CP-6

INTERNALS (C00 LEVEL)

SEMINAR

AUGUST 12-23, 1985

Description: CP-6 INTERNALS COURSE (C00 LEVEL)
Date: AUGUST 12-23, 1985

This course details the features of the DPS-8 and Datanet-8 hardware architectures and describes how the CP-6 Host Operating System and the LCP-6 FEP Operating system makes use of these features. The course covers basic system architecture as well as descriptions of major functional areas of the operating systems, including file management, comgroups, connection of terminals, and monitor services. The process of booting, system recovery, and dump analysis will also be covered.

Lab work will include writing a command processor, shared library, and some comgroup related programs, as well as some work with system administration utilities.

Requirements:

Experienced PL-6/Monitor services programmers, familiarity with system administration utilities, such as, SUPER, ANLZ, and PIG. This course

CLASS SCHEDULE

MORNING		AFTERNOON
MON	Administrative Info, Overview, DPS-8 Hardware Arch	CP-6 Use of Hard. Mem. Arch.
TUE	Cont. CP-6 use of Hard. Mem. Arch.	Anatomy of a User
WED	CP-6 User Modes, Intro to CP-6 OS	LAB 1 (SUPER) & LAB 2 (Shared Lib)
THUR	Booting the System, Recovery	LAB 3 Use of ANLZ, STATS, CONTROL
FRI	Comgroup Architecture	LAB 4 Using COMGROUPS
MON	MM, FM, RES & Prog Man Functions	LAB 5 Command Processor
TUE	Walking Tours of CP-6 Functions	LAB 5 cont
WED	FEP Arch & LCP-6 Architecture	LAB 6 FPRG programs
THUR	FEP Users and System Components	LAB 7 ANLZ (FEP)
FRI		OPEN

CP-6 IS A SYSTEM

- COLLECTION OF SOFTWARE INTEGRATED AROUND HARDWARE
- PRIMARY CHARACTERISTICS OF THE HARDWARE ARCHITECTURE THAT INFLUENCES THE SOFTWARE ARCHITECTURE IS THE MEMORY ADDRESSING SYSTEM
- DPS-8 MEMORY SYSTEM IS CALLED VM&S OR NSA

NSA EXTENDS LEVEL 66 BASIC ADDRESSING

- L66 – PROCESS COULD ADDRESS UP TO 256K WORDS OF ADDRESS SPACE. THIS SPACE MUST BE CONTIGUOUS IN REAL MEMORY.

NSA EXTENDS LEVEL 66 BASIC ADDRESSING (CONT)

- NSA – TWO MAJOR EXTENSIONS
  - MULTIPLE 256K ADDRESS SPACES AVAILABLE TO PROCESS
  - SEPARATION OF "VIRTUAL" ADDRESS CONTINUITY FOR REAL ADDRESS VIA PAGE TABLES

(ALSO MAJOR SECURITY FEATURES BUT NOT DETAILED HERE)

NSA ADDED HARDWARE REGISTERS:

- SSR – SAFE STORE REGISTER
- LSR – LINKAGE SEGMENT REGISTER
- ASR/PSR – ARG SEGMENT REGISTER/PARAMETER SEGMENT
- ISR – INSTRUCTION SEGMENT REGISTER
- DR0–7 PAIRED W/AR0–7 – DESCRIPTOR REGISTERS/ADDRESS REGISTERS

NSA ADDED HARDWARE REGISTERS: (CONT)

- PTDBR – PAGE TABLE DIRECTORY BASE REGISTER
- WSR0-7 – WORKING SPACE REGISTERS

NSA USES NEW ENTITIES

- SSS – SAFE STORE STACK (FRAMED BY SSR)
- LS – LINKAGE SEGMENT (FRAMED BY LSR)
- AS/PS – ARGUMENT/PARAMETER SEGMENTS (FRAMED BY ASR/PSR)
- IS – INSTRUCTION SEGMENT (FRAMED BY ISR)
- SEGMENTS – ADDRESS SPACE (FRAMED BY SOME DESCRIPTOR)

NSA USES NEW ENTITIES (CONT.)

- DESCRIPTORS/VECTORS/POINTERS
  (VECTORS/POINTERS REFERENCE DESCRIPTORS)
  (DESCRIPTORS FRAME SEGMENTS)

- WS – WORKING SPACE (PAGE TABLE, POINTED TO BY WSPTD)

- WSPTD – WORKING SPACE PAGE TABLE DIRECTORY (POINTED TO BY PTDBR)

- PAGE – 1024 WORDS OF CONTIGUOUS MEMORY

FIRST STEP TO UNDERSTANDING NSA ADDRESSING INVOLVES THE CONCEPTS OF SEGMENTATION AND WORK SPACES

WHAT IS A SEGMENT?

- CHUNK OF PROGRAM ADDRESS SPACE FROM ONE BYTE TO 256K WORDS
- DEFINED BY A 72 BIT DESCRIPTION WHICH SPECIFIES
  - BOUND (SIZE IN BYTES -1)
  - ACCESS CONTROLS
  - TYPE OF SEGMENT (DATA, DESCRIPTOR, SPECIAL)

WHAT IS A SEGMENT?
(CONT)

- WHAT WS SEGMENT IS IN

- BASE (BYTE OFFSET INTO WS THAT IS VIRT 0 FOR THIS SEGMENT)

WHAT IS A WORKING SPACE?

- CHUNK OF VIRTUAL ADDRESS SPACE 1->N PAGES

- DEFINED BY A PAGE TABLE (ONE WORD/PAGE)
  EACH PAGE TABLE ENTRY GIVES ADDRESS OF
  PAGE IN REAL MEMORY

  - SPECIFIES ACCESS CONTROL

  - PAGE TABLE FOUND BY AN ENTRY IN WSPTD
    (INDEXED BY WS#)

WHAT IS A WORKING SPACE (CONT)

- SPECIFIES ADDRESS OF PAGE TABLE AND SIZE OF PAGE TABLE
- WSPTD IS POINTED TO BY HARDWARE REGISTER PTDBR

THE DOMAIN CONCEPT

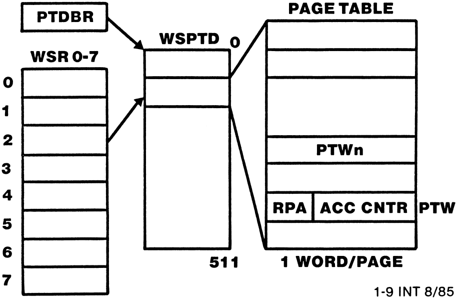
- DOMAIN IS THE ALLOWABLE/ADDRESSABLE WORLD OF SEGMENTS AND WORKING SPACES AS "SEEN" BY AN EXECUTING PROCESS

- THE SET OF HARDWARE ADDRESSING REGISTERS IN CONCEPT WITH THE DESCRIPTOR SEGMENTS AND PAGE TABLES SET UP BY A CONTROLLING (PRIVILEGED) PROCESS CONSTRAIN THE CURRENTLY EXECUTING PROCESS TO ITS DOMAIN

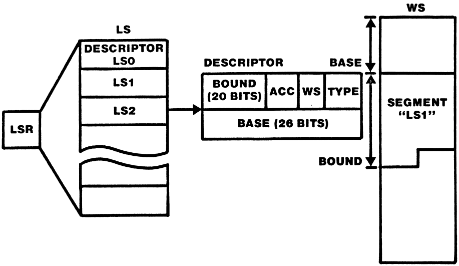
THE DOMAIN CONCEPT (CONT)

- CLIMB INSTRUCTION ALLOWS SWITCHING FROM ONE DOMAIN TO ANOTHER, PUSHING/PULLING UP TO 64 WORDS OF CONTEXT ON SSS

WS CONTROL


PTDBR
WSR 0-7
0
1
2
3
4
5
6
7
WSPTD
0
511
PAGE TABLE
PTWn
RPA
ACC CNTR
PTW
1 WORD/PAGE

SEGMENT CONTROL (PARTIAL)



A (NORMAL) DESCRIPTOR DESCRIBES A SEGMENT

<table>
<tr><td rowspan="2">BOUND IN BYTES (20)</td><td rowspan="2">ACCESS CONTROL FLAGS</td><td colspan="2">WSQ (9)</td><td rowspan="2">TYPE (4)</td></tr>
<tr><td>MISCELLANEOUS FLAGS (6)</td><td>WSR (3)</td></tr>
<tr><td colspan="5">BASE AS BYTE OFFSET INTO WS (26) (NORMALLY)</td></tr>
</table>

- BOUND SPECIFIES SIZE OF SEGMENT (1 BYTE UP TO 256K WORDS)
- BASE LOCATES SEGMENT WITHIN SOME WORKING SPACE
- WSQ/WSR SPECIFIES WORKING SPACE NUMBER

A (NORMAL) DESCRIPTOR DESCRIBES A SEGMENT (cont.)

<table>
<tr><td rowspan="2">BOUND IN BYTES (20)</td><td rowspan="2">ACCESS CONTROL FLAGS</td><td colspan="2">WSQ (0)</td><td rowspan="2">TYPE (4)</td></tr>
<tr><td>MISCELLANEOUS FLAGS (~~0~~6)</td><td>WSR (3)</td></tr>
<tr><td colspan="5">BASE AS BYTE OFFSET INTO WS (26) (NORMALLY)</td></tr>
</table>

- **TYPE SPECIFIES:**
  - **WHETHER SEGMENT CONTAINS DATA OR DESCRIPTORS**
  - **WHETHER WS SPECIFIED DIRECTLY (WSQ) OR INDIRECTLY (WSR)**
  - **INDICATES IF THIS IS NOT NORMAL DESCRIPTOR (IF NOT NORMAL THE REST ALL LOOKS DIFFERENT)**

A (NORMAL) DESCRIPTOR DESCRIBES A SEGMENT (cont.)

<table>
<tr><td rowspan="2">BOUND IN BYTES (20)</td><td rowspan="2">ACCESS CONTROL FLAGS</td><td colspan="2">WSQ (0)</td><td rowspan="2">TYPE (4)</td></tr>
<tr><td>MISCELLANEOUS FLAGS (0)</td><td>WSR (3)</td></tr>
<tr><td colspan="5">BASE AS BYTE OFFSET INTO WS (26) (NORMALLY)</td></tr>
</table>

- ENTRY DESCRIPTOR
- SUPER DESCRIPTOR

● ARE CONTAINED IN DESCRIPTOR SEGMENTS (FRAMED BY A DESCRIPTOR TYPE DESCRIPTOR)

A VECTOR REFERENCES A SEGMENT OR PARTIAL SEGMENT VIA SOME DESCRIPTOR

BOUND (20)	ACCESS CONTROL FLAGS (9)		CONTROL INFO
BASE (20)	▨	S (2)	DESCRIPTOR # (10)

- VECTOR IS COMBINED WITH SPECIFIED DESCRIPTOR FOR FORMING NEW DESCRIPTOR TO BE LOADED INTO DR (LDD INSTR) OR PUSHED ON PS VIA CLIMB

- VECTOR BASE & BOUND CAN SPECIFY SUBSET ALL OF THE SEGMENT

A VECTOR REFERENCES A SEGMENT OR PARTIAL SEGMENT VIA SOME DESCRIPTOR (cont.)

<table>
<tr><td>BOUND (20)</td><td colspan="3">ACCESS CONTROL FLAGS (9)</td><td>CONTROL INFO</td></tr>
<tr><td>BASE (20)</td><td>▨</td><td>S (2)</td><td colspan="2">DESCRIPTOR # (10)</td></tr>
</table>

- ACCESS FLAGS LOGICALLY ANDED WITH DESCRIPTOR FLAGS THEREFORE CAN SUBSET PRIVILEGES, BUT NOT ADD ANY
- S SPECIFIES LS,AS, PS AS DESCRIPTOR SOURCE
- DESCRIPTOR # SPECIFIES WHICH DESCRIPTOR IN LS/AS/PS

A VECTOR REFERENCES A SEGMENT OR PARTIAL SEGMENT VIA SOME DESCRIPTOR (cont.)

<table>
<tr><td>BOUND (20)</td><td colspan="3">ACCESS CONTROL FLAGS (9)</td><td>CONTROL INFO</td></tr>
<tr><td>BASE (20)</td><td>▨</td><td>S (2)</td><td colspan="2">DESCRIPTOR # (10)</td></tr>
</table>

- SPECIAL VALUES OF S, D RESERVED TO SPECIFY CERTAIN HARDWARE REGISTERS (DRn, ISR, LSR, SSR, PSR, ETC.)

A POINTER REFERENCES A SEGMENT

VIA SOME DESCRIPTOR AND CAN SPECIFY AN OFFSET INTO THE SEGMENT TO THE BIT LEVEL

WORD OFFSET 18	BYTE (2)	BIT (4)	S (2)	DESCRIPTOR # (10)

- S, D FIELDS (SEGID) REFERENCE A DESCRIPTOR (SAME AS VECTOR)
- WHEN LOAD POINTER INSTRUCTION EXECUTED
  - WORD, BYTE, BIT OFFSETS LOADED INTO AR
  - S, D LOADED INTO SEGID REGISTER (INFO ONLY)
  - SPECIFIED DESCRIPTOR LOADED INTO DR
- OFFSET VALUES DO NOT PROSCRIBE ACCESS BELOW SPECIFIED ADDRESS

SOME SPECIAL REGISTERS WHICH WILL BE COMMONLY REFERENCED

LSR - CONTAINS DESCRIPTOR WHICH FRAMES CURRENT LS

- LOADED ONLY AS A RESULT OF CLIMB INSTRUCTION

SOME SPECIAL REGISTERS WHICH WILL BE COMMONLY REFERENCED (CONT)

ISR – CONTAINS DESCRIPTOR WHICH FRAMES CURRENT INSTRUCTION SEGMENT

- IC IS OFFSET INTO CURRENT IS
- LOADED VIA CLIMB OR TSX INSTRUCTION

SSR – CONTAINS DESCRIPTOR WHICH FRAMES CURRENT SAFE STORE STACK

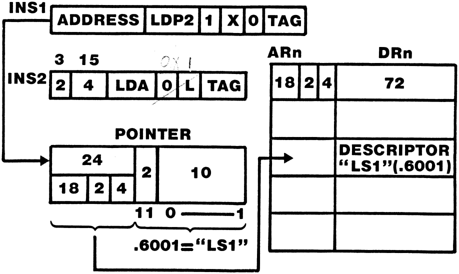
SOME SPECIAL REGISTERS WHICH WILL BE COMMONLY REFERENCED (CONT)

-SS IS STACK FOR CONTEXTS PUSHED/PULLED
-VIA CLIMB INSTRUCTION

ASR/PSR-CONTAIN DESCRIPTORS WHICH FRAME
-CURRENT ARGUMENT/PARAMETER SEGMENTS

-AS/PS IS STACK USED TO PASS DESCRIPTORS
-FROM ONE DOMAIN TO ANOTHER VIA CLIMB
-INSTRUCTION

SEGMENT SPECIFICATION IN INSTRUCTIONS



CONCEPT OF AN EXECUTING PROCESS

- CURRENT PROCESS CONTAINED BY DOMAIN AS ESTABLISHED

- MACHINE HAS SLAVE MODE/PRIVILEGED MODE

- ONLY PRIVILEGED MODE CAN "SET UP" DOMAINS

CONCEPT OF AN EXECUTING PROCESS (CONT)

- SLAVE MODE SWITCHED TO PRIVILEGED AS RESULT OF
  - TRAPS (INCLUDING PMME)
  - INTERRUPTS (TIMERS, I/O)
- HARDWARE TOGETHER WITH "CORRECT" PRIVILEGED PROCESS AND "CORRECT" SYSTEM ARCHITECTURE ARE WHAT PROVIDES SYSTEM SECURITY

THE USE OF DPS-8 MEMORY ARCHITECTURE (NSA) BY CP-6

CP-6 WORKING SPACE USAGE

WS#0 ADDRESSES REAL MEMORY

WS#1 MONITOR PROCEDURE AND DATA

WS#2 SCRATCH USE BY XDELTA

WS#3 USED BY DELTA/ANLZ

WS#4 COMMAND PROCESSORS PROCEDURE
(IBEX AND OTHERS)

CP-6 WORKING SPACE USAGE (CONT)

WS#5 DEBUGGERS PROCEDURE (DELTA AND OTHER)

WS#6 ALTERNATE SHARED LIBRARIES PROCEDURE (I-D-S/II AND OTHERS)

WS#7 SCRATCH USE BY I/O END ACTION

WS#8 CURRENT USERS WORKING SPACE

CP-6 WORKING SPACE USAGE (CONT)

WS#9
WS#10 CURRENT USERS VIRTUAL SEGMENTS
WS#11

WS>11 USED BY FEP I/O AND COMGROUPS

USER VIRTUAL SPACE

- "THE USER" FROM STANDPOINT OF MONITOR CONTROL
- CONTAINS ALL "GLUE" HOLDING IT TOGETHER
  - PAGE TABLE
  - LINKAGE SEGMENTS

USER VIRTUAL SPACE (CONT)

- SAFE STORE STACK
- ARGUMENT/PARAMETER SEGMENTS

- COLLECTS ALL PHYSICAL MEMORY ALLOCATED TO THIS USER

USER VIRTUAL SPACE (CONT)

- LOCATES BUT DOES NOT CONTAIN SHARED PROCEDURE
  - SHARED PROCESSOR
  - RUN-TIME LIBRARY

SIMPLIFIED VIEW OF USER'S WS (1MW)

PAGE TABLE OF WS
HJIT - 4LS's SSS, AS/PS
JIT AND MONITOR TSTACK
FILE BUFFERS
DCBs
INSTRUCTION SEGMENT
USERS 8 DYNAMIC SEGMENTS
CP'S DYNAMIC SEGMENTS
DB'S 8 DYNAMIC SEGMENTS
ASL'S 8 DYNAMIC SEGMENTS

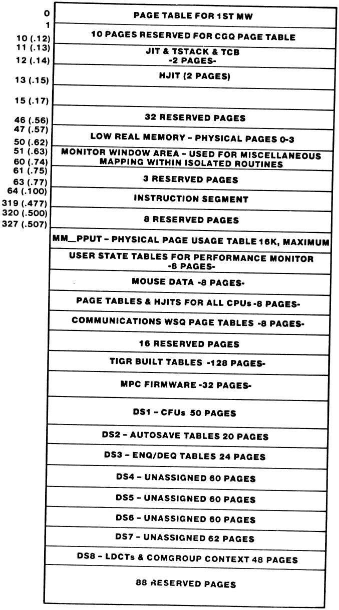
1 MW USER VIRTUAL LAYOUT

	0	PAGE TABLE	MONITOR
	1 10 (.12)	10-PAGE GAP CORRESPONDING TO MON CGQ MAP PAGE 5	
USER RD	11 (.13)	JIT & TSTACK	MONITOR W
	12 (.14)	RESERVED PAGE	
USER	13 (.15)	HJIT	MONITOR
	14 (.16) 15 (.17)	2 RESERVED PAGES	
	16 (.20) 46 (.56)	FPOOLS -31 PAGES-	MONITOR
USER RD	47 (.57) 60 (.74)	ROSEG -14 PAGES-	MONITOR W
	61 (.75) 63 (.77)	3 RESERVED PAGES	
USER R,W,E	64 (.100) 319 (.477)	INSTRUCTION SEGMENT	
	320 (.500) 321 (.501)	SBUF2 SBUF1 STEP SPECIAL BUFFERS - 2 PAGES	MONITOR
	322 (.502) 327 (.507)	6 RESERVED PAGES	
	328 (.510) 391 (.607)	DB DATA SEGMENTS -UP TO 64 PAGES-	DELTA
	392 (.610) 519 (.1007)	ASL DATA SEGMENTS -UP TO 128 PAGES-	ASL
	520 (.1010) 551 (.1047)	CP DATA SEGMENTS -UP TO 32 PAGES-	CP
USER R,W	552 (.1050) 935 (.1647)	USER DATA SEGMENTS -UP TO 384 PAGES-	
	936 (.1650) 1023 (.1777)	88 RESERVED PAGES	

HEAVILY ACCESSED PAGES

NAME	V.P.#	AM ROW#
JIT	.13	.13
HJIT	.15	.15
ROSEG(1)	.57	.17
ISR(1)	.100	0
U. DATA SEG. (1)	.1050	.10
STEP BUFFER #1	.501	.01

1MW MONITOR VIRTUAL LAYOUT



0
1
10 (.12)
11 (.13)
12 (.14)
13 (.15)
15 (.17)
46 (.56)
47 (.57)
50 (.62)
51 (.63)
60 (.74)
61 (.75)
63 (.77)
64 (.100)
319 (.477)
320 (.500)
327 (.507)

PAGE TABLE FOR 1ST MW

10 PAGES RESERVED FOR CGQ PAGE TABLE

JIT & TSTACK & TCB
-2 PAGES-

HJIT (2 PAGES)

32 RESERVED PAGES

LOW REAL MEMORY - PHYSICAL PAGES 0-3

MONITOR WINDOW AREA - USED FOR MISCELLANEOUS
MAPPING WITHIN ISOLATED ROUTINES

3 RESERVED PAGES

INSTRUCTION SEGMENT

8 RESERVED PAGES

MM_PPUT - PHYSICAL PAGE USAGE TABLE 16K, MAXIMUM

USER STATE TABLES FOR PERFORMANCE MONITOR
-8 PAGES-

MOUSE DATA -8 PAGES-

PAGE TABLES & HJITS FOR ALL CPUs -8 PAGES-

COMMUNICATIONS WSQ PAGE TABLES -8 PAGES-

16 RESERVED PAGES

TIGR BUILT TABLES -128 PAGES-

MPC FIRMWARE -32 PAGES-

DS1 - CFUs 50 PAGES

DS2 - AUTOSAVE TABLES 20 PAGES

DS3 - ENQ/DEQ TABLES 24 PAGES

DS4 - UNASSIGNED 60 PAGES

DS5 - UNASSIGNED 60 PAGES

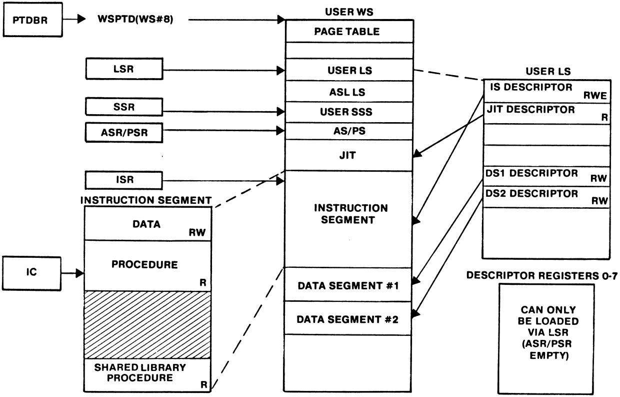
DS6 - UNASSIGNED 60 PAGES

DS7 - UNASSIGNED 62 PAGES

DS8 - LDCTs & COMGROUP CONTEXT 48 PAGES

88 RESERVED PAGES

HOW HARDWARE DEFINES (CONTAINS) USERS DOMAIN
(USER PROGRAM CURRENTLY IN CONTROL)



PTDBR

WSPTD(WS#8)

LSR

SSR

ASR/PSR

ISR

IC

INSTRUCTION SEGMENT

DATA RW

PROCEDURE R

SHARED LIBRARY PROCEDURE R

USER WS

PAGE TABLE

USER LS

ASL LS

USER SSS

AS/PS

JIT

INSTRUCTION SEGMENT

DATA SEGMENT #1

DATA SEGMENT #2

USER LS

IS DESCRIPTOR RWE

JIT DESCRIPTOR R

DS1 DESCRIPTOR RW

DS2 DESCRIPTOR RW

DESCRIPTOR REGISTERS 0-7

CAN ONLY BE LOADED VIA LSR (ASR/PSR EMPTY)

MORE ABOUT THE INSTRUCTION SEGMENT

FRAMED BY ISR DESCRIPTOR (LS#0)

DESCRIPTOR ALWAYS SAYS SIZE IS 256K

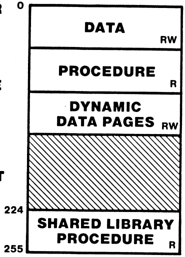
DESCRIPTOR ALLOWS R, W, E ACCESS

WRITEABILITY CONTROLLED BY PAGE TABLE

SHARED LIBRARY & PROCEDURE & PROGRAM NOT WRITEABLE

UNALLOCATED PAGES MARKED NOT PRESENT

CONTAINS POINTERS TO GET TO OTHER SEGMENTS



THE MONITOR DOMAIN

- USES 1MW WORKING SPACE
- HAS ONLY ONE LINKAGE SECTION
- MANY OF THE DESCRIPTORS IN THE LS REFER TO USER WS THESE DESCRIPTORS HAVE READ AND WRITE ACCESS
    - JIT
    - FILE BUFFERS

THE MONITOR DOMAIN (CONT)

- RO SEGMENT (TCB, ECCB, DCBs)
- HJIT
- USERS MAP

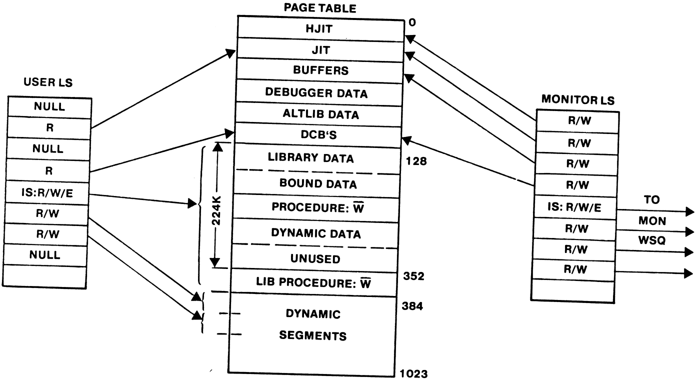
- THERE ARE MANY SPECIAL PURPOSE DESCRIPTORS

THE MONITOR DOMAIN (CONT)

- THE PSR FRAMES THE PS OF DESCRIPTORS PASSED TO THE MONITOR BY THE USER VIA THE PMME INSTRUCTION. THESE DESCRIPTORS FRAME THE PARAMETERS/BUFFERS OF THE SERVICE REQUEST

USER VIRTUAL ADDRESS SPACE

AS SEEN BY USER LS, MONITOR LS



"USER" HAS FOUR LINKAGE SEGMENTS

- ONE FOR EACH SLAVE DOMAIN WITH ITS CONTEXT
  - USER LS– DEFINES WS AS SEEN BY USER
  - CP LS – DEFINES WS AS SEEN BY CP
  - DB LS – DEFINES WS AS SEEN BY DB
  - ASL LS – DEFINES WS AS SEEN BY ASL
- EACH DOMAIN'S IS DESCRIPTOR (LS#0) REFERENCES PROCEDURE IN OWN WS

"USER" HAS FOUR LINKAGE SEGMENTS (CONT)

- EACH DOMAIN'S DYNAMIC DATA SEGMENT DESCRIPTOR REFERENCES APPROPRIATE PAGES IN USERS WS
- OTHER DESCRIPTORS SAME FOR ALL FOUR DOMAINS

THE CP, DB, ASL DOMAINS

- EACH IS SIMILAR
  - PROCEDURE IN SEPARATE WS
  - 8 DATA SEGMENTS IN USER WS (PRIVATE)
  - SHARES JIT, DCBs WITH USER PROGRAM, EACH OTHER
- EACH IS UNIQUE TO SATISFY DIFFERENT REQUIREMENTS

THE CP, DB, ASL DOMAINS (CONT)

- CP, DB HAVE RESERVED DCB SLOTS
- ENTRY, EXIT DIFFERENT FOR EACH, PARTLY BY NEED, PARTLY FOR BEST PERFORMANCE (ASL)

- SEPARATE DOMAINS MINIMIZE OVERHEAD, ALLOW FOR TRULY EXTERNAL DEBUGGER AND COMMON PROCESSOR

THE COMMAND PROCESSOR DOMAIN

- JIT DESCRIPTOR HAS READ, WRITE ACCESS
- CANNOT SEE USER IS OR DATA SEGMENTS
- CAN SEE DCBs ROSEG
- GETS CONTROL VIA MONITOR INTERVENTION AT PROGRAM ABORT, EXIT, OR ATTENTION REQUEST
- GIVES CONTROL BACK VIA SPECIAL MONITOR SERVICE

THE DEBUGGER DOMAIN

- LS CONTAINS DESCRIPTOR GIVING IT ACCESS TO THE USERS LS AND HENCE ALL THAT USER PROGRAM CAN SEE WITH SAME ACCESS (R, W)
- USERS PROGRAM PAGES ARE MARKED WRITEABLE BY THE MONITOR PROVIDING THE PROCEDURE IS NOT SHARED (SHARED PROCEDURE AND LIBRARY MAY BE UNSHARED BY REQUEST)
- GETS CONTROL OF ALL SPECIAL EVENTS, TRAPS, ETC. VIA MONITOR INTERVENTION

THE DEBUGGER DOMAIN (CONT)

- GIVES CONTROL BACK VIA SPECIAL MONITOR SERVICE

THE ASL DOMAIN

- LS HAS NO SPECIAL DESCRIPTORS, NO SPECIAL ACCESS
- PS CONTAINS DESCRIPTORS PASSED BY USER PROGRAM WHICH FRAME THOSE PORTIONS OF USERS IS AND DATA SEGMENTS TO WHICH ASL IS TO HAVE ACCESS
- GETS CONTROL DIRECTLY FROM USER VIA CLIMB INSTRUCTION (USER LS HAS ENTRY DESCRIPTOR LOCATING ASL LS)

THE ASL DOMAIN (CONT)

- GIVES CONTROL DIRECTLY BACK TO USER VIA CLIMB (OUTWARD) INSTRUCTION

GENERAL INFORMATION

- THE MONITOR IS THE ONLY PROCESS WHICH EXECUTES IN PRIVILEGED MODE (PLUS XDELTA)
- THE USER PROCESS RUNS IN SLAVE MODE NO MATTER WHICH OF FOUR DOMAINS (USER, CP, DB, ASL) IS IN CONTROL
- ALL USER DOMAINS USE THE PMME CLIMB INSTRUCTION TO MAKE SERVICE REQUESTS ON THE MONITOR

GENERAL INFORMATION (CONT)

- THE PMME CLIMB INSTRUCTION:
  - ENTERS PRIVILEGED MODE
  - SWITCHES DOMAINS TO THE MONITOR VIA ENTRY DESCRIPTOR IN LOW REAL MEMORY (RESERVED LOCATION)
  - CAN PASS DESCRIPTORS FROM THE USER DOMAIN TO THE MONITOR

GENERAL INFORMATION (CONT)

- TRAPS AND INTERRUPTS CAUSE SIMILAR ENTRY TO THE MONITOR BUT PASS NO USER DESCRIPTORS

SUMMARY

- MONITOR HAS 1MW VIRTUAL WS CONTROLLED BY PAGE TABLE
- EACH USER HAS 1MW VIRTUAL WS CONTROLLED BY PAGE TABLE
- EACH USER HAS FOUR LINKAGE SEGMENTS – USER, CP, DB, ASL
- MONITOR HAS ONE LINKAGE SEGMENT

SUMMARY (CONT)

- MONITOR, CP, DB, AND ASL LS REFER TO USER WS FOR CONTEXT SPECIFIC TO THAT USER
- MONITOR SETS UP PAGE TABLE, LINKAGE SECTIONS, HARDWARE TAKES CARE OF ALL ADDRESSING ACCESS AND CONTROL

SUMMARY

- INTERACTIONS BETWEEN DOMAINS ALL HANDLED BY CLIMB INSTRUCTION
  - SIMPLE CLIMB FOR USER-ASL
  - PMME CLIMB FOR ALL OTHERS
    - STRAIGHTFORWARD FOR USER-MONITOR
    - VIA SPECIAL MONITOR MANIPULATION FOR USER/CP/DB

ANATOMY OF A USER

SYSTEM SERVICE INTERFACE

- UNIFORM ABSTRACT INTERFACE
- ISOLATES USER PROGRAMS FROM HARDWARE/SYSTEM SOFTWARE
- ENABLES COMPLETE DEVICE INDEPENDENCE
- MAKES FILES INTERCHANGEABLE AMONG LANGUAGES
- CLIMB IS EXCELLENT VEHICLE

CP-6 USER ENVIRONMENT

- L66B SLAVE INSTRUCTION SET
- MONITOR SERVICES AS EXTENSION OF INSTRUCTION SET
- MEMORY AS DEFINED BY USER DOMAIN
- JIT, TCB, DCBs

CP-6 USER ENVIRONMENT (CONT)

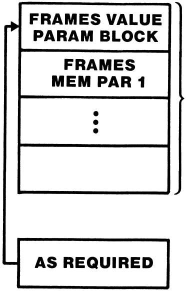
PLUS OPTIONALLY:

- SHARED RUN-TIME LIBRARY
- ALTERNATE SHARED LIBRARY

MONITOR CALL FORMAT:

```
EPPRO   α
PMME    CODE, n
[TRA    β]
```

β EXCEPTION
PROCESSING



FRAMES VALUE
PARAM BLOCK

FRAMES
MEM PAR 1

•
•
•

AS REQUIRED

n VECTORS,
1 FOR VALUE
PARAM
BLOCK PLUS
1 FOR EACH
'MEMORY
TYPE'

3-4 INT 8/85

PROGRAM BINDING INTERFACE

- OBJECT LANGUAGE COMMON FOR ALL LANGUAGES
- DEBUG SCHEMA INCLUDED PERMITTING COMMON DEBUGGER
- ORIENTED TO SHARED PROCEDURE ENVIRONMENT
- PERMITS GENERAL LINK TIME BINDING

FEATURES OF OBJECT LANGUAGE/ LINKER

- GENERAL RELOCATION OF FIELDS
- COMPLETE DESCRIPTION OF VARIABLE/ PROCEDURES
- DETECTION OF PROCEDURE DEFINITION/CALL MISMATCH
- SYSTEMIC DEFINITIONS SUPPLIED BY LINKER

FEATURES OF OBJECT LANGUAGE/ LINKER

- PERMITS GENERAL LINK TIME BINDING TO PROMOTE MODULAR PROGRAMMING
- DESIGNED FOR EASE OF GENERATION AND SPEED OF LINKING
- LINKER PRODUCES RUN UNIT

CP-6 OBJECT UNIT

- SECTION IS BASIC UNIT OF ALLOCATION (SIZE, ATTRIBUTES)

- ALL CODE AND DATA LOADED INTO SOME SECTION

- RELOCATION MAY BE DONE RELATIVE TO ANY SECTION OR EXTERNAL REFERENCE

- EXTERNAL ENTRY (ENTDEFs) AND DATA (SYMDEFS) ARE DECLARED RELATIVE TO SOME SECTION (OR CONSTANT)

CP-6 OBJECT UNIT (CONT)

- ENTREFS, SYMREFS, AND SEGREFS ARE USED TO ACQUIRE THE VALUES OF ADDRESSES (OR CONSTANTS) DEFINED ELSEWHERE AND TO ACQUIRE VALUES OF SEGIDS (SUPPLIED BY THE LINKER)
- CONTAINS INFORMATION TO ALLOW LINKER CONSISTENCY CHECKS

CP-6 OBJECT UNIT (CONT)

- INFORMATION ABOUT STATEMENT NUMBERS, STATEMENT LABELS, VARIABLE NAMES AND DATA TYPES IS PROVIDED VIA THE DEBUG TABLES

- STANDARD FILE, OF COURSE

DEBUG SCHEMA:

- STATEMENT DEFINITION
- VARIABLE DEFINITION
- ACCOMODATES ALL LANGUAGES
- GENERAL ENOUGH TO PRODUCE LO
- DELTA MAKES IS SWING

FEATURES OF RUN UNIT

- SAME FORMAT USED FOR ALL TYPES OF PROGRAMS
- EXECUTABLE FORM CONTAINING STATIC, PROCEDURE, DCBS AND TCBS
- IDENTIFIES REQUIRED LIBRARY AND ASL
- STANDARD FILE, OF COURSE

CP-6 RUN UNIT:

- RESULT OF LINKING ONE OR MORE OBJECT UNITS

- ALL CROSS REFERENCES ARE SATISFIED, ENTDEFS AND SYMDEFS HAVE VALUES

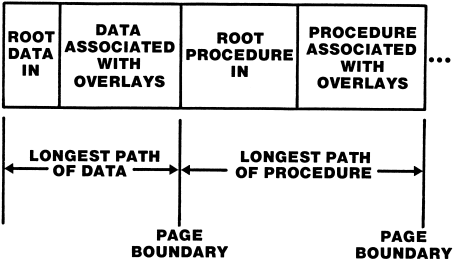
- DEBUG INFORMATION COPIED TO RUN UNIT WITH ADDRESSES RESOLVED

- OVERLAY TREE STRUCTURE REFLECTS IN DATA AND PROCEDURE

CP-6 RUN UNIT: (CONT)

- COMMON BLOCKS COALESCED
- DCBS COALESCED

CP-6 RUN UNIT (WHEN BROUGHT IN FOR EXECUTION)


ROOT DATA IN
DATA ASSOCIATED WITH OVERLAYS
ROOT PROCEDURE IN
PROCEDURE ASSOCIATED WITH OVERLAYS
•••
LONGEST PATH OF DATA
LONGEST PATH OF PROCEDURE
PAGE BOUNDARY
PAGE BOUNDARY

PROGRAM CALLING INTERFACE

- STANDARD SYSTEM CALLING SEQUENCE
- ACCOMODATES NEED OF ALL LANGUAGES
- FACILITATES MIXED LANGUAGE PROGRAMS
- DESIGNED FOR EFFICIENT FORMAL INTERFACE AMONG PROGRAMS IN A RUN UNIT
- PROMOTES COMMON LIBRARY ROUTINES

CALLING SEQUENCE ATTRIBUTES

- DESIGNED FOR NSA ENVIRONMENT
- ORIENTED TO PURE PROCEDURE ENVIRONMENT
- CONTAINS INFORMATION USEFUL TO DEBUGGER
- INTEGRATED WITH PL/1, PL-6 STACK FRAME MANAGEMENT
- ENCOMPASSES LIBRARY FUNCTION CALL FORMAT

PL-6 RELATIONSHIP TO SYSTEM

- LANGUAGE BUILT TO FIT SYSTEM, NOT VICE VERSA
    - OPERATING SYSTEM IS THE RUN-TIME "LIBRARY"
- FACILITIES INCLUDED TO FACILITATE
    - BUILDING THE SYSTEM
    - USING THE SYSTEM

PL–6 RELATIONSHIP TO SYSTEM (CONT)

- DOES NOT PROVIDE HIDDEN CONTROL MECHANISMS
- SYSTEM IS NOT PREJUDICED TO ANY LANGUAGE
  - BUT PROVIDES FACILITIES NECESSARY TO IMPLEMENT ALL
- DESIGNED TO MAKE DATA DEFINITIONS VISIBLE AND CONTROLLABLE

NOT AN APPLICATIONS LANGUAGE

- NO COMPLEX RUN–TIME
- NO FLOAT OR DECIMAL DATA TYPES
- INTENDED FOR USE BY SKILLED PROGRAMMERS
- ATTEMPT TO STRIKE PROPER BALANCE AMONG STRUCTURE, PERFORMANCE, CHECKING, ETC.
- BUT IT IS RIGHT FOR MANY APPLICATIONS

STRUCTURE, PERFORMANCE, PROGRAMM CONTROL, CHECKING

- ATTEMPT TO GIVE NO BIG "SURPRISES"
- MODULARITY ENCOURAGED, GLOBAL AND INTERNAL
- PARAMETER PASSING ENCOURAGED, NO DYNAMIC DATA TYPE CHECKING

STRUCTURE, PERFORMANCE, PROGRAM CONTROL, CHECKING (CONT)

- NO DATA TYPE COERCION, INTERNAL OR EXTERNAL
- EXPLICIT DATA REDEFINITION FREELY ALLOWED

IT'S A SYSTEM:

- MONITOR ROUNDS OUT THE MACHINE
- LANGUAGES PLAY TOGETHER
- DEBUGGER HANDLES ALL LANGUAGE

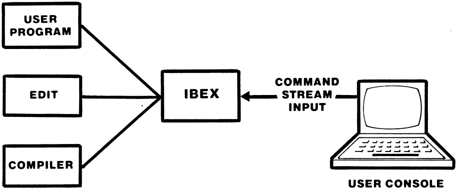
ONLINE/BATCH/GHOST/TP MODE

- DIFFERENT WAY TO GLUE TOGETHER SAME THING
- ALL USERS ARE MUCH MORE ALIKE THAN DIFFERENT
- MAJOR DIFFERENCES:
  - INITIATION OF USER
  - ACQUISITION OF RESOURCES

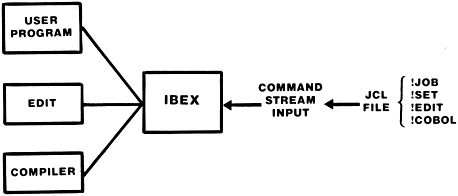
ONLINE/BATCH/GHOST/TP MODE (CONT)

- SOURCE OF COMMAND STREAM
- AUTHORIZATION LIMITS
- DEFAULT ASSIGNMENT OF DCBs

A TIMESHARING USER



A BATCH USER



USER PROGRAM

EDIT

COMPILER

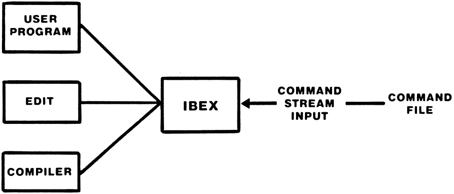
IBEX

COMMAND STREAM INPUT

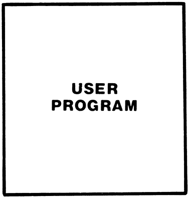
JCL FILE

!JOB
!SET
!EDIT
!COBOL

A GHOST USER

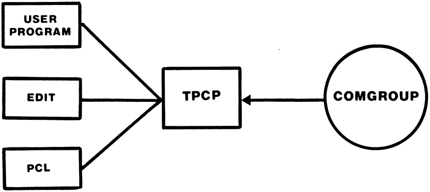

USER PROGRAM
EDIT
COMPILER
IBEX
COMMAND STREAM INPUT
COMMAND FILE

SPECIAL SYSTEM GHOST

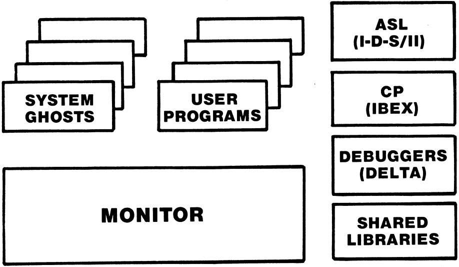


USER
PROGRAM

TRANSACTION PROCESSING USER



OVERALL PICTURE OF CP-6 OPERATING SYSTEM



THE CP-6 MONITOR

- THE MONITOR IS THE ONE PRIVILEGED PROCESS WHICH CONTROLS THE OPERATION OF THE SYSTEMS AND PROVIDES SERVICE FUNCTIONS TO ALL OTHER PROCESSES

- MOST OF THE CODE WHICH COMPRISES THE MONITOR CAN BE EXECUTED SIMUTANEOUSLY BY MULTIPLE CPUs – EACH CPU HAS SOME SMALL AMOUNT OF PRIVATE DATA, BUT BY AND LARGE MOST DATA IS SHARED

THE CP-6 MONITOR (CONT)

- THE MONITOR RUNS IN TWO DISTINCT MODES

  - PERFORMING A SERVICE ON BEHALF OF A SPECIFIC USER (BY REQUEST)

  - PERFORMING OVERHEAD FUNCTION NOT ON BEHALF OF ANY SPECIFIC USER

CP-6 SYSTEM GHOSTS

- SLUG/MBS - START-UP AND LOGON USER GHOST/MULTI-BATCH SCHEDULER
- KEYIN - GHOST TO HANDLE OPERATOR COMMUNICATION AND LOGGING
- OUTSYM - OUTPUT SYMBIONT CONTROL GHOST
- PRESCAN - CHECKS BATCH JCL

CP-6 SYSTEM GHOSTS (CONT)

- INSYM – INPUT SYMBIONT CONTROL GHOST
- ELF – ERROR LOG FILE WRITER GHOST
- PIG – PACK INITIALIZATION GHOST
- FROG – FRONTEND GHOST

CP-6 SYSTEM GHOSTS (CONT)

• DOG	- DISPLAY ONLINE USERS GHOST
• GOOSE	- GHOST TO GOOSE GHOSTS AND LAY - KEYINS
• MAILMAN	- DELIVERS MAIL
• JAYS	- JOURNAL ALL YOUR STUFF GHOST

CP-6 SYSTEM GHOST (CONT)

- SCOTTY – PERFORMS FILE MOVES
- TPA – TRANSACTION PROCESSING ADMINISTRATION GHOST (NOT REALLY SYSTEM GHOST)

SPECIAL SHARED PROCESSORS

- COMMAND PROCESSOR – IBEX, TPCP
- DEBUGGERS – DELTA
- ALTERNATE SHARED LIBRARY – I-D-S/II, ARES

SYSTEM PROGRAMS

- EXAMPLES OF OTHER PROCESSORS PROVIDED WITH CP-6 AS PART OF SYSTEM

CONTROL
NETCON
STATS
SUPER
RATES
EFT
IMP
ELAN
ANLZ

SYSCON
LABEL
SPIDER
DEF
DEF
VOLINIT
PIGETTE
REPLAY
TRADER

BOOTING A SYSTEM

Release contents on multiple tape reels

#CP6P01, #CP6P02 [,#CP6P03]
[DEF-created]
- contain bootable information and run units

#CP6T1, #CP6T2 [,#CP6T3]
[EFT-created]
- contain "tools"... electronic manuals, the X account, QUAC tests, :LIBRARY, DEMO, SUPPORT, :CONVERT, :SRB, :xxxPRC

BOOTING A SYSTEM

Other stuff in box:

- Software Release Bulletin
  [installation instructions]
- Packing slip

Other stuff you'll need

- Other good California stuff
- Supplemental sedatives
- Computer

Advice: read open STARs against this
release FIRST...especially sev "A"s

PO tape, volume 1

- Bootstrap
- COYOTE
- AARDVARK
- FIRMWARE
- SCHEMA for M:MON
- XDELTA and XDELTALS
- MONITOR (M:MON) and MONITOR HJIT
- GHOST1 and GHOST1 HJIT
- PATCHES
- TAPE LABEL (VOL1)
- $XINSTALL

PO tape, volumes 2 and beyond

Rest of stuff destined for life in :SYS

- service processors
  - IBEX, DELTA, LOGON, PCL, EDIT, IMP, PLOVER, PARTRGE, TPA, TPCP, TRADER, LINK, LEMUR, FEPLINK, TEXT, PL6, CALF, ELSIE
- system maintenance processors
  - EFT, ARCOM, CONTROL, SUPER, NETCON, PIG, STATS, ELAN, GOOSE, SYSCON, DEER, REPLAY, VOLINIT, TURTLE, SPIDER, PIGETTE, ANLZ, DEF, RATES, TOLTS

PO tape, volumes 2 and beyond

- system ghosts
  - SLUG, KEYIN, OUTSYM, INSYM, ELF, PRESCAN, PIG, GOOSE, DOG, [JAYS,] [MAILMAN,] FROG

- needed libraries
  - :SHARED_SYSTEM, :SHARED_SPECIAL

- host orphans
  - RCVR2, ALTKEY

PO tape, volumes 2 and beyond

- FEP software
  - M:FEP
  - :SHARED_LCP6_SYSTEM <sans 3270>
  - :SHARED_LCP6_RELEASE <with 3270>

- handlers
  - NODEADMN, COUPLER, ASYNC, BISYNC, UNITREC, HDLCX25

- FEP orphans
  - PIGLET, ANLZ_FPRG, DELTA_FPRG

PO tape, volumes 2 and beyond

- Other stuff
  - :SHARED_FPL, COMMAND_FP
  - TND_MLC16_HNDLR, MDC, MLCP

- What? Still more?
  - HELP files
  - :?ERRMSG files

- Separately priced software
  - COBOL
  - COBOLE
  - SORT/MERGE
  - DIGS
  - FORTRAN
  - FPL

PO tape, volumes 2 and beyond

- BASIC
- APL
- 6EDIT
- ARES/ARGENT/XARGENT
- IDS (DBUTIL, DBACS)
- IDP
- RPG
- GMAP6
- MAIL/SEND/NODEATER

- Each has libraries, HELP files

BOOT SEQUENCE

- Hardware boot button starts 1 record read
- COYOTE uses primitive reads to read rest
- transfer to AARDVARK
- read firmware modules into memory
- load firmware into UR, MT, DP controllers
- rewind tape
- search for "CP-6" record
- read rest of PO (schema, M:MON, XDELTA)
- build system disk area
- patch using MINI
- exit to XDELTA for monitor patching
- XDELTA uses MINIQ (in AARDVARK) for I/O

BOOT SEQUENCE

- climb to monitor, into TIGR
- TIGR uses MINIQ to get configuration
- builds tables...
- before MINIQ returns EOF to TIGR read, RUMs, PLOVERs, and BOOTIMEs xferred to system area
- TIGR gets EOF, kills VOLINIT and self (FRITOLAY)
- calls scheduler
- GHOST1, poised for execution, starts by building #SYS accounts & reconstructing then restores labelled PO files (timewarp)
- convert XDELTA output to :PF?.:SYSTAC

BOOT SEQUENCE

- convert RUMs into :RUM
- convert PLOVERs into ::PLOVER
- convert BOOTIME
- "SPIDER" in :SHARED_SPECIAL,DELTA
- :RUM exist? ->
  - delete ::RUM, mod :RUM->::RUM
- OUTPUT INTO :PF?.:SYSTAC
- ALIB to DELTA "READ ::RUM.:SYS"
- when rums done, reinstall DELTA & lib
- install LOGON, IBEX, :SHARED?, ASLs
- mod JIT -> LDTRC to SLUG
- SLUG starts rest of ghosts

BOOT SEQUENCE

- reconstruct JOBSTATS, sysid ranges
- SLUG starts SUPER if files not there
- then KEYIN, PIG, DINGO
- when DINGO done, ELF, OUTSYM, INSYM, PRESCAN, FROG, CONTROL, DOG, GOOSE
- SCOTTY, MAILMAN, JAYS, ELAN (TB)
- boot FEPs

RECOVERY SEQUENCE

- calls to SCREECH are individual monitor entry points that are defined by caller with flags indicating what to dump
- M$SCREECH PMME has default flags
- climb using ASL form using ASL slot in MON LS into IRM$SCREECH
- setup flags, IRM_SCODE, etc.
- check for same SUA user, etc.
- message on console if IT_XDELTA there
- goto XDELTA if there, else LTRAD to AARDVARK (to save "good" SSF)

RECOVERY SEQUENCE

- AARDVARK/RECOVERY takes dump based on flags (dump area overflow, bypass) into system area
- close files (SUA, SCREECH)
- RETURN to IRM$SCREECH if SUA or SNAP
- save JITs in system area, too
- reboot (fetch new M:MON, GHOST1 into memory)
- restore system tables
- call scheduler for GHOST1 restart
- do accounting from system area JITs
- restart system ghosts
- phew!

TOOLS TAPES

- SUPPORT
- documentation (SRBs, manuals)
- X
- DEMO
- QUAC
-
- CONVERT
- :COOPRC
- :LIBRARY

ALTERNATE SCHEMA

Alternate schema enhances XDELTA debugging of DEBUGGERs, ASLs, COMMAND PROCESSORs, and SYSTEM GHOSTS

Functional code groups = ++

AS tape = i-xx-n

```
!SET M$AS FT#XX
!DEF
ASCHEMA ru_fid, { USER | IDB | ICP | ASL }
```

ALTERNATE SCHEMA

XDELTA's USE command "activates" alternate schema use based on domain desired and DEF'd onto ASCHEMA tape

```
UU#.3          " would use USER domain ASCHEMA
UU#.3,ICP      " would use CP (IBEX?) ASCHEMA
UU#.22,ASL     " would use IDS/ARES ASCHEMA
```

Note: ASCHEMA is only selected by domain. Debugging OUTSYM with KEYIN's ASCHEMA won't work.

SYSTEM OVERHEAD FUNCTIONS

MACHINE CONTEXT IN VARIOUS STATES:

CONTEXT	USER RUNNING	MONITOR SERVICE	MONITOR RUNNING
LSR	USER LS IN USER HJIT (PER USER)	MONITOR LS → IN MON HJIT (PER CPU)	
PSR/ASR	USER PS/AS → IN USER HJIT		NOT USED
SSR	USER SAFE → STONE IN USER HJIT		MONITOR SS IN MON HJIT
MASTER/ SLAVE	SLAVE	MASTER → (PRIV)	
WSR7	8	~~0~~ 8	0

MULTIPROCESSING

- USERS RUN STRICTLY IN USER CONTEXT – ANY CPU
- MOST MONITOR SERVICES RUN IN USER CONTEXT – ANY CPU
- FAULT HANDLER RUNS IN USER/CPU CONTEXT – ANY CPU

MULTIPROCESSING (CONT)

- SCHEDULER/PHYSICAL I/O – ANY CPU
- I/O INTERRUPT – MASTER ONLY

THE MONITOR DOMAIN:

- INSTRUCTION SEGMENT DESCRIPTOR LOCATES MONITOR PROCEDURE AND STATIC DATA IN THE MONITOR WS

- DATA SEGMENT DESCRIPTORS LOCATE MONITOR DYNAMIC DATA

- JIT, FILE BUFFERS, READ ONLY SEGMENT, HJIT LOCATES THESE AREAS IN ALL USERS, BUT WITH READ/WRITE ACCESS

THE MONITOR DOMAIN: (CONT)

- OTHER SPECIAL PURPOSE DESCRIPTORS ALSO PRESENT

MONITOR EXECUTION MODES:

- MONITOR SERVICE (EXECUTING PMMES)
    - RUNNING ON BEHALF OF USER
    - TIME CHARGED TO USER
    - RUNS IN USER CONTEXT

MONITOR EXECUTION MODES: (CONT)

- MONITOR EXECUTION (SCHEDULING, SERVICING INTERRUPTS, ETC.)
  - NOT RUNNING FOR ANY USER
  - TIME CHARGED TO SYSTEM OVERHEAD
  - RUNS IN CPU CONTEXT

TYPES OF MONITOR DATA:

- USER SPECIFIC (JIT, HJIT, PAGE TABLE, FILE BUFFERS, USER TSTACK)
- CPU SPECIFIC (MONITOR, JIT, HJIT, PAGE TABLE, PAGE TABLE DIRECTORY, PART OF INSTRUCTION SEGMENT DATA)
- CPU GLOBAL
  - PART OF INSTRUCTION SEGMENT DATA

TYPES OF MONITOR DATA: (CONT)

- REAL (BUILT BY TIGR BASED ON CONFIGURATION NEEDS)
  - REA~~D~~L MEMORY AVAILABILITY LIST (PPUT)
  - DEVICE AND CHANNEL CONTROL TABLES

TYPES OF MONITOR DATA: (CONT)

- AND QUEUE BLOCKS
- I/O CACHE CONTROL TABLES
- RESOURCE TABLES

TYPES OF MONITOR DATA: (CONT)

  - SHARED PROGRAM TABLES

- DYNAMIC SEGMENTS
  - CFUS
  - ENQ/DEQ TABLES

TYPES OF MONITOR DATA: (CONT)

- USER TABLES
- DYNAMIC REAL PAGES
- I/O CACHE

SCHEDULER:

- EVENT DRIVEN, PRIORITIZED QUEUE SCHEDULING
- ALL TYPES OF JOBS SAME EXCEPT FOR PRIORITY
- CONTROLS TO PREVENT EXCESS SCHEDULING

SCHEDULER: (CONT)

- MOST (VOLUME) SERVICES RUN ON ANY CPU
- PRIORITY INCREMENTS FOR CERTAIN EVENTS

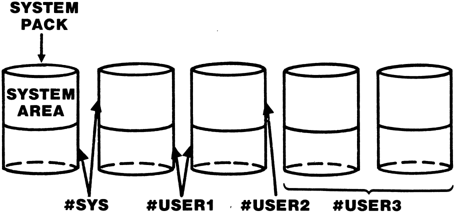
MONITOR SERVICES

FILE MANAGEMENT STRUCTURE

OVERVIEW

7-9 INT 8/85

PACKS AND PACK SETS



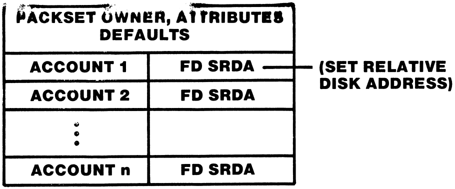
- **EACH VOLUME CONTAINS VID (Volume IDentification)**

MAD (Master Account Directory)

PUBLIC ACCOUNT 1	CONTAINING PACKSET
PUBLIC ACCOUNT 2	CONTAINING PACKSET
⋮	
PUBLIC ACCOUNT n	CONTAINING PACKSET

- MAD CONTAINED ON #SYS
- INDICATES WHERE ACCOUNT MAY BE
- MAINTAINED BY M$MADMUCK, USUALLY VIA PIG AS RESULT OF PUBL KEYIN

PAD (Packset Account Directory)



PACKSET OWNER, ATTRIBUTES DEFAULTS	
ACCOUNT 1	FD SRDA
ACCOUNT 2	FD SRDA
• • •	
ACCOUNT n	FD SRDA

(SET RELATIVE DISK ADDRESS)

- RESIDES ON APPROPRIATE PACKSET
- ACCOUNTS MAY BE ADDED BY PIG, OR DYNAMICALLY IF ATTRIBUTES ALLOW

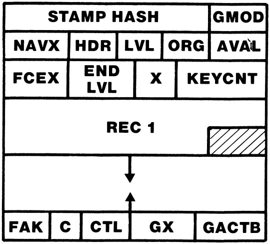
FD (File Directory)

<table>
<tr><td colspan="3">ACCOUNT OWNER, ATTRIBUTES, DEFAULTS</td></tr>
<tr><td>FILE 1</td><td>FIT SRDA</td><td>FLAGS</td></tr>
<tr><td>FILE 2</td><td>FIT SRDA</td><td>FLAGS</td></tr>
<tr><td>•
•
•</td><td></td><td></td></tr>
<tr><td>FILE n</td><td>FIT SRDA</td><td>FLAGS</td></tr>
</table>

FIT (File Information Table)

ACCESS CONTROLS
ORG, NRECS, UGRANS, GAVAL
DATES (OPEN UPDATE INDICATOR)
UATTR, INSTATTR
EXTEN~~D~~T LIST
TDA (Top Disk Address) FDA (First Disk Address) LDA (Last Disk Address) ALL ARE FRDA (File Relative Disk Address)
GRANULE STAMP HASH

CONSECUTIVE, UR, SYMBIONT FILE STRUCTURE



STAMP HASH | GMOD

NAVX | HDR | LVL | ORG | AVAIL

FCEX | END LVL | X | KEYCNT

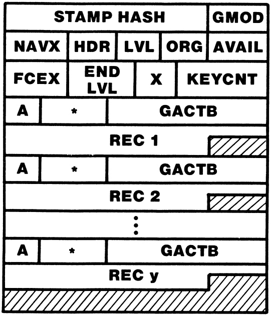
REC 1

FAK | C | CTL | GX | GACTB

GRANULE NUMBER MODULO 512

- RECORD SPANNING FOR CONSEC, UR

RELATIVE FILE STRUCTURE



STAMP HASH | GMOD

NAVX | HDR | LVL | ORG | AVAIL

FCEX | END LVL | X | KEYCNT

A | * | GACTB

REC 1

A | * | GACTB

REC 2

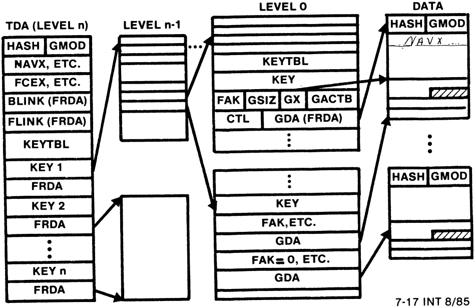
⋮

A | * | GACTB

REC y

- NO RECORD SPANNING
- GRANULE NUMBER CALCULATED FROM RECORD NUMBER

KEYED (INDEXED) FILE STRUCTURE



7-17 INT 8/85

FILE MANAGEMENT SERVICE CATEGORIES

- DCB CONTROL
- RECORD MANIPULATION
- DEVICE CONTROL
- PRIVILEGED OPERATIONS

RESOURCE MANAGEMENT SERVICES

- RESOURCE TYPES
  - PHYSICAL RESOURCES
  - POOLED RESOURCES
  - PSEUDO RESOURCES
- RESOURCE MANAGEMENT SERVICES
- DIFFERENT TREATMENT AND LIMITS BY MODE

MEMORY MANAGEMENT SERVICE CATEGORIES

- ACQUIRE AND RELEASE MEMORY
- MANIPULATE PAGE TABLE AND LINKAGE SEGMENT/ARGUMENT SEGMENT
- PRIVILEGED AND T & D SERVICES

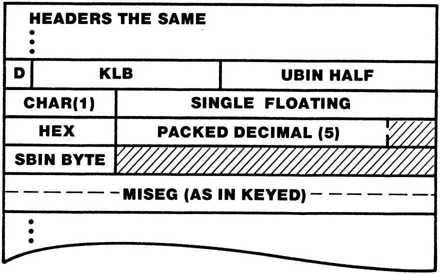
EXECUTION CONTROL SERVICE CATEGORIES

- PROGRAM FLOW CONTROL SERVICES
- EXCEPTION CONDITION CONTROL SERVICES
- EXCEPTION CONDITION HANDLING SERVICES

EXECUTION CONTROL SERVICE CATEGORIES (CONT)

- PRIVILEGED/COMMAND PROGRAM SERVICES

IREL KEY STRUCTURE



HEADERS THE SAME

•
•
•

D | KLB | UBIN HALF

CHAR(1) | SINGLE FLOATING

HEX | PACKED DECIMAL (5) ▨

SBIN BYTE ▨

– – – – – – MISEG (AS IN KEYED) – – – – – – –

•
•
•

IO CACHE

- SET ASSOCIATIVE (4 SLOTS/SET)
- HASH TO SET, SEARCH
- MANY CONTROLS AND STATS
- INTEGRATED WITH AUTOSHARE AND CG MEM

GRANULE TYPES

- MAD, PAD, GP, FD, FIT, ~~VL~~, 4L
  INDEX, DATA, REL, CONSEC, ELSE

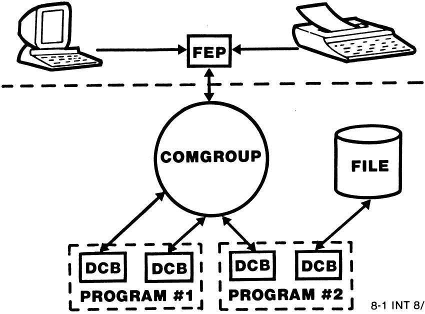
- CONTROLS AND STATS BY GRANULE TYPE

- STATS/ANLZ GIVE INFO

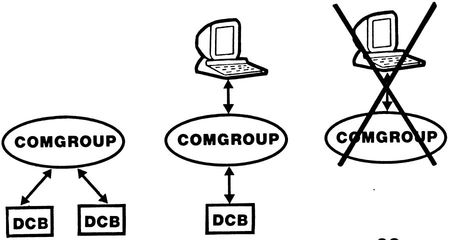
CACHE TABLE

<table>
<tr><td colspan="1">TYPE</td><td colspan="7">SRDA</td></tr>
<tr><td colspan="2">FLINK</td><td colspan="6">BLINK</td></tr>
<tr><td colspan="2">PAGE</td><td colspan="6">USECNT</td></tr>
<tr><td colspan="8">UPCOUNT</td></tr>
<tr><td colspan="2">SETX</td><td>USER</td><td>*</td><td>I</td><td>W</td><td>E</td><td>B</td></tr>
<tr><td colspan="8">AGE</td></tr>
</table>

COMMUNICATION GROUPS



CONNECTIONS TO COMGROUPS



COMGROUP

DCB DCB

CG
WITH ONLY
DCBs

COMGROUP

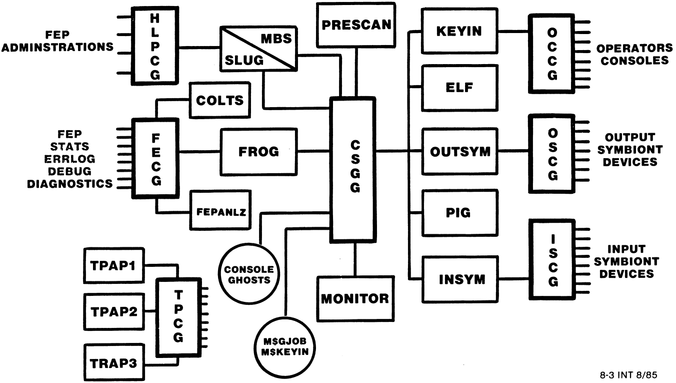
DCB

CG
WITH DCBs
AND
STATIONS

COMGROUP

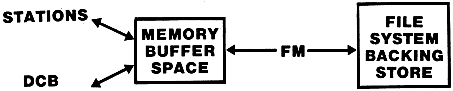
CG
WITH ONLY
STATIONS
(NO SUCH THING)

COMGROUPS INSIDE CP-6



8-3 INT 8/85

OPEN COMGROUP



STATIONS

DCB

MEMORY BUFFER SPACE

FM

FILE SYSTEM BACKING STORE

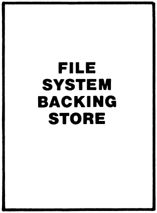
DISK FILE:

- M$OPEN
- ACCESS CONTROLS
- BACKUP
- OVERFLOW
- PACKAGING

HOST MEMORY:

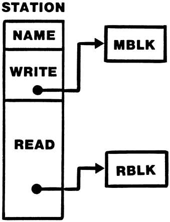
- M$OPEN
- MESSAGES
- TERMINALS (ACTIVE COMGROUP ONLY)

CLOSED COMGROUP



FILE
SYSTEM
BACKING
STORE

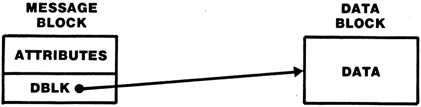
STATION I/O



READ: MESSAGE TYPE
ORIGIN STATION

WRITE: MESSAGE TYPE
DESTINATION STATION
(DIRECT OR ANONYMOUS
QUEUE)

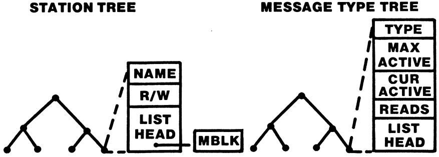
COMGROUP MESSAGE



- SMALL
- IDENTIFY BY MEMORY ADDRESS
- ATTRIBUTES
  - TYPE
  - ORIGIN STATION
  - DESTINATION (STATION/QUEUE)
  - PRIORITY
  - MESSAGE ID

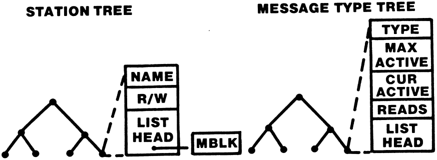
- ARBITRARY SIZE
- IDENTIFIED BY DISK ADDRESS - CACHED
- DATA PART OF MESSAGE

QUEUE ORGANIZATION



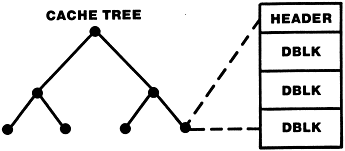
- BINARY TREE BY STATION NAME
  - FAST ACCESS FOR DIRECT WRITES
- STATION NODE
  - CONTROLS STATION I/O
  - CONTAINS LIST OF MESSAGES TO THIS STATION
- ANONYMOUS QUEUE

QUEUE ORGANIZATION



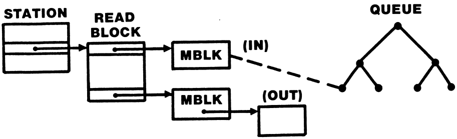
- BINARY TREE BY MESSAGE TYPE
  - FAST ACCESS FOR READS/QUEUE WRITES
- MESSAGE TYPE NODE
  - MAXIMUM AND CURRENT ACTIVE
  - LIST OF READS
  - LIST OF MESSAGES OF THIS TYPE
- TOTAL SIZE LIMITED BY AU
- AUTOMATIC SPILL TO DISK

DISK CACHE



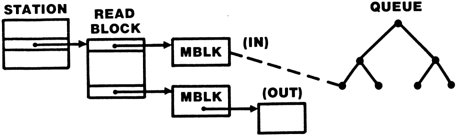
- BINARY TREE BY DISK ADDRESS
  - FAST ACCESS
- NODE IS UNIT OF DISK I/O: MEMORY PAGE
  - CONTAINS MULTIPLE DATA BLOCKS
- SIZE LIMITED BY AU
- GROWS AND SHRINKS (TO DISK)
- CHARGE COMGROUP OWNER

READ AND LATCH



- SPECIFY TYPE (AND ORIGIN)
- LOOK UP IN TREE
- PICK OUT MESSAGE/LEAVE READ PENDING
- REMEMBER MESSAGE 'CURRENT LATCHED INPUT'
- CURRENT # ACTIVE OF TYPE
- MUST SUCCEED
  - REREAD
  - UNLATCH HOLD/RERUN

READ AND LATCH



- LATCHED OUTPUTS
  - MESSAGE ID 'SPAWN' TRAIL
  - INVISIBLE
  - REPORTS/NEW TRANSACTIONS
- NEXT READ TRIGGERS UNLATCH
  - DELETE INPUT
  - SEND OUTPUTS

I/O FEATURES

- WILD-CARDED WRITE DESTINATION
  - BROADCAST
  - FIRST-FOUND
- WILD-CARDED READ 'KEYS'
  - MESSAGE TYPE
  - ORIGIN STATION
- READ: DIRECT ONLY/QUEUE
- WRITE DIRECT TO READER BUFFER

I/O FEATURES (CONT)

- LATCH/SECURE: BUFFERS THRU DATA BLOCKS
- WRITE CONTINUED MESSAGES
  - READER MAY IGNORE SEGMENTING OR PAGE THRU SEGMENTS
- MESSAGE PRIORITY FUNCTION OF TYPE AND ORIGIN STATION
- READ: WAIT/AVAILABLE/ONE ONLY

ADMINISTRATIVE USER FUNCTIONS

CONTROL

- ACTIVATE/DEACTIVATE STATION
  - AU CONNECT/DISCONNECT
  - FLUSH
- EXTEND DISK FILE
- REDIRECT/DELETE MESSAGE

ADMINISTRATIVE USER FUNCTIONS (CONT)

- SET CONTROL PARAMETERS
  - READ/WRITE ABSENT STATION
  - SECURE
  - MEMORY LIMITS
  - MAXIMUM MESSAGE SIZE
  - INPUT/OUTPUT LEGAL FOR TERMINALS
- SET LIST OF MESSAGE TYPES
  - PRIORITY
  - MAXIMUM ACTIVE

ADMINISTRATIVE USER FUNCTIONS (CONT)

- SET LIST OF STATIONS
  - PRIORITY

ADMINISTRATIVE USER FUNCTIONS (CONT)

INFORMATION

- CONNECT/DISCONNECT MESSAGE
- FREE SPACE WARNING MESSAGE
- "LATCH ABORT" MESSAGES
- RETRIEVE CONTROL PARAMETERS

ADMINISTRATIVE USER FUNCTIONS (CONT)

- STATISTICS
  - QUEUE DEPTH
  - TRANSACTION RATE
  - DISK CACHE

- CURRENT MESSAGE TYPES
  - QUEUE DEPTH
  - STATISTICS

ADMINISTRATIVE USER FUNCTIONS (CONT)

- CURRENT STATIONS
  - DEPTH OF DIRECTED MESSAGES
  - STATISTICS
  - PRESENT ABSENT

COMGROUP MONITOR SERVICES

M$OPEN
M$CLOSE
M$READ
M$WRITE
M$UNLATCH

M$ACTIVATE
M$DEACTIVATE
M$CGCTL
M$CGINFO
M$FWCG

WALKING TOURS OF SOME CP-6

FUNCTIONS

CREATING A TIME SHARING USER

- LINE CONNECTS, HANDLER INFORMS NODE ADMINISTRATOR (FEP)

- NODE ADMINISTRATOR CONVERSES WITH TERMINAL (IF NECESSARY) TO ACQUIRE LOGON STRING

- NODE ADMINISTRATOR SENDS LOGON STRING AND END POINT ADDRESS (ON ADMIN PATH) TO SLUG IN THE HOST

CREATING A TIME SHARING USER (CONT)

- SLUG CONSULTS :HLP TO DETERMINE IF LOGON STRING IS AUTHORIZED

- IF INVALID, SLUG RESPONDS ACCORDINGLY TO NODE ADMIN, AND HOST HAS NO FURTHER ACTION IN THIS LOGON ATTEMPT

- IF VALID, SLUG DETERMINES THE KIND OF CONNECTION DEFINED BY THIS LOGON LOGON (IN THIS CASE TIME SHARING)

CREATING A TIME SHARING USER (CONT)

- SLUG CHECKS TO INSURE MAX ONLINE USERS NOT EXCEEDED, AND MAX TOTAL USERS NOT EXCEEDED
- SLUG RESPONDS TO NODE ADMIN WITH THE HOST END POINT ADDRESS (LDCT)
- NODE ADMIN COMPLETES THE PATH AND RELINQUISHES CONTROL OF THE END POINT

CREATING A TIME SHARING USER (CONT)

- SLUG STEALS THREE PHYSICAL PAGES (CAN FAIL)

- SLUG BUILDS SKELETON JIT, HJIT, AND ROSEG, INSERTS NAME AND ACCOUNT INTO JIT AND LDCT INTO M$UC (ROSEG)

- SLUG EXECUTES CALL M$MAKUSER (CAN FAIL)

CREATING A TIME SHARING USER (CONT)

- M$MAKUSER ACQUIRES ADDITIONAL PHYSICAL PAGE FOR PAGE TABLE, INITIALIZES IT WITH THE FOUR USER CONTEXT PAGES, INITIALIZES TSTACK, REPORTS SCHEDULER EVENT E_AU (ADD USER)

CREATING A TIME SHARING USER (CONT)

- SCHEDULER GETS AVAILABLE USER TABLE SLOTS (SS_NULL), ASSIGNS SYSID, PUTS USER NUMBER AND SYSID INTO JIT, INITIALIZES USER TABLE SLOT (PAGE TABLE, ETC), AND PUTS NEW USER INTO EXECUTABLE STATE

CREATING A TIME SHARING USER (CONT)

- WHEN SCHEDULER PUTS NEW USER INTO EXECUTION, IT NOTICES THAT NO COMMAND PROGRAM IS ASSOCIATED AND INVOKES LOGON

- LOGON DETERMINES IF USERS SUSPENDED, AND CONVERSES ABOUT WHAT TO DO

CREATING A TIME SHARING USER (CONT)

- LOGON INITIALIZES JIT (FROM :USERS) WITH RESOURCES, LIMITS, DEFAULTS, SETUP (TO CCBUF)
- LOGON EXECUTES M$CPEXIT TO ASSOCIATE COMMAND PROGRAM SPECIFIED IN :USERS

CREATING A TIME SHARING USER (CONT)

- WHEN (IF) IBEX GETS CONTROL, IT EXECUTES COMMAND IN CCBUF
- PHEW!

THE TRIP A TRANSACTION TAKES THROUGH TP

- FPL PROGRAM DETERMINES IT HAS A COMPLETE TRANSACTION, AND SEND IT
- HOST RECEIVES TRANSACTION AND INSERTS IT INTO COMGROUP

THE TRIP A TRANSACTION TAKES THROUGH TP (CONT)

- IF TRANSACTION TYPE SPECIFIES JOURNAL, TRANSACTION DIRECTED TO JAYS, ELSE TRANSACTION PUT ON ANONYMOUS QUEUE

- TRANSACTION TYPE SATISFIES THE READ OF TPCP ASSOCIATED WITH SOME TPU FOR THE TP INSTANCE, OR THE READ OF A TPAP ASSOCIATED...

THE TRIP A TRANSACTION TAKES THROUGH TP (CONT)

- IF TPCP RECEIVES THE TRANSACTION, IT DETERMINES THE APPROPRIATE TPAP, AND EXECUTES M$CPEXIT TO FETCH IT

- TPAP PROCESSES THE TRANSACTION, POSSIBLY SENDS RESPONSES/REPORTS, AND EVENTUALLY EXECUTES M$READ (OR M$UNLATCH), REMOVING THE TRANSACTION FROM THE COMGROUP

THE TRIP A TRANSACTION TAKES THROUGH TP (CONT)

- IF THE TRANSACTION TYPE REQUIRES JOURNAL, JAYS IS NOTIFIED TO WRITE 'END TRANSACTION' RECORD TO JOURNAL

TIME SHARING USER HITS BREAK

- HANDLER DETECTS BREAK AND DOES APPROPRIATE LOCAL ACTION (CANCEL INPUT/OUTPUT)

- HANDLER SENDS BREAK MESSAGE ON THE PATH TO HOST

- HOST (FRONT END INTERFACE) SEES BREAK MESSAGE AND REPORTS E_BRK TO SCHEDULER

TIME SHARING USER HITS BREAK (CONT)

- SCHEDULER TAKES ONE OF TWO TYPES OF ACTION, DEPENDING ON CURRENT STATE OF USER. IF 'BREAKABLE' STATE (E.G., SLEEP, ENQUEUE WAIT, ETC) STATE IS CHANGED TO EXECUTABLE (SS_n), ELSE NO STATE CHANGE OCCURS. IN EITHER CASE, BREAK FLAG IS SET IN USER TABLE ENTRY

TIME SHARING USER HITS BREAK (CONT)

- WHEN USER IS NEXT SCHEDULED, IF BREAK FLAG IS SET, SCHEDULER ALTRETS FROM REG. CALLER OF REG SETS IC TO REEXECUTE PMME IF APPROPRIATE

- ON NEXT EXIT FROM MONITOR TO USER, BREAK FLAG IS NOTICED. IF SET, APPROPRIATE ACTION IS TAKEN.

TIME SHARING USER HITS BREAK (CONT)

- IF DEBUGGER HAS REQUESTED BREAK CONTROL, CONTROL GOES TO DEBUGGER
- IF USER HAS REQUESTED BREAK CONTROL, CONTROL GOES TO USERS BREAK ROUTINE
- OTHERWISE CONTROL GOES TO COMMAND PROGRAM

I/O COMPLETE EVEN ON NOWAIT I/O FOR USER

- I/O INTERRUPT OCCURS, HANDLER GET CONTROL TO DO CLEANUP, NIO$COMP IS CALLED TO SIGNAL OPERATION COMPLETED
- IF I/O ASSOCIATED WITH A DCB, DCB FUNCTION COUNT IS DECREMENTED

I/O COMPLETE EVENT ON NOWAIT I/O FOR USER (CONT)

- IF EVENT WAS REQUESTED, SCHEDULER IS CALLED TO REPORT USER EVENT. SCHEDULER SETS USER TABLE FLAG INDICATING CACHE CLEAR IS REQUIRED

- E_IOC IS REPORTED TO SCHEDULER (TO MAINTAIN MF)

I/O COMPLETE EVENT ON NOWAIT I/O FOR USER (CONT)

- ON NEXT EXIT FROM MONITOR TO USER, USER EVENT REQUESTS IS NOTED, CACHE CLEAR FLAG IS NOTED AND HONORED, AND USER EVENT ROUTINE IS ENTERED

- IF NO EVENT WAS REQUESTED, USER MUST EXECUTE M$CHECK TO DETERMINE I/O COMPLETION (AND GET CACHE CLEARED)

RUNNING DOWN A USER (STEP)

- USER PROGRAM DOES M$EXIT ~~ON~~ OR ERRORS OR ABORTS FOLLOWED BY M$CPEXIT (QUIT)
- ALL DCBS (DCBNUM>=10 ARE CLOSED
- CURRENT SHARED PROGRAM, SHARED LIBRARY, ASL, DEBUGGER (IF ANY) ARE DISASSOCIATED AFTER GETTING EXIT CONTROL IF REQUESTED

RUNNING DOWN A USER (STEP)
(CONT)

- ALL MEMORY EXCEPT CONTEXT IS RELEASED

- IF PROPRIETARY ACCOUNTING REQUIRED M$ACCT CALLED

- IF STEP ACCOUNTING REQUIRED, M$ACCT CALLED

RUNNING DOWN A USER (STEP)
(CONT)

- IF CP__LOGOFF SET IN JIT, USER IS LOGGED OFF (NEXT SLIDE)

RUNNING DOWN A USER (LOGOFF)

- ~~COMMAND PROGRAM DECIDES TO LOG OFF USER, ISSUES M$CPEXIT TO LOGON~~
- STEP RUNDOWN IS PERFORMED (IF NOT ALREADY DONE)
- ~~IBEX LOGON IS ASSOCIATED, DOES FINAL ACCOUNTING FUNCTIONS~~
- IBEX ~~LOGON~~ ISSUES M$CPEXIT (OFF)

RUNNING DOWN A USER (LOGOFF) (CONT)

- DISCONNECT RECORD IS SENT TO FEP (IF TS)
- MBS IS NOTIFIED (IF BATCH)
- E_OFF EVENT IS REPORTED TO SCHEDULER
- SCHEDULER RELEASES 4 CONTEXT PAGES

RUNNING DOWN A USER (LOGOFF)
(CONT)

- SCHEDULER CHANGES USER STATE TO SS_NULL
- USER NOT LONGER EXISTS
- FEP EITHER DROPS LINE OR REISSUES SALUATATION

READ A RECORD FROM A T/S TERMINAL

- ASSUMPTIONS: NO TYPEAHEAD WAITING, NON-TRANSPARENT, VANILLA READ

- USER ISSUES M$READ

- KI MODULES OF MONITOR GET ENTERED FROM PMME, READ SENT TO FEI

READ A RECORD FROM A T/S TERMINAL (CONT)

- FEI PUTS READ REQUESTS INTO CIRCULAR QUEUE AND REGS USER (STI) – SIZE OF READ, DOMAIN, REREAD?, PATH ID (FROM LDCT ENTRY FROM M$~~US~~)
  UC

- READ REQUEST ARRIVES IN DESTINATION FEP (MAYBE VIA X.25 TO REMOTE

- READ REQUEST DELIVERED TO VDH, SCHEDULED ON BEHALF OF THIS PATH CONTEXT (TERMINAL)

READ A RECORD FROM A T/S TERMINAL (CONT)

- IF PROMPT FOR THIS DOMAIN NO NULL, TRANSLATE PROMPT (FROM CONTEXT) AND WRITE TO TERMINAL

- ALLOCATE 16 WORD (32 CHAR) INPUT BUFFER

- AS CHARACTERS ENTERED, IMP THEM, ECHO THEM, TRANSLATE THEM, AND INSERT INTO INPUT BUFFER

READ A RECORD FROM A T/S TERMINAL (CONT)

- IF INPUT BUFFER FULL, GET ONE TWICE AS BIG, MOVE EVERYTHING, CONTINUE

- UPON ACTIVATION (EOM CHAR/COUNT/TIMEOUT) STOP ECHOING CHARS, ANY MORE TYPED GO INTO TYPEAHEAD BUFFERS

- READ RESPONSE MSG SEND BACK TOWARDS HOST END (CONTAINS PTR TO INPUT BUFFER)

READ A RECORD FROM A T/S TERMINAL (CONT)

- HOST END HANDLER IN FEP PUTS READ RESPONSE WITH DATA INTO INPUT CIRCULAR QUEUE

- FEI RECEIVES READ RESPONSE IN ICQ, GETS CONTROL VIA I/O OR TIMER INTERRUPT AND USING CONTEXT IN LDCT ENTRY DELIVERS DATA TO USERS BUFFER, SETS ARE DCB

READ A RECORD FROM A T/S TERMINAL (CONT)

- FEI CALL SCHEDULER TO REPORT TERMINAL INPUT COMPLETE
- SCHEDULER SET USER TO COMPUTE STATE (BASE PRIORITY + TIC INCREMENT)
- KI GETS CONTROL WHEN USER SCHEDULED AND RETURNS TO USER

OPEN A FILE

- USER ISSUES M$OPEN, FM MODULES ENTERED VIA PMME
- MERGE OPEN PARAMETERS INTO DCB (MAYBE FMA FROM JIT)
- CHECK CPUS FOR THIS FILE ALREADY OPEN/RECENTLY OPENED. IF FOUND SKIP DIRECTORY SEARCH, SRDA OF FIT IN CFU

OPEN A FILE (CONT)

- DIRECTORY SEARCH

  FPOOL AREA USED TO MAP IN CACHE DIRECTORY GRANS OR TO READ THEM IN FROM DISK IF NECESSARY

  SEARCH MAD FOR ACCOUNT NAME IF PACKSET NOT SPECIFIED IN M$OPEN

OPEN A FILE (CONT)

SEARCH PAD FOR SRDA OF FILE ACCOUNT DIRECTORY FOR THIS ACCOUNT NAME

SEARCH FAD FOR SDRA OF FIT FOR THIS FILE NAME

OPEN A FILE (CONT)

- READ FIT INTO FPOOL PAGE
- CHECK USER ACCOUNT NAME AGAINST ACCESS CONTROL LIST, IF FAILS TRY OTHER TRICKS (ACCESS VEHICLE, FMREAD/FMSEC PRIV)
- ALLOCATE CFU ENTRY IF NOT ALREADY THERE, FILL IT IN

OPEN A FILE (CONT)

- MOVE INFO FROM FIT INTO DCB,
  MARK DCB OPEN
- MOVE FPARMS TO USER IF REQUESTED
  ON M$OPEN
- RETURN CONTROL TO USER

READ A RECORD FROM A (KEYED) FILE

- IS DCB OPEN? NO GO BACK ONE CHART, DO OPEN, RETURN HERE

- ASSUME ONE GRANULE FOR FIT, ONE GRANULE FOR KEYS, DATA ON SEPARATE GRANULES

- ALLOCATE FPOOL FOR INDEX GRANULE (TRUNIC OTHER IS NECESSARY)

READ A RECORD FROM A (KEYED) FILE (CONT)

- READ INDEX GRANULE (FRDA IN FIT)
- SEARCH FOR KEY (BIN SEARCH)
- FROM KEY GET FRDA OF DATA GRANULE(S), WORD OFFSET, BYTE COUNT OF RECORD
- ALLOCATE FPOOL FOR DATA GRANULE

READ A RECORD FROM A (KEYED) FILE (CONT)

- READ IN DATA GRANULE
- MOVE DATA FROM GRANULE TO USER BUFFER
- REPEAT ABOVE FOR EACH CONTINUED CHUNK OF RECORD

READ A RECORD FROM A (KEYED) FILE (CONT)

- SET ARS IN DCB
- RETURN TO USER

CONNECT A DCB TO A (CLOSED) COMGROUP

- USING FM ROUTINES OPEN THE COMGROUP FILE
- GET A PAGE FOR CB CONTEXT BLOCK
- GET A PAGE FOR DESCRIPTOR SEGMENT BLOCK

CONNECT A DCB TO A (CLOSED) COMGROUP (CONT)

- READ CG CONTEXT BLOCK FROM GRAN 0 OF FILE

- USING CGCTX ALLOCATE APPROPRIATE DESCRIPTORS, GET THE RIGHT PAGES IN THE 10K CHUNKS FO CGWS, AND READ IN THE GRANULES OF THE SAVED "ACTIVE" IMAGE INTO THE APPROPRIATE PAGES.

CONNECT A DCB TO A (CLOSED) COMGROUP (CONT)

- START CLOCK
- RUN AROUND MAKING SURE THINGS ARE O.K. (RECOVERY DIDN'T CLOSE RIGHT?)
- INSERT NODE INTO "EMPTY" STATION FOR THIS DCB STATION
- FILL IN NODE WITH STATION NAME, NOTE IF THIS IS AU IN NODE AND IN CGCTX

CONNECT A DCB TO A (CLOSED) COMGROUP (CONT)

- RETURN CONTROL TO USER

CONNECT A TERMINAL TO A COMGROUP

- SLUG RECEIVES LOGON STRING AS MESSAGE ON HLPCG FROM FEI WHICH LOOKS LIKE A TERMINAL STATION ON HLPCG FOR EACH FEP

- LOGON STRING KEYED INTO :HLP FILE FOR TERMINAL INFO:

  DEFAULT PROFILE (IT OR SPECIFIED PROFILE SENT TO FEP FOR THIS PATH)

CONNECT A TERMINAL TO A COMGROUP (CONT)

STATION NAME

COMGROUP TO CONNECT TO

- SLUG ISSUES M$TRMCON TO COMGROUP CODE IN MONITOR (KQ)
- KQ MODULES FIND CONTEXT BLOCK FOR THIS COMGROUP - IF NOT THERE ALTRET (*NOCG)

CONNECT A TERMINAL TO A COMGROUP (CONT)

- FROM COMGROUP CONTEXT BLOCK ALL IS KNOWN ABOUT LOCATING COMGROUP

- STATION MODE INSERTED IN CG IF ITS NOT THERE

CONNECT A TERMINAL TO A COMGROUP (CONT)

IF NODE ALREADY THERE (NOT CONNECTED, WITH MSGS WAITING) THEN IF CGCTL ALLOWS ACTIVATION, START SENDING QUEUED MSG'S TO TERMINAL

- IF AU CONNECTED BUILD *AUEV MSG, DELIVER TO AU STATION

DATA NET 8 (L6) ARCHITECTURE

GENERAL ARCHITECTURE

- 16 BIT MINI-COMPUTER
- 7-16 BIT GENERAL REGISTER
- 7-20 BIT BASE REGISTER
- COMMERICAL INSTRUCTION PROCESSOR (CIP)

MEMORY MANAGEMENT

- 1 MW VIRTUAL ADDRESS SPACE
- 16 4KW SEGMENTS
- 15 64KW SEGMENTS
- R,W,E PROTECTION
- SEGMENT MUST BE CONTIGUOUS REAL

MEMORY ARCHITECTURE

- SEGMENT DESCRIPTOR

  BASE, SIZE (MOD 256 WORDS)

  VALID BIT

  RING PROTECTION R,W,E

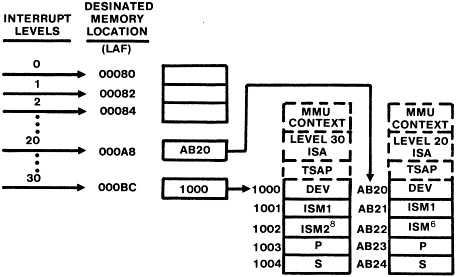
MEMORY ARCHITECTURE (CONT)

- ASD
- TRAP, INTERRUPT

INTERRUPTS

- 64 LEVELS (0-63)
  0,1,2 RESERVED
- CONTEXT SWITCH CONTROLLED BY MASK
- LEV INSTRUCTION

INTERRUPT SAVE AREA

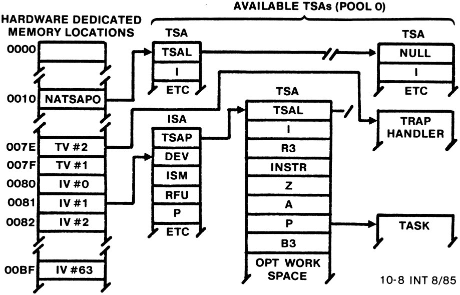


TRAPS

- TRAP VECTOR IN REAL MEMORY
- CONTEXT STORED IN TSA

  TSA'S IN FOUR POOLS
  PARTIAL CONTEXT AUTOMATIC
- TSA LINKED TO CURRENT INTERRUPT
- MCL IS A TRAP

TRAP VECTOR AND INTERRUPT VECTOR LINKAGE



10-8 INT 8/85

LCP–6 OPERATING SYSTEM FEATURES

- GENERAL FEATURES
- USE OF MEMORY MANAGEMENT
- USE OF INTERRUPTS
- KINDS OF USERS
- DEBUGGING

VIRTUAL MEMORY ALLOCATION

- 1 MEG VIRTUAL SPACE LIKE WS ON HOST

  ONLY ONE WS VISIBLE

- NO PAGE MAP
  SHUFFLING
  I/O MEMORY
  FRAGMENTATION

- ADDRESS SPACE DEFINED BY ASDT IN HJIT

```
16:11 AUG 09 '85 FEP_SEGS


    ACTIVE PROCESS ------->   USER/HANDLER        MON FOR USER       MONITOR
    MMU IMAGE SOURCE ---->    UHJIT.ASDT_USR      UHJIT.ASDT_MCL     MHJIT.ASDT_MON
 #   UASDT   MASDT   VADDR
--   -----   -----   ------   ---------------     ---------------    --------------
00   .5000   .503E   .00000   NULLSEG             NULLSEG            NULLSEG
01   .5002   .5040   .01000   ROS                 ROS                ROS
02   .5004   .5042   .02000   DB_DS               RDB_DS             RDB_DS
03   .5006   .5044   .03000   LOW_MEM             LOW_MEM            LOW_MEM
04   .5008   .5046   .04000   TSTACKU             TSTACKU            TSTACKM
05   .500A   .5048   .05000   UHJIT               UHJIT              UMHJIT
06   .500C   .504A   .06000   MHJIT               MHJIT              MHJIT
07   .500E   .504C   .07000   MHJIT               MHJIT              MHJIT
08   .5010   .504E   .08000   MON_ENTRY_DATA      MON_ENTRY_DATA     MON_ENTRY_DATA
09   .5012   .5050   .09000   MON_ENTRY           MON_ENTRY          MON_ENTRY
10   .5014   .5052   .0A000   USER_DS1            LPAR1              *
11   .5016   .5054   .0B000   USER_DS2            LPAR2              *
12   .5018   .5056   .0C000   CP_DS               LPAR3              *
13   .501A   .5058   .0D000   *                   LPAR4              *
14   .501C   .505A   .0E000   *                   LPAR5              *
15   .501E   .505C   .0F000   *                   LPAR6              *

16   .5020   .505E   .10000   USER_IS1            MON_IS1            MON_IS1
17   .5022   .5060   .20000   USER_IS2            MON_IS2            MON_IS2
18   .5024   .5062   .30000   USER_IS3            MON_IS3            MON_IS3
19   .5026   .5064   .40000   USER_IS4            MON_IS4            MON_IS4
20   .5028   .5066   .50000   USER_IS5            MON_IS5            MON_IS5
21   .502A   .5068   .60000   USER_IS6 (LIB)      BPAR1              *
22   .502C   .506A   .70000   USER_IS7 (LIB)      BPAR2              *
23   .502E   .506C   .80000   DB_PROC             DB_PROC            DB_PROC
24   .5030   .506E   .90000   CP_PROC             WINDOW1            WINDOW1
25   .5032   .5070   .A0000   UAUTO_DS            *                  *
26   .5034   .5072   .B0000   USER_DS3            *                  *
27   .5036   .5074   .C0000   USER_DS4            BOBCAT             BOBCAT
28   .5038   .5076   .D0000   HAND_Q              HAND_Q             *
29   .503A   .5078   .E0000   *                   *                  *
30   .503C   .507A   .F0000   *                   BIGFOOT            BIGFOOT
```

SCHEDULING IN LCP-6

- SCHEDULED EXECUTION AT LEVEL 63
  - FPRGS, HANDLERS BASE LEVEL
- SCHEDULER RUNS AT LEVEL 62
- REAL TIME CLOCK AT LEVEL 61

SCHEDULING IN LCP-6
(CONT)

- LEVELS 12-60 AVAILABLE FOR HANDLERS CONNECTED INDIRECTLY VIA M$INTCON
- LEVEL 3 COMMON INHIBIT LEVEL

KINDS OF FEP USERS

- USER FRONT END PROGRAMS
- COMGROUP FRONT END PROGRAMS
- GHOST FRONT END PROGRAMS
- HANDLER FRONT END PROGRAMS

USER FPRG

- IN PATH BETWEEN HOST USER AND DEVICE
- ABORTED WHEN HOST DCB CLOSED
- OPEN RES='UCnn', ORG=FPRG

USER FPRG (CONT)

- USES
  - DATA REDUCTION/CONVERSION
  - SCREEN ORIENTED FUNCTIONS
  - E.G. DIGIS

COMGROUP FPRG

- IN PATH BETWEEN HOST COMGROUP AND TERMINAL STATION
- STARTED/STOPPED BY CG ADMINISTRATOR
  - M$ACTIVE MAKEFPRG=YES
  - M$DEACTIVATE OR CG AU DISCONNECT

COMGROUP FPRG (CONT)

- USES
  - FORMS CONTROL
  - E.G. FPL FOR TP

GHOST FPRG

- CONNECTED TO HOST DCB ONLY
- ABORTED WHEN HOST DCB CLOSED
- OPEN RES='FEnn', ORG=FPRG
- USES
  - E.G. COLTS, PIGETTE, ANLZ

HANDLER FPRG

- STARTED THROUGH HOST DCB, THEN INDEPENDENT
- MAY CONNECT TO DEVICES, LINES
- OPEN RES='FEnn', ORG=HANDLER

HANDLER FPRG (CONT)

- USES
  - DEVICE HANDLERS E.G. ASYNC
  - NODE ADMINISTRATOR
  - GATEWAYS

DEBUGGING FEPS AND FPRGS

- DELTA REPLACES FOX AND FEPANLZ DEBUGGING FUNCTION

- CAN BE USED TO DEBUG

  - USER FPRG (WITH HOST PROGRAM)

  - COMGROUP FPRG (W/O HOST PROGRAM)

DEBUGGING FEPS AND FPRGS (CONT)

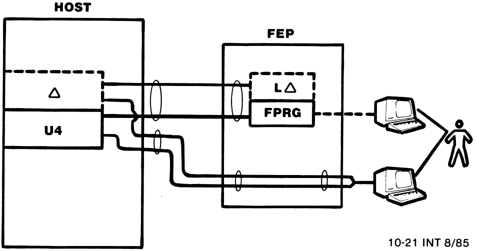
- ENTIRE FEP IN EXECUTIVE MODE VIA SA ASYNC

- FPL PROGRAMS IN CONCERT WITH FPL INTERPRETER

DEBUGGING HOST FPRGS

- AUTOMATIC IF DEBUGGING HOST PROGRAM
- USE UCnn/USE HOST/USE FEnn

DELTA USED TO DEBUG HOST PROGRAM AND FPRG SIMULTANEOUSLY



10-21 INT 8/85

DEBUGGING COMGROUP FPRGS

- DELTA STARTED ONLINE W/O HOST PROGRAM

- DEBUG STATION_NAME AT CG/COMGROUP

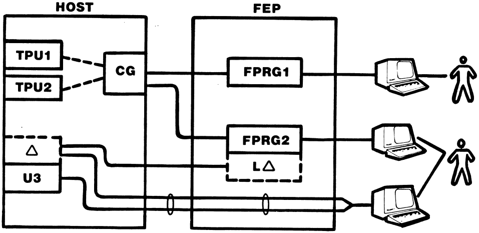
- IF FPRG HAS FPL ASSOCIATED

    - FPL COOPERATES WITH DELTA

DEBUGGING COMGROUP FPRGS (CONT)

- STATEMENT BREAKPOINTS ONLY
- DISPLAY/LET
- NO MODIFY/DUMP

DELTA USED TO DEBUG TP FPRG



10-23 INT 8/85

FEP USERS

AND

SYSTEM COMPONENTS

SYSTEM SERVICE INTERFACE

- MCL IS USED FOR ENTRY
- FPT IS SIMILAR TO HOST
- VECTORS INTERFACED BY SOFTWARE

LCP6 OU/RU

- FORMATS/SCHEMA SAME AS ON HOST
- FEPLINK OFFERS SUBSET OF LINK FEATURES
- DELTA PROCESSES HOST/FEP SCHEMA WITH SAME CODE

LCP6 OU/RU (CONT)

- DELTA HAS SEPARATE CODE FOR MEMORY REFERENCE, INSTRUCTION INTERPRETATION

CALLING SEQUENCE

- SAME PRINCIPLE AS ON HOST
- LIMITED ARGUMENT DESCRIPTORS
- NO RUNTIME CHECKING

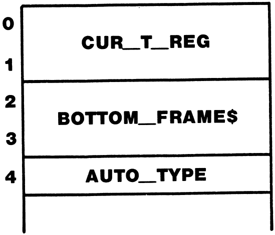
CALL

```
[LAB, B3   ptrs  ]
 LAB, B4   descs
 LNJ, B6   sub
 DC        altret, PREL (i if none)
```

RECEIVING SEQUENCE

```
LNJ, B5    X6A__AUTO__N
DC         Frameinfo
DC         numargs
:;
:
LNJ, B4    X6A__ARET
```

AUTO STORAGE BASE



0

CUR_T_REG

1

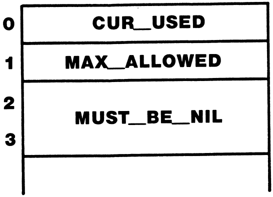
2

BOTTOM_FRAME$

3

4 AUTO_TYPE

AUTO HEAD



0 CUR_USED

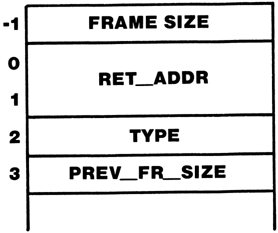
1 MAX_ALLOWED

2 MUST_BE_NIL

3

AUTO FRAME FORMAT

B7 LOCATES AUTO FRAME



-1	FRAME SIZE
0 1	RET_ADDR
2	TYPE
3	PREV_FR_SIZE

FRAMES ALLOCATED UP/DOWN

- MONITOR (FIXED AUTO) USES STACK
- USER PERVERTS STACK

LCP6 LIBRARIES

- FPL
- VDH

FPL LIBRARY

- SUPPORTS INTERPRETIVE PROCEDURE
- IN-LINE CODE FOR ENTRY, EXIT, CALL
- SUPPORTS DELTA

VDH

- PROVIDES FORMATTING, IMPS, ETC.
- ALSO ACTS AS PROTOCOL ENGINE
- CONTAINS HANDLER COMMON
- HASP/3780 + 3270 (OPTIONAL)

HANDLER COMMON

- STATS
- CONTROL/NETCON
- ERROR LOG

ASYNC

- LINE MANAGEMENT
- WORKS WITH VDH

BISYNC

- HANDLERS HASP, 2780/3780, 3270
- WORKS WITH VDH
- FUTURE NJE

UNIT REC

- HANDLES LP, CR, CP
- INTERFACE SAME AS BISYNC

HDLCX25

- REMOTE FEPS
- X.29
- FUTURE X.28, PROGRAMMING X.25

NODEADMN

- CONDUCTS LOGON DIALOGUE
- PROCESSES NETCON ACTIONS
- COMMUNICATES WITH HANDLERS VIA BOBCAT DATA

COUPLER

- HAS COUNTERPART IN HOST
- MANAGES CIRCULAR QUEUES
- FORWARDS DATA TO CORRECT HANDLER
- DETECTS HOST REQUESTED OPERATIONS

ANLZ_FPRG, DELTA_FPRG

- AGENTS FOR HOST COUNTERPARTS
- DO MEMORY FETCH/STORE
- ALSO BREAKPOINTS, ETC.

OTHER FPRGS

- TND (COLTS)
- COMMAND_FP
- PIGLET
- XCC$MCS_FORM

```
17:07 AUG 29 '85 DRIB:STATS.ZZINT

DRIBBLE ON @  22:45 08/22/85
$STATS
 STATS C00 here
#FILE STATDATA.:SYS
#SPAN 9:00-15:00
#SUM ONL,BAT,EXE,SERV,MON,IOS,PMME,MEM,ETMF
#REPL

Interval end  Online  Batch  % exec  % serv  % mon  I/Os  PMMEs  Free Pgs  ETMF
         *** Data follows for THU. AUG 22 '85 ***
09:18:04.20     58      3    152.6   124.4   21.0  3631  22769      2903     1
09:48:03.74     63      3    158.8    98.9   17.5  3038  21185      3145     1
10:18:03.63     68      3    146.8   128.1   16.4  2393  29091      3274     1
10:48:03.47     63      3    159.0    94.2   19.8  3434  19106      2889     1
11:18:03.91     68      4    180.2    96.6   15.2  2190  26267      3310     2
11:48:04.46     71      3    156.8   112.1   18.8  3269  22614      3138     2
12:18:04.90     69      4    177.5    99.6   17.3  2854  21174      3297     1
12:48:04.98     77      4    186.0    95.5   15.0  2299  23432      3001     1
13:18:04.69     76      2    171.2   103.2   14.3  2498  25371      3842     1
13:48:05.21     77      4    146.2   109.0   19.3  2446  26529      2365     2
14:18:05.93     75      4    183.3    99.2   16.7  2730  23095      2396     2
14:48:05.62     79      4    162.1   118.0   15.8  2215  25507      3229     1


#GLOM

Interval end  Online  Batch  % exec  % serv  % mon  I/Os  PMMEs  Free Pgs  ETMF
14:48:05.62     79      4    166.1   105.0   16.9  2675  23824      3229     1


#PLOT MON(0-20) VS IOS(0-4000)
```

```
                   Y-axis: % mon (linear, 0 - 20)
                   X-axis: I/Os (linear, 0 - 4000)
                12 intervals met the selection criteria.

                /---|---|---|---|---|---|---|---|---|---|\
23 and above    |                                        |
22              |                                        |
21              |                                   +    |
20              |                                 +      |
19              |                        +       +       |
18              |                                        |
17              |                          + ++          |
16              |                      ++                |
15              |                     ++                 |
14              |                        +               |
13              |                                        |
12              |                                        |
11              |                                        |
10              |                                        |
 9              |                                        |
 8              |                                        |
 7              |                                        |
 6              |                                        |
 5              |                                        |
 4              |                                        |
 3              |                                        |
 2              |                                        |
 1              |                                        |
 0              |                                        |
                \---|---|---|---|---|---|---|---|---|---|/
                 | 320  | 1160  | 2000  | 2840  | 3680 |
                 0     740    1580    2420    3260   4000

                 Character     +
                 Min hits      1
                 Max hits      1
```

```
#HIST ALL
#GLOM
```

```
             STATS interval from 09:17:33.05 to 14:48:05.62

              "Snap" histogram of interactive response time

(milliseconds)   /---|---|---|---|---|---|---|---|---|---|\
    0 to 1       |******                                  |     7564 ( 16 %)
    2 to 2       |***   `-\                               |     3558 (  8 %)
    3 to 5       |******   `----\                         |     6770 ( 14 %)
    6 to 10      |*******        `-----\                  |     8107 ( 17 %)
   11 to 20      |*******               `-----\           |     8870 ( 19 %)
   21 to 50      |********                     `------\   |     9532 ( 20 %)
   51 to 100     |**                                   `-\|     2870 (  6 %)
  101 to 200     |*                                      !|      121 (  0 %)
  201 to 500     |*                                      !|        5 (  0 %)
  501 to 1000    |                                       !|        0 (  0 %)
 1001 to 2000    |                                       !|        0 (  0 %)
 2001 and above  |                                       !|        0 (  0 %)
                 \---|---|---|---|---|---|---|---|---|---|/  -------
                                                              47397
```

```
               "All" histogram of interactive response time

(milliseconds)  /---|---|---|---|---|---|---|---|---|---|---|\
    0 to 1      |*************                           |   115033 ( 32 %)
    2 to 2      |***          `\                         |    22641 (  6 %)
    3 to 5      |****           `---\                    |    38336 ( 11 %)
    6 to 10     |*****               `---\               |    45750 ( 13 %)
   11 to 20     |******                   `----\         |    56290 ( 16 %)
   21 to 50     |*******                        `-----\  |    60379 ( 17 %)
   51 to 100    |**                                    `\|    17647 (  5 %)
  101 to 200    |*                                      !|      553 (  0 %)
  201 to 500    |*                                      !|       19 (  0 %)
  501 to 1000   |                                       !|        0 (  0 %)
 1001 to 2000   |*                                      !|        1 (  0 %)
 2001 and above |*                                      !|        2 (  0 %)
                \---|---|---|---|---|---|---|---|---|---|---|/  --------
                                                             356651
               "Snap" histogram of user memory sizes

(pages)      /---|---|---|---|---|---|---|---|---|---|\
 0 to 5      |*                                       |      189 (  1 %)
 6 to 10     |****************                        |    12756 ( 39 %)
11 to 15     |******          `----\                  |     5221 ( 16 %)
16 to 20     |***                   `--\              |     2799 (  9 %)
21 to 25     |****                      `--\          |     3405 ( 10 %)
26 to 30     |*                            !          |      424 (  1 %)
31 to 35     |*                             \         |      383 (  1 %)
36 to 40     |*                             !         |       93 (  0 %)
41 to 45     |*                             !         |      201 (  1 %)
46 to 50     |***                            `-\      |     2261 (  7 %)
51 to 55     |*                                 \     |      689 (  2 %)
56 to 60     |*                                 !     |      121 (  0 %)
61 to 65     |*                                 !     |      213 (  1 %)
66 to 70     |*                                 !     |      197 (  1 %)
71 to 75     |*                                  \    |       53 (  0 %)
76 to 80     |*                                  !    |      739 (  2 %)
81 to 85     |*                                  !    |       58 (  0 %)
86 to 90     |**                                  `\  |     1630 (  5 %)
91 to 95     |*                                     \ |       79 (  0 %)
96 and above |*                                      \|     1160 (  4 %)
             \---|---|---|---|---|---|---|---|---|---|/   -------
                                                           32671
```

```
                "All" histogram of user memory sizes

(pages)      /---|---|---|---|---|---|---|---|---|---|\
 0 to 5      |*                                       |    409 (  0 %)
 6 to 10     |*************                           |  26618 ( 31 %)
11 to 15     |******       `----\                     |  12804 ( 15 %)
16 to 20     |*****              `---\                |  11343 ( 13 %)
21 to 25     |*****                   `---\           |   9842 ( 12 %)
26 to 30     |*                           !           |    845 (  1 %)
31 to 35     |*                            \          |   1555 (  2 %)
36 to 40     |*                            !          |    246 (  0 %)
41 to 45     |*                            !          |    836 (  1 %)
46 to 50     |***                           `--\      |   7267 (  9 %)
51 to 55     |*                                 \     |   1579 (  2 %)
56 to 60     |*                                 !     |    289 (  0 %)
61 to 65     |*                                 !     |   1174 (  1 %)
66 to 70     |*                                 !     |    395 (  0 %)
71 to 75     |*                                  \    |    103 (  0 %)
76 to 80     |*                                   \   |   3067 (  4 %)
81 to 85     |*                                   !   |    119 (  0 %)
86 to 90     |**                                   `\ |   3457 (  4 %)
91 to 95     |*                                     ! |    248 (  0 %)
96 and above |*                                      \|   2605 (  3 %)
             \---|---|---|---|---|---|---|---|---|---|/  -------
                                                          84801


"Snap" histogram of time required to complete user service requests

       No events occurred in this interval - histogram skipped.


 "All" histogram of time required to complete user service requests

       No events occurred in this interval - histogram skipped.
```

```
                "Snap" histogram of compute time between interactions

(milliseconds)  /---|---|---|---|---|---|---|---|---|---|\
    0 to 10     |*******                                 |   7854 ( 17 %)
   11 to 15     |************------\                     |  12932 ( 28 %)
   16 to 25     |********           `-----\              |   8858 ( 19 %)
   26 to 45     |**                       `\             |   2350 (  5 %)
   46 to 75     |**                         `\           |   1892 (  4 %)
   76 to 130    |**                           \          |   1875 (  4 %)
  131 to 215    |*                             `\        |   1547 (  3 %)
  216 to 360    |*                               \       |    922 (  2 %)
  361 to 600    |*                               !       |    399 (  1 %)
  601 to 1000   |*                               !       |    345 (  1 %)
 1001 and above |*******                          `-----\|   7998 ( 17 %)
                \---|---|---|---|---|---|---|---|---|---|/  -------
                                                              46972
```

```
               "All" histogram of compute time between interactions

(milliseconds)  /---|---|---|---|---|---|---|---|---|---|\
     0 to 10    |************                            |   110110 ( 31 %)
    11 to 15    |***************___________\             |   134183 ( 38 %)
    16 to 25    |********                   `-------\    |    73518 ( 21 %)
    26 to 45    |*                                  !    |     5968 (  2 %)
    46 to 75    |*                                   \   |     4208 (  1 %)
    76 to 130   |*                                   !   |     4296 (  1 %)
   131 to 215   |*                                    \  |     3268 (  1 %)
   216 to 360   |*                                    !  |     2021 (  1 %)
   361 to 600   |*                                    !  |      989 (  0 %)
   601 to 1000  |*                                    !  |      762 (  0 %)
 1001 and above |**                                    `\|    17161 (  5 %)
                \---|---|---|---|---|---|---|---|---|---|/  --------
                                                             356484
```

```
#DISPLAY CPU,SCPU,RES,IO
#GLOM
```

```
                 STATS interval from 09:17:33.05 to 14:48:05.62

                        {all} {snap}                         {all} {snap}
% batch execution        97.8  114.3   ETMF                      1      1
% batch service          25.4   35.8   90% response time        50     50
% online execution       12.1   29.9   I/O load factor          11     42
% online service         22.6   38.8   # of batch users          2      4
% ghost execution        23.4   21.8   # of online users        30     79
% ghost service          25.9   29.5   # of ghost users         22     24
% TP execution            0.0    0.0   # of TP users             1      2
% TP service              0.0    0.0   I/Os per minute        2023   2675
% monitor execution      14.1   16.9   Schedules per minute   2665   3270
% I/O wait               29.7    9.1   Interactions per min    242    143
% resource wait           0.0    0.0   Events per minute      4673   5662
% I/O&resource wait       0.1    0.2   PMMEs per minute      19273  23824
% true idle              47.0    2.0   Avg. usec per PMME     2318   2648
Total                   298.9  299.5   Minutes in interval    1475    330
```

```
                     Master CPU   Slave CPU 2   Slave CPU 3   Slave CPU 4
                    {all} {snap} {all} {snap} {all} {snap}  {all} {snap}
% user execution      34     45     0      0      0      0     49     60
% user service        30     39     0      0      0      0     21     33
% monitor exec.       10     12     0      0      0      0      2      2
% I/O wait            11      3     0      0      0      0      9      3
% resource wait        0      0     0      0      0      0      0      0
% I/O & res wait       0      0     0      0      0      0      0      0
% true idle           11      1     0      0      0      0     17      1
% unaccountable        0      0   100    100    100    100      0      0
Schedules/minute    1426   1527     0      0      0      0    622    873
Events/minute       3717   4362     0      0      0      0    480    652
PMME starts/min.    9898  11191     0      0      0      0   4724   6402
PMME ends/min.      9897  11183     0      0      0      0   4724   6405


                    Slave CPU 5   Slave CPU 6
                    {all} {snap} {all} {snap}
% user execution      49     60     0      0
% user service        21     33     0      0
% monitor exec.        2      3     0      0
% I/O wait             9      3     0      0
% resource wait        0      0     0      0
% I/O & res wait       0      0     0      0
% true idle           17      1     0      0
% unaccountable        0      0   100    100
Schedules/minute     617    865     0      0
Events/minute        476    646     0      0
PMME starts/min.    4650   6231     0      0
PMME ends/min.      4651   6235     0      0

                    CP-6 monitor resource utilization

{Resource name}                 {  # in }  {---since system boot---}  {  Total  }
                                {use now}  { (max) (min) (average) }  {available}

IOQ packets                        44        100    0        45          110
IOS packets                        103       104    57       85          397
I/O cache entries                  2918      4212   3        3008        4096
Enqueue/Dequeue data blocks        649       2239   11       509         2560
Scheduler Do-list entries          1         12     0        0           50
```

I/O cache activity (actions per minute)

	Attempted Gets	Hits UC=0	Hits UC>0	Percent Hits	Attempted Puts	Failed Puts	Unused Pages	
MAD	36	35	0	99	0	0	15	{all}
	77	77	0	99	0	0	15	{snap}
PAD	13	13	0	96	0	0	11	{all}
	7	6	0	90	1	0	22	{snap}
GP	73	65	7	99	0	0	26	{all}
	115	104	11	99	0	0	18	{snap}
FD	500	485	4	97	11	0	175	{all}
	528	514	6	98	9	0	163	{snap}
FIT	426	276	7	66	143	0	710	{all}
	671	493	13	75	166	0	233	{snap}
UL	40	36	3	97	2	0	30	{all}
	63	54	7	97	3	0	50	{snap}
INDEX	197	160	8	85	41	0	99	{all}
	271	210	12	81	67	0	290	{snap}
DATA	824	605	43	78	263	0	349	{all}
	1274	882	55	73	445	0	1698	{snap}
REL	0	0	0	49	0	0	0	{all}
	0	0	0	0	0	0	0	{snap}
CONSEC	135	80	0	59	133	0	1099	{all}
	277	125	1	45	343	0	41	{snap}
ELSE	20	0	0	1	23	0	0	{all}
	35	0	0	1	40	0	0	{snap}
Total	2266	1757	73	80	617	0	2514	{all}
	3324	2469	105	77	1074	0	2530	{snap}

```
                     CP-6 memory utilization

AARDVARK and RECOVERY                                   45
XDELTA and monitor debug schema                        112
Monitor procedure and static data                      258
Monitor context (JITs, HJITs, PPUT, page tables)        47
Monitor dynamic data segments                           33
TIGR-built tables                                       50
Communications WSQs                                     74
Comgroup queue                                          71
Total pages held back for monitor use                   15
Resident system ghosts                                 388
Required processors (IBEX, DELTA, LOGON)               228
All other special shared (resident) processors         613
                                                     -------
Total dedicated memory                                1934

Available to users                                    6258
Currently allocated to users                          2136
Automatically shared run units in use                  733
Shared data segments in use                             22
Free pages                                             284
Automatically shared run units not in use              401
I/O cache pages (Use Count = 0)                       2529
Total pages currently available                       3229
I/O cache pages                                       2672
Number of pages not accounted for                       10
Total physical pages in system                        8192
```

```
 Device      # of     connects      %         %         %         %       load
 name      connects   per min.     idle      wait      busy    backlog   factor

SC010000      3773         11     100.0      0.0       0.0       0.0       0.0   {snap}
             12537          8     100.0      0.0       0.0       0.0       0.0   {all}

DC010000    145124        439     100.0      0.0       0.0       0.0       0.0   {snap}
            359950        243     100.0      0.0       0.0       0.0       0.0   {all}

DC020000    284827        861     100.0      0.0       0.0       0.0       0.0   {snap}
            715820        484     100.0      0.0       0.0       0.0       0.0   {all}

DP010000     98149        296      85.8      0.1      11.7       2.4      17.5   {snap}
            283045        191      91.4      0.0       7.2       1.4      16.6   {all}

DP020000    146134        442      78.8      0.1      16.4       4.7      22.5   {snap}
            342938        232      89.1      0.0       8.6       2.2      20.7   {all}

DP030000    163738        495      77.3      0.1      16.7       5.9      26.5   {snap}
            391650        265      88.1      0.1       9.8       2.1      17.7   {all}

DP040000        12          0     100.0      0.0       0.0       0.0       1.7   {snap}
              3273          2      99.9      0.0       0.1       0.1      58.0   {all}

DP050000     10615         32      97.9      0.0       1.5       0.7      32.1   {snap}
             24654         16      99.1      0.0       0.7       0.2      21.6   {all}

DP060000     11227         33      97.8      0.0       1.5       0.6      29.7   {snap}
             25289         17      99.0      0.0       0.8       0.2      19.1   {all}

DP080000        49          0     100.0      0.0       0.0       0.0       1.1   {snap}
               398          0     100.0      0.0       0.0       0.0       1.0   {all}

DP090000        15          0     100.0      0.0       0.0       0.0       0.5   {snap}
              1446          0     100.0      0.0       0.0       0.0       0.9   {all}

DP110000        12          0     100.0      0.0       0.0       0.0       0.0   {snap}
              2692          1      99.9      0.0       0.1       0.0       0.7   {all}

DC030000    210720        637     100.0      0.0       0.0       0.0       0.0   {snap}
            875570        593     100.0      0.0       0.0       0.0       0.0   {all}

DC040000    210754        637     100.0      0.0       0.0       0.0       0.0   {snap}
            874614        592     100.0      0.0       0.0       0.0       0.0   {all}

DP210000     12000         36      98.6      0.0       1.3       0.0       2.7   {snap}
             31481         21      99.2      0.0       0.8       0.0       3.6   {all}

DP230000      2503          7      99.7      0.0       0.3       0.0       7.0   {snap}
              6620          4      99.8      0.0       0.2       0.0       4.0   {all}

DP250000     35786        108      94.6      0.0       5.2       0.1       2.8   {snap}
```

	160651	108	94.8	0.0	4.9	0.3	6.6	{all}
DP270000	4945	14	99.3	0.0	0.7	0.0	5.6	{snap}
	11926	8	99.6	0.0	0.4	0.0	3.8	{all}
DP290000	55102	166	93.7	0.1	6.2	0.0	1.5	{snap}
	237124	160	94.2	0.0	5.7	0.0	1.5	{all}
DP310000	7461	22	99.0	0.0	1.0	0.0	1.5	{snap}
	30029	20	99.2	0.0	0.7	0.1	9.1	{all}
DP330000	42301	127	94.8	0.0	4.4	0.8	16.3	{snap}
	82900	56	97.8	0.0	1.9	0.3	14.2	{all}
DP350000	72	0	100.0	0.0	0.0	0.0	43.2	{snap}
	346	0	99.7	0.3	0.0	0.0	97.1	{all}
DP370000	22783	68	97.0	0.0	2.8	0.3	9.4	{snap}
	59977	40	98.3	0.0	1.6	0.1	7.3	{all}
DP390000	59	0	100.0	0.0	0.0	0.0	0.9	{snap}
	171	0	100.0	0.0	0.0	0.0	0.8	{all}
DP410000	4428	13	99.5	0.0	0.4	0.0	4.3	{snap}
	8366	5	99.8	0.0	0.2	0.0	4.1	{all}
DP430000	14490	43	98.6	0.0	1.3	0.1	8.0	{snap}
	35147	23	99.2	0.0	0.7	0.1	7.2	{all}
DP450000	10696	32	98.8	0.0	1.2	0.1	4.7	{snap}
	22040	14	99.4	0.0	0.6	0.0	4.1	{all}
DP470000	7148	21	99.2	0.0	0.7	0.0	5.9	{snap}
	34928	23	99.2	0.0	0.8	0.0	4.6	{all}
DP490000	32440	98	96.1	0.0	3.5	0.4	10.7	{snap}
	114148	77	96.8	0.0	2.7	0.6	18.0	{all}
DP510000	169260	512	79.6	0.1	19.8	0.4	2.7	{snap}
	914330	619	77.3	0.2	22.1	0.4	2.5	{all}
TC010000	26134	79	100.0	0.0	0.0	0.0	0.0	{snap}
	138807	94	100.0	0.0	0.0	0.0	0.0	{all}
MT030000	12091	36	97.7	0.0	2.3	0.0	0.5	{snap}
	51114	34	98.0	0.0	1.9	0.0	2.7	{all}
MT040000	13486	40	97.4	0.0	2.6	0.0	0.5	{snap}
	68136	46	96.4	0.1	3.5	0.0	2.6	{all}
MT050000	557	1	100.0	0.0	0.0	0.0	5.7	{snap}
	19002	12	99.8	0.0	0.2	0.0	3.8	{all}
UC010000	1335	4	100.0	0.0	0.0	0.0	0.0	{snap}

	7504	5	100.0	0.0	0.0	0.0	0.0	{all}
LP010000	1002	3	86.8	0.2	13.0	0.0	1.8	{snap}
	3666	2	89.5	0.2	10.3	0.0	2.0	{all}
LP020000	333	1	95.7	0.0	4.3	0.0	0.0	{snap}
	3838	2	89.3	0.3	10.4	0.0	2.5	{all}
UC030000	323	0	100.0	0.0	0.0	0.0	0.0	{snap}
	323	0	100.0	0.0	0.0	0.0	0.0	{all}
LP100000	323	0	95.7	0.1	4.2	0.0	1.2	{snap}
	323	0	99.0	0.0	0.9	0.0	1.2	{all}

```
 IOM-chan    # of     connects     %        %        %        %       load
  number   connects   per min.    idle     wait     busy   backlog   factor

   0-08      35869        108     95.0     0.0      5.0     0.0       0.0   {snap}
             89471         60     97.3     0.0      2.7     0.0       0.0   {all}

   0-09      37009        111     94.9     0.0      5.1     0.0       0.0   {snap}
             90966         61     97.3     0.0      2.7     0.0       0.0   {all}

   0-10      34899        105     95.0     0.0      5.0     0.0       0.0   {snap}
             88389         59     97.3     0.0      2.7     0.0       0.0   {all}

   0-11      37347        112     94.9     0.0      5.1     0.0       0.0   {snap}
             91120         61     97.3     0.0      2.7     0.0       0.0   {all}

   0-12      36444        110     94.8     0.0      5.2     0.0       0.0   {snap}
             90643         61     97.2     0.0      2.8     0.0       0.0   {all}

   0-13      35007        105     94.8     0.0      5.2     0.0       0.0   {snap}
             88993         60     97.2     0.0      2.8     0.0       0.0   {all}

   0-14      35063        106     94.9     0.0      5.1     0.0       0.0   {snap}
             89020         60     97.3     0.0      2.7     0.0       0.0   {all}

   0-15      36184        109     94.8     0.0      5.2     0.0       0.0   {snap}
             89513         60     97.2     0.0      2.8     0.0       0.0   {all}

   0-16      13068         39     97.5     0.0      2.5     0.0       0.0   {snap}
             69413         47     97.3     0.0      2.7     0.0       0.0   {all}

   0-17      13066         39     97.5     0.0      2.5     0.0       0.0   {snap}
             69394         47     97.2     0.0      2.8     0.0       0.0   {all}

   0-20      52643        159     93.7     0.0      6.3     0.0       0.0   {snap}
            218965        148     94.5     0.0      5.5     0.0       0.0   {all}

   0-21      52661        159     93.7     0.0      6.3     0.0       0.0   {snap}
            218562        148     94.5     0.0      5.5     0.0       0.0   {all}

   0-22      52695        159     93.7     0.0      6.3     0.0       0.0   {snap}
            218754        148     94.5     0.0      5.5     0.0       0.0   {all}

   0-23      52756        159     93.8     0.0      6.2     0.0       0.0   {snap}
            218812        148     94.5     0.0      5.5     0.0       0.0   {all}

   0-24       1002          3     81.9     0.0     18.1     0.0       0.0   {snap}
              3666          2     86.6     0.0     13.4     0.0       0.0   {all}

   0-25        333          1     95.1     0.0      4.9     0.0       0.0   {snap}
              3838          2     88.C     0.0     12.0     0.0       0.0   {all}

   0-28        323          0     95.5     0.0      4.5     0.0       0.0   {snap}
```

	323	0	99.0	0.0	1.0	0.0	0.0	{all}
0-30	3773	11	81.3	0.0	18.7	0.0	0.0	{snap}
	12537	8	86.2	0.0	13.8	0.0	0.0	{all}
1-12	35722	108	94.8	0.0	5.2	0.0	0.0	{snap}
	89816	60	97.2	0.0	2.8	0.0	0.0	{all}
1-13	35882	108	94.9	0.0	5.1	0.0	0.0	{snap}
	90038	61	97.2	0.0	2.8	0.0	0.0	{all}
1-14	35451	107	94.9	0.0	5.1	0.0	0.0	{snap}
	89428	60	97.2	0.0	2.8	0.0	0.0	{all}
1-15	35074	106	94.9	0.0	5.1	0.0	0.0	{snap}
	88369	59	97.3	0.0	2.7	0.0	0.0	{all}
1-20	52686	159	93.7	0.0	6.3	0.0	0.0	{snap}
	218892	148	94.5	0.0	5.5	0.0	0.0	{all}
1-21	52730	159	93.7	0.0	6.3	0.0	0.0	{snap}
	219151	148	94.5	0.0	5.5	0.0	0.0	{all}
1-22	52713	159	93.7	0.0	6.3	0.0	0.0	{snap}
	218547	148	94.4	0.0	5.6	0.0	0.0	{all}
1-23	52590	159	93.7	0.0	6.3	0.0	0.0	{snap}
	218501	148	94.5	0.0	5.5	0.0	0.0	{all}

NETCON

- NEW IN C00
  - REPLACES FEPCON
  - REPLACES PART OF FEPANLZ
- INITIALLY USED IN $XINSTALL
  - DEFINE NODES
  - DEFINED DEFAULTS

NETCON (CONT)

- NEXT USED TO REDO IT ALL
  - TO MAKE CORRECT CONFIG FOR SITE
  - COULD USE $XINSTALL__LOCAL.:SYS
- TWO PARTS OF THE WORLD
  - LOCAL FEPS (LFEPS)
  - REMOTE FEPS (RFEPS)

BOOT INFO

		LFEP	RFE
• MONITOR		M:FEP	M:FEP
• LIBRARY ACCOUNT		:SYS	:SYS
• HANDLERS	#1->	NODEADMN	NODEADMN
	#2->	COUPLER	HDLCX25

BOOT PARAMETERS

- DEFAULTS CAN BE USED (MOSTLY)
- EXCEPTION IF TP OR FPRGS
- EXCEPTION: NSHUF

HANDLER PARAMETERS

- SELECT H=HANDLER COMMAND
- COMMON PARAMETER NAMES
  - MEM
  - IOMEM
- SPECIAL PARAMETERS PER HANDLER

HANDLER DEFAULTS

- DEFAULT HANDLER COMMAND
- COMMON PARAMETERS NAMES
  - ENABLE, REENABLE
- SPECIAL PARAMETERS PER HANDLER
  - BLOCK, UNBLOCK
  - BUFSIZE

CONFIG COMMANDS

- USED ON SPECIFIC LINES
- USED TO SUPPLY AUTO LOGON
- USED TO SET SPECIAL PARAMETER VALUES

NETWORKING (RFEPS)

- MULTIPLE PROCESSORS
  - NETCON
  - SUPER
  - PIGETTE

NETWORKING (RFEPS) (CONT)

- XEQ FILES
  - STARTING POINT
  - EDIT GLOBALS AND NETCON
  - XEQ SUPER AND NETCON
  - REBOOT LOCAL
  - MAKE DISKETTE
  - PUT DISKETTE IN RFEP
  - BOOT RFEP MANUALLY

```
 1.000 !ECHO
 1.500 !"!REPORT STEP=FULL
 2.000 !M DINGO ghost starting using !XEQ $XINSTALL.:SYS
 3.000 !SET M$LL ME
 4.000 !" File: $XINSTALL.:SYS (C00 version)
 5.000 !"
 6.000 !" This file is XEQ'd by the system DINGO ghost after tape boots
 7.000 !" with the command:
 8.000 !" !XEQ $XINSTALL.:SYS BOOTFLAG=nn
 9.000 !"
10.000 !" where nn=
11.000 !"    02 for tape boot, Y to new file system
12.000 !"    03 for tape boot, S to new file system
13.000 !"    04 for tape boot, N to new file system
14.000 !"    05 for tape boot, S to new file system (with reconstruct)
15.000 !"    06 for tape boot, N to new file system (with reconstruct)
16.000 !"    07 for disk boot
17.000 !"    08 for disk boot (with reconstruct)
18.000 !"    09 for recovery
19.000 !"    10 for recovery (lost dumpfile(?))
20.000 !"    11 for operator recovery (from tape)
21.000 !"
22.000 !IF BOOTFLAG=11 THEN M DINGO is running after a tape-initiated OR
23.000 !IF BOOTFLAG=10 THEN M DINGO is running after a recovery (DUMPF)
24.000 !IF BOOTFLAG=9  THEN M DINGO is running after a recovery
25.000 !IF BOOTFLAG=8  THEN M DINGO is running after a disk boot, reconstruct
26.000 !IF BOOTFLAG=7  THEN M DINGO is running after a disk boot
27.000 !IF BOOTFLAG=6 THEN M DINGO is running after a TAPE boot, N to new files, reconstruct
28.000 !IF BOOTFLAG=5 THEN M DINGO is running after a TAPE boot, S to new files, reconstruct
29.000 !IF BOOTFLAG=4 THEN M DINGO is running after a TAPE boot, N to new files
30.000 !IF BOOTFLAG=3 THEN M DINGO is running after a TAPE boot, S to new files
31.000 !IF BOOTFLAG=2 THEN M DINGO is running after a TAPE boot, Y to new files
32.000 !PRIV ALL
33.000 !"
34.000 !" THIS PART GETS EXECUTED ON EVERY SYSTEM STARTUP
35.000 !"
36.000 !" ***************************************************************
```

```
37.000 !" NOTE THAT PACKSETS OTHER THAN #SYS MAY NOT BE UP AT THE TIME
38.000 !" THIS IS RUN, SO YOU SHOULDN'T DEPEND ON BEING ABLE TO GET TO
39.000 !" FILES OTHER THAN THOSE ON #SYS!
40.000 !" ************************************************************
41.000 !"
42.000 !PCL
43.000 MOD TP_CNTRL_D.:SYS TO (ACS+((TPA?,TPCP?),WNEW,READ,UPD))
44.000 " Andrew and J.L., this is for you. Love, JJ
45.000 MOD :JOBSTATS.:SYS(SH) TO (ACC((:SYS),READ,DELR,WNEW,UPD,DELF,REATTR),;
46.000                                ACC+((?),EXEC),;
47.000                                ACS+((IBEX),READ,WNEW,UPD),;
48.000                                ACS+((A.X),READ),;
49.000                                ACS+((SYMBO),READ),;
50.000                                ACS+((PEEK.X),READ),;
51.000                                ACS+((BASS),READ),;
52.000                                ACS+((FILER.X),READ),;
53.000                                ACS+((MOM.X),READ),;
54.000                                ACS+((MOM),READ),;
55.000                                ACS+((TERM),READ),;
56.000                                ACS+((WOODPECKER),READ),;
57.000                                ACS+((RQ.X),READ))
58.000 END
59.000 !"
60.000 !" Special step: lets you create a file in :SYS that gets XEQd
61.000 !" whenever this job does.  For example, on the LADC L66B machine,
62.000 !" since that machine has 3270's configured, the file $XINSTALL_LOCAL
63.000 !" contains a line like:
64.000 !"
65.000 !" !C :SHARED_LCP6_RELEASE.:SYS OVER :SHARED_LCP6_SYSTEM.:SYS
66.000 !"
67.000 !" that the LADC L66A doesn't need.
68.000 !"
69.000 !" note that when this file runs, there is no guarantee that packsets
70.000 !" other than DP#SYS are mounted, so don't expect anything extra-fancy
71.000 !" to work here.  Also, you may find that changing user-maximums and
72.000 !" other CONTROLlable values may not work as you think they might.
73.000 !"
74.000 !" This job is primarily to guarantee that a system comes up right!
75.000 !" It performs LADC-defined specific tasks, that are mostly support
```

```
 76.000 !" oriented.  If this job or $XINSTALL_LOCAL can be adapted to do what
 77.000 !" you want, that's great.  If not: Sorry, Charlie.
 78.000 !"
 79.000 !IF $FID_EXIST('$XINSTALL_LOCAL.:SYS') THEN XEQ $XINSTALL_LOCAL.:SYS THISBOOT=BOOTFLAG
 80.000 !IF BOOTFLAG>6 THEN GOTO END_OF_TAPE_BOOT
 81.000 !"
 82.000 !"   ************************************************************
 83.000 !"   The stuf* in this section only gets EXECUTED after a tape boot
 84.000 !"   ************************************************************
 85.000 !TAPE_BOOT_ONLY:
 86.000 !"
 87.000 !"   This section copies the system authorization files over themselves
 88.000 !"   so they get their upper_level key granules rebuilt and data space
 89.000 !"   recovered.  You may not notice it, but average logon time drops
 90.000 !"   dramatically if the authorization files aren't fragmented.  Also,
 91.000 !"   the COO :USERS records are fixed size, so changing a password or
 92.000 !"   profile allows the record to be rewritten IN PLACE, also getting
 93.000 !"   rid of a primary source of fragmentation.
 94.000 !"
 95.000 !IF $FID_EXIST(':RATES.:SYS') THEN CA :RATES.:SYS OVER .:SYS
 96.000 !IF $FID_EXIST(':PRO.:SYS') THEN CA :PRO.:SYS OVER .:SYS
 97.000 !IF $FID_EXIST(':WSN.:SYS') THEN CA :WSN.:SYS OVER .:SYS
 98.000 !IF $FID_EXIST(':OSYMB.:SYS') THEN CA :OSYMB.:SYS OVER .:SYS
 99.000 !IF $FID_EXIST(':NETCON.:SYS') THEN CA :NETCON.:SYS OVER .:SYS
100.000 !IF $FID_EXIST(':FORM.:SYS') THEN CA :FORM.:SYS OVER .:SYS
101.000 !IF $FID_EXIST(':HLP.:SYS') THEN CA :HLP.:SYS OVER .:SYS
102.000 !IF $FID_EXIST(':USERS.:SYS') THEN CA :USERS.:SYS OVER .:SYS
103.000 !IF $FID_EXIST(':NAME.:MAIL') THEN CA :NAME.:MAIL OVER .:MAIL
104.000 !IF $FID_EXIST(':MAIL_CENTRAL.:MAIL') THEN CA :MAIL_CENTRAL.:MAIL OVER .:MAIL
105.000 !IF $FID_EXIST('::PATCH.:SYS') THEN CA ::PATCH.:SYS OVER .:SYS(LN,C)
106.000 !IF $FID_EXIST('::RUM.:SYS') THEN CA ::RUM.:SYS OVER .:SYS(LN,C)
107.000 !LET PATCHFILE=':PF'||$VERSION||'.:SYSTAC'
108.000 !IF $FID_EXIST(':PLOVER.:SYS')=0 THEN GOTO SKIP_PLOVERING
109.000 !" Remove OUTPUT INTO, ECHO,   KILL RUM, and ALIB RET   from :PLOVER.
110.000 !C :PLOVER.:SYS(3-%($FID_RECS(':PLOVER.:SYS') - 2)) OVER ::PLOVER.:SYS(LN,C)
111.000 !IF STEPCC=0 THEN DEL :PLOVER.:SYS
112.000 !PLOVER ,ME OVER ,*PLOVERFILE
113.000 READ ::PLOVER.:SYS
114.000 END
```

```
115.000 !IF $FID_EXIST('%PATCHFILE') THEN C %PATCHFILE,*PLOVERFILE OVER %PATCHFILE(LN,C)
116.000 !GOTO PLOVER_OR_NOT
117.000 !SKIP_PLOVERING: IF $FID_EXIST('%PATCHFILE') THEN CA %PATCHFILE OVER .:SYSTAC(LN,C)
118.000 !PLOVER_OR_NOT: LET DELETE PATCHFILE
119.000 !M Dingo says: this boot/dumpfile revision is %$VERSION
120.000 !"
121.000 !"  build some default stuff on Y to new file system boot
122.000 !"  and/or we've lost :NETCON
123.000 !"
124.000 !IF $FID_EXIST(':NETCON.:SYS') THEN GOTO SKIP_NETCON_STEP
125.000 !NETCON
126.000 DEF NODE=0,NAME=HOST,TYPE=ME
127.000 DEF NODE=1,TYPE=FEP
128.000 DEF NODE=2,TYPE=FEP
129.000 DEF NODE=3,TYPE=FEP
130.000 DEF NODE=4,TYPE=FEP
131.000 DEF NODE=5,TYPE=FEP
132.000 DEF NODE=6,TYPE=FEP
133.000 DEF NODE=7,TYPE=FEP
134.000 DEF NODE=8,TYPE=FEP
135.000 DEF NODE=9,TYPE=FEP
136.000 DEF NODE=10,TYPE=FEP
137.000 DEF NODE=11,TYPE=FEP
138.000 DEF NODE=12,TYPE=FEP
139.000 DEF NODE=13,TYPE=FEP
140.000 DEF NODE=14,TYPE=FEP
141.000 DEF NODE=15,TYPE=FEP
142.000 DEF NODE=16,TYPE=FEP
143.000 SEL N=1
144.000 DEFAULT ASYNC PRO='DFLPRF'
145.000 SEL N=2
146.000 DEFAULT ASYNC PRO='DFLPRF'
147.000 SEL N=3
148.000 DEFAULT ASYNC PRO='DFLPRF'
149.000 SEL N=4
150.000 DEFAULT ASYNC PRO='DFLPRF'
151.000 SEL N=5
152.000 DEFAULT ASYNC PRO='DFLPRF'
153.000 SEL N=6
```

```
154.000 DEFAULT ASYNC PRO='DFLPRF'
155.000 SEL N=7
156.000 DEFAULT ASYNC PRO='DFLPRF'
157.000 SEL N=8
158.000 DEFAULT ASYNC PRO='DFLPRF'
159.000 SEL N=9
160.000 DEFAULT ASYNC PRO='DFLPRF'
161.000 SEL N=10
162.000 DEFAULT ASYNC PRO='DFLPRF'
163.000 SEL N=11
164.000 DEFAULT ASYNC PRO='DFLPRF'
165.000 SEL N=12
166.000 DEFAULT ASYNC PRO='DFLPRF'
167.000 SEL N=13
168.000 DEFAULT ASYNC PRO='DFLPRF'
169.000 SEL N=14
170.000 DEFAULT ASYNC PRO='DFLPRF'
171.000 SEL N=15
172.000 DEFAULT ASYNC PRO='DFLPRF'
173.000 SEL N=16
174.000 DEFAULT ASYNC PRO='DFLPRF'
175.000 END
176.000 !SUPER
177.000 REM PROFILE $ASYNC
178.000 CREATE PROFILE $ASYNC ASYNC
179.000   "NULL"
180.000 M PRO DFLHC FROM $ASYNC          "default hardcopy"
181.000 TABSIM=YES; TABRELATIVE=YES; SPACEINSERT=YES; DISPINPUT=YES
182.000 FULLDUXPAPERTAPE=NO; HALFDUXPAPERTAPE=NO; LOWERCASE=NO
183.000 HEIGHT=0; PROGDTAB=NO; LIMBOCLM=YES
184.000 NOOPTMIZ=NO
185.000 DEVICEBS=YES; EDITOVR=YES; BLANKERASES=NO; LOWERCASEPRINTS=NO
186.000 CHARSETNAME=ASC64; INPUT=YES; OUTPUT=YES
187.000 SENDBKSPACE=NO; ACTONTRN=NO; APLLCNRM=YES; OPDTAB=NO
188.000 DEVICECR=YES; DEVICELF=YES; AUTONL=NO; CRISNL=NO; LFISNL=NO
189.000 DEVPOSOPTS=NO; APL=NO; BIN=NO; PRINTTYPE=YES; TTYTYPE=YES
190.000 RETYPOVR=NO; SCROLL=NO; PAGEHALT=NO
191.000 PRINTHALT=NO; RELPAGE=NO; DEVSCROLL=NO; CURSORUP=NO; TRUOVRPRT=YES
192.000 TMNALG_PRM1=0; TMNALG_PRM2=0; TMNALG_PRM7=0; TMNALG_AFTCR=0; TMNALG_AFTLF=0; ERSTIM=0
```

```
193.000   "NULL"
194.000 M PRO DFLCRT FROM $ASYNC          "default CRT"
195.000 TABSIM=YES; TABRELATIVE=YES; SPACEINSERT=YES; DISPINPUT=YES
196.000 FULLDUXPAPERTAPE=NO; HALFDUXPAPERTAPE=NO; LOWERCASE=NO
197.000 HEIGHT=0; PROGDTAB=NO; LIMBOCLM=YES
198.000 NOOPTMIZ=NO
199.000 DEVICEBS=YES; BLANKERASES=YES; LOWERCASEPRINTS=NO
200.000 CHARSETNAME=ASC64; INPUT=YES; OUTPUT=YES
201.000 SENDBKSPACE=NO; ACTONTRN=NO; APLLCNRM=YES; OPDTAB=NO
202.000 DEVICECR=YES; DEVICELF=YES; AUTONL=NO; CRISNL=NO; LFISNL=NO
203.000 DEVPOSOPTS=NO; APL=NO; BIN=NO; PRINTTYPE=YES; TTYTYPE=YES
204.000 RETYPOVR=NO; EDITOVR=NO; SCROLL=YES; PAGEHALT=NO
205.000 PRINTHALT=YES; RELPAGE=YES; DEVSCROLL=YES; CURSORUP=NO; TRUOVRPRT=NO
206.000 TMNALG_PRM1=0; TMNALG_PRM2=0; TMNALG_PRM7=0; TMNALG_AFTCR=0; TMNALG_AFTLF=0; ERSTIM=0
207.000   "NULL"
208.000 M PRO TTY FROM DFLHC              "Very basic and slow"
209.000 DEVICEBS=NO; EDITOVR=NO
210.000 AWIDTH=72; PWIDTH=72; HEIGHT=0
211.000 CHARSETNAME=ASC64; LOWERCASEPRINTS=NO
212.000 DEVICECR=YES; DEVICELF=YES; AUTONL=NO; CRISNL=NO; LFISNL=NO
213.000 TMNALG_PRM1=2.5; TMNALG_PRM2=166.7; TMNALG_PRM7=66.7; TMNALG_AFTCR=1; TMNALG_AFTLF=2; ERSTI
M=0
214.000   "NULL"
215.000 M PRO DFLPRF FROM TTY             "Default profile (before logon)"
216.000 FCNTBL=CP5
217.000   "NULL"
218.000 END
219.000 !C ME OVER $XPIGALL.:SYS(LN)
220.000 !!DEFAULT PACKSET=SYS
221.000 !!PIG
222.000 CR DP#PACKSET.SUPPORT GR=5000
223.000 CR DP#PACKSET.X GR=10000,EXEC=?
224.000 CR DP#PACKSET.:XSI GR=15000
225.000 CR DP#PACKSET.:LIBRARY GR=5000,READ=?
226.000 CR DP#PACKSET.DEMO GR=5000,READ=?
227.000 CR DP#PACKSET.:DEMOSI GR=10000,READ=?
228.000 CR DP#PACKSET.:CONVERT GR=5000
229.000 CR DP#PACKSET.:QUAC GR=5000
230.000 CR DP#PACKSET.:COOPRC GR=5000
```

```
231.000 CR DP#PACKSET.:SRB GR=2500,READ=?
232.000 CR DP#PACKSET.:DOCUM GR=6000,READ=?
233.000 CR DP#PACKSET.:FED GR=3000
234.000 CR DP#PACKSET.NULLACCT GR=1,READ=?
235.000 DISM DP#PACKSET
236.000 END
237.000 !!RELEASE DP#PACKSET
238.000 !SKIP_NETCON_STEP:
239.000 !END_OF_TAPE_BOOT:
240.000 !M DINGO exiting, stepcc = %STEPCC
```

```
13:04 AUG 22 '85 :NETCON_A.ZZGLG3MK

 1.000 !NETCON
 2.000 DEL NODE=0
 3.000 DEL NODE=1
 4.000 DEL NODE=2
 5.000 DEL NODE=3
 6.000 DEL NODE=4
 7.000 DEL NODE=5
 8.000 DEL NODE=6
 9.000 DEL NODE=7
10.000 DEL NODE=8
11.000 DEL NODE=9
12.000 DEL NODE=10
13.000 DEL NODE=11
14.000 DEL NODE=12
15.000 DEL NODE=13
16.000 DEL NODE=14
17.000 DEL NODE=15
18.000 DEL NODE=16
19.000 DEL NODE=17
20.000 DEL NODE=18
21.000 DEL NODE=19
22.000 DEL NODE=20
23.000 DEL NODE=21
24.000 DEL NODE=22
25.000 DEL NODE=32
26.000 DEL NODE=33
27.000 DEL NODE=34
28.000 DEF NODE=0,NAME=L66A,TYPE=ME
29.000 DEF NODE=1,NAME=L6I,TYPE=FEP,CONTROL=L66A
30.000 DEF NODE=2,NAME=L6II,TYPE=DEBUG,CONTROL=L66B
31.000 DEF NODE=3,NAME=L6III,TYPE=DEBUG,CONTROL=L66B
32.000 DEF NODE=4,NAME=L6IV,TYPE=FEP,CONTROL=L66A
33.000 DEF NODE=5,NAME=L6V,TYPE=DEBUG,CONTROL=L66B
34.000 DEF NODE=6,NAME=L6VI,TYPE=FEP,CONTROL=L66C
35.000 DEF NODE=7,NAME=L6VII,TYPE=DEBUG,CONTROL=L66B
36.000 DEF NODE=8,NAME=L6VIII,TYPE=FEP,CONTROL=L66A
37.000 DEF NODE=9,NAME=L6IX,TYPE=FEP,CONTROL=L66B
```

```
38.000 DEF NODE=10,NAME=L6X,TYPE=DEBUG,CONTROL=L66B
39.000 DEF NODE=11,NAME=L6XI,TYPE=FEP,CONTROL=L66B
40.000 DEF NODE=12,NAME=L6XII,TYPE=FEP,CONTROL=L66B
41.000 DEF NODE=13,NAME=L6XIII,TYPE=DEBUG,CONTROL=L66B
42.000 DEF NODE=15,NAME=L6XV,TYPE=DEBUG,CONTROL=L66A
43.000 DEF NODE=20,NAME=L66B,TYPE=HOST
44.000 DEF NODE=21,NAME=L66D,TYPE=HOST
45.000 DEF NODE=22,NAME=L66C,TYPE=HOST
46.000 DEF NODE=32,NAME=DVFEP,TYPE=FEP,CONTROL=L66A
47.000 DEF NODE=33,NAME=CRFEP,TYPE=FEP,CONTROL=L66A
48.000 DEF NODE=34,NAME=CP6FEP,TYPE=FEP,CONTROL=L66A
49.000 " L66A :NETCON - DEFAULT COMMANDS AND CONFIGS FOR FEP 4 " ;
50.000 SEL NODE=4
51.000 SET BOOTINFO
52.000 M:FEP.:SYS
53.000 5
54.000 NODEADMN
55.000 NODEADMN.:SYS
56.000 COUPLER
57.000 COUPLER.:SYS
58.000 ASYNC
59.000 ASYNC.:SYS
60.000 BISYNC
61.000 BISYNC.:SYS
62.000 HDLCX25
63.000 HDLCX25.:SYS
64.000 :SYS
65.000 DEFAULT ASYNC ;
66.000                 INPUT = YES,;
67.000                 OUTPUT = YES,;
68.000                 SALUTATION = YES,;
69.000                 SPEED = AUTO,;
70.000                 ENABLE = YES,;
71.000                 REENABLE = YES,;
72.000                 DROPDTR = NO,;
73.000                 HARDWIRE = YES,;
74.000                 ECHOLOGON = NO,;
75.000                 LOGONTIMEOUT = 5,;
76.000                 READTIMEOUT = 0,;
```

```
 77.000             PROFILE='DFLPRF'
 78.000 DEFAULT BISYNC ;
 79.000             INPUT = YES,;
 80.000             OUTPUT = YES,;
 81.000             SPEED = 2400,;
 82.000             REMOTE = TERMINAL,;
 83.000             ENABLE = YES,;
 84.000             HARDWIRE = YES,;
 85.000             CLOCKING = NO,;
 86.000             REENABLE = YES,;
 87.000             READTIMEOUT = 0
 88.000 DEFAULT HDLCX25 ;
 89.000             INPUT = YES,;
 90.000             OUTPUT = YES,;
 91.000             SPEED = AUTO,;
 92.000             REMOTE = HOST,;
 93.000             ENABLE = NO,;
 94.000             HARDWIRE = YES,;
 95.000             CLOCKING = NO,;
 96.000             REENABLE = YES,;
 97.000             READTIMEOUT = 0
 98.000 CONFIG .4A80 HARDWIRE=NO,DROPDTR=YES,LOGONTIMEOUT=5,READTIMEOUT=2
 99.000 CONFIG .4880 HARDWIRE=NO,DROPDTR=YES,LOGONTIMEOUT=5,READTIMEOUT=2
100.000 CONFIG .5000 ENABLE=YES,LOGON='LNKT034A'   "Phoenix HLSUA RFEP
101.000 CONFIG .5100 ENABLE=YES,LOGON='LNKT034B'   "Phoenix HLSUA RFEP
102.000 CONFIG .5200 ENABLE=YES,LOGON='LNKT033A'   "Phoenix MEW-CRF RFEP
103.000 CONFIG .5300 ENABLE=YES,LOGON='LNKT033B'   "Phoenix MEW-CRF RFEP
104.000 CONFIG .1080 BUF=256    "rm 284  Fosnight
105.000 CONFIG .1800 SPEED=300,HARD=NO,LOGON='ALEXDEV PRO=ALEXPRO',SAL=NO  "ALEX modem
105.500 CONFIG .1880 SPEED=1200,HARD=NO,LOGON='ICOM1',SAL=NO
106.000 CONFIG .2800 BUF=256    "rm 212  Anderson
107.000 CONFIG .1580 BUF=128 "JLJ RM241 IBMPC"
107.500 SET NSHUF=324
108.000 SEL H=BISYNC
109.000 SET MEM=84
110.000 " L66C :NETCON - DEFAULT COMMANDS AND CONFIGS FOR FEP 6 " ;
111.000 SEL NODE=6
112.000 SET BOOTINFO
113.000 M:FEP.:SYS
```

```
116 -    114.000 5
117 -    115.000 NODEADMN
118 -    116.000 NODEADMN.:SYS
119 -    117.000 COUPLER
120 -    118.000 COUPLER.:SYS
121 -    119.000 ASYNC
122 -    120.000 ASYNC.:SYS
123 -    121.000 BISYNC
124 -    122.000 BISYNC.:SYS
125 -    123.000 HDLCX25
126 -    124.000 HDLCX25.:SYS
127 -    125.000 :SYS
128 -    126.000 DEFAULT ASYNC ;
129 -    127.000           INPUT = YES,;
130 -    128.000           OUTPUT = YES,;
131 -    129.000           SALUTATION = YES,;
132 -    130.000           SPEED = AUTO,;
133 -    131.000           ENABLE = YES,;
134 -    132.000           REENABLE = YES,;
135 -    133.000           DROPDTR = NO,;
136 -    134.000           HARDWIRE = YES,;
137 -    135.000           ECHOLOGON = NO,;
138 -    136.000           LOGONTIMEOUT = 5,;
139 -    137.000           READTIMEOUT = 0,;
140 -    138.000           PROFILE='DFLPRF'
141 -    139.000 DEFAULT BISYNC ;
142 -    140.000           INPUT = YES,;
143 -    141.000           OUTPUT = YES,;
144 -    142.000           SPEED = 2400,;
145 -    143.000           REMOTE = TERMINAL,;
146 -    144.000           ENABLE = YES,;
147 -    145.000           HARDWIRE = YES,;
148 -    146.000           CLOCKING = NO,;
149 -    147.000           REENABLE = YES,;
150 -    148.000           READTIMEOUT = 0
151 -    149.000 DEFAULT HDLCX25 ;
152 -    150.000           INPUT = YES,;
153 -    151.000           OUTPUT = YES,;
154 -    152.000           SPEED = AUTO,;
```

```
155 -    153.000           REMOTE = HOST,;
156 -    154.000           ENABLE = NO,;
157 -    155.000           HARDWIRE = YES,;
158 -    156.000           CLOCKING = NO,;
159 -    157.000           REENABLE = YES,;
160 -    158.000           READTIMEOUT = 0
161 -    159.000 CONFIG .4900 REMOTE=HOST,LOGON='L66A'
162 -    160.000 " L66A :NETCON - DEFAULT COMMANDS AND CONFIGS FOR FEP 8 " ;
163 -    161.000 SEL NODE=8
164 -    162.000 SET BOOTINFO
165 -    163.000 M:FEP.:SYS
166 -    164.000 5
167 -    165.000 NODEADMN
168 -    166.000 NODEADMN.:SYS
169 -    167.000 COUPLER
170 -    168.000 COUPLER.:SYS
171 -    169.000 ASYNC
172 -    170.000 ASYNC.:SYS
173 -    171.000 BISYNC
174 -    172.000 BISYNC.:SYS
175 -    173.000 HDLCX25
176 -    174.000 HDLCX25.:SYS
177 -    175.000 :SYS
178 -    176.000 DEFAULT ASYNC ;
179 -    177.000           INPUT = YES,;
180 -    178.000           OUTPUT = YES,;
181 -    179.000           SALUTATION = YES,;
182 -    180.000           SPEED = AUTO,;
183 -    181.000           ENABLE = YES,;
184 -    182.000           HARDWIRE = YES,;
185 -    183.000           REENABLE = YES,;
186 -    184.000           DROPDTR = NO,;
187 -    185.000           ECHOLOGON = NO,;
188 -    186.000           LOGONTIMEOUT = 5,;
189 -    187.000           READTIMEOUT = 0,;
190 -    188.000           PROFILE='DFLPRF'
191 -    189.000 DEFAULT BISYNC ;
192 -    190.000           INPUT = YES,;
193 -    191.000           OUTPUT = YES,;
```

```
         SPEED = 2400,;
         REMOTE = TERMINAL,;
         ENABLE = YES,;
         HARDWIRE = YES,;
         CLOCKING = NO,;
         REENABLE = YES,;
         READTIMEOUT = 0
DEFAULT HDLCX25 ;
         INPUT = YES,;
         OUTPUT = YES,;
         SPEED = AUTO,;
         REMOTE = HOST,;
         ENABLE = NO,;
         HARDWIRE = YES,;
         CLOCKING = NO,;
         REENABLE = YES,;
         READTIMEOUT = 0
CONFIG .0C00 BUF=256    "rm 2105 Sickler
CONFIG .1200 BUF=256    "rm 252  Catozzi
CONFIG .1980 BUF=256    "rm 2110 Hatfield
CONFIG .2880 LOGON='DEBUG12',HARD=NO,SAL=NO,SPEED=9600,DROP=YES,BUF=128,ENABLE=NO
CONFIG .2900 BUF=256    "rm 219  Towe
CONFIG .2C00 BUF=256    "rm 213  Heying
CONFIG .2F00 LOGON='DEBUG9',HARD=NO,SAL=NO,SPEED=9600,DROP=YES,BUF=128,ENABLE=NO
CONFIG .2F80 LOGON='DEBUG3',HARD=NO,SAL=NO,SPEED=9600,DROP=YES,BUF=128
CONFIG .3400 SPEED=1200,HARD=YES,SAL=NO,LOGON='name,acct,pass',PROFILE='TI855',LOGONTIMEOUT
=0
"CONFIG .3500 SPEED=1200,HARD=YES,SAL=NO,LOGON='name,acct,pass',PROFILE='TI855',LOGONTIMEOU
T=0
CONFIG .3500 SPEED=9600,HARD=YES,REENABLE=YES,LOGON='name,acct,pass',SAL=NO,LOGONTIMEOUT=0,
READTIMEOUT=0,PROFILE='QMS1200',FLOWC=YES
CONFIG .4980 LOGON='L66B'
CONFIG .4300 LOGON='LADC3270',SPEED=9600      "Upstairs MOD400 filetran Beaumont
CONFIG .4C00 ENABLE=YES,LOGON='LNKT032'     "Phoenix DVCP RFEP
SET NSHUF=324
SEL H=BISYNC
SET MEM=84
" L66A :NETCON - DEFAULT COMMANDS AND CONFIGS FOR FEP 32 " ;
SEL N=32
```

```
230 -    226.000 SET BOOTINFO
231 -    227.000 M:FEP.:SYS
232 -    228.000 4
233 -    229.000 NODEADMN
234 -    230.000 NODEADMN.:SYS
235 -    231.000 HDLCX25
236 -    232.000 HDLCX25.:SYS
237 -    233.000 ASYNC
238 -    234.000 ASYNC.:SYS
239 -    235.000 UNITREC
240 -    236.000 UNITREC.:SYS
241 -    237.000 :SYS
242 -    238.000 DEF LINK .F000,.F100
243 -    239.000 DEFAULT ASYNC ;
244 -    240.000            INPUT = YES,;
245 -    241.000            OUTPUT = YES,;
246 -    242.000            SALUTATION = YES,;
247 -    243.000            SPEED = AUTO,;
248 -    244.000            ENABLE = YES,;
249 -    245.000            REENABLE = YES,;
250 -    246.000            DROPDTR = NO,;
251 -    247.000            HARDWIRE = YES,;
252 -    248.000            ECHOLOGON = NO,;
253 -    249.000            LOGONTIMEOUT = 5,;
254 -    250.000            READTIMEOUT = 0,;
255 -    251.000            PROFILE='DFLPRF'
256 -    252.000 DEFAULT BISYNC ;
257 -    253.000            INPUT = YES,;
258 -    254.000            OUTPUT = YES,;
259 -    255.000            SPEED = 2400,;
260 -    256.000            REMOTE = TERMINAL,;
261 -    257.000            ENABLE = YES,;
262 -    258.000            HARDWIRE = YES,;
263 -    259.000            CLOCKING = NO,;
264 -    260.000            REENABLE = YES,;
265 -    261.000            READTIMEOUT = 0
266 -    262.000 DEFAULT HDLCX25 ;
267 -    263.000            INPUT = YES,;
268 -    264.000            OUTPUT = YES,;
```

```
269 -   265.000              SPEED = AUTO,;
270 -   266.000              REMOTE = HOST,;
271 -   267.000              ENABLE = NO,;
272 -   268.000              HARDWIRE = YES,;
273 -   269.000              CLOCKING = NO,;
274 -   270.000              REENABLE = YES,;
275 -   271.000              READTIMEOUT = 0
276 -   272.000 CONFIG .0580 LOGON='name,acct,pass'
277 -   273.000 CONFIG .0680 LOGON='name,acct,pass'
278 -   274.000 CONFIG .0700 LOGON='name,acct,pass'
279 -   275.000 CONFIG .F000 LOGON='LINK32',ENABLE=YES,REENABLE=YES
280 -   276.000 CONFIG .F100 LOGON='LINK32',ENABLE=YES,REENABLE=YES
281 -   277.000 " L66A :NETCON - DEFAULT COMMANDS AND CONFIGS FOR FEP 33 " ;
282 -   278.000 SEL N=33
283 -   279.000 SET BOOTINFO
284 -   280.000 M:FEP.:SYS
285 -   281.000 4
286 -   282.000 NODEADMN
287 -   283.000 NODEADMN.:SYS
288 -   284.000 HDLCX25
289 -   285.000 HDLCX25.:SYS
290 -   286.000 ASYNC
291 -   287.000 ASYNC.:SYS
292 -   288.000 UNITREC
293 -   289.000 UNITREC.:SYS
294 -   290.000 :SYS
295 -   291.000 DEF LINK .9000,.9800
296 -   292.000 DEFAULT ASYNC ;
297 -   293.000              INPUT = YES,;
298 -   294.000              OUTPUT = YES,;
299 -   295.000              SALUTATION = YES,;
300 -   296.000              SPEED = AUTO,;
301 -   297.000              ENABLE = YES,;
302 -   298.000              REENABLE = YES,;
303 -   299.000              DROPDTR = NO,;
304 -   300.000              HARDWIRE = YES,;
305 -   301.000              ECHOLOGON = NO,;
306 -   302.000              LOGONTIMEOUT = 15,;
307 -   303.000              READTIMEOUT = 0,;
```

```
304.000          PROFILE='DFLPRF'
305.000 DEFAULT BISYNC ;
306.000          INPUT = YES,;
307.000          OUTPUT = YES,;
308.000          SPEED = 2400,;
309.000          REMOTE = TERMINAL,;
310.000          ENABLE = YES,;
311.000          HARDWIRE = YES,;
312.000          CLOCKING = NO,;
313.000          REENABLE = YES,;
314.000          READTIMEOUT = 0
315.000 DEFAULT HDLCX25 ;
316.000          INPUT = YES,;
317.000          OUTPUT = YES,;
318.000          SPEED = AUTO,;
319.000          REMOTE = HOST,;
320.000          ENABLE = NO,;
321.000          HARDWIRE = YES,;
322.000          CLOCKING = NO,;
323.000          REENABLE = YES,;
324.000          READTIMEOUT = 0
325.000 CONFIG .0580 LOGON='name,acct,pass'
326.000 CONFIG .0B80 LOGON='name,acct,pass'
327.000 CONFIG .6980 LOGON='name,acct,pass',HARDWIRE=YES,SPEED=4800,READTIME=0,LOGONTIME=0
328.000 CONFIG .9000 LOGON='LINK33A',ENABLE=YES,REENABLE=YES
329.000 CONFIG .9800 LOGON='LINK33B',ENABLE=YES,REENABLE=YES
330.000 " L66A :NETCON - DEFAULT COMMANDS AND CONFIGS FOR FEP 34 " ;
331.000 SEL N=34
332.000 SET BOOTINFO
333.000 M:FEP.:SYS
334.000 4
335.000 NODEADMN
336.000 NODEADMN.:SYS
337.000 HDLCX25
338.000 HDLCX25.:SYS
339.000 ASYNC
340.000 ASYNC.:SYS
341.000 UNITREC
342.000 UNITREC.:SYS
```

```
347 -    343.000 :SYS
348 -    344.000 DEF LINK .8000,.8100,.9000,.9100
349 -    345.000 DEFAULT ASYNC ;
350 -    346.000              INPUT = YES,;
351 -    347.000              OUTPUT = YES,;
352 -    348.000              SALUTATION = YES,;
353 -    349.000              SPEED = AUTO,;
354 -    350.000              ENABLE = YES,;
355 -    351.000              REENABLE = YES,;
356 -    352.000              DROPDTR = NO,;
357 -    353.000              HARDWIRE = YES,;
358 -    354.000              ECHOLOGON = NO,;
359 -    355.000              LOGONTIMEOUT = 15,;
360 -    356.000              READTIMEOUT = 0,;
361 -    357.000              PROFILE='DFLPRF'
362 -    358.000 DEFAULT BISYNC ;
363 -    359.000              INPUT = YES,;
364 -    360.000              OUTPUT = YES,;
365 -    361.000              SPEED = 2400,;
366 -    362.000              REMOTE = TERMINAL,;
367 -    363.000              ENABLE = YES,;
368 -    364.000              HARDWIRE = YES,;
369 -    365.000              CLOCKING = NO,;
370 -    366.000              REENABLE = YES,;
371 -    367.000              READTIMEOUT = 0
372 -    368.000 DEFAULT HDLCX25 ;
373 -    369.000              INPUT = YES,;
374 -    370.000              OUTPUT = YES,;
375 -    371.000              SPEED = AUTO,;
376 -    372.000              REMOTE = HOST,;
377 -    373.000              ENABLE = NO,;
378 -    374.000              HARDWIRE = YES,;
379 -    375.000              CLOCKING = NO,;
380 -    376.000              REENABLE = YES,;
381 -    377.000              READTIMEOUT = 0
382 -    378.000 CONFIG .8000 LOGON='LINK34A',ENABLE=YES,REENABLE=YES
383 -    379.000 CONFIG .8100 LOGON='LINK34B',ENABLE=YES,REENABLE=YES
384 -    380.000 CONFIG .9000 LOGON='LINK34A',ENABLE=YES,REENABLE=YES
385 -    381.000 CONFIG .9100 LOGON='LINK34B',ENABLE=YES,REENABLE=YES
```

```
386 -    382.000 END
```

```
 0.500 !"
 0.510 !"        Define the host node number and name for NETCON.
 0.520 !"
 0.530 !LET      HOST_NODE                = 0
 0.540 !LET      HOST_NAME                = 'L66A'
 0.550 !GLOBAL   %('HOST_NODE'||'$')      = '%HOST_NODE'
 0.560 !GLOBAL   %('HOST_NAME'||'$')      = '%HOST_NAME'
 1.000 !"
 2.000 !"        Define the local FEP node number and name which the RFEP
 3.000 !"        is connected to.  Define the channel(s) that is(are)
 3.100 !"        used to connect to the remote FEP.
 4.000 !"
 5.000 !LET      FEP_NODE                 = 1
 6.000 !LET      FEP_NAME                 = 'NODE0001'
 6.100 !LET      FEP_CHAN                 = '.F000'
 7.000 !GLOBAL   %('FEP_NODE'||'$')       = '%FEP_NODE'
 8.000 !GLOBAL   %('FEP_NAME'||'$')       = '%FEP_NAME'
 8.100 !GLOBAL   %('FEP_CHAN'||'$')       = '%FEP_CHAN'
 9.000 !"
10.000 !"        Define the remote FEP (RFEP) node number and name.  Define
10.100 !"        the channel(s) that is(are) used to connect to the local FEP.
11.000 !"
12.000 !LET      RFEP_NODE                = 34
13.000 !LET      RFEP_NAME                = 'CP6FEP'
13.100 !LET      RFEP_CHAN                = '.F000'
14.000 !GLOBAL   %('RFEP_NODE'||'$')      = '%RFEP_NODE'
15.000 !GLOBAL   %('RFEP_NAME'||'$')      = '%RFEP_NAME'
15.100 !GLOBAL   %('RFEP_CHAN'||'$')      = '%RFEP_CHAN'
16.000 !"
17.000 !"        Define the profile names for the link profiles at the
18.000 !"        local and remote ends of the links.  (11 char max.)
19.000 !"
20.000 !LET      LOCAL_LINK               = 'LOCALLINK'
21.000 !LET      REMOTE_LINK              = 'REMOTELINK'
22.000 !GLOBAL   %('LOCAL_LINK'||'$')     = '%LOCAL_LINK'
23.000 !GLOBAL   %('REMOTE_LINK'||'$')    = '%REMOTE_LINK'
24.000 !"
```

```
!"          Define the virtual circuit names for the virtual circuits
!"          at the local and remote ends of the links.  (11 char max.)
!"          Note:  The remove vc are not actually generated because
!"          changing the remote vc would involve rewritting the
!"          diskette every time the remote vc was changed.
!"
!LET        LOCAL_VC                  = 'LOCALVC'
!LET        REMOTE_VC                 = 'REMOTEVC'
!GLOBAL     %('LOCAL_VC'||'$')        = '%LOCAL_VC'
!GLOBAL     %('REMOTE_VC'||'$')       = '%REMOTE_VC'
!"
!"          Form the names of the links from the FEP to the RFEP and
!"          from the RFEP to the FEP.  The name is of the form
!"          LINKmmnn, where 'mm' is the two-digit node number the
!"          link is from, and 'nn' is the two-digit node number the
!"          link is to.
!"
!LET        NN                        = '0' || FEP_NODE
!LET        FEP_NODEXX                = $SUBSTR ( NN, $LEN(NN)-2, 2 )
!LET        NN                        = '0' || RFEP_NODE
!LET        RFEP_NODEXX               = $SUBSTR ( NN, $LEN(NN)-2, 2 )
!LET        FEP_RFEP_LINK             = 'LINK' || FEP_NODEXX || RFEP_NODEXX
!LET        RFEP_FEP_LINK             = 'LINK' || RFEP_NODEXX || FEP_NODEXX
!GLOBAL     %('FEP_RFEP_LINK'||'$') = '%FEP_RFEP_LINK'
!GLOBAL     %('RFEP_FEP_LINK'||'$') = '%RFEP_FEP_LINK'
!"
!"          Form the X.25 address for the local and remote ends of
!"          the link.  The address is of the form mmnnii, where 'mm'
!"          is the two-digit node number the address is to, 'nn' is
!"          the two-digit node number the address is from, and 'ii'
!"          is the number of the physical link (i.e., 01, 02, etc.)
!"
!LET        LOCAL_ADDRESS             = FEP_NODEXX || RFEP_NODEXX || '01'
!LET        REMOTE_ADDRESS            = RFEP_NODEXX || FEP_NODEXX || '01'
!GLOBAL     %('LOCAL_ADDRESS'||'$') = '%LOCAL_ADDRESS'
!GLOBAL     %('REMOTE_ADDRESS'||'$') = '%REMOTE_ADDRESS'
```

```
 1.000 !ECHO
 4.000 !DRIBBLE OVER RFEP_NETCON_DRIBBLE
 7.000 !IF       'FEP_NODE$' = '%('FEP_NODE'||'$')' THEN XEQ RFEP_GLOBALS
 8.000 !COPY     ME                                 over *RFEP_NETCON
 9.000 !!NETCON
 9.500  delete  node = HOST_NODE$
 9.510  define  node = HOST_NODE$, name = HOST_NAME$, type = me
10.000  delete  node = RFEP_NODE$
11.000  define  node = RFEP_NODE$, name = RFEP_NAME$, type = fep, control = HOST_NAME$
12.000  update
12.500  select  node = FEP_NODE$
12.510  define  link  FEP_CHAN$
12.520  config  FEP_CHAN$ ;
12.530          enable = yes, ;
12.540          logon = 'FEP_RFEP_LINK$', ;
12.550          reenable = yes
13.000  select  node = RFEP_NODE$
14.000  set     bootinfo
15.000 M:FEP.:SYS
16.000 3
17.000 NODEADMN
18.000 NODEADMN.:SYS
19.000 HDLCX25
20.000 HDLCX25.:SYS
21.000 ASYNC
22.000 ASYNC.:SYS
23.000 :SYS
24.000  default ASYNC ;
25.000          breakrequired = no, ;
26.000          bufsize = 256, ;
27.000          dropdtr = no, ;
28.000          echologon = no, ;
29.000          enable = yes, ;
30.000          hardwire = yes, ;
31.000          input = yes, ;
32.000          logontimeout = 0, ;
33.000          output = yes, ;
```

```
34.000                  profile = 'DFLPRF', ;
35.000                  readtimeout = 0, ;
36.000                  reenable = yes, ;
37.000                  salutation = yes, ;
38.000                  speed = auto
39.000  default HDLCX25 ;
40.000                  breakrequired = no, ;
41.000                  clocking = no, ;
42.000                  enable = yes, ;
43.000                  hardwire = no, ;
44.000                  input = yes, ;
45.000                  output = yes, ;
46.000                  readtimeout = 0, ;
47.000                  reenable = yes, ;
48.000                  speed = auto
49.000  select  handler = ASYNC
50.000  set     RCVCQBYTSIZ = 16384
51.000  set     NSHUF = 600
52.000  define  link  RFEP_CHAN$
53.000  config  RFEP_CHAN$ ;
53.010          enable = yes, ;
53.020          logon = 'RFEP_FEP_LINK$', ;
53.030          reenable = yes
57.000  end
58.000 !EOD
59.000 !XEQ     *RFEP_NETCON
60.000 !DONT    DRIBBLE
61.000 !ELBBIRD.X RFEP_NETCON_DRIBBLE
62.000 !DIR     R
```

```
 1.000 !ECHO
 2.000 !DRIBBLE OVER RFEP_SUPER_DRIBBLE
 3.000 !IF       'FEP_NODE$' = '%('FEP_NODE'||'$')' THEN XEQ RFEP_GLOBALS
 4.000 !COPY     ME                                  over *RFEP_SUPER
 5.000 !!SUPER
 6.000 "
 7.000 "                            Profile for local end of links
 8.000 "
 9.000  remove  profile LOCAL_LINK$
10.000  create  profile LOCAL_LINK$ link
11.000   circuits = 10
12.000   default packet size = 1024
13.000   default response timer = 0
14.000   default window = 7
15.000   frame = 1024
16.000   mode = DTE
17.000   retransmission = 10
18.000   timeout = 3
19.000   window = 7
20.000   end
21.000 "
22.000 "                            Profile for RFEP end of links
23.000 "
24.000  remove  profile REMOTE_LINK$
25.000  create  profile REMOTE_LINK$ link
26.000   circuits = 10
27.000   default packet size = 1024
28.000   default response timer = 0
29.000   default window = 7
30.000   frame = 1024
31.000   mode = DCE
32.000   retransmission = 10
33.000   timeout = 3
34.000   window = 7
35.000   end
36.000 "
37.000 "                            Profile for local end of virtual circuit
```

```
38.000 "
39.000  remove  profile LOCAL_VC$
40.000  create  profile LOCAL_VC$ virtual circuit
41.000   delays = 2
42.000   maxvircir = 6
43.000   minvircir = 1
44.000   receive size = 1024
45.000   receive window = 7
46.000   respond to complete = yes
47.000   response delay = 2
48.000   response timer = 0
49.000   retrys = 0
50.000   send size = 1024
51.000   send window = 7
52.000   timeout = 3
53.000   type = primary
54.000   end
55.000 "
56.000 "                        Profile for remote end of virtual circuit
57.000 "
58.000  remove  profile REMOTE_VC$
59.000  create  profile REMOTE_VC$ virtual circuit
60.000   delays = 2
61.000   maxvircir = 6
62.000   minvircir = 1
63.000   receive size = 1024
64.000   receive window = 7
65.000   respond to complete = yes
66.000   response delay = 2
67.000   response timer = 0
68.000   retrys = 0
69.000   send size = 1024
70.000   send window = 7
71.000   timeout = 3
72.000   type = secondary
73.000   end
74.000 "
75.000 "                        Links for local end of link to RFEP_NAME$
76.000 "
```

```
77.000 remove  link FEP_RFEP_LINK$
78.000 create  link FEP_RFEP_LINK$
79.000  address = LOCAL_ADDRESS$
80.000  profile = LOCAL_LINK$
81.000  end
82.000 "
83.000 "                       Links for remote end of link to RFEP_NAME$
84.000 "
85.000 remove  link RFEP_FEP_LINK$
86.000 create  link RFEP_FEP_LINK$
87.000  address = REMOTE_ADDRESS$
88.000  profile = REMOTE_LINK$
89.000  end
90.000 "
91.000 "                       Virtual circuits for local end of link
92.000 "
93.000 remove  virtual circuit 1 for link FEP_RFEP_LINK$
94.000 create  virtual circuit 1 for link FEP_RFEP_LINK$
95.000  address = REMOTE_ADDRESS$
96.000  destination = RFEP_NAME$
97.000  profile = LOCAL_VC$
98.000  end
98.500 create  virtual circuit 2 for link FEP_RFEP_LINK$
98.510  address = REMOTE_ADDRESS$
98.520  destination = RFEP_NAME$
98.530  profile = LOCAL_VC$
98.540  end
99.000 "
100.000 "                      Virtual circuits for remote end of link
101.000 "
102.000 " remove  virtual circuit 1 for link RFEP_FEP_LINK$
103.000 " create  virtual circuit 1 for link RFEP_FEP_LINK$
104.000 "  address = LOCAL_ADDRESS$
105.000 "  destination = FEP_NAME$
106.000 "  profile = REMOTE_VC$
107.000 "  end
108.000 end
109.000 !EOD
110.000 !XEQ     *RFEP_SUPER
```

12HO-26

```
116 -   111.000 !DONT     DRIBBLE
117 -   112.000 !ELBBIRD.X REEP_SUPER_DRIBBLE
118 -   113.000 !DIR      R
```

```
 2.000 !LET     ECHO = $FLAG ( ECHO )
 3.000 !DONT    ECHO
 4.500 !WHAT.X  PA
 8.000 !FEP:
 9.000 !LET     FEP = $INPUT ( 'Which FEP is diskette to be built ON  (DVFEP|CFEP|CP6FEP):  ' )
10.000 !IF      FEP = ''                         THEN GOTO END
11.000 !IF      FEP = 'DVFEP'                    THEN GOTO FEP_END
12.000 !IF      FEP = 'CFEP'                     THEN GOTO FEP_END
13.000 !IF      FEP ~= 'CP6FEP'                  THEN GOTO FEP
14.000 !FEP_END:
16.000 !RFEP:
17.000 !LET     RFEP = $INPUT ( 'Which RFEP is diskette to be built FOR (DVFEP|CRFEP|CP6FEP):  ' )
18.000 !IF      RFEP = ''                        THEN GOTO END
19.000 !IF      RFEP = 'DVFEP'                   THEN GOTO RFEP_END
20.000 !IF      RFEP = 'CRFEP'                   THEN GOTO RFEP_END
21.000 !IF      RFEP ~= 'CP6FEP'                 THEN GOTO RFEP
22.000 !RFEP_END:
45.000 !DRIVE1:
46.000 !OUTPUT  'Put the new diskette for %RFEP in drive 1.'
47.000 !LET     ANS = $INPUT ( 'Enter GO when diskette installed:  ' )
48.000 !IF      ANS ~= 'GO'                      THEN GOTO DRIVE1
49.000 !DRIVE1_END:
50.000 !BATCH:
51.000 !LET     NAME = RFEP || '_DISKETTE'
52.000 !BATCH   J:RFEP_DISKETTE         %('NAME'||'$') = '%NAME', ;
53.000 !                                %('FEP'||'$') = '%FEP', ;
54.000 !                                %('RFEP'||'$') = '%RFEP'
66.000 !END:
67.000 !IF      ECHO = 'YES'            THEN ECHO
```

```
 1.000 !DEFAULT NAME$=RFEP_DISKETTE
 2.000 !DEFAULT WSN$=DVPSIG3, DEFER$=0:00, SCHED$=RERUN
 3.000 !DEFAULT TIME$=1:00, MEM$=64, FPOOLS$=31
 4.000 !JOB     NAME=NAME$, WSN=WSN$, DEFER=(DEFER$), SCHED$
 5.000 !RES     TIME=TIME$, MEM=MEM$
 6.000 !LIMIT   FPOOLS=FPOOLS$
 7.000 !M       Job to build RFEP diskette for RFEP$.
 8.000 !M       Diskette will be built on drive 1 of FEP$.
13.000 !PIGETTE
14.000  use     fep FEP$
15.000  time
16.000  build   RFEP$ on fep FEP$ drive 1
17.000  time
18.000  list    fep FEP$ drive 1
19.000  end
20.000 !copy    me over *mess
21.000 TO:Me
22.000 SUBJECT:NAME$ completed
23.000 FCOPY:NO
24.000 RR:NO
25.000 !EOD
26.000 !SEND    *mess
```

LAB OVERVIEW

Lab formats generally take the form of:

Scenario
- where scenario is as real-to-life as we can make it

Discussion/hints
- where we are as obscure as possible

Problem
- where we really let you have it

LAB 1

ASSIGNMENT

LAB #1 [SUPER]

SCENARIO:

- You have been asked to make a special line printer form for a room full of schmucks. The schmucks are divided into groups of two, known as schmucklets. Each schmucklet has a different logon account and would like their output to come out uniquely identified.

  What the heck can you do?

LAB #1 [SUPER]

DISCUSSION:

- You don't want to have to change the forms man every few minutes. [Hint: What's a pseudo form?]

- You'd like to be able to use the same form for all schmucklets, but uniquely identify each schmucklet's output. [Hint: What's a WSN?]

LAB #1 [SUPER]

DISCUSSION:

- You don't want to confuse the poor development programmers who tear listings off the damn printers? [Hint: What's a DFORM? For which WSN?]

- You'd like the listings to be identifiable by the members of the schmucklet [schmucklettes]. How can that be done? [Hint: What's a BANNERTEXT?]

LAB #1 [SUPER]

PROBLEM:

Create a form named U40XXX, where XXX is your group number. It must look a lot like the standard form at your W00, and must be a pseudo form for that form. Test the form, using LDEV. When you're done, the instructor will continue with a group exercise designed to make a single form the permanent default for all schmucklets for the rest of the sessions.

LAB 1

ANSWER

```
!SUPER
   R FORM STDINT
   CR PSEUDO STDINT FOR STDLP
      BANNER
         REPEATS = 2
         ENTRY = 45, 1, 0, 1, 0, 1, 0, 0
%$
         Entry = 01, 001, 0, 1, 0, 1, 0, 0
%DATE
         Entry = 01, 010, 0, 1, 0, 1, 0, 0
%TIME
         Entry = 01, 016, 0, 1, 0, 1, 0, 0
%ACCN
         Entry = 01, 025, 0, 1, 0, 1, 0, 0
%SYSID
         Entry = 01, 031, 0, 1, 0, 1, 0, 0
.
         Entry = 01, 032, 0, 1, 0, 1, 0, 0
%SUBFILE
         Entry = 01, 039, 0, 1, 0, 1, 0, 0
(
         Entry = 01, 040, 0, 1, 0, 1, 0, 0
%JOBNAME
         Entry = 01, 071, 0, 1, 0, 1, 0, 0
)
         Entry = 01, 074, 0, 1, 0, 1, 0, 0
WOO=
         Entry = 01, 078, 0, 1, 0, 1, 0, 0
%WOO
         Entry = 01, 088, 0, 1, 0, 1, 0, 0
WOD=@
         Entry = 01, 093, 0, 1, 0, 1, 0, 0
%WOD
         Entry = 01, 103, 0, 1, 0, 1, 0, 0
CP-6/
         Entry = 01, 108, 0, 1, 0, 1, 0, 0
%VERSION
         ENTRY = 4, 1, 0, 1, 0, 1, 0, 0
Schmucklet # 1 is
         ENTRY = 4, 19, 0, 1, 0, 1, 0, 0
%BANNERTEXT1
         ENTRY = 5, 1, 0, 1, 0, 1, 0, 0
Schmucklet # 2 is
         ENTRY = 5, 19, 0, 1, 0, 1, 0, 0
%BANNERTEXT2
         ENTRY = 10, 16, 2, 3, 2
CP-6
         ENTRY = 24, 10, 1, 2, 1
Internals
         ENTRY = 33, 1, 1, 2, 1, 1, 0, 0
%BANNERTEXT3
      END
   END
END
!"
!SUPER
CR WSN INTERNAL
DEVICE=LP@UPSTAIRS
DFORM LP STDINT
END
MOD ZZINT?
WSN=INTERNAL

END
```

LAB 2

ASSIGNMENT

LAB #2 [SHARED LIBRARIES]

SCENARIO:

- The head of the computer science department just read about shared libraries in the CP-6 System Support manual. He's absolutely sure that the computer science graphics class can benefit from your writing a shared library for them. Who are you to argue? As an exercise, you decide to write your own simple shared library to learn the "ins and outs" of shared library creation.

LAB #2 [SHARED LIBRARIES]

PROBLEM:

The rest of the specifics of the problem are described in detail in the handouts. Bonus points are awarded for stealing the System Support Manual that the head of the computer science department was reading.

17:43 AUG 29 '85 LAB2_ASSIGNMENT

U40 (CP-6 INTERNALS) CLASS

SHARED LIBRARIES LAB

1) Make a subroutine library. The library will have 5 entry points:

a) OPEN_DCB - this subroutine accepts a DCB number from the calling program, and opens the DCB to the timesharing terminal.

b) WRITE_ALINE - this subroutine accepts a BUFFER and a SIZE and writes the SIZE number of bytes from the BUFFER through the DCB previously passed to the OPEN_DCB subroutine.

c) READ_ALINE - this subroutine accepts a BUFFER and a maximum SIZE from the calling program and performs a read through the DCB opened in OPEN_DCB. The number of bytes read is returned through the SIZE parameter passed to the routine.

d) CLOSE_DCB - closes the DCB opened by OPEN_DCB.

e) EXIT_ALL - performs an M$EXIT.

HINT: This library is best written in PL-6.

HINT: See the DCBNUM function in the PL-6 reference manual.

HINT: You must get the DCB's number from the calling program.

HINT: Your lab instructor has included a listing of his solution to the problem in this handout.

HINT: Save the object unit(s) from this exercise, as you'll need them below.

2) Use LEMUR to build an UNSHARED, SUBROUTINE library from the object unit(S) created from exercise 1.

The UNSHARED version of your library will be named:

:LIB_U40nn

where:

nn   is your group number

HINT: See the LEMUR and LINK sections of the PROGRAMMER reference manual.

3) Write a MAIN program (in PL-6) that uses this library. Link your MAIN program with the UNSHARED version of the library. Does it check out okay?

HINT: This isn't even a shared library exercise. Use the standard object unit linking commands from the PROGRAMMER REFERENCE manual.

HINT: Save the "MAIN" program OU from this exercise. Use a star file for the run unit.

4) Use LINK to create a shareable version of the library. Place the shareable version of the library in a file called:

:SHARED_U40nn

where

nn is your group number

HINT: See the LINK section of the PROGRAMMERs REFERENCE MANUAL.

HINT: See your Lab Instructor's solution to the problem (in the handout)

HINT: Ask yourself why the person who wrote the Lab Instructor's example extracted X66_AUTO_O from :LIB_SYSTEM.:SYS and then used that in the link. What is it anyway? Maybe the explanation of what's in :LIBRARY in the back of the Monitor Services Reference Manual will tell you.

5) Use SPIDER to install the shareable version of the library on the running machine. Use the same name to install the library as it is already named.

HINT: See the instructor's handout to see how he did it.

HINT: Ask yourself why he deletes it before installing it.

HINT: Don't use SPIDER to mess with anything else on the system. The Lab Instructor used to teach martial arts and small arms weaponry for the CIA.

6) Link your MAIN program OU (from exercise 3) with the shared library you just created. Run it. SAVE THE RUN UNIT FROM THIS EXERCISE FOR LATER. YOU MAY NOT RELINK THIS RUN UNIT ONCE IT HAS RUN CORRECTLY.

HINT: You may, once again, find the Lab Instructor's example handy.

HINT: Ask questions about anything you don't understand.

HINT: Why are you linking with the SH option? The UNSAT option?

7) Use the MOD command (of PCL) to change the name of your library to

:SHARED_U40nn_OLD.

HINT: See MODIFY under PCL in the PROGRAMMER REFERENCE MANUAL.

8) Modify your subroutine library. You have determined that you don't like the prompt character issued by the system. You have also decided that you need your EXIT_ALL routine to display the fact that it indeed has exited. You decide the way to do this is to write a 'GOOM-BYE' message to the terminal.

Change your library accordingly. Recreate the OU. Check out the new, unshared version of the library.

HINT: See M$PROMPT.

HINT: What DCBs can a shared library reference?

9) Relink the shared version of your library as

:SHARED_U40nn

Remember that you must keep the "transfer vector" intact. How can you do this?

HINT: See the VECTOR option of LINK, PROGRAMMER REFERENCE MANUAL.

HINT: Remember that you saved your old library as
:SHARED_U40nn_OLD.

10) Replace the old library (in memory) with the new library.

HINT: Use SPIDER.

11) Re-run the run unit you created in exercise 6. YOU MAY NOT RELINK THIS RUN UNIT. Does it work correctly? Why? Why not? Aren't I frustrating?

HINT: Transfer vectors are real important.

17:44 AUG 29 '85 LAB2_JCL

```
!JOB WSN=LOCAL
!RES MEM=300,TIME=2
!PL6 LAB21_SI6 OVER LAB21_OU6(LS,SR(.:LIBRARY),SCHEMA)
!LEMUR
COPY :LIB_SYSTEM.:SYS(X66_AUTO_0) INTO *LIB
END
!LINK LAB21_OU6,*LIB ;
      OVER :SHARED_INT (MAP(VALUE,NAME),NODEBUG,NOSH,SLIB,REP=0, ;
      VECTOR(ENTRIES(OPEN_DCB,WRITE_ALINE,READ_ALINE,CLOSE_DCB,EXIT_ALL),;
      REMOVE_E,REMOVE_S,DATA=247))
!PL6 LAB2M_SI6 OVER LAB2M_OU6(LS,SR(.:LIBRARY),SCHEMA)
!LINK LAB2M_OU6 OVER LAB2M_RU(SHAREL=:SHARED_INT.ZZINT,;
      UNSAT=:LIB_SYSTEM.:SYS)
!SPIDER
DEL :SHARED_INT
IN :SHARED_INT,LIB FROM :SHARED_INT.ZZINT
L :SHARED_INT
END
!LAB2M_RU.
TUNA
!C LAB2_JCL TO LP(K)
```

LAB 2

ANSWER

```
/*M* JJ'S SHARED LIBRARY (PHASE 1) */
/*T***********************************************************
 *T*                                                         *
 *T* COPYRIGHT, (C) HONEYWELL INFORMATION SYSTEMS INC., 1981 *
 *T*                                                         *
 *T***********************************************************/
/*X* DMR,PLM=5,IND=5,CTI=5,SDI=5,MCL=10,CSI=0,ECI=0 */

%SET LISTSUB='1'B;

OPEN_DCB: PROC (DCBNO);

DCL DCBNO SBIN WORD;

     %INCLUDE CP_6;

     %FPT_OPEN (FPTN=OPEN_DCBNO,
                        CTG=YES,
                        RES='ME',
                        DISP=NAMED);

/* GLOBAL DATA DEFINITIONS */

DCL GLOBAL_DCBNO SBIN WORD STATIC SYMDEF;

                                   /*
                                        BEGIN PROCEDURE
                                   */

     GLOBAL_DCBNO = DCBNO;

     OPEN_DCBNO.V.DCB# = GLOBAL_DCBNO;

     CALL M$OPEN (OPEN_DCBNO)
          WHENALTRETURN
     DO;
          CALL M$MERC;
```

```
            CALL M$EXIT;
          END;                                    /* DO IF ALTRETURN */

      RETURN;

END OPEN_DCB;
%EOD;
%SET LISTSUB = '1'B;

READ_ALINE: PROC (BUF_, SIZE_);

DCL BUF_ CHAR(132);

DCL SIZE_ SBIN WORD;

DCL 1 BUFFER_ BASED,
          2 BUFFER_CHARS_ CHAR(SIZE_) CALIGNED;


      %INCLUDE CP_6;

/* GLOBAL REFERENCES */

DCL GLOBAL_DCBNO SBIN WORD SYMREF;

/* LOCAL FPTS */

      %FPT_READ       (FPTN=READ_ALINE_,
                           WAIT=YES);

          %F$DCB;

                                        /*
                                              BEGIN PROCEDURE
                                        */

      READ_ALINE_.V.DCB# = GLOBAL_DCBNO;

      READ_ALINE_.BUF_.BUF$ = ADDR(BUF_);

      READ_ALINE_.BUF_.BOUND = SIZE_ - 1;

      IF SIZE_ <= 0
      THEN
      DO;
            READ_ALINE_.BUF_ = VECTOR (NIL);
            ADDR(BUF_) -> BUFFER_.BUFFER_CHARS_ = ' ';
         END;                                    /* DO IF SIZE <= 0 */

      CALL M$READ (READ_ALINE_)
            WHENALTRETURN
      DO;
            CALL M$MERC;
            CALL M$EXIT;
         END;                                    /* DO IF ALTRETURN */

      IF SIZE_ <= 0
      THEN
            SIZE_ = 0;
      ELSE
            SIZE_ = DCBADDR(GLOBAL_DCBNO) -> F$DCB.ARS#;

      RETURN;

END READ_ALINE;
%EOD;

%SET LISTSUB='1'B;

WRITE_ALINE: PROC (BUF_, SIZE_);

DCL BUF_ CHAR(132);

DCL SIZE_ SBIN WORD;

/* GLOBAL REFERENCES */

DCL GLOBAL_DCBNO SBIN WORD SYMREF;
```

```

     %INCLUDE CP_6;

     %FPT_WRITE         (FPTN=WRITE_ALINE_,
                              WAIT=YES);

                                   /*
                                        BEGIN PROCEDURE
                                   */

     IF SIZE_ <= 0
     THEN
          WRITE_ALINE_.BUF_ = VECTOR(NIL);
     ELSE
     DO;
          WRITE_ALINE_.BUF_.BOUND = SIZE_ - 1;
          WRITE_ALINE_.BUF_.BUF$ = ADDR(BUF_);
        END;                                    /* DO IF SIZE > 0 */

     WRITE_ALINE_.V.DCB# = GLOBAL_DCBNO;

     CALL M$WRITE (WRITE_ALINE_)
          WHENALTRETURN
     DO;
          CALL M$MERC;
          CALL M$EXIT;
        END;                                    /* DO IF ALTRETURN */

     RETURN;
END WRITE_ALINE;
%EOD;

%SET LISTSUB='1'B;

CLOSE_DCB: PROC;

/* GLOBAL REFERENCES */

DCL GLOBAL_DCBNO SBIN WORD SYMREF;
```

```

     %INCLUDE CP_6;

     %FPT_CLOSE    (FPTN=CLOSE_DCB_,
                         DISP=SAVE);

                                   /*
                                        BEGIN PROCEDURE
                                   */
     CLOSE_DCB_.V.DCB# = GLOBAL_DCBNO;

     CALL M$CLOSE (CLOSE_DCB_)
          WHENALTRETURN
     DO;
          CALL M$MERC;
          CALL M$EXIT;
        END;                            /* DO IF ALTRETURN */

     RETURN;
END CLOSE_DCB;

%EOD;

EXIT_ALL: PROC;

     %INCLUDE CP_6;

     CALL M$EXIT;

END EXIT_ALL;
```

```
/*M* JJ'S SHARED LIBRARY TEST PROGRAM */
/*T***********************************************************
 *T*                                                         *
 *T* COPYRIGHT, (C) HONEYWELL INFORMATION SYSTEMS INC., 1981 *
 *T*                                                         *
 *T***********************************************************/
/*X* DMR,PLM=5,IND=5,CTI=5,SDI=5,MCL=10,CSI=0,ECI=0 */


%SET LISTSUB='1'B;


MYPROG: PROC MAIN;


     /* LIBRARY ENTRY POINTS */

DCL OPEN_DCB ENTRY(1);
DCL READ_ALINE ENTRY(2);
DCL WRITE_ALINE ENTRY(2);
DCL CLOSE_DCB ENTRY;
DCL EXIT_ALL ENTRY;

DCL BUFFER_ CHAR(120) STATIC;
DCL BUFSIZ_ SBIN WORD STATIC INIT(SIZEC(BUFFER_));

DCL M$MINE DCB;

     CALL OPEN_DCB(DCBNUM(M$MINE));

     CALL READ_ALINE(BUFFER_,BUFSIZ_);

     CALL WRITE_ALINE(BUFFER_, BUFSIZ_);

     CALL CLOSE_DCB;

     CALL EXIT_ALL;
```

```
38 -
39 - END MYPROG;
40 -
```

LAB 3

ANLZ, STATS, CONTROL

- Look at some dumps
- Look at running system
- Get a feel for problem causes
- If time allows, performance work

SOME USEFUL SEGIDS (AND OFFSETS)

```
$LS16 FOR x   - Auto for Mon. for user
$LS33         - Mon auto
$LS37->.500 FOR x - bottom SS frame of User SS
                                  (.600 if CP running)
$LS35->.560   - bottom SS frame of Mon SS
$LS4 using x  - Auto of user
                        (x{ICP,IDB,ASL} for other domains)
$LS30 for x   - User page table
$LS31         - Mon page table
$LS1 usi x    - JIT
$LS2 usi/for x - file buffers
$LS3 usi/for x - ROS
$LS83         - TIGR data
```

SOME USEFUL COMMANDS

REC – Recovery buffer
SCH – Scheduler Queues
USRT – User Tables
CPUS – List of CPUs
EVB – Event Buffer (Sched Ev & CPU Connect)
JIT, DCB, TCB (domain), AUTO
SSF – Safe Store frame
SYM – use RU for symbols.
:, GOD, ALIB RET – Use Delta, go to it and return
WHO HAS x OPEN
IOCACHE

SOME DUMPA WE WILL LOOK AT

- A100 - UDE-501
- A084 - HFC-533
- A112 - CPC-700

SOME DUMPS YOU WILL LOOK AT

- A071 - UDE-501
- e013 - UDE-501
- e015 - UDE-501
- A185 - QXA-751
- i024 - HFC-530

POSSIBLE OTHER EXERCISES

- Find out what a user is doing
- Attack an inefficient program
- Talk about system support section

LAB 4

ASSIGNMENT

17:51 AUG 29 '85 INT_CG_ASGN

Task: Write a simple version of OUTSYM.

You have the following options:

1) Do both programs. Work on #1 first, in case you run out of time.

2) Do only the AU program. Use the provided program to generate file input to your comgroup.

3) Do only the terminal IO portion of the AU program by adding code to the provided partial solution.

PROGRAM #1 - The AU.

The AU must be able to perform the following tasks.

1) Accept DCB connects.

2) Once a DCB user connects, open a file called PRINT:sname where sname is the station name of the DCB user. Send all records recieved from this station to that file. Since you don't know how many DCB stations will connect at once, you'd better use M$GETDCB. You'd also better add something to PRINT:sname to avoid collisions if the same user sends you more than one file.

3) When the DCB station closes, close the matching file. You'd better stick the file name in a table, however, you'll need it later. Make sure you write all the records from this DCB to the PRINT: file before you close it.

4) Accept terminal connects.

5) If a terminal is connected, and a PRINT:sname file is available, then open the PRINT: file and send the contents to the terminal station. Delete the print file when done.

6) Think up a way to get the terminal stations logged off on request.

Want some bonus points? Do something the solution doesn't! Add a simulated operator's console using a second comgroup. Add the following capability to the AU.

7) Have it open a second comgroup. This comgroup should allow only terminal connects.

8) Printing of files thru the first comgroup will be controlled via commands recieved from a terminal connected to the second comgroup. Implement the following commands:

WHAT - When received from a terminal device connected to to the command comgroup causes a list of PRINT: files waiting to be printed.

PRINT fid - Will print fid if a print device is connected to the first comgroup, else prints 'No Way' on the command terminal.

OFF - Does about what you'd expect.

Use no-wait IO to accomplish this task. An ASYNC routine is provided for you if you don't want to develop your own. See INT_CGLAB_ASYNC. Get steps 1-3 running before you go on to 4-6. If you choose not to do steps 1-3, the partial solution is in INT_CGLAB_PART_SI1.

PROGRAM #2 - The COPY program.

1) Open to the comgroup. Use your sysid as your station name.

2) Write LO type output to the comgroup.

3) Close the comgroup.

If you chose not to do this program, the solution is in INT_CGLAB_SI2. The run unit is INT_CGLAB_2.

LIST of available programs in LJSHOST:

INT_CGEX_SI1	
INT_CGEX_SI2	example programs
INT_CGLAB_SI1	program #1
INT_CGLAB_SI2	program #2
INT_CGLAB_ASYNC	async routine for no-wait IO
INT_CGLAB_PART_SI1	partial solution of program #2

INTERNALS COMGROUP LAB

Purpose: to learn to manipulate CP6 comgroups

Your group will be provided with sample programs which illustrate some comgroup programming techniques. These programs involve a comgroup administrative user that reads messages from a terminal station and sends the messages to a DCB station, where the messages are written back to the terminal station and displayed.

The example source files are INT_CGEX_SI1 and INT_CGEX_SI2 in account LJSHOST @L66B.

Some comgroup X account tools that you might find useful are:
CCG
LARK
CGU

It is necessary to create device logons for comgroup terminal stations. The following is an example of how to create a device logon:

```
!SUPER
CMD*CRE DEV LJS1HOST
OPT*USE=CG
OPT*PROFILE=VIP7205
OPT*COMGROUP=CG/INTERNALS_CG.LJSHOST
OPT*NOCG=INFORM
OPT*PASS=CG
OPT*END
CMD*END
```

17:22 AUG 29 '85 INT_CGEX_CRU

```
!JOB NAME=INT_CGEX_CRU,WSN=UPSTAIRS
!RESOURCE TIME=3,MEM=256
!PL6 INT_CGEX_SI1 OVER *INT_CGEX_OU1,INT_CGEX_LS1
!LINK *INT_CGEX_OU1 OVER INT_CGEX_1
!PL6 INT_CGEX_SI2 OVER *INT_CGEX_OU2,INT_CGEX_LS2
!LINK *INT_CGEX_OU2 OVER INT_CGEX_2
!C INT_CGEX_LS1 TO LP
!C INT_CGEX_LS2 TO LP
```

```
/*M* INT_CGEX_SI1 internals class comgroup example - SI1 of 2 */
ADMINISTRATIVE_USER : PROC MAIN ;

/*
   NAME:           ADMINISTRATIVE_USER

   PURPOSE:        Provides a comgroup AU that reads messages from a terminal station
                   and writes those messages to a DCB station who sends the messages
                   back to the terminal station as 'ECHO: message'.

   DESCRIPTION:    This AU is run from online.  It opens (creates) a comgroup and
                   modifies it to fit this application via M$CGCTL.  It then waits
                   for the DCB station and the terminal station to connect to the
                   comgroup.  When both stations have connected, the AU writes a
                   "logged on" notification message to the terminal station and his
                   own terminal and also sends a message to the DCB station informing
                   him of the STATION name of the terminal station.  The AU then reads
                   messages from the terminal station and writes them to the DCB
                   station.  If the message 'OFF' is received from the terminal
                   station, this is sent to the DCB station (not echoed).  The DCB
                   station recognizes this as a signal to close its station and exit.
                   The AU then sends a "logged off" message to the terminal station
                   and the AU's terminal, disconnects the terminal station, and exits.
                   If break is typed at the AU's terminal, the AU tells the DCB
                   station to close and exit by sending the message 'AU says goodbye'.
                   The AU then sends the "logged off" message to the terminal station
                   and disconnects the terminal station.  The AU then exits.
*/


%INCLUDE     CP_6 ;
%INCLUDE     CP_6_SUBS ;

%B$CGAURD ;

%EQU_CG ;

```

```
%EQU           AU_MESSAGE        = '*AUEV    ' ;

DCL            DATE               CHAR ( 8 ) STATIC ;
DCL         1  FLAGS              STATIC SYMDEF,
               2  DCB_CON         BIT ( 1 ) UNAL INIT ( %NO# ),
               2  TERM_CON        BIT ( 1 ) UNAL INIT ( %NO# ),
               2  AU_BREAK        BIT ( 1 ) UNAL INIT ( %NO# ),
               2  TERM_LOGOFF     BIT ( 1 ) UNAL INIT ( %NO# ) ;
DCL            IO_BUFFER          CHAR ( 200 ) STATIC SYMDEF ;
DCL            IO_BUFFER$         PTR STATIC INIT ( ADDR ( IO_BUFFER ) ) ;
DCL            M$CG               DCB ;
DCL            M$CG$              PTR STATIC SYMDEF ;
DCL            M$DO               DCB ;
DCL            M$LO               DCB ;
DCL            MSGTYP             CHAR ( 8 ) ;
DCL            TIME               CHAR ( 11 ) STATIC ;
DCL            INTERRUPT_HANDLER ENTRY ASYNC ;

%F$DCB ;

%FPT_ACTIVATE
             ( FPTN               = FPT_ACTIVATE_CG,
               STCLASS            = STATIC SYMDEF,
               DCB                = M$CG,
               DISCONNECT         = YES,
               STATION            = VLP_STATION_TERM ) ;

%FPT_CGCTL
             ( FPTN               = FPTCGCTL_CG,
               STCLASS            = CONSTANT,
               DCB                = M$CG,
               CGCP               = VLPCGCP ) ;

%FPT_CLOSE
             ( FPTN               = FPT_CLOSE_CG,
               STCLASS            = STATIC SYMDEF,
               DCB                = M$CG ) ;

%FPT_INT
```

```
            ( FPTN          = FPT_INT,
              UENTRY        = INTERRUPT_HANDLER ) ;

%FPT_OPEN
            ( FPTN          = FPTOPEN_CG,
              STCLASS       = CONSTANT,
              ACCT          = VLPACCT,
              ASN           = COMGROUP,
              AU            = YES,
              CTG           = YES,
              DCB           = M$CG,
              EXIST         = OLDFILE,
              FUN           = CREATE,
              NAME          = VLPNAME,
              QISS          = YES,
              SCRUB         = YES,
              SETSTA        = VLPSETSTA_AU ) ;

%FPT_READ
            ( FPTN          = FPT_READ_CG,
              BUF           = IO_BUFFER,
              DCB           = M$CG,
              STATION       = VLP_STATION ) ;

%FPT_TIME
            ( FPTN          = FPT_TIME,
              DATE          = DATE,
              DEST          = LOCAL,
              TIME          = TIME ) ;

%FPT_WRITE
            ( FPTN          = FPT_WRITE_CG,
              STCLASS       = STATIC SYMDEF,
              BUF           = IO_BUFFER,
              DCB           = M$CG ) ;

%FPT_WRITE
            ( FPTN          = FPT_WRITE_LO,
              BUF           = IO_BUFFER,
            DCB            = M$LO ) ;

%VLP_CGCP
          ( FPTN           = VLPCGCP,
            STCLASS        = CONSTANT,
            RAS            = YES,
            TERMCONAU      = NO,
            TERMCONNAU     = NO,
            TRMRDSIZ       = 74 ) ;

%VLP_NAME
          ( FPTN           = VLPNAME,
            STCLASS        = CONSTANT,
            NAME           = 'INTERNALS_CG' ) ;

%VLP_ACCT
          ( FPTN           = VLPACCT,
            STCLASS        = CONSTANT
/*
           ,ACCT           = 'LJSHOST'
*/
                                       ) ;

%VLP_SETSTA
          ( FPTN           = VLPSETSTA_AU,
            STCLASS        = CONSTANT,
            MYSTATION      = 'AU' ) ;

%VLP_STATION
          ( FPTN           = VLP$STATION,
            STCLASS        = BASED ) ;

%VLP_STATION
          ( FPTN           = VLP_STATION ) ;

%VLP_STATION
          ( FPTN           = VLP_STATION_TERM,
            MSGTYP         = 'LOG_MSG' ) ;

```

```
%VLP_STATION
            ( FPTN              = VLP_STATION_DCB,
              STATION           = 'DCB',
              MSGTYP            = 'TRM_ECHO' ) ;

%EJECT ;
M$CG$ = DCBADDR ( DCBNUM ( M$CG ) ) ;
FPT_WRITE_CG.STATION_ = VECTOR ( VLP_STATION_DCB ) ;

/*
    Create a new comgroup and modify it's control parameters.
*/

CALL M$OPEN ( FPTOPEN_CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
CALL M$CGCTL ( FPTCGCTL_CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;

/*
    Set break control.
*/

CALL M$INT ( FPT_INT ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;

/*
    Loop until the TERM station and DCB station have both connected or
    break has been typed at the AU's terminal.
*/

DO UNTIL ( FLAGS.TERM_CON AND FLAGS.DCB_CON ) OR FLAGS.AU_BREAK ;

    /*
        Issue a read for a message directed to the AU.  This should catch only
        *AUEV messages.
    */

    VLP_STATION.CTL.DIRONLY# = %YES# ;
    CALL M$READ ( FPT_READ_CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;

    IF ( M$CG$->F$DCB.LASTSTA$->VLP$STATION.MSGTYP# = %AU_MESSAGE ) THEN DO ;
        DO CASE IO_BUFFER$->B$CGAURD.EVENT ;
```

```
         CASE ( %CG_TCON# ) ;
            VLP_STATION_TERM.STATION# = IO_BUFFER$->B$CGAURD.STATION ;
            FLAGS.TERM_CON = %YES# ;
         CASE ( %CG_DOPN# ) ;
            IF ( VLP_STATION_DCB.STATION# = IO_BUFFER$->B$CGAURD.STATION ) THEN DO ;
               FLAGS.DCB_CON = %YES# ;
            END ; ELSE DO ;
               /*
                  Let's just ignore this station - it's not the one we expected.
               */
            END ;
         CASE ( %CG_TDSC# ) ;
            IF ( VLP_STATION_TERM.STATION# = IO_BUFFER$->B$CGAURD.STATION ) THEN DO;
               FLAGS.TERM_CON = %NO# ;
            END ;
         CASE ( %CG_DCLS# ) ;
            IF ( VLP_STATION_DCB.STATION# = IO_BUFFER$->B$CGAURD.STATION ) THEN DO ;
               FLAGS.DCB_CON = %NO# ;
            END ;
         CASE ( ELSE ) ;
            /*
               Ignore the other possible AU events.
            */
      END ;
   END ; ELSE DO ;
      /*
         This is an unexpected message - just throw it away.
      */
   END ;

END ;

IF FLAGS.AU_BREAK THEN DO ;
   GOTO CLOSE_AND_EXIT ;
END ;

/*
   Activate the terminal station.
*/
```

```

CALL M$ACTIVATE ( FPT_ACTIVATE_CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;

/*
   Write a logon message to the terminal station and M$LO.  Also send the
    terminal ID to the DCB station.
*/

FPT_TIME.DATE_ = VECTOR ( DATE ) ;
FPT_TIME.TIME_ = VECTOR ( TIME ) ;
CALL M$TIME ( FPT_TIME ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
CALL CONCAT ( IO_BUFFER , VLP_STATION_TERM.STATION#, ' on at ', TIME, ' ', DATE ) ;
CALL M$WRITE ( FPT_WRITE_LO ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
FPT_WRITE_CG.STATION_ = VECTOR ( VLP_STATION_TERM ) ;
CALL M$WRITE ( FPT_WRITE_CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
IO_BUFFER = VLP_STATION_TERM.STATION# ;
FPT_WRITE_CG.STATION_ = VECTOR ( VLP_STATION_DCB ) ;
CALL M$WRITE ( FPT_WRITE_CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;

VLP_STATION.CTL.DIRONLY# = %NO# ;

/*
   The AU is now set up to read all messages, directed or not.  The AU will send
   messages read from the terminal station to the DCB station until "OFF" is typed
   at the terminal station or break is entered at the AU's terminal.
*/

DO UNTIL FLAGS.TERM_LOGOFF OR FLAGS.AU_BREAK ;

   IO_BUFFER = ' ' ;
   CALL M$READ ( FPT_READ_CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;

   MSGTYP = M$CG$->F$DCB.LASTSTA$->VLP$STATION.MSGTYP# ;
   IF ( MSGTYP ~= %AU_MESSAGE ) THEN DO ;
      IF ( IO_BUFFER = 'OFF' ) OR ( IO_BUFFER = 'off' ) THEN DO ;
         FLAGS.TERM_LOGOFF = %YES# ;
      END ;
      CALL M$WRITE ( FPT_WRITE_CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
   END ; ELSE DO ;
```

```
      DO CASE IO_BUFFER$->B$CGAURD.EVENT ;
         CASE ( %CG_DCLS# ) ;
            IF ( VLP_STATION_DCB.STATION# = IO_BUFFER$->B$CGAURD.STATION ) AND
                  NOT FLAGS.AU_BREAK THEN DO ;
               IO_BUFFER = 'DCB station closed unexpectedly' ;
               CALL M$WRITE ( FPT_WRITE_LO ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
               FLAGS.TERM_LOGOFF = %YES# ;   /* let's fake the terminal logoff */
            END ;
         CASE ( %CG_TDSC# ) ;
            IF ( VLP_STATION_TERM.STATION# = IO_BUFFER$->B$CGAURD.STATION ) THEN DO ;
               IO_BUFFER = 'TERM station disconnected unexpectedly' ;
               CALL M$WRITE ( FPT_WRITE_LO ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
               IO_BUFFER = 'AU says goodbye' ;
               CALL M$WRITE ( FPT_WRITE_CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
               GOTO CLOSE_AND_EXIT ;
            END ;
         CASE ( ELSE ) ;
            /*
               Ignore the other possible AU events.
            */
      END ;
   END ;

END ;

/*
   Send the logoff message to the terminal station and to M$LO.
*/

CALL M$TIME ( FPT_TIME ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
CALL CONCAT ( IO_BUFFER, VLP_STATION_TERM.STATION#, ' off at ', TIME, ' ', DATE ) ;
CALL M$WRITE ( FPT_WRITE_LO ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
FPT_WRITE_CG.STATION_ = VECTOR ( VLP_STATION_TERM ) ;
CALL M$WRITE ( FPT_WRITE_CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;

/*
   Disconnect the terminal and close the comgroup.
*/

```

```
CALL M$DEACTIVATE ( FPT_ACTIVATE_CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;

CLOSE_AND_EXIT: ;
   CALL M$CLOSE ( FPT_CLOSE_CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
   CALL M$EXIT ;

REPORT_ERROR_AND_EXIT: ;
   IF M$CG$->F$DCB.FCD# THEN DO ;
      CALL M$CLOSE ( FPT_CLOSE_CG ) ;
   END ;
   CALL M$MERC ;
   CALL M$EXIT ;

END ADMINISTRATIVE_USER ;

%EOD ;
INTERRUPT_HANDLER : PROC ASYNC ;

/*
   NAME:             INTERRUPT_HANDLER
   PURPOSE:          Takes care of interrupts caused by typing break at the AU's
                     terminal.
   DESCRIPTION:      If the DCB station is connected to the comgroup, the 'AU says
                     goodbye' message is sent and the AU_BREAK flag is sent.  The AU
                     will then finish processing this break when the DCB station's
                     close is announced in an AU event message.  If the DCB station is
                     not present, the terminal station is disconnected and the AU
                     exits.  This is done because the AU is hung in a wait read that
                     won't complete until the DCB station connects and the break must
                     be taken care of immediately.
*/


%INCLUDE CP_6 ;
%INCLUDE CP_6_SUBS ;

DCL      1  FLAGS               SYMREF,
            2  DCB_CON          BIT ( 1 ) UNAL,
            2  TERM_CON         BIT ( 1 ) UNAL,
            2  AU_BREAK      BIT ( 1 ) UNAL,
            2  TERM_LOGOFF   BIT ( 1 ) UNAL ;
DCL      IO_BUFFER           CHAR ( 200 ) SYMREF ;
DCL     M$CG                 DCB ;
DCL     M$CG$                PTR SYMREF ;

%F$DCB ;

%FPT_ACTIVATE
          ( FPTN             = FPT_ACTIVATE_CG,
            STCLASS          = SYMREF ) ;

%FPT_CLOSE
          ( FPTN             = FPT_CLOSE_CG,
            STCLASS          = SYMREF ) ;

%FPT_WRITE
          ( FPTN             = FPT_WRITE_CG,
            STCLASS          = SYMREF ) ;

%EJECT ;
IF FLAGS.DCB_CON THEN DO ;
   IO_BUFFER = 'AU says goodbye' ;
   CALL M$WRITE ( FPT_WRITE_CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
   FLAGS.AU_BREAK = %YES# ;
   RETURN ;
END ; ELSE DO ;
   IF FLAGS.TERM_CON THEN DO ;
      CALL M$DEACTIVATE ( FPT_ACTIVATE_CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
   END ;
   CALL M$CLOSE ( FPT_CLOSE_CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
   CALL M$EXIT ;
END ;

REPORT_ERROR_AND_EXIT: ;
   CALL M$CLOSE ( FPT_CLOSE_CG ) ;
   CALL M$MERC ;
   CALL M$EXIT ;
END INTERRUPT_HANDLER ;
```

```
/*M* INT_CGEX_SI2 internals class comgroup example - SI2 of 2 */
DCB_ECHO : PROC MAIN ;

/*
   NAME:          DCB_ECHO
   PURPOSE:       Echoes the messages sent from the AU back to the terminal station.
   DESCRIPTION:   The DCB station receives the name of the terminal station in the
                  first message from the AU.  It then sends the messages received
                  from the AU to the terminal station in the form 'ECHO: message'.
                  If the message 'OFF' or 'AU says goodbye' is received,
                  it simply exits.
*/


%INCLUDE    CP_6 ;
%INCLUDE    CP_6_SUBS ;

DCL      1  FLAGS          STATIC,
              2  TERM_LOGOFF BIT ( 1 ) UNAL INIT ( %NO# ),
              2  NO_AU       BIT ( 1 ) UNAL INIT ( %NO# ) ;
DCL         IN_BUFFER      CHAR ( 200 ) STATIC;
DCL         OUT_BUFFER     CHAR ( 80 ) ;
DCL         M$CG           DCB ;
DCL         M$CG$          PTR ;

%F$DCB ;

%FPT_CLOSE
           ( FPTN          = FPTCLOSE_CG,
             STCLASS       = CONSTANT,
             DCB           = M$CG ) ;

%FPT_OPEN
           ( FPTN          = FPTOPEN_CG,
             STCLASS       = CONSTANT,
             ACCT          = VLPACCT,
             ASN           = COMGROUP,
            DCB            = M$CG,
            FUN            = UPDATE,
            NAME           = VLPNAME,
            SCRUB          = YES,
            SETSTA         = VLPSETSTA_DCB ) ;

%FPT_READ
         ( FPTN            = FPTREAD_CG,
            STCLASS        = CONSTANT,
            BUF            = IN_BUFFER,
            DCB            = M$CG,
            STATION        = VLPSTATION_AU ) ;

%FPT_WRITE
         ( FPTN            = FPT_WRITE_CG,
            DCB            = M$CG,
            STATION        = VLP_STATION_TERM ) ;

%VLP_ACCT
         ( FPTN            = VLPACCT,
            STCLASS        = CONSTANT
/*
           ,ACCT           = 'LJSHOST'
*/
                                       ) ;

%VLP_NAME
         ( FPTN            = VLPNAME,
            STCLASS        = CONSTANT,
            NAME           = 'INTERNALS_CG' ) ;

%VLP_SETSTA
         ( FPTN            = VLPSETSTA_DCB,
            STCLASS        = CONSTANT,
            MYSTATION      = 'DCB' ) ;

%VLP_STATION
         ( FPTN            = VLPSTATION_AU,
            STCLASS        = CONSTANT,
            DIRONLY        = YES,
            MSGTYP         = 'TRM_ECHO',
            STATION        = 'AU' ) ;

%VLP_STATION
          ( FPTN           = VLP_STATION_TERM,
            MSGTYP         = 'TRM_ECHO' ) ;

%EJECT ;
M$CG$ = DCBADDR ( DCBNUM ( M$CG ) ) ;

/*
   Open the comgroup.
*/

CALL M$OPEN ( FPTOPEN_CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;

/*
   Read the name of the terminal station.
*/

CALL M$READ ( FPTREAD_CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
IF IN_BUFFER = 'AU says goodbye' THEN DO ;
    FLAGS.NO_AU = %YES# ;
END ; ELSE DO ;
   VLP_STATION_TERM.STATION# = IN_BUFFER ;
END ;

/*
   Read the messages from the AU and forward them to the terminal station
   until the AU goes away or the terminal logs off.
*/

DO WHILE NOT FLAGS.TERM_LOGOFF AND NOT FLAGS.NO_AU ;

   CALL M$READ ( FPTREAD_CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
   IF ( IN_BUFFER = 'OFF' ) OR ( IN_BUFFER = 'off' ) THEN DO ;
      FLAGS.TERM_LOGOFF = %YES# ;
   END ; ELSE IF ( IN_BUFFER = 'AU says goodbye' ) THEN DO ;
```

```
        FLAGS.NO_AU = %YES# ;
    END ; ELSE DO ;
        CALL CONCAT ( OUT_BUFFER, 'ECHO: ',IN_BUFFER ) ;
        FPT_WRITE_CG.BUF_ = VECTOR ( OUT_BUFFER ) ;
        FPT_WRITE_CG.BUF_.BOUND = M$CG$->F$DCB.ARS# - 1 ;
        CALL M$WRITE ( FPT_WRITE_CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
    END ;

END ;

CALL M$CLOSE ( FPTCLOSE_CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
CALL M$EXIT ;

REPORT_ERROR_AND_EXIT: ;
    IF M$CG$->F$DCB.FCD# THEN DO ;
        CALL M$CLOSE ( FPTCLOSE_CG ) ;
    END ;
    CALL M$MERC ;
    CALL M$EXIT ;

END DCB_ECHO ;
```

LAB 4

ANSWER

```
EVENT_HANDLER : PROC ASYNC ;

/*
    NAME:                EVENT_HANDLER
    PURPOSE:             Takes care of events caused by IO completion.
    DESCRIPTION:         Sets the appropriate flags and returns.
*/


%INCLUDE CP_6 ;
%INCLUDE CP_6_SUBS ;

%B$TCB ;

%B$NWIO
              ( STCLASS                  = "BASED ( B$NWIO$ )" ) ;

%EQU            PRINT_READ_EVENT  = 1001 ;
%EQU            CONSOLE_READ_EVENT= 1002 ;

DCL             B$NWIO$                  PTR ;
DCL             B$TCB$                   PTR SYMREF ;
DCL         1   EVENTS                   SYMREF,
                2  PRINT_COMP            BIT ( 1 ) UNAL,
                2  CONSOLE_COMP          BIT ( 1 ) UNAL,
                2  BAD_COMP              BIT ( 1 ) UNAL,
                2  ERR                   BIT ( 36 ),
                2  TYC                   BIT ( 36 ) ;

%EJECT ;
DO INHIBIT ;
    B$NWIO$ = B$TCB$->B$TCB.STK$ ;
    DO CASE ( B$NWIO.EVID ) ;
        CASE ( %PRINT_READ_EVENT ) ;
            EVENTS.PRINT_COMP = %YES# ;
        CASE ( %CONSOLE_READ_EVENT ) ;   /* not currently implemented */
            EVENTS.CONSOLE_COMP = %YES# ;
        CASE ( ELSE ) ;
            EVENTS.BAD_COMP = %YES# ;
    END ;
    EVENTS.ERR = B$NWIO.ERR ;
    EVENTS.TYC = B$NWIO.TYC ;
END ;

RETURN ;

END EVENT_HANDLER ;
```

```
17:34 AUG 29 '85 INT_CGLAB_CRU..

!JOB NAME=INT_CGLAB_CRU,WSN=UPSTAIRS
!RESOURCE TIME=3,MEM=256
!PL6 INT_CGLAB_SI1 OVER *INT_CGLAB_OU1,INT_CGLAB_LS1
!LINK *INT_CGLAB_OU1 OVER INT_CGLAB_1
!PL6 INT_CGLAB_SI2 OVER *INT_CGLAB_OU2,INT_CGLAB_LS2
!LINK *INT_CGLAB_OU2 OVER INT_CGLAB_2
!C INT_CGLAB_LS1 TO LP
!C INT_CGLAB_LS2 TO LP
```

```
17:34 AUG 29 '85 INT_CGLAB_PART_CRU

!JOB NAME=INT_CGLAB_CRU,WSN=UPSTAIRS
!RESOURCE TIME=3,MEM=256
!PL6 INT_CGLAB_PART_SI1 OVER *INT_CGLAB_PART_OU1,INT_CGLAB_PART_LS1
!LINK *INT_CGLAB_PART_OU1 OVER INT_CGLAB_PART_1
!C INT_CGLAB_PART_LS1 TO LP
```

```
/*M* INT_CGLAB_PART_SI1 internals class comgroup lab - SI1 of 2 */
OUTSYM_AU : PROC MAIN ;

/*
   NAME:           OUTSYM_AU

   PURPOSE:          This is the administrative user for the partial version of OUTSYM.
                     Files are copied into the comgroup by DCB stations identified
                     by SYSID.  These files are copied to PRINT:sysid_ext files.

   DESCRIPTION:      This AU is run from online.  It opens (creates) a comgroup and
                     modifies it to fit this application via M$CGCTL.  It recognizes
                     DCB connections as a signal to open a new PRINT file.  It then
                     takes all records provided by that station and writes them to
                     the PRINT file.  It recognizes a DCB close as a signal that all
                     records have been written into the comgroup (but possibly not
                     processed yet).  Because of this, a second DCB station exists
                     (a non-AU station) which does wait reads to process the
                     messages remaining in the comgroup written by the closing DCB.
                     NOTE: The comgroup AU cannot do this because if *AUEV
                     messages exist in the comgroup (having highest message prio)
                     an AU read specifying a message of some other message type will
                     be satisfied by this *AUEV message - welcome to the wonderful
                     world of comgroups!

*/


%INCLUDE       CP_6 ;
%INCLUDE       CP_6_SUBS ;
%INCLUDE       B_ERRORS_C ;
%INCLUDE       XU_MACRO_C ;
%INCLUDE       XUG_ENTRY ;

%EQU           AU_MESSAGE          = '*AUEV    ' ;
%EQU           PRINT_READ_EVENT    = 1001 ;
%EQU           CONSOLE_READ_EVENT= 1002 ;
```

```

DCL         B$TCB$            PTR SYMREF ;
DCL         DATE              CHAR ( 8 ) STATIC ;
DCL         DCB_NUM           UBIN STATIC ;
DCL       1 EVENTS            STATIC SYMDEF,
            2  PRINT_COMP     BIT ( 1 ) UNAL INIT ( %NO# ),
            2  CONSOLE_COMP   BIT ( 1 ) UNAL INIT ( %NO# ),
            2  BAD_COMP       BIT ( 1 ) UNAL INIT ( %NO# ),
            2  ERR            BIT ( 36 ),
            2  TYC            BIT ( 36 ) ;
DCL         EXTENSION         UBIN STATIC INIT ( 0 ) ;
DCL         I                 UBIN ;
DCL         IDX               SBIN ;
DCL         IO_BUFFER         CHAR ( 200 ) STATIC ;
DCL         IO_BUFFER$        PTR STATIC INIT ( ADDR ( IO_BUFFER ) ) ;
DCL         M$CG              DCB ;
DCL         M$CG$             PTR ;
DCL         M$SPECIAL         DCB ;
DCL         M$SPECIAL$        PTR ;
DCL       1 PRINT_FILES       ( 0:99 ) STATIC,
            2  NAME_EXT       CHAR ( 12 ) INIT ( ' '*100 ),
            2  NAME           REDEF NAME_EXT,
               3  STATION     CHAR ( 8 ),
               3  EXTENSION   CHAR ( 4 ),
            2  DCB_NUM        UBIN INIT ( 0*100 ) ;
DCL         PRINTNUM          UBIN STATIC INIT ( 0 ) ;
DCL         TIME              CHAR ( 11 ) STATIC ;
DCL         EVENT_HANDLER     ENTRY ASYNC ;
DCL         INTERRUPT_HANDLER ENTRY ASYNC ;

%B$CGAURD
          ( FPTN             = B$CGAURD,
            STCLASS          = "BASED ( IO_BUFFER$ )" ) ;

%B$ALT ;

%B$TCB ;

%EQU_CG ;
```

```

%F$DCB ;

%FPT_ACTIVATE
          ( FPTN           = FPT_ACTIVATE_CG,
            STCLASS        = STATIC,
            DCB            = M$CG,
            DISCONNECT     = YES ) ;

%FPT_CGCTL
          ( FPTN           = FPT_CGCTL_CG,
            DCB            = M$CG,
            CGCP           = VLP_CGCP_SPECIAL ) ;

%FPT_CLOSE
          ( FPTN           = FPT_CLOSE_CG,
            STCLASS        = STATIC,
            DCB            = M$CG ) ;

%FPT_CLOSE
          ( FPTN           = FPT_CLOSE_INPUT,
            DISP           = SAVE ) ;

%FPT_EVENT
          ( FPTN           = FPT_EVENT,
            UENTRY         = EVENT_HANDLER ) ;

%FPT_GETDCB
          ( FPTN           = FPT_GETDCB,
            DCBNAME        = VLP_NAME_DCB,
            DCBNUM         = DCB_NUM ) ;

%FPT_INT
          ( FPTN           = FPT_INT,
            UENTRY         = INTERRUPT_HANDLER ) ;

%FPT_OPEN
          ( FPTN           = FPTOPEN_CG,
            STCLASS        = CONSTANT,
            ACCT           = VLPACCT,
            ASN            = COMGROUP,
            AU             = YES,
            CTG            = YES,
            DCB            = M$CG,
            EXIST          = OLDFILE,
            FUN            = CREATE,
            NAME           = VLPNAME,
            QISS           = YES,
            SCRUB          = YES,
            SETSTA         = VLPSETSTA_AU ) ;

%FPT_OPEN
          ( FPTN           = FPTOPEN_SPECIAL,
            STCLASS        = CONSTANT,
            ACCT           = VLPACCT,
            ASN            = COMGROUP,
            DCB            = M$SPECIAL,
            FUN            = UPDATE,
            NAME           = VLPNAME,
            SCRUB          = YES,
            SETSTA         = VLPSETSTA_SPECIAL ) ;

%FPT_OPEN
          ( FPTN           = FPT_OPEN_INPUT,
            ASN            = FILE,
            EXIST          = NEWFILE,
            FUN            = CREATE,
            NAME           = VLP_NAME_PRINT,
            ORG            = CONSEC,
            SCRUB          = YES ) ;

%FPT_READ
          ( FPTN           = FPT_READ_CG,
            BUF            = IO_BUFFER,
            DCB            = M$CG,
            EVENT          = %PRINT_READ_EVENT,
            STATION        = VLP_STATION,
            WAIT           = NO ) ;
```

```

%FPT_READ
           ( FPTN           = FPT_READ_SPECIAL,
             BUF            = IO_BUFFER,
             DCB            = M$SPECIAL,
             STATION        = VLP_STATION_SPECIAL ) ;

%FPT_RELDCB
           ( FPTN           = FPT_RELDCB ) ;

%FPT_WAIT
           ( FPTN           = FPTWAIT_ASECOND,
             STCLASS        = CONSTANT,
             UNITS          = 1 ) ;

%FPT_WRITE
           ( FPTN           = FPT_WRITE_INPUT,
             BUF            = IO_BUFFER ) ;

%VLP_CGCP
           ( FPTN           = VLP_CGCP_OUTSYM,
             CONMSG         = YES,
             DCBCONAU       = NO,
             DCBCONWA       = YES,
             MINPG          = 4,
             RAS            = YES ) ;

%VLP_CGCP
           ( FPTN           = VLP_CGCP_SPECIAL,
             CONMSG         = NO ) ;

%VLP_NAME
           ( FPTN           = VLP_NAME_DCB,
             LEN            = 31 ) ;

%VLP_NAME
           ( FPTN           = VLP_NAME_PRINT,
             LEN            = 31 ) ;

```

```
%VLP_NAME
            ( FPTN              = VLPNAME,
              STCLASS           = CONSTANT,
              NAME              = 'INTERNALS_CG' ) ;

%VLP_ACCT
            ( FPTN              = VLPACCT,
              STCLASS           = CONSTANT
/*
             ,ACCT              = 'LJSHOST'
*/
                                              ) ;

%VLP_SETSTA
            ( FPTN              = VLPSETSTA_AU,
              STCLASS           = CONSTANT,
              MYSTATION         = 'AU' ) ;

%VLP_SETSTA
            ( FPTN              = VLPSETSTA_SPECIAL,
              STCLASS           = CONSTANT,
              MYSTATION         = 'SPECIAL' ) ;

%VLP_STATION
            ( FPTN              = VLP$STATION,
              STCLASS           = BASED ) ;

%VLP_STATION
            ( FPTN              = VLP_STATION ) ;

%VLP_STATION
            ( FPTN              = VLP_STATION_SPECIAL,
              EOFNONE           = YES,
              MSGTYP            = 'PRINT   ' ) ;

%VLP_STATION
            ( FPTN              = VLP_STATION_DCB ) ;

%XUG_GETCMD
```

```
            ( NAME           = XUGGETCMD,
              STCLASS        = CONSTANT ) ;

%EJECT ;
/*
   Set break control.
*/

CALL M$INT ( FPT_INT ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;

M$CG$ = DCBADDR ( DCBNUM ( M$CG ) ) ;
M$SPECIAL$ = DCBADDR ( DCBNUM ( M$SPECIAL ) ) ;

/*
   Create a new comgroup and modify its control parameters.
   The first CGCTL specifies CONMSG=NO so the SPECIAL station can connect
   without requiring activation.  After that, another CGCTL is done specifying
   the normal parameters that we should use.
*/

CALL M$OPEN ( FPTOPEN_CG ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
CALL M$CGCTL ( FPT_CGCTL_CG ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
CALL M$OPEN ( FPTOPEN_SPECIAL ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
FPT_CGCTL_CG.CGCP_ = VECTOR ( VLP_CGCP_OUTSYM ) ;
CALL M$CGCTL ( FPT_CGCTL_CG ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;

/*
   Set event control.
*/

CALL M$EVENT ( FPT_EVENT ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;

/*
   Start the first no-wait read.
*/

CALL M$READ ( FPT_READ_CG ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;

/*
```

```
    Start normal operation of getting PRINT files and printing them if possible.
*/

DO WHILE %YES# ;

    DO INHIBIT ;
        DO WHILE EVENTS.PRINT_COMP ;
            CALL PROCESS_READ_EVENT ;
            EVENTS.PRINT_COMP = %NO# ;
            IO_BUFFER = ' ' ;
            CALL M$READ ( FPT_READ_CG ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
        END ;
        CALL M$WAIT ( FPTWAIT_ASECOND ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
    END ;

END ;

%EJECT ;
PROCESS_READ_EVENT : PROC ;

    /*
        The message just read could be an AU event or a record to be written into
        one of the print files being created.  Take care of it appropriately.
    */

DCL          EOF                  BIT ( 1 ) ;

    IF ( M$CG$->F$DCB.LASTSTA$->VLP$STATION.MSGTYP# = %AU_MESSAGE ) THEN DO ;
        DO CASE B$CGAURD.EVENT ;

            /*
                This is a new DCB station.  This station needs to be activated so
                that it can start shoveling data into the comgroup.  An entry is
                then built in the print file table containing the station name and
                file extension, along with a non-zero DCB number.  This number
                signifies that this print file is not complete (it can't be printed
                yet).  The DCB number is also necessary when writing records to the
                file.
            */
```

```

         CASE ( %CG_DOPN# ) ;
            VLP_STATION_DCB.STATION# = B$CGAURD.STATION ;
            FPT_ACTIVATE_CG.STATION_ = VECTOR ( VLP_STATION_DCB ) ;
            CALL M$ACTIVATE ( FPT_ACTIVATE_CG ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
            CALL GET_NEW_INDEX ( IDX ) ;
            PRINT_FILES.NAME.STATION ( IDX ) = B$CGAURD.STATION ;
            EXTENSION = EXTENSION + 1 ;
            CALL BINCHAR ( PRINT_FILES.NAME.EXTENSION ( IDX ),
                           EXTENSION ) ;
            CALL M$GETDCB ( FPT_GETDCB ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
            PRINT_FILES.DCB_NUM ( IDX ) = DCB_NUM ;
            CALL CONCAT ( VLP_NAME_PRINT.NAME#,
                          'PRINT:',
                          PRINT_FILES.NAME_EXT ( IDX ) ) ;
            CALL INDEX ( I,
                         ' ',
                         VLP_NAME_PRINT.NAME#,
                         0 ) ;
            VLP_NAME_PRINT.L# = I ;
            FPT_OPEN_INPUT.V.DCB# = DCB_NUM ;
            CALL M$OPEN ( FPT_OPEN_INPUT ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;

         /*
            This event signifies that the DCB station is finished writing data
            into the comgroup and is going away.  The records in the comgroup
            that haven't been processed yet are read by the SPECIAL station
            using WAIT IO and EOFNONE and written to the print file.  The
            print file is then closed and the DCB number in the print file table
            is set to zero to signal that this file is ready to print.

            The reason that the DCB stations must be activated upon connection
            to the comgroup is now apparent - if this closing station immediately
            reconnected and started writing records, we would put the new data
            is this old print file.
         */

         CASE ( %CG_DCLS# ) ;
            CALL GET_STATION_INDEX ( B$CGAURD.STATION,
                              IDX ) ;
         IF ( IDX >= 0 ) THEN DO ;
            VLP_STATION_SPECIAL.STATION# = PRINT_FILES.NAME.STATION ( IDX ) ;
            FPT_WRITE_INPUT.V.DCB# = PRINT_FILES.DCB_NUM ( IDX ) ;
            EOF = %NO# ;
            DO WHILE NOT EOF ;
               CALL M$READ ( FPT_READ_SPECIAL )
                     WHENALTRETURN DO ;
                        IF ( B$TCB$->B$TCB.ALT$->B$ALT.ERR.CODE = %E$EOF ) THEN DO ;
                           EOF = %YES# ;
                        END ; ELSE DO ;
                           CALL ERROR_EXIT ; /* DOESN'T RETURN */
                        END ;
                     END ;
               IF NOT EOF THEN DO ;
                  FPT_WRITE_INPUT.BUF_.BOUND = M$SPECIAL$->F$DCB.ARS# - 1 ;
                  CALL M$WRITE ( FPT_WRITE_INPUT ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
               END ;
            END ;
            FPT_CLOSE_INPUT.V.DCB# = PRINT_FILES.DCB_NUM ( IDX ) ;
            CALL M$CLOSE ( FPT_CLOSE_INPUT ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
            FPT_RELDCB.V.DCB# = PRINT_FILES.DCB_NUM ( IDX ) ;
            CALL M$RELDCB ( FPT_RELDCB ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
            PRINT_FILES.DCB_NUM ( IDX ) = 0 ;
         END ;

      CASE ( ELSE ) ;
         /*
            Ignore the other possible AU events.
         */
   END ;

END ; ELSE DO ;
   /*
      This is a record to be written to a print file.
   */
   CALL GET_STATION_INDEX ( M$CG$->F$DCB.LASTSTA$->VLP$STATION.STATION#,
                           IDX ) ;
   IF ( IDX >= 0 ) THEN DO ;
```

```
         FPT_WRITE_INPUT.V.DCB# = PRINT_FILES.DCB_NUM ( IDX ) ;
         FPT_WRITE_INPUT.BUF_.BOUND = M$CG$->F$DCB.ARS# - 1 ;
         CALL M$WRITE ( FPT_WRITE_INPUT ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
      END ;
   END ;

   RETURN ;

END PROCESS_READ_EVENT ;

%EJECT ;
GET_NEW_INDEX : PROC ( IDX ) ;

   /*
      Get an unused entry in the print file table.  Returns -1 if none
      are available.
   */

DCL         IDX           SBIN ;

DCL         I             UBIN ;


   I = 0 ;
   DO WHILE ( I <= PRINTNUM ) AND ( PRINT_FILES.NAME.STATION ( I ) ~= ' ' ) ;
      I = I + 1 ;
   END ;

   IF ( I = PRINTNUM ) AND ( PRINTNUM < 100 ) THEN DO ;
      PRINTNUM = PRINTNUM + 1 ;
      IDX = I ;
   END ; ELSE IF ( I < PRINTNUM ) THEN DO ;
      IDX = I ;
   END ; ELSE DO ;
      IDX = -1 ;
   END ;

   RETURN ;

```

```
END GET_NEW_INDEX ;

%EJECT ;
GET_PRINTABLE_INDEX : PROC ( IDX ) ;

   /*
      Get the index of a printable file - name not blank and DCB number zeroed.
      -1 is returned if none exist.
   */

DCL          IDX          SBIN ;

DCL          I            UBIN ;


   I = 0 ;
   IDX = -1 ;
   DO WHILE ( I <= PRINTNUM ) AND ( IDX = -1 ) ;
      IF ( PRINT_FILES.NAME.STATION ( I ) ~= ' ' ) AND ( PRINT_FILES.DCB_NUM ( I ) = 0 ) THEN DO ;
         IDX = I ;
      END ;
      I = I + 1 ;
   END ;

   RETURN ;

END GET_PRINTABLE_INDEX ;

%EJECT ;
GET_STATION_INDEX : PROC ( STATION, IDX ) ;

   /*
      Get the index of the given station.  -1 is returned if it doesn't exist.
   */

DCL          STATION      CHAR ( 8 ) ;
DCL          IDX          SBIN ;

DCL          I            UBIN ;
```

```


   I = 0 ;
   IDX = -1 ;
   DO WHILE ( I <= PRINTNUM ) AND ( IDX = -1 ) ;
      IF ( STATION = PRINT_FILES.NAME.STATION ( I ) ) AND ( PRINT_FILES.DCB_NUM ( I ) ~= 0 ) THEN D
         IDX = I ;
      END ;
      I = I + 1 ;
   END ;

   RETURN ;

END GET_STATION_INDEX ;

%EJECT ;
ERROR_EXIT : PROC ;

   CALL XUG$CLOSE_DCBS ( XUGGETCMD ) ;
   CALL M$MERC ;
   CALL M$EXIT ;

END ERROR_EXIT ;



END OUTSYM_AU ;

%EOD ;
INTERRUPT_HANDLER : PROC ASYNC ;

/*
   NAME:             INTERRUPT_HANDLER
   PURPOSE:          Takes care of interrupts caused by typing break at the AU's
                     terminal.
   DESCRIPTION:      Closes all DCB's and exits.
*/


```

```
%INCLUDE CP_6 ;
%INCLUDE CP_6_SUBS ;
%INCLUDE XU_MACRO_C ;
%INCLUDE XUG_ENTRY ;

%XUG_GETCMD
          ( NAME                = XUGGETCMD,
            STCLASS             = CONSTANT ) ;

%EJECT ;
CALL XUG$CLOSE_DCBS ( XUGGETCMD ) ;
CALL M$EXIT ;

END INTERRUPT_HANDLER ;

%EOD ;
EVENT_HANDLER : PROC ASYNC ;

/*
    NAME:            EVENT_HANDLER
    PURPOSE:         Takes care of events caused by IO completion.
    DESCRIPTION:     Sets the appropriate flags and returns.
*/


%INCLUDE CP_6 ;
%INCLUDE CP_6_SUBS ;

%B$TCB ;

%B$NWIO
          ( STCLASS             = "BASED ( B$NWIO$ )" ) ;

%EQU        PRINT_READ_EVENT  = 1001 ;
%EQU        CONSOLE_READ_EVENT= 1002 ;

DCL         B$NWIO$           PTR ;
DCL         B$TCB$            PTR SYMREF ;
DCL       1 EVENTS            SYMREF,
            2  PRINT_COMP     BIT ( 1 ) UNAL,
            2  CONSOLE_COMP   BIT ( 1 ) UNAL,
            2  BAD_COMP       BIT ( 1 ) UNAL,
            2  ERR            BIT ( 36 ),
            2  TYC            BIT ( 36 ) ;

%EJECT ;
DO INHIBIT ;
   B$NWIO$ = B$TCB$->B$TCB.STK$ ;
   DO CASE ( B$NWIO.EVID ) ;
      CASE ( %PRINT_READ_EVENT ) ;
        EVENTS.PRINT_COMP = %YES# ;
      CASE ( %CONSOLE_READ_EVENT ) ;   /* not currently implemented */
        EVENTS.CONSOLE_COMP = %YES# ;
      CASE ( ELSE ) ;
        EVENTS.BAD_COMP = %YES# ;
   END ;
   EVENTS.ERR = B$NWIO.ERR ;
   EVENTS.TYC = B$NWIO.TYC ;
END ;

RETURN ;

END EVENT_HANDLER ;
```

```
/*M* INT_CGLAB_SI1 internals class comgroup lab - SI1 of 2 */
OUTSYM_AU : PROC MAIN ;

/*
   NAME:          OUTSYM_AU

   PURPOSE:        This is the administrative user for a simple version of OUTSYM.
                   Files are copied into the comgroup by DCB stations identified
                   by SYSID.  These files are copied to PRINT:sysid_ext files
                   and are printed at a terminal station if one is connected.

   DESCRIPTION:    This AU is run from online.  It opens (creates) a comgroup and
                   modifies it to fit this application via M$CGCTL.  It recognizes
                   DCB connections as a signal to open a new PRINT file.  It then
                   takes all records provided by that station and writes them to
                   the PRINT file.  It recognizes a DCB close as a signal that all
                   records have been written into the comgroup (but possibly not
                   processed yet).  Because of this, a second DCB station exists
                   (a non-AU station) which does wait reads to process the
                   messages remaining in the comgroup written by the closing DCB.
                   NOTE: The comgroup AU cannot do this because if *AUEV
                   messages exist in the comgroup (having highest message prio)
                   an AU read specifying a message of some other message type will
                   be satisfied by this *AUEV message - welcome to the wonderful
                   world of comgroups!
                   A terminal connect signals that a "printer" is available so
                   PRINT files can be sent to it.  A terminal disconnect signals
                   that we can no longer print files.  If a file is currently
                   being printed, that file is closed and saved to be printed
                   again.  A terminal break signals that the terminal requests
                   to be disconnected.  The print file is also saved in this case,
                   if necessary.

*/


%INCLUDE    CP_6 ;
```

```
%INCLUDE    CP_6_SUBS ;
%INCLUDE    B_ERRORS_C ;
%INCLUDE    XU_MACRO_C ;
%INCLUDE    XUG_ENTRY ;

%EQU        AU_MESSAGE        = '*AUEV    ' ;
%EQU        FORM_FEED         = 12 ;
%EQU        PRINT_READ_EVENT  = 1001 ;
%EQU        CONSOLE_READ_EVENT= 1002 ;

DCL         B$TCB$            PTR SYMREF ;
DCL         CUR_PRINTFILE     SBIN STATIC INIT ( -1 ) ;
DCL         DATE              CHAR ( 8 ) STATIC ;
DCL         DCB_NUM           UBIN STATIC ;
DCL      1  EVENTS            STATIC SYMDEF,
               2  PRINT_COMP     BIT ( 1 ) UNAL INIT ( %NO# ),
               2  CONSOLE_COMP   BIT ( 1 ) UNAL INIT ( %NO# ),
               2  BAD_COMP       BIT ( 1 ) UNAL INIT ( %NO# ),
               2  ERR            BIT ( 36 ),
               2  TYC            BIT ( 36 ) ;
DCL         EXTENSION         UBIN STATIC INIT ( 0 ) ;
DCL         I                 UBIN ;
DCL         IDX               SBIN ;
DCL         IO_BUFFER         CHAR ( 200 ) STATIC ;
DCL         IO_BUFFER$        PTR STATIC INIT ( ADDR ( IO_BUFFER ) ) ;
DCL         M$CG              DCB ;
DCL         M$CG$             PTR ;
DCL         M$SPECIAL           DCB ;
DCL         M$SPECIAL$          PTR ;
DCL         M$PRINT           DCB ;
DCL         M$PRINT$          PTR ;
DCL         PRINT_BUFFER      CHAR ( 200 ) STATIC ;
DCL      1  PRINT_FILES       ( 0:99 ) STATIC,
               2  NAME_EXT       CHAR ( 12 ) INIT ( ' '*100 ),
               2  NAME           REDEF NAME_EXT,
                  3  STATION     CHAR ( 8 ),
                  3  EXTENSION   CHAR ( 4 ),
               2  DCB_NUM        UBIN INIT ( 0*100 ) ;
DCL         PRINTNUM          UBIN STATIC INIT ( 0 ) ;
DCL         TIME              CHAR ( 11 ) STATIC ;
DCL         EVENT_HANDLER     ENTRY ASYNC ;
DCL         INTERRUPT_HANDLER ENTRY ASYNC ;

%B$CGAURD
          ( FPTN           = B$CGAURD,
            STCLASS        = "BASED ( IO_BUFFER$ )" ) ;

%B$ALT ;

%B$TCB ;

%EQU_CG ;

%F$DCB ;

%FPT_ACTIVATE
          ( FPTN           = FPT_ACTIVATE_CG,
            STCLASS        = STATIC,
            DCB            = M$CG,
            DISCONNECT     = YES ) ;

%FPT_CGCTL
          ( FPTN           = FPT_CGCTL_CG,
            DCB            = M$CG,
            CGCP           = VLP_CGCP_SPECIAL ) ;

%FPT_CLOSE
          ( FPTN           = FPT_CLOSE_CG,
            STCLASS        = STATIC,
            DCB            = M$CG ) ;

%FPT_CLOSE
          ( FPTN           = FPT_CLOSE_DELETE,
            DCB            = M$PRINT,
            DISP           = RELEASE ) ;

%FPT_CLOSE
          ( FPTN           = FPT_CLOSE_INPUT,
              DISP          = SAVE ) ;

%FPT_EVENT
            ( FPTN          = FPT_EVENT,
              UENTRY        = EVENT_HANDLER ) ;

%FPT_GETDCB
            ( FPTN          = FPT_GETDCB,
              DCBNAME       = VLP_NAME_DCB,
              DCBNUM        = DCB_NUM ) ;

%FPT_INT
            ( FPTN          = FPT_INT,
              UENTRY        = INTERRUPT_HANDLER ) ;

%FPT_OPEN
            ( FPTN          = FPTOPEN_CG,
              STCLASS       = CONSTANT,
              ACCT          = VLPACCT,
              ASN           = COMGROUP,
              AU            = YES,
              CTG           = YES,
              DCB           = M$CG,
              EXIST         = OLDFILE,
              FUN           = CREATE,
              NAME          = VLPNAME,
              QISS          = YES,
              SCRUB         = YES,
              SETSTA        = VLPSETSTA_AU ) ;

%FPT_OPEN
            ( FPTN          = FPTOPEN_SPECIAL,
              STCLASS       = CONSTANT,
              ACCT          = VLPACCT,
              ASN           = COMGROUP,
              DCB           = M$SPECIAL,
              FUN           = UPDATE,
              NAME          = VLPNAME,
              SCRUB         = YES,
            SETSTA          = VLPSETSTA_SPECIAL ) ;

%FPT_OPEN
          ( FPTN            = FPT_OPEN_INPUT,
            ASN             = FILE,
            EXIST           = NEWFILE,
            FUN             = CREATE,
            NAME            = VLP_NAME_PRINT,
            ORG             = CONSEC,
            SCRUB           = YES ) ;

%FPT_OPEN
          ( FPTN            = FPT_OPEN_OUTPUT,
            ASN             = FILE,
            DCB             = M$PRINT,
            FUN             = IN,
            NAME            = VLP_NAME_PRINT,
            ORG             = CONSEC,
            SCRUB           = YES ) ;

%FPT_READ
          ( FPTN            = FPT_READ_CG,
            BUF             = IO_BUFFER,
            DCB             = M$CG,
            EVENT           = %PRINT_READ_EVENT,
            STATION         = VLP_STATION,
            WAIT            = NO ) ;

%FPT_READ
          ( FPTN            = FPT_READ_SPECIAL,
            BUF             = IO_BUFFER,
            DCB             = M$SPECIAL,
            STATION         = VLP_STATION_SPECIAL ) ;

%FPT_READ
          ( FPTN            = FPT_READ_OUTPUT,
            BUF             = PRINT_BUFFER,
            DCB             = M$PRINT ) ;

```

```
%FPT_RELDCB
          ( FPTN           = FPT_RELDCB ) ;

%FPT_WAIT
          ( FPTN           = FPTWAIT_ASECOND,
            STCLASS        = CONSTANT,
            UNITS          = 1 ) ;

%FPT_WRITE
          ( FPTN           = FPT_WRITE_CG,
            STCLASS        = STATIC,
            BUF            = PRINT_BUFFER,
            DCB            = M$CG,
            STATION        = VLP_STATION_TERM ) ;

%FPT_WRITE
          ( FPTN           = FPT_WRITE_INPUT,
            BUF            = IO_BUFFER ) ;

%VLP_CGCP
          ( FPTN           = VLP_CGCP_OUTSYM,
            CONMSG         = YES,
            DCBCONAU       = NO,
            DCBCONWA       = YES,
            MINPG          = 4,
            RAS            = YES,
            TERMCONAU      = NO,
            TERMCONNAU     = NO ) ;

%VLP_CGCP
          ( FPTN           = VLP_CGCP_SPECIAL,
            CONMSG         = NO ) ;

%VLP_NAME
          ( FPTN           = VLP_NAME_DCB,
            LEN            = 31 ) ;

%VLP_NAME
          ( FPTN           = VLP_NAME_PRINT,
            LEN            = 31 ) ;

%VLP_NAME
          ( FPTN           = VLPNAME,
            STCLASS        = CONSTANT,
            NAME           = 'INTERNALS_CG' ) ;

%VLP_ACCT
          ( FPTN           = VLPACCT,
            STCLASS        = CONSTANT
/*
           ,ACCT           = 'LJSHOST'
*/
                                         ) ;

%VLP_SETSTA
          ( FPTN           = VLPSETSTA_AU,
            STCLASS        = CONSTANT,
            MYSTATION      = 'AU' ) ;

%VLP_SETSTA
          ( FPTN           = VLPSETSTA_SPECIAL,
            STCLASS        = CONSTANT,
            MYSTATION      = 'SPECIAL' ) ;

%VLP_STATION
          ( FPTN           = VLP$STATION,
            STCLASS        = BASED ) ;

%VLP_STATION
          ( FPTN           = VLP_STATION ) ;

%VLP_STATION
          ( FPTN           = VLP_STATION_SPECIAL,
            EOFNONE        = YES,
            MSGTYP         = 'PRINT    ' ) ;

%VLP_STATION
          ( FPTN           = VLP_STATION_TERM ) ;
```

```

%VLP_STATION
            ( FPTN              = VLP_STATION_DCB ) ;

%XUG_GETCMD
            ( NAME              = XUGGETCMD,
              STCLASS           = CONSTANT ) ;

%EJECT ;
/*
    Set break control.
*/

CALL M$INT ( FPT_INT ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;

M$CG$ = DCBADDR ( DCBNUM ( M$CG ) ) ;
M$PRINT$ = DCBADDR ( DCBNUM ( M$PRINT ) ) ;
M$SPECIAL$ = DCBADDR ( DCBNUM ( M$SPECIAL ) ) ;

/*
    Create a new comgroup and modify its control parameters.
    The first CGCTL specifies CONMSG=NO so the SPECIAL station can connect
    without requiring activation.  After that, another CGCTL is done specifying
    the normal parameters that we should use.
*/

CALL M$OPEN ( FPTOPEN_CG ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
CALL M$CGCTL ( FPT_CGCTL_CG ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
CALL M$OPEN ( FPTOPEN_SPECIAL ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
FPT_CGCTL_CG.CGCP_ = VECTOR ( VLP_CGCP_OUTSYM ) ;
CALL M$CGCTL ( FPT_CGCTL_CG ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;

/*
    Set event control.
*/

CALL M$EVENT ( FPT_EVENT ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;

/*
```

```
   Start the first no-wait read.
*/

CALL M$READ ( FPT_READ_CG ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;

/*
   Start normal operation of getting PRINT files and printing them if possible.
*/

DO WHILE %YES# ;

   DO INHIBIT ;
      DO WHILE EVENTS.PRINT_COMP ;
         CALL PROCESS_READ_EVENT ;
         EVENTS.PRINT_COMP = %NO# ;
         IO_BUFFER = ' ' ;
         CALL M$READ ( FPT_READ_CG ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
      END ;
      IF ( VLP_STATION_TERM.STATION# ~= ' ' ) THEN DO ;
         CALL PREPARE_PRINT_RECORD
            WHENRETURN DO ;
               CALL M$WRITE ( FPT_WRITE_CG ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
            END ;
      END ; ELSE DO ;
         CALL M$WAIT ( FPTWAIT_ASECOND ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
      END ;
   END ;

END ;

%EJECT ;
PREPARE_PRINT_RECORD : PROC ALTRET ;

   /*
      Puts a record in the buffer to send to the "printer", if we have any files
      that are ready to be printed.

   */

```

```
/*

   If there isn't a print file currently open, then try to find one and open
   it.  ALTRETURN if none exist.

*/

IF ( CUR_PRINTFILE < 0 ) THEN DO ;
   CALL GET_PRINTABLE_INDEX ( CUR_PRINTFILE ) ;
   IF ( CUR_PRINTFILE < 0 ) THEN DO ;
      ALTRETURN ;
   END ; ELSE DO ;
      CALL CONCAT ( VLP_NAME_PRINT.NAME#,
                    'PRINT:',
                    PRINT_FILES.NAME_EXT ( CUR_PRINTFILE ) ) ;
      CALL INDEX ( I,
                   ' ',
                   VLP_NAME_PRINT.NAME#,
                   0 ) ;
      VLP_NAME_PRINT.L# = I ;
      CALL M$OPEN ( FPT_OPEN_OUTPUT ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
   END ;
END ;

/*
   Put a record in the buffer.  When at EOF, send a formfeed and delete
   the file.
*/

CALL M$READ ( FPT_READ_OUTPUT )
      WHENALTRETURN DO ;
         IF ( B$TCB$->B$TCB.ALT$->B$ALT.ERR.CODE = %E$EOF ) THEN DO ;
            PRINT_BUFFER = BINASC ( %FORM_FEED ) ;
            FPT_WRITE_CG.BUF_.BOUND = 0 ;
            CALL M$CLOSE ( FPT_CLOSE_DELETE ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
            PRINT_FILES.NAME.STATION ( CUR_PRINTFILE ) = ' ' ;
            CUR_PRINTFILE = -1 ;
            RETURN ;
         END ; ELSE DO ;
```

```
                    CALL ERROR_EXIT ; /* DOES NOT RETURN */
                END ;
            END ;
    FPT_WRITE_CG.BUF_.BOUND = M$PRINT$->F$DCB.ARS# - 1 ;

    RETURN ;

END PREPARE_PRINT_RECORD ;

%EJECT ;
PROCESS_READ_EVENT : PROC ;

    /*
        The message just read could be an AU event or a record to be written into
        one of the print files being created.  Take care of it appropriately.
    */

DCL        EOF                     BIT ( 1 ) ;

    IF ( M$CG$->F$DCB.LASTSTA$->VLP$STATION.MSGTYP# = %AU_MESSAGE ) THEN DO ;
        DO CASE B$CGAURD.EVENT ;

            /*
                This is a new DCB station.  This station needs to be activated so
                that it can start shoveling data into the comgroup.  An entry is
                then built in the print file table containing the station name and
                file extension, along with a non-zero DCB number.  This number
                signifies that this print file is not complete (it can't be printed
                yet).  The DCB number is also necessary when writing records to the
                file.
            */

            CASE ( %CG_DOPN# ) ;
                VLP_STATION_DCB.STATION# = B$CGAURD.STATION ;
                FPT_ACTIVATE_CG.STATION_ = VECTOR ( VLP_STATION_DCB ) ;
                CALL M$ACTIVATE ( FPT_ACTIVATE_CG ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
                CALL GET_NEW_INDEX ( IDX ) ;
                PRINT_FILES.NAME.STATION ( IDX ) = B$CGAURD.STATION ;
                EXTENSION = EXTENSION + 1 ;
    CALL BINCHAR ( PRINT_FILES.NAME.EXTENSION ( IDX ),
                   EXTENSION ) ;
    CALL M$GETDCB ( FPT_GETDCB ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
    PRINT_FILES.DCB_NUM ( IDX ) = DCB_NUM ;
    CALL CONCAT ( VLP_NAME_PRINT.NAME#,
                  'PRINT:',
                  PRINT_FILES.NAME_EXT ( IDX ) ) ;
    CALL INDEX ( I,
                 ' ',
                 VLP_NAME_PRINT.NAME#,
                 0 ) ;
    VLP_NAME_PRINT.L# = I ;
    FPT_OPEN_INPUT.V.DCB# = DCB_NUM ;
    CALL M$OPEN ( FPT_OPEN_INPUT ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;

/*
    This event signifies that the DCB station is finished writing data
    into the comgroup and is going away.  The records in the comgroup
    that haven't been processed yet are read by the SPECIAL station
    using WAIT IO and EOFNONE and written to the print file.  The
    print file is then closed and the DCB number in the print file table
    is set to zero to signal that this file is ready to print.

    The reason that the DCB stations must be activated upon connection
    to the comgroup is now apparent - if this closing station immediately
    reconnected and started writing records, we would put the new data
    is this old print file.
*/

CASE ( %CG_DCLS# ) ;
    CALL GET_STATION_INDEX ( B$CGAURD.STATION,
                             IDX ) ;
    IF ( IDX >= 0 ) THEN DO ;
        VLP_STATION_SPECIAL.STATION# = PRINT_FILES.NAME.STATION ( IDX ) ;
        FPT_WRITE_INPUT.V.DCB# = PRINT_FILES.DCB_NUM ( IDX ) ;
        EOF = %NO# ;
        DO WHILE NOT EOF ;
            CALL M$READ ( FPT_READ_SPECIAL )
                    WHENALTRETURN DO ;
                  IF ( B$TCB$->B$TCB.ALT$->B$ALT.ERR.CODE = %E$EOF ) THEN DO ;
                     EOF = %YES# ;
                  END ; ELSE DO ;
                     CALL ERROR_EXIT ; /* DOESN'T RETURN */
                  END ;
               END ;
         IF NOT EOF THEN DO ;
            FPT_WRITE_INPUT.BUF_.BOUND = M$SPECIAL$->F$DCB.ARS# - 1 ;
            CALL M$WRITE ( FPT_WRITE_INPUT ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
         END ;
      END ;
      FPT_CLOSE_INPUT.V.DCB# = PRINT_FILES.DCB_NUM ( IDX ) ;
      CALL M$CLOSE ( FPT_CLOSE_INPUT ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
      FPT_RELDCB.V.DCB# = PRINT_FILES.DCB_NUM ( IDX ) ;
      CALL M$RELDCB ( FPT_RELDCB ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
      PRINT_FILES.DCB_NUM ( IDX ) = 0 ;
   END ;

/*
   A terminal station, "printer", wants to connect.  Activate it and
   keep track of its station name so we can print files.
*/

CASE ( %CG_TCON# ) ;
   VLP_STATION_TERM.STATION# = B$CGAURD.STATION ;
   FPT_ACTIVATE_CG.STATION_ = VECTOR ( VLP_STATION_TERM ) ;
   CALL M$ACTIVATE ( FPT_ACTIVATE_CG ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;

/*
   A terminal station, "printer", just disconnected.  Save the print
   file if it's only partially printed and blank the terminal name so
   we don't try to print any files.
*/

CASE ( %CG_TDSC# ) ;
   IF ( VLP_STATION_TERM.STATION# = B$CGAURD.STATION ) THEN DO ;
      VLP_STATION_TERM.STATION# = ' ' ;
      IF M$PRINT$->F$DCB.FCD# THEN DO ;
         FPT_CLOSE_INPUT.V.DCB# = DCBNUM ( M$PRINT ) ;
```

```
                  CALL M$CLOSE ( FPT_CLOSE_INPUT ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
               END ;
            END ;

         /*
            Someone just typed break at the terminal.  Deactivate the station
            and close a partially printed file so it can be reprinted.
         */

         CASE ( %CG_TBRK# ) ;
            FPT_ACTIVATE_CG.STATION_ = VECTOR ( VLP_STATION_TERM ) ;
            CALL M$DEACTIVATE ( FPT_ACTIVATE_CG ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
            VLP_STATION_TERM.STATION# = ' ' ;
            IF M$PRINT$->F$DCB.FCD# THEN DO ;
               FPT_CLOSE_INPUT.V.DCB# = DCBNUM ( M$PRINT ) ;
               CALL M$CLOSE ( FPT_CLOSE_INPUT ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
            END ;
         CASE ( ELSE ) ;
            /*
               Ignore the other possible AU events.
            */
      END ;

   END ; ELSE DO ;
      /*
         This is a record to be written to a print file.
      */
      CALL GET_STATION_INDEX ( M$CG$->F$DCB.LASTSTA$->VLP$STATION.STATION#,
                               IDX ) ;
      IF ( IDX >= 0 ) THEN DO ;
         FPT_WRITE_INPUT.V.DCB# = PRINT_FILES.DCB_NUM ( IDX ) ;
         FPT_WRITE_INPUT.BUF_.BOUND = M$CG$->F$DCB.ARS# - 1 ;
         CALL M$WRITE ( FPT_WRITE_INPUT ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
      END ;
   END ;

   RETURN ;

END PROCESS_READ_EVENT ;
```

```

%EJECT ;
GET_NEW_INDEX : PROC ( IDX ) ;

   /*
      Get an unused entry in the print file table.  Returns -1 if none
      are available.
   */

DCL          IDX              SBIN ;

DCL          I                UBIN ;


   I = 0 ;
   DO WHILE ( I <= PRINTNUM ) AND ( PRINT_FILES.NAME.STATION ( I ) ~= ' ' ) ;
      I = I + 1 ;
   END ;

   IF ( I = PRINTNUM ) AND ( PRINTNUM < 100 ) THEN DO ;
      PRINTNUM = PRINTNUM + 1 ;
      IDX = I ;
   END ; ELSE IF ( I < PRINTNUM ) THEN DO ;
      IDX = I ;
   END ; ELSE DO ;
      IDX = -1 ;
   END ;

   RETURN ;

END GET_NEW_INDEX ;

%EJECT ;
GET_PRINTABLE_INDEX : PROC ( IDX ) ;

   /*
      Get the index of a printable file - name not blank and DCB number zeroed.
      -1 is returned if none exist.
   */
```

```

DCL         IDX         SBIN ;

DCL         I              UBIN ;


   I = 0 ;
   IDX = -1 ;
   DO WHILE ( I <= PRINTNUM ) AND ( IDX = -1 ) ;
      IF ( PRINT_FILES.NAME.STATION ( I ) ~= ' ' ) AND ( PRINT_FILES.DCB_NUM ( I ) = 0 ) THEN DO ;
         IDX = I ;
      END ;
      I = I + 1 ;
   END ;

   RETURN ;

END GET_PRINTABLE_INDEX ;

%EJECT ;
GET_STATION_INDEX : PROC ( STATION, IDX ) ;

   /*
      Get the index of the given station.  -1 is returned if it doesn't exist.
   */

DCL         STATION     CHAR ( 8 ) ;
DCL         IDX         SBIN ;

DCL         I              UBIN ;


   I = 0 ;
   IDX = -1 ;
   DO WHILE ( I <= PRINTNUM ) AND ( IDX = -1 ) ;
      IF ( STATION = PRINT_FILES.NAME.STATION ( I ) ) AND ( PRINT_FILES.DCB_NUM ( I ) ~= 0 ) THEN D
         IDX = I ;
      END ;
      I = I + 1 ;
```

```
    END ;

    RETURN ;

END GET_STATION_INDEX ;

%EJECT ;
ERROR_EXIT : PROC ;

    IF ( VLP_STATION_TERM.STATION# ~= ' ' ) THEN DO ;
        FPT_ACTIVATE_CG.STATION_ = VECTOR ( VLP_STATION_TERM ) ;
        CALL M$DEACTIVATE ( FPT_ACTIVATE_CG ) WHENALTRETURN DO ; END ;
    END ;
    CALL XUG$CLOSE_DCBS ( XUGGETCMD ) ;
    CALL M$MERC ;
    CALL M$EXIT ;

END ERROR_EXIT ;



END OUTSYM_AU ;

%EOD ;
INTERRUPT_HANDLER : PROC ASYNC ;

/*
    NAME:                   INTERRUPT_HANDLER
    PURPOSE:                Takes care of interrupts caused by typing break at the AU's
                            terminal.
    DESCRIPTION:            Closes all DCB's and exits.
*/


%INCLUDE CP_6 ;
%INCLUDE CP_6_SUBS ;
%INCLUDE XU_MACRO_C ;
%INCLUDE XUG_ENTRY ;

```

```
%XUG_GETCMD
           ( NAME                  = XUGGETCMD,
             STCLASS               = CONSTANT ) ;

%EJECT ;
CALL XUG$CLOSE_DCBS ( XUGGETCMD ) ;
CALL M$EXIT ;

END INTERRUPT_HANDLER ;

%EOD ;
EVENT_HANDLER : PROC ASYNC ;

/*
    NAME:            EVENT_HANDLER
    PURPOSE:         Takes care of events caused by IO completion.
    DESCRIPTION:     Sets the appropriate flags and returns.
*/


%INCLUDE CP_6 ;
%INCLUDE CP_6_SUBS ;

%B$TCB ;

%B$NWIO
            ( STCLASS              = "BASED ( B$NWIO$ )" ) ;

%EQU          PRINT_READ_EVENT  = 1001 ;
%EQU          CONSOLE_READ_EVENT= 1002 ;

DCL           B$NWIO$              PTR ;
DCL           B$TCB$               PTR SYMREF ;
DCL       1   EVENTS               SYMREF,
                2  PRINT_COMP      BIT ( 1 ) UNAL,
                2  CONSOLE_COMP    BIT ( 1 ) UNAL,
                2  BAD_COMP        BIT ( 1 ) UNAL,
                2  ERR             BIT ( 36 ),
                2  TYC             BIT ( 36 ) ;
```

```

%EJECT ;
DO INHIBIT ;
    B$NWIO$ = B$TCB$->B$TCB.STK$ ;
    DO CASE ( B$NWIO.EVID ) ;
        CASE ( %PRINT_READ_EVENT ) ;
            EVENTS.PRINT_COMP = %YES# ;
        CASE ( %CONSOLE_READ_EVENT ) ;    /* not currently implemented */
            EVENTS.CONSOLE_COMP = %YES# ;
        CASE ( ELSE ) ;
            EVENTS.BAD_COMP = %YES# ;
    END ;
    EVENTS.ERR = B$NWIO.ERR ;
    EVENTS.TYC = B$NWIO.TYC ;
END ;

RETURN ;

END EVENT_HANDLER ;
```

```
/*M* INT_CGEX_SI2 internals class comgroup example - SI2 of 2 */
DCB_PRINT : PROC MAIN ;

/*
   NAME:            DCB_PRINT
   PURPOSE:         Copies the file specified to the OUTSYM comgroup.
   DESCRIPTION:     The DCB station opens to the comgroup using its sysid as its
                    name.  It opens the print file specified in the CCBUF:
                    "!INT_CGLAB_2. (fid)" and writes those records to the comgroup.
*/


%INCLUDE      CP_6 ;
%INCLUDE      CP_6_SUBS ;
%INCLUDE      B_ERRORS_C ;
%INCLUDE      XUX$INTERFACE_M ;
%INCLUDE      B$JIT ;

DCL           B$TCB$           PTR SYMREF ;
DCL           B$JIT$           PTR SYMREF ;
DCL           EOF              BIT ( 1 ) STATIC INIT ( '0'B ) ;
DCL           IO_BUFFER        CHAR ( 200 ) STATIC ;
DCL           JUNK             CHAR ( 12 ) STATIC ;
DCL           M$CG             DCB ;
DCL           M$CG$            PTR ;
DCL           M$PRINT          DCB ;
DCL           M$PRINT$         PTR ;
DCL           POS1             UBIN ;

DCL           XUX$GETLINE      ENTRY ( 1 ) ALTRET ;
DCL           XUX$CLEANUP      ENTRY ALTRET ;

%B$ALT ;

%B$TCB ;

%F$DCB ;
```

```

%FPT_CLOSE
          ( FPTN           = FPTCLOSE_CG,
            STCLASS        = CONSTANT,
            DCB            = M$CG ) ;

%FPT_CLOSE
          ( FPTN           = FPTCLOSE_PRINT,
            STCLASS        = CONSTANT,
            DCB            = M$PRINT ) ;

%FPT_FID
          ( FPTN           = FPT_FID_PRINT,
            ACCT           = VLP_ACCT_PRINT,
            ASN            = JUNK,
            NAME           = VLP_NAME_PRINT,
            PASS           = JUNK,
            RES            = JUNK,
            RESULTS        = JUNK,
            SN             = JUNK,
            WSN            = JUNK ) ;

%FPT_OPEN
          ( FPTN           = FPTOPEN_CG,
            STCLASS        = CONSTANT,
            ACCT           = VLPACCT,
            ASN            = COMGROUP,
            DCB            = M$CG,
            FUN            = UPDATE,
            NAME           = VLPNAME,
            SCRUB          = YES,
            SETSTA         = VLP_SETSTA_DCB ) ;

%FPT_OPEN
          ( FPTN           = FPTOPEN_PRINT,
            STCLASS        = CONSTANT,
            ACCT           = VLP_ACCT_PRINT,
            ASN            = FILE,
            DCB            = M$PRINT,
            FUN            = IN,
            NAME           = VLP_NAME_PRINT,
            ORG            = CONSEC,
            SCRUB          = YES ) ;

%FPT_READ
          ( FPTN           = FPT_READ_PRINT,
            BUF            = IO_BUFFER,
            DCB            = M$PRINT ) ;

%FPT_WRITE
          ( FPTN           = FPT_WRITE_CG,
            BUF            = IO_BUFFER,
            DCB            = M$CG,
            STATION        = VLPSTATION_AU ) ;

%VLP_ACCT
          ( FPTN           = VLPACCT,
            STCLASS        = CONSTANT
/*
           ,ACCT           = 'LJSHOST'
*/
                                       ) ;

%VLP_ACCT
          ( FPTN           = VLP_ACCT_PRINT ) ;

%VLP_NAME
          ( FPTN           = VLPNAME,
            STCLASS        = CONSTANT,
            NAME           = 'INTERNALS_CG' ) ;

%VLP_NAME
          ( FPTN           = VLP_NAME_PRINT ) ;

%VLP_SETSTA
          ( FPTN           = VLP_SETSTA_DCB ) ;

%VLP_STATION
```

```
          ( FPTN           = VLPSTATION_AU,
            STCLASS        = CONSTANT,
            ANYDCB         = YES,
            MSGTYP         = 'PRINT   ' ) ;

%XUX$PARAM_NO_PARSE
          ( NAME           = XUX_CCBUF,
            STCLASS        = STATIC,
            BUFFER         = IO_BUFFER,
            DISP_ONLY      = YES ) ;

%EJECT ;
M$CG$ = DCBADDR ( DCBNUM ( M$CG ) ) ;
M$PRINT$ = DCBADDR ( DCBNUM ( M$PRINT ) ) ;

/*
   Use SYSID as station name.
*/

CALL BINCHAR ( VLP_SETSTA_DCB.MYSTATION#,
              B$JIT$->B$JIT.SYSID ) ;

/*
   Get the name of the file to print out of the CCBUF and open it.
*/

DO UNTIL ( XUX_CCBUF.CMD_LEN > 0 ) ;
   CALL XUX$GETLINE ( XUX_CCBUF ) ALTRET ( XIT ) ;
END ;
CALL XUX$CLEANUP ;
CALL INDEX ( POS1,
            ')',
            IO_BUFFER,
            1 ) ;
FPT_FID_PRINT.TEXTFID_ = VECTOR ( SUBSTR ( IO_BUFFER,
                                           1,
                                           POS1 - 1 ) ) ;
CALL M$FID ( FPT_FID_PRINT ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
CALL M$OPEN ( FPTOPEN_PRINT ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
```

```

/*
    Open the comgroup.
*/

CALL M$OPEN ( FPTOPEN_CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;

/*
    Read the messages from the print file and send them to the AU station.
*/

DO WHILE NOT EOF ;

    CALL M$READ ( FPT_READ_PRINT )
            WHENALTRETURN DO ;
                IF ( B$TCB$->B$TCB.ALT$->B$ALT.ERR.CODE = %E$EOF ) THEN DO ;
                    EOF = %YES# ;
                END ; ELSE DO ;
                    GOTO REPORT_ERROR_AND_EXIT ;
                END ;
            END ;
    IF NOT EOF THEN DO ;
        FPT_WRITE_CG.BUF_.BOUND = M$PRINT$->F$DCB.ARS# - 1 ;
        CALL M$WRITE ( FPT_WRITE_CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
    END ;

END ;

CALL M$CLOSE ( FPTCLOSE_CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
CALL M$CLOSE ( FPTCLOSE_PRINT ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
CALL M$EXIT ;

REPORT_ERROR_AND_EXIT: ;
    IF M$CG$->F$DCB.FCD# THEN DO ;
        CALL M$CLOSE ( FPTCLOSE_CG ) ;
    END ;
    IF M$PRINT$->F$DCB.FCD# THEN DO ;
        CALL M$CLOSE ( FPTCLOSE_PRINT ) ;
    END ;
    CALL M$MERC ;

XIT: ;
    CALL M$EXIT ;

END DCB_PRINT ;
```

LAB 5

ASSIGNMENT

LAB #5 [COMMAND PROCESSORS]

SCENARIO:

- You managed to hide the System Support Manual but the head of the computer science department has just read the System Programmer Guide. He's absolutely sure that the computer science department can benefit from the security a command program could provide. Who are you to argue? As an exercise, you decide to write your own simple command program to learn the "ins and outs" of command program creation.

LAB #5 [COMMAND PROCESSORS]

PROBLEM:

1. Make a command program, called INTCPnn where nn is your group number.

This command program will have several functions.

a. It will prompt with a '#'.

LAB #5 [COMMAND PROCESSORS]

b. It will recognize when control-Y or break is hit. If control-Y is struck, it will prompt with '##'. If break is struck, it will "ignore" it (i.e. just return). When the '##' is issued, it will accept two responses: DELTA or GO. (Do not use the parser, just accept th string and do a sized compare against a literal).

LAB #5 [COMMAND PROCESSORS]

c. You will write a small program, called, INTCPTEST. This program will simply wait (M$READ from the terminal) for you to strike control-Y so that you can test step b. Your command program will have to know how to call this program from you account.

LAB #5 [COMMAND PROCESSORS]

d. Your command program will also accept "PCL" as a command, and call up PCL.:SYS when typed in. This command will only be legal at jobstep (the '#' prompt, not the '##' prompt).

e. Your command program will also accept "IBEX" as a command, and will exit to IBEX when "IBEX" is typed in. This command is only legal at jobstep.

LAB #5 [COMMAND PROCESSORS]

2. Install your command program with SPIDER.

3. Check it out.

17:45 AUG 29 '85 LAB5_ASSIGNMENT

INTERNALS CLASS

COMMAND PROCESSORS LAB

1) Make a command processor, called:

INTCPnn

where

nn is your group number.

This command processor will have several functions:

a) It will prompt with a '#'.

b) It will recognize when control-Y or break is hit. If control-Y is struck, it will prompt with '##'. If break is struck, it will "ignore" it...i.e. just return. When the '##' is issued, it will accept two responses: DELTA or GO. (Do not use the parser, just accept the string and do a sized compare against a literal).

c) You will write a small program, called

INTCPTEST.

This program will simply wait (M$READ from the terminal) for you to strike control-Y so that you can test step b. Your command program will have to know how to call this program from your account.

d) Your command program will also accept "PCL" as a command, and call up PCL.:SYS when typed in. This command will only be legal at jobstep (the '#' prompt, not the '##' prompt).

e) Your command program will also accept "IBEX" as a command, and will exit to IBEX when "IBEX" is typed in. This command is only legal at jobstep.

2) Install your command program with SPIDER.

3) Check it out.

LAB 5

ANSWER

17:46 AUG 29 '85 LAB5_JCL.

```
!JOB WSN=UPSTAIRS
!RES MEM=300,TIME=2
!PL6 LAB5_CP_SI OVER LAB5_CP_OU6(LS,SR(.:LIBRARY),SCHEMA)
!LINK LAB5_CP_OU6 OVER INTCP(MAP(VALUE,NAME),NOSH,SYS, ;
        UNSAT=:LIB_SYSTEM.:SYS,TCB=3,PRIV(JIT))
!PL6 LAB5_CPTEST_SI OVER LAB5_CPTEST_OU6(LS,SR(.:LIBRARY),SCHEMA)
!LINK LAB5_CPTEST_OU6 OVER INTCPTEST
!SPIDER
DEL INTCP
IN INTCP,CP FROM INTCP.CKWHOST
L INTCP
END
!C LAB5_JCL TO LP(K)
```

```
17:46 AUG 29 '85 LAB5_CPTEST_SI.

U40CPTEST: PROC MAIN;
%INCLUDE CP_6;
%B$TCB;
%B$ALT;
%INCLUDE B_ERRORS_C;
DCL M$SI DCB;
DCL B$TCB$ PTR SYMREF;
%FPT_READ (FPTN=FPT$READ,STCLASS=CONSTANT,DCB=M$SI,BUF=BUFFER);
DCL BUFFER CHAR(120) STATIC;

REREAD: ;
      CALL M$READ(FPT$READ)
            WHENALTRETURN DO;
                  IF B$TCB$ -> B$TCB.ALT$ -> B$ALT.ERR.CODE = %E$EOF
                  THEN GOTO XIT;
                  ELSE GOTO REREAD;
            END;
      GOTO REREAD;
XIT: ;
      CALL M$EXIT;
      END U40CPTEST;
```

```
U40CP: PROC MAIN;
%INCLUDE CP_6;
%INCLUDE B$JIT;

DCL B$JIT$      PTR SYMREF;
DCL B$TCB$      PTR SYMREF;
%FPT_READ (FPTN=FPT$READ, STCLASS=CONSTANT, DCB=M$UC);
%FPT_READ (STCLASS=AUTO);
%FPT_WRITE (FPTN=WRITE_SYNERR, STCLASS=CONSTANT, BUF=SYNERR, DCB=M$UC);
DCL SYNERR CHAR(0) CONSTANT INIT('U40CP can''t do that.');
%FPT_PROMPT (FPTN=PROMPT_SINGLE, PROMPT=SINGLE_SPLAT, STCLASS=CONSTANT, VFC=YES);
%FPT_PROMPT (FPTN=PROMPT_DOUBLE, PROMPT=DOUBLE_SPLAT, STCLASS=CONSTANT, VFC=YES);
DCL SINGLE_SPLAT CHAR(2) CONSTANT INIT('p#');
DCL DOUBLE_SPLAT CHAR(3) CONSTANT INIT('p##');
%FPT_CPEXIT (FPTN=FPT$CPEXIT, STCLASS=CONSTANT);
%FPT_CPEXIT (STCLASS="");
%VLP_ACCT (FPTN=ACCT_SYS, ACCT=':SYS    ', STCLASS=CONSTANT);
%VLP_NAME (FPTN=NAME_DELTA, NAME='DELTA', STCLASS=CONSTANT);
%VLP_NAME (FPTN=NAME_PCL, NAME='PCL', STCLASS=CONSTANT);
%VLP_NAME (FPTN=NAME_U40CPTEST, NAME='U40CPTEST', STCLASS=CONSTANT);
%VLP_NAME (FPTN=NAME_IBEX, NAME='IBEX', STCLASS=CONSTANT);
%FPT_ERRMSG (FPTN=FPT$ERRMSG, STCLASS=CONSTANT, SOURCE=PASS, OUTDCB1=M$UC);
%FPT_ERRMSG (STCLASS=AUTO);
DCL BUFFER CHAR(120);
%F$DCB;
%B$TCB;
%B$ALT;

   FPT_CPEXIT = FPT$CPEXIT; FPT_CPEXIT.V_ = VECTOR (FPT_CPEXIT.V);
   FPT_READ = FPT$READ; FPT_READ.V_ = VECTOR (FPT_READ.V);
   FPT_READ.BUF_ = VECTOR (B$JIT.CCBUF);
   FPT_ERRMSG = FPT$ERRMSG; FPT_ERRMSG.V_ = VECTOR (FPT_ERRMSG.V);
   FPT_ERRMSG.BUF_ = VECTOR (BUFFER);
   IF B$JIT.CPFLAGS1 & %CP_JSTEP# THEN DO;
      CALL M$PROMPT (PROMPT_SINGLE);
      READ_SINGLE: CALL M$READ (FPT_READ)
         WHENALTRETURN DO;
         FPT_ERRMSG.CODE_ = VECTOR (B$TCB$->B$TCB.ALT$->B$ALT.ERR);
         CALL M$ERRMSG (FPT_ERRMSG);
         GOTO IBEX;
      END;
   B$JIT.CCARS = DCBADDR(DCBNUM(M$UC))->F$DCB.ARS#;
   DO SELECT (SUBSTR(B$JIT.CCBUF,0,B$JIT.CCARS));
    SELECT ('PCL');
      FPT_CPEXIT.ACCT_ = VECTOR (ACCT_SYS);
      FPT_CPEXIT.NAME_ = VECTOR (NAME_PCL);

    SELECT('IBEX');
      IBEX: FPT_CPEXIT.ACCT_ = VECTOR (ACCT_SYS);
      FPT_CPEXIT.NAME_ = VECTOR (NAME_IBEX);

    SELECT('U40CPTEST');
      FPT_CPEXIT.NAME_ = VECTOR (NAME_U40CPTEST);
      FPT_CPEXIT.ACCT_ = VECTOR (B$JIT.ACCN);
      FPT_CPEXIT.ACCT_ = VECTOR (B$JIT.ACCN);

    SELECT(ELSE);
      CALL M$WRITE (WRITE_SYNERR);
      GOTO READ_SINGLE;
   END;
   CALL M$CPEXIT (FPT_CPEXIT)
   WHENALTRETURN DO;
      FPT_ERRMSG.CODE_ = VECTOR (B$TCB$->B$TCB.ALT$->B$ALT.ERR);
      CALL M$ERRMSG (FPT_ERRMSG);
      FPT_CPEXIT = FPT$CPEXIT; FPT_CPEXIT.V_ = VECTOR (FPT_CPEXIT.V);
      GOTO READ_SINGLE;
   END;
END; ELSE DO;
   CALL M$PROMPT (PROMPT_DOUBLE);
   READ_DOUBLE: CALL M$READ (FPT_READ) ALTRET (QUIT);
   B$JIT.CCARS = DCBADDR(DCBNUM(M$UC))->F$DCB.ARS#;
   DO SELECT (SUBSTR(B$JIT.CCBUF,0,B$JIT.CCARS));
    SELECT('DELTA');
      FPT_CPEXIT.DEBUG_ = VECTOR (NAME_DELTA);
      FPT_CPEXIT.V.CONT# = '1'B;

```

```
       SELECT('GO');
         FPT_CPEXIT.V.CONT# = '1'B;

       SELECT('QUIT');
         QUIT: FPT_CPEXIT.V.QUIT# = '1'B;

       SELECT(ELSE);
         CALL M$WRITE(WRITE_SYNERR);
         FPT_CPEXIT.V.CONT# = '1'B;
      END;
      CALL M$CPEXIT (FPT_CPEXIT)
      WHENALTRETURN DO;
         FPT_ERRMSG.CODE_ = VECTOR (B$TCB$->B$TCB.ALT$->B$ALT.ERR);
         CALL M$ERRMSG (FPT_ERRMSG);
         FPT_CPEXIT = FPT$CPEXIT; FPT_CPEXIT.V_ = VECTOR (FPT_CPEXIT.V);
         GOTO READ_DOUBLE;
      END;
   END;
END;
```

LAB 6

ASSIGNMENT

LAB #6 [FPRGS]

SCENARIO:

- You have just gotten a prerelease of C00 CP-6. Being a prerelease, everything doesn't quite work - one thing is ANLZ doesn't know how to ANLZ running FEPs yet. You call LADC and are told in order to look at running FEPs, you need to do A, B, and C - click.

LAB #6 [FPRGS]

PROBLEM – PART I

Write a host program that starts an FPRG, reads three pieces of information from the FPRG – the FEP number, the NODENAME, and the SYSID of user number 1, and writes the values out.

LAB #6 [FPRGS]

The FPRG must M$CVM onto BOBCAT data to get the FEP number, the NODENAME, and the pointer to the user tables. The pointer to the user tables can then be used to get the SYSID of user number 1.

Run your programs under DELTA, displaying the pointer to BOBCAT data and the pointer to the user tables.

LAB #6 [FPRGS]

PROBLEM – PART II

Write a host program what creates three windows on your terminal. The host program will read a character string from one window and write the character string to an FPRG. The FPRG will write the reverse character string out in the second window. The third window is for DELTA.

LAB #6 [FPRGS]

Run your programs under DELTA, displaying the character string before it is sent to the FPRG, after the FPRG reads it from the host program, and the reverse character string before the FPRG writes it out.

HINT: See M$LDEV and VLP_WINDOW in the Host Monitor Services Manual.

LAB #6 [FPRGS]

HINT: See Appendix B of the Fep Programming Concepts Manual for a sample FPRG and host program.

HINT: The structure for BOBCAT data is in G$BOBCAT_M.:LIBRARY.

HINT: The structure for the user tables, G$USER, is in GH_SCHD_M.ZZINT

17:47 AUG 29 '85 LAB6_ASSIGNMENT

CP-6 INTERNALS CLASS

FPRGS LAB

Part I

Write a host program that starts an FPRG, reads three pieces of information from the FPRG - the FEP number, the NODENAME, and the SYSID of user number 1, and writes the values out.

The FPRG must M$CVM onto BOBCAT data to get the FEP number, the NODENAME, and the pointer to the user tables. The pointer to the user tables can then be used to get the SYSID of user number 1.

Run your programs under DELTA, displaying the pointer to BOBCAT data and the pointer to the user tables.

HINT: See Appendix B of the FEP Programming Concepts Manual for a sample FPRG and host program.

HINT: The structure for BOBCAT data is in G$BOBCAT_M.:LIBRARY.

HINT: The structure for the user tables, G$USER, is in GH_SCHD_M.ZZINT

HINT: The pointer to the user tables cannot be used directly.

Part II

Write a host program that creates three windows on your terminal. The host program will read a character string from one window and write the character string to an FPRG. The FPRG will write the reverse character string out in the second window. The third window is for DELTA.

Run your programs under DELTA, displaying the character string before it is sent to the FPRG, after the FPRG reads it, and the reverse character string before the FPRG writes it out.

HINT: See M$LDEV and VLP_WINDOW in the Host Monitor Services Manual.

LAB 6

ANSWER

```
17:47 AUG 29 '85 LAB6_JCL

     1 - !JOB WSN=UPSTAIRS
     2 - !RES MEM=300,TIME=2
     3 - !PL6 LAB6_HOST_SI1 OVER LAB6_HOST_OU61(LS,SR(.:LIBRARY),SCHEMA)
     4 - !LINK LAB6_HOST_OU61 OVER LAB6:H1
     5 - !PL6 LAB6_FPRG_SI1 OVER LAB6_FPRG_OU61(LS,SR(.:LIBRARY),SCHEMA)
     6 - !FEPLINK LAB6_FPRG_OU61 OVER LAB6:F1
     7 - !PL6 LAB6_HOST_SI2 OVER LAB6_HOST_OU62(LS,SR(.:LIBRARY),SCHEMA)
     8 - !LINK LAB6_HOST_OU62 OVER LAB6:H2
     9 - !PL6 LAB6_FPRG_SI2 OVER LAB6_FPRG_OU62(LS,SR(.:LIBRARY),SCHEMA)
    10 - !FEPLINK LAB6_FPRG_OU62 OVER LAB6:F2
    11 - !C LAB6_JCL TO LP(K)
```

```
LAB6_FPRG_SI1: PROC MAIN;
%INCLUDE LCP_6;
%INCLUDE G$BOBCAT_M;
%G$BOBCAT(FPTN=G$BOBCAT,STCLASS="BASED(G$DS4$)");
%INCLUDE GM_VIRTUAL_E;
%INCLUDE GH_SCHD_M;
%G$USER(FPTN="G$USER(0:0)",STCLASS=BASED);
DCL G$DS4$ PTR SYMREF READONLY;
DCL G$BOBCATPTR UBIN(32) CONSTANT INIT(%GM_BOBCAT_BASE);
DCL G$BOBCAT$ REDEF G$BOBCATPTR PTR;
DCL M$HOST DCB;
%FPT_CVM(USERSEG=USEG, FROMSEG=FSEG, PAGES=256);
%VLP_SEGMENT(FPTN=USEG, PAGES=256);
%VLP_SEGMENT(FPTN=FSEG, PAGES=256);
%FPT_WRITE (FPTN=HOST_WRITE, BUF=HOST_BUF, DCB=M$HOST, STCLASS=CONSTANT);

DCL 1 HOST_BUF STATIC,
        2 FEP# UBIN(16),
        2 SYSID REDEF FEP# UBIN(16),
        2 NODENAME CHAR(8);
DCL USRT$ PTR;
DCL USRT_ADDR REDEF USRT$ UBIN(32);
DCL SEGOFFSET UBIN;
DCL NILPTR UBIN(32) CONSTANT INIT(0);
DCL NIL$ REDEF NILPTR PTR;

      USEG.BASE$ = G$DS4$;
      FSEG.BASE$ = G$BOBCAT$;
      CALL M$CVM (FPT_CVM);
      HOST_BUF.FEP# = G$BOBCAT.FEP#;
      HOST_BUF.NODENAME = G$BOBCAT.NODENAME;
      CALL M$WRITE (HOST_WRITE);
      USRT$ = G$BOBCAT.USRT$;
      FPT_CVM.V.TYPE = %G_CVM_MON#;
      IF USRT_ADDR > (1024*64)
      THEN SEGOFFSET = MOD(USRT_ADDR, 1024*64);
      ELSE SEGOFFSET = MOD(USRT_ADDR, 1024*4);
```

```
        FSEG.BASE$ = PINCRW(NIL$, USRT_ADDR - SEGOFFSET);
        CALL M$CVM (FPT_CVM);
        HOST_BUF.SYSID = PINCRW(G$DS4$, SEGOFFSET) -> G$USER.SYSID(1);
        CALL M$WRITE (HOST_WRITE);

XIT:
CALL M$EXIT;
END LAB6_FPRG_SI1;
```

17:47 AUG 29 '85 LAB6_FPRG_SI2

```
LAB6_FPRG_SI2: PROC MAIN;
%INCLUDE LCP_6;
DCL M$ME DCB;
DCL M$HOST DCB;
DCL I UBIN;
DCL SIZE UBIN;
%FPT_READ(FPTN=HOST_READ,BUF=CHAR_BUF,DCB=M$HOST,STCLASS=CONSTANT);
%FPT_WRITE(FPTN=TERMINAL_WRITE,BUF=REV_CHAR_BUF,DCB=M$ME,STCLASS=CONSTANT);
%FPT_WRITE (FPTN=HOST_WRITE, BUF=CHAR_BUF, DCB=M$HOST, STCLASS=CONSTANT);

DCL CHAR_BUF(0:80) CHAR(1) CALIGNED STATIC;
DCL CHAR_BUF1(0:80) REDEF CHAR_BUF UBIN BYTE CALIGNED;
DCL CHAR_BUF2 REDEF CHAR_BUF CHAR(81) CALIGNED;
DCL REV_CHAR_BUF(0:79) CHAR(1) CALIGNED STATIC;

      CALL M$READ (HOST_READ) ALTRET(XIT);
      SIZE = CHAR_BUF1(0);
      DO I = 0 TO SIZE-1;
            REV_CHAR_BUF(SIZE- I) = CHAR_BUF(I+1);
         END;
      CALL M$WRITE (TERMINAL_WRITE);
      CHAR_BUF2 = ' ';
      CHAR_BUF1(0) = SIZE;
      DO I = 0 TO 79;
            CHAR_BUF(I+1) = REV_CHAR_BUF(I);
         END;
      CALL M$WRITE (HOST_WRITE);

XIT:
CALL M$EXIT;
END LAB6_FPRG_SI2;
```

17:47 AUG 29 '85 LAB6_HOST_SI1

```
LAB6_HOST_SI1: PROC MAIN;
%INCLUDE CP_6;
DCL B$JIT$ PTR SYMREF;
%INCLUDE B$JIT;
DCL FPRG DCB;
DCL TEMP UBIN STATIC;
DCL 1 TEMP1 REDEF TEMP,
         2 * BIT(20),
         2 Y1 UBIN(8) UNAL,
         2 Y2 UBIN(8) UNAL;
DCL FPRG_BUF(0:9) UBIN BYTE CALIGNED STATIC INIT(0*0);
DCL 1 BUF STATIC,
         2 * CHAR(5) INIT('FEP#='),
         2 FEP# CHAR(3),
         2 * CHAR(11) INIT('  NODENAME='),
         2 NODENAME(0:7) CHAR(1),
         2 NODENAME1(0:7) REDEF NODENAME UBIN BYTE UNAL;
DCL 1 BUF1 STATIC,
         2 * CHAR(6) INIT('SYSID='),
         2 SYSID CHAR(3);
DCL I UBIN;
%FPT_WRITE (DCB=M$UC,FPTN=UC_WRITE,BUF=BUF);
%FPT_OPEN (DCB=FPRG, ORG=FPRG, RES='UC02', FPRG=VLP_FPRG);
%VLP_FPRG (NAME='LAB6:F1');
%FPT_READ (FPTN=FPRG_READ,BUF=FPRG_BUF,DCB=FPRG);

     CALL M$OPEN (FPT_OPEN);
     CALL M$READ (FPRG_READ);
     TEMP = 0;
     TEMP1.Y1 = FPRG_BUF(0);
     TEMP1.Y2 = FPRG_BUF(1);
     CALL BINCHAR(BUF.FEP#, TEMP);
     DO I = 0 TO 7;
          BUF.NODENAME1(I) = FPRG_BUF(I+2);
        END;
     CALL M$WRITE (UC_WRITE);
     CALL M$READ (FPRG_READ);
```

```
      TEMP = 0;
      TEMP1.Y1 = FPRG_BUF(0);
      TEMP1.Y2 = FPRG_BUF(1);
      CALL BINCHAR(BUF1.SYSID, TEMP);
      UC_WRITE.BUF_ = VECTOR(BUF1);
      CALL M$WRITE (UC_WRITE);
END LAB6_HOST_SI1;
```

```
LAB6_HOST_SI2: PROC MAIN;
%INCLUDE CP_6;
DCL B$JIT$ PTR SYMREF;
%INCLUDE B$JIT;
%F$DCB;
DCL FPRG DCB;
DCL I UBIN;
DCL SIZE UBIN;
DCL CHAR_BUF(0:79) CHAR CALIGNED STATIC INIT(' ');
DCL CHAR_BUF1 REDEF CHAR_BUF CHAR(80) CALIGNED;
DCL FPRG_BUF(0:80) CHAR(1) CALIGNED STATIC;
DCL FPRG_BUF1(0:80) REDEF FPRG_BUF UBIN BYTE CALIGNED;
%FPT_WRITE (DCB=M$UC,FPTN=UC_WRITE,BUF=CHAR_BUF);
%FPT_WRITE (DCB=FPRG,FPTN=FPRG_WRITE,BUF=FPRG_BUF);
%FPT_OPEN (DCB=FPRG, ORG=FPRG, RES='UC02', FPRG=VLP_FPRG);
%VLP_FPRG (NAME='LAB6:F2');
%FPT_READ (FPTN=FPRG_READ,BUF=FPRG_BUF,DCB=FPRG);
%FPT_READ (FPTN=UC_READ,BUF=CHAR_BUF,DCB=M$UC);
%FPT_LDEV(FPTN=LDEV_FPRG,STREAMNAME='UC02',WINDOW=VLP_WINDOW_FPRG);
%VLP_WINDOW(FPTN=VLP_WINDOW_FPRG,FWINDOW='UC01',LENGTH=6,WIDTH=80,
            POSITION=TOP);
%FPT_LDEV(FPTN=LDEV_DELTA,STREAMNAME='UC99',WINDOW=VLP_WINDOW_DELTA);
%VLP_WINDOW(FPTN=VLP_WINDOW_DELTA,FWINDOW='UC01',LENGTH=8,WIDTH=80,
            POSITION=BOTTOM);

     CALL M$LDEV (LDEV_FPRG);
     CALL M$LDEV (LDEV_DELTA);
     CALL M$OPEN (FPT_OPEN);
     CALL M$READ (UC_READ);
     SIZE = DCBADDR( DCBNUM(M$UC)) -> F$DCB.ARS#;
     FPRG_BUF1(0) = SIZE;
     DO I = 1 TO SIZE+1;
          FPRG_BUF(I) = CHAR_BUF(I-1);
        END;
     CALL M$WRITE (FPRG_WRITE);
     CALL M$READ (FPRG_READ);
     SIZE = FPRG_BUF1(0);
     CHAR_BUF1 = ' ';
     DO I = 1 TO SIZE+1;
          CHAR_BUF(I-1) = FPRG_BUF(I);
        END;
     CALL M$WRITE (UC_WRITE);
END LAB6_HOST_SI2;
```

LAB 7

17:06 AUG 29 '85 C00.

C01 - ANLZ Command Summary - FEPs
C02 - REC Command - FEPs

C03 - Monitor Trap Example
C04 - Handler Trap Example

C05 - LCP6 Trap Numbers
C06 - MCL Codes
C07 - ASDTs
C08 - Users HJIT
C09 - Low Real Memory
C10 - Status Register

17:06 AUG 29 '85 C01

ANLS Commands for FEPs

- Combinations of dumps/ANLZ
  - C00 dump / C00 ANLZ
  - C00 dump / Bootleg ANLZ
  - C01 dump / C01 ANLZ

ANLZ FEP Commands - Available on C00

ADD	Identifies channel context for FMT command to display.
BOB[CAT]	Displays the contents of the base of the BOBCAT data segment
CHN[TBL]	Produces formatted display of a channel table entry
DROP	The opposite of ADD
DU[MP]	Produces hex dump of specified area of FEP memory
FMT	Displays the context for a channel
ISA	Formats an Interrupt Save Area
LCT	Dumps the Line Control Table for a channel
MLCP	Produces a hex dump of specified MLCP memory
PL[UGH]	Displays the chain of calls in an AUTO Stack
REC[OVERY]	Formats the contents of the FEP Recovery Buffer
SPY	Displays all users currently on the FEP
TSA	Formats a Trap Save Area

Additional Commands with Bootleg ANLZ or C01 ANLZ:

ACC[RES]	Displays the contents of the Account Resource Table.
CHANNELS	Displays channel ids and channel types
DCB[S]	Displays DCBs for selected user(s).
ECCB	Displays the ECCB for selected user(s).
INT[CON]	Displays the contents of the Interrupt Table.
JIT	Displays selected items from the JIT for selected user(s).
LDCT	Displays the LDCT.
MEM[ORY]	Displays memory usage information
ROUTE	Displays network routiong info
SFI[LES]	Displays the contents of the Shared File Table.
SPY	Displays selected FEP users.
STAT[US]	Displays memory usage for selected user(s).
TCB	Displays the Task Control Block for selected user(s).
USR[T]	Displays the contents of the User Table.

C01 ANLZ cannot be used with C00 systems as there is a slight change to the DCB structure.

To get the latest and greatest Bootleg ANLZ, BEAM both ANLZ and ANLZ_FPRG from account ZZZTEST @L66A to your site.

You cannot always believe the info from the MEMORY and STAT commands when they are used on-line. They are not valid for C00 in any event.

17:06 AUG 29 '85 C02.

REC Command

- FCG-nnnn-3 See System Support Manual - Appendix B
  GHT-trap no-3 - Monitor Trap
  GHH-trap no-3 - Handler Trap

- S: status register See G$STATUS_REG

  The important thing here is the Level;
  .04 - inhibit level - could be monitor or handler
  .06 through .0A - its XDELTA
  .0C through .3B - Its a handler - use INTCON command
  .3D - Real Time Clock
  .3E - Scheduler Level
  .3F - User Execution

  To find all the active levels, look at low real memory
  by saying DU .3020,4 - see G$LOW_MEM

  To find the active domain, look in the user's HJIT at .507E
  See G$UHJIT.

%G_DMN_MON_SVC	0	%G_DMN_MON	4
%G_DMN_DB_SVC	1	%G_DMN_DB	5
%G_DMN_INT_SVC	2	%G_DMN_INT	6
%G_DMN_USR_SVC	3	%G_DMN_USR	7

- ISM2 Interrupt Save Mask2 - See G$ISM2
  The interesting field is the NATSAP:
  2 - Handler at interrupt level
  1 - User or User Service
  0 - Idle Loop

- TSA Trap Save Area - must determine the valid frame if C00.

  Check Status Register to see what level was running
  If monitor - default SYM of M:FEP.:SYS is the right one
  Say PL .4000 to see TSTACKU

  If Handler - use INTCON command to find user number (SPY CUN if C01)
  - use SPY to get M$LM fid
  - use JIT to get SLIB number
  - use SFILE to get library fid

  If Program Counter is ›= .60000 use SYM library fid
  If Program Counter is ‹ .60000 use SYM M$LM fid
  Say REC again to get valid P$, etc. (or say EV .addr)
  Say PL .A0000 for CUN to see Handler's Auto

  If C00 or Bootleg ANLZ you must get B3 and R3 from here;
  the B3 and R3 in the following ISA display will be as destroyed
  by the software to build the TSA.

If TRAP: is .01 or .63 - its an MCL
- R3 has the MCL code - see attached list
- B3 has the address of FPT -see manual

If TRAP: is .02 through .1F - its a hardware trap
See list

If TRAP: is .2F through .62 - its a software pseudo trap
See list

If TRAP: is .20 through .2E - its a bug

- Current User:

  If not running at level .3F this is not necessarily the current user, but instead the user that was running when we got the interrupt.

  This is the user that ANLZ displays on subsequent commands that specify CUN on C00, however. C01 gets it right.

  ANLZ doesn't understand CUN until after you have said REC if using C00 or Bootleg ANLZ.

  A couple of useful addresses: .6351 contains CUN
  .6352 and .6253 contain CU$

```
17:06 AUG 29 '85 C03

DRIBBLE ON @  18:21 08/21/85
!ANLZ A099
ANLZ C01
 :DFC01A099 for GHT-001304-3 at 13:24 AUG 06 '85 on LADC L66B
 KR is to blame

 Nodes:     L6XII      L66B
L6XII (node 12) Selected
L6XII    -REC

Screech Code:

 GHT-M01304-0  LCP6 Abort - Disabled Too Long


PPUT inconsistent.  Index = 1427
Current User: .5                TSA$: .50FF

TSA @.50FF:

  Watch Dog Timer Runout
TRAP: 4  TSAL$: .50B6      I: .0402  INST: .0000  Z: .0080

A$: .0          P: .25F99     B3: .452D      R3: .3A    S: .4004

Trap IC: GMA$LGP+.4D9

ISA:

P: .25F99     S: .6004  CHN: .FFFF  ISM1: .FFFF  ISM2: .9103

Interrupted address: GMA$LGP+.4D9

                   1       2       3       4       5       6       7
                ------ ------ ------ ------ ------ ------ ------
B Registers:     506E   9D10   452D  C0040  9003E  90000   44E3
R Registers:     FFFF    B29     3A    593    598      0   A9FF
M Registers:     FF00   FF00   FF00   FF00   FF00   FF00   FF00

ASV$: .503E      TSAP$: .0          NATSAP: 1  I: .0002  T$: .45FA

CI: .0000  RDBR$: .0

TSA @.50B6:

  MCL - Monitor Service Request.
TRAP: 63  TSAL$: .0          I: .3F04  INST: .0001  Z: .8000

A$: .2004F      P: .20050     B3: .10288     R3: .902    S: .003F

Trap IC: GFM$MCL+.18
```

```
ISA:

P: .20050     S: .0000  CHN: .0000  ISM1: .FFFF  ISM2: .8103

Interrupted address: GFM$MCL+.18

                      1       2       3       4       5       6       7
                  ------ ------ ------ ------ ------ ------ ------
B Registers:          0       0   10288       0       0       0   A0010
R Registers:          0       0     902       0       0       0       0
M Registers:       FF00       0       0       0       0       0       0

ASV$: .0          TSAP$: .0            NATSAP: 0  I: .0000  T$: .A00FA

CI: .0000  RDBR$: .0


L6XII     -DUA .10288,8 F CUN
         0    1    2    3    4    5    6    7
010288  0003 0001 028E 0005 0001 0298 0100 4000   ..............@.
L6XII     -DUA .10298,6 F CUN
         0    1    2    3    4    5    6    7
010298  000C 0000 0000 0001 0001 029E             ............
L6XII     -
L6XII     -SPY CUN
 Usr#     Identification          Sysid     CPU     M$LM
G  .5 LLAHOST,104AVERY              11     0:00     QGDS.LLAHOST


L6XII     -JIT CUN
JIT for User# .05 Sysid 11  Mode: Ghost  Prog entry: M$SETFP  LLAHOST,104AVERY

DLL .0100  DUL .0102  PLL .0200  PUL .0201  LLL .0600  LUL .07FF  MAXMEM 511

PCD 3   PCP 2   PCL 0   PCDS(non-IO) 1  PCDS(IO) 0
PCC 11  PCROS 1  PCCQ 0  PCHHJIT 0  PCDDS 0

PRIV.ACTIVE: .00000000   AUTH: .00300C4F  PRC: .00000000   PPRIV .00000000

MCLs: 6  Steps: 1   CVM_REAL: .00000000  SPROC: 5  SLIB  2  DB: 0

RNST .0000  FRS .0000  XLIMFLG .00 STEPCC .00 RUNF .04   JUNK .0000

Interrupts: MAX 5  CURR 0  SPEAK 0  JPEAK 0  Rtime Clk 0  Xtime Clk 0

JIT.ERR and JIT_FD_ALTERR
    -00000-0
    -00000-0

L6XII     -TCB CUN
TCB for User# .05    ALT$ .012C9   STK$ .012F7   AVSZ .008A   CURRSZ .0000

Altret frame at .012C9  ECC: .0063  PREVSZ: .0000
```

```
P: .74B56     S: .003F  ISM1: .FFFF  ISM2: .8103


                   1      2      3      4      5      6      7
             ------ ------ ------ ------ ------ ------ ------
B Registers:      0      0  74C61      0      0      0  2001D
R Registers:      0      0    905      0      0      0      0
M Registers:      0   FF00      0      0      0      0      0

I: .3F00  T$: .0            CI: .45FA  RDBR$: .0
SUBC: .0905  EVID: .0000  ERR: .3B5B1318  P#: .0000

 GMM-M00611-0  That page does not belong to you.


L6XII    -ECCB CUN
ECCB for User# .05:
EVENT$ .00000  INT$ .00000  XCON$ .00000  TRAP$ .00000  DBCONTROL$ .00000
FLAGS: .0000  FLTFLG: .8000 .0000 .0000


L6XII    -DCBS CUN
DCBs for User# .05
  # DCB$    FCD ORG RES  NODE GEN LDCTX STRM DDEV HMI$   SSN$   DCBNAME
--- ------ --- --- ---- ---- --- ----- ---- ---- ------ ------ -------------
.01 .011A1 CLS 0   HO   .00  .00 .0000  .00  .00 .00000 .00000 M$DEBUG
.02 .011D7 OPN 0   HO   .14  .35 .008F  .01  .00 .00000 .C1CE4 M$LM
       Name: QGDS.LLAHOST
.03 .01215 CLS 0   UC01 .00  .00 .0000  .00  .00 .00000 .00000 M$ME
.04 .0124A OPN 0   HO   .14  .35 .008F  .01  .00 .00000 .C1CE4 M$HOST
.05 .01280 CLS 0        .00  .00 .0000  .00  .00 .00000 .00000 M$DO


L6XII    -DU .5000,.3E F CUN
           0    1    2    3    4    5    6    7
005000  0000 FC00 85AD 2C03 0000 5C00 8000 AC0F
005008  85A7 AC05 85A5 5C01 8009 AC06 0000 AC00
005010  8010 AC07 8018 B80F 0000 0C00 0000 0C00
005018  0000 5C00 0000 FC00 0000 FC00 0000 FC00
005020  85B1 0C02 85B4 3001 0000 1000 0000 1000
005028  0000 1000 81EA 30FF 82EA 3055 0000 6400
005030  0000 6400 85B6 0C00 0000 0C00 0000 0C00
005038  0000 AC00 0000 FC00 8700 A8FF
L6XII    -DU .503E,.3E F CUN
           0    1    2    3    4    5    6    7
005038                                0000 FC00
005040  85AD 2C03 8175 AC07 8000 AC0F 85A7 AC05
005048  85A5 5C01 8009 AC06 0000 AC00 8010 AC07
005050  8018 B80F 85B3 0C00 85B3 0C00 0000 FC00
005058  0000 FC00 0000 FC00 0000 FC00 8028 AC05
005060  802E B8F8 0000 B800 0000 B800 0000 B800
005068  85B1 0C02 8594 A8FF 8127 B84A 858B A9FF
005070  0000 FC00 0000 FC00 817D ACFF 0000 AC00
```

```
005078  0000 FC00 8700 A8FF
L6XII     -SPY
 Usr#     Identification                Sysid      CPU       M$LM
H  .1 :SYS,NODEADMN                        1       2:17      NODEADMN.:SYS
H  .2 :SYS,HDLCX25                         2       5:34      HDLCX25.:SYS
H  .3 :SYS,ASYNC                           3       9:51      ASYNC.:SYS
G  .4 LLAHOST,104AVERY                     8       0:00      ANLZ_FPRG.:SYS
G  .5 LLAHOST,104AVERY                    11       0:00      QGDS.LLAHOST


L6XII     -USRT
 #  FL    BL    Sysid   Mode   State Flags   Async  DL$     MISC        HMI$
--- ---   ---   -----   ----   ----- -----   -----  ------ ---------   ------
.01 .00   .03       1     H    SW     .0000   .0000  .00000 .00008E4F  .C1B86
.02 .03   .00       2     H    SW     .0000   .0000  .00000 .00008A06  .C1B9F
.03 .01   .02       3     H    SW     .0000   .0000  .00000 .00008E07  .C1BB8
.04 .00   .00       8     G    SCI    .0800   .0000  .00000 .000C1542  .00000
.05 .00   .00      11     G    SCU    .0000   .0000  .00000 .000C16D2  .00000

 #  Sysid   HJIT  MF  Clock  UTS          Prio   PrioB   USRT$
--- -----  -----  --- -----  ----------- -----   -----   ------
.01     1  .01C8    0 .0000  780101 0000     4       4   .C1058
.02     2  .01D3    0 .0000  780101 0000     4       4   .C1070
.03     3  .01DD    0 .0000  780101 0000     4       4   .C1088
.04     8  .058B    0 .0000  780101 0000     1       1   .C10A0
.05    11  .05A5    0 .0000  780101 0000     1       1   .C10B8



L6XII     -SFILE
 #  File Identification                    Flags Modtime     Instime     UC  FRQ
--- ---------------------------------- ----- ----------- ----------- --- ---
  1 DELTA_F.:SYS                          .0800 780101 0000 780101 0000   1   1
  2 :SHARED_LCP6_SYSTEM.:SYS              .4000 780101 0000 780101 0000   4   8
  4 ANLZ_FPRG.:SYS                        .A080 781016 1214 850806 1223   1   3
  5 QGDS.LLAHOST                          .8080 781017 0101 850806 1321   1   1

 #   PGs ROS DATA   PROC   PP# ROS    SEG1  SEG2  SEG3  SEG4  SEG5  SEG6  SEG7
---      --- -----  -----      -----  ----- ----- ----- ----- ----- ----- -----
  1     .000 .0000 .004B      .0000 .0127 .0000 .0000 .0000 .0000 .0000 .0000
  2     .000 .0002 .0156      .0000 .01E9 .0000 .0000 .0000 .0000 .01EB .02EB
  4     .000 .0000 .000A      .0000 .0000 .059A .0000 .0000 .0000 .0000 .0000
  5     .000 .0000 .0002      .0000 .0000 .05B4 .0000 .0000 .0000 .0000 .0000

L6XII     -INTCON
 #  TIMER   TSAP$   P$      IENTRY   DEV  S    HHJIT USR LVL FLAGS ISM1 ISM2
--- ------ ------ ------ -------- ---- ---- ----- --- --- ----- ---- ----
.20 .0000  .00000 .094FA 00000000 0020 4000 .0000 .03 .00 .8400 0000 2000
```

```
 .21 .17E3   .00000 .094FA 00020A5C 0000 4004 .81DB .03 .20 .C000 0000 2000
 .22 .0000   .00000 .094FA 00021432 0000 4004 .81DC .03 .20 .C000 0000 2000
 .24 .0000   .00000 .094FA 00000000 0024 4000 .0000 .02 .00 .8400 0000 2000
 .25 .17E3   .00000 .094FA 00021920 0000 4004 .81E6 .02 .24 .C000 0000 2000
 .26 .0000   .00000 .094FA 00021920 0000 4004 .81E7 .02 .24 .C000 0000 2000
```

```
L6XII     -MEMORY
                                  FEP Memory Utilization

MONITOR:BIGFOOT                                 0  .0000
MONITOR:Procedure and Static Data             279  .0117
MONITOR:Cntx-TSTKM.UMHJIT,MHJIT,ROS,LRM        19  .0013
MONITOR:BOBCAT                                 75  .004B
MONITOR:DELTA-Procedure and Dyn Data Seg       83  .0053
MONITOR:Total Dedicated Memory..........      456  .01C8


                                            USER   COMGP  HAND GHOST
FPRGS   :User Procedure                        0       0   288     0
FPRGS   :User Data-Static and Dynamic          0       0   495     7
FPRGS   :Context-TSTACKM,ROS,UHJIT,HHJITs      0       0    40    24
FPRGS   :Handler Q                             0       0    99     0
FPRGS   :DELTA-Dynamic Data                    0       0     0     0
FPRGS   :Individual Totals...............      0       0   922    31
FPRGS   :Individual Totals............... .0000   .0000 .039A .001F


TOTAL   :Memory Available on L6..........   2048  .0800

TOTAL   :Memory Dedicated by LCP-6           456  .01C8
TOTAL   :Available for FPRGs and SFILEs     1592  .0638

TOTAL   :System Shuffable Memory             282  .011A
TOTAL   :System Non-shuffable Memory         380  .017C

TOTAL   :Memory used by FPRGs                953  .03B9
TOTAL   :Memory for SFILE Procedure          354  .0162
TOTAL   :Memory for SFILE Data                 2  .0002
TOTAL   :Memory Free - Shuffable               0  .0000
TOTAL   :Memory Free - Non-shuffable          77  .004D
TOTAL   :Memory Unaccounted For              206  .00CE
```

```
L6XII     -STAT
                                       Max Phys Pro Data Dyn Dyn Con  Ro Dbg  Cq HHJT
M Identification                 Sysid  Pgs  Pgs Pgs  Pgs Pgs I/O Pgs Pgs Pgs Pgs  Pgs
H :SYS,NODEADMN                      1 65535  181  68    8  60   0  11   1   0  33    0
H :SYS.HDLCX25                       2 65535  370 167   19  89  48  11   1   0  33    2
H :SYS,ASYNC                         3 65535  371  53    7   9 255  11   1   0  33    2
G LLAHOST,104AVERY                   8   511   15 10S    2   1   0  11   1   0   0    0
G LLAHOST,104AVERY                  11   511   16  2S    3   1   0  11   1   0   0    0
```

```
L6XII     -USRT
 #  FL   BL    Sysid  Mode  State Flags  Async  DL$    MISC       HMI$
--- ---  ---   -----  ----  ----- -----  -----  ------ ---------  ------
.01 .00  .03       1    H   SW    .0000  .0000  .00000 .00008E4F  .C1B86
.02 .03  .00       2    H   SW    .0000  .0000  .00000 .00008A06  .C1B9F
.03 .01  .02       3    H   SW    .0000  .0000  .00000 .00008E07  .C1BB8
.04 .00  .00       8    G   SCI   .0800  .0000  .00000 .000C1542  .00000
.05 .00  .00      11    G   SCU   .0000  .0000  .00000 .000C16D2  .00000

 #  Sysid   HJIT   MF  Clock  UTS               Prio   PrioB   USRT$
--- -----  -----  --- -----  ----------- -----  -----  ------
.01     1  .01C8    0 .0000  780101 0000     4      4   .C1058
.02     2  .01D3    0 .0000  780101 0000     4      4   .C1070
.03     3  .01DD    0 .0000  780101 0000     4      4   .C1088
.04     8  .058B    0 .0000  780101 0000     1      1   .C10A0
.05    11  .05A5    0 .0000  780101 0000     1      1   .C10B8

L6XII     -PPUT
PPUT$: .C0040   MEM_LO: .01C8   MEM_HI: .07FF
PPUT Entry - first page .593 : 6800 0006  last page .598 : A804 0002


L6XII     -EV .593
Dec: 1427  Oct: 2623  Hex: 593  Entdef: GD_DATA_D+.593
L6XII     -PL .4000 F CUN
CW=.1AA  MW=.5F0

Frame @.4450 called from .22335 (GID$WRDUMP+.35)
Frame @.4489 called from .22A81 (GIR$INITDUMP+.177)
Frame @.44A7 called from .22EB6 (GIR$SCREECH+.142)
Frame @.44C2 called from .91DA (GHT$TRAP+.1D9)
Frame @.44DB called from .90EB (GHT$TRAP+.EA)
Frame @.44E2 called from .26B19 (GMM$MCLS+.135)
Frame @.4534 called from .9A26 (GUD$MCL_HAND+.2E4)
Frame @.4561 called from .9238 (GHT$TRAP+.237)
Frame @.45EE called from .9746 (GUD$MCL_HAND+.4)
Frame @.45F5 called from .21386 (GHS$ADDUSR+.196)
L6XII     -DUA .4450,.4000 F CUN
            0    1    2    3    4    5    6    7
004450    0038 0002 2338 0000 0000 0000 449E 0000   .8..#8......D...
004458    4496 0000 4494 0002 235B 0498 8100 04C9   D...D...#[.....I
004460    000E 0000 0000 0800 0006 0004 003C 0000   ..............‹..
004468    0000 0800 0800 000E 0000 1184 0001 0012   ................
004470    0000 0000 4451 0002 244C 0000 4451 0002   ....DQ..$L..DQ..
004478    2476 0002 244E 0002 2561 0002 2584 0000   $v..$N..%a..%...
004480    445D 0000 4466 0000 4465 0000 446D 0000   D]..Df..De..Dm..
004488    446E 001D 0002 2A84 0000 0000 0000 44B4   Dn....*.......D4
004490    0000 44B3 0000 44B5 000E 0000 0000 0800   ..D3..D5........
004498    0002 2316 0000 44B4 0000 4494 1184 0000   ..#...D4..D.....
0044A0    449E 0000 4496 0000 4494 0002 235B 001A   D...D...D...#[..
0044A8    0002 2EB9 0000 0000 0000 3200 0000 0002   ...9......2.....
0044B0    0002 2961 016A 08C0 05A8 0004 013E 403F   ..)a.j.@.(...›@?
```

```
0044B8   0000 44B4 0000 44B3 0000 44B5 000C 3D6C   ..D4..D3..D5..=l
0044C0   0000 44BC 0018 0000 91DD 0000 0000 0000   ..D‹.....].......
0044C8   50FF 0000 3200 0593 0592 07FF 01C8 000C   P...2........H..
0044D0   0B66 0002 2EA6 0000 9D30 005C 0001 001F   .f...&...0.\....
0044D8   0003 0000 6000 0006 0000 90EE 0000 0002   ....`......n....
0044E0   0000 50FF 0051 0002 6B1C 0000 03FA 0000   ..P..Q..k....z..
0044E8   4551 0000 4552 0002 6FCE 0002 6FCE 0002   EQ..ER..oN..oN..
0044F0   6FCF 0000 0593 000C 0BAE 0000 0000 8115   oO..............
0044F8   0000 0000 0C00 0010 0007 058B 0000 8145   ...............E
            0    1    2    3    4    5    6    7
004500   0594 000C 0B68 000C 1F70 000C 00CE 0001   .....h...p...N..
004508   000F 0009 0000 0009 003E 0007 003E 0002   .........›...›..
004510   5B0A 0000 8177 8594 A8FF 000E 0002 001C   [....w..(.......
004518   0002 0002 5E2A 0000 0004 0004 001D 458D   ....~*........E.
004520   0000 454B 0000 453B 0000 454A 0000 0002   ..EK..E;..EJ....
004528   639E 0002 5F8D 0000 44FD 0000 9D10 0000   c..._...D}......
004530   506E 452D 0000 44F2 002C 0000 9A29 0000   PnE-..Dr.,....)..
004538   00F9 0000 458A 0000 0000 0C00 0008 0000   .y..E...........
004540   0008 0002 0000 10B8 A068 0000 458A 000C   .......8 h..E...
004548   10B8 0000 8000 0000 0000 A08E 0000 B098   .8......... ...0.
004550   0008 001B 00CE 0002 6A4F 0000 4551 0000   .....N..jO..EQ..
004558   4552 0002 6FCE 0002 6FCE 0002 6FCF 0000   ER..oN..oN..oO..
004560   453F 008C 0000 923B 0000 0000 0000 50B6   E?.....;......P6
004568   0000 8292 2F9E 0000 0108 0002 0002 69EB   ..../.........ik
004570   0000 503E 0001 028E 0000 4592 0002 0010   ..P›......E.....
004578   000B 0000 0005 0001 029B 0002 0000 0000   ................
004580   0001 0000 0000 0298 0000 0098 0002 000B   ................
004588   0001 0288 0902 0000 50B6 3B43 26A8 0003   ........P6;C&(..
004590   0000 A08E 0000 B098 0000 0000 0000 0000   .. ...0.........
004598   0000 0000 0000 0000 0000 0000 0000 0000   ................
0045A8*  0000 0000 0000 0001 003F 0000 0000 0000   .........?......
0045B0   0003 0005 0000 0000 0000 0000 0000 0000   ................
0045B8   0000 0000 0000 0000 0000 0000 0000 0000   ................
0045D8*  0000 0000 0007 4B58 0063 0905 0000 3B5B   ......KX.c....;[
0045E0   1318 0000 0000 0000 0000 0000 50B6 0000   ............P6..
0045E8   0000 000B 0000 9926 0000 458A 0006 0000   .......&..E.....
0045F0   9749 0000 0000 0000 50B6 0004 0002 1389   .I......P6......
0045F8   FFFF 0000 01AA 05F0 0000 0000 0000 0000   .....*.p........
```

```
17:06 AUG 29 '85 C04

DRIBBLE ON @  17:16 08/21/85
!ANLZ A090
ANLZ C01
 :DFC01A090 for GHH-001304-3 at 18:30 AUG 05 '85 on LADC L66B
 JL is to blame

 Nodes:       L6IX       L66B
L6IX (node 9) Selected
L6IX      -REC

Screech Code:

 GHH-M01304-0  Handler Abort - Disabled Too Long

Current User: .4                TSA$: .50B6

TSA @.50B6:

  Watch Dog Timer Runout
TRAP: 4  TSAL$: .0          I: .041A  INST: .0001  Z: .0080

A$: .0          P: .6ED4D     B3: .A0250     R3: .FFFF  S: .4020


ISA:

P: .6ED4D      S: .0000  CHN: .FFFF  ISM1: .FFFF  ISM2: .A103


                    1       2       3       4       5       6       7
                 ------  ------  ------  ------  ------  ------  ------
B Registers:     323A0   323A0   A0250   32B00   33AE0   33A00   A02B7
R Registers:         4      16    FFFF      50       4       3       1
M Registers:      FF00    FF00    FF00    FF00    FF00    FF00    FF00

ASV$: .5000      TSAP$: .0         NATSAP: 2  I: .001A  T$: .A07FA

CI: .0002  RDBR$: .0
L6IX      -SPY CUN
 Usr#    Identification          Sysid     CPU      M$LM
H  .3 :SYS,ASYNC                    3    56:40      ASYNC.:SYS


L6IX      -INTCON
 #  TIMER  TSAP$   P$      IENTRY    DEV  S     HHJIT USR LVL FLAGS ISM1 ISM2
--- ------ ------ ------ -------- ---- ----  ----- --- --- ----- ---- ----
.1A .0000  .00000 .094FA 00000000 001A 4000  .0000 .04 .00 .8400 0000 2000
.1B .1801  .00000 .094FA 000246AA 0000 4004  .8202 .04 .1A .C000 0000 2000
.1C .0000  .00000 .094FA 000247B4 0000 4004  .8203 .04 .1A .C000 0000 2000
.1D .0000  .00000 .094FA 000248C2 0000 4004  .81F6 .04 .1A .C000 0000 2000
.20 .0000  .00000 .094FA 00000000 0020 4000  .0000 .03 .00 .8400 0000 2000
.21 .1802  .00000 .094FA 00020A5C 0000 4004  .820E .03 .20 .C800 0000 2000
```

```
.22 .0000   .00000 .094FA 00021432 0000 4004 .8201 .03 .20 .C000 0000 2000
.24 .0000   .00000 .094FA 00000000 0024 4000 .0000 .05 .00 .8400 0000 2000
.25 .1802   .00000 .094FA 00021920 0000 4004 .81F7 .05 .24 .C000 0000 2000
.26 .0000   .00000 .094FA 00021920 0000 4004 .81F8 .05 .24 .C000 0000 2000
.32 .0000   .00000 .094FA 00000000 0032 4000 .0000 .02 .00 .8400 0000 2000
.33 .172B   .00000 .094FA 00020A9A 0000 4004 .86CC .02 .32 .C000 0000 2000
.34 .0000   .00000 .094FA 00000000 0034 4000 .0000 .06 .00 .8400 0000 2000
.35 .1801   .00000 .094FA 00020A9A 0000 4004 .86EE .06 .34 .C000 0000 2000


L6IX      -DU .3020,4
           0    1    2    3    4    5    6    7
003020  0800 0000 C000 0005
L6IX      -DU .507E,1
           0    1    2    3    4    5    6    7
005078                                0002
L6IX      -JIT CUN
JIT for User# .03 Sysid 3  Mode: Handler  Prog entry: Bigfoot  :SYS,ASYNC

DLL .0100  DUL .0106  PLL .0200  PUL .0234  LLL .0600  LUL .07FF  MAXMEM 65535

PCD 7   PCP 53   PCL 0   PCDS(non-IO) 9  PCDS(IO) 192
PCC 11  PCROS 1  PCCQ 33  PCHHJIT 2  PCDDS 0

PRIV.ACTIVE: .00300C08   AUTH: .FFFFFFFF  PRC: .00000000    PPRIV .00000000

MCLs: 58664  Steps: 1   CVM_REAL: .00000010  SPROC: 0  SLIB  2  DB: 0

RNST .0000  FRS .0000  XLIMFLG .00 STEPCC .00 RUNF .04   JUNK .0008

Interrupts: MAX 10  CURR 4  SPEAK 4  JPEAK 0  Rtime Clk 0  Xtime Clk 0

JIT.ERR and JIT_FD_ALTERR
    -00000-0
    -00000-0

L6IX      -SFILE 2
 #  File Identification                Flags Modtime     Instime     UC  FRQ
--- ------------------------------- ----- ----------- ----------- --- ---
  2 :SHARED_LCP6_SYSTEM.:SYS         .4000 780101 0000 780101 0000   6  10


 #  PGs ROS DATA  PROC  PP# ROS   SEG1  SEG2  SEG3  SEG4  SEG5  SEG6  SEG7
---     --- ----- -----     ----- ----- ----- ----- ----- ----- ----- -----
  2    .000 .0002 .018D     .0000 .021A .0000 .0000 .0000 .0000 .021C .031C


L6IX      -SYM :SHARED_LCP6_SYSTEM.:SYS
L6IX      -REC

Screech Code:

 GHH-M01304-0  Handler Abort - Disabled Too Long
```

```
Current User: .4              TSA$: .50B6

TSA @.50B6:

  Watch Dog Timer Runout
TRAP: 4  TSAL$: .0           I: .041A  INST: .0001  Z: .0080

A$: .0          P: .6ED4D     B3: .A0250     R3: .FFFF  S: .4020

Trap IC: KVO$GNRPST+.191

ISA:

P: .6ED4D     S: .0000  CHN: .FFFF  ISM1: .FFFF  ISM2: .A103

Interrupted address: KVO$GNRPST+.191

                    1      2      3      4      5      6      7
                 ------ ------ ------ ------ ------ ------ ------
B Registers:     323A0  323A0  A0250  32B00  33AE0  33A00  A02B7
R Registers:         4     16   FFFF     50      4      3      1
M Registers:      FF00   FF00   FF00   FF00   FF00   FF00   FF00

ASV$: .5000      TSAP$: .0          NATSAP: 2  I: .001A  T$: .A07FA

CI: .0002  RDBR$: .0
L6IX      -PL .A0000 F .3
CW=.544  MW=.7ED

Frame @.A02B6 called from .7481B (KVV$VDI_+.8D7)
Frame @.A0273 called from .6AC07 (KVI$INT+.1915)
Frame @.A0234 called from .68DD9 (KVI$INPCHR+.931)
Frame @.A0210 called from .745E2 (KVV$VDI_+.69E)
Frame @.A01CD called from .20CB6
Frame @.A01AE called from .20AFC
Frame @.A0192 called from .20A5C
Frame @.A0173 called from .6BD53 (KVM$RCV_+.323)
Frame @.A001C called from .21B15
Frame @.A000F called from .21A28
L6IX      -SYM ASYNC.:SYS
L6IX      -PL .A0000 F .3
CW=.544  MW=.7ED

Frame @.A02B6 called from .7481B (X6A_MAUTO+.14717)
Frame @.A0273 called from .6AC07 (X6A_MAUTO+.AB03)
Frame @.A0234 called from .68DD9 (X6A_MAUTO+.8CD5)
Frame @.A0210 called from .745E2 (X6A_MAUTO+.144DE)
Frame @.A01CD called from .20CB6 (KAI$INPINT+.10C)
Frame @.A01AE called from .20AFC (KAI$INP+.A0)
Frame @.A0192 called from .20A5C (KAI$INP+.0)
Frame @.A0173 called from .6BD53 (X6A_MAUTO+.BC4F)
Frame @.A001C called from .21B15 (KAS$BOT+.ED)
Frame @.A000F called from .21A28 (KAS$BOT+.0)
```

17:06 AUG 29 '85 C05.

LCP-6 Trap Numbers

E$TRAP=1300   E$TRAP + G$TSA.I.TRAP# yields the following:

Err	TSA		
13xx	X'xx'	E$ or %G_	:ERRMSG file
01	01	E$MCL	MCL
02	02	E$TRACE	Trace Breakpoint Trap
03	03	E$NO_SIP	Uninstalled SIP Trap
04	04	E$TROT	Watch Dog Timer Runout
05	05	E$UNIMPL	Unimplemented Instruction Trap
06	06	E$INT_REG_OV	Integer Register Overflow Trap
07	07	E$S_DBZ	SIP Divide by Zero Trap
08	08	E$S_EXP_OV	SIP Exponent Overflow Trap
09	09	E$STK_UF	Stack Underflow Trap
10	0A	E$STK_OV	Stack Overflow Trap
12	0C	E$REMOTE_DESC	Remote Data Descriptor Trap
13	0D	E$PRIV	Privilege Violation Trap
14	0E	E$MEM_PROT	Memory Protection Trap
15	0F	E$INT_UR	Internal Unavailable Resource Trap
16	10	E$PROG_ERR	Program Error Trap
17	11	E$INT_MBE	Internal Memory or Bus Error Trap
19	13	E$S_EXP_UF	SIP Exponent Underflow Trap
20	14	E$S_PROG_ERR	SIP Program Error Trap
21	15	E$S_SIGNIF	SIP Significance Error Trap
22	16	E$S_PRECISION	SIP Precision Error Trap
23	17	E$EXT_UR	External (CIP or SIP) Unavailable R
24	18	E$EXT_MBE	External (CIP or SIP) Memory or Bus
25	19	E$C_DBZ	CIP Divide by Zero Trap
26	1A	E$C_SPEC	CIP Illegal Specification Trap
27	1B	E$C_CHAR	CIP Illegal Character Trap
28	1C	E$C_TRUNC	CIP Truncation Trap
29	1D	E$C_OV	CIP Overflow Trap
30	1E	E$CIP_QLT	CIP QLT Fault
31	1F	E$SIP_QLT	SIP QLT Fault

1347 - 1362   used to communicate to host debugger.
1347 - 1354   reserved to reflect GJ_LCP6_M FPT code.

47	2F	E$FPRG_EXIT	FPRG M$EXIT.
48	30	E$FPRG_ERR	FPRG M$ERR.
49	31	E$FPRG_XXX	FPRG M$XXX.
50	32	E$FPRG_LDTRC	FPRG requesting M$LDTRC.
51	33	E$FPRG_XBREAK	FPRG (FPL interpreter) M$XBREAK.
52	34	E$FPRG_SCREECH	FPRG M$SCREECH
55	37	E$HOST_INT	FPRG interrupted by Host Debugger.
56	38	E$FPRG_EVENT	FPRG event condition.
57	39	E$FPRG_BRK	FPRG break condition.
58	3A	E$FPRG_XCON	FPRG exit/abort condition.
59	3B	E$FPRG_MCLTRAP	FPRG bad MCL - Entry to trap handler.

60	3C	E$DB_TIMER	Debugger Timer Runout event
61	3D	E$FPRG_START	FPRG started via M$SETFP.
62	3E	E$FPRG_START2	FPRG started via M$LDTRC.
63	3F	E$MCL2	MCL

```
17:06 AUG 29 '85 C06

Monitor service codes  - FPT specified

      Sorted by $R3 value                             Sorted by M$name

  Service Name       Code  Vect#                  Service Name     Code  Vect#

  M$INTRET           0440   01                    M$AFD            04D1   01
  M$INTCON           0441   01                    M$CHGUNIT        04D5   01
  M$INTREL           0442   01                    M$CLOCK          08C6   02
  M$SYS              0443   01                    M$CLOSE          1541   05
  M$XBREAK           0484   01                    M$CLRSTK         04CD   01
  M$SCREECH          0485   01                    M$CPEXIT         1486   05
  M$MAKEUSR          04C7   01                    M$CVM            0D08   03
  M$MERC             04C9   01                    M$DBCONTROL      08C4   02
  M$RETRY            04CA   01                    M$DCLFLD         0D8F   03
  M$CLRSTK           04CD   01                    M$DEVICE         0D47   03
  M$MERCS            04CE   01                    M$DRTN           04D2   01
  M$RETRYS           04CF   01                    M$EOM            0983   02
  M$AFD              04D1   01                    M$ERASE          0992   02
  M$DRTN             04D2   01                    M$ERR            0881   02
  M$UNSHARE          04D3   01                    M$ERRMSG         2CD6   11
  M$CHGUNIT          04D5   01                    M$EVENT          08C0   02
  M$RUE              04D7   01                    M$EXIT           0880   02
  M$GDDL             0506   01                    M$FAUTO          0905   02
  M$RELDCB           0546   01                    M$FDS            0903   02
  M$TRMPRG           0584   01                    M$GAUTO          0904   02
  M$RCHAN            0594   01                    M$GCHAN          0993   02
  M$EXIT             0880   02                    M$GDDL           0506   01
  M$ERR              0881   02                    M$GDS            0902   02
  M$XXX              0882   02                    M$GETDCB         0D45   03
  M$EVENT            08C0   02                    M$GLINEATTR      0985   02
  M$INT              08C1   02                    M$GPROMPT        0995   02
  M$XCON             08C2   02                    M$GTRMATTR       0987   02
  M$TRAP             08C3   02                    M$GTRMCTL        0981   02
  M$DBCONTROL        08C4   02                    M$INT            08C1   02
  M$CLOCK            08C6   02                    M$INTCON         0441   01
  M$TRTN             08CC   02                    M$INTREL         0442   01
  M$GDS              0902   02                    M$INTRET         0440   01
  M$FDS              0903   02                    M$LDTRC          1483   05
  M$GAUTO            0904   02                    M$MAKEUSR        04C7   01
  M$FAUTO            0905   02                    M$MDFFLD         118E   04
  M$PDS              0907   02                    M$MERC           04C9   01
  M$STRMCTL          0980   02                    M$MERCS          04CE   01
  M$GTRMCTL          0981   02                    M$OPEN           3540   13
  M$PROMPT           0982   02                    M$PDS            0907   02
  M$EOM              0983   02                    M$PLATEN         0988   02
  M$GLINEATTR        0985   02                    M$PROMPT         0982   02
  M$STRMATTR         0986   02                    M$RCHAN          0594   01
  M$GTRMATTR         0987   02                    M$READ           1542   05
  M$PLATEN           0988   02                    M$RELDCB         0546   01
  M$STRMTAB          098A   02                    M$RETRY          04CA   01
  M$SINPUT           098B   02                    M$RETRYS         04CF   01
```

M$RLSFLD	0990	02	M$RLSFLD	0990	02
M$SLCFLD	0991	02	M$RUE	04D7	01
M$ERASE	0992	02	M$SCREECH	0485	01
M$GCHAN	0993	02	M$SINPUT	098B	02
M$GPROMPT	0995	02	M$SLCFLD	0991	02
M$WRSYSLOG	09C0	02	M$SPRIV	0CD0	03
M$WAIT	0CC5	03	M$STRMATTR	0986	02
M$SPRIV	0CD0	03	M$STRMCTL	0980	02
M$CVM	0D08	03	M$STRMTAB	098A	02
M$GETDCB	0D45	03	M$SYS	0443	01
M$DEVICE	0D47	03	M$TIME	14D4	05
M$DCLFLD	0D8F	03	M$TRAP	08C3	02
M$WRTMLT	1144	04	M$TRMPRG	0584	01
M$MDFFLD	118E	04	M$TRTN	08CC	02
M$LDTRC	1483	05	M$UNSHARE	04D3	01
M$CPEXIT	1486	05	M$WAIT	0CC5	03
M$TIME	14D4	05	M$WRITE	1543	05
M$CLOSE	1541	05	M$WRSYSLOG	09C0	02
M$READ	1542	05	M$WRTMLT	1144	04
M$WRITE	1543	05	M$XBREAK	0484	01
M$ERRMSG	2CD6	11	M$XCON	08C2	02
M$OPEN	3540	13	M$XXX	0882	02

Code is expressed in hexidecimal.
Number of vectors is expressed in decimal.

Monitor service codes - no FPT specified

1.000	M$INTRET	0040	M$GDS	0102
2.000	M$INTCON	0041	M$FDS	0103
3.000	M$INTREL	0042	M$GAUTO	0104
4.000	M$SYS	0043	M$FAUTO	0105
5.000			M$GDDL	0106
6.000	M$EXIT	0080	M$PDS	0107
7.000	M$ERR	0081	M$CVM	0108
8.000	M$XXX	0082		
9.000	M$LDTRC	0083	M$OPEN	0140
10.000	M$XBREAK	0084	M$CLOSE	0141
11.000	M$SCREECH	0085	M$READ	0142
12.000	M$CPEXIT	0086	M$WRITE	0143
13.000			M$WRTMLT	0144
14.000	M$EVENT	00C0	M$GETDCB	0145
15.000	M$INT	00C1	M$RELDCB	0146
16.000	M$XCON	00C2	M$DEVICE	0147
17.000	M$TRAP	00C3		
18.000	M$DBCONTROL	00C4	M$STRMCTL	0180
19.000	M$WAIT	00C5	M$GTRMCTL	0181
20.000	M$CLOCK	00C6	M$PROMPT	0182

21.000	M$MAKEUSR	00C7	M$EOM	0183
22.000	M$SENV	00C8	M$TRMPRG	0184
23.000	M$MERC	00C9	M$GLINEATTR	0185
24.000	M$RETRY	00CA	M$STRMATTR	0186
25.000	M$RENV	00CB	M$GTRMATTR	0187
26.000	M$TRTN	00CC	M$PLATEN	0188
27.000	M$CLRSTK	00CD	M$STRMTAB	018A
28.000	M$MERCS	00CE	M$SINPUT	018B
29.000	M$RETRYS	00CF	M$MDFFLD	018E
30.000	M$SPRIV	00D0	M$DCLFLD	018F
31.000	M$AFD	00D1	M$RLSFLD	0190
32.000	M$DRTN	00D2	M$SLCFLD	0191
33.000	M$UNSHARE	00D3	M$ERASE	0192
34.000	M$TIME	00D4	M$GCHAN	0193
35.000	M$CHGUNIT	00D5	M$RCHAN	0194
36.000	M$ERRMSG	00D6	M$GPROMPT	0195
37.000	M$RUE	00D7		
			M$WRSYSLOG	01C0

```
17:06 AUG 29 '85 C07


  Active Process ------>  USER/HANDLER     MON FOR USER     MONITOR
  MMU Image source ---->  UHJIT.ASDT_USR   UHJIT.ASDT_MCL   MHJIT.ASDT_MON
 #  UASDT  MASDT  VADDR
--  -----  -----  ------  ---------------  ---------------  --------------
00  .5000  .503E  .00000  NULLSEG          NULLSEG          NULLSEG
01  .5002  .5040  .01000  ROS              ROS              ROS
02  .5004  .5042  .02000  DB_DS            RDB_DS           RDB_DS
03  .5006  .5044  .03000  LOW_MEM          LOW_MEM          LOW_MEM
04  .5008  .5046  .04000  TSTACKU          TSTACKU          TSTACKM
05  .500A  .5048  .05000  UHJIT            UHJIT            UMHJIT
06  .500C  .504A  .06000  MHJIT            MHJIT            MHJIT
07  .500E  .504C  .07000  MHJIT            MHJIT            MHJIT
08  .5010  .504E  .08000  MON_ENTRY_DATA   MON_ENTRY_DATA   MON_ENTRY_DATA
09  .5012  .5050  .09000  MON_ENTRY        MON_ENTRY        MON_ENTRY
10  .5014  .5052  .0A000  USER_DS1         LPAR1            *
11  .5016  .5054  .0B000  USER_DS2         LPAR2            *
12  .5018  .5056  .0C000  CP_DS            LPAR3            *
13  .501A  .5058  .0D000  *                LPAR4            *
14  .501C  .505A  .0E000  *                LPAR5            *
15  .501E  .505C  .0F000  *                LPAR6            *

16  .5020  .505E  .10000  USER_IS1         MON_IS1          MON_IS1
17  .5022  .5060  .20000  USER_IS2         MON_IS2          MON_IS2
18  .5024  .5062  .30000  USER_IS3         MON_IS3          MON_IS3
19  .5026  .5064  .40000  USER_IS4         MON_IS4          MON_IS4
20  .5028  .5066  .50000  USER_IS5         MON_IS5          MON_IS5
21  .502A  .5068  .60000  USER_IS6 (LIB)   BPAR1            *
22  .502C  .506A  .70000  USER_IS7 (LIB)   BPAR2            *
23  .502E  .506C  .80000  DB_PROC          DB_PROC          DB_PROC
24  .5030  .506E  .90000  CP_PROC          WINDOW1          WINDOW1
25  .5032  .5070  .A0000  UAUTO_DS         *                *
26  .5034  .5072  .B0000  USER_DS3         *                *
27  .5036  .5074  .C0000  USER_DS4         BOBCAT           BOBCAT
28  .5038  .5076  .D0000  HAND_Q           HAND_Q           *
29  .503A  .5078  .E0000  *                *                *
30  .503C  .507A  .F0000  *                BIGFOOT          BIGFOOT
```

G$UHJIT

```
                              0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
                              I---------------I---------------I
                        .5000 !ASDT_USER                      !W
                              I - - - - - - - I - - - - - - - I
                               !                               !
                               I---------------I---------------I
                         .503E !ASDT_MCL                       !
                               I - - - - - - - I - - - - - - - I
                               !                               !
                               I---------------I---------------I
                         .5072 !ISA_USR                        !
                               I - - - - - - - I - - - - - - - I
                               !  .507E contains .DMN.ID       !
                               I---------------I---------------I
                         .50B6 !TSA_USR                        !
                               I - - - - - - - I - - - - - - - I
                              7!                               !
                               I---------------I---------------I
                               ~                               ~
                               ~                               ~
                               I---------------I---------------I
      CNCT_LVLS          .517F !@!@!@!@!@!@!@!@!@!@!@!@!@!@!@!@!
                               I---------------I---------------I
                               !@!@!@!@!@!@!@!@!@!@!@!@!@!@!@!@!
                               I---------------I---------------I
                               !@!@!@!@!@!@!@!@!@!@!@!@!@!@!@!@!
                               I---------------I---------------I
                               !@!@!@!@!@!@!@!@!@!@!@!@!@!@!@!@!
                               I---------------I---------------I
                         .5183 !SHRD_SEG       !. . . . . . . .!
                               I---------------I---------------I
                                ARRAY:'  20'O ENTRIES TOTAL.
                               I---------------I---------------I
                         .5193 !MMFLGS !. . . . . . . . . . . .!
                               I---------------I---------------I
                                ARRAY:'  20'O ENTRIES TOTAL.
                               I---------------I---------------I
```

The User Housekeeping Job Information Table (UHJIT) contains the data required by the hardware and the LCP-6 Schedular for running a user.

The G$UHJIT macro, which is contained in the G_HJIT_M include file, may be used to generate a structure defining the UHJIT. Because the G$UHJIT macro requires some of the macros contained in the GH_LCP6_M include file, GH_LCP6_M (or LCP_6) must also be specified as an include file in a compile unit that requires the G$UHJIT structure.

A user may reference the UHJIT through the pointer G$UHJIT$. This pointer is defined in the G_UPTRS_D object unit file.

The fields within the UHJIT of particular interest to the system programmer are:

ASDT_MCL - The MCL Address Space Descriptor Table contains the segment descriptors that are loaded into the Memory Management Unit while processing a user's monitor service request.

ASDT_USR - The User Address Space Descriptor Table contains the segment descriptors that are loaded into the Memory Management Unit for user program execution.

ISA_USR - The User Interrupt Save Area is accessed by the hardware on the occurrence of the user's interrupt level. Refer to the description of G$ISA.

TSA_USR - The User Trap Save Area is used by the hardware and the LCP-6 system to store the environment at the time of a user trap. Refer to the description of G$TSA.

UHJIT.DMN.ID - UBIN contains the DoMaiN IDentification which indicates which process is running; Monitor, Debugger, User, user Interrupt Level, or monitor service. The value will be one of the following EQUated values from the G_LCP6_E (or LCP_6) include file:

%G_DMN_MON_SVC	0	%G_DMN_MON	4
%G_DMN_DB_SVC	1	%G_DMN_DB	5
%G_DMN_INT_SVC	2	%G_DMN_INT	6
%G_DMN_USR_SVC	3	%G_DMN_USR	7

G$LOW_MEM

```
               0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
              I---------------I---------------I
             0!Contains the TSA overflow     !
              I logic used to generate the    I
              ! GHT-1396-3 Screech Dumps      !
              ~                               ~
              ~                               ~
              !                               !
             B!                               !
              I---------------I---------------I
             C!NATSAP2$    Handler Int Level  !
              I - - - - - - - I - - - - - - - I
             D!                               !
              I---------------I---------------I
             E!NATSAP1$    User Level         !
              I - - - - - - - I - - - - - - - I
             F!                               !
              I---------------I---------------I
            10!NATSAP0$    Monitor - Idle     !
              I - - - - - - - I - - - - - - - I
            11!                               !
              I---------------I---------------I
            12!*                              !
              I---------------I---------------I
            13!*                              !
              I---------------I---------------I
            14!RTC_INIT                       !
              I---------------I---------------I
            15!RTC_CURR                       !
              I---------------I---------------I
            16!RTC_LVL                        !
              I---------------I---------------I
            17!WDT_CURR                       !
              I---------------I---------------I
            18!*                              !
              I---------------I---------------I
               ARRAY:'   7'O ENTRIES TOTAL.
              I---------------I---------------I
            1F!MEM_ERR_CNT                    !
              I---------------I---------------I
LVL_ACT     20!     Levels 0 - 15             !
              I---------------I---------------I
LVL_ACT     21!     Levels 16 - 31            !
              I---------------I---------------I
LVL_ACT     22!     Levels 32 - 47            !
              I---------------I---------------I
LVL_ACT     23!     Levels 48 - 63            !
              I---------------I---------------I
```

The G$LOW_MEM macro may be used to generate a structure of the hardware dedicated memory locations in low core. No initial values may be specified.

This structure contains the following fields:

IV = The interrupt vector. Refer to G$INTERRUPT_VECTOR for the fields within the interrupt vector.

LVL_ACT = ARRAY(0:63)-BIT(1) set when the corresponding interrupt level is active.

MEM_ERR_CNT = VALUE-SBIN WORD contains the hardware count of memory errors.

NATSAPn$ = PTR (where n = 0-7) contains the address of a pool of available Trap Save Areas. When a trap occurs, the firmware uses the NATSAP_SEL field in ISM2 to access one of the pools 0-7. The linkage between TSAs in a pool is initialized by the LCP-6 software and maintained by the firmware.

NATSAP0$ will be set to the address of MHJIT.TSA_MON and NATSAP1$ will be set to the address of UHJIT.TSA_USR.

NATSAP4$ through NAPTSAP7$ are Reserved for Future Use.

PSF_ENT$ = PTR is an entry to the Power Failsafe Routine. This is the address to be entered on power-up.

RTC_CURR = VALUE-UBIN WORD contains the Real time clock current value.

RTC_INIT = VALUE-UBIN WORD contains the Real time clock initial value.

RTC_LVL = VALUE-UBIN WORD contains the Real time clock interrupt level.

TV = The trap vector. Refer to G$TRAP_VECTOR for the fields within the trap vector.

WDT_CURR = VALUE-UBIN WORD contains the Watchdog Timer current value.

G$STATUS_REG

```
                    0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
                    I---------------I---------------I
         QLT        0!@!. . . . . . . . . . . . . .!W
                    I---------------I---------------I
         RING       0!.!@  !*        !ID !LEVEL     !
                    I---------------I---------------I
```

The G$STATUS_REG macro may be used to generate a structure that contains the Status Register.

The fields of the Status Register are:

ID = VALUE-BIT(2) is the processor identification which is hard-wired and cannot be changed under program control. These 2 bits are used as the least significant bits of the 10-bit channel number for the processor itself, where the 8 high order bits are always ZERO. Ther default is '00'B.

LEVEL = VALUE-UBIN(6) contains the interrupt priority level on which the processor is currently executing. Level 0 is the highest priority level and 63 is the lowest. The default is 0.

LCP-6 Interrupt Level Assignments

Dec	Hex	Usage
00	00	Power Fail Safe
01	01	Watch Dog Timer
02	02	Trap Save Overflow
03	03	Bigfoot - Used during the boot process
04	04	Monitor Inhibit Level
05	05	Monitor Inhibit Level2
06	06	XDELTA
-	-	
0A	0A	XDELTA
12	0C	Available for Handlers
-	-	
59	3B	Available for Handlers
60	3C	Common Interrupt Cleanup
61	3D	Real Time Clock
62	3E	Scheduler - Idle Level
63	3F	User Execution

QLT = VALUE-BIT(1) indicates whether a unit in the system has successfully completed its Quality Logic Test or not.

0 = QLT successfully completed.
1 = QLT either still running or failed

Default = '0'B.

RING = VALUE-BIT(2) contains the currently active Ring value. Note that ring values are always encoded as ones complement whenever they appear. Thus this field will contain one of the following:

11 = Ring 0 - the most privileged; used by the Monitor
10 = Ring 1 - also privileged and used by the Monitor
01 = Ring 2 - Command Processor or Debugger
00 = Ring 3 - User

This field may be initialized by specifying {G_RING0# | G_R_MON# | G_R_CPDB# | G_R_USR#}. The default is G_R_USR#.

```
17:07 AUG 29 '85 CXX

DRIBBLE ON @  12:19 08/22/85
!ANLZ. 5060
ANLZ C01JTA
 :DFC005060 for GHB-000001-3 at 19:19 AUG 20 '85 on LADC L66A
 LA is to blame

 Nodes:        L6I      L66A
L6I (node 1) Selected
L6I       -REC

Screech Code:

 GHB-M00001-6   Invalid domain number in UHJIT.

Current User: .0               TSA$: .6087

TSA @.6087:

TRAP: 0  TSAL$: .50B6       I: .000C  INST: .000A  Z: .00C0

A$: .20674     P: .20707     B3: .2077B     R3: .1     S: .6032

Trap IC: GHB$TRAP1+.8F

ISA:

P: .9026      S: .6004  CHN: .FFFF  ISM1: .FFFF  ISM2: .A103

Interrupted address: GHT$TRAP+.25

                      1      2      3      4      5      6      7
                   ------ ------ ------ ------ ------ ------ ------
B Registers:      503E   50B6   5081  2077A   5000  20707   44F8
R Registers:        10      F   A103      2     32      2    996
M Registers:      FF00   FF00   FF00   FF00   FF00   FF00   FF0A

ASV$: .503E      TSAP$: .0          NATSAP: 2  I: .000C  T$: .45FA

CI: .0000  RDBR$: .0

TSA @.50B6:

TRAP: 15  TSAL$: .0          I: .0F18  INST: .D800  Z: .0083

A$: .200        P: .20996     B3: .A0052     R3: .15    S: .4032

Trap IC: GHM$MCLS+.116

ISA:

P: .9026      S: .0000  CHN: .FFFF  ISM1: .FFFF  ISM2: .A103
```

```
Interrupted address: GHT$TRAP+.25

                     1      2      3      4      5      6      7
                  ------ ------ ------ ------ ------ ------ ------
B Registers:      A005C  A0068   5081  20C8D  D0018  2087B  A0058
R Registers:         10      8   A103      2      1      3     1D
M Registers:       FF00   FF00   FF00   FF00   FF00   FF00   FF0A

ASV$: .5000      TSAP$: .0          NATSAP: 2  I: .0018  T$: .A09FA

CI: .0000  RDBR$: .0


L6I       -INTCON
 #  TIMER  TSAP$  P$      IENTRY   DEV  S     HHJIT USR LVL FLAGS ISM1 ISM2
--- ------ ------ ------ -------- ---- ----  ----- --- --- ----- ---- ----
.1A .0000  .00000 .094EA 00000000 001A 6000 .0000 .04 .00 .8400 0000 2000
.1B .1803  .00000 .094EA 000246E8 0000 6004 .8226 .04 .1A .C000 0000 2000
.1C .0000  .00000 .094EA 000247F2 0000 6004 .8219 .04 .1A .C000 0000 2000
.1D .0000  .00000 .094EA 00024900 0000 6004 .821A .04 .1A .C000 0000 2000
.1E .0000  .00000 .094EA 00000000 001E 6000 .0000 .06 .00 .8400 0000 2000
.1F .17FB  .00000 .094EA 00020C66 0000 6004 .820F .06 .1E .C000 0000 2000
.20 .0000  .00000 .094EA 00000000 0020 6000 .0000 .03 .00 .8400 0000 2000
.21 .1803  .00000 .094EA 00020A46 0000 6004 .8210 .03 .20 .C000 0000 2000
.22 .0000  .00000 .094EA 000211AC 0000 6004 .8203 .03 .20 .C000 0000 2000
.24 .0000  .00000 .094EA 00000000 0024 6000 .0000 .05 .00 .8400 0000 2000
.25 .17FB  .00000 .094EA 0002190C 0000 6004 .821B .05 .24 .C000 0000 2000
.26 .0000  .00000 .094EA 0002190C 0000 6004 .820E .05 .24 .C000 0000 2000
.32 .0000  .00000 .094EA 00000000 0032 6000 .0000 .02 .00 .8400 0000 2000
.33 .1793  .00000 .094EA 00020A1E 0000 6004 .8204 .02 .32 .C000 0000 2000


L6I       -PL .4000 F .2
CW=.123  MW=.5F0

Frame @.44D7 called from .91D8 (GHT$TRAP+.1D7)
Frame @.44F0 called from .90EB (GHT$TRAP+.EA)
Frame @.44F7 called from .91D8 (GHT$TRAP+.1D7)
Frame @.4514 called from .9121 (GHT$TRAP+.120)
Frame @.451B called from .2794B (GUS$CLOCK+.35B)
Frame @.452C called from .9A3B (GUD$MCL_HAND+.30D)
Frame @.4561 called from .922D (GHT$TRAP+.22C)
Frame @.45EE called from .9732 (GUD$MCL_HAND+.4)
Frame @.45F5 called from .2123C (GHS$ADDUSR+.17A)

L6I       -SPY .2
 Usr#    Identification         Sysid     CPU      M$LM
H  .2 :SYS,COUPLER                 2    15:17     COUPLER.:SYS


L6I       -DUA .3020,4
          0    1    2    3    4    5    6    7
003020  0800 0000 0000 2001                          ...... .
L6I       -DU .507E,1 F .2
```

```
           0    1    2    3    4    5   6    7
005078                                  0003
L6I       -DU .507E,1 ASDT .20400
           0    1    2    3    4    5   6    7
005078                                  0002



L6I       -SYM COUPLER.:SYS
L6I       -TSA .50B6 F .2
TRAP: 63  TSAL$: .0          I: .3F0C  INST: .0001  Z: .8080

A$: .61E45    P: .61E46    B3: .61A00     R3: .CC5    S: .403F

Trap IC: X6A_MAUTO+.1D48

ISA:

P: .9148      S: .4000  CHN: .0000  ISM1: .FFFF  ISM2: .9103


                      1      2      3      4      5      6      7
                 ------ ------ ------ ------ ------ ------ ------
B Registers:      5000   9F21   5081  62300   9CC5  61E7C  A0025
R Registers:        B2     3F      0   FFFF     3F      8   9103
M Registers:      FF00   FF00   FF00   FF00   FF00   FF00   FF0A

ASV$: .5000      TSAP$: .0        NATSAP: 1  I: .000C  T$: .A09FA

CI: .0004  RDBR$: .0
L6I       -TSA .50B6 ASDT .20400
TRAP: 15  TSAL$: .0          I: .0F18  INST: .D800  Z: .0083

A$: .200      P: .20996    B3: .A0052     R3: .15     S: .4032

Trap IC: KJF$FIX+.170

ISA:

P: .9026      S: .0000  CHN: .FFFF  ISM1: .FFFF  ISM2: .A103


                      1      2      3      4      5      6      7
                 ------ ------ ------ ------ ------ ------ ------
B Registers:     A005C  A0068   5081  20C8D  D0018  2087B  A0058
R Registers:        10      8   A103      2      1      3     1D
M Registers:      FF00   FF00   FF00   FF00   FF00   FF00   FF0A

ASV$: .5000      TSAP$: .0        NATSAP: 2  I: .0018  T$: .A09FA

CI: .0000  RDBR$: .0


L6I       -PL .A0000 F .2
```

```
CW=.9A3  MW=.9ED

Frame @.A0057 called from .20C71 (KJF$INTHAND+.253)
Frame @.A0041 called from .20A1E (KJF$INTHAND+.0)
Frame @.A0024 called from .20E51 (G_UPTRS_D+.6D)
Frame @.A000F called from .2043A (KJF$FEI+.0)
```