```
-- ExternalLoadState.mesa
-- Edited by:
--            Sandman, April 6, 1978  2:02 PM
--            Barbara, May 15, 1978  10:43 AM

DIRECTORY
  AltoFileDefs: FROM "altofiledefs" USING [eofDA, SN],
  ControlDefs: FROM "controldefs" USING [GFTIndex],
  DebugUtilityDefs: FROM "debugutilitydefs" USING [MREAD],
  InlineDefs: FROM "inlinedefs" USING [COPY],
  LoaderBcdUtilDefs: FROM "loaderbcdutildefs",
  LoadStateDefs: FROM "loadstatedefs" USING [
    BcdAddress, BcdArrayLength, ConfigIndex, ConfigNull, EnumerationDirection,
    FileSegmentHandle, GFTIndex, LoadState, LoadStateGFT, Relocation],
  SDDefs: FROM "sddefs" USING [SD, sGFTLength],
  SegmentDefs: FROM "segmentdefs" USING [
    DeleteFileSegment, FileHandle, FileHint, FileSegmentAddress,
    FileSegmentHandle, InsertFile, NewFileSegment, Read, SwapIn, SwapOut,
    Unlock],
  SystemDefs: FROM "systemdefs" USING [AllocateHeapNode, FreeHeapNode];

DEFINITIONS FROM LoadStateDefs;

ExternalLoadState: PROGRAM
  IMPORTS DebugUtilityDefs, SegmentDefs, SystemDefs
  EXPORTS LoaderBcdUtilDefs, LoadStateDefs, DebugUtilityDefs = PUBLIC

  BEGIN

  state: SegmentDefs.FileSegmentHandle ← NIL;
  loadstate: LoadState ← NIL;
  gft: LoadStateGFT;
  nbcds: ConfigIndex;

  LoadStateInvalid: SIGNAL = CODE;

  InputLoadState: PROCEDURE RETURNS [ConfigIndex] =
    BEGIN OPEN ControlDefs, SDDefs, SegmentDefs;
    i: GFTIndex;
    IF state = NIL THEN SIGNAL LoadStateInvalid;
    SwapIn[state];
    loadstate ← FileSegmentAddress[state];
    gft ← DESCRIPTOR[loadstate+BcdArrayLength, DebugUtilityDefs.MREAD[SD+sGFTLength]];
    nbcds ← 0;
    FOR i IN [0..LENGTH[gft]) DO
      IF gft[i].config # ConfigNull THEN nbcds ← MAX[nbcds, gft[i].config];
      ENDLOOP;
    nbcds ← nbcds + 1;
    RETURN[nbcds]
    END;

  ReleaseLoadState: PROCEDURE =
    BEGIN OPEN SegmentDefs;
    IF ~state.swappedin THEN RETURN;
    Unlock[state];
    IF state.lock = 0 THEN
      BEGIN
      SwapOut[state];
      loadstate ← NIL;
      nbcds ← 0;
      END;
    END;

  MapConfigToReal: PROCEDURE [cgfi: GFTIndex, config: ConfigIndex] RETURNS [rgfi: GFTIndex] =
    BEGIN
    IF cgfi = 0 THEN RETURN[0];
    FOR rgfi IN [0..LENGTH[gft]) DO
      IF gft[rgfi] = [config, cgfi] THEN RETURN [rgfi];
      ENDLOOP;
    RETURN[0];
    END;

  MapRealToConfig: PROCEDURE [rgfi: GFTIndex] RETURNS [GFTIndex, ConfigIndex] =
    BEGIN
    RETURN[gft[rgfi].gfi, gft[rgfi].config];
    END;
```

```
InitializeRelocation: PROCEDURE [config: ConfigIndex] RETURNS [rel: Relocation] =
  BEGIN
  max: CARDINAL ← 0;
  i: GFTIndex;
  FOR i IN [0..LENGTH[gft]) DO
    IF gft[i].config = config THEN max ← MAX[max, gft[i].gfi];
    ENDLOOP;
  rel ← DESCRIPTOR[SystemDefs.AllocateHeapNode[max+1], max+1];
  rel[0] ← 0;
  InlineDefs.COPY[from: BASE[rel], to: BASE[rel]+1, nwords: max];
  FOR i IN [0..LENGTH[gft]) DO
    IF gft[i].config = config THEN rel[gft[i].gfi] ← i;
    ENDLOOP;
  END;

ReleaseRelocation: PROCEDURE [rel: Relocation] =
  BEGIN
  SystemDefs.FreeHeapNode[BASE[rel]];
  END;

BcdSegFromLoadState: PROCEDURE [bcd: ConfigIndex] RETURNS [seg: FileSegmentHandle] =
  BEGIN OPEN SegmentDefs, b: loadstate.bcds[bcd];
  bcdfile: FileHandle;
  NullSN: AltoFileDefs.SN = [1,0,1,17777B,177777B];
  IF b.fp.serial = NullSN THEN b.fp ← state.file.fp;
  bcdfile ← InsertFile[@b.fp, Read];
  seg ← NewFileSegment[bcdfile, b.base, b.pages, Read];
  seg.class ← other;
  IF b.da # AltoFileDefs.eofDA THEN
    WITH s: seg SELECT FROM
      disk => s.hint ← SegmentDefs.FileHint[b.da, b.base];
      ENDCASE;
  END;

UpdateLoadStateDA: PROCEDURE [bcdseg: FileSegmentHandle] =
  BEGIN OPEN SegmentDefs;
  FindSeg: PROCEDURE [c: ConfigIndex, b: BcdAddress] RETURNS [BOOLEAN] =
    BEGIN
    IF b.base = bcdseg.base AND b.pages = bcdseg.pages AND
      b.fp = bcdseg.file.fp THEN
      BEGIN
      WITH s: bcdseg SELECT FROM
        disk => IF s.hint.da # AltoFileDefs.eofDA THEN b.da ← s.hint.da;
        ENDCASE;
      RETURN[TRUE];
      END;
    RETURN[FALSE];
    END;
  IF loadstate = NIL THEN RETURN;  -- loadstate not in
  [] ← EnumerateLoadStateBcds[recentfirst, FindSeg];
  RETURN
  END;

EnumerateLoadStateGFT: PROCEDURE [
 proc: PROCEDURE [GFTIndex, GFTIndex, ConfigIndex] RETURNS [BOOLEAN]]
  RETURNS [GFTIndex] =
  BEGIN
  i: GFTIndex;
  FOR i IN [0..LENGTH[gft]) DO
    IF proc[i, gft[i].gfi, gft[i].config] THEN RETURN[i];
    ENDLOOP;
  RETURN[0]
  END;

EnumerateLoadStateBcds: PROCEDURE [dir: EnumerationDirection,
 proc: PROCEDURE [ConfigIndex, BcdAddress] RETURNS [BOOLEAN]]
  RETURNS [ConfigIndex, BcdAddress] =
  BEGIN
  i: CARDINAL;
  SELECT dir FROM
    recentfirst =>
      FOR i DECREASING IN [0..nbcds) DO
        IF proc[i, @loadstate.bcds[i]] THEN RETURN[i, @loadstate.bcds[i]];
        ENDLOOP;
    recentlast =>
```

```
        FOR i IN [0..nbcds) DO
          IF proc[i, @loadstate.bcds[i]] THEN RETURN[i, @loadstate.bcds[i]];
          ENDLOOP;
      ENDCASE;
    RETURN[ConfigNull, NIL]
    END;

  SetLoadState: PROCEDURE [stateseg: FileSegmentHandle] =
    BEGIN
    state ← stateseg;
    END;

  GetLoadState: PROCEDURE RETURNS [FileSegmentHandle] =
    BEGIN
    RETURN[state];
    END;

  ReleaseBcdSeg: PROCEDURE [bcdseg: SegmentDefs.FileSegmentHandle] =
    BEGIN OPEN SegmentDefs;
    UpdateLoadStateDA[bcdseg];
    Unlock[bcdseg];
    IF bcdseg.lock = 0 THEN DeleteFileSegment[bcdseg];
    END;

END...
```