CP-6

INTERNALS (C00 LEVEL)

SEMINAR

AUGUST 12-23, 1985
Description: CP-6 INTERNALS COURSE (COO 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

MON  Administrative Info, Overview, DPS-8 Hardware Arch
WED  CP-6 User Modes, Intro to CP-6 OS
THUR Booting the System, Recovery
FRI  Comgroup Architecture
MON  MM, FM, RES & Prog Man Functions
TUE  Walking Tours of CP-6 Functions
WED  FEP Arch & LCP-6 Architecture
THUR FEP Users and System Components
FRI  

AFTERNOON

CP-6 Use of Hard. Mem. Arch.

Anatomy of a User

LAB 1 (SUPER) & LAB 2 (Shared Lib)
LAB 3 Use of ANLZ, STATS, CONTROL
LAB 4 Using COMGROUPS
LAB 5 Command Processor
LAB 5 cont
LAB 6 FPRG programs
LAB 7 ANLZ (FEP)
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

1-1 INT 8/85
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)

1-2B INT 8/85
NSA ADDED HARDWARE REGISTERS:

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

1–3A INT 8/85
NSA ADDED HARDWARE REGISTERS: (CONT)

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

1–3B INT 8/85
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

1-4B INT 8/85
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)

1-6A INT 8/85
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#)

1–7A INT 8/85
WHAT IS A WORKING SPACE (CONT)

- Specifies address of page table and size of page table

- WSPTD is pointed to by hardware register PTDBR

1-7B INT 8/85
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
WSPTD 0
WSR 0-7

PAGE TABLE
PTWn
RPA
ACC CNTR
PTW

511
1 WORD/PAGE

1-9 INT 8/85
SEGMENT CONTROL (PARTIAL)
A (NORMAL) DESCRIPTOR DESCRIBES A SEGMENT

<table>
<thead>
<tr>
<th>BOUND IN BYTES (20)</th>
<th>ACCESS CONTROL FLAGS</th>
<th>WSQ (Q)</th>
<th>WSQ/WSR</th>
<th>TYPE (4)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MISCELLANEOUS FLAGS (6)</td>
<td>WSR (3)</td>
<td>WSR (3)</td>
<td></td>
</tr>
</tbody>
</table>

BASE AS BYTE OFFSET INTO WS (26) (NORMALLY)

- 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>
<thead>
<tr>
<th>BOUND IN BYTES (20)</th>
<th>ACCESS CONTROL FLAGS</th>
<th>WSQ (0)</th>
<th>MISCELLANEOUS FLAGS (7)</th>
<th>WSR (3)</th>
<th>TYPE (4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASE AS BYTE OFFSET INTO WS (26) (NORMALY)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</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>
<thead>
<tr>
<th>BOUND IN BYTES (20)</th>
<th>ACCESS CONTROL FLAGS</th>
<th>WSQ (0)</th>
<th>MISCELLANEOUS FLAGS (0)</th>
<th>WSR (3)</th>
<th>TYPE (4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASE AS BYTE OFFSET INTO WS (26) (NORMALLY)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</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

<table>
<thead>
<tr>
<th></th>
<th>ACCESS CONTROL FLAGS (9)</th>
<th>CONTROL INFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOUND (20)</td>
<td>S (2)</td>
<td>DESCRIPTOR # (10)</td>
</tr>
<tr>
<td>BASE (20)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

1-12A INT 8/85
A VECTOR REFERENCES A SEGMENT OR PARTIAL SEGMENT VIA SOME DESCRIPTOR (cont.)

<table>
<thead>
<tr>
<th>BOUND (20)</th>
<th>ACCESS CONTROL FLAGS (9)</th>
<th>CONTROL INFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASE (20)</td>
<td>S (2)</td>
<td>DESCRIPTOR # (10)</td>
</tr>
</tbody>
</table>

- ACCESS FLAGS LOGICALLY ANDED WITH DESCRIPTOR FLAGS THEREFORE CAN SUBSET PRIVILEGES, BUT NOT ADD ANY
- S SPECIFIES LS, S, 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>
<thead>
<tr>
<th>BOUND (20)</th>
<th>ACCESS CONTROL FLAGS (9)</th>
<th>CONTROL INFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASE (20)</td>
<td>S (2)</td>
<td>DESCRIPTOR # (10)</td>
</tr>
</tbody>
</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

<table>
<thead>
<tr>
<th>WORD OFFSET 18</th>
<th>BYTE (2)</th>
<th>BIT (4)</th>
<th>S (2)</th>
<th>DESCRIPTOR # (10)</th>
</tr>
</thead>
</table>

- 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

1-13 INT 8/85
SOME SPECIAL REGISTERS WHICH WILL BE COMMONLY REFERENCED

LSR  - CONTAINS DESCRIPATOR WHICH FRAMES CURRENT LS

- LOADED ONLY AS A RESULT OF CLIMB INSTRUCTION

1-14A INT 8/85
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

1-14B INT 8/85
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

INS1

ADDRESS LDP2 1 X 0 TAG

INS2

3 15

2 4 LDA 0 L TAG

POINTER

24

18 2 4

2 10

11 0 1

.6001 = "LS1"

ARn DRn

18 2 4 72

DESCRIPTOR "LS1" (.6001)

.6001 = "LS1"

1-15 INT 8/85
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

1-16A INT 8/85
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

1-16B INT 8/85
THE USE OF DPS-8
MEMORY ARCHITECTURE (NSA)
BY CP-6

2-1 INT 8/85
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)

2-2A INT 8/85
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

2-2B INT 8/85
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

2-2C INT 8/85
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

2–3C INT 8/85
SIMPLIFIED VIEW OF USER’S WS (1MW)

<table>
<thead>
<tr>
<th>PAGE TABLE OF WS</th>
</tr>
</thead>
<tbody>
<tr>
<td>HJIT - 4LS’s SSS, AS/PS</td>
</tr>
<tr>
<td>JIT AND MONITOR TSTACK</td>
</tr>
<tr>
<td>FILE BUFFERS</td>
</tr>
<tr>
<td>DCBs</td>
</tr>
<tr>
<td>INSTRUCTION SEGMENT</td>
</tr>
<tr>
<td>USERS 8 DYNAMIC SEGMENTS</td>
</tr>
<tr>
<td>CP’S DYNAMIC SEGMENTS</td>
</tr>
<tr>
<td>DB’S 8 DYNAMIC SEGMENTS</td>
</tr>
<tr>
<td>ASL’S 8 DYNAMIC SEGMENTS</td>
</tr>
</tbody>
</table>

2-4 INT 8/85
# 1 MW USER VIRTUAL LAYOUT

<table>
<thead>
<tr>
<th>PAGE TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>10 (.12)</td>
</tr>
<tr>
<td>11 (.13)</td>
</tr>
<tr>
<td>12 (.14)</td>
</tr>
<tr>
<td>USER RD</td>
</tr>
<tr>
<td>13 (.15)</td>
</tr>
<tr>
<td>14 (.16)</td>
</tr>
<tr>
<td>15 (.17)</td>
</tr>
<tr>
<td>16 (.20)</td>
</tr>
<tr>
<td>USER</td>
</tr>
<tr>
<td>46 (.56)</td>
</tr>
<tr>
<td>47 (.57)</td>
</tr>
<tr>
<td>60 (.74)</td>
</tr>
<tr>
<td>61 (.75)</td>
</tr>
<tr>
<td>63 (.77)</td>
</tr>
<tr>
<td>64 (.100)</td>
</tr>
<tr>
<td>USER R,W,E</td>
</tr>
<tr>
<td>319 (.477)</td>
</tr>
<tr>
<td>320 (.500)</td>
</tr>
<tr>
<td>321 (.501)</td>
</tr>
<tr>
<td>322 (.502)</td>
</tr>
<tr>
<td>327 (.507)</td>
</tr>
<tr>
<td>328 (.510)</td>
</tr>
<tr>
<td>391 (.607)</td>
</tr>
<tr>
<td>392 (.610)</td>
</tr>
<tr>
<td>SBUF2</td>
</tr>
<tr>
<td>SBUF1</td>
</tr>
<tr>
<td>STEP SPECIAL BUFFERS - 2 PAGES</td>
</tr>
<tr>
<td>6 RESERVED PAGES</td>
</tr>
<tr>
<td>DB DATA SEGMENTS -UP TO 64 PAGES-</td>
</tr>
<tr>
<td>ASL DATA SEGMENTS -UP TO 128 PAGES-</td>
</tr>
<tr>
<td>CP DATA SEGMENTS -UP TO 32 PAGES-</td>
</tr>
<tr>
<td>USER DATA SEGMENTS -UP TO 384 PAGES-</td>
</tr>
<tr>
<td>88 RESERVED PAGES</td>
</tr>
</tbody>
</table>

## HEAVILY ACCESSED PAGES

<table>
<thead>
<tr>
<th>NAME</th>
<th>V.P.#</th>
<th>AM</th>
<th>ROW#</th>
</tr>
</thead>
<tbody>
<tr>
<td>JIT</td>
<td>.13</td>
<td>.13</td>
<td></td>
</tr>
<tr>
<td>HJIT</td>
<td>.15</td>
<td>.15</td>
<td></td>
</tr>
<tr>
<td>ROSEG(1)</td>
<td>.67</td>
<td>.17</td>
<td></td>
</tr>
<tr>
<td>ISR(1)</td>
<td>.100</td>
<td>.0</td>
<td></td>
</tr>
<tr>
<td>U. DATA SEG. (1)</td>
<td>.1050</td>
<td>.10</td>
<td></td>
</tr>
<tr>
<td>STEP BUFFER #1</td>
<td>.501</td>
<td>.01</td>
<td></td>
</tr>
</tbody>
</table>

2-5 INT 8/85
1MW MONITOR VIRTUAL LAYOUT

<table>
<thead>
<tr>
<th>Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PAGE TABLE FOR 1ST MW</td>
</tr>
<tr>
<td>1</td>
<td>10 PAGES RESERVED FOR CGQ PAGE TABLE</td>
</tr>
<tr>
<td>10 (.12)</td>
<td>JIT &amp; TSTACK &amp; TCB -2 PAGES-</td>
</tr>
<tr>
<td>11 (.13)</td>
<td>HJIT (2 PAGES)</td>
</tr>
<tr>
<td>12 (.14)</td>
<td></td>
</tr>
<tr>
<td>13 (.15)</td>
<td></td>
</tr>
<tr>
<td>15 (.17)</td>
<td></td>
</tr>
<tr>
<td>46 (.56)</td>
<td>32 RESERVED PAGES</td>
</tr>
<tr>
<td>47 (.57)</td>
<td>LOW REAL MEMORY - PHYSICAL PAGES 0-3</td>
</tr>
<tr>
<td>50 (.62)</td>
<td>MONITOR WINDOW AREA - USED FOR MISCELLANEOUS</td>
</tr>
<tr>
<td>51 (.63)</td>
<td>MAPPING WITHIN ISOLATED ROUTINES</td>
</tr>
<tr>
<td>60 (.74)</td>
<td>3 RESERVED PAGES</td>
</tr>
<tr>
<td>61 (.75)</td>
<td>INSTRUCTION SEGMENT</td>
</tr>
<tr>
<td>63 (.77)</td>
<td>8 RESERVED PAGES</td>
</tr>
<tr>
<td>64 (.100)</td>
<td>MM_PUT - PHYSICAL PAGE USAGE TABLE 16K, MAXIMUM</td>
</tr>
<tr>
<td>319 (.477)</td>
<td>USER STATE TABLES FOR PERFORMANCE MONITOR</td>
</tr>
<tr>
<td>320 (.500)</td>
<td>-8 PAGES-</td>
</tr>
<tr>
<td>327 (.507)</td>
<td>MOUSE DATA -8 PAGES-</td>
</tr>
<tr>
<td></td>
<td>PAGE TABLES &amp; HJITS FOR ALL CPUS -8 PAGES-</td>
</tr>
<tr>
<td></td>
<td>COMMUNICATIONS WSQ PAGE TABLES -8 PAGES-</td>
</tr>
<tr>
<td></td>
<td>16 RESERVED PAGES</td>
</tr>
<tr>
<td></td>
<td>TIGR BUILT TABLES -128 PAGES-</td>
</tr>
<tr>
<td></td>
<td>MPC Firmware -32 PAGES-</td>
</tr>
<tr>
<td></td>
<td>DS1 - CFUs 50 PAGES</td>
</tr>
<tr>
<td></td>
<td>DS2 - AUTOSAVE TABLES 20 PAGES</td>
</tr>
<tr>
<td></td>
<td>DS3 - ENQ/DEQ TABLES 24 PAGES</td>
</tr>
<tr>
<td></td>
<td>DS4 - UNASSIGNED 60 PAGES</td>
</tr>
<tr>
<td></td>
<td>DS5 - UNASSIGNED 60 PAGES</td>
</tr>
<tr>
<td></td>
<td>DS6 - UNASSIGNED 60 PAGES</td>
</tr>
<tr>
<td></td>
<td>DS7 - UNASSIGNED 62 PAGES</td>
</tr>
<tr>
<td></td>
<td>DS8 - LDCTs &amp; COMGROUP CONTEXT 48 PAGES</td>
</tr>
<tr>
<td></td>
<td>88 RESERVED PAGES</td>
</tr>
</tbody>
</table>

2-6 INT 8/85
HOW HARDWARE DEFINES (CONTAINS) USERS DOMAIN
(USER PROGRAM CURRENTLY IN CONTROL)
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

2-8 INT 8/85
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

2-9A INT 8/85
THE MONITOR DOMAIN (CONT)

- RO SEGMENT (TCB, ECCB, DCBs)
- HJIT
- USERS MAP
- THERE ARE MANY SPECIAL PURPOSE DESCRIPTORS

2-9B INT 8/85
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 LS
NULL
R
NULL
R
IS:R/W/E
R/W
R/W
NULL

PAGE TABLE
HJIT
JIT
BUFFERS
DEBUGGER DATA
ALTLIB DATA
DCB'S
LIBRARY DATA
BOUND DATA
PROCEDURE: W
DYNAMIC DATA
UNUSED
LIB PROCEDURE: W
DYNAMIC
SEGMENTS

MONITOR LS
R/W
R/W
R/W
R/W
IS: R/W/E
R/W
R/W
R/W

2-10 INT 8/85
"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

2-11A INT 8/85
"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

2-12A INT 8/85
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

2-12B INT 8/85
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

2-13 INT 8/85
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

2-14A INT 8/85
THE DEBUGGER DOMAIN
(CONT)

- GIVES CONTROL BACK VIA SPECIAL MONITOR SERVICE

2-14B INT 8/85
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

2-16B INT 8/85
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

2-17A INT 8/85
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

2-17C INT 8/85
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

3-2 INT 8/85
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 $\alpha$
PMME  CODE, n
[TRA $\beta$]

$\beta$ EXCEPTION PROCESSING

FRAMES VALUE PARAM BLOCK

FRAMES MEM PAR 1

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

AS REQUIRED

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

3-5 INT 8/85
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

3-6A INT 8/85
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

3-6B INT 8/85
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)

3-7A INT 8/85
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

3-7C INT 8/85
DEBUG SCHEMA:

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

3-8 INT 8/85
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

3-9 INT 8/85
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

3-10A INT 8/85
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

PAGE BOUNDARY

LONGEST PATH OF PROCEDURE

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

3-12 INT 8/85
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

3-13 INT 8/85
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

3-14A INT 8/85
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

3-14B INT 8/85
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

3-15 INT 8/85
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

4-1A INT 8/85
ONLINE/BATCH/ Ghost/ TP MODE (CONT)

- SOURCE OF COMMAND STREAM
- AUTHORIZATION LIMITS
- DEFAULT ASSIGNMENT OF DCBs
A TIMESHARING USER

USER PROGRAM

EDIT

COMPILER

IBEX

COMMAND STREAM INPUT

USER CONSOLE

4-2 INT 8/85
A BATCH USER
A GHOST USER

USER PROGRAM

EDIT

COMPILER

IBEX

COMMAND STREAM INPUT

COMMAND FILE

4-4 INT 8/85
SPECIAL SYSTEM GHOST

USER PROGRAM

4-5 INT 8/85
TRANSACTION PROCESSING USER

USER PROGRAM

EDIT

PCL

TPCP

COMGROUP

4-6 INT 8/85
OVERALL PICTURE OF CP-6 OPERATING SYSTEM

SYSTEM GHOSTS

USER PROGRAMS

ASL (I-D-S/II)

CP (IBEX)

DEBUGGERS (DELTA)

SHARED LIBRARIES

MONITOR

5-1 INT 8/85
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 simultaneously 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

5–2B INT 8/85
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

5-3B INT 8/85
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

5-3C INT 8/85
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

5-4 INT 8/85
SYSTEM PROGRAMS

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

<table>
<thead>
<tr>
<th>CONTROL</th>
<th>SYSCON</th>
</tr>
</thead>
<tbody>
<tr>
<td>NETCON</td>
<td>LABEL</td>
</tr>
<tr>
<td>STATS</td>
<td>SPIDER</td>
</tr>
<tr>
<td>SUPER</td>
<td>DEF</td>
</tr>
<tr>
<td>RATES</td>
<td>DEF</td>
</tr>
<tr>
<td>EFT</td>
<td>VOLINIT</td>
</tr>
<tr>
<td>IMP</td>
<td>PIGETTE</td>
</tr>
<tr>
<td>ELAN</td>
<td>REPLAY</td>
</tr>
<tr>
<td>ANLZ</td>
<td>TRADER</td>
</tr>
</tbody>
</table>

5-5 INT 8/85
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

6-1 INT 8/85
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 ev "A"s

6-2 INT 8/85
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

6–3 INT 8/85
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

6-4 INT 8/85
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
  - NODEADMIN, COUPLER, ASYNC, BISYNC, UNITREC, HDLCX25

- FEP orphans
  - PIGLET, ANLZ_FPRG, DELTA_FPRG

6-6 INT 8/85
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

6-7 INT 8/85
PO tape, volumes 2 and beyond

- BASIC
- APL
- 6EDIT
- ARE5/ARGENT/XARGENT
- IDS (DBUTIL, DBACS)
- IDP
- RPG
- GMAP6
- MAIL/SEND/NODEATER

- Each has libraries, HELP files

6-8 INT 8/85
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

6–9 INT 8/85
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?:SYSSTAC

6–10 INT 8/85
BOOT SEQUENCE

- convert RUMs into :RUM
- convert PLOVERs into ::PLOVER
- convert BOOTTIME
- "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

6-11 INT 8/85
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

6–12 INT 8/85
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)

6-13 INT 8/85
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!

6-14 INT 8/85
TOOLS TAPES

- SUPPORT
- documentation (SRBs, manuals)
- X
- DEMO
- QUAC
- CONV
- :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 }

6–16 INT 8/85
ALTERNATE SCHEMA

XDELTAs 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

7-1 INT 8/85
## MACHINE CONTEXT IN VARIOUS STATES:

<table>
<thead>
<tr>
<th>CONTEXT</th>
<th>USER RUNNING</th>
<th>MONITOR SERVICE</th>
<th>MONITOR RUNNING</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSR</td>
<td>USER LS IN USER HJIT (PER USER)</td>
<td>MONITOR LS IN MON HJIT (PER CPU)</td>
<td></td>
</tr>
<tr>
<td>PSR/ASR</td>
<td>USER PS/AS IN USER HJIT</td>
<td></td>
<td>NOT USED</td>
</tr>
<tr>
<td>SSR</td>
<td>USER SAFE STONE IN USER HJIT</td>
<td></td>
<td>MONITOR SS IN MON HJIT</td>
</tr>
<tr>
<td>MASTER/S</td>
<td>SLAVE</td>
<td>MASTER (PRIV)</td>
<td></td>
</tr>
<tr>
<td>SLAVE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WSR7</td>
<td>8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7-2 INT 8/85
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

7–3A INT 8/85
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

7-4A INT 8/85
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

7-5A INT 8/85
MONITOR EXECUTION MODES:
(CONT)

• MONITOR EXECUTION (SCHEDULING, SERVICING INTERRUPTS, ETC.)

• NOT RUNNING FOR ANY USER

• TIME CHARGED TO SYSTEM OVERHEAD

• RUNS IN CPU CONTEXT

7-5B INT 8/85
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

7-6A INT 8/85
TYPES OF MONITOR DATA:
(CONT)

- REAL (BUILT BY TIGR BASED ON CONFIGURATION NEEDS)

- READ 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

7-6E INT 8/85
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

7-8 INT 8/85
FILE MANAGEMENT STRUCTURE

OVERVIEW

7-9 INT 8/85
PACKS AND PACK SETS

SYSTEM PACK

SYSTEM AREA

#SYS #USER1 #USER2 #USER3

- EACH VOLUME CONTAINS VID (Volume IDentification)
MAD (Master Account Directory)

<table>
<thead>
<tr>
<th>PUBLIC ACCOUNT 1</th>
<th>CONTAINING PACKSET</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUBLIC ACCOUNT 2</td>
<td>CONTAINING PACKSET</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>PUBLIC ACCOUNT n</td>
<td>CONTAINING PACKSET</td>
</tr>
</tbody>
</table>

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

7-11 INT 8/85
**PAD (Packset Account Directory)**

<table>
<thead>
<tr>
<th>PACKSET OWNER, ATTRIBUTES</th>
<th>DEFAULTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCOUNT 1</td>
<td>FD SRDA</td>
</tr>
<tr>
<td>ACCOUNT 2</td>
<td>FD SRDA</td>
</tr>
<tr>
<td></td>
<td>(SET RELATIVE DISK ADDRESS)</td>
</tr>
<tr>
<td>ACCOUNT n</td>
<td>FD SRDA</td>
</tr>
</tbody>
</table>

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

7-12 INT 8/85
## FD (File Directory)

<table>
<thead>
<tr>
<th>ACCOUNT OWNER, ATTRIBUTES, DEFAULTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE 1</td>
</tr>
<tr>
<td>FILE 2</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>FILE n</td>
</tr>
</tbody>
</table>
### FIT (File Information Table)

<table>
<thead>
<tr>
<th>ACCESS CONTROLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORG, NRECS, UGRANS, GAVAL</td>
</tr>
<tr>
<td>DATES (OPEN UPDATE INDICATOR)</td>
</tr>
<tr>
<td>UATTR, INSTATTR</td>
</tr>
<tr>
<td>EXTEND LIST</td>
</tr>
<tr>
<td>TDA (Top Disk Address)</td>
</tr>
<tr>
<td>FDA (First Disk Address)</td>
</tr>
<tr>
<td>LDA (Last Disk Address)</td>
</tr>
<tr>
<td>ALL ARE FRDA (File Relative Disk Address)</td>
</tr>
<tr>
<td>GRANULE STAMP HASH</td>
</tr>
</tbody>
</table>

7-14 INT 8/85
CONSECUTIVE, UR, SYMBIONT FILE STRUCTURE

<table>
<thead>
<tr>
<th>STAMP HASH</th>
<th>GMOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAVX</td>
<td>HDR</td>
</tr>
<tr>
<td>LVL</td>
<td>ORG</td>
</tr>
<tr>
<td>AVAL</td>
<td></td>
</tr>
<tr>
<td>FCEX</td>
<td>END</td>
</tr>
<tr>
<td>LVL</td>
<td>X</td>
</tr>
<tr>
<td>KEYCNT</td>
<td></td>
</tr>
</tbody>
</table>

GRANULE NUMBER MODULO 512

• RECORD SPANNING FOR CONSEC, UR

FAK C CTL GX GACTB

7-15 INT 8/85
**RELATIVE FILE STRUCTURE**

<table>
<thead>
<tr>
<th>STAMP HASH</th>
<th>GMOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAVX</td>
<td>HDR</td>
</tr>
<tr>
<td></td>
<td>LVL</td>
</tr>
<tr>
<td></td>
<td>ORG</td>
</tr>
<tr>
<td></td>
<td>AVAIL</td>
</tr>
<tr>
<td>FCEX</td>
<td>END</td>
</tr>
<tr>
<td></td>
<td>LVL</td>
</tr>
<tr>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>KEYCNT</td>
</tr>
<tr>
<td>A</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>GACTB</td>
</tr>
</tbody>
</table>

**REC 1**

**REC 2**

... etc.

**REC y**

- **NO RECORD SPANNING**
- **GRANULE NUMBER CALCULATED FROM RECORD NUMBER**

7-16 INT 8/85
KEYED (INDEXED) FILE STRUCTURE
FILE MANAGEMENT SERVICE CATEGORIES

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

7-18 INT 8/85
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

7-21A INT 8/85
EXECUTION CONTROL SERVICE CATEGORIES (CONT)

- PRIVILEGED/COMMAND PROGRAM SERVICES
IREL KEY STRUCTURE

<table>
<thead>
<tr>
<th>D</th>
<th>KLB</th>
<th>UBIN HALF</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR(1)</td>
<td>SINGLE FLOATING</td>
<td></td>
</tr>
<tr>
<td>HEX</td>
<td>PACKED DECIMAL (5)</td>
<td></td>
</tr>
<tr>
<td>SBIN BYTE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

MISEG (AS IN KEYED)

---

7-22 INT 8/85
IO CACHE

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

7-23 INT 8/85
GRANULE TYPES

- MAD, PAD, GP, FD, FIT, UL
  INDEX, DATA, REL, CONSEC, ELSE

- CONTROLS AND STATS BY GRANULE TYPE

- STATS/ANLZ GIVE INFO

7-24 INT 8/85
## CACHE TABLE

<table>
<thead>
<tr>
<th>TYPE</th>
<th>SRDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLINK</td>
<td>BLINK</td>
</tr>
<tr>
<td>PAGE</td>
<td>USECNT</td>
</tr>
<tr>
<td>UPCOUNT</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SETX</th>
<th>USER</th>
<th>*</th>
<th>I</th>
<th>W</th>
<th>E</th>
<th>B</th>
</tr>
</thead>
</table>

**AGE**
COMMUNICATION GROUPS

COMGROUP

DCB  DCB  DCB  DCB
PROGRAM #1  PROGRAM #2

FILE

8-1 INT 8/85
CONNECTIONS TO COMGROUPS

COMGROUP

DCB  DCB
CG WITH ONLY DCBs

COMGROUP

DCB
CG WITH DCBs AND STATIONS

COMGROUP

X
CG WITH ONLY STATIONS (NO SUCH THING)

8-2 INT 8/85
OPEN COMGROUP

STATIONS

DCB

MEMORY BUFFER SPACE

FM

FILE SYSTEM BACKING STORE

HOST MEMORY:

- MSOPEN
- MESSAGES
- TERMINALS (ACTIVE COMGROUP ONLY)

DISK FILE:

- MSOPEN
- ACCESS CONTROLS
- BACKUP
- OVERFLOW
- PACKAGING

8-4 INT 8/85
CLOSED COMGROUP

FILE
SYSTEM
BACKING
STORE

8-5 INT 8/85
STATION I/O

READ: MESSAGE TYPE
ORIGIN STATION

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

8-6 INT 8/85
COMGROUP MESSAGE

MESSAGE BLOCK

- ATTRIBUTES
- DBLK

DATA BLOCK

- DATA

- 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

8-7 INT 8/85
QUEUE ORGANIZATION

STATION TREE

- 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

MESSAGE TYPE TREE

8-8A INT 8/85
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

8-8B INT 8/85
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

8-9 INT 8/85
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

8-10A INT 8/85
READ AND LATCH

- LATCHED OUTPUTS
  - MESSAGE ID 'SPAWN' TRAIL
  - INVISIBLE
  - REPORTS/NEW TRANSACTIONS

- NEXT READ TRIGGERS UNLATCH
  - DELETE INPUT
  - SEND OUTPUTS

8-10B INT 8/85
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

8-11A INT 8/85
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

8-12A INT 8/85
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

8-12B INT 8/85
ADMINISTRATIVE USER FUNCTIONS
(CONT)

- SET LIST OF STATIONS
- PRIORITY

8-12C INT 8/85
ADMINISTRATIVE USER FUNCTIONS (CONT)

INFORMATION

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

8-12D INT 8/85
ADMINISTRATIVE USER FUNCTIONS (CONT)

- STATISTICS
  - QUEUE DEPTH
  - TRANSACTION RATE
  - DISK CACHE

- CURRENT MESSAGE TYPES
  - QUEUE DEPTH
  - STATISTICS

8-12E INT 8/85
ADMINISTRATIVE USER FUNCTIONS (CONT)

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

8-12F INT 8/85
COMGROUP MONITOR SERVICES

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

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

8-13 INT 8/85
WALKING TOURS OF SOME CP-6 FUNCTIONS

9-1 INT 8/85
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

9-2A INT 8/85
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)

9-2B INT 8/85
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

9-2C INT 8/85
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$UG (ROSEG)

- SLUG EXECUTES CALL M$MAKUSER (CAN FAIL)

9-2D INT 8/85
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

9-2G INT 8/85
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

9-2H INT 8/85
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

9-3A INT 8/85
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...

9-3B INT 8/85
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

9-3C INT 8/85
THE TRIP A TRANSACTION TAKES THROUGH TP (CONT)

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

9-3D INT 8/85
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

9-4B INT 8/85
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.

9-4C INT 8/85
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

9-5A INT 8/85
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)

9-5B INT 8/85
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).

9-5C INT 8/85
RUNNING DOWN A USER (STEP)

- USER PROGRAM DOES M$EXIT ON ERRORS OR ABORTS FOLLOWED BY M$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

9-6A INT 8/85
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

9-6B INT 8/85
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)

- LOGON IS ASSOCIATED, DOES FINAL ACCOUNTING FUNCTIONS

- LOGON ISSUES M$CPEXIT (OFF)

9-7A INT 8/85
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

9-7B INT 8/85
RUNNING DOWN A USER (LOGOFF) (CONT)

- SCHEDULER CHANGES USER STATE TO SS_NULL

- USER NOT LONGER EXISTS

- FEP EITHER DROPS LINE OR REISSUES SALUATATION

9-7C INT 8/85
READ A RECORD FROM A T/S TERMINAL

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

- USER ISSUES M$READ

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

9-8A INT 8/85
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)

- 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)

9-88 INT 8/85
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

9–8C INT 8/85
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)

9–8D INT 8/85
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

9-8E INT 8/85
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

9–8F INT 8/85
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

9-9A INT 8/85
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

9-9B INT 8/85
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

9–9C INT 8/85
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

9-9D INT 8/85
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

9-9E INT 8/85
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)

9-10A INT 8/85
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

9-10B INT 8/85
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

9-10C INT 8/85
READ A RECORD FROM A (KEYED) FILE (CONT)

- SET ARS IN DCB
- RETURN TO USER

9-10D INT 8/85
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

9-11A INT 8/85
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.

9-11B INT 8/85
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

9-11C INT 8/85
CONNECT A DCB TO A (CLOSED) COMGROUP (CONT)

- RETURN CONTROL TO USER

9-11D INT 8/85
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)

9-12A INT 8/85
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)

9-12B INT 8/85
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

9-12C INT 8/85
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

9-12D INT 8/85
DATA NET 8 (L6) ARCHITECTURE
GENERAL ARCHITECTURE

• 16 BIT MINI-COMPUTER
• 7–16 BIT GENERAL REGISTER
• 7–20 BIT BASE REGISTER
• COMMERCIAL INSTRUCTION PROCESSOR (CIP)

10-2 INT 8/85
MEMORY MANAGEMENT

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

10-3 INT 8/85
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

<table>
<thead>
<tr>
<th>INTERRUPT LEVELS</th>
<th>DESIGNATED MEMORY LOCATION (LAF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00080</td>
</tr>
<tr>
<td>1</td>
<td>00082</td>
</tr>
<tr>
<td>2</td>
<td>00084</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>20</td>
<td>000A8</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>30</td>
<td>000BC</td>
</tr>
</tbody>
</table>

10-6 INT 8/85
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

10-7 INT 8/85
TRAP VECTOR AND INTERRUPT VECTOR LINKAGE

HARDWARE DEDICATED MEMORY LOCATIONS

0000  NATSAPO
0010  TV #2
007E  TV #1
007F  TV #1
0080  IV #0
0081  IV #1
0082  IV #2
00BF  IV #63

AVAILABLE TSA's (POOL 0)

TSA
  TSAL
  I
  ETC

TSA
  TSA
  TSAL
  I
  ETC

TSA
  NULL
  I
  ETC

TSA
  TRAP HANDLER

TSA
  TASK

OPT WORK SPACE

10-8 INT 8/85
LCP–6 OPERATING SYSTEM FEATURES

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

10–9 INT 8/85
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

10-11 INT 8/85
<table>
<thead>
<tr>
<th>Active Process</th>
<th>MMU Image Source</th>
<th>USER/HANDLER</th>
<th>MON FOR USER</th>
<th>MONITOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>UASDT</td>
<td>MASDT</td>
<td>VADDR</td>
<td>UHJIT.ASDT_USR</td>
</tr>
<tr>
<td>00</td>
<td>.5000</td>
<td>.503E</td>
<td>.00000</td>
<td>NULLSEG</td>
</tr>
<tr>
<td>01</td>
<td>.5002</td>
<td>.5040</td>
<td>.01000</td>
<td>ROS</td>
</tr>
<tr>
<td>02</td>
<td>.5004</td>
<td>.5042</td>
<td>.02000</td>
<td>DB_DS</td>
</tr>
<tr>
<td>03</td>
<td>.5006</td>
<td>.5044</td>
<td>.03000</td>
<td>LOW_MEM</td>
</tr>
<tr>
<td>04</td>
<td>.5008</td>
<td>.5046</td>
<td>.04000</td>
<td>TSTACKU</td>
</tr>
<tr>
<td>05</td>
<td>.500A</td>
<td>.5048</td>
<td>.05000</td>
<td>UHJIT</td>
</tr>
<tr>
<td>06</td>
<td>.500C</td>
<td>.504A</td>
<td>.06000</td>
<td>MHJIT</td>
</tr>
<tr>
<td>07</td>
<td>.500E</td>
<td>.504C</td>
<td>.07000</td>
<td>MHJIT</td>
</tr>
<tr>
<td>08</td>
<td>.5010</td>
<td>.504E</td>
<td>.08000</td>
<td>MON_ENTRY_DATA</td>
</tr>
<tr>
<td>09</td>
<td>.5012</td>
<td>.5050</td>
<td>.09000</td>
<td>MON_ENTRY</td>
</tr>
<tr>
<td>10</td>
<td>.5014</td>
<td>.5052</td>
<td>.0A000</td>
<td>USER_DS1</td>
</tr>
<tr>
<td>11</td>
<td>.5016</td>
<td>.5054</td>
<td>.0B000</td>
<td>USER_DS2</td>
</tr>
<tr>
<td>12</td>
<td>.5018</td>
<td>.5056</td>
<td>.0C000</td>
<td>CP_DS</td>
</tr>
<tr>
<td>13</td>
<td>.501A</td>
<td>.5058</td>
<td>.0D000</td>
<td>*</td>
</tr>
<tr>
<td>14</td>
<td>.501C</td>
<td>.505A</td>
<td>.0E000</td>
<td>*</td>
</tr>
<tr>
<td>15</td>
<td>.501E</td>
<td>.505C</td>
<td>.0F000</td>
<td>*</td>
</tr>
<tr>
<td>16</td>
<td>.5020</td>
<td>.505E</td>
<td>.10000</td>
<td>USER_IS1</td>
</tr>
<tr>
<td>17</td>
<td>.5022</td>
<td>.5060</td>
<td>.20000</td>
<td>USER_IS2</td>
</tr>
<tr>
<td>18</td>
<td>.5024</td>
<td>.5062</td>
<td>.30000</td>
<td>USER_IS3</td>
</tr>
<tr>
<td>19</td>
<td>.5026</td>
<td>.5064</td>
<td>.40000</td>
<td>USER_IS4</td>
</tr>
<tr>
<td>20</td>
<td>.5028</td>
<td>.5066</td>
<td>.50000</td>
<td>USER_IS5</td>
</tr>
<tr>
<td>21</td>
<td>.502A</td>
<td>.5068</td>
<td>.60000</td>
<td>USER_IS6 (LIB)</td>
</tr>
<tr>
<td>22</td>
<td>.502C</td>
<td>.506A</td>
<td>.70000</td>
<td>USER_IS7 (LIB)</td>
</tr>
<tr>
<td>23</td>
<td>.502E</td>
<td>.506C</td>
<td>.80000</td>
<td>DB_PROC</td>
</tr>
<tr>
<td>24</td>
<td>.5030</td>
<td>.506E</td>
<td>.90000</td>
<td>CP_PROC</td>
</tr>
<tr>
<td>25</td>
<td>.5032</td>
<td>.5070</td>
<td>.A0000</td>
<td>UAUTO_DS</td>
</tr>
<tr>
<td>26</td>
<td>.5034</td>
<td>.5072</td>
<td>.B0000</td>
<td>USER_DS3</td>
</tr>
<tr>
<td>27</td>
<td>.5036</td>
<td>.5074</td>
<td>.C0000</td>
<td>USER_DS4</td>
</tr>
<tr>
<td>28</td>
<td>.5038</td>
<td>.5076</td>
<td>.D0000</td>
<td>HAND_Q</td>
</tr>
<tr>
<td>29</td>
<td>.503A</td>
<td>.5078</td>
<td>.E0000</td>
<td>*</td>
</tr>
<tr>
<td>30</td>
<td>.503C</td>
<td>.507A</td>
<td>.F0000</td>
<td>*</td>
</tr>
</tbody>
</table>
SCHEDULING IN LCP-6

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

10-13A INT 8/85
SCHEDULING IN LCP-6 (CONT)

• LEVELS 12–60 AVAILABLE FOR HANDLERS CONNECTED INDIRECTLY VIA M$INTCON

• LEVEL 3 COMMON INHIBIT LEVEL

10-13B INT 8/85
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

10-16A INT 8/85
COMGROUP FPRG (CONT)

• USES

  – FORMS CONTROL
  – E.G. FPL FOR TP

10-168 INT 8/85
GHOST FPRG

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

10-17 INT 8/85
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. SYNC
  - NODE ADMINISTRATOR
  - GATEWAYS

10-18B INT 8/85
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)

10-19A INT 8/85
DEBUGGING FEPS AND FPRGS (CONT)

- ENTIRE FEP IN EXECUTIVE MODE VIA SA ASYNC

- FPL PROGRAMS IN CONCERT WITH FPL INTERPRETER

10-19B INT 8/85
DEBUGGING HOST FPRGS

- AUTOMATIC IF DEBUGGING HOST PROGRAM
- USE UCnn/USE HOST/USE FEnn
DELTA USED TO DEBUG HOST PROGRAM AND FPRG SIMULTANEOUSLY
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

10-22B INT 8/85
DELTA USED TO DEBUG TP FPRG
FEP USERS

AND

SYSTEM COMPONENTS

11-1 INT 8/85
SYSTEM SERVICE INTERFACE

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

11-2 INT 8/85
LCP6 OU/RU

- FORMATS/SCHEMA SAME AS ON HOST

- FEPLINK OFFERS SUBSET OF LINK FEATURES

- DELTA PROCESSES HOST/FEP SCHEMA WITH SAME CODE

11-3A INT 8/85
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  desc
LNJ, B6  sub
DC       altret, PREL (i if none)

11-5 INT 8/85
RECEIVING SEQUENCE

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

11–6 INT 8/85
<table>
<thead>
<tr>
<th>AUTO STORAGE BASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

11-7 INT 8/85
### AUTO HEAD

<table>
<thead>
<tr>
<th>Index</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CUR_USED</td>
</tr>
<tr>
<td>1</td>
<td>MAX_ALLOWED</td>
</tr>
<tr>
<td>2</td>
<td>MUST_BE_NIL</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
## AUTO FRAME FORMAT

**B7 LOCATES AUTO FRAME**

<table>
<thead>
<tr>
<th>Index</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>FRAME SIZE</td>
</tr>
<tr>
<td>0</td>
<td>RET_ADDR</td>
</tr>
<tr>
<td>1</td>
<td>TYPE</td>
</tr>
<tr>
<td>3</td>
<td>PREV_FR_SIZE</td>
</tr>
</tbody>
</table>

11-9 INT 8/85
FRAMES AlLOCATED UP/DOWN

• MONITOR (FIXED AUTO) USES STACK

• USER PERVERTS STACK

11–10 INT 8/85
LCP6 LIBRARIES

- FPL
- VDH

11-11 INT 8/85
FPL LIBRARY

- SUPPORTS INTERPRETIVE PROCEDURE

- IN-LINE CODE FOR ENTRY, EXIT, CALL

- SUPPORTS DELTA

11–12 INT 8/85
VDH

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

11-13 INT 8/85
HANDLER COMMON

- STATS
- CONTROL/NETCON
- ERROR LOG

11-14 INT 8/85
ASYNC

- LINE MANAGEMENT
- WORKS WITH VDH

11-15 INT 8/85
BISYNC

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

11-16 INT 8/85
UNIT REC

- HANDLES LP, CR, CP

- INTERFACE SAME AS BISYNC

11-17 INT 8/85
HDLCX25

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

11-18 INT 8/85
NODEADMIN

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

11-19 INT 8/85
COUPLER

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

11-20 INT 8/85
ANLZ__FPRG, DELTA__FPRG

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

11-21 INT 8/85
OTHER FPRGS

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

11-22 INT 8/85
17:07 AUG 29 '85 DRIB:STATS.ZZINT

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

<table>
<thead>
<tr>
<th>Interval end</th>
<th>Online</th>
<th>Batch</th>
<th>% exec</th>
<th>% serv</th>
<th>% mon</th>
<th>I/Os</th>
<th>PMMEs</th>
<th>Free Pgs</th>
<th>ETMF</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:18:04.20</td>
<td>58</td>
<td>3</td>
<td>152.6</td>
<td>124.4</td>
<td>21.0</td>
<td>3631</td>
<td>22769</td>
<td>2903</td>
<td>1</td>
</tr>
<tr>
<td>09:48:03.74</td>
<td>63</td>
<td>3</td>
<td>158.8</td>
<td>98.9</td>
<td>17.5</td>
<td>3036</td>
<td>21185</td>
<td>3145</td>
<td>1</td>
</tr>
<tr>
<td>10:18:03.63</td>
<td>68</td>
<td>3</td>
<td>146.8</td>
<td>128.1</td>
<td>16.4</td>
<td>2393</td>
<td>29091</td>
<td>3274</td>
<td>1</td>
</tr>
<tr>
<td>10:48:03.47</td>
<td>63</td>
<td>3</td>
<td>159.0</td>
<td>94.2</td>
<td>19.9</td>
<td>3434</td>
<td>19106</td>
<td>2889</td>
<td>1</td>
</tr>
<tr>
<td>11:18:03.91</td>
<td>68</td>
<td>4</td>
<td>180.2</td>
<td>98.6</td>
<td>15.2</td>
<td>2190</td>
<td>26267</td>
<td>3310</td>
<td>2</td>
</tr>
<tr>
<td>11:48:04.46</td>
<td>71</td>
<td>3</td>
<td>156.8</td>
<td>112.1</td>
<td>18.3</td>
<td>3269</td>
<td>22614</td>
<td>3138</td>
<td>2</td>
</tr>
<tr>
<td>12:18:04.90</td>
<td>69</td>
<td>4</td>
<td>177.5</td>
<td>99.6</td>
<td>17.3</td>
<td>2854</td>
<td>21174</td>
<td>3297</td>
<td>1</td>
</tr>
<tr>
<td>12:48:04.98</td>
<td>77</td>
<td>4</td>
<td>186.0</td>
<td>95.5</td>
<td>15.0</td>
<td>2299</td>
<td>23432</td>
<td>3001</td>
<td>1</td>
</tr>
<tr>
<td>13:18:04.69</td>
<td>76</td>
<td>2</td>
<td>171.2</td>
<td>103.2</td>
<td>14.3</td>
<td>2495</td>
<td>25371</td>
<td>3842</td>
<td>1</td>
</tr>
<tr>
<td>13:48:05.21</td>
<td>77</td>
<td>4</td>
<td>146.2</td>
<td>109.0</td>
<td>19.3</td>
<td>2446</td>
<td>26529</td>
<td>2365</td>
<td>2</td>
</tr>
<tr>
<td>14:18:05.93</td>
<td>75</td>
<td>4</td>
<td>183.3</td>
<td>99.2</td>
<td>16.7</td>
<td>2730</td>
<td>23095</td>
<td>2396</td>
<td>2</td>
</tr>
<tr>
<td>14:48:05.62</td>
<td>79</td>
<td>4</td>
<td>162.1</td>
<td>118.0</td>
<td>15.8</td>
<td>2215</td>
<td>25507</td>
<td>3229</td>
<td>1</td>
</tr>
</tbody>
</table>

*GLOM

<table>
<thead>
<tr>
<th>Interval end</th>
<th>Online</th>
<th>Batch</th>
<th>% exec</th>
<th>% serv</th>
<th>% mon</th>
<th>I/Os</th>
<th>PMMEs</th>
<th>Free Pgs</th>
<th>ETMF</th>
</tr>
</thead>
<tbody>
<tr>
<td>14:48:05.62</td>
<td>79</td>
<td>4</td>
<td>166.1</td>
<td>105.0</td>
<td>16.9</td>
<td>2675</td>
<td>23824</td>
<td>3229</td>
<td>1</td>
</tr>
</tbody>
</table>

*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 | 740 | 1160 | 1580 | 2000 | 2420 | 2840 | 3260 | 3680 | 4000 |

Character +
Min hits 1
Max hits 1

#HIST ALL
#GLOM
"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

<table>
<thead>
<tr>
<th>(milliseconds)</th>
<th>0 to 1</th>
<th>2 to 2</th>
<th>3 to 5</th>
<th>6 to 10</th>
<th>11 to 20</th>
<th>21 to 50</th>
<th>51 to 100</th>
<th>101 to 200</th>
<th>201 to 500</th>
<th>501 to 1000</th>
<th>1001 to 2000</th>
<th>2001 and above</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>115033 (32 %)</td>
<td>22641 (6 %)</td>
<td>38336 (11 %)</td>
<td>45750 (13 %)</td>
<td>56290 (16 %)</td>
<td>60379 (17 %)</td>
<td>17647 (5 %)</td>
<td>553 (0 %)</td>
<td>19 (0 %)</td>
<td>0 (0 %)</td>
<td>1 (0 %)</td>
<td>2 (0 %)</td>
</tr>
</tbody>
</table>

---

356651
"Snap" histogram of user memory sizes

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

---

32671
"All" histogram of user memory sizes

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

"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

<table>
<thead>
<tr>
<th>(milliseconds)</th>
<th>--------</th>
<th>--------</th>
<th>--------</th>
<th>--------</th>
<th>--------</th>
<th>--------</th>
<th>--------</th>
<th>--------</th>
<th>--------</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 to 10</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>11 to 15</td>
<td>*********</td>
<td>*********</td>
<td>*********</td>
<td>*********</td>
<td>*********</td>
<td>*********</td>
<td>*********</td>
<td>*********</td>
<td>*********</td>
</tr>
<tr>
<td>16 to 25</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
</tr>
<tr>
<td>26 to 45</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>46 to 75</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
</tr>
<tr>
<td>76 to 130</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
</tr>
<tr>
<td>131 to 215</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
</tr>
<tr>
<td>216 to 360</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
</tr>
<tr>
<td>361 to 600</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
</tr>
<tr>
<td>601 to 1000</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
</tr>
<tr>
<td>1001 and above</td>
<td>*********</td>
<td>*********</td>
<td>*********</td>
<td>*********</td>
<td>*********</td>
<td>*********</td>
<td>*********</td>
<td>*********</td>
<td>*********</td>
</tr>
</tbody>
</table>

7854 (17 %) 12932 (28 %) 8858 (19 %) 2350 (5 %) 1892 (4 %) 1875 (4 %) 1547 (3 %) 922 (2 %) 399 (1 %) 345 (1 %) 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

<table>
<thead>
<tr>
<th></th>
<th>{all}</th>
<th>{snap}</th>
<th>{all}</th>
<th>{snap}</th>
</tr>
</thead>
<tbody>
<tr>
<td>% batch execution</td>
<td>97.8</td>
<td>114.3</td>
<td>ETMF</td>
<td>1</td>
</tr>
<tr>
<td>% batch service</td>
<td>25.4</td>
<td>35.8</td>
<td>90% response time</td>
<td>50</td>
</tr>
<tr>
<td>% online execution</td>
<td>12.1</td>
<td>29.9</td>
<td>I/O load factor</td>
<td>11</td>
</tr>
<tr>
<td>% online service</td>
<td>22.6</td>
<td>38.8</td>
<td>* of batch users</td>
<td>2</td>
</tr>
<tr>
<td>% ghost execution</td>
<td>23.4</td>
<td>21.8</td>
<td>* of online users</td>
<td>30</td>
</tr>
<tr>
<td>% ghost service</td>
<td>25.9</td>
<td>29.5</td>
<td>* of ghost users</td>
<td>22</td>
</tr>
<tr>
<td>% TP execution</td>
<td>0.0</td>
<td>0.0</td>
<td>* of TP users</td>
<td>1</td>
</tr>
<tr>
<td>% TP service</td>
<td>0.0</td>
<td>0.0</td>
<td>I/Os per minute</td>
<td>2023</td>
</tr>
<tr>
<td>% monitor execution</td>
<td>14.1</td>
<td>16.9</td>
<td>Schedules per minute</td>
<td>2665</td>
</tr>
<tr>
<td>% I/O wait</td>
<td>29.7</td>
<td>9.1</td>
<td>Interactions per min</td>
<td>242</td>
</tr>
<tr>
<td>% resource wait</td>
<td>0.0</td>
<td>0.0</td>
<td>Events per minute</td>
<td>4673</td>
</tr>
<tr>
<td>% I/O@resource wait</td>
<td>0.1</td>
<td>0.2</td>
<td>PMMEs per minute</td>
<td>19273</td>
</tr>
<tr>
<td>% true idle</td>
<td>47.0</td>
<td>2.0</td>
<td>Avg. usec per PMME</td>
<td>2318</td>
</tr>
<tr>
<td>Total</td>
<td>298.9</td>
<td>299.5</td>
<td>Minutes in interval</td>
<td>1475</td>
</tr>
<tr>
<td>Master CPU</td>
<td>Slave CPU 2</td>
<td>Slave CPU 3</td>
<td>Slave CPU 4</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>{all} {snap}</td>
<td>{all} {snap}</td>
<td>{all} {snap}</td>
<td>{all} {snap}</td>
<td></td>
</tr>
<tr>
<td>% user execution</td>
<td>34 45</td>
<td>0 0</td>
<td>0 0</td>
<td>49 60</td>
</tr>
<tr>
<td>% user service</td>
<td>30 39</td>
<td>0 0</td>
<td>0 0</td>
<td>21 33</td>
</tr>
<tr>
<td>% monitor exec.</td>
<td>10 12</td>
<td>0 0</td>
<td>0 0</td>
<td>2 2</td>
</tr>
<tr>
<td>% I/O wait</td>
<td>11 3</td>
<td>0 0</td>
<td>0 0</td>
<td>9 3</td>
</tr>
<tr>
<td>% resource wait</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>% I/O &amp; res wait</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>% true idle</td>
<td>11 1</td>
<td>0 0</td>
<td>0 0</td>
<td>17 1</td>
</tr>
<tr>
<td>% unaccountable</td>
<td>0 0</td>
<td>100 100</td>
<td>100 100</td>
<td>0 0</td>
</tr>
</tbody>
</table>

| Schedules/minute | 1426 1527 | 0 0 | 0 0 | 622 873 |
| Events/minute    | 3717 4362 | 0 0 | 0 0 | 480 652 |
| PMME starts/min. | 9895 11191 | 0 0 | 0 0 | 4724 6402 |
| PMME ends/min.   | 9897 11183 | 0 0 | 0 0 | 4724 6405 |

<table>
<thead>
<tr>
<th>Slave CPU 5</th>
<th>Slave CPU 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>{all} {snap}</td>
<td>{all} {snap}</td>
</tr>
<tr>
<td>% user execution</td>
<td>49 60</td>
</tr>
<tr>
<td>% user service</td>
<td>21 33</td>
</tr>
<tr>
<td>% monitor exec.</td>
<td>2 3</td>
</tr>
<tr>
<td>% I/O wait</td>
<td>9 3</td>
</tr>
<tr>
<td>% resource wait</td>
<td>0 0</td>
</tr>
<tr>
<td>% I/O &amp; res wait</td>
<td>0 0</td>
</tr>
<tr>
<td>% true idle</td>
<td>17 1</td>
</tr>
<tr>
<td>% unaccountable</td>
<td>0 0</td>
</tr>
</tbody>
</table>

| 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**

<table>
<thead>
<tr>
<th>(Resource name)</th>
<th>{ # in }</th>
<th>{---since system boot---}</th>
<th>{ Total }</th>
</tr>
</thead>
<tbody>
<tr>
<td>{use now}</td>
<td></td>
<td>(max) (min) (average)</td>
<td>{available}</td>
</tr>
</tbody>
</table>

- **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)

<table>
<thead>
<tr>
<th></th>
<th>Attempted Gets</th>
<th>Hits UC=0</th>
<th>Hits UC&gt;0</th>
<th>Percent Hits</th>
<th>Attempted Puts</th>
<th>Failed Puts</th>
<th>Unused Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAD</td>
<td>36</td>
<td>35</td>
<td>0</td>
<td>99</td>
<td>0</td>
<td>0</td>
<td>15 {all}</td>
</tr>
<tr>
<td></td>
<td>77</td>
<td>77</td>
<td>0</td>
<td>99</td>
<td>0</td>
<td>0</td>
<td>15 {snap}</td>
</tr>
<tr>
<td>PAD</td>
<td>13</td>
<td>13</td>
<td>0</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>11 {all}</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>6</td>
<td>0</td>
<td>90</td>
<td>1</td>
<td>0</td>
<td>22 {snap}</td>
</tr>
<tr>
<td>GP</td>
<td>73</td>
<td>65</td>
<td>7</td>
<td>99</td>
<td>0</td>
<td>0</td>
<td>26 {all}</td>
</tr>
<tr>
<td></td>
<td>115</td>
<td>104</td>
<td>11</td>
<td>99</td>
<td>0</td>
<td>0</td>
<td>18 {snap}</td>
</tr>
<tr>
<td>FD</td>
<td>500</td>
<td>485</td>
<td>4</td>
<td>97</td>
<td>11</td>
<td>0</td>
<td>175 {all}</td>
</tr>
<tr>
<td></td>
<td>528</td>
<td>514</td>
<td>6</td>
<td>98</td>
<td>9</td>
<td>0</td>
<td>163 {snap}</td>
</tr>
<tr>
<td>FIT</td>
<td>426</td>
<td>276</td>
<td>7</td>
<td>66</td>
<td>143</td>
<td>0</td>
<td>710 {all}</td>
</tr>
<tr>
<td></td>
<td>671</td>
<td>493</td>
<td>13</td>
<td>75</td>
<td>166</td>
<td>0</td>
<td>233 {snap}</td>
</tr>
<tr>
<td>UL</td>
<td>40</td>
<td>36</td>
<td>3</td>
<td>97</td>
<td>2</td>
<td>0</td>
<td>30 {all}</td>
</tr>
<tr>
<td></td>
<td>63</td>
<td>54</td>
<td>7</td>
<td>97</td>
<td>3</td>
<td>0</td>
<td>50 {snap}</td>
</tr>
<tr>
<td>INDEX</td>
<td>197</td>
<td>160</td>
<td>8</td>
<td>85</td>
<td>41</td>
<td>0</td>
<td>99 {all}</td>
</tr>
<tr>
<td></td>
<td>271</td>
<td>210</td>
<td>12</td>
<td>81</td>
<td>67</td>
<td>0</td>
<td>290 {snap}</td>
</tr>
<tr>
<td>DATA</td>
<td>824</td>
<td>605</td>
<td>43</td>
<td>78</td>
<td>263</td>
<td>0</td>
<td>349 {all}</td>
</tr>
<tr>
<td></td>
<td>1274</td>
<td>882</td>
<td>55</td>
<td>73</td>
<td>445</td>
<td>0</td>
<td>1698 {snap}</td>
</tr>
<tr>
<td>REL</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>49</td>
<td>0</td>
<td>0</td>
<td>0 {all}</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 {snap}</td>
</tr>
<tr>
<td>CONSEC</td>
<td>135</td>
<td>80</td>
<td>0</td>
<td>59</td>
<td>133</td>
<td>0</td>
<td>1099 {all}</td>
</tr>
<tr>
<td></td>
<td>277</td>
<td>125</td>
<td>1</td>
<td>45</td>
<td>343</td>
<td>0</td>
<td>41 {snap}</td>
</tr>
<tr>
<td>ELSE</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>23</td>
<td>0</td>
<td>0 {all}</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>40</td>
<td>0</td>
<td>0 {snap}</td>
</tr>
<tr>
<td>Total</td>
<td>2266</td>
<td>1757</td>
<td>73</td>
<td>80</td>
<td>617</td>
<td>0</td>
<td>2514 {all}</td>
</tr>
<tr>
<td></td>
<td>3324</td>
<td>2469</td>
<td>105</td>
<td>77</td>
<td>1074</td>
<td>0</td>
<td>2530 {snap}</td>
</tr>
</tbody>
</table>
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
<table>
<thead>
<tr>
<th>Device name</th>
<th># of connects</th>
<th>connects per min.</th>
<th>% idle</th>
<th>% wait</th>
<th>% busy</th>
<th>% backlog</th>
<th>load factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC010000</td>
<td>3773</td>
<td>11</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0 {snap}</td>
</tr>
<tr>
<td></td>
<td>12537</td>
<td></td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0 {all}</td>
</tr>
<tr>
<td>DC010000</td>
<td>145124</td>
<td>439</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0 {snap}</td>
</tr>
<tr>
<td></td>
<td>359950</td>
<td>243</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0 {all}</td>
</tr>
<tr>
<td>DC020000</td>
<td>284827</td>
<td>861</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0 {snap}</td>
</tr>
<tr>
<td></td>
<td>715820</td>
<td>484</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0 {all}</td>
</tr>
<tr>
<td>DP010000</td>
<td>98149</td>
<td>296</td>
<td>85.8</td>
<td>0.1</td>
<td>11.7</td>
<td>2.4</td>
<td>17.5 {snap}</td>
</tr>
<tr>
<td></td>
<td>283045</td>
<td>191</td>
<td>91.4</td>
<td>0.0</td>
<td>7.2</td>
<td>1.4</td>
<td>16.6 {all}</td>
</tr>
<tr>
<td>DP020000</td>
<td>146134</td>
<td>442</td>
<td>78.8</td>
<td>0.1</td>
<td>16.4</td>
<td>4.7</td>
<td>22.5 {snap}</td>
</tr>
<tr>
<td></td>
<td>342938</td>
<td>232</td>
<td>89.1</td>
<td>0.0</td>
<td>8.6</td>
<td>2.2</td>
<td>20.7 {all}</td>
</tr>
<tr>
<td>DP030000</td>
<td>163738</td>
<td>495</td>
<td>77.3</td>
<td>0.1</td>
<td>16.7</td>
<td>5.9</td>
<td>26.5 {snap}</td>
</tr>
<tr>
<td></td>
<td>391650</td>
<td>265</td>
<td>88.1</td>
<td>0.1</td>
<td>9.8</td>
<td>2.1</td>
<td>17.7 {all}</td>
</tr>
<tr>
<td>DP040000</td>
<td>12</td>
<td>0</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.7 {snap}</td>
</tr>
<tr>
<td></td>
<td>3273</td>
<td></td>
<td>99.9</td>
<td>0.0</td>
<td>0.1</td>
<td>0.1</td>
<td>58.0 {all}</td>
</tr>
<tr>
<td>DP050000</td>
<td>10615</td>
<td>32</td>
<td>97.9</td>
<td>0.0</td>
<td>1.5</td>
<td>0.7</td>
<td>32.1 {snap}</td>
</tr>
<tr>
<td></td>
<td>24654</td>
<td>16</td>
<td>99.1</td>
<td>0.0</td>
<td>0.7</td>
<td>0.2</td>
<td>21.6 {all}</td>
</tr>
<tr>
<td>DP060000</td>
<td>11227</td>
<td>33</td>
<td>97.8</td>
<td>0.0</td>
<td>1.5</td>
<td>0.6</td>
<td>29.7 {snap}</td>
</tr>
<tr>
<td></td>
<td>25289</td>
<td>17</td>
<td>99.0</td>
<td>0.0</td>
<td>0.8</td>
<td>0.2</td>
<td>19.1 {all}</td>
</tr>
<tr>
<td>DP080000</td>
<td>49</td>
<td>0</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.1 {snap}</td>
</tr>
<tr>
<td></td>
<td>398</td>
<td></td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0 {all}</td>
</tr>
<tr>
<td>DP090000</td>
<td>15</td>
<td>0</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.5 {snap}</td>
</tr>
<tr>
<td></td>
<td>1446</td>
<td></td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.9 {all}</td>
</tr>
<tr>
<td>DP110000</td>
<td>12</td>
<td>0</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0 {snap}</td>
</tr>
<tr>
<td></td>
<td>2692</td>
<td></td>
<td>99.9</td>
<td>0.0</td>
<td>0.1</td>
<td>0.0</td>
<td>0.7 {all}</td>
</tr>
<tr>
<td>DC030000</td>
<td>210720</td>
<td>637</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0 {snap}</td>
</tr>
<tr>
<td></td>
<td>875570</td>
<td>593</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0 {all}</td>
</tr>
<tr>
<td>DC040000</td>
<td>210754</td>
<td>637</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0 {snap}</td>
</tr>
<tr>
<td></td>
<td>874614</td>
<td>592</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0 {all}</td>
</tr>
<tr>
<td>DP210000</td>
<td>12000</td>
<td>36</td>
<td>96.6</td>
<td>0.0</td>
<td>1.3</td>
<td>0.0</td>
<td>2.7 {snap}</td>
</tr>
<tr>
<td></td>
<td>31481</td>
<td>21</td>
<td>99.2</td>
<td>0.0</td>
<td>0.8</td>
<td>0.0</td>
<td>3.6 {all}</td>
</tr>
<tr>
<td>DP230000</td>
<td>2503</td>
<td>7</td>
<td>99.7</td>
<td>0.0</td>
<td>0.3</td>
<td>0.0</td>
<td>7.0 {snap}</td>
</tr>
<tr>
<td></td>
<td>6620</td>
<td>4</td>
<td>99.8</td>
<td>0.0</td>
<td>0.2</td>
<td>0.0</td>
<td>4.0 {all}</td>
</tr>
<tr>
<td>DP250000</td>
<td>35786</td>
<td>108</td>
<td>94.6</td>
<td>0.0</td>
<td>5.2</td>
<td>0.1</td>
<td>2.8 {snap}</td>
</tr>
<tr>
<td></td>
<td>160651</td>
<td>108</td>
<td>94.8</td>
<td>0.0</td>
<td>4.9</td>
<td>0.3</td>
<td>6.6</td>
</tr>
<tr>
<td>---</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>DP270000</td>
<td>4945</td>
<td>14</td>
<td>99.3</td>
<td>0.0</td>
<td>0.7</td>
<td>0.0</td>
<td>5.6</td>
</tr>
<tr>
<td></td>
<td>11926</td>
<td>8</td>
<td>99.6</td>
<td>0.0</td>
<td>0.4</td>
<td>0.0</td>
<td>3.8</td>
</tr>
<tr>
<td>DP290000</td>
<td>55102</td>
<td>166</td>
<td>93.7</td>
<td>0.1</td>
<td>6.2</td>
<td>0.0</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>237124</td>
<td>160</td>
<td>94.2</td>
<td>0.0</td>
<td>5.7</td>
<td>0.0</td>
<td>1.5</td>
</tr>
<tr>
<td>DP310000</td>
<td>7461</td>
<td>22</td>
<td>99.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>30029</td>
<td>20</td>
<td>99.2</td>
<td>0.0</td>
<td>0.7</td>
<td>0.1</td>
<td>9.1</td>
</tr>
<tr>
<td>DP330000</td>
<td>42301</td>
<td>127</td>
<td>94.8</td>
<td>0.0</td>
<td>4.4</td>
<td>0.8</td>
<td>16.3</td>
</tr>
<tr>
<td></td>
<td>82900</td>
<td>56</td>
<td>97.8</td>
<td>0.0</td>
<td>1.9</td>
<td>0.3</td>
<td>14.2</td>
</tr>
<tr>
<td>DP350000</td>
<td>72</td>
<td>0</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>43.2</td>
</tr>
<tr>
<td></td>
<td>346</td>
<td>0</td>
<td>99.7</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>97.1</td>
</tr>
<tr>
<td>DP370000</td>
<td>22783</td>
<td>68</td>
<td>97.0</td>
<td>0.0</td>
<td>2.8</td>
<td>0.3</td>
<td>9.4</td>
</tr>
<tr>
<td></td>
<td>59977</td>
<td>40</td>
<td>98.3</td>
<td>0.0</td>
<td>1.6</td>
<td>0.1</td>
<td>7.3</td>
</tr>
<tr>
<td>DP390000</td>
<td>59</td>
<td>0</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>171</td>
<td>0</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.8</td>
</tr>
<tr>
<td>DP410000</td>
<td>4428</td>
<td>13</td>
<td>99.5</td>
<td>0.0</td>
<td>0.4</td>
<td>0.0</td>
<td>4.3</td>
</tr>
<tr>
<td></td>
<td>8366</td>
<td>5</td>
<td>99.8</td>
<td>0.0</td>
<td>0.2</td>
<td>0.0</td>
<td>4.1</td>
</tr>
<tr>
<td>DP430000</td>
<td>14490</td>
<td>43</td>
<td>98.6</td>
<td>0.0</td>
<td>1.3</td>
<td>0.1</td>
<td>8.0</td>
</tr>
<tr>
<td></td>
<td>35147</td>
<td>23</td>
<td>99.2</td>
<td>0.0</td>
<td>0.7</td>
<td>0.1</td>
<td>7.2</td>
</tr>
<tr>
<td>DP450000</td>
<td>10696</td>
<td>32</td>
<td>98.8</td>
<td>0.0</td>
<td>1.2</td>
<td>0.1</td>
<td>4.7</td>
</tr>
<tr>
<td></td>
<td>22040</td>
<td>14</td>
<td>99.4</td>
<td>0.0</td>
<td>0.6</td>
<td>0.0</td>
<td>4.1</td>
</tr>
<tr>
<td>DP470000</td>
<td>7148</td>
<td>21</td>
<td>99.2</td>
<td>0.0</td>
<td>0.7</td>
<td>0.0</td>
<td>5.9</td>
</tr>
<tr>
<td></td>
<td>34928</td>
<td>23</td>
<td>99.2</td>
<td>0.0</td>
<td>0.8</td>
<td>0.0</td>
<td>4.6</td>
</tr>
<tr>
<td>DP490000</td>
<td>32440</td>
<td>98</td>
<td>96.1</td>
<td>0.0</td>
<td>3.5</td>
<td>0.4</td>
<td>10.7</td>
</tr>
<tr>
<td></td>
<td>114148</td>
<td>77</td>
<td>96.8</td>
<td>0.0</td>
<td>2.7</td>
<td>0.6</td>
<td>18.0</td>
</tr>
<tr>
<td>DP510000</td>
<td>169260</td>
<td>512</td>
<td>79.6</td>
<td>0.1</td>
<td>19.8</td>
<td>0.4</td>
<td>2.7</td>
</tr>
<tr>
<td></td>
<td>914330</td>
<td>619</td>
<td>77.3</td>
<td>0.2</td>
<td>22.1</td>
<td>0.4</td>
<td>2.5</td>
</tr>
<tr>
<td>TC010000</td>
<td>26134</td>
<td>79</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>138807</td>
<td>94</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>MT030000</td>
<td>12091</td>
<td>36</td>
<td>97.7</td>
<td>0.0</td>
<td>2.3</td>
<td>0.0</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>51114</td>
<td>34</td>
<td>98.0</td>
<td>0.0</td>
<td>1.9</td>
<td>0.0</td>
<td>2.7</td>
</tr>
<tr>
<td>MT040000</td>
<td>13486</td>
<td>40</td>
<td>97.4</td>
<td>0.0</td>
<td>2.6</td>
<td>0.0</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>68136</td>
<td>46</td>
<td>96.4</td>
<td>0.1</td>
<td>3.5</td>
<td>0.0</td>
<td>2.6</td>
</tr>
<tr>
<td>MT050000</td>
<td>557</td>
<td>1</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>5.7</td>
</tr>
<tr>
<td></td>
<td>19002</td>
<td>12</td>
<td>99.8</td>
<td>0.0</td>
<td>0.2</td>
<td>0.0</td>
<td>3.8</td>
</tr>
<tr>
<td>UC010000</td>
<td>1335</td>
<td>4</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Code</td>
<td>Value1</td>
<td>Value2</td>
<td>Value3</td>
<td>Value4</td>
<td>Value5</td>
<td>Value6</td>
<td>Value7</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>7504</td>
<td>5</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>LP010000</td>
<td>1002</td>
<td>3</td>
<td>86.8</td>
<td>0.2</td>
<td>13.0</td>
<td>0.0</td>
<td>1.8</td>
</tr>
<tr>
<td></td>
<td>3666</td>
<td>2</td>
<td>89.5</td>
<td>0.2</td>
<td>10.3</td>
<td>0.0</td>
<td>2.0</td>
</tr>
<tr>
<td>LP020000</td>
<td>333</td>
<td>1</td>
<td>95.7</td>
<td>0.0</td>
<td>4.3</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>333</td>
<td>2</td>
<td>89.3</td>
<td>0.3</td>
<td>10.4</td>
<td>0.0</td>
<td>2.5</td>
</tr>
<tr>
<td>UCO30000</td>
<td>323</td>
<td>0</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>323</td>
<td>0</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>LP100000</td>
<td>323</td>
<td>0</td>
<td>95.7</td>
<td>0.1</td>
<td>4.2</td>
<td>0.0</td>
<td>1.2</td>
</tr>
<tr>
<td></td>
<td>323</td>
<td>0</td>
<td>99.0</td>
<td>0.0</td>
<td>0.9</td>
<td>0.0</td>
<td>1.2</td>
</tr>
<tr>
<td>IOM-chan number</td>
<td># of connects</td>
<td>connects per min.</td>
<td>% idle</td>
<td>% wait</td>
<td>% busy</td>
<td>% backlog</td>
<td>load factor</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------</td>
<td>------------------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>-----------</td>
<td>------------</td>
</tr>
<tr>
<td>0-08</td>
<td>35869</td>
<td>108</td>
<td>95.0</td>
<td>0.0</td>
<td>5.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>89471</td>
<td>60</td>
<td>97.3</td>
<td>0.0</td>
<td>2.7</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0-09</td>
<td>37009</td>
<td>111</td>
<td>94.9</td>
<td>0.0</td>
<td>5.1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>90966</td>
<td>61</td>
<td>97.3</td>
<td>0.0</td>
<td>2.7</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0-10</td>
<td>34899</td>
<td>105</td>
<td>95.0</td>
<td>0.0</td>
<td>5.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>88389</td>
<td>59</td>
<td>97.3</td>
<td>0.0</td>
<td>2.7</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0-11</td>
<td>37347</td>
<td>112</td>
<td>94.9</td>
<td>0.0</td>
<td>5.1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>91120</td>
<td>61</td>
<td>97.3</td>
<td>0.0</td>
<td>2.7</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0-12</td>
<td>36444</td>
<td>110</td>
<td>94.8</td>
<td>0.0</td>
<td>5.2</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>90643</td>
<td>61</td>
<td>97.2</td>
<td>0.0</td>
<td>2.8</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0-13</td>
<td>35007</td>
<td>105</td>
<td>94.8</td>
<td>0.0</td>
<td>5.2</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>88993</td>
<td>60</td>
<td>97.2</td>
<td>0.0</td>
<td>2.8</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0-14</td>
<td>35063</td>
<td>106</td>
<td>94.9</td>
<td>0.0</td>
<td>5.1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>89020</td>
<td>60</td>
<td>97.3</td>
<td>0.0</td>
<td>2.7</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0-15</td>
<td>36184</td>
<td>109</td>
<td>94.8</td>
<td>0.0</td>
<td>5.2</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>89513</td>
<td>60</td>
<td>97.2</td>
<td>0.0</td>
<td>2.8</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0-16</td>
<td>13068</td>
<td>39</td>
<td>97.5</td>
<td>0.0</td>
<td>2.5</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>69413</td>
<td>47</td>
<td>97.3</td>
<td>0.0</td>
<td>2.7</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0-17</td>
<td>13066</td>
<td>39</td>
<td>97.5</td>
<td>0.0</td>
<td>2.5</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>69394</td>
<td>47</td>
<td>97.2</td>
<td>0.0</td>
<td>2.8</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0-20</td>
<td>52643</td>
<td>159</td>
<td>93.7</td>
<td>0.0</td>
<td>6.3</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>218965</td>
<td>148</td>
<td>94.5</td>
<td>0.0</td>
<td>5.5</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0-21</td>
<td>52661</td>
<td>159</td>
<td>93.7</td>
<td>0.0</td>
<td>6.3</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>218562</td>
<td>148</td>
<td>94.5</td>
<td>0.0</td>
<td>5.5</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0-22</td>
<td>52695</td>
<td>159</td>
<td>93.7</td>
<td>0.0</td>
<td>6.3</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>218754</td>
<td>148</td>
<td>94.5</td>
<td>0.0</td>
<td>5.5</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0-23</td>
<td>52756</td>
<td>159</td>
<td>93.8</td>
<td>0.0</td>
<td>6.2</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>218812</td>
<td>148</td>
<td>94.5</td>
<td>0.0</td>
<td>5.5</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0-24</td>
<td>1002</td>
<td>3</td>
<td>81.9</td>
<td>0.0</td>
<td>18.1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>3666</td>
<td>2</td>
<td>86.6</td>
<td>0.0</td>
<td>13.4</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0-25</td>
<td>333</td>
<td>1</td>
<td>95.1</td>
<td>0.0</td>
<td>4.9</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>3838</td>
<td>2</td>
<td>88.0</td>
<td>0.0</td>
<td>12.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0-28</td>
<td>323</td>
<td>0</td>
<td>95.5</td>
<td>0.0</td>
<td>4.5</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>323</td>
<td>0</td>
<td>99.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>---</td>
<td>------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>0-30</td>
<td>3773</td>
<td>11</td>
<td>81.3</td>
<td>0.0</td>
<td>18.7</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>12537</td>
<td>8</td>
<td>86.2</td>
<td>0.0</td>
<td>13.8</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1-12</td>
<td>35722</td>
<td>108</td>
<td>94.8</td>
<td>0.0</td>
<td>5.2</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>89816</td>
<td>60</td>
<td>97.2</td>
<td>0.0</td>
<td>2.8</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1-13</td>
<td>35882</td>
<td>108</td>
<td>94.9</td>
<td>0.0</td>
<td>5.1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>90038</td>
<td>61</td>
<td>97.2</td>
<td>0.0</td>
<td>2.8</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1-14</td>
<td>35451</td>
<td>107</td>
<td>94.9</td>
<td>0.0</td>
<td>5.1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>89428</td>
<td>60</td>
<td>97.2</td>
<td>0.0</td>
<td>2.8</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1-15</td>
<td>35074</td>
<td>106</td>
<td>94.9</td>
<td>0.0</td>
<td>5.1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>88369</td>
<td>59</td>
<td>97.3</td>
<td>0.0</td>
<td>2.7</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1-20</td>
<td>52686</td>
<td>159</td>
<td>93.7</td>
<td>0.0</td>
<td>6.3</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>218892</td>
<td>148</td>
<td>94.5</td>
<td>0.0</td>
<td>5.5</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1-21</td>
<td>52730</td>
<td>159</td>
<td>93.7</td>
<td>0.0</td>
<td>6.3</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>219151</td>
<td>148</td>
<td>94.5</td>
<td>0.0</td>
<td>5.5</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1-22</td>
<td>52713</td>
<td>159</td>
<td>93.7</td>
<td>0.0</td>
<td>6.3</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>218547</td>
<td>148</td>
<td>94.4</td>
<td>0.0</td>
<td>5.6</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1-23</td>
<td>52590</td>
<td>159</td>
<td>93.7</td>
<td>0.0</td>
<td>6.3</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>218501</td>
<td>148</td>
<td>94.5</td>
<td>0.0</td>
<td>5.5</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>
NETCON

- NEW IN COO
  - REPLACES FEPCON
  - REPLACES PART OF FEPANLZ

- INITIALLY USED IN $XINSTALL
  - DEFINE NODES
  - DEFINED DEFAULTS

12-1A INT 8/85
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)

12-1B INT 8/85


BOOT INFO

- MONITOR
  - LFEP
  - ----
  - M:FEP

- LIBRARY ACCOUNT
  - :SYS

- HANDLERS
  - #1-> NODEADMIN
  - #2-> COUPLER
  - NODEADMIN
  - HDLCX25

12-2 INT 8/85
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

12-4 INT 8/85
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

12-6 INT 8/85
NETWORKING (RFEPS)

- MULTIPLE PROCESSORS
  - NETCON
  - SUPER
  - PIGETTE

12-7A INT 8/85
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

12-7B INT 8/85
1 - 1.000 !ECHO
2 - 1.500 !"REPORT STEP=FULL
3 - 2.000 !"M DINGO ghost starting using !XEQ $XINSTALL..SYS
4 - 3.000 !"SET MSLL ME
5 - 4.000 !" File: $XINSTALL..SYS (C00 version)
6 - 5.000 !"
7 - 6.000 !" This file is XEQ'd by the system DINGO ghost after tape boots
8 - 7.000 !" with the command:
9 - 8.000 !" !XEQ $XINSTALL..SYS BOOTFLAG=nn
10 - 9.000 !"
11 - 10.000 !" where nn=
12 - 11.000 !" 02 for tape boot, Y to new file system
13 - 12.000 !" 03 for tape boot, S to new file system
14 - 13.000 !" 04 for tape boot, N to new file system
15 - 14.000 !" 05 for tape boot, S to new file system (with reconstruct)
16 - 15.000 !" 06 for tape boot, N to new file system (with reconstruct)
17 - 16.000 !" 07 for disk boot
18 - 17.000 !" 08 for disk boot (with reconstruct)
19 - 18.000 !" 09 for recovery
20 - 19.000 !" 10 for recovery (lost dump file(?))
21 - 20.000 !" 11 for operator recovery (from tape)
22 - 21.000 !"
23 - 22.000 !IF BOOTFLAG=11 THEN M DINGO is running after a tape-initiated OR
24 - 23.000 !IF BOOTFLAG=10 THEN M DINGO is running after a recovery (DUMPF)
25 - 24.000 !IF BOOTFLAG=9 THEN M DINGO is running after a recovery
26 - 25.000 !IF BOOTFLAG=8 THEN M DINGO is running after a disk boot, reconstruct
27 - 26.000 !IF BOOTFLAG=7 THEN M DINGO is running after a disk boot
28 - 27.000 !IF BOOTFLAG=6 THEN M DINGO is running after a TAPE boot, N to new files, reconstruct
29 - 28.000 !IF BOOTFLAG=5 THEN M DINGO is running after a TAPE boot, S to new files, reconstruct
30 - 29.000 !IF BOOTFLAG=4 THEN M DINGO is running after a TAPE boot, N to new files
31 - 30.000 !IF BOOTFLAG=3 THEN M DINGO is running after a TAPE boot, S to new files
32 - 31.000 !IF BOOTFLAG=2 THEN M DINGO is running after a TAPE boot, Y to new files
33 - 32.000 !PRIV ALL
34 - 33.000 !"
35 - 34.000 !" THIS PART GETS EXECUTED ON EVERY SYSTEM STARTUP
36 - 35.000 !"
37 - 36.000 !" ********************************************
38 -  37.000 !" NOTE THAT PACKSETS OTHER THAN SYS MAY NOT BE UP AT THE TIME
39 -  38.000 !" THIS IS RUN SO YOU SHOULDN'T DEPEND ON BEING ABLE TO GET TO
40 -  39.000 !" FILES OTHER THAN THOSE ON #SYS!
41 -  40.000 !"******************************************************************************
42 -  41.000 !"
43 -  42.000 !PCL
44 -  43.000 MOD TP_CNTRL_D::SYS TO (ACS+((TPA?, TPC?), WNEW, READ, UPD))
45 -  44.000 " Andrew and J.L., this is for you. Love, JJ
46 -  45.000 MOD :JOBSTATS::SYS(SH) TO (ACC((SYS), READ, DELR, WNEW, UPD, DELF, REATTR),
47 -  46.000 ACC+((?), EXEC),
48 -  47.000 ACS+((IBEX), READ, WNEW, UPD),
49 -  48.000 ACS+((A.X), READ),
50 -  49.000 ACS+((SYMO), READ),
51 -  50.000 ACS+((PEEK.X), READ),
52 -  51.000 ACS+((BASS), READ),
53 -  52.000 ACS+((FILER.X), READ),
54 -  53.000 ACS+((MOM.X), READ),
55 -  54.000 ACS+((TERM), READ),
56 -  55.000 ACS+((WOODPECKER), READ),
57 -  56.000 ACS+((RQ.X), READ)
58 -  57.000 ENO
59 -  58.000 !"
60 -  59.000 !"
61 -  60.000 !" Special step: lets you create a file in :SYS that gets XEQd
62 -  61.000 !" whenever this job does. For example, on the LADC L66B machine,
63 -  62.000 !" since that machine has 3270's configured, the file $INSTALL_LOCAL
64 -  63.000 !" contains a line like:
65 -  64.000 !"
66 -  65.000 !" IC :SHARED_LCP6_RELEASE::SYS OVER :SHARED_LCP6_SYSTEM::SYS
67 -  66.000 !"
68 -  67.000 !" that the LADC L66A doesn't need.
69 -  68.000 !"
70 -  69.000 !" note that when this file runs, there is no guarantee that packsets
71 -  70.000 !" other than OP#SYS are mounted, so don't expect anything extra-fancy
72 -  71.000 !" to work here. Also, you may find that changing user-maximums and
73 -  72.000 !" other CONTROLLable values may not work as you think they might.
74 -  73.000 !"
75 -  74.000 !" This job is primarily to guarantee that a system comes up right!
76 -  75.000 !" It performs LADC-defined specific tasks, that are mostly support
77 - 76.000 " oriented. If this job or $XINSTALL_LOCAL can be adapted to do what
78 - 77.000 " you want, that's great. If not: Sorry, Charlie.
79 - 79.000 "
80 - 79.000 !IF SFID_EXIST("$XINSTALL_LOCAL::SYS") THEN XEQ $XINSTALL_LOCAL::SYS THISBOOT=BOOTFLAG
81 - 80.000 !IF BOOTFLAG>6 THEN GOTO END_OF_TAPE_BOOT
82 - 81.000 "
83 - 82.000 " ****************************
84 - 83.000 " The stuff in this section only gets EXECUTED after a tape boot
85 - 84.000 " ****************************
86 - 85.000 !TAPE_BOOT_ONLY:
87 - 86.000 "
88 - 87.000 " This section copies the system authorization files over themselves
89 - 88.000 " so they get their upper_level key granules rebuilt and data space
90 - 89.000 " recovered. You may not notice it, but average logon time drops
91 - 90.000 " dramatically if the authorization files aren't fragmented. Also,
92 - 91.000 " the COO:USERS records are fixed size, so changing a password or
93 - 92.000 " profile allows the record to be rewritten IN PLACE, also getting
94 - 93.000 " rid of a primary source of fragmentation.
95 - 94.000 "
96 - 95.000 !IF SFID_EXIST("::RATES::SYS") THEN CA :RATES::SYS OVER ::SYS
97 - 96.000 !IF SFID_EXIST("::PRO::SYS") THEN CA :PRO::SYS OVER ::SYS
98 - 97.000 !IF SFID_EXIST("::WSN::SYS") THEN CA :WSN::SYS OVER ::SYS
99 - 98.000 !IF SFID_EXIST("::OSYM::SYS") THEN CA :OSYM::SYS OVER ::SYS
100 - 99.000 !IF SFID_EXIST("::NETCON::SYS") THEN CA :NETCON::SYS OVER ::SYS
101 - 100.000 !IF SFID_EXIST("::FORM::SYS") THEN CA :FORM::SYS OVER ::SYS
102 - 101.000 !IF SFID_EXIST("::HLP::SYS") THEN CA :HLP::SYS OVER ::SYS
103 - 102.000 !IF SFID_EXIST("::USERS::SYS") THEN CA :USERS::SYS OVER ::SYS
104 - 103.000 !IF SFID_EXIST("::NAME::MAIL") THEN CA :NAME::MAIL OVER ::MAIL
105 - 104.000 !IF SFID_EXIST("::MAILCENTRAL::MAIL") THEN CA :MAILCENTRAL::MAIL OVER ::MAIL
106 - 105.000 !IF SFID_EXIST("::PATCH::SYS") THEN CA :PATCH::SYS OVER ::SYS(LN\C)
107 - 106.000 !IF SFID_EXIST("::RUM::SYS") THEN CA :RUM::SYS OVER ::SYS(LN\C)
108 - 107.000 !LET PATCHFILE="PF"||$VERSION||"::SYSTAC"
109 - 108.000 !IF SFID_EXIST("::PLOVER::SYS") THEN GOTO SKIP_PLOVERING
110 - 109.000 " Remove OUTPUT INTO/ ECHO/ KILL RUM/ and ALIB RET from :PLOVER.
111 - 110.000 !C :PLOVER::SYS(3-(SFID_REGS("::PLOVER::SYS") - 2)) OVER ::PLOVER::SYS(LN\C)
112 - 111.000 !IF STEPCC=0 THEN DEL :PLOVER::SYS
113 - 112.000 !PLOVER /ME OVER /*PLOVERFILE
114 - 113.000 READ ::PLOVER::SYS
115 - 114.000 END
116 - 115.000 !IF $FID_EXIST('PATCHFILE') THEN C %PATCHFILE,PLOVERFILE OVER %PATCHFILE(LN,C)1200-4
117 - 116.000 !GOTO PLOVER OR NOT
118 - 117.000 !SKIP_PLOVERING: IF $FID_EXIST('PATCHFILE') THEN CA %PATCHFILE OVER ..SYSTAC(LN,C)
119 - 118.000 !PLOVER OR NOT: LET DELETE PATCHFILE
120 - 119.000 !" Dingo says: this boot/dumpfile revision is $VERSION
121 - 120.000 !"
122 - 121.000 !" build some default stuff on Y to new file system boot
123 - 122.000 !" and/or we've lost :NETCON
124 - 123.000 !"
125 - 124.000 !IF $FID_EXIST('NETCON,SYS') THEN GOTO SKIP_NETCON_STEP
126 - 125.000 !NETCON
127 - 126.000 DEF NODE=0,NAME=HOST,TYPE=ME
128 - 127.000 DEF NODE=1,TYPE=FEP
129 - 128.000 DEF NODE=2,TYPE=FEP
130 - 129.000 DEF NODE=3,TYPE=FEP
131 - 130.000 DEF NODE=4,TYPE=FEP
132 - 131.000 DEF NODE=5,TYPE=FEP
133 - 132.000 DEF NODE=6,TYPE=FEP
134 - 133.000 DEF NODE=7,TYPE=FEP
135 - 134.000 DEF NODE=8,TYPE=FEP
136 - 135.000 DEF NODE=9,TYPE=FEP
137 - 136.000 DEF NODE=10,TYPE=FEP
138 - 137.000 DEF NODE=11,TYPE=FEP
139 - 138.000 DEF NODE=12,TYPE=FEP
140 - 139.000 DEF NODE=13,TYPE=FEP
141 - 140.000 DEF NODE=14,TYPE=FEP
142 - 141.000 DEF NODE=15,TYPE=FEP
143 - 142.000 DEF NODE=16,TYPE=FEP
144 - 143.000 SEL N=1
145 - 144.000 DEFAULT ASYNC PRO="DFLPRE"  
146 - 145.000 SEL N=2
147 - 146.000 DEFAULT ASYNC PRO="DFLPRE"
148 - 147.000 SEL N=3
149 - 148.000 DEFAULT ASYNC PRO="DFLPRE"
150 - 149.000 SEL N=4
151 - 150.000 DEFAULT ASYNC PRO="DFLPRE"
152 - 151.000 SEL N=5
153 - 152.000 DEFAULT ASYNC PRO="DFLPRE"
154 - 153.000 SEL N=6
155 - 154.000 DEFAULT SYNC PRO="DFLPRE"
156 - 155.000 SEL N=7
157 - 156.000 DEFAULT SYNC PRO="DFLPRE"
158 - 157.000 SEL N=8
159 - 158.000 DEFAULT SYNC PRO="DFLPRE"
160 - 159.000 SEL N=9
161 - 160.000 DEFAULT SYNC PRO="DFLPRE"
162 - 161.000 SEL N=10
163 - 162.000 DEFAULT SYNC PRO="DFLPRE"
164 - 163.000 SEL N=11
165 - 164.000 DEFAULT SYNC PRO="DFLPRE"
166 - 165.000 SEL N=12
167 - 166.000 DEFAULT SYNC PRO="DFLPRE"
168 - 167.000 SEL N=13
169 - 168.000 DEFAULT SYNC PRO="DFLPRE"
170 - 169.000 SEL N=14
171 - 170.000 DEFAULT SYNC PRO="DFLPRE"
172 - 171.000 SEL N=15
173 - 172.000 DEFAULT SYNC PRO="DFLPRE"
174 - 173.000 SEL N=16
175 - 174.000 DEFAULT SYNC PRO="DFLPRE"
176 - 175.000 END
177 - 176.000 I SUPER
178 - 177.000 REM PROFILE $SYNC
179 - 178.000 CREATE PROFILE $SYNC $SYNC
180 - 179.000 "NULL"
181 - 180.000 M PROC DELHC FROM $SYNC "default hardcopy"
182 - 181.000 TABSYM=YES; TABRELATIVE=YES; SPACEINSERT=YES; DISINPUT=YES
183 - 182.000 FULLCXPAPERTAPE=NO; HALFUXPAPERTAPE=NO; LOWERCASE=NO
184 - 183.000 HEIGHT=0; PROGDTAB=NO; LIMBOCLM=YES
185 - 184.000 NOOPTMIZ=NO
186 - 185.000 DEVICES=YES; EDITORV=YES; BLANKERASES=NO; LOWERCASEPRINTS=NO
187 - 186.000 CHARSETNAME="ASCII"; INPUT=YES; OUTPUT=YES
188 - 187.000 SEND9KSPACE=NO; ACTONTRN=NO; APPLCNRM=YES; OPDATAB=NO
189 - 188.000 DEVICE=CR=YES; DEVELF=YES; AUTONL=NO; CRISNL=NO; LFISNL=NO
190 - 189.000 DEVFINDOPTS=NO; APL=NO; 8IN=NO; PRINTTYPE=YES; TTYTYPE=YES
191 - 190.000 RETYPENV=NO; SCROLL=NO; PAGEMALT=NO
192 - 191.000 PRINTHALT=NO; RELPAGE=NO; DEVSCROLL=NO; CURSORUP=NO; TRUEVRPT=NO
193 - 192.000 TMNALG_PRM1=0; TMNALG_PRM2=0; TMNALG_PRM7=0; TMNALG_AFTRC=0; TMNALG_AFTR=0; ERSTIM=0
194 - 194.000 "NULL"
195 - 194.000 M PRO DFLCRT FROM $ASYNC  "default CRT"
196 - 195.000 TABSIM=YES; TABRELATIVE=YES; SPACEINSERT=YES; DISPINPUT=YES
197 - 196.000 FULLDUPAPER TAPE=no; HALFDUPERAPE TAPE=no; LOWERCASE=NO
198 - 197.000 HEIGHT=0; PROGDTAB=NO; LIMBOCLM=YES
199 - 198.000 NOOPTMIZ=NO
200 - 199.000 DEVICE=SYS=YES; BLANKERASES=YES; LOWERCASEPRINTS=NO
201 - 200.000 CHARSETNAME=ASCII; INPUT=YES; OUTPUT=YES
202 - 201.000 ENDBKSPACE=NO; ACONTRNS=NO; APPLCNR=SYS=YES; OPDIN=NO
203 - 202.000 DEVICECTL=YES; DEVICEF=SYS=; AUTONL=NO; CRISNL=NO; LFISNL=NO
204 - 203.000 DEVPOSOPTS=NO; APL=NO; BIN=NO; PRINTTYPE=YES; TTYTYPE=YES
205 - 204.000 RETYPEN=NO; EDITOR=NO; SCROLL=YES; PAGEHALT=NO
206 - 205.000 PRINTHALT=YES; RELPAGE=NO; DEVSCROLL=NO; CURSORUP=NO; TRUOVRPRT=NO
207 - 206.000 TMNALG_PRM1=0; TMNALG_PRM2=0; TMNALG_PRM7=0; TMNALG_AFTCR=0; TMNALG_AFTLF=0; ERSTIM=0
208 - 207.000 "NULL"
209 - 208.000 M PRO TTY FROM DFLHC  "Very basic and slow"
210 - 209.000 DEVICES=NO; EDITOR=NO
211 - 210.000 AWIDTH=72; PWIDTH=72; HEIGHT=0
212 - 211.000 CHARSETNAME=ASCII; LOWERCASEPRINTS=NO
213 - 212.000 DEVICECTL=YES; DEVICEF=SYS=; AUTONL=NO; CRISNL=NO; LFISNL=NO
214 - 213.000 TMNALG_PRM1=2.5; TMNALG_PRM2=166.7; TMNALG_PRM7=66.7; TMNALG_AFTCR=1; TMNALG_AFTLF=2; ERSTIM=0
215 - 214.000 "NULL"
216 - 215.000 M PRO DFLPRF FROM $TYP  "Default profile (before logon)"
217 - 216.000 FCNTBL=CPS
218 - 217.000 "NULL"
219 - 218.000 END
220 - 219.000 IC ME OVER $XPISALL::SYS(LN)
221 - 220.000 !!CFALF PACKSET=SYS
222 - 221.000 !!PIG
223 - 222.000 CR DP#PACKSET.SUPPORT GR=5000
224 - 223.000 CR DP#PACKSET.X GR=10000; EXEC=*
225 - 224.000 CR DP#PACKSET.XSI GR=15000
226 - 225.000 CR DP#PACKSET.LIBRARY GR=5000; exec=*
227 - 226.000 CR DP#PACKSET.DEMO GR=5000; READ=*
228 - 227.000 CR DP#PACKSET.DEPOSRO GR=10000; READ=*
229 - 228.000 CR DP#PACKSET.Convert GR=5000
230 - 229.000 CR DP#PACKSET.QUAC GR=5000
231 - 230.000 CR DP#PACKSET.COOPRC GR=5000
230 - 231.000 CR DP#PACKSET.:SRB GR=2500,READ=?
233 - 232.000 CR DP#PACKSET.:DOCUM GR=6000,READ=?
234 - 233.000 CR DP#PACKSET.:FED GR=3000
235 - 234.000 CP DP#PACKSET:NULLACCT GR=1,READ=?
236 - 235.000 DISM DP#PACKSET
237 - 236.000 END
239 - 237.000 !!RELEASE DP#PACKSET
239 - 238.000 !!SKIP_NETCON_STEP:
240 - 239.000 !!END_GF_TAPE_BOOT:
241 - 240.000 !M DINGO exiting, stepcc = %STEPCC
1 - 1.000 NETCON
2 - 2.000 DEL NODE=0
3 - 3.000 DEL NODE=1
4 - 4.000 DEL NODE=2
5 - 5.000 DEL NODE=3
6 - 6.000 DEL NODE=4
7 - 7.000 DEL NODE=5
8 - 8.000 DEL NODE=6
9 - 9.000 DEL NODE=7
10 - 10.000 DEL NODE=8
11 - 11.000 DEL NODE=9
12 - 12.000 DEL NODE=10
13 - 13.000 DEL NODE=11
14 - 14.000 DEL NODE=12
15 - 15.000 DEL NODE=13
16 - 16.000 DEL NODE=14
17 - 17.000 DEL NODE=15
18 - 18.000 DEL NODE=16
19 - 19.000 DEL NODE=17
20 - 20.000 DEL NODE=18
21 - 21.000 DEL NODE=19
22 - 22.000 DEL NODE=20
23 - 23.000 DEL NODE=21
24 - 24.000 DEL NODE=22
25 - 25.000 DEL NODE=32
26 - 26.000 DEL NODE=33
27 - 27.000 DEL NODE=34
28 - 28.000 DEF NODE=0, NAME=L66A, TYPE=ME
29 - 29.000 DEF NODE=1, NAME=L66I, TYPE=FEP, CONTROL=L66A
30 - 30.000 DEF NODE=2, NAME=L66II, TYPE=DEBUG, CONTROL=L66B
31 - 31.000 DEF NODE=3, NAME=L66III, TYPE=DEBUG, CONTROL=L66B
32 - 32.000 DEF NODE=4, NAME=L66IV, TYPE=FEP, CONTROL=L66A
33 - 33.000 DEF NODE=5, NAME=L66V, TYPE=DEBUG, CONTROL=L66B
34 - 34.000 DEF NODE=6, NAME=L66VI, TYPE=FEP, CONTROL=L66A
35 - 35.000 DEF NODE=7, NAME=L66VII, TYPE=DEBUG, CONTROL=L66B
36 - 36.000 DEF NODE=8, NAME=L66VIII, TYPE=FEP, CONTROL=L66A
37 - 37.000 DEF NODE=9, NAME=L66IX, TYPE=FEP, CONTROL=L66B
38 - 38.000 DEF NODE=10, NAME=L6X, TYPE=DEBUG, CONTROL=L66B
39 - 39.000 DEF NODE=11, NAME=L6XI, TYPE=FEP, CONTROL=L66B
40 - 40.000 DEF NODE=12, NAME=L6XII, TYPE=FEP, CONTROL=L66B
41 - 41.000 DEF NODE=13, NAME=L6XIII, TYPE=DEBUG, CONTROL=L66B
42 - 42.000 DEF NODE=15, NAME=L6XV, TYPE=DEBUG, CONTROL=L66A
43 - 43.000 DEF NODE=20, NAME=L66A, TYPE=HOST
44 - 44.000 DEF NODE=21, NAME=L660, TYPE=HOST
45 - 45.000 DEF NODE=22, NAME=L66C, TYPE=HOST
46 - 46.000 DEF NODE=32, NAME=DVFEP, TYPE=FEP, CONTROL=L66A
47 - 47.000 DEF NODE=33, NAME=CRFEP, TYPE=FEP, CONTROL=L66A
48 - 48.000 DEF NODE=34, NAME=CPFEP, TYPE=FEP, CONTROL=L66A
49 - 49.000 " L66A: NETCON - DEFAULT COMMANDS AND CONFIGS FOR FEP 4 " ;
50 - 50.000 SET NODE=4
51 - 51.000 SET BOOTINFO
52 - 52.000 M:FEP SYS
53 - 53.000 5
54 - 54.000 NODE=ADM
55 - 55.000 NODE=ADM SYS
56 - 56.000 COUPLER
57 - 57.000 COUPLER SYS
58 - 58.000 ASYNC
59 - 59.000 ASYNC SYS
60 - 60.000 BISYNC
61 - 61.000 BISYNC SYS
62 - 62.000 HOCX25
63 - 63.000 HOCX25 SYS
64 - 64.000 SYS
65 - 65.000 DEFAULT ASYNC ;
66 - 66.000 INPUT = YES ;
67 - 67.000 OUTPUT = YES ;
68 - 68.000 SALUTATION = YES ;
69 - 69.000 SPEED = AUTO ;
70 - 70.000 ENABLE = YES ;
71 - 71.000 REENABLE = YES ;
72 - 72.000 DROPSTR = NO ;
73 - 73.000 HARDWIRE = YES ;
74 - 74.000 ECHOLOGON = NO ;
75 - 75.000 LOGONTIMEOUT = 5 ;
76 - 76.000 REACTIMEOUT = 0 ;
PROFILE="DFLPREF"

DEFAULT BISYNC;
INPUT = YES,;
OUTPUT = YES,;
SPEED = 2400,;
REMOTE = TERMINAL,;
ENABLE = YES,;
HARDWIRE = YES,;
CLOCKING = NO,;
REENABLE = YES,;
READTIMEOUT = 0,

DEFAULT HOLC25;
INPUT = YES,;
OUTPUT = YES,;
SPEED = AUTO,;
REMOTE = HOST,;
ENABLE = NO,;
HARDWIRE = YES,;
CLOCKING = NO,;
REENABLE = YES,;
READTIMEOUT = 0,

CONFIG .4480 HARDWIRE=NO,DROPDTR=YES,LOGONTIMEOUT=5,READTIMEOUT=2

CONFIG .4480 HARDWIRE=NO,DROPDTR=YES,LOGONTIMEOUT=5,READTIMEOUT=2

Phoenix HLSUA RFEP
Phoenix HLSUA RFEP
Phoenix MEW-CRF RFEP
Phoenix MEW-CRF RFEP

rm 284 Fosnight
ALEX modem
rm 212 Anderson

JLJ RM241 IBMPC

" LetC :NETCON - DEFAULT COMMANDS AND CONFIGS FOR FEP 6 ";

SEL NO=6
SET BOOTINFO
M:FEP,;SYS
116 - 114.000 S
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 DROPSTR = 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='OFPLPRF'
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 HARDWARE = 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 CONFGS FOR FEP B";
163 - 161.000 SEL NODE=S
164 - 162.000 SET BOOTINFO
165 - 163.000 M:FEP:SYS
166 - 164.000 S
167 - 165.000 NODEADMIN
168 - 166.000 NODEADMIN: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 HARDWARE = YES;
185 - 183.000 REENABLE = YES;
186 - 184.000 DROPODTR = NO;
187 - 185.000 ECHOLOGON = NO;
188 - 186.000 LOGONTIMEOUT = 5;
189 - 187.000 READTIMEOUT = 0;
190 - 188.000 PROFILE="DFLPREF"
191 - 189.000 DEFAULT BISYNC ;
192 - 190.000 INPUT = YES;
193 - 191.000 OUTPUT = YES;
SPEED = 2400;
REMOTE = TERMINAL;
ENABLE = YES;
HARDWARE = YES;
CLOCKING = NO;
REENABLE = YES;
READEMOUT = 0
DEFAULT HDLCX25;
INPUT = YES;
OUTPUT = YES;
SPEED = AUTO;
REMOTE = HOST;
ENABLE = NO;
HARDWARE = YES;
CLOCKING = NO;
REENABLE = YES;
READEMOUT = 0
CONFIG .O600 Buf=256 "rm 2105 Sickler
CONFIG .O700 Buf=256 "rm 252 Coatozzi
CONFIG .O900 Buf=256 "rm 2110 Metfield
CONFIG .O480 LOGON="DEBUG12",HARD=NO,SAL=NO,SPEED=9600,DRIP=YES,BUF=128,ENABLE=NO
CONFIG .O480 LOGON="DEBUG9",HARD=NO,SAL=NO,SPEED=9600,DRIP=YES,BUF=128,ENABLE=NO
CONFIG .O480 LOGON="DEBUG3",HARD=NO,SAL=NO,SPEED=9600,DRIP=YES,BUF=128
CONFIG .O480 SPEED=1200,HARD=YES,SAL=NO,LOGON=\"name acct pass\",PROFILE=\"TI855\",LOGONTIMEOUT=0
"CONFIG .O500 SPEED=1200,HARD=YES,SAL=NO,LOGON=\"name acct pass\",PROFILE=\"TI855\",LOGONTIMEOUT=0,READTIMEOUT=0,PROFILE=\"QMS1200\",FLOWC=ES
CONFIG .O600 LOGON="L668"
CONFIG .O300 LOGON="MAOC3270",SPEED=9600 "Upstairs MOD400 filetran Beaumont
CONFIG .O400 ENABLE=YES,LOGON="LNK032" "Phoenix DVCP RFEP
SET NSHF=24
SEL H=9ISYNC
SET MEM=84
"LOC A :NETCON - DEFAULT COMMANDS AND CONFIGS FOR FEP 32 "
SEL \n=32
230 - 225.000 SET BOOTINFO
231 - 227.000 M:FEP..SYS
232 - 228.000 4
233 - 229.000 NODENAME
234 - 230.000 NODENAME..SYS
235 - 231.000 HOLC25
236 - 232.000 HOLC25..SYS
237 - 233.000 ASYNC
238 - 234.000 ASYNC..SYS
239 - 235.000 UNITPEC
240 - 236.000 UNITPEC..SYS
241 - 237.000 :SYS
242 - 238.000 DEF LINK .FO00,.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 DROPCTR = NO,;
251 - 247.000 HARDWIRE = YES,;
252 - 248.000 ECHOLGON = 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 HOLC25 ;
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  HARDWARE = YES;
273 - 269.000  CLOCKING = NO;
274 - 270.000  REENABLE = YES;
275 - 271.000  READTIMEOUT = 0
276 - 272.000  CONFIG .0550 LOGON="name,acct,pass"
277 - 273.000  CONFIG .0680 LOGON="name,acct,pass"
278 - 274.000  CONFIG .0730 LOGON="name,acct,pass"
279 - 275.000  CONFIG .F000 LOGON="LINK32",ENABLE=NO,REENABLE=YES
280 - 276.000  CONFIG .F100 LOGON="LINK32",ENABLE=NO,REENABLE=YES
281 - 277.000  " L66A : NETCON - DEFAULT COMMANDS AND CONFIGS FOR FEP 33 " ;
282 - 278.000  SEL N=33
283 - 279.000  SET P00INFO
284 - 280.000  M:FEP:.SYS
285 - 281.000  4
286 - 282.000  NODEADMIN
287 - 283.000  NODEADMIN:.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,.9300
296 - 292.000  DEFAULT ASYNC ;
297 - 293.000  IMUPT = 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  DROPOUT = NO;
304 - 300.000  HARDWARE = YES;
305 - 301.000  ECHOCLOGON = NO;
306 - 302.000  LOGONTIMEOUT = 15;
307 - 303.000  READTIMEOUT = 0;
308 - 304.000 PROFILE="DFLPREF"
309 - 305.000 DEFAULT BISYNC;
310 - 306.000 INPUT = YES;
311 - 307.000 OUTPUT = YES;
312 - 308.000 SPEED = 2400;
313 - 309.000 REMOTE = TERMINAL;
314 - 310.000 ENABLE = YES;
315 - 311.000 HARDWIRE = YES;
316 - 312.000 CLOCKING = NO;
317 - 313.000 REENABLE = YES;
318 - 314.000 READTIMEOUT = 0
319 - 315.000 DEFAULT MDLX25;
320 - 316.000 INPUT = YES;
321 - 317.000 OUTPUT = YES;
322 - 318.000 SPEED = AUTO;
323 - 319.000 REMOTE = HOST;
324 - 320.000 ENABLE = NO;
325 - 321.000 HARDWIRE = YES;
326 - 322.000 CLOCKING = NO;
327 - 323.000 REENABLE = YES;
328 - 324.000 READTIMEOUT = 0
329 - 325.000 CONFIG .0590 LOGON="name\acct\pass"
330 - 326.000 CONFIG .0680 LOGON="name\acct\pass"
331 - 327.000 CONFIG .6960 LOGON="name\acct\pass" HARDWIRE=YES SPEED=4800 READTIME=0 LOGONTIME=0
332 - 328.000 CONFIG .9800 LOGON="LINK33A" ENABLE=YES REENABLE=YES
333 - 329.000 CONFIG .9800 LOGON="LINK33B" ENABLE=YES REENABLE=YES
334 - 330.000 "L66A\NETCON DEFAULT COMMANDS AND CONFIGS FOR FEP 34";
335 - 331.000 SEL N=34
336 - 332.000 SET BOOTINFO
337 - 333.000 M:\FEP..SYS
338 - 334.000 4
339 - 335.000 NODFAWMN
340 - 336.000 NODEADMIN..SYS
341 - 337.000 MDLX25
342 - 338.000 MDLX25..SYS
343 - 339.000 ASYNC
344 - 340.000 ASYNC..SYS
345 - 341.000 UNITREC
346 - 342.000 UNITREC..SYS
347 - 343.000 :SYS
348 - 344.000 DEF LINK .3000,.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 DROPDT = 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 HOLC25 ;
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 .9000 LOGON = "LINK34A", ENABLE = YES, REENABLE = YES
383 - 379.000 CONFIG .100 LOGON = "LINK34B", ENABLE = YES, REENABLE = YES
384 - 380.000 CONFIG .9000 LOGON = "LINK34A", ENABLE = YES, REENABLE = YES
385 - 381.000 CONFIG .100 LOGON = "LINK34B", ENABLE = YES, REENABLE = YES
Define the host node number and name for NETCON.

Define the local FEP node number and name which the RFEP is connected to. Define the channel(s) that is(are) used to connect to the remote FEP.

Define the remote FEP (RFEP) node number and name. Define the channel(s) that is(are) used to connect to the local FEP.

Define the profile names for the link profiles at the local and remote ends of the links. (11 char max.)
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 rewriting the diskette every time the remote vc was changed.

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.

Form the X.25 address for the local and remote ends of the link. The address is of the form mmnni, where 'mm' is the two-digit node number the address is to, 'nn' is the two-digit node number the address is from, and 'i' is the number of the physical link (i.e., 01, 02, etc.).
1 - 1.000 !ECHO
2 - 4.000 !DPIBBLE OVER RFEP_NETCON_Dribble
3 - 7.000 !IF "FEP_NODES$ = "%(FEP_NODE$)|| "$)" THEN XEQ RFEP_GLOBALS
4 - 8.000 !COPY ME over *RFEP_NETCON
5 - 9.000 !NETCON
6 - 9.300 delete node = HOST_NODE$,
7 - 9.510 define node = HOST_NODE$, name = HOST_NAMES$, type = me
8 - 10.000 delete node = RFEP_NODE$
9 - 11.030 define node = RFEP_NODE$, name = RFEP_NAME$, type = fep, control = HOST_NAME$
10 - 12.000 update
11 - 12.500 select node = RFEP_NODE$
12 - 12.510 define link FEP_CHAN$
13 - 12.520 config FEP_CHAN$;
14 - 12.530 enable = yes;
15 - 12.540 logon = 'FEP_RFEP_LINK$';
16 - 12.550 reenable = yes
17 - 13.000 select node = RFEP_NODE$
18 - 14.000 set bootinfo
19 - 15.000 M:FEP:.SYS
20 - 15.000 3
21 - 17.000 NODEADMIN
22 - 18.000 NODEADMIN:.SYS
23 - 19.000 HDLCX25
24 - 20.000 HDLCX25:.SYS
25 - 21.000 ASYNC
26 - 22.000 ASYNC:.SYS
27 - 23.000 :SYS
28 - 24.000 default ASYNC;
29 - 25.000 breakrequired = no;
30 - 26.000 bufsize = 256;
31 - 27.000 droptr = no;
32 - 28.000 echologon = no;
33 - 29.000 enable = yes;
34 - 30.000 hardwire = yes;
35 - 31.000 input = yes;
36 - 32.000 logontimeout = 0;
37 - 33.000 output = yes;
profile = 'DFLPRF',
readtimeout = 0,
reenable = yes,
salutation = yes,
speed = auto

default HCLCX25
breakrequired = no,
clocking = no,
enable = yes,
hardwire = no,
inout = yes,
output = yes,
readtimeout = 0,
reenable = yes,
speed = auto
select handler = ASYNC
set RCVCQ@YTSIZ = 16384
set NSHUF = 600
define link RFEP_CHAN
config RFEP_CHAN
enable = yes,
logon = 'RFEP_FEP_LINKS',
reenable = yes
end
! EOD
! SEQAR RFEP_NETCON
! DONT DRIABLE
! ELDSTRO RFEP_NETCON_DRIABLE
! DIR R
Profile for local end of links

```
1 - 1.000 'ECHO
2 - 2.000 !DRIBBLE OVER RFEP_SUPER_DRIBBLE
3 - 3.000 !IF "RFEP_NODES" = "X('RFEP_NODE'||'S')" THEN XEQ RFEP_GLOBALS
4 - 4.000 !COPY ME
5 - 5.000 !!SUPER
6 - 6.000 "
7 - 7.000 "
8 - 8.000 "
9 - 9.000 remove profile LOCAL_LINKS
10 - 10.000 create profile LOCAL_LINKS link
11 - 11.000 circuits = 10
12 - 12.000 default packet size = 1024
13 - 13.000 default response timer = 0
14 - 14.000 default window = 7
15 - 15.000 frame = 1024
16 - 16.000 mode = CTE
17 - 17.000 retransmission = 10
18 - 18.000 timeout = 3
19 - 19.000 window = 7
20 - 20.000 end
21 - 21.000 "
22 - 22.000 "
23 - 23.000 "
24 - 24.000 remove profile REMOTE_LINKS
25 - 25.000 create profile REMOTE_LINKS link
26 - 26.000 circuits = 10
27 - 27.000 default packet size = 1024
28 - 28.000 default response timer = 0
29 - 29.000 default window = 7
30 - 30.000 frame = 1024
31 - 31.000 mode = DCE
32 - 32.000 retransmission = 10
33 - 33.000 timeout = 3
34 - 34.000 window = 7
35 - 35.000 end
36 - 36.000 "
37 - 37.000 "
```

Profile for RFEP end of links

Profile for local end of virtual circuit
remove profile LOCAL_VCS
create profile LOCAL_VCS virtual circuit

delays = 2
maxvircir = 6
minvircir = 1
receive size = 1024
receive window = 7
response to complete = yes
response delay = 2
response timer = 0
retrys = 0
send size = 1024
send window = 7
timeout = 3
type = primary
end

Profile for remote end of virtual circuit

remove profile REMOTE_VCS
create profile REMOTE_VCS virtual circuit
delays = 2
maxvircir = 6
minvircir = 1
receive size = 1024
receive window = 7
response to complete = yes
response delay = 2
response timer = 0
retrys = 0
send size = 1024
send window = 7
timeout = 3
type = secondary
end

Links for local end of link to RFEP_NAMES$
77 - 77.000 remove link FEP_RFEP_LINKS
78 - 78.000 create link FEP_RFEP_LINKS
79 - 79.000 address = LOCAL_ADDRESS$
80 - 80.000 profile = LOCAL_LINKS
81 - 81.000 end
82 - 82.000"
83 - 83.000"
84 - 84.000"
85 - 85.000 remove link RFEP_RFEP_LINKS
86 - 86.000 create link RFEP_RFEP_LINKS
87 - 87.000 address = REMOTE_ADDRESS$
88 - 88.000 profile = REMOTE_LINKS
89 - 89.000 end
90 - 90.000"
91 - 91.000"
92 - 92.000"
93 - 93.000 remove virtual circuit 1 for link FEP_RFEP_LINKS
94 - 94.000 create virtual circuit 1 for link FEP_RFEP_LINKS
95 - 95.000 address = REMOTE_ADDRESS$
96 - 96.000 destination = RFEP_NAME$
97 - 97.000 profile = LOCAL_VCS
98 - 98.000 end
99 - 99.500 create virtual circuit 2 for link FEP_RFEP_LINKS
100 - 98.510 address = REMOTE_ADDRESS$
101 - 98.520 destination = RFEP_NAME$
102 - 98.530 profile = LOCAL_VCS
103 - 98.540 end
104 - 99.900"
105 - 100.000"
106 - 101.000"
107 - 102.000" remove virtual circuit 1 for link RFEP_RFEP_LINKS
108 - 103.000" create virtual circuit 1 for link RFEP_RFEP_LINKS
109 - 104.000" address = LOCAL_ADDRESS$
110 - 105.000" destination = FEP_NAME$
111 - 106.000" profile = REMOTE_VCS
112 - 107.000" end
113 - 108.000 end
114 - 109.000 !END
115 - 110.000 !END *RFEP_SUPER
1 - 2.000 !LET ECHO = $FLAG ( ECHO )
2 - 3.000 !CONT ECHO
3 - 4.500 !WHAT_X PA
4 - 6.000 !FEP:
5 - 9.000 !LET FEP = $INPUT ( 'Which FEP is diskette to be built ON (DVFEP|CFEP|CP6FEP): ' )
6 - 10.000 !IF FEP = '' THEN GOTO END
7 - 11.000 !IF FEP = 'DVFEP' THEN GOTO FEP_END
8 - 12.000 !IF FEP = 'CFEP' THEN GOTO FEP_END
9 - 13.000 !IF FEP = 'CP6FEP' THEN GOTO FEP
10 - 14.000 !FEP_END:
11 - 16.000 !RFEP:
12 - 17.000 !LET RFEP = $INPUT ( 'Which RFEP is diskette to be built FOR (DVFEP|CFEP|CP6FEP): ' )
13 - 18.000 !IF RFEP = '' THEN GOTO END
14 - 19.000 !IF RFEP = 'DVFEP' THEN GOTO RFEP_END
15 - 20.000 !IF RFEP = 'CFEP' THEN GOTO RFEP_END
16 - 21.000 !IF RFEP = 'CP6FEP' THEN GOTO RFEP
17 - 22.000 !RFEP_END:
18 - 45.000 !DRIVE1:
19 - 46.000 !OUTPUT "Put the new diskette for %RFEP in drive 1."
20 - 47.000 !LET ANS = $INPUT ( 'Enter GO when diskette installed: ' )
21 - 48.000 !IF ANS = 'GO' THEN GOTO DRIVE1
22 - 49.000 !DRIVE1_END:
23 - 50.000 !BATCH:
24 - 51.000 !LET NAME = RFEP || '_DISKETTE'
25 - 52.000 !BATCH J:RFEP_DISKETTE %("NAME" || "$") = "XNAME" ;
26 - 53.000 !FEP = "RFEP" ;
27 - 54.000 !RFEP = "RFEP" ;
28 - 65.000 !END:
29 - 67.000 !IF ECHO = 'YES' THEN ECHO
1.000 !DEFUALT NAM$=RFEP_DISKETTE
2.000 !DEFUALT WSN=QVPSIG3$; DEFERS=0:00; SCHED$=RERUN
3.000 !DEFUALT TIMES=1:00; MEMS=64; FPOOL$=31
4.000 !JOB NAME=NAMES$; WSN=WSN$, DEFER=(DEFERS$); SCHED$
5.000 !RES TIME=TIMES$, MEM=MEM$
6.000 !LIMIT FPOOL$=FPOOL$
7.000 !M Job to build RFEP diskette for RFEP$.
8.000 !M Diskette will be built on drive 1 of FEPS.
9.000 !PIGETTE
10.000 use fep FEPS
11.000 time
12.000 build RFEP$ on fep FEPS drive 1
13.000 time
14.000 list fep FEPS drive 1
15.000 end
16.000 !COPY me over *mass
17.000 TO:*m
18.000 SUBJECT:NAMES completed
19.000 FCOPY:NO
20.000 R=$:NC
21.000 !EOD
22.000 !SEND *mass
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 manually 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 WOO, 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
1 - !SUPER
2 -  R FORM STDINT
3 -  CR PSEUDO STDINT FOR STDLP
4 -  BANNER
5 -  REPEATS = 2
6 -  ENTRY = 45, 1, 0, 1, 0, 1, 0, 0
7 -  %$
8 -  Entry $ = 01, 001, 0, 1, 0, 1, 0, 0
9 -  %DATE
10 -  Entry $ = 01, 010, 0, 1, 0, 1, 0, 0
11 -  %TIME
12 -  Entry $ = 01, 016, 0, 1, 0, 1, 0, 0
13 -  %ACCN
14 -  Entry $ = 01, 025, 0, 1, 0, 1, 0, 0
15 -  %SYSID
16 -  Entry $ = 01, 031, 0, 1, 0, 1, 0, 0
17 - .
18 -  Entry $ = 01, 032, 0, 1, 0, 1, 0, 0
19 -  %SUBFILE
20 -  Entry $ = 01, 039, 0, 1, 0, 1, 0, 0
21 - ( 
22 -  Entry $ = 01, 040, 0, 1, 0, 1, 0, 0
23 -  %JOBNAME
24 -  Entry $ = 01, 071, 0, 1, 0, 1, 0, 0
25 - )
26 -  Entry $ = 01, 074, 0, 1, 0, 1, 0, 0
27 -  WOO= 
28 -  Entry $ = 01, 078, 0, 1, 0, 1, 0, 0
29 -  %WOO
30 -  Entry $ = 01, 088, 0, 1, 0, 1, 0, 0
31 -  WOD=@
32 -  Entry $ = 01, 093, 0, 1, 0, 1, 0, 0
33 -  %WOD
34 -  Entry $ = 01, 103, 0, 1, 0, 1, 0, 0
35 -  CP-6/
36 -  Entry $ = 01, 108, 0, 1, 0, 1, 0, 0
37 -  %VERSION
38 -  ENTRY $ = 4, 1, 0, 1, 0, 1, 0, 0
39 -  Schmucklet # 1 is
40 -  ENTRY $ = 4, 19, 0, 1, 0, 1, 0, 0
41 -  %BANNE TEXT1
42 -  ENTRY $ = 5, 1, 0, 1, 0, 1, 0, 0
43 -  Schmucklet # 2 is
44 -  ENTRY $ = 5, 19, 0, 1, 0, 1, 0, 0
45 -  %BANNE TEXT2
46 -  ENTRY $ = 10, 16, 2, 3, 2
47 -  CP-6
48 -  ENTRY $ = 24, 10, 1, 2, 1
49 -  Internals
50 -  ENTRY $ = 33, 1, 1, 2, 1, 1, 0, 0
51 -  %BANNE TEXT3
52 -  END
53 - END
54 - END
55 - !"
56 - !SUPER
57 - CR WSN INTERNAL
58 - DEVICE=LP@UPSTAIRS
59 - DFORM LP STDINT
60 - END
61 - MOD ZZINT?
62 - WSN=INTERNAL
63 -
64 - 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.
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 $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_0 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 'GOOD-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.
1 - !JOB WSN=LOCAL
2 - !RES MEM=300,TIME=2
3 - !PL6 LAB21_SI6 OVER LAB21_OU6(LS,SR(.:LIBRARY),SCHEMA)
4 - !LEMUR
5 - COPY :LIB_SYSTEM.:SYS(X66_AUTO_0) INTO *LIB
6 - END
7 - !LINK LAB21_OU6,*LIB :
8 - OVER :SHARED_INT (MAP(VALUE,NAMESPACE),NODEBUG,NOSH,SLIB,REP=0,)
9 - VECTOR(ENTRIES(OPEN_DCBL,WRITE_ALINE,READ_ALINE,CLOSE_DCBL,EXIT_ALL),
10 - REMOVE_E,REMOVE_S,DATA=247)
11 - !PL6 LAB2M_SI6 OVER LAB2M_OU6(LS,SR(.:LIBRARY),SCHEMA)
12 - !LINK LAB2M_OU6 OVER LAB2M_RU(SHAREL=:SHARED_INT.ZZINT,;
13 - UNSAT=:LIB_SYSTEM.:SYS)
14 - !SPIDER
15 - DEL :SHARED_INT
16 - IN :SHARED_INT,LIB FROM :SHARED_INT.ZZINT
17 - L :SHARED_INT
18 - END
19 - !LAB2M_RU.
20 - TUNA
21 - !C LAB2_JCL TO LP(K)
LAB 2

ANSWER
1 - /*M* JJ'S SHARED LIBRARY (PHASE 1) */
2 - /*T*******************************************************************/
3 - /*T* *
4 - /*T* COPYRIGHT, (C) HONEYWELL INFORMATION SYSTEMS INC., 1981 *
5 - /*T* *
6 - /*T*******************************************************************/
7 - /*X* DMR,PLM=5,IND=5,CTI=5,SDI=5,MCL=10,CSI=0,ECI=0 */
8 - *
9 - %SET LISTSUB='1'B;
10 - *
11 - OPEN_DCB: PROC (DCBNO);
12 - *
13 - DCL DCBNO SBIN WORD;
14 - *
15 - %INCLUDE CP_6;
16 - *
17 - %FPT_OPEN (FPTN=OPEN_DCBNO,
18 - CTG=YES,
19 - RES='ME',
20 - DISP=NAMED);
21 - *
22 - /* GLOBAL DATA DEFINITIONS */
23 - *
24 - DCL GLOBAL_DCBNO SBIN WORD STATIC SYMDEF;
25 - *
26 - /*
27 - BEGIN PROCEDURE
28 - */
29 - *
30 - GLOBAL_DCBNO = DCBNO;
31 - *
32 - OPEN_DCBNO,V.DCB# = GLOBAL_DCBNO;
33 - *
34 - CALL M$OPEN (OPEN_DCBNO)
35 - WHENALTRETURN
36 - DO;
37 - CALL M$MERC;
38 -   CALL M$EXIT;
39 -   END; /* DO IF ALTRETURN */
40 -
41 - RETURN;
42 -
43 - END OPEN_DCB;
44 - %EOD;
45 - %SET LISTSUB = '1'B;
46 -
47 - READ_ALINE: PROC (BUF_, SIZE_);
48 -
49 - DCL BUF_ CHAR(132);
50 -
51 - DCL SIZE_ SBIN WORD;
52 -
53 - DCL 1 BUFFER_ BASED,
54 -      2 BUFFER_CHARS_ CHAR(SIZE_) CALIGNED;
55 -
56 -   %INCLUDE CP_6;
57 -
58 - /* GLOBAL REFERENCES */
59 -
60 - DCL GLOBAL_DCBNO SBIN WORD SYMREF;
61 -
62 - /* LOCAL FPTS */
63 -      %FPT_READ       (FPTN=READ_ALINE_,
64 -        WAIT=YES);
65 -
66 -      %F$DCB;
67 -
68 -
69 -   /*
70 -   */
71 -
72 -   /* BEGIN PROCEDURE */
73 -
74 - READ_ALINE_.V.DCB# = GLOBAL_DCBNO;
75 -
76 - 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$EXIT;
END EXIT_ALL;
1 - /** JJ'S SHARED LIBRARY TEST PROGRAM */
2 - /***************************************************************************/
3 - *T*
4 - *T* COPYRIGHT, (C) HONEYWELL INFORMATION SYSTEMS INC., 1981 *
5 - *T*
6 - *T***************************************************************************/
7 - /*X* DMR,PLM=5,IND=5,CTL=5,SDL=5,MCL=10,CSI=0,ECI=0 */
8 -
9 -
10 - %SET LISTSUB='1'B;
11 -
12 -
13 - MYPROG: PROC MAIN;
14 -
15 -
16 -    /* LIBRARY ENTRY POINTS */
17 -
18 - DCL OPEN_DC B ENTRY(1);
19 - DCL READ_ALINE ENTRY(2);
20 - DCL WRITE_ALINE ENTRY(2);
21 - DCL CLOSE_DC B ENTRY;
22 - DCL EXIT_ALL ENTRY;
23 -
24 - DCL BUFFER_CHAR(120) STATIC;
25 - DCL BUFSIZ_SBIN_WORD STATIC INIT(SIZEC(BUFFER_));
26 -
27 - DCL M$MINE DC B;
28 -
29 -    CALL OPEN_DC B(DCBNUM(M$MINE));
30 -
31 -    CALL READ_ALINE(BUFFER_,BUFSIZ_);
32 -
33 -    CALL WRITE_ALINE(BUFFER_, BUFSIZ_);
34 -
35 -    CALL CLOSE_DC B;
36 -
37 -    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
($LS37$ -> $.600$ if CP running)
$LS35$ -> $.560$ — bottom SS frame of Mon SS
$LS4$ using $x$ — Auto of user
($LS4$ using $x$ {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
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 received 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 received 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 IO 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 omgroups

Your group will be provided with sample programs which illustrate some omgroup programming techniques. These programs involve a omgroup administrative user that reads messages from a terminal station and sends the messages to a DCE 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 omgroup X account tools that you might find useful are:

CCG
LARK
CGU

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

!SUPER
CMD*CRE DEV LJ$1HOST
OPT*USE=CG
OPT*PROFILE=VIP7205
OPT*OMGROUP=CG/INTERNALS.CG.LJ$HOST
OPT*NOCG=INFORM
OPT*PASS=CG
OPT*END
CMD*END
17:22 AUG 29 '85 INT_CGEX_CRU

1 - !JOB NAME=INT_CGEX_CRU,WSN=UPSTAIRS
2 - !RESOURCE TIME=3, MEM=256
3 - !PL6 INT_CGEX_S11 OVER *INT_CGEX_OU1, INT_CGEX_LS1
4 - !LINK *INT_CGEX_OU1 OVER INT_CGEX_1
5 - !PL6 INT_CGEX_S12 OVER *INT_CGEX_OU2, INT_CGEX_LS2
6 - !LINK *INT_CGEX_OU2 OVER INT_CGEX_2
7 - !C INT_CGEX_LS1 TO LP
8 - !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.

*/
77 - ( FPTN  = FPT_INT,
78 -     UENTRY = INTERRUPT_HANDLER ) ;
79 -
80 - %FPT_OPEN
81 - ( FPTN  = FPTOPEN_CG,
82 -     STCLASS = CONSTANT,
83 -     ACCT   = VLPACCT,
84 -     ASN    = COMGROUP,
85 -     AU     = YES,
86 -     CTG    = YES,
87 -     DCB    = M$CG,
88 -     EXIST  = OLDFILE,
89 -     FUN    = CREATE,
90 -     NAME   = VLPNAME,
91 -     QISS   = YES,
92 -     SCRUB  = YES,
93 -     SETSTA = VLPSETSTA_AU ) ;
94 -
95 - %FPT_READ
96 - ( FPTN  = FPT_READ_CG,
97 -     BUF    = IO_BUFFER,
98 -     DCB    = M$CG,
99 -     STATION = VLP_STATION ) ;
100 -
101 - %FPT_TIME
102 - ( FPTN  = FPT_TIME,
103 -     DATE   = DATE,
104 -     DEST   = LOCAL,
105 -     TIME   = TIME ) ;
106 -
107 - %FPT_WRITE
108 - ( FPTN  = FPT_WRITE_CG,
109 -     STCLASS = STATIC SYMDEF,
110 -     BUF    = IO_BUFFER,
111 -     DCB    = M$CG ) ;
112 -
113 - %FPT_WRITE
114 - ( FPTN  = FPT_WRITE_LO,
115 -     BUF    = IO_BUFFER,
116 -       DCB = M$LO ;
117 -
118 - %VLP.CGCP
119 -    ( FPTN = VLPCGCP,
120 -    STCLASS = CONSTANT,
121 -    RAS = YES,
122 -    TERMCONAU = NO,
123 -    TERMCONNAU = NO,
124 -    TRMRDSIZ = 74 ) ;
125 -
126 - %VLP_NAME
127 -    ( FPTN = VLPNAME,
128 -    STCLASS = CONSTANT,
129 -    NAME = 'INTERNALS.CG' ) ;
130 -
131 - %VLP_ACCT
132 -    ( FPTN = VLPACCT,
133 -    STCLASS = CONSTANT
134 -    /*
135 -    ,ACCT = 'LJSHOST'
136 -    */
137 -
138 -    ) ;
139 -
140 - %VLP_SETSTA
141 -    ( FPTN = VLPSETSTA_AU,
142 -    STCLASS = CONSTANT,
143 -    MYSTATION = 'AU' ) ;
144 -
145 - %VLP_STATION
146 -    ( FPTN = VLP$STATION,
147 -    STCLASS = BASED ) ;
148 -
149 - %VLP_STATION
150 -    ( FPTN = VLP_STATION ) ;
151 -
152 - %VLP_STATION
153 -    ( FPTN = VLP_STATION_TERM,
154 -    MSGTYP = 'LOG_MSG' ) ;
155 - %VLP_STATION
156 -   ( FPTN = VLP_STATION_DCB,
157 -       STATION = 'DCB',
158 -       MSGTYP = 'TRM_ECHO' );
159 -
160 - %EJECT ;
161 - M$CG$ = DCBADDR ( DCBNUM ( M$CG$ ) ) ;
162 - FPT_WRITE.CG.STATION_ = VECTOR ( VLP_STATION_DCB ) ;
163 -
164 - /*
165 -       Create a new comgroup and modify its control parameters.
166 - */
167 -
168 - CALL M$OPEN ( FPTOPEN.CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
169 - CALL M$CGCTL ( FPTCGCTL.CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
170 -
171 - /*
172 -       Set break control.
173 - */
174 -
175 - CALL M$INT ( FPT_INT ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
176 -
177 - /*
178 -       Loop until the TERM station and DCB station have both connected or
179 -       break has been typed at the AU's terminal.
180 - */
181 -
182 - DO UNTIL ( FLAGS.TERM_CON AND FLAGS.DCB_CON ) OR FLAGS.AU_BREAK ;
183 -
184 - /*
185 -       Issue a read for a message directed to the AU. This should catch only
186 -       *AUEV messages.
187 - */
188 -
189 - VLP_STATION.CTL.DIRONLY# = %YES# ;
190 - CALL M$READ ( FPT_READ.CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
191 -
192 - IF ( M$CG$->F$DCB.LASTSTA$->VLP$STATION.MSGTYP# = %AU_MESSAGE ) THEN DO ;
193 -   DO CASE IO_BUFFER$->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 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$WRITE ( FPT_WRITE_LO ) ALTRET ( REPORT_ERROR_AND_EXIT );
CALL M$WRITE ( FPT_WRITE.CG ) ALTRET ( REPORT_ERROR_AND_EXIT );
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,
350 - 2 AU_BREAK BIT (1) UNAL,
351 - 2 TERM_LOGOFF BIT (1) UNAL;
352 - DCL IO_BUFFER CHAR (200) SYMREF;
353 - DCL M$CG DCB;
354 - DCL M$CG$ PTR SYMREF;
355 -
356 - %F$DCB;
357 -
358 - %FPT_ACTIVATE
359 - (FPTN = FPT_ACTIVATE.CG,
360 - STCLASS = SYMREF);
361 -
362 - %FPT_CLOSE
363 - (FPTN = FPT_CLOSE.CG,
364 - STCLASS = SYMREF);
365 -
366 - %FPT_WRITE
367 - (FPTN = FPT_WRITE.CG,
368 - STCLASS = SYMREF);
369 -
370 - %EJECT;
371 - IF FLAGS.DCB_CON THEN DO:
372 - IO_BUFFER = 'AU says goodbye';
373 - CALL M$WRITE (FPT_WRITE.CG) ALTRET (REPORT_ERROR_AND_EXIT);
374 - FLAGS.AU_BREAK = %YES#;
375 - RETURN;
376 - END; ELSE DO:
377 - IF FLAGS.TERM_CON THEN DO:
378 - CALL M$DEACTIVATE (FPT_ACTIVATE.CG) ALTRET (REPORT_ERROR_AND_EXIT);
379 - END;
380 - CALL M$CLOSE (FPT_CLOSE.CG) ALTRET (REPORT_ERROR_AND_EXIT);
381 - CALL M$EXIT;
382 - END;
383 - REPORT_ERROR_AND_EXIT:
384 - CALL M$CLOSE (FPT_CLOSE.CG);
385 - CALL M$MERC;
386 - CALL M$EXIT;
387 - END INTERRUPT_HANDLER;
/* 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,

17:22 AUG 29 '85 INT_CGEX_SI2
DCB = M$CG,
FUN = UPDATE,
NAME = VLPNAME,
SCRUB = YES,
SETSTA = VLPSETSTA_DCB

FPTN = FPTREAD.CG,
STCLASS = CONSTANT,
BUF = IN_BUFFER,
DCB = M$CG,
STATION = VLPSTATION_AU

FPTN = FPT_WRITE.CG,
DCB = M$CG,
STATION = VLPSTATION_TERM

FPTN = VLPACCT,
STCLASS = CONSTANT

.ACCT = 'LJSHOST'

FPTN = VLPNAME,
STCLASS = CONSTANT,
NAME = 'INTERNALS.CG'

FPTN = VLPSETSTA_DCB,
STCLASS = CONSTANT,
MYSTATION = 'DCB'

FPTN = VLPSTATION_AU,
STCLASS = CONSTANT,
77 -         DIRONLY = YES,
78 -         MSGTYP = 'TRM_ECHO',
79 -         STATION = 'AU' );
80 -
81 - %VLP_STATION
82 -     ( FPTN = VLP_STATION_TERM,
83 -     MSGTYP = 'TRM_ECHO' );
84 -
85 - %EJECT ;
86 - M$CG$ = DCBADDR ( DCBNUM ( M$CG ) );
87 -
88 - /*
89 -     Open the comgroup.
90 - */
91 -
92 - CALL M$OPEN ( FPTOPEN.CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
93 -
94 - /*
95 -     Read the name of the terminal station.
96 - */
97 -
98 - CALL M$READ ( FPTREAD.CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
99 - IF IN_BUFFER = 'AU says goodbye' THEN DO :
100 -     FLAGS.NO_AU = %YES# ;
101 -     END ; ELSE DO :
102 -     VLPS.TSTN.TERM STATION# = IN_BUFFER ;
103 -  END ;
104 -
105 - /*
106 -     Read the messages from the AU and forward them to the terminal station
107 -     until the AU goes away or the terminal logs off.
108 - */
109 -
110 - DO WHILE NOT FLAGS.TERM_LOGOFF AND NOT FLAGS.NO_AU ;
111 -
112 - CALL M$READ ( FPTREAD.CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
113 - IF ( IN_BUFFER = 'OFF' ) OR ( IN_BUFFER = 'off' ) THEN DO :
114 -     FLAGS.TERM LOGOFF = %YES# ;
115 -     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
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$NWIO.EVID ) ;
  CASE ( %PRINT_READ_EVENT ) ;
    EVENTS.PRINT_COMP = %YES# ;
  CASE ( %CONSOLE_READ_EVENT ) ; /* not currently implemented */
    EVENTS.CONSOLE_COMP = %YES# ;
38 - CASE ( ELSE ) :
39 - EVENTS.BAD_COMP = %YES# ;
40 - END ;
41 - EVENTS.ERR = B$NW1O.ERR ;
42 - EVENTS.TYC = B$NW1O.TYC ;
43 - END ;
44 -
45 - RETURN ;
46 -
47 - END EVENT_HANDLER ;
1 - !JOB NAME=INT_CGLAB_CRU,WSN=UPSTAIRS
2 - !RESOURCE TIME=3,MEM=256
3 - !PL6 INT_CGLAB_SI1 OVER *INT_CGLAB_OU1,INT_CGLAB_LS1
4 - !LINK *INT_CGLAB_OU1 OVER INT_CGLAB_1
5 - !PL6 INT_CGLAB_SI2 OVER *INT_CGLAB_OU2,INT_CGLAB_LS2
6 - !LINK *INT_CGLAB_OU2 OVER INT_CGLAB_2
7 - !C INT_CGLAB_LS1 TO LP
8 - !C INT_CGLAB_LS2 TO LP
17:34 AUG 29 '85 INT_CGLAB_PART_CRU.

1 - JOB_NAME=INT_CGLAB_PART_WSN_UPSTAIRS
2 - RESOURCE TIME=3, MEM=256
3 - PL6 INT_CGLAB_PART_S11 OVER *INT_CGLAB_PART_OU1, INT_CGLAB_PART_LS1
4 - LINK *INT_CGLAB_PART_OU1 OVER INT_CGLAB_PART_1
5 - C INT_CGLAB_PART_LS1 TO LP
/* INT_CGLAB_PART_S11 internals class comgroup lab - S11 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 ;
38 - DCL B$TCB$ PTR SYMREF ;
40 - DCL DATE CHAR ( 8 ) STATIC ;
41 - DCL DCB_NUM UBIN STATIC ;
42 - DCL 1 EVENTS STATIC SYMDEF ,
43 - 2 PRINT_COMP BIT ( 1 ) UNAL INIT ( %NO# ) ,
44 - 2 CONSOLE_COMP BIT ( 1 ) UNAL INIT ( %NO# ) ,
45 - 2 BAD_COMP BIT ( 1 ) UNAL INIT ( %NO# ) ,
46 - 2 ERR BIT ( 36 ) ,
47 - 2 TYC BIT ( 36 ) ;
48 - DCL EXTENSION UBIN STATIC INIT ( 0 ) ;
49 - DCL 1 UBIN ;
50 - DCL IDX SBIN ;
51 - DCL IO_BUFFER CHAR ( 200 ) STATIC ;
52 - DCL IO_BUFFER$ PTR STATIC INIT ( ADDR ( IO_BUFFER ) ) ;
53 - DCL M$CG DCB ;
54 - DCL M$CG$ PTR ;
55 - DCL M$SPECIAL DCB ;
56 - DCL M$SPECIAL$ PTR ;
57 - DCL 1 PRINT_LICENSES ( 0:99 ) STATIC ,
58 - 2 NAME_EXT CHAR ( 12 ) INIT ( ' '*100 ) ,
59 - 2 NAME REDEF NAME_EXT ,
60 - 3 STATION CHAR ( 8 ) ;
61 - 3 EXTENSION CHAR ( 4 ) ;
62 - 2 DCB_NUM UBIN INIT ( 0*100 ) ;
63 - DCL PRINTNUM UBIN STATIC INIT ( 0 ) ;
64 - DCL TIME CHAR ( 11 ) STATIC ;
65 - DCL EVENT_HANDLER ENTRY ASYNC ;
66 - DCL INTERRUPT_HANDLER ENTRY ASYNC ;
67 -
68 - %B$CGAURD
69 - ( FPTN = B$CGAURD ,
70 - STCLASS = "BASED ( IO_BUFFER$ )" ) ;
71 -
72 - %B$ALT ;
73 -
74 - %B$TCB ;
75 -
76 - %EQU.CG ;
77 -
78 - %F$DCB ;
79 -
80 - %FPT_A(CTIVATE
81 - ( FPTN = FPT_4CTIVATE_CG,
82 - STCLASS = STATIC,
83 - DCB = M$CG,
84 - DISCONNECT = YES ) ;
85 -
86 - %FPT_CGCTRL
87 - ( FPTN = FPT_CGCTRL_CG,
88 - DCB = M$CG,
89 - CGCP = VLP_CGCP_SPEC1AL ) ;
90 -
91 - %FPT_CLOSE
92 - ( FPTN = FPT_CLOSE_CG,
93 - STCLASS = STATIC,
94 - DCB = M$CG ) ;
95 -
96 - %FPT_CLOSE
97 - ( FPTN = FPT_CLOSE_INPUT,
98 - DISP = SAVE ) ;
99 -
100 - %FPT_EVE(NT
101 - ( FPTN = FPT_EVENT,
102 - UENTRY = EVENT_HANDLER ) ;
103 -
104 - %FPT_GETDCB
105 - ( FPTN = FPT_GETDCB,
106 - DCBNAME = VLP_NAME_DCB,
107 - DCBNUM = DCB_NUM ) ;
108 -
109 - %FPT_IN( T
110 - ( FPTN = FPT_INT,
111 - UENTRY = INTERRUPT_HANDLER ) ;
112 -
113 - %FPT_OPEN
114 - ( FPTN = FPTOPEN_CG,
115 - 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$CG,
EVENT = %PRINT_READ_EVENT,
STATION = VLP_STATION,
WAIT = NO );
155 -  
156 -  %FPT_READ
157 -  ( FPTN = FPT_READ_SPECIAL,  
158 -  BUF = IO_BUFFER,  
159 -  DCB = M$SPECIAL,  
160 -  STATION = VLP_STATION_SPECIAL ) ;
161 -  
162 -  %FPT_RELDCB
163 -  ( FPTN = FPT_RELDCB ) ;
164 -  
165 -  %FPT_WAIT
166 -  ( FPTN = FPTWAIT_ASECOND,  
167 -  STCLASS = CONSTANT,  
168 -  UNITS = 1 ) ;
169 -  
170 -  %FPT_WRITE
171 -  ( FPTN = FPT_WRITE_INPUT,  
172 -  BUF = IO_BUFFER ) ;
173 -  
174 -  %VLP_CGCP
175 -  ( FPTN = VLP_CGCP_OUTSYM,  
176 -  CONMSG = YES,  
177 -  DCBCONAU = NO,  
178 -  DCBCOMMA = YES,  
179 -  MINPG = 4,  
180 -  RAS = YES ) ;
181 -  
182 -  %VLP_CGCP
183 -  ( FPTN = VLP_CGCP_SPECIAL,  
184 -  CONMSG = NO ) ;
185 -  
186 -  %VLP_NAME
187 -  ( FPTN = VLP_NAME_DCB,  
188 -  LEN = 31 ) ;
189 -  
190 -  %VLP_NAME
191 -  ( FPTN = VLP_NAME_PRINT,  
192 -  LEN = 31 ) ;
193 -  

194 - %VLP_NAME
195 -  ( FPTN     = VLPNAME,
196 -       STCLASS = CONSTANT,
197 -       NAME    = 'INTERNALS_CG' ) ;
198 -
199 - %VLP_ACCT
200 -  ( FPTN     = VLP_ACCT,
201 -       STCLASS  = CONSTANT
202 - */
203 - , ACCT    = 'LJHOST'
204 - */
205 - ) ;
206 -
207 - %VLP_SETSTA
208 -  ( FPTN     = VLP_SETSTA_AU,
209 -       STCLASS  = CONSTANT,
210 -       MYSTATION = 'AU' ) ;
211 -
212 - %VLP_SETSTA
213 -  ( FPTN     = VLP_SETSTA_SPECIAL,
214 -       STCLASS  = CONSTANT,
215 -       MYSTATION = 'SPECIAL' ) ;
216 -
217 - %VLP_STATION
218 -  ( FPTN     = VLP$STATION,
219 -       STCLASS  = BASED ) ;
220 -
221 - %VLP_STATION
222 -  ( FPTN     = VLP_STATION ) ;
223 -
224 - %VLP_STATION
225 -  ( FPTN     = VLP_STATION_SPECIAL,
226 -       EOFNONE  = YES,
227 -       MSGTYP   = 'PRINT ' ) ;
228 -
229 - %VLP_STATION
230 -  ( FPTN     = VLP_STATION_DCB ) ;
231 -
232 - %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$DCB.LASTSTAS$->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 ( 1 ,
             VLP_NAME_PRINT.NAME# ,
             0 ) ;
VLP_NAME_PRINT.L# = 1 ;
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 ,
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$READ ( FPT_READ_SPECIAL )
        WHENALTERNRETURN DO ;
            IF ( B$TCB$->B$TCB_ALT$->B$ALT.ERR.CODE = %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 ) WHENALTERNRETURN DO ; CALL ERROR_EXIT ; END ;
        END ;
    END ;
    FPT_CLOSE_INPUT.V.DCB# = PRINT_FILES.DCB_NUM ( IDX ) ;
    CALL M$CLOSE ( FPT_CLOSE_INPUT ) WHENALTERNRETURN DO ; CALL ERROR_EXIT ; END ;
    FPT_RELDCB.V.DCB# = PRINT_FILES.DCB_NUM ( IDX ) ;
    CALL M$RELDCB ( FPT_RELDCB ) WHENALTERNRETURN 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$GC$->F$DCB.LASTSTA$->VLP$STATION.STATION# ,
        IDX ) ;
    IF ( IDX >= 0 ) THEN DO ;
389 -   FPT_WRITE_INPUT.V.DCB# = PRINT_FILES.DCB_NUM ( IDX ) ;
390 -   FPT_WRITE_INPUT.BUF_BOUND = M$CG$->F$DCB.ARS# - 1 ;
391 -   CALL M$WRITE ( FPT_WRITE_INPUT ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
392 -   END ;
393 -   END ;
394 -
395 -   RETURN ;
396 -
397 -   END PROCESS_READ_EVENT ;
398 -
399 -   %EJECT ;
400 -   GET_NEW_INDEX : PROC ( IDX ) ;
401 -
402 -   /*
403 -     Get an unused entry in the print file table. Returns -1 if none
404 -     are available.
405 -   */
406 -
407 -   DCL IDX SBIN ;
408 -
409 -   DCL I UBIN ;
410 -
411 -
412 -   I = 0 ;
413 -   DO WHILE ( I <= PRINTNUM ) AND ( PRINT_FILES.NAME.STATION ( I ) ~= ' ' ) ;
414 -     I = I + 1 ;
415 -   END ;
416 -
417 -   IF ( I = PRINTNUM ) AND ( PRINTNUM < 100 ) THEN DO ;
418 -     PRINTNUM = PRINTNUM + 1 ;
419 -     IDX = I ;
420 -   END ; ELSE IF ( I < PRINTNUM ) THEN DO ;
421 -     IDX = I ;
422 -   END ; ELSE DO ;
423 -     IDX = -1 ;
424 -   END ;
425 -
426 -   RETURN ;
427 -
428 - END GET_NEW_INDEX ;
429 -
430 - %EJECT ;
431 - GET_PRINTABLE_INDEX : PROC ( IDX ) ;
432 -
433 - /*
434 - Get the index of a printable file - name not blank and DCB number zeroed.
435 - -1 is returned if none exist.
436 - */
437 -
438 - DCL IDX SBIN ;
439 -
440 - DCL I UBIN ;
441 -
442 -
443 -   I = 0 ;
444 -   IDX = -1 ;
445 -   DO WHILE ( I <= PRINTNUM ) AND ( IDX = -1 ) ;
446 -     IF ( PRINT_FILES.NAME.STATION ( I ) == ' ') AND ( PRINT_FILES.DCB_NUM ( I ) = 0 ) THEN DO :
447 -       IDX = I ;
448 -     END ;
449 -     I = I + 1 ;
450 -   END ;
451 -
452 -   RETURN ;
453 -
454 - END GET_PRINTABLE_INDEX ;
455 -
456 - %EJECT ;
457 - GET_STATION_INDEX : PROC ( STATION, IDX ) ;
458 -
459 - /*
460 - Get the index of the given station. -1 is returned if it doesn't exist.
461 - */
462 -
463 - DCL STATION CHAR ( 8 ) ;
464 - DCL IDX SBIN ;
465 -
466 - DCL I UBIN ;
467 -
468 -
469 -     I = 0 ;
470 -     IDX = -1 ;
471 -     DO WHILE ( I <= PRINTNUM ) AND ( IDX = -1 ) ;
472 -         IF ( STATION = PRINT_FILES.NAME.STATION ( I ) ) AND ( PRINT_FILES.DCB_NUM ( I ) = 0 ) THEN IDX = I ;
473 -         END ;
474 -     I = I + 1 ;
475 -     END ;
476 -
477 -     RETURN ;
478 -
479 -     END GET_STATION_INDEX ;
480 -
481 -     %EJECT ;
482 -     ERROR_EXIT : PROC ;
483 -
484 -     CALL XUG$CLOSE_DCBS ( XUGGETCMD ) ;
485 -     CALL M$MERC ;
486 -     CALL M$EXIT ;
487 -
488 -     END ERROR_EXIT ;
489 -
490 -
491 -
492 -
493 -     END OUTSYM_AU ;
494 -
495 -     %EOD ;
496 -     INTERRUPT_HANDLER : PROC ASYNC ;
497 -
498 -  /*
499 -      NAME: INTERRUPT_HANDLER
500 -      PURPOSE: Takes care of interrupts caused by typing break at the AU's
501 -      terminal.
502 -      DESCRIPTION: Closes all DCB's and exits.
503 -   */
504 -
505 -
506 - %INCLUDE CP_6 ;
507 - %INCLUDE CP_6_SUBS ;
508 - %INCLUDE XU_MACRO_C ;
509 - %INCLUDE XUG_ENTRY ;
510 -
511 - %XUG_GETCMD
512 - ( NAME = XUGGETCMD,
513 -    STCLASS = CONSTANT ) ;
514 -
515 - %EJECT ;
516 - CALL XUG$CLOSE_DCBS ( XUGGETCMD ) ;
517 - CALL M$EXIT ;
518 -
519 - END INTERRUPT_HANDLER ;
520 -
521 - %EOD ;
522 - EVENT_HANDLER : PROC ASYNC ;
523 -
524 - /*
525 -  NAME: EVENT_HANDLER
526 -  PURPOSE: Takes care of events caused by IO completion,
527 -  DESCRIPTION: Sets the appropriate flags and returns.
528 - */
529 -
530 -
531 - %INCLUDE CP_6 ;
532 - %INCLUDE CP_6_SUBS ;
533 -
534 - %B$TCB ;
535 -
536 - %B$NWIO
537 - ( STCLASS = "BASED ( B$NWIO$ )" ) ;
538 -
539 - %EQU PRINT_READ_EVENT = 1001 ;
540 - %EQU CONSOLE_READ_EVENT = 1002 ;
541 -
542 - DCL B$NWIO$ PTR ;
543 - DCL B$TCB$ PTR SYMREF ;
544 - DCL 1 EVENTS SYMREF,
545 -  2  PRINT_COMP   BIT ( 1 ) UNAL,
546 -  2  CONSOLE_COMP BIT ( 1 ) UNAL,
547 -  2  BAD_COMP    BIT ( 1 ) UNAL,
548 -  2  ERR         BIT ( 36 ),
549 -  2  TYC         BIT ( 36 );
550 -
551 -  %EJECT ;
552 -  DO INHIBIT ;
553 -  B$NWIO$ = B$TCB$->B$TCB.STK$ ;
554 -  DO CASE ( B$NWIO.EVID ) ;
555 -    CASE ( %PRINT_READ_EVENT ) :
556 -      EVENTS.PRINT_COMP = %YES# ;
557 -    CASE ( %CONSOLE_READ_EVENT ) ; /* not currently implemented */
558 -      EVENTS.CONSOLE_COMP = %YES# ;
559 -    CASE ( ELSE ) :
560 -      EVENTS.BAD_COMP = %YES# ;
561 -  END ;
562 -  EVENTS.ERR = B$NWIO.ERR ;
563 -  EVENTS.TYC = B$NWIO.TYC ;
564 -  END ;
565 -
566 -  RETURN ;
567 -
568 -  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 ;
38 - %INCLUDE CP_6_SUBS ;
39 - %INCLUDE B_ERRORS_C ;
40 - %INCLUDE XU_MACRO_C ;
41 - %INCLUDE XUG_ENTRY ;
42 -
43 - %EQU AU_MESSAGE = '*AUEV ' ;
44 - %EQU FORM_FEED = 12 ;
45 - %EQU PRINT_READ_EVENT = 1001 ;
46 - %EQU CONSOLE_READ_EVENT= 1002 ;
47 -
48 - DCL B$TCB$ PTR SYMREF ;
49 - DCL CUR_PRINTFILE SBIN STATIC INIT ( -1 ) ;
50 - DCL DATE CHAR ( 8 ) STATIC ;
51 - DCL DCB_NUM UBIN STATIC ;
52 - DCL 1 EVENTS STATIC SYMDEF,
53 - 2 PRINT_COMP BIT ( 1 ) UNAL INIT ( %NO# ) ,
54 - 2 CONSOLE_COMP BIT ( 1 ) UNAL INIT ( %NO# ) ,
55 - 2 BAD_COMP BIT ( 1 ) UNAL INIT ( %NO# ) ,
56 - 2 ERR BIT ( 36 ) ,
57 - 2 TYC BIT ( 36 ) ;
58 - DCL EXTENSION UBIN STATIC INIT ( 0 ) ;
59 - DCL I UBIN ;
60 - DCL IDX SBIN ;
61 - DCL IO_BUFFER CHAR ( 200 ) STATIC ;
62 - DCL IO_BUFFER$ PTR STATIC INIT ( ADDR ( IO_BUFFER ) ) ;
63 - DCL M$CG DCB ;
64 - DCL M$CG$ PTR ;
65 - DCL M$SPECIAL DCB ;
66 - DCL M$SPECIAL$ PTR ;
67 - DCL M$PRINT DCB ;
68 - DCL M$PRINT$ PTR ;
69 - DCL PRINT_BUFFER CHAR ( 200 ) STATIC ;
70 - DCL 1 PRINT_FILES ( 0:99 ) STATIC,
71 - 2 NAME_EXT CHAR ( 12 ) INIT ( ' '*100 ) ,
72 - 2 NAME REDEF NAME_EXT ,
73 - 3 STATION CHAR ( 8 ) ,
74 - 3 EXTENSION CHAR ( 4 ) ,
75 - 2 DCB_NUM UBIN INIT ( 0*100 ) ;
76 - DCL PRINTNUM UBIN STATIC INIT ( 0 ) ;
77 - DCL TIME CHAR (11) STATIC;
78 - DCL EVENT_HANDLER ENTRY SYNC;
79 - DCL INTERRUPT_HANDLER ENTRY SYNC;
80 -
81 - %B$CGAURD
82 - ( FPTN = B$CGAURD,
83 - STCLASS = "BASED ( IO_BUFFER$ )" );
84 -
85 - %B$ALT ;
86 -
87 - %B$TCB ;
88 -
89 - %EQU_CG ;
90 -
91 - %F$DCB ;
92 -
93 - %FPT_ACTIVATE
94 - ( FPTN = FPT_ACTIVATE_CG,
95 - STCLASS = STATIC,
96 - DCB = M$CG,
97 - DISCONNECT = YES );
98 -
99 - %FPT_CGCTL
100 - ( FPTN = FPT_CGCTL_CG,
101 - DCB = M$CG,
102 - CGCP = VLP_CGC_SPESIAL );
103 -
104 - %FPT_CLOSE
105 - ( FPTN = FPT_CLOSE_CG,
106 - STCLASS = STATIC,
107 - DCB = M$CG );
108 -
109 - %FPT_CLOSE
110 - ( FPTN = FPT_CLOSE_DELETE,
111 - DCB = M$PRINT,
112 - DISP = RELEASE );
113 -
114 - %FPT_CLOSE
115 - ( 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,
    OISSL = YES,
    SCRUB = YES,
    SETSTA = VLPSETSTA_AU ) ;

%FPT_OPEN
  ( FPTN = FPTOPEN_SPECIAL,
    STCLASS = CONSTANT,
    ACCT = VLPACCT,
    ASN = COMGROUP,
    DCB = M$SPECIAL,
    FUN = UPDATE,
    NAME = VLPNAME,
    SCRUB = YES,
155 - SETSTA = VLPSETSTA_SPECIAL ;
156 -
157 - %FPT_OPEN
158 - ( FPTN = FPT_OPEN_INPUT,
159 - ASN = FILE,
160 - EXIST = NEWFILE,
161 - FUN = CREATE,
162 - NAME = VLP_NAME_PRINT,
163 - ORG = CONSEC,
164 - SCRUB = YES );
165 -
166 - %FPT_OPEN
167 - ( FPTN = FPT_OPEN_OUTPUT,
168 - ASN = FILE,
169 - DCB = M$PRINT,
170 - FUN = IN,
171 - NAME = VLP_NAME_PRINT,
172 - ORG = CONSEC,
173 - SCRUB = YES );
174 -
175 - %FPT_READ
176 - ( FPTN = FPT_READ.CG,
177 - BUF = IO_BUFFER,
178 - DCB = M$CG,
179 - EVENT = %PRINT_READ_EVENT,
180 - STATION = VLP_STATION,
181 - WAIT = NO );
182 -
183 - %FPT_READ
184 - ( FPTN = FPT_READ_SPECIAL,
185 - BUF = IO_BUFFER,
186 - DCB = M$SPECIAL,
187 - STATION = VLP_STATION_SPECIAL );
188 -
189 - %FPT_READ
190 - ( FPTN = FPT_READ_OUTPUT,
191 - BUF = PRINT_BUFFER,
192 - DCB = M$PRINT );
193 -
194 - %FPT_RELD CB
195 - ( FPTN = FPT_RELD CB ) ;
196 -
197 - %FPT_WAIT
198 - ( FPTN = FPTWAIT_ASECOND,
199 - STCLASS = CONSTANT,
200 - UNITS = 1 ) ;
201 -
202 - %FPT_WRITE
203 - ( FPTN = FPT_WRITE.CG,
204 - STCLASS = STATIC,
205 - BUF = PRINT_BUFFER,
206 - DCB = M$CG,
207 - STATION = VLP_STATION TERM ) ;
208 -
209 - %FPT_WRITE
210 - ( FPTN = FPT_WRITE_INPUT,
211 - BUF = IO_BUFFER ) ;
212 -
213 - %VLP.CGCP
214 - ( FPTN = VLP.CGCP_OUTSYM,
215 - CONMSG = YES,
216 - DCBCONAU = NO,
217 - DCBCONWA = YES,
218 - MINPG = 4,
219 - RAS = YES,
220 - TERMCONAU = NO,
221 - TERMCONNAU = NO ) ;
222 -
223 - %VLP.CGCP
224 - ( FPTN = VLP.CGCP_SPECIAL,
225 - CONMSG = NO ) ;
226 -
227 - %VLP_NAME
228 - ( FPTN = VLP_NAME_DCB,
229 - LEN = 31 ) ;
230 -
231 - %VLP_NAME
232 - ( 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 )
272 -
273 - %VLP_STATION
274 - ( FPTN        = VLP_STATION_DCB ) ;
275 -
276 - %XUG_GETCMD
277 - ( NAME        = XUGGETCMD ,
278 -     STCLASS    = CONSTANT ) ;
279 -
280 - %EJECT ;
281 - /*
282 -   Set break control.
283 - */
284 -
285 - CALL M$INT ( FPT_INT ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
286 -
287 - M$CG$ = DCBADDR ( DCBNUM ( M$CG  ) ) ;
288 - M$PRINT$ = DCBADDR ( DCBNUM ( M$PRINT ) ) ;
289 - M$SPECIAL$ = DCBADDR ( DCBNUM ( M$SPECIAL ) ) ;
290 -
291 - /*
292 -   Create a new comgroup and modify its control parameters.
293 -   The first CGCTL specifies CONMSG=NO so the SPECIAL station can connect
294 -   without requiring activation. After that, another CGCTL is done specifying
295 -   the normal parameters that we should use.
296 - */
297 -
298 - CALL M$OPEN ( FPTOPEN.CG ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
299 - CALL M$CGCTL ( FPT.CGCTL.CG ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
300 - CALL M$OPEN ( FPTOPEN.SPECIAL ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
301 - FPT.CGCTL.CG.CGCP = VECTOR ( VLP.CGCP_OUTSYM ) ;
302 - CALL M$CGCTL ( FPT.CGCTL.CG ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
303 -
304 - /*
305 - Set event control.
306 - */
307 -
308 - CALL M$EVENT ( FPT.EVENT ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
309 -
310 - /*
311 -   Start the first no-wait read.
312 -   */
313 -   
314 -   CALL M$READ ( FPT_READ.CG ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
315 -   
316 -   /*
317 -   Start normal operation of getting PRINT files and printing them if possible.
318 -   */
319 -   
320 -   DO WHILE %YES# ;
321 -   
322 -   DO INHIBIT :
323 -   DO WHILE EVENTS.PRINT_COMP ;
324 -   CALL PROCESS_READ_EVENT ;
325 -   EVENTS.PRINT_COMP = %NO# ;
326 -   IO_BUFFER = ' ' ;
327 -   CALL M$READ ( FPT_READ.CG ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
328 -   END ;
329 -   IF ( VLP_STATION_TERM.STATION# ~ = ' ' ) THEN DO ;
330 -   CALL PREPARE_PRINT_RECORD
331 -   WHENRETURN DO :
332 -   CALL M$WRITE ( FPT_WRITE.CG ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
333 -   END ; ELSE DO :
334 -   CALL M$WAIT ( FPTWAIT_ASECOND ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
335 -   END ;
336 -   END ;
337 -   END ;
338 -   END ;
339 -   %EJECT ;
340 -   
341 -   PREPARE_PRINT_RECORD : PROC ALTRET ;
342 -   
343 -   /*
344 -   Puts a record in the buffer to send to the "printer", if we have any files
345 -   that are ready to be printed.
346 -   */
347 -   
348 -   */
349 -   

/*

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 ( 1,
                 ' ',
                 VLP_NAME_PRINT.NAME#, 0 );
   VLP_NAME_PRINT.L# = 1;
   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$TCB.ALTS->B$ALT.ERR.CODE = %E$EOF ) THEN DO;
  EOF = %YES#;
END; ELSE DO;
  CALL ERROR_EXIT ; /* DOESN'T RETURN */
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;

FPT_CLOSE_INPUT.V.DCB# = PRINT_FILES.DCB_NUM ( IDX ) ;
CALL M$CLOSE ( FPT_CLOSE_INPUT ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END;
FPT_RELDLCB.V.DCB# = PRINT_FILES.DCB_NUM ( IDX ) ;
CALL M$RELDLCB ( FPT_RELDLCB ) 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$CGAUDR.STATION ;
  FPT_ACTIVATE.CG.STATION_ = VECTOR ( VLP_STATION_TERM ) ;
  CALL M$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$CGAUDR.STATION ) THEN DO ;
    VLP_STATION_TERM.STATION# = ' ' ;
  IF M$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$STATION.STATION#, IDX ) ;
  IF ( IDX >= 0 ) THEN DO ;
    FPT_WRITE_INPUT.V.DCB# = PRINTFILES.DCB_NUM ( IDX ) ;
    FPT_WRITE_INPUT.BUF_BOUND = M$CG$->F$DCB.ARS# - 1 ;
    CALL M$WRITE ( FPT_WRITE_INPUT ) WHENALTRETURN DO ; CALL ERROR_EXIT ; END ;
END ;
RETURN ;
END PROCESS_READ_EVENT ;
545 -
546 - %EJECT ;
547 - GET_NEW_INDEX : PROC ( IDX ) ;
548 -
549 - /*
550 - Get an unused entry in the print file table. Returns -1 if none
551 - are available.
552 - */
553 -
554 - DCL IDX SBIN ;
555 -
556 - DCL I UBIN ;
557 -
558 -
559 - I = 0 ;
560 - DO WHILE ( I <= PRINTNUM ) AND ( PRINT_FILES.NAME.STATION ( I ) == ' ' ) ;
561 - I = I + 1 ;
562 - END ;
563 -
564 - IF ( I = PRINTNUM ) AND ( PRINTNUM < 100 ) THEN DO ;
565 - PRINTNUM = PRINTNUM + 1 ;
566 - IDX = I ;
567 - END ; ELSE IF ( I < PRINTNUM ) THEN DO ;
568 - IDX = I ;
569 - END ; ELSE DO ;
570 - IDX = -1 ;
571 - END ;
572 - RETURN ;
573 - END GET_NEW_INDEX ;
574 -
575 - %EJECT ;
576 -
577 - GET_PRINTABLE_INDEX : PROC ( IDX ) ;
578 -
579 - /*
580 - Get the index of a printable file. - name not blank and DCB number zeroed.
581 - -1 is returned if none exist.
582 - */
584 - DCL    IDX    SBIN  ;
585 -
586 -
587 - DCL    i    UBIN  ;
588 -
589 -
590 -    i = 0  ;
591 -    IDX = -1  ;
592 -    DO WHILE ( i <= PRINTNUM ) AND ( IDX = -1 ) ;
593 -    IF ( PRINT_FILES.NAME.STATION ( i ) != ' ' ) AND ( PRINT_FILES.DCB_NUM ( i ) = 0 ) THEN DO :
594 -        IDX = i  ;
595 -    END  ;
596 -    i = i + 1  ;
597 -    END  ;
598 -
599 -    RETURN  ;
600 -
601 -    END GET_PRINTABLE_INDEX  ;
602 -
603 -    %EJECT  ;
604 -    GET_STATION_INDEX : PROC ( STATION, IDX )  ;
605 -
606 -    /*
607 -    Get the index of the given station. -1 is returned if it doesn't exist.
608 -    */
609 -
610 - DCL    STATION    CHAR ( 8 )  ;
611 - DCL    IDX    SBIN  ;
612 -
613 - DCL    i    UBIN  ;
614 -
615 -
616 -    i = 0  ;
617 -    IDX = -1  ;
618 -    DO WHILE ( i <= PRINTNUM ) AND ( IDX = -1 ) ;
619 -    IF ( STATION = PRINT_FILES.NAME.STATION ( i ) ) AND ( PRINT_FILES.DCB_NUM ( i ) == 0 ) THEN DO :
620 -        IDX = i  ;
621 -    END  ;
622 -    i = i + 1  ;
623 -     END ;
624 -
625 -     RETURN ;
626 -
627 -     END GET_STATION_INDEX ;
628 -
629 -     %EJECT ;
630 -     ERROR_EXIT : PROC ;
631 -
632 -     IF ( VLP_STATION_TERM.STATION# ^= "" ) THEN DO :
633 -         FPT_ACTIVATE.CG.STATION_ = VECTOR ( VLP_STATION_TERM ) ;
634 -         CALL MSDEACTIVATE ( FPT_ACTIVATE.CG ) WHENALTRETURN DO ; END ;
635 -     END ;
636 -     CALL XUG$CLOSE_DCBS ( XUGGETCMD ) ;
637 -     CALL MS%MERC ;
638 -     CALL MS$EXIT ;
639 -
640 -     END ERROR_EXIT ;
641 -
642 -
643 -  *
644 -     END OUTSYM_AU ;
645 -
646 -     %EOD ;
647 -     INTERRUPT_HANDLER : PROC ASYNC ;
648 -
649 -     /*
650 -         NAME: INTERRUPT_HANDLER
651 -         PURPOSE: Takes care of interrupts caused by typing break at the AU's
652 -         terminal.
653 -         DESCRIPTION: Closes all DCB's and exits.
654 -     */
655 -
656 -
657 -     %INCLUDE CP_6 ;
658 -     %INCLUDE CP_6_SUBS ;
659 -     %INCLUDE XU_MACRO_C ;
660 -     %INCLUDE XUG_ENTRY ;
661 -
662 - %XUG_GETCMD
663 - ( NAME = XUG_GETCMD, 
664 -         STCLASS = CONSTANT );
665 -
666 - %EJECT ;
667 - CALL XUG$CLOSE_DCBS ( XUG_GETCMD ) ;
668 - CALL M$EXIT ;
669 -
670 - END INTERRUPT_HANDLER ;
671 -
672 - %EOD ;
673 - EVENT_HANDLER : PROC ASYNC ;
674 -
675 - /*
676 -        NAME: EVENT_HANDLER
677 -        PURPOSE: Takes care of events caused by IO completion.
678 -        DESCRIPTION: Sets the appropriate flags and returns.
679 - */
680 -
681 -
682 - %INCLUDE CP_6 ;
683 - %INCLUDE CP_6_SUBS ;
684 -
685 - %B$TCB ;
686 -
687 - %B$NVI0 ;
688 - ( STCLASS = "BASED ( B$NVI0$ )" );
689 -
690 - %EQU PRINT_READ_EVENT = 1001 ;
691 - %EQU CONSOLE_READ_EVENT = 1002 ;
692 -
693 - DCL B$NVI0$ PTR ;
694 - DCL B$TCB$ PTR SYMREF ;
695 - DCL EVENTS SYMREF ,
696 - 2 PRINT_COMP BIT ( 1 ) UNAL ,
697 - 2 CONSOLE_COMP BIT ( 1 ) UNAL ,
698 - 2 BAD_COMP BIT ( 1 ) UNAL ,
699 - 2 ERR BIT ( 36 ) ,
700 - 2 TYC BIT ( 36 ) ;
701 -
702 - %EJECT ;
703 - DO INHIBIT :
704 - B$NWIO$ = B$TCB$->B$TCB.STK$ ;
705 - DO CASE ( B$NWIO.EVID$ ) :
706 - CASE ( %PRINT_READ_EVENT ) :
707 - EVENTS.PRINT_COMP = %YES# ;
708 - CASE ( %CONSOLE_READ_EVENT ) ; /* not currently implemented */
709 - EVENTS.CONSOLE_COMP = %YES# ;
710 - CASE ( ELSE ) :
711 - EVENTS.BAD_COMP = %YES# ;
712 - END ;
713 - EVENTS.ERR = B$NWIO.ERR$ ;
714 - EVENTS.TYC = B$NWIO.TYC$ ;
715 - END ;
716 -
717 - RETURN ;
718 -
719 - END EVENT_HANDLER ;
//M* INT_CGLEX_SL2 internals class comgroup example - SL2 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_CGLEX_SL2. (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$DCB ;
38 -
39 - %FPT_CLOSE
40 - ( FPTN = FPTCLOSE_CG,
41 - STCLASS = CONSTANT,
42 - DCB = M$CG ) ;
43 -
44 - %FPT_CLOSE
45 - ( FPTN = FPTCLOSE_PRINT,
46 - STCLASS = CONSTANT,
47 - DCB = M$PRINT ) ;
48 -
49 - %FPT_FID
50 - ( FPTN = FPT_FID_PRINT,
51 - ACCT = VLP_ACCT_PRINT,
52 - ASN = JUNK,
53 - NAME = VLP_NAME_PRINT,
54 - PASS = JUNK,
55 - RES = JUNK,
56 - RESULTS = JUNK,
57 - SN = JUNK,
58 - WSN = JUNK ) ;
59 -
60 - %FPT_OPEN
61 - ( FPTN = FPTOPEN_CG,
62 - STCLASS = CONSTANT,
63 - ACCT = VLPACCT,
64 - ASN = COMGROUP,
65 - DCB = M$CG,
66 - FUN = UPDATE,
67 - NAME = VLPNAME,
68 - SCRUB = YES,
69 - SETSTA = VLP_SETSTA_DCB ) ;
70 -
71 - %FPT_OPEN
72 - ( FPTN = FPTOPEN_PRINT,
73 - STCLASS = CONSTANT,
74 - ACCT = VLP_ACCT_PRINT,
75 - ASN = FILE,
76 - DCB = M$PRINT,
77 - FUN = IN,
78 - NAME = VLP_NAME_PRINT,
79 - ORG = CONSEC,
80 - SCRUB = YES ;
81 -
82 - %FPT_READ
83 - ( FPTN = FPT_READ_PRINT,
84 - BUF = IO_BUFFER,
85 - DCB = M$PRINT ) ;
86 -
87 - %FPT_WRITE
88 - ( FPTN = FPT_WRITE.CG,
89 - BUF = IO_BUFFER,
90 - DCB = M$CG,
91 - STATION = VLPSTATION_AU ) ;
92 -
93 - %VLP_ACCT
94 - ( FPTN = VLPACCT,
95 - STCLASS = CONSTANT
96 - /*
97 - ACCT = 'LJSHOST'
98 - */
99 - ) ;
100 -
101 - %VLP_ACCT
102 - ( FPTN = VLP_ACCT_PRINT ) ;
103 -
104 - %VLP_NAME
105 - ( FPTN = VLPNAME,
106 - STCLASS = CONSTANT,
107 - NAME = 'INTERNALS.CG' ) ;
108 -
109 - %VLP_NAME
110 - ( FPTN = VLP_NAME_PRINT ) ;
111 -
112 - %VLP_SETSTA
113 - ( FPTN = VLP_SETSTA_DCB ) ;
114 -
115 - %VLP_STATION
( FPTN = 'VLPSTATION_AU',
  STCLASS = 'CONSTANT',
  ANYDCB = 'YES',
  MSGTYP = 'PRINT' )

%!XUX$PARAM_NOPARSE
( 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 BINFCHAR ( 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,
  POS1 - 1 ) ) ;

CALL M$FID ( FPT_FID_PRINT ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;

CALL M$OPEN ( FPTOPEN_PRINT ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
155 - 
156 - /*
157 - Open the comgroup.
158 - */
159 - 
160 - CALL M$OPEN ( FPTOPEN.CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
161 - 
162 - /*
163 - Read the messages from the print file and send them to the AU station.
164 - */
165 - 
166 - DO WHILE NOT EOF ;
167 - 
168 - CALL M$READ ( FPT_READ_PRINT )
169 - WHENALTRETURN DO :
170 - IF ( B$TCB$->B$TCB.ALT$->B$ALT.ERR.CODE = %E$EOF ) THEN DO :
171 - EOF = %YES# ;
172 - END ; ELSE DO :
173 - GOTO REPORT_ERROR_AND_EXIT ;
174 - END ;
175 - END ;
176 - IF NOT EOF THEN DO :
177 - FPT_WRITE.CG.BUF_BOUND = M$PRINT$->F$DCB.ARS# - 1 ;
178 - CALL M$WRITE ( FPT_WRITE.CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
179 - END ;
180 - 
181 - END ;
182 - 
183 - CALL M$CLOSE ( FPTCLOSE.CG ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
184 - CALL M$CLOSE ( FPTCLOSE_PRINT ) ALTRET ( REPORT_ERROR_AND_EXIT ) ;
185 - CALL M$EXIT ;
186 - 
187 - REPORT_ERROR_AND_EXIT ;
188 - IF M$CG$->F$DCB.FCD# THEN DO :
189 - CALL M$CLOSE ( FPTCLOSE.CG ) ;
190 - END ;
191 - IF M$PRINT$->F$DCB.FCD# THEN DO :
192 - CALL M$CLOSE ( FPTCLOSE_PRINT ) ;
193 - END ;
194 -      CALL M$MERC ;
195 -
196 -  XIT: ;
197 -      CALL M$EXIT ;
198 -
199 -  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, INTCPTTEST. 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.
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.
17:46 AUG 29 '85 LAB5_JCL.

1 - !JOB WSN-UPSTAIRS
2 - !RES MEM=300,TIME=2
3 - !PL6 LAB5_CP_SI OVER LAB5_CP_OU6(LS,SR(.:LIBRARY),SCHEMA)
4 - !LINK LAB5_CP_OU6 OVER INTCP(MAP(VALUE,NAMEN),NOSH,SYN, ;
5 - UNSAT=:LIB_SYSTEM.:SYS,TCB=3,PRIV(JIT))
6 - !PL6 LAB5_CPTEST_SI OVER LAB5_CPTEST_OU6(LS,SR(.:LIBRARY),SCHEMA)
7 - !LINK LAB5_CPTEST_OU6 OVER INTCP
8 - !SPIDER
9 - DEL INTCP
10 - IN INTCP,CP FROM INTCP.CKWHOST
11 - L INTCP
12 - END
13 - !C LAB5_JCL TO LP(K)
U40CPTEST: PROC MAIN;
  %INCLUDE CP_6;
  %B$TCB;
  %B$ALT;
  %INCLUDE B_ERRORS_C;
  DCL M$S1 DCB;
  DCL B$TCB$ PTR SYMREF;
  %FPT_READ (FPTN=FPT$READ,STCLASS=CONSTANT,DCB=M$S1,BUF=BUFFER);
  DCL BUFFER CHAR(120) STATIC;

  REREAD: 
  CALL M$READ(FPT$READ)
  WHENALTRETURN DO;
  IF B$TCB$ -> B$TCB.ALTS -> B$ALT.ERR.CODE = %E$EOF
  THEN GOTO XIT;
  ELSE GOTO REREAD;
  END;
  GOTO REREAD;
XIT: 
  CALL M$EXIT;
END U40CPTEST;
1 - U40CP: PROC MAIN;
2 - %INCLUDE CP_6;
3 - %INCLUDE B$JIT;
4 -
5 - DCL B$JIT $ PTR SYMREF;
6 - DCL B$TCB $ PTR SYMREF;
7 - %FPT_READ (FPTN=FPT$READ, STCLASS=CONSTANT, DCB=M$UC);
8 - %FPT_READ (STCLASS=AUTO);
9 - %FPT_WRITE (FPTN=WRITE_SYNERR, STCLASS=CONSTANT, BUF=SYNERR, DCB=M$UC);
10 - DCL SYNERR CHAR(0) CONSTANT INIT('U40CP can't do that.');
11 - %FPT_PROMPT (FPTN=PROMPT_SINGLE, PROMPT=SINGLE_SPLAT, STCLASS=CONSTANT, VFC=NO);
12 - %FPT_PROMPT (FPTN=PROMPT_DOUBLE, PROMPT=DOUBLE_SPLAT, STCLASS=CONSTANT, VFC=NO);
13 - DCL SINGLE_SPLAT CHAR(2) CONSTANT INIT('p#');
14 - DCL DOUBLE_SPLAT CHAR(3) CONSTANT INIT('p##');
15 - %FPT_CPEXIT (FPTN=FPT$CPEXIT, STCLASS=CONSTANT);
16 - %FPT_CPEXIT (STCLASS="");
17 - %VLP_ACCT (FPTN=ACCT_SYS, ACCT=':SYS ', STCLASS=CONSTANT);
18 - %VLP_NAME (FPTN=NAME_DELTA, NAME='DELTA', STCLASS=CONSTANT);
19 - %VLP_NAME (FPTN=NAME_PCL, NAME='PCL', STCLASS=CONSTANT);
20 - %VLP_NAME (FPTN=NAME_U40CPTEST, NAME='U40CPTEST', STCLASS=CONSTANT);
21 - %VLP_NAME (FPTN=NAME_IBEX, NAME='IBEX', STCLASS=CONSTANT);
22 - %FPT_ERRMSG (FPTN=FPT$ERRMSG, STCLASS=CONSTANT, SOURCE=PASS, OUTDCB1=M$UC);
23 - %FPT_ERRMSG (STCLASS=AUTO);
24 - DCL BUFFER CHAR(120);
25 - %F$DCB;
26 - %B$TCB;
27 - %B$ALT;
28 -
29 - FPT_CPEXIT = FPT$CPEXIT; FPT_CPEXIT.V_ = VECTOR (FPT_CPEXIT.V);
30 - FPT_READ = FPT$READ; FPT_READ.V_ = VECTOR (FPT_READ.V);
31 - FPT_READ.BUF_ = VECTOR (B$JIT.CCBUF);
32 - FPT_ERRMSG = FPT$ERRMSG; FPT_ERRMSG.V_ = VECTOR (FPT_ERRMSG.V);
33 - FPT_ERRMSG.BUF_ = VECTOR (BUFFER);
34 - IF B$JIT.CPFLAGS1 & %CP_JSTEP* THEN DO;
35 -   CALL M$PROMPT (PROMPT_SINGLE);
36 -   READ_SING: CALL M$READ (FPT_READ)
37 - 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$JIT.CCARS));
SELECT ('DELTAB');
FPT_CPEXIT.DEBUG_ = VECTOR (NAME_DELTA);
FPT_CPEXIT.V.CONT# = '1'B;
SELECT('GO');
    FPT_CPEXIT.V.CONT# = '1'B;
SELECT('QUIT');
    QUIT: FPT_CPEXIT.V.QUIT# = '1'B;
SELECT(ELSE);
    CALL M$WRITE(WRITE_SYNER);  
    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 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.
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 Fp 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
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_S11 OVER LAB6_HOST_OU61(LS,SR(.,LIBRARY),SCHEMA)
4 - !LINK LAB6_HOST_OU61 OVER LAB6:H1
5 - !PL6 LAB6_FPRG_S11 OVER LAB6_FPRG_OU61(LS,SR(.,LIBRARY),SCHEMA)
6 - !FEPLINK LAB6_FPRG_OU61 OVER LAB6:F1
7 - !PL6 LAB6_HOST_S12 OVER LAB6_HOST_OU62(LS,SR(.,LIBRARY),SCHEMA)
8 - !LINK LAB6_HOST_OU62 OVER LAB6:H2
9 - !PL6 LAB6_FPRG_S12 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_S11: 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.VTYPE = %G_CVM_MON#;
  IF USRT_ADDR > (1024*64)
    THEN SEGOFFSET = MOD(USRT_ADDR, 1024*64);
  ELSE SEGOFFSET = MOD(USRT_ADDR, 1024*4);
38 - FSEG.BASE$ = PINCRW(NIL$, USRT_ADDR - SEGOFFSET);
39 - CALL M$CVM (FPT_CVM);
40 - HOST_BUF.SYSID = PINCRW(G$DS4$, SEGOFFSET) -> G$USER.SYSID(1);
41 - CALL M$WRITE (HOST_WRITE);
42 -
43 - XIT:
44 - CALL M$EXIT;
45 - END LAB6_FPRG_S11;
LAB6_FPRG_S12: PROC MAIN;
  %INCLUDE LCP_6;
  DCL M$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$EXIT;
END LAB6_FPRG_S12;
LAB6_HOST_S11: PROC MAIN;
  %INCLUDE CP_6;
  DCL B$JIT$ PTR SYMREF;
  %INCLUDE B$JIT;
  DCL FPRG DCB;
  DCL TEMP UBIN STATIC;
  DCL 1 TEMP1 REDEF TEMP,
       2 * BIT(20),
       2 Y1 UBIN(8) UNAL,
       2 Y2 UBIN(8) UNAL;
  DCL FPRG_BUF(0:9) UBIN BYTE CALIGNED STATIC INIT(0*0);
  DCL 1 BUF STATIC,
       2 * CHAR(5) INIT('FEP#='),
       2 FEP# CHAR(3),
       2 * CHAR(11) INIT(' NODENAME='),
       2 NODENAME(0:7) CHAR(1),
       2 NODENAME1(0:7) REDEF NODENAME UBIN BYTE UNAL;
  DCL 1 BUF1 STATIC,
       2 * CHAR(6) INIT('SYSID='),
       2 SYSID CHAR(3);
  DCL 1 UBIN;
  %FPT_WRITE (DCB=M$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);
38 -    TEMP = 0;
39 -    TEMP1.Y1 = FPRG_BUF(0);
40 -    TEMP1.Y2 = FPRG_BUF(1);
41 -    CALL BINCHAR(BUF1.SYSID, TEMP);
42 -    UC_WRITE.BUF_ = VECTOR(BUF1);
43 -    CALL M$WRITE (UC_WRITE);
44 -    END LAB6_HOST_S11;
1 - LAB6_HOST_S12: PROC MAIN;
2 - INCLUDE CP_6;
3 - DCL B$JIT$ PTR SYMREF;
4 - INCLUDE B$JIT;
5 - %F$DCB;
6 - DCL FPRG DCB;
7 - DCL I UBIN;
8 - DCL SIZE UBIN;
9 - DCL CHAR_BUF(0:79) CHAR CALIGNED STATIC INIT(' ');
10 - DCL CHAR_BUF1 REDEF CHAR_BUF CHAR(80) CALIGNED;
11 - DCL FPRG_BUF(0:80) CHAR(1) CALIGNED STATIC;
12 - DCL FPRG_BUF1(0:80) REDEF FPRG_BUF UBIN BYTE CALIGNED;
13 - %FPT_WRITE (DCB=M$UC,FPTN=UC_WRITE,BUF=CHAR_BUF);
14 - %FPT_WRITE (DCB=FPRG,FPTN=FPRG_WRITE,BUF=FPRG_BUF);
15 - %FPT_OPEN (DCB=FPRG, ORG=FPRG, RES='UC02', FPRG=VLP_FPRG);
16 - %VLP_FPRG (NAME='LAB6:F2');
17 - %FPT_READ (FPTN=FPRG_READ,BUF=FPRG_BUF,DCB=FPRG);
18 - %FPT_READ (FPTN=UC_READ,BUF=CHAR_BUF,DCB=M$UC);
19 - %FPT_LDEV(FPTN=LDEV_FPRG,STREAMNAME='UC02',WINDOW=VLP_WINDOW_FPRG);
20 - %VLP_WINDOW(FPTN=VLP_WINDOW_FPRG,FWINDOW='UC01',LENGTH=6,WIDTH=80,
21 - POSITION=TOP);
22 - %FPT_LDEV(FPTN=LDEV_DELTA,STREAMNAME='UC99',WINDOW=VLP_WINDOW_DELTA);
23 - %VLP_WINDOW(FPTN=VLP_WINDOW_DELTA,FWINDOW='UC01',LENGTH=8,WIDTH=80,
24 - POSITION=BOTTOM);
25 -
26 - CALL M$LDEV (LDEV_FPRG);
27 - CALL M$LDEV (LDEV_DELTA);
28 - CALL M$OPEN (FPT_OPEN);
29 - CALL M$READ (UC_READ);
30 - SIZE = DCBADDR (DCBNUM(M$UC)) -> F$DCB.ARSH;
31 - FPRG_BUF1(0) = SIZE;
32 - DO I = 1 TO SIZE+1;
33 -   FPRG_BUF(I) = CHAR_BUF(I-1);
34 - END;
35 - CALL M$WRITE (FPRG_WRITE);
36 - CALL M$READ (FPRG_READ);
37 - SIZE = FPRG_BUF1(0);
38 - CHAR_BUF1 = ' ';  
39 - DO I = 1 TO SIZE+1;  
40 - CHAR_BUF(I-1) = FPRG_BUF(I);  
41 - END;  
42 - CALL M$WRITE (UC_WRITE);  
43 - END LAB6_HOST_S12;
CO1 - ANLZ Command Summary - FEPs
CO2 - REC Command - FEPs

CO3 - Monitor Trap Example
CO4 - Handler Trap Example

CO5 - LCP6 Trap Numbers
CO6 - MCL Codes
CO7 - ASDTs
CO8 - Users HJIT
CO9 - Low Real Memory
C10 - Status Register
ANLS Commands for FEPs

. Combinations of dumps/ANLZ
   - COO dump / COO ANLZ
   - COO dump / Bootleg ANLZ
   - CO1 dump / CO1 ANLZ

ANLZ FEP Commands - Available on COO

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 CO1 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 routing 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.

CO1 ANLZ cannot be used with COO 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 COO in any event.
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 XDELTAA
.0C through .3E  - Its a handler   - use INTOCON 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:FEPI:SYS is the right one
   Say PL .4000 to see TSTACKU

If Handler   - use INTOCON 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
   - E3 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 COO, however. CO1 gets it right.

  ANLZ doesn't understand CUN until after you have said REC
  if using COO 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

Secret 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$LG+.4D9

ISA:


Interrupted address: GMA$LG+.4D9

+---+---+---+---+---+---+---+
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
+---+---+---+---+---+---+---+
B Registers:  506E  9D10  452D  C0040  9003E  90000  4AE3
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

Trap IC: GFM$MCL+.18
ISA:

P: .20050   S: .0000   CHN: .0000   ISM1: .FFFF   ISM2: .8103

Interrupted address: GFM$MCL+.18

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>B Registers:</td>
<td>0</td>
<td>0</td>
<td>10288</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>R Registers:</td>
<td>0</td>
<td>0</td>
<td>902</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>M Registers:</td>
<td>FF00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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 .10288,6 F CUN
        0 1 2 3 4 5 6 7
010298  000C 0000 0000 0001 0001 029E ...............          
L6XII   -SPY CUN
        Usr# Identification  Sysid  CPU  M$LM
        G .5 LLHOST,104AVERY  11  0:00  QGDS.LLHOST

L6XII   -JIT CUN
JIT for User# .05 Sysid 11 Mode: Ghost  Prog entry: M$SETFP  LLHOST,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

MCL: 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  CURRSSZ .0000

Altret frame at .012C9  ECC: .0063  PREVSZ: .0000
The page contains a listing of registers and flags, along with internal data structures and instructions. The document includes a message in English that reads: "That page does not belong to you."
### L6XII -SPY

<table>
<thead>
<tr>
<th>Usr#</th>
<th>Identification</th>
<th>Sysid</th>
<th>CPU</th>
<th>M$LM</th>
</tr>
</thead>
<tbody>
<tr>
<td>H 1</td>
<td>SYS.NODEADMN</td>
<td>1</td>
<td>2:17</td>
<td>NODEADMN:SYS</td>
</tr>
<tr>
<td>H 2</td>
<td>SYS.HDLCX25</td>
<td>2</td>
<td>5:34</td>
<td>HDLCX25:SYS</td>
</tr>
<tr>
<td>H 3</td>
<td>SYS.ASYNC</td>
<td>3</td>
<td>9:51</td>
<td>ASYNC:SYS</td>
</tr>
<tr>
<td>G 4</td>
<td>LLAHOST, 104AVERY</td>
<td>8</td>
<td>0:00</td>
<td>ANLZ_FPRG:SYS</td>
</tr>
<tr>
<td>G 5</td>
<td>LLAHOST, 104AVERY</td>
<td>11</td>
<td>0:00</td>
<td>QGDS.LLAHOST</td>
</tr>
</tbody>
</table>

### L6XII -USRT

<table>
<thead>
<tr>
<th>* FL</th>
<th>BL</th>
<th>Sysid</th>
<th>Mode</th>
<th>State</th>
<th>Flags</th>
<th>Async</th>
<th>DL$</th>
<th>MISC</th>
<th>HMI$</th>
</tr>
</thead>
<tbody>
<tr>
<td>.01</td>
<td>.00</td>
<td>.03</td>
<td>1</td>
<td>H</td>
<td>SW</td>
<td>.000</td>
<td>.000</td>
<td>.0000000008E4F</td>
<td>.C1B86</td>
</tr>
<tr>
<td>.02</td>
<td>.03</td>
<td>.00</td>
<td>2</td>
<td>H</td>
<td>SW</td>
<td>.000</td>
<td>.000</td>
<td>.0000000008A06</td>
<td>.C1B9F</td>
</tr>
<tr>
<td>.03</td>
<td>.01</td>
<td>.02</td>
<td>3</td>
<td>H</td>
<td>SW</td>
<td>.000</td>
<td>.000</td>
<td>.0000000008E07</td>
<td>.C1BB8</td>
</tr>
<tr>
<td>.04</td>
<td>.00</td>
<td>.00</td>
<td>8</td>
<td>G</td>
<td>SCI</td>
<td>.0800</td>
<td>.000</td>
<td>.00000000C1542</td>
<td>.00000</td>
</tr>
<tr>
<td>.05</td>
<td>.00</td>
<td>.00</td>
<td>11</td>
<td>G</td>
<td>SCU</td>
<td>.0000</td>
<td>.000</td>
<td>.00000000C16D2</td>
<td>.00000</td>
</tr>
</tbody>
</table>

### L6XII -SFILE

<table>
<thead>
<tr>
<th>* File Identification</th>
<th>Flags</th>
<th>Modtime</th>
<th>Instime</th>
<th>UC</th>
<th>FRQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELTA_F:SYS</td>
<td>.0800</td>
<td>780101</td>
<td>0000</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SHARED_LCP6_SYSTEM:SYS</td>
<td>.4000</td>
<td>780101</td>
<td>0000</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>ANLZ_FPRG:SYS</td>
<td>.A080</td>
<td>781016</td>
<td>1214</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>QGDS.LLAHOST</td>
<td>.8080</td>
<td>781017</td>
<td>0101</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### L6XII -INTCON

<table>
<thead>
<tr>
<th>* TIMER</th>
<th>TSAP$</th>
<th>P$</th>
<th>IENTRY</th>
<th>DEV</th>
<th>S</th>
<th>HHJIT</th>
<th>USR</th>
<th>LVL</th>
<th>FLGS</th>
<th>ISM1</th>
<th>ISM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>.20</td>
<td>.0000</td>
<td>.0000</td>
<td>.094PA</td>
<td>0000</td>
<td>0000</td>
<td>.03</td>
<td>.00</td>
<td>.8400</td>
<td>0000</td>
<td>2000</td>
<td></td>
</tr>
</tbody>
</table>
L6XII -MEMORY

FEP Memory Utilization

| MONITOR:BIGFOOT                      | 0  .0000 |
| MONITOR:Procedure and Static Data   | 279 .0117 |
| MONITOR:Ctx-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 |

<table>
<thead>
<tr>
<th>FPRGS :User Procedure</th>
<th>USER</th>
<th>COMGP</th>
<th>HAND</th>
<th>GHOST</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPRGS :User Data-Static and Dynamic</td>
<td>0</td>
<td>0</td>
<td>288</td>
<td>0</td>
</tr>
<tr>
<td>FPRGS :Context-TSTACKM,ROS,UMHJIT,HHJITs</td>
<td>0</td>
<td>0</td>
<td>495</td>
<td>7</td>
</tr>
<tr>
<td>FPRGS :Handler Q</td>
<td>0</td>
<td>0</td>
<td>99</td>
<td>0</td>
</tr>
<tr>
<td>FPRGS :DELTA-Dynamic Data</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FPRGS :Individual Totals.............</td>
<td>0</td>
<td>0</td>
<td>922</td>
<td>31</td>
</tr>
<tr>
<td>FPRGS :Individual Totals.............</td>
<td>.0000</td>
<td>.0000</td>
<td>.039A</td>
<td>.001F</td>
</tr>
</tbody>
</table>

TOTAL :Memory Available on L6............ | 2048  | .0500 |
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

<table>
<thead>
<tr>
<th>M Identification</th>
<th>Max</th>
<th>Phys</th>
<th>Pro</th>
<th>Data</th>
<th>Dyn</th>
<th>Dyn</th>
<th>Con</th>
<th>Ro</th>
<th>Dbg</th>
<th>Cq</th>
<th>HHJT</th>
</tr>
</thead>
<tbody>
<tr>
<td>M Identification</td>
<td>Sysid</td>
<td>Pgs</td>
<td>Pgs</td>
<td>Pgs</td>
<td>Pgs</td>
<td>I/O</td>
<td>Pgs</td>
<td>Pgs</td>
<td>Pgs</td>
<td>Pgs</td>
<td>Pgs</td>
</tr>
<tr>
<td>H :SYS.NODEADMIN</td>
<td>1 65535</td>
<td>161</td>
<td>68</td>
<td>60</td>
<td>0</td>
<td>11</td>
<td>1</td>
<td>0</td>
<td>33</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>H :SYS.HDLCX25</td>
<td>2 65535</td>
<td>370</td>
<td>167</td>
<td>19</td>
<td>89</td>
<td>48</td>
<td>11</td>
<td>1</td>
<td>0</td>
<td>33</td>
<td>2</td>
</tr>
<tr>
<td>H :SYS,ASYNC</td>
<td>3 65535</td>
<td>371</td>
<td>53</td>
<td>7</td>
<td>9</td>
<td>255</td>
<td>11</td>
<td>1</td>
<td>0</td>
<td>33</td>
<td>2</td>
</tr>
<tr>
<td>G LLAHOST,104AVERY</td>
<td>8</td>
<td>511</td>
<td>15</td>
<td>105</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>11</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G LLAHOST,104AVERY</td>
<td>11</td>
<td>511</td>
<td>16</td>
<td>25</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>11</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>L6XII</td>
<td>-USRT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>FL</td>
<td>BL</td>
<td>Sysid</td>
<td>Mode</td>
<td>State</td>
<td>Flags</td>
<td>Async</td>
<td>DL$</td>
<td>MISC</td>
<td>HMI$</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-----</td>
<td>------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>.01</td>
<td>.00</td>
<td>.03</td>
<td>1</td>
<td>H</td>
<td>SW</td>
<td>.0000</td>
<td>.0000</td>
<td>.0000</td>
<td>.00008E4F</td>
<td>.C1B86</td>
<td></td>
</tr>
<tr>
<td>.02</td>
<td>.03</td>
<td>.00</td>
<td>2</td>
<td>H</td>
<td>SW</td>
<td>.0000</td>
<td>.0000</td>
<td>.0000</td>
<td>.00008A06</td>
<td>.C1B9F</td>
<td></td>
</tr>
<tr>
<td>.03</td>
<td>.01</td>
<td>.02</td>
<td>3</td>
<td>H</td>
<td>SW</td>
<td>.0000</td>
<td>.0000</td>
<td>.0000</td>
<td>.00008E07</td>
<td>.C1BB8</td>
<td></td>
</tr>
<tr>
<td>.04</td>
<td>.00</td>
<td>.00</td>
<td>8</td>
<td>G</td>
<td>SCI</td>
<td>.0800</td>
<td>.0000</td>
<td>.0000</td>
<td>.0000C1542</td>
<td>.00000</td>
<td></td>
</tr>
<tr>
<td>.05</td>
<td>.00</td>
<td>.00</td>
<td>11</td>
<td>G</td>
<td>SCU</td>
<td>.0000</td>
<td>.0000</td>
<td>.0000</td>
<td>.0000C16D2</td>
<td>.00000</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#</th>
<th>Sysid</th>
<th>HJIT</th>
<th>MF</th>
<th>Clock</th>
<th>UTS</th>
<th>Prio</th>
<th>PrioB</th>
<th>USRT$</th>
</tr>
</thead>
<tbody>
<tr>
<td>.01</td>
<td>1</td>
<td>.01C8</td>
<td>0</td>
<td>.0000</td>
<td>780101</td>
<td>0000</td>
<td>1</td>
<td>1.058</td>
</tr>
<tr>
<td>.02</td>
<td>2</td>
<td>.01D3</td>
<td>0</td>
<td>.0000</td>
<td>780101</td>
<td>0000</td>
<td>4</td>
<td>1.070</td>
</tr>
<tr>
<td>.03</td>
<td>3</td>
<td>.01DD</td>
<td>0</td>
<td>.0000</td>
<td>780101</td>
<td>0000</td>
<td>4</td>
<td>1.088</td>
</tr>
<tr>
<td>.04</td>
<td>8</td>
<td>.05B</td>
<td>0</td>
<td>.0000</td>
<td>780101</td>
<td>0000</td>
<td>1</td>
<td>1.0A0</td>
</tr>
<tr>
<td>.05</td>
<td>11</td>
<td>.05A5</td>
<td>0</td>
<td>.0000</td>
<td>780101</td>
<td>0000</td>
<td>1</td>
<td>1.0B8</td>
</tr>
</tbody>
</table>

L6XII   -PPUT
PPUT$: .C0040  MEM_LO: .01C8  MEM_HI: .07FF
PPUT Entry - first page .593 : 6600 0006 last page .598 : A804 0002

L6XII   -EV .593
L6XII   -PL .4000 F CUN
CW=.1AA  MW=.5F0

Frame @.4450 called from .2235 (GID$WRDUMP+.35)
Frame @.4489 called from .22A81 (GIR$INITDUMP+.177)
Frame @.44A7 called from .22EB6 (GIR$SRECH+.142)
Frame @.44C2 called from .91DA (GHT$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$MCL_HAND+.4)
Frame @.45F5 called from .21366 (GHS$ADDUSR+.196)
L6XII   -DUA .4450..4000 F CUN

Frame @.4450 called from .2235 (GID$WRDUMP+.35)
Frame @.4489 called from .22A81 (GIR$INITDUMP+.177)
Frame @.44A7 called from .22EB6 (GIR$SRECH+.142)
Frame @.44C2 called from .91DA (GHT$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$MCL_HAND+.4)
Frame @.45F5 called from .21366 (GHS$ADDUSR+.196)
L6XII   -DUA .4450..4000 F CUN

Frame @.4450 called from .2235 (GID$WRDUMP+.35)
Frame @.4489 called from .22A81 (GIR$INITDUMP+.177)
Frame @.44A7 called from .22EB6 (GIR$SRECH+.142)
Frame @.44C2 called from .91DA (GHT$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$MCL_HAND+.4)
Frame @.45F5 called from .21366 (GHS$ADDUSR+.196)

Frame @.4450 called from .2235 (GID$WRDUMP+.35)
Frame @.4489 called from .22A81 (GIR$INITDUMP+.177)
Frame @.44A7 called from .22EB6 (GIR$SRECH+.142)
Frame @.44C2 called from .91DA (GHT$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$MCL_HAND+.4)
Frame @.45F5 called from .21366 (GHS$ADDUSR+.196)

L6XII   -DUA .4450..4000 F CUN

Frame @.4450 called from .2235 (GID$WRDUMP+.35)
Frame @.4489 called from .22A81 (GIR$INITDUMP+.177)
Frame @.44A7 called from .22EB6 (GIR$SRECH+.142)
Frame @.44C2 called from .91DA (GHT$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$MCL_HAND+.4)
Frame @.45F5 called from .21366 (GHS$ADDUSR+.196)
0044B8 0000 44B4 0000 44B3 0000 44B5 0000 3D6C ..D4..D3..D5..=1
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 001F 0000 f...&...O..\n0044D8 0003 0000 6000 0000 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 059A 000C 0B68 000C 1F70 000C 00CE 0001 ......h..p..N..
004508 000F 0009 0000 0009 003F 0007 003F 0002 ...........?.....'
004510 5BOA 0000 8177 8594 A8FF 000E 0002 01C5 [.....\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 0000 0008 0000 ..y..E...........
004540 0008 0002 0000 10B8 A068 0000 458A 000C.......8h..E...
004548 10B8 0000 8000 0000 0000 A06E 0000 B098 ..8............O.
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 5OB6 E?..........P6
004568 0000 8292 2F9E 0000 0108 0002 0002 69EB .../.....ik
004570 0000 503E 0001 028E 0000 4592 0002 0100 ....P>......E....
004578 000B 0000 0005 0001 028B 0002 0000 0000 .............
004580 0001 0000 0000 0286 0000 0980 0002 000B ...........
004588 0001 0286 0902 0000 50B6 3B43 26A8 0003 ........P6;Cf(.....
004590 0000 A06E 0000 B096 0000 0000 0000 0000 ...........O..
004598 0000 0000 0000 0000 0000 0000 0000 0000 ...........
0045A0* 0000 0000 0000 001 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 0000 4B58 0063 0905 0000 3B5B ......Kx..c...[...
0045E0 1318 0000 0000 0000 0000 0000 0000 50B6 ........P6..
0045E8 0000 0003 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 CO4

DRIBBLE ON @ 17:16 08/21/85
!ANLZ A090
ANLZ C01
:DFO1A090 for GH0-001304-3 at 18:30 AUG 05 '85 on LADC L66B
JL is to blame

Nodes: L6IX L66B
L6IX (node 9) Selected
L6IX -REC

Screech Code:

GH0-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 | .0000 | .094FA | 00000000 | 001A | 4000 | .0000 | .04 | .00 | .8400 | 0000 | 2000 |
| .1B | .A01 | .0000 | .094FA | 000246AA | 0000 | 4004 | .8202 | .04 | .1A | .C000 | 0000 | 2000 |
| .1C | .0000 | .0000 | .094FA | 000247B4 | 0000 | 4004 | .8203 | .04 | .1A | .C000 | 0000 | 2000 |
| .1D | .0000 | .0000 | .094FA | 000248C2 | 0000 | 4004 | .81F6 | .04 | .1A | .C000 | 0000 | 2000 |
| .20 | .0000 | .0000 | .094FA | 00000000 | 0020 | 4000 | .0000 | .03 | .00 | .8400 | 0000 | 2000 |
| .21 | .A02 | .0000 | .094FA | 00020A5C | 0000 | 4004 | .820E | .03 | .20 | .C800 | 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
POC 11 PCQO 83 PCQ 33 PCHHJIT 2 PCDDS 0

PRIV.ACTIVE: .00300C08 AUTH: FFFFFFFFF PRC: .00000000 PPRIV .00000000

MCLs: 58664 Steps: 1 CVM_REAL: .00000010 SPROC: 0 SLIB 2 DB: 0

RUNS .0000 FRS .0000 XLMFLG .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_ALTERRR
  -00000-0
  -00000-0

L6IX  -SPFILE 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
--- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ------
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

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>323A0</td>
<td>323A0</td>
<td>A0250</td>
<td>32B00</td>
<td>33AE0</td>
<td>33A00</td>
<td>A02B7</td>
</tr>
<tr>
<td>R</td>
<td>4</td>
<td>16</td>
<td>FFFF</td>
<td>50</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>M</td>
<td>FF00</td>
<td>FF00</td>
<td>FF00</td>
<td>FF00</td>
<td>FF00</td>
<td>FF00</td>
<td>FF00</td>
</tr>
</tbody>
</table>

ASV$: .5000       TSAP$: .0      NATSAP: 2  I: .001A  T$: .A07FA

CI: .0002    RDBR$: .0
L6IX         -PL .A0000  P .3
CW=.544    MW=.7ED

Frame @ .A02B6 called from .7481B (KV$VDI+.8D7)
Frame @ .A0273 called from .6AC07 (KV$INT+.1915)
Frame @ .A0254 called from .68DD9 (KV$INPCHR+.931)
Frame @ .A0210 called from .745E2 (KV$VDI+.69E)
Frame @ .A01CD called from .20CB6
Frame @ .A01AE called from .20APC
Frame @ .A01B2 called from .20A5C
Frame @ .A0173 called from .6BD53 (KV$RCV+.323)
Frame @ .A001C called from .21B15
Frame @ .A000F called from .21A28
L6IX         -SYM ASYNC.:SYS
L6IX         -PL .A0000  P .3
CW=.544    MW=.7ED

Frame @ .A02B6 called from .7481B (X6A_MAUTO+.14717)
Frame @ .A0273 called from .6AC07 (X6A_MAUTO+.AB03)
Frame @ .A0254 called from .68DD9 (X6A_MAUTO+.8CD5)
Frame @ .A0210 called from .745E2 (X6A_MAUTO+.144DE)
Frame @ .A01CD called from .20CB6 (KAI$INPINT+.10C)
Frame @ .A01AE called from .20APC (KAI$INP+.A0)
Frame @ .A01B2 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)
LCP-6 Trap Numbers

E$TRAP=1300  E$TRAP + G$TSA.I.TRAP# yields the following:

<table>
<thead>
<tr>
<th>Err</th>
<th>TSA</th>
<th>E$ or %G_</th>
<th>:ERRMSG file</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>01</td>
<td>E$MCL</td>
<td>MCL</td>
</tr>
<tr>
<td>02</td>
<td>02</td>
<td>E$TRACE</td>
<td>Trace Breakpoint Trap</td>
</tr>
<tr>
<td>03</td>
<td>03</td>
<td>E$NO_SIP</td>
<td>Uninstalled SIP Trap</td>
</tr>
<tr>
<td>04</td>
<td>04</td>
<td>E$TROT</td>
<td>Watch Dog Timer Runout</td>
</tr>
<tr>
<td>05</td>
<td>05</td>
<td>E$UNIMPL</td>
<td>Unimplemented Instruction Trap</td>
</tr>
<tr>
<td>06</td>
<td>06</td>
<td>E$INT_REG_OV</td>
<td>Integer Register Overflow Trap</td>
</tr>
<tr>
<td>07</td>
<td>07</td>
<td>E$S_DBZ</td>
<td>SIP Divide by Zero Trap</td>
</tr>
<tr>
<td>08</td>
<td>08</td>
<td>E$S_EXP_OV</td>
<td>SIP Exponent Overflow Trap</td>
</tr>
<tr>
<td>09</td>
<td>09</td>
<td>E$STK_UF</td>
<td>Stack Underflow Trap</td>
</tr>
<tr>
<td>10</td>
<td>0A</td>
<td>E$STK_OV</td>
<td>Stack Overflow Trap</td>
</tr>
<tr>
<td>12</td>
<td>0C</td>
<td>E$REMOTE_DESC</td>
<td>Remote Data Descriptor Trap</td>
</tr>
<tr>
<td>13</td>
<td>0D</td>
<td>E$PRIV</td>
<td>Privilege Violation Trap</td>
</tr>
<tr>
<td>14</td>
<td>0E</td>
<td>E$MEM_PROT</td>
<td>Memory Protection Trap</td>
</tr>
<tr>
<td>15</td>
<td>0F</td>
<td>E$INT.UR</td>
<td>Internal Unavailable Resource Trap</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>E$PROG_ERR</td>
<td>Program Error Trap</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>E$INT.MBE</td>
<td>Internal Memory or Bus Error Trap</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>E$S_EXP.UF</td>
<td>SIP Exponent Underflow Trap</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>E$S_PROG.ERR</td>
<td>SIP Program Error Trap</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>E$S_SIGNIF</td>
<td>SIP Significance Error Trap</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>E$S_PRECISION</td>
<td>SIP Precision Error Trap</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>E$EXT.UR</td>
<td>External (CIP or SIP) Unavailable R</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>E$EXT.MBE</td>
<td>External (CIP or SIP) Memory or Bus</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>E$C_DBZ</td>
<td>CIP Divide by Zero Trap</td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>E$C_SPEC</td>
<td>CIP Illegal Specification Trap</td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>E$C_CHAR</td>
<td>CIP Illegal Character Trap</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>E$C_TRUNC</td>
<td>CIP Truncation Trap</td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>E$C_OV</td>
<td>CIP Overflow Trap</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>E$CIP_QLT</td>
<td>CIP QLT Fault</td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>E$SIP_QLT</td>
<td>SIP QLT Fault</td>
</tr>
</tbody>
</table>

1347 - 1362 used to communicate to host debugger.
1347 - 1354 reserved to reflect GJ_LCP6_M FPT code.

<p>| 47  | 2F  | E$FFPRG_EXIT | FPRG M$EXIT.                      |
| 48  | 30  | E$FFPRG_ERR  | FPRG M$ERR.                       |
| 49  | 31  | E$FFPRG.XXX  | FPRG M$XXX.                      |
| 50  | 32  | E$FFPRG_LDTRC | FPRG requesting M$LDTRC.          |
| 51  | 33  | E$FFPRG_XBREAK | FPRG (FPL interpreter) M$XBREAK. |
| 52  | 34  | E$FFPRG_SCRREECH | FPRG M$SCRREECH.                 |
| 55  | 37  | E$HOST_INT  | FPRG interrupted by Host Debugger. |
| 56  | 38  | E$FFPRG_EVENT | FPRG event condition.             |
| 57  | 39  | E$FFPRG_BRK  | FPRG break condition.             |
| 58  | 3A  | E$FFPRG_XCON  | FPRG exit/abort condition.        |
| 59  | 3B  | E$FFPRG_MCLRAP | FPRG bad MCL - Entry to trap handler. |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>3C</td>
<td>E$DB_TIMER</td>
<td>Debugger Timer Runout event</td>
</tr>
<tr>
<td>61</td>
<td>3D</td>
<td>E$FPRG_START</td>
<td>FPRG started via M$SETFP.</td>
</tr>
<tr>
<td>62</td>
<td>3E</td>
<td>E$FPRG_START2</td>
<td>FPRG started via M$LDTRC.</td>
</tr>
<tr>
<td>63</td>
<td>3F</td>
<td>E$MCL2</td>
<td>MCL</td>
</tr>
<tr>
<td>Service Name</td>
<td>Code</td>
<td>Vect#</td>
<td>Service Name</td>
</tr>
<tr>
<td>--------------</td>
<td>------</td>
<td>-------</td>
<td>--------------</td>
</tr>
<tr>
<td>M$INTRET</td>
<td>0440</td>
<td>01</td>
<td>M$AFD</td>
</tr>
<tr>
<td>M$INTCON</td>
<td>0441</td>
<td>01</td>
<td>M$CHGUNIT</td>
</tr>
<tr>
<td>M$INTREL</td>
<td>0442</td>
<td>01</td>
<td>M$CLOCK</td>
</tr>
<tr>
<td>M$SYS</td>
<td>0443</td>
<td>01</td>
<td>M$CLOSE</td>
</tr>
<tr>
<td>M$XBREAK</td>
<td>0484</td>
<td>01</td>
<td>M$CLRSTK</td>
</tr>
<tr>
<td>M$SCREECH</td>
<td>0485</td>
<td>01</td>
<td>M$CPEXIT</td>
</tr>
<tr>
<td>M$MAKEUSR</td>
<td>04C7</td>
<td>01</td>
<td>M$CVM</td>
</tr>
<tr>
<td>M$MERC</td>
<td>04C9</td>
<td>01</td>
<td>M$DBCONTROL</td>
</tr>
<tr>
<td>M$RETRY</td>
<td>04CA</td>
<td>01</td>
<td>M$DCIFLD</td>
</tr>
<tr>
<td>M$CLRSTK</td>
<td>04CD</td>
<td>01</td>
<td>M$DEVICE</td>
</tr>
<tr>
<td>M$MERC5</td>
<td>04CE</td>
<td>01</td>
<td>M$DRTN</td>
</tr>
<tr>
<td>M$RETRY5</td>
<td>04CF</td>
<td>01</td>
<td>M$EOM</td>
</tr>
<tr>
<td>M$AFD</td>
<td>04D1</td>
<td>01</td>
<td>M$ERASE</td>
</tr>
<tr>
<td>M$DRTN</td>
<td>04D2</td>
<td>01</td>
<td>M$ERR</td>
</tr>
<tr>
<td>M$UNSHARE</td>
<td>04D3</td>
<td>01</td>
<td>M$ERRMSG</td>
</tr>
<tr>
<td>M$CHGUNIT</td>
<td>04D5</td>
<td>01</td>
<td>M$EVENT</td>
</tr>
<tr>
<td>M$RUE</td>
<td>04D7</td>
<td>01</td>
<td>M$EXIT</td>
</tr>
<tr>
<td>M$GDDL</td>
<td>0506</td>
<td>01</td>
<td>M$FAUTO</td>
</tr>
<tr>
<td>M$RELCDB</td>
<td>0546</td>
<td>01</td>
<td>M$FDS</td>
</tr>
<tr>
<td>M$TRMPRG</td>
<td>0584</td>
<td>01</td>
<td>M$GAUTO</td>
</tr>
<tr>
<td>M$RCHAN</td>
<td>0594</td>
<td>01</td>
<td>M$GCHAN</td>
</tr>
<tr>
<td>M$EXIT</td>
<td>0880</td>
<td>02</td>
<td>M$GDDL</td>
</tr>
<tr>
<td>M$ERR</td>
<td>0881</td>
<td>02</td>
<td>M$GDS</td>
</tr>
<tr>
<td>M$XXX</td>
<td>0882</td>
<td>02</td>
<td>M$GETDCB</td>
</tr>
<tr>
<td>M$EVENT</td>
<td>08C0</td>
<td>02</td>
<td>M$GLINEATTR</td>
</tr>
<tr>
<td>M$INT</td>
<td>08C1</td>
<td>02</td>
<td>M$GPROMPT</td>
</tr>
<tr>
<td>M$XCON</td>
<td>08C2</td>
<td>02</td>
<td>M$GTRMATTR</td>
</tr>
<tr>
<td>M$TRAP</td>
<td>08C3</td>
<td>02</td>
<td>M$GTRMCTL</td>
</tr>
<tr>
<td>M$DBCONTROL</td>
<td>08C4</td>
<td>02</td>
<td>M$INT</td>
</tr>
<tr>
<td>M$CLOCK</td>
<td>08C6</td>
<td>02</td>
<td>M$INTCON</td>
</tr>
<tr>
<td>M$RTN</td>
<td>08CC</td>
<td>02</td>
<td>M$INTREL</td>
</tr>
<tr>
<td>M$GDS</td>
<td>0902</td>
<td>02</td>
<td>M$INTRET</td>
</tr>
<tr>
<td>M$FDS</td>
<td>0903</td>
<td>02</td>
<td>M$LDTRC</td>
</tr>
<tr>
<td>M$GAUTO</td>
<td>0904</td>
<td>02</td>
<td>M$MAKEUSR</td>
</tr>
<tr>
<td>M$FAUTO</td>
<td>0905</td>
<td>02</td>
<td>M$MDFFLD</td>
</tr>
<tr>
<td>M$PDS</td>
<td>0907</td>
<td>02</td>
<td>M$MERC</td>
</tr>
<tr>
<td>M$STRMCTL</td>
<td>0980</td>
<td>02</td>
<td>M$MERC5</td>
</tr>
<tr>
<td>M$GTRMATTR</td>
<td>0981</td>
<td>02</td>
<td>M$OPEN</td>
</tr>
<tr>
<td>M$PROMPT</td>
<td>0982</td>
<td>02</td>
<td>M$PDS</td>
</tr>
<tr>
<td>M$EOM</td>
<td>0983</td>
<td>02</td>
<td>M$PLATEN</td>
</tr>
<tr>
<td>M$LINEATTR</td>
<td>0985</td>
<td>02</td>
<td>M$PROMPT</td>
</tr>
<tr>
<td>M$STRMATTR</td>
<td>0986</td>
<td>02</td>
<td>M$RCHAN</td>
</tr>
<tr>
<td>M$GTRMATTR</td>
<td>0987</td>
<td>02</td>
<td>M$READ</td>
</tr>
<tr>
<td>M$PLATEN</td>
<td>0988</td>
<td>02</td>
<td>M$RELCDB</td>
</tr>
<tr>
<td>M$STRMTAB</td>
<td>098A</td>
<td>02</td>
<td>M$RETRY</td>
</tr>
<tr>
<td>M$INPUT</td>
<td>098B</td>
<td>02</td>
<td>M$RETRY5</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
<td>Hex Value</td>
<td>Decimal Value</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------</td>
<td>-----------</td>
<td>---------------</td>
</tr>
<tr>
<td>M$RLSFLD</td>
<td>0990 02</td>
<td>M$RLSFLD</td>
<td>0990 02</td>
</tr>
<tr>
<td>M$SLCFLD</td>
<td>0991 02</td>
<td>M$SLCFLD</td>
<td>0991 02</td>
</tr>
<tr>
<td>M$ERASE</td>
<td>0992 02</td>
<td>M$ERASE</td>
<td>0992 02</td>
</tr>
<tr>
<td>M$GCHAN</td>
<td>0993 02</td>
<td>M$GCHAN</td>
<td>0993 02</td>
</tr>
<tr>
<td>M$GPROMPT</td>
<td>0995 02</td>
<td>M$GPROMPT</td>
<td>0995 02</td>
</tr>
<tr>
<td>M$WRSLUG</td>
<td>0997 02</td>
<td>M$WRSLUG</td>
<td>0997 02</td>
</tr>
<tr>
<td>M$WAIT</td>
<td>0C5 03</td>
<td>M$WAIT</td>
<td>0C5 03</td>
</tr>
<tr>
<td>M$SPRIV</td>
<td>0CD0 03</td>
<td>M$SPRIV</td>
<td>0CD0 03</td>
</tr>
<tr>
<td>M$GETDCB</td>
<td>OD45 03</td>
<td>M$GETDCB</td>
<td>OD45 03</td>
</tr>
<tr>
<td>M$DEVICE</td>
<td>OD47 03</td>
<td>M$DEVICE</td>
<td>OD47 03</td>
</tr>
<tr>
<td>M$DCLFLD</td>
<td>OD8F 03</td>
<td>M$DCLFLD</td>
<td>OD8F 03</td>
</tr>
<tr>
<td>M$WRTMLT</td>
<td>1144 04</td>
<td>M$WRTMLT</td>
<td>1144 04</td>
</tr>
<tr>
<td>M$MDFLD</td>
<td>118E 04</td>
<td>M$MDFLD</td>
<td>118E 04</td>
</tr>
<tr>
<td>M$LDTRC</td>
<td>1483 05</td>
<td>M$LDTRC</td>
<td>1483 05</td>
</tr>
<tr>
<td>M$CPEXIT</td>
<td>1486 05</td>
<td>M$CPEXIT</td>
<td>1486 05</td>
</tr>
<tr>
<td>M$TIME</td>
<td>14D4 05</td>
<td>M$TIME</td>
<td>14D4 05</td>
</tr>
<tr>
<td>M$CLOSE</td>
<td>1541 05</td>
<td>M$CLOSE</td>
<td>1541 05</td>
</tr>
<tr>
<td>M$READ</td>
<td>1542 05</td>
<td>M$READ</td>
<td>1542 05</td>
</tr>
<tr>
<td>M$WRITE</td>
<td>1543 05</td>
<td>M$WRITE</td>
<td>1543 05</td>
</tr>
<tr>
<td>M$ERRMSG</td>
<td>2CD6 11</td>
<td>M$ERRMSG</td>
<td>2CD6 11</td>
</tr>
<tr>
<td>M$OPEN</td>
<td>3540 13</td>
<td>M$OPEN</td>
<td>3540 13</td>
</tr>
</tbody>
</table>

Code is expressed in hexadecimal.
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$BREAK | 0084 | M$CLOSE | 0141 |
11.000 M$SCREEX | 0085 | M$READ | 0142 |
12.000 M$CPEXIT | 0086 | M$WRITE | 0143 |
13.000 | M$WRMLT | 0144 |
14.000 M$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 |
<table>
<thead>
<tr>
<th>Page</th>
<th>Symbol</th>
<th>Offset</th>
<th>Description</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>21.000</td>
<td>M$MAKEUSR</td>
<td>OOC7</td>
<td>M$EOM</td>
<td>0183</td>
</tr>
<tr>
<td>22.000</td>
<td>M$SENV</td>
<td>OOC8</td>
<td>M$TRMPRG</td>
<td>0184</td>
</tr>
<tr>
<td>23.000</td>
<td>M$MERC</td>
<td>OOC9</td>
<td>M$GLINEATTR</td>
<td>0185</td>
</tr>
<tr>
<td>24.000</td>
<td>M$RETRY</td>
<td>OOCa</td>
<td>M$STRMATTR</td>
<td>0186</td>
</tr>
<tr>
<td>25.000</td>
<td>M$RENV</td>
<td>OOCb</td>
<td>M$GTRMATTR</td>
<td>0187</td>
</tr>
<tr>
<td>26.000</td>
<td>M$TRTN</td>
<td>OOCc</td>
<td>M$PLATEN</td>
<td>0188</td>
</tr>
<tr>
<td>27.000</td>
<td>M$CLRSTK</td>
<td>OOCd</td>
<td>M$STRMTAB</td>
<td>018a</td>
</tr>
<tr>
<td>28.000</td>
<td>M$MERCs</td>
<td>OOCe</td>
<td>M$INPUT</td>
<td>018b</td>
</tr>
<tr>
<td>29.000</td>
<td>M$RETRYs</td>
<td>OOCf</td>
<td>M$MDFFLD</td>
<td>018e</td>
</tr>
<tr>
<td>30.000</td>
<td>M$SPRIV</td>
<td>OOD0</td>
<td>M$DCLFLD</td>
<td>018f</td>
</tr>
<tr>
<td>31.000</td>
<td>M$APD</td>
<td>OOD1</td>
<td>M$RLSFLD</td>
<td>0190</td>
</tr>
<tr>
<td>32.000</td>
<td>M$DRTN</td>
<td>OOD2</td>
<td>M$SLCFLD</td>
<td>0191</td>
</tr>
<tr>
<td>33.000</td>
<td>M$UNSHARE</td>
<td>OOD3</td>
<td>M$ERASE</td>
<td>0192</td>
</tr>
<tr>
<td>34.000</td>
<td>M$TIME</td>
<td>OOD4</td>
<td>M$GCHAN</td>
<td>0193</td>
</tr>
<tr>
<td>35.000</td>
<td>M$CHGUNIT</td>
<td>OOD5</td>
<td>M$RCHAN</td>
<td>0194</td>
</tr>
<tr>
<td>36.000</td>
<td>M$ERRMSG</td>
<td>OOD6</td>
<td>M$GPROMPT</td>
<td>0195</td>
</tr>
<tr>
<td>37.000</td>
<td>M$RUE</td>
<td>OOD7</td>
<td>M$WRSYSLOG</td>
<td>01c0</td>
</tr>
<tr>
<td>Active Process</td>
<td>USER/HANDLER</td>
<td>MON FOR USER</td>
<td>MONITOR</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>--------------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UHJIT.ASDT_USR</td>
<td>UHJIT.ASDT_MCL</td>
<td>MHJIT.ASDT_MON</td>
<td></td>
</tr>
<tr>
<td># UASDT MASDT VADDR</td>
<td>----------------</td>
<td>----------------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>00 .5000 .503E .00000</td>
<td>NULLSEG</td>
<td>NULLSEG</td>
<td>NULLSEG</td>
<td></td>
</tr>
<tr>
<td>01 .5002 .5040 .01000</td>
<td>ROS</td>
<td>ROS</td>
<td>ROS</td>
<td></td>
</tr>
<tr>
<td>02 .5004 .5042 .02000</td>
<td>DB_DS</td>
<td>RDB_DS</td>
<td>RDB_DS</td>
<td></td>
</tr>
<tr>
<td>03 .5006 .5044 .03000</td>
<td>LOW_MEM</td>
<td>LOW_MEM</td>
<td>LOW_MEM</td>
<td></td>
</tr>
<tr>
<td>04 .5008 .5046 .04000</td>
<td>TSTACKU</td>
<td>TSTACKU</td>
<td>TSTACKM</td>
<td></td>
</tr>
<tr>
<td>05 .500A .5048 .05000</td>
<td>UHJIT</td>
<td>UHJIT</td>
<td>UHJIT</td>
<td></td>
</tr>
<tr>
<td>06 .500C .504A .06000</td>
<td>MHJIT</td>
<td>MHJIT</td>
<td>MHJIT</td>
<td></td>
</tr>
<tr>
<td>07 .500E .504C .07000</td>
<td>MHJIT</td>
<td>MHJIT</td>
<td>MHJIT</td>
<td></td>
</tr>
<tr>
<td>08 .5010 .504E .08000</td>
<td>MON_ENTRY_DATA</td>
<td>MON_ENTRY_DATA</td>
<td>MON_ENTRY_DATA</td>
<td></td>
</tr>
<tr>
<td>09 .5012 .5050 .09000</td>
<td>MON_ENTRY</td>
<td>MON_ENTRY</td>
<td>MON_ENTRY</td>
<td></td>
</tr>
<tr>
<td>10 .5014 .5052 .0A000</td>
<td>LPAR1</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 .5016 .5054 .0B000</td>
<td>LPAR2</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 .5018 .5056 .0C000</td>
<td>LPAR3</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13 .501A .5058 .0D000</td>
<td>LPAR4</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 .501C .505A .0E000</td>
<td>LPAR5</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 .501E .505C .0F000</td>
<td>LPAR6</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16 .5020 .505E .10000</td>
<td>USER_IS1</td>
<td>MON_IS1</td>
<td>MON_IS1</td>
<td></td>
</tr>
<tr>
<td>17 .5022 .5060 .20000</td>
<td>USER_IS2</td>
<td>MON_IS2</td>
<td>MON_IS2</td>
<td></td>
</tr>
<tr>
<td>18 .5024 .5062 .30000</td>
<td>USER_IS3</td>
<td>MON_IS3</td>
<td>MON_IS3</td>
<td></td>
</tr>
<tr>
<td>19 .5026 .5064 .40000</td>
<td>USER_IS4</td>
<td>MON_IS4</td>
<td>MON_IS4</td>
<td></td>
</tr>
<tr>
<td>20 .5028 .5066 .50000</td>
<td>USER_IS5</td>
<td>MON_IS5</td>
<td>MON_IS5</td>
<td></td>
</tr>
<tr>
<td>21 .502A .5068 .60000</td>
<td>USER_IS6 (LIB)</td>
<td>BPAR1</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>22 .502C .506A .70000</td>
<td>USER_IS7 (LIB)</td>
<td>BPAR2</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>23 .5030 .506C .80000</td>
<td>DB_PROC</td>
<td>DB_PROC</td>
<td>DB_PROC</td>
<td></td>
</tr>
<tr>
<td>24 .5032 .5070 .90000</td>
<td>CP_PROC</td>
<td>WINDOW1</td>
<td>WINDOW1</td>
<td></td>
</tr>
<tr>
<td>25 .5034 .5072 .A0000</td>
<td>UAUTO_DS</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>26 .5036 .5074 .B0000</td>
<td>USER_DS3</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>27 .5038 .5076 .C0000</td>
<td>USER_DS4</td>
<td>BOBCAT</td>
<td>BOBCAT</td>
<td></td>
</tr>
<tr>
<td>28 .503A .5078 .D0000</td>
<td>HAND_Q</td>
<td>HAND_Q</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>29 .503C .507A .E0000</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30 .503E .507C .F0000</td>
<td>*</td>
<td>*</td>
<td>BIGFOOT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BIGFOOT</td>
<td></td>
</tr>
</tbody>
</table>
The User Housekeeping Job Information Table (UHJIT) contains the data required by the hardware and the LCP-6 Scheduler 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

Contains the TSA overflow
I logio used to generate the
! GHT-1396-3 Screech Dumps
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
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 NATSAP7$ 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.
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. Their 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**

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hex</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>---</td>
<td>---</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
<td>Power Fail Safe</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>Watch Dog Timer</td>
</tr>
<tr>
<td>02</td>
<td>02</td>
<td>Trap Save Overflow</td>
</tr>
<tr>
<td>03</td>
<td>03</td>
<td>Bigfoot - Used during the boot process</td>
</tr>
<tr>
<td>04</td>
<td>04</td>
<td>Monitor Inhibit Level</td>
</tr>
<tr>
<td>05</td>
<td>05</td>
<td>Monitor Inhibit Level2</td>
</tr>
<tr>
<td>06</td>
<td>06</td>
<td>XDELTA</td>
</tr>
<tr>
<td>0A</td>
<td>0A</td>
<td>XDELTA</td>
</tr>
<tr>
<td>12</td>
<td>0C</td>
<td>Available for Handlers</td>
</tr>
<tr>
<td>59</td>
<td>3B</td>
<td>Available for Handlers</td>
</tr>
<tr>
<td>60</td>
<td>3C</td>
<td>Common Interrupt Cleanup</td>
</tr>
<tr>
<td>61</td>
<td>3D</td>
<td>Real Time Clock</td>
</tr>
<tr>
<td>62</td>
<td>3E</td>
<td>Scheduler - Idle Level</td>
</tr>
<tr>
<td>63</td>
<td>3F</td>
<td>User Execution</td>
</tr>
</tbody>
</table>

**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 = 'O'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_RINGO# | 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 CO1JT
: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

Soreeh 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$TRAP+.8F

ISA:


Interrupted address: GHT$TRAP+.25

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>B Registers:</td>
<td>503E</td>
<td>50B6</td>
<td>5081</td>
<td>2077A</td>
<td>5000</td>
<td>20707</td>
<td>44F8</td>
</tr>
<tr>
<td>R Registers:</td>
<td>10</td>
<td>F</td>
<td>A103</td>
<td>2</td>
<td>32</td>
<td>2</td>
<td>996</td>
</tr>
<tr>
<td>M Registers:</td>
<td>FF00</td>
<td>FF00</td>
<td>FF00</td>
<td>FF00</td>
<td>FF00</td>
<td>FF00</td>
<td>FF00</td>
</tr>
</tbody>
</table>

ASV$: .503E TSAP$: .0 NATSAP: 2 I: .000C T$: .45FA

CI: .0000 RDBR$: .0

TSA @.50B6:

TRAP: 15 TSAL$: .0 I: .OF18 INST: .D800 Z: .0083

Trap IC: GHM$MCLS+.116

ISA:

P: .9026 S: .0000 CHN: .FFFF ISM1: .FFFF ISM2: .A103
Interrupted address: GHT$TRAP+.25

<table>
<thead>
<tr>
<th>B Registers:</th>
<th>A005C</th>
<th>A0068</th>
<th>5081</th>
<th>20C8D</th>
<th>D001B</th>
<th>20878</th>
<th>A0058</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Registers:</td>
<td>10</td>
<td>8</td>
<td>A103</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1D</td>
</tr>
<tr>
<td>M Registers:</td>
<td>FF00</td>
<td>FF00</td>
<td>FF00</td>
<td>FF00</td>
<td>FF00</td>
<td>FF00</td>
<td>FFO0</td>
</tr>
</tbody>
</table>

ASV$: .5000  TSAP$: .0  NATSAP: 2 I$: .0018  T$: .A09FA

CI: .0000  RDR$: .0

<table>
<thead>
<tr>
<th>L6I -INTCON</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
</tr>
<tr>
<td>.1A</td>
</tr>
<tr>
<td>.1B</td>
</tr>
<tr>
<td>.1C</td>
</tr>
<tr>
<td>.1D</td>
</tr>
<tr>
<td>.1E</td>
</tr>
<tr>
<td>.20</td>
</tr>
<tr>
<td>.21</td>
</tr>
<tr>
<td>.22</td>
</tr>
<tr>
<td>.24</td>
</tr>
<tr>
<td>.25</td>
</tr>
<tr>
<td>.26</td>
</tr>
<tr>
<td>.32</td>
</tr>
<tr>
<td>.33</td>
</tr>
</tbody>
</table>

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
L6I  -DU  .507E1  ASDT  .20400
     0 1 2 3 4 5 6 7 0003
L6I  -SYM COUPLER.:SYS
L6I  -TSA  .50B6 F  .2
TRAP: 63  TSAL$:  .0  I:  .3FOC  INST:  .0001  Z:  .8080
A$:  .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  9C56  61E7C  A0025
R Registers:  B2  3F  0  FFFF  3F  8  9103
M Registers:  FFOO  FFOO  FFO0  FFOO  FFOO  FFO0  FFOA

ASV$:  .5000  TSAP$:  .0  NATSAP:  1  I:  .000C  T$:  .A09FA

CI:  .0004  RDBR$:  .0
L6I  -TSA  .50B6 ASDT  .20400
TRAP: 15  TSAL$:  .0  I:  .OF18  INST:  .D800  Z:  .0083
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  20878  A0058
R Registers:  10  8  A103  2  1  3  1D
M Registers:  FFO0  FFO0  FFO0  FFO0  FFO0  FFO0  FFOA

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)