ;change definition of T9Key and T10Key
;--     JAC         31-Aug-85 14:32:41          ;pop the stack after ChainUpICONDataStructure loop
;--     JAC         29-Aug-85 14:50:07          ;Fix icon booting proc
;--     JAC         27-Aug-85 17:28:35          ;Zero bootDeviceIORSpace since ICONs munged it.
;--     RDH         23-Aug-85 18:08:59          ;Fix bug above icon and SI smashed by InvertIcon bugs.
;--     RDH         23-Aug-85  8:29:27          ;Fix 2 matching bugs in background painting.
;--     RDH         22-Aug-85 16:14:24          ;Fix half icon inverted, and mouse click offset problems.
;--     RDH         22-Aug-85 13:29:10          ;Fix bugs in mouse tracking, icon inversion, and keyboard watching.
;--     RDH         22-Aug-85 10:56:05          ;Add mouse tracking, selection, and selection inversion.
;--     RDH         20-Aug-85  9:50:54          ;Fix munge of DS after calling display handler procs.
;--     RDH         17-Aug-85 16:36:50          ;Delete error commenting out DisableInterruptsUntilNextWait.
;--     RDH         17-Aug-85 16:36:50          ;fix up cursor stuff.  Add zillions of WaitForSystem's.  Comment out
%'DisableInterruptsUntilNextWait.
;--     RDH         17-Aug-85 14:47:02          ;Changed AH to AL near ButtonInnerLoop.  Add use of SetCursorPattern.  Set up DS for
display handler procs.
;--     RDH         17-Aug-85 13:23:34          ;Saved ES at NoUnderflow. Restored ES at EnableDisplay.
;--     RDH         17-Aug-85 11:45:58          ;Added DisableInterruptsUntilNextWait.
;--     RDH         16-Aug-85 20:42:11          ;Add set up of BX for ChainUpICONDataStructure. Fix byteswap problem in SetICONbutton.
Remove skip over UI.
;--     RDH         16-Aug-85 15:42:13          ;Add WaitForSystem after before PaintScreenDeskTopGrey.  Fix error in loop control on
WholeDisplayLoop.
;--     RDH         16-Aug-85  8:32:56          ;Fold in JPM.s changes: ";Change reg for new parm to AH" and "Add parm to
EtherInitialize".
;--     RDH         14-Aug-85 16:15:07          ;Fold in changes for UI.
;--     BKI         11-Aug-85 19:30:28          ;Convert to new EthIOFce
;--     JPM         22-Jul-85 15:19:49          ;Change IOPEInROM alignment to WORD, set up clientCondition in EtherInitialize IOCBs.
;--     JPM         15-Jul-85 17:20:38          ;Opie redesign conversion.
;--     JMM         26-Jun-85 14:44:07          ;Fixed alternate etherbooting support.

ROMBoot.asm         15-Jan-87 13:54:53 PST                                                                                    1
```

```
;--     BKI         21-Jun-85 13:12:45          ;Fixed alternate etherbooting support.
;--     JMM         19-Jun-85 17:17:56          ;Added alternate etherbooting support.
;--     JMM/BKI     19-Jun-85 15:14:42          ;Changed ethernet fifo limit because of slowed clock.
;--     JPM          24-May-85 14:58:06         ;Overlay boot changes.
;--     JMM          15-Apr-85 16:50:14         ;Initialize ethernet if present.
;--     JMM          25-Feb-85 15:30:45         ;Removed IOPLRAM.asm.
;--     JMM           7-Dec-84 12:02:48         ;First release.

NAME            ROMBoot

;-------------------------------------------------------------------------------
;
$NOLIST
$INCLUDE        (HardDefs.asm)
$INCLUDE        (IOPDefs.asm)
$INCLUDE        (QueDefs.asm)
$INCLUDE        (QueMacro.asm)
$INCLUDE        (IOPMacro.asm)
$INCLUDE        (ROMBDefs.asm)
$INCLUDE        (EthIOFce.asm)
$INCLUDE        (DsplDefs.asm)
$INCLUDE        (ROMEEP.asm)
$INCLUDE        (RAMEEP.asm)
$LIST

;-------------------------------------------------------------------------------

EXTRN           BootStrapHandlerID: ABS
EXTRN           DisplayHandlerID: ABS
EXTRN           EthernetHandlerID: ABS
EXTRN           KeyBoardAndMouseHandlerID: ABS
EXTRN           MaintPanelHandlerID: ABS

;---------------
;*****************************************************************
;---------------

IOPELocalRAM    SEGMENT AT 0


;from IOPLRAM.asm:
;-----------------

EXTRN           HandlerInitProcTable: DWORD
EXTRN           prebootSwitches: WORD
EXTRN           bootType: BYTE
EXTRN           diagType: BYTE
EXTRN           finishMode: BYTE
EXTRN           skipUserInterface: BYTE
EXTRN           device: WORD    ;VMMDefs.asm
EXTRN           baseEthernetFileID: WORD
;EXTRN          cleanFloppyHeadsPROC: WORD
EXTRN           timeoutEnable: BYTE

IOPELocalRAM    ENDS

;---------------
;*****************************************************************
;---------------

;Imported Variables:

;from IORROM8t.asm:
;------------------

BootStrapIOR    SEGMENT COMMON

EXTRN           bootTask: TaskContextBlock      ;
EXTRN           bootStrapTask: TaskContextBlock ;
;EXTRN          jumpTable: BootJumpTable        ;
EXTRN           bootDeviceICON: ICONDataStructure ;
EXTRN           displaySegment: WORD            ;
EXTRN           displayOffset: WORD             ;
;EXTRN          currentCursor: cursorPosition   ;
EXTRN           bootDeviceIORSpace: EtherBootContext ;
EXTRN           ICONsBottomEdge: WORD ;
EXTRN           ICONsTopEdge: WORD ;
EXTRN           displayWidthInBytes: WORD ;
;EXTRN          EndBootstrapIOR: FAR
EXTRN           OverlayLength: ABS
EXTRN           displayStartOffset: WORD
EXTRN           displayStartSegment: WORD
EXTRN           bigTimeOut: BYTE
EXTRN           littleTimeOut: WORD
EXTRN           allowTimeout: BYTE

BootStrapIOR    ENDS

;from IORDisp.asm:
;-----------------

DisplayIOR      SEGMENT COMMON

EXTRN           numberBitsPerLine: WORD         ;These two words contain the size
EXTRN           numberDisplayLines: WORD        ;of the display in bits.
EXTRN           bitMapOrg: WORD                 ;This = 0 for Daybreak
                                                ;or "n" Achip Daisy;


ROMBoot.asm         15-Jan-87 13:54:53 PST
```

```
                                                ;40000 for Daisy with one AChip.
EXTRN           cursorXCoord: WORD              ;After putting cursor position here
EXTRN           cursorYCoord: WORD              ;set a bit in chngdInfo.
EXTRN           borderHigh: BYTE
EXTRN           borderLow: BYTE
EXTRN           displCntl: BYTE                 ;Bit-3 <- 0 means display
border.

EXTRN           chngdInfo: WORD
EXTRN           cursorPattern: BYTE             ;place the display handler looks
                                                ;for cursor bit map.  It is not
                                                ;a BYTE but rather 32 bytes.

DisplayIOR      ENDS

;from IOREther.asm:
;------------------

EthernetIOR     SEGMENT COMMON

EXTRN           etherCmdAvail: Condition
EXTRN           etherOutQueue: QueueBlock

EthernetIOR     ENDS

;from IORKeyMo.asm:
;------------------

KeyBoardAndMouseIOR     SEGMENT COMMON

EXTRN           HexValue: BYTE                  ;ASCII value of key down.
;EXTRN          MouseX: WORD, MouseY: WORD      ;Delta motion change of mouse.

KeyBoardAndMouseIOR     ENDS


;from IORMaint.asm:
;------------------
MaintPanelIOR           SEGMENT COMMON

EXTRN                   maintPanelCode: WORD
EXTRN                   maintPanelChanged: Condition

MaintPanelIOR           ENDS


;-------------------------------------------------------------------------------

BootStrapSTK    SEGMENT COMMON

EXTRN           BootStack: WORD                 ;initial stack for boot task.
EXTRN           BootStrapStack: WORD            ;initial stack for bootstrap task.

BootStrapSTK    ENDS

;---------------
;*****************************************************************
;---------------

IOPEInROM       SEGMENT WORD PUBLIC
                ASSUME  CS:IOPEInROM

PUBLIC          BootStrapInit, EtherInitialize, DisplayMPCode

EXTRN           DiskBootStrap: NEAR             ;ROMDskBt.asm
EXTRN           EthernetBootStrap: NEAR         ;ROMEthBt.asm
EXTRN           FloppyBootStrap: NEAR           ;ROMFlpBt.asm

;
;-------------------------------------------------------------------------------
;For initializing ethernet chips:
;--------------------------------

configureParms  DB      11                      ;configure byte count
                DB      00BH                    ;fifo limit (008H for B2)
                DB      080H                    ;bad frame, etc
                DB      02EH                    ;loopback, etc
                DB      000H                    ;backoff parms
                DB      96                      ;interframe spacing
                DB      000H                    ;slot time
                DB      0F1H                    ;retry number
                DB      000H                    ;promiscuous, etc
                DB      000H                    ;carrier sense, collision detect
                DB      64                      ;min frame length
configureParmsSize      EQU     $-configureParms

                EVEN
ethernetBootFilePrefix  DW      %ByteSwap(00000H)       ;Daisy boot file number
                        DW      %ByteSwap(0AA00H)
                        DW      %ByteSwap(008C0H)

;-------------------------------------------------------------------------
;Boot Keyboard knowledge:

softKeyWidth            EQU             64      ;in bits
narrowGap               EQU             8       ;in bits the gap between two softkeys
                                                ;in the same group.
```

```
wideGap                 EQU             56      ;in bits the gap between two groups of softkeys.
;softKeyWidthInWords    EQU             softKeyWidth/16

;bit offsets of the softkeys relative to the left side of the first softkey.
f1SoftKeyOffset         EQU             0
f2SoftKeyOffset         EQU             f1SoftKeyOffset + softKeyWidth + narrowGap
f3SoftKeyOffset         EQU             f2SoftKeyOffset + softKeyWidth + narrowGap
f4SoftKeyOffset         EQU             f3SoftKeyOffset + softKeyWidth + narrowGap
f5SoftKeyOffset         EQU             f4SoftKeyOffset + softKeyWidth + wideGap
f6SoftKeyOffset         EQU             f5SoftKeyOffset + softKeyWidth + narrowGap
f7SoftKeyOffset         EQU             f6SoftKeyOffset + softKeyWidth + narrowGap
f8SoftKeyOffset         EQU             f7SoftKeyOffset + softKeyWidth + narrowGap
f9SoftKeyOffset         EQU             f8SoftKeyOffset + softKeyWidth + wideGap
f10SoftKeyOffset        EQU             f9SoftKeyOffset + softKeyWidth + narrowGap

totalWidthICONS         EQU             f10SoftKeyOffset + softKeyWidth

bootDeviceCode  DB      disk,floppy,ethernet,rs232C

;sizeOfPageInBytes      DW      pageSizeInBytes
;bitsPerWord            DW      wordBitSize
;bitsPerByte            DW      8
linesFromBottom         DW      distanceFromBottom
;wordsPerLine           DW      16
;bytesPerLine           DW      8

STOPkey         EQU     01010100B


KeyBoardBootSelect      EQU     $
diagBootKeyBaseValue    EQU     01100110B
bootKeyBaseValue        EQU     01100010B
numberKeyBaseValue      EQU     10000100B
leftMouseButton         EQU     00000001B
rightMouseButton        EQU     00000010B

T1key           EQU     01100011B
T9key           EQU     01101011B
T4key           EQU     01100110B
diagnostics     EQU     4


;T1key          DB      01100011B               ;Disk boot.
;T2key          DB      01100100B               ;Floppy boot.
;T3key          DB      01100101B               ;Ethernet boot.
;T4key          DB      01100110B               ;.

;diagnostics    EQU     $ - T1key

;T5key          DB      01100111B               ;Disk diag boot.
;T6key          DB      01101000B               ;Floppy diag boot.
;T7key          DB      01101001B               ;Ethernet diag boot.
;T8key          DB      01101010B               ;.

;T9key          DB      01101011B               ;Not used.
;T10key         DB      01101100B               ;Floppy head clean.

;-------------------------------------------------------------------------------

iconRelativeXcoordinates        DW      f1SoftKeyOffset
                                DW      f2SoftKeyOffset
                                DW      f3SoftKeyOffset
                                DW      f4SoftKeyOffset
                                DW      f5SoftKeyOffset
                                DW      f6SoftKeyOffset
                                DW      f7SoftKeyOffset
                                DW      f8SoftKeyOffset
                                DW      f9SoftKeyOffset
                                DW      f10SoftKeyOffset

;-------------------------------------------------------------------------------
;Boot Display:
;-------------
;1's show as white. 0's as black.

desktopGray     EQU     1011101110111011B       ;Two full length lines of
                                                ;each of this then shift right
                                                ;twice for two more lines. Repeat
                                                ;over display for desktop gray.

;---------------


;-----------------------------------
;
diskBootICON    DW      1111111111111111B
                DW      1110000000000111B
                DW      1100000000000011B
                DW      1100111111110011B
                DW      1100111111110011B
                DW      1100111111110011B
                DW      1100111111110011B
                DW      1100111111110011B
                DW      1100111111110011B
                DW      1100000000000011B
                DW      1110000000000111B
                DW      1111100000011111B
```

```
                DW      1100000000000011B
                DW      1001111111111001B
                DW      0000000000000000B
                DW      0000000000000000B

sizeOfICON      EQU     $ - diskBootICON

;-----------------
;
ethernetBootICON DW     1000001100000111B
                DW      1000001100000111B
                DW      1001001100100111B
                DW      1001001100100111B
                DW      1000001100000111B
                DW      1000001100000111B
                DW      1110111111011111B
                DW      0000000000000000B
                DW      0000000000000000B
                DW      1111101111110111B
                DW      1110000011000001B
                DW      1110000011000001B
                DW      1110010011001001B
                DW      1110010011001001B
                DW      1110000011000001B
                DW      1110000011000001B
;-----------------
;
floppyBootICON  DW      1111111111111111B
                DW      0000000000000011B
                DW      0000000000000011B
                DW      0000000000000011B
                DW      0000000000000011B
                DW      0000000000000011B
                DW      0000001100000011B
                DW      0111011110000011B
                DW      0000001100000011B
                DW      0000000000000011B
                DW      0000000000000011B
                DW      0000000000000011B
                DW      0000000000000011B
                DW      0000000000000011B
                DW      0000000000000011B
                DW      1111111111111111B
;-----------------
;
;floppyHdCleanICON DW    1111111111111111B
;               DW      1111111110111111B
;               DW      1101011100011111B
;               DW      1010110000000001B
;               DW      0101010000000000B
;               DW      1010111100011101B
;               DW      1101011100000111B
;               DW      1111111011111011B
;               DW      1111111011111011B
;               DW      1111111011111011B
;               DW      1111111011111011B
;               DW      1111111011111011B
;               DW      1111111011111011B
;               DW      1111111011111011B
;               DW      1111111000000011B
;               DW      1111111111111111B

;-----------------

availableICONs  DW      diskBootICON,floppyBootICON,ethernetBootICON

;-----------------

;arrowCursor    DW      0000000001111111B
;               DW      0000000011111111B
;               DW      0000111111111111B
;               DW      0000011111111111B
;               DW      0010001111111111B
;               DW      0011000111111111B
;               DW      0011100011111111B
;               DW      0111110001111111B
;               DW      0111111000111111B
;               DW      1111111100011111B
;               DW      1111111110001111B
;               DW      1111111111100011B
;               DW      1111111111110001B
;               DW      1111111111111000B
;               DW      1111111111111110B
;-----------------

;arrowCursor    DW      0111111111111111B
;               DW      0011111111111111B
;               DW      0001111111111111B
;               DW      0000111111111111B
;               DW      0000011111111111B
;               DW      0000001111111111B
;               DW      0000000111111111B
;               DW      0000000011111111B
;               DW      0010001111111111B
;               DW      0110011111111111B
;               DW      1111001111111111B
```

```
;               DW      1111001111111111B
;               DW      1111100111111111B
;               DW      1111100111111111B
;               DW      1111110011111111B
;               DW      1111110011111111B
;------------------

diagnosticsICON DW      1110000011111111B
                DW      1111001001111111B
                DW      1111100101111111B
                DW      0111110101111111B
                DW      0011100101111111B
                DW      0001001101111111B
                DW      0100011100111111B
                DW      0011111110111111B
                DW      1000000011011111B
                DW      1111111001001111B
                DW      1111111100100111B
                DW      1111111110110011B
                DW      1111111111011001B
                DW      1111111111001100B
                DW      1111111111100001B
                DW      1111111111110011B
;------------------

;iconBorder     DW      1000000000000000B, 0000000000000000B, 0000000000000000B, 0000000000000010B
;               DW      0111111111111111B, 1111111111111111B, 1111111111111111B, 1111111111111001B
;               DW      0111111111111111B, 1111111111111111B, 1111111111111111B, 1111111111111000B
;               DW      0111111111111111B, 1111111111111111B, 1111111111111111B, 1111111111111000B
;               DW      0111111111111111B, 1111111111111111B, 1111111111111111B, 1111111111111000B
;               DW      0111111111111111B, 1111111111111111B, 1111111111111111B, 1111111111111000B
;               DW      0111111111111111B, 1111111111111111B, 1111111111111111B, 1111111111111000B
;               DW      0111111111111111B, 1111111111111111B, 1111111111111111B, 1111111111111000B
;               DW      0111111111111111B, 1111111111111111B, 1111111111111111B, 1111111111111000B
;               DW      0111111111111111B, 1111111111111111B, 1111111111111111B, 1111111111111000B
;               DW      0111111111111111B, 1111111111111111B, 1111111111111111B, 1111111111111000B
;               DW      0111111111111111B, 1111111111111111B, 1111111111111111B, 1111111111111000B
;               DW      0111111111111111B, 1111111111111111B, 1111111111111111B, 1111111111111000B
;               DW      0111111111111111B, 1111111111111111B, 1111111111111111B, 1111111111111000B
;               DW      0111111111111111B, 1111111111111111B, 1111111111111111B, 1111111111111000B
;               DW      0111111111111111B, 1111111111111111B, 1111111111111111B, 1111111111111000B
;               DW      0111111111111111B, 1111111111111111B, 1111111111111111B, 1111111111111000B
;               DW      0111111111111111B, 1111111111111111B, 1111111111111111B, 1111111111111000B
;               DW      0111111111111111B, 1111111111111111B, 1111111111111111B, 1111111111111000B
;               DW      0111111111111111B, 1111111111111111B, 1111111111111111B, 1111111111111000B
;               DW      0111111111111111B, 1111111111111111B, 1111111111111111B, 1111111111110000B
;               DW      0000000000000000B, 0000000000000000B, 0000000000000000B, 0000000000000000B
;               DW      1000000000000000B, 0000000000000000B, 0000000000000000B, 0000000000000001B
;               DW      1100000000000000B, 0000000000000000B, 0000000000000000B, 0000000000000010B

;------------------
;
;--------------------------------------------------------------------------------
;EPROM Boot Data:

bootDeviceProc  DW      unKnownDevice, DiskBootStrap, FloppyBootStrap
                DW      AltEthernetBoot, unKnownDevice, DiskDiag
                DW      FloppyDiag, EtherDiag, unKnownDevice
                DW      unKnownDevice, unKnownDevice


bootDevicesMask         EQU     01E0H
bootDevicesToLow        EQU     5
floppyAndEthernet       EQU     6
dontTrustEEProm         EQU     8000H
shortDiag               EQU     0
longDiag                EQU     1
DiagLEDsTo2             EQU     200H            ;off-on-off  This should match def in Preboot.
;--------------------------------------------------------------------------------


;-- Initial entry:
;-------------- Initialize the task bootTask, Jam it, and initialize
;-------------- bootStrapTask to start with the user interface.
;-------------- bootTask is the boot device independent task.  It sets up
;--------------|buffers for bootStrapTask to fill.  It then interprets those
;--------------|buffers, running initial code, loading control store with
;--------------|microcode, loading ram handlers, etc.  bootStrapTask is the
;--------------|device specific task.  It knows something about the boot
;--------------|devices.  It fills the buffers with data from the boot device.
;--------------|Communication between these two tasks is accomplished by the
;--------------|conditions getBootFile, bootBufferFull, bootBufferEmpty, and
;--------------|finishedLoaderFileFetch.
;--------------|--------                                                 |
;--------------------------------------------------------------------------------


BootStrapInit           PROC    FAR
                        %InitializeTask (BootStrapHandlerID,OFFSET bootTask,JamBootTask,OFFSET BootStack)
JamBootTask:            %Jam    (BootStrapHandlerID,OFFSET bootTask)     ;to be restarted by
                                                        ;device specific code.
                                                        ;to be resolved later.
                        %InitializeTask (BootStrapHandlerID,OFFSET bootStrapTask,UserInterface,OFFSET BootStrapStack)
                        RET
```

```
BootStrapInit           ENDP

;-- User Interface:
;Put up the icons.  Wait for a key stroke or a mouse click selecting what
;type of boot to do.
;-----------------------------------------------------------------------------


UserInterface   PROC    FAR
                ASSUME  DS:BootStrapIOR
                MOV     AX, IOPELocalRAM        ;load low RAM address
                MOV     ES, AX                  ;  into ES
                ASSUME  ES:IOPELocalRAM
                CMP     skipUserInterface, normal
                JE      StartUserInterface
                JMP     BootDeviceSelection     ;if doing diagnostics just continue
StartUserInterface:
                %CallHandlerInitProc    (KeyBoardAndMouseHandlerID)
                %WaitForSystem

                PUSH    ES
                MOV     diagType, shortDiag
                CMP     prebootSwitches, dontTrustEEProm        ;did preboot indicate checksum is bad?
                JNE     CheckFunctionKey9       ;no, go see if user wants to ignore anyway
                MOV     AX, KeyBoardAndMouseIOR ;yes, can't proceed until user hits function key 9
                MOV     ES, AX
                ASSUME  ES: KeyBoardAndMouseIOR
                MOV     AX, DiagLEDsTo2         ;Set LED's to give indication
                OUT     WriteConfigReg, AX      ; that EEProm is bad.
WaitForF9:      MOV     AL, HexValue
                CMP     AL, T9key
                JE      ClearLEDs
                %WaitForSystem
                JMP     SHORT WaitForF9
ClearLEDs:      MOV     AX, 0                   ;turn off leds that preboot left on to indicate
                OUT     WriteConfigReg, AX      ;that checksum was bad
SetPrebootSwitches:
                MOV     HexValue, 0             ;just in case.....
                POP     ES
                ASSUME  ES:IOPELocalRAM
                MOV     prebootSwitches, dontTrustEEProm
                JMP     SHORT InitDisplay

PauseForF9:     PUSH    BX
                %WaitForSystem
                JMP     SHORT TryAgain
CheckFunctionKey9:
                %GetIntervalTimer               ;returns value in AX
                ADD     AX, 2000                ;wait for ~2 seconds
                PUSH    AX
                %EstablishHandlerAccess(KeyBoardAndMouseHandlerID)
                ASSUME  ES: KeyBoardAndMouseIOR
TryAgain:
                MOV     AL, HexValue
                POP     BX
                CMP     AL, T9key               ;does user want to ignore EEPROM?
                JE      SetPrebootSwitches      ;yes, indicate that
                CALL    HasTimeElapsed          ;hasn't said so yet; see if our waiting time is up
                JL      PauseForF9

ProceedWithUserInterface:
                POP     ES
                ASSUME  ES:IOPELocalRAM
InitDisplay:
                %CallHandlerInitProc    (MaintPanelHandlerID)
                %CallHandlerInitProc    (DisplayHandlerID)
                %WaitForSystem          ;Let display and keyboard handlers run.
PaintScreenDeskTopGrey:
                MOV     AX, DisplayIOR
                MOV     ES, AX
                ASSUME  ES:DisplayIOR
SetUpDisplayMemoryAccess:

                MOV     CX, bitMapOrg
                MOV     DX, CX                          ;The first thing we want to do is paint
                SHR     CX, byteLength                  ;the whole display desktop grey. So first
                SHL     DX, byteLength                  ;get the display memory mapping from
                MOV     CH, extendedBusOpieAddress      ;Opie.
                PUSH    ES
                %EstablishIOPAccess(generalMapRegister,CX-DX)
GotAccess:      MOV     DisplaySegment, ES      ;ES:DI points to beginning of
                MOV     DisplayOffset, DI       ;display memory.
                MOV     displayStartSegment, ES ;Save pointer for later use in
                MOV     displayStartOffset, DI  ; making booting MP codes legible.
                POP     ES                      ;Get ES back to DisplayIOR.
                MOV     BX, DI
                MOV     AX, numberBitsPerLine   ;We are going to write out the pattern a word
                SHR     AX, 4                   ;at a time per each two lines. We therefore
                PUSH    AX                      ;need to know how may words per line this
                MOV     SI, AX                  ;display has. This information is made
                SHL     SI, 1                   ;available to us by the Display handler.
                                                ;The desktop grey pattern is made up of
                                                ;lines of an alternating pattern where each
                MOV     DI, numberDisplayLines  ;pattern takes up two lines at a time. So
                SHR     DI, 2                   ;since we will fill up four lines at a time


```

```
                MOV     displayWidthInBytes, SI  ;we need to know how many sets of four full
                                                 ;display lines the current machine has.



                MOV     CX, DisplaySegment       ;Set ES up to point into display.
                MOV     ES, CX                   ;Use segment override for display.
WholeDisplayLoop:
                PUSHA                            ;Save state and let someone else have a chance.
                %WaitForSystem
                POPA

                MOV     AX, desktopGray ;get background pattern for even lines
                POP     CX
                PUSH    CX
                CALL    DoLine                   ;paint even lines of background pattern
                ROR     AX, 2                    ;get background pattern for odd lines
                POP     CX
                PUSH    CX
                CALL    DoLine                   ;paint odd lines of background pattern
                DEC     DI                       ;are we done yet?
                JNZ     WholeDisplayLoop


DonePainting:   ;we need to special case the last line because the number of display lines is not
                ;an even multiple of 4.  this is true for both 17 and 19 inch displays
                MOV     AX, desktopGray ;get background pattern for even lines
                POP     CX
                PUSH    CX
                CALL    FillLine
AllLinesPainted:
                MOV     AX, ES                   ;Save possibly incremented ES into
                MOV     DisplaySegment, AX       ;DisplaySegment
                MOV     AX, DisplayIOR
                MOV     ES, AX                   ;Now use ES to access displayIOR.
                ASSUME  ES:DisplayIOR
                POP     AX                       ;now figure out how to position the ICONs.
                MOV     CX, AX
                SHL     AX, 1                    ;AX gets bytes per line
                MUL     CS: linesFromBottom      ;The ICONs will be a given distance from the
                SUB     BX, AX                   ;the bottom of the screen regardless of the
                JNC     NoUnderflow              ;size of the screen. Some display memories
                MOV     DX, DisplaySegment       ;are larger than can be represented by 16 bits.
                SUB     DX, crossover64KbBank    ;Move DisplaySegment back a segment
                MOV     DisplaySegment, DX
NoUnderflow:
                ;Calculate bit offset in display line to first icon.
                MOV     AX, numberBitsPerLine    ;Keep icons centered
                SUB     AX, totalWidthICONS      ;If there is ever a display narrower than
                SHR     AX,1                     ;the icons, look out.
                PUSH    AX                       ;Save bit offset of first ICON on stack.
                ;SAR    AX, 3                    ;Convert bit offset to bytes
                ;ADD    BX, AX                   ;BX points to corner of first ICON
                MOV     displayOffset, BX        ;Save the base value of the beginning of the icon area.
                ;displayOffset is now a pointer to the upper lefthand corner of the first icon.

InitializeICONs:

SetUpICONs:     ;Set up the icon's data structure.
                MOV     CX, numberOfSoftKeys             ;The first thing we want to do is to
                MOV     DX, OFFSET bootDeviceICON        ;set up the ICON data
                MOV     BX, DX                           ;structures such
                MOV     DI, Null                         ;that the ICONs are positioned appropriately
                                                         ;on the screen. i.e. towards the bottom
;               MOV     AX, numberDisplayLines           ;of the screen and centered. At the same
;               SUB     AX, distanceFromBottom           ;time we want to link the ICON data
;               MOV     ICONsTopEdge, AX                 ;structures so that we can manipulate the
;               ADD     AX, ICONInnerDepth               ;user interface bitmap a bit more easily.
;               MOV     ICONsBottomEdge, AX
                XOR     SI, SI
                POP     AX                               ;We had saved first ICON bit offset here.
                PUSH    AX                               ;Keep it available.

ChainUpICONDataStructure:

                PUSHA                   ;Save state and let someone else have a chance.
                %WaitForSystem
                POPA


                ADD     AX, iconRelativeXcoordinates[SI]        ;Get bit offset for this icon.
                ADD     DX, SIZE bootDeviceICON          ;Each ICON's data structure will point to
                MOV     [BX].ICONLeftImagePtr, Null      ;both the next ICON on the right and on the
                MOV     [BX].ICONRightImagePtr, Null     ;left. For any side without an ICON, the
                ;MOV    [BX].leftBoundary, AX            ;pointer there is NULL (=0).
                ;MOV    [BX].rightBoundary, AX
                ;ADD    [BX].rightBoundary, softKeyWidth
                MOV     [BX].bootingProcedure, OFFSET unKnownDevice      ;This is not used!!!!!
                MOV     [BX].nextICON, DX
                MOV     [BX].invertedICON, notInverted
                MOV     [BX].previousICON, DI
;               XOR     DX, DX
                SHR     AX, 3
                ADD     AX, displayOffset                ;Make AX pointer to corner of icon
                MOV     [BX].ICONOffset, AX
                ;Prepare for next iteration
                POP     AX                               ;AX gets first ICON bit offset.
```

```
                PUSH    AX                              ;Keep it on the stack.
                MOV     DX, [BX].nextICON
                MOV     DI, BX
                MOV     BX, DX
                INC     SI
                INC     SI
                LOOP    ChainUpICONDataStructure
                POP     AX                              ;Return stack to normal
                MOV     [DI].nextICON, Null             ;Make last pointer null.


GetAvailableDevices:
                MOV     AX, IOPELocalRAM
                MOV     ES, AX
                ASSUME  ES: IOPELocalRAM
                CMP     prebootSwitches, dontTrustEEProm
                JNE     GetDevicesFromEEProm
                MOV     AX, floppyAndEthernet
                JMP     SHORT InitPointers
GetDevicesFromEEProm:
                %ReadEEProm(eePromBooting,1)
                AND     AX, bootDevicesMask     ;just get available devices
                SAR     AX, bootDevicesToLow    ;put in least significant byte
                ;ReadEEPROM(AvailableDevices)   ;jmm:85-03-27
                ;MOV    AX,07                   ;Disk, floppy and ethernet available.

InitPointers:
                MOV     CX, numberOfICONs
                MOV     BX, OFFSET availableICONs               ;BX points to array of pointers
                                                                ;to icon bitmaps.
                MOV     SI, OFFSET bootDeviceProc+2             ;SI points into array of boot procs.
                MOV     DI, OFFSET bootDeviceCode               ;DI points to array of device codes.
                MOV     BP, OFFSET bootDeviceICON               ;BP points to first icon data structure.
offsetToDiag            EQU             (SIZE bootDeviceICON)*4
offsetToBootDiag        EQU             8       ;offset in bootDeviceProc to get from device to device diag
                                                ;4 devices *2 words each

Initialize:     PUSHA                           ;Save state and let someone else have a chance.
                %WaitForSystem
                POPA

                ROR     AX,1
                JNC     NextDevice                                      ;Skip unavailable devices.
                MOV     DX, CS: [BX]                                    ;DX points to icon bitmap
                MOV     DS: [BP].ICONLeftImagePtr, DX                   ;Set up bitmap pointers
                MOV     DS: [BP][offsetToDiag].ICONLeftImagePtr, DX
                MOV     DS: [BP][offsetToDiag].ICONRightImagePtr, OFFSET diagnosticsICON
                MOV     DX, CS: [DI]                                    ;DX holds device code.
                MOV     DS: [BP].bootDevice, DL                         ;Set boot device.
                MOV     DS: [BP][offsetToDiag].bootDevice, DL           ;Set boot device for diagnostic button
                MOV     DS: [BP].typeOfBoot, normal                     ;Set type of boot
                MOV     DS: [BP][offsetToDiag].typeOfBoot, diagnostic
                MOV     DX, CS: [SI]                                    ;DX points to boot proc
                MOV     DS: [BP].bootingProcedure, DX                   ;Set boot proc
                MOV     DX, CS: [SI+offsetToBootDiag]
                MOV     DS: [BP][offsetToDiag].bootingProcedure, DX     ;Set diagnostic boot proc
NextDevice:     INC     BX                                              ;Point to next available
                INC     BX                                              ;icon
                INC     DI                                              ;Next boot device
                INC     SI                                              ;Next boot device proc
                INC     SI
                MOV     BP, DS: [BP].nextICON                           ;Prepare endloop.
                LOOP    Initialize



SetUpICONSinDisplayMemory:
                ;Dump the ICON images into display memory.
                MOV     BX, DisplaySegment
                MOV     ES, BX                          ;Make ES point into display memory.
                MOV     BX, OFFSET bootDeviceICON       ;Point to first icon data structure.

SetUpAllICONS:  PUSHA                           ;Save state and let someone else have a chance.
                %WaitForSystem
                POPA

                CALL    SetICONbutton                   ;Paint in button.
                CMP     [BX].ICONLeftImagePtr, 0
                JE      GetTheNextICON
                CALL    SetUpDeviceSymbol
                CALL    SetUPDeviceDiagSymbol
GetTheNextICON: MOV     BX, [BX].nextICON
                OR      BX, BX                          ;This will be zero if the ICON we
                JNZ     SetUpAllICONS                   ;just finished was the last one.


EnableDisplay:  MOV     AX, DisplayIOR                  ;Allow access to Display's IO region.
                MOV     ES, AX
                ASSUME  ES: DisplayIOR
;ES must be pointing into DisplayIOR when this code is executed.
;SetCursorPattern:
;               XOR     SI, SI
;               XOR     DI, DI
;               MOV     CX, 16
;CursorFillIn:  MOV     AX, arrowCursor[SI]     ;The cursor pattern location is also provided
;               MOV     cursorPattern[DI], AH   ;by the Display handler and is a 32 byte array.
```

```
;               MOV     cursorPattern[DI+1], AL ;
;               INC     SI                      ;We should not initialize the cursor until
;               INC     SI                      ;after we have initialized the display pattern.
;               INC     DI
;               INC     DI
;               LOOP    CursorFillIn

                MOV     displCntl, hideCursorDisplayOn
                MOV     borderLow, 0BBH
                MOV     borderHigh, 0EEH
                ;The display handler procs called through commandProc and
                ;cursorPatternProc think that DS contains displayIOR.
                OR      chngdInfo, commandChngd
                OR      chngdInfo, cursorPatternChngd
                OR      chngdInfo, borderPatternChngd
Selection:      ;Timer stuff.
                MOV     bigTimeOut, timeoutInterval     ;Put number of seconds to wait in timeout counter.

                MOV     AX, IOPELocalRAM                ;Make ES point into
                MOV     ES, AX                          ;data area for
                ASSUME  ES: IOPELocalRAM                ;timeoutEnable.

                MOV     finishMode, backToUI            ;Set flag for diagnostics to know
                                                        ; to go back through the user interface
                                                        ; A default diagnostics boot will
                                                        ; have finishMode set to bootSystem.
                MOV     AL, timeoutEnable               ;Get timeoutEnable flag.
                MOV     allowTimeout, AL                ;Save flag in own ioregion.


SetShortTimer:  %GetIntervalTimer                       ;Put timeOfDay in AX.
                ADD     AX, oneSecond                   ;AX has short termination time in milliseconds
                MOV     littleTimeOut, AX               ;Save short timeout.

SelectionLoop:  %WaitForSystem                          ;Let something happen.
;               XOR     AX, AX                          ;
;               MOV     SI, OFFSET KeyBoardBootSelect   ;
                MOV     DL, T1key
                MOV     AX, KeyBoardAndMouseIOR         ;Set up ES for access
                MOV     ES, AX                          ; to keyboard values.
                ASSUME  ES:KeyBoardAndMouseIOR
                MOV     AL, HexValue                    ;
                MOV     CX, 10                          ;
WhatDevice:     CMP     AL, DL                          ;check for normal boot
                JE      DeviceKnown                     ;
                INC     DL                              ;
                LOOP    WhatDevice

;CheckMouse:    CMP     AL, leftMouseButton             ;Check mouse buttons
;               JE      Click
;               CMP     AL, rightMouseButton
;               JE      Click
;               PUSH    AX                              ;Save AX for stop key checking.  This will come out with the mouse tracking code.
;TrackMouse:    CALL    GetMouseCoords                  ;Get current mouse position.
;               PUSH    DS                              ;Save DS.
;               MOV     CX, DisplayIOR                  ;Set DS to DisplayIOR
;               MOV     DS, CX                          ;
;               MOV     DS: cursorXCoord, AX            ;Tell display handler where cursor is.
;               MOV     DS: cursorYCoord, BX
;               OR      DS: chngdInfo, cursorPositionChngd  ;Have display handler set cursor position.
;               POP     DS                              ;Restore DS to BootStrapIOR>

;               POP     AX                              ;Get the value of HexValue back into  AL..  This should come out with the mouse
tracking ode.

                ;Check for stop key and timeout.
                CMP     AL, STOPkey                     ;Check for stop key.
                JE      StopKeyPushed                   ;Go clear the timeoutEnabled flag.
                TEST    allowTimeout, timeOutEnab       ;Check timeoutEnabled flag.
                JZ      TimeOutDisabled                 ;Go fix stack and do loop again.
                MOV     BX, littleTimeOut               ;Get timeout time into BX.
                CALL    HasTimeElapsed                  ;Returns compare of current-time-of-day, BX (BX is timeout time)
                JL      SelectionLoop                   ;If second over check bigTimeOut otherwise wait some more.
                DEC     bigTimeOut                      ;Decrement number of seconds left.
                JNZ     SetShortTimer                   ;If nonzero go back and restart short timer.

TimedOut:       ;Check EEProm for default boot device and default mode.
                ;This implementation assumes that if the eeprom is bad the default default
                ; is to wait indefinitely for user input.
                MOV     AX, IOPELocalRAM
                MOV     ES, AX
                ASSUME  ES:IOPELocalRAM
                MOV     finishMode, bootSystem          ;Tell diagnostics not to return to UI
                %ReadEEProm(eePromBooting, 1)
                JNC     HaveDefaultDevice               ;See if eeprom is ok.
BadEeprom:      ;PUSH   AX                              ;Put filler on stack.
TimeOutDisabled:
StopKeyPushed:  MOV     allowTimeout, timeOutDisab      ;Clear timeout enabled flag.
                JMP     SHORT SelectionLoop             ;Wait for action.
HaveDefaultDevice:
                AND     AX, 0FH                         ;Mask out all but default device bits.
                JZ      TimeOutDisabled                 ;if no default boot devices in EEPROM then disable timeouts
                PUSH    AX                              ;Save AL to make into index for bootdeviceProc
                %ReadEEProm(eePromMisc, 1)              ;Do diags by default?
                JC      BadEeprom                       ;See if eeprom is ok.
                TEST    AL, 1                           ;Test the def diags bit.
                POP     AX                              ;POP does not affect flags.
```

```
                JZ      NoDefDiags
                ADD     AL, diagnostics                 ;Change index to indicate the analagous diag boot.
NoDefDiags:     DEC     AL                              ;There is no zero icon data structure.  The index must be shifted by 1.
                JMP     SHORT GetIconDataStrucPtr       ;Go use AL to get the right icon data structure and boot therefrom.

;FoundKeyMatch: POP     BX                              ;Clear timeoutEnable and
;               POP     BX                              ; timeout time from stack.
;               JMP     DeviceKnown

;Click:         ;A mouse button has been pushed.  See if cursor is over icon.
;               %'EstablishHandlerAccess(KeyBoardAndMouseHandlerID)
;               ASSUME ES: KeyBoardAndMouseIOR
;               CALL    GetMouseCoords                  ;Get current mouse position.
;               CMP     BX, ICONsBottomEdge             ;See if cursor is below icons.
;               JG      NotOverIcon
;               CMP     BX, ICONsTopEdge                ;See if cursor is above icons.
;               JL      NotOverIcon
;CheckXPos:
;               MOV     CX, numberOfSoftKeys            ;Check the icon data structures for the xoocrdinate.
;               MOV     DI, OFFSET bootDeviceICON       ;DI points to first icon datat structure.
;CheckXPosLoop: CMP     AX, DS:[DI].rightBoundary       ;See if cursor is right of the icon.
;               JG      NotThisOne
;               CMP     AX, DS:[DI].leftBoundary        ;See if cursor is left of the icon.
;               JGE     IconSelected
;NotThisOne:    ADD     DI, SIZE ICONDataStructure      ;Point to next one.
;               LOOP    CheckXPosLoop

;NotOverIcon:   MOV     HexValue, 0                     ;Don't see another click until there is one.
;               JMP     TrackMouse                      ;Move the cursor and wait again.


DeviceKnown:    ;Invert the icon corresponding to the key pressed.
                ;PUSH   AX                              ;Save key value.
                ASSUME  ES: KeyBoardAndMouseIOR         ;why does the assembler get confused here?
                MOV     HexValue, 0
                SUB     AL, T1Key                       ;Get key offset.
                CMP     AL, diagnostics                 ;Before we join with the default code
                JL      GetIconDataStrucPtr             ; check for a diagnostic boot and
                MOV     allowTimeOut, timeOutDisab      ; disable timeouts for after diagnostics run.
GetIconDataStrucPtr:                                    ;Default booting enters here.
                MOV     BL, SIZE ICONDataStructure      ;Convert key value to pointer to icon data structure.
                MUL     BL                              ;Make AX index the array of icon data structures.
                MOV     DI, OFFSET bootDeviceICON       ;Set up base of array
                ADD     DI, AX                          ;DI points to correct icon  data structure.
                ;PUSH   SI
                ;CALL   InvertICON                      ;Invert that icon!
                ;POP    SI
                ;POP    AX                              ;Restore key value.
                ;JMP    IconSelected


IconSelected:
;DI points to ICONDataStructure to be inverted.
;After this proc the icon is inverted and AX, CX, SI, BX are smashed.
InvertICON:
                NOT     DS:[DI].invertedICON            ;Toggle inverted flag.
                PUSH    ES
                MOV     AX, DS:displaySegment
                MOV     ES, AX
                MOV     CX, ICONInnerDepth              ;Prepare line counter.
                MOV     SI, DS:[DI].ICONOffset          ;ES:SI points to corner of icon.
InvertOuterLoop:
                XOR     BX, BX                          ;Prepare byte counter.
InvertInnerLoop:
                NOT     BYTE PTR ES:[SI][BX]            ;invert a byte.
                INC     BX
                CMP     BX, softKeyWidth/8              ;bytes across softkey.
                JNE     InvertInnerLoop
                ADD     SI, DS: displayWidthInBytes
                LOOP    InvertOuterLoop
                POP     ES

                ;Paint a square of the upper left hand corner of display memory white
                ; to provide a pretty background for the maintenance panel.  It must be
                ; a little bit bigger than the 16x16 cursor to provide good legibility.
                MOV     SI, displayStartOffset          ;Get pointer to the beginning
                MOV     AX, displayStartSegment         ; of the display bank.
                MOV     ES, AX
                ASSUME  ES:NOTHING                      ;Assembler doesn't know about display bank.
                MOV     AX, OFFFFH                      ;Load big paint brush with white paint.
                MOV     BL, maintPanelBkGrdRightPad     ;Load little brush with a little paint.
                MOV     CX, maintPanelBkGrdHeight       ;Mark out height of area to paint.
PaintSquare:    MOV     ES:[SI], AX                     ;A stroke with the big brush.
                OR      ES:[SI+2], BL                   ;A dainty stroke with the little brush.
                ADD     SI, displayWidthInBytes         ;Move scaffolding down to the next line.
                LOOP    PaintSquare                     ;Are we done yet??

                MOV     CX, DisplayIOR                  ;Return cursor to upper left corner of screen
                MOV     ES, CX
                ASSUME  ES: DisplayIOR
                MOV     cursorXCoord, mpcodeCursorXPos
                MOV     cursorYCoord, mpcodeCursorYPos
                MOV     displCntl, niceMPDisplayOn
                OR      chngdInfo, cursorPositionChngd
                OR      chngdInfo, backgroundChngd
                ;quick slip in a slimy mp code!
                MOV     AX, mpStartBooting
```

```
                CALL    DisplayMPCode                   ;this does a wait for system
                MOV     BX, IOPELocalRAM                ;Set up access to local ram segment.
                MOV     ES, BX
                ASSUME  ES:IOPELocalRAM
                MOV     AL, DS:[DI].bootDevice          ;Set boot device variable.
                CBW                                     ;Make it a word
                MOV     device, AX                      ;

                MOV     AL, allowTimeout                ;Save the state of timeouts
                MOV     timeOutEnable, AL               ; for after diagnostics run,
                                                        ; if they do.

                CMP     DS:[DI].typeOfBoot, normal      ;See if its a diagnostic bot.
                JE      ItsNormal
                MOV     bootType, diagnostic            ;It's diagnostic boot.
                JMP     SHORT CallBootProc
ItsNormal:      MOV     bootType, normal                ;It's normal boot.
CallBootProc:   MOV     AX, [DI].bootingProcedure       ;clearing ioregion will smash this
ZeroBootDeviceIORSpace:
                MOV     BX, OFFSET(bootDeviceIORSpace)  ;clear ioregion for device specific code
                MOV     CX, OverlayLength               ;get number of bytes to be cleared
                XOR     SI, SI
ZeroLoop:       MOV     BYTE PTR [BX + SI], 0
                INC     SI
                LOOP    ZeroLoop
                JMP     AX                              ;Boot.


                ;Set parameters in low memory and start booting.
                ;MOV    BX, IOPELocalRAM
                ;MOV    ES, BX
                ;ASSUME ES:IOPELocalRAM
                ;CMP    AL, CS: [SI]                    ;check for diag boot
                ;JE     NonDiagBoot                     ;
                ;SUB    AX, diagBootKeyBaseValue
                ;MOV    bootType, diagnostic            ;
                ;JMP    GoBoot
;NonDiagBoot:   SUB     AX, bootKeyBaseValue
                ;MOV    bootType, normal                ;

;GoBoot:                MOV     device, AX              ;
                ;CALL   ZeroBootDeviceIORSpace  ;device specific code assumes this is zero and icons munged it

MouseInterpreter:
KeyBoardInterpreter:
DisplayInterface:
BootDeviceSelection:
                ASSUME  ES:IOPELocalRAM
                MOV     SI, device
                SHL     SI, 1
                JMP     WORD PTR CS: bootDeviceProc[SI]


DiskDiag:       CALL    ShortOrLongDiag
                JMP     DiskBootStrap

FloppyDiag:     CALL    ShortOrLongDiag
                JMP     FloppyBootStrap

EtherDiag:      CALL    ShortOrLongDiag                 ;Note this assumes AltEthernetBoot is directly below

AltEthernetBoot:
                CMP     skipUserInterface, normal       ;see note at EtherDiag
                JNE     StartEthernetBoot
                %WaitForTime    (3000)                  ;Give user time to select
                MOV     AX, KeyBoardAndMouseIOR
                MOV     ES, AX
                ASSUME  ES:KeyBoardAndMouseIOR
                MOV     AL, HexValue                    ;an alternate boot.
                SUB     AL, numberKeyBaseValue          ;Choices are 1 thru 7.
                CMP     AL, 01                          ;
                JL      DefaultAltBoot                  ;
                CMP     AL, 07
                JLE     SelectedAltBoot
DefaultAltBoot: MOV     AL, 0                           ;alternate suffix
SelectedAltBoot:
                MOV     BX, IOPELocalRAM
                MOV     ES, BX
                ASSUME  ES:IOPELocalRAM
                ;JMP    SHORT ProceedEthernetBoot

;EthernetBoot:  XOR     AL, AL                          ;primary suffix

ProceedEthernetBoot:
                MOV     CX, 6                           ;
                MOV     DI, OFFSET baseEthernetFileID   ;
                MOV     SI, OFFSET ethernetBootFilePrefix
                CLD
StoreEthernetFileID:
                MOVS    BYTE PTR ES:[DI], CS:[SI]       ;
                LOOP    StoreEthernetFileID
                OR      ES:[DI - 1], AL ;set suffix  (by OR'ing, not MOV'ing!).
StartEthernetBoot:
                JMP     EthernetBootStrap

ShortOrLongDiag PROC    NEAR
                PUSH    ES
```

```
                %GetIntervalTimer
                ADD     AX, 1000                        ;wait for ~ 1 second
                PUSH    AX
                MOV     CX, device                              ;push pointer to IOPELocalRam
                ADD     CL, T4key                       ;determine if same key was pressed again to
WaitForLong:    %EstablishHandlerAccess(KeyBoardAndMouseHandlerID)
                ASSUME ES: KeyBoardAndMouseIOR
                CMP     HexValue, CL
                POP     BX
                POP     ES
                ASSUME ES: IOPELocalRam
                JE      SetLongDiag
                CALL    HasTimeElapsed
                JGE     ShortOrLongRet
                PUSH    ES
                PUSH    BX
                JMP     SHORT WaitForLong
SetLongDiag:    MOV     diagType, longDiag


ShortOrLongRet: RET
ShortOrLongDiag ENDP


;PARAMETERS:    BX has been set with the value that the interval timer should read when
;time has elapsed.
;SMASHES:       AX
;RETURNS        ends with a compare so flags are set accordingly
HasTimeElapsed  PROC    NEAR
                %GetIntervalTimer
                CMP     AX, BX
                RET
HasTimeElapsed  ENDP



FloppyHeadClean:
                ;quick slip in another slimy mp code!
                ;MOV    AX, mpFloppyCleaning
                ;CALL   DisplayMPCode
                ;CALL   DWORD PTR cleanFloppyHeadsPROC

unKnownDevice:
                MOV     AX, mpDeviceUnknown
                CALL    DisplayMPCode
StayHere:       JMP     SHORT StayHere          ;this really should do a Jam



UserInterface   ENDP
                ASSUME  DS:NOTHING, ES:NOTHING

;--------
BitsToDisplayPROCs      PROC    NEAR

                ASSUME  DS:BootStrapIOR

topAndBottomEdgePat     EQU     0h
leftEdgePatByteSw       EQU     0FF7Fh
rightEdgePatByteSw      EQU     0F8FFh
centerPat               EQU     0FFFFh
linesInCenterPat        EQU     20
byteButt                EQU     4

; entry, BX has icon index
;ES to display mem
SetICONbutton:  MOV     SI, [BX].ICONOffset
                CALL    PaintTopBottomEdges
                MOV     CX, linesInCenterPat
PaintCenterPat:
                PUSH    SI
                MOV     ES:[SI], leftEdgePatByteSw
                MOV     ES:[SI+2], centerPat
                MOV     ES:[SI+4], centerPat
                MOV     ES:[SI+6], rightEdgePatByteSw
                POP     SI
                ADD     SI, displayWidthInBytes
                LOOP    PaintCenterPat
                CALL    PaintTopBottomEdges
                CALL    PaintTopBottomEdges
                CALL    PaintTopBottomEdges
                RET
;entry SI point where painting to begin. exit SI points to byte below it in icon pix
;ES to display mem
PaintTopBottomEdges:
                MOV     CX, byteButt
                PUSH    SI
PaintTopBottomLoop:
                MOV     WORD PTR ES:[SI], topAndBottomEdgePat
                INC     SI
                INC     SI
                LOOP    PaintTopBottomLoop
                POP     SI
                ADD     SI, displayWidthInBytes
                RET
```

```
;--------

SetUpDeviceSymbol:
                MOV     DI, [BX].ICONLeftImagePtr       ;If there is a symbol for this ICON
                OR      DI, DI                          ;button, this value should be non-zero
                JZ      LetsGetOut                      ;i.e. it will point to the symbol.
                MOV     AX, displayWidthInBytes
;               XOR     DX, DX
                MOV     SI, vertIconOffsetInButton      ;Icon starts a few lines below top of button.
                MUL     SI                              ;
                MOV     SI, [BX].ICONOffset
                ADD     SI, AX
                ADD     SI, leftIconOffsetInButton      ;Icon starts a few bytes inside of the edge of the button.
                MOV     CX, 16
MoveDeviceSymbol:
                MOV     AX, CS: [DI]
                MOV     ES: [SI], AH
                MOV     ES: [SI+1], AL
                INC     DI
                INC     DI
                ADD     SI, displayWidthInBytes
                LOOP    MoveDeviceSymbol
LetsGetOut:     RET

;--------

SetUPDeviceDiagSymbol:
                MOV     DI, [BX].ICONRightImagePtr      ;If there is a symbol for this ICON
                OR      DI, DI                          ;button, this value should be non-zero
                JZ      GetOut                          ;i.e. it will point to the symbol.
                MOV     AX, displayWidthInBytes
;               XOR     DX, DX
                MOV     SI, vertIconOffsetInButton
                MUL     SI
                MOV     SI, [BX].ICONOffset
                ADD     SI, AX
                ADD     SI, rightIconOffsetInButton     ;Icon starts a few bytes inside of the edge of the button.
                MOV     CX, 16
MoveDiagSymbol:
                MOV     AX, CS: [DI]
                MOV     ES: [SI], AH
                MOV     ES: [SI+1], AL
                INC     DI
                INC     DI
                ADD     SI, displayWidthInBytes
                LOOP    MoveDiagSymbol
GetOut:         RET

BitsToDisplayPROCs      ENDP
;--------
;Background pattern painting procs

DoLine          PROC    NEAR
                PUSH    CX                      ;leave loop counter on stack
                CALL    FillLine                ;paint one line
                POP     CX
                ;ADD    BX, SI
                ;JNC    FinishSecondLine
                ;CALL   BumpES                  ;watch out for crossing 64kb boundary
FinishSecondLine:
                CALL    FillLine                ;paint the second line
                ;ADD    BX, SI                  ;point to the next line
                ;JNC    DoLineDone
                ;CALL   BumpES                  ;watch out for crossing 64kb boundary
DoLineDone:     RET
DoLine          ENDP

FillLine        PROC    NEAR
PaintLine:      MOV     ES:[BX], AX             ;paint a word
                INC     BX                      ;point to the next word
                INC     BX
                JNZ     LoopFillLine
;Move ES into the next block of 64kb
BumpES:         PUSH    AX                      ;Save pattern
                MOV     AX, ES                  ;Change segment register to point
                ADD     AX, crossover64KbBank   ;into next 64K
                MOV     ES, AX
                POP     AX                      ;Recover pattern
LoopFillLine:   LOOP    PaintLine
                RET
FillLine        ENDP


;--------

;GetMouseCoords         PROC            NEAR
;ES points into KB's IORegion.
;Returns mouseX and mouseY byteswapped in AX and BX
;               ASSUME  ES: KeyBoardAndMouseIOR
;GetMouseX:     MOV     AX, mouseX
;               XCHG    AH, AL
;               MOV     BX, mouseY
;               XCHG    BH, BL
;               RET

;GetMouseCoords         ENDP
;--------
```

```
                ASSUME  DS:NOTHING, ES:NOTHING

;-- Ethernet chip initialization:
;--------------  Assume parameter locations upon entry into System procedure
;--------------  are as follows:
;--------------
;--------------|        DS:[DI] points to an Ethernet IOCB                        |
;--------------|        DS:[BX] points to a condition variable                    |
;--------------|        AL contains handler ID for calling task                   |
;--------------|        AH contains broadcast disable & other framing parameters|
;--------------|--------                                                          |
;--------------| Upon exiting this procedure the following will be true:          |
;--------------|                                                                  |
;--------------|                                                                  |
;--------------|                                                                  |
;--------------|                                                                  |
;-------------------------------------------------------------------------------


EtherInitialize PROC    NEAR

                PUSH    ES

SetUpReset:     MOV     DH, iocbReset SHL 4
                CALL    InitEtherIOCB
                CALL    ExecuteCommand
                JZ      i82586InitError

SetUpConfigure: MOV     DH, (iocbCommand SHL 4)+acConfigure
                CALL    InitEtherIOCB
                MOV     SI, OFFSET configureParms
                PUSH    DI
                LEA     DI, [DI].iocbVariant    ;ES already set by InitEtherIOCB
                MOV     CX, configureParmsSize
                CLD
SetUpConfigureLoop:
                MOVS    BYTE PTR ES:[DI], BYTE PTR CS:[SI]
                LOOP    SetUpConfigureLoop
                POP     DI
                MOV     ES:[DI].iocbVariant[8], AH
                CALL    ExecuteCommand
                JZ      i82586InitError

SetUpIndAddr:   MOV     DH, (iocbCommand SHL 4)+acIndAddr
                CALL    InitEtherIOCB
                PUSH    BX
                XOR     BX, BX
                MOV     CX, addrSize
                MOV     DX, ReadHostProm
                PUSH    AX
FirstSetHostProm:
                IN      AL, DX
                MOV     [DI].iocbVariant.cmdParms.indAddr[BX], AL
                INC     BX
                INC     DX
                INC     DX
                LOOP    FirstSetHostProm
                POP     AX
                POP     BX
                CALL    ExecuteCommand
                JZ      i82586InitError

ReceiveFrames:  MOV     DH, iocbStartRU SHL 4
                CALL    InitEtherIOCB
                CALL    ExecuteCommand          ;for startRU, always returns iocbOkay=TRUE

i82586InitError:
                POP     ES
                IRET


EtherInitialize ENDP

;-------------------

ExecuteCommand  PROC    NEAR
                MOV     [DI].iocbCondition.handlerID, AL
                MOV     [DI].iocbCondition.conditionPtr, BX
                OR      [DI].iocbCondition.conditionPtr, nonNilPtr
                MOV     CH, IOPIORegionOpieAddress
                MOV     CL, AL
                PUSH    AX
                %EstablishHandlerAccess (EthernetHandlerID)
                ASSUME  ES:EthernetIOR
                MOV     etherOutQueue.queueNext.OpieAddressLow, DI
                MOV     etherOutQueue.queueNext.OpieAddressHigh, CX
                PUSH    DI
                PUSH    BX
                %NotifyHandlerCondition (EthernetHandlerID,OFFSET etherCmdAvail)
                POP     BX
                PUSH    BX
                %WaitForCondition       (BX,noTimeout)
                POP     BX
                POP     DI
                POP     AX
                TEST    [DI].iocbStatus, MASK iocbOkay
```

```
                RET
ExecuteCommand  ENDP

InitEtherIOCB   PROC    NEAR
                PUSH    AX
                XOR     AL, AL
                MOV     CX, DS
                MOV     ES, CX
                MOV     CX, SIZE IOCB
                PUSH    DI
                CLD
                REP     STOSB
                POP     DI
                MOV     [DI].iocbType, DH
                POP     AX
                RET
InitEtherIOCB   ENDP
                ASSUME  ES:NOTHING


;--


; Call this to put a number into the curser as an MP code.
; On Entry, AX contains number to be displayed.
DisplayMPCode   PROC
                PUSH    ES
                PUSHA
                PUSH    AX                      ;EstablishHandlerAccess smashes this register
                %EstablishHandlerAccess (MaintPanelHandlerID)
                POP     AX
                ASSUME  ES:MaintPanelIOR
                MOV     maintPanelCode, AX
                %NotifyHandlerCondition (MaintPanelHandlerID,OFFSET maintPanelChanged)
                %WaitForSystem
                POPA
                POP     ES
                RET
DisplayMPCode   ENDP



IOPEInROM               ENDS

;*****************************************************************************

                        END
```

```
$MOD186
$PAGELENGTH     (72)
$PAGEWIDTH      (136)

;Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.

;-- stored as [Iris]<WMicro>Dove>ROMDskBt.asm
;-- created on  30-Nov-84 13:44:00

;-- last edited by:

;--     kek                      4-Sep-85 17:36:42      ;mp codes
;--     JPM                      3-Aug-85 10:35:55      ;Change EEPDefs.asm to ROMEEP.asm.
;--     JPM                     25-Jul-85 12:56:27      ;Fix bug in DoDiskOperation (shouldn't use ES).
;--     JPM                     22-Jul-85 13:57:39      ;Change IOPEInROM alignment to WORD, fix bugs.
;--     JPM                     18-Jul-85  8:30:17      ;Opie redesign conversion.
;--     JMM                      9-Jul-85 20:17:58      ;Upgraded to new Disk handler.
;--     JMM                     19-Jun-85 17:10:06      ;Upgraded to new IOPLRAM.asm
;--     JPM                     17-Jun-85  9:07:30      ;Align bootloaded code on page boundary (for disk DMA)
;--     JMM                      4-Apr-85 20:23:06      ;Misc. edit.
;--     DEG                     19-Jan-85 21:06:55      ;First release.
;--     JMM                     30-Nov-84 13:44:00      ;First release.

NAME            ROMDskBt

;-------------------------------------------------------------------------------
;
$NOLIST
$INCLUDE        (HardDefs.asm)
$INCLUDE        (IOPDefs.asm)
$INCLUDE        (IOPMacro.asm)
$INCLUDE        (ROMBDefs.asm)
$INCLUDE        (ROMEEP.asm)
$INCLUDE        (DskIOFce.def)
$LIST

;-------------------------------------------------------------------------------

EXTRN                   BootStrapHandlerID: ABS
EXTRN                   DiskHandlerID: ABS
EXTRN                   DisplayMPCode: NEAR

;-------------------------------------------------------------------------------

;DiskAddress            STRUC   ;Copied from [Iris]<WMicro>Dove>DskHdFce.def

;diskCylinder   DW      ?
;diskSector     DB      ?
;diskHead       DB      ?

;DiskAddress            ENDS


;---------------
;*****************************************************************
;---------------

IOPELocalRAM    SEGMENT AT 0

EXTRN           opieReentry: DWORD
EXTRN           bootType: BYTE
EXTRN           bootRetryCount: WORD
EXTRN           startOfBootBufferSpace: WORD
EXTRN           endOfBootBufferSpace: WORD
EXTRN           HandlerInitProcTable: DWORD

IOPELocalRAM    ENDS

;---------------
;*****************************************************************
;---------------


;from IORROMBt.asm:
;------------------

BootStrapIOR    SEGMENT COMMON

EXTRN           bootStrapTask: TaskContextBlock
EXTRN           bootDeviceIORSpace: DiskBootContext
EXTRN           jumpTable: BootJumpTable

BootStrapIOR    ENDS

;from IORdisk.asm:
;-----------------

DiskIOR         SEGMENT COMMON

EXTRN           diskFCB: DiskFCBRecord

DiskIOR         ENDS

;from STKDisk.asm:
;-----------------

DiskSTK         SEGMENT COMMON
```

```
EXTRN           EndOfDiskStackSeg: WORD

DiskSTK         ENDS

;---------------
;*****************************************************************
;---------------

IOPEInROM       SEGMENT WORD PUBLIC

                ASSUME  CS:IOPEInROM
                ASSUME  DS:BootStrapIOR

;from ROMBoot.asm:
;------------------

PUBLIC          DiskBootStrap

;
;-------------------------------------------------------------------------------
;

;-- Disk BootStrap:
;--------------  This module reads in the first page of DiskInitialDove.db into
;--------------  IOP local RAM and jumps to it. That is all it does other than
;--------------  handling errors encounterd during the attempt to read in the
;--------------  page.
;--------------|                                                               |
;--------------|                                                               |
;--------------|                                                               |
;--------------|                                                               |
;--------------|                                                               |
;--------------|                                                               |
;--------------|--------                                                       |
;--------------| Upon exiting this module the following will be true:          |
;--------------|                                                               |
;--------------|                                                               |
;--------------|                                                               |
;--------------|                                                               |
;--------------|                                                               |
;--------------|                                                               |
;-------------------------------------------------------------------------------

DiskBootStrap           PROC

StartDiskBoot:          MOV     AX, IOPELocalRAM
                        MOV     ES, AX
                        ASSUME  ES:IOPELocalRAM
                        %CallHandlerInitProc    (DiskHandlerID)
                        %WaitForSystem                          ;Give disk a chance to initialize
                        MOV     WORD PTR bootDeviceIORSpace.diskIOCBDone, Null   ; Initialize condition

;   The first thing that gets done is the IOCB less the Disk Operation Block (DOB) portion is zeroed out.

                        MOV     DI, OFFSET bootDeviceIORSpace.diskBootIOCB
                        MOV     CX, (SIZE DiskIOCBRecord - SIZE DiskDOB) / 2
                        XOR     AX, AX
ZeroIOCBLoop:           MOV     WORD PTR [DI], AX
                        ADD     DI, 2
                        LOOP    ZeroIOCBLoop


;   The next thing to do is to copy the image of the initial DOB to the real
; space for the DOB in the IOCB.

                        PUSH    ES
                        MOV     AX, DS
                        MOV     ES, AX
                        ASSUME  ES:NOTHING
                        MOV     SI, OFFSET DiskInitialDOB
                        MOV     DI, OFFSET bootDeviceIORSpace.diskBootIOCB.diskOperationBlock
                        PUSH    DI                              ;Save the offset to the DOB on the stack for now.
                        MOV     CX, (SIZE DiskDOB / 2)
CopyLoop:               MOVS    WORD PTR ES:[DI], CS:[SI]       ;passed on to the handler.
                        LOOP    CopyLoop                        ;can't use REP MOVS w/seg override: might get interrupted
                        POP     DI                              ;Fetch the offset to the DOB from the stack.


;   The next series of steps page aligns the buffer to be used to read in the first record from the disk.

                        POP     ES                              ;Booting uses IOP local RAM
                        ASSUME  ES:IOPELocalRAM                 ;for all its work.
                        MOV     DX, DiskSTK                     ;The first record of
                        SHL     DX, Nibble                      ;DiskInitialDove.db goes
                        ADD     DX, OFFSET EndOfDiskStackSeg    ;after the disk stack segment.
                        ADD     DX, pageSizeInBytes-1           ;The location to jump to
                        AND     DX, NOT(pageSizeInBytes-1)      ;must be at a page
                        MOV     startOfBootBufferSpace, DX      ;boundary.
                        MOV     bootDeviceIORSpace.diskBootIOCB.diskDataPtrLow, DX
                        MOV     CH, IOPLogicalOpieAddress
                        MOV     CL, 0
                        MOV     bootDeviceIORSpace.diskBootIOCB.diskDataPtrHigh, CX
                        SHR     DX, Nibble                      ;Calculate the CS value of
                        MOV     jumpTable.iopEntry, nullOffset  ;the loaded code's
                        MOV     jumpTable.iopEntryCS, DX        ;entry point.
```

```
                        MOV     DX, DiskIOR                     ;Limit the size
                        SHL     DX, Nibble                      ;of the boot
                        MOV     endOfBootBufferSpace, DX        ;area.


;the next instructions set up the disk handler's IOP client condition
; to point to the first word of boot IOR space (overlays first word of IOCB)
                        PUSH    ES
                        MOV     AX, DiskIOR
                        MOV     ES, AX
                        ASSUME  ES:DiskIOR
                        MOV     diskFCB.rd0.diskIOPClientCondition.handlerID, LOW BootStrapHandlerID
                        MOV     diskFCB.rd0.diskIOPClientCondition.conditionPtr, OFFSET bootDeviceIORSpace.diskIOCBDone+nonNilPtr
                        MOV     diskFCB.rd0.diskIOPClientCondition.clientMask, 0
                        POP     ES
                        ASSUME  ES:IOPELocalRAM

ResetDiskHeads:
                        MOV     bootDeviceIORSpace.diskBootIOCB.diskPageCount, 1
                        MOV     [DI].diskOperation, RecalibrateDisk
                        MOV     bootDeviceIORSpace.diskBootIOCB.diskDataXferDirection, noDataOp

; display mp code that initial is ready to start fetching
                        PUSH    AX
                        MOV     AX, mpFetchInitial
                        CALL    DisplayMPCode
                        POP     AX

                        CALL    DoDiskOperation

                        JNC     DiskReadLabelAndData
                        DEC     bootRetryCount
                        JZ      FatalDiskBootError
                        JMP     ResetDiskHeads

DiskReadLabelAndData:
                        MOV     bootDeviceIORSpace.diskBootIOCB.diskPageCount, 1
                        MOV     [DI].diskOperation, ReadDiskLabelAndData
                        MOV     bootDeviceIORSpace.diskBootIOCB.diskDataXferDirection, read

                        CALL    DoDiskOperation

                        JNC     GoToRAM
                        DEC     bootRetryCount
                        JZ      FatalDiskBootError
                        JMP     DiskReadLabelAndData


GoToRAM:                JMP     DWORD PTR jumpTable.iopEntry

FatalDiskBootError:     MOV     AX, mpInitialError
                        CALL    DisplayMPCode
                        %Jam    (BootStrapHandlerID,OFFSET bootStrapTask)
                        %WaitForSystem          ; never returns!


DiskBootStrap           ENDP

;-------------------------------------------------------------------------------

;-- Disk Operations:
;--------------  This procedure after some initialization notifies the disk
;--------------  handler to perform the required operation.
;--------------|                                                               |
;--------------|                                                               |
;--------------|                                                               |
;--------------|                                                               |
;--------------|                                                               |
;--------------|                                                               |
;--------------|--------                                                       |
;--------------| Upon exiting this procedure the following will be true:       |
;--------------|                                                               |
;--------------|                                                               |
;--------------|        CARRY flag is set on error and reset on success        |
;--------------|                                                               |
;--------------|                                                               |
;--------------|                                                               |
;--------------|                                                               |
;-------------------------------------------------------------------------------


DoDiskOperation         PROC    NEAR

DoDiskOperationEntry:

;   The next block of code (re-)initializes some fields within the IOCB.
                        MOV     AL, FALSE
                        MOV     bootDeviceIORSpace.diskBootIOCB.diskInProgress, AL
                        MOV     bootDeviceIORSpace.diskBootIOCB.diskComplete, AL
                        MOV     bootDeviceIORSpace.diskBootIOCB.diskOperationBlockError, AL
                        MOV     bootDeviceIORSpace.diskBootIOCB.controllerErrorType, AL
                        MOV     bootDeviceIORSpace.diskBootIOCB.dmaErrorType, AL
                        MOV     bootDeviceIORSpace.diskBootIOCB.diskError, AL
                        XOR     AX, AX
```

```
                        MOV     [DI].diskHeaderError, AX
                        MOV     [DI].diskLabelError, AX
                        MOV     [DI].diskDataError, AX
                        MOV     [DI].diskLastError, AX
                        MOV     [DI].diskMinusSectorCount, -1


;   These next few lines of code setup the address of the Input/Output Context Block (IOCB) that is to be used to read the
; initial record from the DoveDiskInitial.db file.  First CX:DX will be setup with the segment and offset to the space for
; the IOCB.  Next the address is back converted to an OPIE IOP logical address.  And finally the address is enqueued for the
; rigid disk handler.
                        MOV     DX, OFFSET bootDeviceIORSpace.diskBootIOCB
                        MOV     CH, IOPIORegionOpieAddress
                        MOV     CL, LOW BootStrapHandlerID
                        PUSH    ES
                        MOV     AX, DiskIOR
                        MOV     ES, AX
                        ASSUME  ES:DiskIOR
                        MOV     diskFCB.rd0.diskIOPNextHigh, CX
                        MOV     diskFCB.rd0.diskIOPNextLow, DX
                        MOV     diskFCB.diskStartHandlerForIOP, TRUE
                        POP     ES
                        ASSUME  ES:IOPELocalRAM
                        PUSH    DI


;   Everything is now setup to carry out the disk operation.  The rigid disk
; handler is notified that it has work to do and then this disk booting task falls
; asleep waiting for the disk code to complete.

                        %NotifyHandlerCondition (DiskHandlerID,OFFSET diskFCB.diskConditionWork)
                        %WaitForCondition       (OFFSET bootDeviceIORSpace,noTimeout)


;   Upon completion check to see if there has been an error or not.  If an error
; occured or if the handler is not in its normal state then go through the error
; exit path which attempts to get the handler back to a valid state.

OperationComplete:
                        POP     DI
                        TEST    bootDeviceIORSpace.diskBootIOCB.diskError, TRUE
                        JNZ     ErrorDoDiskOperationExit
                        CLC
                        RET

ErrorDoDiskOperationExit:
                        STC
                        RET


DoDiskOperation         ENDP
;-------------------------------------------------------------------------------

;-------------------------------------------------------------------------------
;Disk EPROM Boot Data:


;   As far as the IOP is concerned the constants in this region are byte-swapped.
; Any word quantity assigned must be byte-swapped.

;   The fields should be accessed through (diskIOCB).diskOperationBlock.(mumble)
; where (mumble) is a field within the DiskDOB STRUCture in DskHdFce.def .

DiskInitialDOB:                 ;Actually this is of type DiskDOB (DiskOperationBlock)
                DD      0       ;the first four bytes are for the ECC syndrome
                DW      -1      ;the two complement of the number of sectors in transfer
                DB      16      ;number of sectors per track.
                DB      0
                DB      4       ;heads per cylinder
                DB      0
                DB      32h     ;132h = 306d  number of cylinders per drive plus one
                DB      1
                DB      0       ;sectors are numbered 0-15
                ;Above five byte values should be gotten from EEPROM but since we are
                ;only reading one record it does not matter unless the RAM code uses
                ;the values passed here. - jmm :85-07-08.
                DB      0
                DW      -1      ;cylinder where reduced write begins (-1 if not used)
                DB      80h     ;0080h = 128   cylinder where precompensation begins
                DB      0
                DB      OFFH    ;write end count = OFFH
                DB      0
                DW      0       ;header error field
                DW      0       ;label error field
                DW      0       ;data error field
                DW      0       ;last error field
                DW      -1      ;current cylinder
                DB      1       ;1 = CRC, 0 = ECC
                DB      0
                DiskAddress     <0,0,1>         ;Location of DiskInitialDove.db
                DW      0
                DW      0
                DB      0       ;DiskCtlrStatusRec
                DB      0       ;DiskDriveStatusRec
                DB      0       ;restore operation
                DB      0
                DW      0       ;tracks to format count 2's complement
```

```
                DB      (SIZE DiskLabelImage) DUP (0)    ;end of DOB image/template


IOPEInROM               ENDS
;*****************************************************************************

                        END
```

```
;Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.

;-- stored as [Idun]<WMicro>Dove>ROMEEP.asm
;-- created on  14-Feb-84 11:13:22
;-- This file is intended to contain ROM-resident portion of EEProm constants.
;
;-- last edited by:

;--     JPM      2-Aug-85 17:20:34       :Make eePromLowMem and eePromHighMem ROM-resident (display handler uses them).
;--     KEK     31-Jul-85 16:32:25       :created ROMEEP.asm from orig EEPDefs.asm
;--     JPM     14-Nov-84 14:52:53       :Updated for new layout.
;--     JPM     12-Oct-84 16:11:44       :Updated for new layout.
;--     VXS     16-Aug-84 11:04:01       :made EEPromSegment COMMON.
;--     VXS      8-Aug-84 15:43:40       :Made tempEEPromImage so we
                ;know where this thing is. (fixes bug where if wordsInEEProm
                ;wasn't on even 4 word boundary, would be overlap)
;--     JPM     19-Jul-84 11:04:57       :Added eePromLockMode.
;--     VXS     11-Jul-84 16:28:56       :Creation. See IOPDefs.asm for earlier history.

;-------------------------------------------------------------------------------
;
;Definitions for ROM-used offsets within EEProm:
;(only the ROM-resident defs are contained here in uncommented-out form)
;(for actual RAM and BadPage contents, see latest RAMEEP.asm and BadPage.asm)

eePromROMVersion        EQU     ROMSegment+wordEEPromOffset+0
eePromRAMVersion        EQU     ROMSegment+wordEEPromOffset+2
eePromBadPageVersion    EQU     ROMSegment+wordEEPromOffset+4

eePromBooting           EQU     ROMSegment+wordEEPromOffset+6
;       [0..3] = default boot device type
;       [4..4] = with icons or without icons
;       [5..5] = rigid booting allowed
;       [6..6] = floppy booting allowed
;       [7..7] = ethernet booting allowed
;       [8..8] = rs-232-c booting allowed
;       [9..15] = spare

eePromDispType          EQU     ROMSegment+byteEEPromOffset+8
;       [0..0] = 0=no display present, 1=display present
;       [1..1] = 0=black and white display, 1= color display
;       [2..6] = spare
;       [7..7] = 0=bitmap display type, 1=non-bitmap display type
eePromXAlign            EQU     ROMSegment+byteEEPromOffset+9
eePromYAlign            EQU     ROMSegment+byteEEPromOffset+10

eePromKBType            EQU     ROMSegment+byteEEPromOffset+11

eePromRigidSctPerTrk    EQU     ROMSegment+byteEEPromOffset+12
eePromRigidHdPerCyl     EQU     ROMSegment+byteEEPromOffset+13
eePromRigidCylCt        EQU     ROMSegment+wordEEPromOffset+14
eePromRigidRWC          EQU     ROMSegment+wordEEPromOffset+16
eePromRigidPCC          EQU     ROMSegment+wordEEPromOffset+18
eePromRigidType         EQU     ROMSegment+byteEEPromOffset+20

eePromSpare1            EQU     ROMSegment+byteEEPromOffset+21

eePromLowMem            EQU     ROMSegment+wordEEPromOffset+22
eePromHighMem           EQU     ROMSegment+byteEEPromOffset+24

eePromFloppy            EQU     ROMSegment+wordEEPromOffset+42

;-------------------------------------------------------------------------------
;
;Definitions for checksum offsets within EEProm:

eePromChecksum          EQU     ROMSegment+wordEEPromOffset+124
eePromInvChecksum       EQU     ROMSegment+wordEEPromOffset+126


;-------------------------------------------------------------------------------
;
;Definitions for RAM-used offsets within EEProm:
;(these are included in comment form just to be complete initially)
;(for actual RAM and BadPage contents, see latest RAMEEP.asm and BadPage.asm)

;eePromMemSize          EQU     RAMSegment+byteEEPromOffset+25
;       [0..3] = encoded VM size (in increments of 64 map pages)
;                0 => none
;                1 => 64 VM map pages
;                2 => 128 VM map pages
;                3 => 256 VM map pages
;       [4..7] = encoded control store sizes (in increments of 4K pages)
;                0 => none
;                1 => 4K control store
;                2 => 8K control store

;eePromHardwareBuild    EQU     RAMSegment+byteEEPromOffset+26
;       0 => ?none
;       1 => B0/B1
;       2 => B2
;       3-255 => spare incremental encodings

;eePromMisc             EQU     RAMSegment+byteEEPromOffset+27
;       [0..0] = Default boot type bit2 (with or without diagnostics)
;       [1..1] = Default diagnostic boot type bit (short or long)

ROMEEP.asm          2-Aug-85 17:20:36 PDT                                  1
```

```
;       [2..7] = spare

;eePromRS232DCEtype      EQU     RAMSegment+byteEEPromOffset+28
;eePromRS232DCEattr      EQU     RAMSegment+byteEEPromOffset+29
;eePromRS232DTEtype      EQU     RAMSegment+byteEEPromOffset+30
;eePromRS232DTEattr      EQU     RAMSegment+byteEEPromOffset+31

;eePromPCEMemSize        EQU     RAMSegment+wordEEPromOffset+32
;eePromPCEConfig         EQU     RAMSegment+wordEEPromOffset+34

;eePromOption1           EQU     RAMSegment+wordEEPromOffset+36
;eePromOption2           EQU     RAMSegment+wordEEPromOffset+38
;eePromOption3           EQU     RAMSegment+wordEEPromOffset+40

;this is actually defined in the ROM section above.
;eePromFloppy            EQU     ROMSegment+wordEEPromOffset+42

;eePromSpare2            EQU     RAMSegment+byteEEPromOffset+44
;eePromSpare3            EQU     RAMSegment+byteEEPromOffset+45
;eePromSpare4            EQU     RAMSegment+byteEEPromOffset+46
;eePromSpare5            EQU     RAMSegment+byteEEPromOffset+47
;eePromSpare6            EQU     RAMSegment+byteEEPromOffset+48
;eePromSpare7            EQU     RAMSegment+byteEEPromOffset+49
;eePromSpare8            EQU     RAMSegment+byteEEPromOffset+50
;eePromSpare9            EQU     RAMSegment+byteEEPromOffset+51
;eePromSpare10           EQU     RAMSegment+byteEEPromOffset+52
;eePromSpare11           EQU     RAMSegment+byteEEPromOffset+53
;eePromSpare12           EQU     RAMSegment+byteEEPromOffset+54
;eePromSpare13           EQU     RAMSegment+byteEEPromOffset+55
;eePromSpare14           EQU     RAMSegment+byteEEPromOffset+56
;eePromSpare15           EQU     RAMSegment+byteEEPromOffset+57
;eePromSpare16           EQU     RAMSegment+byteEEPromOffset+58
;eePromSpare17           EQU     RAMSegment+byteEEPromOffset+59
;eePromSpare18           EQU     RAMSegment+byteEEPromOffset+60
;eePromSpare19           EQU     RAMSegment+byteEEPromOffset+61
;eePromSpare20           EQU     RAMSegment+byteEEPromOffset+62
;eePromSpare21           EQU     RAMSegment+byteEEPromOffset+63
;eePromSpare22           EQU     RAMSegment+byteEEPromOffset+64
;eePromSpare23           EQU     RAMSegment+byteEEPromOffset+65
;eePromSpare24           EQU     RAMSegment+byteEEPromOffset+66
;eePromSpare25           EQU     RAMSegment+byteEEPromOffset+67

;-------------------------------------------------------------------------------
;
;Definitions for Bad Page offsets within EEProm:
;(these are included in comment form just to be complete initially)
;(for actual RAM and BadPage contents, see latest RAMEEP.asm and BadPage.asm)

;eePromLastParityErrLow  EQU     badPageSegment+wordEEPromOffset+68
;eePromLastParityErrHigh         EQU     badPageSegment+wordEEPromOffset+70

;eePromBadPageLow1       EQU     badPageSegment+wordEEPromOffset+72
;eePromBadPageHigh1      EQU     badPageSegment+wordEEPromOffset+76
;eePromBadPageLow2       EQU     badPageSegment+wordEEPromOffset+74
;eePromBadPageHigh2      EQU     badPageSegment+wordEEPromOffset+78
;eePromBadPageLow3       EQU     badPageSegment+wordEEPromOffset+80
;eePromBadPageHigh3      EQU     badPageSegment+wordEEPromOffset+82
;eePromBadPageLow4       EQU     badPageSegment+wordEEPromOffset+84
;eePromBadPageHigh4      EQU     badPageSegment+wordEEPromOffset+86
;eePromBadPageLow5       EQU     badPageSegment+wordEEPromOffset+88
;eePromBadPageHigh5      EQU     badPageSegment+wordEEPromOffset+90
;eePromBadPageLow6       EQU     badPageSegment+wordEEPromOffset+92
;eePromBadPageHigh6      EQU     badPageSegment+wordEEPromOffset+94
;eePromBadPageLow7       EQU     badPageSegment+wordEEPromOffset+96
;eePromBadPageHigh7      EQU     badPageSegment+wordEEPromOffset+98
;eePromBadPageLow8       EQU     badPageSegment+wordEEPromOffset+100
;eePromBadPageHigh8      EQU     badPageSegment+wordEEPromOffset+102
;eePromBadPageLow9       EQU     badPageSegment+wordEEPromOffset+104
;eePromBadPageHigh9      EQU     badPageSegment+wordEEPromOffset+106
;eePromBadPageLow10      EQU     badPageSegment+wordEEPromOffset+108
;eePromBadPageHigh10     EQU     badPageSegment+wordEEPromOffset+110
;eePromBadPageLow11      EQU     badPageSegment+wordEEPromOffset+112
;eePromBadPageHigh11     EQU     badPageSegment+wordEEPromOffset+114
;eePromBadPageLow12      EQU     badPageSegment+wordEEPromOffset+116
;eePromBadPageHigh12     EQU     badPageSegment+wordEEPromOffset+118
;eePromBadPageLow13      EQU     badPageSegment+wordEEPromOffset+120
;eePromBadPageHigh13     EQU     badPageSegment+wordEEPromOffset+122
```

```
$MOD186
$PAGELENGTH (72)
$PAGEWIDTH  (136)

;Copyright (C) 1985 by Xerox Corporation. All rights reserved.


;-- STKBoot.asm defines stacks used by the bootstrap handler.

;-- stored as [Iris]<WMicro>Dove>STKBoot.asm
;-- created on  11-Jul-85  8:54:24
;
;-- last edited by:

;--     JPM             11-Jul-85  8:54:24      :Created.

NAME                    STKBoot
;-------------------------------------------------------------------------------
$NOLIST
$INCLUDE                (IOPStack.asm)
$LIST
;*******************************************************************************

BootStrapSTK            SEGMENT         COMMON

                        %StackAllocation        (BootStack,)
                        %StackAllocation        (BootStrapStack,)

BootStrapSTK            ENDS

;*******************************************************************************

                        END
```

```
;$MOD186
;$PAGELENGTH (95)
;$PAGEWIDTH  (136)

;Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.


;-- IORegion locations for the rigid disk handler.

;-- stored as [Idun]<WDLion>Dove>DskHdFce.def
;
;-- last edited by:


;--     DEG             13-Oct-84 19:34:12      :created by
; taking the hardware constants from DskIOfce.asm.
;--     DEG             15-Oct-84 14:01:01      :introduced
; diskMapRegister which is currently EQUated to 0.  It really needs to be
; equated to generalMapRegister in the real system.
;--     DEG             22-Oct-84 23:58:51      :introduced
; new definitons and redefined DOB according to Bob RXS    's msg of
; 19-Oct-84 10:06:00 PDT.
;--     DEG             26-Oct-84  2:01:54      :edited
;--     DEG              2-Dec-84 14:54:50      :include harddefs
;--     DEG             10-Dec-84  2:03:52      :convert to use the generalMapRegister
;--     DEG             13-Dec-84 14:25:20      :convert back to using map register 0 for the blue-box.
;--     DEG             19-Dec-84 15:03:27      :use the generalMapRegister.
;--     DEG              9-Jan-85 15:36:57      :change name to DskHdFce.def from DskHdFce.asm.
;--     DEG             11-Jan-85 15:24:55      :add definition for disk address.
;--     DEG             13-Jan-85  2:49:40      :Add constants for FIFO problems.
;--     DEG             15-Jan-85 23:00:05      :Fix problems in definitions.
;--     DEG             18-Jan-85 21:59:30      :Correct a definition.
;--     DEG             20-Jan-85  9:02:45      :Correct a definition.
;--     DEG             18-Feb-85 23:08:00      :Make DOB
; definition match the definition in DiskIOFaceDove.mesa.
;--     DEG             28-Feb-85 12:10:16      :Add a new
; constant diskPageSizeInBytes.
;--     DEG             27-Mar-85  2:34:29      :remove INCLUDEs.
;--     DEG             15-Apr-85 13:49:41      :add special error to disk status register.


; All sizeMumbles are in bytes

$NOGEN

DiskAddress             STRUC

diskCylinder    DW      ?
diskSector      DB      ?
diskHead        DB      ?

DiskAddress             ENDS


DiskLabelImage          STRUC

diskFileID              DB      10 DUP (?)
diskFilePageLow DW      ?
diskFPHighPage0Attr     DW      ?
diskAttrInAllPages      DW      ?
diskDontCare            DD      ?

DiskLabelImage          ENDS

sizeDiskLabelImage      EQU     SIZE DiskLabelImage


DiskDOB                 STRUC

diskECCSyndrome         DD      ?
diskMinusSectorCount    DW      ?
diskSecPerTrack         DB      ?
diskZero1               DB      ?
diskHdsPerCyl           DB      ?
diskZero2               DB      ?
cylPerDrive             DW      ?
diskStartSec            DB      ?
diskZero3               DB      ?
diskReducedWriteCyl     DW      ?
diskPreCompCyl          DW      ?
diskWriteEndCnt         DB      ?
diskZero4               DB      ?
diskHeaderError         DW      ?
diskLabelError          DW      ?
diskDataError           DW      ?
diskLastError           DW      ?
diskCurrentCyl          DW      ?
diskECCFlag             DB      ?
diskZero5               DB      ?
diskHeader              DD      ?
diskReserved1           DW      ?
diskReserved2           DW      ?
diskCtlrStatus          DB      ?       ;Actually of DiskCtlrStatusRec
diskDriveStatus         DB      ?       ;actually of DiskDriveStatusRec
diskOperation           DB      ?
diskZero6               DB      ?
diskMinusFmtCnt         DW      ?

DskHdFce.def        21-Apr-85 12:31:12 PDT                                  1
```

```
diskLabel               DB      (SIZE DiskLabelImage) DUP (?)

DiskDOB                 ENDS

sizeDiskDOB             EQU     SIZE DiskDOB
dobSize                 EQU     sizeDiskDOB
minusDOBSize            EQU     0 - dobSize

;-------------------------------------------------

;  --> disk status record definitions <--

DiskDriveStatusRec      RECORD  dskDriveNotReady: 1, dskSeekNotComplete: 1, dskUnu0: 1, dskAddrMark: 1, dskNotStoredIndxMrk: 1,
dskNotTrack0: 1, dskNotWriteFault: 1, dskLock: 1

DiskCtlrStatusRec       RECORD  dskReadDataFound: 1, dskNotBDone: 1, dskFIFOEmptyAtRead: 1, dskNotSPMA3: 1, dskNotSPMAMaxCnt: 1,
dskA1B1Same: 1, dskFIFOEmpty: 1, dskFIFOFull: 1

;-------------------------------------------------

;  --> disk controller's commands <--

DiskCommandReg  RECORD  crStopBit: 1, crDiagMode: 1, crUnused: 2, crUnuCmd: 2, crCmd: 2

goToIdleLoop            EQU     0
xferDOBToController     EQU     1
executeDOB              EQU     2
xferDOBFromController   EQU     3
inIdleLoop              EQU     goToIdleLoop


;-------------------------------------------------

;  --> disk controller's status <--

DiskStatusReg   RECORD  srErrorBit: 1, srDone: 1, srSpecialError: 2, srUnused: 2, srCmd: 2

statusRegisterErrorBit  EQU     MASK srErrorBit
diskRDCDone             EQU     MASK srDone

noDiskSpecialError      EQU     0
diskFIFOError           EQU     10h
diskSpecialError0       EQU     20h
diskSpecialError1       EQU     30h


;-------------------------------------------------

;  --> disk DMA's status <--

DiskDMAReg      RECORD  diskDMAFIFODir: 1, diskDMABb1: 1, diskDMAFIFOFullBar: 1, diskDMAFIFOEmptyBar: 1, diskDMAFIFOBoundsBar: 1,
diskDMAEndOfXferBar: 1, diskDMARunStateMachBar: 1, diskDMAErrorBit: 1

DiskDMAStatusMask       EQU     33h
DiskDMAIsInResetState   EQU     20h


;-------------------------------------------------

;  --> Rigid Disk DMA Hardware Constants <--

diskDMAMode0    EQU     0
diskDMAMode1    EQU     2
diskDMAMode2    EQU     4
diskDMAMode3    EQU     6

diskDMAMode     EQU     diskDMAMode3

;-------------------------------------------------

;  --> FIFO commands <--

fifoToMemory    EQU     0
memoryToFIFO    EQU     -1
noDataXfer      EQU     1


;-------------------------------------------------

;  --> Interesting Rigid Disk Hardware Ports <--

DiskReadDMAWordCount            EQU     0204h
DiskReadDMAAddressWordOffset    EQU     0206h
DiskDMAAddressPage              EQU     0208h
DiskDMAAddressWordOffset        EQU     020Ah
DiskWriteDMAWordCount           EQU     020Ch
DiskDMADirection                EQU     0210h
DiskDMAStatus                   EQU     0210h
DiskControllerCommandRegister   EQU     0214h
DiskControllerStatusRegister    EQU     0214h
DiskStartDMA                    EQU     0216h

DiskPresetA1                    EQU     0212h
DiskWrite2942CR2+CR0            EQU     0200h
DiskRead2942CR2+CR0             EQU     0202h
DiskEnable2942Counters          EQU     020Eh
```

```
;-------------------------------------

;  --> Disk Operations <--

RecalibrateDisk          EQU     0
FormatDiskTracks         EQU     1
ReadDiskData             EQU     2
WriteDiskData            EQU     3
WriteDiskLabelAndData    EQU     4
ReadDiskLabel            EQU     5
ReadDiskLabelAndData     EQU     6
VerifyDiskData           EQU     7


;-------------------------------------

;  --> Disk Operations <--

DiagDiskRead             EQU     1


;-------------------------------------------------

;  --> Other Interesting Hardware Constants <--

diskPageSizeInBytes              EQU     512
diskPageSizeInWords              EQU     256

nilCondition             EQU     0
preNotifiedCondition     EQU     1

nybble          EQU     4
nybbleMask      EQU     0Fh

;   In the real system diskMapRegister needs to be EQU to generalMapRegister.
; For testing on an IOP-only system this value is set to 0.
diskMapRegister          EQU     generalMapRegister
```

```
;$MOD186
;$PAGELENGTH (95)
;$PAGEWIDTH  (136)

;Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.


;-- IORegion locations for the rigid disk handler.

;-- stored as [Idun]<WDLion>Dove>DskIOFce.def
;
;-- last edited by:


;--     DEG             20-Aug-84 10:35:36      :created
;--     DEG             12-Sep-84 16:59:34      :edited for initial cut.
;--     DEG             30-Sep-84 22:56:49      :edited for use SIZE operator.
;--     DEG              2-Oct-84  0:40:21      :fixed DOB definition.
;--     DEG              4-Oct-84 16:29:13      :fixed some constant declarations.
;--     DEG             14-Oct-84  0:11:59      :introduce new values and change definitions.
;--     DEG             18-Oct-84  4:27:25      :introduce new values.
;--     DEG             25-Oct-84 10:01:31      :change disk operation constants.
;--     DEG             26-Oct-84 11:14:18      :change layout of a few records..
;--     DEG             10-Dec-84  2:02:44      :Edited for Opie 17 compatibility.
;--     DEG             16-Dec-84 23:34:50      :Add
; complementDOB to DiskDataRec RECORD so that the client can specify whether
; complementing is necessary or not.
;--     DEG              9-Jan-85 15:35:20      :changed name to DskIOFce.def from DskIOFce.asm
;--     DEG             13-Jan-85  0:42:02      :add handler states 6 & 7.
;--     DEG             14-Jan-85  2:39:20      :add HNH   for getting EEPROM data to disk handler clients.
;--     DEG             15-Jan-85 23:03:30      :remove contention in naming.
;--     DEG             16-Jan-85 10:11:32      :place hook for testing both etch1 and etch2 boards.
;--     DEG             20-Jan-85 16:37:04      :correct some definitions of STRUCtures.
;--     DEG             22-Jan-85  4:10:12      :added another condition variable.
;--     DEG              2-Feb-85 18:39:22      :switched dataPtr high and low in IOCB.
;--     DEG             25-Feb-85 18:47:55      :use defined EEPROM indices.
;--     DEG              2-Mar-85 23:13:53      :size of DOB was
; in words instead of in bytes as it should have been.  Add HNH   for the LEDs to
; be used to display page count.
;--     DEG              4-Mar-85  1:57:23      :Add
; diskPresentCylinder to the DiskDCBRecord so that we can maintain the correct
; cylinder information.
;--     DEG              8-Mar-85  1:51:57      :Adjust double
; word quantities such that the low word appears first.
;--     DEG             27-Mar-85  2:35:02      :remove INCLUDEs.
;--     DEG             29-Mar-85 15:54:38      :convert to Opie!19.
;--     DEG             16-Apr-85 13:43:22      :make changes to data structures
; for proper synchronization between the head and the handler; correct a typo as well.
;--     BKI             10-May-85 20:47:07      :Add controllerErrorType and dmaErrorType.

; All sizeMumbles are in bytes.

$NOGEN

$INCLUDE                (DskHdFce.def)

; Disk data structures

; DiskClientType: TYPE = MACHINE DEPENDENT
;   {drive0MesaClient, drive0IOPClient, drive1MesaClient, drive1IOPClient,
;   drive2MesaClient, drive2IOPClient, drive3MesaClient, drive3IOPClient}


DiskDCBRecord           STRUC

diskMesaHead            DW      ?
diskMesaNext            DW      ?
diskMesaTail            DW      ?
diskBlockMesaQueue      DW      ?
diskIOPHeadLow          DW      ?
diskIOPHeadHigh         DW      ?
diskIOPNextLow          DW      ?
diskIOPNextHigh         DW      ?
diskIOPTailLow          DW      ?
diskIOPTailHigh         DW      ?
diskBlockIOPQueue       DW      ?
diskCurrentDriveMask    DB      ?
diskCommand             DB      ?
diskMesaClientCondition DB      (SIZE ClientCondition) DUP (?)   ;ClientCondition        <>
diskIOPClientCondition  DB      (SIZE ClientCondition) DUP (?)   ;ClientCondition        <>
diskCurrentIOCBLow      DW      ?
diskCurrentIOCBHigh     DW      ?
diskMisc                DW      ?       ;word used exclusively by the Client.
diskSpare0              DB      ?
diskDriveType           DB      ?
diskSectorsPerTrack     DB      ?
diskHeadsPerCylinder    DB      ?
diskCylindersPerDrive   DW      ?
diskReducedWriteCurrentCylinder DW      ?
diskPreCompensationCylinder     DW      ?


DiskDCBRecord           ENDS

sizeDiskDCBRecord       EQU     SIZE DiskDCBRecord



DskIOFce.def        10-May-85 20:53:41 PDT
```

```
DiskFCBRecord           STRUC

diskTask                DB      SIZE taskContextBlock DUP (?)           ;TaskContextBlock       <>
diskDMATask             DB      SIZE taskContextBlock DUP (?)           ;TaskContextBlock       <>
diskConditionDMAWork    DB      SIZE Condition DUP (?)          ;Condition              <>
diskConditionDMADone    DB      SIZE Condition DUP (?)          ;Condition              <>
diskConditionWork       DB      SIZE Condition DUP (?)          ;Condition              <>
diskWorkMask            DW      ?               ;work mask
diskLockMask            DW      ?               ;lock mask
diskMesaCleanupRequest  DW      ?
diskIOPCleanupRequest   DW      ?
diskHandlerStoppedForMesa               DW      ?
diskHandlerStoppedForIOP                DW      ?
diskHandlerStoppedForMesaCleanup        DW      ?
diskHandlerStoppedForIOPCleanup         DW      ?
diskStartHandlerForMesa DW      ?
diskStartHandlerForIOP  DW      ?
diskHandlerState        DW      ?
diskCurrentClient       DB      ?               ;DiskClientType
diskClientsToTest       DB      ?               ;CARDINAL[0..numDiskClients]
diskNumPossibleClients  DB      ?
diskLastDriveMask       DB      ?
diskCurrentDrivePtr     DW      ?               ;Used by handler to save ptr to current drive.
diskStatusRegister      DB      ?
diskCommandRegister     DB      ?
diskTaskRetryCount      DB      ?
diskBadInterruptReason  DB      ?
diskDMAStatusRegister   DB      ?
diskBadDMAInterruptReason       DB      ?
unexpectedDiskInterruptCount    DW      ?
unexpectedDiskDMAInterruptCount DW      ?
rd0                     DB      SIZE DiskDCBRecord DUP (?)      ;DiskDCBRecord  <>
rd1                     DB      SIZE DiskDCBRecord DUP (?)      ;DiskDCBRecord  <>

DiskFCBRecord           ENDS

sizeDiskFCBRecord       EQU     SIZE DiskFCBRecord


DiskDataRec     RECORD  incrementDataPtr:1, complementDOB: 1, etch2: 1, ddrunu1: 2, diskUseLEDs: 1, diskHalt: 1, diagDiskCmd: 1,
dataCommandTransfer: 1, unused2: 6, commandDirection: 1


DiskIOCBRecord          STRUC

diskPrivateStuff        DW      35 DUP (?)

diskDataPtrLow          DW      ?
diskDataPtrHigh         DW      ?
diskDataInfoRec         DW      ?               ;actually of type DiskDataRec.
diskPageCount           DW      ?
diskStopHandlerOnCompletion     DB      ?
diskOnlyDOBFromController       DB      ?
diskError               DB      ?
diskOperationBlockError DB      ?
controllerErrorType     DB      ?
dmaErrorType            DB      ?
diskComplete            DB      ?
diskInProgress          DB      ?
diskDataXferDirection   DB      ?
diskDMATimedOut         DB      ?
diskNextLow             DW      ?
diskNextHigh            DW      ?

diskOperationBlock      DB      SIZE DiskDOB DUP (?)

DiskIOCBRecord          ENDS


sizeDiskIOCBRecord      EQU     SIZE DiskIOCBRecord

;-------------------------------------

fromMesa        EQU     0
toMesa          EQU     MASK commandDirection

;-------------------------------------

NILPtr                  EQU     0
NILOpieAddressHigh      EQU     nilOpieAddress*100h
NILOpieAddressLow       EQU     0

mesaShortPtrOpieAddressType     EQU     mesaEnvBaseWord*100h

;-------------------------------------

;  --> Handler states <--

normalDiskHandlerState  EQU     0
diskControllerNotIdling EQU     1
badDiskInterrupt        EQU     2
badDiskDMAInterrupt     EQU     3
;diskDMAError           EQU     4
resettingDiskDMATask    EQU     5
initialStartDiskDMATask EQU     6
resettingHandler        EQU     -1
```

```
;-------------------------------------

diskControllerError     EQU     1
diskDMAError            EQU     1
diskInterruptTimeout    EQU     2
controllerNotInIdleLoop EQU     3
fifoNotEmpty            EQU     3

;-------------------------------------

DiskClientType  RECORD  dctUnused: 5, diskDrive: 2, diskClient: 1, dctZero: 1

diskClientMask          EQU     MASK    diskClient
diskDriveMask           EQU     MASK    diskDrive

diskMesaClient          EQU     0
diskIOPClient           EQU     2

diskDriveRD0            EQU     0
diskDriveRD1            EQU     4 * 1
diskDriveRD2            EQU     4 * 2
diskDriveRD3            EQU     4 * 3

driveOMesaClient        EQU     diskMesaClient + diskDriveRD0
drive0IOPClient         EQU     diskIOPClient + diskDriveRD0
drive1MesaClient        EQU     diskMesaClient + diskDriveRD1
drive1IOPClient         EQU     diskIOPClient + diskDriveRD1
drive2MesaClient        EQU     diskMesaClient + diskDriveRD2
drive2IOPClient         EQU     diskIOPClient + diskDriveRD2
drive3MesaClient        EQU     diskMesaClient + diskDriveRD3
drive3IOPClient         EQU     diskIOPClient + diskDriveRD3

lastDriveClient         EQU     drive0IOPClient

;-------------------------------------

;       Indexes for EEPROM

eeDiskTypeIndex                 EQU     eePromRigidType
eeDiskSectorsPerTrackIndex      EQU     eePromRigidSctPerTrk
eeDiskHeadsPerCylinderIndex     EQU     eePromRigidHdPerCyl
eeDiskCylinderCountIndex        EQU     eePromRigidCylCt
eeDiskReducedWriteCylinderIndex EQU     eePromRigidRWC
eeDiskPreCompCylinderIndex      EQU     eePromRigidPCC

;-------------------------------------

;       Miscellany

numSupportedDiskDrives  EQU     1
numPossibleDiskDrives   EQU     2
maxNumOfDiskDrives      EQU     4

numDiskClientsPerDrive  EQU     2

numDiskClients          EQU     numSupportedDiskDrives * numDiskClientsPerDrive
numPossibleDiskClients  EQU     numPossibleDiskDrives * numDiskClientsPerDrive

defaultInitialDiskCylinder      EQU     -1

start           EQU     0

read            EQU     0
write           EQU     1
execute         EQU     2
noDataOp        EQU     3

wordSize        EQU     2

noWatchDogTimeOuts      EQU     255

tenMilliseconds         EQU     10

UnexpectedInterrupt     EQU     1
```

```
REM DiskBoot.bat
REM Copyright (C) 1987 by Xerox Corporation. All rights reserved.
BREAK ON

REM ----------------------
REM Build DiskBoot.loc!
REM MUST RUN RAMBOOT.BAT BEFORE THIS!
REM ----------------------

XFILE RET DskBDefs.asm -o
XFILE RET DskHdFce.def -o
XFILE RET DskIOFce.def -o
XFILE RET RAMBDefs.asm -o
XFILE RET ROMBDefs.asm -o
XFILE RET RAMDskBt.asm -o
XFILE RET RAMBoot.lnk -o
XFILE RET IORDisk.obj -o

CRLFTool /lf DskBDefs.asm
DEL DskBDefs.bak
CRLFTool /lf DskHdFce.def
DEL DskHdFce.bak
CRLFTool /lf DskIOFce.def
DEL DskIOFce.bak
CRLFTool /lf RAMBDefs.asm
DEL RAMBDefs.bak
CRLFTool /lf ROMBDefs.asm
DEL ROMBDefs.bak
CRLFTool /lf RAMDskBt.asm
DEL RAMDskBt.bak

RUN ASM86 RAMDskBt.asm DEBUG > RAMDskBt.log
TYPE RAMDskBt.log
PAUSE

RUN LINK86 RAMDskBt.obj, IORDisk.obj, RAMBoot.lnk to DiskBoot.lnk
REM This will cause an unresoved external warning on TransmitFrame.
REM Ignore it.

RUN LOC86 DiskBoot.lnk to DiskBoot.loc PC(PURGE) ADDRESSES (SEGMENTS(IOPEInRAM(0A00H)))

XFILE STORE DiskBoot.loc -o
XFILE STORE DiskBoot.mp2 -o
XFILE STORE DiskBoot.lnk -o
XFILE STORE RAMDskBt.obj -o

REM
REM ****** Then do the following in XDE ******
REM
REM MakeInitialMicroBoot DiskInitialDove.db ← DiskBoot.loc
REM
REM
REM fini
```

```
REM Initial.bat
REM Copyright (C) 1987 by Xerox Corporation. All rights reserved.
BREAK ON

REM ----------------------
REM Build FlpyBoot/DiskBoot/EthrBoot.loc!
REM MUST RUN RAMBOOT.BAT BEFORE THIS!
REM ----------------------

REM This batch file is the concatenation of DiskBoot.bat, EthrBoot.bat, and FlpyBoot.bat.

XFILE RET DsplDefs.asm -o
XFILE RET FlopFace.asm -o
XFILE RET ROMBDefs.asm -o
XFILE RET RAMBDefs.asm -o
XFILE RET FlpBDefs.asm -o
XFILE RET RAMFlpBt.asm -o
XFILE RET DskBDefs.asm -o
XFILE RET DskHdFce.def -o
XFILE RET DskIOFce.def -o
XFILE RET RAMDskBt.asm -o
XFILE RET RAMEthBt.asm -o
XFILE RET EthIOFce.asm -o
XFILE RET EthHdFce.asm -o
XFILE RET EthBDefs.asm -o
XFILE RET IORFlop.obj -o
XFILE RET IORDisk.obj -o
XFILE RET IOREther.obj -o
XFILE RET RAMBoot.lnk -o

CRLFTool /lf DsplDefs.asm
DEL DsplDefs.bak
CRLFTool /lf FlopFace.asm
DEL FlopFace.bak
CRLFTool /lf ROMBDefs.asm
DEL ROMBDefs.bak
CRLFTool /lf RAMBDefs.asm
DEL RAMBDefs.bak
CRLFTool /lf FlpBDefs.asm
DEL FlpBDefs.bak
CRLFTool /lf RAMFlpBt.asm
DEL RAMFlpBt.bak
CRLFTool /lf DskBDefs.asm
DEL DskBDefs.bak
CRLFTool /lf DskHdFce.def
DEL DskHdFce.bak
CRLFTool /lf DskIOFce.def
DEL DskIOFce.bak
CRLFTool /lf RAMDskBt.asm
DEL RAMDskBt.bak
CRLFTool /lf RAMEthBt.asm
DEL RAMEthBt.bak
CRLFTool /lf EthIOFce.asm
DEL EthIOFce.bak
CRLFTool /lf EthHdFce.asm
DEL EthHdFce.bak
CRLFTool /lf EthBDefs.asm
DEL EthBDefs.bak

RUN ASM86 RAMFlpBt.asm DEBUG > RAMFlpBt.log
RUN ASM86 RAMDskBt.asm DEBUG > RAMDskBt.log
RUN ASM86 RAMEthBt.asm DEBUG > RAMEthBt.log
TYPE RAMFlpBt.log
TYPE RAMDskBt.log
TYPE RAMEthBt.log
PAUSE

RUN LINK86 RAMFlpBt.obj, IORFlop.obj, RAMBoot.lnk to FlpyBoot.lnk
RUN LINK86 RAMDskBt.obj, IORDisk.obj, RAMBoot.lnk to DiskBoot.lnk
RUN LINK86 RAMEthBt.obj, IOREther.obj,RAMBoot.lnk to EthrBoot.lnk
REM This will cause an unresoved external warning on TransmitFrame.
REM Ignore it.

RUN LOC86 FlpyBoot.lnk to FlpyBoot.loc PC(PURGE) ADDRESSES (SEGMENTS(IOPEInRAM(08F0H)))
RUN LOC86 DiskBoot.lnk to DiskBoot.loc PC(PURGE) ADDRESSES (SEGMENTS(IOPEInRAM(0A00H)))
RUN LOC86 EthrBoot.lnk to EthrBoot.loc PC(PURGE) ADDRESSES (SEGMENTS (IOPEInRAM(0980H)))

XFILE STORE FlpyBoot.loc -o
XFILE STORE DiskBoot.loc -o
XFILE STORE EthrBoot.loc -o
XFILE STORE FlpyBoot.lnk -o
XFILE STORE DiskBoot.lnk -o
XFILE STORE EthrBoot.lnk -o
XFILE STORE FlpyBoot.mp2 -o
XFILE STORE DiskBoot.mp2 -o
XFILE STORE EthrBoot.mp2 -o
XFILE STORE RAMFlpBt.obj -o
XFILE STORE RAMDskBt.obj -o
XFILE STORE RAMEthBt.obj -o

REM
REM  ****** Then do the following in XDE ******
REM
REM  MakeInitialMicroBoot FloppyInitialDove.db ← FlpyBoot.loc
REM  MakeInitialMicroBoot DiskInitialDove.db ← DiskBoot.loc
REM  MakeInitialMicroBoot EtherInitialDove.db ← EthrBoot.loc
REM
```

```
REM
REM fini
```

```
REM RAMBoot.bat
REM Copyright (C) 1987 by Xerox Corporation. All rights reserved.
BREAK ON

REM ----------------------
REM Build RAMBoot.lnk!
REM ----------------------

XFILE RET STKBoot.obj -o
XFILE RET IORMaint.obj -o
XFILE RET IOPLRAM.obj -o
XFILE RET RAMBDefs.asm -o
XFILE RET RamBoot.asm -o
XFILE RET EthHdFce.asm -o
XFILE RET HardDefs.asm -o
XFILE RET IOPDefs.asm -o
XFILE RET IOPMacro.asm -o
XFILE RET EthBDefs.asm -o
XFILE RET Handlers.asm -o
XFILE RET IORRAMBt.asm -o
XFILE RET CSBankDf.asm -o
XFILE RET RAMEEP.asm -o

CRLFTool /lf RAMBDefs.asm
DEL RAMBDefs.bak
CRLFTool /lf RamBoot.asm
DEL RamBoot.bak
CRLFTool /lf EthHdFce.asm
DEL EthHdFce.bak
CRLFTool /lf HardDefs.asm
DEL HardDefs.bak
CRLFTool /lf IOPDefs.asm
DEL IOPDefs.bak
CRLFTool /lf IOPMacro.asm
DEL IOPMacro.bak
CRLFTool /lf EthBDefs.asm
DEL EthBDefs.bak
CRLFTool /lf Handlers.asm
DEL Handlers.bak
CRLFTool /lf IORRAMBt.asm
DEL IORRAMBt.bak
CRLFTool /lf CSBankDf.asm
DEL CSBankDf.bak
CRLFTool /lf RAMEEP.asm
DEL RAMEEP.bak

RUN ASM86 RAMBoot.asm DEBUG > RAMBoot.log
RUN ASM86 IORRAMBt.asm DEBUG > IORRAMBt.log
TYPE RAMBoot.log
TYPE IORRAMBt.log
PAUSE

RUN LINK86 RAMBoot.obj,IOPLRAM.obj,IORRAMBt.obj,STKBoot.obj,IORMaint.obj TO RAMBoot.lnk

XFILE STORE RAMBoot.lnk -o
XFILE STORE RAMBoot.mp1 -o
XFILE STORE RAMBoot.obj -o
XFILE STORE IORRAMBt.obj -o

REM ----------------------
REM Build FlpyBoot/DiskBoot/EthrBoot.loc!
REM ----------------------

REM If you don't want to run Initial.bat, type Control C
PAUSE

XFILE RET Initial.bat -o

CRLFTool /lf Initial.bat
DEL Initial.bak

Initial.bat

REM Initial.bat depends on components rebuilt in here!
REM fini
```

```
;Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.
;CSBankDf.asm
;For coordinating between Ramboot.asm and Csbank.asm.
;Last Edited By:
;       RDH      13-Jan-86 15:17:57      :Create.

bankConfigMask  EQU     30H     ;For the convenience of *InitialDove that it
                                ; may be as small as possible.
```

```
;Copyright (C) 1985, 1987 by Xerox Corporation. All rights reserved.

;-- Definitions for the disk boot handler.
;-- stored as [Iris]<WMicro>Dove>DskBDefs.asm
;
;-- last edited by:


;--     JPM     18-May-87  8:42:53       ;Changed value of diskShapeHeadAndSector (from 15).
;--     JPM      4-May-87 16:18:25       ;Added DiskShape STRUC & constants.
;--     JPM     16-Jul-85 12:50:41       ;bootDataBegins moved to RAMBDefs.
;--     JMM     26-Jun-85 16:36:59       ;Removed imbedded defs.
;--     JPM     17-Jun-85 10:33:30       ;Created from FlpBDefs.asm.

;------------------------------------------------------------------------------
;

;-------------------------------------------------------

;-- STRUC for file descriptor

FileSpec                STRUC

Cyl                     DW      ?       ; cylinder in entire word
HdSct                   DW      ?       ; head in high byte, sector number in low byte

FileSpec                ENDS

;-- STRUC for device-specific IORegion area (overlays deviceSpecificArea)

DiskBootArea            STRUC

diskIOCBDone            DB      SIZE(Condition) DUP (?)
;diskBootIOCB           DB      156 DUP (?)     ; first 70 bytes can be overlaid by workspace
fileCount               DB      ?
                        DB      ?
file1                   DB      SIZE(FileSpec) DUP (?)
file2                   DB      SIZE(FileSpec) DUP (?)
file3                   DB      SIZE(FileSpec) DUP (?)

DiskBootArea            ENDS

diskBootIOCB            EQU     fileCount

;-- STRUC for file id in root page

DiskRootFileID          STRUC

DFIDfileID              DB      10 DUP (?)      ; file ID
DFIDfirstPg             DB      4 DUP (?)       ; first page
DFIDcylinderHigh        DB      ?       ; cylinder high byte
DFIDcylinderLow         DB      ?       ; cylinder low byte
DFIDhead                DB      ?       ; head
DFIDsector              DB      ?       ; sector

DiskRootFileID          ENDS

;-- STRUC for disk shape

DiskShape               STRUC

DSseal                  DW      ?       ; must equal diskShapeSeal
DSversion               DW      ?       ; must equal diskShapeVersion
DStype                  DW      ?
                        DB      ?
DSsectorsPerTrack       DB      ?
                        DB      ?
DSheadsPerCylinder      DB      ?
DScylinderCount         DW      ?
DSreducedWriteCyl       DW      ?
DSpreCompCyl            DW      ?
                        DW      246 DUP (?)
DSchecksum              DW      ?
DSinvertedChecksum      DW      ?

DiskShape               ENDS

;-- constants

sectorSize              EQU     onePage ; sector size in bytes
rootPageCylinder        EQU     0       ; location of root page
rootPageHeadAndSector   EQU     0       ;  is cyl. 0, head 0, sector 0
rootPageHeaderSize      EQU     16      ; 16 bytes of header before 1st file ID
diskShapeCylinder       EQU     0       ; location of disk shape
diskShapeHeadAndSector  EQU     14      ;  is cyl. 0, head 0, sector 14
diskShapeSeal           EQU     9665H   ; byteswapped 6596H = 62626 octal
diskShapeVersion        EQU     100H    ; byteswapped 1

;--                     END     DskBDefs
```

```
; File: DsplDefs.asm - last edit:
; MXT                   6-Mar-86 10:57:11

; Copyright (C) 1986 by Xerox Corporation. All rights reserved.

;-- stored as [Idun]<WDLion>Dove>DsplDefs.asm
;
;Copyright (C) 1984 by Xerox Corporation. All rights reserved.
;
;Last edited by:
;;--     MXT      2-May-86 10:30:53       :Changed borderOnlyMask and videoDisableMask
;--      MXT     10-Feb-86 19:28:06       :Added Daisy specific I/O Addresses
;--      JPM/kek  1-Sep-85 14:23:55       :Added *chngd (flags for chngdInfo word) (taken from B0 edits)
;--      ANK     10-Jan-85 12:25:45       :Changed to use the 2nd etch IO port addresses
;--      ANK      3-Jan-85 15:47:18       :Added Software Reset port
;--      ANK     14-Nov-84  9:00:56       :Deleted DisplMemStrtLow and High
;--      ANK     26-Sep-84 12:53:16       :Deleted MapRegistersDayBreak
;--      ANK     13-Sep-84 10:13:43       :Added MapRegistersDayBreak
;
;-----------------------------------------------------------------------------------------------------
;
;Constants used for display handler/ Maintainence panel handler synchronization
;
MPCursor                EQU     0FFH    ;CursorUser=FF when MP handler is setting the cursor pattern
DisplayCursor           EQU     0       ;CursorUser=0 when display handler is setting the cursor pattern
;
;-----------------------------------------------------------------------------------------------------
;
;Set up constants for I/O addresses
;
;DAYBREAK/DAISY ADDRESSES
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
;
;DAYBREAK ADDRESSES
;
VertStoreDayBreak               EQU   0E800H;   Starting locn of vertical control store
HorzStoreDayBreak               EQU   0EC00H;   Starting locn of horizontal control store
;
DisplCntlRegDayBreak            EQU   0EC80H;   Data display control register
BorderPatternLowDayBreak        EQU   0EC81H;   Border pattern low byte
BorderPatternHighDayBreak       EQU   0EC82H;   Border pattern high byte
;
CursorWordPortDayBreak          EQU   0EC83H;   Word number of cursor position
CursorOffsetPortDayBreak        EQU   0EC84H;   Word offset of cursor position
CursorLinePortLowDayBreak       EQU   0EC85H;   Line number of cursor position (Low byte)
CursorLinePortHighDayBreak      EQU   0EC86H;   Line number of cursor position (High byte)
;
DMCWordsPerLineDayBreak         EQU   0EC88H;   DMC register that contains number of words per line
;
DMCDisplStrtAddrLowDayBreak     EQU   0EC89H;   DMC register that holds low byte of display mem. strt addr.
DMCDisplStrtAddrHighDayBreak    EQU   0EC8AH;   DMC reg. that holds high byte of display mem. strt addr.
;
DisplMemStatReg1stDayBreak      EQU   0EC8CH
SysMemStatReg2ndDayBreak        EQU   0EC8DH
SysMemStatReg3rdDayBreak        EQU   0EC8EH
SysMemStatReg4thDayBreak        EQU   0EC8FH
;
DisplaySoftReset                EQU   0ED60H;   A read from this port resets display HW
CursorBufferPortsDayBreak       EQU   0ED00H;   Cursor buffer starting locn.
;
;
; - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
;
;DAISY ADDRESSES -- All these I/O addresses are TBD
;
;Daisy A chip registers are accessed via IN/OUT instructions and their addresses are calcurate from following equation.
;       0800 + 0100 * AID + Location
;               here AID is the A chip ID: CARDINAL [0..4)
;following definitions are Location.
;in actual use, the AID should be ORed into upper byte.
;AID0 = 0800H
;AID1 = 0900H
;AID2 = 0A00H
;AID3 = 0B00H

AChip0                  EQU     08H;
CursorBufferReg         EQU     880H;   Cursor buffer starting location.

HCursor                 EQU     8A0H;   Cursor x-coodinate in words.
                                ;       b15,,b14 = Cursor Nibble Offset. b13,,b12 = Cursor Rotate parameter
HBorder1                EQU     8A4H;   Starting point of display bitmap.
                                ;       b15 = DCReset, b14 = ColorMode', b13 = DCTest', b12 = PICyBDR'
HBorder0                EQU     8A6H;   End point of display bitmap.
                                ;       b15,,14 = HSyncOff1-0, b13 = Interlace, b12 = BaseP10
HBlank0                 EQU     8A8H;   Start left border location. (b15,,b12 = unused).
HBlank1                 EQU     8AAH;   Stop right border and start blank location.
HSync1                  EQU     8ACH;   Start Horiz Sync.
HSync0                  EQU     8AEH;   Stop Horiz Sync.

VCursor                 EQU     8B0H;   Cursor y-coodinate in words.
                                ;       b15,,b12 = Mix Rule
VInt                    EQU     8B2H;   Vertical interrupt point.
VBorder1                EQU     8B4H;   Start lower border location.
VBorder0                EQU     8B6H;   Start display bitmap.
VBlank1                 EQU     8B8H;   End of lower border location.
VBlank0                 EQU     8BAH;   Start upper border location.
VSync0                  EQU     8BCH;   End of VSync

DsplDefs.asm          2-May-86 10:30:55 PDT
```

```
VSync1                  EQU     8BEH;   Start of VSync

BaseP                   EQU     8C0H;   page address of start of bitmap.
QRandLR                 EQU     8C2H;   Line count (b15,,b7) and num of quad words (b5,,b0)

Border1                 EQU     8C4H;   border pattern for even line.
Border0                 EQU     8C6H;   border pattern for odd line.


;-------------------------------------------------------------------------------------------------
;
;Constants used for  initialization
;
NumHorzBytes            EQU     0AH;    Number of values in horizontal control store
NumVertBytes            EQU     09H;    Number of values in vertical control store
;
DefaultMixRule          EQU     0E0H;   Default value for Cursor/Data mix (Daisy)
DCResetBit                      EQU     8000H;
;
;-------------------------------------------------------------------------------------------------
;
;Misc
;
NumBitMapBytes                  EQU     32;     Number of bytes in cursor pattern
NumBitMapBytesDaisy             EQU     16; Number of words in cursor pattern.
;
; Flags in chngdInfo word
cursorPositionChngd             EQU     8000H   ;b15 = change in cursor position
cursorPatternChngd              EQU     4000H   ;b14 = change in cursor pattern
borderPatternChngd              EQU     2000H   ;b13 = change in border pattern
backgroundChngd                 EQU     1000H   ;b12 = change in backgrond
commandChngd                    EQU     0800H   ;b11 = change in command to display control register
;
;constatns to determine displCntl contents for Daisy.
pictureBorderCheckMask  EQU     08H
borderOnlyMask          EQU     10H
videoDisableCheckMask   EQU     02H
videoDisableMask        EQU     00H
MSNibbleMask            EQU     0FFFH
;-------------------------------------------------------------------------------------------------
;
```

```
$MOD186
$PAGELENGTH (72)
$PAGEWIDTH  (136)

;Copyright (C) 1987 by Xerox Corporation. All rights reserved.
;-- DybrkCP.asm
;-- last edited by:

;--  RDH  22-Jan-86 12:06:22 ;Create from parts of RAMBoot.asm.


NAME                    DybrkCP
;-------------------------------------------------------------------------------
;
$NOLIST
;$INCLUDE               (EthHdFce.asm)
$INCLUDE                (HardDefs.asm)
$INCLUDE                (IOPDefs.asm)
$INCLUDE                (IOPMacro.asm)
$INCLUDE                (RAMBDefs.asm)
;$INCLUDE               (EthBDefs.asm)
;$INCLUDE               (Handlers.asm)   ;to resolve handler IDs
;$INCLUDE               (CSBankDf.asm)
;$INCLUDE               (RAMEEP.asm)
$LIST

;-------------------------------------------------------------------------------

EXTRN                   mesaProcessorInterrupt :ABS

;-------------------------------------------------------------------------------
;from IORRamBt.asm:
IOPELocalRAM            SEGMENT AT 0

EXTRN                   resetRegData: WORD

IOPELocalRAM            ENDS

;-------------------------------------------------------------------------------
;from IORRamBt.asm:
BootStrapIOR            SEGMENT COMMON

EXTRN                   IncSIFarProc: WORD

BootStrapIOR            ENDS

;-------------------------------------------------------------------------------
IOPEInRAM               SEGMENT         PUBLIC
                        ASSUME  CS:IOPEInRAM
                        ASSUME  DS:BootStrapIOR
;-------------------------------------------------------------------------------

PUBLIC                  DybrkCPConditioning
PUBLIC                  WriteDybrkControlStore
PUBLIC                  InitDybrkCP

EXTRN   LoadAXFromBootBuf: NEAR
EXTRN   LoadAXFromBootBufLateEntry: NEAR
EXTRN   LoadCXFromBootBuf: NEAR
EXTRN   LoadCXFromBootBufLateEntry: NEAR
EXTRN   CallDumpCSAddrBlock: NEAR
;-------------------------------------------------------------------------------

;-------------------------------------------------------------------------------
DybrkCPCond     PROC    FAR

DybrkCPConditioning:
                MOV     AX, IOPELocalRAM
                MOV     ES, AX
                ASSUME  ES: IOPELocalRAM
                CALL    DaybreakCPHalt
                CLI                     ;to protect reset register
                MOV     AX, resetRegData
                AND     AX, NOT resetMesaProcessor
                OUT     WriteResetReg, AX
                CALL    DaybreakCPStart
                OR      resetRegData, resetMesaProcessor
                MOV     AX, resetRegData
                OUT     WriteResetReg, AX
                CALL    DaybreakCPHalt
                STI                     ;done with reset register
                RET
                ASSUME  ES:NOTHING

DybrkCPCond     ENDP


;-------------------------------------------------------------------------------


WriteDybrkControlStore  PROC    FAR

WriteCntlStore:         CALL    LoadAXFromBootBufLateEntry
                        ;We first want to set the CS
                        ;bank register then get the CS
                        ;address that this block is to

DybrkCP.asm         28-Jan-87 15:00.56 PST
```

```
                        ;be written to.

                        ;DayBreak's control store is an I/O port.
                        ;Put the cs address into DX.
                        CALL    LoadCXFromBootBuf
                        MOV     DX, CX
                        ;We want to point to the count
                        ;of CS words in this block,
                        ;from which we will fetch the
                        ;CS block size counter and
                        ;save it in DI - we are again out of registers!
                        CALL    LoadCXFromBootBuf
                        MOV     DI, CX                  ;
                        MOV     BP, DX

                        MOV     DX, daybreakBankRegister ;First let us set up the CS bank
                        OUT     DX, AL                  ;register.

WriteDaybreakCSBlock:   MOV     DX, BP                  ;Also need to save initial CS adrs.
                        OR      DX, daybreakCSPortMask  ;
                        MOV     CX, CSWordByteSize      ;A single control store word
WriteDaybreakCSWord:    CALL DWORD PTR  IncSIFarProc            ;is made up of six bytes. The
                        MOV     AL, ES: [BX][SI]        ;layout of the control store with
                        OUT     DX, AL                  ;respect to the 4K ports is as
                        ADD     DX, nextCSByte          ;follows: 8###H - D###H where
                        LOOP    WriteDaybreakCSWord     ;8###H has the MSB and D###H has
                        INC     BP                      ;the LSB. The CS is loaded in six
                        DEC     DI                      ;byte data streams. Update CS word count.
                        JNZ     WriteDaybreakCSBlock    ;Loop while not done with CS block.
                        CALL DWORD PTR  IncSIFarProc

                        RET


WriteDybrkControlStore  ENDP

;-------------------------------------------------------------------------------

DybrkInitializeCP       PROC    FAR

InitDybrkCP:    PUSHA
                %Enable(mesaProcessorInterrupt)  ;turn on the interrupt, but
                %DisableInterruptsTillNextWait   ; don't allow it till we're ready
                CALL    DaybreakCPStart          ;let CP run
                %WaitForInterrupt       (1)      ;init routines should take < 1 sec
                IN      AX, ClrMesaIntr          ;clear Mesa interrupt
                %Disable(mesaProcessorInterrupt); and turn it off
                %WaitForSystem                   ;drop to system level
                POPA
                CALL    DaybreakCPHalt           ;stop the CP (so can load more code)
InitializeDaybreakCPRet:
                JMP     CallDumpCSAddrBlock

DybrkInitializeCP               ENDP

;-------------------------------------------------------------------------------

waitForMesaInterrupt    PROC    NEAR

                                                 ;Restore registers, then
WaitForMesaInt:         ;CALL   DaisyCPHalt      ;halt the CP and keep
                        RET                      ;processing the boot file.


waitForMesaInterrupt    ENDP
;-------------------------------------------------------------------------------

BadMesaInterrupt        PROC    NEAR

BadMesaInt:             RET

BadMesaInterrupt        ENDP


;-------------------------------------------------------------------------------

DaybreakCPHalt          PROC    NEAR

DybrkCPHalt:            MOV     AX, 0
                        OUT     WriteCSReg, AX
                        SHL     AX, 15  ; delay
                        SHL     AX, 13  ;  at least 38 cycles
                        RET

DaybreakCPHalt          ENDP

;-------------------------------------------------------------------------------

DaybreakCPStart         PROC    NEAR

DybrkCPStart:           MOV     AX, 0200H
                        OUT     WriteCSReg, AX
                        RET

DaybreakCPStart         ENDP


DybrkCP.asm         28-Jan-87 15:00:56 PST
```

```
;-------------------------------------------------------------------------
IOPEInRAM               ENDS

                        END
```

```
;Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.

;-- stored as [Iris]<WMicro>Dove>Handlers.asm
;-- created on  23-Jul-84 15:29:07

;--     JAC             28-Aug-85 15:17:17      :add bogus handler for and Opie stack
;--     kek             28-Aug-85 15:17:17      :add parity ROM handler
;--     JPM    .es      14-May-85 10:00:59      :Customize for next integration
;--     VXS  .pa         7-Dec-84 17:10:38      :Customize for first integration
;--     VXS  .pa         6-Nov-84 14:36:15      :Creation

;This file enumerates all handlers which will be linked with
; HandInit.asm, and is INCLUDEd in that module.

%' Handler writers: to customize this file, comment out the %Handler calls
%'  below and/or add %Handler calls with your handler name(s) and ID(s).

%*DEFINE(HandlersLinked)(

%Handler(Beep,1,PROC)
%Handler(Disk,2,PROC)
%Handler(Display,3,PROC)
%Handler(Ethernet,4,PROC)
%Handler(Floppy,5,PROC)
%Handler(KeyBoardAndMouse,6,PROC)
%Handler(MaintPanel,7,PROC)
%Handler(BootStrap,8,CALL)
%'Handler(Parity,9,PROC)
%'Handler(Umbilical,10,CALL)
%'Handler(RemoteMemory,11,CALL)
%Handler(System,12,)

)
```

```
; DAISOGO02.def= 5 revs up from DaisyRevC.Def


;Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.
;-- stored as [Iris]<WMicro>Dove>HardDefs.asm

%*DEFINE(Revision)(Hardware: G, DefsVersion: 002)
;end Revision macro definition

;Last Edited By:
;       TXM      30-Oct-87  9:03:14     :Add Dahlia distinction
;       KEK      17-Apr-86  9:21:29     :add Daisy defs
;       JAC      24-Jan-86 10:49:04     :change 8259OptionsSlave definitions
;       JGS      27-Sep-85 12:25:47     :DisplayTypePort is OECCCH not OECCOH.
;       JPM      11-Sep-85  8:57:09     :Add i8274NonVectored and appropriate constant changes.
;       JPM       2-Aug-85 16:32:59     :Fix EEProm segment offsets.
;       KEK      19-Jun-85 13:29:31     :remove fixed EEPDefs definitions (put into ROMEEP.asm, RAMEEP.asm, and BadPage.asm), added eep
segment offsets. Remove duplicate defs 8259Mas, 8259Slv, also Ctl and IntCtl...
;       JPM       6-Jul-85 10:36:35     :Add i80186Flags (from HardOpie).
;       JPM       2-Jul-85  8:09:55     :Add i8274VarVect.
;       JPM      25-Jun-85 16:56:49     :Take out software-determined EEProm constants (except byte/word flag for IOPKernl).
;       KEK      19-Jun-85 13:29:31     :add in EEProm section, containing old EEPDefs.asm and HardEEP.asm.
;       JPM      30-May-85 16:53:35     :change i8259MasterICW3 for B0.
;       KEK      21-May-85 11:04:37     :add DisplayTypePort and DisplayTypeMask.
;       KEK      25-Apr-85 16:10:33     :more 8274 defs (used by IOPInit). Corrected 18259OptionsSlaveICW2 vector number. add some
i8259OptionsSlave defs.
;       KEK       5-Mar-85 11:18:16     :add ETCH TWO comment (WriteCtlReg defs).
;       KEK       26-Feb-85 21:00:32    :Add machine distinguishing constants, retraceLatch port constant, add 8274 equates.
;       VXS     15-Nov-84 12:06:42      :Add symbols for Control Register.
;       VXS      7-Nov-84 13:33:59      :Remove ClearResetsMask
;       VXS      5-Nov-84 15:44:53      :Add defs for daybreak map register numbers
;       VXS     11-Oct-84 10:07:23      :Add Device Reset information
;       VXS      9-Oct-84 11:52:19      :Changed FDCDMADataReg to base+4 - 6 works too, but it decodes the bits seperately anyway.
;       VXS      4-Oct-84 14:41:15      :Add daybreakMapIOAddressBase
;       VXS      3-Oct-84 16:59:26      :Added FloppyTimer, since the timer is hardwired to timer 1.
;       VXS      1-Oct-84 18:46:27      :Added definitions for Floppy DMA (uses internal 186 DMA controller)
;       VXS     17-Sep-84 14:05:10      :Change def of TRUE to -1 instead of OFFFF so that can use it for bytes
;       FXB     15-Aug-84 16:29:29      :Made changes for Rev 0G002 for AKTsang
;       VXS      6-Aug-84 18:30:34      :Add OptionsSlaveInServiceRegAddr
;       VXS      6-Aug-84 12:53:55      :Use new symbols instead of *CtlrBase ones
;       VXS      3-Aug-84 18:54:20      :Added def for 8259 masterInServceRegAddr
;       VXS      3-Aug-84 17:52:15      :Added Mas and Slv defs back in as duplicates for compatibility, also Ctl and IntCtl
;       VXS      3-Aug-84 14:08:21      ;changed i186*CtlAddr to i186*IntCtlAddr for clarity
;       VXS      3-Aug-84 14:05:39      :Added i186Intrchannelfor*
;       VXS     3-Aug-84 11:49:18       :Changed convention for 8259 expansion from *Exp* to *Optionsslave*
;       VXS     3-Aug-84 11:31:41       :Put in Revision macro, which is documentation of what version of this file and what hardware
revision its for
;       VXS     3-Aug-84 10:31:18       :Removed defs specific to chips and moved them to *Defs.asm
;       VXS     3-Aug-84 10:21:15       :changes *Mas symbols to *Master, *Slv to *Slave
;       FXB     1-Aug-84 16:23:51        :added expansion intr cntlr constants
;       JBinkley 22-Jun-84 16:20:40
;       P. PxE  for JBinkley 6-Jun-84 13:49:52
;                                       :Changed ICW3 to reflect slave mode of 8274
;                                         changed ISR & IRR for 8259 read.
;       JBinkley 20-Apr-84 13:14:22

; updated to RevC Build by JBinkley 12-Apr-84  6:39:12
; first written by JBinkley 21-Oct-83 with Geoff Thompson

; This file defines IO addresses for the Daisy IOP.
; It also includes operation constants for hardware that is central to the system,
; such as the 8259 interrupt controller, the 80186 processor operation constants, etc.
; There are now seperate files for the peripheral chip operating constants, of
; the form <chipname>Defs.asm.

; It should be INCLUDE-ed  in all
;   hardware dependent code modules.

;*******************************************************
; DO NOT REFER TO "iAPX 86/88, 186/188 User's Manual
;                  Programmer's Reference",   May 1983.
; IT CONTAINS NUMEROUS MISTAKES.
; Refer, instead, to the 186 Application's Note
;     by Ken Shoemaker, March, 1983
;     or to the 186 Data Sheet.
;*******************************************************

; Conventions for labels:
;       Intn    =       Internal
;       Intr    =       Interrupt
;       Ctl     =       Control
;       Ctlr    =       Controller
; Chip identifiers --
;       i186    =       Intel 80186
;       i8259=  Intel 8259
;       i8251=  Intel 8251
; Normally, each new word in a label occurs with the first character
; capitalized. The exception is when the previous word is all CAPS,
; as in: PACSvalue. (When a name is defined by Intel, we use their
; convention.)


;*******************************************************


; i80186 Internal Peripheral Control Block  (PCB)


HardDefs.asm        29-Oct-87 16:03:15 PST
```

```
; This section defines the various control register addresses in the
;   80186  on-chip Peripheral Control Block  (PCB)

; After RESET, the relocation reg = 20FFH.  This means the
;   PCB will be mapped into I/O space, &
;   base address of  PCB = 0FF00H.

; PCB can be relocated to any 256-byte boundary.
; For Daisy, the PCB is specified at the upper-most 256 bytes of
;   the I/O space  (i.e., default i186 locations)

PCBbase EQU     0FF00h



; Chip  Select  Control  Register  Locations
;
UMCSaddr        EQU     PCBbase + 0A0h  ;Upper Mem (ROM) Chip Select
LMCSaddr        EQU     PCBbase + 0A2h  ;Lower Mem (RAM) Chip Select
PACSaddr        EQU     PCBbase + 0A4h  ;Peripherial Chip Select
                                        ; & PCS 0-3 Ready bits
MMCSaddr        EQU     PCBbase + 0A6h  ;Middle Mem (RAM) Chip Select
MPCSaddr        EQU     PCBbase + 0A8h  ;Middle Mem Range
                                        ; & PCS 4-6 Ready bits

;-------------------------------------------------------

; Chip  Select  Control  Register  Values

; (See i186ControlBlockProgramming.doc for more detailed documentation)
; See Intel  uP & P Handbook, 1983,  pp.3-39  to 3-43.
; All memory sizes are specified in bytes.



; Upper Memory Chip Select
;   16 KBytes of EPROM, base @ 0FC000h, 0 WS, RDY ignored
UMCSvalue       EQU     0FC3Ch  ; ← (0038h -OR- 0004h -OR- FC00h)



; Lower Memory Chip Select
;   16 KBytes of SRAM, base @ 000000h, 0 WS, wait for RDY
LMCSvalue       EQU     003F8h  ; ← (0038h -OR- 0000h -OR- 03C0h)



; Middle Memory Chip Select
;   Possible 64 KBytes of SRAM, as 4  16K chunks
;   Only first 16 KBytes chunk is implemented,
;   base @ 010000h, 0 WS, wait for RDY
MMCSvalue       EQU     011F8h  ; ← (01F8h -OR- 0000h -OR- 1000h)



; Middle Memory / Peripheral Chip Se ect
;   Possible 64 KBytes of SRAM, as 4  16K chunks
;   (addresses 010000 to 013FFF installed)
;   7 PCS' lines, mapped into I/O space
;   PCS4-6: 0 WS, RDY ignored
MPCSvalue       EQU     088BCh  ; ← (8038h -OR- 0800h -OR- 0080h -OR- 0004h)



; Peripheral Chip Select
;   Base @ 0000H,
;   PCS0-3: 1 WS, RDY ignored
PACSvalue       EQU     0003Dh  ; ← (0038h -OR- 0005h -OR- 0000h)


;-------------------------------------------------------------------------

;  Other 80186 Internal Control registers addresses
i186RelocationRegAddr   EQU     PCBbase+0FEh    ;Relocation Register
i186Timer0IntCtlAddr    EQU     PCBbase+032h    ;Timer 0 Interrupt Control Register
i186Timer1IntCtlAddr    EQU     PCBbase+038h    ;Timer 1 Interrupt Control Register
i186Timer2IntCtlAddr    EQU     PCBbase+03Ah    ;Timer 2 Interrupt Control Register

;Definitions for i186 internal DMA 0

i186DMA0IntCtlAddr      EQU     PCBbase+034h    ;DMA 0 Interrupt Control Register
i186DMA0LowSourcePtr    EQU     PCBbase+0C0h    ;DMA 0 Low order source pointer
i186DMA0HighSourcePtr   EQU     PCBbase+0C2h    ;DMA 0 High order source pointer
i186DMA0LowDestPtr      EQU     PCBbase+0C4h    ;DMA 0 Low order destination pointer
i186DMA0HighDestPtr     EQU     PCBbase+0C6h    ;DMA 0 High order destination pointer
i186DMA0TransferCount   EQU     PCBbase+0C8h    ;DMA 0 transfer count
i186DMA0ControlWord     EQU     PCBbase+0CAh    ;DMA 0 control word

;Definitions for i186 internal DMA 0

i186DMA1IntCtlAddr      EQU     PCBbase+036h    ;DMA 1 Interrupt Control Register
i186DMA1LowSourcePtr    EQU     PCBbase+0D0h    ;DMA 1 Low order source pointer
i186DMA1HighSourcePtr   EQU     PCBbase+0D2h    ;DMA 1 High order source pointer
i186DMA1LowDestPtr      EQU     PCBbase+0D4h    ;DMA 1 Low order destination pointer

HardDefs.asm        29-Oct-87 16:03:15 PST
```

```
i186DMA1HighDestPtr     EQU     PCBbase+0D6h    ;DMA 1 High order destination pointer
i186DMA1TransferCount   EQU     PCBbase+0D8h    ;DMA 1 transfer count
i186DMA1ControlWord     EQU     PCBbase+0DAh    ;DMA 1 control word

i186IntVectorRegAddr    EQU     PCBbase+020h    ;Interrupt Vector Register
i186EOIRegAddr          EQU     PCBbase+022h    ;Specific EOI Register
i186IntrMaskRegAddr     EQU     PCBbase+028h    ;Interrupt mask register
i186PriorityMaskAddr    EQU     PCBbase+02Ah    ;Priority Level Register
i186InServiceRegAddr    EQU     PCBbase+02Ch    ;In Service register
i186IntrRequestAddr     EQU     PCBbase+02Eh    ;Interrupt Request Register
i186IntrStatusAddr      EQU     PCBbase+030h    ;Interrupt Status Register

;-------------------------------------------------------------------------
; 80186 Timer I/O Addresses

i186Timer0MCWAddr       EQU     PCBbase+056h    ;Mode/Control Word
i186Timer1MCWAddr       EQU     PCBbase+05Eh    ;Mode/Control Word
i186Timer2MCWAddr       EQU     PCBbase+066h    ;Mode/Control Word

i186Timer0CountBAddr    EQU     PCBbase+054h    ;Max Count B
i186Timer1CountBAddr    EQU     PCBbase+05Ch    ;Max Count B

i186Timer0CountAAddr    EQU     PCBbase+052h    ;Max Count A
i186Timer1CountAAddr    EQU     PCBbase+05Ah    ;Max Count A
i186Timer2CountAAddr    EQU     PCBbase+062h    ;Max Count A

i186Timer0CountRegAddr  EQU     PCBbase+050h    ;Count Reg
i186Timer1CountRegAddr  EQU     PCBbase+058h    ;Count Reg
i186Timer2CountRegAddr  EQU     PCBbase+060h    ;Count Reg



;-------------------------------------------------------------------------

; 80186 Internal Control registers values

i186RelocationRegvalue  EQU     060FFh  ;RMX mode & default location
i186IntrVectorRegvalue  EQU     0038h   ;Vectors types start at 038h

;The following channel requirements are imposed by the 186 hardware in iRMX mode.

i186IntrChannelforTimer0        EQU     0
i186IntrChannelforDMA0          EQU     2
i186IntrChannelforDMA1          EQU     3
i186IntrChannelforTimer1        EQU     4
i186IntrChannelforTimer2        EQU     5

i186IntrMaskforTimer0   EQU     NOT(1 SHL i186IntrChannelforTimer0)
i186IntrMaskforDMA0     EQU     NOT(1 SHL i186IntrChannelforDMA0)
i186IntrMaskforDMA1     EQU     NOT(1 SHL i186IntrChannelforDMA1)
i186IntrMaskforTimer1   EQU     NOT(1 SHL i186IntrChannelforTimer1)
i186IntrMaskforTimer2   EQU     NOT(1 SHL i186IntrChannelforTimer2)


; EOI Commands for internal interrupt controller

i186EOItimer0           EQU     i186IntrChannelforTimer0
i186EOIdma0             EQU     i186IntrChannelforDMA0
i186EOIdma1             EQU     i186IntrChannelforDMA1
i186EOItimer1           EQU     i186IntrChannelforTimer1
i186EOItimer2           EQU     i186IntrChannelforTimer2


;-------------------------------------------------------------------------

;80186 Flag structure

i80186Flags     RECORD  reserved15to12:4, i80186OF:1, i80186DF:1, i80186IF:1,
&                       i80186TF:1, i80186SF:1, i80186ZF:1, reserved5:1,
&                       i80186AF:1, reserved3:1, i80186PF:1, reserved1:1,
&                       i80186CF:1
                PURGE   reserved15to12, reserved5, reserved3, reserved1

;-------------------------------------------------------------------------

; Peripherial  Device  Base  Addresses
;   Peripherial Chip Selects are mapped into I/O space
;   and start at address 0000H.
;  Note : Base address of PCS's must be an integer multiple of 1K.

PCSBase EQU     0h              ; defined in PACS
PCS0Base        EQU     PCSBase + 0h
PCS1Base        EQU     PCSBase + 080h
PCS2Base        EQU     PCSBase + 100h
PCS3Base        EQU     PCSBase + 180h
PCS4Base        EQU     PCSBase + 200h
PCS5Base        EQU     PCSBase + 280h
PCS6Base        EQU     PCSBase + 300h

; ------------------------------------------------

; PCS.0'  --  used for Peripherial Controllers,  1 w.s.
;   (8 bit Data Bus Devices)


; --- new names as of August 3, 1984 ---
```

```
i8259MasterBase EQU     PCS0Base + 00h  ;A6-A4 = 0
i8259SlaveBase  EQU     PCS0Base + 10h  ;A6-A4 = 1
i8254Base       EQU     PCS0Base + 20h  ;A6-A4 = 2
i8251Base       EQU     PCS0Base + 30h  ;A6-A4 = 3
i8274DCommBase  EQU     PCS0Base + 40h  ;A6-A4 = 4
i8272Base       EQU     PCS0Base + 50h  ;A6-A4 = 5
i8259OptionsSlaveBase   EQU     PCS0Base + 60h  ;A6-A4 = 6
i8255Base               EQU     PCS0Base + 70h  ;A6-A4 = 7


; -------------------------------------------------

; PCS.1'  --  used for miscellaneous I/O,  0 w.s.
;   (16 bit Data Bus Devices)

DisplayTypePort EQU     0ECCCH          ;read this to get display size data
DisplayTypeMask EQU     01H             ;if bit0 = 0,then 19" display else 15".

ReadInputPort   EQU     PCS1Base + 0h           ;A6-A4 = 000b, R --80h
ReadHostProm    EQU     PCS1Base + 10h          ;A6-A4 = 001b, R --90h
ClrRingLatch    EQU     PCS1Base + 20h          ;A6-A4 = 010b, R --A0h
ClrMesaIntr     EQU     PCS1Base + 30h          ;A6-A4 = 011b, R --B0h
ClrENetIntr     EQU     PCS1Base + 40h          ;A6-A4 = 100b, R --C0h
ClrRetraceIntr  EQU     PCS1Base + 50h          ;A6-A4 = 101b, R --D0h

;a word IN instruction to this port will address the arbiter:

ArbCmdBase      EQU     PCS1Base + 70h          ;A6-A4 = 111b, R --Fxh

;The arbiter control will accept any combination of the bits below, and perform
; the multiple functions specified:

AllowPCCmdOffset        EQU     08H             ;add this to ArbCmdBase
AllowRDCmdOffset        EQU     04H             ;add this to ArbCmdBase
HoldIOPCmd              EQU     02H             ;add this to ArbCmdBase

WriteCtlReg     EQU     PCS1Base + 0h           ;A6-A4 = 000b, W --80h
;ETCH ONE DEFS!
  CRSpeakerData                 EQU     8000H
  CREnableTimer0                EQU     4000H
  CRFDDMotorOn                  EQU     2000H
  CRFDDInUse                    EQU     1000H
  CRTimer1GenerateFloppyTC      EQU     0800H
  CREEPromAccess                EQU     0400H
  CRRS232AInternalClock         EQU     0200H
  CRRS232BEnableClockSend       EQU     0100H
;ETCH TWO DEFS! (some are already defined by the ETCH ONE defs above)
  CRNotBlockSysMem              EQU     8000H   ;'1 enables memory!
; CREnableTimer0                EQU     4000H
; CRFDDMotorOn                  EQU     2000H
; CRFDDInUse                    EQU     1000H
; CRTimer1GenerateFloppyTC      EQU     0800H
  CRFDDLowSpeed                 EQU     0400H   ;low = '1, hi = '0
; CRRS232AInternalClock         EQU     0200H
; CRRS232BEnableClockSend       EQU     0100H
  CRDriveSel3                   EQU     0080H
  CRDriveSel2                   EQU     0040H
  CRDriveSel1                   EQU     0020H
  CRDriveSel0                   EQU     0010H
  CRSelect250KbDataRate         EQU     0008H   ;signal "5H/8L"
  CRPcomp2                      EQU     0004H
  CRPcomp1                      EQU     0002H
  CRPcomp0                      EQU     0001H
  CRFloppyMask                  EQU     3CFFH
WriteLED        EQU     PCS1Base + 10h          ;A6-A4 = 001b, W --90h
ENetAttn        EQU     PCS1Base + 20h          ;A6-A4 = 010b, W --A0h
WriteCSReg      EQU     PCS1Base + 30h          ;A6-A4 = 011b, W --B0h
WriteResetReg   EQU     PCS1Base + 40h          ;A6-A4 = 100b, W --C0h
WriteConfigReg  EQU     PCS1Base + 50h          ;A6-A4 = 101b, W --D0h

allResetBits    EQU     07FFH           ;All bits

;The following constant is used for a LOOP $ between clearing the reset bit
; for a device and setting it again to ensure that the reset signal is held
; low for the proper amount of time. It should be adjusted so that the device
; with the longest reset time is accounted for.

clocksPerusec                   EQU     8       ;running at 8MHz
clocksPerLOOP                   EQU     16D     ;from Intel handbook

usecsPerLOOP                    EQU     clocksPerLOOP/clocksPerusec

maximumResetDelayinusecs        EQU     32D     ;twice the floppy's requirement.
maximumResetDelayCount          EQU     maximumResetDelayinusecs/usecsPerLOOP

;Here are the individual device reset bit masks.

resetEthernetController         EQU     1
resetRS232CController           EQU     2
resetFloppyController           EQU     4
resetKeyboardUART               EQU     8
resetUmbilicalController        EQU     10h
resetKeyboardController         EQU     20h
resetMesaProcessor              EQU     40h
resetPCProcessor                EQU     80h
resetDiskController             EQU     100h
```

```
resetDiskDMAController          EQU     200h
resetExpansionChannel           EQU     400h

;Here are some composite device reset masks.

resetKeyboardHardware   EQU     resetKeyboardUART + resetKeyboardController
resetDiskHardware       EQU     resetDiskController + resetDiskDMAController




;-------------------------------------------------------------------------

; Interrupt  Controller -- Master i8259A

; [Programming information on Page 2-120 of
; the Intel '84 Microsystem Components Handbook]

%SET(i8274NonVectored,1)

;
i8259MasterAddr0        EQU     i8259MasterBase +0h      ;A1 = 0
i8259MasterAddr1        EQU     i8259MasterBase +2h      ;A1 = 1

i8259MasterInServiceRegAddr     EQU     i8259MasterAddr0
                                ;OCW3 (A1=0) is in-service register when i8259ISRread
                                ; is given

i8259MasterRequestRegAddr       EQU     i8259MasterAddr0
                                ;OCW3(A1=0) is also interrupt request register when
                                ; i8259IRRread is given

i8259MasterMaskRegAddr          EQU     i8259MasterAddr1
                                ;This is always the mask register (read from where its written)


;
i8259MasterICW1 EQU     011h    ;ICW1, edge triggered, cascade mode, ICW4 needed
i8259MasterICW2 EQU     020h    ;interrupt types 20h-27h
%IF(%i8274NonVectored) THEN (
i8259MasterICW3         EQU     060H    ;i8274 not a slave in non-vectored mode
) ELSE (
i8259MasterICW3         EQU     070H    ;IR4 - slave i8274 --for ETCH TWO!!!
;i8259MasterICW3        EQU     0E8h    ;IR3 - slave i8274 --for ETCH ONE!!!
) FI
                                ;IR5 - slave i8259
                                ;IR6 - slave i80186
                                ;IR7 - slave expansion slot -- for ETCH ONE!!!
i8259MasterICW4 EQU     011h    ;SFNM, Not Buffered, Normal EOI, 86/88 mode


i8259MasterOCW1 EQU     0FFh    ;Nothing is enabled
i8259MasterOCW2 EQU     0C7h    ;IR7 has lowest priority
i8259MasterOCW3 EQU     008h    ;Not special mask mode
i8259MasterDebuggerInEOI EQU 61h  ;specific EOI for debugger Int Handler

i8259AllEnabled         EQU     00000000B ;All interrupts are enabled...
i8259AllInhibited       EQU     0FFh    ;All interrupts are inhibited...
i8259EnableIR0          EQU     0FEh    ;Enable IR0 (for OCW1)
i8259EnableIR1          EQU     0FDh    ;Enable IR1 (for OCW1)
i8259EnableIR2          EQU     0F8h    ;Enable IR2 (for OCW1)
i8259EnableIR3          EQU     0F7h    ;Enable IR3 (for OCW1)
i8259EnableIR4          EQU     0EFh    ;Enable IR4 (for OCW1)
i8259EnableIR5          EQU     0DFh    ;Enable IR5 (for OCW1)
i8259EnableIR6          EQU     0BFh    ;Enable IR6 (for OCW1)
i8259EnableIR7          EQU     07Fh    ;Enable IR7 (for OCW1)


i8259EOIforIR0          EQU     060h    ;Specific EOI for IR0 (for OCW2)
i8259EOIforIR1          EQU     061h    ;Specific EOI for IR1 (for OCW2)
i8259EOIforIR2          EQU     062h    ;Specific EOI for IR2 (for OCW2)
i8259EOIforIR3          EQU     063h    ;Specific EOI for IR3 (for OCW2)
i8259EOIforIR4          EQU     064h    ;Specific EOI for IR4 (for OCW2)
i8259EOIforIR5          EQU     065h    ;Specific EOI for IR5 (for OCW2)
i8259EOIforIR6          EQU     066h    ;Specific EOI for IR6 (for OCW2)
i8259EOIforIR7          EQU     067h    ;Specific EOI for IR7 (for OCW2)

i8259EOINonSpecific     EQU     20h     ;Non-specific EOI for 8259s


i8259ISRread            EQU     00Bh    ;Read In-Service-Register on
                                        ;next Rd pulse (for OCW3)
i8259IRRread            EQU     00Ah    ;Read Intr-Request-Register on
                                        ;next pulse Rd pulse (for OCW3)
;To read IMR of i8259, set A1 = 1

;-------------------------------------------------------------------

; Interrupt  Controller -- Slave i8259A

; [Programming information on Page 2-120 of
; the Intel '84 Microsystem Components Handbook]


;
i8259SlaveAddr0 EQU     i8259SlaveBase +0h       ;A1 = 0

HardDefs.asm        29-Oct-87 16:03:15 PST
```

```
i8259SlaveAddr1 EQU     i8259SlaveBase +2h      ;A1 = 1
;
i8259SlaveICW1  EQU     011h    ;ICW1, edge triggered, cascade mode, ICW4 needed
i8259SlaveICW2  EQU     030h    ;interrupt types 30h-37h
i8259SlaveICW3  EQU     005h    ;This slave is connected to IR5 of the master
i8259SlaveICW4  EQU     001h    ;not SFNM, Not Buffered, Normal EOI, 86/88 mode


i8259SlaveOCW1  EQU     0FFh    ;Nothing is enabled
i8259SlaveOCW2  EQU     0C7h    ;IR7 has lowest priority
i8259SlaveOCW3  EQU     008h    ;Not special mask mode

i8259SlaveInServiceRegAddr      EQU     i8259SlaveAddr0
                                ;OCW3 (A1=0) is in-service register when i8259ISRread
                                ; is given

i8259SlaveRequestRegAddr        EQU     i8259SlaveAddr0
                                ;OCW3(A1=0) is also interrupt request register when
                                ; i8259IRRread is given

i8259SlaveMaskRegAddr           EQU     i8259SlaveAddr1
                                ;This is always the mask register (read from where its written)



;---------------------------------------------------------------

; Interrupt  Controller -- Expansion slot i8259A

; [Programming information on Page 2-120 of
; the Intel '84 Microsystem Components Handbook]


;
i8259OptionsSlaveAddr0  EQU     i8259OptionsSlaveBase +0h       ;A1 = 0
i8259OptionsSlaveAddr1  EQU     i8259OptionsSlaveBase +2h       ;A1 = 1

i8259OptionsSlaveICW1   EQU     013h    ;ICW1, edge triggered, single, ICW4 needed
i8259OptionsSlaveICW2   EQU     000h    ;no interrupt vector generated
;i8259OptionsSlaveICW3 ******** not used *********
i8259OptionsSlaveICW4   EQU     001h    ;not SFNM, Not Buffered, Normal EOI, 86/88 mode

i8259Poll       EQU     00CH            ;i8259 in poll mode

i8259OptionsSlaveOCW1   EQU     0FFh    ;Nothing is enabled
i8259OptionsSlaveOCW2   EQU     0C7h    ;IR7 has lowest priority
i8259OptionsSlaveOCW3   EQU     008h    ;Not special mask mode

i8259OptionsSlaveInServiceRegAddr       EQU     i8259OptionsSlaveAddr0
                                        ;OCW3 (A1=0) is in-service register when i8259ISRread
                                        ; is given

i8259OptionsSlaveRequestRegAddr EQU     i8259OptionsSlaveAddr0
                                        ;OCW3(A1=0) is also interrupt request register when
                                        ; i8259IRRread is given

i8259OptionsSlaveMaskRegAddr    EQU     i8259OptionsSlaveAddr1
                                        ;This is always the mask register (read from where its written)

;-------------------------------------------------------------------------
;-- This section is for Daybreak dependent hardware parameters.
;--
daybreakMapIOAddressBase                EQU     0E010H
daybreakMapRegisterNumberBase           EQU     0
daybreakMapIOAddressPCBase              EQU     0E010H
daybreakMapRegisterNumberPCBase         EQU     daybreakMapRegisterNumberBase
daybreakMapIOAddressIOPBase             EQU     0E018H
daybreakMapRegisterNumberIOPBase        EQU     8
nilMapData                              EQU     0FFH; illegal map reg data for init.
;-------------------------------------------------------------------------
;-- This section is for Daisy dependent hardware parameters.
;--
daisyMapIOAddressBase           EQU     0804H
daisyMapRegisterNumberBase      EQU     0
daisyMapIOAddressPCBase         EQU     daisyMapIOAddressBase
daisyMapRegisterNumberPCBase    EQU     daisyMapRegisterNumberBase
daisyMapIOAddressIOPBase        EQU     daisyMapIOAddressBase
daisyMapRegisterNumberIOPBase   EQU     8
maxChipCount                    EQU     4 ;number Of A-chips at one time.
;-------------------------------------------------------------------------


;-- This section defines IOP device addresses.
;-- Device control symbols appear in the respective definitions file for the
;--  device.

;-------------------------------------------------------------------------
;  Intel (i8251)
KeyBdUartData           EQU     i8251Base + 0h  ;A1 = 0, R/W
KeyBdUartCtl            EQU     i8251Base + 2h  ;A1 = 1, W
KeyBdUartStatus         EQU     i8251Base + 2h  ;A1 = 1, R


;-------------------------------------------------------------------------


;Timers  (i8254)
```

```
; [Programming information on Page 2- of
; the Intel '84 Microsystem Components Handbook]

i8254Count0     EQU     i8254Base + 0h   ;A2-A1 = 00h, R/W
i8254Count1     EQU     i8254Base + 2h   ;A2-A1 = 01h, R/W
i8254Count2     EQU     i8254Base + 4h   ;A2-A1 = 10h, R/W
i8254Ctlr       EQU     i8254Base + 6h   ;A2-A1 = 11h, W


;---------------------------------------------------------------

; RS232C Channels (i8274)

; [Programming information on Page ???]


i8274DCommADataAddr     EQU     i8274DCommBase + 0h      ;A2-A1 = 00b
i8274DCommACtlrAddr     EQU     i8274DCommBase + 4h      ;A2-A1 = 10b
i8274DCommBDataAddr     EQU     i8274DCommBase + 2h      ;A2-A1 = 01b
i8274DCommBCtlrAddr     EQU     i8274DCommBase + 6h      ;A2-A1 = 11b

i8274WriteRegister0     EQU     0
i8274WriteRegister1     EQU     1
i8274WriteRegister2     EQU     2
i8274WriteRegister3     EQU     3
i8274WriteRegister4     EQU     4
i8274WriteRegister5     EQU     5
i8274WriteRegister6     EQU     6
i8274WriteRegister7     EQU     7
i8274ReadRegister0      EQU     0
i8274ReadRegister1      EQU     1
i8274ReadRegister2      EQU     2

i8274EOIPort            EQU     i8274DCommACtlrAddr
i8274EOICommand         EQU     038h                    ;can only be sent to chA.
i8274RstChannelCommand  EQU     18H
i8274RstRxCRCCommand    EQU     50H
i8274RstTXCRCCommand    EQU     90H
i8274RstIntrCommand     EQU     010h                    ;for either ch.
i8274RstErrorCommand    EQU     030h                    ;for either ch.
%IF(%i8274NonVectored) THEN (
i8274OpieInitCommand    EQU     00010100B               ;RTS',non-vectored, 8086 mode,
                                ;  Rx* priority, both interrupt run.
) ELSE (
i8274OpieInitCommand    EQU     00110100B               ;RTS',vectored, 8086 mode,
                                ;  Rx* priority, both interrupt run.
) FI
i8274VarVect            EQU     04H                     ;WR1, ch. B: variable vectored interrupts


;---------------------------------------------------------------

; Burdock Umbilical Port (i8255)

; [Programming information on
; the Intel '82 Data Component Catalog]

i8255portA      EQU     i8255Base+0h    ;(R,W)
i8255portB      EQU     i8255Base+2h    ;(R,W)
i8255portC      EQU     i8255Base+4h    ;(R,W)
i8255ctl        EQU     i8255Base+6h    ;(W only) mode instruction, bit set/reset

;
;---------------------------------------------------------------

; Floppy Disc Controller (i8272)
; [Programming information on Page 9-146 of
; the Intel '82 Data Component Catalog]

FDCStatusReg            EQU     i8272Base + 0h          ;A1 = 0
FDCDataReg              EQU     i8272Base + 2h          ;A1 = 1
FDCDMADataReg           EQU     i8272Base + 4h          ;A2, A1 = 1, 0
FDCMotorPort            EQU     WriteCtlReg             ;its in the general control register

;Timer 1 external clock is connected to FDC

FloppyTimerMCWAddr      EQU     i186Timer1MCWAddr       ;Timer 1 is used by Floppy.
FloppyTimerMaxCountReg  EQU     i186Timer1CountAAddr    ;Timer 1 is used by Floppy.
FloppyTimerCountReg     EQU     i186Timer1CountRegAddr  ;Timer 1 is used by Floppy.
FloppyTimerIntCntrlReg  EQU     i186Timer1IntCtlAddr    ;Timer 1 is used by Floppy.

;Definitions for 186 DMA controller connected to floppy disk:

FloppyDMAIntCtlAddr     EQU     i186DMA0IntCtlAddr      ;DMA 0 Interrupt Control Register
FloppyDMALowSourcePtr   EQU     i186DMA0LowSourcePtr    ;DMA 0 Low order source pointer
FloppyDMAHighSourcePtr  EQU     i186DMA0HighSourcePtr   ;DMA 0 High order source pointer
FloppyDMALowDestPtr     EQU     i186DMA0LowDestPtr      ;DMA 0 Low order destination pointer
FloppyDMAHighDestPtr    EQU     i186DMA0HighDestPtr     ;DMA 0 High order destination pointer
FloppyDMATransferCount  EQU     i186DMA0TransferCount   ;DMA 0 transfer count
FloppyDMAControlWord    EQU     i186DMA0ControlWord     ;DMA 0 control word

;ResetFDC               EQU     PCS1Base + 4h           See Above...
;ResetFDCnRS232 EQU     PCS1Base + 4h           See Above...

;---------------------------------------------------
```

```
; Ethernet controller equates (i82586)
; (Fill in later...)


;-------------------------------------------------
;
; Processor Interrupt Source types

;
; Constant Equates
TRUE            EQU     -1h
FALSE           EQU     0h
ZERO            EQU     0h
;
; System memory sizes
OneK            EQU     1024            ;Used for calculations
LowRamStart     EQU     0               ;Start at Absolute 0
LowRamSize      EQU     16*OneK         ;16K bytes (16384d=04000h)
MidRamStart     EQU     01000h          ;Start at Address 010000
MidRamSize      EQU     16*OneK         ;16K bytes (16384d=04000h)
StackSize       EQU     256             ;256 bytes (100h)
RomStart        EQU     0FC00h          ;Start at absolute FFC00h
RomSize EQU     16*OneK         ;16K bytes (16384d=04000h)
NumOfInterrupts EQU     256             ;256 Interrupt Types


;---------------------------------------------
; EEProm Definitions:
;---------------------------------------------

;ETCH ONE/HYBRID Data for WriteConfigReg (from HardDefs)
;EEPEnable              EQU     8000H
;EEPWriteDataMask       EQU     1000H
;EEPClk                 EQU     0100H

;ETCH TWO Data for WriteConfigReg (from HardDefs)
EEPEnable               EQU     1000H
EEPWriteDataMask        EQU     8000H
EEPClk                  EQU     2000H

;Data for ReadInputReg (from HardDefs)
EEPReadDataMask         EQU     0800H
EEPStatusReady          EQU     0800H

;Command codes
EEPCmdRead              EQU     80H     ;10aaaaaa, where aaaaaa = word address
EEPCmdWrite             EQU     40H     ;01aaaaaa
EEPCmdErase             EQU     0C0H    ;11aaaaaa
EEPCmdEWEnable          EQU     30H     ;0011xxxx, where xxxx = don't care
EEPCmdEWDisable         EQU     00H     ;0000xxxx
EEPCmdReset             EQU     20H     ;0010xxxx

;Word or byte offset constants

bytesInEEProm           EQU     128

byteEEPromOffset        EQU     0000H
wordEEPromOffset        EQU     0100H

;Segment offset constants (high byte must be 00/10/20H shl'ed by 1 for IOPKernl)

ROMsegment              EQU     0000H
RAMSegment              EQU     2000H
badPageSegment          EQU     4000H


; The bit 5 and 6 of machineIDPort and DaisyDisplayTypePort encode the machine type.
; These assignments are the following:
;
;  Bit 6  Bit 5      Machine  ID
;  -----  -----     -------------
;    0      0     Daisy with 19" display
;    0      1     Daisy with 15" display
;    1      0     Dahlia
;    1      1     Daybreak


;---------------------------------------------
; Daisy Daybreak Distinction:
;---------------------------------------------

machineIDPort   EQU     0080H           ;query here to get a machine ID.
machineIDMask   EQU     0040H           ;only these 2 bits hold the ID.
Daisy           EQU     0000H           ;machine ID = this if Daisy.
Daybreak        EQU     0040H           ;machine ID = this if Daybreak.

;---------------------------------------------
; Daybreak Dahlia Distinction:
;---------------------------------------------

machineIDMask2  EQU     0020H           ;only these 2 bits hold the ID.
Dahlia          EQU     0000H           ;machine ID = this if Dahlia.
Daybreak2       EQU     0020H           ;machine ID = this if Daybreak.

;---------------------------------------------
; Daisy (only) 15"/19" screen size Distinction:
```

```
;-----------------------------------------

DaisyDisplayTypePort    EQU     0080H   ;query here to get display type.
DaisyDisplayTypeMask    EQU     0020H   ;only this bit holds the type.
DaisyfifteenInch        EQU     0020H   ;.
DaisynineteenInch       EQU     0000H   ;.

;-----------------------------------------

;       End of HardDefs
;-----------------------------------------
```

```
;Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.

;-- stored as [Idun]<WDLion>Dove>IOPDefs.asm
;-- created on  14-Feb-84 11:13:22
;-- This file contains public definitions which Opie exports to its clients.
;-- Any hardware dependent defintions which Opie uses can be found in
;-- hardOpie.asm, and Opie's private definitions are found in OpieDefs.asm.
;
;-- last edited by:

;--     JPM     24-Jul-85 12:00:42      :Added extendedBusPageOpieAddress
;--     JPM     26-Jun-85 15:40:03      :Removed Crash macro
;--     JPM     24-Jun-85  9:11:00      :Change taskContextBlock again
;--     JPM     20-Jun-85 10:25:05      :Change taskContextBlock
;--     JPM     15-May-85 13:42:34      :Opie redesign
;--     KEK     11-Mar-85 17:40:56      :fixed returnSPSS fatfinger
;--     KEK      2-Mar-85 19:07:37      :add returnSPSS, remove unexpaectedInterrupt and WatchDogTimeout.
;--     VXS     27-Nov-84 11:48:31      :Change DW to DB SIZE QueueEntry in TCB definition.
;--     VXS     20-Nov-84 22:15:55      :Move def of conditionTimeout to IOPMacro.asm
;--     VXS     20-Nov-84 17:06:21      :Take out hardware dependent non-exported definitions and move them into HardOpie.asm
;--     VXS     20-Nov-84 15:13:46      :change TCB structure item queue to taskQueue to avoid naming conflict.
;--     VXS     15-Nov-84 11:53:21      :Add ControlRegData to IOPEFCB
;--     VXS     14-Nov-84 12:59:30      :Introduce QueueEntry structure
;--     VXS      5-Nov-84 16:07:55      :Add map register memory image.
;--     VXS      1-Nov-84 13:19:02      :add conditionTimeout definition
;--     VXS     17-Oct-84 16:34:14      :change floppy and options DMA channels to i186IntrChannelforDMA0 and i186IntrChannelforDMA1
                                        : instead of 1 and 2.  Also change to using i186IntrChannelforTimer2 for the IOPE timer.
;--     VXS     16-Oct-84 19:26:32      :Add byte to TCB to make it an even number of words.
;--     VXS     12-Oct-84 18:19:09      :add new location to TCB so can restart a task.
;--     VXS     11-Oct-84 11:39:44      :Added mesa page map stuff to IOPE FCB
;--     VXS     11-Oct-84 10:10:30      :Remove Allow? macros
;--     VXS      4-Oct-84 19:15:42      :Add OpieAddressLow to be consistent with OpieAddressHigh alias.
;--     VXS      4-Oct-84 18:05:23      :Add AllowRDC and AllowPC macro definitions for SystemLoop
;--     VXS      4-Oct-84 17:24:28      :Add map register assignments.
;--     VXS      4-Oct-84 15:48:25      :Add TCB location generalMapData
;--     VXS      3-Oct-84 16:58:18      :Moved FloppyTimer stuff to HardDefs since timer 1 is hardwired to floppy stuff.
;--     VXS      1-Oct-84 19:54:31      :Changed some "sizeof" variables to use SIZE of the structure rather than use number by hand
;--     VXS      1-Oct-84 19:04:27      :Added definitions to allocate timer 0 to the floppy disk handler
;--                                      Corrected Spelling of IOPETimeIntCntrlReg to IOPETimerIntCntrlReg
;--     VXS      1-Oct-84 16:13:19      :Remove client condition definition bit (8000)
;--     VXS     26-Sep-84 15:46:20      :Changed GENONLY listing stuff at Interruptcontrollers macro call
;--     VXS     25-Sep-84 12:48:20      :Eliminate MesaLogicalByte from Opie Addresses
;--     VXS     24-Sep-84 18:30:22      :Change GEN to GENONLY for better listings.
;--     VXS     18-Sep-84 20:22:01      :Fix save,gen, restore stuff to be in first column
;--     VXS     17-Sep-84 18:47:01      :Add symbols for interrupt trouble routine AX indicator values
;--     VXS     17-Sep-84 17:04:20      :Add save,gen,restore around macro expansions to show symbol definitions
;--     VXS      6-Sep-84 14:26:26      :Add new Crash macro.
;--     VXS     27-Aug-84 15:58:46      :Add i8259OptionsSlaveIntrptMaskPort symbol.
;--     VXS     27-Aug-84 15:26:33      :Redefine Opie Addresses
;--     VXS     23-Aug-84 15:53:58      :change sizeOf???Available in IOPEFCB to endOf
;--     VXS     22-Aug-84 16:57:58      :Added i186LogicalOpieAddress (for ds:offset things)
;--     VXS     22-Aug-84 11:12:35      :Put in equates for Opie Addresses, Make IOPEFCB same size as TCBs are (why? find out later)
;--     VXS     14-Aug-84 12:32:57      :Change taskcontextblock for private stacks (take out entryipcs, put in taskSPSS)
;--     VXS     14-Aug-84 12:05:25      :Get rid of sizeofTCB
;--     VXS      7-Aug-84 13:58:01      :Change IntrptVctBase to IntrptVctType for correctness
;--                                      : Also SoftwareInterruptBase ← Type
;--     VXS      3-Aug-84 13:20:26      :Take out hardware stuff, will now be found in HardDefs.asm
;--     VXS     19-Jul-84 18:16:47      :Put in $GEN directive
;--     VXS     12-Jul-84 16:35:09      :Moved Devices macro here so IOregion can use it
;--     VXS     11-Jul-84 16:28:56      :Took out device specific symbol defs, put them into
                                        : seperate Definitions files.
;--     JPM     11-Jul-84 11:24:39      :added EEProm indexes (first pass per PJT, 25-Jun-84)
;--     VXS      9-Jul-84 18:23:32      :fixed erroneous definition of i8259EOINonSpecific
;--     VXS      5-Jul-84 16:14:29      :changed sizeofICB def to use SIZE
                                        :removed individual ICB symbol definitions, replaced
                                        : with Devices macro in IOPData.asm
                                        :Fixed error in EOI command port
;--     JPM      3-Jul-84 14:31:37      :added wordsInEEProm
;--     FXB     29-Jun-84 16:18:18      :added Umbilical Handler constants
;--     JMM     27-Jun-84 15:07:10      :Compatible with Opie Version 1 release.
;--     VXS     24-Jun-84 14:36:36      :Changed i80186SlaveIntrptOn to include all
                                         slave interrupt channels on the master
;--     ETN     26-Jun-84 15:30:15      :TCB struc.
;--     JMM     22-Jun-84  7:06:32      :Misc. fixes.
;--     ETN     21-Jun-84 18:33:26      :Remove IORegion EQUs
;--     JMM     19-Jun-84 22:05:24      :Misc. fixes.
;--     JMM     17-Jun-84  6:39:19      :structure updates.
;--     JMM     10-Jun-84 17:29:33      :Version 1 release.

$NOGEN

;-------------------------------------------------
;
; Constant Equates

LowNibbleMask   EQU     0FH
HighNibbleMask  EQU     0F0H
Null            EQU     0H
Nibble          EQU     4H
;

;-----------------------------------------------------------------------------
;Opie DATA STRUCTURES:

;
;------------------------

IOPDefs.asm         24-Jul-85 12:00:46 PDT
```

```
;
Condition               STRUC

TCBLinkPtr              DW      ?

Condition               ENDS
;
sizeOfCondition EQU     SIZE Condition

nonNilPtr               EQU     8000H           ;Since 0 is a valid offset, high bit in TCBLinkPtr
                                                ; signifies non-null ptr
preNotifyFlag           EQU     0001H           ;If in the TCBLinkPtr for condition variable,
                                                ; means prenotify has happened.
;-----------------------
;
ClientCondition STRUC

handlerID               DB      ?       ;ID of client
conditionRelMaskPtr     DB      ?       ; = (maskPtr - conditionPtr)
conditionPtr            DW      ?       ;in client's IORegion segment
clientMask              DW      ?       ;if 0, conditionRelMaskPtr ignored

ClientCondition ENDS

sizeOfClientCondition   EQU     SIZE ClientCondition

;-----------------------
;

;Here are definitions for Opie Addresses.
;An Opie address is a 32 bit quantity which is capable of describing the various
; address spaces and variations on those address spaces encountered in the IOP.
;The addresses are defined as a structure here for clarity, but it may be
; more convenient in actual coding to just use the type equates, and not use
; the structure offsets.
;
;Although this is not guaranteed for all future releases, the type field can
; be thought of as being broken up into 3 fields:
;       Bits 7-6        Logical Address Space Selector
;       Bits 5-4        Format Determination (Nil, Byte, Word, Page)
;       Bits 3-0        Base Specification (dependent on Logical Address Space)
;
;There are several types of nil Opie Address, but only one is really supported.
;
;See OpieAddresses.txt for further discussion of these types.
;
;mesaLogical is mesaVirtual with real memory guaranteed to be behind the involved pages.


nilOpieAddress                  EQU     0
extendedBusOpieAddress          EQU     010H
extendedBusPageOpieAddress      EQU     030H
IOPLogicalOpieAddress           EQU     050H
IOPIORegionOpieAddress          EQU     051H
PCLogicalOpieAddress            EQU     090H
mesaLogicalWordOpieAddress      EQU     0E0H
mesaEnvBaseWord                 EQU     0E1H
mesaLogicalPageOpieAddress      EQU     0F0H




OpieAddress             STRUC

OpieAddressA15toA0      DW      ?       ;bits 15 to 0
OpieAddressA23toA16     DB      ?       ;bits 23 to 16
OpieAddressType         DB      ?       ;first byte is type, see equates above

OpieAddress             ENDS

;If OpieAddressType = IOPIORegionOpieAddress

OpieAddressHandlerID    EQU     BYTE PTR OpieAddressA23toA16

;For accessing the high word as a word (low provided for consistency):

OpieAddressHigh         EQU     WORD PTR OpieAddressA23toA16
OpieAddressLow          EQU     WORD PTR OpieAddressA15toA0


;-----------------------
;Map register assignments (machine independent)

PCEMapRegisterBase      EQU     0       ;base is zero

IORegionMapRegister     EQU     8+0             ;Has lower 16k shadowed by EPROM.
mesaVMMapRegister       EQU     8+1
comRecMapRegister       EQU     8+2
comSendMapRegister      EQU     8+3
floppyDMAMapRegister    EQU     8+4
optionDMAMapRegister    EQU     8+5
generalMapRegister      EQU     8+6
spareMapRegister        EQU     8+7             ;Has upper 16K shadowed by EPROM.

;
;-----------------------
```

```
;
QueueEntry              STRUC

IOPEQueueType           DB      ?               ;e.g. system, timer
nextHandlerID           DB      ?               ;ID of next task in queue
nextTCBLinkPtr          DW      ?               ;offset of next task

QueueEntry              ENDS


;-----------------------
;
taskContextBlock        STRUC

taskQueue               DB      (SIZE QueueEntry) DUP (?)
taskCondition           DW      ?       ;if in waitForCondition state
taskICPtr               DW      ?       ;set by ThisTaskServices
taskSP                  DW      ?       ;holds stack pointer while waiting
returnSPSS              DW      ?       ;holds return-from-int addresses
                        DW      ?
taskState               DB      ?       ;Bits 7-4 Previous state, Bits 3-0 Present state.
taskHandlerID           DB      ?       ;set by InitializeTask
timerValue              DW      ?       ;counted down by timer

taskContextBlock        ENDS


;-----------------------
;
ICBcodeBytes            EQU     6       ;PUSHA, CALL FAR GenericInterruptProcessing

interruptContext        STRUC           ;do not alter the order of the fields!

interruptStatus         DB      ?       ;task waiting, active, timed out, ...
interruptHandlerID      DB      ?       ;for task servicing this interrupt
interruptTCBLinkPtr     DW      ?       ;task (set by ThisTaskServices)
interruptTimerValue     DW      ?       ;counted down by watchdog (set by WaitForInterrupt)
watchdogLinkPtr         DW      ?       ;next IC in watchdog queue
troubleIPCS             DW      ?       ;proc called for unexpected interrupt
                        DW      ?       ; (set by ThisTaskServices)
interruptMask           DB      ?       ;used for enable/disable
interruptSlaveEOIcmd    DB      ?       ;used to clear interrupt
interruptController     DW      ?       ;link to controller STRUC (private)

interruptContext        ENDS
;
sizeOfIC                EQU     (SIZE interruptContext) ;in bytes!

variableSizeOfIC        EQU     interruptMask-interruptStatus
                                ;***equals number of bytes in the
                                ; interruptContext that are variable


interruptContextBlock   STRUC

ICBcode                 DB      ICBCodeBytes DUP (?)
ICBcontext              DB      (SIZE InterruptContext) DUP (?)

interruptContextBlock   ENDS
;
sizeOfICB               EQU     (SIZE interruptContextBlock)    ;in bytes!

%*DEFINE(softwareIntrptVctType)()       ;tell IOPMacro this is not defined.
```

```
;Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.

;-- stored as [Idun]<WDLion>Dove>IOPMacro.asm
;-- created on  14-Feb-84 11:13:22
;
;-- last edited by:
;
;--     KEK     11-Aug-86 16:57:43      :changes for multiple options support
;--     KEK      1 Aug 85 18:37:33      :Changed ReadEEProm macro.
;--     JPM     22-Jul-85 13:46:58      :Changed order of loading parameters in NotifyClientCondition (load BX last since might be used
in address specification).
;--     JPM     18-Jul-85  9:34:57      :Changed SetupOpieAddressInCXDX to use struc fields.
;--     JPM     16-Jul-85  9:25:16      :Added WORD PTR to client condition MOVs.
;--     JPM     28-Jun-85 13:07:01      :Separated SystemCalls into ROM and RAM portions (RAM empty for now).
;--     JPM     25-Jun-85 17:33:54      :Changed macros which use interruptName (no EXTRNs).
;--     JPM     24-Jun-85  9:13:53      :Changed ReadEEProm.
;--     JPM     18-Jun-85 13:15:01      :Add GetIntervalTimer; use SI for all taskPtr parms.
;--     JPM     24-May-85 13:40:15      :Remove alternate locking from MesaLockedOut macro
;--     JPM     15-May-85 13:52:31      :Opie redesign
;--     kEK      2-Mar-85 19:07:10      :add interruptTimeout
;--     kEK     19-Feb-85 14:49:37      :Add PUSH/POP ES to WaitForMumble. cleaned up code format
;--     VXS     26-Nov-84 16:19:25      :Fix bug in ClientCondition (include code inside generateEScall)
;--     VXS     20-Nov-84 22:17:23      :add conditionTimeout definition.
;--     VXS     15-Nov-84 13:50:25      :Add ControlRegister macro.
;--     VXS      5-Nov-84 18:04:31      :add RegisterPCEStartRoutine, CallPCEStartRoutine
;--     VXS      5-Nov-84 17:59:29      :Add SI-DI arg to restart for runtime determination of restart address
;--     VXS     15-Oct-84 10:19:33      :Install Reset macro
;--     VXS     12-Oct-84 18:32:25      :Install Restart macro
;--     VXS     10-Oct-84 18:36:41      :Changed Reset macro to do a software interrupt instead of inline code.
;--     VXS      3-Oct-84 11:35:44      :Fix confusion on SetupOpieAddressInCXDX macro
;--     VXS     28-Sep-84 18:11:40      :Put delay test cx,[bx] instruction in MesaLockedOut macro
;--     VXS     20-Sep-84 19:50:58      :Fix NotifyClientCondition macro bugs
;--     VXS     19-Sep-84 17:30:01      :Add EstablishIOPAccess
;--     VXS     17-Sep-84 16:52:02      :Change NotifyClientcondition to use LEA instead of macro
;--     VXS     31-Aug-84 15:37:59      :Use a macro to generate System call instructions, and so can use it in IOPData.asm.
;--     VXS     31-Aug-84 15:28:58      :Added NotifyClientcondition macro, INT
;--     VXS     28-Aug-84 12:06:40      :Changed arg to SubInterrupt to be just interrupt name rather than interruptType
;--     VXS     27-Aug-84 17:48:59      :Added new SubInterrupt macro definition
;--     JPM     20-Aug-84 16:15:31      :Took spaces out of MesaLockedOut string compares
;--     VXS     21-Aug-84 18:01:35      :Added MinimumStackSize defintiion
;--     VXS     16-Aug-84 12:36:59      :Change Initialize Task for private stacks.
;--     VXS      6-Aug-84 17:38:28      :Changed SoftwareInterruptBase to Type
;--     JPM      6-Jul-84 11:20:29      :Removed MesaLockedOutMemToMem.
;--     JPM      3-Jul-84 13:57:45      :Added ReadEEProm.
;--     FXB      2-Jul-84 14:25:36      :added BindweedIntr Codemacro
;--     JPM      2-Jul-84 13:57:56      :Added MesaLockedOutMemToMem.
;--     JMM     27-Jun-84 15:17:44      :Compatible with Opie Version 1 release.
;--     ETN     26-Jun-84 15:20:54      :InitializeTask change.
;--     ETN     21-Jun-84 18:33:54      :Restored EstablishIOPAccess.
;--     JMM/ETN 21-Jun-84 18:33:54      :Deleted Converts, etc.
;--     JMM     10-Jun-84 17:14:13      :Version 1 release.


;                       IOPMacro
;-------------------------------------------------------------------------------
;IMPORTED VARIABLES:

;Generate INT Instructions using two data bytes

InterruptCode                   EQU     0CDH

;The following must match the equivalent definition in HardOpie.asm. If it doesn't, an error will be generated by the linker.

%IF (%NES(%softwareIntrptVctType,Defined))
     THEN (
     PUBLIC     softwareIntrptVctType
     softwareIntrptVctType      EQU     96
     )FI

;-------------------------------------------------------------------------------
;
;System Macros:

%*DEFINE                (SystemCalls)
                          (
                           %SET(n,0)
                           %*DEFINE(whereDefined)(ROM)
                           %SystemCall(CallPCEStartRoutine)
                           %SystemCall(ControlRegister)
                           %SystemCall(ConvertAddress)
                           %SystemCall(Disable)
                           %SystemCall(Enable)
                           %SystemCall(EstablishHandlerAccess)
                           %SystemCall(EstablishIOPAccess)
                           %SystemCall(GetIntervalTimer)
                           %SystemCall(GetLockMask)
                           %SystemCall(GetWorkMask)
                           %SystemCall(InitializeTask)
                           %SystemCall(Jam)
                           %SystemCall(MesaLockedOut)
                           %SystemCall(NotifyClientCondition)
                           %SystemCall(NotifyCondition)
                           %SystemCall(NotifyHandlerCondition)
                           %SystemCall(ReadEEProm)
                           %SystemCall(RegisterPCEStartRoutine)
                           %SystemCall(Reset)
                           %SystemCall(Restart)

IOPMacro.asm        11-Aug-86 17:05:33 PDT
```

```
                            %SystemCall(ThisTaskServices)
                            %SystemCall(WaitForCondition)
                            %SystemCall(WaitForInterrupt)
                            %SystemCall(WaitForSystem)
                            %SystemCall(WaitForTime)
                            %*DEFINE(whereDefined)(RAM)
                            %SystemCall(GetOptionsInterrupt)
                            %SystemCall(ReleaseDMAChannel)
                            %SystemCall(WaitForDMAChannel)
                                )

%*DEFINE                    (SystemCall (Name))
                               (
                                %Name%(SIVType)      EQU      softwareIntrptVctType+%n
                                %SET(n,%EVAL(%n+1))
                                    )

        %SystemCalls

;-------------------------------------------------------------------------------

;Lower level support macros:

;Macro to setup CX-DX with an Opie address if the argument isn't CX-DX

%*DEFINE                    (SetupOpieAddressInCXDX (OpieAddressEA))
                               (
                                %IF (%NES(%OpieAddressEA,CX-DX))
                                     THEN (
                                     MOV     CX,%OpieAddressEA.OpieAddressHigh
                                     MOV     DX,%OpieAddressEA.OpieAddressLow
                                     )FI
                                    )

;-------------------------------------------------------------------------------

;User called Macros:

%*DEFINE                    (CallPCEStartRoutine)
                               (
                                DB  InterruptCode, LOW CallPCEStartRoutineSIVType
                                    )

%*DEFINE                    (ControlRegister (mask, value))
                               (
                                %IF (%NES (%mask,CX))
                                     THEN (MOV       CX, %mask
                                     )FI
                                %IF (%NES (%value,AX))
                                     THEN (MOV       AX, %value
                                     )FI
                                DB  InterruptCode, LOW ControlRegisterSIVType
                                    )

%*DEFINE                    (ConvertAddress (OpieAddressEA))
                               (
                                %SetupOpieAddressInCXDX(%OpieAddressEA)
                                DB  InterruptCode, LOW ConvertAddressSIVType
                                    )

%*DEFINE                    (Disable (interruptName))
                               (
                                %IF (%NES (%interruptName,BX))
                                     THEN (MOV  BX, %interruptName
                                     )FI
                                DB  InterruptCode, LOW DisableSIVType
                                    )

%*DEFINE                    (DisableInterruptsTillNextWait)
                               (
                                CLI
                                    )

%*DEFINE                    (Enable (interruptName))
                               (
                                %IF (%NES (%interruptName,BX))
                                     THEN (MOV  BX, %interruptName
                                     )FI
                                DB  InterruptCode, LOW EnableSIVType
                                    )

%*DEFINE                    (EstablishHandlerAccess (handlerID))
                               (
                                %IF (%NES (%handlerID,AX))
                                     THEN (MOV       AX, %handlerID
                                     )FI
                                DB  InterruptCode, LOW EstablishHandlerAccessSIVType
                                    )

%*DEFINE                    (EstablishIOPAccess (MapNo,OpieAddressEA))
                               (
                                %IF (%NES (%MapNo,AX))
                                     THEN (MOV       AX, %MapNo
                                     )FI
                                %SetupOpieAddressInCXDX(%OpieAddressEA)
                                DB  InterruptCode, LOW EstablishIOPAccessSIVType
                                    )
```

```
%*DEFINE                (GetLockMask)
                        (
                         DB  InterruptCode, LOW GetLockMaskSIVType
                             )

%*DEFINE                (GetIntervalTimer)
                        (
                         DB  InterruptCode, LOW GetIntervalTimerSIVType
                             )

%*DEFINE                (GetWorkMaskForCondition        (conditionPtr))
                        (
                         %IF (%NES (%conditionPtr,BX))
                              THEN (MOV       BX, %conditionPtr
                              )FI
                         DB  InterruptCode, LOW GetWorkMaskSIVType
                             )

%*DEFINE                (InitializeTask (handlerID, taskPtr, initLoc, initialStackPtr))
                        (
                         %IF (%NES (%handlerID,AX))
                              THEN (MOV       AX, %handlerID
                              )FI
                         %IF (%NES (%taskPtr,SI))
                              THEN (MOV       SI, %taskPtr
                              )FI
                         %IF (%NES (%initLoc,CX-DX))
                              THEN (
                              MOV     CX, OFFSET %initLoc
                              MOV     DX, CS
                              )FI
                         %IF (%NES (%initialStackPtr,DI))
                              THEN (MOV       DI, %initialStackPtr
                              )FI
                         DB  InterruptCode, LOW InitializeTaskSIVType
                             )

%*DEFINE                (Jam    (handlerID, taskPtr))
                        (
                         %IF (%NES (%handlerID,AX))
                              THEN (MOV       AX, %handlerID
                              )FI
                         %IF (%NES (%taskPtr,SI))
                              THEN (MOV       SI, %taskPtr
                              )FI
                         DB  InterruptCode, LOW JamSIVType
                             )

%*DEFINE                (MesaLockedOut  (operation, dataPtr, dataRegOrVal, lockMask))
                        (
                         %IF (%NES (%dataRegOrVal,AX))
                              THEN (MOV       AX, %dataRegOrVal
                              )FI
                         %IF (%NES (%dataPtr,BX))
                              THEN (MOV       BX, %dataPtr
                              )FI
                         %IF (%NES (%lockMask,CX))
                              THEN (MOV       CX, %lockMask
                              )FI
                         %IF (%EQS (%operation,ADD))
                              THEN (MOV       DX, 0
                              )ELSE
                                 (
                                  %IF (%EQS (%operation,AND))
                                       THEN (MOV      DX, 1
                                       )ELSE
                                          (
                                           %IF (%EQS (%operation,OR))
                                                THEN (MOV     DX, 2
                                                )ELSE
                                                   (
                                                    %IF (%EQS (%operation,XCHG))
                                                         THEN (MOV    DX, 3
                                                         )ELSE (MOV   DX, 4
                                                         )FI
                                                )FI
                                       )FI
                                 )FI
                         DB           InterruptCode, LOW MesaLockedOutSIVType
                             )

%*DEFINE                (NotifyClientCondition (clientCondition))
                        (
                         %IF (%NES (%clientCondition,AX-BX-CX))
                              THEN (
                              MOV     AX, WORD PTR %clientCondition
                              MOV     CX, WORD PTR %clientCondition[4]
                              MOV     BX, WORD PTR %clientCondition[2]
                              )FI
                         DB InterruptCode, LOW NotifyClientConditionSIVType
                              )

%*DEFINE                (NotifyCondition (conditionPtr))
                        (
                         %IF (%NES (%conditionPtr,BX))
                              THEN (MOV       BX, %conditionPtr
```

```
                    )FI
                DB  InterruptCode, LOW NotifyConditionSIVType
                    )

%*DEFINE        (NotifyHandlerCondition (handlerID, conditionPtr))
                   (
                    %IF (%NES (%handlerID,AX))
                         THEN (MOV      AX, %handlerID
                         )FI
                    %IF (%NES (%conditionPtr,BX))
                         THEN (MOV      BX, %conditionPtr
                         )FI
                    DB  InterruptCode, LOW NotifyHandlerConditionSIVType
                    )

%*DEFINE        (ReadEEProm (eePromAddress, eePromVersion))
                   (
                    %IF (%NES (%eePromVersion,AX))
                         THEN (MOV      AX, %eePromVersion
                         )FI
                    %IF (%NES (%eePromAddress,BX))
                         THEN (MOV      BX, %eePromAddress
                         )FI
                    DB  InterruptCode, LOW ReadEEPromSIVType
                    )

%*DEFINE        (RegisterPCEStartRoutine        (location))
                   (
                    %IF (%NES (%location,CX-DX))
                         THEN (
                         MOV    CX, OFFSET %location
                         MOV    DX, CS
                         )FI
                    DB  InterruptCode, LOW RegisterPCEStartRoutineSIVType
                    )

%*DEFINE        (Reset  (deviceResetMask))
                   (
                    %IF (%NES (%deviceResetMask,AX))
                         THEN (MOV      AX, %deviceResetMask
                         )FI
                    DB  InterruptCode, LOW ResetSIVType
                    )

%*DEFINE        (Restart        (handlerID, taskPtr, initLoc, initialStackPtr))
                   (
                    %IF (%NES (%handlerID,AX))
                         THEN (MOV      AX, %handlerID
                         )FI
                    %IF (%NES (%taskPtr,SI))
                         THEN (MOV      SI, %taskPtr
                         )FI
                    %IF (%NES (%initLoc,CX-DX))
                         THEN (
                         MOV    CX, OFFSET %initLoc
                         MOV    DX, CS
                         )FI
                    %IF (%NES (%initialStackPtr,DI))
                         THEN (MOV      DI, %initialStackPtr
                         )FI
                    DB  InterruptCode, LOW RestartSIVType
                    )

%*DEFINE        (SubInterrupt (interruptName))
                   (
                    DB InterruptCode, LOW %interruptName
                    )

%*DEFINE        (ThisTaskServices (interruptName, badInterruptProcLoc))
                   (
                    %IF (%NES (%interruptName,BX))
                         THEN (MOV  BX, %interruptName
                         )FI
                    %IF (%NES (%badInterruptProcLoc,CX-DX))
                         THEN (
                         MOV    CX, OFFSET %badInterruptProcLoc
                         MOV    DX, CS
                         )FI
                    DB  InterruptCode, LOW ThisTaskServicesSIVType
                    )

%*DEFINE        (WaitForCondition       (conditionParms))
                   (
                    %MATCH      (conditionPtr,timeoutInterval)  (%conditionParms)
                    %IF (%EQS (%timeoutInterval,noTimeout) OR %EQS (%timeoutInterval,%()))
                        THEN (XOR       AX, AX
                        )ELSE
                           (
                            %IF (%NES (%timeoutInterval,AX))
                                 THEN (MOV      AX, %timeoutInterval
                                 )FI
                        )FI
                    %IF (%NES (%conditionPtr,BX))
                        THEN (MOV       BX, %conditionPtr
                        )FI
                    DB  InterruptCode, LOW WaitForConditionSIVType
                    )
```

```
%*DEFINE                (WaitForInterrupt       (timeoutInterval))
                           (
                            %IF (%EQS (%timeoutInterval,noTimeout) OR %EQS (%timeoutInterval,%()))
                                THEN (XOR       AX, AX
                                )ELSE
                                   (
                                    %IF (%NES (%timeoutInterval,AX))
                                         THEN (MOV       AX, %timeoutInterval
                                         )FI
                                )FI
                            DB  InterruptCode, LOW WaitForInterruptSIVType
                                )

%*DEFINE                (WaitForSystem)
                           (
                            DB  InterruptCode, LOW WaitForSystemSIVType
                                )

%*DEFINE                (WaitForTime    (interval))
                           (
                            %IF (%NES (%interval,AX))
                                THEN (MOV       AX, %interval
                                )FI
                            DB  InterruptCode, LOW WaitForTimeSIVType
                                )

;The following are defined in RAM:


%*DEFINE                (GetOptionsInterrupt(interruptMask, handlerInterruptNumber))
                           (
                            %IF (%NES (%interruptMask,DX))
                                THEN (MOV       DX, %interruptMask
                                )FI
                            %IF (%NES (%handlerInterruptNumber,CX))
                                THEN (MOV       CX, %handlerInterruptNumber
                                )FI
                            DB  InterruptCode, LOW GetOptionsInterruptSIVType
                                )

%*DEFINE                (ReleaseDMAChannel)
                           (
                            DB  InterruptCode, LOW ReleaseDMAChannelSIVType
                                )

%*DEFINE                (WaitForDMAChannel)
                           (
                            DB  InterruptCode, LOW WaitForDMAChannelSIVType
                                )

;-------------------------------------------------------------------------------
```

```
;Copyright (C) 1985 by Xerox Corporation. All rights reserved.

;-- stored as [Iris]<WMicro>Dove>IOPStack.asm
;-- created on  15-May-85 15:47:00

;--      JPM     .es       25-Jun-85 12:00:52       :Changed MinimumStackSize from 14 to 16
;--      JPM     .es       15-May-85 15:47:00       :Creation

;This file provides support for stack segment generation.

;stack size constants (in words)

MinimumStackSize        EQU     16      ;for interrupt processing
DefaultStackSize        EQU     40      ;allows several PUSHA, PUSH, CALL

;stack allocation macro

%*DEFINE        (StackAllocation        (label, size))
                   (
                            PUBLIC  %label
                    %IF (%EQS (%size,%()))
                        THEN (  DW      DefaultStackSize DUP %((?))
                        )ELSE ( DW      %size DUP %((?))
                        )FI
                %label  LABEL   WORD
                        )

;-------------------------------------------------------------------------------
```

```
$MOD186
$PAGELENGTH (95)
$PAGEWIDTH  (136)

;Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.

;-- IORegion locations for the bootstrap handler.
;-- stored as [Iris]<WMicro>Dove>IORRAMBt.asm
;
;-- last edited by:

;--     RDH        27-Jan-87 10:53:06   ;Insert ending label.
;--     RDH        26-Jan-87 16:31:44   ;Added stuff for cp to be handled from iop init block.
;--     kek         6-May-86 18:12:25   ;added daisy stuff.
;--     JAC        10-Dec-85 11:08:40   ;added bootMemoryPtr
;--     JPM        18-Jul-85  8:26:14   ;Opie redesign conversion.
;--     JMM        17-Jun-85 11:46:33   ;Removed doneWithGermFile and doneWithDBFile.
;--     JPM         5-Jun-85 15:52:01   ;Added doneWithGermFile and doneWithDBFile.
;--     JMM        21-Feb-85 11:30:26   ;Misc. edits.
;--     JMM        20-Jan-85 14:22:27   ;Opie 16 upgrade.
;--     JMM        30-Nov-84 15:41:11   ;First release.

NAME                    IORRAMBt
;-------------------------------------------------------------------------------
;
$NOLIST
$INCLUDE                (IOPDefs.asm)
$INCLUDE                (RAMBDefs.asm)
$LIST

;-------------------------------------------------------------------------------
;
;
;*******************************************************************************

BootstrapIOR            SEGMENT         COMMON

                        Assume DS:BootstrapIOR

;EXPORTED variables:
;-------------------

PUBLIC          bootBufferFull, bootStrapTask, bootTask
PUBLIC          jumpTable, bootDeviceIORSpace
PUBLIC          bootBufferEmpty,bootBufferPtr,bootFileChoice
PUBLIC          currentBlockSequence ,currentBootBuffer
PUBLIC          finishedLoaderFileFetch, getBootFile, startOfBootBufferPool
PUBLIC          loaderVirtualMemoryLocation, bootMemoryPtr
PUBLIC          firstMicrocodeStartAddr, secondMicrocodeStartAddr
PUBLIC          firstMicrocodeInitAddr, secondMicrocodeInitAddr
PUBLIC          firstMicrocodeData, secondMicrocodeData, thirdMicrocodeData
PUBLIC          csBankConfiguration, CPType, WriteCSProc
PUBLIC          CPStartOrInitProc, IncSIFarProc
PUBLIC          EndRAMBootstrapIOR

;-----------------------------------------------


bootTask                TaskContextBlock <>     ;Generic boot task in RAM.
bootStrapTask           TaskContextBlock <>     ;ROM boot task -> RAM*Bt task.
bootBufferFull          Condition       <>      ;Condition variable for
                                                ;handler - bootstrap
                                                ;coordination.
jumpTable               BootJumpTable   <>      ;Use to do indirect jumps.


bootDeviceIORSpace      DB      200 DUP (?)     ;This section is used by the
                                                ;device boot heads for whatever
                                                ;needs they may have. i.e. One
                                                ;could define device specific
                                                ;structures in the device boot
                                                ;head then load a register with
                                                ;"deviceSpecificArea" and use
                                                ;.extension to access the needed
                                                ;variables.

;-------------------------------------------------------------------------------
;everything above is identical with IORROMBt.asm
;-------------------------------------------------------------------------------

bootBufferEmpty         Condition       <>
finishedLoaderFileFetch Condition       <>
getBootFile             Condition       <>
bootFileChoice          DB      6 DUP (?)       ;For Ethernet, this would
                                                ;contain the boot file number
                                                ;of the file we want. For disk,
                                                ;this would . . .
startOfBootBufferPool   DW      ?               ;pointer to (n) linked buffers,
                                                ;its value changes depending on
                                                ;whether we are doing a diagnostic
                                                ;boot or a normal boot (i.e. a
                                                ;diagnostic boot requires more
                                                ;room to load into)
bootBufferPtr           DW      ?               ;Pointer to load area for booting.
currentBootBuffer       DW      ?               ;For keeping track of buffer to
                                                ;process.

IORRAMBt.asm        27-Jan-87 10:58:02 PST                                  1
```

```
currentBlockSequence    DW      ?               ;For keeping track of where we are
                                                ;at in processing a boot buffer.

;The variables below are initialized at main memory set-up  for the particular religion
;and happens during an emulator load.

loaderVirtualMemoryLocation     DD      ?       ;This is an SEG:INDEX pair of where the
                                                ;loader goes.
bootMemoryPtr                   DW      ?


;for Daisy, addresses of first and second microinstruction to be started.
firstMicrocodeStartAddr         DW      ?
secondMicrocodeStartAddr        DW      ?

;for Daisy, addresses of first and second microinstruction to be inited.
firstMicrocodeInitAddr          DW      ?
secondMicrocodeInitAddr         DW      ?

;for Daisy, first, second, and third microinstruction data.
firstMicrocodeData              DW      ?
secondMicrocodeData             DW      ?
thirdMicrocodeData              DW      ?

csBankConfiguration             DB      ?
;10H, 20H, 30H, 40H => 4K, 8K, 12K, 16K etc.

CPType                  DB      ?       ;20H => Daisy, 60H => Daybreak

;Offsets and code segments to be called from RAMBoot.asm
WriteCSProc             DW 2 DUP (?)    ;Set up by InitCPSpecific in DoveCP.asm
CPStartOrInitProc       DW 2 DUP (?)

;Offsets and code segments to be called from here back into RAMBoot.asm
IncSIFarProc            DW 2 DUP (?)    ;Set up by RAMBoot.asm

EndRAMBootstrapIOR      LABEL   FAR

BootstrapIOR            ENDS


                        END
```

```
$MOD186
$PAGELENGTH (72)
$PAGEWIDTH  (136)

;Copyright (C) 1984, 1985 by Xerox Corporation. All rights reserved.

;-- IORegion locations for the bootstrap handler.
;-- stored as IORROMBt.asm
;
;-- last edited by:

;--     RDH     14-Sep-85  9:46:10      ;Added bigTimeOut, littleTimeout, and allowTimeout to make longer (35 sec) timeout period work.
;--     RDH     13-Sep-85 20:40:51      ;Added displayStartOffset and displayStartSegment for maint panel visibility.
;--     RDH     10-Sep-85 20:05:53      ;Commented out currentCursor since there is no more mouse tracking.
;--     RDH     10-Sep-85 20:05:53      ;added OverlayLength
;--     JAC     29-Aug-85  9:29:12      ;added EndBootstrapIOR
;--     RDH                             22-Aug-85 12:35:47      ;Overlay bootDeviceIORSpace with icon data strucs.
;--     RDH                             14-Aug-85 15:08:08      ;Folded in UI stuff.
;--     JPM                             18-Jul-85  8:26:07      ;Removed idledBootTasks.
;--     JPM                             11-Jul-85  8:49:13      ;Opie redesign conversion.
;--     JMM                              4-Apr-85 20:30:35      ;Misc. edits.
;--     JMM                             20-Jan-85 14:23:17      ;Opie 16 upgrade.
;--     BKI/JMM                         24-Oct-84 16:18:41      ;First release.

NAME                    IORROMBt

;-------------------------------------------------------------------------------
;
$NOLIST
$INCLUDE                (IOPDefs.asm)
$INCLUDE                (ROMBDefs.asm)
$LIST

;-------------------------------------------------------------------------------
;
;
;*******************************************************************************

BootstrapIOR            SEGMENT         COMMON

                        Assume DS:BootstrapIOR

;EXPORTED variables:
;-------------------

PUBLIC                  bootBufferFull, bootStrapTask, bootTask
PUBLIC                  jumpTable, bootDeviceIORSpace
PUBLIC                  bootDeviceICON, displaySegment, displayOffset
;PUBLIC                 currentCursor
PUBLIC                  ICONsBottomEdge, ICONsTopEdge
PUBLIC                  displayWidthInBytes, selectedICON
PUBLIC                  displayStartOffset, displayStartSegment
PUBLIC                  bigTimeOut
PUBLIC                  littleTimeOut
PUBLIC                  allowTimeout
PUBLIC                  EndBootstrapIOR
PUBLIC                  OverlayLength

;-----------------------------------------


bootTask                TaskContextBlock <>     ;Generic boot task in RAM.
bootStrapTask           TaskContextBlock <>     ;ROM boot task -> RAM*Bt task.
bootBufferFull          Condition       <>      ;Condition variable for
                                                ;handler - bootstrap
                                                ;coordination.
jumpTable               BootJumpTable   <>      ;Use to do indirect jumps.


bootDeviceIORSpace      DB      200 DUP (?)     ;This section is used by the
                                                ;device boot heads for whatever
                                                ;needs they may have. i.e. One
                                                ;could define device specific
                                                ;structures in the device boot
                                                ;head then load a register with
                                                ;"deviceSpecificArea" and use
                                                ;.extension to access the needed
                                                ;variables.

;-------------------------------------------------------------------------------
;everything above is identical with IORRAMBt.asm
;-------------------------------------------------------------------------------

ORG OFFSET bootDeviceIORSpace                   ;Overlay since icon data structures
                                                ;will not be used once boot device is
                                                ;determined.

bootDeviceICON          EQU     $
bootDevice1ICON         ICONDataStructure       <>
bootDevice2ICON         ICONDataStructure       <>
bootDevice3ICON         ICONDataStructure       <>
bootDevice4ICON         ICONDataStructure       <>

bootDevice5ICON         ICONDataStructure       <>
bootDevice6ICON         ICONDataStructure       <>
bootDevice7ICON         ICONDataStructure       <>

IORROMBt.asm        14-Sep-85  9:46:13 PDT
```

```
bootDevice8ICON         ICONDataStructure       <>

bootDevice9ICON         ICONDataStructure       <>
bootDevice10ICON        ICONDataStructure       <>


displayWidthInBytes     DW      ?
displaySegment          DW      ?       ;These are used to remember where the
displayOffset           DW      ?       ; icons go in display memory.
selectedICON            DW      ?
firstICONOffset         DW      ?
ICONsBottomEdge         DW      ?
ICONsTopEdge            DW      ?
displayStartOffset      DW      ?       ;These are used to remember the beginning
displayStartSegment     DW      ?       ; of the display.
bigTimeOut              DB      ?       ;Used to count down seconds in timeout.
littleTimeOut           DW      ?       ;Used to compare to interval timer to see if one second has passed.
allowTimeout            DB      ?       ;Flag to control if default booting will occur.
;currentCursor          cursorPosition  <>
EndBootstrapIOR         EQU     $
OverlayLength           EQU     $ - bootDeviceIORSpace

BootstrapIOR            ENDS


                        END
```

```
;Copyright (C) 1984 by Xerox Corporation. All rights reserved.

;-- stored as [Idun]<WDLion>Dove>QueDefs.asm
;-- created on  20-Aug-84 12:00:16
;
;-- last edited by:
;
;--     JoM    .es              20-Aug-84 12:00:16      :Created.

;--     Queue Block Structure

QueueBlock              STRUC
queueHead               DW      2 DUP (?)       ; OpieAddress for first item on queue, or 0 if NIL
queueTail               DW      2 DUP (?)       ; OpieAddress for last item on queue, or 0 if NIL
queueNext               DW      2 DUP (?)       ; OpieAddress for next item to process, or 0 if NIL
QueueBlock              ENDS
```

```
;Copyright (C) 1984 by Xerox Corporation. All rights reserved.

;-- stored as [Idun]<WDLion>Dove>QueMacro.asm
;-- created on  20-Aug-84 16:12:01
;
;-- last edited by:
;
;--     JPM    .es              26-Jun-85 13:13:46      :Fixed parentheses mismatch.
;--     JPM    .es              14-May-85 14:34:41      :Opie redesign.
;--     JPM    .es              29-Oct-84 8:41:32       :Substituted generalMapRegister for systemMapNo.
;--     JPM    .es              26-Oct-84 16:17:48      :Replaced constant offsets by OpieAddress field offsets.
;--     JPM    .es              2-Oct-84 17:43:09       :Revised for new EstablishIOPAccess macro.
;--     JPM    .es              12-Sep-84 15:16:59      :Added JumpIfMesaIOCBNextNil and JumpIfQueueNextNil.

; These macros define queue operations used in the IOP.
; Note: clients should call these macros only at system level.

;-------------------------------------------------------------

;-- Definition of STRUC QueueBlock is in QueDefs.asm
;-- Definition of STRUC OpieAddress is in IOPDefs.asm
;-- Additional macros referenced herein are in IOPMacro.asm

;-- AdvanceMesaIOCB advances to the next IOCB in a Mesa-maintained chain.
;--  (Note that only one word of the link is used, and locking must be done)
;-- mappedOrNot = "mapped" iff ES:DI points to the current IOCB, and
;--  at end of macro, ES:DI points to that IOCB and (ZF) is 1 iff next = NIL

%*DEFINE                (AdvanceMesaIOCB        (iocbNext, linkOffset, lockMask, mappedOrNot))
                           LOCAL Lb10 Lb11
                           ( %IF        (%NES (mapped,%mappedOrNot))
                                  THEN (%EstablishIOPAccess     (generalMapRegister,%iocbNext))  FI
                                        MOV     AX, ES:[DI].%linkOffset.OpieAddressLow
                                        MOV     BX, OFFSET %iocbNext.OpieAddressLow
                                        MOV     CX, %lockMask
                                    %Lb10:  %MesaLockedOut  (MOV,BX,AX,CX)
                                        OR      AX, AX
                                        JNZ     %Lb11
                                        MOV     AX, ES:[DI].%linkOffset.OpieAddressLow
                                        OR      AX, AX
                                        JNZ     %Lb10
                                    %Lb11:
                                    )

;-- AdvanceQueue advances to the next IOCB in an IOP-maintained chain.
;-- mappedOrNot = "mapped" iff ES:DI points to the current IOCB, and
;--  at end of macro, ES:DI points to that IOCB and (ZF) is 1 iff next = NIL

%*DEFINE                (AdvanceQueue   (queueBlock, linkOffset, mappedOrNot))
                           ( %IF        (%NES (mapped,%mappedOrNot))
                                  THEN (%EstablishIOPAccess     (generalMapRegister,%queueBlock.queueNext))  FI
                                        MOV     DX, ES:[DI].%linkOffset.OpieAddressLow
                                        MOV     %queueBlock.queueNext.OpieAddressLow, DX
                                        MOV     CX, ES:[DI].%linkOffset.OpieAddressHigh
                                        MOV     %queueBlock.queueNext.OpieAddressHigh, CX
                                        OR      CH, CH
                                    )


;-- DeQueue removes the first IOCB from an IOP-maintained chain.
;--  (caller must ensure that queueBlock.queueHead is not NIL)

%*DEFINE                (DeQueue        (queueBlock, linkOffset, preserveESOrNot))
                           LOCAL Lb10
                           ( %IF        (%EQS (%preserveESOrNot,preserveES))
                                  THEN (PUSH    ES)FI
                                        %EstablishIOPAccess     (generalMapRegister,%queueBlock.queueHead)
                                        MOV     DX, ES:[DI].%linkOffset.OpieAddressLow
                                        MOV     CX, ES:[DI].%linkOffset.OpieAddressHigh
                                  %IF        (%EQS (%preserveESOrNot,preserveES))
                                  THEN (POP     ES)FI
                                        MOV     %queueBlock.queueHead.OpieAddressLow, DX
                                        MOV     %queueBlock.queueHead.OpieAddressHigh, CX
                                        CMP     CH, nilOpieAddress
                                        JNZ     %Lb10
                                        MOV     %queueBlock.queueTail.OpieAddressLow, DX
                                        MOV     %queueBlock.queueTail.OpieAddressHigh, CX
                                    %Lb10:
                                    )


;-- EnQueue adds an IOCB to the end of an IOP-maintained chain.
;-- CX:DX contains the OpieAddress of the IOCB to be added

%*DEFINE                (EnQueue        (queueBlock, linkOffset, preserveESOrNot))
                           LOCAL Lb10 Lb11 Lb12
                           (            CMP     %queueBlock.queueHead.OpieAddressType, nilOpieAddress
                                        JNZ     %Lb10
                                        MOV     %queueBlock.queueHead.OpieAddressLow, DX
                                        MOV     %queueBlock.queueHead.OpieAddressHigh, CX
                                        JMP     SHORT %Lb11
                                    %Lb10:
                                  %IF        (%EQS (%preserveESOrNot,preserveES))
                                  THEN (PUSH    ES)FI
                                        PUSH    CX
                                        PUSH    DX
                                        %EstablishIOPAccess     (generalMapRegister,%queueBlock.queueTail)


```

```
                                        POP     DX
                                        POP     CX
                                        MOV     ES:[DI].%linkOffset.OpieAddressLow, DX
                                        MOV     ES:[DI].%linkOffset.OpieAddressHigh, CX
                                %IF     (%EQS (%preserveESOrNot,preserveES))
                                     THEN (POP  ES)FI
                                   %Lbl1: MOV   %queueBlock.queueTail.OpieAddressLow, DX
                                        MOV     %queueBlock.queueTail.OpieAddressHigh, CX
                                        CMP     %queueBlock.queueNext.OpieAddressType, nilOpieAddress
                                        JNZ     %Lbl2
                                        MOV     %queueBlock.queueNext.OpieAddressLow, DX
                                        MOV     %queueBlock.queueNext.OpieAddressHigh, CX
                                   %Lbl2:
                                   )

;-- JumpIfMesaIOCBNextNil does a jump if the queueNext field is NIL.
;-- Note: for Mesa queues, "NIL" means the low-order word is 0.

%*DEFINE                (JumpIfMesaIOCBNextNil  (iocbNext, jumpLabel))
                           (            CMP     %iocbNext.OpieAddressLow, 0
                                        JZ      %jumpLabel
                                )

;-- JumpIfQueueNextNil does a jump if the queueNext field is NIL.
;-- Note: for IOP queues, "NIL" means the high-order byte is nilOpieAddress.

%*DEFINE                (JumpIfQueueNextNil     (queueBlock, jumpLabel))
                           (            CMP     %queueBlock.queueNext.OpieAddressType, nilOpieAddress
                                        JZ      %jumpLabel
                                )
```

```
;Copyright (C) 1984, 1985, 1987 by Xerox Corporation. All rights reserved.

;-- IORegion locations for the boot handler.
;-- stored as [Iris]<WMicro>Dove>RAMBDefs.asm
;
;-- last edited by:

;--     JPM                      11-May-87  9:05:32    ;Add new MP codes.
;--     RDH                      26-Jan-87 19:38:32    ;Add CSWordByteSize and fourKEEPromFormat.
;--     KEK                       7-May-86  9:46:14    ;update daisy constants.
;--     RDH                      24-Oct-85 16:56:16    ;Add device types defs.
;--     kek                       4-Sep-85 14:28:38    ;mp code definitions
;--     JPM                      18-Jul-85 12:27:26    ;Add bootDataBegins and uncomment BootJumpTable STRUC.
;--     JMM                       2-Jul-85 17:38:06    ;Increased normal load area.
;--     JPM                      22-May-85  9:19:21    ;Diagnostic boot changes.
;--     JMM                      22-Apr-85 12:06:08    ;Adjusted load area.
;--     JMM                       4-Apr-85 10:40:07    ;.
;--     JMM                      28-Jan-85 18:28:27    ;Add macro section.
;--     JMM                      23-Jan-85 15:04:01    ;Increase buffer size.
;--     BKI/JMM                   9-Dec-84 20:03:43    ;First release.

;NAME                   RAMBDefs
;-------------------------------------------------------------------------------
;

;-----------------------------------------------------

;Constants for STRUC definitions.
;--------------------------------

;Most things require a page (512 bytes) except Ethernet which has some overhead.
;(one page + packet overhead + simple data overhead)
maximumBootBufferSize   EQU     512+44+12
;-----------------------

BootBuffer              STRUC

nextBootBuffer          DW      ?       ;We have to abe able to do the following
bootDataStart           DW      ?       ;SI←[BX].bootDataStart and have [BX][SI]
bootDataEnd             DW      ?       ;point to data. Likewise for "bootDataEnd".
bootData                DB      maximumBootBufferSize DUP(?)

BootBuffer              ENDS

bootDataBegins          EQU     bootData-nextBootBuffer

;-----------------------

BootJumpTable           STRUC

startRAMOpie            DW      ?       ;An IOP Start block will result in an IOP
startRAMOpieCS          DW      ?       ;address being saved here for later entry.
iopEntry                DW      ?
iopEntryCS              DW      ?
processBootBlock        DW      ?

BootJumpTable           ENDS

;-----------------------

;BootTimeVariables      STRUC           ;located in ROMBDefs.asm
;
;device                 DW      ?       ;1-Disk, 2-Ethernet, 3-Floppy, 4-RS232C.
;mode                   DW      ?       ;0-Normal, 1-Fast boot.
;showUserInterface      DW      ?       ;0-yes, #0-no.
;bootRetryCount         DW      ?       ;initially 0, incremented per failure

;reincarnationFlag      DW      ?       ;For task reinitialization - 0 => tasks
;                                       ;to use local RAM otherwise main memory.

;emulatorID             DW      3 DUP(?) ;Initially 0, updated by boot executive.
;loaderID               DW      3 DUP(?) ;Initially 0, updated by boot executive.


;diagnosticsType        DW      ?       ;Initialized at boot-time, 0 => short
                                        ;diagnostics - user has set EEPROM to
                                        ;indicate desire for a diagnostics boot.
                                        ;#0=> long diagnostics boot - user has
                                        ;manually requested diagnostics at boot
                                        ;time!
;
;BootTimeVariables      ENDS

;-----------------------
;Boot types:
;-----------

normal                  EQU     0
diagnostic              EQU     1

;-----------------------
;Device types:
;-----------

disk                    EQU     1

RAMBDefs.asm        11-May-87  9:05:35 PDT                                  1
```

```
floppy                  EQU     2
ethernet                EQU     3
experimental            EQU     4
rs232C                  EQU     4

;-------------------------------------------------------------------------------
;Booting error numbers:
;----------------------
;
bootingError    EQU     -1
unKnownBootBlock EQU    0303H
noRAMStartAddress EQU   0404H
;-----------------------
;Other constants:

EOF                     EQU     0FFH
highByteMask            EQU     0FF00H
lowByteMask             EQU     0FFH
oneBit                  EQU     1
onePage                 EQU     512     ;In bytes.
diagnosticsLoadAreaSize EQU     2000H
normalLoadAreaSize      EQU     2000H   ;Too big!.
loaderTimeout           EQU     04H


;Control Store constants:

fourKEEPromFormat       EQU     10H
CSWordByteSize          EQU     6

;Daybreak:

daybreakBankRegister    EQU     0E000H
daybreakCSPortMask      EQU     08000H
nextCSByte              EQU     01000H

;Daisy:

BytesInShiftRegister            EQU 6
BytesPerMicroInstruction        EQU 6
;Data from control store is port 080H bit 13 i.e.
SiriusPort      EQU     0B0H    ;The port for controlling the Sirius chip
                                ;bit 7: RAMWrEnable
                                ;bit 6: CSShift'
                                ;bit 5: CSDOe
                                ;bit 4: shift data
                                ;bit 3: ShiftClk
                                ;bit 2: Halt'
                                ;bit 1: ResetSChip'
                                ;bit 0: IOPIntSChip
Ram             EQU  8000H
NCSShift        EQU  4000H
CSShift         EQU  0
CSDOe           EQU  2000H
SData           EQU  1000H
SClock          EQU  800H
NHalt           EQU  200H
IOPInt          EQU  100H
CSDataBit       EQU  2000H


;Daisy & Daybreak:
;-------------------------------------------------------------------------------
;
;
;-------------------------------------------------------------------------------
;Convenience macros:
;-------------------
;
%*Define (MultiplyByTwo (register))
        (ROL    %register, 1)

;-------------------------------------------------------------------------------
;Maintenance Panel Codes:
;------------------------
;
mpStartBooting  EQU     0100D
mpFetchInitial  EQU     0149D
mpRunInitial    EQU     0150D
mpInitialError  EQU     0151D
mpFetchMesaDove EQU     0199D
mpRunMesaDove   EQU     0200D
mpMesaDoveError EQU     0201D
mpSDDReadError  EQU     0209D
mpSDDSealError  EQU     0211D
mpSDDVrsnError  EQU     0212D
mpSDDCksmError  EQU     0213D
mpRunGerm       EQU     0501D
mpFloppyCleaning        EQU     0077D
```

```
$MOD186
$PAGELENGTH (72)
$PAGEWIDTH  (136)

;Copyright (C) 1984, 1985, 1986 by Xerox Corporation. All rights reserved.

;-- stored as RAMBoot.asm
;-- created on  19-Jul-84 13:20:18

;-- last edited by:

;--     RDH             19-Jan-87 10:12:01      ;Remove CP specific stuff to iop init block.
        ;Added EXTRNs: WriteCSProc CPStartOrInitProc IncSIFarProc.
        ;Added init of IncSIFarProc IncSIFarProc+2. Added IncSIFar.
        ;Changed call to DumpCSBlock in Checksum to 2 calls to IncrementSI
        ;At ReadOffset+2 change CALL LoadCXFromBootBuf to CALL LoadAXFromBootBuf; MOV CX, AX
        ;Commented out all of LoadCXFromBootBuf.
;--     JPM             29-Jul-86 15:57:30      ;Activate InitializeCP code (uncomment code, add calls to Start/StopCP,
enable/clear/disable interrupt).
;--     RDH             22-Jan-86 12:06:22      ;Adjust code for cs config of "any-type", and fix buglet in InitializeIOP.
;--     RDH             17-Jan-86 12:06:08      ;Create and use procs
        ; LoadCXFromBootBuf and LoadAXFromBootBuf to save space.
        ; Comment out useless code for doing Daybreak cp inits and starts.
        ; Alter IncrementSI, ProcessEmulatorFile, and ProcessMultipleDBsFile
        ; to use new proc ReloadBuffer created from part of ProcessEmulatorFile.
;--     RDH             14-Jan-86 15:25:14      ;Add code for handling fb's for various control store configurations, and add labels near
beginning of all proc's for debugging ease.
;--     RDH             13-Jan-86 13:39:34      ;Change AX to AL in setting control store bank register for 8K.
;--     RDH              9-Jan-86 11:45:38      ;Add stuff for handling fb files for bank configs not on the machine.
;--     JAC              6-Jan-86 10:00:59      ;comment out rest of Daisy stuff because the addns to initial kill ether booting
diagnostics
;--     JAC             10-Dec-85 11:04:03      ;add checking in WriteIOPMemory, StartIOP and InitializeIOP for main memory
;--     RDH             28-Oct-85 12:33:27      ;Add code to fix etherbooting of diagnostics by sending an error packet when EOF block is
found and it is an ethernet diagnostic boot.  Commented out some Daisy code to make room.
;--     JPM             20-Sep-85  9:29:24      ;Move boot buffer allocation to device-specific code.
;--     bki              6-Sep-85 11:52:40      ;change some JMP's to SHORT's.
;--     kek              4-Sep-85 14:58:06      ;mp codes
;--     JPM              6-Aug-85 16:25:43      ;Remove extra POP in ProcessLoaderFile.
;--     JPM              1-Aug-85 15:04:33      ;Fix bug in OutOfLoadSpace.
;--     JPM             18-Jul-85 15:17:48      ;Opie redesign conversion.
;--     JMM             20-Jun-85 14:55:53      ;New IOPLRAM.asm upgrade.
;--     JMM             17-Jun-85 12:25:18      ;Deleted doneWithDBFile and doneWithGermFile and restored previous protocol.
;--     JPM              6-Jun-85  8:19:41      ;Switch to word offset in multiple db files; share buffer loading code.
;--     JPM              5-Jun-85 15:48:02      ;Fix byte-swap bug in MultipleDBs; use doneWithDBFile and doneWithGermFile for control
flow.
;--     JPM             30-May-85 15:01:34      ;Fixed bug in StartIOP.
;--     JPM             22-May-85 10:11:58      ;Revised for overlay booting.
;--     JMM             12-Apr-85 18:44:28      ;Debugging additions.
;--     JMM             31-Jan-85 11:43:32      ;Diagnostics handling.
;--     JMM             15-Jan-85 16:41:11      ;First release.

NAME                    RAMBoot
;-------------------------------------------------------------------------------
;
$NOLIST
$INCLUDE                (EthHdFce.asm)
$INCLUDE                (HardDefs.asm)
$INCLUDE                (IOPDefs.asm)
$INCLUDE                (IOPMacro.asm)
$INCLUDE                (RAMBDefs.asm)
$INCLUDE                (EthBDefs.asm)
$INCLUDE                (Handlers.asm)  ;to resolve handler IDs
;$INCLUDE               (CSBankDf.asm)
;$INCLUDE               (RAMEEP.asm)
$LIST

;-------------------------------------------------------------------------------
%*DEFINE        (ByteSwap       (wordToSwap))
                (%wordToSwap SHL 8 OR %wordToSwap SHR 8)


%*DEFINE(Handler(name,id,initProcAction))       (
PUBLIC                  %name%(HandlerID)
%name%(HandlerID)       EQU     %id
)

                        %HandlersLinked

EXTRN                   mesaProcessorInterrupt :ABS

;---------------
;*****************************************************************
;---------------

IOPELocalRAM            SEGMENT AT 0


;from IOPLRAM.asm:
;------------------

EXTRN                   bootOverlayReentry: WORD
EXTRN                   bootType: BYTE
EXTRN                   device: WORD
EXTRN                   skipUserInterface: BYTE
EXTRN                   bootOverlayRequest: WORD
EXTRN                   startOfBootBufferSpace: WORD
EXTRN                   endOfBootBufferSpace: WORD

RamBoot.asm         21-Jan-87 15:32:18 PST                                                                                    1
```

```
EXTRN                   resetRegData: WORD

IOPELocalRAM            ENDS

;---------------
;*****************************************************************
;---------------

;Imported Variables:


;from IORMaint.asm:
;------------------
MaintPanelIOR           SEGMENT COMMON

EXTRN                   maintPanelCode: WORD
EXTRN                   maintPanelChanged: Condition

MaintPanelIOR           ENDS


;from IORRAMBt.asm:
;------------------

BootStrapIOR            SEGMENT COMMON

EXTRN                   bootTask: TaskContextBlock
EXTRN                   bootBufferPtr :WORD
EXTRN                   bootBufferEmpty  :Condition, bootBufferFull :Condition
EXTRN                   finishedLoaderFileFetch :Condition, getBootFile :Condition
EXTRN                   bootFileChoice :WORD
EXTRN                   currentBootBuffer :WORD
;EXTRN                  currentBlockSequence :WORD
EXTRN                   jumpTable :BootJumpTable
EXTRN                   startOfBootBufferPool :WORD
EXTRN                   loaderVirtualMemoryLocation :DWORD
EXTRN                   BootDeviceIORSpace :RamEtherBootContext
EXTRN                   bootMemoryPtr: WORD
EXTRN                   WriteCSProc: WORD
EXTRN                   CPStartOrInitProc: WORD
EXTRN                   IncSIFarProc: WORD

BootStrapIOR            ENDS

;-------------------------------------------------------------------------------

;*******************************************************************************


IOPEInRAM               SEGMENT         PUBLIC

                        ASSUME  CS:IOPEInRAM
                        ASSUME  DS:BootStrapIOR

;
;-------------------------------------------------------------------------------

PUBLIC                  EndOfInitial
PUBLIC                  BootTaskInit
PUBLIC                  DisplayMPCode
;Needed for etherbooting diagnostics.  This will cause a link time warning,
;Unresolved External when building diskboot.lnk and flpyboot.lnk.
EXTRN                   TransmitFrame:          NEAR

bootBufferSize          DW      SIZE BootBuffer
;-------------------------------------------------------------------------------

;Dove Boot Blocks:
;
;
;BootFile => BootBlock | BootBlock . BootFile
;BootBlock => Type . Address . DataLength . Data
;Type => CP . Command | IOP . Command | Checksum | MultipleFiles | EOF
;Command => Operation . MachineType . MemorySection . BankConfiguration . Bit . Byte
;Operation => Initialize | Start | writeData
;CP => 1
;IOP => 0
;Initialize => 00
;Start => 01
;writeData => 10
;MachineType => Daisy | DayBreak
;Daisy => 0
;DayBreak => 1
;MemorySection => LocalRam | MainMemory
;LocalRam => 0
;MainMemory => 1
;BankConfiguration => fourK | eightK | twelveK | sixteenK
;fourK => 00
;eightK => 01
;twelveK => 10
;sixteenK => 11
;Checksum => 1110 . 0000 . 0000 . 0000
;MultipleFiles => 1111 . 1111 . DBFileCount . FileOffset
;FileOffset => FileOffset . FileOffset | FileOffset
;Address => DoveCPAddress | DoveIOPAddress

RamBoot.asm         21-Jan-87 15:32:18 PST
```

```
;DoveCPAddress => CPAddress | BankValue . CPAddress | CPAddress . CPNextAddress
;BankValue => DaybreakBankValue | DaisyBankValue
;DaybreakBankValue => DaybreakBank0 | DaybreakBank1 | DaybreakBank2 | DaybreakBank3
;DaisyBankValue => DaisyBank0 | DaisyBank1 | DaisyBank2 | DaisyBank3
;CPNextAddress => CPAddress
;CPAddress => m = {0,1,2, . . .2↑16-1}
;DoveIOPAddress => AddressSegment . AddressOffset
;DataLength => j = {0,1,2, . . .2↑16-1}
;DBFileCount => k = {1,2, . . .254}
;FileOffset => i = {0,1,2, . . .2↑16-1}
;DaybreakBank0 => NullByte . 0000 . 0000
;DaybreakBank1 => NullByte . 0011 . 0000
;DaybreakBank2 => NullByte . 0101 . 0000
;DaybreakBank3 => NullByte . 1001 . 0000
;DaisyBank0 => 0001 . 0000 . NullByte
;DaisyBank1 => 0010 . 0000 . NullByte
;DaisyBank2 => 0100 . 0000 . NullByte
;DaisyBank3 => 1000 . 0000 . NullByte
;Data => CPData | IOPData
;CPData => (Word . Word . Word)*
;IOPData => Byte*
;AddressSegment => n = {0,1,2, . . .2↑16-1}
;AddressOffset => n = {0,1,2, . . .2↑16-1}
;EOF => 1111 . 1111 . 1111 . 1111
;Word => Byte . Byte
;Byte => Nibble . Nibble
;NullByte => 0000 . 0000
;Nibble => (0+1)(0+1)(0+1)(0+1)
;Bit => (0+1)
;
;
;
;       Hex     Binary
;
;       0x      0000.xxxx       IOP Daisy Initialize
;       1x      0001.xxxx       IOP Daybreak Initialize
;       2x      0010.xxxx       IOP Daisy Start
;       3x      0011.xxxx       IOP Daybreak Start
;       4x      0100.xxxx       IOP Daisy Write
;       5x      0101.xxxx       IOP Daybreak Write
;       6x      0110.xxxx       IOP Daisy Unknown
;       7x      0111.xxxx       IOP Daybreak Unknown
;       8x      1000.xxxx       CP  Daisy Initialize
;       9x      1001.xxxx       CP  Daybreak Initialize
;       Ax      1010.xxxx       CP  Daisy Start
;       Bx      1011.xxxx       CP  Daybreak Start
;       Cx      1100.xxxx       CP  Daisy Write
;       Dx      1101.xxxx       CP  Daybreak Write
;       Ex      1110.xxxx       Checksum
;       Fx      1111.xxxx       EOF or MultipleDB's
;
;--
;-- Boot task initialization:
;--------------  This task communicates with the device specific task to load in
;--------------  the Mesa emulator file and RAM based opie and to also prepare the:
;--------------  virtual memory map, Germ variables and load the Germ.
;--------------|                                                                  |
;--------------|          startOfBootBufferSpace = This is supplied by Opie and  |
;--------------|          points to the beginning of a huge buffer that lies     |
;--------------|          right after STACK space but    right before local RAM  |
;--------------|          IORegion. *Initial updates it so that it points just   |
;--------------|          beyond *Initial. Boot then adjusts it so that there is|
;--------------|          loadable space between *Initial and the beginning of   |
;--------------|          buffer. It is up to Boot task to divide this buffer    |
;--------------|          into a pool of buffers to be used in conjuction        |
;--------------|          with RAM*Bt.asm to load files. Any Opie files must be |
;--------------|          loaded last since we start loading RAM based Opie in   |
;--------------|          low RAM going upwards.                                 |
;--------------|                                                                 |
;--------------|                                                                 |
;--------------|                                                                 |
;--------------|--------                                                         |
;--------------|                                                                 |
;--------------|                                                                 |
;--------------|                                                                 |
;--------------|                                                                 |
;-------------------------------------------------------------------------------


BootTaskInit            PROC    NEAR

BeginBootTask:          %'ThisTaskServices (mesaProcessorInterrupt,BadMesaInterrupt)
                        ;This will be done in the CP specific code that is loaded from
                        ; the last IOP Init block before the cp oriented blocks.
                                ;Recall that any boot block of the type CP-
                                ;Initialize causes the CP to be started which
                                ;in turn is expected to issue an interrupt when
                                ;it is done initializing the CP. This interrupt
                                ;servicing will be taken over by Mesa processor
                                ;task when RAM based opie starts. In fact on
                                ;loading CP code, CP is not started even though
                                ;its start address is progrmmed into it, rather
                                ;it is left to Mesa processor task to start CP
                                ;since by then all of Opie should be ready.

                        MOV     AX, IOPELocalRAM        ;We will occassionally refer to variables
```

```
                        MOV     ES, AX                  ;in IOP low memory (which will cover all of
                        ASSUME  ES:IOPELocalRAM         ;boot buffer and load space).
                        MOV     AX, startOfBootBufferPool ;Pick up buffer(s) supplied
                        MOV     currentBootBuffer, AX   ;by device-specific code.
                        MOV     AX, startOfBootBufferSpace ;Also get boot area address,
                        ADD     AX, LowNibbleMask       ;adjust it so that it is
                        AND     AX, NOT(LowNibbleMask)  ;at a "paragraph" boundary and then
                                                        ;save it for loading IOP code
                        MOV     bootBufferPtr, AX       ;into later.
                        MOV     jumpTable.startRAMOpie, OFFSET NoRAMOpieEntryPoint
                        MOV     jumpTable.startRAMOpieCS, CS    ;Just in case we never get a
                                                                ;RAMOpie entry IOPStart block.
                        MOV     IncSIFarProc, OFFSET IncSIFar   ;Initialize point for CP
                        MOV     IncSIFarProc+2, CS              ; specific code to call
                                                        ; to get IncrementSI.


                        %NotifyCondition (OFFSET getBootFile)

;display mp code for fetching MesaDove
                        MOV     AX, mpFetchMesaDove
                        CALL    DisplayMPCode


                        ;IN     AX, machineIDPort       ;The Mesa processor should be halted
                        ;AND    AX, machineIDMask       ;before we do anything to it.
                        ;CMP    AX, Daisy               ;before we do anything to it.
                        ;JNE    DaybreakCPConditioning

;DaisyCPConditioning:   CALL    DaisyCPHalt
                        ;JMP    SHORT ProcessEmulatorFile


;DaybreakCPConditioning:        CALL    DaybreakCPHalt
;                       CLI                                     ;to protect reset register
;                       MOV     AX, resetRegData
;                       AND     AX, NOT resetMesaProcessor
;                       OUT     WriteResetReg, AX
;                       CALL    DaybreakCPStart
;                       OR      resetRegData, resetMesaProcessor
;                       MOV     AX, resetRegData
;                       OUT     WriteResetReg, AX
;                       CALL    DaybreakCPHalt
;                       STI                                     ;done with reset register

                        ;Put the number of extra control store banks in the
                        ; variable, csBankConfiguration.
;                       CALL    GetCSBankConfig


                        JMP     SHORT ProcessEmulatorFile

BootTaskInit            ENDP


;--

;--
;-- Boot file processing:
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                                       |
;--------------|       currentBootBuffer = pointer to current boot buffer.             |
;--------------|       At this point, boot block type is as follows:                   |
;--------------|               'zxxx' where 'x' is a don't care bit and                |
;--------------|               'z' is the bit we are testing for. Except               |
;--------------|               initially we do test all of 'zxxx' for EOF              |
;--------------|               or MultipleDBfile.                                      |
;--------------|                                                                       |
;--------------|                                                                       |
;--------------|       BX = currentBootBuffer                                          |
;--------------|       SI↑ = BootBlock.Type                                            |
;--------------|                                                                       |
;--------------|--------                                                               |
;--------------|                                                                       |
;--------------|       AH = xxx0 - remaining bits to be tested.                        |
;--------------|       BX = pointer to the buffer to process.                          |
;--------------|       SI = BootBlock.Type                                             |
;--------------|                                                                       |
;-------------------------------------------------------------------------------------


ProcessEmulatorFile     PROC    NEAR

                        CALL    WaitForFullBuffer

StartSequence:          CALL    LoadAXFromBootBufLateEntry      ;Get the boot block type indicator
CouldBeCPorIOPType:     SHL     AH, oneBit
                        JC      ItsaCPBlockType
                        MOV     jumpTable.processBootBlock, OFFSET ProcessIOPBlock
                        JMP      SHORT BootBlockIdentified
ItsaCPBlockType:        MOV     jumpTable.processBootBlock, OFFSET ProcessCPBlock

BootBlockIdentified:    CALL    WORD PTR jumpTable.processBootBlock
```

```
                        CMP     SI, ES: [BX].bootDataEnd
                        JLE     StartSequence
                        CALL    PrepareToFetchBuffer
                        JMP     StartSequence

ProcessEmulatorFile     ENDP

;--

;--
;-- ReloadBuffer:
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|--------                                                      |
;--------------|                                                              |
;--------------|                                                              |
;-------------------------------------------------------------------------------

ReloadBuffer    PROC    NEAR


WaitForFullBuffer:      PUSHA
                        %WaitForCondition (OFFSET bootBufferFull,noTimeout)
                        POPA

FetchTheNextBuffer:     MOV     BX, currentBootBuffer           ;Every time we get a new buffer
                        MOV     SI, ES: [BX].bootDataStart      ;we have to first check that it
                        CMP     SI, ES: [BX].bootDataEnd        ;is really full before we start
                        JE      WaitForFullBuffer               ;processing it.
                        RET

PrepareToFetchBuffer:   MOV     BX, ES: [BX].nextBootBuffer     ;Update the pointer to the
                        MOV     currentBootBuffer, BX           ;next buffer to process, then do
                        PUSH    BX                              ;the obligatory rest.
                        %NotifyCondition (OFFSET bootBufferEmpty)
                        POP     BX
                        JMP     FetchTheNextBuffer


ReloadBuffer    ENDP


;--

;--
;-- IOP Boot block early interpretation:
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                              |
;--------------|       AH = zzx0 - remaining bits to be tested.               |
;--------------|       BX = pointer to the buffer to process.                 |
;--------------|       SI↑ = BootBlock.Type                                   |
;--------------|                                                              |
;--------------|--------                                                      |
;--------------|                                                              |
;--------------|       BX = pointer to the buffer to process.                 |
;--------------|       SI↑ = BootBlock.Type                                   |
;--------------|                                                              |
;-------------------------------------------------------------------------------


ProcessIOPBlock         PROC    NEAR

DoIOPBlock:             SHL     AH, oneBit              ;This could be a Start, Initialize
                        JC      IOPWriteORUnknown       ;Checksum or Write block for IOP.

ItsStartORInitialize:   SHL     AH, oneBit              ;Sure, but which one is it?
                        JC      ItsStart                ;Go process an IOP Start block!
                        JMP     InitializeIOP           ;Go process an IOP Initialize block!
ItsStart:               JMP     StartIOP

IOPWriteORUnknown:      SHL     AH, oneBit              ;If it is a checksum block go to
                        JC      ItsUnknown              ;the checksum routine otherwise
                        JMP     SHORT WriteIOPMemory            ;go write IOP memory.
ItsUnknown:             JMP     UnknownBlock

;Notice that we never bothered checking BootBlock.Type(Daisy|Daybreak) because for the
;IOP, the two worlds are identical!

ProcessIOPBlock         ENDP


;--
;--
;--
;-- CP Boot block early interpretation:
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                              |
```

```
;--------------|        AH = zzx0 - remaining bits to be tested.                |
;--------------|        BX = pointer to the buffer to process.                  |
;--------------|        SI↑ = BootBlock.Type                                    |
;--------------|                                                                |
;--------------|--------                                                        |
;--------------|                                                                |
;--------------|                                                                |
;--------------|        AH = x000 - remaining bit to be tested.                 |
;--------------|        BX = pointer to the buffer to process.                  |
;--------------|        SI↑ = BootBlock.Type                                    |
;--------------|                                                                |
;--------------|                                                                |
;--------------|                                                                |
;-------------------------------------------------------------------------------

ProcessCPBlock          PROC    NEAR


DoCPBlock:              ROL     AH, oneBit              ;Save this bit for use
                                                        ; later in CheckBankConfig.
                                                        ;This could be a Start, Initialize
                        JC      CPWriteORChecksum       ;or Write block for CP.

StartORInitialize:
                        CALL DWORD PTR CPStartOrInitProc
                        RET     ;Return into ProcessEmulatorFile
                        ;Code for starting the CP should have been loaded and
                        ; started from the last IOP init block which will have
                        ; initialized CPStartORInitialize.



                        ;SHL     AH, oneBit             ;Sure, but which one is it?
                        ;JC      ItsStartCP             ;Go process a CP Start block!
                        ;JMP     InitializeCP           ;Go process a CP Initialize block!
;ItsStartCP:            JMP     StartCP

CPWriteORChecksum:      SHL     AH, oneBit              ;If it is an unknown block go to
                        JC      ItsChecksum             ;the unknown block routine otherwise
                        CALL    DWORD PTR WriteCSProc
                        RET             ;Return into ProcessEmulatorFile
                        ;Code for loading the CP should have been loaded and
                        ; started from the last IOP init block which will have
                        ; initialized CPStartORInitialize.

                        ;JMP     WriteControlStore      ;go write CP control store.
ItsChecksum:            JMP     Checksum


ProcessCPBlock          ENDP


;--
;--
;-- Boot section #
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                                |
;--------------|        BX = pointer to the buffer to process.                  |
;--------------|        SI↑ = BootBlock.Type                                    |
;--------------|                                                                |
;--------------|--------                                                        |
;--------------|                                                                |
;--------------|                                                                |
;--------------|        BX = pointer to the buffer to process.                  |
;--------------|        SI↑ = BootBlock.Type.                                   |
;--------------|                                                                |
;--------------|                                                                |
;--------------|                                                                |
;-------------------------------------------------------------------------------


WriteIOPMemory          PROC    NEAR
WriteIOPMem:            SHL     AX, oneBit              ;Ignore Daisy or Daybreak bit
                        SHL     AX, oneBit              ;Should this be stored in main memory?
                        JNC     WriteInLRAM
WriteInMemory:          MOV     DI, bootMemoryPtr
                        SHL     DI, Nibble
                        JMP     ReadOffset
WriteInLRAM:            MOV     DI, bootBufferPtr
ReadOffset:             CALL    LoadAXFromBootBuf       ;The next word in the
                        ADD     DI, AX                  ; boot block is the
                                                        ; address to be written to.
                                                        ;Boot clients should offset their
                                                        ;code from zero! jmm:84-12-11:later
                        CALL    LoadAXFromBootBuf
                        MOV     CX, AX
                        ;And now we want to get the data
                        ;count to use in emptying this
                        ;boot block, then point to the data itself.
                        CALL    IncrementSI

UnloadBootBlock:        MOV     AL, ES: [BX][SI]        ;As we empty the block,
                        MOV     BYTE PTR ES: [DI], AL   ;we have to keep testing to make sure
```

```
                        CALL    IncrementSI             ;that we have not reached the end of the
                        INC     DI                      ;boot buffer.
                        LOOP    UnloadBootBlock
                        RET


WriteIOPMemory          ENDP


;--
;--
;--
;-- Boot section #
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                              |
;--------------|       BX = pointer to the buffer to process.                 |
;--------------|       SI↑ = BootBlock.Type                                   |
;--------------|                                                              |
;--------------|--------                                                      |
;--------------|                                                              |
;--------------|       BX = pointer to the buffer to process.                 |
;--------------|       SI↑ = BootBlock.Type                                   |
;--------------|                                                              |
;--------------|                                                              |
;-------------------------------------------------------------------------------


InitializeIOP           PROC    NEAR

InitIOP:                SHL     AX, oneBit              ;Ignore Daisy or Daybreak bit
                        SHL     AX, oneBit              ;should this be started in main memory?
                        JNC     LRAMInit
MemoryInit:             MOV     AX, bootMemoryPtr       ;Set up CS for main memory
                                                        ;This had better not happen until
                                                        ;AllocSgs has run since it is responsible
                                                        ;for setting up bootMemoryPtr
                        JMP     JumpTableInit
LRAMInit:               MOV     AX, bootBufferPtr       ;Set up CS for boot code in local RAM
                        SHR     AX, Nibble              ;to point to start of boot buffer
JumpTableInit:          MOV     jumpTable.iopEntryCS, AX ;and store into jump table.

                        CALL    LoadAXFromBootBuf
                        ;The next word in the Boot
                        ;block is the address to go to
                        ;and which should return to us
                        ;upon completion.
                        MOV     jumpTable.iopEntry, AX  ;Notice that we are giving
                        PUSHA                           ;boot clients the ability
                        PUSH    ES                      ;to specify only the offset
EnterInitIOP:           CALL    DWORD PTR jumpTable.iopEntry ;to their entry point.
                        POP     ES
                        POPA
                        CALL    IncrementSI
                        RET

InitializeIOP           ENDP


;--
;--
;-- Boot section #
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                              |
;--------------|       BX = pointer to the buffer to process.                 |
;--------------|       SI↑ = BootBlock.Type                                   |
;--------------|                                                              |
;--------------|--------                                                      |
;--------------|                                                              |
;--------------|       BX = pointer to the buffer to process.                 |
;--------------|       SI↑ = BootBlock.Type                                   |
;--------------|                                                              |
;-------------------------------------------------------------------------------


StartIOP                PROC    NEAR

StartTheIOP:            SHL     AX, oneBit              ;Ignore Daisy or Daybreak bit
                        SHL     AX, oneBit              ;should this be started in main memory?
                        JNC     LRAMStart
MemoryStart:            MOV     AX, bootMemoryPtr       ;Set up CS for main memory
                        JMP     JumpTableStore
LRAMStart:              MOV     AX, bootBufferPtr       ;Set up CS for boot code in local RAM
                        SHR     AX, Nibble              ;to point to start of boot buffer
JumpTableStore:         MOV     jumpTable.startRAMOpieCS, AX ;and store into jump table.

                        CALL    LoadAXFromBootBuf
                        ; The address we get here and save
                        ;is the entry point of RAMOpie.
                        ;Prior to this we had saved the
                        ;address of a "No Opie entry point"
                        MOV     jumpTable.startRAMOpie, AX ; error routine. Again notice
```

```
                                                ;that we are giving boot clients
                                                ;the ability to specify only
                        CALL    IncrementSI     ;the offset to their entry point.
                        RET

StartIOP                ENDP


;--
;--
;-- Boot section #
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                              |
;--------------|       AH = z000 - remaining bit to be tested.                |
;--------------|       BX = pointer to the buffer to process.                 |
;--------------|       SI↑ = BootBlock.Type                                   |
;--------------|                                                              |
;--------------|--------                                                      |
;--------------|                                                              |
;--------------|                                                              |
;--------------|       BX = pointer to the buffer to process.                 |
;--------------|       SI↑ = BootBlock.Type.                                  |
;--------------|                                                              |
;-------------------------------------------------------------------------------

;ALL OF WriteControlStore IS COMMENTED OUT.

;WriteControlStore      PROC    NEAR

;WriteCntlStore:                CALL    CheckBankConfig         ;Make sure its the right fb.
                        ;CALL   IncrementSI             ;We want to point to the
                        ;SHL    AX, oneBit              ;address part of the buffer.
                        ;JC     WriteDaybreakControlStore ;But first see if the block
                                                        ;is for Daisy or Daybreak.

;WriteDaisyControlStore:
                        ;IN     AX, machineIDPort       ;Find out if this is a Daisy
                        ;AND    AX, machineIDMask       ;or a Daybreak and act accordingly.
                        ;CMP    AX, Daisy               ;or a Daybreak and act accordingly.
                        ;JE     WriteDaisysCS
                        ;JMP    DumpCSBlock
;WriteDaisysCS:
                        ;MOV    AH, ES: [BX][SI]        ;Get the low byte of the
                        ;CALL   IncrementSI             ;address, fetch the high byte,
                        ;MOV    AL, ES: [BX][SI]        ;then save in BP. Recall shift
                        ;MOV    BP, AX                  ;order for the Sirius chip access.
                        ;CALL   IncrementSI             ;After which, we want to fetch
                        ;MOV    CH, ES: [BX][SI]        ;the CS block length and we will
                        ;CALL   IncrementSI             ;use the BP register as a counter
                        ;MOV    CL, ES: [BX][SI]        ;since we are out of registers.
;LetsGetACSWord:                ;PUSH   CX      ;jmm-84-10-24!!!!!!!stack trouble!!!!!!
                        ;MOV    CX, CSsizeWordCount
;LoopForCSWord:         ;CALL   IncrementSI             ;Get the control store word to
                        ;MOV    AH, ES: [BX][SI]        ;be written and pop it onto stack
                        ;CALL   IncrementSI             ;so as to restore it to its correct
                        ;MOV    AL, ES: [BX][SI]        ;order when writing it later.
                        ;PUSH   AX
                        ;LOOP   LoopForCSWord

                        ;MOV    CX, CSsizeWordCount
                        ;MOV    AL, loadShiftRegister   ;Get CS ready to write a word!
                        ;OUT    CPoutputPort, AL

;WriteDaisyCSWord:      ;POP    AX                      ;Send the 48-bit control store
                        ;PUSH   CX                      ;word to the shift register of
                        ;MOV    CX, dataWordBitCount    ;Sirius chip.
                        ;CALL   ShiftDataToCS
                        ;POP    CX
                        ;LOOP   WriteDaisyCSWord

                        ;MOV    AX, BP                  ;Finally get the saved address
                        ;CALL   ShiftDataToCS           ;of the word to write.
                        ;MOV    AL, (notLoadShiftRegister OR writeCS OR interruptCP)
                        ;OUT    CPoutputPort, AL
                        ;MOV    AL, grabData
                        ;OUT    CPoutputPort, AL
                        ;MOV    AL, (loadShiftRegister OR notWriteCS)
                        ;OUT    CPoutputPort, AL
                        ;MOV    AL, notGrabData
                        ;OUT    CPoutputPort, AL
                        ;INC    BP                      ;Update to next address

                        ;POP    CX
                        ;LOOP   LetsGetACSWord

                        ;CALL   IncrementSI

                        ;RET

;WriteDaybreakControlStore:
                        ;IN     AX, machineIDPort       ;Find out if this is a Daisy
                        ;AND    AX, machineIDMask       ;or a Daybreak machine and
                        ;CMP    AX, Daybreak            ;dump block if it is for
                        ;JE     WriteDaybreakCS
                        ;JMP    DumpCSBlock             ;Daisy.
```

```
;WriteDaybreakCS:       CALL    LoadAXFromBootBufLateEntry
                        ;We first want to set the CS
                        ;bank register then get the CS
                        ;address that this block is to
                        ;be written to.

                        ;DayBreak's control store is an I/O port.
                        ;Put the cs address into DX.
                        ;CALL    LoadCXFromBootBuf
                        ;MOV     DX, CX
                        ;We want to point to the count
                        ;of CS words in this block,
                        ;from which we will fetch the
                        ;CS block size counter and
                        ;save it in DI - we are again out of registers!
                        ;CALL    LoadCXFromBootBuf
                        ;MOV     DI, CX                  ;
                        ;MOV     BP, DX

                        ;MOV     DX, daybreakBankRegister ;First let us set up the CS bank
                        ;OUT     DX, AL                  ;register.

;WriteDaybreakCSBlock:  MOV     DX, BP                  ;Also need to save initial CS adrs.
                        ;OR      DX, daybreakCSPortMask  ;
                        ;MOV     CX, CSWordByteSize      ;A single control store word
;WriteDaybreakCSWord:   CALL    IncrementSI             ;is made up of six bytes. The
                        ;MOV     AL, ES: [BX][SI]        ;layout of the control store with
                        ;OUT     DX, AL                  ;respect to the 4K ports is as
                        ;ADD     DX, nextCSByte          ;follows: 8###H - D###H where
                        ;LOOP    WriteDaybreakCSWord     ;8###H has the MSB and D###H has
                        ;INC     BP                      ;the LSB. The CS is loaded in six
                        ;DEC     DI                      ;byte data streams. Update CS word count.
                        ;JNZ     WriteDaybreakCSBlock    ;Loop while not done with CS block.
                        ;CALL    IncrementSI

                        ;RET


;WriteControlStore      ENDP


;--

;--
;--
;--
;-- Boot section #
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                                  |
;--------------|        AH = z000 - remaining bits to be tested.                  |
;--------------|        AL = xxxx - remaining bits to be dumped in this case.     |
;--------------|        BX = pointer to the buffer to process.                    |
;--------------|        SI↑ = BootBlock.Type                                      |
;--------------|                                                                  |
;--------------|--------                                                          |
;--------------|                                                                  |
;--------------|                                                                  |
;--------------|        BX = pointer to the buffer to process.                    |
;--------------|        SI↑ = BootBlock.Type.                                     |
;--------------|                                                                  |
;--------------|                                                                  |
;-------------------------------------------------------------------------------

;ALL OF InitializeCP IS COMMENTED OUT.

;InitializeCP           PROC    NEAR

;InitCP:                        CALL    CheckBankConfig         ;Make sure its the right fb.
                        ;CALL   IncrementSI             ;We want to point to the
                                                        ;address part of the buffer.
                        ;SHL    AX, oneBit              ;But first see if the block
                        ;JC     InitializeDaybreakCP    ;is Daisy or Daybreak.

;InitializeDaisyCP:
                        ;IN     AX, machineIDPort       ;Find out if this is a Daisy
                        ;AND    AX, machineIDMask       ;or a Daybreak and act accordingly.
                        ;CMP    AX, Daisy
                        ;JE     WriteDaisyCPAddress
                        ;JMP    DumpCSBlock
;WriteDaisyCPAddress:                                    ;Load CP address for Daisy before
                                                        ;starting the CP.

                        ;JMP    SHORT StartDaisyCPANDWait

;InitializeDaybreakCP:
                        ;MOV    CX, 5                   ;5 bytes to end of boot block.
                        ;CALL   DumpCSBlock             ;skip over address (not used)
                        ;IN     AX, machineIDPort       ;Find out if this is a Daisy
                        ;AND    AX, machineIDMask       ;or a Daybreak and act accordingly.
                        ;CMP    AX, Daybreak
                        ;JNE    InitializeDaybreakCPRet

;StartDaybreakCPANDWait:
```

```
                        ;PUSHA
                        ;%'Enable(mesaProcessorInterrupt)       ;turn on the interrupt, but
                        ;%'DisableInterruptsTillNextWait        ; don't allow it till we're ready
                        ;CALL   DaybreakCPStart         ;let CP run
                        ;%'WaitForInterrupt     (1)     ;init routines should take < 1 sec
                        ;IN     AX, ClrMesaIntr         ;clear Mesa interrupt
                        ;%'Disable(mesaProcessorInterrupt); and turn it off
                        ;%'WaitForSystem                        ;drop to system level
                        ;POPA
                        ;CALL   DaybreakCPHalt          ;stop the CP (so can load more code)
;InitializeDaybreakCPRet:
                        ;RET

;InitializeCP           ENDP


;--
;--
;-- Boot section #
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                               |
;--------------|        AH = z000 - remaining bits to be tested.               |
;--------------|        AL = xxxx - remaining bits to be dumped in this case.  |
;--------------|        BX = pointer to the buffer to process.                 |
;--------------|        SI↑ = BootBlock.Type                                   |
;--------------|                                                               |
;--------------|--------                                                       |
;--------------|                                                               |
;--------------|                                                               |
;--------------|        BX = pointer to the buffer to process.                 |
;--------------|        SI↑ = BootBlock.Type.                                  |
;--------------|                                                               |
;--------------|                                                               |
;-------------------------------------------------------------------------------

;ALL OF StartCP IS COMMENTED OUT.

;StartCP                        PROC    NEAR


;StartTheCP:            CALL    CheckBankConfig         ;Make sure its the right fb.
                                                        ;We want to point to the
                        ;CALL   IncrementSI             ;address part of the buffer.
                        ;SHL    AX, oneBit              ;But first see if the block
                        ;JC     StartDaybreakCP ;is Daisy or Daybreak.

;StartDaisyCP:
                        ;IN     AX, machineIDPort       ;Find out if this is a Daisy
                        ;AND    AX, machineIDMask       ;or a Daybreak and act accordingly.
                        ;CMP    AX, Daisy
                        ;JE     DoDaisyStart
                        ;JMP    DumpCSBlock

;DoDaisyStart:                                          ;Load CP address for Daisy before
                                                        ;leaving this procedure.
                        ;CALL   IncrementSI

                        ;RET


;StartDaybreakCP:       ;Daybreak cannot currently use a start address, so dump its
                        ;Start Block right here!
                        ;MOV    CX, 5           ;5 bytes to end of boot block
                        ;JMP SHORT DumpCSBlock  ;Note that a call to IncrementSI
                                                ;was commented out above giving us
                                                ;5 instead of the 4 below.

                        ;IN     AX, machineIDPort       ;Find out if this is a Daisy
                        ;AND    AX, machineIDMask       ;or a Daybreak and act accordingly.
                        ;CMP    AX, Daybreak
                        ;JNE    DumpCSBlock

                        ;CALL   IncrementSI             ;jmm:1984-11-08:For now dump adrs.
                        ;CALL   IncrementSI             ;jmm:1984-11-08:For now dump adrs.
                        ;CALL   IncrementSI             ;jmm:1984-11-08:For now dump adrs.


                        ;CALL   IncrementSI

                        ;RET


;StartCP                        ENDP


;--

;--
;-- Boot section #
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                               |
;--------------|                                                               |
```

```
;--------------|                                                                |
;--------------|                                                                |
;--------------|--------                                                        |
;--------------|                                                                |
;--------------|                                                                |
;--------------|                                                                |
;--------------|                                                                |
;-------------------------------------------------------------------------------


;waitForMesaInterrupt   PROC    NEAR

                                                ;Restore registers, then
;WaitForMesaInt:                ;CALL   DaisyCPHalt             ;halt the CP and keep
                        RET                     ;processing the boot file.

;waitForMesaInterrupt   ENDP


;--
;--
;-- Boot section #
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                                |
;--------------|                                                                |
;--------------|                                                                |
;--------------|                                                                |
;--------------|--------                                                        |
;--------------|                                                                |
;--------------|                                                                |
;--------------|                                                                |
;--------------|                                                                |
;-------------------------------------------------------------------------------


;BadMesaInterrupt       PROC    NEAR

;BadMesaInt:            RET

;BadMesaInterrupt       ENDP


;--


;--
;--
;-- Boot section #
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                                |
;--------------|       AH = x000 - remaining bit to be tested.                  |
;--------------|       BX = pointer to the buffer to process.                   |
;--------------|       SI↑ = BootBlock.Type                                     |
;--------------|                                                                |
;--------------|--------                                                        |
;--------------|                                                                |
;--------------|                                                                |
;--------------|                                                                |
;--------------|                                                                |
;--------------|                                                                |
;-------------------------------------------------------------------------------


Checksum                PROC    NEAR

ChecksumBegin:          SHL     AH, oneBit              ;This could be a Start, Initialize
                        JNC     DoChecksum              ;or Write block for CP.
                        JMP     EOForMultipleDBs        ;We might be done with this file.
DoChecksum:

                        ;MOV    CX, 2
                        ;JMP SHORT DumpCSBlock
                        CALL    IncrementSI             ;jmm:1984-11-06 :For now we will
                        CALL    IncrementSI             ;jmm:1984-11-06 :just dump the
                                                        ;jmm:1984-11-06 :checksum type and
                                                        ;jmm:1984-11-06 :the checksum.
                        RET

Checksum                ENDP


;--

;--
;--
;-- Boot section #
;--------------  Assume parameter locations upon entry into this procedure
```

```
;--------------  are as follows:
;--------------
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|--------                                                      |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;-------------------------------------------------------------------------------


;DaisyCPHalt            PROC    NEAR

;DsyCPHalt:             RET

;DaisyCPHalt            ENDP


;DaybreakCPHalt         PROC    NEAR

;DybrkCPHalt:           MOV     AX, 0
                        ;OUT    WriteCSReg, AX
                        ;SHL    AX, 15  ; delay
                        ;SHL    AX, 13  ;  at least 38 cycles
                        ;RET

;DaybreakCPHalt         ENDP

;DaybreakCPStart                PROC    NEAR

;DybrkCPStart:          MOV     AX, 0200H
                        ;OUT    WriteCSReg, AX
                        ;RET

;DaybreakCPStart                ENDP


;--
;--
;--
;-- Boot section #
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                              |
;--------------|                                                              |
;--------------|        AL = BootBlock.Type.lowByte.                          |
;--------------|        BX = pointer to the buffer to process.                |
;--------------|        SI = BootBlock.Type                                   |
;--------------|                                                              |
;--------------|--------                                                      |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;-------------------------------------------------------------------------------


EOForMultipleDBs        PROC    NEAR

EOForMultDBs:           CMP     AH, ((EOF AND highNibbleMask) OR 4)
;The 4 is there because back at DoCPBlock we Rotated AH instead of shifting it.
                        JNE     UnknownBlock
                        CMP     AL, EOF
                        JNE     MultipleDBs
                        ;If this is diagnostics and ether booting
                        ; send a shut up message to boot server.
                        MOV     AL, ES:bootType
                        CMP     AL, diagnostic
                        JNE     GoToLoader
                        MOV     AX, device
                        CMP     AX, ethernet
                        JNE     GoToLoader
                        MOV     BootDeviceIORSpace.useStreamProtocol, shutUp
                        ;TransmitFrame will send error packet.
                        CALL    TransmitFrame           ;in RAMEthBt.  TransmitFrame will
                                                        ; cause an unresoved
                                                        ;  external warning when
                                                        ;   building Disk and Floppy
                                                        ;    initials.
GoToLoader:             JMP     ProcessLoaderFile

MultipleDBs:            JMP     ProcessMultipleDBsFile

                        RET

EOForMultipleDBs        ENDP

;--
```

```
;--
;--
;--
;-- Boot section #
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                                    |
;--------------|                                                                    |
;--------------|                                                                    |
;--------------|                                                                    |
;--------------|--------                                                            |
;--------------|                                                                    |
;--------------|                                                                    |
;--------------|                                                                    |
;--------------|                                                                    |
;--------------|                                                                    |
;-------------------------------------------------------------------------------


IncrementSI             PROC    NEAR

IncSI:                  INC     SI
                        CMP     SI, ES: [BX].bootDataEnd
                        JLE     BufferNotYetEmpty

                        MOV     ES: [BX].bootDataStart, Null
                        MOV     ES: [BX].bootDataEnd, Null

                        CALL    PrepareToFetchBuffer
                        ;POP    jumpTable.processBootBlock
                        ;MOV    currentBlockSequence, OFFSET BootBlockIdentified

BufferNotYetEmpty:      RET


IncrementSI             ENDP

;-------------------------------------------------------------------------------
;
;IncSIFar is what CP specific code will use to call IncrementSI.  The address
; will be loaded into the IO Region prior to beginning to interpret blocks.

IncSIFar                PROC    FAR

CallLocal:      CALL    IncrementSI
                RET

IncSIFar        ENDP



;--
;--
;--
;-- DumpCSBlock: Call IncrementSI CX times and return.  Some procedures Jmp
;--  here and use this return as their own in order to dump the rest of the
;--  block.  Others Call it to move past uninteresting data in the boot block.
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                                    |
;--------------|        CX = Count of bytes to dump.                                |
;--------------|        BX = pointer to the buffer to process.                      |
;--------------|        SI = Offset within buffer                                   |
;--------------|                                                                    |
;--------------|--------                                                            |
;--------------|                                                                    |
;--------------|                                                                    |
;--------------|        BX = pointer to the buffer to process.                      |
;--------------|        SI = CX bytes further through stream.                       |
;--------------|                                                                    |
;--------------|                                                                    |
;-------------------------------------------------------------------------------


;DumpCSBlock     PROC    NEAR

;DumpBlock:      CALL    IncrementSI             ;Bump SI with checking for end of
                ;LOOP   DumpBlock               ;boot buffer until CX runs down.
                ;RET

;DumpCSBlock     ENDP


;--
;--
;--
;-- Boot section #
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
```

```
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|--------                                                      |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;-------------------------------------------------------------------------------


UnknownBlock            PROC    NEAR

UnkBlock:               PUSH    bootingError
                        PUSH    unKnownBootBlock
                        %Jam    (BootStrapHandlerID,OFFSET bootTask)
                        %WaitForSystem                          ;never returns!

UnknownBlock            ENDP


;--

;--
;-- Boot section #
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|--------                                                      |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;-------------------------------------------------------------------------------


NoRAMOpieEntryPoint     PROC    NEAR

NoStart:                PUSH    bootingError
                        PUSH    noRAMStartAddress
                        %Jam    (BootStrapHandlerID,OFFSET bootTask)
                        %WaitForSystem                          ;never returns!

NoRAMOpieEntryPoint     ENDP


;--

;--
;--
;-- Boot section #
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                              |
;--------------|                                                              |
;--------------|       AX = input data left justified.                        |
;--------------|       CX = count of data to shift.                           |
;--------------|                                                              |
;--------------|--------                                                      |
;--------------|                                                              |
;--------------|       DX was trashed here.                                   |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;-------------------------------------------------------------------------------


;ShiftDataToCS           PROC    NEAR

;ShiftDataToCS:         MOV     DX, AX          ;We want to rotate the data so that
                        ;ROL    DX, Nibble      ;we can shift it out a bit at time
                                                ;from the 5-th bit which is the LSB.
;DataShiftLoop:         MOV     AX, ((CSInputBitMask SHL 8) OR CSToggleBit) ;We have
                        ;AND    AH, DL          ;the data into control store one bit
                        ;OR     AL, AH          ;at a time. So we shift the appropriate
                        ;OUT    CPoutputPort, AL ;bit to bit five in DX which corresponds
                        ;MOV    AL, notCSToggleBit ;to the correct bit for the output
                        ;OR     AL, AH          ;port and then do the appropriate
                        ;OUT    CPoutputPort, AL ;toggling to get the bit accepted.
                        ;ROR    DX, oneBit      ;Get the next bit ready.
                        ;LOOP   DataShiftLoop   ;We will loop until all the bits
                                                ;are sent!
                        RET

;ShiftDataToCS           ENDP
```

```
;-- Boot section #
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                              |
;--------------|        BX = pointer to the buffer to process.                |
;--------------|        SI↑ = BootBlock.Address                               |
;--------------|                                                              |
;--------------|--------                                                      |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;-------------------------------------------------------------------------------


;WriteSiriusStartAddress        PROC    NEAR

                        ;MOV    CX, CSaddressLoopCount
;GetCPStartAddress:     CALL    IncrementSI
                        ;MOV    AH, ES: [BX][SI]
                        ;CALL   IncrementSI
                        ;MOV    AL, ES: [BX][SI]
                        ;PUSH   AX
                        ;LOOP   GetCPStartAddress

                        ;MOV    AL, loadShiftRegister
                        ;OUT    CPoutputPort, AL
                        ;MOV    AX, parityInterruptTrapFlags
                        ;MOV    CX, CPflagsBitCount
                        ;CALL   ShiftDataToCS
                        ;MOV    CX, dataWordBitCount
                        ;POP    AX
                        ;CALL   ShiftDataToCS
                        ;MOV    CX, dataWordBitCount*CSsizeWordCount
                        ;CALL   ShiftDataToCS           ;Data here is garbage.
                        ;POP    AX
                        ;CALL   ShiftDataToCS
                        ;MOV    AL, notLoadShiftRegister
                        ;OUT    CPoutputPort, AL

                        ;RET

;WriteSiriusStartAddress        ENDP

;--


;-- Boot section #
;--------------  This procedure gets called after the virtual memory map has
;--------------  been initialized, map registers set. Its task is to load the
;--------------  Loader in the specified virtual memory location then we are
;--------------  ready to give up booting ghost!
;--------------
;--------------
;--------------|        loaderVMLocBase = loaders base location in virtual      |
;--------------|          memory a 32-bit word containing virtual memory jmm????|
;--------------|          address should have been set up at VMM initialization.|
;--------------|                                                               |
;--------------|--------                                                       |
;--------------|                                                               |
;--------------|                                                               |
;--------------|                                                               |
;--------------|                                                               |
;--------------|                                                               |
;-------------------------------------------------------------------------------


ProcessLoaderFile       PROC    NEAR

DoLoaderFile:           POP     AX                              ;Clean the stack.
                        MOV     ES: [BX].bootDataStart, Null    ;We are done loading the
                        MOV     ES: [BX].bootDataEnd, Null      ;emulator so release buffer.
                        MOV     BX, startOfBootBufferPool       ;This is a new file so start
                        MOV     currentBootBuffer, BX           ;processing buffers anew.
                        CMP     bootType, normal                ;If it is a diagnostics boot
                        JE      GoGetTheLoaderFile              ;we are done, otherwise we
                        JMP     BootFadeOut                     ;should go get the loader file.

                        ;Outstanding issue - termination of communcation here when doing
                        ;ethernet diagnostics booting.

GoGetTheLoaderFile:     %NotifyCondition (OFFSET getBootFile)
                        %NotifyCondition (OFFSET bootBufferEmpty)

;display mp code for fetching MesaDove
                        MOV     AX, mpFetchMesaDove
                        CALL    DisplayMPCode

ThereMightBeSomeMore:   %WaitForCondition (OFFSET bootBufferFull,loaderTimeout)
                        JNC     MoreLoader                      ;if not timed out
                        %WaitForCondition (OFFSET finishedLoaderFileFetch,loaderTimeout)
```

```
                        JC      ThereMightBeSomeMore            ;if timed out
                        JMP     BootFadeOut

MoreLoader:             MOV     BX, currentBootBuffer           ;Every time we get a new buffer
                        MOV     SI, ES: [BX].bootDataStart      ;we have to first check that it
                        CMP     SI, ES: [BX].bootDataEnd        ;is really empty before we start
                        JE      ThereMightBeSomeMore            ;processing it.
                        PUSH    ES


                        %EstablishIOPAccess     (generalMapRegister,loaderVirtualMemoryLocation)
                        INC     WORD PTR loaderVirtualMemoryLocation
                        MOV     DX, ES
                        POP     ES
                        PUSH    DS
                        MOV     DS, DX
                        ASSUME  DS:NOTHING
                        MOV     CX, ES: [BX].bootDataEnd        ;We will load a page of the germ at
                        SUB     CX, ES: [BX].bootDataStart      ;a time.
                        INC     CX
                        ADD     SI, BX

WriteAPageOfLoader:     MOV     AL, ES: [SI]
                        MOV     DS: [DI], AL
                        INC     DI
                        INC     SI
                        LOOP    WriteAPageOfLoader

                        POP     DS
                        ASSUME  DS:BootStrapIOR
                        MOV     BX, currentBootBuffer
                        MOV     ES: [BX].bootDataStart, Null
                        MOV     ES: [BX].bootDataEnd, Null
                        MOV     BX, ES: [BX].nextBootBuffer
                        MOV     currentBootBuffer, BX
                        %NotifyCondition (OFFSET bootBufferEmpty)
                        JMP     MoreLoader

ProcessLoaderFile       ENDP

;--

;--

;-- Boot section #
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|--------                                                      |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;-------------------------------------------------------------------------------


BootFadeOut             PROC    NEAR

BootDone:               MOV     AX, mpRunMesaDove
                        CALL    DisplayMPCode
StartRam:               JMP     DWORD PTR jumpTable.startRAMOpie ;Boot task should be jammed by
                                                                 ;the RAMOpie task that is being loaded.

BootFadeOut             ENDP

;--

;-- Boot section #
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                              |
;--------------|                                                              |
;--------------|       AL = BootBlock.Type.lowByte.                           |
;--------------|       BX = pointer to the buffer to process.                 |
;--------------|       SI = BootBlock.Type                                    |
;--------------|                                                              |
;--------------|--------                                                      |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;--------------|                                                              |
;-------------------------------------------------------------------------------
```

```
ProcessMultipleDBsFile  PROC    NEAR


DoMultipleDBsFile:      POP     AX                              ;Clean the stack.

                        CMP     skipUserInterface, normal ;Is this first time?
                        JNE     MoveReentryPointToJumpTable ;If not, parms are set
                        MOV     bootOverlayRequest, 1   ;else default to 1
                        JMP     SHORT LoadOverlayNumber
MoveReentryPointToJumpTable:
                        MOV     DI, bootOverlayReentry  ;Monitor has set up a
                        MOV     jumpTable.startRAMOpie, DI ;reentry point for
                        MOV     DI, bootOverlayReentry+2 ;overlay booting to use.
                        MOV     jumpTable.startRAMOpieCS, DI ;so move it to jump table
LoadOverlayNumber:      MOV     DI, bootOverlayRequest
                        %MultiplyByTwo (DI)
                        MOV     SI, ES: [BX].bootDataStart ;We may now fetch the offset of
                        ADD     DI, SI                  ;the diagnostics boot file desired.
                        MOV     AH, ES: [BX][DI]        ;We want the n-th *.db file in this boot
                        MOV     AL, ES: [BX][DI]+1      ;file so let's get its offset.
                        MOV     DI, AX
                        XOR     DL, DL                  ;table has word offset --
                        SHL     DI, 1                   ;change to byte offset
                        RCL     DL, 1                   ;with overflow in DL.

IsOurDBInThisBuffer:    MOV     AX, ES: [BX].bootDataEnd
                        SUB     AX, SI                  ;Get the size of the buffer and check
                        INC     AX
                        MOV     CX, AX                  ;to see if this is the buffer that the
                        CMP     DL, 0
                        JNE     TryNextBuffer
                        SUB     AX, DI                  ;desired db file is in. If it is we want
                        JNC     TheWantedDBIsHere       ;to go keep processing it as normal.

TryNextBuffer:          SUB     DI, CX
                        SBB     DL, 0
                        MOV     ES: [BX].bootDataStart, Null
                        MOV     ES: [BX].bootDataEnd, Null
                        CALL    PrepareToFetchBuffer
                        JMP     IsOurDBInThisBuffer

TheWantedDBIsHere:      ADD     SI, DI                  ;We now have the offset of the file we
                        JMP     StartSequence           ;have been looking for.



ProcessMultipleDBsFile  ENDP


;--
;  Space could be saved by using the version of this proc that lives in ROM.
;
;DisplayMPCode -- Call this to put a number into the curser as an MP code.
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------|                                                                |
;--------------|        AX = mp code to be displayed                            |
;--------------|                                                                |
;--------------|--------                                                        |
;--------------|                                                                |
;--------------|                                                                |
;--------------|                                                                |
;-------------------------------------------------------------------------------

DisplayMPCode   PROC
RamDispMPCode:  PUSH    ES
                PUSHA
                PUSH    AX
                %EstablishHandlerAccess (MaintPanelHandlerID)
                POP     AX
                ASSUME  ES:MaintPanelIOR
                MOV     maintPanelCode, AX
                %NotifyHandlerCondition (MaintPanelHandlerID,OFFSET maintPanelChanged)
                %WaitForSystem
                POPA
                POP     ES
                ASSUME  ES:IOPELocalRAM
                RET
DisplayMPCode   ENDP

;--
;--
;--
;-- Determination of control store configuration:
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                                |
;--------------|        No assumptions                                          |
;--------------|                                                                |
;--------------|                                                                |
;--------------|                                                                |


```

```
;--------------|--------                                                        |
;--------------|                                                                |
;--------------| Upon return:                                                   |
;--------------|         csBankConfiguration is set as follows:                 |
;--------------|           four K control store          =>  1                  |
;--------------|           eight K control store         =>  2                  |
;--------------|         AX smashed                                             |
;--------------|         BX smashed                                             |
;--------------|                                                                |
;--------------|                                                                |
;-------------------------------------------------------------------------------

;csBankConfiguration     DB      0               ;This will go into IORegion.

;GetCSBankConfig         PROC    NEAR
                         ;Figure out how much control store we have.
                         ;This code could be moved into an IOP initial to save
                         ; Local ram space.
;GetCSBankCfg:           %'ReadEEProm(eePromMemSize, 1)   ;AX ← virt mem,,eeprom.
                         ;JNC    GoodEEProm              ;Check eeprom validity.
;fourKEEPromFormat       EQU     10H
                         ;MOV    AX, fourKEEPromFormat   ;Default is 1 bank.
;GoodEEProm:             AND     AL, 0F0H                ;Get rid of VM size.
                         ;MOV    CS: csBankConfiguration, AL     ;Save it for later.
                         ;RET

;GetCSBankConfig                 ENDP


;-------------------------------------------------------------------------------

;--
;--
;--
;-- LoadAXFromBootBuf --.
;--
;This procedure just loads AX from the boot buffer byte at a time calling the
; procedure IncrementSI to move the pointer along.  This assures that the end
; of the boot buffer will be detected and handled correctly.
;
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                                |
;--------------|         ES:[BX][SI]     byte before the first we want          |
;--------------|--------                                                        |
;--------------|                                                                |
;--------------| Upon return:                                                   |
;--------------|         ES:[BX][SI]     2 bytes further in the stream.         |
;--------------|         AX loaded from ES:[BX][SI]                             |
;--------------|                                                                |
;--------------|                                                                |
;-------------------------------------------------------------------------------


LoadAXFromBootBuf               PROC            NEAR

                         CALL   IncrementSI             ;
LoadAXFromBootBufLateEntry:
                         MOV    AH, ES: [BX][SI]        ;
                         CALL   IncrementSI             ;
                         MOV    AL, ES: [BX][SI]        ;
                         RET

LoadAXFromBootBuf               ENDP

;-------------------------------------------------------------------------------

;--
;--
;--
;-- LoadCXFromBootBuf --.
;--
;This procedure just loads CX from the boot buffer byte at a time calling the
; procedure IncrementSI to move the pointer along.  This assures that the end
; of the boot buffer will be detected and handled correctly.
;
;--------------  Assume parameter locations upon entry into this procedure
;--------------  are as follows:
;--------------
;--------------|                                                                |
;--------------|         ES:[BX][SI]     byte before the first we want          |
;--------------|--------                                                        |
;--------------|                                                                |
;--------------| Upon return:                                                   |
;--------------|         ES:[BX][SI]     2 bytes further in the stream.         |
;--------------|         CX loaded from ES:[BX][SI]                             |
;--------------|                                                                |
;--------------|                                                                |
;-------------------------------------------------------------------------------


;LoadCXFromBootBuf               PROC            NEAR

                         ;CALL   IncrementSI            ;
;LoadCXFromBootBufLateEntry:
                         ;MOV    CH, ES: [BX][SI]       ;
```

```
                        ;CALL   IncrementSI             ;
                        ;MOV    CL, ES: [BX][SI]        ;
                        ;RET

;LoadCXFromBootBuf              ENDP

;-------------------------------------------------------------------------------

                EVEN
EndOfInitial    EQU     $


IOPEInRAM               ENDS

;*******************************************************************************

                        END
```

```
-- File: ShowCharsImpl.mesa - last edit:
-- Mader.ES              7-Aug-85  9:26:25

-- Copyright (C) 1985 by Xerox Corporation. All rights reserved.

DIRECTORY
  Attention USING [AddMenuItem, Post],
  Heap USING [Create],
  MenuData USING [CreateItem, MenuProc],
  Selection USING [Free, CanYouConvert, Convert, Enumerate, EnumerationProc, Value],
  XChar USING [Code, Set],
  XFormat USING [Object, Octal, WriterObject],
  XString USING [AppendReader, Character, Empty, FreeWriterBytes, FromSTRING, Map, MapCharProc, NewWriterBody, Reader, ReaderBody,
  ReaderFromWriter, Writer, WriterBody];

ShowCharsImpl: PROGRAM
  IMPORTS Attention, Heap, MenuData, Selection, XChar, XFormat, XString =
BEGIN

  zone: UNCOUNTED ZONE = Heap.Create [initial: 1];
  first: BOOLEAN ← TRUE;

  AppendOctal: PROCEDURE [to: XString.Writer, n: CARDINAL] =
  BEGIN
  xfo: XFormat.Object ← XFormat.WriterObject [to];

    XFormat.Octal [@xfo, n];
  END;

  ShowString: Selection.EnumerationProc =
  BEGIN
  r: XString.Reader = element.value;

    Attention.Post [s: r, clear: first];
    first ← FALSE;
  END;

  ShowCodes: Selection.EnumerationProc =
  BEGIN
  r: XString.Reader = element.value;
  w: XString.WriterBody ← XString.NewWriterBody [100, zone];
  open: XString.ReaderBody ← XString.FromSTRING ["["L];
  close: XString.ReaderBody ← XString.FromSTRING ["]"L];
  comma: XString.ReaderBody ← XString.FromSTRING [", "L];

    MapChars: XString.MapCharProc =
    BEGIN
      IF ~XString.Empty [XString.ReaderFromWriter [@w]] OR ~first
      THEN XString.AppendReader [to: @w, from: @comma];

      XString.AppendReader [to: @w, from: @open];
      AppendOctal [to: @w, n: XChar.Set [c]];
      XString.AppendReader [to: @w, from: @comma];
      AppendOctal [to: @w, n: XChar.Code [c]];
      XString.AppendReader [to: @w, from: @close];
      RETURN [stop: FALSE];
    END;

    [] ← XString.Map [r, MapChars];
    Attention.Post [s: XString.ReaderFromWriter [@w], clear: first];
    first ← FALSE;
    XString.FreeWriterBytes [@w];
  END;

  ShowChars: MenuData.MenuProc =
  BEGIN
  displayProc: Selection.EnumerationProc = LOOPHOLE [itemData];

    first ← TRUE;
    SELECT TRUE FROM
    Selection.CanYouConvert [target: string, enumeration: FALSE] =>
    BEGIN
    v: Selection.Value ← Selection.Convert [string];

      [] ← displayProc [v, NIL];
      Selection.Free [@v];
    END;

    Selection.CanYouConvert [target: string, enumeration: TRUE] =>
      [] ← Selection.Enumerate [target: string, proc: displayProc];

    ENDCASE =>
    BEGIN
    needString: XString.ReaderBody ← XString.FromSTRING ["Please select some characters and try again."L];

      Attention.Post [@needString];
    END;
  END;

  Init: PROCEDURE =
  BEGIN
  showChars: XString.ReaderBody ← XString.FromSTRING ["Show Characters"L];
  showCodes: XString.ReaderBody ← XString.FromSTRING ["Show Character Codes"L];

    Attention.AddMenuItem [
      MenuData.CreateItem [
        zone: zone, name: @showChars, proc: ShowChars, itemData: LOOPHOLE [ShowString]]];
```

```
    Attention.AddMenuItem [
      MenuData.CreateItem [
        zone: zone, name: @showCodes, proc: ShowChars, itemData: LOOPHOLE [ShowCodes]]];
  END;

 Init [];
END.
```

```
<<File K4.mesa

deLaBeaujardiere:OSBU North:Xerox27-Apr-87 12:57:34

>>

DIRECTORY Ascii, Containee, FormWindow,
          NSFile, NSFileStream,  NSString,
          PropertySheet, Prototype, StarWindowShell, Stream,
          Window, XString;


K4: DEFINITIONS =
BEGIN
OPEN SWS: StarWindowShell;



GeneralItems: TYPE = {iconName, channelSpeed, folderName, others};

TextItems: TYPE = {folderName, docName,
                   justification, lineHeight,
                   preLeading, postLeading,
                   underlining,
                   guessMark, dropKeepMark,
                   pageBreak,
                   others};

MapItems: TYPE = {header, from, to};

FontItems: TYPE = {font0, size0, italics0, bold0,
                   font1, size1, italics1, bold1,
                   font2, size2, italics2, bold2,
                   font3, size3, italics3, bold3,
                   font4, size4, italics4, bold4,
                   font5, size5, italics5, bold5,
                   font6, size6, italics6, bold6,
                   font7, size7, italics7, bold7,
                   font8, size8, italics8, bold8,
                   font9, size9, italics9, bold9};

IconParms:              TYPE = LONG POINTER TO IconParmsRecord;
IconParmsRecord:        TYPE = RECORD[
        heap:           UNCOUNTED ZONE ← NIL,
        propSheet:      SWS.Handle ← SWS.nullHandle,
        iconData:       Containee.DataHandle ← NIL,
        changeProc:     Containee.ChangeProc ← NIL,
        changeProcData: LONG POINTER ← NIL,
        iconFile:       NSFile.Handle ← NSFile.nullHandle,
        signature:      CARDINAL ← 0,
        genProps:       GeneralProperties ← NIL,
        docProps:       DocumentProperties ← NIL,
        mapProps:       MapProperties ← NIL,
        fonProps:       FontProperties ← NIL];

ConvertParms:           TYPE = LONG POINTER TO ConvertParmsRecord;
ConvertParmsRecord:     TYPE = RECORD [
        zone:           UNCOUNTED ZONE ← NIL,
        optionSheet:    SWS.Handle ← SWS.nullHandle,
        logFile:        NSFile.Handle ← NSFile.nullHandle,
        docProps:       DocumentProperties ← NIL,
        mapProps:       MapProperties ← NIL,
        fonProps:       FontProperties ← NIL];

GeneralProperties:      TYPE = LONG POINTER TO GeneralPropertyRecord;
GeneralPropertyRecord:  TYPE = RECORD [
        iconName:       XString.ReaderBody,
        channelSpeed:   ChannelSpeed,
        folderName:     XString.ReaderBody,
        others:         XString.ReaderBody,
        tagSize:        LONG POINTER TO GeneralTagSizes ← NIL];

DocumentProperties:     TYPE = LONG POINTER TO DocumentPropertyRecord;
DocumentPropertyRecord: TYPE = RECORD [
        folderName:     XString.ReaderBody,
        docName:        XString.ReaderBody,
        justification:  BOOLEAN,
        lineHeight:     LineSpacing,
        preLeading:     LineSpacing,
        postLeading:    LineSpacing,
        underlining:    Underlining,
        guessMark:      XString.ReaderBody,
        dropKeepMark:   DropKeep,
        pageBreak:      PageBreak,
        others:         XString.ReaderBody,
        tagSize:        LONG POINTER TO TextTagSizes ← NIL];

MapProperties:          TYPE = LONG POINTER TO MapPropertyRecord;
MapPropertyRecord:      TYPE = RECORD [
        header:         XString.ReaderBody,
        number:         CARDINAL ← 0,
        map:            Mapping ← NIL,
        tagSize:        LONG POINTER TO MapTagSizes ← NIL];

FontProperties: TYPE = LONG POINTER TO FontPropertyRecord;
```

```
    FontPropertyRecord:     TYPE = RECORD [
            fonts:          ARRAY [0..9] OF FontRecord,
            tagSize:        LONG POINTER TO FontTagSizes ← NIL];

    Mapping:                TYPE = LONG POINTER TO MappingRecord;
    MappingRecord:          TYPE = RECORD [
            next:           Mapping ← NIL,
            from:           XString.ReaderBody,
            to:             XString.ReaderBody];

    FontRecord:             TYPE = RECORD [
            font:           FontStyle,
            size:           FontSize,
            italics:        BOOLEAN,
            bold:           BOOLEAN];

    ChannelSpeed:   TYPE = {ninetySix, fortyEight, three}; -- hectobauds
    FontStyle:      TYPE = {modern, classic,  titan};
    FontSize:       TYPE = {eight, ten, twelve, fourteen,
                            eighteen, twentyFour};
    Underlining:    TYPE = {underline, italics, plain};
    LineSpacing:    TYPE = {single, singleHalf, double, triple};
    DropKeep:       TYPE = {drop, keep};
    PageBreak:      TYPE = {drop, keep, unfilled};
    GeneralTagSizes: TYPE = ARRAY GeneralItems OF CARDINAL ← ALL[0];
    TextTagSizes:   TYPE = ARRAY TextItems OF CARDINAL ← ALL[0];
    MapTagSizes:    TYPE = ARRAY MapItems OF CARDINAL ← ALL[0];
    FontTagSizes:   TYPE = ARRAY FontItems OF CARDINAL ← ALL[0];



    -- Procedures implemented in K4WindowImpl

    OpenWindow: PROCEDURE [iconData: Containee.DataHandle,
                           changeProc: Containee.ChangeProc,
                           changeProcData: LONG POINTER,
                           tinyIcon: XString.Character]
                RETURNS [shell: SWS.Handle ← SWS.nullHandle];

    PutFileInFolder: PROCEDURE [file: NSFile.Handle,
                                folderName: NSString.String];


    -- Procedures implemented in K4PSheetImpl

    OpenPSheet: PROCEDURE [iconData: Containee.DataHandle,
                           changeProc: Containee.ChangeProc,
                           changeProcData: LONG POINTER]
                RETURNS [SWS.Handle];

    OpenDocOptionSheet: PROCEDURE [
                           convertParm: ConvertParms,
                           takeDown: PropertySheet.MenuItemProc];


    -- Procedures implemented in K4FiledDataImpl

    TypeAndVersion: PROCEDURE RETURNS [fileType: NSFile.Type,
                                       version:  Prototype.Version];

    LoadFiledData:  PROCEDURE [parm: IconParms]
                      RETURNS [mismatch: BOOLEAN ← FALSE];

    StoreFiledData: PROCEDURE [parm: IconParms];

    FreeIconProps:  PROCEDURE [props: GeneralProperties,
                               z: UNCOUNTED ZONE];

    FreeTextProps:  PROCEDURE [props: DocumentProperties,
                               z: UNCOUNTED ZONE];

    FreeMapProps:   PROCEDURE [props: MapProperties,
                               z: UNCOUNTED ZONE];

    FreeFontProps:  PROCEDURE [props: FontProperties,
                               z: UNCOUNTED ZONE];


    -- Procedures implemented in K4DocumentImpl

    ConvertToDocument: PROCEDURE[logFile: NSFile.Handle,
                                 docProps: DocumentProperties,
                                 mapProps: MapProperties,
                                 fonProps: FontProperties];



    END.  -- of K4
    14-Jan-87 14:35:51 created from DestText and KDEM3.
    28-Jan-87 16:01:53 changed parms of ConvertToViewPoint.
    27-Feb-87 15:56:30 added preliminary ConvertToCanvas.
     3-Mar-87 11:05:03 unified parms for ConvertToViewPoint, ConvertToCanvas.
     4-Mar-87 15:55:03 added PutFileInFolder.
     9-Mar-87 14:08:49 added CanvasName, IconParmRecord, CanvasParmRecord.
```

```
 9-Mar-87 16:43:35 elimination of Courier and addition of options from linked property sheet. Changed iconFileType because of filed props
 changes.
13-Mar-87 10:12:24 added options and modified others.
16-Mar-87  9:43:17 version 9; added signature.
17-Mar-87 13:15:28 added character mapping.
18-Mar-87 11:04:54 defined OpenDocOptionSheet, OpenCanvasOptionSheet, and dropped all other option sheet interfaces.
18-Mar-87 14:22:38 version 11: preset number to 0, not 1.
19-Mar-87 10:51:46 moved application file type and version into proc TypeAndVersion to avoid full recompilation of all impls at every
version change.
31-Mar-87 12:58:48 removed iconName from StoreFiledData.
 2-Apr-87 11:26:48 added parameter saveLog to ConvertTo...
 6-Apr-87 15:06:18 changed parameters of ConvertToCanvas, ConvertToDocument.
14-Apr-87 12:56:02 added paragraph line height and leadings, added drop/keep questionable marks.
24-Apr-87 16:08:12 dropped canvas and ArtScan processing, moved fonts to separate prop sheet.
27-Apr-87 12:56:49 added folder of transmisions in general items, and choice for page breaks.
/
```

```
<< File: K4DocumentImpl.mesa      14-Sep-88 16:02:55

        Parses K4 output and produces structured data
        for input to CommonConversionImpl
        (which makes a ViewPoint document from the structured data)

Owner: Advanced Design - User Interfaces  - deLaBeaujardiere  >>


DIRECTORY Ascii, BackgroundProcess,
          DocInterchangeDefs, DocInterchangePropsDefs,
          FormWindow, Heap, K4, NSFile, NSFileStream,
          NSString,
          Process, PropertySheet,
          Runtime, StarFileTypes,
          StarWindowShell, Stream, Window,
          XChar, XCharSet0, XCharSet360, XString;

K4DocumentImpl: PROGRAM
  IMPORTS BackgroundProcess,
          DocInterchangeDefs, DocInterchangePropsDefs, FormWindow, Heap, K4,
          NSFile, NSFileStream, NSString,
          Process, PropertySheet, Runtime,
          Stream,
          XChar, XCharSet0, XCharSet360, XString
  EXPORTS K4
  SHARES  XString =
BEGIN
OPEN DI: DocInterchangeDefs, DIP: DocInterchangePropsDefs,
     FW: FormWindow, XS: XString;



Baseline:               TYPE = {null, super, sub};
QuestionHandling:       TYPE = {keep, dropCharacter, dropString};

Line: TYPE = LONG POINTER TO LineRecord;
LineRecord: TYPE = RECORD [
        next:           Line ← NIL,
        chunk:          Chunk ← NIL,
        interLine:      CARDINAL,       -- empty lines above
        leadingSpaces:  CARDINAL,
        paragraphHere:  BOOLEAN ← FALSE,
        text:           XS.WriterBody];

Chunk: TYPE = LONG POINTER TO ChunkRecord;
ChunkRecord: TYPE = RECORD [
        next:           Chunk ← NIL,
        aspect:         Aspect ← plain,
        firstChar:      CARDINAL ← 0,   -- pointer to beginning position
        nChars:         CARDINAL ← 0];  -- number of characters

Aspect: TYPE = MACHINE DEPENDENT {
                notAnAspect, plain,
                onSubscript, offSubscript,
                onSuperscript, offSuperscript,
                onUnderline, offUnderline,
                onFont0, (9),                   -- avoid TAB
                onFont1, onFont2, (12), (13),   -- avoid FF and CR
                onFont3, onFont4, onFont5, onFont6,
                onFont7, onFont8, onFont9,
                offFont0, offFont1, offFont2, offFont3,
                offFont4, offFont5, (27),       -- avoid ESC
                offFont6, offFont7, offFont8, offFont9};
                -- the above enumeration is to replace in the source
                -- stream the font/underline/subscript/.. markers
                -- with characters that cannot be accessed through
                -- the keyboard or are not K4 encodings.
                -- Thus the mapping options typed by
                -- the user cannot interfere with the markers.

ConversionHandle: TYPE = LONG POINTER TO ConversionData;
ConversionData: TYPE = RECORD [
        docHandle:      DI.Doc,
        fontProps:      DIP.FontPropsRecord,
        paraProps:      DIP.ParaPropsRecord,
        pageProps:      DIP.PagePropsRecord,
        tabProps:       ARRAY [0..maxTabs) OF DIP.TabStop];

WarningLines: TYPE = [0..10);


tab:            XChar.Character = XCharSet0.Make [tab];
space:          XChar.Character = XCharSet0.Make [space];
lastInSet0:     XChar.Character = XCharSet0.Make [lowerEng];
hyphen:         XChar.Character = XCharSet0.Make [minus];
questionMark:   XChar.Character = XCharSet0.Make [questionMark];
substitute:     XChar.Character = XCharSet360.Make [blackRectGraphic];
paraTab:        XChar.Character = XCharSet0.Make[LOOPHOLE[211C]];
pageEnd:        XChar.Character = XCharSet0.Make[LOOPHOLE[014C]];
lineEnd:        XChar.Character = XCharSet0.Make[LOOPHOLE[015C]];
propsBegin:     XChar.Character = XCharSet0.Make[LOOPHOLE['<.ORD]];
propsEnd:       XChar.Character = XCharSet0.Make[LOOPHOLE['>.ORD]];
  subscript:    XChar.Character = XCharSet0.Make[LOOPHOLE['i.ORD]];
  superscript:  XChar.Character = XCharSet0.Make[LOOPHOLE['s.ORD]];
  underline:    XChar.Character = XCharSet0.Make[LOOPHOLE['u.ORD]];
  fontStyle0:   XChar.Character = XCharSet0.Make[LOOPHOLE['0.ORD]];
```

```
    fontStyle1:    XChar.Character = XCharSet0.Make[LOOPHOLE['1.ORD]];
    fontStyle2:    XChar.Character = XCharSet0.Make[LOOPHOLE['2.ORD]];
    fontStyle3:    XChar.Character = XCharSet0.Make[LOOPHOLE['3.ORD]];
    fontStyle4:    XChar.Character = XCharSet0.Make[LOOPHOLE['4.ORD]];
    fontStyle5:    XChar.Character = XCharSet0.Make[LOOPHOLE['5.ORD]];
    fontStyle6:    XChar.Character = XCharSet0.Make[LOOPHOLE['6.ORD]];
    fontStyle7:    XChar.Character = XCharSet0.Make[LOOPHOLE['7.ORD]];
    fontStyle8:    XChar.Character = XCharSet0.Make[LOOPHOLE['8.ORD]];
    fontStyle9:    XChar.Character = XCharSet0.Make[LOOPHOLE['9.ORD]];
  escape:          XChar.Character = XCharSet0.Make[LOOPHOLE[033C]];

  oneInch:         CARDINAL = 1440;
  spaceWidth:      CARDINAL = oneInch / 10;
  maxTabs:         CARDINAL = 8;    -- 8 arbitrary tabs
                                    -- half an inch apart
  fourthInch:      CARDINAL = oneInch / 4;
  fiveInches:      CARDINAL = oneInch * 5;
  linesFor8Inches:         CARDINAL = 48;   -- 6 lines per inch X 8"
  pointsPerLine:   CARDINAL = 12;



  ConvertToDocument: PUBLIC PROCEDURE [
                               logFile: NSFile.Handle,
                               docProps: K4.DocumentProperties,
                               mapProps: K4.MapProperties,
                               fonProps: K4.FontProperties] =

    BEGIN
    -- This procedure copies the parameters in a record
    -- of its own, so that the original can be freed by the client.
    -- It then paints either an option sheet
    -- or a warning sheet requesting that Interpress be loaded.
    -- Control returns to client as soon as the sheet is painted.
    -- Zone and nodes acquired here are freed by ReleaseParm.

    ownZone: UNCOUNTED ZONE  ← Heap.Create[1];
    cp:      K4.ConvertParms ← ownZone.NEW[K4.ConvertParmsRecord];
    source, sink, map: K4.Mapping ← NIL;

    cp.zone ← ownZone;
    cp.logFile ← logFile;

    cp.docProps ← ownZone.NEW[K4.DocumentPropertyRecord];
    cp.docProps↑ ← docProps↑;
    cp.docProps.folderName ← XS.CopyToNewReaderBody[
                                  @docProps.folderName, cp.zone];
    cp.docProps.docName ← XS.CopyToNewReaderBody[
                                  @docProps.docName, cp.zone];
    cp.docProps.guessMark ← XS.CopyToNewReaderBody[
                                  @docProps.guessMark, cp.zone];
    cp.docProps.others ← XS.CopyToNewReaderBody[
                                  @docProps.others, cp.zone];

    cp.mapProps ← ownZone.NEW[K4.MapPropertyRecord];
    cp.mapProps.header ← XS.CopyToNewReaderBody[@mapProps.header, cp.zone];
    cp.mapProps.number ← mapProps.number;
    source ← mapProps.map;
    FOR k: CARDINAL IN [1..mapProps.number] DO
      map ← cp.zone.NEW [K4.MappingRecord ← [
                from: XS.CopyToNewReaderBody[@source.from, cp.zone],
                to:   XS.CopyToNewReaderBody[@source.to, cp.zone],
                next: NIL]];
      IF cp.mapProps.map = NIL
          THEN cp.mapProps.map ← map
          ELSE sink.next ← map;
      sink ← map;
      source ← source.next;
      ENDLOOP;

    cp.fonProps ← ownZone.NEW[K4.FontPropertyRecord];
    cp.fonProps↑ ← fonProps↑;

    IF Runtime.IsBound [LOOPHOLE [DI.StartCreation]]
      THEN K4.OpenDocOptionSheet [cp, TakeDownSheet]
      ELSE ShowWarning [cp];
    END;  -- of ConvertToDocument


  TakeDownSheet: PropertySheet.MenuItemProc =
    BEGIN
    cp: K4.ConvertParms ← LOOPHOLE[clientData];
    IF menuItem = start
      THEN Process.Detach [FORK ConvertProcess[cp]]
      ELSE ReleaseParm [cp];
    ok ← TRUE;
    END;  -- of TakeDownSheet


  ShowWarning: PROC [cp: K4.ConvertParms] =
    BEGIN
    pSheetName: XS.ReaderBody ← XS.FromSTRING ["Kurzweil Converter"L];
    pSize: Window.Dims ← [420, 300];
    pPlace: Window.Place ← [550, 100];
```

```
    [] ← PropertySheet.Create [formWindowItems: MakeWarning,
                               formWindowItemsLayout: LayoutWarning,
                               menuItemProc: ExitWarning,
                               menuItems: [done: TRUE, cancel: TRUE],
                               title: @pSheetName,
                               size: pSize,
                               placeToDisplay: pPlace,
                               clientData: cp];
    END;  -- of ShowWarning


  MakeWarning: FW.MakeItemsProc =
    BEGIN
    txt: ARRAY WarningLines OF LONG STRING ← [
        "VP Editor is needed to make a document, but is not running."L,
        "Please run the following applications (if they are idle,"L,
        "then press <Done>:"L,
        "      Font Manager"L,
        "      WorkStation Keyboards"L,
        "      Keyboards"L,
        "      Interscript Converter"L,
        "      VP Document Editor"L,
        "  "L,
        "If want to drop the document creation, press <Cancel>."L];
    msg: XString.ReaderBody;

    FOR k: WarningLines IN WarningLines DO
      msg ← XString.FromSTRING[txt[k]];
      FW.MakeTextItem [window: window, myKey: k,
                       tag: NIL, readOnly: TRUE, boxed: FALSE,
                       width: 400, initString: @msg];
      ENDLOOP;
    END;  -- of MakeWarning


  LayoutWarning: FW.LayoutProc =
    BEGIN
    FOR k: WarningLines IN WarningLines DO
      line: FW.Line ← FW.AppendLine [window: window, spaceAboveLine: 0];
      FW.AppendItem [window: window, item: k,
                     line: line, preMargin: 10];
      ENDLOOP;
    END;  -- of LayoutWarning


  ExitWarning: PropertySheet.MenuItemProc =
    BEGIN
    cp: K4.ConvertParms ← LOOPHOLE[clientData];
    newMsg: XString.ReaderBody ← XString.FromSTRING[
          "The Document Editor is still not running..."L];

    IF menuItem = cancel THEN
      BEGIN
      ReleaseParm [cp];
      RETURN [TRUE];
      END;
    IF Runtime.IsBound[LOOPHOLE[DI.StartCreation]] THEN
      BEGIN
      K4.OpenDocOptionSheet [cp, TakeDownSheet];
      RETURN [TRUE];
      END;

    FW.SetTextItemValue [formWindow, 0, @newMsg, TRUE];
    RETURN [FALSE];
    END;  -- of ExitWarning


  ConvertProcess: PROCEDURE [cp: K4.ConvertParms] =
    BEGIN
    processName: XS.ReaderBody ← XS.FromSTRING["Making Document"L];

    ManagedProcess: BackgroundProcess.CallBackProc =
                    -- ManagedProcess must be within ConvertProcess
                    -- to be supervised by the Background Processor
    BEGIN
    ENABLE UNWIND => {finalStatus ← aborted; CONTINUE};

    zone:                   UNCOUNTED ZONE ← cp.zone;
    rawText:                XS.WriterBody;
    firstLine, currentLine: Line ← NIL;
    line:                   Line;
    map:                    K4.Mapping ← NIL;
    maxMapLength:           CARDINAL ← 0;
    leadingSpaces:          CARDINAL;
    emptyLinesAbove:        CARDINAL ← 0;
    conversionHandle:       ConversionHandle;
    docFile:                NSFile.Handle;
    docSession:             NSFile.Session;
    status:                 DI.FinishCreationStatus;
    docNSName, folderNSName: NSString.String;
    docAttributes:          ARRAY [0..1) OF NSFile.Attribute;
    logStream:              Stream.Handle;
    hasProps, endOfPage, endOfStream: BOOLEAN ← FALSE;
    qHandling:              QuestionHandling;
    qCharacter:             XS.Character;    -- accelerator
    qString:                XS.Reader;       -- accelerator
```

```
    conversionHandle ← BeginDocument [zone, cp.docProps, cp.fonProps];
    IF conversionHandle = NIL THEN RETURN;          -- creation failed

    IF  (cp.mapProps # NIL)
    AND (cp.mapProps.map # NIL)
    AND (NOT XS.Empty [@cp.mapProps.map.from]) THEN
          [map, maxMapLength] ← OrderMap [cp.mapProps.map];

    qString ← @cp.docProps.guessMark;
    SELECT TRUE FROM
      cp.docProps.dropKeepMark = keep => qHandling ← keep;
      XS.CharacterLength[qString] = 1 =>
          BEGIN
          qHandling ← dropCharacter;
          qCharacter ← XS.Lop[qString];
          END;
      ENDCASE => qHandling ← dropString;

    logStream ← NSFileStream.Create [cp.logFile, FALSE];
    Stream.SetInputOptions [logStream, [signalEndOfStream: TRUE]];

    rawText ← XS.NewWriterBody [300, zone];
                          -- large enough to contain a whole line
                          -- in small font. Will be expanded if needed.

    DO
      -- get a raw line of text
      [leadingSpaces, hasProps, endOfPage, endOfStream] ← FillBuffer [
                                  logStream, @rawText,
                                  qHandling, qCharacter, qString];

      IF XS.Empty [XS.ReaderFromWriter[@rawText]] THEN
        emptyLinesAbove ← emptyLinesAbove + 1
      ELSE
        BEGIN
        line ← zone.NEW [LineRecord];
        IF firstLine = NIL THEN currentLine ← firstLine ← line
        ELSE currentLine ← currentLine.next ← line;
        line.leadingSpaces ← leadingSpaces;
        line.interLine ← emptyLinesAbove;
        emptyLinesAbove ← 0;
        line.text ← XS.CopyToNewWriterBody[
                               XS.ReaderFromWriter[@rawText], zone];
        IF map # NIL THEN Map [XS.ReaderFromWriter[@rawText], @line.text, map];
        IF hasProps THEN Parcel [line, zone];
        END;

      IF endOfPage THEN -- end of page found after current line
        BEGIN
        linesInPage: CARDINAL ← FillDocument [conversionHandle, firstLine,
                                              cp.fonProps,
                                              cp.docProps.underlining];
        SELECT cp.docProps.pageBreak FROM
          drop     => NULL;
          keep     => DI.AppendPageBreak [conversionHandle.docHandle,
                                          @conversionHandle.fontProps];
          unfilled => IF linesInPage < linesFor8Inches THEN
                        DI.AppendPageBreak [conversionHandle.docHandle,
                                            @conversionHandle.fontProps];
          ENDCASE;
        ReleaseLines [firstLine, zone];
        firstLine ← NIL;
        emptyLinesAbove ← 0;
        END;

      IF endOfStream THEN EXIT;    -- end of stream found after current line

      ENDLOOP;                     -- loop to get next raw line

    [] ← FillDocument [conversionHandle, firstLine,
                       cp.fonProps, cp.docProps.underlining];
    ReleaseLines [firstLine, zone];

    [docFile, docSession, status] ← DI.FinishCreation[@conversionHandle.docHandle];

    IF docFile # NSFile.nullHandle THEN
      BEGIN       -- reopen docFile in null session
      ENABLE NSFile.Error =>
              {
              NSFile.Close[docFile, docSession ! NSFile.Error => CONTINUE];
              docFile ← NSFile.nullHandle;
              CONTINUE;
              };
      tmpRef: NSFile.Reference ← NSFile.GetReference[file: docFile,
                                                     session: docSession];
      tmpFile: NSFile.Handle ← docFile;
      docFile ← NSFile.OpenByReference[reference: tmpRef];
      NSFile.Close[tmpFile, docSession];
      NSFile.Logoff[docSession ! NSFile.Error => CONTINUE];
      END;

    docNSName ← XS.NSStringFromReader [
                        @cp.docProps.docName, zone];
    docAttributes[0] ← [name[docNSName]];
    NSFile.ChangeAttributes [docFile, DESCRIPTOR [docAttributes]];
    NSString.FreeString [zone, docNSName];
```

```
    folderNSName ← XS.NSStringFromReader [
                        @cp.docProps.folderName, zone];
    K4.PutFileInFolder [docFile, folderNSName];
    NSString.FreeString [zone, folderNSName];
    NSFile.Close [docFile];

    Stream.Delete [logStream];
    XS.FreeWriterBytes [@rawText];
    ReleaseParm [cp];       -- done last, beause it releases the zone
    END;  -- of ManagedProcess

    Process.SetPriority[Process.priorityBackground];
    [] ← BackgroundProcess.ManageMe[name: @processName,
                                    callBackProc: ManagedProcess,
                                    abortable: FALSE];  --** TRUE later
    END;  -- of ConvertProcess


FillBuffer: PROC [stream:       Stream.Handle,
                  bufferW:      XS.Writer,
                  qHandling:    QuestionHandling,
                  qCharacter:   XS.Character,
                  qString:      XS.Reader]
          RETURNS [leadingSpaces: CARDINAL ← 0,
                   hasProps:      BOOLEAN ← FALSE,
                   endOfPage:     BOOLEAN ← FALSE,
                   endOfStream:   BOOLEAN ← FALSE] =
  BEGIN
  -- Gets a line of text in buffer, up to next CR or FF.
  -- Drops the leading spaces.
  -- Replace strings of 3 or more spaces by a tab.
  -- Replaces props marker by ESC+code so that they cannot be
  -- changed by user mapping options.
  ENABLE Stream.EndOfStream => GOTO EndStream;
  xChar, yChar: XChar.Character;
  aspect: Aspect;
  insideSpaces, k: CARDINAL;

  NextChar: PROC RETURNS [XS.Character] = INLINE
        {RETURN [XCharSet0.Make[LOOPHOLE[Stream.GetByte [stream]]]]};

  XS.ClearWriter [bufferW];
  insideSpaces ← 0;

  DO            -- count and drop leading spaces
    xChar ← NextChar[];
    IF xChar # space THEN EXIT;
    leadingSpaces ← leadingSpaces + 1;
    ENDLOOP;

  DO            -- get remainder of line
    SELECT TRUE FROM
      xChar = lineEnd => RETURN;
      xChar = pageEnd => {endOfPage ← TRUE; RETURN};
      qHandling = dropCharacter
      AND xChar = qCharacter => NULL;
      xChar = propsBegin =>     -- replace begin-props markers
                BEGIN
                yChar ← NextChar[];
                aspect ← SELECT yChar FROM
                  underline => onUnderline,
                  subscript => onSubscript,
                  superscript => onSuperscript,
                  fontStyle0 => onFont0,
                  fontStyle1 => onFont1,
                  fontStyle2 => onFont2,
                  fontStyle3 => onFont3,
                  fontStyle4 => onFont4,
                  fontStyle5 => onFont5,
                  fontStyle6 => onFont6,
                  fontStyle7 => onFont7,
                  fontStyle8 => onFont8,
                  fontStyle9 => onFont9,
                  ENDCASE => notAnAspect;
                IF aspect = notAnAspect THEN
                  {XS.AppendChar [bufferW, xChar];
                   XS.AppendChar [bufferW, yChar]}
                ELSE
                  {XS.AppendChar [bufferW, escape];
                   XS.AppendChar [bufferW,
                            XCharSet0.Make[LOOPHOLE[aspect]]];
                   hasProps ← TRUE};
                END;
      xChar = propsEnd =>       -- replace end-props markers
                BEGIN
                yChar ← NextChar[];
                aspect ← SELECT yChar FROM
                underline => offUnderline,
                  subscript => offSubscript,
                  superscript => offSuperscript,
                  fontStyle0 => offFont0,
                  fontStyle1 => offFont1,
                  fontStyle2 => offFont2,
                  fontStyle3 => offFont3,
                  fontStyle4 => offFont4,
                  fontStyle5 => offFont5,
                  fontStyle6 => offFont6,
```

```
                    fontStyle7 => offFont7,
                    fontStyle8 => offFont8,
                    fontStyle9 => offFont9,
                    ENDCASE => notAnAspect;
                  IF aspect = notAnAspect THEN
                   {XS.AppendChar [bufferW, xChar];
                    XS.AppendChar [bufferW, yChar]}
                  ELSE
                   {XS.AppendChar [bufferW, escape];
                    XS.AppendChar [bufferW,
                                   XCharSet0.Make[LOOPHOLE[aspect]]];
                    hasProps ← TRUE};
                  END;
        xChar = space => insideSpaces ← insideSpaces + 1;
                                   -- count embedded spaces
        xChar IN (space..lastInSet0] =>
                  BEGIN
                       -- replace space strings by tab or space
                  SELECT insideSpaces FROM
                    0     => NULL;
                    > 5   => XS.AppendChar [bufferW, tab];
                    ENDCASE =>
                      BEGIN
                      FOR k IN [1..insideSpaces] DO
                        XS.AppendChar [bufferW, space];
                        ENDLOOP;
                      END;
                  XS.AppendChar [bufferW, xChar];
                  insideSpaces ← 0;
                  END;
        xChar = tab =>
                  XS.AppendChar [bufferW, xChar];
        ENDCASE =>
                  XS.AppendChar [bufferW, substitute];
        -- ** handle questionable char "dropString" later

      xChar ← NextChar [];
      ENDLOOP;
    EXITS EndStream => {endOfStream ← TRUE};

    END;  -- of FillBuffer


  Parcel: PROC [line: Line, zone: UNCOUNTED ZONE] =
    BEGIN
    -- Font or aspect markers found in line.text generate
    -- a linked list of "chunks".
    -- There is at least one marker when we enter this proc.

    code:         XChar.Character;
    index:        CARDINAL ← 0;
    source:       XS.Reader ← XS.ReaderFromWriter[@line.text];
    newChunk:     Chunk ← NIL;
    currentChunk: Chunk ← zone.NEW [ChunkRecord];

    line.chunk ← currentChunk;

    WHILE index < XS.ByteLength[source] DO         -- parse source text
      code ← XS.NthCharacter [source, index];
      IF code # escape THEN
        BEGIN
        index ← index + 1;
        END
      ELSE
        BEGIN
        currentChunk.nChars ← index - currentChunk.firstChar;
        -- Make a new chunk, unless the current chunk is at beginning of line.
        IF index > 0 THEN
          BEGIN
          newChunk ← zone.NEW [ChunkRecord];
          currentChunk.next ← newChunk;
          currentChunk ← newChunk;
          END;
        code ← XS.NthCharacter [source, index + 1];
        currentChunk.aspect ← LOOPHOLE[XChar.Code[code], Aspect];
        index ← index + 2;
        currentChunk.firstChar ← index;
        END;
      ENDLOOP;
    currentChunk.nChars ← index - currentChunk.firstChar;
    END;  -- of Parcel



  ReleaseLines: PROC [firstLine: Line, z: UNCOUNTED ZONE] =
    BEGIN
    line: Line ← firstLine;
    holdLine: Line ← NIL;
    chunk, holdChunk: Chunk ← NIL;

    DO
      IF line = NIL THEN RETURN;
      chunk ← line.chunk;
      DO
        IF chunk = NIL THEN EXIT;
        holdChunk ← chunk.next;
        z.FREE [@chunk];
```

```
      chunk ← holdChunk;
      ENDLOOP;
    XS.FreeWriterBytes [@line.text];
    holdLine ← line.next;
    z.FREE [@line];
    line ← holdLine;
    ENDLOOP;
  END;  -- of ReleaseLines


ReleaseParm: PROC [cp: K4.ConvertParms] =
  BEGIN
  z: UNCOUNTED ZONE ← cp.zone;

  NSFile.Close [cp.logFile];
  K4.FreeTextProps [cp.docProps, z];
  K4.FreeMapProps [cp.mapProps, z];
  K4.FreeFontProps [cp.fonProps, z];
  z.FREE [@cp];
  Heap.Delete [z];
  END;  -- of ReleaseParm


BeginDocument: PROCEDURE [z: UNCOUNTED ZONE,
                          docProps: K4.DocumentProperties,
                          fonProps: K4.FontProperties]
                RETURNS [document: LONG POINTER] =
  BEGIN
  h: ConversionHandle ← z.NEW [ConversionData];

  DIP.GetPagePropsDefaults[@h.pageProps];

  DIP.GetFontPropsDefaults[@h.fontProps];
  h.fontProps.fontDesc.pointSize ←
              SELECT fonProps.fonts[0].size FROM
                                    twelve  => 12,
                                    ten     => 10,
                                    eight   =>  8,
                                    fourteen => 14,
                                    ENDCASE => 12;
  h.fontProps.fontDesc.weight ← IF fonProps.fonts[0].bold
              THEN bold ELSE medium;
  h.fontProps.fontDesc.designVariant ← IF fonProps.fonts[0].italics
              THEN italic ELSE roman;
  h.fontProps.fontDesc.family ←
              SELECT fonProps.fonts[0].font FROM
                                    classic => century,
                                    modern  => frutiger,
                                    ENDCASE => titan;

  DIP.GetParaPropsDefaults[@h.paraProps];
  h.paraProps.basicProps.preLeading ← SELECT docProps.preLeading FROM
                                    single => 0,
                                    singleHalf => 6,
                                    double => 12,
                                    ENDCASE => 24;
  h.paraProps.basicProps.postLeading ← SELECT docProps.postLeading FROM
                                    single => 0,
                                    singleHalf => 6,
                                    double => 12,
                                    ENDCASE => 24;
  h.paraProps.basicProps.lineHeight ← SELECT docProps.lineHeight FROM
                                    single => 12,
                                    singleHalf => 18,
                                    double => 24,
                                    ENDCASE => 36;
  h.paraProps.basicProps.justified ← docProps.justification;
  h.paraProps.tabStops ← DESCRIPTOR [h.tabProps];

  FOR k: CARDINAL IN [0..maxTabs) DO            -- set 12 tabs at 1/2 inch
    h.tabProps[k].dotLeader ← FALSE;
    h.tabProps[k].tabStopOffset ← 36 * k;       -- 72 points/inch
    h.tabProps[k].tabStopAlignment ← left;
    ENDLOOP;

  [h.docHandle,,,,,] ← DI.StartCreation [
                       simple,  -- simple pagination
                       FALSE,   -- no header
                       FALSE,   -- no footer
                       @h.fontProps,
                       @h.paraProps,
                       @h.pageProps];
  document ← h;
  END;  -- of BeginDocument


FillDocument: PROC [h: ConversionHandle,
                    lineList: Line,
                    fontOptions: K4.FontProperties,
                    underOption: K4.Underlining]
                    RETURNS [linesInPage: CARDINAL ← 0] =
  BEGIN
  partialBody:  XS.ReaderBody;
  fullReader:   XS.Reader;
  context:      XS.Context ← XS.vanillaContext;
  line:         Line;
  lastChar:     XChar.Character;
```

```
    IF lineList = NIL THEN RETURN;

    MarkParagraphs [lineList, FALSE];
                -- Find the paragraph boundaries

    FOR line ← lineList, line.next WHILE line # NIL DO

      linesInPage ← linesInPage + 1 + line.interLine;
          -- count to see where is the last line in the page;
          -- if it lies "far up enough" from the bottom
          -- (arbitrarily, 3" from bottom, 8" from top),
          -- we will issue a page break.

      IF line.paragraphHere THEN
        BEGIN
        h.fontProps.nUnderlines ← 0;  -- turn off underline at
                                      -- beginning of a paragraph
        FOR k: CARDINAL IN [2..line.interLine) DO  -- make white space
          DI.AppendNewParagraph [[doc[h.docHandle]],
                                 @h.paraProps,
                                 @h.fontProps];
          ENDLOOP;
        DI.AppendNewParagraph [[doc[h.docHandle]],
                               @h.paraProps, @h.fontProps];
        END;

      fullReader ← XS.ReaderFromWriter[@line.text];
      IF line.chunk = NIL THEN
        BEGIN
        DI.AppendText [[doc[h.docHandle]], fullReader,
                       XS.vanillaContext, @h.fontProps];
        END
      ELSE
        BEGIN
        FOR chunk: Chunk ← line.chunk, chunk.next WHILE chunk # NIL DO
          SetAspect [chunk, fontOptions, underOption, @h.fontProps];
          IF chunk.nChars > 0 THEN
            BEGIN
            [partialBody, context] ← XS.Piece [fullReader,
                                               chunk.firstChar,
                                               chunk.nChars];
          DI.AppendText [[doc[h.docHandle]],
                         @partialBody, context, @h.fontProps];
            END;
          ENDLOOP;  -- loop to process next chunk in line
        END;
      -- Put a space after the last character of the line
      -- unless there is already a space or an hyphen.
      lastChar ← XS.NthCharacter [fullReader,
                                  XS.CharacterLength[fullReader] - 1];
      IF lastChar # space AND lastChar # hyphen THEN
        DI.AppendChar [[doc[h.docHandle]], space, @h.fontProps];
      ENDLOOP;  -- loop to process next line

  END;  -- of FillDocument


SetAspect: PROC [chunk: Chunk,
                 fontOptions: K4.FontProperties,
                 underOption: K4.Underlining,
                 vpFont: DIP.FontProps] =
  BEGIN
  -- Given a chunk of text with an aspect code,
  -- and given the font choices made by the user,
  -- adjust a property of the VP font as required.

  SetFont: PROC [fontChoice: K4.FontRecord] =
        BEGIN
        vpFont.fontDesc.family ← SELECT fontChoice.font FROM
                                 classic => century,
                                 modern  => frutiger,
                                 titan   => titan,
                                 ENDCASE => century;
        vpFont.fontDesc.weight ← IF fontChoice.bold
                                 THEN bold ELSE medium;
        vpFont.fontDesc.designVariant ← IF fontChoice.italics
                                 THEN italic ELSE roman;
        vpFont.fontDesc.pointSize ← SELECT fontChoice.size FROM
                                 twelve   => 12,
                                 ten      => 10,
                                 eight    =>  8,
                                 fourteen => 14,
                                 ENDCASE  => 12;
        IF fontChoice.font = titan THEN
          BEGIN
          vpFont.fontDesc.designVariant ← roman; -- in case it was italics
          IF vpFont.fontDesc.pointSize < 10 THEN
                                 vpFont.fontDesc.pointSize ← 10;
          IF vpFont.fontDesc.pointSize > 12 THEN
                                 vpFont.fontDesc.pointSize ← 12;
          END;
        END;  -- of SetFont

  SELECT chunk.aspect FROM
    onSubscript    => vpFont.placement ← sub;
    offSubscript   => vpFont.placement ← null;
```

```
      onSuperscript  => vpFont.placement ← super;
      offSuperscript => vpFont.placement ← null;
      onUnderline    => SELECT underOption FROM
                          underline => vpFont.nUnderlines ← 1;
                          italics   => {vpFont.nUnderlines ← 0;
                                        vpFont.fontDesc.designVariant ← italic};
                          plain     => vpFont.nUnderlines ← 0;
                          ENDCASE;
      offUnderline   => SELECT underOption FROM
                          underline => vpFont.nUnderlines ← 0;
                          italics   => {vpFont.nUnderlines ← 0;
                                        vpFont.fontDesc.designVariant ← roman};
                          plain     => vpFont.nUnderlines ← 0;
                          ENDCASE;
      onFont0        => SetFont [fontOptions.fonts[0]];
      onFont1        => SetFont [fontOptions.fonts[1]];
      onFont2        => SetFont [fontOptions.fonts[2]];
      onFont3        => SetFont [fontOptions.fonts[3]];
      onFont4        => SetFont [fontOptions.fonts[4]];
      onFont5        => SetFont [fontOptions.fonts[5]];
      onFont6        => SetFont [fontOptions.fonts[6]];
      onFont7        => SetFont [fontOptions.fonts[7]];
      onFont8        => SetFont [fontOptions.fonts[8]];
      onFont9        => SetFont [fontOptions.fonts[9]];
      ENDCASE        => SetFont [fontOptions.fonts[0]];  -- restore font 0

    END;  -- of SetAspect


  Map: PROC [fromR: XS.Reader, toW: XS.Writer, mapOptions: K4.Mapping] =
    BEGIN
    fromIndex: CARDINAL ← 0;
    fromMapLg: CARDINAL;
    piece: XS.ReaderBody;
    context: XS.Context;  -- not used...

    XS.ClearWriter [toW];

    WHILE fromIndex < XS.CharacterLength [fromR] DO
      FOR map: K4.Mapping ← mapOptions, map.next WHILE map # NIL DO
        fromMapLg ← XS.CharacterLength [@map.from];
        [piece, context] ← XS.Piece [fromR, fromIndex, fromMapLg];
        IF XS.Equal [@piece, @map.from] THEN
          BEGIN
          XS.AppendReader [toW, @map.to];
          fromIndex ← fromIndex + fromMapLg;
          EXIT;
          END;
        REPEAT
        FINISHED => BEGIN -- no mapping match
                    XS.AppendChar [toW, XS.NthCharacter [fromR, fromIndex]];
                    fromIndex ← fromIndex + 1;
                    END;
      ENDLOOP;
    ENDLOOP;

  END;  -- of Map


OrderMap: PROC [map: K4.Mapping] RETURNS [K4.Mapping, CARDINAL] =
  BEGIN
  -- This procedure sorts the mapping entries by string length,
  -- placing the longest string first.
  -- The sort technique is a bit crude, because we don't have
  -- back pointers, but the lists are generally short.

  prior, current, next, hold: K4.Mapping ← NIL;
  entryHasMoved: BOOLEAN ← FALSE;

  IF map = NIL THEN RETURN [NIL, 0];

  <<** not working correctly yet
  entryHasMoved ← TRUE;
  WHILE entryHasMoved DO
    entryHasMoved ← FALSE;
    prior ← current ← map;
    next ← map.next;
    DO
      IF next = NIL OR current = NIL THEN EXIT;
      IF XS.ByteLength [@next.from] >
         XS.ByteLength [@current.from] THEN
        BEGIN
        IF current = prior THEN
          BEGIN
          map ← next;
          map.next ← current;
          current.next ← next.next;
          END
        ELSE
          BEGIN
          hold ← prior.next;
          prior.next ← current.next;
          current.next ← next.next;
          next.next ← hold;
          END;
        entryHasMoved ← TRUE;
        EXIT;
```

```
        END;
      prior ← prior.next;
      current ← current.next;
      next ← next.next;
      ENDLOOP;
    ENDLOOP;
  **>>
  RETURN [map, 10];      --**10 is temp...
  END;  -- of OrderMap


MarkParagraphs: PROC [firstLine: Line,
                      keepLineBreak: BOOLEAN] =  --** unused for now
  BEGIN
  -- We think there is a paragraph if one of the following is true:
  -- 1) the line has an interval (i.e distance from line above)
  --    greater than the smallest line interval;
  -- 2) the line above is "clearly" shorter than the current line,
  --    ("clearly" arbitrarily defined as 1 inch).
  -- 3) the line is "clearly" indented from the prior line's,
  --    ("clearly" arbitrarily defined as 1/4 inch).

  line: Line;
  priorMargin, smallestMargin: CARDINAL ← LAST[CARDINAL];
  priorLength, lineLength: CARDINAL;
  smallestInterline: CARDINAL ← LAST[CARDINAL];
  LineLength: PROC [r: XS.Reader] RETURNS [length: CARDINAL] =
        BEGIN
        length ← XS.CharacterLength [r] * oneInch / 10;
        -- Grossly approximative. Need better way...
        END;  -- of LineLength

  -- If we must keep line breaks, no point looking further...
  IF keepLineBreak THEN
    BEGIN
    FOR line ← firstLine, line.next WHILE line # NIL DO
      line.paragraphHere ← TRUE;
      ENDLOOP;
    RETURN;
    END;

  -- Find the smallest line interval, to tell us if
  -- the text is generally single-spaced, or double-spaced, etc..
  -- and find the smallest left margin.
  FOR line ← firstLine, line.next WHILE line # NIL DO
    smallestInterline ← MIN [smallestInterline, line.interLine];
    smallestMargin ← MIN [smallestMargin, line.leadingSpaces];
    ENDLOOP;

  priorLength ← LineLength [XS.ReaderFromWriter[@firstLine.text]];

  FOR line ← firstLine.next, line.next WHILE line # NIL DO
    lineLength ← LineLength [XS.ReaderFromWriter[@line.text]];
    line.paragraphHere ←
                (line.interLine > smallestInterline)
             OR (priorLength + oneInch < lineLength)
             OR (line.leadingSpaces > priorMargin + fourthInch)
             OR (line.paragraphHere);   -- already a paragraph
    priorLength ← lineLength;
    priorMargin ← line.leadingSpaces;
    ENDLOOP;

  END;  -- of MarkParagraphs


END.  -- of K4DocumentImpl
 5-Feb-87 12:48:28 upgraded to new version of CommonConversion, and to new parameter for ConvertToDocument.
10-Feb-87 14:05:39 added auto creation of folder, movement of document into folder.
18-Feb-87 17:51:47 added NIL header/footer parameters for BeginDocument.
 4-Mar-87  9:48:28 adapted to new common parameters for ConvertToDocument and ConvertToCanvas.
 4-Mar-87 13:55:18 added presetting of fontChoices to user options.
10-Mar-87 15:06:03 adapted to linked pSheet.
11-Mar-87 17:41:42 moved here from K4WindowImpl the test for Editor loaded.
14-Mar-87 10:25:11 forgot to call SetTextOptions.
18-Mar-87 11:02:41 replaced all option sheet logic by call to OpenDocOptionSheet.
19-Mar-87 10:20:41 released mapProps in ReleaseParm.
23-Mar-87 16:01:01 forgot to test for "cancel" in option sheet.
23-Mar-87 16:14:13 merged in ComonConversionImpl to handle string substitutions.
26-Mar-87 11:29:57 increased width of warning text.  Redesigned entire line/subLines concept, replaced with line/pieces, dropped String
interface and used XString for buffer.
 2-Apr-87 12:43:03 named logFile according to document name.
 3-Apr-87 11:06:45 implemented mapping.
 5-Apr-87 11:23:54 was quitting on blank lines.
 5-Apr-87 11:48:11 characters outside [space..3768] not caught.
 6-Apr-87 10:33:28 upgraded paragraph recognition procedure, removed leading spaces.
 6-Apr-87 15:26:20 creation of Convertparms moved here from K4WindowImpl.
 7-Apr-87 14:30:31 moved out saving of logFile; had problem with holding same handle in two processes.
 7-Apr-87 17:25:45 reduced interline in warning window.
13-Apr-87 14:41:18 checked for empty fist map.from; release bytes of bufferWB.
14-Apr-87 11:27:58 released resources when cancelling document creation.
14-Apr-87 13:13:16 added pre/post/lineHeight.
20-Apr-87 13:53:25 picked up initial font size/style/slant/weight from docProps.
24-Apr-87 16:47:32 new parameter fonProps, fonts moved out of docProps.
27-Apr-87 13:44:28 page break choices.
28-Apr-87 18:22:02 underlining vs italics vs plain choice.
 8-May-87 13:22:14 looks like automatic expansion of bufferW did not work; added code to expand it manually.
```

8-May-87 13:23:18 when font aspect change occurs after first character, all chars between first and the aspect change are dropped; extensive change, including new design of the way to translate the input text according to the mapping options.
14-May-87 9:31:20 insured that titan font is not italics and in [10, 12].
23-Mar-88 15:18:13 mapped tab into tab (was changed to substitute before); fixed the algorithm assigning a paragraph mark on certain lines (it was making a paragraph of the last line of real paragraphs!); made paragraphs of lines less than 5 inches long.
5-Apr-88 14:36:41 when a string of spaces are found inside a line, replace it with a tab if there are more than 5 spaces (rather than 3 spaces); removed the criteria linelength < 5 inches for paragraph decision (caused problems on documents with narrow paragraphs).
3-Aug-88 10:03:47 upgraded to BWS4.3/VP2.0: numerous changes in DocInterchangeDefs and company; added session support because of DocInterchangeDefs.FinishCreation.
/

```
<< File: K4FiledDataImpl.mesa            27-Apr-87 13:00:46

deLaBeaujardiere:OSBU North:Xerox (deLaBeaujardiere.PA)

Copyright (C) 1986 by Xerox Corporation. All rights reserved.

>>

DIRECTORY Containee, Environment, K4, NSFile, NSFileStream,
          Prototype, StarWindowShell, Stream, XString;

K4FiledDataImpl: PROGRAM

IMPORTS NSFileStream, Stream, XString
EXPORTS K4 =

BEGIN
OPEN XS: XString;


<<========== PUBLIC PROCEDURES ==========>>


TypeAndVersion: PUBLIC PROCEDURE
                RETURNS [fileType: NSFile.Type,
                         version:  Prototype.Version] =
  BEGIN
  RETURN [7389325,       -- randomly-chosen number for file type
          19];           -- current version of the application
  END;                   -- The file type must be assigned an
                         -- official number at productization.


LoadFiledData: PUBLIC PROCEDURE [parm: K4.IconParms]
        RETURNS [mismatch: BOOLEAN ← FALSE] =
  BEGIN
  -- This procedure loads into parm the names and values
  -- recorded in the icon file.  Memory allocated here to hold
  -- the bytes for icon/folder/document names
  -- must be freed by the client.

  z: UNCOUNTED ZONE ← parm.heap;
  map, currentMap: K4.Mapping ← NIL;
  signature: CARDINAL ← Signature [];
  stream: Stream.Handle ← NSFileStream.Create [parm.iconFile, FALSE];

  Stream.SetPosition [stream, 0];

  BEGIN ENABLE Stream.EndOfStream => {mismatch ← TRUE;
                                     GOTO Termination};
                                  -- we should not run out of stream
  -- 1. Signature.
  parm.signature ← Stream.GetWord [stream];
  IF parm.signature # signature THEN {mismatch ← TRUE;
                                     GOTO Termination};

  -- 2. General Properties
  parm.genProps ← parm.heap.NEW [K4.GeneralPropertyRecord];
  parm.genProps.iconName ← LoadReaderBody [stream, z];
  parm.genProps.channelSpeed ← VAL[LoadEnumerated[stream]];
  parm.genProps.folderName ← LoadReaderBody [stream, z];
  parm.genProps.others ← LoadReaderBody [stream, z];

  -- 3. Document Properties
  parm.docProps ← parm.heap.NEW [K4.DocumentPropertyRecord];
  parm.docProps.folderName ← LoadReaderBody [stream, z];
  parm.docProps.docName ← LoadReaderBody [stream, z];
  parm.docProps.justification ← LoadBoolean [stream];
  parm.docProps.lineHeight ← VAL[LoadEnumerated [stream]];
  parm.docProps.preLeading ← VAL[LoadEnumerated [stream]];
  parm.docProps.postLeading ← VAL[LoadEnumerated [stream]];
  parm.docProps.underlining ← VAL[LoadEnumerated [stream]];
  parm.docProps.guessMark ← LoadReaderBody [stream, z];
  parm.docProps.dropKeepMark ← VAL[LoadEnumerated [stream]];
  parm.docProps.pageBreak ← VAL[LoadEnumerated [stream]];
  parm.docProps.others ← LoadReaderBody [stream, z];

  -- 4. String Mapping Properties
  parm.mapProps ← parm.heap.NEW [K4.MapPropertyRecord ← [
                       header:  LoadReaderBody [stream, z],
                       number:  Stream.GetWord [stream],
                       map:     NIL,
                       tagSize: NIL]];
  FOR k: CARDINAL IN [1..parm.mapProps.number] DO
    map ← parm.heap.NEW [K4.MappingRecord ← [
                       next: NIL,
                       from: LoadReaderBody [stream, z],
                       to:   LoadReaderBody [stream, z] ]];
    IF parm.mapProps.map = NIL
      THEN parm.mapProps.map ← map
      ELSE currentMap.next ← map;
    currentMap ← map;
    ENDLOOP;

  -- 5. Font Properties
  parm.fonProps ← parm.heap.NEW [K4.FontPropertyRecord];

K4FiledDataImpl.mesa        27-Apr-87 13:00:48 PDT                    1
```

```
    FOR k: CARDINAL IN [0..9] DO
     parm.fonProps.fonts[k].font ← VAL[LoadEnumerated [stream]];
     parm.fonProps.fonts[k].size ← VAL[LoadEnumerated [stream]];
     parm.fonProps.fonts[k].italics ← LoadBoolean [stream];
     parm.fonProps.fonts[k].bold ← LoadBoolean [stream];
     ENDLOOP;

   EXITS  Termination => {};
   END;  -- of ENABLE

   Stream.Delete [stream];
   END;  -- of LoadFiledData


StoreFiledData: PUBLIC PROCEDURE [parm: K4.IconParms] =
  BEGIN
  -- This procedure writed back into the icon file the values
  -- recorded in the property sheet.

  map:       K4.Mapping ← parm.mapProps.map;
  signature: CARDINAL ← Signature[];
  stream:    Stream.Handle ← NSFileStream.Create [parm.iconFile,
                                                  FALSE];

  NSFileStream.SetLength [[stream], 0];  -- truncate old stream

  -- 1. Signature.
  Stream.PutWord [stream, signature];

  -- 2. General Properties
  StoreReaderBody [stream, @parm.genProps.iconName];
  StoreEnumerated [stream, parm.genProps.channelSpeed.ORD];
  StoreReaderBody [stream, @parm.genProps.folderName];
  StoreReaderBody [stream, @parm.genProps.others];

  -- 3. Document Properties
  StoreReaderBody [stream, @parm.docProps.folderName];
  StoreReaderBody [stream, @parm.docProps.docName];
  StoreBoolean [stream, parm.docProps.justification];
  StoreEnumerated [stream, parm.docProps.lineHeight.ORD];
  StoreEnumerated [stream, parm.docProps.preLeading.ORD];
  StoreEnumerated [stream, parm.docProps.postLeading.ORD];
  StoreEnumerated [stream, parm.docProps.underlining.ORD];
  StoreReaderBody [stream, @parm.docProps.guessMark];
  StoreEnumerated [stream, parm.docProps.dropKeepMark.ORD];
  StoreEnumerated [stream, parm.docProps.pageBreak.ORD];
  StoreReaderBody [stream, @parm.docProps.others];

  -- 4. Mapping Properties
  StoreReaderBody [stream, @parm.mapProps.header];
  Stream.PutWord [stream, parm.mapProps.number];
  FOR k: CARDINAL IN [1..parm.mapProps.number]
  WHILE map # NIL DO     -- should diagnose if map is NIL
    StoreReaderBody [stream, @map.from];
    StoreReaderBody [stream, @map.to];
    map ← map.next;
    ENDLOOP;

  -- 5. Font Properties
  FOR k: CARDINAL IN [0..9] DO
    StoreEnumerated [stream, parm.fonProps.fonts[k].font.ORD];
    StoreEnumerated [stream, parm.fonProps.fonts[k].size.ORD];
    StoreBoolean [stream, parm.fonProps.fonts[k].italics];
    StoreBoolean [stream, parm.fonProps.fonts[k].bold];
    ENDLOOP;

  Stream.SendNow [stream];
  Stream.Delete [stream];
  END;  -- of StoreFiledData


FreeIconProps: PUBLIC PROC [props: K4.GeneralProperties,
                            z: UNCOUNTED ZONE] =
  BEGIN
  IF props = NIL THEN RETURN;
  XS.FreeReaderBytes [@props.iconName, z];
  XS.FreeReaderBytes [@props.folderName, z];
  XS.FreeReaderBytes [@props.others, z];
  IF props.tagSize # NIL THEN z.FREE [@props.tagSize];
  z.FREE [@props];
  END;  -- of FreeIconProps


FreeTextProps: PUBLIC PROC [props: K4.DocumentProperties,
                            z: UNCOUNTED ZONE] =
  BEGIN
  IF props = NIL THEN RETURN;
  XS.FreeReaderBytes [@props.folderName, z];
  XS.FreeReaderBytes [@props.docName, z];
  XS.FreeReaderBytes [@props.guessMark, z];
  XS.FreeReaderBytes [@props.others, z];
  IF props.tagSize # NIL THEN z.FREE [@props.tagSize];
  z.FREE [@props];
  END;  -- of FreeTextProps
```

```
FreeMapProps: PUBLIC PROC [props: K4.MapProperties,
                           z: UNCOUNTED ZONE] =
  BEGIN
  map, hold: K4.Mapping;

  IF props = NIL THEN RETURN;
  XS.FreeReaderBytes [@props.header, z];
  FOR map ← props.map, hold WHILE map # NIL DO
    hold ← map.next;
    XS.FreeReaderBytes [@map.from, z];
    XS.FreeReaderBytes [@map.to, z];
    z.FREE [@map];
    ENDLOOP;
  IF props.tagSize # NIL THEN z.FREE [@props.tagSize];
  z.FREE [@props];
  END;  -- of FreeMapProps


FreeFontProps: PUBLIC PROC [props: K4.FontProperties,
                            z: UNCOUNTED ZONE] =
  BEGIN
  IF props = NIL THEN RETURN;
  IF props.tagSize # NIL THEN z.FREE [@props.tagSize];
  z.FREE [@props];
  END;  -- of FreeFontProps



<<==========  PRIVATE PROCEDURES  ==========>>


LoadBoolean: PROC [stream: Stream.Handle] RETURNS [BOOLEAN] =
  INLINE {RETURN [Stream.GetWord[stream] # 0]};


StoreBoolean: PROC [stream: Stream.Handle, boolean: BOOLEAN] =
  INLINE {Stream.PutWord [stream, boolean.ORD]};


LoadEnumerated: PROC [stream: Stream.Handle]
            RETURNS [Environment.Word] =
  INLINE {RETURN [Stream.GetWord[stream]]};


StoreEnumerated: PROC [stream: Stream.Handle,
                       value: Environment.Word] =
  INLINE {Stream.PutWord [stream, value]};


LoadReaderBody: PROC [stream: Stream.Handle, z: UNCOUNTED ZONE]
        RETURNS [rb: XS.ReaderBody] =
  BEGIN
  length: CARDINAL ← Stream.GetWord [stream];
  wb: XS.WriterBody ← XS.NewWriterBody [length, z];

  FOR k: CARDINAL IN [0..length) DO
    XS.AppendChar [@wb, Stream.GetWord[stream]];
    ENDLOOP;
  rb ← XS.CopyToNewReaderBody[XS.ReaderFromWriter[@wb], z];
        -- the bytes allocated here must be released by the client

  XS.FreeWriterBytes[@wb];
  END;


StoreReaderBody: PROC [stream: Stream.Handle, r: XS.Reader] =
  BEGIN
  length:  CARDINAL ← XS.CharacterLength[r];
  lambdaB: XS.ReaderBody ← XS.Dereference [r];

  Stream.PutWord [stream, length];
  FOR k: CARDINAL IN [0..length) DO
    Stream.PutWord [stream, XS.Lop[@lambdaB]];
    ENDLOOP;
  END;


Signature: PROC RETURNS [CARDINAL] = INLINE
   -- calculates some number likely to change whenever a change is
   -- made to K4.IconParmsRecord.  This is to try to catch instances
   -- where the user has a Kurzweil icon not matching the current
   -- software version.
  {RETURN [     SIZE [K4.DocumentPropertyRecord] +
            3 * SIZE [K4.MapPropertyRecord]   +
            5 * SIZE [K4.FontPropertyRecord]  +
            7 * SIZE [K4.GeneralPropertyRecord]  +
           11 * TypeAndVersion[].version]};



END.  -- of K4FiledDataImpl

10-Mar-87  9:47:45 created from code moved from K4IconImpl.
13-Mar-87 10:11:40 added options and modified others.
```

```
16-Mar-87  9:43:36 added signature.
17-Mar-87 14:12:24 added character mapping.
19-Mar-87 10:55:28 implemented TypeAndVersion.
20-Mar-87 10:25:13 version 14: mapProps.number starts at 0 instead of 1.
31-Mar-87 12:30:25 version 16: called ReduceList before writing map.number and only if not creating prototype.
31-Mar-87 13:21:36 filed data initialization moved to K4IconImpl.
14-Apr-87 14:02:42 version 17: paragraph properties.
24-Apr-87 16:15:03 version 18: dropped canvas, moved fonts to separate prop sheet.
27-Apr-87 13:00:24 version 19: folder name for transmissions, pageBreak choice.
/
```

```
<< File: K4IconImpl.mesa        27-Apr-87 13:06:14

deLaBeaujardiere:OSBU North:Xerox (deLaBeaujardiere.PA)

Copyright (C) 1986 by Xerox Corporation. All rights reserved.

>>

DIRECTORY Atom, Containee,
          Display, Environment,
          Heap, K4,
          NSFile,
          Prototype,
          SimpleTextDisplay, SimpleTextFont,
          StarWindowShell,
          XChar, XString;

K4IconImpl: PROGRAM

IMPORTS Atom, Containee, Display,
        Heap, K4, NSFile,
        Prototype,
        SimpleTextDisplay, SimpleTextFont,
        XString =

BEGIN
        OPEN XS: XString;

<<=====  TYPE DEFINITIONS AND CONSTANTS =====>>

IconRecord: TYPE = RECORD [
  heap:           UNCOUNTED ZONE,
  oldGenericProc: Containee.GenericProc ← NIL,
  keyOpen:        Atom.ATOM,
  keyProps:       Atom.ATOM,
  smallIcon:      XS.Character ← XChar.not];


<<=====  GLOBAL VARIABLES =====>>

icon: LONG POINTER TO IconRecord ← NIL;


<<=====  PROCEDURES  =====>>


InstallK4Icon: PROCEDURE =
  BEGIN
  heap:         UNCOUNTED ZONE ← Heap.Create[1];
  rows:         CARDINAL = 13;  -- 13 rows in small icon picture
  smallPicture: PACKED ARRAY[0..rows) OF WORD ← [
    177770B, 100010B, 125010B, 100010B, 177770B, 100010B, 137750B,
    110110B, 170170B, 010100B, 010700B, 010600B, 017400B];
  iconName:     XS.ReaderBody ← XS.FromSTRING ["Kurzweil 4000"L];
  impl:         Containee.Implementation;
  iconFileType: NSFile.Type;
  currentVersion: Prototype.Version;

  [iconFileType, currentVersion] ← K4.TypeAndVersion[];
  icon ← heap.NEW[IconRecord];
  icon.heap ← heap;
  icon.keyOpen ← Atom.MakeAtom["Open"L];
  icon.keyProps ← Atom.MakeAtom["Props"L];

  IF Prototype.Find [type: iconFileType,
                     version: currentVersion] = NSFile.nullReference THEN
      MakeNewVersion [@iconName, iconFileType, currentVersion, heap];

  impl ← Containee.GetImplementation [iconFileType];

  icon.oldGenericProc ← impl.genericProc;
  icon.smallIcon ← SimpleTextFont.AddClientDefinedCharacter[
                                width: 13,
                                height: rows,
                                bitsPerLine: 16,
                                bits: @smallPicture];

  impl.genericProc ← GenericProc;
  impl.name ← iconName;
  impl.smallPictureProc ← PaintSmallIcon;
  impl.pictureProc ← PaintBigIcon;

  [] ← Containee.SetImplementation [iconFileType, impl];
  END;  -- of InstallK4Icon


MakeNewVersion: PROC [iconName: XS.Reader,
                      iconType: NSFile.Type,
                      iconVersion: Prototype.Version,
                      z: UNCOUNTED ZONE] =
  BEGIN
  question: LONG STRING ← "@"L;
  parm: K4.IconParms ← z.NEW[K4.IconParmsRecord];

  parm.heap ← z;
```

```
    parm.genProps ← z.NEW [K4.GeneralPropertyRecord ← [
                  iconName:      iconName↑,
                  channelSpeed:  ninetySix,
                  folderName:    XS.FromSTRING ["Kurzweil Transmissions"L],
                  others:        XS.FromSTRING [
          "Bits per Char: 8, Stop bits: 1, Parity: None, Asynchronous"L]]];
    parm.docProps ← z.NEW [K4.DocumentPropertyRecord ← [
                  folderName:    XS.FromSTRING ["Kurzweil Documents"L],
                  docName:       XS.FromSTRING ["Interpreted Document"L],
                  justification: TRUE,
                  lineHeight:    single,
                  preLeading:    single,
                  postLeading:   singleHalf,
                  underlining:   underline,
                  guessMark:     XS.FromSTRING [question],
                  dropKeepMark:  drop,
                  pageBreak:     drop,
                  others:        XS.FromSTRING [
        "End Line: HOD, End Page: HOC, End Column/Para: None, Horizontal: Preserve, Vertical: Preserve Breaks"L]]];
    parm.mapProps ← z.NEW [K4.MapPropertyRecord ← [
                  header:        XS.FromSTRING ["Character Substitutions"L],
                  number:        0,
                  map: NIL]];
    parm.fonProps ← z.NEW [K4.FontPropertyRecord ← [
                  fonts: [[classic, ten,      FALSE, FALSE],
                          [classic, ten,      FALSE, TRUE ],
                          [classic, ten,      TRUE,  FALSE],
                          [classic, eight,    FALSE, FALSE],
                          [classic, eight,    FALSE, TRUE ],
                          [classic, eight,    TRUE,  FALSE],
                          [classic, twelve,   FALSE, FALSE],
                          [classic, twelve,   FALSE, TRUE ],
                          [classic, twelve,   TRUE,  FALSE],
                          [classic, fourteen, FALSE, FALSE]]]];

    parm.iconFile ← Prototype.Create [name: iconName,
                                      type: iconType,
                                      subtype: 0,
                                      version: iconVersion,
                                      isDirectory: FALSE];
    Prototype.PurgeOldVersions [type: iconType,
                                subtype: 0,
                                current: iconVersion];
    K4.StoreFiledData [parm];
    z.FREE [@parm.genProps];
    z.FREE [@parm.docProps];
    z.FREE [@parm.mapProps] ;
    z.FREE [@parm.fonProps] ;
    NSFile.Close [parm.iconFile];
    z.FREE [@parm];
    END;  -- of MakeNewVersion

GenericProc: Containee.GenericProc =
    -- defined as PROC [atom: Atom.ATOM,
    --                  data: DataHandle,
    --                  changeProc: ChangeProc ← NIL,
    --                  changeProcData: LONG POINTER ← NIL,
    --          RETURNS [LONG UNSPECIFIED]
    BEGIN
    shell: LONG POINTER;

    shell ← SELECT atom FROM
              icon.keyOpen  => K4.OpenWindow [data, changeProc,
                                              changeProcData,
                                              icon.smallIcon],
              icon.keyProps => K4.OpenPSheet [data, changeProc,
                                              changeProcData],
              ENDCASE => icon.oldGenericProc [atom, data, changeProc,
                                              changeProcData];

    RETURN [shell];
    END;  -- of GenericProc


Err: PROCEDURE[message: LONG STRING] = BEGIN
    msgRB: XString.ReaderBody ← XString.FromSTRING[message];
    Containee.Error [@msgRB];
    END;  -- of Err


PaintSmallIcon: Containee.SmallPictureProc = {RETURN[icon.smallIcon]};


PaintBigIcon: Containee.PictureProc = BEGIN
    widthInPixels: CARDINAL = 65;
    widthInWords: CARDINAL = 5;
    heightInPixels: CARDINAL = 60;
    IF new=garbage THEN RETURN
    ELSE BEGIN
      fileName: XS.ReaderBody;
      cacheTicket: Containee.Ticket;
      mask: ARRAY[0..widthInWords*heightInPixels) OF WORD ← [
        037777B, 177777B, 177777B, 177776B, 000000B,
        077777B, 177777B, 177777B, 177777B, 000000B,
        177777B, 177777B, 177777B, 177777B, 100000B,
        177777B, 177777B, 177777B, 177777B, 100000B,
```

```
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      177777B, 177777B, 177777B, 177777B, 100000B,
      077777B, 177777B, 177777B, 177777B, 000000B,
      037777B, 177777B, 177777B, 177776B, 000000B,
      000077B, 177777B, 177777B, 177000B, 000000B,
      000077B, 177777B, 177777B, 177000B, 000000B,
      000077B, 177777B, 177777B, 177000B, 000000B,
      000077B, 177777B, 177777B, 177000B, 000000B,
      000077B, 177777B, 177777B, 177000B, 000000B,
      000077B, 177777B, 177777B, 177000B, 000000B,
      000077B, 177777B, 177777B, 177000B, 000000B,
      000077B, 177777B, 177777B, 177000B, 000000B,
      000077B, 177777B, 177777B, 177000B, 000000B,
      000077B, 177777B, 177777B, 177000B, 000000B,
      000077B, 177777B, 177777B, 177000B, 000000B,
      000077B, 177777B, 177777B, 177000B, 000000B,
      000077B, 177777B, 177777B, 177000B, 000000B,
      000077B, 177777B, 177777B, 177000B, 000000B,
      000077B, 177777B, 177777B, 177000B, 000000B,
      000077B, 177777B, 177777B, 177000B, 071145B,
      000077B, 177777B, 177777B, 176000B, 000022B,
      000077B, 177777B, 177777B, 174000B, 007144B,
      000077B, 177777B, 177777B, 170000B, 000022B,
      000077B, 177777B, 177777B, 160000B, 002312B,
      000077B, 177777B, 177777B, 140000B, 000005B,
      000077B, 177777B, 177777B, 100000B, 000005B,
      000077B, 177777B, 177777B, 000000B, 000025B,
      000077B, 177777B, 177776B, 000000B, 064556B];

  IF new=reference OR new=referenceHighlighted THEN new ← normal;
  box.place.x ← box.place.x + 2;
  box.dims.w ← widthInPixels;
  box.place.y ← box.place.y + 8;
  box.dims.h ← heightInPixels;

  Display.Bitmap[
    window: window,
    box: box,
    address: [word: @mask, bit: 0],
    bitmapBitWidth: widthInWords*Environment.bitsPerWord,
    flags: SELECT new FROM
      highlighted, referenceHighlighted => [  --for inverting picture
        disjoint: TRUE,
        srcFunc: null,
        dstFunc: or],
      ENDCASE => [
        disjoint: TRUE,
        srcFunc: complement,
        dstFunc: and]];

  IF new#ghost THEN BEGIN
    picture: ARRAY[0..widthInWords*heightInPixels) OF WORD ← [
      037777B, 177777B, 177777B, 177776B, 000000B,
      077777B, 177777B, 177777B, 177777B, 000000B,
      160000B, 000000B, 000000B, 000003B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
      140000B, 000000B, 000000B, 000001B, 100000B,
```

```
        140000B, 000000B, 000000B, 000001B, 100000B,
        140000B, 000000B, 000000B, 000001B, 100000B,
        140000B, 000000B, 000000B, 000001B, 100000B,
        140000B, 000000B, 000000B, 000001B, 100000B,
        177777B, 177777B, 177777B, 177777B, 100000B,
        140000B, 000000B, 000000B, 000001B, 100000B,
        140000B, 000000B, 000000B, 000001B, 100000B,
        140000B, 000000B, 000000B, 000001B, 100000B,
        140000B, 000000B, 000000B, 000001B, 100000B,
        140000B, 000000B, 000000B, 000001B, 100000B,
        141777B, 177777B, 177777B, 177741B, 100000B,
        141760B, 000000B, 000000B, 003741B, 100000B,
        140020B, 000000B, 000000B, 002001B, 100000B,
        140020B, 052525B, 052525B, 002001B, 100000B,
        140020B, 025252B, 125252B, 002001B, 100000B,
        140020B, 000000B, 000000B, 002001B, 100000B,
        160020B, 000000B, 000000B, 002003B, 100000B,
        077760B, 052525B, 052525B, 003777B, 000000B,
        037760B, 025252B, 125252B, 003776B, 000000B,
        000060B, 000000B, 000000B, 003000B, 000000B,
        000060B, 000000B, 000000B, 003000B, 000000B,
        000060B, 052525B, 052525B, 003000B, 000000B,
        000060B, 025252B, 125252B, 003000B, 000000B,
        000060B, 000000B, 000000B, 003000B, 000000B,
        000060B, 000000B, 000000B, 003000B, 000000B,
        000060B, 052525B, 052525B, 003000B, 000000B,
        000060B, 025252B, 125252B, 003000B, 000000B,
        000060B, 000000B, 000000B, 003000B, 000000B,
        000060B, 000000B, 000000B, 003000B, 000000B,
        000060B, 000000B, 000000B, 003000B, 000000B,
        000060B, 000000B, 000007B, 177000B, 000000B,
        000060B, 000000B, 000004B, 016000B, 000000B,
        000060B, 000000B, 000004B, 034000B, 000000B,
        000060B, 000000B, 000004B, 070000B, 000000B,
        000060B, 000000B, 000004B, 160000B, 000000B,
        000060B, 000000B, 000005B, 140000B, 000000B,
        000060B, 000000B, 000007B, 100000B, 000000B,
        000077B, 177777B, 177777B, 000000B, 000000B,
        000077B, 177777B, 177776B, 000000B, 000000B];

      Display.Bitmap[
        window: window,
        box: box,
        address: [word: @picture, bit: 0],
        bitmapBitWidth: widthInWords*Environment.bitsPerWord,
        flags: SELECT new FROM
          highlighted, referenceHighlighted => [
            disjoint: TRUE,
            srcFunc: complement,
            dstFunc: and],
          ENDCASE => [
            disjoint: TRUE,
            srcFunc: null,
            dstFunc: or]];
      END;  --of drawing in the picture per se (not a ghost)

    [fileName, cacheTicket] ← Containee.GetCachedName[data];
    [] ← SimpleTextDisplay.StringIntoWindow[
      string: @fileName,
      window: window,
      place: [x: box.place.x + 3, y: box.place.y + 3],
      lineWidth: widthInPixels - 6,
      maxNumberOfLines: 1,
      wordBreak: FALSE,
      flags: SELECT new FROM
          highlighted, referenceHighlighted => [
            disjoint: TRUE,
            srcFunc: complement,
            dstFunc: and],
          ENDCASE => Display.paintFlags];
    Containee.ReturnTicket[cacheTicket];
    END;  --of drawing; i.e., not garbage
  END;  -- of PaintBigIcon



<<========== MAIN LINE CODE ==========>>

InstallK4Icon[];

END.  -- of K4IconImpl
14-Jan-87 15:03:40 created from DestTextIconImpl.
 6-Feb-87 15:35:08 changed initial preset of pSheet.
10-Mar-87  9:50:14 removed Courier usage, moved Load/StoreFiledData to K4FiledDataImpl.
19-Mar-87 11:05:26 used TypeAndVersion to get fileType/version.
31-Mar-87 13:50:53 moved initialization of icon main data store here from K4FiledDataImpl.
14-Apr-87 13:53:19 added paragraph properties.
24-Apr-87 16:22:04 dropped canvas and ArtScan, moved fonts to separate prop sheet.
27-Apr-87 13:05:58 foldername for transmissions, page break.
/
```

```
<< File: K4PSheetImpl.mesa        28-Apr-87 16:59:05

deLaBeaujardiere:OSBU North:Xerox (deLaBeaujardiere.PA)

Copyright (C) 1986 by Xerox Corporation. All rights reserved.

>>

DIRECTORY
        Attention, Containee,
        Environment, FormWindow, Heap,
        K4,
        NSFile, NSString,
        PropertySheet,
        SimpleTextDisplay,
        StarWindowShell, Window, XString;

K4PSheetImpl: PROGRAM

IMPORTS Attention,
        FormWindow, Heap, K4, NSFile, NSString,
        PropertySheet,
        SimpleTextDisplay, XString

EXPORTS K4 =

BEGIN

OPEN FW: FormWindow,
     K4,
     XS: XString;



<<========== PUBLIC PROCEDURES ==========>>


OpenPSheet: PUBLIC PROCEDURE [iconData: Containee.DataHandle,
                              changeProc: Containee.ChangeProc,
                              changeProcData: LONG POINTER]
                    RETURNS [StarWindowShell.Handle] =
  BEGIN
  zone:          UNCOUNTED ZONE ← Heap.Create [1];
                              -- deleted by DoPropsCommands
  pSheetPlace:   Window.Place ← PropertySheet.nullPlace;
  pSheetSize:    Window.Dims ← [0, 0];
  pSheetName:    XS.ReaderBody ← XS.FromSTRING [
                        "Kurzweil 4000 Properties"L];
  versionMismatch: BOOLEAN;
  iconParms:     K4.IconParms ← zone.NEW [K4.IconParmsRecord ← [
                        heap: zone,
                        iconData: iconData,
                        changeProc: changeProc,
                        changeProcData: changeProcData,
                        iconFile: NSFile.OpenByReference
                                    [iconData.reference] ]];
                        -- freed by TakeDownPSheet

  versionMismatch ← K4.LoadFiledData [iconParms];
                              -- unloaded by TakeDownPSheet
  IF versionMismatch THEN
    BEGIN
    Msg ["Obsolete icon does not work with new software"L];
    NSFile.Close [iconParms.iconFile];
    zone.FREE [@iconParms];
    Heap.Delete [zone];
    RETURN [StarWindowShell.nullHandle];
    END;

  -- Create the Property Sheet.
  iconParms.propSheet ← PropertySheet.CreateLinked [
                linkWindowItems: MakeLinkProps,
                linkWindowItemsLayout: LayLinkItem,
                formWindowItems: MakeTextProps,        -- first shown
                formWindowItemsLayout: LayTextProps,   -- is text sheet
                menuItemProc: SetTextProps,
                menuItems: [done: TRUE, cancel: TRUE],
                title: @pSheetName,
                size: pSheetSize,
                placeToDisplay: pSheetPlace,
                afterTakenDownProc: TakeDownPSheet,
                clientData: iconParms];

  RETURN [iconParms.propSheet];
  END;  -- of OpenPSheet


OpenDocOptionSheet: PUBLIC PROC [
                        convertParm: K4.ConvertParms,
                        takeDown: PropertySheet.MenuItemProc] =
  BEGIN
  pSheetName: XS.ReaderBody ← XS.FromSTRING ["Document Options"L];
  pSize: Window.Dims ← [500, 400];
  pPlace: Window.Place ← [500, 50];

  convertParm.optionSheet  ← PropertySheet.CreateLinked [
```

```
                linkWindowItems: MakeLinkOptions,
                linkWindowItemsLayout: LayLinkItem,
                formWindowItems: MakeTextOptions,       -- first shown
                formWindowItemsLayout: LayTextOptions, -- is text
                menuItemProc: SetTextOptions,
                menuItems: [start: TRUE, cancel: TRUE],
                title: @pSheetName,
                size: pSize,
                placeToDisplay: pPlace,
                afterTakenDownProc: takeDown,
                clientData: convertParm];
  END;  -- of OpenDocOptionSheet


<<==========  PRIVATE PROCEDURES FOR DOCUMENT SUBWINDOW  ==========>>


MakeTextOptions: FW.MakeItemsProc =
  BEGIN
  parm: K4.ConvertParms ← LOOPHOLE[clientData];
  MakeTextItems [window, parm.docProps, parm.zone];
  END;  -- of MakeTextOptions


MakeTextProps: FW.MakeItemsProc =
  BEGIN
  iconParm: K4.IconParms ← LOOPHOLE[clientData];
  MakeTextItems [window, iconParm.docProps, iconParm.heap];
  END;  -- of MakeTextProps


MakeTextItems: PROC [window: Window.Handle,
                     docProps: K4.DocumentProperties,
                     z: UNCOUNTED ZONE] =
  BEGIN
  label: XS.ReaderBody;

  underliningChoices: FW.ChoiceItems ← DESCRIPTOR [uChoices];
  uChoices: ARRAY [0..3) OF FW.ChoiceItem ← [
              [string [choiceNumber: Underlining.underline.ORD,
                       string: XString.FromSTRING["Underline"L]]],
              [string [choiceNumber: Underlining.italics.ORD,
                       string: XString.FromSTRING["Italics"L]]],
              [string [choiceNumber: Underlining.plain.ORD,
                       string: XString.FromSTRING["Plain"L]]] ];

  lineChoices: FW.ChoiceItems ← DESCRIPTOR [lChoices];
  lChoices: ARRAY [0..4) OF FW.ChoiceItem ← [
              [string [choiceNumber: LineSpacing.single.ORD,
                       string: XString.FromSTRING["Single"L]]],
              [string [choiceNumber: LineSpacing.singleHalf.ORD,
                       string: XString.FromSTRING["1 1/2"L]]],
              [string [choiceNumber: LineSpacing.double.ORD,
                       string: XString.FromSTRING["Double"L]]],
              [string [choiceNumber: LineSpacing.triple.ORD,
                       string: XString.FromSTRING["Triple"L]]] ];

  guessChoices: FW.ChoiceItems ← DESCRIPTOR [gChoices];
  gChoices: ARRAY [0..2) OF FW.ChoiceItem ← [
              [string [choiceNumber: DropKeep.drop.ORD,
                       string: XString.FromSTRING["Drop"L]]],
              [string [choiceNumber: DropKeep.keep.ORD,
                       string: XString.FromSTRING["Keep"L]]] ];

  pageBreakChoices: FW.ChoiceItems ← DESCRIPTOR [pChoices];
  pChoices: ARRAY [0..3) OF FW.ChoiceItem ← [
              [string [choiceNumber: Underlining.underline.ORD,
                       string: XString.FromSTRING["None"L]]],
              [string [choiceNumber: Underlining.italics.ORD,
                       string: XString.FromSTRING["Every Page"L]]],
              [string [choiceNumber: Underlining.plain.ORD,
                       string: XString.FromSTRING["Unfilled Pages"L]]] ];

  IF docProps.tagSize = NIL THEN
    docProps.tagSize ← z.NEW[TextTagSizes];
                                        -- released by FreeTextProps

  docProps.tagSize[folderName] ← Measure [@label, "Document Folder Name"L];
  FW.MakeTextItem [window: window, myKey: K4.TextItems.folderName.ORD,
                   tag: @label,
                   initString: @docProps.folderName,
                   width: 200];

  docProps.tagSize[docName] ← Measure [@label, "Document Name"L];
  FW.MakeTextItem [window: window, myKey: K4.TextItems.docName.ORD,
                   tag: @label,
                   initString: @docProps.docName,
                   width: 200];

  docProps.tagSize[justification] ← Measure [@label,
                                             "Paragraph Right Edge"L];
  FW.MakeBooleanItem[window: window,
    myKey: K4.TextItems.justification.ORD, tag: @label,
    label: [string [XString.FromSTRING["Justify"L]]],
    initBoolean: docProps.justification];
```

```
    docProps.tagSize[lineHeight] ← Measure [@label,
                                            "Paragraph Line Height"L];
    FW.MakeChoiceItem[window: window,
      myKey: K4.TextItems.lineHeight.ORD, tag: @label,
      values: lineChoices,
      initChoice: VAL[docProps.lineHeight]];

    docProps.tagSize[preLeading] ← Measure [@label,
                                            "Lines before Paragraph"L];
    FW.MakeChoiceItem[window: window,
      myKey: K4.TextItems.preLeading.ORD, tag: @label,
      values: lineChoices,
      initChoice: VAL[docProps.preLeading]];

    docProps.tagSize[postLeading] ← Measure [@label,
                                            "Lines after Paragraph"L];
    FW.MakeChoiceItem[window: window,
      myKey: K4.TextItems.postLeading.ORD, tag: @label,
      values: lineChoices,
      initChoice: VAL[docProps.postLeading]];

    docProps.tagSize[underlining] ← Measure [@label, "Underlining"L];
    FW.MakeChoiceItem[window: window,
      myKey: K4.TextItems.underlining.ORD, tag: @label,
      values: underliningChoices,
      initChoice: VAL[docProps.underlining]];

    docProps.tagSize[guessMark] ← Measure [@label,
                                           "Questionable Character"L];
    FW.MakeTextItem [window: window, myKey: K4.TextItems.guessMark.ORD,
                     tag: @label,
                     initString: @docProps.guessMark,
                     width: 40];
    FW.MakeChoiceItem[window: window,
      myKey: K4.TextItems.dropKeepMark.ORD, tag: NIL,
      values: guessChoices,
      initChoice: VAL[docProps.dropKeepMark]];

    docProps.tagSize[pageBreak] ← Measure [@label, "Page Break"L];
    FW.MakeChoiceItem[window: window,
      myKey: K4.TextItems.pageBreak.ORD, tag: @label,
      values: pageBreakChoices,
      initChoice: VAL[docProps.pageBreak]];

    docProps.tagSize[others] ← Measure [@label,
                                        "Required Tailor Choices"L];
    FW.MakeTextItem [window: window, myKey: K4.TextItems.others.ORD,
                     tag: @label,
                     initString: @docProps.others,
                     width: 300, boxed: FALSE, readOnly: TRUE];
    END;  -- of MakeTextItems


  LayTextOptions: FW.LayoutProc =
    BEGIN
    parm: K4.ConvertParms ← LOOPHOLE[clientData];
    LayTextItems [window, parm.docProps, parm.zone];
    END;  -- of LayTextOptions


  LayTextProps: FW.LayoutProc =
    BEGIN
    iconParm: K4.IconParms ← LOOPHOLE[clientData];
    LayTextItems [window, iconParm.docProps, iconParm.heap];
    END;  -- of LayTextProps


  LayTextItems: PROC [window: Window.Handle,
                      docProps: K4.DocumentProperties,
                      z: UNCOUNTED ZONE] =
    BEGIN
    margin: CARDINAL = 5;
    maxTag: CARDINAL ← 0;
    line: FW.Line;
    ts: LONG POINTER TO TextTagSizes ← docProps.tagSize;  -- accelerator

    LaySingleItem: PROC [itemKey: K4.TextItems,
                         tagSize, interline: CARDINAL] =
          BEGIN
          IF interline > 0 THEN
                  line ← FW.AppendLine[window: window,
                            spaceAboveLine: interline];
          FW.AppendItem[window: window, line: line,
                  item: itemKey.ORD,
                  preMargin: IF interline = 0 THEN 8
                             ELSE Prespace[tagSize, maxTag, margin]];
          END;  -- of LaySingleItem

    FOR k: TextItems IN TextItems DO
      maxTag ← MAX [maxTag, ts[k]];
      ENDLOOP;

    LaySingleItem[folderName, ts[folderName], 6];
    LaySingleItem[docName, ts[docName], 6];
    LaySingleItem[justification, ts[justification], 6];
    LaySingleItem[lineHeight, ts[lineHeight], 6];
```

```
    LaySingleItem[preLeading, ts[preLeading], 6];
    LaySingleItem[postLeading, ts[postLeading], 6];
    LaySingleItem[underlining, ts[underlining], 6];
    LaySingleItem[guessMark, ts[guessMark], 6];
    LaySingleItem[dropKeepMark, ts[dropKeepMark], 0];
    LaySingleItem[pageBreak, ts[pageBreak], 6];
    LaySingleItem[others, ts[others], 12];

    FW.Repaint [window];
    END;  -- of LayTextItems


  SetTextOptions: PropertySheet.MenuItemProc =
    BEGIN
    parm: K4.ConvertParms ← LOOPHOLE[clientData];
    RETURN [SetTextItems [formWindow, parm.docProps, parm.zone]];
    END;  -- of SetTextOptions


  SetTextProps: PropertySheet.MenuItemProc =
    BEGIN
    iconParm: K4.IconParms ← LOOPHOLE[clientData];
    RETURN [SetTextItems [formWindow, iconParm.docProps, iconParm.heap]];
    END;  -- of SetTextProps


  SetTextItems: PROC [window: Window.Handle,
                      docProps: K4.DocumentProperties,
                      z: UNCOUNTED ZONE]
              RETURNS [BOOLEAN] =
    BEGIN

    Boolean: PROCEDURE [item: TextItems] RETURNS [BOOLEAN] = INLINE
          {RETURN [FW.GetBooleanItemValue [window, item.ORD]]};

    Choice: PROCEDURE [item: TextItems] RETURNS [CARDINAL] = INLINE
          {RETURN [FW.GetChoiceItemValue [window, item.ORD] ]};

    IF NOT FW.HasAnyBeenChanged [window] THEN RETURN [TRUE];

    IF FW.HasBeenChanged [window, TextItems.folderName.ORD] THEN
      BEGIN
      docProps.folderName ← UpdateBody [window,
                                        TextItems.folderName.ORD,
                                        @docProps.folderName, z];
      IF XS.Empty [@docProps.folderName] THEN
        BEGIN
        Msg ["Folder name cannot be empty"L];
        RETURN [FALSE];
        END;
      END;

    IF FW.HasBeenChanged [window, TextItems.docName.ORD] THEN
      BEGIN
      docProps.docName ← UpdateBody [
                                window, TextItems.docName.ORD,
                                @docProps.docName, z];
      IF XS.Empty [@docProps.docName] THEN
        BEGIN
        Msg ["Document name cannot be empty"L];
        RETURN [FALSE];
        END;
      END;

      docProps.justification ← Boolean [K4.TextItems.justification];
      docProps.lineHeight    ← VAL [Choice [K4.TextItems.lineHeight]];
      docProps.preLeading    ← VAL [Choice [K4.TextItems.preLeading]];
      docProps.postLeading   ← VAL [Choice [K4.TextItems.postLeading]];
      docProps.underlining   ← VAL [Choice [K4.TextItems.underlining]];
      docProps.dropKeepMark  ← VAL [Choice [K4.TextItems.dropKeepMark]];
      docProps.pageBreak     ← VAL [Choice [K4.TextItems.pageBreak]];

    IF FW.HasBeenChanged [window, TextItems.guessMark.ORD] THEN
      docProps.guessMark ← UpdateBody [
                                window, TextItems.guessMark.ORD,
                                @docProps.guessMark, z];

    RETURN [TRUE];
    END;  -- of SetTextItems


<<==========  PRIVATE PROCEDURES FOR MAPPING SUBWINDOW  ==========>>

<< Road map to their usage:
        MakeMapOptions          => MakeMapItems
                                => MakeNextMapItems
        MakeMapProps            => MakeMapItems
                                => MakeNextMapItems
        LayMapOptions           => LayMapItems
        LayMapProps             => LayMapItems
        AddNextMapOptions       => MakeNextMapItems
                                => LayNextMapItems
        AddNextMapProps         => MakeNextMapItems
                                => LayNextMapItems              >>
```

```
MakeMapOptions: FW.MakeItemsProc =
  BEGIN
  parm: K4.ConvertParms ← LOOPHOLE[clientData];

  MakeMapItems [window, AddNextMapOptions, parm.mapProps, parm.zone];

  -- If no mapping option exists yet, must display an empty pair
  IF parm.mapProps.number = 0 THEN
      MakeNextMapItems [window, AddNextMapOptions,
                        parm.mapProps, parm.zone];
  END;  -- of MakeMapOptions


MakeMapProps: FW.MakeItemsProc =
  BEGIN
  ip: K4.IconParms ← LOOPHOLE[clientData];

  MakeMapItems [window, AddNextMapProps, ip.mapProps, ip.heap];
  IF ip.mapProps.number = 0 THEN
      MakeNextMapItems [window, AddNextMapProps,
                        ip.mapProps, ip.heap];
  END;  -- of MakeMapProps


MakeMapItems: PROC [window: Window.Handle,
                    nextOut: FW.NextOutOfProc,
                    mapProps: K4.MapProperties,
                    z: UNCOUNTED ZONE] =
  BEGIN
  -- Create window items for the header and the
  -- existing from/to pairs.
  label: XS.ReaderBody;
  map: K4.Mapping ← mapProps.map;  -- may be NIL if no items yet.

  IF mapProps.tagSize = NIL THEN
    mapProps.tagSize ← z.NEW[MapTagSizes]; -- released by FreeMapProps

  mapProps.tagSize[header] ← 0;
  FW.MakeTextItem[window: window,
                  myKey: 0,
                  tag: NIL,
                  initString: @mapProps.header,
                  width: 300, boxed: FALSE, readOnly: TRUE];

  mapProps.tagSize[from] ← Measure[@label, "          "L];
  mapProps.tagSize[to] ← 0;

  FOR k: CARDINAL IN [1..mapProps.number] DO
    FW.MakeTextItem[window: window, tag: @label, width: 50,
                    myKey: 2*k - 1, initString: @map.from];
    FW.MakeTextItem[window: window, tag: NIL, width: 50,
                    myKey: 2*k, initString: @map.to,
                    nextOutOfProc: nextOut];
    map ← map.next;
    ENDLOOP;
  END;  -- of MakeMapItems


MakeNextMapItems: PROC [window: Window.Handle,
                        nextOut: FW.NextOutOfProc,
                        mapProps: K4.MapProperties,
                        z: UNCOUNTED ZONE] =
  BEGIN
  -- Add an entry in mapProps, and create
  -- the corresponding pair of window items.

  empty: XS.ReaderBody ← XS.FromSTRING [""L];
  map:   K4.Mapping ← z.NEW [K4.MappingRecord ← [
                        from: XS.CopyToNewReaderBody [@empty, z],
                        to:   XS.CopyToNewReaderBody [@empty, z],
                        next: NIL]];

  IF mapProps.map = NIL THEN mapProps.map ← map
  ELSE
    BEGIN
    FOR last: K4.Mapping ← mapProps.map, last.next DO
      IF last.next = NIL THEN {last.next ← map; EXIT};
      ENDLOOP;
    END;

  mapProps.number ← mapProps.number + 1;    -- update number of pairs

  FW.MakeTextItem[window: window, tag: NIL, width: 50,
                  myKey: 2*mapProps.number - 1,
                  initString: @map.from];
  FW.MakeTextItem[window: window, tag: NIL, width: 50,
                  myKey: 2*mapProps.number,
                  initString: @map.to,
                  nextOutOfProc: nextOut];
  END;  -- of MakeNextMapItems


LayMapOptions: FW.LayoutProc =
  BEGIN
  parm: K4.ConvertParms ← LOOPHOLE[clientData];
  LayMapItems [window, parm.mapProps, parm.zone];
  END;  -- of LayMapOptions
```

```
  LayMapProps: FW.LayoutProc =
    BEGIN
    iconParm: K4.IconParms ← LOOPHOLE[clientData];
    LayMapItems [window, iconParm.mapProps, iconParm.heap];
    END;  -- of LayMapProps


  LayMapItems: PROC [window: Window.Handle,
                     mapProps: K4.MapProperties,
                     z: UNCOUNTED ZONE] =
    BEGIN
    margin: CARDINAL = 5;
    spaceBetweenLines: CARDINAL = 3;
    preMargin, maxTag: CARDINAL ← 0;
    line: FW.Line;

    FOR k: MapItems IN MapItems DO
      maxTag ← MAX [maxTag, mapProps.tagSize[k]];
      ENDLOOP;

    line ← FW.AppendLine[window: window,
                         spaceAboveLine: 2 * spaceBetweenLines];
    FW.AppendItem[window: window, line: line,
                  item: K4.MapItems.header.ORD,
                  preMargin: Prespace[mapProps.tagSize[K4.MapItems.header],
                                      maxTag, margin]];

    preMargin ← Prespace[mapProps.tagSize[K4.MapItems.from],
                                      maxTag, margin];
    FOR k: CARDINAL IN [1..mapProps.number] DO
      line ← FW.AppendLine[window: window,
                           spaceAboveLine: 2 * spaceBetweenLines];
      FW.AppendItem[window: window, line: line,
                    item: 2*k - 1, preMargin: preMargin];
      FW.AppendItem[window: window, line: line,
                    item: 2*k, preMargin: 8];
      ENDLOOP;

    FW.Repaint [window];
    END;  -- of LayMapItems


  LayNextMapItems: PROC [window: Window.Handle,
                         mapProps: K4.MapProperties] =
    BEGIN
    margin: CARDINAL = 5;
    spaceBetweenLines: CARDINAL = 3;
    preMargin: CARDINAL;
    maxTag: CARDINAL ← 0;
    line: FW.Line;
    lastTo: FW.ItemKey ← 2*mapProps.number;

    FOR k: MapItems IN MapItems DO
      maxTag ← MAX [maxTag, mapProps.tagSize[k]];
      ENDLOOP;

    line ← FW.AppendLine[window: window,
                         spaceAboveLine: 2 * spaceBetweenLines];
    preMargin ← Prespace[mapProps.tagSize[K4.MapItems.from],
                         maxTag, margin];
    FW.AppendItem[window: window, line: line,
                  item: lastTo - 1, preMargin: preMargin];
    FW.AppendItem[window: window, line: line,
                  item: lastTo, preMargin: 8];
    FW.Repaint [window];
    END;  -- of LayNextMapItems


  AddNextMapOptions: FW.NextOutOfProc =
    BEGIN
    -- If NEXTing out of the last field, make and lay a new pair

    parm: K4.ConvertParms ← LOOPHOLE[FW.GetClientData [window]];

    IF item = FW.NumberOfItems[window] THEN
      BEGIN
      MakeNextMapItems [window, AddNextMapOptions,
                        parm.mapProps, parm.zone];
      LayNextMapItems [window, parm.mapProps];
      END;
    END;  -- of AddNextMapOptions


  AddNextMapProps: FW.NextOutOfProc =
    BEGIN
    -- If NEXTing out of the last field, make and lay a new pair

    iconParm: K4.IconParms ← LOOPHOLE[FW.GetClientData [window]];

    IF item = FW.NumberOfItems[window] THEN
      BEGIN
      MakeNextMapItems [window, AddNextMapProps,
                        iconParm.mapProps, iconParm.heap];
      LayNextMapItems [window, iconParm.mapProps];
      END;
```

```
    END;  -- of AddNextMapProps


  SetMapOptions: PropertySheet.MenuItemProc =
    BEGIN
    parm: K4.ConvertParms ← LOOPHOLE[clientData];
    RETURN [SetMapItems [formWindow, parm.mapProps, parm.zone]];
    END;  -- of SetMapOptions


  SetMapProps: PropertySheet.MenuItemProc =
    BEGIN
    iconParm: K4.IconParms ← LOOPHOLE[clientData];
    RETURN [SetMapItems [formWindow, iconParm.mapProps, iconParm.heap]];
    END;  -- of SetMapProps


  SetMapItems: PROC [window: Window.Handle,
                     mapProps: K4.MapProperties,
                     z: UNCOUNTED ZONE]  RETURNS [BOOLEAN]=
    BEGIN
    current, prior: K4.Mapping;
    itemFrom, itemTo: FW.ItemKey;
    emptyFrom, emptyTo: BOOLEAN;

    IF NOT FW.HasAnyBeenChanged [window]
    OR mapProps = NIL THEN RETURN [TRUE];

    current ← mapProps.map;

    FOR itemFrom ← 1, itemFrom + 2
    WHILE itemFrom < FW.NumberOfItems[window] DO
      itemTo ← itemFrom + 1;

      IF FW.HasBeenChanged [window, itemFrom] THEN
          current.from ← UpdateBody [window, itemFrom, @current.from, z];
      IF FW.HasBeenChanged [window, itemTo] THEN
          current.to ← UpdateBody [window, itemTo, @current.to, z];

      emptyFrom ← XS.Empty [@current.from];
      emptyTo ← XS.Empty [@current.to];
      SELECT TRUE FROM
        emptyFrom AND NOT emptyTo =>
                  BEGIN
                  Msg ["Please Fill Left Field or clear Right Field"L];
                  FW.SetInputFocus [window, itemFrom];
                  RETURN [FALSE];
                  END;
        emptyFrom AND emptyTo =>
                  BEGIN
                  -- remove empty pair
                  mapProps.number ← mapProps.number - 1;
                  XS.FreeReaderBytes [@current.from, z];
                  XS.FreeReaderBytes [@current.to, z];
                  IF current = mapProps.map THEN -- empty pair is first
                    BEGIN
                    mapProps.map ← current.next;
                    z.FREE [@current];
                    current ← mapProps.map;
                    prior ← mapProps.map;
                    END
                  ELSE
                    BEGIN
                    prior.next ← current.next;
                    z.FREE [@current];
                    current ← prior.next
                    END;
                  END;
        ENDCASE =>
                  BEGIN
                  prior ← current;
                  current ← current.next;
                  END;
      ENDLOOP;

    RETURN [TRUE];
    END;  -- of SetMapItems



<<==========  PRIVATE PROCEDURES FOR FONT SUBWINDOW  ==========>>


  MakeFontOptions: FW.MakeItemsProc =
    BEGIN
    parm: K4.ConvertParms ← LOOPHOLE[clientData];
    MakeFontItems [window, parm.fonProps, parm.zone];
    END;  -- of MakeFontOptions


  MakeFontProps: FW.MakeItemsProc =
    BEGIN
    iconParm: K4.IconParms ← LOOPHOLE[clientData];
    MakeFontItems [window, iconParm.fonProps, iconParm.heap];
    END;  -- of MakeFontProps
```

```
  MakeFontItems: PROC [window: Window.Handle,
                       fonProps: K4.FontProperties,
                       z: UNCOUNTED ZONE] =
    BEGIN
    label: XS.ReaderBody;
    fontChoices: FW.ChoiceItems ← DESCRIPTOR [fChoices];
    fChoices: ARRAY [0..3) OF FW.ChoiceItem ← [
                [string [choiceNumber: FontStyle.modern.ORD,
                         string: XString.FromSTRING["Modern"L]]],
                [string [choiceNumber: FontStyle.classic.ORD,
                         string: XString.FromSTRING["Classic"L]]],
                [string [choiceNumber: FontStyle.titan.ORD,
                         string: XString.FromSTRING["Titan"L]]] ];

    sizeChoices: FW.ChoiceItems ← DESCRIPTOR [sChoices];
    sChoices: ARRAY [0..5) OF FW.ChoiceItem ← [
                [string [choiceNumber: 0,
                         string: XString.FromSTRING["8"L]]],
                [string [choiceNumber: 1,
                         string: XString.FromSTRING["10"L]]],
                [string [choiceNumber: 2,
                         string: XString.FromSTRING["12"L]]],
                [string [choiceNumber: 3,
                         string: XString.FromSTRING["14"L]]],
                [string [choiceNumber: 4,
                         string: XString.FromSTRING["18"L]]] ];
    MakeFontItems: PROC [fontNumber: CARDINAL,
                         styleKey: K4.FontItems,
                         sizeKey: K4.FontItems,
                         italicsKey: K4.FontItems,
                         boldKey: K4.FontItems] =
          BEGIN
          tagString: LONG STRING ← "Font x"L;
          tagString[tagString.length - 1] ← '0 + VAL[fontNumber];
          fonProps.tagSize[styleKey] ← Measure [@label, tagString];
          fonProps.tagSize[sizeKey] ← 0;
          fonProps.tagSize[boldKey] ← 0;
          fonProps.tagSize[italicsKey] ← 0;
          FW.MakeChoiceItem[
            window: window, tag: @label,
            myKey: styleKey.ORD,
            fullyDisplayed: FALSE,
            values: fontChoices,
            initChoice: VAL[fonProps.fonts[fontNumber].font]];
          FW.MakeChoiceItem[
            window: window, tag: NIL,
            myKey: sizeKey.ORD,
            fullyDisplayed: FALSE,
            values: sizeChoices,
            initChoice: VAL[fonProps.fonts[fontNumber].size]];
          FW.MakeBooleanItem[
            window: window, tag: NIL,
            myKey: boldKey.ORD,
            label: [string [XString.FromSTRING["Bold"L]]],
            initBoolean: fonProps.fonts[fontNumber].bold];
          FW.MakeBooleanItem[
            window: window, tag: NIL,
            myKey: italicsKey.ORD,
            label: [string [XString.FromSTRING["Italics"L]]],
            initBoolean: fonProps.fonts[fontNumber].italics];
          END;  -- of MakeFontItems


    IF fonProps.tagSize = NIL THEN
      fonProps.tagSize ← z.NEW[FontTagSizes];
                                        -- released by FreeFontProps

    MakeFontItems [0, font0, size0, italics0, bold0];
    MakeFontItems [1, font1, size1, italics1, bold1];
    MakeFontItems [2, font2, size2, italics2, bold2];
    MakeFontItems [3, font3, size3, italics3, bold3];
    MakeFontItems [4, font4, size4, italics4, bold4];
    MakeFontItems [5, font5, size5, italics5, bold5];
    MakeFontItems [6, font6, size6, italics6, bold6];
    MakeFontItems [7, font7, size7, italics7, bold7];
    MakeFontItems [8, font8, size8, italics8, bold8];
    MakeFontItems [9, font9, size9, italics9, bold9];
    END;  -- of MakeFontItems


LayFontOptions: FW.LayoutProc =
  BEGIN
  parm: K4.ConvertParms ← LOOPHOLE[clientData];
  LayFontItems [window, parm.fonProps, parm.zone];
  END;  -- of LayFontOptions


LayFontProps: FW.LayoutProc =
  BEGIN
  iconParm: K4.IconParms ← LOOPHOLE[clientData];
  LayFontItems [window, iconParm.fonProps, iconParm.heap];
  END;  -- of LayFontProps


LayFontItems: PROC [window: Window.Handle,
                    fonProps: K4.FontProperties,
```

```
                    z: UNCOUNTED ZONE] =
    BEGIN
    margin: CARDINAL = 5;
    maxTag: CARDINAL ← 0;
    line: FW.Line;
    ts: LONG POINTER TO FontTagSizes ← fonProps.tagSize;  -- accelerator

    LaySingleItem: PROC [itemKey: K4.FontItems,
                         tagSize, interline: CARDINAL] =
          BEGIN
          IF interline > 0 THEN
                  line ← FW.AppendLine[window: window,
                            spaceAboveLine: interline];
          FW.AppendItem[window: window, line: line,
                  item: itemKey.ORD,
                  preMargin: IF interline = 0 THEN 8
                             ELSE Prespace[tagSize, maxTag, margin]];
          END;  -- of LaySingleItem

    FOR k: FontItems IN FontItems DO
      maxTag ← MAX [maxTag, fonProps.tagSize[k]];
      ENDLOOP;

    LaySingleItem[font0, ts[font0], 6];
    LaySingleItem[size0, ts[size0], 0];
    LaySingleItem[italics0, ts[italics0], 0];
    LaySingleItem[bold0, ts[bold0], 0];
    LaySingleItem[font1, ts[font1], 6];
    LaySingleItem[size1, ts[size1], 0];
    LaySingleItem[italics1, ts[italics1], 0];
    LaySingleItem[bold1, ts[bold1], 0];
    LaySingleItem[font2, ts[font2], 6];
    LaySingleItem[size2, ts[size2], 0];
    LaySingleItem[italics2, ts[italics2], 0];
    LaySingleItem[bold2, ts[bold2], 0];
    LaySingleItem[font3, ts[font3], 6];
    LaySingleItem[size3, ts[size3], 0];
    LaySingleItem[italics3, ts[italics3], 0];
    LaySingleItem[bold3, ts[bold3], 0];
    LaySingleItem[font4, ts[font4], 6];
    LaySingleItem[size4, ts[size4], 0];
    LaySingleItem[italics4, ts[italics4], 0];
    LaySingleItem[bold4, ts[bold4], 0];
    LaySingleItem[font5, ts[font5], 6];
    LaySingleItem[size5, ts[size5], 0];
    LaySingleItem[italics5, ts[italics5], 0];
    LaySingleItem[bold5, ts[bold5], 0];
    LaySingleItem[font6, ts[font6], 6];
    LaySingleItem[size6, ts[size6], 0];
    LaySingleItem[italics6, ts[italics6], 0];
    LaySingleItem[bold6, ts[bold6], 0];
    LaySingleItem[font7, ts[font7], 6];
    LaySingleItem[size7, ts[size7], 0];
    LaySingleItem[italics7, ts[italics7], 0];
    LaySingleItem[bold7, ts[bold7], 0];
    LaySingleItem[font8, ts[font8], 6];
    LaySingleItem[size8, ts[size8], 0];
    LaySingleItem[italics8, ts[italics8], 0];
    LaySingleItem[bold8, ts[bold8], 0];
    LaySingleItem[font9, ts[font9], 6];
    LaySingleItem[size9, ts[size9], 0];
    LaySingleItem[italics9, ts[italics9], 0];
    LaySingleItem[bold9, ts[bold9], 0];
    FW.Repaint [window];
    END;  -- of LayFontItems


SetFontOptions: PropertySheet.MenuItemProc =
  BEGIN
  parm: K4.ConvertParms ← LOOPHOLE[clientData];
  RETURN [SetFontItems [formWindow, parm.fonProps, parm.zone]];
  END;  -- of SetFontOptions


SetFontProps: PropertySheet.MenuItemProc =
  BEGIN
  iconParm: K4.IconParms ← LOOPHOLE[clientData];
  RETURN [SetFontItems [formWindow, iconParm.fonProps, iconParm.heap]];
  END;  -- of SetFontProps


SetFontItems: PROC [window: Window.Handle,
                    fonProps: K4.FontProperties,
                    z: UNCOUNTED ZONE]
          RETURNS [BOOLEAN] =
  BEGIN

  Boolean: PROCEDURE [item: FontItems] RETURNS [BOOLEAN] = INLINE
        {RETURN [FW.GetBooleanItemValue [window, item.ORD]]};

  Choice: PROCEDURE [item: FontItems] RETURNS [CARDINAL] = INLINE
        {RETURN [FW.GetChoiceItemValue [window, item.ORD] ]};

  IF NOT FW.HasAnyBeenChanged [window] THEN RETURN [TRUE];

  fonProps.fonts[0].font ← VAL [Choice [K4.FontItems.font0]];
  fonProps.fonts[0].size ← VAL [Choice [K4.FontItems.size0]];
```

```
    fonProps.fonts[0].bold ← Boolean [K4.FontItems.bold0];
    fonProps.fonts[0].italics ← Boolean [K4.FontItems.italics0];
    fonProps.fonts[1].font ← VAL [Choice [K4.FontItems.font1]];
    fonProps.fonts[1].size ← VAL [Choice [K4.FontItems.size1]];
    fonProps.fonts[1].bold ← Boolean [K4.FontItems.bold1];
    fonProps.fonts[1].italics ← Boolean [K4.FontItems.italics1];
    fonProps.fonts[2].font ← VAL [Choice [K4.FontItems.font2]];
    fonProps.fonts[2].size ← VAL [Choice [K4.FontItems.size2]];
    fonProps.fonts[2].bold ← Boolean [K4.FontItems.bold2];
    fonProps.fonts[2].italics ← Boolean [K4.FontItems.italics2];
    fonProps.fonts[3].font ← VAL [Choice [K4.FontItems.font3]];
    fonProps.fonts[3].size ← VAL [Choice [K4.FontItems.size3]];
    fonProps.fonts[3].bold ← Boolean [K4.FontItems.bold3];
    fonProps.fonts[3].italics ← Boolean [K4.FontItems.italics3];
    fonProps.fonts[4].font ← VAL [Choice [K4.FontItems.font4]];
    fonProps.fonts[4].size ← VAL [Choice [K4.FontItems.size4]];
    fonProps.fonts[4].bold ← Boolean [K4.FontItems.bold4];
    fonProps.fonts[4].italics ← Boolean [K4.FontItems.italics4];
    fonProps.fonts[5].font ← VAL [Choice [K4.FontItems.font5]];
    fonProps.fonts[5].size ← VAL [Choice [K4.FontItems.size5]];
    fonProps.fonts[5].bold ← Boolean [K4.FontItems.bold5];
    fonProps.fonts[5].italics ← Boolean [K4.FontItems.italics5];
    fonProps.fonts[6].font ← VAL [Choice [K4.FontItems.font6]];
    fonProps.fonts[6].size ← VAL [Choice [K4.FontItems.size6]];
    fonProps.fonts[6].bold ← Boolean [K4.FontItems.bold6];
    fonProps.fonts[6].italics ← Boolean [K4.FontItems.italics6];
    fonProps.fonts[7].font ← VAL [Choice [K4.FontItems.font7]];
    fonProps.fonts[7].size ← VAL [Choice [K4.FontItems.size7]];
    fonProps.fonts[7].bold ← Boolean [K4.FontItems.bold7];
    fonProps.fonts[7].italics ← Boolean [K4.FontItems.italics7];
    fonProps.fonts[8].font ← VAL [Choice [K4.FontItems.font8]];
    fonProps.fonts[8].size ← VAL [Choice [K4.FontItems.size8]];
    fonProps.fonts[8].bold ← Boolean [K4.FontItems.bold8];
    fonProps.fonts[8].italics ← Boolean [K4.FontItems.italics8];
    fonProps.fonts[9].font ← VAL [Choice [K4.FontItems.font9]];
    fonProps.fonts[9].size ← VAL [Choice [K4.FontItems.size9]];
    fonProps.fonts[9].bold ← Boolean [K4.FontItems.bold9];
    fonProps.fonts[9].italics ← Boolean [K4.FontItems.italics9];

    RETURN [TRUE];
    END;   -- of SetFontItems


<<========== PRIVATE PROCEDURES FOR GENERAL SUBWINDOW ==========>>


MakeGeneralItems: FW.MakeItemsProc =
  BEGIN
    iconParm: K4.IconParms ← LOOPHOLE[clientData];
    label: XS.ReaderBody;
    speedChoices: FW.ChoiceItems ← DESCRIPTOR [sChoices];
    sChoices: ARRAY [0..3) OF FW.ChoiceItem ← [
                  [string [choiceNumber: 0,
                           string: XString.FromSTRING["9600"L]]],
                  [string [choiceNumber: 1,
                           string: XString.FromSTRING["4800"L]]],
                  [string [choiceNumber: 2,
                           string: XString.FromSTRING["300"L]]] ];

    IF iconParm.genProps.tagSize = NIL THEN
      iconParm.genProps.tagSize ← iconParm.heap.NEW[GeneralTagSizes];
                        -- released by FreeIconProps

    iconParm.genProps.tagSize[iconName] ← Measure[@label, "Icon Name"L];
    FW.MakeTextItem[window: window,
                    myKey: K4.GeneralItems.iconName.ORD,
                    tag: @label,
                    initString: @iconParm.genProps.iconName,
                    width: 200];

    iconParm.genProps.tagSize[channelSpeed] ← Measure [@label, "RS232C Speed"L];
    FW.MakeChoiceItem[window: window,
      myKey: K4.GeneralItems.channelSpeed.ORD, tag: @label,
      fullyDisplayed: FALSE,
      values: speedChoices,
      initChoice: VAL[iconParm.genProps.channelSpeed]];

    iconParm.genProps.tagSize[folderName] ← Measure[@label, "Transmission Folder Name"L];
    FW.MakeTextItem[window: window,
                    myKey: K4.GeneralItems.folderName.ORD,
                    tag: @label,
                    initString: @iconParm.genProps.folderName,
                    width: 200];

    iconParm.genProps.tagSize[others] ← Measure[@label, "Required Tailor Choices"L];
    FW.MakeTextItem[window: window,
                    myKey: K4.GeneralItems.others.ORD,
                    tag: @label,
                    initString: @iconParm.genProps.others,
                    width: 300, boxed: FALSE, readOnly: TRUE];

    END;  -- of MakeGeneralItems
```

```
  LayGeneralItems: FW.LayoutProc =
    BEGIN
    iconParm: K4.IconParms ← LOOPHOLE[clientData];
    margin: CARDINAL = 5;
    spaceBetweenLines: CARDINAL = 3;
    maxTag: CARDINAL ← 0;
    line: FW.Line;

    FOR k: K4.GeneralItems IN K4.GeneralItems DO
      maxTag ← MAX [maxTag, iconParm.genProps.tagSize[k]];
      ENDLOOP;

    line ← FW.AppendLine[window: window,
                         spaceAboveLine: 2 * spaceBetweenLines];
    FW.AppendItem[
        window: window, item: K4.GeneralItems.iconName.ORD, line: line,
        preMargin: Prespace[iconParm.genProps.tagSize[iconName], maxTag, margin]];

    line ← FW.AppendLine[window: window,
                         spaceAboveLine: 2 * spaceBetweenLines];
    FW.AppendItem[
        window: window, item: K4.GeneralItems.channelSpeed.ORD, line: line,
        preMargin: Prespace[iconParm.genProps.tagSize[channelSpeed], maxTag, margin]];

    line ← FW.AppendLine[window: window,
                         spaceAboveLine: 2 * spaceBetweenLines];
    FW.AppendItem[
        window: window, item: K4.GeneralItems.folderName.ORD, line: line,
        preMargin: Prespace[iconParm.genProps.tagSize[folderName], maxTag, margin]];

    line ← FW.AppendLine[window: window,
                         spaceAboveLine: 4 * spaceBetweenLines];
    FW.AppendItem[
        window: window, item: K4.GeneralItems.others.ORD, line: line,
        preMargin: Prespace[iconParm.genProps.tagSize[others], maxTag, margin]];

    FW.Repaint [window];
    END;  -- of LayGeneralItems


  SetGeneralItems: PropertySheet.MenuItemProc =
    BEGIN
    nsSelections: NSFile.Selections ← [];
    nsName: NSString.String;
    attributeList: ARRAY[0..1) OF NSFile.Attribute;
    ps: K4.IconParms ← LOOPHOLE[clientData];

    IF FW.HasBeenChanged [formWindow, GeneralItems.iconName.ORD] THEN
      BEGIN
      ps.genProps.iconName ← UpdateBody [formWindow,
                                         GeneralItems.iconName.ORD,
                                         @ps.genProps.iconName,
                                         ps.heap];
      IF XS.Empty [@ps.genProps.iconName] THEN
        BEGIN
        Msg ["Icon name cannot be empty"L];
        RETURN [FALSE];
        END;
      nsName ← XS.NSStringFromReader [@ps.genProps.iconName, ps.heap];
      attributeList[0] ← [name [nsName]];
      NSFile.ChangeAttributes [ps.iconFile, DESCRIPTOR[attributeList]];
      NSString.FreeString [ps.heap, nsName];
      END;

    IF FW.HasBeenChanged [formWindow, GeneralItems.folderName.ORD] THEN
      BEGIN
      ps.genProps.folderName ← UpdateBody [formWindow,
                                           GeneralItems.folderName.ORD,
                                           @ps.genProps.folderName,
                                           ps.heap];
      IF XS.Empty [@ps.genProps.folderName] THEN
        BEGIN
        Msg ["Folder name cannot be empty"L];
        RETURN [FALSE];
        END;
      END;

    ps.genProps.channelSpeed ← VAL [FW.GetChoiceItemValue[
                                    formWindow,
                                    K4.GeneralItems.channelSpeed.ORD]];
    RETURN [TRUE];
    END;  -- of SetGeneralItems



<<==========  PRIVATE PROCEDURES FOR LINK SUBWINDOW  ==========>>


MakeLinkOptions: FW.MakeItemsProc =
  BEGIN
  parm: K4.ConvertParms ← LOOPHOLE[clientData];
  sheetChoices: FW.ChoiceItems ← DESCRIPTOR [sChoices];
  sChoices: ARRAY [0..3) OF FW.ChoiceItem ← [
              [string [choiceNumber: 0,
                       string: XString.FromSTRING["Document"L]]],
```

```
                [string [choiceNumber: 1,
                         string: XString.FromSTRING["Mapping"L]]],
                [string [choiceNumber: 2,
                         string: XString.FromSTRING["Fonts"L]]] ];

    MakeLinkItems [window, sheetChoices, 0, SwapOptions];
    END;  -- of MakeLinkOptions


  MakeLinkProps: FW.MakeItemsProc =
    BEGIN
    iconParm: K4.IconParms ← LOOPHOLE[clientData];
    sheetChoices: FW.ChoiceItems ← DESCRIPTOR [sChoices];
    sChoices: ARRAY [0..4) OF FW.ChoiceItem ← [
                [string [choiceNumber: 0,
                         string: XString.FromSTRING["Icon"L]]],
                [string [choiceNumber: 1,
                         string: XString.FromSTRING["Document"L]]],
                [string [choiceNumber: 2,
                         string: XString.FromSTRING["Mapping"L]]],
                [string [choiceNumber: 3,
                         string: XString.FromSTRING["Font"L]]] ];

    MakeLinkItems [window, sheetChoices, 1, SwapProps];
    END;  -- of MakeLinkProps


  MakeLinkItems: PROC [window: Window.Handle,
                       choices: FW.ChoiceItems,
                       firstChoice: FW.ItemKey,
                       swapProc: FW.ChoiceChangeProc] =
    BEGIN
    label: XS.ReaderBody ← XS.FromSTRING["Sheet for: "L];

    FW.MakeChoiceItem[window: window,
                      myKey: 0, tag: @label,
                      values: choices,
                      initChoice: firstChoice,
                      changeProc: swapProc];
    END;  -- of MakeLinkItems


  LayLinkItem: FW.LayoutProc =
    BEGIN
    line: FW.Line ← FW.AppendLine[window: window,
                                  spaceAboveLine: 9];
    FW.AppendItem[window: window, item: 0, line: line, preMargin: 5];
    END;  -- of LayLinkItem


  SwapOptions: FW.ChoiceChangeProc =
    BEGIN
    parm: K4.ConvertParms ← LOOPHOLE[FW.GetClientData [window]];
    newMake: FW.MakeItemsProc;
    newLay: FW.LayoutProc;
    newSet: PropertySheet.MenuItemProc;

    SELECT newValue FROM
      0       => {newMake ← MakeTextOptions;
                  newLay ← LayTextOptions;
                  newSet ← SetTextOptions};
      1       => {newMake ← MakeMapOptions;
                  newLay ← LayMapOptions;
                  newSet ← SetMapOptions};
      ENDCASE => {newMake ← MakeFontOptions;
                  newLay ← LayFontOptions;
                  newSet ← SetFontOptions};

    [] ← PropertySheet.SwapFormWindows [
                  shell: parm.optionSheet,
                  apply: TRUE,
                  newFormWindowItems: newMake,
                  newFormWindowItemsLayout: newLay,
                  newMenuItemProc: newSet];
    END;  -- of SwapOptions


  SwapProps: FW.ChoiceChangeProc =
    BEGIN
    iconParm: K4.IconParms ← LOOPHOLE[FW.GetClientData [window]];
    newMake: FW.MakeItemsProc;
    newLay: FW.LayoutProc;
    newSet: PropertySheet.MenuItemProc;

    SELECT newValue FROM
      1       => {newMake ← MakeTextProps;
                  newLay ← LayTextProps;
                  newSet ← SetTextProps};
      2       => {newMake ← MakeMapProps;
                  newLay ← LayMapProps;
                  newSet ← SetMapProps};
      3       => {newMake ← MakeFontProps;
                  newLay ← LayFontProps;
                  newSet ← SetFontProps};
      ENDCASE => {newMake ← MakeGeneralItems;
                  newLay ← LayGeneralItems;
                  newSet ← SetGeneralItems};
```

```
    [] ← PropertySheet.SwapFormWindows [
                shell: iconParm.propSheet,
                apply: TRUE,
                newFormWindowItems: newMake,
                newFormWindowItemsLayout: newLay,
                newMenuItemProc: newSet];

    END;  -- of SwapProps


  TakeDownPSheet: PropertySheet.MenuItemProc =
    BEGIN
    data: Containee.Data;
    nsSelections: NSFile.Selections ← [];
    iconParm: K4.IconParms ← LOOPHOLE[clientData];
    z: UNCOUNTED ZONE ← iconParm.heap;

    IF menuItem = done THEN
      BEGIN
      IF iconParm.mapProps # NIL AND iconParm.mapProps.number = 1 THEN
        BEGIN      -- check if only entry is empty
        IF  XS.Empty [@iconParm.mapProps.map.from]
        AND XS.Empty [@iconParm.mapProps.map.to]
                THEN iconParm.mapProps.number ← 0;
        END;
      K4.StoreFiledData [iconParm];
      IF iconParm.changeProc # NIL THEN
        BEGIN
        data ← [reference: NSFile.GetReference[iconParm.iconFile]];
        nsSelections.interpreted[name] ← TRUE;
        iconParm.changeProc [changeProcData: iconParm.changeProcData,
                        data: @data,
                        changedAttributes: nsSelections,
                        noChanges: FALSE];
        END;
      END;

    K4.FreeIconProps [iconParm.genProps, z];
    K4.FreeTextProps [iconParm.docProps, z];
    K4.FreeMapProps [iconParm.mapProps, z];
    K4.FreeFontProps [iconParm.fonProps, z];
    NSFile.Close [iconParm.iconFile];
    z.FREE [@iconParm];
    Heap.Delete [z];
    RETURN [TRUE];
    END;  -- of TakeDownPSheet


<<========== OTHER PRIVATE PROCEDURES ==========>>


Measure: PROCEDURE[label: XS.Reader, string: LONG STRING]
          RETURNS [size: CARDINAL] =
  {label↑ ← XS.FromSTRING[string];
   size ← SimpleTextDisplay.MeasureString[label].width};


Prespace: PROCEDURE [size, maxSize, margin: CARDINAL]
            RETURNS [preMargin: CARDINAL] =
  BEGIN
  preMargin ← margin + (maxSize - size) + (IF size = 0
                                          THEN 8 ELSE 0);
  END;


UpdateBody: PROC [window: Window.Handle, item: FW.ItemKey,
                  oldReader: XS.Reader, zone: UNCOUNTED ZONE]
          RETURNS [newReaderBody: XS.ReaderBody] =
  BEGIN
  rb: XS.ReaderBody ← FW.LookAtTextItemValue [window, item];

  XS.FreeReaderBytes [oldReader, zone];
  newReaderBody ← XS.CopyToNewReaderBody [@rb, zone];
  FW.DoneLookingAtTextItemValue [window, item];
  END;  -- of UpdateBody


Msg: PROCEDURE [message: LONG STRING] =
  BEGIN
  msgRB: XS.ReaderBody ← XS.FromSTRING [message];
  Attention.Post [@msgRB];
  END;  -- of Msg




END.  -- of K4PSheetImpl
14-Jan-87 15:42:32 created from DestTextPSheetImpl.
 6-Feb-87 10:12:41 fixed storage leak (folderName/outputName not deallocated).
12-Mar-87 13:48:25 made SetText/FontOptions PUBLIC.
16-Mar-87 10:00:54 added signature.
17-Mar-87 15:37:09 added character mapping.
26-Mar-87 11:27:35 made channel speed singly-displayed.
31-Mar-87 14:12:20 changes to StoreFiledData, no longer releasing memory.
```

```
31-Mar-87 16:30:15 tried to fix nexting out of mappings.
 7-Apr-87 17:23:53 Taylor => Tailor.
13-Apr-87 13:13:26 set mapProps.number to 0 if only entry is empty.
14-Apr-87 14:32:18 added paragraph properties.
20-Apr-87 17:45:17 initial value of channel speed not picked up from filed data.
24-Apr-87 16:56:45 removed canvas property/option sheets.
27-Apr-87 13:21:20 transmission folder name, page breaks.
28-Apr-87 16:57:56 checked for empty icon, folder or document names
```

```
<< File: K4WindowImpl.mesa -   15-Sep-88  9:16:10

deLaBeaujardiere:OSBU North:Xerox (deLaBeaujardiere.PA)

Copyright (C) 1986 by Xerox Corporation. All rights reserved.

>>

DIRECTORY Attention, Containee, Environment, Heap,
          K4, MenuData, MessageWindow,
          NSFile, NSFileStream, NSString, Process,
          RS232C, RS232CCorrespondents, RS232CEnvironment,
          Selection, StarDesktop, StarFileTypes,
          StarWindowShell, StarWindowShellExtra2,
          Stream, TIP, Window, XFormat, XString;

K4WindowImpl: MONITOR

IMPORTS   Attention, Heap, K4, MenuData, MessageWindow,
          NSFile, NSFileStream, NSString,
          Process, RS232C, Selection,
          StarDesktop, StarWindowShell, StarWindowShellExtra2,
          Stream, XFormat, XString

EXPORTS   K4 =

BEGIN
       OPEN SWS: StarWindowShell, XS: XString;


Variables: TYPE = LONG POINTER TO VariableObject;
VariableObject: TYPE = RECORD [
  window:                Window.Handle,
  channel:               RS232C.ChannelHandle,
  commParamObject:       RS232C.CommParamObject,
  data:                  RS232C.PhysicalRecord,
  windowClosing:         BOOLEAN ← FALSE,
  listener:              PROCESS ← NIL];


parms:  K4.IconParms;
vars:   Variables;

bufferSize: CARDINAL = 512;


OpenWindow: PUBLIC PROCEDURE [iconData: Containee.DataHandle,
                              changeProc: Containee.ChangeProc,
                              changeProcData: LONG POINTER,
                              tinyIcon: XS.Character]
                      RETURNS [shell: SWS.Handle ← SWS.nullHandle] =
  BEGIN
  wDims:    Window.Dims ← [500, 230];
  wPlace:   Window.Place ← [50, 30];
  wLines:   CARDINAL ← 10;
  mismatch: BOOLEAN;
  zone:     UNCOUNTED ZONE ← Heap.Create [4];  --** why 4?
                       -- deleted by ShellClosing
  reconvert: XString.ReaderBody ← XString.FromSTRING["Make Document"L];
  command: ARRAY[0..1) OF MenuData.ItemHandle ← [
                                MenuData.CreateItem[
                                                zone: zone,
                                                name: @reconvert,
                                                proc: Reconvert]];

              --get memory for IconParms and fill it
              -- (freed by ShellClosing, unless there is
              --   a software/icon mismatch).
  parms ← zone.NEW [K4.IconParmsRecord];
  parms.heap ← zone;
  parms.iconFile ← NSFile.OpenByReference [iconData.reference];
                                -- closed by ShellClosing
  mismatch ← K4.LoadFiledData [parms]; -- unloaded by ShellClosing
  IF mismatch THEN
    BEGIN
    msg: XS.ReaderBody ← XS.FromSTRING [
              "Obsolete icon does not work with new software"L];
    Attention.Post [@msg];
    NSFile.Close [parms.iconFile];
    zone.FREE [@parms];
    Heap.Delete [zone];
    RETURN [SWS.nullHandle];
    END;
              -- get memory for Variable
              -- (will be freed by ShellClosing).
  vars ← zone.NEW[VariableObject];

              -- create window shell and message window
  shell ← SWS.Create [namePicture: tinyIcon,
                      name: @parms.genProps.iconName,
                      scrollData: SWS.vanillaScrollData,
                      isCloseLegalProc: ShellClosing];
  SWS.SetRegularCommands [sws: shell,
                          commands: MenuData.CreateMenu [
                                zone: zone,
                                title: NIL,
                                array: DESCRIPTOR[command]]];
```

```
    vars.window ← SWS.CreateBody [shell];
    StarWindowShellExtra2.SetPreferredInteriorDims[sws: shell,
                                                   dims: wDims];
    MessageWindow.Create [vars.window, zone, wLines];
    MsgDate [];

    ReadyRS232C [vars, parms.genProps.channelSpeed];

    vars.listener ← FORK GetTransmissionAndConvert [vars, parms];
    END;  -- of OpenWindow


  PutFileInFolder: PUBLIC PROCEDURE [file: NSFile.Handle,
                                     folderName: NSString.String] =
    BEGIN
    -- This proc puts a file
    -- in the named folder on the desktop.
    -- If the folder is not found, a new one is created.
    -- If the folder creation fails, we leave the file
    -- on the desktop as a last resort...

    dateOrdered: key NSFile.Ordering;
    folderFile: NSFile.Handle ← NSFile.nullHandle;
    folderAttrs: ARRAY [0..4) OF NSFile.Attribute;
    desktopFile: NSFile.Handle ← NSFile.OpenByReference [
                  StarDesktop.GetCurrentDesktopFile []];

    BEGIN
    folderAttrs[0] ← [name [folderName]];
    folderFile ← NSFile.Open [
          attributes: DESCRIPTOR [BASE[folderAttrs], 1],
          directory: desktopFile
          ! NSFile.Error =>
                BEGIN
                WITH error SELECT FROM
                  access =>
                    SELECT problem FROM
                      fileNotFound =>
                        BEGIN
                        dateOrdered.ascending ← TRUE;
                        dateOrdered.key ← createdOn;
                        folderAttrs[1] ← [type[StarFileTypes.folder]];
                        folderAttrs[2] ← [isDirectory [TRUE]];
                        folderAttrs[3] ← [ordering[dateOrdered]];
                        folderFile ← NSFile.Create [
                                directory:  desktopFile,
                                attributes: DESCRIPTOR[folderAttrs]];
                        StarDesktop.AddReferenceToDesktop [
                                NSFile.GetReference [folderFile]];
                        END;
                      ENDCASE;
                  ENDCASE;
                GOTO Done;
                END];
    EXITS Done => {};
    END;

    IF folderFile = NSFile.nullHandle THEN -- folder creation failed
      StarDesktop.AddReferenceToDesktop[NSFile.GetReference [file]]
    ELSE
      BEGIN
      NSFile.Move [file, folderFile];
      NSFile.Close [folderFile];
      END;
    NSFile.Close [desktopFile];
    END;  -- of PutFileInFolder



  -- PRIVATE PROCEDURES


  ShellClosing: ENTRY SWS.IsCloseLegalProc =
    BEGIN
    ENABLE UNWIND => NULL;
    z: UNCOUNTED ZONE;
    clearMask: RS232C.DeviceStatus ← [
                  statusAborted: FALSE, dataLost: FALSE,
                  breakDetected: FALSE, clearToSend: TRUE,
                  dataSetReady: TRUE, carrierDetect: TRUE,
                  ringHeard: FALSE, ringIndicator: FALSE,
                  deviceError: FALSE];

    vars.windowClosing ← TRUE;
    IF vars.listener # NIL THEN
      BEGIN
      RS232C.Suspend [vars.channel, all];
      JOIN vars.listener;
      RS232C.SetParameter [vars.channel, [latchBitClear [clearMask]]];
      RS232C.Delete [vars.channel];
      vars.listener ← NIL;
      END;

    IF parms.changeProc # NIL THEN parms.changeProc[
                          changeProcData: parms.changeProcData,
                          data: parms.iconData,
```

```
                        noChanges: TRUE];
    MessageWindow.Destroy [vars.window];  -- clear message resources
    NSFile.Close [parms.iconFile];
    z ← parms.heap;
    z.FREE [@vars];
    K4.FreeIconProps [parms.genProps, z];
    K4.FreeTextProps [parms.docProps, z];
    K4.FreeMapProps [parms.mapProps, z];
    K4.FreeFontProps [parms.fonProps, z];
    z.FREE [@parms];
    Heap.Delete [z];
    RETURN[TRUE];
    END;  -- of ShellClosing


  Msg: PROC [message: LONG STRING, startOnNewLine: BOOLEAN] =
    BEGIN
    picture: XFormat.Object ← MessageWindow.XFormatObject [vars.window];
    IF startOnNewLine THEN
      BEGIN
      MessageWindow.PostSTRING [vars.window, "
"L, TRUE];
      XFormat.Date [h: @picture, format: timeOnly];
      MessageWindow.PostSTRING [vars.window, "
"L, FALSE];
      END;
    MessageWindow.PostSTRING [vars.window, message, FALSE];
    END;  -- of Msg


  MsgDate: PROC =
    BEGIN
    <<
    pic: XFormat.Object ← MessageWindow.XFormatObject [vars.window];
    MessageWindow.PostSTRING [vars.window, "
"L, TRUE];
    XFormat.Date [h: @pic, format: dateOnly];
    >>
    END;  -- of MsgDate


  MsgDecimal: PROC [number: LONG CARDINAL] =
    BEGIN
    picture: XFormat.Object ← MessageWindow.XFormatObject [vars.window];
    XFormat.Decimal [h: @picture, n: number];
    END;  -- of MsgDecimal


  ReadyRS232C: PROCEDURE [vars: Variables,
                          optionSpeed: K4.ChannelSpeed] =
    BEGIN
    speed: RS232C.LineSpeed ← SELECT optionSpeed FROM
                                    ninetySix     => bps9600,
                                    fortyEight    => bps4800,
                                    three         => bps300,
                                    ENDCASE       => bps9600;
    clearMask: RS232C.DeviceStatus ← [
                  statusAborted: FALSE, dataLost: FALSE,
                  breakDetected: FALSE, clearToSend: TRUE,
                  dataSetReady: TRUE, carrierDetect: TRUE,
                  ringHeard: FALSE, ringIndicator: FALSE,
                  deviceError: FALSE];
    vars.commParamObject  ← [duplex: full,
                             lineType: asynchronous,
                             lineSpeed: speed,
                             accessDetail: directConn[]];

    vars.channel ← RS232C.Create [
              lineNumber: RS232C.GetNextLine [RS232C.nullLineNumber],
              commParams: @vars.commParamObject,
              preemptOthers: preemptAlways,
              preemptMe: preemptAlways];

    RS232C.SetParameter [vars.channel, [charLength [8]]];
    RS232C.SetParameter [vars.channel,
      [correspondent [RS232CCorrespondents.ttyHost]]];
    RS232C.SetParameter [vars.channel, [frameTimeout [1000]]];
    RS232C.SetParameter [vars.channel, [lineSpeed [speed]]];
    RS232C.SetParameter [vars.channel, [parity [none]]];
    RS232C.SetParameter [vars.channel, [stopBits [1]]];
    RS232C.SetParameter [vars.channel, [latchBitClear [clearMask]]];
    RS232C.SetParameter [vars.channel, [dataTerminalReady [TRUE]]];
    RS232C.SetParameter[vars.channel, [requestToSend [TRUE]]];
    RS232C.SetParameter[vars.channel,
                 [flowControl [[type: xOnXOff,
                               xOn: 17,          -- DC1 (↑q)
                               xOff: 19]]]];     -- DC3 (↑s)
    END;  -- of ReadyRS232C


GetTransmissionAndConvert: PROCEDURE [vars: Variables,
                                      parms: K4.IconParms] =
    BEGIN
    byteArray:    PACKED ARRAY [0..bufferSize) OF Environment.Byte;
    logAttrs:     ARRAY[0..2) OF NSFile.Attribute ← [
                        [name [NSString.StringFromMesaString["K4TEXT.LOG"L]]],
                        [type [2]]];
    logFile:      NSFile.Handle;
    logStream:    NSFileStream.Handle;
    folderName:   NSString.String;
```

```
    dots:          CARDINAL;        -- to count dots announcing reception
    bytesLogged:   LONG CARDINAL;
    bytesReceived:         CARDINAL;
    transferStatus:        RS232C.TransferStatus;

    Process.SetPriority [Process.priorityNormal];
    vars.data ← [header: Environment.nullBlock,
                 body: [blockPointer: @byteArray,
                        startIndex: 0,
                        stopIndexPlusOne: bufferSize],
                 trailer: Environment.nullBlock];

    DO  -- run while window is open
      dots ← 0;
      bytesLogged ← 0;
      Msg ["Waiting for transmission from Kurzweil 4000"L, TRUE];
      [bytesReceived, transferStatus] ← GetBlock[vars];     -- request 1st block

      IF vars.windowClosing THEN RETURN;

      Msg ["Receiving"L, TRUE];    -- announce initial reception
      logFile ← NSFile.Create [directory:  NSFile.nullHandle,
                               attributes: DESCRIPTOR[logAttrs]];
      logStream ← NSFileStream.Create [logFile, FALSE];


      DO
        IF vars.windowClosing THEN
          BEGIN
          Stream.Delete [logStream];
          NSFile.Close [logFile];
          RETURN;
          END;

        IF transferStatus # success THEN EXIT;
        IF bytesReceived = 0 THEN LOOP;

        vars.data.body.stopIndexPlusOne ← bytesReceived;
        bytesLogged ← bytesLogged + bytesReceived;
        dots ← IF dots > 50 THEN 0 ELSE dots + 1;
        Msg [".L, (dots = 0)];    -- continue announcing block reception

        Stream.PutBlock [logStream, vars.data.body];
        IF  byteArray[bytesReceived - 1] = 200B THEN EXIT;

        [bytesReceived, transferStatus] ← GetBlock[vars];
        ENDLOOP;

      IF bytesLogged < 5 THEN  -- don't bother saving 4 bytes or less...
        BEGIN
        Msg ["  Less than 5 bytes received. Ignored."L, FALSE];
        Stream.Delete [logStream];
        NSFile.Close [logFile];
        END
      ELSE
        BEGIN
        bytesLogged ← bytesLogged - 1; -- to drop the 200B added by K4000
        MsgDecimal [bytesLogged];
        Msg [" bytes received."L, FALSE];
        Stream.SendNow [logStream];
        NSFileStream.SetLength [logStream, bytesLogged];
        Stream.Delete [logStream];
        folderName ← XS.NSStringFromReader [@parms.genProps.folderName,
                                            parms.heap];
        PutFileInFolder [logFile, folderName];
        NSString.FreeString [parms.heap, folderName];
        Msg ["Spawning document creation."L, TRUE];
        K4.ConvertToDocument [logFile, parms.docProps,
                              parms.mapProps, parms.fonProps];
        END;
      ENDLOOP;

  END;  -- of GetTransmissionAndConvert


GetBlock: PROC [vars: Variables]
        RETURNS [count: CARDINAL, status: RS232C.TransferStatus] =
  BEGIN
  completionHandle: RS232C.CompletionHandle;

  vars.data.body.stopIndexPlusOne ← bufferSize;
  completionHandle ← RS232C.Get[vars.channel, @vars.data];
  [count, status] ← RS232C.TransferWait[vars.channel, completionHandle];
  END;  -- of GetBlock


Reconvert: MenuData.MenuProc =
  BEGIN
  --*** use here selection interpretation in K4ToVPUtility
  OPEN Selection;
  -- User has selected a log stream in document folder
  -- and wants to make a new document from it.
  fileSelection: Value ← Convert [Target.file];
  typeSelection: Value ← Convert [Target.fileType];
  typeAscii: NSFile.Type = 2;

  IF fileSelection.value = nullValue.value
```

```
    OR typeSelection.value = nullValue.value
    OR typeSelection.value↑ # typeAscii THEN
      BEGIN
      Msg ["Reconverting: there is no selection to convert."L,
          TRUE];
      END
    ELSE
      BEGIN
      refLoc: LONG POINTER TO NSFile.Reference ←
                                LOOPHOLE[fileSelection.value];
      logFile: NSFile.Handle ← NSFile.OpenByReference [refLoc↑];
                                -- file closed by conversion job
      Msg ["Spawning document creation."L, TRUE];
      K4.ConvertToDocument [logFile, parms.docProps, parms.mapProps, parms.fonProps];
      END;
    END;  -- of Reconvert


END.  -- of K4WindowImpl
LOG:
23-Jan-87 16:42:05 created from DestTextWindowImpl and K43WindowImpl.
 4-Feb-87 11:47:24 moved pSheet items into body window, removed Options command.
 6-Feb-87 10:13:34 fixed storage leak (folderName/outputName not deallocated).
 6-Feb-87 17:58:04 implemented ReceiveScannerData as background job.
13-Feb-87 10:56:09 redid entire RS232C interaction to prevent hanging in RS232C.Suspend.
16-Feb-87 14:06:07 added warning sheet to run Editor if idle.
18-Feb-87 15:47:10 added inspection of header to determine output type.
26-Feb-87 17:22:49 implemented IntervalTimer facility because text files from K4000 do not always have etx at the end, and we need to
time out after a while.
 3-Mar-87 10:20:32 incorporated call to ConvertToCanvas, request to run VP Free-hand Drawing, etc...
 4-Mar-87 15:58:47 made PutFileInFolder a public proc.
10-Mar-87 14:24:23 adapted to linked property sheet, removed panel window in preparation of installing history window.
11-Mar-87 14:05:31 turned into a message window.
12-Mar-87 10:58:39 fixed messages/transmission coordination.
16-Mar-87 10:12:10 added signature.
18-Mar-87 14:38:46 cleared channel mask bits after deleting.
19-Mar-87 10:12:40 mapProps not initialized in SpinOffDocument.
23-Mar-87 15:21:14 added Reconvert command.
24-Mar-87 13:30:21 ordering folder by creation date.
 2-Apr-87 10:43:20 saving/naming of logFile moved to K4CanvasImpl or K4DocumentImpl.
 6-Apr-87 15:45:27 creation of ConvertParms moved from here to K4DocumentImpl and K4CanvasImpl.
 7-Apr-87 13:54:40 removed useless start code.
 7-Apr-87 14:28:32 moved back here saving of logFile; had problem with holding same handle in two processes.
 9-Apr-87 11:46:09 Looks like the logFile does not have all the data received; so now using block/PutBlock instead of string/PutString to
save bytes received from channel.
14-Apr-87 14:44:48 removed saveLog.
15-Apr-87 15:07:44 remove flow control on RS232C channel, set timeout to 1000 millisecs.
18-Apr-87 16:27:55 restored flow control (still no clues about lost bytes: tried slower baud rates, larger timeouts, 7 bits, high
priority processes, flow control/no flow control, larger buffers, ..).
20-Apr-87 14:48:27 truncated the log stream by one byte because it seems that we always get one extra byte (200B) at the end, perhaps
generated by the Suspend[channel].
20-Apr-87 14:49:40 picked up channel speed from option sheet.
20-Apr-87 18:09:56 adjusted length of silence to channel speed.
24-Apr-87 16:43:38 dropped canvas and ArtScan processing.
27-Apr-87 10:08:52 "Reconvert" becomes "Make Document".
27-Apr-87 13:25:32 transmission folder name.
14-Sep-88  9:59:55 discovered that in VP2.0, we no longer get the last block; perhaps something with the 5 seconds wait in lower priority
process; also discovered with DLM that the last byte of last transmission, 80X, is sent by Kurzweil, not added by Suspend or some such;
thus, the code to wait a while to see if transmission is ended is replaced by testing 80X.
15-Sep-88  9:13:53 cleared latch bits before deleting channel: I noticed that the DEST claims that the port is not ready after Kurzweil
application has been used; perhaps clearing the latch bits will help.
/
```

```
-- OctalConvertToolImpl.mesa
-- Trow:PARC:Xerox  7-Sep-89 21:37:23

-- Copyright (c) 1989 by Xerox Corporation.  All rights reserved.

DIRECTORY
  Ascii USING [CR, SP],
  Environment USING [Byte],
  Event USING [DoneWithProcess, Handle, StartingProcess, toolWindow],
  EventTypes USING [deactivate],
  Exec USING [AddCommand, ExecProc, RemoveCommand],
  FormSW USING [AllocateItemDescriptor, BooleanItem, ClientItemsProcType,
    CommandItem, ItemHandle, line0, line1, line2, ProcType, StringItem],
  Heap USING [Create, Delete],
  Inline USING [BITAND, BITSHIFT],
  MFile USING [Type],
  MStream USING [Error, GetLength, ReadOnly, SetLength, WriteOnly],
  Process USING [Detach],
  Put USING [Line, Text],
  Runtime USING [GetBcdTime],
  Stream USING [Delete, EndOfStream, GetChar, GetByte, Handle, PutByte, PutChar,
    PutString],
  String USING [AppendLongNumber, AppendString],
  Supervisor USING [AddDependency, AgentProcedure, CreateSubsystem,
    EnumerationAborted, RemoveDependency, SubsystemHandle],
  Time USING [Append, AppendCurrent, Unpack],
  Tool USING [Create, Destroy, MakeFileSW, MakeFormSW, MakeMsgSW, MakeSWsProc,
    UnusedLogName],
  ToolWindow USING [Activate, GetState, TransitionProcType],
  Window USING [GetChild, GetParent, Handle, Stack, ValidateTree];

OctalConvertToolImpl: PROGRAM
  IMPORTS Event, Exec, FormSW, Heap, Inline, MStream, Process, Put,
    Runtime, Stream, String, Supervisor, Time, Tool, ToolWindow, Window
  =
  BEGIN

  ----------------------
  -- Types
  ----------------------

  FormIndex: TYPE = {protectBinary, binaryFile, protectOctal, octalFile, binaryToOctal,
    octalToBinary};

  ToolData: TYPE = MACHINE DEPENDENT RECORD [
    msgSW(0): Window.Handle ← NIL,
    formSW(2): Window.Handle ← NIL,
    logSW(4): Window.Handle ← NIL,
    protectBinary(6): BOOLEAN ← TRUE,
    protectOctal(7): BOOLEAN ← TRUE,
    binaryFileName(8): LONG STRING ← NIL,
    octalFileName(10): LONG STRING ← NIL,
    commandIsRunning(12): BOOLEAN ← FALSE];

  ----------------------
  -- Constants
  ----------------------

  agent: Supervisor.SubsystemHandle = Supervisor.CreateSubsystem[CheckDeactivate];

  ----------------------
  -- Globals
  ----------------------

  data: LONG POINTER TO ToolData ← NIL;
  wh: Window.Handle ← NIL;
  heap: UNCOUNTED ZONE ← NIL;

  ----------------------
  -- Initialization
  ----------------------

  Init: PROC =
    BEGIN
    Exec.AddCommand["OctalConvertTool.~"L, MakeTool, NIL, Unload];
    END; -- Init

  MakeTool: Exec.ExecProc =
    BEGIN
    IF wh = NIL THEN
      BEGIN
      name: LONG STRING ← [60];
      String.AppendString[to: name, from : "Octal Convert Tool of "L];
      Time.Append[s: name, unpacked: Time.Unpack[Runtime.GetBcdTime[]]];
      name.length ← name.length - 3; -- lop the seconds
      wh ← Tool.Create[name: name, makeSWsProc: MakeSWs,
        clientTransition: ClientTransition, cmSection: "OctalConvertTool"L,
        tinyName1: "Octal"L, tinyName2: "Convert"L];
      END
    ELSE IF ToolWindow.GetState[wh] = active THEN
      BEGIN
      newSibling: Window.Handle = Window.GetChild[Window.GetParent[wh]];
      IF wh # newSibling THEN {
        Window.Stack[wh, newSibling]; Window.ValidateTree[wh]};
      END
    ELSE
      ToolWindow.Activate[wh];
```

```
    END; -- MakeTool

  Unload: Exec.ExecProc =
    BEGIN
    IF wh # NIL THEN {Tool.Destroy[wh]; wh ← NIL};
    Exec.RemoveCommand[h, "OctalConvertTool.~"L];
    END; -- Unload

  --====================
  -- State change
  --====================

  CheckDeactivate: Supervisor.AgentProcedure =
    BEGIN
    IF event = EventTypes.deactivate AND wh # NIL
     AND wh = eventData AND data.commandIsRunning THEN
      BEGIN
      Put.Line[data.msgSW, "Tool is busy!"L];
      ERROR Supervisor.EnumerationAborted;
      END;
    END; -- CheckDeactivate

  ClientTransition: ToolWindow.TransitionProcType =
    BEGIN
    SELECT TRUE FROM
      old = inactive =>
        BEGIN
        IF heap = NIL THEN heap ← Heap.Create[initial: 1, increment: 1];
        IF data = NIL THEN data ← heap.NEW[ToolData ← []];
        END;
      new = inactive =>
        BEGIN
        Supervisor.RemoveDependency[client: agent, implementor: Event.toolWindow];
        IF data # NIL THEN heap.FREE[@data];
        IF heap # NIL THEN {Heap.Delete[heap]; heap ← NIL};
        END;
      ENDCASE;
    END; -- ClientTransition

  --====================
  -- Tool window
  --====================

  MakeSWs: Tool.MakeSWsProc =
    BEGIN
    logFileName: STRING = [50];
    Tool.UnusedLogName[unused: logFileName, root: "OctalConvertTool.log"L];
    data.msgSW ← Tool.MakeMsgSW[window: window];
    data.formSW ← Tool.MakeFormSW[window: window, formProc: MakeForm];
    data.logSW ← Tool.MakeFileSW[window: window, name: logFileName];
    Supervisor.AddDependency[client: agent, implementor: Event.toolWindow];
    END; -- MakeSWs

  MakeForm: FormSW.ClientItemsProcType =
    BEGIN
    OPEN FormSW;
    nItems: CARDINAL = FormIndex.LAST.ORD + 1;
    items ← AllocateItemDescriptor[nItems];
    items[FormIndex.protectBinary.ORD] ← BooleanItem[
      tag: "ProtectBinary"L, place: [0, line0], switch: @data.protectBinary];
    items[FormIndex.binaryFile.ORD] ← StringItem[
      tag: "Binary File"L, place: [100, line0], inHeap: TRUE,
      string: @data.binaryFileName];
    items[FormIndex.protectOctal.ORD] ← BooleanItem[
      tag: "ProtectOctal"L, place: [0, line1], switch: @data.protectOctal];
    items[FormIndex.octalFile.ORD] ← StringItem[
      tag: "Octal File"L, place: [100, line1], inHeap: TRUE,
      string: @data.octalFileName];
    items[FormIndex.binaryToOctal.ORD] ← CommandItem[
      tag: "Convert Binary To Octal"L, place: [0, line2], proc: Convert];
    items[FormIndex.octalToBinary.ORD] ← CommandItem[
      tag: "Convert Octal To Binary"L, place: [200, line2], proc: Convert];
    RETURN[items: items, freeDesc: TRUE];
    END; -- MakeForm

  --====================
  -- Convert procs
  --====================

  Convert: FormSW.ProcType =
    BEGIN
    IF data.commandIsRunning THEN
      Put.Line[data.msgSW, "Tool is busy!"L]
    ELSE
      BEGIN
      convertProc: PROC ← SELECT index FROM
        FormIndex.binaryToOctal.ORD => BinaryToOctal,
        FormIndex.octalToBinary.ORD => OctalToBinary,
        ENDCASE => ERROR;
      data.commandIsRunning ← TRUE;
      Process.Detach[FORK convertProc[]];
      END;
    END; -- Convert

  BinaryToOctal: PROC =
    BEGIN
    PutByteOctal: PROC [byte: Environment.Byte] =
```

```
      BEGIN
      NibbleToChar: PROC [n: CARDINAL] RETURNS [c: CHAR] = INLINE
        BEGIN
        c ← SELECT n FROM
          IN [0..7] => VAL[ORD['0] + n],
          ENDCASE => ERROR;
        END; -- NibbleToChar
      Stream.PutChar[octalStream, NibbleToChar[Inline.BITSHIFT[Inline.BITAND[byte, 300B], -6]]];
      Stream.PutChar[octalStream, NibbleToChar[Inline.BITSHIFT[Inline.BITAND[byte, 070B], -3]]];
      Stream.PutChar[octalStream, NibbleToChar[Inline.BITAND[byte, 007B]]];
      END; -- PutByteOctal
    wordCount: LONG INTEGER ← 0;
    wordString: LONG STRING ← [20];
    binaryStream, octalStream: Stream.Handle ← NIL;
    handle: Event.Handle ← Event.StartingProcess["Octal Convert Tool is running"L];
    binaryStream ← MStream.ReadOnly[data.binaryFileName, [NIL, NIL] !
      MStream.Error => CONTINUE];
    IF binaryStream = NIL THEN {
      CleanUp["File not available!\n"L, handle];
      RETURN};
    octalStream ← GetWriteStream[data.octalFileName, ~data.protectOctal, text];
    IF octalStream = NIL THEN {
      Stream.Delete[binaryStream];
      CleanUp["Output file is protected!\n"L, handle];
      RETURN};
    PutBoth["Converting binary file """L];
    PutBoth[data.binaryFileName];
    PutBoth["" to octal file """L];
    PutBoth[data.octalFileName];
    PutBoth[""" ..."L];
    WriteHeader[octalStream];
    Stream.PutString[octalStream, "\n-- "L];
    String.AppendLongNumber[wordString, wordCount, 8];
    Stream.PutString[octalStream, wordString];
    Stream.PutChar[octalStream, Ascii.CR];
    Stream.PutChar[octalStream, Ascii.CR];
    DO
      ENABLE Stream.EndOfStream => EXIT;
      THROUGH [0..8) DO
        THROUGH [0 .. 8) DO
          PutByteOctal[Stream.GetByte[binaryStream]];
          Stream.PutChar[octalStream, '.];
          PutByteOctal[Stream.GetByte[binaryStream]];
          Stream.PutChar[octalStream, Ascii.SP];
          Stream.PutChar[octalStream, Ascii.SP];
        ENDLOOP;
        Stream.PutChar[octalStream, Ascii.CR];
      ENDLOOP;
      wordCount ← wordCount + 64;
      Stream.PutString[octalStream, "\n\n-- "L];
      wordString.length ← 0;
      String.AppendLongNumber[wordString, wordCount, 8];
      Stream.PutString[octalStream, wordString];
      Stream.PutChar[octalStream, Ascii.CR];
      Stream.PutChar[octalStream, Ascii.CR];
    ENDLOOP;
    Stream.PutChar[octalStream, Ascii.CR];
    Stream.PutChar[octalStream, Ascii.CR];
    Stream.Delete[binaryStream];
    MStream.SetLength[octalStream, MStream.GetLength[octalStream]];
    Stream.Delete[octalStream];
    CleanUp[".. Done!\n"L, handle];
    END; -- BinaryToOctal

  OctalToBinary: PROC =
    BEGIN
    binaryStream, octalStream: Stream.Handle ← NIL;
    byte: Environment.Byte ← 0;
    count: INTEGER ← 0;
    char: CHARACTER;
    handle: Event.Handle ← Event.StartingProcess["Octal Convert Tool is running"L];
    octalStream ← MStream.ReadOnly[data.octalFileName, [NIL, NIL] !
      MStream.Error => CONTINUE];
    IF octalStream = NIL THEN {
      CleanUp["File not available!\n"L, handle];
      RETURN};
    binaryStream ← GetWriteStream[data.binaryFileName, ~data.protectBinary, binary];
    IF binaryStream = NIL THEN {
      Stream.Delete[octalStream];
      CleanUp["Output file is protected!\n"L, handle];
      RETURN};
    PutBoth["Converting octal file """L];
    PutBoth[data.octalFileName];
    PutBoth[""" to binary file """L];
    PutBoth[data.binaryFileName];
    PutBoth[""" ..."L];
    DO
      BEGIN
      ENABLE Stream.EndOfStream => EXIT;
      char ← Stream.GetChar[octalStream];
      SELECT char FROM
        IN ['0..'7] => {
          byte ← byte * 8 + char - '0;
          IF (count ← count + 1) = 3 THEN {
            Stream.PutByte[binaryStream, byte];
            byte ← 0;
            count ← 0;
```

```
            };
          };
        Ascii.CR, Ascii.SP, '. => NULL;
        '- =>
          BEGIN
          IF Stream.GetChar[octalStream] = '- THEN
            UNTIL Stream.GetChar[octalStream] = Ascii.CR DO ENDLOOP
          ELSE
            GOTO badOctalChar;
          END;
        ENDCASE => GOTO badOctalChar;
      EXITS badOctalChar =>
        BEGIN
        Stream.Delete[octalStream];
        MStream.SetLength[binaryStream, 0];
        Stream.Delete[binaryStream];
        CleanUp["Error: Illegal character encountered in octal file!\n"L, handle];
        RETURN;
        END;
      END;
    ENDLOOP;
    Stream.Delete[octalStream];
    MStream.SetLength[binaryStream, MStream.GetLength[binaryStream]];
    Stream.Delete[binaryStream];
    CleanUp[".. Done!\n"L, handle];
    END; -- OctalToBinary

  GetWriteStream: PROC [name: LONG STRING, overwrite: BOOLEAN, type: MFile.Type]
   RETURNS [stream: Stream.Handle ← NIL] =
    BEGIN
    stream ← MStream.ReadOnly[name, [NIL, NIL] ! MStream.Error => CONTINUE];
    IF stream # NIL THEN {
      Stream.Delete[stream]; IF ~overwrite THEN RETURN[NIL]};
    stream ← MStream.WriteOnly[name, [NIL, NIL], type];
    END; -- GetWriteStream

  WriteHeader: PROC [stream: Stream.Handle] =
    BEGIN
    dateAndTime: LONG STRING ← [40];
    Time.AppendCurrent[dateAndTime, TRUE];
    Stream.PutString[stream, "-- File: "L];
    Stream.PutString[stream, data.octalFileName];
    Stream.PutString[stream, "\n-- From: "L];
    Stream.PutString[stream, data.binaryFileName];
    Stream.PutString[stream, "\n-- Date: "L];
    Stream.PutString[stream, dateAndTime];
    Stream.PutString[stream, "\n\n"L];
    END; -- WriteHeader

  CleanUp: PROC [reason: LONG STRING, event: Event.Handle] =
    BEGIN
    PutBoth[reason];
    data.commandIsRunning ← FALSE;
    Event.DoneWithProcess[event];
    END; -- CleanUp

  PutBoth: PROC [s: LONG STRING] = {
    Put.Text[data.msgSW, s]; Put.Text[data.logSW, s]};


  ---------------------
  -- Mainline code
  ---------------------

   Init[];

END...

LOG (Editor/Date/Comment):
  Trow   7-Sep-89 17:22:43      Adapted from HexConvertToolImpl to deal with files of XCharacters.
```

```
-- TelegraphCodeConvertToolImpl.mesa
-- Trow:PARC:Xerox  9-Sep-89 16:03:23

-- Copyright (c) 1989 by Xerox Corporation.  All rights reserved.

DIRECTORY
  Ascii USING [CR, SP],
  Environment USING [Byte],
  Event USING [DoneWithProcess, Handle, StartingProcess, toolWindow],
  EventTypes USING [deactivate],
  Exec USING [AddCommand, ExecProc, RemoveCommand],
  FormSW USING [AllocateItemDescriptor, BooleanItem, ClientItemsProcType,
    CommandItem, ItemHandle, line0, line1, line2, ProcType, StringItem],
  Heap USING [Create, Delete],
  Inline USING [BITAND, BITSHIFT],
  MFile USING [Type],
  MStream USING [Error, GetLength, ReadOnly, SetLength, WriteOnly],
  Process USING [Detach],
  Put USING [Line, Text],
  Runtime USING [GetBcdTime],
  Stream USING [Delete, EndOfStream, GetChar, GetByte, Handle, PutByte, PutChar,
    PutString],
  String USING [AppendString, AppendLongDecimal],
  Supervisor USING [AddDependency, AgentProcedure, CreateSubsystem,
    EnumerationAborted, RemoveDependency, SubsystemHandle],
  Time USING [Append, AppendCurrent, Unpack],
  Tool USING [Create, Destroy, MakeFileSW, MakeFormSW, MakeMsgSW, MakeSWsProc,
    UnusedLogName],
  ToolWindow USING [Activate, GetState, TransitionProcType],
  Window USING [GetChild, GetParent, Handle, Stack, ValidateTree];

TelegraphCodeConvertToolImpl: PROGRAM
  IMPORTS Event, Exec, FormSW, Heap, Inline, MStream, Process, Put,
    Runtime, Stream, String, Supervisor, Time, Tool, ToolWindow, Window
  =
  BEGIN

  ----------------------
  -- Types
  ----------------------

  FormIndex: TYPE = {protectBinary, binaryFile, protectOctal, octalFile, binaryToOctal,
    octalToBinary};

  ToolData: TYPE = MACHINE DEPENDENT RECORD [
    msgSW(0): Window.Handle ← NIL,
    formSW(2): Window.Handle ← NIL,
    logSW(4): Window.Handle ← NIL,
    protectBinary(6): BOOLEAN ← TRUE,
    protectOctal(7): BOOLEAN ← TRUE,
    binaryFileName(8): LONG STRING ← NIL,
    octalFileName(10): LONG STRING ← NIL,
    commandIsRunning(12): BOOLEAN ← FALSE];

  ----------------------
  -- Constants
  ----------------------

  agent: Supervisor.SubsystemHandle = Supervisor.CreateSubsystem[CheckDeactivate];

  ----------------------
  -- Globals
  ----------------------

  data: LONG POINTER TO ToolData ← NIL;
  wh: Window.Handle ← NIL;
  heap: UNCOUNTED ZONE ← NIL;

  ----------------------
  -- Initialization
  ----------------------

  Init: PROC =
    BEGIN
    Exec.AddCommand["TelegraphCodeConvertTool.~"L, MakeTool, NIL, Unload];
    END; -- Init

  MakeTool: Exec.ExecProc =
    BEGIN
    IF wh = NIL THEN
      BEGIN
      name: LONG STRING ← [60];
      String.AppendString[to: name, from : "TelegraphCode Convert Tool of "L];
      Time.Append[s: name, unpacked: Time.Unpack[Runtime.GetBcdTime[]]];
      name.length ← name.length - 3; -- lop the seconds
      wh ← Tool.Create[name: name, makeSWsProc: MakeSWs,
        clientTransition: ClientTransition, cmSection: "TelegraphCodeConvertTool"L,
        tinyName1: "TelegraphCode"L, tinyName2: "Convert"L];
      END
    ELSE IF ToolWindow.GetState[wh] = active THEN
      BEGIN
      newSibling: Window.Handle = Window.GetChild[Window.GetParent[wh]];
      IF wh # newSibling THEN {
        Window.Stack[wh, newSibling]; Window.ValidateTree[wh]};
      END
    ELSE
      ToolWindow.Activate[wh];
```

```
  END; -- MakeTool

Unload: Exec.ExecProc =
  BEGIN
  IF wh # NIL THEN {Tool.Destroy[wh]; wh ← NIL};
  Exec.RemoveCommand[h, "TelegraphCodeConvertTool.~"L];
  END; -- Unload

---------------------
-- State change
---------------------

CheckDeactivate: Supervisor.AgentProcedure =
  BEGIN
  IF event = EventTypes.deactivate AND wh # NIL
   AND wh = eventData AND data.commandIsRunning THEN
    BEGIN
    Put.Line[data.msgSW, "Tool is busy!"L];
    ERROR Supervisor.EnumerationAborted;
    END;
  END; -- CheckDeactivate

ClientTransition: ToolWindow.TransitionProcType =
  BEGIN
  SELECT TRUE FROM
    old = inactive =>
      BEGIN
      IF heap = NIL THEN heap ← Heap.Create[initial: 1, increment: 1];
      IF data = NIL THEN data ← heap.NEW[ToolData ← []];
      END;
    new = inactive =>
      BEGIN
      Supervisor.RemoveDependency[client: agent, implementor: Event.toolWindow];
      IF data # NIL THEN heap.FREE[@data];
      IF heap # NIL THEN {Heap.Delete[heap]; heap ← NIL};
      END;
    ENDCASE;
  END; -- ClientTransition

---------------------
-- Tool window
---------------------

MakeSWs: Tool.MakeSWsProc =
  BEGIN
  logFileName: STRING = [50];
  Tool.UnusedLogName[unused: logFileName, root: "TelegraphCodeConvertTool.log"L];
  data.msgSW ← Tool.MakeMsgSW[window: window];
  data.formSW ← Tool.MakeFormSW[window: window, formProc: MakeForm];
  data.logSW ← Tool.MakeFileSW[window: window, name: logFileName];
  Supervisor.AddDependency[client: agent, implementor: Event.toolWindow];
  END; -- MakeSWs

MakeForm: FormSW.ClientItemsProcType =
  BEGIN
  OPEN FormSW;
  nItems: CARDINAL = FormIndex.LAST.ORD + 1;
  items ← AllocateItemDescriptor[nItems];
  items[FormIndex.protectBinary.ORD] ← BooleanItem[
    tag: "ProtectBinary"L, place: [0, line0], switch: @data.protectBinary];
  items[FormIndex.binaryFile.ORD] ← StringItem[
    tag: "Binary File"L, place: [100, line0], inHeap: TRUE,
    string: @data.binaryFileName];
  items[FormIndex.protectOctal.ORD] ← BooleanItem[
    tag: "ProtectOctal"L, place: [0, line1], switch: @data.protectOctal];
  items[FormIndex.octalFile.ORD] ← StringItem[
    tag: "Octal File"L, place: [100, line1], inHeap: TRUE,
    string: @data.octalFileName];
  items[FormIndex.binaryToOctal.ORD] ← CommandItem[
    tag: "Convert Binary To Octal"L, place: [0, line2], proc: Convert];
  items[FormIndex.octalToBinary.ORD] ← CommandItem[
    tag: "Convert Octal To Binary"L, place: [200, line2], proc: Convert];
  RETURN[items: items, freeDesc: TRUE];
  END; -- MakeForm

---------------------
-- Convert procs
---------------------

Convert: FormSW.ProcType =
  BEGIN
  IF data.commandIsRunning THEN
    Put.Line[data.msgSW, "Tool is busy!"L]
  ELSE
    BEGIN
    convertProc: PROC ← SELECT index FROM
      FormIndex.binaryToOctal.ORD => BinaryToOctal,
      FormIndex.octalToBinary.ORD => OctalToBinary,
      ENDCASE => ERROR;
    data.commandIsRunning ← TRUE;
    Process.Detach[FORK convertProc[]];
    END;
  END; -- Convert

B   yToOctal: PROC =
    IN
    ByteOctal: PROC [byte: Environment.Byte] =
```

```
    BEGIN
    NibbleToChar: PROC [n: CARDINAL] RETURNS [c: CHAR] = INLINE
      BEGIN
      c ← SELECT n FROM
        IN [0..7] => VAL[ORD['0] + n],
        ENDCASE => ERROR;
      END; -- NibbleToChar
    Stream.PutChar[octalStream, NibbleToChar[Inline.BITSHIFT[Inline.BITAND[byte, 300B], -6]]];
    Stream.PutChar[octalStream, NibbleToChar[Inline.BITSHIFT[Inline.BITAND[byte, 070B], -3]]];
    Stream.PutChar[octalStream, NibbleToChar[Inline.BITAND[byte, 007B]]];
    END; -- PutByteOctal
  wordCount: LONG INTEGER ← 0;
  wordString: LONG STRING ← [20];
  binaryStream, octalStream: Stream.Handle ← NIL;
  handle: Event.Handle ← Event.StartingProcess["TelegraphCode Convert Tool is running"L];
  binaryStream ← MStream.ReadOnly[data.binaryFileName, [NIL, NIL] !
    MStream.Error => CONTINUE];
  IF binaryStream = NIL THEN {
    CleanUp["File not available!\n"L, handle];
    RETURN};
  octalStream ← GetWriteStream[data.octalFileName, ~data.protectOctal, text];
  IF octalStream = NIL THEN {
    Stream.Delete[binaryStream];
    CleanUp["Output file is protected!\n"L, handle];
    RETURN};
  PutBoth["Converting binary file """L];
  PutBoth[data.binaryFileName];
  PutBoth[""" to octal file """L];
  PutBoth[data.octalFileName];
  PutBoth[""" ..."L];
  WriteHeader[octalStream];
  Stream.PutString[octalStream, "\n-- "L];
  String.AppendLongDecimal[wordString, wordCount];
  Stream.PutString[octalStream, wordString];
  Stream.PutChar[octalStream, Ascii.CR];
  Stream.PutChar[octalStream, Ascii.CR];
  DO
    ENABLE Stream.EndOfStream => EXIT;
    THROUGH [0..10) DO
      THROUGH [0 .. 10) DO
        PutByteOctal[Stream.GetByte[binaryStream]];
        Stream.PutChar[octalStream, '.];
        PutByteOctal[Stream.GetByte[binaryStream]];
        Stream.PutChar[octalStream, Ascii.SP];
        Stream.PutChar[octalStream, Ascii.SP];
      ENDLOOP;
      Stream.PutChar[octalStream, Ascii.CR];
    ENDLOOP;
    wordCount ← wordCount + 100;
    Stream.PutString[octalStream, "\n\n-- "L];
    wordString.length ← 0;
    String.AppendLongDecimal[wordString, wordCount];
    Stream.PutString[octalStream, wordString];
    Stream.PutChar[octalStream, Ascii.CR];
    Stream.PutChar[octalStream, Ascii.CR];
  ENDLOOP;
  Stream.PutChar[octalStream, Ascii.CR];
  Stream.PutChar[octalStream, Ascii.CR];
  Stream.Delete[binaryStream];
  MStream.SetLength[octalStream, MStream.GetLength[octalStream]];
  Stream.Delete[octalStream];
  CleanUp[".. Done!\n"L, handle];
  END; -- BinaryToOctal

OctalToBinary: PROC =
  BEGIN
  binaryStream, octalStream: Stream.Handle ← NIL;
  byte: Environment.Byte ← 0;
  count: INTEGER ← 0;
  char: CHARACTER;
  handle: Event.Handle ← Event.StartingProcess["TelegraphCode Convert Tool is running"L];
  octalStream ← MStream.ReadOnly[data.octalFileName, [NIL, NIL] !
    MStream.Error => CONTINUE];
  IF octalStream = NIL THEN {
    CleanUp["File not available!\n"L, handle];
    RETURN};
  binaryStream ← GetWriteStream[data.binaryFileName, ~data.protectBinary, binary];
  IF binaryStream = NIL THEN {
    Stream.Delete[octalStream];
    CleanUp["Output file is protected!\n"L, handle];
    RETURN};
  PutBoth["Converting octal file """L];
  PutBoth[data.octalFileName];
  PutBoth[""" to binary file """L];
  PutBoth[data.binaryFileName];
  PutBoth[""" ..."L];
  DO
    BEGIN
    ENABLE Stream.EndOfStream => EXIT;
    char ← Stream.GetChar[octalStream];
    SELECT char FROM
      IN ['0..'7] => {
        byte ← byte * 8 + char - '0;
        IF (count ← count + 1) = 3 THEN {
          Stream.PutByte[binaryStream, byte];
          byte ← 0;
          count ← 0;
```

```
            };
          };
        Ascii.CR, Ascii.SP, '. => NULL;
        '- =>
          BEGIN
          IF Stream.GetChar[octalStream] = '- THEN
            UNTIL Stream.GetChar[octalStream] = Ascii.CR DO ENDLOOP
          ELSE
            GOTO badOctalChar;
          END;
        ENDCASE => GOTO badOctalChar;
      EXITS badOctalChar =>
        BEGIN
        Stream.Delete[octalStream];
        MStream.SetLength[binaryStream, 0];
        Stream.Delete[binaryStream];
        CleanUp["Error: Illegal character encountered in octal file!\n"L, handle];
        RETURN;
        END;
      END;
    ENDLOOP;
    Stream.Delete[octalStream];
    MStream.SetLength[binaryStream, MStream.GetLength[binaryStream]];
    Stream.Delete[binaryStream];
    CleanUp[".. Done!\n"L, handle];
    END; -- OctalToBinary

  GetWriteStream: PROC [name: LONG STRING, overwrite: BOOLEAN, type: MFile.Type]
   RETURNS [stream: Stream.Handle ← NIL] =
    BEGIN
    stream ← MStream.ReadOnly[name, [NIL, NIL] ! MStream.Error => CONTINUE];
    IF stream # NIL THEN {
      Stream.Delete[stream]; IF ~overwrite THEN RETURN[NIL]};
    stream ← MStream.WriteOnly[name, [NIL, NIL], type];
    END; -- GetWriteStream

  WriteHeader: PROC [stream: Stream.Handle] =
    BEGIN
    dateAndTime: LONG STRING ← [40];
    Time.AppendCurrent[dateAndTime, TRUE];
    Stream.PutString[stream, "-- File: "L];
    Stream.PutString[stream, data.octalFileName];
    Stream.PutString[stream, "\n-- From: "L];
    Stream.PutString[stream, data.binaryFileName];
    Stream.PutString[stream, "\n-- Date: "L];
    Stream.PutString[stream, dateAndTime];
    Stream.PutString[stream, "\n\n"L];
    END; -- WriteHeader

  CleanUp: PROC [reason: LONG STRING, event: Event.Handle] =
    BEGIN
    PutBoth[reason];
    data.commandIsRunning ← FALSE;
    Event.DoneWithProcess[event];
    END; -- CleanUp

  PutBoth: PROC [s: LONG STRING] = {
    Put.Text[data.msgSW, s]; Put.Text[data.logSW, s]};


  --====================
  -- Mainline code
  --====================

   Init[];

END...

LOG (Editor/Date/Comment):
  Trow   7-Sep-89 22:43:58      Adapted from HexConvertToolImpl to deal with files of XCharacters arranged in telegraph code order.
```

```
-- File: CvXlit.mesa
-- Trow 7-Sep-89 16:32:38

-- Last Revised by: Erickson                                    24-Nov-16.53:30
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (c) 1984, 1985, 1986, 1987, 1989 by Xerox Corporation  All rights reserved

DIRECTORY
  Converter
    USING [ConvertProc, CvData, DestroyProc, DependentOptionProc, MenuItemProc],
  NSFile
    USING [Reference],
  Window
    USING [Handle],
  XChar
    USING [Character],
  XString
    USING [Reader, ReaderBody, Writer];

<<
--=================================================================================
-- OVERVIEW:

Private definitions interface for the ascii conversion.
--=================================================================================
>>

CvXlit: DEFINITIONS =
BEGIN

--====================
-- CONSTANTS
--====================

modern: CARDINAL = 0;
classic: CARDINAL = 1;

twelve: CARDINAL = 0;
eightteen: CARDINAL = 1;
twentyFour: CARDINAL = 2;

unlimited: CARDINAL = 0;
limited: CARDINAL = 1;

dfltFont: CARDINAL = classic;
dfltFontSize: CARDINAL = twelve;
dfltTrailing: BOOLEAN = FALSE;
dfltLineLen: CARDINAL = unlimited;
dfltChars: CARDINAL = 80;
dfltWordWrap: BOOLEAN = TRUE;

leadingMargin: CARDINAL = 2;
pointsBetweenItems: CARDINAL = 10;

--====================
-- TYPES
--====================

AsciiToVPTable: TYPE = LONG POINTER TO ARRAY CHARACTER OF XChar.Character;
VPToAsciiTable: TYPE = LONG POINTER TO ARRAY [0..256) OF LONG POINTER TO ARRAY [0..256) OF CHARACTER;

Boolean: TYPE = MACHINE DEPENDENT RECORD[
  zeros(0:0..14): [0..7777B], value(0:15..15): BOOLEAN];

Common: TYPE = LONG POINTER TO CommonData;
CommonData: TYPE = RECORD [
  cvData: Converter.CvData,
  options: BOOLEAN,
  window: Window.Handle,
  owner: Owners,
  ref: NSFile.Reference,
  f: CommonObj,
  textRb: FiledXStrings,
  text: EncodedText,
  z: UNCOUNTED ZONE];
<<
The same data strcture is used by all the client formwindows/details sections.
>>

Filed: TYPE = LONG POINTER TO CommonObj;
CommonObj: TYPE = MACHINE DEPENDENT RECORD [
  font: CARDINAL ← dfltFont,
  fontSize: CARDINAL ← dfltFontSize,
  ignoreTrailing: Boolean ← [0, dfltTrailing],
  lineLen: CARDINAL ← dfltLineLen,
  charsSuffix: CARDINAL ← dfltChars,
  wordWrap: Boolean ← [0, dfltWordWrap]
  ];
<<
This data structure is the filed data object, along with the various strings/text items that come from the formwindows.
>>

EncodedText: TYPE = ARRAY TextIDs OF LONG STRING;
<<
Use long strings internally, since they are better suited to ASCII text.
>>
```

```
FiledXStrings: TYPE = ARRAY TextIDs OF XString.ReaderBody;
<<
Filed strings are kept here.
>>

Owners: TYPE = {AtoVsrc, AtoVdst, VtoAdst, backstop};

TextIDs: TYPE = {
  paraEndsWith,
  atovReplaceUnknown,
  endLine,
  endPara,
  vtoaReplaceUnknown,
  avTableName,
  vaTableName
  };


--====================
-- SIGNALS
--====================

Problem: SIGNAL [err: ProblemType];

ProblemType: TYPE = {obsoleteDataFile, fatalError, doDflts, other};

--====================
-- PROCEDURES
--====================

AsciiToVP: Converter.ConvertProc;
<< = PROCEDURE [source: NSFile.Handle, cvData: Converter.CvData, session: NSFile.Session, srcInstance: LONG POINTER ← NIL, dstInstance:
LONG POINTER ← NIL, background: BOOLEAN ← FALSE] RETURNS [dest: NSFile.Handle ← LOOPHOLE[0]];

Exported by CvXlitToVPImpl.
>>


AsciiToVPSrcOps: Converter.DependentOptionProc;
<< = PROCEDURE [options: BOOLEAN ← TRUE, cvData: Converter.CvData, which: Converter.FormatToUse, srcFormat: XString.Reader, destFormat:
XString.Reader, window: Window.Handle, oldInstance: LONG POINTER ← NIL] RETURNS [menuItemProc: Converter.MenuItemProc, destroy:
Converter.DestroyProc, instance: LONG POINTER];

Exported by CvXlitToVPImpl.
>>


AsciiToVPDstOps: Converter.DependentOptionProc;
<< = PROCEDURE [options: BOOLEAN ← TRUE, cvData: Converter.CvData, which: Converter.FormatToUse, srcFormat: XString.Reader, destFormat:
XString.Reader, window: Window.Handle, oldInstance: LONG POINTER ← NIL] RETURNS [menuItemProc: Converter.MenuItemProc, destroy:
Converter.DestroyProc, instance: LONG POINTER];

Exported by CvXlitToVPImpl.
>>


CommonMenu: Converter.MenuItemProc;
<< = PROCEDURE [instance: LONG POINTER, menuItem: PropertySheet.MenuItemType] RETURNS [ok: BOOLEAN ← TRUE];

Exported by CvXlitFWImpl.
>>


CreateCommon: PROC [cvData: Converter.CvData, options: BOOLEAN, window: Window.Handle, owner: Owners] RETURNS [my: Common]; --!
NSFile.Error
<<
Exported by CvXlitDataImpl.
>>


CreateFW: PROC [my: Common, window: Window.Handle, owner: Owners];
<<
Exported by CvXlitFWImpl.
>>


DataFromWindow: PROC [w: Window.Handle] RETURNS [my: Common];
<<
Exported by CvXlitMainImpl
>>


DataToWindow: PROC [my: Common, w: Window.Handle];
<<
Exported by CvXlitMainImpl
>>


DestroyCommon: Converter.DestroyProc;
<< = PROCEDURE [instance: LONG POINTER];

Exported by CvXlitDataImpl.
>>

GetAVTable: PROC RETURNS [avTable: AsciiToVPTable];
```

```
<<
Exported by CvXlitToVPImpl.
>>

GetPreMargin: PROC [item: MessageKey] RETURNS [leads: CARDINAL];
<<
Exported by CvXlitMainImpl.
>>

GetMessage: PROC [msg: MessageKey] RETURNS [msgRb: XString.ReaderBody];
<<
Exported by CvXlitMsgFileImpl.
>>

GetVATable: PROC RETURNS [vaTable: VPToAsciiTable];
<<
Exported by CvXlitFromVPImpl.
>>

InitFiledData: PROC [my: Common]; -- ! NSFile.Error
<<
Create and initialize client file. Exported by CvXlitDataImpl.
>>

LoadFiledData: PROC [my: Common]; -- ! NSFile.Error, Problem
<<
Read filed data. Exported by CvXlitDataImpl.
>>

ParseItem: PROC [my: Common, r: XString.Reader, item: MessageKey, buf: XString.Writer ← NIL] RETURNS [ok: BOOLEAN, ls: LONG STRING];
<<
Exported by CvXlitParseImpl. If ok is FALSE, error during parse. ls is NIL if item has null text. buf is a temporary buffer that will
be created and destroyed each time the proc is called if defaulted, otherwise it will just be used.
>>

StoreFiledData: PROC [my: Common]; -- ! NSFile.Error
<<
Write filed data. Exported by CvXlitDataImpl.
>>

VPToAscii: Converter.ConvertProc;
<< = PROCEDURE [source: NSFile.Handle, cvData: Converter.CvData, session: NSFile.Session, srcInstance: LONG POINTER ← NIL, dstInstance:
LONG POINTER ← NIL, background: BOOLEAN ← FALSE] RETURNS [dest: NSFile.Handle ← LOOPHOLE[0]];

Exported by CvXlitFromVPImpl.
>>

VPToAsciiDstOps: Converter.DependentOptionProc;
<< = PROCEDURE [options: BOOLEAN ← TRUE, cvData: Converter.CvData, which: Converter.FormatToUse, srcFormat: XString.Reader, destFormat:
XString.Reader, window: Window.Handle, oldInstance: LONG POINTER ← NIL] RETURNS [menuItemProc: Converter.MenuItemProc, destroy:
Converter.DestroyProc, instance: LONG POINTER];

Exported by CvXlitFromVPImpl.
>>

-- ==========================
-- MESSAGES
-- ==========================

MessageKey: TYPE = {
    asciiDoc,
    paraEndsWith,
    fontSize,
    fontSizeChoices,
    font,
    fontChoices,
    ignoreTrailing,
    lineLen,
    lineLenChoices,
    charsSuffix,
    wordWrap,
    endLine,
    endPara,
    replaceUnknown,
    transliterationTable,
    spareP1,
    spareP2,
    spareP3,
    spareP4,
    spareP5,
    lastPsheetItem,
    left,
    right,
    cr,
    lf,
    nl,
    ff,
    tab,
```

```
    createError,
    notPF,
    paginating,
    skippedTableData,
    dfltAVEndParagraph,
    dfltAVReplaceCharacter,
    prefix,
    doneFailed,
    backstop,
    metaError,
    charsOutOfBounds,
    fatalError,
    extraErr0,
    extraErr1,
    dfltVAEndLine,
    dfltVAEndParagraph,
    dfltVAReplaceCharacter
    };

END.

LOG
 5--Dec-84 15:01:26 - MSchneider.pa - CREATED
19--Dec-84 15:31:39 - MSchneider - update to BWS 4.0
16--Apr-85 10:40:52 - MSchneider - added some comments and owner statement
28--May-85 9:23:59 - MSchneider - took out messages now in common interface
26--Feb-87 16:17:12 - Caro - Added paginating and spares
18--Mar-87 14:02:39 - Caro - Completely rewritten for Enhancements I
24--Nov-87 16:51:13 - Erickson - added aToVDfltMeta to change A to V paraEndsWith default
                                 from <CR><LF> to <CR>
```

```
-- File: CvXlitDataImpl.mesa
-- Trow 31-Aug-89 15:17:25

-- Last Revised by: Erickson                    24-Nov-87 16:54:21
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (c) 1987, 1989 by Xerox Corporation.  All rights reserved.

DIRECTORY
  Courier
    USING [Description, DeserializeParameters, Error, Free,
    Parameters, SerializeParameters],
  Converter
    USING [CreateClientFile, CvData, DestroyProc, FindClientFile],
  ConverterMsg
    USING [Get, kvpDocument],
  CvXlit
    USING [Common, CommonData, CommonObj,
    GetMessage, Owners, Problem, TextIDs],
  Environment
    USING [bytesPerPage],
  Heap
    USING [Create, Delete],
  NSFile
    USING [Delete, Error, Handle, nullReference, OpenByReference],
  NSFileStream
    USING [Create, GetLength, Handle, SetLength],
  Stream
    USING [Delete, InvalidOperation],
  Window
    USING [Handle],
  XString
    USING [CopyToNewReaderBody, DescribeReaderBody, FromSTRING, nullReaderBody, ReaderBody];

<<
-- ============================================================================
-- OVERVIEW:

Data and filed data procedures
-- ============================================================================
>>

CvXlitDataImpl: PROGRAM
IMPORTS
  Converter, ConverterMsg, Courier, CvXlit, Heap,
  NSFile, NSFileStream, Stream, XString
EXPORTS
  CvXlit =
BEGIN

-- ====================
-- CONSTANTS
-- ====================

keyBits: Key = 2707974433;  --/* never change this value! */

emptyRb: XString.ReaderBody = XString.FromSTRING[""L];

currentVersion: Version = 4;  --/* change this value if you alter the filed data format */
-- ============================================================================
-- History of Versions (update each time version number changes)
--  18-Mar-87 11:48:29 -  1 - First version
--   2-Mar-89 13:42:14 -  2 - Xlit version
--  29-Aug-89 21:03:36 -  3 - Xlit version
--  31-Aug-89 12:01:31 -  4 - Xlit version
-- ============================================================================

-- ====================
-- TYPES
-- ====================

Key: TYPE = LONG CARDINAL;

Version: TYPE = INTEGER;

-- ====================
-- PUBLIC PROCEDURES
-- ====================

CreateCommon: PUBLIC PROC [cvData: Converter.CvData, options: BOOLEAN, window: Window.Handle, owner: CvXlit.Owners] RETURNS [my:
CvXlit.Common] = {
    z: UNCOUNTED ZONE ← Heap.Create[initial: 16, increment: 28];

    my ← z.NEW[CvXlit.CommonData ← [
      cvData: cvData,
      options: options,
      window: window,
      owner: owner,
      ref: NSFile.nullReference,
      textRb: ALL[XString.nullReaderBody],
      text: ALL[NIL],
      z: z]];

    --/* find client file */
    BEGIN
      ENABLE UNWIND => Heap.Delete[z];
    prefix: XString.ReaderBody ← CvXlit.GetMessage[prefix];
```

```
    src: XString.ReaderBody ← CvXlit.GetMessage[asciiDoc];
    dst: XString.ReaderBody ← ConverterMsg.Get[ConverterMsg.kvpDocument];

    my.ref ← Converter.FindClientFile[
      cvData: cvData,
      srcFormat: @src,
      destFormat: @dst,
      prefix: @prefix];

    IF my.ref = NSFile.nullReference THEN
        {
        --/* file never created, so initialize */
        InitFiledData[my]; --/* fills in my.ref */
        };

    --/* read data */
    BEGIN
      ENABLE CvXlit.Problem => {
        file: NSFile.Handle ← NSFile.OpenByReference[my.ref];
        avPara: XString.ReaderBody ← CvXlit.GetMessage[dfltAVEndParagraph];
        avChar: XString.ReaderBody ← CvXlit.GetMessage[dfltAVReplaceCharacter];
        vaLine: XString.ReaderBody ← CvXlit.GetMessage[dfltVAEndLine];
        vaPara: XString.ReaderBody ← CvXlit.GetMessage[dfltVAEndParagraph];
        vaChar: XString.ReaderBody ← CvXlit.GetMessage[dfltVAReplaceCharacter],
        --/* get rid of old file, reinitialize */
        NSFile.Delete[file];
        InitFiledData[my];
        my.textRb ← [
          paraEndsWith: avPara,
          atovReplaceUnknown: avChar,
          endLine: vaLine,
          endPara: vaPara,
          vtoaReplaceUnknown: vaChar,
          avTableName: emptyRb,
          vaTableName: emptyRb];
        CONTINUE;
        };

    LoadFiledData[my];
    END;
    END;
};

DestroyCommon: PUBLIC Converter.DestroyProc = {
<< = PROCEDURE [instance: LONG POINTER];
>>
    my: CvXlit.Common ← instance;
    z: UNCOUNTED ZONE;

    IF my = NIL THEN RETURN;
    z ← my.z;
    Heap.Delete[z];
};

InitFiledData: PUBLIC PROC [my: CvXlit.Common] = {
    myObj: CvXlit.CommonData;
    avPara: XString.ReaderBody ← CvXlit.GetMessage[dfltAVEndParagraph];
    avChar: XString.ReaderBody ← CvXlit.GetMessage[dfltAVReplaceCharacter];
    vaLine: XString.ReaderBody ← CvXlit.GetMessage[dfltVAEndLine];
    vaPara: XString.ReaderBody ← CvXlit.GetMessage[dfltVAEndParagraph];
    vaChar: XString.ReaderBody ← CvXlit.GetMessage[dfltVAReplaceCharacter];

    --/* make dummy filed data */
    myObj.textRb ← [
      paraEndsWith: avPara,
      atovReplaceUnknown: avChar,
      endLine: vaLine,
      endPara: vaPara,
      vtoaReplaceUnknown: vaChar,
      avTableName: emptyRb,
      vaTableName: emptyRb];

    --/* create client file */
    BEGIN
    prefix: XString.ReaderBody ← CvXlit.GetMessage[prefix];
    src: XString.ReaderBody ← CvXlit.GetMessage[asciiDoc];
    dst: XString.ReaderBody ← ConverterMsg.Get[ConverterMsg.kvpDocument];

    my.ref ← Converter.CreateClientFile[
      cvData: my.cvData,
      srcFormat: @src,
      destFormat: @dst,
      prefix: @prefix];
    END;

    myObj.ref ← my.ref;
    myObj.z ← my.z;
    myObj.owner ← backstop; --/* let StoreFiledData know we are initializing */
    --/* store */
    StoreFiledData[@myObj];
};

LoadFiledData: PUBLIC PROC [my: CvXlit.Common] = {
```

```
    sh: NSFileStream.Handle ← [NIL];
    file: NSFile.Handle;
    parms: Courier.Parameters;
    tz: UNCOUNTED ZONE ← NIL;

    --/* read filed data */
    BEGIN
      ENABLE
        {
        Courier.Error, Stream.InvalidOperation => NSFile.Error[[access[fileDamaged]]];
        UNWIND =>
            {
            IF sh # NSFileStream.Handle[NIL] THEN Stream.Delete[sh];
            IF tz # NIL THEN Heap.Delete[tz];
            };
        };

    --/* open data file */
    file ← NSFile.OpenByReference[my.ref];

    --/* open read stream on data file */
    sh ← NSFileStream.Create[file: file, closeOnDelete: TRUE];

    --/* create temporary zone for disjoint data */
    tz ← Heap.Create[(NSFileStream.GetLength[sh]/Environment.bytesPerPage) + 2];

    --/* read key */
    BEGIN
    key: Key;
    parms ← [location: @key, description: DescribeKey];
    Courier.DeserializeParameters[parms, sh, tz];
    IF key # keyBits THEN
        {
        --/* quit */
        Courier.Free[parms, tz];
        Stream.Delete[sh];
        sh ← [NIL];
        SIGNAL CvXlit.Problem[obsoleteDataFile];
        };
    Courier.Free[parms, tz];
    END;

    --/* read version */
    BEGIN
    ver: Version;
    parms ← [location: @ver, description: DescribeVersion];
    Courier.DeserializeParameters[parms, sh, tz];
    IF ver # currentVersion THEN
        {
        --/* quit */
        Courier.Free[parms, tz];
        Stream.Delete[sh];
        sh ← [NIL];
        SIGNAL CvXlit.Problem[obsoleteDataFile];
        };
    Courier.Free[parms, tz];
    END;

    --/* read commonObj */
    parms ← [location: @my.f, description: DescribeCommonObj];
    Courier.DeserializeParameters[parms, sh, tz];

    --/* read paraEndsWith */
    BEGIN
    rb: XString.ReaderBody;
    parms ← [location: @rb, description: XString.DescribeReaderBody];
    Courier.DeserializeParameters[parms, sh, tz];
    my.textRb[paraEndsWith] ← XString.CopyToNewReaderBody[@rb, my.z];
    Courier.Free[parms, tz];
    END;

    --/* read atovReplaceUnknown */
    BEGIN
    rb: XString.ReaderBody;
    parms ← [location: @rb, description: XString.DescribeReaderBody];
    Courier.DeserializeParameters[parms, sh, tz];
    my.textRb[atovReplaceUnknown] ← XString.CopyToNewReaderBody[@rb, my.z];
    Courier.Free[parms, tz];
    END;

    --/* read endLine */
    BEGIN
    rb: XString.ReaderBody;
    parms ← [location: @rb, description: XString.DescribeReaderBody];
    Courier.DeserializeParameters[parms, sh, tz];
    my.textRb[endLine] ← XString.CopyToNewReaderBody[@rb, my.z];
    Courier.Free[parms, tz];
    END;

    --/* read endPara */
    BEGIN
    rb: XString.ReaderBody;
    parms ← [location: @rb, description: XString.DescribeReaderBody];
    Courier.DeserializeParameters[parms, sh, tz];
    my.textRb[endPara] ← XString.CopyToNewReaderBody[@rb, my.z];
    Courier.Free[parms, tz];
    END;
```

```
        --/* read vtoaReplaceUnknown */
        BEGIN
        rb: XString.ReaderBody;
        parms ← [location: @rb, description: XString.DescribeReaderBody];
        Courier.DeserializeParameters[parms, sh, tz];
        my.textRb[vtoaReplaceUnknown] ← XString.CopyToNewReaderBody[@rb, my.z];
        Courier.Free[parms, tz];
        END;

        --/* read avTableName */
        BEGIN
        rb: XString.ReaderBody;
        parms ← [location: @rb, description: XString.DescribeReaderBody];
        Courier.DeserializeParameters[parms, sh, tz];
        my.textRb[avTableName] ← XString.CopyToNewReaderBody[@rb, my.z];
        Courier.Free[parms, tz];
        END;

        --/* read vaTableName */
        BEGIN
        rb: XString.ReaderBody;
        parms ← [location: @rb, description: XString.DescribeReaderBody];
        Courier.DeserializeParameters[parms, sh, tz];
        my.textRb[vaTableName] ← XString.CopyToNewReaderBody[@rb, my.z];
        Courier.Free[parms, tz];
        END;

        END;

        --/* clean up */
        Stream.Delete[sh];
        Heap.Delete[tz];

};  -- LoadFiledData


--==========================================================================================
-- StoreFiledData
-- * This is tricky, since common data is used.  This routine could be called
-- * three different times, with different subsets of data, but the whole
-- * file must be written each time.
--==========================================================================================

StoreFiledData: PUBLIC PROC [my: CvXlit.Common] = {
        dataFile: NSFile.Handle;
        sh: NSFileStream.Handle;
        parms: Courier.Parameters;
        tmpMy: CvXlit.CommonData;

        --/* fill out dummy */
        tmpMy ← my↑;
        IF my.owner # backstop THEN
                LoadFiledData[@tmpMy];

        --/* open data file */
        dataFile ← NSFile.OpenByReference[my.ref];

        --/* open stream on file */
        sh ← NSFileStream.Create[file: dataFile, closeOnDelete: TRUE];
        NSFileStream.SetLength[fileStream: sh, lengthInBytes: 0];

        --/* write data */
        BEGIN
          ENABLE
            {
            Courier.Error, Stream.InvalidOperation => NSFile.Error[[access[fileDamaged]]];
            UNWIND => Stream.Delete[sh];
            };

        --/* write key */
        BEGIN
        key: Key ← keyBits;
        parms ← [location: @key, description: DescribeKey];
        Courier.SerializeParameters[parms, sh];
        END;

        --/* write version */
        BEGIN
        ver: Version ← currentVersion;
        parms ← [location: @ver, description: DescribeVersion];
        Courier.SerializeParameters[parms, sh];
        END;

        --/* update portions of data record */
        SELECT my.owner FROM
                AtoVsrc =>
                        {
                        tmpMy.textRb[paraEndsWith] ← my.textRb[paraEndsWith];
                        };
                AtoVdst =>
                        {
                        tmpMy.f.font ← my.f.font;
                        tmpMy.f.fontSize ← my.f.fontSize;
                        tmpMy.textRb[atovReplaceUnknown] ← my.textRb[atovReplaceUnknown];
                        tmpMy.f.ignoreTrailing ← my.f.ignoreTrailing;
```

```
            tmpMy.textRb[avTableName] ← my.textRb[avTableName];
            };
        VtoAdst => 
            {
            tmpMy.f.lineLen ← my.f.lineLen;
            tmpMy.f.charsSuffix ← my.f.charsSuffix;
            tmpMy.f.wordWrap ← my.f.wordWrap;
            tmpMy.textRb[endLine] ← my.textRb[endLine];
            tmpMy.textRb[endPara] ← my.textRb[endPara];
            tmpMy.textRb[vtoaReplaceUnknown] ← my.textRb[vtoaReplaceUnknown];
            tmpMy.textRb[vaTableName] ← my.textRb[vaTableName];
            };
        ENDCASE;

    --/* write filed data record */
    parms ← [location: @tmpMy.f, description: DescribeCommonObj];
    Courier.SerializeParameters[parms, sh];

    --/* write paraEndsWith string */
    parms ← [location: @tmpMy.textRb[paraEndsWith], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write atovReplaceUnknown string */
    parms ← [location: @tmpMy.textRb[atovReplaceUnknown], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write endLine string */
    parms ← [location: @tmpMy.textRb[endLine], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write endPara string */
    parms ← [location: @tmpMy.textRb[endPara], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write vtoaReplaceUnknown string */
    parms ← [location: @tmpMy.textRb[vtoaReplaceUnknown], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write avTableName string */
    parms ← [location: @tmpMy.textRb[avTableName], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    --/* write vaTableName string */
    parms ← [location: @tmpMy.textRb[vaTableName], description: XString.DescribeReaderBody];
    Courier.SerializeParameters[parms, sh];

    END;

    Stream.Delete[sh];
};

--=====================
-- PROCEDURES
--=====================

DescribeKey: Courier.Description = {
    p: LONG POINTER TO Key = notes.noteSize[SIZE[Key]];
    notes.noteLongCardinal[p];
};

DescribeVersion: Courier.Description = {
    p: LONG POINTER TO Version = notes.noteSize[SIZE[Version]];
};

DescribeCommonObj: Courier.Description = {
    p: LONG POINTER TO CvXlit.CommonObj = notes.noteSize[
      SIZE[CvXlit.CommonObj]];
};

END...

LOG
16-Mar-87 14:06:16 - Caro - Created
24-Nov-87 16:55:56 - Erickson - Changed default setting of paraEndsWith
                                to <CR> instead of <CR><LF>
```

```
-- File: CvXlitFromVPImpl.mesa
-- Trow 5-Sep-89 23:14:31

-- Last Revised by: Caro                                16-Sep-87 12:21:45
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (c) 1987, 1989 by Xerox Corporation.  All rights reserved.

DIRECTORY
  Ascii
    USING [CR, FF, LF, SP, TAB],
  BackgroundProcess
    USING [UserAbort],
  BWSZone
    USING [Permanent],
  Converter
    USING [ConvertProc, CvData, DependentOptionProc, DestroyProc, MenuItemProc, PostMessage],
  ConverterMsg,
  CvXlit
    USING [Common, CommonMenu, CreateCommon, CreateFW,
    DestroyCommon, GetMessage, limited, MessageKey, Owners,
    ParseItem, Problem, unlimited],
  DocInterchangeDefs
    USING [Close, Doc, Enumerate, EnumProcsRecord, Error, NewParagraphProc, Open, OpenStatus, PFCProc, PageBreakProc, TextProc],
  DocInterchangePropsDefs
    USING [StreakSuccession],
  Environment
    USING [wordsPerPage],
  NSFile
    USING [Attribute, Create, Delete, Error, GetReference, Handle, nullHandle, Session],
  NSFileStream
    USING [Create, Handle],
  Space
    USING [ScratchMap],
  StarFileTypes
    USING [text],
  Stream
    USING [PutChar, Delete],
  String
    USING [MakeString],
  TIP
    USING [UserAbort],
  XChar
    USING [Code, Set],
  XMessage
    USING [MsgKey],
  XString
    USING [InvalidEncoding, ReaderBody, Map, MapCharProc];

<<
--==========================================================================================
-- OVERVIEW:

VP to ASCII encoded Xlit conversion.
--==========================================================================================
>>

CvXlitFromVPImpl: PROGRAM
IMPORTS
  BackgroundProcess, BWSZone, Converter, ConverterMsg, CvXlit,
  DocInterchangeDefs,
  NSFile, NSFileStream, Space, Stream, String, TIP, XChar, XString
EXPORTS
  CvXlit =
BEGIN

--==================
-- CONSTANTS
--==================

tabInterval: CARDINAL = 8;

setMapSize: CARDINAL = SIZE[VPToAsciiSetMap];
charMapSize: CARDINAL = SIZE[VPToAsciiCharMap];

--==================
-- TYPES
--==================

VAData: TYPE = LONG POINTER TO VADataObj;
VADataObj: TYPE = RECORD [
    source: NSFile.Handle,
    output: NSFileStream.Handle,  --/* created from dest */
    cvData: Converter.CvData,
    session: NSFile.Session,
    dst: CvXlit.Common,
    background: BOOLEAN,
    doc: DocInterchangeDefs.Doc,
    putc: PutCProc,
    firstPara: BOOLEAN,
    line: LONG STRING,                  --/* line buffer */
    n: CARDINAL,                        --/* index of next char in line buffer */
    pos: CARDINAL,                      --/* current position on virtual line */
    max: CARDINAL,                      --/* last column in line */
    lastWhite: CARDINAL,                --/* last white */
    wordWrap: BOOLEAN,
    after: CARDINAL,                    --/* number of eop strings to output */
```

```
                                                --/* for previous paragraph */
        streak: DocInterchangePropsDefs.StreakSuccession,
        z: UNCOUNTED ZONE];

PutCProc: TYPE = PROC [va: VAData, c: CHARACTER];

VPToAsciiSetMap: TYPE = ARRAY [0..256) OF LONG POINTER TO VPToAsciiCharMap;
VPToAsciiCharMap: TYPE = ARRAY [0..256) OF CHARACTER;

--=====================
-- GLOBALS
--=====================

Global: TYPE = RECORD [
        defaultMap: LONG POINTER TO VPToAsciiSetMap,
        userMap: LONG POINTER TO VPToAsciiSetMap,
        gz: UNCOUNTED ZONE];

g: Global;

--=====================
-- PUBLIC PROCEDURES
--=====================

VPToAscii: PUBLIC Converter.ConvertProc = {
<< = PROCEDURE [source: NSFile.Handle, cvData: Converter.CvData, session: NSFile.Session, srcInstance: LONG POINTER ← NIL, dstInstance:
LONG POINTER ← NIL, background: BOOLEAN ← FALSE] RETURNS [dest: NSFile.Handle ← LOOPHOLE[0]];
>>
    ENABLE CvXlit.Problem, NSFile.Error, XString.InvalidEncoding => {
        msgRb: XString.ReaderBody ← CvXlit.GetMessage[fatalError];

        Post[msgRb, cvData];
        CONTINUE;
        };

    IF source = NSFile.nullHandle THEN RETURN;
    dest ← VtoA[source, cvData, session, srcInstance, dstInstance, background];
};


<<
--=====================================================================================
This DependentOptionProc creates instance data with CreateCommon. The data is distinguished by the owner variable. The CommonObj within
CvXlit.CommonData is the data structure written to the client file stored as the icon properties. Only those fields pertaining to the
owner are used.
--=====================================================================================
>>


VPToAsciiDstOps: PUBLIC Converter.DependentOptionProc = {
<< = PROCEDURE [options: BOOLEAN ← TRUE, cvData: Converter.CvData, which: Converter.FormatToUse, srcFormat: XString.Reader, destFormat:
XString.Reader, window: Window.Handle, oldInstance: LONG POINTER ← NIL] RETURNS [menuItemProc: Converter.MenuItemProc, destroy:
Converter.DestroyProc, instance: LONG POINTER];
>>
    owner: CvXlit.Owners ← VtoAdst;

    menuItemProc ← CvXlit.CommonMenu;
    destroy ← CvXlit.DestroyCommon;
    IF oldInstance = NIL THEN
        instance ← CvXlit.CreateCommon[cvData, options, window, owner ! NSFile.Error, CvXlit.Problem => {owner ← backstop; instance ←
        NIL; CONTINUE}]
    ELSE
        {
        my: CvXlit.Common ← oldInstance;
        my.window ← window; --/* AR 13535: update window handle */
        instance ← my;
        };


    --/* make formwindow */
    CvXlit.CreateFW[instance, window, owner];
};


GetVATable: PUBLIC PROC RETURNS [vaTable: LONG POINTER TO VPToAsciiSetMap] = {
    RETURN[g.userMap];
};


--=====================
-- PROCEDURES
--=====================

VtoA: Converter.ConvertProc = {
    aborted: BOOLEAN ← FALSE;
    dataSkipped: BOOLEAN ← FALSE;
    attr: ARRAY [0..1) OF NSFile.Attribute ← [[type[StarFileTypes.text]]];
    enumProcs: DocInterchangeDefs.EnumProcsRecord ← [
      newParagraphProc: EndPrevAsciiPara,
      pageBreakProc: AddAsciiPage,
      textProc: AddAsciiText,
      pfcProc: AddAsciiPFC];
    openStatus: DocInterchangeDefs.OpenStatus;
    vaData: VADataObj; --/* only works if Enumerate doesn't FORK */
    dst: CvXlit.Common ← NIL;
```

```
--/* initialize instance data */
IF dstInstance = NIL THEN
    {
    ENABLE NSFile.Error, CvXlit.Problem => 
        {
        msgRb: XString.ReaderBody ← CvXlit.GetMessage[extraErr0];  --" Unrecoverable ASCII conversion error: damaged converter icon.
        "
        Converter.PostMessage[
          msg: @msgRb,
          cvData: cvData,
          cr: FALSE,
          clear: FALSE];
        GOTO terminate;
        };
    key: CvXlit.MessageKey ← CvXlit.MessageKey.FIRST;  --/* dummy */

    --/* we only care about dst */
    dst ← CvXlit.CreateCommon[cvData, FALSE, NIL, VtoAdst];

    dst.text[endLine] ← CvXlit.ParseItem[
      my: dst,
      r: @dst.textRb[endLine],
      item: key].ls;

    dst.text[endPara] ← CvXlit.ParseItem[
      my: dst,
      r: @dst.textRb[endPara],
      item: key].ls;

    dst.text[vtoaReplaceUnknown] ← CvXlit.ParseItem[
      my: dst,
      r: @dst.textRb[vtoaReplaceUnknown],
      item: key].ls;

    EXITS terminate => RETURN;
    }
ELSE
    {
    dst ← dstInstance;
    };

vaData ← [
    source: source,
    output: [NIL],
    cvData: cvData,
    session: session,
    dst: dst,
    background: background,
    doc: TRASH,
    putc: IF dst.f.lineLen = CvXlit.unlimited THEN UnbufferedPutC ELSE BufferedPutC,
    firstPara: TRUE,
    line: NIL,
    n: 0,
    pos: 0,
    max: 0,
    lastWhite: CARDINAL.LAST,
    wordWrap: dst.f.wordWrap.value,
    after: 0,
    streak: rightToLeft,
    z: dst.z];

IF dst.f.lineLen = CvXlit.limited THEN
    {
    --/* ASSERT: charsSuffix IN [10..256] */
    --/* create line buffer */
    vaData.line ← String.MakeString[z: vaData.z, maxlength: dst.f.charsSuffix];
    vaData.line.length ← vaData.line.maxlength;
    vaData.max ← dst.f.charsSuffix - 1;
    IF dst.text[endLine] # NIL AND dst.text[endLine].length < vaData.max THEN
        {
        --/* max column is limit less visible end-of-line characters */
        FOR i: CARDINAL IN [0..dst.text[endLine].length) DO
            SELECT dst.text[endLine][i] FROM
                Ascii.CR, Ascii.LF, Ascii.FF => NULL;
                ENDCASE => vaData.max ← vaData.max - 1;
        ENDLOOP;
        };
    };

BEGIN
  ENABLE
    {
    NSFile.Error => GOTO nsErr;
    DocInterchangeDefs.Error => GOTO docErr;
    UNWIND => IF dstInstance = NIL THEN
        {
        CvXlit.DestroyCommon[dst];
        dst ← NIL;
        };
    };

[vaData.doc, openStatus] ← DocInterchangeDefs.Open[
  docFileRef: NSFile.GetReference[source, vaData.session],
  session: vaData.session];
IF openStatus # ok THEN GOTO docErr;

dest ← NSFile.Create[
```

```
        directory: NSFile.nullHandle,
        attributes: DESCRIPTOR[attr],
        session: session ! NSFile.Error => {
            IF error = [space[mediumFull]] THEN
                Post[ConverterMsg.Get[ConverterMsg.koutOfSpace], vaData.cvData]
            ELSE
                Post[CvXlit.GetMessage[createError], vaData.cvData];
            GOTO nsErr}];

    vaData.output ← NSFileStream.Create[
        file: dest,
        closeOnDelete: FALSE,
        session: vaData.session];

    dataSkipped ← DocInterchangeDefs.Enumerate[
        textContainer: [doc[vaData.doc]],
        procs: @enumProcs,
        clientData: @vaData ! ABORTED => {
            dataSkipped ← TRUE;
            aborted ← TRUE;
            Post[ConverterMsg.Get[ConverterMsg.kuserAbort], vaData.cvData];
            CONTINUE}];

    --/* AR 13705: flush any remaining text */
    --/* ASSERT: n = 0 IF dst.f.lineLen # CvXlit.limited */
    IF NOT aborted AND vaData.n > 0 THEN
        {
        RawPuts[@vaData, vaData.line, vaData.n];
        --/* AR 14393: terminate last paragraph */
        RawPuts[@vaData, vaData.dst.text[endPara]];
        };

    Stream.Delete[vaData.output ! NSFile.Error => {
        IF error = [space[mediumFull]] THEN
            Post[ConverterMsg.Get[ConverterMsg.koutOfSpace], vaData.cvData]
        ELSE
            Post[ConverterMsg.Get[ConverterMsg.kunknownProblem], vaData.cvData];
        NSFile.Delete[dest, vaData.session];
        dest ← NSFile.nullHandle;
        GOTO nsErr}];
    IF dataSkipped THEN
            Post[ConverterMsg.Get[ConverterMsg.kdataSkipped], vaData.cvData];

    DocInterchangeDefs.Close[@vaData.doc];

    EXITS
        nsErr => {
            IF vaData.doc # NIL THEN
                DocInterchangeDefs.Close[@vaData.doc ! DocInterchangeDefs.Error => CONTINUE]};
        docErr => {
            key: XMessage.MsgKey ←
                SELECT openStatus FROM
                    malFormed, incompatible => ConverterMsg.kincompatible,
                    outOfDiskSpace, outOfVM => ConverterMsg.koutOfSpace,
                    ENDCASE => ConverterMsg.kcantOpen;

            Post[ConverterMsg.Get[key], vaData.cvData];
            IF vaData.doc # NIL THEN
                DocInterchangeDefs.Close[@vaData.doc ! DocInterchangeDefs.Error => CONTINUE];
            dest ← NSFile.nullHandle};
    END;
    IF vaData.line # NIL THEN vaData.z.FREE[@vaData.line];
    --/* destroy instance data if created by this proc */
    IF dstInstance = NIL AND dst # NIL THEN CvXlit.DestroyCommon[dst];
};


Post: PROC [msgRb: XString.ReaderBody, cvData: Converter.CvData] = {
    Converter.PostMessage[
        msg: @msgRb,
        cvData: cvData,
        cr: TRUE,
        clear: FALSE];
};


CheckAbort: PROC [background: BOOLEAN] RETURNS [yes: BOOLEAN] = INLINE {
    yes ← (background AND BackgroundProcess.UserAbort[]) OR
        (NOT background AND TIP.UserAbort[NIL]);
};


--/* Enumeration Procs */

AddAsciiPage: DocInterchangeDefs.PageBreakProc = {
<< = PROCEDURE [clientData: LONG POINTER, fontProps: DocInterchangePropsDefs.ReadonlyFontProps] RETURNS [stop: BOOL ← FALSE];
>>
    va: VAData ← clientData;
    -- form feed appended for a new page
    va.putc[va, Ascii.FF];
};


EndPrevAsciiPara:  DocInterchangeDefs.NewParagraphProc = {
<< = PROCEDURE [clientData: LONG POINTER, fontProps: DocInterchangePropsDefs.ReadonlyFontProps, paraProps:
DocInterchangePropsDefs.ReadonlyParaProps] RETURNS [stop: BOOL ← FALSE];
>>
```

```
    va: VAData ← clientData;

    --/* ASSERT: n = 0 IF dst.f.lineLen # CvXlit.limited */
    IF va.n > 0 THEN RawPuts[va, va.line, va.n]; --/* flush any pending text */

    --/* a new para char means we terminate the previous ASCII paragraph */
    IF va.firstPara AND paraProps.basicProps.preLeading < paraProps.basicProps.lineHeight THEN
        va.firstPara ← FALSE
    ELSE
        {
        --/* ceiling to next highest line */
        newlines: CARDINAL ←
          (paraProps.basicProps.preLeading + paraProps.basicProps.lineHeight - 1)
          paraProps.basicProps.lineHeight;

        IF NOT va.firstPara THEN
            {
            --/* end previous paragraph */
            RawPuts[va, va.dst.text[endPara]];

            --/* append endLine strings for AFTER paragraph spacing */
            THROUGH [1..va.after] DO
                RawPuts[va, va.dst.text[endLine]];
            ENDLOOP;
            };

        --/* this newPara character contains properties for the FOLLOWING *
        --/*   paragraph, therefore output BEFORE line spacing first */
        THROUGH [1..newlines] DO
            RawPuts[va, va.dst.text[endLine]];
        ENDLOOP;
        va.firstPara ← FALSE;
        };
    va.n ← 0; --/* reset line index */
    va.pos ← 0; --/* reset line position */
    va.lastWhite ← CARDINAL.LAST; --/* reset last white */
    --/* save AFTER line spacing */
    va.after ←
      (paraProps.basicProps.postLeading + paraProps.basicProps.lineHeight - 1) /
      paraProps.basicProps.lineHeight;
    va.streak ← paraProps.basicProps.streakSuccession;
};


AddAsciiPFC: DocInterchangeDefs.PFCProc = {};

<<
--=============================================================================================
AddAsciiText
This procedure does the bulk of the text handling.  Its main purpose is to translate VP characters into ASCII characters, according to
the user's encoding selection.
--=============================================================================================
>>

AddAsciiText: DocInterchangeDefs.TextProc = {
<< = PROCEDURE [clientData: LONG POINTER, fontProps: DocInterchangePropsDefs.ReadonlyFontProps, text: XString.Reader, textEndContext:
XString.Context] RETURNS [stop: BOOL ← FALSE];
>>
    va: VAData = clientData;

    --/* local procs */
    XnToX0: XString.MapCharProc = {
        xset: [0..256) ← XChar.Set[c];
        xcode: [0..256) ← XChar.Code[c];
        mapc: CHARACTER;
        putc: PutCProc = va.putc;

        IF g.userMap[xset] # NIL THEN {
            mapc ← g.userMap[xset][xcode];
            IF mapc # 0C THEN {
                IF (mapc = 12C OR mapc = 15C) THEN {
                    Puts[va, va.dst.text[endLine]];
                    IF va.n > 0 THEN FlushLine[va];
                    }
                ELSE {
                    IF va.streak = rightToLeft THEN {
                        SELECT mapc FROM
                            '( => mapc ← ');
                            ') => mapc ← '(;
                            '[ => mapc ← '];
                            '] => mapc ← '[;
                            '{ => mapc ← '};
                            '} => mapc ← '{;
                            ENDCASE;
                        putc[va, mapc];
                        }
                    ELSE
                        putc[va, mapc];
                    }
                }
            ELSE
                Puts[va, va.dst.text[vtoaReplaceUnknown]];
            }
        ELSE
            Puts[va, va.dst.text[vtoaReplaceUnknown]];
        stop ← FALSE;
    };
```

```
    --/* begin code */
    IF CheckAbort[va.background] THEN ERROR ABORTED;
    [] ← XString.Map[r: text, proc: XnToX0];
};


--/* put procs */

UnbufferedPutC: PutCProc = {
    Stream.PutChar[va.output, c];
};


BufferedPutC: PutCProc = {
    line: LONG STRING ← va.line;
    output: NSFileStream.Handle ← va.output;

    IF va.pos > va.max THEN
        {
        IF va.wordWrap THEN
            {
            offset: CARDINAL;

            --/* determine offset to new text */
            IF va.lastWhite = CARDINAL.LAST THEN
                {
                IF va.n > 0 THEN
                    {
                    va.lastWhite ← va.n - 1;
                    offset ← va.n;
                    }
                ELSE
                    offset ← va.lastWhite ← 0;
                }
            ELSE
                offset ← va.lastWhite + 1;

            --/* flush to mark */
            FOR i: CARDINAL IN [0..va.lastWhite] DO
                Stream.PutChar[output, line[i]];
            ENDLOOP;

            --/* end line */
            RawPuts[va, va.dst.text[endLine]];

            --/* restore line */
            FOR i: CARDINAL IN [offset..va.n) DO
                line[i-offset] ← line[i];
            ENDLOOP;
            va.n ← va.n - offset;
            va.lastWhite ← CARDINAL.LAST;

            --/* reset pos */
            va.pos ← 0;
            FOR i: CARDINAL IN [0..va.n) DO
                va.pos ← IF line[i] = Ascii.TAB THEN
                    ((va.pos / tabInterval) + 1) * tabInterval
                ELSE IF (c = Ascii.CR OR c = Ascii.LF) THEN
                    va.pos
                ELSE
                    va.pos + 1;
            ENDLOOP;
            }
        ELSE
            {
            RawPuts[va, line, va.n];
            --/* end line */
            RawPuts[va, va.dst.text[endLine]];
            va.n ← 0;
            va.pos ← 0;
            };
        };

    IF va.n >= line.length THEN
        {
        RawPuts[va, line];
        va.n ← va.pos ← 0;
        va.lastWhite ← CARDINAL.LAST;
        };

    --/* append character */
    line[va.n] ← c;

    IF c = Ascii.SP THEN va.lastWhite ← va.n;
    va.pos ← IF c = Ascii.TAB THEN
        ((va.pos / tabInterval) + 1) * tabInterval
    ELSE IF (c = Ascii.CR OR c = Ascii.LF) THEN
        va.pos
    ELSE
        va.pos + 1;

    va.n ← va.n + 1;
};


--/* put a string */
```

```
Puts: PROC [va: VAData, s: LONG STRING] = {
    IF s = NIL THEN RETURN;
    IF s.length = 0 THEN RETURN;

    FOR i: CARDINAL IN [0..s.length) DO
        va.putc[va, s[i]];
    ENDLOOP;
};


--/* raw put string */
RawPuts: PROC [va: VAData, s: LONG STRING, limit: CARDINAL ← CARDINAL.LAST] = {
    IF s = NIL THEN RETURN;
    IF s.length = 0 THEN RETURN;

    IF limit = CARDINAL.LAST THEN limit ← s.length;
    FOR i: CARDINAL IN [0..limit) DO
        Stream.PutChar[va.output, s[i]];
    ENDLOOP;
};


FlushLine: PROC [va: VAData] = {
    RawPuts[va, va.line, va.n];
    va.n ← va.pos ← 0;
    va.lastWhite ← CARDINAL.LAST;
};


ZzInit: PROC = {
    gz: UNCOUNTED ZONE = BWSZone.Permanent[];

    --/* these Spaces should not be unmapped while this application is loaded */
    g ← [
      defaultMap: Space.ScratchMap[(setMapSize + Environment.wordsPerPage-1) / Environment.wordsPerPage],
      userMap: Space.ScratchMap[(setMapSize + Environment.wordsPerPage-1) / Environment.wordsPerPage],
      gz: gz];

    --/* initialize conversion maps */

    FOR s: CARDINAL IN [0..256) DO
        g.defaultMap[s] ← NIL;
        g.userMap[s] ← NIL;
    ENDLOOP;

    g.defaultMap[0] ← Space.ScratchMap[(charMapSize + Environment.wordsPerPage-1) / Environment.wordsPerPage];
    g.defaultMap[41B] ← Space.ScratchMap[(charMapSize + Environment.wordsPerPage-1) / Environment.wordsPerPage];
    g.defaultMap[357B] ← Space.ScratchMap[(charMapSize + Environment.wordsPerPage-1) / Environment.wordsPerPage];

    g.userMap[0] ← Space.ScratchMap[(charMapSize + Environment.wordsPerPage-1) / Environment.wordsPerPage];
    g.userMap[41B] ← Space.ScratchMap[(charMapSize + Environment.wordsPerPage-1) / Environment.wordsPerPage];
    g.userMap[357B] ← Space.ScratchMap[(charMapSize + Environment.wordsPerPage-1) / Environment.wordsPerPage];

    FOR c: CARDINAL IN [0..256) DO
        g.defaultMap[0][c] ← VAL[c];
        g.userMap[0][c] ← VAL[c];
        g.defaultMap[41B][c] ← VAL[0];
        g.userMap[41B][c] ← VAL[0];
        g.defaultMap[357B][c] ← VAL[0];
        g.userMap[357B][c] ← VAL[0];
    ENDLOOP;

    g.defaultMap[0][211B] ← VAL[11B];          -- 211B -> tab
    g.defaultMap[0][244B] ← VAL[44B];          -- dollar -> $
    g.defaultMap[0][252B] ← VAL[42B];          -- leftDoubleQuote -> "
    g.defaultMap[0][272B] ← VAL[42B];          -- rightDoubleQuote -> "
    g.defaultMap[0][251B] ← VAL[47B];          -- leftSingleQuote -> '
    g.defaultMap[0][271B] ← VAL[47B];          -- rightSingleQuote -> '
    g.defaultMap[41B][76B] ← VAL[55B];         -- hyphen -> minus
    g.defaultMap[357B][42B] ← VAL[55B];        -- nonBreakingHyphen -> minus
    g.defaultMap[357B][41B] ← VAL[40B];        -- nonBreakingSpace -> space

    g.userMap[0][211B] ← VAL[11B];             -- 211B -> tab
    g.userMap[0][244B] ← VAL[44B];             -- dollar -> $
    g.userMap[0][252B] ← VAL[42B];             -- leftDoubleQuote -> "
    g.userMap[0][272B] ← VAL[42B];             -- rightDoubleQuote -> "
    g.userMap[0][251B] ← VAL[47B];             -- leftSingleQuote -> '
    g.userMap[0][271B] ← VAL[47B];             -- rightSingleQuote -> '
    g.userMap[41B][76B] ← VAL[55B];            -- hyphen -> minus
    g.userMap[357B][42B] ← VAL[55B];           -- nonBreakingHyphen -> minus
    g.userMap[357B][41B] ← VAL[40B];           -- nonBreakingSpace -> space
};


--/* main line code */

ZzInit[];


END...

LOG
16-Mar-87 14:06:16 - Caro - Created
26-Jun-87 11:30:20 - Caro - Added error catcher in ConvertProc over CreateCommon,
                            ISO8 now has correct ENDCASE
10-Jul-87 11:31:10 - Caro - Added before/after line spacing
19-Aug-87 11:03:02 - Caro - Fixed AR 13535 by updating oldInstance window
```

Fixed AR 13705 by flushing remaining text
16–Sep–87 12:21:09 – Caro – Fixed AR 14393 by terminating with endPara

```
-- File: CvXlitFWImpl.mesa
-- Trow 7-Sep-89 16:52:38

-- Last Revised by: Erickson                    17-Dec-87 16:03:15
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (c) 1987, 1989 by Xerox Corporation.  All rights reserved.

DIRECTORY
  Attention
    USING [Post],
  BWSZone
    USING [logonSession, Permanent],
  Catalog
    USING [GetFile],
  Converter
    USING [MenuItemProc, ResizeDetailWindow],
  CvXlit,
  Environment
    USING [Byte, wordsPerPage],
  FormWindow
    USING [AppendItem, AppendLine, ChoiceChangeProc, ChoiceHintsProc, ChoiceIndex,
    ChoiceItem, ChoiceItems, Create, GetBooleanItemValue, GetChoiceItemValue,
    GetIntegerItemValue, GetTextItemValue,
    HasBeenChanged, HasAnyBeenChanged, LayoutProc, Line, MakeItemsProc,
    MakeBooleanItem, MakeChoiceItem, MakeIntegerItem, MakeTextItem,
    MinDimsChangeProc,
    SetBooleanItemValue, SetChoiceItemValue, SetIntegerItemValue,
    SetTabStops, SetTextItemValue,
    SetVisibility, TabStops, SetSelection, SetInputFocus],
  FormWindowMessageParse
    USING [FreeChoiceItems, ParseChoiceItemMessage],
  NSAssignedTypes
    USING [tUnspecified],
  NSFile
    USING [AttributesProc, Close, Error, Filter, Find, GetReference, Handle,
    List, nullHandle, nullReference, OpenByReference, Reference, Scope, Selections],
  NSFileStream
    USING [Create, Handle],
  NSString
    USING [FreeString, String, StringFromMesaString],
  Space
    USING [ScratchMap, Unmap],
  Stream
    USING [Delete, EndOfStream, GetByte, GetWord],
  Window
    USING [Handle],
  XChar
    USING [Character, Code, Set],
  XCharSet0
    USING [Make],
  XString
    USING [CopyReader, Empty, FreeReaderBytes, FreeWriterBytes, FromNSString, FromSTRING,
    NewWriterBody, NSStringFromReader, nullReaderBody, Reader,
    ReaderBody, WriterBody, InvalidNumber, Overflow];

<<
--=================================================================================
-- OVERVIEW:

Formwindow procedures
--=================================================================================
>>

CvXlitFWImpl: PROGRAM
IMPORTS
  Attention, BWSZone, Catalog, Converter, CvXlit, FormWindow, FormWindowMessageParse,
  NSFile, NSFileStream, NSString, Space, Stream, XChar, XCharSet0, XString
EXPORTS
  CvXlit =
BEGIN

--====================
-- CONSTANTS
--====================

textWidth: CARDINAL = 320;
tabStopInterval: CARDINAL = CvXlit.pointsBetweenItems/2;

charMapSize: CARDINAL = SIZE[VPToAsciiCharMap];

--====================
-- TYPES
--====================

Global: TYPE = RECORD [
    avCount: CARDINAL,
    vaCount: CARDINAL,
    avChoices: ARRAY [0..31) OF FormWindow.ChoiceItem,
    vaChoices: ARRAY [0..31) OF FormWindow.ChoiceItem,
    cz: UNCOUNTED ZONE];

HintArray: TYPE = ARRAY [0..30) OF FormWindow.ChoiceIndex;

VPToAsciiCharMap: TYPE = ARRAY [0..256) OF CHARACTER;

--====================
```

```
--- GLOBALS
---====================

g: Global ← [
    avCount: 0,
    vaCount: 0,
    avChoices: ALL[[string[0, XString.nullReaderBody]]],
    vaChoices: ALL[[string[0, XString.nullReaderBody]]],
    cz: BWSZone.Permanent[]];

hintsObject: HintArray ←
    [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30];

avHints: LONG DESCRIPTOR FOR ARRAY OF FormWindow.ChoiceIndex;
vaHints: LONG DESCRIPTOR FOR ARRAY OF FormWindow.ChoiceIndex;

---====================
--- PUBLIC PROCEDURES
---====================


CommonMenu: PUBLIC Converter.MenuItemProc = {
<< = PROCEDURE [instance: LONG POINTER, menuItem: PropertySheet.MenuItemType] RETURNS [ok: BOOLEAN ← TRUE]; >>
    my: CvXlit.Common = instance;
    avPara: XString.ReaderBody ← CvXlit.GetMessage[dfltAVEndParagraph];
    avChar: XString.ReaderBody ← CvXlit.GetMessage[dfltAVReplaceCharacter];
    vaLine: XString.ReaderBody ← CvXlit.GetMessage[dfltVAEndLine];
    vaPara: XString.ReaderBody ← CvXlit.GetMessage[dfltVAEndParagraph];
    vaChar: XString.ReaderBody ← CvXlit.GetMessage[dfltVAReplaceCharacter];

    IF my = NIL THEN RETURN[ok: TRUE];

    SELECT menuItem FROM
        defaults => {
            SELECT my.owner FROM
                AtoVsrc => {
                    FormWindow.SetTextItemValue[
                        window: my.window,
                        item: CvXlit.MessageKey.paraEndsWith.ORD,
                        newValue: @avPara,
                        repaint: FALSE];
                    };
                AtoVdst => {
                    FormWindow.SetChoiceItemValue[
                        window: my.window,
                        item: 100,
                        newValue: 0,
                        repaint: FALSE];
                    FormWindow.SetChoiceItemValue[
                        window: my.window,
                        item: CvXlit.MessageKey.font.ORD,
                        newValue: CvXlit.dfltFont,
                        repaint: FALSE];
                    FormWindow.SetChoiceItemValue[
                        window: my.window,
                        item: CvXlit.MessageKey.fontSize.ORD,
                        newValue: CvXlit.dfltFontSize,
                        repaint: FALSE];
                    FormWindow.SetTextItemValue[
                        window: my.window,
                        item: CvXlit.MessageKey.replaceUnknown.ORD,
                        newValue: @avChar,
                        repaint: FALSE];
                    FormWindow.SetBooleanItemValue[
                        window: my.window,
                        item: CvXlit.MessageKey.ignoreTrailing.ORD,
                        newValue: CvXlit.dfltTrailing,
                        repaint: TRUE];
                    };
                VtoAdst => {
                    FormWindow.SetChoiceItemValue[
                        window: my.window,
                        item: 101,
                        newValue: 0,
                        repaint: FALSE];
                    FormWindow.SetChoiceItemValue[
                        window: my.window,
                        item: CvXlit.MessageKey.lineLen.ORD,
                        newValue: CvXlit.dfltLineLen,
                        repaint: FALSE];
                    FormWindow.SetIntegerItemValue[
                        window: my.window,
                        item: CvXlit.MessageKey.charsSuffix.ORD,
                        newValue: CvXlit.dfltChars,
                        repaint: FALSE];
                    FormWindow.SetBooleanItemValue[
                        window: my.window,
                        item: CvXlit.MessageKey.wordWrap.ORD,
                        newValue: CvXlit.dfltWordWrap,
                        repaint: FALSE];
                    FormWindow.SetTextItemValue[
                        window: my.window,
                        item: CvXlit.MessageKey.endLine.ORD,
```

```
                        newValue: @vaLine,
                        repaint: FALSE];
                    FormWindow.SetTextItemValue[
                        window: my.window,
                        item: CvXlit.MessageKey.endPara.ORD,
                        newValue: @vaPara,
                        repaint: FALSE];
                    FormWindow.SetTextItemValue[
                        window: my.window,
                        item: CvXlit.MessageKey.replaceUnknown.ORD,
                        newValue: @vaChar,
                        repaint: TRUE];
                    };
                ENDCASE;
            };
        done => {
            ENABLE NSFile.Error, CvXlit.Problem => {
                msgRb: XString.ReaderBody ← CvXlit.GetMessage[doneFailed];
                Attention.Post[@msgRb];
                GOTO notOK;
                };
            IF FormWindow.HasAnyBeenChanged[my.window] THEN {
                ok ← ApplyChanges[my];
                IF NOT ok THEN GOTO notOK;
                CvXlit.StoreFiledData[my];
                };
            EXITS notOK => RETURN[ok: FALSE];
            };
        start => {
            ok ← ApplyChanges[my];
            };
        ENDCASE;
};

CreateFW: PUBLIC PROC [my: CvXlit.Common, window: Window.Handle, owner: CvXlit.Owners] = {
    SELECT owner FROM
        AtoVsrc => {
            FormWindow.Create[
                window: window,
                makeItemsProc: MakeAtoVSrc,
                layoutProc: LayoutAtoVSrc,
                minDimsChangeProc: GrowParent,
                clientData: my];
            CvXlit.DataToWindow[my, window];
            };
        AtoVdst => {
            FormWindow.Create[
                window: window,
                makeItemsProc: MakeAtoVDst,
                layoutProc: LayoutAtoVDst,
                clientData: my];
            };
        VtoAdst => {
            FormWindow.Create[
                window: window,
                makeItemsProc: MakeVtoADst,
                layoutProc: LayoutVtoADst,
                minDimsChangeProc: GrowParent,
                clientData: my];
            CvXlit.DataToWindow[my, window];
            };
        backstop => {
            FormWindow.Create[
                window: window,
                makeItemsProc: MakeBackstop];
            };
        ENDCASE;
};

--=====================
-- PROCEDURES
--=====================

ApplyChanges: PROC [my: CvXlit.Common] RETURNS [ok: BOOLEAN ← TRUE] = {
    bufWb: XString.WriterBody ← XString.NewWriterBody[maxLength: 30, z: my.z];
    tf: NSFile.Handle;
    ts: NSFileStream.Handle;
    zero: [0..256);
    char: CHARACTER;
    xchar: XChar.Character;
    avt: CvXlit.AsciiToVPTable;
    vat: CvXlit.VPToAsciiTable;
    tableName: XString.ReaderBody;
    table0: XString.ReaderBody;
    tempChoice: FormWindow.ChoiceItem;
    tableChoice: FormWindow.ChoiceIndex;

    SELECT my.owner FROM
```

```
AtoVsrc => 
    {
    IF FormWindow.HasBeenChanged[window: my.window, item: CvXlit.MessageKey.paraEndsWith.ORD] THEN
        {
        IF my.textRb[paraEndsWith] # XString.nullReaderBody THEN
            XString.FreeReaderBytes[@my.textRb[paraEndsWith], my.z];
        my.textRb[paraEndsWith] ← FormWindow.GetTextItemValue[
          window: my.window,
          item: CvXlit.MessageKey.paraEndsWith.ORD,
          zone: my.z];
        };
    [ok: ok, ls: my.text[paraEndsWith]] ← CvXlit.ParseItem[
      my: my,
      r: @my.textRb[paraEndsWith],
      item: CvXlit.MessageKey.paraEndsWith,
      buf: @bufWb];
    IF NOT ok THEN RETURN;
    };
AtoVdst => 
    {
    IF FormWindow.HasBeenChanged[my.window, 100] THEN
        {
        --/* set conversion map */

        tableChoice ← FormWindow.GetChoiceItemValue[window: my.window, item: 100];

        avt ← CvXlit.GetAVTable[];
        FOR c: CHARACTER IN CHARACTER DO
            avt[c] ← XCharSet0.Make[LOOPHOLE[c]];
        ENDLOOP;

        tempChoice ← g.avChoices[tableChoice];
        WITH tempChoice SELECT FROM
            string => {tableName ← string};
            ENDCASE;
        IF ~XString.Empty[@tableName] THEN
            {
            tf ← GetTableFile[tableName];
            ts ← NSFileStream.Create[file: tf];
            DO
                ENABLE {Stream.EndOfStream => EXIT};
                [] ← Stream.GetByte[ts];
                char ← LOOPHOLE[Stream.GetByte[ts]];
                xchar ← LOOPHOLE[Stream.GetWord[ts]];
                avt[char] ← xchar;
            ENDLOOP;
            Stream.Delete[ts];
            ts ← [NIL];
            tempChoice ← g.avChoices[0];
            WITH tempChoice SELECT FROM
                string => {table0 ← string};
                ENDCASE;
            IF table0 # XString.nullReaderBody THEN
                XString.FreeReaderBytes[@table0, g.cz];
            IF my.textRb[avTableName] # XString.nullReaderBody THEN
                XString.FreeReaderBytes[@my.textRb[avTableName], my.z];
            g.avChoices[0] ← [string[choiceNumber: 0, string: XString.CopyReader[@tableName, g.cz] ↑ ]];
            my.textRb[avTableName] ← XString.CopyReader[@tableName, g.cz] ↑ ;
            };
        FOR i: CARDINAL IN [1..g.avCount] DO
            tempString: XString.ReaderBody;
            WITH g.avChoices[i] SELECT FROM
                string => {tempString ← string};
                ENDCASE;
            XString.FreeReaderBytes[@tempString, g.cz];
            g.avChoices[i] ← [string[i, XString.nullReaderBody]];
        ENDLOOP;
        };
    IF FormWindow.HasBeenChanged[my.window, CvXlit.MessageKey.font.ORD] THEN
        {
        my.f.font ← FormWindow.GetChoiceItemValue[
          window: my.window,
          item: CvXlit.MessageKey.font.ORD];
        };
    IF FormWindow.HasBeenChanged[my.window, CvXlit.MessageKey.fontSize.ORD] THEN
        {
        my.f.fontSize ← FormWindow.GetChoiceItemValue[
          window: my.window,
          item: CvXlit.MessageKey.fontSize.ORD];
        };
    IF FormWindow.HasBeenChanged[my.window, CvXlit.MessageKey.replaceUnknown.ORD] THEN
        {
        IF my.textRb[atovReplaceUnknown] # XString.nullReaderBody THEN
            XString.FreeReaderBytes[@my.textRb[atovReplaceUnknown], my.z];
        my.textRb[atovReplaceUnknown] ← FormWindow.GetTextItemValue[
          window: my.window,
          item: CvXlit.MessageKey.replaceUnknown.ORD,
          zone: my.z];
        };
    [ok: ok, ls: my.text[atovReplaceUnknown]] ← CvXlit.ParseItem[
      my: my,
      r: @my.textRb[atovReplaceUnknown],
      item: CvXlit.MessageKey.replaceUnknown,
      buf: @bufWb];
    IF NOT ok THEN RETURN;

    IF FormWindow.HasBeenChanged[my.window, CvXlit.MessageKey.ignoreTrailing.ORD] THEN
```

```
            {
            my.f.ignoreTrailing.value ← FormWindow.GetBooleanItemValue[
               window: my.window,
               item: CvXlit.MessageKey.ignoreTrailing.ORD];
            };
      };
VtoAdst => 
      {
      IF FormWindow.HasBeenChanged[my.window, 101] THEN
            {
            charMap: LONG POINTER TO VPToAsciiCharMap;
            xset: [0..256);

            --/* set conversion map */

            tableChoice ← FormWindow.GetChoiceItemValue[window: my.window, item: 101];

            vat ← CvXlit.GetVATable[];
            NewUserMap[vat];

            tempChoice ← g.vaChoices[tableChoice];
            WITH tempChoice SELECT FROM
                  string => {tableName ← string};
                  ENDCASE;
            IF ~XString.Empty[@tableName] THEN
                  {
                  tf ← GetTableFile[tableName];
                  ts ← NSFileStream.Create[file: tf];
                  DO
                        ENABLE {Stream.EndOfStream => EXIT};
                        xchar ← LOOPHOLE[Stream.GetWord[ts]];
                        zero ← Stream.GetByte[ts];
                        char ← LOOPHOLE[Stream.GetByte[ts]];
                        IF zero # 0 THEN LOOP;
                        xset ← XChar.Set[xchar];
                        charMap ← vat[xset];
                        IF charMap = NIL THEN {
                              vat[xset] ← Space.ScratchMap[(charMapSize + Environment.wordsPerPage-1) / Environment.wordsPerPage];
                              FOR c: CARDINAL IN [0..256) DO
                                    vat[xset][c] ← VAL[0];
                              ENDLOOP;
                              charMap ← vat[xset];
                        };
                        charMap[XChar.Code[xchar]] ← char;
                  ENDLOOP;
                  Stream.Delete[ts];
                  ts ← [NIL];

                  tempChoice ← g.vaChoices[0];
                  WITH tempChoice SELECT FROM
                        string => {table0 ← string};
                        ENDCASE;
                  IF table0 # XString.nullReaderBody THEN
                        XString.FreeReaderBytes[@table0, g.cz];
                  IF my.textRb[vaTableName] # XString.nullReaderBody THEN
                        XString.FreeReaderBytes[@my.textRb[vaTableName], my.z];
                  g.vaChoices[0] ← [string[choiceNumber: 0, string: XString.CopyReader[@tableName, g.cz] ↑ ]];
                  my.textRb[vaTableName] ← XString.CopyReader[@tableName, g.cz] ↑ ;
                  };
            FOR i: CARDINAL IN [1..g.vaCount] DO
                  tempString: XString.ReaderBody;
                  WITH g.vaChoices[i] SELECT FROM
                        string => {tempString ← string};
                        ENDCASE;
                  XString.FreeReaderBytes[@tempString, g.cz];
                  g.vaChoices[i] ← [string[i, XString.nullReaderBody]];
            ENDLOOP;
            };



      IF FormWindow.HasBeenChanged[my.window, CvXlit.MessageKey.lineLen.ORD] THEN
            {
            my.f.lineLen ← FormWindow.GetChoiceItemValue[
               window: my.window,
               item: CvXlit.MessageKey.lineLen.ORD];
            };
      IF FormWindow.HasBeenChanged[my.window, CvXlit.MessageKey.charsSuffix.ORD] THEN
            {
            my.f.charsSuffix ← CARDINAL[FormWindow.GetIntegerItemValue[window: my.window,
               item: CvXlit.MessageKey.charsSuffix.ORD !
                  XString.InvalidNumber => {
                     msgRb: XString.ReaderBody ← CvXlit.GetMessage[extraErr1];
                     Attention.Post[@msgRb];
                     GOTO Badnum;
                     };
                  XString.Overflow => {
                     my.f.charsSuffix ← 0;
                     CONTINUE;
                     }]];
            IF my.f.charsSuffix NOT IN [10..256] THEN
                  {
                  msgRb: XString.ReaderBody ← CvXlit.GetMessage[charsOutOfBounds];
                  Attention.Post[@msgRb];
                  GOTO Badnum;
```

```
                    };
                EXITS
                  Badnum => {
                    FormWindow.SetSelection[window: my.window,
                                            item: CvXlit.MessageKey.charsSuffix.ORD,
                                            firstChar: 0, lastChar: CARDINAL.LAST];
                    FormWindow.SetInputFocus[window: my.window,
                                             item: CvXlit.MessageKey.charsSuffix.ORD,
                                             beforeChar: CARDINAL.LAST];
                    RETURN[ok: FALSE];
                  };
                };
              IF FormWindow.HasBeenChanged[my.window, CvXlit.MessageKey.wordWrap.ORD] THEN
                  {
                  my.f.wordWrap.value ← FormWindow.GetBooleanItemValue[
                    window: my.window,
                    item: CvXlit.MessageKey.wordWrap.ORD];
                  };
              IF FormWindow.HasBeenChanged[my.window, CvXlit.MessageKey.endLine.ORD] THEN
                  {
                  IF my.textRb[endLine] # XString.nullReaderBody THEN
                      XString.FreeReaderBytes[@my.textRb[endLine], my.z];
                  my.textRb[endLine] ← FormWindow.GetTextItemValue[
                    window: my.window,
                    item: CvXlit.MessageKey.endLine.ORD,
                    zone: my.z];
                  };
              [ok: ok, ls: my.text[endLine]] ← CvXlit.ParseItem[
                my: my,
                r: @my.textRb[endLine],
                item: CvXlit.MessageKey.endLine,
                buf: @bufWb];
              IF NOT ok THEN RETURN;

              IF FormWindow.HasBeenChanged[my.window, CvXlit.MessageKey.endPara.ORD] THEN
                  {
                  IF my.textRb[endPara] # XString.nullReaderBody THEN
                      XString.FreeReaderBytes[@my.textRb[endPara], my.z];
                  my.textRb[endPara] ← FormWindow.GetTextItemValue[
                    window: my.window,
                    item: CvXlit.MessageKey.endPara.ORD,
                    zone: my.z];
                  };
              [ok: ok, ls: my.text[endPara]] ← CvXlit.ParseItem[
                my: my,
                r: @my.textRb[endPara],
                item: CvXlit.MessageKey.endPara,
                buf: @bufWb];
              IF NOT ok THEN RETURN;

              IF FormWindow.HasBeenChanged[my.window, CvXlit.MessageKey.replaceUnknown.ORD] THEN
                  {
                  IF my.textRb[vtoaReplaceUnknown] # XString.nullReaderBody THEN
                      XString.FreeReaderBytes[@my.textRb[vtoaReplaceUnknown], my.z];
                  my.textRb[vtoaReplaceUnknown] ← FormWindow.GetTextItemValue[
                    window: my.window,
                    item: CvXlit.MessageKey.replaceUnknown.ORD,
                    zone: my.z];
                  };
              [ok: ok, ls: my.text[vtoaReplaceUnknown]] ← CvXlit.ParseItem[
                my: my,
                r: @my.textRb[vtoaReplaceUnknown],
                item: CvXlit.MessageKey.replaceUnknown,
                buf: @bufWb];
              IF NOT ok THEN RETURN;

              };
            ENDCASE;

        XString.FreeWriterBytes[@bufWb];

    }; -- ApplyChanges


    GrowParent: FormWindow.MinDimsChangeProc = {
    << = PROCEDURE [window: Window.Handle, old: Window.Dims, new: Window.Dims];
    >>
        my: CvXlit.Common = CvXlit.DataFromWindow[window];
        oldHeight: INTEGER;

        --/* don't adjust the first time window is viewed */
        IF my = NIL THEN RETURN;
        IF old = new THEN RETURN;

        --/* defaulting newHeight returns oldHeight without resizing */
        oldHeight ← Converter.ResizeDetailWindow[
          cvData: my.cvData,
          window: window,
          which: IF my.owner = AtoVsrc THEN source ELSE destination];

        --/* now resize window */
        [] ← Converter.ResizeDetailWindow[
          cvData: my.cvData,
          window: window,
          which: IF my.owner = AtoVsrc THEN source ELSE destination,
          newHeight: oldHeight + (new.h - old.h)];
    };
```

```
MakeBackstop: FormWindow.MakeItemsProc = {
    tag: XString.ReaderBody ← CvXlit.GetMessage[backstop];

    FormWindow.MakeTextItem[
      window: window,
      myKey: CvXlit.MessageKey.backstop.ORD,
      boxed: FALSE,
      readOnly: TRUE,
      width: 400,
      initString: @tag];
};


AVHints: FormWindow.ChoiceHintsProc = {
  RETURN[hints: avHints, freeHints: NIL];
};

VAHints: FormWindow.ChoiceHintsProc = {
  RETURN[hints: vaHints, freeHints: NIL];
};

MakeAtoVDst: FormWindow.MakeItemsProc = {
<< = PROCEDURE [window: Window.Handle, clientData: LONG POINTER];
>>
    my: CvXlit.Common = clientData;
    tag: XString.ReaderBody;
    tmp: XString.ReaderBody;

    tag ← CvXlit.GetMessage[transliterationTable];
    BEGIN
    folder: NSFile.Handle ← GetTableFolder[];
    ListTables[folder];
    NSFile.Close[folder];
    avHints ← DESCRIPTOR[BASE[hintsObject], g.avCount];
    FormWindow.MakeChoiceItem[
      window: window,
      myKey: 100,
      tag: @tag,
      values: DESCRIPTOR[g.avChoices],
      initChoice: 0,
      fullyDisplayed: FALSE,
      hintsProc: AVHints];
    END;

    tag ← CvXlit.GetMessage[font];
    tmp ← CvXlit.GetMessage[fontChoices];
    BEGIN
    values: FormWindow.ChoiceItems ← FormWindowMessageParse.ParseChoiceItemMessage[choiceItemMessage: @tmp, zone: my.z];
    FormWindow.MakeChoiceItem[
      window: window,
      myKey: CvXlit.MessageKey.font.ORD,
      tag: @tag,
      values: values,
      initChoice: my.f.font,
      fullyDisplayed: TRUE];
    FormWindowMessageParse.FreeChoiceItems[choiceItems: values, zone: my.z];
    END;

    tag ← CvXlit.GetMessage[fontSize];
    tmp ← CvXlit.GetMessage[fontSizeChoices];
    BEGIN
    values: FormWindow.ChoiceItems ← FormWindowMessageParse.ParseChoiceItemMessage[choiceItemMessage: @tmp, zone: my.z];
    FormWindow.MakeChoiceItem[
      window: window,
      myKey: CvXlit.MessageKey.fontSize.ORD,
      tag: @tag,
      values: values,
      initChoice: my.f.fontSize,
      fullyDisplayed: TRUE];
    FormWindowMessageParse.FreeChoiceItems[choiceItems: values, zone: my.z];
    END;

    tag ← CvXlit.GetMessage[replaceUnknown];
    FormWindow.MakeTextItem[
      window: window,
      myKey: CvXlit.MessageKey.replaceUnknown.ORD,
      tag: @tag,
      width: textWidth,
      initString: @my.textRb[atovReplaceUnknown]];

    tag ← CvXlit.GetMessage[ignoreTrailing];
    FormWindow.MakeBooleanItem[
      window: window,
      myKey: CvXlit.MessageKey.ignoreTrailing.ORD,
      label: [string[tag]],
      initBoolean: my.f.ignoreTrailing.value];
};


MakeAtoVSrc: FormWindow.MakeItemsProc = {
<< = PROCEDURE [window: Window.Handle, clientData: LONG POINTER];
>>
    my: CvXlit.Common = clientData;
```

```
    tag: XString.ReaderBody;

    tag ← CvXlit.GetMessage[paraEndsWith];
    FormWindow.MakeTextItem[
      window: window,
      myKey: CvXlit.MessageKey.paraEndsWith.ORD,
      tag: @tag,
      width: textWidth,
      initString: @my.textRb[paraEndsWith]];
};


MakeVtoADst: FormWindow.MakeItemsProc = {
<< = PROCEDURE [window: Window.Handle, clientData: LONG POINTER]; >>
    my: CvXlit.Common = clientData;
    tag: XString.ReaderBody;
    tmp: XString.ReaderBody;

    tag ← CvXlit.GetMessage[transliterationTable];
    BEGIN
    folder: NSFile.Handle ← GetTableFolder[];
    ListVATables[folder];
    NSFile.Close[folder];
    vaHints ← DESCRIPTOR[BASE[hintsObject], g.vaCount];
    FormWindow.MakeChoiceItem[
      window: window,
      myKey: 101,
      tag: @tag,
      values: DESCRIPTOR[g.vaChoices],
      initChoice: 0,
      fullyDisplayed: FALSE,
      hintsProc: VAHints];
    END;

    tag ← CvXlit.GetMessage[lineLen];
    tmp ← CvXlit.GetMessage[lineLenChoices];
    BEGIN
    values: FormWindow.ChoiceItems ← FormWindowMessageParse.ParseChoiceItemMessage[choiceItemMessage: @tmp, zone: my.z];
    FormWindow.MakeChoiceItem[
      window: window,
      myKey: CvXlit.MessageKey.lineLen.ORD,
      tag: @tag,
      values: values,
      initChoice: my.f.lineLen,
      changeProc: LineLenXProc,
      fullyDisplayed: TRUE];
    FormWindowMessageParse.FreeChoiceItems[choiceItems: values, zone: my.z];
    END;

    tag ← CvXlit.GetMessage[charsSuffix];
    FormWindow.MakeIntegerItem[
      window: window,
      myKey: CvXlit.MessageKey.charsSuffix.ORD,
      suffix: @tag,
      visibility: IF my.f.lineLen = CvXlit.limited THEN visible ELSE invisible,
      signed: FALSE,
      width: 30,
      initInteger: INTEGER[my.f.charsSuffix]];

    tag ← CvXlit.GetMessage[wordWrap];
    FormWindow.MakeBooleanItem[
      window: window,
      myKey: CvXlit.MessageKey.wordWrap.ORD,
      visibility: IF my.f.lineLen = CvXlit.limited THEN visible ELSE invisible,
      label: [string[tag]],
      initBoolean: my.f.wordWrap.value];

    tag ← CvXlit.GetMessage[endLine];
    FormWindow.MakeTextItem[
      window: window,
      myKey: CvXlit.MessageKey.endLine.ORD,
      tag: @tag,
      width: textWidth,
      initString: @my.textRb[endLine]];

    tag ← CvXlit.GetMessage[endPara];
    FormWindow.MakeTextItem[
      window: window,
      myKey: CvXlit.MessageKey.endPara.ORD,
      tag: @tag,
      width: textWidth,
      initString: @my.textRb[endPara]];

    tag ← CvXlit.GetMessage[replaceUnknown];
    FormWindow.MakeTextItem[
      window: window,
      myKey: CvXlit.MessageKey.replaceUnknown.ORD,
      tag: @tag,
      width: textWidth,
      initString: @my.textRb[vtoaReplaceUnknown]];
};


LayoutAtoVDst: FormWindow.LayoutProc = {
<< = PROCEDURE [window: Window.Handle, clientData: LONG POINTER];
>>
    leadingMargin: CARDINAL = CvXlit.leadingMargin;
```

```
        spaceAboveLine: CARDINAL = 5;
        line: FormWindow.Line;
        tabChoice: fixed FormWindow.TabStops = [fixed[tabStopInterval]];

        FormWindow.SetTabStops[window, tabChoice];

        line ← FormWindow.AppendLine[window, spaceAboveLine];
        FormWindow.AppendItem[
          window: window,
          item: 100,
          line: line,
          preMargin: CvXlit.GetPreMargin[transliterationTable] MOD tabStopInterval,
          tabStop: CvXlit.GetPreMargin[transliterationTable] / tabStopInterval,
          repaint: FALSE];

        line ← FormWindow.AppendLine[window, spaceAboveLine];
        FormWindow.AppendItem[
          window: window,
          item: CvXlit.MessageKey.font.ORD,
          line: line,
          preMargin: CvXlit.GetPreMargin[font] MOD tabStopInterval,
          tabStop: CvXlit.GetPreMargin[font] / tabStopInterval,
          repaint: FALSE];
        line ← FormWindow.AppendLine[window, spaceAboveLine];
        FormWindow.AppendItem[
          window: window,
          item: CvXlit.MessageKey.fontSize.ORD,
          line: line,
          preMargin: CvXlit.GetPreMargin[fontSize] MOD tabStopInterval,
          tabStop: CvXlit.GetPreMargin[fontSize] / tabStopInterval,
          repaint: FALSE];
        line ← FormWindow.AppendLine[window, spaceAboveLine];
        FormWindow.AppendItem[
          window: window,
          item: CvXlit.MessageKey.replaceUnknown.ORD,
          line: line,
          preMargin: CvXlit.GetPreMargin[replaceUnknown] MOD tabStopInterval,
          tabStop: CvXlit.GetPreMargin[replaceUnknown] / tabStopInterval,
          repaint: FALSE];
        line ← FormWindow.AppendLine[window, spaceAboveLine];
        FormWindow.AppendItem[
          window: window,
          item: CvXlit.MessageKey.ignoreTrailing.ORD,
          line: line,
          preMargin: CvXlit.GetPreMargin[ignoreTrailing] MOD tabStopInterval,
          tabStop: CvXlit.GetPreMargin[ignoreTrailing] / tabStopInterval,
          repaint: FALSE];

};


LayoutAtoVSrc: FormWindow.LayoutProc = {
<< = PROCEDURE [window: Window.Handle, clientData: LONG POINTER];
>>
        leadingMargin: CARDINAL = CvXlit.leadingMargin;
        spaceAboveLine: CARDINAL = 5;
        line: FormWindow.Line;
        tabChoice: fixed FormWindow.TabStops = [fixed[tabStopInterval]];

        FormWindow.SetTabStops[window, tabChoice];

        line ← FormWindow.AppendLine[window, spaceAboveLine];
        FormWindow.AppendItem[
          window: window,
          item: CvXlit.MessageKey.paraEndsWith.ORD,
          line: line,
          preMargin: CvXlit.GetPreMargin[paraEndsWith] MOD tabStopInterval,
          tabStop: CvXlit.GetPreMargin[paraEndsWith] / tabStopInterval,
          repaint: FALSE];
};


LayoutVtoADst: FormWindow.LayoutProc = {
<< = PROCEDURE [window: Window.Handle, clientData: LONG POINTER]; >>
        leadingMargin: CARDINAL = CvXlit.leadingMargin;
        spaceAboveLine: CARDINAL = 5;
        line: FormWindow.Line;
        tabChoice: fixed FormWindow.TabStops = [fixed[tabStopInterval]];

        FormWindow.SetTabStops[window, tabChoice];

        line ← FormWindow.AppendLine[window, spaceAboveLine];
        FormWindow.AppendItem[
          window: window,
          item: 101,
          line: line,
          preMargin: CvXlit.GetPreMargin[transliterationTable] MOD tabStopInterval,
          tabStop: CvXlit.GetPreMargin[transliterationTable] / tabStopInterval,
          repaint: FALSE];

        line ← FormWindow.AppendLine[window, spaceAboveLine];
        FormWindow.AppendItem[
          window: window,
          item: CvXlit.MessageKey.lineLen.ORD,
          line: line,
          preMargin: CvXlit.GetPreMargin[lineLen] MOD tabStopInterval,
          tabStop: CvXlit.GetPreMargin[lineLen] / tabStopInterval,
```

```
            repaint: FALSE];
        FormWindow.AppendItem[
            window: window,
            item: CvXlit.MessageKey.charsSuffix.ORD,
            line: line,
            preMargin: CvXlit.GetPreMargin[charsSuffix],
            tabStop: ,
            repaint: FALSE];
        FormWindow.AppendItem[
            window: window,
            item: CvXlit.MessageKey.wordWrap.ORD,
            line: line,
            preMargin: CvXlit.GetPreMargin[wordWrap],
            tabStop: ,
            repaint: FALSE];
        line ← FormWindow.AppendLine[window, spaceAboveLine];
        FormWindow.AppendItem[
            window: window,
            item: CvXlit.MessageKey.endLine.ORD,
            line: line,
            preMargin: CvXlit.GetPreMargin[endLine] MOD tabStopInterval,
            tabStop: CvXlit.GetPreMargin[endLine] / tabStopInterval,
            repaint: FALSE];
        line ← FormWindow.AppendLine[window, spaceAboveLine];
        FormWindow.AppendItem[
            window: window,
            item: CvXlit.MessageKey.endPara.ORD,
            line: line,
            preMargin: CvXlit.GetPreMargin[endPara] MOD tabStopInterval,
            tabStop: CvXlit.GetPreMargin[endPara] / tabStopInterval,
            repaint: FALSE];
        line ← FormWindow.AppendLine[window, spaceAboveLine];
        FormWindow.AppendItem[
            window: window,
            item: CvXlit.MessageKey.replaceUnknown.ORD,
            line: line,
            preMargin: CvXlit.GetPreMargin[replaceUnknown] MOD tabStopInterval,
            tabStop: CvXlit.GetPreMargin[replaceUnknown] / tabStopInterval,
            repaint: FALSE];
};

--/* Change Procs */

LineLenXProc: FormWindow.ChoiceChangeProc = {
<< = PROCEDURE [window: Window.Handle, item: FormWindow.ItemKey, calledBecauseOf: FormWindow.ChangeReason, oldValue:
FormWindow.ChoiceIndex, newValue: FormWindow.ChoiceIndex];
>>
    IF newValue = oldValue THEN RETURN;
    IF newValue = CvXlit.limited THEN
        {
        FormWindow.SetVisibility[
            window: window,
            item: CvXlit.MessageKey.charsSuffix.ORD,
            visibility: visible,
            repaint: FALSE];
        FormWindow.SetVisibility[
            window: window,
            item: CvXlit.MessageKey.wordWrap.ORD,
            visibility: visible,
            repaint: TRUE];
        }
    ELSE
        {
        FormWindow.SetVisibility[
            window: window,
            item: CvXlit.MessageKey.charsSuffix.ORD,
            visibility: invisible,
            repaint: FALSE];
        FormWindow.SetVisibility[
            window: window,
            item: CvXlit.MessageKey.wordWrap.ORD,
            visibility: invisible,
            repaint: TRUE];
        };
};

--/* Table Procs */

<<
FindTableFolder: PROC [directory: NSFile.Handle]
    RETURNS [tableFolder: NSFile.Handle ← NSFile.nullHandle] = {
    filters: ARRAY [0..2) OF NSFile.Filter ← [
        [matches[[name[NSString.StringFromMesaString["Transliteration Tables"L]]]]],
        [equal[[type[NSAssignedTypes.tDirectory]]]]];

    -- should look on desktop and then in System catalog

    tableFolder ← NSFile.Find[
        directory: directory, scope: [filter: [and[DESCRIPTOR [filters]]]] ! NSFile.Error = > CONTINUE];
    directory ← Catalog.GetFile [name: folderName, readonly: TRUE];
    FileFromName [fileName];
    NSFile.Close [directory];
};
>>
```

```
ListTables: PROC [folder: NSFile.Handle] = {
    filters: ARRAY [0..2) OF NSFile.Filter ← [
      [matches[[name[NSString.StringFromMesaString["*.avTable"L]]]]],
      [equal[[type[NSAssignedTypes.tUnspecified]]]]];
    scope: NSFile.Scope ← [
      count: 30,
      filter: [and[DESCRIPTOR [filters]]]];
    selections: NSFile.Selections;

    CopyTableName: NSFile.AttributesProc = {
        cd: LONG POINTER TO Global ← clientData;
        name: XString.ReaderBody ← XString.FromNSString[attributes.name];
        cd.avCount ← cd.avCount + 1;
        cd.avChoices[cd.avCount] ← [string[
          choiceNumber: cd.avCount,
          string: XString.CopyReader[@name, cd.cz] ↑ ]];
    };

    selections.interpreted[name] ← TRUE;
    g.avCount ← 0;
    NSFile.List[
      directory: folder,
      proc: CopyTableName,
      selections: selections,
      scope: scope,
      clientData: @g ! NSFile.Error = > CONTINUE];
};


ListVATables: PROC [folder: NSFile.Handle] = {
    filters: ARRAY [0..2) OF NSFile.Filter ← [
      [matches[[name[NSString.StringFromMesaString["*.vaTable"L]]]]],
      [equal[[type[NSAssignedTypes.tUnspecified]]]]];
    scope: NSFile.Scope ← [
      count: 30,
      filter: [and[DESCRIPTOR [filters]]]];
    selections: NSFile.Selections;

    CopyTableName: NSFile.AttributesProc = {
        cd: LONG POINTER TO Global ← clientData;
        name: XString.ReaderBody ← XString.FromNSString[attributes.name];
        cd.vaCount ← cd.vaCount + 1;
        cd.vaChoices[cd.vaCount] ← [string[
          choiceNumber: cd.vaCount,
          string: XString.CopyReader[@name, cd.cz] ↑ ]];
    };

    selections.interpreted[name] ← TRUE;
    g.vaCount ← 0;
    NSFile.List[
      directory: folder,
      proc: CopyTableName,
      selections: selections,
      scope: scope,
      clientData: @g ! NSFile.Error = > CONTINUE];
};


GetTableFolder: PROC RETURNS [folder: NSFile.Handle] = {
    -- assume folder is in System catalog
    folderName: XString.ReaderBody ← XString.FromSTRING["Transliteration Tables"L];
    folder ← Catalog.GetFile[name: @folderName, readonly: TRUE];
};


GetTableFile: PROC [tableName: XString.ReaderBody] RETURNS [file: NSFile.Handle] = {
    -- assume folder is in System catalog
    folderName: XString.ReaderBody ← XString.FromSTRING["Transliteration Tables"L];
    ref: NSFile.Reference ← TRASH;
    ref ← GetFile[@folderName, @tableName];
    file ← NSFile.OpenByReference[ref];
};


GetFile: PROC [folderName, fileName: XString.Reader]
    RETURNS [file: NSFile.Reference ← NSFile.nullReference] = {
    directory: NSFile.Handle ← TRASH;

    FileFromName: PROC [value: XString.Reader] = {
        nsName: NSString.String ← XString.NSStringFromReader[
          r: value, z: BWSZone.logonSession];
        handle: NSFile.Handle ← NSFile.nullHandle;

        handle ← NSFile.Find[
          directory: directory,
          scope: [filter: [matches[attribute: [name[nsName]]]]]
            ! NSFile.Error = > {handle ← NSFile.nullHandle; CONTINUE}];

        IF handle # NSFile.nullHandle THEN {
          file ← NSFile.GetReference[handle];
          NSFile.Close[handle]};
        NSString.FreeString[z: BWSZone.logonSession, s: nsName];
    };

    directory ← Catalog.GetFile[name: folderName, readonly: TRUE];
    FileFromName[fileName];
    NSFile.Close[directory];
```

```
};

NewUserMap: PROC [userMap: CvXlit.VPToAsciiTable] = {

    --/* initialize conversion maps */

    FOR s: CARDINAL IN [0..256) DO
        IF userMap[s] # NIL THEN userMap[s] ← Space.Unmap[userMap[s]];
    ENDLOOP;

    userMap[0] ← Space.ScratchMap[(charMapSize + Environment.wordsPerPage-1) / Environment.wordsPerPage];
    userMap[41B] ← Space.ScratchMap[(charMapSize + Environment.wordsPerPage-1) / Environment.wordsPerPage];
    userMap[357B] ← Space.ScratchMap[(charMapSize + Environment.wordsPerPage-1) / Environment.wordsPerPage];

    FOR c: CARDINAL IN [0..256) DO
        userMap[0][c] ← VAL[c];
        userMap[41B][c] ← VAL[0];
        userMap[357B][c] ← VAL[0];
    ENDLOOP;

    userMap[0][211B] ← VAL[11B];          -- 211B -> tab
    userMap[0][244B] ← VAL[44B];          -- dollar -> $
    userMap[0][252B] ← VAL[42B];          -- leftDoubleQuote -> "
    userMap[0][272B] ← VAL[42B];          -- rightDoubleQuote -> "
    userMap[0][251B] ← VAL[47B];          -- leftSingleQuote -> '
    userMap[0][271B] ← VAL[47B];          -- rightSingleQuote -> '
    userMap[41B][76B] ← VAL[55B];         -- hyphen -> minus
    userMap[357B][42B] ← VAL[55B];        -- nonBreakingHyphen -> minus
    userMap[357B][41B] ← VAL[40B];        -- nonBreakingSpace -> space
};

END...
```

LOG
16-Mar-87 14:06:16 - Caro - Created
24-Nov-87 16:58:56 - Erickson - Changed paraEndsWith default to <CR> instead of <CR><LF>
17-Dec-87 15:48:52 - Erickson - AR 16414 - Added to ApplyChanges in the CvXlit.MessageKey.charsSuffix section. The value read from the prop sheet was expected to be a valid number. If text was entered, the converter crashed the system. I added signal checking for InvalidNumber and Overflow. If text is entered, the InvalidNumber signal is raised by FormWindow.GetIntegerItemValue, and is caught here. The user's input is then highlighted, that field of the propsheet is made the input focus, and a message is posted indicating the problem. This message was placed in the extraErr1 position in CvXlitMsgFileImpl.mesa. While I was here, I added a catch phrase for the Overflow signal also, this simply sets the input value to zero and allows the already existing code to treat this as input out of range.

```
-- File: CvXlitMainImpl.mesa
-- Trow 7-Sep-89 16:39:53

-- Last Revised by: Caro                                    30-Jun-87 12:39:53
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (c) 1987, 1988 by Xerox Corporation.  All rights reserved.

DIRECTORY
   Atom
      USING [MakeAtom],
   Attention
      USING [Post],
   BWSZone
      USING [Permanent],
   Context
      USING [Create, Error, Find, NopDestroyProc, Type, UniqueType],
   Converter
      USING [DestinationOptions, GetEventType, Register, Status, SourceOptions],
   ConverterMsg
      USING [Get, kvpDocument],
   ConverterPFOptions
      USING [conASCII],
   CvXlit
      USING [AsciiToVP, AsciiToVPDstOps, AsciiToVPSrcOps, Common,
      GetMessage, leadingMargin, MessageKey,
      pointsBetweenItems, ProblemType, VPToAscii, VPToAsciiDstOps],
   Event
      USING [AddDependency, AgentProcedure, EventType],
   Process
      USING [Detach, Pause, SecondsToTicks],
   ProductFactoring
      USING [Enabled],
   SimpleTextDisplay
      USING [MeasureString],
   StarFileTypes
      USING [document, text, unspecified],
   Window
      USING [Handle],
   XString
      USING [ReaderBody];
<<
--=============================================================================
-- OVERVIEW:

Main code for ascii conversion.  Registations done here.
--=============================================================================
>>

CvXlitMainImpl: PROGRAM
IMPORTS
   Atom, Attention, BWSZone, Context, Converter, ConverterMsg,
   CvXlit, Event, Process, ProductFactoring, SimpleTextDisplay
EXPORTS
   CvXlit =
BEGIN

--====================
-- CONSTANTS
--====================

--====================
-- TYPES
--====================

Globals: TYPE = RECORD [
      leads: ItemLeads,
      ctype: Context.Type,
      z: UNCOUNTED ZONE];

ItemLeads: TYPE = ARRAY CvXlit.MessageKey[paraEndsWith..lastPsheetItem] OF CARDINAL;

--====================
-- GLOBALS
--====================

g: Globals;

--====================
-- PUBLIC SIGNALS
--====================

Problem: PUBLIC SIGNAL [err: CvXlit.ProblemType] = CODE;

--====================
-- PUBLIC PROCEDURES
--====================

DataFromWindow: PUBLIC PROC [w: Window.Handle] RETURNS [my: CvXlit.Common] = {
      my ← Context.Find[type: g.ctype, window: w ! Context.Error => {my ← NIL; CONTINUE}];
};


DataToWindow: PUBLIC PROC [my: CvXlit.Common, w: Window.Handle] = {
      Context.Create[
         type: g.ctype,
         data: my,
```

```
        proc: Context.NopDestroyProc,
        window: w ! Context.Error => CONTINUE];
};

GetPreMargin: PUBLIC PROC [item: CvXlit.MessageKey] RETURNS [leads: CARDINAL] = {
    RETURN[g.leads[item]];
};

--====================
-- PROCEDURES
--====================

Init: PROC = {
    z: UNCOUNTED ZONE = BWSZone.Permanent[];

    g ← [
      leads: ALL[CARDINAL.LAST],
      ctype: Context.UniqueType[],
      z: z];

    MeasureTags[];

    --/* register with converter icon */
    Register[];
};

MeasureTags: PROC = {
    lmarg: CARDINAL = CvXlit.leadingMargin;
    max: CARDINAL ← 0;
    --/* local proc */
    Length: PROC [key: CvXlit.MessageKey] RETURNS [width: CARDINAL] =
        {
        rb: XString.ReaderBody ← CvXlit.GetMessage[key];
        [width: width] ← SimpleTextDisplay.MeasureString[string: @rb];
        RETURN [width];
        };

    --/* begin code */
    g.leads ← [
      paraEndsWith: Length[paraEndsWith],
      fontSize: Length[fontSize],
      fontSizeChoices: 0,
      font: Length[font],
      fontChoices: 0,
      ignoreTrailing: 1,  -- no tag
      lineLen: Length[lineLen],
      lineLenChoices: 0,
      charsSuffix: CARDINAL.LAST,
      wordWrap: CARDINAL.LAST,
      endLine: Length[endLine],
      endPara: Length[endPara],
      replaceUnknown: Length[replaceUnknown],
      transliterationTable: Length[transliterationTable],
      spareP1: 0,
      spareP2: 0,
      spareP3: 0,
      spareP4: 0,
      spareP5: 0,
      lastPsheetItem: 0];

    --/* now determine max */
    FOR i: CvXlit.MessageKey IN CvXlit.MessageKey[paraEndsWith..lastPsheetItem] DO
        IF g.leads[i] = CARDINAL.LAST THEN LOOP;
        max ← MAX[max, g.leads[i]];
    ENDLOOP;

    --/* now adjust */
    max ← max + lmarg;
    FOR i: CvXlit.MessageKey IN CvXlit.MessageKey[paraEndsWith..lastPsheetItem] DO
        SELECT g.leads[i] FROM
            0 => LOOP;
            1 => g.leads[i] ← max + 8;  -- compensate for no tag
            CARDINAL.LAST => g.leads[i] ← CvXlit.pointsBetweenItems;
            ENDCASE => g.leads[i] ← max - g.leads[i];
    ENDLOOP;
};


RegisterNow: PROC [first: BOOLEAN] RETURNS [allOk: BOOLEAN ← TRUE] = {
    doc: XString.ReaderBody ← ConverterMsg.Get[ConverterMsg.kvpDocument];
    asciiDoc: XString.ReaderBody ← CvXlit.GetMessage[asciiDoc];
    status: Converter.Status;
    --/* local proc */
    Check: PROC [status: Converter.Status] =
        {
        SELECT status FROM
            registered, alreadyExisted, overridden => NULL;
            busy =>
                {
                IF first THEN
                    {
                    et: Event.EventType ← Converter.GetEventType[];
                    --/* tell user registration will be done later */
```

```
                        --+$$$ not implemented

                        [] ← Event.AddDependency[
                            agent: RetryRegistration,
                            myData: NIL,
                            event: et];
                        first ← FALSE;  --/* only add once! */
                        };
                    allOk ← FALSE;
                    };
                error => allOk ← FALSE;  --+$$$ should post a message
                ENDCASE;
            };

        --/* begin code */
        status ← Converter.Register[
            srcType: StarFileTypes.text,
            srcFormat: @asciiDoc,
            destFormat: @doc,
            convertProc: CvXlit.AsciiToVP,
            sizeChange: 190,
            forkable: TRUE].status;

        Check[status];

        status ← Converter.Register[
            srcType: StarFileTypes.unspecified,
            srcFormat: @asciiDoc,
            destFormat: @doc,
            convertProc: CvXlit.AsciiToVP,
            sizeChange: 190,
            forkable: TRUE].status;

        Check[status];

        status ← Converter.Register[
            srcType: StarFileTypes.document,
            srcFormat: @doc,
            destFormat: @asciiDoc,
            convertProc: CvXlit.VPToAscii,
            sizeChange: 63,
            forkable: TRUE].status;

        Check[status];

        --/* register ops */

        IF NOT allOk THEN RETURN;

        status ← Converter.DestinationOptions[
            srcFormat: @doc,
            destFormat: @asciiDoc,
            dependentOptions: CvXlit.VPToAsciiDstOps,
            override: TRUE].status;

        Check[status];

        status ← Converter.SourceOptions[
            srcFormat: @asciiDoc,
            destFormat: @doc,
            dependentOptions: CvXlit.AsciiToVPSrcOps,
            override: TRUE].status;

        Check[status];

        status ← Converter.DestinationOptions[
            srcFormat: @asciiDoc,
            destFormat: @doc,
            dependentOptions: CvXlit.AsciiToVPDstOps,
            override: TRUE].status;

        Check[status];

};


RetryRegistration: Event.AgentProcedure = {
    IF RegisterNow[first: FALSE].allOk THEN remove ← TRUE;
};


RetryProductFactoring: Event.AgentProcedure = {
    IF NOT ProductFactoring.Enabled[option: ConverterPFOptions.conASCII] THEN
        {
        msg: XString.ReaderBody ← CvXlit.GetMessage[notPF];

        Attention.Post[@msg];
        remove ← FALSE;
        }
    ELSE
        {
        Process.Detach[FORK AvoidDeadlock[]];
        remove ← TRUE;
        };
};
```

```
<<
--=========================================================================================
AvoidDeadlock
* Finish doing registrations in another process, to make sure we don't try to AddDependency from inside of an AgentProcedure.
--=========================================================================================
>>
AvoidDeadlock: PROC = {
    Process.Pause[Process.SecondsToTicks[2]];  --/* give other process a chance */
    [] ← RegisterNow[first: TRUE];
};


Register: PROCEDURE = {
    IF NOT ProductFactoring.Enabled[option: ConverterPFOptions.conASCII] THEN
        {
        msg: XString.ReaderBody ← CvXlit.GetMessage[notPF];
        logon: Event.EventType ← Atom.MakeAtom["LogonCompleted"L];

        Attention.Post[@msg];
        [] ← Event.AddDependency[
          agent: RetryProductFactoring,
          myData: NIL,
          event: logon];
        }
    ELSE [] ← RegisterNow[first: TRUE]; -- OK
};

--/* MAIN code */

Init[];


END...

LOG
16-Mar-87 14:06:16 - Caro - Created
30-Jun-87 12:39:59 - Caro - MDS relief, RetryProductFactoring
```

```
-- File: CvXlitMsgFileImpl.mesa
-- Trow 7-Sep-89 16:40:41

-- Last Revised by: Erickson                                    17-Dec-87 16:06:35
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (c) 1985, 1986, 1987, 1988, 1989 by Xerox Corporation.  All rights reserved.

DIRECTORY
  ApplicationFolderExtra
    USING [InitMessages],
  CvXlit,
  NSFile
    USING [Error],
  Runtime
    USING [UnboundProcedure],
  XMessage
    USING [AllocateMessages, Get, Handle, MsgEntry, RegisterMessages],
  XString
    USING [FromSTRING, nullReaderBody, ReaderBody];

CvXlitMsgFileImpl: PROGRAM
IMPORTS
  ApplicationFolderExtra, NSFile, Runtime, XMessage, XString
EXPORTS CvXlit =
BEGIN

--=====================
-- GLOBALS
--=====================

h: XMessage.Handle ← NIL;

--=====================
-- SIGNALS
--=====================

NoMessageFile: ERROR = CODE;

--=====================
-- PUBLIC PROCEDURES
--=====================

GetMessage: PUBLIC PROCEDURE [msg: CvXlit.MessageKey] RETURNS [msgRb: XString.ReaderBody] = {
    IF h # NIL THEN RETURN[h.Get[msg.ORD]];
    RETURN[XString.nullReaderBody];
};

--=====================
-- PROCEDURES
--=====================

InitMessages: PROCEDURE = {
    internalName: XString.ReaderBody ← XString.FromSTRING["FC Xlit Documents"L];
    messageFile: XString.ReaderBody ← XString.FromSTRING["MessageFile"L];

    h ← ApplicationFolderExtra.InitMessages[
      internalName: @internalName,
      label: @messageFile,
      domainindex: 0 ! ANY = > {h ← NIL; CONTINUE}];
    IF h = NIL THEN
        InitFromArray[];
};


InitFromArray: PROC = {
    h ← XMessage.AllocateMessages["Xlit Conversion"L, CvXlit.MessageKey.LAST.ORD.SUCC, NIL, NIL];

    Init0to20[];
    Init21toLAST[];
};


Init0to20: PROC = {
    msgArray: ARRAY CvXlit.MessageKey[asciiDoc..lastPsheetItem] OF XMessage.MsgEntry ← [
      asciiDoc: [
        msgKey: CvXlit.MessageKey.asciiDoc.ORD,
        msg: XString.FromSTRING["Transliterated Text"L],
        type: userMsg,
        translationNote: "Label for source or destination of conversion"L,
        translatable: FALSE,
        id: 0],
      paraEndsWith: [
        msgKey: CvXlit.MessageKey.paraEndsWith.ORD,
        msg: XString.FromSTRING["Paragraph ends with"L],
        type: pSheetItem,
        translationNote: "Tag for text item, should read as if user were filling in the blank/completing the sentence"L,
        translatable: TRUE,
        id: 1],
      fontSize: [
        msgKey: CvXlit.MessageKey.fontSize.ORD,
        msg: XString.FromSTRING["Font size"L],
        type: pSheetItem,
        translationNote: "Choice item tag"L,
        translatable: TRUE,
        id: 2],
      fontSizeChoices: [
```

```
    msgKey: CvXlit.MessageKey.fontSizeChoices.ORD,
    msg: XString.FromSTRING["12:0@18:1@24:2"L],
    type: argList,
    translationNote: "Choices that go with id#2"L,
    translatable: FALSE,
    id: 3],
  font: [
    msgKey: CvXlit.MessageKey.font.ORD,
    msg: XString.FromSTRING["Font"L],
    type: pSheetItem,
    translationNote: "Choice item tag"L,
    translatable: TRUE,
    id: 4],
  fontChoices: [
    msgKey: CvXlit.MessageKey.fontChoices.ORD,
    msg: XString.FromSTRING["Modern:0@Classic:1"L],
    type: argList,
    translationNote: "Choices that go with id#4"L,
    translatable: TRUE,
    id: 5],
  ignoreTrailing: [
    msgKey: CvXlit.MessageKey.ignoreTrailing.ORD,
    msg: XString.FromSTRING["IGNORE TRAILING WHITE SPACE"L],
    type: pSheetItem,
    translationNote: "Boolean item"L,
    translatable: TRUE,
    id: 6],
  lineLen: [
    msgKey: CvXlit.MessageKey.lineLen.ORD,
    msg: XString.FromSTRING["Line length"L],
    type: pSheetItem,
    translationNote: "Choice item tag"L,
    translatable: TRUE,
    id: 7],
  lineLenChoices: [
    msgKey: CvXlit.MessageKey.lineLenChoices.ORD,
    msg: XString.FromSTRING["Unlimited:0@Limited:1"L],
    type: argList,
    translationNote: "Choices that go with id#7"L,
    translatable: TRUE,
    id: 8],
  charsSuffix: [
    msgKey: CvXlit.MessageKey.charsSuffix.ORD,
    msg: XString.FromSTRING["characters"L],
    type: pSheetItem,
    translationNote: "Suffix for number item -- to be read e.g. '[80] characters'"L,
    translatable: TRUE,
    id: 9],
  wordWrap: [
    msgKey: CvXlit.MessageKey.wordWrap.ORD,
    msg: XString.FromSTRING["WORD WRAP"L],
    type: pSheetItem,
    translationNote: "Boolean item, indicating that text lines should break only on the white space between words"L,
    translatable: TRUE,
    id: 10],
  endLine: [
    msgKey: CvXlit.MessageKey.endLine.ORD,
    msg: XString.FromSTRING["End line with"L],
    type: pSheetItem,
    translationNote: "Text item tag, should read as if user is filling in the blank/completing sentence"L,
    translatable: TRUE,
    id: 11],
  endPara: [
    msgKey: CvXlit.MessageKey.endPara.ORD,
    msg: XString.FromSTRING["End paragraph with"L],
    type: pSheetItem,
    translationNote: "Text item tag, should read as if user is filling in the blank/completing sentence"L,
    translatable: TRUE,
    id: 12],
  replaceUnknown: [
    msgKey: CvXlit.MessageKey.replaceUnknown.ORD,
    msg: XString.FromSTRING["Replace unknown character with"L],
    type: pSheetItem,
    translationNote: "Text item tag, should read as if user is filling in the blank/completing sentence"L,
    translatable: TRUE,
    id: 13],
  transliterationTable: [
    msgKey: CvXlit.MessageKey.transliterationTable.ORD,
    msg: XString.FromSTRING["Transliteration table"L],
    type: pSheetItem,
    translationNote: "Choice item tag"L,
    translatable: TRUE,
    id: 14],
  spareP1: [
    msgKey: CvXlit.MessageKey.spareP1.ORD,
    msg: XString.FromSTRING[""L],
    type: others,
    translationNote: "DO NOT TRANSLATE -- spare key"L,
    translatable: TRUE,
    id: 15],
  spareP2: [
    msgKey: CvXlit.MessageKey.spareP2.ORD,
    msg: XString.FromSTRING[""L],
    type: others,
    translationNote: "DO NOT TRANSLATE -- spare key"L,
    translatable: TRUE,
    id: 16],
```

```
        spareP3: [
          msgKey: CvXlit.MessageKey.spareP3.ORD,
          msg: XString.FromSTRING[""L],
          type: others,
          translationNote: "DO NOT TRANSLATE -- spare key"L,
          translatable: TRUE,
          id: 17],
        spareP4: [
          msgKey: CvXlit.MessageKey.spareP4.ORD,
          msg: XString.FromSTRING[""L],
          type: others,
          translationNote: "DO NOT TRANSLATE -- spare key"L,
          translatable: TRUE,
          id: 18],
        spareP5: [
          msgKey: CvXlit.MessageKey.spareP5.ORD,
          msg: XString.FromSTRING[""L],
          type: others,
          translationNote: "DO NOT TRANSLATE -- spare key"L,
          translatable: TRUE,
          id: 19],
        lastPsheetItem: [
          msgKey: CvXlit.MessageKey.lastPsheetItem.ORD,
          msg: XString.FromSTRING[""L],
          type: others,
          translationNote: "DO NOT TRANSLATE -- spare key"L,
          translatable: TRUE,
          id: 20]
        ];

    XMessage.RegisterMessages[h, LOOPHOLE[LONG[DESCRIPTOR[msgArray]]], FALSE];
};


Init21toLAST: PROC = {
    msgArray: ARRAY CvXlit.MessageKey[left..CvXlit.MessageKey.LAST] OF XMessage.MsgEntry ← [
      left: [
        msgKey: CvXlit.MessageKey.left.ORD,
        msg: XString.FromSTRING["<"L],
        type: others,
        translationNote: "do not translate"L,
        translatable: FALSE,
        id: 21],
      right: [
        msgKey: CvXlit.MessageKey.right.ORD,
        msg: XString.FromSTRING[">"L],
        type: others,
        translationNote: "do not translate"L,
        translatable: FALSE,
        id: 22],
      cr: [
        msgKey: CvXlit.MessageKey.cr.ORD,
        msg: XString.FromSTRING["CR"L],
        type: others,
        translationNote: "do not translate"L,
        translatable: FALSE,
        id: 23],
      lf: [
        msgKey: CvXlit.MessageKey.lf.ORD,
        msg: XString.FromSTRING["LF"L],
        type: others,
        translationNote: "do not translate"L,
        translatable: FALSE,
        id: 24],
      nl: [
        msgKey: CvXlit.MessageKey.nl.ORD,
        msg: XString.FromSTRING["NL"L],
        type: others,
        translationNote: "do not translate"L,
        translatable: FALSE,
        id: 25],
      ff: [
        msgKey: CvXlit.MessageKey.ff.ORD,
        msg: XString.FromSTRING["FF"L],
        type: others,
        translationNote: "do not translate"L,
        translatable: FALSE,
        id: 26],
      tab: [
        msgKey: CvXlit.MessageKey.tab.ORD,
        msg: XString.FromSTRING["TAB"L],
        type: others,
        translationNote: "do not translate"L,
        translatable: FALSE,
        id: 27],
      createError: [
        msgKey: CvXlit.MessageKey.createError.ORD,
        msg: XString.FromSTRING["The source object was not converted due to an error while creating the output file."L],
        type: errorMsg,
        translationNote: "Posted to attention window"L,
        translatable: TRUE,
        id: 28],
      notPF: [
        msgKey: CvXlit.MessageKey.notPF.ORD,
        msg: XString.FromSTRING["Transliterated Text Conversion cannot be activated because required Software Option not enabled.  Please
        enable Software Option, End Session, then Logon again."L],
        type: errorMsg,
```

```
    translationNote: "posted to attention window"L,
    translatable: TRUE,
    id: 29],
  paginating: [
    msgKey: CvXlit.MessageKey.paginating.ORD,
    msg: XString.FromSTRING[" paginating ... "L],
    type: userMsg,
    translationNote: "posted to attention window following 'Converting xyz ... ' converter icon message. The leading and trailing
    spaces are REQUIRED"L,
    translatable: TRUE,
    id: 30],
  skippedTableData: [
    msgKey: CvXlit.MessageKey.skippedTableData.ORD,
    msg: XString.FromSTRING[" Some data in '<>' was skipped ... "L],
    type: template,
    translationNote: "Some table data skipped. Leading and trailing blanks REQUIRED."L,
    translatable: TRUE,
    id: 31],
  dfltAVEndParagraph: [
    msgKey: CvXlit.MessageKey.dfltAVEndParagraph.ORD,
    msg: XString.FromSTRING["<CR><CR>"L],
    type: others,
    translationNote: "do not translate, default value for text items"L,
    translatable: FALSE,
    id: 32],
  dfltAVReplaceCharacter: [
    msgKey: CvXlit.MessageKey.dfltAVReplaceCharacter.ORD,
    msg: XString.FromSTRING["$"L],
    type: others,
    translationNote: "do not translate, default value for text items"L,
    translatable: FALSE,
    id: 33],
  prefix: [
    msgKey: CvXlit.MessageKey.prefix.ORD,
    msg: XString.FromSTRING["CvXlit"L],
    type: others,
    translationNote: "do not translate, internal file name prefix"L,
    translatable: FALSE,
    id: 34],
  doneFailed: [
    msgKey: CvXlit.MessageKey.doneFailed.ORD,
    msg: XString.FromSTRING["Unrecoverable error writing Transliterated Text conversion properties. Cancel the property sheet and
    use a new converter icon."L],
    type: errorMsg,
    translationNote: "Posted when user selects Done on property sheet, if there is an NSFile or other error"L,
    translatable: TRUE,
    id: 35],
  backstop: [
    msgKey: CvXlit.MessageKey.backstop.ORD,
    msg: XString.FromSTRING["Problem: the details section could not be created."L],
    type: pSheetItem,
    translationNote: "For some reason, creation of the client details window failed. This string is put in the formwindow
    instead."L,
    translatable: TRUE,
    id: 36],
  metaError: [
    msgKey: CvXlit.MessageKey.metaError.ORD,
    msg: XString.FromSTRING["The selected text item contains an error. Please correct it."L],
    type: errorMsg,
    translationNote: "This message is posted to the Attention window when the user tries to Done or Start a sheet with a text/syntax
    error. Text syntax is described in the Reference Library documentation for ASCII."L,
    translatable: TRUE,
    id: 37],
  charsOutOfBounds: [
    msgKey: CvXlit.MessageKey.charsOutOfBounds.ORD,
    msg: XString.FromSTRING["The line length limit must be between 10 and 256 characters, inclusive. Please reenter."L],
    type: errorMsg,
    translationNote: "Posted when user tries to Done or Start a sheet with an invalid numeric value."L,
    translatable: TRUE,
    id: 38],
  fatalError: [
    msgKey: CvXlit.MessageKey.fatalError.ORD,
    msg: XString.FromSTRING[" conversion failed with an unrecoverable error "L],
    type: errorMsg,
    translationNote: "Posted if NSFile or other error in conversion. Note that leading and trailing blanks are required."L,
    translatable: TRUE,
    id: 39],
  extraErr0: [
    msgKey: CvXlit.MessageKey.extraErr0.ORD,
    msg: XString.FromSTRING[" Unrecoverable Transliterated Text conversion error: damaged converter icon. "L],
    type: errorMsg,
    translationNote: "Blanks are required. Posted if the conversion cannot read properties from the converter icon."L,
    translatable: TRUE,
    id: 40],
  extraErr1: [
    msgKey: CvXlit.MessageKey.extraErr1.ORD,
    msg: XString.FromSTRING["The number in the highlighted field is invalid. Please reenter."L],
    type: errorMsg,
    translationNote: "Posted when the user tries to Done or Start a sheet with text in a numeric field."L,
    translatable: TRUE,
    id: 41],
  dfltVAEndLine: [
    msgKey: CvXlit.MessageKey.dfltVAEndLine.ORD,
    msg: XString.FromSTRING["<CR>"L],
    type: others,
    translationNote: "do not translate, default value for text items"L,
    translatable: FALSE,
```

```
            id: 42],
        dfltVAEndParagraph: [
            msgKey: CvXlit.MessageKey.dfltVAEndParagraph.ORD,
            msg: XString.FromSTRING["<CR>"L],
            type: others,
            translationNote: "do not translate, default value for text items"L,
            translatable: FALSE,
            id: 43],
        dfltVAReplaceCharacter: [
            msgKey: CvXlit.MessageKey.dfltVAReplaceCharacter.ORD,
            msg: XString.FromSTRING["$"L],
            type: others,
            translationNote: "do not translate, default value for text items"L,
            translatable: FALSE,
            id: 44]

<<
        «»: [
            msgKey: CvXlit.MessageKey.USEAGAINTOREPLACETHISSTRING.ORD,
            msg: XString.FromSTRING["«»"L],
            type: «»,
            translationNote: "«»"L,
            translatable: TRUE,
            id: «»],
>>
        ];

    XMessage.RegisterMessages[h, LOOPHOLE[LONG[DESCRIPTOR[msgArray]]], FALSE];
};



--/* MAIN line code */

InitMessages[! NSFile.Error, Runtime.UnboundProcedure => NoMessageFile];

END...


LOG
24-Apr-85 12:12:27 - MSchneider - CREATED from SampleBWSApplicationMsgFileImpl
10-May-85 10:56:18 - MSchneider - used correct ApplicationFolder name
28-May-85 9:28:54 - MSchneider - moved localZone into procedure, added use of BWSZone
24-Jun-85 14:33:55 - MSchneider - made "MessageFile" be "MessageFile" in entry name
 9-Jul-85 11:12:31 - MSchneider - added ERROR NoMessageFile
26-Feb-87 14:59:12 - Caro - Upgraded to VP 2.0 (delete 90% of code)
 8-Apr-87 11:43:56 - Caro - Catch ANY error raised from InitMessages
26-Jun-87 11:10:51 - Caro - Made #44 a real error
19-Aug-87 10:51:37 - Caro - Reworded several messages and transNotes
24-Nov-87 17:01:04 - Erickson - added aToVDfltMeta (ID = 46) to change default for ascii to ViewPoint treatment of paraEndsWith.
17-Dec-87 16:04:02 - Erickson - AR 16414 - made #45 a real error, bad number input.
```

```
-- File: CvXlitParseImpl.mesa
-- Trow 17-Aug-89 4:48:30

-- Last Revised by: Caro                                    29-Jun-87 11:31:40
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (c) 1987, 1989 by Xerox Corporation.  All rights reserved.

DIRECTORY
  Ascii
    USING [CR, FF, LF, TAB],
  Attention
    USING [Post],
  CvXlit
    USING [Common, GetMessage, MessageKey],
  FormWindow
    USING [SetSelection, SetInputFocus],
  String
    USING [AppendChar, CopyToNewString, MakeString, StringBoundsFault],
  XChar
    USING [Character, Code, not],
  XString
    USING [AppendChar, ClearWriter, CopyToNewReaderBody,
    Empty, Equal, First, FromSTRING, FreeReaderBytes, FreeWriterBytes,
    InvalidEncoding, Lop, NewWriterBody,
    Reader, ReaderBody, ReaderFromWriter, ValidateReader, Writer, WriterBody];

<<
--=============================================================================
-- OVERVIEW:

Parse text items containing meta characters into strings.
--=============================================================================
>>

CvXlitParseImpl: PROGRAM
IMPORTS
  Attention, CvXlit, FormWindow, String, XChar, XString
EXPORTS
  CvXlit =
BEGIN

--====================
-- CONSTANTS
--====================

max: CARDINAL = 10;
maxAbbr: CARDINAL = 3;  --/* abbreviations only up to 3 characters */
maxOctals: CARDINAL = 3;  --/* need exactly 3 octal digits */

--====================
-- TYPES
--====================

ParseStates: TYPE = {
    entry,
    beginMeta,
    doOctal,
    doAbbrev
    };
--====================
-- SIGNALS
--====================

ParseError: SIGNAL [err: ErrType ← syntaxError, start, pos: CARDINAL] = CODE;

ErrType: TYPE =
    {
    syntaxError,
    invalidMeta,
    unknownAbbr,
    invalidOctal,
    invalidEncoding
    };

--====================
-- PUBLIC PROCEDURES
--====================

ParseItem: PUBLIC PROC [my: CvXlit.Common, r: XString.Reader, item: CvXlit.MessageKey, buf: XString.Writer ← NIL] RETURNS [ok: BOOLEAN,
ls: LONG STRING] = {
    bufRb: XString.WriterBody;
    tmpRb: XString.ReaderBody;
    msgRb: XString.ReaderBody;
    clientBuf: BOOLEAN;

    IF buf = NIL THEN
        {
        bufRb ← XString.NewWriterBody[maxLength: 30, z: my.z];
        buf ← @bufRb;
        clientBuf ← FALSE;
        }
    ELSE
        clientBuf ← TRUE;

    BEGIN
      ENABLE ParseError =>
```

```
            {
            msgRb ← CvXlit.GetMessage[metaError];
            IF my.window = NIL OR item = CvXlit.MessageKey.FIRST THEN GOTO notOK;

            FormWindow.SetSelection[
               window: my.window,
               item: item.ORD,
               firstChar: start,
               lastChar: pos];
            FormWindow.SetInputFocus[
               window: my.window,
               item: item.ORD,
               beforeChar: pos];
            Attention.Post[@msgRb];
            ls ← NIL;
            GOTO notOK;
            };

      tmpRb ← XString.CopyToNewReaderBody[r: r, z: my.z];
      ls ← ParseToLS[text: @tmpRb, z: my.z, buf: buf];

      --/* test for invalid encoding */
      IF my.owner = AtoVdst THEN
            {
            msgRb ← XString.FromSTRING[ls];
            XString.ValidateReader[@msgRb ! XString.InvalidEncoding = >
                SIGNAL ParseError[
                   err: invalidEncoding,
                   start: 0,
                   pos: CARDINAL.LAST]];
             };

      ok ← TRUE;
      EXITS notOK = > ok ← FALSE;
      END;

      IF NOT clientBuf THEN
            XString.FreeWriterBytes[buf];
      XString.FreeReaderBytes[r: @tmpRb, z: my.z];
};

--=====================
-- PROCEDURES
--=====================


ParseToLS: PROC [text: XString.Reader, z: UNCOUNTED ZONE, buf: XString.Writer] RETURNS [ls: LONG STRING ← NIL] = {
      rb: XString.ReaderBody ← CvXlit.GetMessage[left];
      state: ParseStates ← entry;
      start,
      pos: CARDINAL ← 0;
      octals,
      abbrs: CARDINAL ← 0;
      cr: XString.ReaderBody;
      lf: XString.ReaderBody;
      nl: XString.ReaderBody;
      ff: XString.ReaderBody;
      tab: XString.ReaderBody;
      left: XChar.Character;
      right: XChar.Character;
      xc: XChar.Character;
      c: CHARACTER;
      octalValue: CARDINAL[0..255];

      --/* get < and > */
      left ← XString.First[@rb];
      rb ← CvXlit.GetMessage[right];
      right ← XString.First[@rb];

      --/* initialize strings */
      IF XString.Empty[text] THEN
            RETURN[ls: NIL]
      ELSE
            ls ← String.MakeString[z: z, maxlength: max];
      cr ← CvXlit.GetMessage[cr];
      lf ← CvXlit.GetMessage[lf];
      nl ← CvXlit.GetMessage[nl];
      ff ← CvXlit.GetMessage[ff];
      tab ← CvXlit.GetMessage[tab];

      --/* lop through string */
      DO
         ENABLE
            {
            String.StringBoundsFault = >
               {
               ns ← String.CopyToNewString[s: ls, z: z, longer: max];
               z.FREE[@ls];
               ls ← ns;
               RESUME[ns];
               };
            UNWIND = >
               {
               IF ls # NIL THEN z.FREE[@ls];
               };
            };
```

```
xc ← XString.Lop[text];
IF xc = XChar.not THEN
    {
    IF state = entry THEN
        EXIT
    ELSE
        SIGNAL ParseError[err: syntaxError, start: start, pos: pos];
    };

SELECT state FROM
    entry => 
        {
        IF xc = left THEN
            state ← beginMeta
        ELSE
            {
            c ← LOOPHOLE[XChar.Code[xc], CHARACTER];  --/* only Charset 0 */
            String.AppendChar[s: ls, c: c];
            state ← entry;
            };
        pos ← pos + 1;
        };
    beginMeta => 
        {
        start ← pos;
        c ← LOOPHOLE[XChar.Code[xc], CHARACTER];  --/* only Charset 0 */
        SELECT c FROM
            IN ['0..'3] => 
                {
                state ← doOctal;
                octals ← 1;
                octalValue ← c - '0;
                };
            'C, 'F, 'L, 'N, 'T, '< => 
                {
                state ← doAbbrev;
                XString.ClearWriter[buf];  --/* collect abbreviation here */
                XString.AppendChar[to: buf, c: xc];
                abbrs ← 1;
                };
            ENDCASE => 
              SIGNAL ParseError[err: invalidMeta, start: start, pos: pos];
        pos ← pos + 1;
        };
    doOctal => 
        {
        c ← LOOPHOLE[XChar.Code[xc], CHARACTER];  --/* only Charset 0 */
        IF xc = right THEN
            {
            IF start = pos THEN
                SIGNAL ParseError[err: invalidMeta, start: start, pos: pos + 1];
            IF octals < maxOctals OR octalValue > 377B THEN
                SIGNAL ParseError[err: invalidOctal, start: start, pos: pos];
            c ← LOOPHOLE[octalValue, CHARACTER];
            String.AppendChar[s: ls, c: c];
            state ← entry;
            }
        ELSE IF octals >= maxOctals THEN
            SIGNAL ParseError[err: invalidOctal, start: start, pos: pos]
        ELSE IF NOT c IN ['0..'7] THEN
            SIGNAL ParseError[err: invalidOctal, start: start, pos: pos]
        ELSE
            {
            octalValue ← (octalValue * 8) + (c - '0);
            octals ← octals + 1;
            state ← doOctal;
            };
        pos ← pos + 1;
        };
    doAbbrev => 
        {
        IF xc = right THEN
            {
            tmp: XString.Reader ← XString.ReaderFromWriter[buf];

            IF start = pos THEN
                SIGNAL ParseError[err: invalidMeta, start: start, pos: pos + 1];
            IF abbrs > maxAbbr THEN
                SIGNAL ParseError[err: unknownAbbr, start: start, pos: pos];
            SELECT TRUE FROM
                XString.Equal[r1: tmp, r2: @cr] => 
                    String.AppendChar[s: ls, c: Ascii.CR];
                XString.Equal[r1: tmp, r2: @lf] => 
                    String.AppendChar[s: ls, c: Ascii.LF];
                XString.Equal[r1: tmp, r2: @nl] => 
                    {
                    String.AppendChar[s: ls, c: Ascii.CR];
                    String.AppendChar[s: ls, c: Ascii.LF];
                    };
                XString.Equal[r1: tmp, r2: @tab] => 
                    String.AppendChar[s: ls, c: Ascii.TAB];
                XString.Equal[r1: tmp, r2: @ff] => 
                    String.AppendChar[s: ls, c: Ascii.FF];
                abbrs = 1 AND c = '< => 
                    String.AppendChar[s: ls, c: '<];
                ENDCASE => 
                    SIGNAL ParseError[err: unknownAbbr, start: start, pos: pos];
```

```
                    state ← entry;
                    }
                ELSE
                    {
                    XString.AppendChar[to: buf, c: xc];
                    abbrs ← abbrs + 1;
                    state ← doAbbrev;
                    };
                pos ← pos + 1;
                };
            ENDCASE;
    ENDLOOP;
};


END...

LOG
16-Mar-87 14:06:16 - Caro - Created
26-Jun-87 11:28:54 - Caro - Added test for MessageKey.FIRST to ParseItem
29-Jun-87 11:33:00 - Caro - Added validation to ParseItem
```

```
-- File: CvXlitToVPImpl.mesa
-- Trow 7-Sep-89 16:59:04

-- Last Revised by: Shinsato                    12-Feb-88 13:00:11
-- Owner: Workstation Applications - Foreign Conversion Team

-- Copyright (c) 1987, 1988, 1989 by Xerox Corporation.  All rights reserved.

DIRECTORY
  Ascii
    USING [CR, FF, LF, NUL, SP, TAB],
  BackgroundProcess
    USING [ResetUserAbort, UserAbort],
  BWSZone
    USING [Permanent],
  Converter
    USING [ConvertProc, CvData, DependentOptionProc, GetPOption, PostMessage],
  ConverterMsg,
  CvXlit,
  DocInterchangeDefs
    USING [AppendNewParagraph, AppendPageBreak, AppendText, CheckAbortProc,
    Doc, Error, FinishCreation, FinishCreationStatus,
    PaginateOption, StartCreation, StartCreationStatus],
  DocInterchangePropsDefs
    USING [Family, FontPropsRecord, GetFontPropsDefaults, GetPagePropsDefaults,
    GetParaPropsDefaults, PagePropsRecord, ParaPropsRecord, modern, classic],
  Environment
    USING [Block, Byte, bytesPerPage, wordsPerPage],
  NSFile
    USING [Close, Error, GetReference, Handle, Logoff,
    nullHandle, OpenByReference, Reference, Session],
  NSFileStream
    USING [Create, Handle],
  Space
    USING [ScratchMap, Unmap],
  Stream
    USING [CompletionCode, Delete, GetBlock],
  TIP
    USING [ResetUserAbort, UserAbort],
  XCharSet0
    USING [Make],
  XString
    USING [AppendChar, ByteLength,
    Character, CharacterLength, ClearWriter, FreeWriterBytes,
    InvalidEncoding, NewWriterBody, Reader, ReaderBody, ReaderFromWriter,
    Writer, WriterBody, WriterInfo];

<<
--================================================================================
-- OVERVIEW:

Xlit to VP conversion.
--================================================================================
>>

CvXlitToVPImpl: PROGRAM
IMPORTS
  BackgroundProcess, BWSZone, Converter, ConverterMsg, CvXlit,
  DocInterchangeDefs, DocInterchangePropsDefs,
  NSFile, NSFileStream, Space, Stream, TIP,
  XCharSet0, XString
EXPORTS
  CvXlit =
BEGIN

--====================
-- CONSTANTS
--====================

maxPara: CARDINAL = 8 * 1024;
bufPages: CARDINAL = (maxPara + Environment.bytesPerPage - 1) / Environment.bytesPerPage;
paraLen: CARDINAL = maxPara/4;

words: CARDINAL = SIZE[CtoVPCharMap];

stopsAt: CARDINAL = 5;  --/* tab stops every five characters */
tabStopCount: CARDINAL = (132/stopsAt)+1;  --/* 132 columns max */

aHyphen: CHARACTER = 055C;

xNewLine: XString.Character = XCharSet0.Make[newLine];

--====================
-- TYPES
--====================

AVData: TYPE = LONG POINTER TO AVDataObj;
AVDataObj: TYPE = RECORD [
    source: NSFile.Handle,
    input: NSFileStream.Handle,  --/* created from source */
    cvData: Converter.CvData,
    session: NSFile.Session,
    src: CvXlit.Common,  --/* common data distinguished by owning formwindow */
    dst: CvXlit.Common,
    background: BOOLEAN,
    fontProps: DocInterchangePropsDefs.FontPropsRecord,
    paraProps: DocInterchangePropsDefs.ParaPropsRecord,
```

```
    pageProps: DocInterchangePropsDefs.PagePropsRecord,
    doc: DocInterchangeDefs.Doc,
    blk: Environment.Block,  --/* primary input buffer */
    state: AVState,
    z: UNCOUNTED ZONE];

--/* the various states of the StateMachine */
AVState: TYPE =
    {
    entry,
    append,
    ignoreTrailing,
    maxExceeded,
    endPara
    };


CtoVPCharMap: TYPE = ARRAY CHARACTER OF XString.Character;

--=====================
-- GLOBALS
--=====================

Global: TYPE = RECORD [
    isomap: LONG POINTER TO CtoVPCharMap,
    modmap: LONG POINTER TO CtoVPCharMap,
    pz: UNCOUNTED ZONE];

g: Global;

--=====================
-- PUBLIC PROCEDURES
--=====================

AsciiToVP: PUBLIC Converter.ConvertProc = {
<< = PROCEDURE [source: NSFile.Handle, cvData: Converter.CvData, session: NSFile.Session, srcInstance: LONG POINTER ← NIL, dstInstance:
LONG POINTER ← NIL, background: BOOLEAN ← FALSE] RETURNS [dest: NSFile.Handle ← LOOPHOLE[0]];
>>
  ENABLE CvXlit.Problem, NSFile.Error, XString.InvalidEncoding => 
      {
      msgRb: XString.ReaderBody ← CvXlit.GetMessage[fatalError];

      Post[msgRb, cvData];
      CONTINUE;
      };

    IF source = NSFile.nullHandle THEN RETURN;
    dest ← AtoV[source, cvData, session, srcInstance, dstInstance, background];
};

<<
--=====================================================================================================
Both DependentOptionProcs create instance data with CreateCommon.  The data is distinguished by the owner variable.  The CommonObj within
CvXlit.CommonData is the data structure written to the client file stored as the icon properties.  Only those fields pertaining to the
owner are used.
--=====================================================================================================
>>

AsciiToVPSrcOps: PUBLIC Converter.DependentOptionProc = {
<< = PROCEDURE [options: BOOLEAN ← TRUE, cvData: Converter.CvData, which: Converter.FormatToUse, srcFormat: XString.Reader, destFormat:
XString.Reader, window: Window.Handle, oldInstance: LONG POINTER ← NIL] RETURNS [menuItemProc: Converter.MenuItemProc, destroy:
Converter.DestroyProc, instance: LONG POINTER];
>>
    owner: CvXlit.Owners ← AtoVsrc;

    menuItemProc ← CvXlit.CommonMenu;
    destroy ← CvXlit.DestroyCommon;
    IF oldInstance = NIL THEN
        instance ← CvXlit.CreateCommon[cvData, options, window, owner ! NSFile.Error, CvXlit.Problem => {owner ← backstop; instance ←
        NIL; CONTINUE}]
    ELSE
        {
        my: CvXlit.Common ← oldInstance;
        my.window ← window;  --/* AR 13535: update window handle */
        instance ← my;
        };

    --/* make formwindow */
    CvXlit.CreateFW[instance, window, owner];
};


AsciiToVPDstOps: PUBLIC Converter.DependentOptionProc = {
<< = PROCEDURE [options: BOOLEAN ← TRUE, cvData: Converter.CvData, which: Converter.FormatToUse, srcFormat: XString.Reader, destFormat:
XString.Reader, window: Window.Handle, oldInstance: LONG POINTER ← NIL] RETURNS [menuItemProc: Converter.MenuItemProc, destroy:
Converter.DestroyProc, instance: LONG POINTER];
>>
    owner: CvXlit.Owners ← AtoVdst;
    my: CvXlit.Common;

    menuItemProc ← CvXlit.CommonMenu;
    destroy ← CvXlit.DestroyCommon;
    IF oldInstance = NIL THEN
        {
        instance ← CvXlit.CreateCommon[cvData, options, window, owner !
            NSFile.Error, CvXlit.Problem => {owner ← backstop; instance ← NIL; CONTINUE}];
        }
```

```
    ELSE
        {
        my ← oldInstance;
        my.window ← window;  --/* AR 13535: update window handle */
        instance ← my;
        };

    --/* make formwindow */
    CvXlit.CreateFW[instance, window, owner];
};


GetAVTable: PUBLIC PROC RETURNS [avTable: LONG POINTER TO CtoVPCharMap] = {
    RETURN[g.modmap];
};


--=====================
-- PROCEDURES
--=====================

AtoV: Converter.ConvertProc = {
    aborted: BOOLEAN ← FALSE;
    start: DocInterchangeDefs.StartCreationStatus ← lastAvailable;
    finish: DocInterchangeDefs.FinishCreationStatus ← lastAvailable;
    avData: AVDataObj;
    pOption: DocInterchangeDefs.PaginateOption;
    docSession: NSFile.Session;
    dst,
    src: CvXlit.Common ← NIL;
    lineHtInPoints: CARDINAL ← 24;  -- lots of space for Xlit
    --/* local proc */
    POption: PROCEDURE RETURNS [DocInterchangeDefs.PaginateOption] = INLINE
        {
        SELECT Converter.GetPOption[] FROM
          compress => RETURN[compress];
          simple => RETURN[simple];
          none => RETURN[none];
        ENDCASE => ERROR;
        };

    --/* begin code */
    --/* initialize instance data */
    IF dstInstance = NIL THEN  --/* ASSERT: srcInstance also NIL */
        {
        ENABLE NSFile.Error, CvXlit.Problem =>
            {
            msgRb: XString.ReaderBody ← CvXlit.GetMessage[extraErr0];  --" Unrecoverable Xlit conversion error: damaged converter icon. "
            Converter.PostMessage[
              msg: @msgRb,
              cvData: cvData,
              cr: FALSE,
              clear: FALSE];
            IF src # NIL THEN CvXlit.DestroyCommon[src];
            GOTO terminate;
            };
        key: CvXlit.MessageKey ← CvXlit.MessageKey.FIRST;

        --/* assume both are NIL */
        src ← CvXlit.CreateCommon[cvData, FALSE, NIL, AtoVsrc];
        dst ← CvXlit.CreateCommon[cvData, FALSE, NIL, AtoVdst];

        src.text[paraEndsWith] ← CvXlit.ParseItem[
          my: src,
          r: @src.textRb[paraEndsWith],
          item: key].ls;

        dst.text[atovReplaceUnknown] ← CvXlit.ParseItem[
          my: dst,
          r: @dst.textRb[atovReplaceUnknown],
          item: key].ls;
        EXITS terminate => RETURN;
        }
    ELSE
        {
        src ← srcInstance;
        dst ← dstInstance;
        };

    avData ← [
        source: source,
        input: [NIL],
        cvData: cvData,
        session: session,
        src: src,
        dst: dst,
        background: background,
        fontProps: TRASH,
        paraProps: TRASH,
        pageProps: TRASH,
        doc: TRASH,
        blk: [Space.ScratchMap[count: bufPages], 0, maxPara],
        state: entry,
        z: dst.z];

    BEGIN
      ENABLE
```

```
        {
        DocInterchangeDefs.Error => GOTO err;
        UNWIND =>
            {
            avData.blk.blockPointer ← Space.Unmap[pointer: avData.blk.blockPointer];
            IF srcInstance = NIL THEN CvXlit.DestroyCommon[src];
            IF dstInstance = NIL THEN CvXlit.DestroyCommon[dst];
            src ← dst ← NIL;
            };
        };

    --/* open stream on source */
    avData.input ← NSFileStream.Create[
      file: avData.source,
      closeOnDelete: FALSE,
      session: avData.session ! NSFile.Error => {avData.input ← [NIL]; GOTO err}];

    --/* initialize */
    pOption ← POption[];
    DocInterchangePropsDefs.GetFontPropsDefaults[@avData.fontProps];
    DocInterchangePropsDefs.GetParaPropsDefaults[@avData.paraProps];
    DocInterchangePropsDefs.GetPagePropsDefaults[@avData.pageProps];

    --/* apply initial parms */
    SELECT avData.dst.f.font FROM
        CvXlit.modern => avData.fontProps.fontDesc.family ← DocInterchangePropsDefs.modern;
        CvXlit.classic => avData.fontProps.fontDesc.family ← DocInterchangePropsDefs.classic;
        ENDCASE;

    SELECT avData.dst.f.fontSize FROM
        CvXlit.twelve =>
            {
            avData.fontProps.fontDesc.pointSize ← 12;
            avData.paraProps.basicProps.lineHeight ← lineHtInPoints ← 32;
            avData.paraProps.basicProps.defaultTabStopSpacing ← (stopsAt * 12);
            };
        CvXlit.eightteen =>
            {
            avData.fontProps.fontDesc.pointSize ← 18;
            avData.paraProps.basicProps.lineHeight ← lineHtInPoints ← 44;
            avData.paraProps.basicProps.defaultTabStopSpacing ← (stopsAt * 18);
            };
        CvXlit.twentyFour =>
            {
            avData.fontProps.fontDesc.pointSize ← 24;
            avData.paraProps.basicProps.lineHeight ← lineHtInPoints ← 64;
            avData.paraProps.basicProps.defaultTabStopSpacing ← (stopsAt * 24);
            };
        ENDCASE;

    --/* set paragraph properties for Xlit */
    avData.paraProps.basicProps.streakSuccession ← rightToLeft;
    avData.paraProps.basicProps.paraAlignment ← right;

    --/* set "AFTER" paragraph leading by counting CRs in paraEndsWith string */
    BEGIN
    lcount: CARDINAL ← 0;
    eop: LONG STRING ← avData.src.text[paraEndsWith];
    IF eop # NIL THEN
      FOR i: CARDINAL IN [0..eop.length) DO
        IF eop[i] = Ascii.CR THEN lcount ← lcount + 1;
      ENDLOOP;
    --/* lcount = 0 => default */
    --/* lcount = 1 => single spacing */
    --/* lcount = 2 => 1.5, etc. */
    IF lcount > 1 THEN
        avData.paraProps.basicProps.postLeading ← lineHtInPoints * (lcount - 1) / 2;
    END;

    --/* StartCreation checks process priority to determine forkedness */
    [doc: avData.doc, status: start] ← DocInterchangeDefs.StartCreation[
        paginateOption: pOption,
        initialFontProps: @avData.fontProps,
        initialParaProps: @avData.paraProps,
        initialPageProps: @avData.pageProps ! NSFile.Error => {
          IF error = [space[mediumFull]] THEN
            start ← notEnoughDiskSpace
          ELSE
            start ← lastAvailable;
          CONTINUE}];

    SELECT start FROM
        ok => NULL;
        notEnoughDiskSpace =>
            {
            Post[ConverterMsg.Get[ConverterMsg.koutOfSpace], avData.cvData];
            GOTO err;
            };
        ENDCASE =>
            {
            Post[ConverterMsg.Get[ConverterMsg.kunknownProblem], avData.cvData];
            GOTO err;
            };

    --/* enter state graph */
    BEGIN
      ENABLE ABORTED => {aborted ← TRUE; CONTINUE};
```

```
        StateMachine[@avData];
    END;

    --/* paginating */
    IF pOption # none THEN
        {
        mrb: XString.ReaderBody ← CvXlit.GetMessage[paginating];
        Converter.PostMessage[
          msg: @mrb,
          cvData: cvData,
          cr: FALSE,
          clear: FALSE];
        };

    --/* user may have partial doc after an abort, so allow paginate/finish */
    --/* reset abort tests. User must abort paginate separately. */
    IF aborted THEN
        {
        IF avData.background THEN
            BackgroundProcess.ResetUserAbort[]
        ELSE
            TIP.ResetUserAbort[NIL];
        };

    --/* paginate and finish */
    [docFile: dest, session: docSession, status: finish] ← DocInterchangeDefs.FinishCreation[
      docPtr: @avData.doc,
      checkAbortProc: UserAbortsPaginate,
      checkAbortClientData: @avData];

    IF finish = aborted THEN
        {
        aborted ← TRUE;
        Post[ConverterMsg.Get[ConverterMsg.kuserAbort], cvData];
        };

    --/* re-open dest in session */
    IF dest # NSFile.nullHandle THEN
        {
          ENABLE NSFile.Error => {
            NSFile.Close[dest, docSession ! NSFile.Error => CONTINUE];
            dest ← NSFile.nullHandle;
            CONTINUE;
            };
        tmpRef: NSFile.Reference;
        tmp: NSFile.Handle ← dest;

        tmpRef ← NSFile.GetReference[file: dest, session: docSession];
        dest ← NSFile.OpenByReference[reference: tmpRef, session: avData.session];
        NSFile.Close[tmp, docSession];
        --/* if this process is clientBackground, docSession must be logged off */
        IF background THEN NSFile.Logoff[docSession ! NSFile.Error => CONTINUE];
        };

    EXITS err => NULL;
    END;
    IF avData.input # NIL THEN Stream.Delete[avData.input];
    IF avData.blk.blockPointer # NIL THEN
        avData.blk.blockPointer ← Space.Unmap[avData.blk.blockPointer];
    --/* destroy instance data if created by this proc call */
    IF srcInstance = NIL AND src # NIL THEN CvXlit.DestroyCommon[src];
    IF dstInstance = NIL AND dst # NIL THEN CvXlit.DestroyCommon[dst];
    IF finish # ok OR aborted THEN
        Post[ConverterMsg.Get[ConverterMsg.kdataSkipped], cvData];
};

CheckAbort: PROC [background: BOOLEAN] RETURNS [yes: BOOLEAN] = INLINE {
    yes ← (background AND BackgroundProcess.UserAbort[]) OR
      (NOT background AND TIP.UserAbort[NIL]);
};

FlushText: PROC [av: AVData, para: XString.Writer] = {
    r: XString.Reader ← XString.ReaderFromWriter[para];

    IF CheckAbort[av.background] THEN ERROR ABORTED;
    IF XString.ByteLength[r] > 0 THEN
        {
        DocInterchangeDefs.AppendText[
          to: [doc[av.doc]],
          text: r,
          textEndContext: XString.WriterInfo[para].endContext,
          fontProps: @av.fontProps];
        XString.ClearWriter[para];
        };
};

Post: PROC [msgRb: XString.ReaderBody, cvData: Converter.CvData] = {
    Converter.PostMessage[
      msg: @msgRb,
      cvData: cvData,
      cr: TRUE,
      clear: FALSE];
};
```

```
<<
--=============================================================================================
StateMachine
This procedure implements a state graph, which is depicted in auxiliary documentation. The state machine handles the input data
character by character, although the i/o is optimized using block buffers. Note that the XString.Writer "para" is the output buffer
that gets appended to the document every time text is flushed (see FlushText). Hereafter are described, briefly, the states, the entry
conditions, exit conditions, and special circumstances:

- entry
The state machine is always entered here. The entry conditions are that the index "n" references the next character to be handled. The
next state is determined by the value of the character "c". The mode "ignore" determines whether white space is treated as standard
text, or as should be handled by the special ignoreTrailing state. If the character "c" matches the first character of the end of
paragraph string "eop0", then the next state is endPara. Otherwise, the next state is "append". Note that the variable "nextState"
does NOT refer to the state executed after entry, but rather the state that the next state RETURNS TO. Although this violates strict
state machine implementation algorithms, it saves logic.

- append
The state is entered with the character "c", and a valid nextState. It translates the character "c" to a VP character, and appends it to
the output buffer "para". Certain special cases are handled. The exit condition is a valid nextState, which becomes "state".

- ignoreTrailing
The purpose of this state is to implement deletion of white space that precedes an end of line sequence, if the user so desires. The
state is entered either from entry with "c" being whitespace, or from ignoreTrailing, with "n" indicating the next character to handle.
Variables are initialized to indicate the beginning of whitespace characters. The state is exited if eop0 is found, or a nonwhitespace
character is found before the end of line.

- maxExceeded
This state handles an overflow exception. It is entered if "para" is about to exceeded its limits. A new paragraph is forced if this
state is entered. It returns to entry.

- endPara
This state tries to determine if the end of a paragraph has been found. It is entered if the character "c" matched eop0, or (from
endPara itself) if the input text continues to match the string "s". If a paragraph ending is found, the paragraph is flushed. The
state returns to entry either if there is a complete match, or of there is a mismatch. Several special cases are handled.

The state machine loops until input is exhausted.
--=============================================================================================
>>


StateMachine: PROC [av: AVData] = {
    lastBlock: BOOLEAN ← FALSE;
    flushed: BOOLEAN ← FALSE; --/* controls appending text to doc */
    ignore: BOOLEAN ← av.dst.f.ignoreTrailing.value,
    eop: CARDINAL ← 0; --/* index into paraEndsWith string */
    para: XString.WriterBody ← XString.NewWriterBody[maxLength: paraLen, z: av.z];
    state: AVState ← entry;
    blankCount: CARDINAL ← 0; --/* count of "white" characters in buffer */
    blkCount: CARDINAL ← 0; --/* number of blocks read */
    lastBlkCount: CARDINAL; --/* for saving "blkCount" */
    nextState: AVState; --/* the state a state goes back to */
    getNextBlock: BOOLEAN;
    bytes: CARDINAL;
    why: Stream.CompletionCode;
    eop0: CHARACTER; --/* first character of end-of-paragraph text */
    unknown: LONG STRING; --/* copy of user defined replacement text */
    blanksStart: CARDINAL; --/* index into buffer for beginning of blanks */
    imap: LONG POINTER TO CtoVPCharMap;
    amap: LONG POINTER TO CtoVPCharMap;
    blk: LONG POINTER TO PACKED ARRAY INTEGER[0..0) OF Environment.Byte;
    n: CARDINAL; --/* current character in blk */
    last,
    c: CHARACTER;
    convertXlit: BOOLEAN ← TRUE;

    --/* initialize */
    imap ← g.isomap;
    amap ← g.modmap;

    --/* para is a buffer of VP characters that gets appended to the doc */
    XString.ClearWriter[@para];
    eop0 ← IF av.src.text[paraEndsWith] # NIL THEN
        av.src.text[paraEndsWith][0]
    ELSE
        Ascii.NUL;
    last ← Ascii.NUL;
    unknown ← av.dst.text[atovReplaceUnknown];
    IF unknown = NIL THEN
        {
        --/* so we don't have to test for NIL again */
        unknown ← "?"L;
        unknown.length ← 0;
        };
    --/* make sure getNextBlock is TRUE first time */
    n ← av.blk.stopIndexPlusOne;
    blk ← av.blk.blockPointer;

    --/* enter state graph */
    DO
        getNextBlock ← n >= av.blk.stopIndexPlusOne;

        IF getNextBlock THEN
            {
```

```
<<
--=========================================================================================
StateMachine
This procedure implements a state graph, which is depicted in auxiliary documentation. The state machine handles the input data
character by character, although the i/o is optimized using block buffers. Note that the XString.Writer "para" is the output buffer
that gets appended to the document every time text is flushed (see FlushText). Hereafter are described, briefly, the states, the entry
conditions, exit conditions, and special circumstances:

-- entry
The state machine is always entered here. The entry conditions are that the index "n" references the next character to be handled. The
next state is determined by the value of the character "c". The mode "ignore" determines whether white space is treated as standard
text, or as should be handled by the special ignoreTrailing state. If the character "c" matches the first character of the end of
paragraph string "eop0", then the next state is endPara. Otherwise, the next state is "append". Note that the variable "nextState"
does NOT refer to the state executed after entry, but rather the state that the next state RETURNS TO. Although this violates strict
state machine implementation algorithms, it saves logic.

-- append
The state is entered with the character "c", and a valid nextState. It translates the character "c" to a VP character, and appends it to
the output buffer "para". Certain special cases are handled. The exit condition is a valid nextState, which becomes "state".

-- ignoreTrailing
The purpose of this state is to implement deletion of white space that precedes an end of line sequence, if the user so desires. The
state is entered either from entry with "c" being whitespace, or from ignoreTrailing, with "n" indicating the next character to handle.
Variables are initialized to indicate the beginning of whitespace characters. The state is exited if eop0 is found, or a nonwhitespace
character is found before the end of line.

-- maxExceeded
This state handles an overflow exception. It is entered if "para" is about to exceeded its limits. A new paragraph is forced if this
state is entered. It returns to entry.

-- endPara
This state tries to determine if the end of a paragraph has been found. It is entered if the character "c" matched eop0, or (from
endPara itself) if the input text continues to match the string "s". If a paragraph ending is found, the paragraph is flushed. The
state returns to entry either if there is a complete match, or of there is a mismatch. Several special cases are handled.

The state machine loops until input is exhausted.
--=========================================================================================
>>


StateMachine: PROC [av: AVData] = {
    lastBlock: BOOLEAN ← FALSE;
    flushed: BOOLEAN ← FALSE; --/* controls appending text to doc */
    ignore: BOOLEAN ← av.dst.f.ignoreTrailing.value;
    eop: CARDINAL ← 0; --/* index into paraEndsWith string */
    para: XString.WriterBody ← XString.NewWriterBody[maxLength: paraLen, z: av.z];
    state: AVState ← entry;
    blankCount: CARDINAL ← 0; --/* count of "white" characters in buffer */
    blkCount: CARDINAL ← 0; --/* number of blocks read */
    lastBlkCount: CARDINAL; --/* for saving "blkCount" */
    nextState: AVState; --/* the state a state goes back to */
    getNextBlock: BOOLEAN;
    bytes: CARDINAL;
    why: Stream.CompletionCode;
    eop0: CHARACTER; --/* first character of end-of-paragraph text */
    unknown: LONG STRING; --/* copy of user defined replacement text */
    blanksStart: CARDINAL; --/* index into buffer for beginning of blanks */
    imap: LONG POINTER TO CtoVPCharMap;
    amap: LONG POINTER TO CtoVPCharMap;
    blk: LONG POINTER TO PACKED ARRAY INTEGER[0..0) OF Environment.Byte;
    n: CARDINAL; --/* current character in blk */
    last,
    c: CHARACTER;
    convertXlit: BOOLEAN ← TRUE;

    --/* initialize */
    imap ← g.isomap;
    amap ← g.modmap;

    --/* para is a buffer of VP characters that gets appended to the doc */
    XString.ClearWriter[@para];
    eop0 ← IF av.src.text[paraEndsWith] # NIL THEN
        av.src.text[paraEndsWith][0]
    ELSE
        Ascii.NUL;
    last ← Ascii.NUL;
    unknown ← av.dst.text[atovReplaceUnknown];
    IF unknown = NIL THEN
        {
        --/* so we don't have to test for NIL again */
        unknown ← "?"L;
        unknown.length ← 0;
        };
    --/* make sure getNextBlock is TRUE first time */
    n ← av.blk.stopIndexPlusOne;
    blk ← av.blk.blockPointer;

    --/* enter state graph */
    DO
        getNextBlock ← n >= av.blk.stopIndexPlusOne;

        IF getNextBlock THEN
            {
```

```
        IF lastBlock THEN
            {
            --/* might have one last character pending */
            IF state = append THEN
                {
                nextState ← entry;
                GOTO oneLastLoop;
                };
            FlushText[av, @para];
            EXIT;  --/* state graph */
            };
        IF CheckAbort[av.background] THEN ERROR ABORTED;
        av.blk.stopIndexPlusOne ← maxPara;
        [bytesTransferred: bytes, why: why] ← Stream.GetBlock[
          sH: av.input,
          block: av.blk];
        lastBlock ← why # normal;
        av.blk.stopIndexPlusOne ← bytes;
        blk ← av.blk.blockPointer;
        n ← 0;
        --/* guard against blkCount overflow */
        blkCount ← IF blkCount = CARDINAL.LAST THEN 0 ELSE blkCount + 1;
        EXITS oneLastLoop => NULL;
        };

    SELECT state FROM
        entry =>
            {
            --/* get next character */
            c ← LOOPHOLE[blk[n], CHAR];
            --/* set up next state */
            SELECT c FROM
                Ascii.SP, Ascii.TAB => IF ignore THEN
                    {
                    state ← ignoreTrailing;
                    blanksStart ← n;
                    blankCount ← 0;
                    lastBlkCount ← blkCount;
                    }
                  ELSE
                    {
                    state ← append;
                    nextState ← entry;
                    n ← n + 1;
                    };
                eop0 =>
                    {
                    state ← endPara;
                    };
                ENDCASE =>
                    {
                    state ← append;
                    nextState ← entry;
                    n ← n + 1;
                    };
            };
        append =>
            {
            --/* ASSERT: order of select arms is critically important */
            SELECT c FROM
                '# =>           -- 74C, toggle convert switch
                    {
                    convertXlit ← ~convertXlit;
                    XString.AppendChar[to: @para, c: amap['#], extra: paraLen];
                    state ← nextState;
                    };
                '* =>           -- 52C, word separator
                    {
                    XString.AppendChar[to: @para, c: imap[Ascii.SP], extra: paraLen];
                    state ← nextState;
                    };
                '( =>
                    {
                    --/* reverse parentheses in right to left paragraphs */
                    IF av.paraProps.basicProps.streakSuccession = rightToLeft THEN
                      XString.AppendChar[to: @para, c: amap[')], extra: paraLen]
                    ELSE
                      XString.AppendChar[to: @para, c: imap[c], extra: paraLen];
                    state ← nextState;
                    };
                ') =>
                    {
                    --/* reverse parentheses in right to left paragraphs */
                    IF av.paraProps.basicProps.streakSuccession = rightToLeft THEN
                      XString.AppendChar[to: @para, c: amap['(], extra: paraLen]
                    ELSE
                      XString.AppendChar[to: @para, c: imap[c], extra: paraLen];
                    state ← nextState;
                    };
                '[ =>
                    {
                    --/* reverse brackets in right to left paragraphs */
                    IF av.paraProps.basicProps.streakSuccession = rightToLeft THEN
                      XString.AppendChar[to: @para, c: amap[']], extra: paraLen]
                    ELSE
                      XString.AppendChar[to: @para, c: imap[c], extra: paraLen];
                    state ← nextState;
```

```
        };
    '] =>
        {
        --/* reverse brackets in right to left paragraphs */
        IF av.paraProps.basicProps.streakSuccession = rightToLeft THEN
           XString.AppendChar[to: @para, c: amap['[], extra: paraLen]
        ELSE
           XString.AppendChar[to: @para, c: imap[c], extra: paraLen];
        state ← nextState;
        };
    '{ =>
        {
        --/* reverse braces in right to left paragraphs */
        IF av.paraProps.basicProps.streakSuccession = rightToLeft THEN
           XString.AppendChar[to: @para, c: amap['}], extra: paraLen]
        ELSE
           XString.AppendChar[to: @para, c: imap[c], extra: paraLen];
        state ← nextState;
        };
    '} =>
        {
        --/* reverse braces in right to left paragraphs */
        IF av.paraProps.basicProps.streakSuccession = rightToLeft THEN
           XString.AppendChar[to: @para, c: amap['{], extra: paraLen]
        ELSE
           XString.AppendChar[to: @para, c: imap[c], extra: paraLen];
        state ← nextState;
        };
    'P, 'p =>
        {
        IF convertXlit THEN
           FOR i: CARDINAL IN [0..unknown.length) DO
                XString.AppendChar[to: @para, c: imap[unknown[i]], extra: paraLen];
           ENDLOOP
        ELSE
           XString.AppendChar[to: @para, c: imap[c], extra: paraLen];
        state ← nextState;
        };
    IN [40C..176C], 242C, 253C, 273C, 277C => 
        {
        IF convertXlit THEN
           XString.AppendChar[to: @para, c: amap[c], extra: paraLen]
        ELSE
           XString.AppendChar[to: @para, c: imap[c], extra: paraLen];
        state ← nextState;
        };
    Ascii.CR =>
        {
        IF CheckAbort[av.background] THEN ERROR ABORTED;
        IF nextState = entry THEN
             {
             --/* smart white space */
             SELECT last FROM
                  Ascii.SP,
                  Ascii.TAB,
                  Ascii.CR,
                  Ascii.LF,
                  aHyphen => NULL; --/* just drop CR */
                  ENDCASE =>
                       {
                       XString.AppendChar[
                          to: @para,
                          c: imap[Ascii.SP],
                          extra: paraLen];
                       };
             };
        --/* CR is skipped if we came from endPara */
        state ← nextState;
        };
    Ascii.LF =>
        {
        IF last # Ascii.CR AND nextState # endPara THEN
             {
             --/* append newline */
             XString.AppendChar[
                to: @para,
                c: xNewLine,
                extra: paraLen];
             };
        --/* LF is skipped if we came from endPara */
        --/* or if last = CR */
        state ← nextState;
        };
    Ascii.TAB =>
        {
        --/* tab */
        XString.AppendChar[to: @para, c: imap[c], extra: paraLen];
        state ← nextState;
        };
    Ascii.FF =>
        {
        --/* flush page */
        FlushText[av, @para];
        DocInterchangeDefs.AppendPageBreak[
           to: av.doc,
           fontProps: @av.fontProps];
        state ← nextState;
```

```
                    };
                Ascii.NUL => 
                    {
                    --/* skip */
                    state ← nextState;
                    };
                ENDCASE => 
                    {
                    --/* exceptions */
                    FOR i: CARDINAL IN [0..unknown.length) DO
                        XString.AppendChar[to: @para, c: imap[unknown[i]], extra: paraLen];
                    ENDLOOP;
                    state ← nextState;
                    };
            last ← c;
            <<
            * XString.CharacterLength is an expensive operation.
            * We make the observation that
            * ByteLength >= CharacterLength ALWAYS. Therefore
            * use faster ByteLength to determine if CharacterLength should
            * be called
            >>
            IF XString.ByteLength[XString.ReaderFromWriter[@para]] > maxPara THEN
                {
                IF XString.CharacterLength[XString.ReaderFromWriter[@para]] > maxPara
                   AND nextState # endPara THEN
                    state ← maxExceeded;
                };
            };
        ignoreTrailing => 
            {
            --/* get next char if other than first entry */
            IF blanksStart # n THEN
                {
                last ← c;
                c ← LOOPHOLE[blk[n], CHAR];
                };

            SELECT c FROM
                Ascii.SP, Ascii.TAB => 
                    {
                    state ← ignoreTrailing;
                    n ← n + 1;
                    blankCount ← blankCount + 1;
                    };
                eop0 => 
                    {
                    --/* end found, so skip all trailing blanks */
                    state ← endPara;
                    };
                Ascii.CR => 
                    {
                    --/* NOTE: this arm must follow the eop0 arm */
                    --/* ASSERT: eop0 # Ascii.CR by order of execution */
                    --/* replace CR with space, and skip blanks */
                    XString.AppendChar[
                       to: @para,
                       c: imap[Ascii.SP],
                       extra: paraLen];
                    state ← entry;
                    blankCount ← 0;
                    n ← n + 1;
                    };
                ENDCASE => 
                    {
                    IF CheckAbort[av.background] THEN ERROR ABORTED;
                    --/* whoops!  Not eol, so append */
                    IF (lastBlkCount # blkCount) THEN
                        {
                        --/* blanks straddle blocks */
                        THROUGH [1..blankCount] DO
                            XString.AppendChar[
                               to: @para,
                               c: imap[Ascii.SP],
                               extra: paraLen];
                        ENDLOOP;
                        }
                    ELSE
                        FOR i: CARDINAL IN [blanksStart..n) DO
                            XString.AppendChar[
                               to: @para,
                               c: imap[LOOPHOLE[blk[i], CHAR]],
                               extra: paraLen];
                        ENDLOOP;
                    blankCount ← 0;
                    state ← entry;
                    };
            };
        maxExceeded => 
            {
            FlushText[av, @para];
            IF convertXlit THEN
               {
               --/* set paragraph properties for Xlit */
               av.paraProps.basicProps.streakSuccession ← rightToLeft;
               av.paraProps.basicProps.paraAlignment ← right;
               }
```

```
        ELSE
          {
          --/* set paragraph properties for English */
          av.paraProps.basicProps.streakSuccession ← leftToRight;
          av.paraProps.basicProps.paraAlignment ← left;
          };
        DocInterchangeDefs.AppendNewParagraph[
          to: [doc[av.doc]],
          paraProps: @av.paraProps,
          fontProps: @av.fontProps,
          nToAppend: 1];
        state ← entry;
        };
    endPara => {
        s: LONG STRING ← av.src.text[paraEndsWith];

        IF s = NIL THEN
            {
            state ← entry;
            nextState ← entry;
            n ← n + 1;
            flushed ← FALSE;
            GOTO restart;
            };

        IF eop # 0 THEN
            {
            last ← c;
            c ← LOOPHOLE[blk[n], CHAR];
            };

        --/* if we are at the end of s, then match! */
        IF eop >= s.length THEN
            {
            IF NOT flushed THEN
                --/* flush all text */
                FlushText[av, @para];

            IF convertXlit THEN
              {
              --/* set paragraph properties for Xlit */
              av.paraProps.basicProps.streakSuccession ← rightToLeft;
              av.paraProps.basicProps.paraAlignment ← right;
              }
            ELSE
              {
              --/* set paragraph properties for English */
              av.paraProps.basicProps.streakSuccession ← leftToRight;
              av.paraProps.basicProps.paraAlignment ← left;
              };
            DocInterchangeDefs.AppendNewParagraph[
              to: [doc[av.doc]],
              paraProps: @av.paraProps,
              fontProps: @av.fontProps,
              nToAppend: 1];

            IF flushed THEN
                --/* flush following text */
                FlushText[av, @para];

            eop ← 0;
            state ← entry;
            nextState ← entry;
            flushed ← FALSE;
            GOTO restart;
            };

        --/* c match with end-of-paragraph? */
        IF s[eop] = c THEN
            {
            eop ← eop + 1;
            n ← n + 1;
            }
        ELSE
            {
            --/* false alarm */
            IF ignore THEN
                --/* ouch, we interrupted ignoreTrailing */
                FOR j: CARDINAL IN [0..eop) DO
                    IF s[j] = Ascii.CR THEN GOTO oneCR;
                    IF s[j] # Ascii.SP OR s[j] # Ascii.TAB THEN
                      GOTO notWhite;
                REPEAT
                    oneCR => {
                        --/* replace CR with one blank */
                        XString.AppendChar[
                          to: @para,
                          c: imap[Ascii.SP],
                          extra: paraLen];
                        --/* other blanks ignored */
                        blankCount ← 0;
                        };
                    notWhite => {
                        --/* flush blankCount characters */
```

```
                    IF (lastBlkCount # blkCount) THEN
                        {
                        --/* blanks straddle blocks */
                        THROUGH [1..blankCount] DO
                            XString.AppendChar[
                              to: @para,
                              c: imap[Ascii.SP],
                              extra: paraLen];
                        ENDLOOP;
                        }
                    ELSE
                        FOR i: CARDINAL IN [blanksStart..blanksStart+blankCount) DO
                            XString.AppendChar[
                              to: @para,
                              c: imap[LOOPHOLE[blk[i], CHAR]],
                              extra: paraLen];
                        ENDLOOP;
                    blankCount ← 0;
                    };
                FINISHED =>
                    {
                    --/* include current chars in blankCount */
                    blankCount ← blankCount + (MAX[eop,1] - 1);
                    state ← ignoreTrailing;
                    };
            ENDLOOP;

        --/* set up for next state */
        IF (c = Ascii.SP OR c = Ascii.TAB)
          AND ignore
          AND state # ignoreTrailing THEN
            {
            state ← ignoreTrailing;
            blanksStart ← n;
            blankCount ← 0;
            lastBlkCount ← blkCount;
            }
        ELSE
            {
            state ← append;
            n ← n + 1;
            --/* account for any CRs */
            --/* IF last = CR, then kludge handled it */
            IF last # Ascii.CR THEN
                FOR j: CARDINAL IN [0..eop) DO
                    IF s[j] = Ascii.CR THEN GOTO foundCR;
                REPEAT
                    foundCR =>
                        {
                        --/* replace one or more CRs with one blank */
                        XString.AppendChar[
                          to: @para,
                          c: imap[Ascii.SP],
                          extra: paraLen];
                        };
                    FINISHED => NULL;
                ENDLOOP;
            };
        eop ← 0;
        nextState ← entry;
        flushed ← FALSE;
        GOTO restart;
        --/* end of false alarm */
        };

    --/* continue looking for eop */
    IF c = Ascii.CR THEN
        {
        --/* flush preceding text, clear buffer */
        FlushText[av, @para];
        flushed ← TRUE;
        };

    --/* translate character */
    state ← append;
    nextState ← endPara;

    --/* special look-ahead kludge to make naked CR's work */
    IF c = Ascii.CR
      AND NOT ignore
      AND eop < s.length
      AND n < av.blk.stopIndexPlusOne
      AND s[eop] # LOOPHOLE[blk[n], CHAR] THEN
        {
        --/* smart white space */
        SELECT last FROM
            Ascii.SP,
            Ascii.TAB,
            Ascii.CR,
            Ascii.LF,
            aHyphen => NULL; --/* just drop CR */
            ENDCASE =>
                {
                XString.AppendChar[
                  to: @para,
                  c: imap[Ascii.SP],
                  extra: paraLen];
```

```
                                        };
                                };
                            EXITS restart => NULL;
                            };
                        ENDCASE;
        ENDLOOP;

        --/* clean up */
        XString.FreeWriterBytes[@para];
};


UserAbortsPaginate: DocInterchangeDefs.CheckAbortProc = {
<< = PROCEDURE [clientData: LONG POINTER] RETURNS [abort: BOOL];
>>
        data: AVData = clientData;

        abort ← CheckAbort[data.background];
};


ZzInit: PROC = {
        pz: UNCOUNTED ZONE = BWSZone.Permanent[];

        --/* these Spaces should not be unmapped while this application is loaded */
        g ← [
          isomap: Space.ScratchMap[(words + Environment.wordsPerPage-1) / Environment.wordsPerPage],
          modmap: Space.ScratchMap[(words + Environment.wordsPerPage-1) / Environment.wordsPerPage],
          pz: pz];

        --/* initialize conversion maps */

        FOR c: CHARACTER IN CHARACTER DO
            temp: XString.Character ← XCharSet0.Make[LOOPHOLE[c]];
            g.isomap[c] ← temp;
            g.modmap[c] ← temp;
        ENDLOOP;
};


--/* main line code */

ZzInit[];


END...

LOG
16-Mar-87 14:06:16 - Caro - Created
26-Jun-87 11:21:47 - Caro - Added error catcher in ConvertProc over CreateCommon,
                                    Caught NSFile.Error in Logoff
29-Jun-87 13:13:00 - Caro - Added lineHtInPoints, AFTER setting
10-Jul-87 10:55:05 - Caro - Added aHyphen testing for smart spacing
19-Aug-87 11:01:32 - Caro - Fixed AR 13535 by updating oldInstance window
16-Sep-87 13:48:21 - Caro - isomap accentFirst from 241C to 301C
                                    isomap lowGraphFirst from 0 to 241C
                                    isomap lowGraphLast from 0 to 277C
                                    pcmap accentLast from 257C to 245C
                                    pcmap hiGraphFirst from 260C to 246C
12-Feb-88 12:58:57 - Shinsato - In AtoV, made sure eop # NIL before counting CR
                                        in eop.
```