# MEMORY DUMP PROCEDURE

## TABLE OF CONTENTS


1. MEMORY DUMP PROCEDURES . . . . . 10-1- 1

2. B 6700 MEMORY DUMP PROCEDURE . . . . . 10-2- 1

INTRODUCTION . . . . . 10-2- 1

MAINTENANCE PANEL INFORMATION . . . . . 10-2- 1

Maintenance Panels . . . . . 10-2- 1

Setting Processor Flip-Flops . . . . . 10-2- 2

Resetting Processor Flip-Flops . . . . . 10-2- 2

Data Processor Display Panel (Panel B) . . . . . 10-2- 2

MEMORY TESTER INFORMATION . . . . . 10-2- 5

Setting Memory Tester Display Indicators . . . . . 10-2- 5

Resetting Memory Tester Display Indicators . . . . . 10-2- 5

MEMORY TESTER DUMP PROCEDURE (B 6700 ONLY) . . . . . 10-2- 7

P-REGISTER DUMP PROCEDURE (B 6700 and B 6748) . . . . . 10-2- 9

3. B 6800 MEMORY DUMP PROCEDURE . . . . . 10-3- 1

INTRODUCTION . . . . . 10-3- 1

MAINTENANCE DISPLAY PROCESSOR . . . . . 10-3- 1

Programmer's Display Panel . . . . . 10-3- 1

System Control Panel . . . . . 10-3- 1

Status Display Panel . . . . . 10-3- 1

Register Display Panel . . . . . 10-3- 1

Maintenance Display Panel . . . . . 10-3- 4

Maintenance Display Registers . . . . . 10-3- 4

Maintenance Processor Control Panel and Display . . . . . 10-3- 4

Programmer's Display Keyboard . . . . . 10-3- 4

FLIP-FLOP RESET PROCEDURE . . . . . 10-3- 7

MEMORY DUMPING PROCEDURE . . . . . 10-3- 7

4. B7000 MEMORY DUMP PROCEDURE . . . . . 10-4- 1

INTRODUCTION . . . . . 10-4- 1

MEMORY DUMP PROCEDURE . . . . . 10-4- 1

PURGE PROCEDURES . . . . . 10-4- 2

RECORDING PROCEDURES . . . . . 10-4- 3

# 1. MEMORY DUMP PROCEDURES

This section describes manual procedures for forcing memory dumps on B 6700, B 6748, and B 6800 systems. The normal method of initiating a memory dump is through ODT commands. Under some types of hardware or software malfunctions, this procedure may not be effective. In those circumstances, the directions provided in this section should be followed.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# 2. B 6700 MEMORY DUMP PROCEDURE

## INTRODUCTION

On the B 6700 system, two methods are available for forcing a memory dump to tape for analysis by the SYSTEM/DUMPANALYZER program: the memory tester method and the P-register method. (On the B 6748 system, the P-register method is the only available method.) This document describes both methods.

## MAINTENANCE PANEL INFORMATION

The memory dump procedures described by this document utilize various sections of the data processor display panel. This section identifies those parts of the display panel that are used for the memory dump procedures. This information is provided for ease in locating the switches, push buttons, and display flip-flops that are used.

### Maintenance Panels

The maintenance panels referenced in this document are illustrated in Figure 10-1-1.

The maintenance panels are identified below:

| | |
|---|---|
| Panel A | Panel A displays the stack registers (P, C, A, B, X, and Y). |
| Panel B | Panel B contains the display flip-flops for the data processor (identified by the blue lettering on the panel) and the input/output processor (identified by the orange lettering on the panel). |
| Panel C | General controls panel - Panel C contains the control switches and push buttons that control the system clock and the setting and resetting of flip-flops. |
| Panel D | Data processor maintenance controls - Panel D provides the control switches and push buttons that examine and control the data processor registers. |
| Panel E | Input/output processor maintenance controls - Panel E provides the maintenance switches for the input/output processor. |

**Setting Processor Flip-Flops**

A processor flip-flop is set using the following procedure:

1. The FF SET button at the lower right corner of the general controls panel (panel C) is pressed and held.
2. The desired push button indicator is pressed, and the selected flip-flop is set.

**Resetting Processor Flip-Flops**

A processor flip-flop is reset using the following procedure:

1. The FF RESET button at the lower right corner of the general controls panel (panel C) is pressed and held.
2. The desired push button indicator is pressed, and the selected flip-flop is reset.

**Data Processor Display Panel (Panel B)**

Figure 10-1-2 illustrates the data processor display panel. The shaded indicators identify the flip-flops used for the memory dump procedures. These indicators are provided for ease in locating the necessary flip-flops.

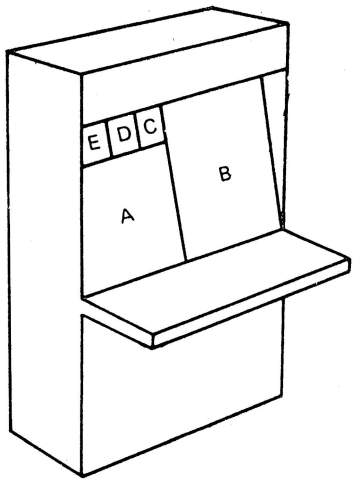



Figure 10-1-1. B 6700 Maintenance Panels

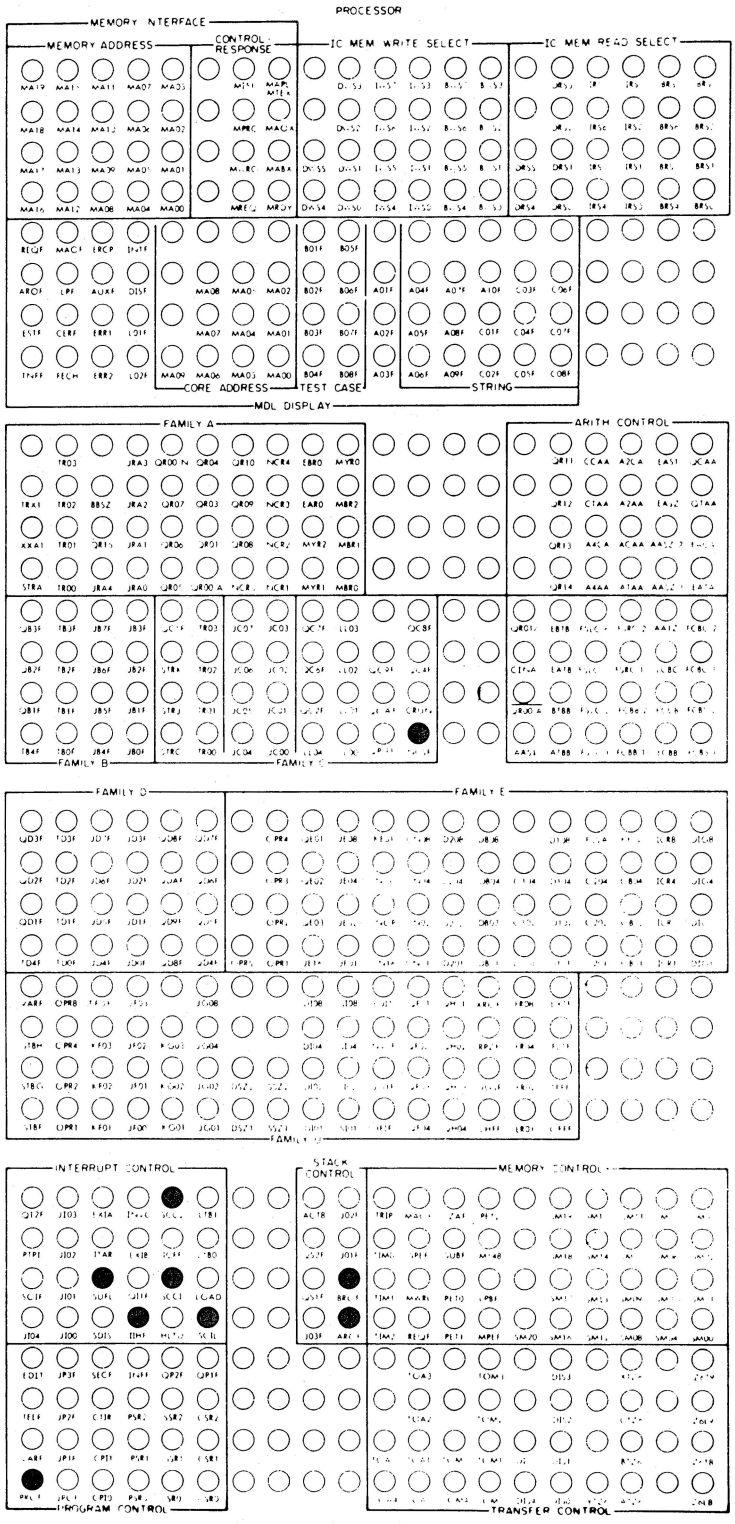



Figure 10-1-2. B 6700 Processor Display Panel B

## MEMORY TESTER INFORMATION

If the DP message entered at the input keyboard is unsuccessful, the memory tester can be used to force a memory dump (B 6700 only).

### Setting Memory Tester Display Indicators

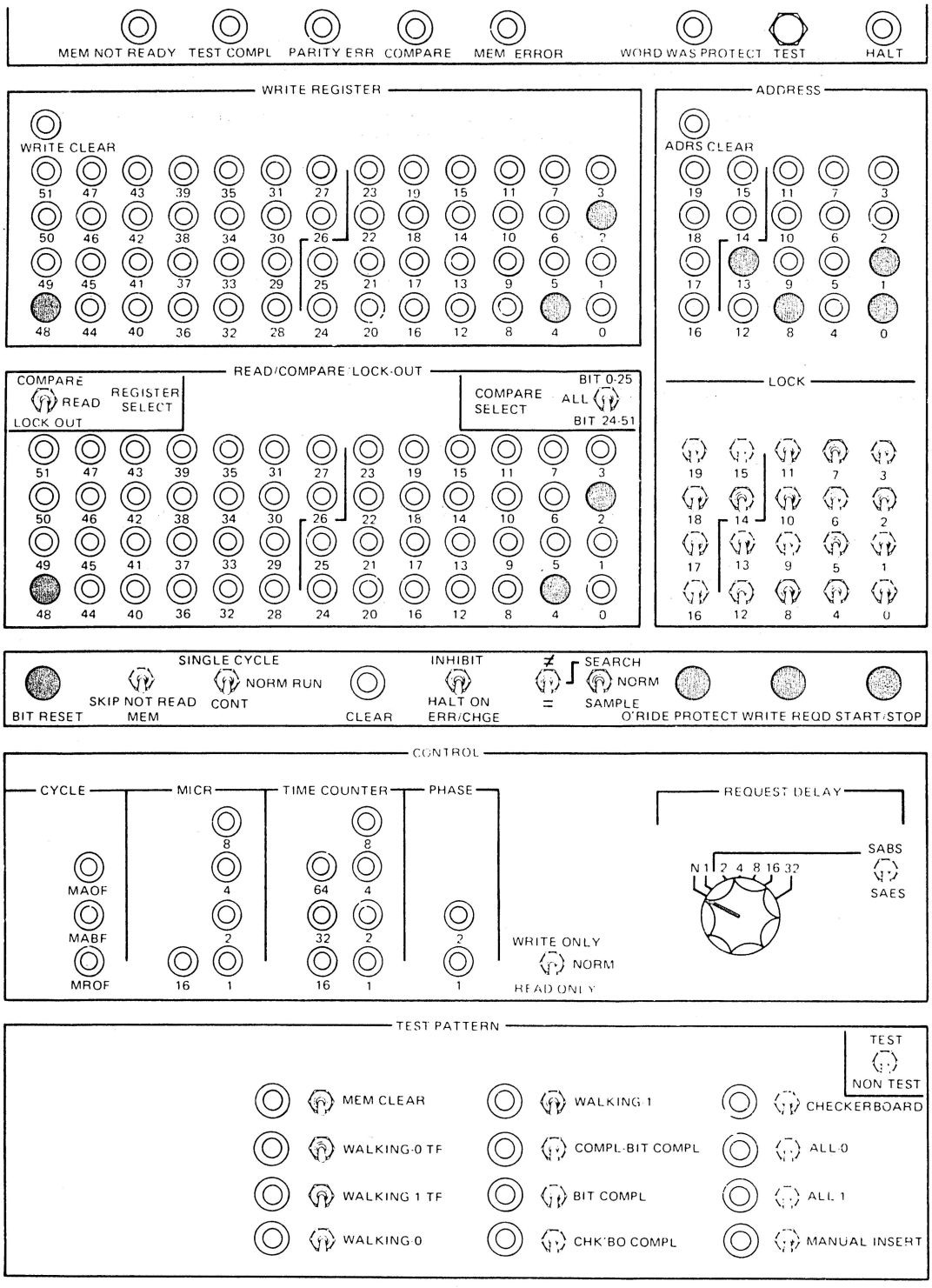
A display indicator on the memory tester panel (Figure 10-1-3) is set by pushing the desired display indicator; the selected bit is then set.

### Resetting Memory Tester Display Indicators

A display indicator on the memory tester panel (Figure 10-1-3), is reset using the following procedure:

1. The BIT RESET button (on the left side of the panel) is pressed and held.
2. Concurrently with step 1, the desired indicator is pushed, and the selected bit is reset.

## MEMORY DUMP PROCEDURE




MV4133

Figure 10-1-3. B 6700 Memory Tester Panel

## MEMORY TESTER DUMP PROCEDURE (B 6700 ONLY)

The following procedure for memory tester dump is performed on the memory tester panel:

1. The SABS/SAES switch (on the right side of the panel) is placed in the SABS (up) position.

2. The hexadecimal value (10000000000014) is placed in the write register. In Figure 10-1-3, the shaded bits indicate the bits that must be set in the write register.

3. The hexadecimal value (02103) is placed in the address register. In Figure 10-1-3, the shaded bits indicate the bits that must be set in the address register.

4. The WRITE REQ button indicator is pressed and released.

5. The O'RIDE PROTECT button is held in, and the START/STOP button is pressed.

6. The START/STOP button is pressed. If step 5 is successful, the read/compare/lockout register displays the hexadecimal code (1000000000014).

These actions cause an IRW (indirect reference word) to the MEMDUMP PCW to be written in address D[0] + 3; all processors go to this address to answer any interrupt. When the IRW has been written in D[0] + 3, the next processor that responds to any interrupt enters the MEMDUMP procedure and begins the dump cycle.

However, if the processor or processors do not answer external or internal interrupts, and no memory dump occurs, then an alarm interrupt must be forced in a processor. (This type of interrupt also causes entry into an interrupt-handling procedure by way of location D[0]+3.) The procedure in steps 2 thru 23 is followed to force a stack-underflow interrupt in a processor and an attempt at another memory dump.

7. The SECL switch on the data processor maintenance panel (panel D) is placed in the up position.

8. The SYST clock in the clock controls section of the general controls panel (panel C) is placed in the up position.

9. The SUFL (Stack-Underflow Interrupt) flip-flop in the interrupt control section of the processor display panel (panel C) is set, and all other flip-flops in the interrupt control are reset. (Refer to the flip-flop setting and resetting procedures listed at the beginning of this chapter.)

10. The SYST clock control is placed in the down position.

11. The SECL switch is placed in the down position.

MEMORY DUMP PROCEDURE

12. The START push button on the processor maintenance control panel is pressed and released. If no dump occurs, the procedure in steps 13 thru 23 below is followed.

13. The SECL switch is placed in the up position.

14. The LOCAL/REMOTE switch on panel D is placed in the LOCAL position.

15. The ADJ 0, 0 button on panel D is pressed and released.

16. The UNIT CLEAR button on panel D is pressed and released.

17. A hexadecimal 2100 is entered in the two least significant characters of the A register. (Refer to Figure 10-1-4.)

18. AROF and BROF in the stack control section of panel B are set. (Refer to Figure 10-1-2.)

19. The WRITE-IC button on panel D is pressed and released.

20. The following syllables are entered in the P register of the processor: 3 AE 40 14 B0 B0 AB (TAG=3; MKST; NAMC MEMDUMP; LIT:0; LIT:0; ENTR). (Refer to Figure 10-1-5.)

21. On panel B, PROF (in the program control section), IIHF (in the interrupt control section), and NCSF (in family C) are set. (Refer to Figure 10-1-2)

22. The SECL and the LOCAL/REMOTE switches are placed in the down position.

23. The START button is pressed and released.

## P-REGISTER DUMP PROCEDURE (B 6700 and B 6748)

The following steps comprise the P-register dump procedure:

**NOTE**

Because a physical continous operations system may be split into two logical systems, SYSTEM in this procedure is defined to be the logical system that does the dump. All steps (unless otherwise stated) apply to the SYSTEM that does the dump.

1. The DP input message is entered at the input keyboard. If a memory dump is not produced, step 2 below follows.

2. The system is halted by pressing the ENABLE and HALT buttons on the console or by pressing the HALT button on the general controls panel (panel C).

3. The ODTs are cleared by changing from local to receive mode a few times or by holding the SHIFT key down and pressing the HOME key. The ODT is then put in receive mode.

4. Dynamic reconfiguration is disallowed (dynamic reconfiguration systems only) using the following method:

   a. All teams are halted. (This step should be verified.)

   b. The SYSTEM MODE switch on the RCC is placed in the STEP position.

   c. The nondumping team may be restarted by pressing the START buttons for all processors on that team.

   d. Steps 5 thru 8 apply to the dumping team.

5. Multiplexor registers are recorded and all multiplexors are unit-cleared by pressing the UNIT CLEAR button (panel E only). All multiplexors remain in remote (LOCAL/REMOTE switch in the down position).

6. On all processors, the following steps are performed:

   a. The LOCAL/REMOTE switch is placed on LOCAL (in the up position on panel D).

   b. The SECL switch is placed in the up position.

   c. The ADJ 0, 0 button is pressed.

   d. Any desired register settings are recorded at this time.

   e. The UNIT CLEAR button is pressed.

f. D[0] is verified as equal to hexadecimal 2100 using the following method:

(1) BROF in the stack controller section of panel B is set. (Refer to Figure 10-1-4.)

(2) The READ-IC button on panel D is pressed and released.

(3) If the A register equals hex 2100, step 10 below follows; otherwise, step 4 below follows.

(4) The UNIT CLEAR button on panel D is pressed and released.

(5) The A register is set equal to hex 2100. (Refer to Figure 10-1-4.)

(6) AROF and BROF in the stack controller section of panel B are set. (Refer to Figure 10-1-4.)

(7) The WRITE-IC button is pressed.

(8) The UNIT CLEAR button is pressed.

(9) The verification procedure returns to step 6f(1).

(10) The UNIT CLEAR button is pressed. All nondumping processors remain in local mode (with LOCAL/REMOTE switches up on panel D).

7. PROF in the program control section of panel B is set on nondumping processors.

NOTE

When PROF is set, stacks running on these processors are marked with an invalid program-word interrupt.

8. The following steps are performed on the processor that does the dump:

a. The P register is set equal to 3 AE 40 14 B1 CE AB (TAG=3; MKST; NAMC MEMDUMP; LIT 1; XTND; ENTR). (Refer to Figure 10-1-6.)

b. PROF (program control), IIHF (interrupt control), and NCSF (family C) are set.

c. The LOCAL/REMOTE switch is placed on REMOTE (in the down position).

d. SCIL (interrupt control) is set. (SCCI and SCC2 in the interrupt control should be ensured to be cycling.

e. The SECL switch on panel D is placed in the down position

f. The START button on panel D is pressed.

B 6700 A Register

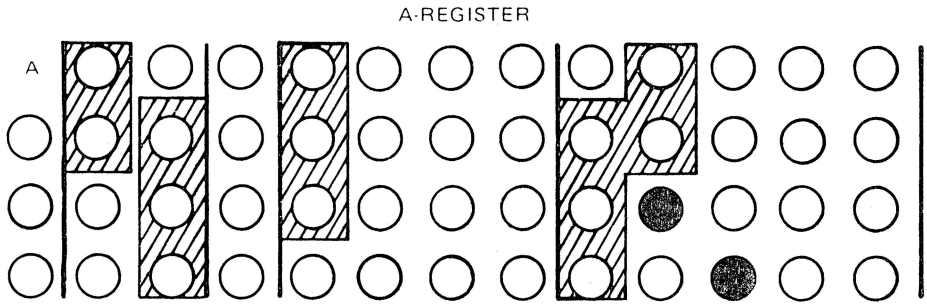



Figure 10-1-4. Hexadecimal 2100
(Step 7e Memory Tester Dump Method and Step 6f(3)
P-Register Dump Method)

B 6700 P Register

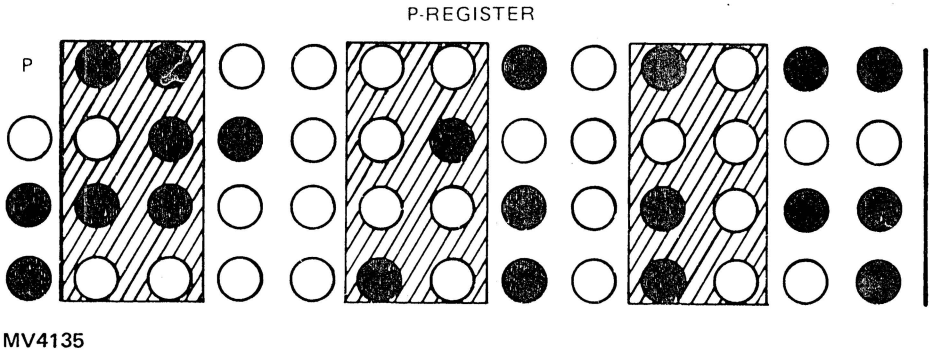



Figure 10-1-5. Hexadecimal 3 AE4014B0B0AB
(Step 20 Memory Tester Dump Method)

B 6700 P Register

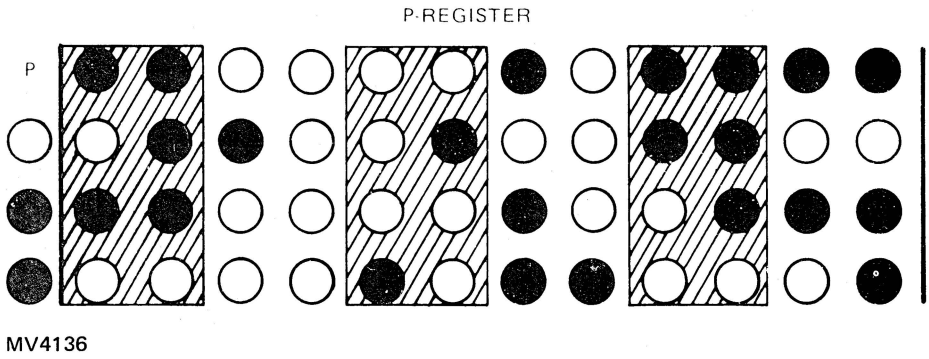



Figure 10-1-6. Hexadecimal 3AE4014B1CEAB
(Step 8a P-Register Dump Method)

# 3. B 6800 MEMORY DUMP PROCEDURE

## INTRODUCTION

This section describes a method for forcing a B 6800 memory dump to tape. The tape can be analyzed with the SYSTEM/DUMPANALYZER.

## MAINTENANCE DISPLAY PROCESSOR

The memory dump procedure uses various parts of the maintenance display processor (MDP). (See Figure 10-2-1.) A brief description of the relevant parts is included in this section. For a more detailed description of the MDP, see the B 6800 System Reference Manual, form 5001290.

The MDP contains the programmer's display panel, the maintenance display panel, and the programmer's display keyboard.

### Programmer's Display Panel

The programmer's display panel contains the system control panel, the status display panel, and the register display panel. (See Figure 10-2-2.)

#### System Control Panel

The eight system control indicators are on the top row of indicators on the programmer's display panel. (See Figure 10-2-2.) The HALT indicator causes the system to halt at the end of the current instruction. The LOAD switch causes a general clear when pressed and causes a LOAD signal to be sent to the mainframe when released. The GENERAL CLEAR switch causes the flip-flops of the system to be reset.

#### Status Display Panel

The status display panel (see Figure 10-2-2) consists of a series of 50 individual indicator lamps. Each indicator displays the state of a major control flip-flop in a B 6800 system. The logical state of the system circuit is true when the indicator is illuminated and is false when the indicator is not illuminated.

#### Register Display Panel

Four display registers are present on the register display panel. Each register has eight push button indicators located below the register. The lit push button indicates the selected CPU register. The contents of the selected register are displayed and may be altered.

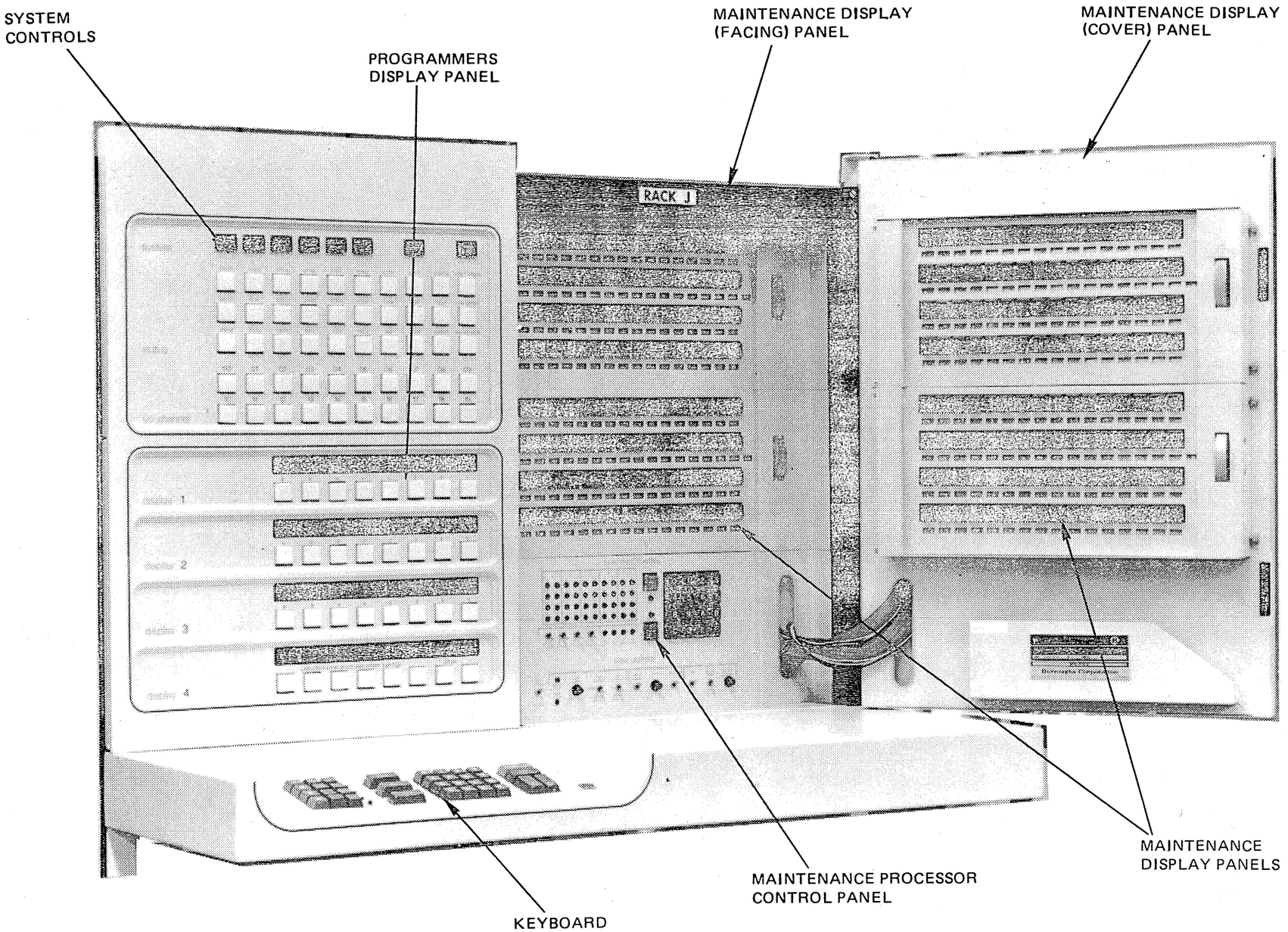



Figure 10-2-1. Maintenance Display Processor

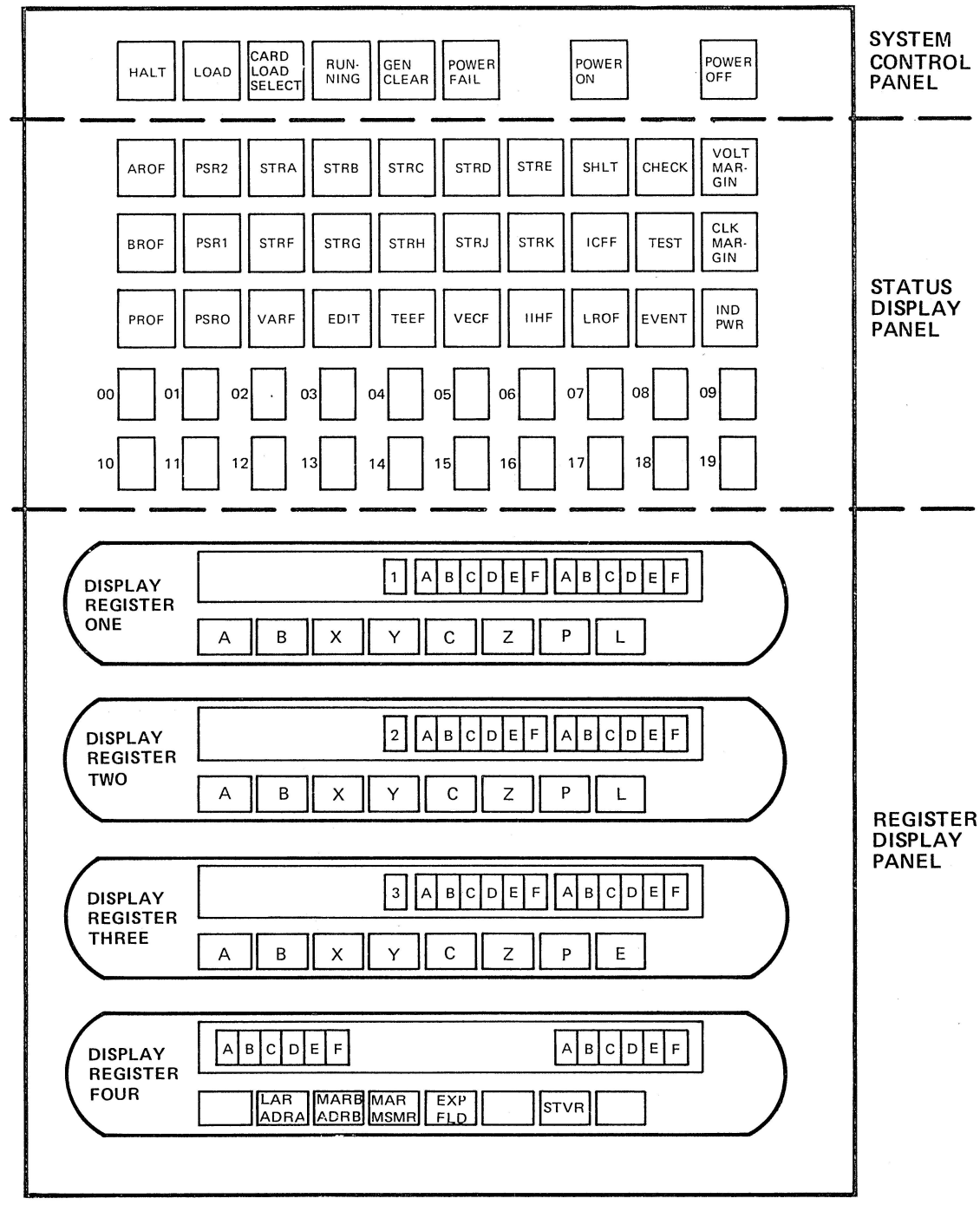



Figure 10-2-2. Programmer's Display Panel

## Maintenance Display Panel

The maintenance display panel (see Figure 10-2-1) consists of the MP control panel and display and the maintenance display registers. The maintenance display panel is covered by a hinged panel and during normal system operations is not visible. When the hinged panel is opened, maintenance display registers 1 and 2 and the MP control panel and display are visible on the facing panel. (See Figure 10-2-3.) Maintenance display registers 3 and 4 are mounted on the back of the hinged panel.

### Maintenance Display Registers

A maintenance display register has 64 indicator lamps organized into four bars of 16 indicators. The thumbwheel selector switch, located to the right of the register, selects one of six pages of 64 signals on a panel of the CPU cabinet to be displayed in a register. The selector switch display window indicates which page and panel are presently selected for display in the register.

### Maintenance Processor Control Panel and Display

The maintenance display processor control panel is located below maintenance display register 2. (See Figure 10-2-1.) The EVENT switch is located in the MAINT CONTROL panel in the maintenance switch group. The EVENT switch has two positions: enabled (up) and disabled (down).

## Programmer's Display Keyboard

The MDP programmer's keyboard is located immediately beneath the programmmer's display panel. (See Figure 10-2-1.) The push button switches and indicators are shown in Figure 10-2-4. The functions of the switches and indicators are as follows:

Register select switches 1, 2, 3, 4.

When one of the four switches is pressed, the associated register is selected. The cursor blinks at the leftmost digit of that register, and the associated register select indicator is on.

HEX DIGITS 0 thru F switches

When a HEX DIGIT switch is pressed, the four flip-flops indicated by the cursor take the state of that hex character. When the switch is released, the cursor moves one position to the right.

CURSOR LEFT switch - left arrow

When the CURSOR LEFT switch is pressed, the cursor moves one position to the left in the selected display register.

CURSOR RIGHT switch - right arrow

When the CURSOR RIGHT switch is pressed, the cursor moves one position to the right in the selected display register.

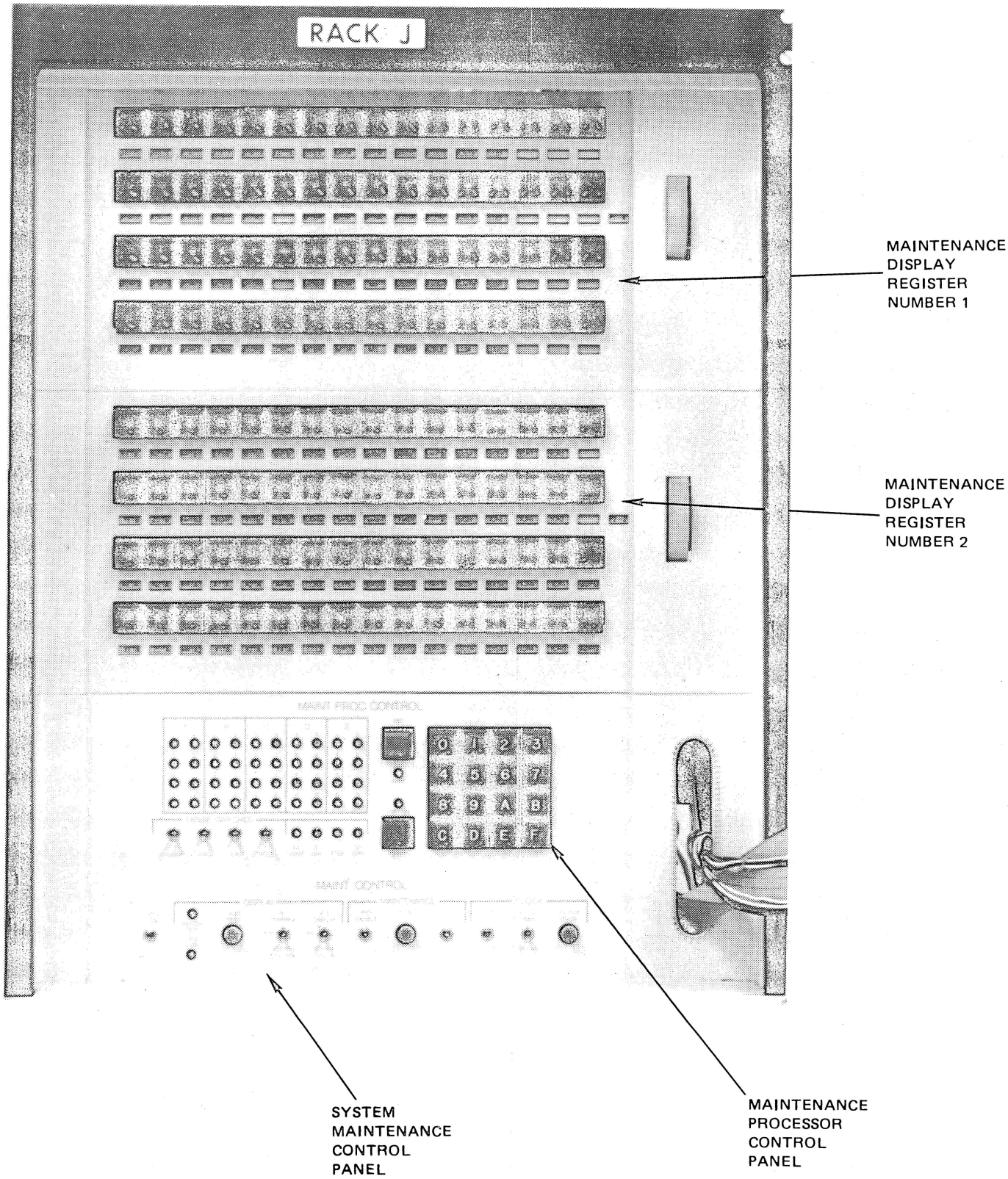



Figure 10-2-3. Maintenance Display Panel (Facing)

REG CLR switch

When the REG CLR switch is pressed, the selected register is cleared (reset).

STEP switch

The STEP switch works in conjunction with the EVENT switch. If the EVENT switch is off and the STEP switch is pressed, the CPU begins execution of the next instruction when the CPU has been stopped by the HALT or SECL switch.

SECL switch

When the SECL switch is off, it has no effect on the mainframe. When the SECL switch is on, the CPU finishes the current instruction and is inhibited from starting the next instruction.

ADJ0 switch

When the ADJO switch is pressed, memory-write cycles necessary to push top of stack into memory are initiated.

BIT RESET button

The BIT RESET button must be pressed to ensure that a flip-flop is left in the reset state when the associated switch of the flip-flop is pressed.

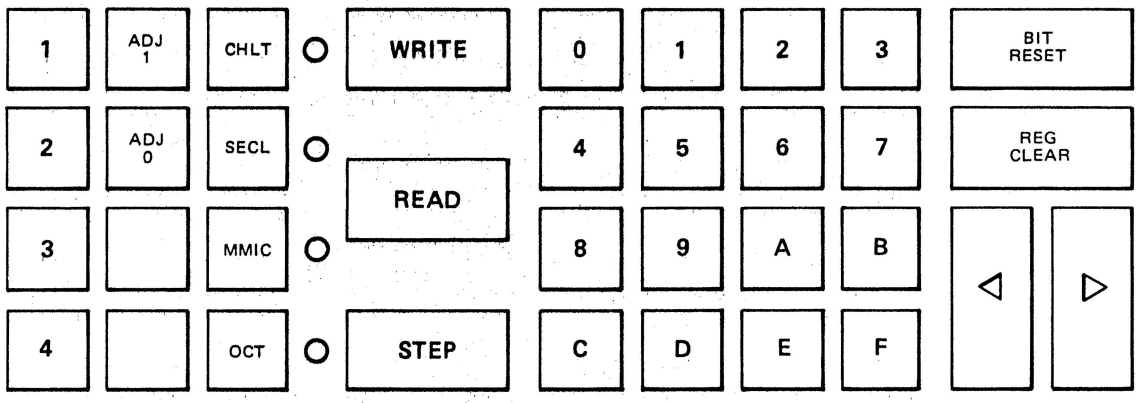



Figure 10-2-4. Keyboard Push Buttons and Indicators

## FLIP-FLOP RESET PROCEDURE

The BIT RESET button on the programmer's display keyboard must be pressed when any flip-flops on the status display panel or maintenance display registers are reset.

## MEMORY DUMPING PROCEDURE

The following procedure is used when the DP ODT message fails to produce a dump. The system should not be general-cleared prior to taking the memory dump.

1. The HALT indicator on the system control panel is pressed.
2. If the EVENT MODE switch on the MAINT CONTROL panel in the maintenance display processor control panel is on, it is turned off.
3. The A register select switch below display register 1 is pressed.
4. The P register select switch below display register 2 is pressed.
5. The ADJ (OP) switch on the programmer's keyboard is pressed.
6. Register select switch 2 is pressed. This switch selects the P register and positions the cursor at the left end of display register 2.
7. 3 AE 40 14 B0 B0 AB is entered in the P register at the programmer's display keyboard. This entry generates a tag of 3, MKST, NAMC(0,14), ZERO, ZERO, and ENTR, and calls the memory dump routine at D[0] + 14.
8. PROF, IIHF, and LROF on the status display panel are set.
9. The PSR2, PSR1, PSR0, VARF, EDIT, TEEF, and VECF indicators on the status display panel are reset.
10. If the SHLT indicator on the status display panel is set:
    a. IML2, IML1 and IML0 on maintenance display register 3, page 5 are reset.
    b. SHLT on the status display panel is reset.
    c. Check the setting of D[0] and reset if required.
    (1) The IC memory location 0 (D[0]) is read.
    (2) The value is verified to be correct. (2100 hex on a one processor system and on a TC system is dependent on memory.)
    (3) If the value is not correct, the correct value is

written into IC memory location 0. (Refer to the B 6800 System Reference Manual, form 5001290, Section 4, Maintenance Processor Control Panel Display for information on reading and writing to IC memory.)

11. On multiprocessor systems, the STEP switch on the other processors is pressed.

12. The STEP switch on the programmer's display keyboard is pressed, an the dump begins.

# 4. B7000 MEMORY DUMP PROCEDURE

## INTRODUCTION

This procedure is used when it is not possible to obtain a memory dump through either the DP or ??DP commands. This procedure supercedes the current procedure, which uses the T/F flip flop, and is to be used with an MCP level equal to or greater than 30.904.

Memory is organized such that the low 2000 hexadecimal words contain only MCP save code. D[0] is located at 2100 hexadecimal.

A memory dump to tape routine is included in the code which is read into memory during a Halt/Load operation. A physical Halt/Load from disk will cause 8192 words (2000 hexadecimal) to be transferred. Thus, information necessary for analysis is not destroyed.

## MEMORY DUMP PROCEDURE

The procedure to produce the tapedump is as follows:

1. Halt the system from the console switches by depressing the ENABLE and HALT buttons simultaneously.

2. Perform the following for each CPM:

   A. Place the CPM in TEST.

   B. Purge internal storage buffers (see Purge Procedures).

      (1) MOD II processor:

          a) Purge the Stack Buffer

      (2) MOD III processor:

          a) Purge the StoreQ

          b) Purge the Stack Buffer

   C. Record the following information (see Recording Procedures):

      (1) S-register

      (2) F-register

      (3) SDI/PIR/PSR

      (4) Instruction Being Executed

   D. Optionally, a processor panel dump taken at this point would provide additional helpful information.

   E. If this CPM is not the Halt/Load CPM, then place the CPM ON-LINE.

3. For the Halt/Load CPM do the following:

   A. Place the CPM in TEST.

   B. Press CONTROL CLEAR.

   C. Purge Code/Data Buffers (MOD III only).

   D. Press PAUSE.

   E. Place COND HALT switch ON.

4. Halt/Load the system. The Halt/Load CPM should halt with ALL LIGHTS ON in the top-of-stack registers.

5. Take the Halt/Load CPM out of COND HALT.

6. Wait at least ten (10) seconds, then press START once. The memory dump should commence.

7. Place the Halt/Load CPM back ON-LINE.

## PURGE PROCEDURES

I. MODEL II CPM:

   1. Purge the stack buffer:

      A. Place the PRG toggle in the UP position.

      B. Toggle the PLS switch.

      C. Restore the PRG toggle switch to the DOWN position.

II. MODEL III CPM:

   1. Purge the StoreQ:

      A. Place the CPM in SERIAL Mode.

      B. Restore the CPM to NON-SERIAL Mode, if desired.

   2. Purge the stack buffer:

      A. Place the CPM in SERIAL Mode.

      B. Set A-D PURGE LIGHT only.

      C. Press STORAGE PULSE.

      D. Reset A-D purge and place CPM in desired MODE.

3. Purge CODE/DATA Buffers:

   A. Set Code and Data purge lights.

   B. Press CONTROL CLEAR.

   C. Press Storage Pulse.

   D. Press CONTROL CLEAR.

## RECORDING PROCEDURES

I. MODEL II CPM

1. Place the CPM in TEST.

2. Purge the Stack Buffer.

3. Record the following registers:

   A. S - Panel 2, Row 5

   B. F - Panel 1, Row 7

   C. PIR/PSR - PC and CS, Panel 1, Row 14

   D. Instruction and variant - IER and DVE bit, Panel 1, Row 14.

   E. Additionally, the Program Address register (PA, Panel 1, Row 10) is sometimes helpful.

4. Read the current segment number (SDI) from display register (hex) 36.

   A. Clear the CPM by pressing CONTROL CLEAR.

   B. Set the PCU read pointer to (hex) 36.

      (1) Set the toggle switches 27,26,24, and 23 UP on Panel 1; all others DOWN.

      (2) Set the thumbwheel switches to: 1-08-9.

      (3) Press LOAD.

   C. The current segment number will appear in the display read register (Panel 1, Row 29).

MEMORY DUMP PROCEDURE

II. MODEL III CPM

1. Place the CPM in TEST.

2. If in NON-SERIAL Mode, Purge the StoreQ.

3. Record the following registers:

   A. S : Bus Select ON, XS - Panel 2, Row 15A

   B. F : Bus Select ON, XF - Panel 2, Row 15B

   C. SDI : PSDI - Panel 1, Row 15A

   D. PIR : PIN 15:16 - Panel 1, Row 15A

   E. PSR : PIN - S 2:3 - Panel 1, Row 15A

   F. Instruction : PID - Panel 1, Row 14A

4. Additionally, the following information is sometimes helpful:

   A. SNR : Super Select DRU, XSN - Panel 2, Row 3B

   B. CMR : Super Select EU, XCM - Panel 2, Row 34

   C. Fault Condition : Super Select PCU/Execute, Bus Select ON, IFR - Panel 1, Row 8A

   D. Processor Fail : Super Select PCU/Execute, Bus Select ON, IPF - Panel 1, Row 8C

# MEMORY MANAGEMENT

## TABLE OF CONTENTS


1. GENERAL INFORMATION . . . . . 11-1- 1

2. MEMORY ORGANIZATION . . . . . 11-2- 1

In-use Areas . . . . . 11-2- 1

Available Areas . . . . . 11-2- 3

3. ALLOCATION OF MEMORY SPACE . . . . . 11-3- 1

Memory Requests . . . . . 11-3- 1

Overlay . . . . . 11-3- 2

4. CONTROLS . . . . . 11-4- 1

SF Description . . . . . 11-4- 2

# 1. GENERAL INFORMATION

The Burroughs B 5000/B 6000/B 7000 series of computer systems uses a virtual memory technique based on variable-length program segments. Program segmentation is performed by the compiler at the COBOL section level, the FORTRAN subroutine level, and the ALGOL block level.

Variable-length program segments optimize storage utilization by:

1. Eliminating unnecessary program segments from memory.

2. Requiring only enough memory to contain the program segment (as opposed to a fixed amount of memory large enough to contain the largest program segment).

On variable-length program segment systems, program segments are resident only when they are required by a task.

A segment descriptor is associated with each program segment. Program segments and a dictionary of program segments are maintained on disk. Data areas also have a descriptor associated with each data array.

Each word of memory consists of 48 data bits and three tag bits. Tags are used by the hardware and the software to protect data and to determine which type of data or descriptor exists in the word. A user program may not attempt to change a word whose tag indicates that it is not an operand; such an attempt may result in program termination.

Descriptors reference each area of virtual memory currently in use and those areas of secondary storage which contain program or data segments related to a currently running task. These descriptors describe the type of area (program or data), the area size, and the memory or disk address of the area. A presence bit is also contained in each descriptor. This bit indicates the presence or absence (in memory) of a data or program segment. When the bit is on, the segment is present in memory; when the bit is off, the segment is on a secondary storage medium.

A unique descriptor called the MOM descriptor exists for each area; however, memory areas may be referenced by more than one descriptor. Multiple descriptors (descriptors in addition to the MOM) which reference the same area have a bit turned on which indicates that they are copy descriptors.

Copy descriptors for currently resident areas are identical to the MOM except for the copy bit. Copy descriptors which reference an area that is not present have the presence bit turned off; the address field contains the address of the MOM rather than the disk address of the area. The MOM contains the disk address of the absent segment. An array parameter passed from one process to another has a copy descriptor passed as the parameter. This copy descriptor references the same area as the MOM descriptor in the calling task.

Data descriptors can reference memory in other than 48-bit word units; this capability includes eight-bit characters, six-bit characters, four-bit characters, and 96-bit double precision words. Data descriptors can also declare areas to be read-only. Attempts to store in an area declared as read-only may result in program termination.

Areas referenced by data descriptors may contain information other than data, such as additional data descriptors. For example, an ALGOL two-dimensional array is represented as a descriptor which points to an area which contains additional data descriptors. Each data descriptor represents an index value of the first dimension of the array. Areas that contain these data descriptors are called "dope vectors".

Each element of a dope vector describes a row of the array. Each element of a dope vector may describe another dope vector that is required in order to describe a multidimensional array.

Large data areas (exceeding 1024 words) are automatically segmented into 256-word segments in order to prevent demands on the system for excessively large contiguous memory areas. An area segmented in this fashion is indicated by having the "segmented" bit turned on in its descriptor.

The segmented bit indicates that the descriptor describes a dope vector of descriptors. Descriptors that describe dope vectors of descriptors are referred to as "seg dope vectors". When indexing operations occur, the hardware locates the proper descriptor within the seg dope vector and indexes it by the correct amount. This type of segmentation is transparent to the user program.

Code segment descriptors are identified by a tag value of three; data descriptors are identified by a tag of five. This distinction allows the hardware and operating system to distinguish between the descriptors that point at code and descriptors that point at data.

# 2. MEMORY ORGANIZATION

The operating system structures memory so that each area is preceded and followed by "memory protected words". Memory protected words provide the link structure used for memory management.

## In-use Areas

In-use areas have four links associated with each area. Three words precede the in-use area, and one word follows it. These link words contain the following information that describes the associated in-use area:

1. Link A.
   a. A tag value of 6.
   b. Bits 47 and 46 ON.
   c. A current save bit, used to indicate that the area is currently saved.
   d. A save bit, used to indicate that the area is not to be overlayed (permanently saved).
   e. An in-use bit, used to indicate that the area is currently in-use.
   f. The memory address of the MOM descriptor.
   g. A temporary save bit, used to indicate that the area is temporarily saved.
   h. A bit that indicates the previous save status of the area. (Memory areas occasionally have their save status changed.)
   i. A bit which indicates that the area cannot be overlayed; however, the area may be relocated (sticky memory).
   j. The size of the area including links.
2. Link B.
   a. A tag value of 7.
   b. An overlay control field. This field designates whether an overlay is pending (value of 1) on this memory space, or that the space may be forgotten (value of 2) by whomever is attempting to overlay this space (a second process does a forgetspace on a memory space which has been targeted for overlay).
   c. The stack number that contains the MOM descriptor.
   d. A delta size field. This field contains a ccunt of additional/unrequested words which are part of a

memory space which is assigned to a stack. These words, though assigned, are unused and are inaccessible to the assigned stack as they are protected (tag value of 3).

e. The address of the location of the area in the overlay file or the code file.

3. Link C.

   a. A tag value of 3.

   b. The remainder of the word is not used directly by the memory management functions; however, it is used by other system functions, used by other system functions, and may contain such things as:

      (a) Object code address (in Rcw/Pcw format) which caused allocation of this memory space.

      (b) An Iocw to be used by the I/o subsystem when an I/o is to be done to/from this memory space.

      (c) An availability mask for this memory space if this memory space contains a DIRECT Array.

4. Link Z.

   a. A tag value of 3.

   b. A bit which indicates that the area is located in a memory module previously "saved" by the system operator.

   c. The size of the area.

   d. A bit which indicates that the area has delta words associated with it.

   e. The current save bit.

   f. A save bit.

   g. An in-use bit.

   h. A memory priority value.

## Available Areas

Available memory areas also have four link words associated with each area. Two words precede the available area and two words follow it. Available area link words contain the following information:

1. Link A.

   a. A tag value of 1.

   b. A stopper bit, used to indicate that the entry is the last entry in the available list.

   c. A usable bit, used to indicate that the area is available for use.

   d. The size of the area.

   e. A saved bit, used to indicate that the area is in the process of being saved.

f. The memory address of the next available area in this available list.

2. Link B.

   a. A tag value of zero.

   b. The address of the previous available area in this available list.

3. Link Y.

   a. A tag value of zero.

   b. The memory address of the previous entry in the available list.

4. Link Z.

   a. A tag value of 1.

   b. A stopper bit.

   c. A usable bit.

   d. A saved bit.

   e. An in-use bit.

   f. The memory address of the next entry in the available list.

   g. The size of the area.

(Links Y and Z are the trailing link words).

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# 3. ALLOCATION OF MEMORY SPACE

In-use memory areas are of two general types: save and non-save. Save areas are required to be resident (present in memory) from the time of allocation to the time of deallocation.

Non-save areas are overlayable areas and may be overlayed to disk by the operating system. These areas are automatically recalled to memory when they are referenced. Indiscriminate interspersion of save and non-save areas causes "checkerboarding" problems.

Checkerboarding problems are avoided by clustering save areas together in the lower portion of memory. This feature is employed by using two lists of memory areas: a save list and an overlayable list.

The save list links all available areas that immediately follow a save area. This list is ordered on area size.

The overlay list links all available areas that adjoin an overlayable area. This list is ordered on a last in/first out basis.

Every available area is referenced in one or both of these lists.

Checkerboarding problems may also be induced by fragmenting memory into many very small areas. This problem is resolved by prohibiting many requests for small amounts of memory. When a memory allocation is requested, the available memory areas are examined for an available area equal to the size requested or an available memory area which exceeds the size requested by no more than a delta factor. (The delta factor is 15 words.) If the area satisfies the request and is not greater than the delta factor, the space is assigned to the request; the delta size is set in link B and the delta indicator bit is set in link Z. Delta words always follow the in-use portion of the area and have a tag value of 3.

## Memory Requests

Requests for memory areas are invoked by a task which references a descriptor (code or data) that has the presence bit off. Such a reference causes a presence-bit interrupt to occur. When a presence-bit interrupt is sensed, the hardware constructs a non-present copy descriptor in which the address field contains the address of the MOM descriptor. After the descriptor is built, the hardware forces entry into the interrupt handling mechanism of the operating system, which:

1. Determines that the interrupt is a presence bit interrupt.

2. Calls the PRESENCEBIT procedure.

Code in the PRESENCEBIT procedure guarantees that other stacks are not permitted to make the same descriptor present asynchronously, by locking the MOM descriptor. The MOM descriptor is locked by turning on all bits in the address field of the absent MOM descriptor.

Before the MOM descriptor is locked, it is checked to ensure that it is in an unlocked state. If the MOM is found to be in a locked state, the task that attempts the presence-bit interrupt is suspended until the area is made present by the task currently performing the presence-bit action. Tasks are suspended in the operating system by waiting on the event DESCUNLOCK, which is caused

when the current presence-bit action is completed. On resumption of processing following DESCUNLOCK, the suspended task checks the MOM descriptor to ensure that it is still absent; if so, a presence-bit interrupt occurs and the presence-bit interrupt procedures described above are repeated.

When the MOM is made present, a stack search is performed on the task. All copy descriptors of the MOM are set to a present state, and their address field is altered to point to the memory area just made present.

The first presence-bit interrupt on data arrays causes the area to be allocated and zeroed by the operating system.

Copy descriptors that reside in stacks of other tasks (inter-program communication, IPC) are not made present until they are referenced by that task.

## Overlay

Memory requests for an area whose size cannot be satisfied by the available list causes the system to overlay non-save in-use areas. The actions performed are: (1) the copy descriptor is marked non-present, and (2) the address field is modified to point to the MOM descriptor. The MOM contains the disk address of the code segment. Data arrays must be written to the overlay file of the owning task. Because code segments are not modifiable, they need not be written to disk because a copy of the information exists in the code file.

To effect an overlay, memory areas are examined to determine if they are overlayable. If they are overlayable (non-save) the overlay control field is turned on in the memory link, which marks this area as being in the process of being overlayed. After the memory link has been marked, the operating system stops all other running processors, and an attempt is made to relocate the area to be overlayed to another memory area (core-to-core overlay). If the area was core-to-core overlayed, the operating system searches the stack of the owning task and alters both the MOM and all copy descriptors to point at the new memory area. Areas that cannot be relocated in memory have a stack search performed for all descriptors referencing the area. Copy descriptors are marked non-present and have their address field altered to point to the MOM descriptor. The MOM descriptor is locked and marked non-present. Because no reference to the area can be made by the owning task without generating a presence-bit interrupt, the processors are now allowed to run. The areas to be overlayed are written to the overlay file of the owning task and the memory areas are made available to the original requestor.

Each task has a unique overlay file assigned at run-time to which memory areas containing data may be overlayed. The operating system has its own stack and associated overlay file for use in overlaying global D[0] data arrays. Unreferenced MOM descriptors are detected by the operating system and have overlay file space allocated to them at their first presence-bit interrupt time. Overlay file space remains with the memory area until the program block that contains the MOM descriptor is exited, or the area is explicitly deallocated or resized by the user program. When the overlay file space is no longer required, it becomes available for use by a new area.

A resident array called OLAYINFO is associated with each overlay file. OLAYINFO has a bit mask with one bit for each segment of disk available within the overlay file. OLAYINFO is used to control the allocation and deallocation of disk space within the overlay file.

# 4. CONTROLS

The system increases throughput in a multiprogramming environment by allowing full utilization of processors and I/O channels. A virtual memory system can achieve full use of the resources of the machine by allowing tasks to run concurrently. An ideal environment is one in which the rate of overlay activity is sufficient to achieve full utilization. Because of the unpredictability of tasks and work loads, this optimum rate of overlay is sometimes exceeded when the aggregate working set requirements for a set of tasks currently running exceeds the amount of physical memory available. (A working set is defined as the amount of physical memory required to effectively run a task.) When the aggregate working set of currently running tasks exceeds physical memory, system performance degrades to an unacceptable level. Under such a condition, the system is constantly servicing presence-bit interrupts for tasks that continually reference non-present memory areas. This "thrashing" state is caused by tasks that have inaccurate working set estimates. Initial working set estimates are provided by the compilers and may be incorrect, because compilers cannot accurately predict the behavior of a program. The problem is compounded by the fact that the working set of a task may vary dramatically from run to run or within a single execution.

Actual working set requirements for each execution are computed at end-of-task time and stored in the program code file. Inaccuracies in the original compiler estimates are corrected by the operating system after a period of running.

The memory priority implementation allows the installation to prevent thrashing to some degree. This algorithm prevents a low priority program from overlaying memory areas belonging to a higher priority program for an installation-specified period of time. If not enough physical memory is available, the higher priority program is allowed access to physical memory at the expense of the lower priority program. The installation-supplied time interval (where the time interval is less than, or equal to, one second) is multiplied by the difference in the priority range. Thus, a memory area made present by a task is guaranteed residency for a period of time calculated as follows:

MRT=T*(HP-LP)

In above the algorithm, MRT is defined to be minimum residency time; T is defined to be installation specified time interval; HP is defined to be high priority; and LP is defined to be low priority.

If tasks are of equal priority, memory areas are guaranteed residency for one time interval. The priority algorithm is invoked through a variation of the Set Factor (SF) operator input message. The fourth variant of SF is used to set the memory priority time interval. The time interval must be specified as a percentage of one second. (For example, SF 4 100 specifies a time interval of one second.) A zero time interval turns off the memory priority algorithm.

The "working set" mechanism is another control that prevents thrashing. This mechanism allows the installation to specify a constant forced overlay rate. Specification of a working set overlay rate causes an independent (invisible) process to be initiated. This process (WSSHERIFF) cyclically examines memory and the overlay rate of each task. A task that exceeds the specified overlay rate is automatically suspended (lowest priority first) until the overlay rate reaches an acceptable level, at which time the task is reactivated. When the overlay rate is less that the specified limit, WSSHERIFF cyclically overlays areas until the specified per minute rate is attained.

The installation may also specify that a certain amount of memory is to be kept in the available list and ready for use on demand. This AVAILMIN specification is expressed as a percentage of the total physical memory on the system.

Invocation of the working set mechanism is also accomplished through variants of the SF operator input message. The percentage of memory to be overlayed per minute (OLAYGOAL) is specified by the first variant of SF. (For example, SF 1 20 causes 20% of the overlayable memory areas to be overlayed each minute.)

AVAILMIN is the percentage of memory to be maintained in the available list as specified by using the second variant of SF. (For example, SF 2 10 causes 10% of physical memory to be kept available.)

## SF Description

Factor 1: OLAYGOAL

The value of OLAYGOAL represents a goal percentage of the overlayable core in the system that is to be overlayed to disk on a per minute basis.

Factor 2: AVAILMIN

This factor only has an effect when factor 1 is greater than zero. AVAILMIN is expressed as a percentage of total core. In an effort to reduce thrashing, the lowest priority job in the mix is suspended when the total amount of available core drops below AVAILMIN. The following may never be suspended:

1. Control programs
2. Programs that are sorting

Factor 3: FACTOR

This factor is only used for scheduling and is expressed as a percentage of total memory. Total core is multiplied by factor to determine how much memory the system thinks it has for scheduling purposes. FACTOR can be greater than 100%.

Factor 4: MEM PRIORITY FACTOR

This factor is used to control the memory priority algorithm. When set to zero, memory priority is not used. If nonzero, it represents the percentage of one second per increment of priority that a low priority job must wait before it can overlay an area owned by a higher priority job.

If MEM PRIORITY FACTOR = 25% and a job of priority 50 makes an an area present in memory, that memory area may not be overlayed for use by a job of priority 30 for 5 seconds, ((50-30)*.25).

The following two examples of SF are hypothetical and are not intended to be recommendations.

```
SF 2 5 3 80

SF 90
```

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

RLTABLEGEN

TABLE OF CONTENTS


1. INTRODUCTION . . . . . 12-1- 1

2. INSTALLATION-DEFINED TAPE LABELS . . . . . 12-2- 1

3. USE OF SYSTEM/RLTABLEGEN . . . . . 12-3- 1

Label Description Format . . . . . 12-3- 1

ID Statement . . . . . 12-3- 2

RECOGNITION Statement . . . . . 12-3- 3

FIELD Statement . . . . . 12-3- 4

RECORD Statement . . . . . 12-3- 5

4. PROGRAM SPECIFICATIONS . . . . . 12-4- 1

# 1. INTRODUCTION

The SYSTEM/RLTABLEGEN program constructs value arrays and binds them to the MCP, thus, allowing system acceptance of nonstandard tape labels. The following paragraphs discuss the ability of the MCP to accept installation-defined tape labels. A description of the operation of the RLTABLEGEN program follows this discussion.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# 2. INSTALLATION-DEFINED TAPE LABELS

In addition to recognizing standard tape labels, the label recognition routine of the MCP (READALABEL) can recognize a large class of installation-defined labels.

To invoke this routine, a description of the desired nonstandard labels must be written and passed through a table-building program, SYSTEM/RLTABLEGEN. The program must bind the resulting value array to the MCP, and the system MCP then changes the MCP (CMs) to resulting MCP.

Tapes with installation-defined labels may then be used in the usual manner with the following restrictions:

1. The label records must be a contiguous group of BCL or EBCDIC records at the beginning of the tape, separated from the data by one tapemark.

2. The label type must be determined by examination of the first record only.

3. Label record size must be less than 230 characters.

4. The MCP does not recognize any user labels (for example, UTL, UHL, and so on) on these label types.

5. No "up tape" labels are recognized. That is, once the file is opened and the tape is positioned to the beginning of the data, the tape is treated as an unlabelled tape.

6. Because the label recognition routine checks for installation labels before checking for standard labels, the recognition sequence must be sufficiently complete so that it does not interfere with standard label recognition.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# 3. USE OF SYSTEM/RLTABLEGEN

The following section describes input to the program SYSTEM/RLTABLEGEN. This program is used to create the value array installation labels that are used by the MCP to identify nonstandard labels.

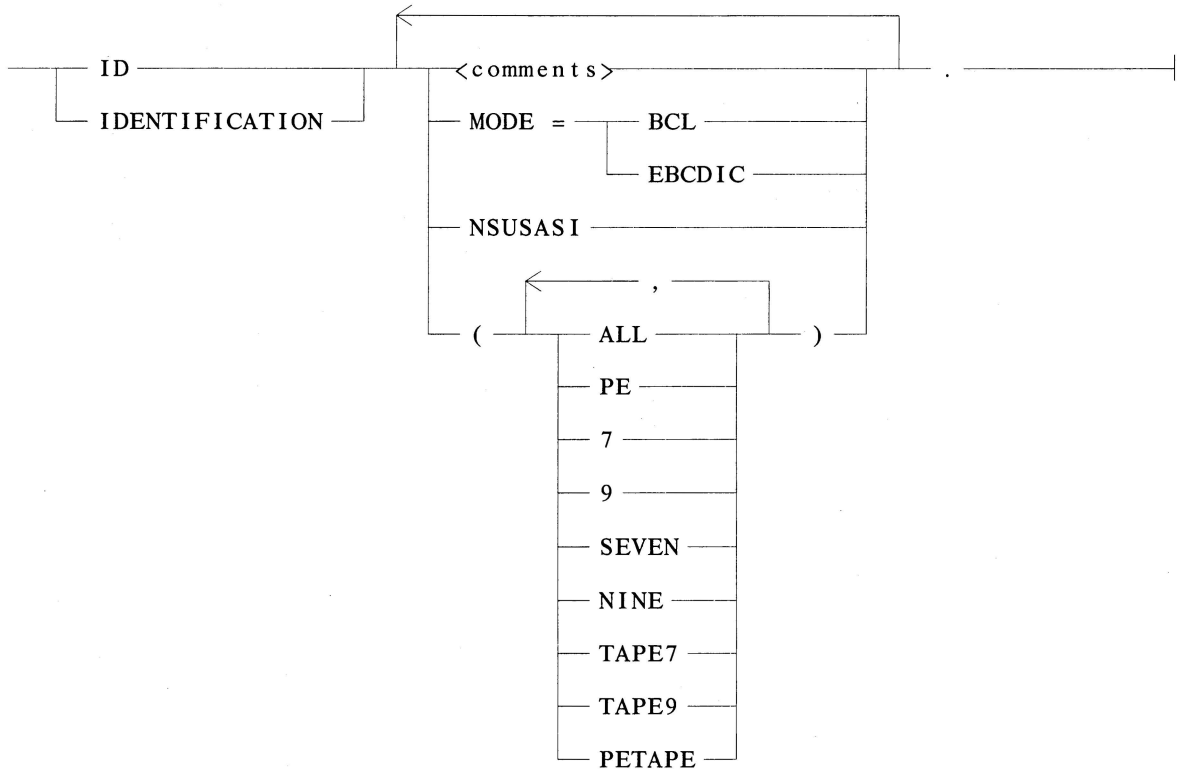
The input to SYSTEM/RLTABLEGEN may consist of any number of label descriptions. The elements of these label descriptions are discussed in the following sections.

## Label Description Format

A label description must consist of two divisions: the ID (or IDENTIFICATION) division and the FIELD (or FIELDS) division. A label description is constructed from the items discussed in the following paragraphs.

## ID Statement

### Syntax




### Semantics

The ID division is demarcated by the ID statement.

NSUSASI is required for nonstandard USASI tapes in which HDR2 and EOF2 label records are omitted. (Most forms of DOS omit these label records.) If RLTABLEGEN is used for such tapes and the NSUSASI attribute is not specified, the tape file closes with a label error because the MCP expects to find the non-present EOF2 record.

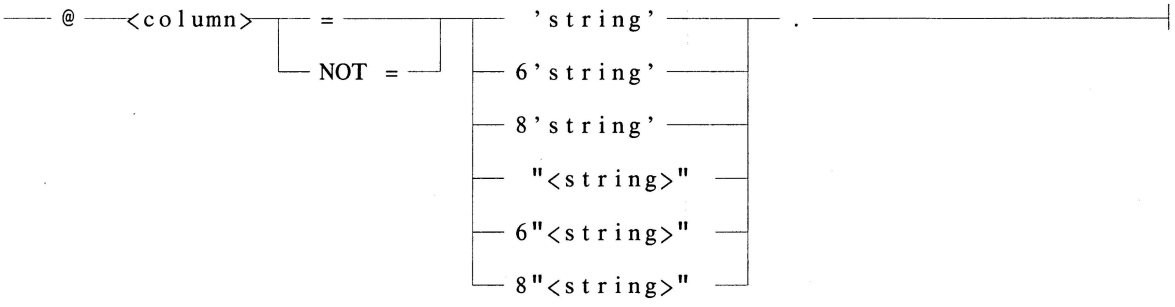
### Examples

```
ID MODE = EBCDIC FOR (PE, NINE) TRACK TAPES.

IDENTIFICATION FOR BRAND-X LABELS (7) MODE = BCL.
```

# RECOGNITION Statement

## Syntax




## Semantics

The RECOGNITION statement follows the ID statement.

<column> is an integer. String is any string except the quote character within the quotes and excluding the apostrophe in the upper three strings.

As many recognition statements as needed may appear.

# FIELD Statement

## Syntax

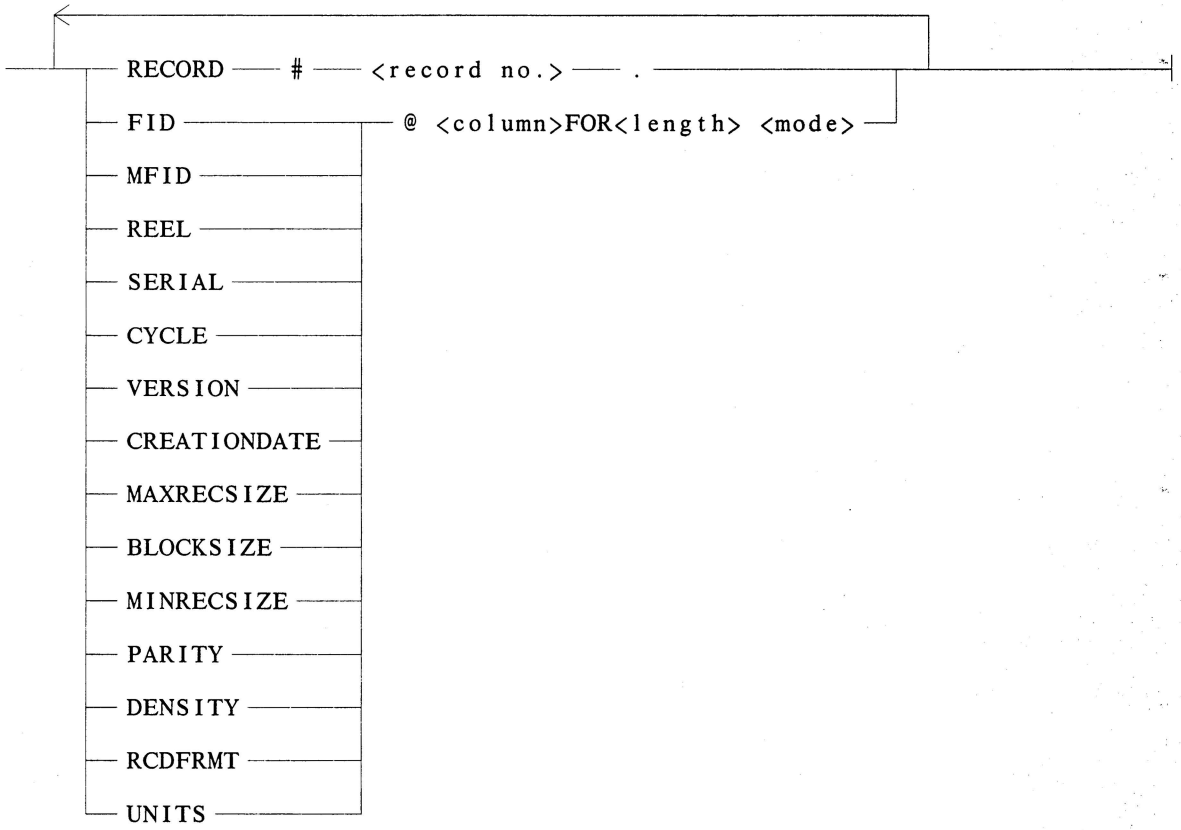



## Semantics

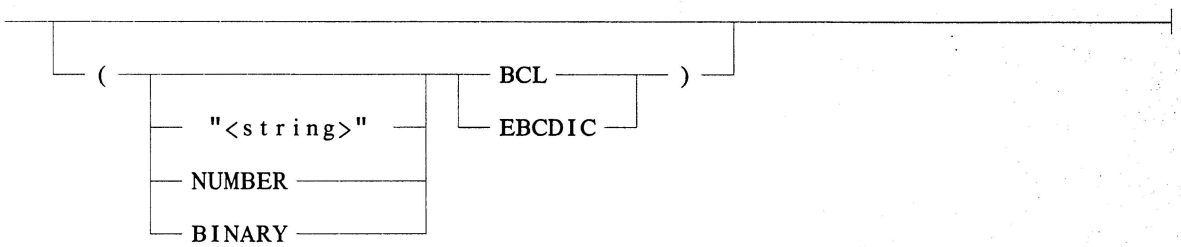
The FIELD division statement denotes the beginning of the description of the label.

## RECORD Statement

**Syntax**




<mode>



**Semantics**

The description of the various fields found in labels must follow the FIELD statement. The three types of label fields are bit fields, strings, and numbers.

Each field is used to store file attribute information.

The two basic statements that describe these fields are the record-number statement and the field-location statement. The record-number statement is used to group field location statements according to the records in which they appear. Counting within records of records and columns begins at one. Each record statement must reference a higher record than the previous one.

Field-location statements consist of a reserved word that describes the file attribute with which the value in the field is associated, followed by the starting column location within the current record and the length of the field.

<column> and <length> are integers that represent the one-relative location and length of the field being described.

Each attribute value is of an assumed type. The character type (BCL or EBCDIC) is assumed to be the type specified by the MODE clause in the ID statement. Attributes of STRING type may be requalified to override the expressed MODE but may not be changed to type BINARY or NUMBER.

NUMBER or BINARY attributes may be requalified to any mode or type except STRING.

The attributes are described below.

| | |
|---|---|
| MFID,FID | Type STRING. The maxumum length is 17 characters. Either field is optional. If both fields are missing or if both reference fields that are all blank, the tape is labeled "untitled". All leading and trailing blanks are deleted from the two fields. |

RCDFRMT — Type BINARY. The low-order eight bits of this field are assumed to be a BCL or EBCDIC letter. The following is a conversion table from letter to file type:

| Letter | File Type |
|---|---|
| F | 0 |
| D | 1 |
| V | 2 |
| I | 4 |
| L | 5 |
| Z | 6 |
| Any other letter | 3 |

PARITY — Type BINARY. This field is automatically set to the parity of the label records; but if it is specified, it must be in accordance with the way in which this label attribute works on B 6700 standard labels. (If the low-order bits are equal to one, standard parity is implied.)

DENSITY — Type BINARY. This field is automatically set based on the density of the label records. If specified, it must reflect the density in its low-order two bits.

UNITS — Type BINARY.

SERIAL — Type STRING. Maximum length is six characters. If this field is missing, the serial number is assumed to be "000000" (all zeros).

REEL, CYCLE, VERSION, CREATIONDATE, MAXRECSIZE, BLOCKSIZE — These attributes are all assumed to be integers; (digit strings of no more than 11 digits). They correspond to the file attributes of the same name. They are significant only if filetype 7 is set.

MINRECSIZE — These attributes are all assumed to be integers (digit strings of no more than 11 digits). They correspond to the file attributes of the same name. They are significant only if filetype 7 is set.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# 4. PROGRAM SPECIFICATIONS

Source-input records are accepted by the program via the reader file with the internal name CARD. The program output file consists of the printer listing named LINE.

The line printer listing contains a pseudo-reproduction of the input statements that the program generates from the table produced from the input statements. Successful operation has occurred when the essential information in the input records is contained in the output listing. The end of the output listing contains the value array generated by the program.

The RLTABLEGEN program automatically binds the generated value array to the MCP by zipping the program binder. RLTABLEGEN requires that the present MCP have the TITLE SYSTEM/MCP and that the code file generated by the binder have the TITLE SYSTEM/NEWMCP.

**Example**

In the following example, BCL labels on seven- and nine-track drives that can be identified by an ANSI VOL header can be recognized by use of the following deck, which produces an MCP that accepts these labels. (The only difference between program recognition of this type of BCL label and program recognition of B 6700 tape labels is that, in the former case, the SYSTEM-ID field is blank.)

```
<I>RUN SYSTEM/RLTABLEGEN; EBCDIC CARD
IDENTIFCATION FOR LOCAL LABEL SEQUENCE MODE=BCL (SEVEN, NINE).

      @ 01="VOL 1"
      @ 80='0'.
      @ 11 NOT=" ".
      @ 29=" ".
    FIELDS.
      RECORD # 1.
         MFID  @ 12 FOR 17.
      RECORD # 2.
         FID              @ 05 FOR 17.
         REEL             @ 32 FOR 4.
         CYCLE            @ 36 FOR 4.
         VERSION          @ 40 FOR 2.
         CREATIONDATE     @ 43 FOR 5.
      RECORD # 3.
         RCDFRMT          @ 05 FOR 1.
         BLOCKSIZE        @ 06 FOR 5 (NUMBER EBCDIC).
         MAXRECSIZE       @ 11 FOR 5.
         DENSITY          @ 16 FOR 1  (BINARY EBCDIC).
         PARITY           @ 18 FOR 1.
    <I>END
```

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

SECURITY

TABLE OF CONTENTS


1. INTRODUCTION . . . . . 13-1- 1

2. USERCODE SECURITY . . . . . 13-2- 1

PRIVILEGED USER . . . . . 13-2- 1

SHOWFILES . . . . . 13-2- 1

PASSWORD . . . . . 13-2- 1

SUSPENDED . . . . . 13-2- 1

3. FILE ATTRIBUTE SECURITY . . . . . 13-3- 1

SECURITYTYPE . . . . . 13-3- 1

PUBLIC . . . . . 13-3- 1

PRIVATE . . . . . 13-3- 1

GUARDED AND CONTROLLED . . . . . 13-3- 1

SECURITYUSE . . . . . 13-3- 1

SENSITIVEDATA . . . . . 13-3- 2

4. SYSTEM/MAKEUSER . . . . . 13-4- 1

5. DCALGOL COMPILER . . . . . 13-5- 1

6. OTHER SECURITY AREAS . . . . . 13-6- 1

ODT INPUT . . . . . 13-6- 1

WFL . . . . . 13-6- 1

CANDE . . . . . 13-6- 1

SYSTEM/SUMLOG . . . . . 13-6- 1

# 1. INTRODUCTION

This chapter gives a brief overview of the system security features. This information should prove useful for establishing computer system security procedures.

The file security system consists of several parts. These parts are: file attributes, usercodes, SYSTEM/MAKEUSER, the DCALGOL compiler, the WFL compiler, CANDE, SYSTEM/SUMLOG, and Operator Display Terminal (ODT) input. This chapter briefly describes each of these areas. Each of these parts is described more comprehensively in separate documents or in other chapters of the SOG volumes.

SECURITY

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# 2. USERCODE SECURITY

Files that belong to a user have that user's usercode appended to the front of the file title [for example, (AUZER)MY/FILE]. In this manner, which files belong to which users can be determined. Users may obtain access to other user files according to rules based on file attributes and usercode attributes. Usercode attributes applicable to file access are: PU, SHOWFILES, PASSWORD, and SUSPENDED.

By default, system access is granted without requiring usercodes. Sites may elect to require all users to present a valid usercode before granting access to the system. The SR (secure reader) ODT command provides enforcement of usercode checking for all batch jobs. The individual MCSs are responsible for validating usercodes of their attached Data Comm users.

## PRIVILEGED USER

When the PU (privileged user) bit in the USERDATAFILE is set for a usercode, all security checking is bypassed for that user. A privileged user is not denied access to any file on the system for security reasons. Privileged user status might, for example, be granted to responsible individuals performing systems programming functions.

## SHOWFILES

The SHOWFILES bit, when set for a user, allows all other users the privilege of seeing (via SYSTEM/FILEDATA, CANDE LFILES, and so forth) all PUBLIC files of that user.

## PASSWORD

A password list is associated with each usercode (each usercode may have between zero and 15 passwords). Passwords are used to validate the identity of a user. Each time a user is asked for his usercode he is also asked for his password, provided the associated MAXPW value of the USERDATAFILE is non-zero. Passwords should be changed frequently to take full advantage of password protection. Both CANDE and Work Flow Language (WFL) have syntax to allow password changes.

## SUSPENDED

The SUSPENDED bit in the USERDATAFILE provides a convenient way to suspend the service of a particular user. Refer to the MAKEUSER/USERDATA documentation in Chapter 9 of this manual for details.

SECURITY

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# 3. FILE ATTRIBUTE SECURITY

Users may specify, via file attributes, how other users may be granted access to their files. Applicable file attributes are: SECURITYTYPE, SECURITYUSE, SECURITYGUARD, and SENSITIVEDATA. These attributes are described in detail in the B 7000/B 6000 Series I/O Subsystem Reference Manual, form 5001779.

Security errors are frequently treated like no-file situations. Some errors generate error messages, while others do not; many cause the attending program to be DSed. All security errors are logged in SYSTEM/SUMLOG.

## SECURITYTYPE

The SECURITYTYPE attribute determines whether or not a particular file can be accessed or known to exist by another user. If a user fails to pass the security check specified by this attribute, the file management system returns a no-file indication. Possible values for this attribute are PRIVATE, PUBLIC, GUARDED, and CONTROLLED.

### PUBLIC

The security type PUBLIC (or CLASSA) allows any user to open the file. Permissible usage of the file is determined by the SECURITYUSE attribute.

### PRIVATE

The security type PRIVATE allows only the owner of the file to access that file. Use of the file is also governed by the SECURITYUSE attribute.

### GUARDED AND CONTROLLED

The security type GUARDED (or CLASSB) specifies that security checking is to be done according to the rules specified in the associated guardfile. A guardfile is associated with a file via the SECURITYGUARD attribute. SYSTEM/GUARDFILE is a utility program used to generate guardfiles.

## SECURITYUSE

The SECURITYUSE attribute allows a user to specify how a file may be used. Possible values are IN, OUT, IO, and SECURED. IN means input; OUT means output; IO means both input and output; and SECURED means for execution only.

## SENSITIVEDATA

If the SENSITIVEDATA attribute is set for a disk-type file, the MCP overwrites the space occupied by the file to erase any residual data when the file is purged. When SENSITIVEDATA is not specified, disk areas are not erased and may be read by a subsequent user of the disk space. Setting the SENSITIVEDATA attribute can help keep information confidential.

# 4. SYSTEM/MAKEUSER

The SYSTEM/MAKEUSER utility is used to add, delete, and modify usercode entries, which reside in the SYSTEM/USERDATAFILE. The SYSTEM/MAKEUSER utility is described in detail in Chapter 9 of this manual. No attempt is made to describe MAKEUSER details here; however, the following three points should be considered:

1. When SYSTEM/USERDATAFILE is not present on the HALT/LOAD unit (or another unit specified via the DL ODT command), SYSTEM/MAKEUSER may be run without a usercode to establish a userdatafile. All other executions of SYSTEM/MAKEUSER require it to be run under a privileged usercode.

2. Care should be taken to ensure that only authorized individuals have access to SYSTEM/MAKEUSER. Because execution of the program will probably be infrequent, it may be advisable not to leave the program on disk.

3. SYSTEM/USERDATAFILE contains an MU model. The MU model provides a mechanism that can prevent operators from establishing new usercodes and/or making established usercodes privileged.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# 5. DCALGOL COMPILER

The DCALGOL compiler provides extensions and features implemented for writing system software. For this reason, an installation may consider limiting access to this compiler. Some of the features that might be considered a breach of security are: MAKEUSERCODE, DCWRITE and other Data Comm constructs, GETSTATUS, SETSTATUS, USERDATA, and DCKEYIN. Although most of these features require a privileged usercode, access to the DCALGOL compiler should be limited.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# 6. OTHER SECURITY AREAS

## ODT INPUT

The MU and PU ODT commands allow an operator some control over the USERDATAFILE. The MU (make usercode) command allows the operator to establish a new usercode. The PU (privileged user) command allows the operator to make a usercode privileged or not privileged. Both MU and PU inputs may be controlled through the use of the MU model.

## WFL

The WFL compiler allows the SECURITYTYPE, SECURITYUSE, and SECURITYGUARD attributes of a file to be changed via the SECURITY STATEMENT.

## CANDE

The CANDE MCS allows the SECURITYTYPE, SECURITYUSE, and SECURITYGUARD attributes of a file to be changed via the SECURITY COMMAND.

## SYSTEM/SUMLOG

All security violations are logged in the SYSTEM/SUMLOG. SYSTEM/LOGANALYZER can print a listing of these security violations when run with a privileged usercode. (Refer to LOGANALYZER documentation in Chapter 6 of this manual for syntax.)

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

SOFTWARE COMPILATION GUIDE

TABLE OF CONTENTS


1. INTRODUCTION . . . . . 14-1- 1

2. GENERAL PROCEDURE . . . . . 14-2- 1

General Descriptions . . . . . 14-2- 1

PASS TWO DESCRIPTION . . . . . 14-2- 2

Compilation Rules . . . . . 14-2- 3

File Dependency . . . . . 14-2- 4

Required Files . . . . . 14-2- 5

Sample Decks . . . . . 14-2- 7

PASS ONE SAMPLE DECK . . . . . 14-2- 7

PASS TWO SAMPLE DECK . . . . . 14-2- 15

3. PROCEDURE EXPLANATION . . . . . 14-3- 1

General Explanation . . . . . 14-3- 1

PASS ONE DETAILS . . . . . 14-3- 1

PASS TWO DETAILS . . . . . 14-3- 2

Intrinsics Generation . . . . . 14-3- 3

COMPILING INTRINSICS . . . . . 14-3- 3

BINDING INTRINSICS . . . . . 14-3- 3

Standard Intrinsics . . . . . 14-3- 3

Installation Intrinsics . . . . . 14-3- 4

SOURCENDL . . . . . 14-3- 6

Table Generating Programs . . . . . 14-3- 7

TABLEGEN . . . . . 14-3- 7

ALGOLTABLEGEN . . . . . 14-3- 8

COBOLTABLEGEN . . . . . 14-3- 9

ATTABLEGEN . . . . . 14-3- 10

CCTABLEGEN . . . . . 14-3- 11

LTTABLEGEN . . . . . 14-3- 12

RLTABLEGEN . . . . . 14-3- 12

SCTABLEGEN . . . . . 14-3- 12

4. SOFTWARE GENERATION OPTIONS . . . . . 14-4- 1

ALGOL Compile-Time Options . . . . . 14-4- 1
ALGOL, DCALGOL AND DMALGOL . . . . . 14-4- 1
B7000CODE (RESET) . . . . . 14-4- 1
CTPROC (RESET) . . . . . 14-4- 1
BINDER Compile-Time Option . . . . . 14-4- 1
COMPBIND (RESET) . . . . . 14-4- 1
COBOL Compile-Time Options . . . . . 14-4- 1
ANSI74 (RESET) . . . . . 14-4- 1
B2500 (RESET) . . . . . 14-4- 1
B7700CODE (RESET) . . . . . 14-4- 2
ESPOL Compile-Time Options . . . . . 14-4- 2
BCL (SET) . . . . . 14-4- 2
B7700CODE (RESET) . . . . . 14-4- 2
FORTRAN Compile-Time Options . . . . . 14-4- 2
BIGOPTJOB (RESET) . . . . . 14-4- 2
B7700CODE (RESET) . . . . . 14-4- 2
FASTGRAPH (SET) . . . . . 14-4- 2
GETFIRSTDOLLARCARDFROMSPO (RESET) . . . . . 14-4- 2
MODEL1 (RESET) . . . . . 14-4- 2
MULTI (SET) . . . . . 14-4- 2
NOSEGS (RESET) . . . . . 14-4- 3
OPTLSS1 (RESET) . . . . . 14-4- 3
SETOWNARRAYS (RESET) . . . . . 14-4- 3
SINGLEBYDEFAULT (SET) . . . . . 14-4- 3
GENSUPP Compile-Time Options . . . . . 14-4- 3
MINUSZEROFORBLANKFIELD (RESET) . . . . . 14-4- 3
MCP Compile-Time Options . . . . . 14-4- 3
ACTIVETIME (RESET) . . . . . 14-4- 3
B7000 (RESET) . . . . . 14-4- 3
DEBUG (RESET) . . . . . 14-4- 3
DEBUGMCP (RESET) . . . . . 14-4- 4
DIAGNOSTICS (RESET) . . . . . 14-4- 4
EXPERIMENTAL (RESET) . . . . . 14-4- 4

INTERNAL (RESET) . . . . . 14-4- 4
LINEINFO (SET) . . . . . 14-4- 4
LOCKTRACE (RESET) . . . . . 14-4- 4
NODUMP (RESET) . . . . . 14-4- 4
OVERHEADCHARGED (RESET) . . . . . 14-4- 4
PRESENCEBITCHARGED (RESET) . . . . . 14-4- 5
READLOCK (RESET) . . . . . 14-4- 5
READLOCKTIMEOUT (SET) . . . . . 14-4- 5
STATISTICSET (RESET) . . . . . 14-4- 5
SORTSTAT (RESET) . . . . . 14-4- 5
SWAPTRACE . . . . . 14-4- 5
TAPEOP (RESET) . . . . . 14-4- 5
USERDATATROUBLE (RESET) . . . . . 14-4- 5
PL/I Compile-Time Options . . . . . 14-4- 6
B7700CODE . . . . . 14-4- 6
DUMP (RESET) . . . . . 14-4- 6
5. B 7000 SOFTWARE GENERATION . . . . . 14-5- 1
INTRODUCTION . . . . . 14-5- 1
FILE DEPENDENCIES . . . . . 14-5- 1
COMPILER COMPILE-TIME OPTIONS . . . . . 14-5- 2
B7700CODE . . . . . 14-5- 2
MCP COMPILE-TIME OPTIONS . . . . . 14-5- 2
B7800ONLY (RESET) . . . . . 14-5- 2
FREEZEOUTTC (RESET) . . . . . 14-5- 2
COMPILE DECK EXAMPLE . . . . . 14-5- 3
Introduction . . . . . 14-5- 3
Compilation Job Deck . . . . . 14-5- 4

# 1. INTRODUCTION

The material contained in this section may be used as a guide when compiling system software. The WFL statements in Chapter 2 are provided to exemplify compilation order, appropriate dollar options, and label equation of compiler files.

The information in the following documents pertains to the material discussed in this manual:

| | Publication Form No. |
|---|---|
| B 5000/B 6000/B 7000 Series ALGOL Reference Manual | 5011760 |
| B 7000/B 6000 BINDER Reference Manual | 5001456 |
| B 5000/B 6000/B 7000 Series Operator Display Terminal Reference Manual | 5011687 |
| B 7000/B 6000 Series System Software Operational Guide, Volume 1 | 5011661 |
| B 5000/B 6000/B 7000 Series System Software Operational Guide, Volume 2 | 5011679 |
| B 5000/B 6000/B 7000 Series Work Flow Language Reference Manual | 5011794 |

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# 2. GENERAL PROCEDURE

When system software is compiled, a two-pass procedure is used. A new MCP is generated on the first pass. After changing to the newly generated MCP by performing a Change MCP (CM) Operator Display Terminal (ODT) command, the rest of system software is recompiled under the new MCP.

## General Descriptions

### Symbolic Consolidation and Compilation of the NEWP MCP

The MCP is grouped into logical modules. Each global item belongs to a particular module. Various "interfacing" defines have been added for accessing data from other modules.

Several external procedures that used MCP data structures have been moved into the MCP symbolic; these procedures include SORT, MAINTENANCE, READER/ SORTER routines, and some intrinsics that were previously written in ESPOL.

These changes have affected MCP generation in the following ways:

1. DUMPINFO and LOADINFO are not supported.

2. Dollar cards that do not begin with SET or RESET do not affect unmentioned options. To RESET all options, $CLEAR can be used.

3. The compiler produces its own cross-reference; SYSTEM/XREFANALYZER need not be run.

4. NEWP has a MAKEHOST and a SEPCOMP facility similar to ALGOL, except that NEWP does its own binding.

5. SORT and MAINTENANCE are not separately compiled and bound. They can, however, be changed using SEPCOMP. An example of a SEPCOMP job is listed below.

```
JOB NEWP/SEP;
BEGIN
PROCESS SYSTEM/PATCH [PT];
 FILE TAPE (TITLE=SYMBOL/MCP),
      PATCH (TITLE=PATCH/NEWPSEP);
DATA
$.COMPARE MARK
&#
$CLEAR LINEINFO MCP SEPCOMP "SYSTEM/MCP."
<PATCHES>
 ? % END PATCH INPUT
WAIT(PT(VALUE)=1);
IF PT(VALUE) NEQ 1 THEN ABORT "BAD PATCH";

 COMPILE SEP/MCP WITH NEWP [CT] LIBRARY;
 COMPILER FILE TAPE (TITLE=SYMBOL/MCP),
                CARD (TITLE=PATCH/NEWPSEP);
IF CT ISNT COMPILEDOK THEN ABORT "BAD COMPILE";
REMOVE PATCH/NEWPSEP;
END JOB.
```

6. The DCALGOL components of the MCP are compiled with DCALGOL and bound to the MCP.

7. The ESPOLINTRINSICS no longer do a LOADINFO from the MCP.

**PASS ONE DESCRIPTION**

The following steps are performed in pass one:

1. Standard intrinsics and support libraries are ensured to be in use and are changed, if necessary, using the SI and SL Operator Display Terminal (ODT) commands.

2. ALGOL is compiled first because it is the basis for system software.

3. The ESPOL, DCALGOL, and NEWP compilers are generated. A new USERSTRUCTURE program and BINDER are created. These software items are used when compiling the MCP and generating new INTRINSICS and system support libraries.

4. The intrinsic files (ESPOLINTRINSICS and BASEINTRINSICS) are compiled separately and bound together with any installation intrinsics to create SYSTEM/INTRINSICS.

5. USERSTRUCTURE is patched and recompiled, generating a NEWTAPE ($SET NEW), and then executed. The output is a patch to GENERALSUPPORT.

6. The system support libraries (GENERALSUPPORT, PLISUPPORT, and BASICSUPPORT) are compiled.

7. The new MCPHOST is compiled with the new NEWP compiler.

8. WFL, along with its associated separate procedures (CCSTRINGCONV, CCVARIABLEPPB, and CCSTRINGFUNCTION), and CONTROLLER are compiled referencing the new MCP SYMBOLIC. JOBFORMATTER and ALGOLSUPPLEMENT are compiled with the new DCALGOL.

9. These newly generated files are bound with the MCP HOST to generate the new MCP.

**PASS TWO DESCRIPTION**

After changing to the new MCP using the CM ODT command, changing to the new INTRINSICS using the SI ODT command, and changing to the new system support libraries using the SL ODT command, all system software is compiled under the new MCP during pass two.

1. COMPBIND is compiled first because it is automatically bound into ALGOL and FORTRAN.

2. ALGOL is compiled again under the new MCP.

3. The various compilers are compiled using the new ALGOL.

4. The remaining software is compiled using the new compilers.

## Compilation Rules

A few general rules apply when software is recompiled. The following rules apply to a thorough recompilation; however, all these compilations need not necessarily be performed.

1. In general, when a compiler is recompiled, all programs compiled with that compiler should be recompiled.

2. When ALGOL is recompiled, all system software should be recompiled (procedures similar to both the Pass One and Pass Two decks should be performed).

3. When the MCP HOST is patched, the following steps should be performed:

   a. All code files bound into the MCP should be recompiled and bound with the new MCP HOST to create a new MCP.

   b. The intrinsics should be recompiled and bound.

   c. All system software should be recompiled under the new MCP (a procedure similar to the Pass Two deck should be performed).

4. When ESPOL is recompiled, the following steps should be performed:

   a. The ESPOLINTRINSICS should be recompiled, and new INTRINSICS should be bound.

   b. All system software should be recompiled under the new MCP to insure integrity (a procedure similar to the Pass Two deck should be performed).

5. A patch to the BINDER should be followed by these steps:

   a. The MCP and INTRINSICS should be rebound with the new BINDER. They need not be recompiled.

   b. COMPBIND should be recompiled with the new BINDER symbolic.

   c. The ALGOL and FORTRAN compilers should be recompiled to utilize the new COMPBIND.

6. When NEWP is recompiled, the following steps should be performed:

   a. The MCP host should be recompiled.

   b. A new MCP should be bound; CONTROLLER and WFL need not be recompiled.

   c. All system software should be recompiled under the new MCP to insure integrity (a procedure similar to the Pass Two deck should be performed).

7. When the system support libraries are recompiled locally, the installation manager may mark them as control programs (CP) or privileged transparent programs (PP:TRANSPARENT), or both, if desired. The following system support libraries are supplied as indicated:

| FILE NAME | CPed | PPed |
|---|---|---|
| BASICSUPPORT | YES | YES |
| BILLINGSUPPORT | YES | |
| BNA | YES | |
| CANDE | YES | |
| GENERALSUPPORT | YES | YES |
| HARDCOPY | YES | |
| HOSTNSPSUPPORT | YES | |
| KEYEDIO | YES | YES |
| PLISUPPORT | YES | YES |
| RPGSUPPORT | YES | YES |
| RJE | YES | |
| SECURITYSUPPORT | YES | |
| SUSPENDER | YES | |
| EDITOR | YES | |

## File Dependency

Most system software is compiled using ALGOL, with the following exceptions:

1. The NEWP compiler is used to compile the following software: BASISSUPPORT, GENERALSUPPORT, HOSTNSPSUPPORT, LOADER, PLISUPPORT, MCPHOST, PRINTERDUMP, RPGSUPPORT, and UTILOADER.

2. The ESPOL compiler is used to compile the ESPOLINTRINSICS.

3. The DCALGOL compiler is used to compile the following software: ALGOLSUPPLEMENT, BACKUP, BARS, BILLINGSUPPORT, BNA, CANDE, CONFIGURATOR, CONTROLLER, DCAUDITOR, DCSTATUS, DIAGNOSTICMCS, DUMPALL, FILECOPY, FILEDATA, EDITOR, HARDCOPY, HOSTSERVICES, IADMAPPER, JOBFORMATTER, LISTVOLUMELIB, LOGANALYZER, LOGGER, MAKEUSER, RJE, SECURITYSUPPORT, STATSTARTER, STATSTOPPER, SUSPENDER, and WFL.

4. The NDL compiler is used to compile SOURCENDL.

5. The following software references the MCPSYMBOLIC file: CONTROLLER, WFL, and LISTVOLUMELIB.

6. BINDER is used when generating the following software: ALGOL and FORTRAN (COMPBIND for AUTOBIND), INTRINSICS, MCP, and NDL (DCPPROGEN is bound in).

7. The following software references the JOBFORMATTER symbolic: LOGGER and LOGANALYZER.

8. Any patches to the table-generating programs ATTABLEGEN, SCTABLEGEN, CCTABLEGEN, COBOLTABLEGEN, TABLEGEN, ALGOLTABLEGEN, and COBOL74TABLEGEN must be made before these programs are compiled. The resulting patch deck generated by running any of these programs must be included in the patches for the final compilation.

## Required Files

The following symbol files should be copied and compared from the latest release tapes when recompiling all system software:

SYMBOL/ALGOL
SYMBOL/ALGOLTABLEGEN
SYMBOL/ATTABLEGEN
SYMBOL/BACKUP
SYMBOL/BASEINTRINSICS
SYMBOL/BASIC
SYMBOL/BASICSUPPORT
SYMBOL/BINDER
SYMBOL/CANDE
SYMBOL/CARDLINE
SYMBOL/CCTABLEGEN
SYMBOL/COBOL
SYMBOL/COBOLTABLEGEN
SYMBOL/COMPARE
SYMBOL/CONFIGURATOR
SYMBOL/CONTROLLER
SYMBOL/DCAUDITOR
SYMBOL/DCPPROGEN
SYMBOL/DCPTESTGEN
SYMBOL/DCSTATUS
SYMBOL/DIAGNOSTICMCS
SYMBOL/DUMPALL
SYMBOL/DUMPANALYZER
SYMBOL/ESPOL (B 5000/B 6000 series systems)
SYMBOL/B7X/ESPOL (B 7000 series systems)
SYMBOL/ESPOLINTRINSICS
SYMBOL/FILECOPY
SYMBOL/FILEDATA
SYMBOL/FORTRAN
SYMBOL/FORTRAN77
SYMBOL/GENERALSUPPORT
SYMBOL/GUARDFILE
SYMBOL/HARDCOPY
SYMBOL/HOSTSERVICES
SYMBOL/IADMAPPER
SYMBOL/INTERACTIVEXREF
SYMBOL/JOBFORMATTER
SYMBOL/LISTVOLUMELIB
SYMBOL/LOADER (B 5000/B 6000 series systems)
SYMBOL/B7X/LOADER (B 7000 series systems)
SYMBOL/LOGANALYZER
SYMBOL/LOGGER
SYMBOL/LTTABLEGEN
SYMBOL/MAKEUSER
SYMBOL/MCPnnnnn (nnnnn indicates current release level)
SYMBOL/MINILOADER (B 7000 series systems)
SYMBOL/NDL
SYMBOL/NDLII
SYMBOL/NEWP
SYMBOL/NSPDUMPANALYZER
SYMBOL/PATCH
SYMBOL/PLISUPPORT
SYMBOL/PL/I
SYMBOL/PRINTBINDINFO
SYMBOL/PRINTCOPY
SYMBOL/RESEQBASIC

SYMBOL/RJE
SYMBOL/RLTABLEGEN
SYMBOL/SCTABLEGEN
SYMBOL/SORTSTAT
SYMBOL/SOURCENDL
SYMBOL/STATSTARTER
SYMBOL/STATSTOPPER
SYMBOL/SUSPENDER
SYMBOL/TABLEGEN
SYMBOL/USERSTRUCTURE
SYMBOL/UTILOADER (B 5000/B 6000 series systems)
SYMBOL/WFL
SYMBOL/XREFANALYZER

If COBOL74 is desired, the following files should also be present:

SYMBOL/COBOL74
SYMBOL/COBOLTABLEGEN74
SYMBOL/KEYEDIO

If LCOBOL is desired, the following files should also be present:

SYMBOL/LCOBOL
SYMBOL/LCOBDOWNLINE
SYMBOL/LCOBLIBRARY

If BNA is desired, the following file should also be present:

SYMBOL/BNA

If FORTRAN77 is desired, the following file should also be present:

SYMBOL/FORTRAN77

If MAD (B 7000 series systems) is desired, the following file should also be present:

SYMBOL/MDUTILITY

The following system files should be present when recompiling all system software:

SYSTEM/ALGOL
SYSTEM/BINDER
SYSTEM/COMPBIND
SYSTEM/GENERALSUPPORT
SYSTEM/INTRINSICS

## Sample Decks

The following listings are examples of a compilation order, appropriate dollar options, and compiler files for the various software items. The user may use procedures similar to those described in this section, but the user's WFL deck may not be exactly like these examples.

### PASS ONE SAMPLE DECK

The following example may be followed when recompiling the MCP and its bound procedures:

```
?BEGIN JOB PASSONE;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%      INSERT ALL VARIABLE DECLARATIONS      %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

INTEGER TEMP; %COUNTER FOR SUCCESSFUL ALGOL COMPILE

TASK T;       %USED IN OUTER BLOCK

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%         ALGOL COMPILE GLOBAL DECK          %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   DATA ALGOLDECK
$ ALGOL MERGE SEQERR NEWSEQERR
$ SET LINEINFO
$ SET NEW
?


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%                    ESPOL                    %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE ESPOLCOMP;
BEGIN  TASK T;
  COMPILE SYSTEM/ESPOL ALGOL[T] LIBRARY;
     ALGOL OPTION = (BDBASE,FAULT,FILE,DSED);
     ALGOL FILE TAPE(TITLE = SYMBOL/ESPOL);
     ALGOL FILE NEWTAPE(TITLE = SYMBOL/ESPOL);
  ALGOL DATA
$ MERGE SEQERR NEWSEQERR
$ SET LINEINFO
$ SET NEW
?
 IF T IS COMPILEDOK
   THEN WAIT ("MC ESPOL",OK)
   ELSE ABORT "RECOMPILE ESPOL";
END; %ESPOLCOMP
```

```
%%%
```

```
%%%
```

```
%%%
```

```
%%%
```

```
%%%
```

```
%%%
```

```
%%%
```

**PASS TWO SAMPLE DECK**

This sample deck shows one possible compilation procedure:

```
?BEGIN JOB PASSTWO;
OPTION = (ARRAYS,FAULT,FILES,BASE);

%%%
```

```
%%%
```

```
%%%
```

```
%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%                    FORTRAN                    %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE FORTRANCOMP;
BEGIN  TASK T;
  COMPILE SYSTEM/FORTRAN  ALGOL[T]  LIBRARY;
     ALGOL OPTION = (BDBASE,FAULT,FILE,DSED);
     ALGOL FILE TAPE(TITLE = SYMBOL/FORTRAN);
     ALGOL FILE NEWTAPE(TITLE = SYMBOL/FORTRAN);
  ALGOL DATA
$ MERGE SEQERR NEWSEQERR
$ SET NEW   LINEINFO
?
    IF T IS COMPILEDOK
            THEN WAIT ("MC FORTRAN",OK)
            ELSE ABORT "RECOMPILE FORTRAN";
END;



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%                  FORTRAN77                    %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE FORTRAN77COMP;
BEGIN  TASK T;
  COMPILE SYSTEM/FORTRAN77  ALGOL[T]  LIBRARY;
     ALGOL OPTION = (BDBASE,FAULT,FILE,DSED);
     ALGOL FILE TAPE(TITLE = SYMBOL/FORTRAN77);
     ALGOL FILE NEWTAPE(TITLE = SYMBOL/FORTRAN77);
  ALGOL DATA
$ MERGE SEQERR NEWSEQERR
$ SET NEW   LINEINFO
?
    IF T IS COMPILEDOK
            THEN WAIT ("MC FORTRAN77",OK)
            ELSE ABORT "RECOMPILE FORTRAN77";
END;



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE PLICOMP;
BEGIN  TASK T;
  COMPILE SYSTEM/PL/I ALGOL[T] LIBRARY;
     ALGOL OPTION = (BDBASE,FAULT,FILE,DSED);
     ALGOL FILE TAPE(TITLE = SYMBOL/PL/I);
     ALGOL FILE NEWTAPE(TITLE = SYMBOL/PL/I);
  ALGOL DATA
$ MERGE SEQERR NEWSEQERR
$ SET LINEINFO  NEW
?
  IF T IS COMPILEDOK
     THEN WAIT ("MC PL/I",OK)
     ELSE ABORT "RECOMPILE PL/I";
END;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%          BINDER          %%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE BINDERCOMP;
BEGIN  TASK T;
  COMPILE SYSTEM/BINDER  ALGOL[T]  LIBRARY;
    ALGOL OPTION = (BDBASE,FAULT,FILE,DSED);
    ALGOL FILE TAPE(TITLE = SYMBOL/BINDER);
    ALGOL FILE NEWTAPE(TITLE = SYMBOL/BINDER);
  ALGOL DATA
$ MERGE SEQERR NEWSEQERR
$ SET LINEINFO
$ SET NEW
?
  IF T IS COMPILEDOK
     THEN WAIT ("MC BINDER",OK)
     ELSE ABORT "RECOMPILE BINDER";
END;
```

```
%%%
```

```
%%%
```

```
%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
SUBROUTINE GUARDFILECOMP;
BEGIN  TASK T;
  COMPILE SYSTEM/GUARDFILE ALGOL[T]  LIBRARY;
    ALGOL OPTION = (BDBASE,FAULT,FILE,DSED);
    ALGOL FILE TAPE(TITLE = SYMBOL/GUARDFILE);
    ALGOL FILE NEWTAPE(TITLE = SYMBOL/GUARDFILE);
  ALGOL DATA
$ MERGE SEQERR NEWSEQERR
$ SET  LINEINFO  NEW
?
  IF T ISNT COMPILEDOK THEN
    ABORT "RECOMPILE GUARDFILE";
END;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
SUBROUTINE IXREFCOMP;
BEGIN  TASK T;
  COMPILE SYSTEM/INTERACTIVEXREF ALGOL[T] LIBRARY;
    ALGOL OPTION = (BDBASE,FAULT,FILE,DSED);
    ALGOL FILE TAPE(TITLE = SYMBOL/INTERACTIVEXREF);
    ALGOL FILE NEWTAPE (TITLE = SYMBOL/INTERACTIVEXREF);
  ALGOL DATA
$ MERGE SEQERR NEWSEQERR
$ SET LINEINFO NEW
?
  IF T ISNT COMPILEDOK THEN
     ABORT "ERROR IN INTERACTIVEXREF COMP";
END;
```

```
%%%
```

```
%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%          PRINTBINDINFO          %%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE PRINTBINDCOMP;
BEGIN  TASK T;
  COMPILE SYSTEM/PRINTBINDINFO ALGOL[T]  LIBRARY;
    ALGOL OPTION = (BDBASE,FAULT,FILE,DSED);
    ALGOL FILE TAPE(TITLE = SYMBOL/PRINTBINDINFO);
    ALGOL FILE NEWTAPE(TITLE = SYMBOL/PRINTBINDINFO);
  ALGOL DATA
$ MERGE SEQERR NEWSEQERR
$ SET LINEINFO  NEW
?
  IF T ISNT COMPILEDOK THEN
     ABORT "ERROR IN PRINTBINDINFO COMP";
END;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%          PRINTCOPY          %%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE PRINTCOPYCOMP;
BEGIN  TASK T;
  COMPILE SYSTEM/PRINTCOPY  ALGOL[T]  LIBRARY;
    ALGOL OPTION = (BDBASE,FAULT,FILE,DSED);
    ALGOL FILE TAPE(TITLE = SYMBOL/PRINTCOPY);
    ALGOL FILE NEWTAPE(TITLE = SYMBOL/PRINTCOPY);
    PRIORITY = 98;
  ALGOL DATA
$ MERGE SEQERR NEWSEQERR
$ SET LINEINFO  NEW
?
  IF T ISNT COMPILEDOK THEN
    ABORT "ERROR IN PRINTCOPY COMP";
END;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%          RESEQBASIC          %%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE RESEQCOMP;
BEGIN  TASK T;
  COMPILE SYSTEM/RESEQBASIC  ALGOL[T] LIBRARY;
    ALGOL OPTION = (BDBASE, FAULT,FILE,DSED);
    ALGOL FILE TAPE(TITLE = SYMBOL/RESEQBASIC);
    ALGOL FILE NEWTAPE(TITLE = SYMBOL/RESEQBASIC);
  ALGOL DATA
$ MERGE SEQERR NEWSEQERR
$ SET LINEINFO  NEW
?
  IF T ISNT COMPILEDOK THEN
     ABORT "ERROR IN RESEQBASIC COMP";
END;
```

```
%%%
```

```
%%%
```

```
%%%
```

```
%%%
```

```
%%%
```

```
%%%
```

```
%%%
```

```
%%%
```

```
%%%
```

```
%%%
```

```
%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE LCOBLIBCOMP;
BEGIN  TASK T;
  COMPILE SYSTEM/LCOBLIBRARY ALGOL[T] LIBRARY;
    ALGOL OPTION = (BDBASE,FAULT,FILE,DSED);
    ALGOL FILE TAPE(TITLE = SYMBOL/LCOBLIBRARY);
    ALGOL FILE NEWTAPE(TITLE = SYMBOL/LCOBLIBRARY);
  ALGOL DATA
$ MERGE SEQERR NEWSEQERR
$ SET NEW LINEINFO
?
  IF T IS COMPILEDOK
     THEN WAIT ("PP SYSTEM/LCOBLIBRARY : TRANSPARENT",OK)
     ELSE ABORT "BAD LCOBLIBRARY COMP";
END;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE COBOL74COMP;
BEGIN  TASK T;
  COMPILE SYSTEM/COBOL74 ALGOL[T] LIBRARY;
    ALGOL OPTION = (BDBASE,FAULT,FILE,DSED);
    ALGOL FILE TAPE(TITLE = SYMBOL/COBOL74);
    ALGOL FILE NEWTAPE(TITLE = SYMBOL/COBOL74);
  ALGOL DATA
$ MERGE SEQERR NEWSEQERR
$ SET NEW LINEINFO
?
  IF T ISNT COMPILEDOK THEN
     ABORT "BAD COBOL74 COMP";
END;
```

```
%%%
```

```
%%%%%%%%%%%%%%% COBOL74 COMPILATION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 COBOL74COMP;

%%%%%%%%%%%%%%% LCOBOL COMPILATIONS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 LCOBOLCOMP;
 LCOBDOWNCOMP;
 LCOBLIBCOMP;

%%%%%%%%%%%%%%%%%%%%%%% FORTRAN77 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 FORTRAN77COMP;

%%%%%%%%%%%%%%%%%%%%%%% CLEAN UP %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

WAIT ("OK WHEN ALL COMPILATIONS COMPLETED",OK);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                           %
%                  FILES PRESENCE VERIFICATION                              %
%                                                                           %
%        STATEMENTS LIKE THE FOLLOWING STATEMENTS SHOULD BE                 %
%           USED FOR ALL SOFTWARE SOURCE AND CODE FILES                     %
%                                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

IF FILE SYMBOL/BINDER  ISNT RESIDENT
   THEN DISPLAY"SYMBOL/BINDER ISNT RESIDENT";
IF FILE SYSTEM/BINDER ISNT RESIDENT
   THEN DISPLAY"SYSTEM/BINDER ISNT RESIDENT";

? END JOB;
```

THIS PAGE INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES

# 3. PROCEDURE EXPLANATION

## General Explanation

The preceding examples do not contain examples of patches to software. SYSTEM/PATCH may be used as a preprocessor for patches. If SYSTEM/PATCH is used, the dollar cards in the compiler DATA deck should be included in the SYSTEM/PATCH input. The SYSTEM/PATCH output should be used as the compiler input. For a more detailed explanation of SYSTEM/PATCH and software patches, refer to Chapter 10 of the B 7000/B 6000 Series System Software Operational Guide, Volume 1, form 5011661.

After recompiling a compiler, "MC <compiler name>" is entered at the ODT to designate the code file as a compiler code file. The newly generated compiler is used as a compiler for successive compilations.

SYSTEM/CANDE, SYSTEM/HARDCOPY, SYSTEM/RJE, and SYSTEM/SUSPENDER are designated as "control" programs with the CP ODT command. A control program is never scheduled when entered into the mix and is never programmatically suspended.

**PASS ONE DETAILS**

SYSTEM/COMPBIND must be present, as it is automatically bound into ALGOL. COMPBIND is generated by compiling SYMBOL/BINDER with the COMPBIND compile-time option SET.

ALGOL is compiled three times to ensure integrity. An example of the necessity of this duplication may be seen in the procedure for making a change in the code for an IF statement. Because IF statements are used in the IF expression procedure, multiple compilations are required to effect a change. The first compilation gets the new lines of code into the compiler. The second compilation generates the new code for the change. The third compilation tests the changed code.

The UDSTRUCTURETABLE define in the USERSTRUCTURE program is used to make changes to the USERDATAFILE entry structure. Refer to Chapter 9, MAKEUSER, in the B 5000/B 6000/B 7000 Series System Software Operational Guide, Volume 2, form 5011679, for a discussion of the UDSTRUCTURETABLE.

The DCALGOL compiler is generated by compiling SYMBOL/ALGOL with the DCALGOL compile-time option SET.

BASEINTRINSICS are compiled with ALGOL. ESPOLINTRINSICS are compiled with ESPOL. These files are bound together with the INTRINSICS bind-time option SET to generate new SYSTEM/INTRINSICS. These new intrinsics should be activated using the SI (or CI) ODT command before generating the remainder of the software.

The system support libraries are compiled with NEWP. After successful compilation, a system support library must be PPed through the ODT as follows:

```
PP <library title> : TRANSPARENT
```

This extremely important action establishes the library as privileged transparent, which guarantees that a privileged program invoking the support library remains privileged during the invocation, and that a non-privileged program remains non-privileged.

The new support libraries should then be activated using the SL ODT command. Only GENERALSUPPORT is referenced when generating system software.

SYSTEM/CONTROLLER and SYSTEM/WFL are compiled referencing the new MCPSYMBOLIC. Some MCP code is INCLUDEd from MCPSYMBOLIC during these compilations.

The WFL compilation generates the files CCSTRINGCONV, CCVARIABLEPPB, and CCSTRINGFUNCTION. These files contain MCP procedures that are bound with the MCPHOST file.

WFL (and its associated files), JOBFORMATTER, and CONTROLLER are bound with the MCPHOST to generate the new MCP.

**PASS TWO DETAILS**

COMPBIND is compiled with the new ALGOL compiler. ALGOL must be recompiled to incorporate any changes in COMPBIND.

The remainder of the compilers may then be compiled. The PL/I and COBOL compilations require extensive system resources and should be performed separately. None of the succeeding compilations needs the new PL/I and COBOL compilers.

The remainder of the system software then is compiled.

## NOTES

LOGANALYZER and LOGGER require the new JOBFORMATTER symbolic as FILE JOBFORMATTER.

The LCOBOL and COBOL74 compilations need not be performed unless these compilers are desired at the site.

# Intrinsics Generation

The following sections provide the necessary information for compiling and binding intrinsics. The intrinsic file contains the standard system intrinsics such as SIN, SQRT, and the I/O formatting routines and can also contain user-written intrinsics, referred to as installation intrinsics. Intrinsics must be written in ALGOL, DCALGOL, ESPOL, FORTRAN, or COBOL but may be referenced by programs written in ALGOL, COBOL, DCALGOL, ESPOL, NEWP, FORTRAN, or PL/I. Additional information concerning intrinsics may be found in the B 7000/B 6000 BINDER Reference Manual, form 5001456; in Chapter 9 of the B 7000/B 6000 Series System Software Operational Guide, Volume 1, form 5011661; and in the appropriate language manuals.

## COMPILING INTRINSICS

The INTRINSICS compiler dollar option must be SET when intrinsics are compiled. In the following example, two installation intrinsics, P and Q, are created. The object code for both intrinsics is put in the file INTR/PROC because the ALGOL LIBRARY option is SET. If LIBRARY is not SET, the object code for procedure P is put into file INTR/P, and the object code for procedure Q is put into file INTR/Q.

### Example

```
<i>BEGIN JOB COMP/INSTALLATION;
COMPILE INTR/PROCS ALGOL LIBRARY;
ALGOL DATA
$SET INTRINSICS LIBRARY
REAL PROCEDURE P(A);
   VALUE A;
   REAL A;
   P:=A+1.2;
REAL PROCEDURE Q(B);
   REAL B;
   Q:=2*B;
<i>END JOB
```

## BINDING INTRINSICS

### Standard Intrinsics

The Pass One Example gives an example of compiling and binding standard intrinsics. A deck to bind standard intrinsics contains:

1. An option card with the INTRINSICS compiler option SET. This indicates an intrinsic file. The HOST BINDER option is automatically RESET and no host file is expected.

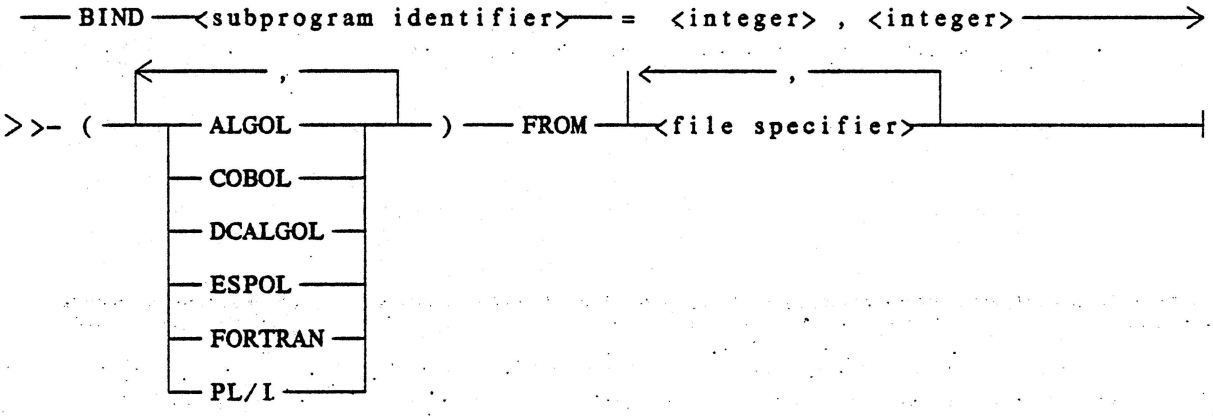
2. A <bind statement> that specifies the source file or files for standard system intrinsics. The "BIND=" statement causes the BINDER to look for all standard intrinsics whose names and intrinsic numbers are tabulated within the BINDER. An EXTERNAL statement must be used in order to leave out a standard intrinsic.

## Installation Intrinsics

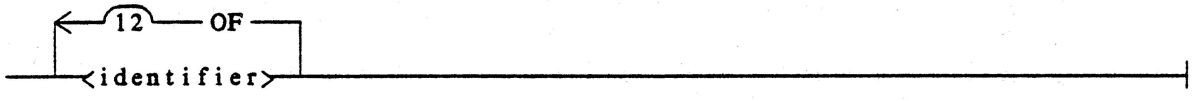
Installation intrinsics are bound using the <installation intrinsics bind statement>.

### Syntax

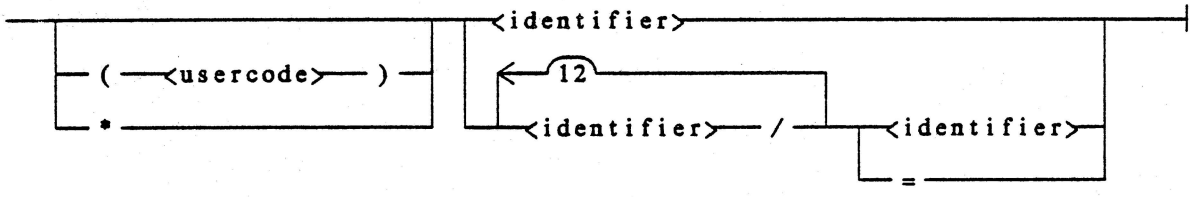
<installation intrinsics bind statement>



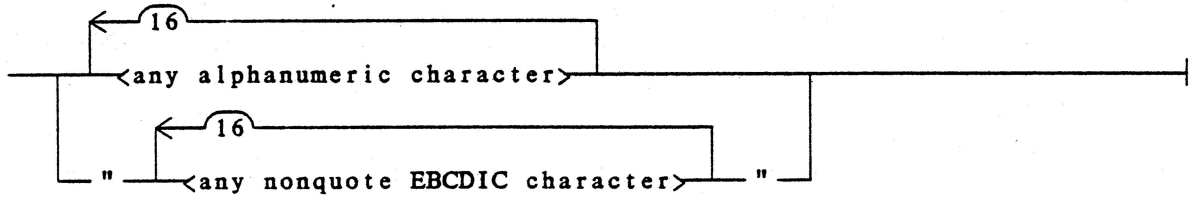

<subprogram identifier>




<file specifier>




<identifier>



**Semantics**

The intrinsic <bind statement> contains an Intrinsic Number Pair and a language identifier for each intrinsic. An Intrinsic Number Pair consists of two <integer>s separated by a comma. The first <integer> specifies an installation number; the second specifies an intrinsic number. An installation number may range in value between 0 and 2047; however, numbers 0 thru 99 are reserved for system intrinsics. An intrinsic number may range in value between 0 and 8191. No two intrinsics within an intrinsic file may be given the same Intrinsic Number Pair or <subprogram identifier>.

Some system intrinsics are part of the MCP symbolic. ALGOL or FORTRAN programs accessing installation intrinsics must be compiled with the INSTALLATION compiler option SET. COBOL and PL/I programs can automatically access installation intrinsics.

The language identifier allows the user to list the compilers authorized to reference the intrinsic. The referencing language is not necessarily the same as that in which the intrinsic is written. The DCALGOL language identifier allows a specified intrinsic to be accessed by the DMALGOL compiler and the DCALGOL compiler.

A new intrinsic file containing installation intrinsics is created using a deck similar to that used for standard intrinsics. A deck to bind installation intrinsics contains:

1. An option card with the INTRINSICS compiler option SET.
2. A <bind statement> specifying the source file or files for standard intrinsics.
3. A <bind statement> for each installation intrinsic.

Installation intrinsics may be added to an intrinsic file by specifying that intrinsic file as the BINDER host file. The BINDER recognizes that the HOST file is an intrinsic file, so the HOST and INTRINSICS BINDER options do not need to be specified. Such a procedure must include:

1. A BINDER file equation specifying the BINDER HOST file.
2. A <bind statement> for each installation intrinsic.

**Example**

```
<i>BEGIN JOB BIND/INTR;
BIND INTR BINDER LIBRARY;
BINDER FILE HOST (TITLE=SYSTEM/INTRINSICS,KIND=DISK);
BINDER DATA
$SET LINEINFO RESET SEGS
 BIND P = 277,1 (ALGOL,FORTRAN) FROM INTR/PROCS;
 BIND Q = 277,2 (ALGOL) FROM INTR/PROCS;
 STOP;
<i>END JOB
```

Once an intrinsic is bound into an intrinsic file, the BINDER does not allow a user to alter the intrinsic number, type of subprogram, or parameter type by performing replacement binding. A new intrinsic file must be bound in order to modify any of these items in an installation intrinsic. The BINDER internal tables must be updated, and a new intrinsic file must be created in order to modify a standard system intrinsic. An installation intrinsic should be used instead of modifying a standard system intrinsic.

**NOTE**

Installation intrinsics will be deimplemented on the 3.4 release. Existing installation intrinsics should be converted to use the LIBRARY mechanism.

## SOURCENDL

SOURCENDL requires patching by the user. SOURCENDL contains line controls, request sets, and examples of DCP, Modem, Line, Terminal, Station, and File declarations. A large amount of the code can be eliminated by omitting those parts of SOURCENDL which are not applicable to the Data Comm network. The remaining relevant part of the SOURCENDL can be modified to fit the network of the site. This software item need not be patched until the rest of system software is running, and the user is ready to initialize Data Comm.

**Example**

The following deck may be used to compile SOURCENDL:

```
<i>BEGIN JOB SOURCENDL/GENERATE;
 COMPILE DUMMY/NDL  NDL LIBRARY;
   NDL OPTION = (BDBASE,FAULT,FILE,DSED);
   NDL FILE TAPE(TITLE = SYMBOL/SOURCENDL);
   NDL FILE NEWTAPE(TITLE = SYMBOL/NEW/SOURCENDL);
   NDL DATA
$ MERGE NEW
<i>END JOB
```

## Table Generating Programs

The table-generating programs should not be patched; these procedures are used only when TIO releases a patch to these programs.

The NEW compiler option should always be set when these programs are compiled.

The patches produced by the table-generating programs must be merged with any other patches to the particular software item. The combined patches are used as compiler input when recompiling the particular software item. If no other patches to the software item are present, the patches produced by the table generating programs may be INCLUDEd after the last dollar card in the CARD input file.

### TABLEGEN

TABLEGEN is used to create many of the defines, value arrays, and switches in the ESPOL compiler. Among the tables generated by TABLEGEN are error messages and defines, operator mnemonics, ELCLASS defines, INFO and ADDL tables, and several switches for ELCLASS handling.

#### Example

A new copy of the TABLEGEN symbolic, including the necessary changes, is generated and compiled with the following deck:

```
<i>BEGIN JOB TABLEGEN/COMP;
 COMPILE SYS/TABLEGEN ALGOL LIBRARY;
   ALGOL OPTION = (BDBASE,FAULT,FILE,DSED);
   ALGOL FILE TAPE(TITLE = SYMBOL/TABLEGEN);
   ALGOL FILE NEWTAPE(TITLE = SYMBOL/NEW/TABLEGEN);
   ALGOL DATA
$ MERGE SEQERR NEWSEQERR
$ SET NEW LINEINFO
   <the TABLEGEN changes go here>
<i>END JOB
```

TABLEGEN must be run with an input parameter of 1; this parameter identifies the ESPOL compiler as the compiler whose tables' patches are generated.

TABLEGEN also requires the following input files:

| | |
|---|---|
| MYSYM | The copy of the TABLEGEN including the necessary changes. These are the data for the TABLEGEN run. |
| TAPE | The input compiler symbolic (that is, SYMBOL/ESPOL). |

The following output file is produced:

| | |
|---|---|
| PATCH | The patch for the compiler symbolic containing all the appropriate defines, value arrays, and switches regenerated using the new input data from the file MYSYM. The default title for this file is PATCH/<compiler name>. |

**Example**

A patch, ESPOLPATCH, for the ESPOL symbolic can be created using the following WFL deck:

```
<i>BEGIN JOB ESPOL/TABLE/PATCH;
 RUN SYSTEM/TABLEGEN(1);
 FILE MYSYM (TITLE=SYMBOL/NEWTABLEGEN);
 FILE TAPE  (TITLE=SYMBOL/ESPOL);
 FILE PATCH (TITLE=ESPOLPATCH);
<i>END JOB
```

**ALGOLTABLEGEN**

ALGOLTABLEGEN is used to create certain defines, arrays, and value arrays for the ALGOL compiler.

**Example**

A new copy of the ALGOLTABLEGEN symbolic, including necessary changes, is generated and compiled with the following deck:

```
<i>BEGIN JOB ALGOLTABLE/COMP;
 COMPILE SYSTEM/ALGOLTABLEGEN ALGOL LIBRARY;
   ALGOL FILE TAPE (TITLE=SYMBOL/ALGOLTABLEGEN);
   ALGOL FILE NEWTAPE (TITLE=SYMBOL/NEW/ALGOLTABLEGEN);
   ALGOL DATA
$MERGE SEQERR NEWSEQERR
$SET NEW LINEINFO
   <the ALGOLTABLEGEN changes go here>
<i>END JOB
```

ALGOLTABLEGEN should be run to generate the patch for the ALGOL compiler. The following input files are used:

| | |
|---|---|
| SYMBOL | The ALGOLTABLEGEN symbolic. The default file title is SYMBOL/ALGOLTABLEGEN. |
| CARD | The patches to the current input data. |

The following output files may be produced:

| | |
|---|---|
| PATCH | The patch to the ALGOL compiler created by running ALGOLTABLEGEN. The default file title is PATCH/ALGOL. |
| LINE | A line printer file that is created if errors are encountered. |

**Example**

The following deck creates an ALGOL patch contained in the file ALGOL/PATCH:

```
<i>BEGIN JOB ALTABLE;
   RUN SYSTEM/ALGOLTABLEGEN;
      FILE SYMBOL(TITLE=SYMBOL/NEW/ATTABLEGEN);
      FILE PATCH(TITLE=PATCH/ALGOL);
<i>END JOB
```

## COBOLTABLEGEN

COBOLTABLEGEN generates the text required for the declaration of reserved word defines and arrays in the COBOL compiler. Like TABLEGEN, COBOLTABLEGEN obtains input from its own symbol file.

### Example

COBOLTABLEGEN is compiled with the following deck:

```
<i>BEGIN JOB COBOLTABLE/COMP;
 COMPILE SYS/COBOLTABLEGEN ALGOL LIBRARY;
   ALGOL OPTION = (BDBASE,FAULT,FILE,DSED);
   ALGOL FILE TAPE(TITLE = SYMBOL/COBOLTABLEGEN);
   ALGOL FILE NEWTAPE(TITLE = SYMBOL/NEW/COBOLTABLEGEN);
   ALGOL DATA
$ MERGE SEQERR NEWSEQERR
$ SET NEW LINEINFO
   <the COBOLTABLEGEN patches go here>
<i>END JOB
```

COBOLTABLEGEN must be run with a string input parameter. The following options are available:

| | |
|---|---|
| LISTUFF | When SET, a listing of the generated defines is produced. |
| LISTACK | When SET, a listing of RSWD stackheads is produced. |

The following input file is required:

| | |
|---|---|
| GEN | The new SYMBOL/COBOLTABLEGEN with the necessary changes. |

The following output file is generated:

| | |
|---|---|
| PATCH | The patch for the compiler symbolic. The default title is PATCH/COBOLTABLEGEN. |

### Example

The following example generates a patch titled PATCHES/COBOLWORDS for the COBOL compiler:

```
<i>RUN SYSTEM/COBOLTABLEGEN("LISTUFF");
   FILE PATCH (TITLE=PATCHES/COBOLWORDS);
```

## ATTABLEGEN

ATTABLEGEN generates patches to the attribute table.

### Example

The following deck recompiles ATTABLEGEN:

```
<i>BEGIN JOB ATT/COMP;
 COMPILE SYSTEM/ATTABLEGEN  ALGOL LIBRARY;
   ALGOL OPTION = (BDBASE,FAULT,FILE,DSED);
   ALGOL FILE TAPE(TITLE = SYMBOL/ATTABLEGEN);
   ALGOL FILE NEWTAPE(TITLE = SYMBOL/NEW/ATTABLEGEN);
   ALGOL DATA
$ MERGE SEQERR NEWSEQERR
$ SET NEW LINEINFO
    <the ATTABLEGEN patches go here>
<i>END JOB
```

ATTABLEGEN requires a string input parameter. The following options are available:

| | |
|---|---|
| LIST | Causes a listing of the input and the generated output deck on the file LINE. |
| PATCH | Generates and locks the patch file. |

The following input file is used:

| | |
|---|---|
| TAPE | The ATTABLEGEN symbolic. |

The following output files may be generated:

| | |
|---|---|
| LINE | The line-printer file containing the input and output listing. |
| PATCH | The attribute table patches. The default title is PATCH/MCP/ATTABLES. |

### Example

A patch, PATCHES/ATTRIBUTES, is generated with the following deck:

```
<i>BEGIN JOB AT/GEN;
   RUN SYSTEM/ATTABLEGEN ("PATCH LIST");
     FILE PATCH (TITLE=PATCHES/ATTRIBUTES);
<i>END JOB
```

## CCTABLEGEN

The CCTABLEGEN program generates patches for reserved word tables used by WFL and the CONTROLLER.

Patches may be generated for three tables:

| | |
|---|---|
| WFL1 | Words needed by "old" WFL. |
| WFL2 | Words used by 2.9 WFL. |
| CON | WFL words that are recognized by the CONTROLLER. |

### Example

The following deck may be used to recompile CCTABLEGEN:

```
<i>BEGIN JOB CCT/COMP;
 COMPILE SYSTEM/CCTABLEGEN  ALGOL LIBRARY;
   ALGOL OPTION = (BDBASE,FAULT,FILE,DSED);
   ALGOL FILE TAPE(TITLE = SYMBOL/CCTABLEGEN);
   ALGOL FILE NEWTAPE(TITLE = SYMBOL/NEW/CCTABLEGEN);
   ALGOL DATA
$ MERGE SEQERR NEWSEQERR
$ SET NEW LINEINFO
    <the CCTABLEGEN patches go here>
<i>END JOB
```

CCTABLEGEN requires a string input parameter. The possible parameters for CCTABLEGEN are:

| | |
|---|---|
| LIST | Causes a printer listing of the input and output on the file LINE. |
| PATCH | Generates and locks the patch decks. |

The following output files are generated:

| | |
|---|---|
| PATCHCON | The patches for the CONTROLLER. The default title is PATCH/CON/CCTABLES. |
| PATCHWFL | The patches for WFL. The default title is PATCH/WFL/CCTABLES. |

### Example

```
<i>RUN SYSTEM/CCTABLEGEN ("LIST PATCH");
```

## LTTABLEGEN

LTTABLEGEN is a utility program that generates SYSTEM/TRAINTABLES. A full description of LTTAGLEGEN is found in Chapter 8 of the B 7000/B 6000 Series System Software Operational Guide, Volume 2, form 5011679.

The following deck may be used to recompile LTTABLEGEN.

```
<i>BEGIN JOB LTT/COMP;
 COMPILE SYSTEM/LTTABLEGEN  ALGOL  LIBRARY;
   ALGOL OPTION = (BDBASE,FAULT,FILE,DSED);
   ALGOL FILE TAPE(TITLE = SYMBOL/LTTABLEGEN);
   ALGOL FILE NEWTAPE(TITLE = SYMBOL/NEW/LTTABLEGEN);
   ALGOL DATA
$ MERGE SEQERR NEWSEQERR
$ SET LINEINFO NEW
    <the LTTABLEGEN patches go here>
<i>END JOB
```

## RLTABLEGEN

RLTABLEGEN is a program that constructs value arrays to allow nonstandard tape labels. A full description of RLTABLEGEN is found in Chapter 12 of the B 5000/B 6000/B 7000 Series System Software Operational Guide, Volume 2 form 5011679.

The following deck may be used to recompile RLTABLEGEN.

```
<i>BEGIN JOB RLT/COMP;
 COMPILE SYSTEM/RLTABLEGEN ALGOL  LIBRARY;
   ALGOL OPTION = (BDBASE,FAULT,FILE,DSED);
   ALGOL FILE TAPE(TITLE = SYMBOL/RLTABLEGEN);
   ALGOL FILE NEWTAPE(TITLE = SYMBOL/NEW/RLTABLEGEN);
   ALGOL DATA
$ MERGE SEQERR NEWSEQERR
$ SET LINEINFO  NEW
    <the RLTABLEGEN patches go here>
<i> END JOB
```

## SCTABLEGEN

SCTABLEGEN generates a table of recognized commands for the CONTROLLER.

### Example

The following deck recompiles SCTABLEGEN:

```
<i> BEGIN JOB SCT/COMP;
 COMPILE SYSTEM/SCTABLEGEN  ALGOL LIBRARY;
   ALGOL OPTION = (BDBASE,FAULT,FILE,DSED);
   ALGOL FILE TAPE(TITLE = SYMBOL/SCTABLEGEN);
   ALGOL FILE NEWTAPE(TITLE = SYMBOL/NEW/SCTABLEGEN);
   ALGOL DATA
$ MERGE SEQERR NEWSEQERR
$ SET LINEINFO  NEW
    <the SCTABLEGEN patches go here>
<i>END JOB
```

SCTABLEGEN requires a string parameter. The following options are available:

| | |
|---|---|
| LIST | Causes a printer listing of the input and generated output deck on the file LINE. |
| PUNCH | Generates and locks the patch file in file LINE. |

The following input file is used:

| | |
|---|---|
| TAPE | The CONTROLLER symbolic. |

The following output file is generated:

| | |
|---|---|
| LINE | The output patch listing. If the PUNCH option is used, this file has a default KIND of PUNCH. To make a disk file containing the patch, LINE should be label-equated to KIND=DISK, and the PUNCH option should be used. |

**Example**

A patch for the CONTROLLER symbolic could be created and stored in a disk file called TABLES with the following deck:

```
<i>BEGIN JOB SC/TABLE;
 RUN SYSTEM/SCTABLEGEN("PUNCH");
   FILE LINE(KIND=DISK,TITLE=TABLES);
<i>END JOB
```

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

## File Dependency

Most system software is compiled with ALGOL, with the following exceptions:

1. The MCP is compiled with the NEWP compiler.

2. The following software is compiled with ESPOL: PLINTRINSICS, ESPOLINTRINSICS, LOADER, and UTILOADER.

3. The following software is compiled with DCALGOL: DCALGOLINTRINSICS, CONTROLLER, WFL, JOBFORMATTER, BACKUP, CANDE, CONFIGURATOR, DCSTATUS, DIAGNOSTICSMCS, FILECOPY, FILEDATA, HARDCOPY, HOSTSERVICES, IADMAPPER, LISTVOLUMELIB, LOGANALYZER, LOGGER, MAKEUSER, RJE, STATSTARTER, STATSTOPPER, and SUSPENDER.

4. UDSTRUCTURETABLE is compiled with the USERSTRUCTURE compiler.

5. SOURCENDL is compiled with SYSTEM/NDL.

6. The following software references the MCPSYMBOLIC file: CONTROLLER, WFL, and LISTVOLUMELIB.

7. SYSTEM/BINDER is used when generating the following software: ALGOL and FORTRAN(COMPBIND for AUTOBIND), INTRINSICS, MCPHOST, and NDL (DCPPROGEN is bound in).

8. The following software references the JOBFORMATTER symbolic: LOGGER and LOGANALYZER.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# 4. SOFTWARE GENERATION OPTIONS

Various compile-time options affect the generated software. Many of the important options are described in this section. The default option values are indicated in parentheses following the option names.

All system software should be compiled with the LINEINFO compile-time option SET. This option provides source-line identification information when an error occurs. The sequence- or line-number information and its relationship to the generated code are saved in a table at the end of the code file.

## ALGOL Compile-Time Options

### ALGOL, DCALGOL AND DMALGOL

When an ALGOL compiler is compiled, either ALGOL, DCALGOL, or DMALGOL must be SET. These options determine which ALGOL compiler is generated. Refer to the DMSII documentation for options relating to DMALGOL.

### B7000CODE (RESET)

Refer to Section 5, B 7700 SOFTWARE GENERATION.

### CTPROC (RESET)

The CTPROC option is SET when compiling ALGOL if the compile-time facility is desired. (See Appendix H in the B 5000/B 6000/B 7000 Series ALGOL Reference Manual, form 5011760, for an explanation of the compile-time facility.)

## BINDER Compile-Time Option

### COMPBIND (RESET)

The COMPBIND option prepares BINDER information for the compilers. The BINDER code is compiled at level 3, and external files are added for compiler interface. The ALGOL and FORTRAN compilers bind in the COMPBIND information to enable automatic binding.

## COBOL Compile-Time Options

### ANSI74 (RESET)

When the COBOL compiler is compiled with the ANSI74 option SET, the ANSI74 compiler option is explicitly SET.

### B2500 (RESET)

When the COBOL compiler is compiled with the B2500 option SET, the B2500 compiler option is explicitly SET.

**B7700CODE (RESET)**

Refer to Section 5, B 7000 SOFTWARE GENERATION.

## ESPOL Compile-Time Options

**BCL (SET)**

The BCL option controls character interpretation and should always be SET.

**B7700CODE (RESET)**

Refer to Section 5, B 7000 SOFTWARE GENERATION.

## FORTRAN Compile-Time Options

**BIGOPTJOB (RESET)**

The BIGOPTJOB option permits optimization of large FORTRAN programs. BIGOPTJOB increases the memory requirements of the FORTRAN compiler. This option can only be SET if the OPTLSS1 compiler generation option is RESET.

**B7700CODE (RESET)**

Refer to Section 5, B 7000 SOFTWARE GENERATION.

**FASTGRAPH (SET)**

When FASTGRAPH is SET, graphs are printed faster by ignoring end-of-page sensing on the line printer. When the FASTGRAPH option is RESET, end-of-page sensing occurs, and the graphs are not drawn across a fanfold in the paper. FASTGRAPH must be SET if OPTLSS1 is RESET.

**GETFIRSTDOLLARCARDFROMSPO (RESET)**

When GETFIRSTDOLLARCARDFROMSPO is SET, the compiler requires the first compiler control card (dollar card) to be input through the ODT. This option should normally be RESET.

**MODEL1 (RESET)**

When MODEL1 is SET, FORTRAN does not perform indexable value calls on arrays. The option should be RESET unless the compiler is to be used on a Model 1 B 6700 processor.

**MULTI (SET)**

The MULTI option allows batch mode operation. Batch facilities are not available if this option is RESET. (See the discussion of batch facilities in Chapter 18 of the B 7000/B 6000 Series FORTRAN Reference Manual.)

**NOSEGS (RESET)**

When NOSEGS is SET, the compiler option SEGS is RESET by default.

**OPTLSS1 (RESET)**

When OPTLSS1 is SET, the FORTRAN compiler control option OPT may only be SET to values less than 1. All OPT values are valid if OPTLSS1 is RESET. A smaller compiler results if OPTLSS1 is SET.

**SETOWNARRAYS (RESET)**

When SETOWNARRAYS is SET, the compiler option OWNARRAYS is SET by default. OWNARRAYS normally defaults to RESET.

**SINGLEBYDEFAULT (SET)**

When SINGLEBYDEFAULT is SET, the compiler option SINGLE is SET by default, causing single-spaced FORTRAN listings. When SINGLEBYDEFAULT is RESET, the listings are double-spaced.

## GENSUPP Compile-Time Options

**MINUSZEROFORBLANKFIELD (RESET)**

A blank string input with an I or J format is normally interpreted as a zero. When MINUSZEROFORBLANKFIELD is SET at sequence number 1600 at ESPOLINTRINSICS compile time, the blank string is translated to a minus zero. This translation allows differentiation between an entered zero and a blank and only affects input operations.

## MCP Compile-Time Options

**ACTIVETIME (RESET)**

When ACTIVETIME is SET, the MCP enters the active time of the stack in the system log at EOJ time. Processor and I/O times are always logged.

**B7000 (RESET)**

When B7000 is SET, the MCP produced is applicable to the B 7000 series systems.

**DEBUG (RESET)**

When DEBUG is SET, the MCP contains the debugging procedure CONDITIONALDUMPER, which stores pertinent information in a trace table each time certain MCP functions are executed. Because of the overhead introduced, this option is usually SET only for system troubleshooting.

**DEBUGMCP (RESET)**

This option causes the following debugging options to be SET: DIAGNOSTICS, TRACE7000, READLOCK, and SWAPTRACE.

**DIAGNOSTICS (RESET)**

When DIAGNOSTICS is SET, extra debugging code is included in the MCP. Because of the additional overhead caused by this option, it is only SET for troubleshooting purposes.

**EXPERIMENTAL (RESET)**

When EXPERIMENTAL is SET, an internal version of the MCP is set which allows extra command and configuration syntax which is dangerous to normal operation. This option should only be used by plant personnel.

**INTERNAL (RESET)**

When INTERNAL is SET, a "diagnostic" version of the MCP is produced. The following options are implicitly SET: DIAGNOSTICS, READLOCK, SWAPTRACE, LOCKTRACE, EXPERIMENTAL, and MAKEHOST.

**LINEINFO (SET)**

When this option is SET, the MCP code file contains line-number information. Memory- and program-dump analyses use this information to associate MCP symbolic line numbers with RCWs in a stack.

**LOCKTRACE (RESET)**

When LOCKTRACE is SET, the directory lock-tracing code is included in the MCP. Inclusion of this code is necessary to utilize the LOCKTRACE run-time option.

**NODUMP (RESET)**

When NODUMP is SET, the resulting MCP contains no memory dump facility. Conditions that normally cause a memory dump generate either fatal or non-fatal ODT messages instead.

**OVERHEADCHARGED (RESET)**

When OVERHEADCHARGED is SET, the processor time incurred by processor-to-processor communication (ANSWER) is charged to the job stack in which the action occurred. This option is meaningful only for a multiple-processor system.

**NOTE**

This option will be deimplemented on the 3.4 release.

**PRESENCEBITCHARGED (RESET)**

Three processor time values are logged for a task: normal processor time, initial presence-bit processor time (INITPBIT), and other presence-bit processor time (OTHERPBIT). Presence-bit times are logged to the process that requests the operation, not necessarily to the process that owns the data. The effect of the PRESENCEBITCHARGED option is to add INITPBIT and OTHERPBIT times to the processor time.

**NOTE**

This option will be deimplemented on the 3.4 release.

**READLOCK (RESET)**

When READLOCK is SET, the NEWP intrinsic BUZZ inserts the stack number of the "owner", the program segment, and the PIR of the program segment into the lock as it is being locked.

**READLOCKTIMEOUT (SET)**

When READLOCKTIMEOUT is SET, the waiting time of the BUZZ and BUZZ47 intrinsics is monitored. As part of this monitoring, the locking SNR is inserted into a lock as it is BUZZed. The MCP procedure READLOCKTIMEOUT is called if a lock is unavailable on the first attempt to BUZZ it. Appropriate action is taken if ownership of the lock has not changed within a specified time limit.

**STATISTICSET (RESET)**

When STATISTICSET is SET, the MCP can monitor procedures. The frequency of procedure calls and time spent in the procedure are monitored.

**SORTSTAT (RESET)**

When SORTSTAT is SET, statistics pertaining to each sort are saved in a disk file titled SORT/STATISTIX. This information may be printed by SYSTEM/SORTSTAT.

**SWAPTRACE**

When SWAPTRACE is SET, debugging information from SWAPPER is inserted in a trace table.

**TAPEOP (RESET)**

When TAPEOP is SET, positioning of a tape retry is done by the hardware using a special 10. This option should be used only by plant personnel.

**USERDATATROUBLE (RESET)**

When USERDATATROUBLE is SET, conditions causing a fault or other problem while the MCP procedure USERDATA is processing the USERDATAFILE cause a program dump to be taken.

# PL/I Compile-Time Options

**B7700CODE**

Refer to Section 5, B 7000 SOFTWARE GENERATION.

**DUMP (RESET)**

A PL/I PROGRAMDUMP can be generated when the DUMP compiler generation option is SET for the PL/I compiler. If the DUMP control option is SET in the PL/I program, the PL/I PROGRAMDUMP is generated; otherwise, a standard program dump is generated. When the DUMP compiler generation option is RESET, a standard program dump is always generated.

# 5. B 7000 SOFTWARE GENERATION

## INTRODUCTION

This section describes procedures that are specific to the B 7000 series systems. Some of these procedures may appear in previous pages of this document. Where conflicts arise, installations with a B 7000 series system should use the syntax and options described in this section.

The following chapter describes procedures that permit the MCP and INTRINSICS to be compiled. The MCP is normally compiled either to change compile-time options or when changes are made to the MCP. The INTRINSICS should be compiled whenever the MCP is compiled.

The SYMBOL files required for the compilations should be copied and compared from the base-level symbolic tapes. The object files for ALGOL, BINDER, DCALGOL, ESPOL, NEWP, AND PATCH should be copied and compared from the current SYSTEM tape.

## FILE DEPENDENCIES

The following tables describe all programs on the appropriate SYMBOL and SYSTEST tapes that require updated program symbolics because of an INCLUDE:

| FILE NAME | REQUIRED DURING COMPILE OF: |
|---|---|
| MCPSYMBOLIC | CONTROLLER |
| | LISTVOLUMELIB |
| | WFL |
| JOBFORMATTER | LOGANALYZER |
| | LOGGER |
| | SYMTEST/UTIL/DKERRANAL |
| | SYMTEST/UTIL/LOGSTRIPPER |
| SCT | CONTROLLER |

## COMPILER COMPILE-TIME OPTIONS

### B7700CODE

When B7700CODE is SET, the compiler is allowed to generate code optimized for B 7700 or B 7800 CPMs. The compiler then generates optimized code when running on a B 7000 series system. This run-time determination can be overridden, either to force code optimization on non-B 7700/B 7800 systems or to suppress code optimization code on B 7700/B 7800 systems.

## MCP COMPILE-TIME OPTIONS

The MCP can be characteristically altered by SETting or RESETting the various compile-time options. These options affect portions of the symbolic to be compiled, the NEWP code file generated, or both.

Some (but not all) of the options are SET or RESET in the MCP symbolic near the beginning of the file. The states of these options should be examined prior to compilation to ensure that they meet the necessary requirements.

The following are the compile-time options that the user may SET for the MCP host. The default settings are shown in parentheses.

### B7800ONLY (RESET)

When B7800ONLY is SET, the resultant MCP is tailored to run on B 7700/B 7800 systems. An MCP compiled in this way does not run on other machine types, but does run faster and uses less memory on B 7700/B 7800 systems than an untailored MCP.

### FREEZEOUTTC (RESET)

When FREEZEOUTTC is SET, the resultant MCP does not have the code used to run a tightly-coupled system. An attempt to bring up a system in a tightly-coupled configuration using such an MCP causes an error message to be displayed and the system to initialize and function in a monolithic state.

# COMPILE DECK EXAMPLE

## Introduction

This section contains an example of a WFL job deck that causes the MCP and MAD software to be compiled. The deck processes SYSTEM/PATCH and the related compiles sequentially for each program to permit its use on a system with limited resources. (Many other options exist in WFL which could be used to permit the simultaneous compilation of several programs once the MCP is completed.)

In the deck, the MCP host is compiled first. The file NEWSYMBOL/MCP produced during this compilation is required to compile the remaining programs bound into the MCP.

The compilation of SCTABLEGEN causes the file NEWSYMBOL/SCTABLEGEN to be produced. This file is required during compilation of CONTROLLER; thus, SCTABLEGEN must precede CONTROLLER.

Successful completion of the job results in the creation of the file NEW/MCP. (The name was chosen so as not to conflict with the MCP file currently in use.)

Files created during the compilation and used later in the job are:

| FILE NAME | PRODUCED DURING COMPILE OF | FILE TITLE ACTUALLY USED |
|---|---|---|
| MCPSYMBOLIC | MCP host | NEWSYMBOL/MCP |
| JOBFORMATTER | JOBFORMATTER | NEWSYMBOL/JOBFORMATTER |
| SCT | SCTABLEGEN | NEWSYMBOL/SCTABLEGEN |

**Compilation Job Deck**

```
<I> BEGIN JOB MCP/COMPILES;

<I> DATA ALGOLSUPPLEMENT/PATCH
$. SET MARK COMPARE
$# COMPILER OPTIONS
$ SET MERGE
$ SET LIST STACK SINGLE XREF
$ SET LINEINFO
   insert patches or "$. PATCHDECK <filename>" card at this point

<I> DATA CONTROLLER/PATCH
$. SET MARK COMPARE
$# COMPILER OPTIONS
$ SET MERGE
$ SET LIST STACK SINGLE XREF
$ SET LINEINFO
   insert patches or "$. PATCHDECK <filename>" card at this point

<I> DATA B7X/ESPOL/PATCH
$. SET MARK COMPARE
$# COMPILER OPTIONS
$ SET MERGE LINEINFO CHECK
$ SET $ LIST LISTINCL FORMAT
$ SET NOXREFLIST STACK XREF
$ SET B7700CODE
   insert patches or "$. PATCHDECK <filename>" card at this point

<I> DATA JOBFORMATTER/PATCH
$. SET MARK COMPARE
$# COMPILER OPTIONS
$ SET MERGE
$ SET NEW
$ SET LIST STACK SINGLE XREF
$ SET LINEINFO
   insert patches or "$. PATCHDECK <filename>" card at this point

<I> DATA B7X/LOADER/PATCH
$. SET MARK COMPARE
$# COMPILER OPTIONS
$ SET MERGE LINEINFO CHECK
$ SET $ LIST LISTINCL FORMAT
$ SET NOXREFLIST STACK XREF
$ SET LISTO
$ RESET LINEINFO
$ SET POOL
   insert patches or "$. PATCHDECK <filename>" card at this point

<I> DATA MCP/PATCH
$. SET MARK COMPARE
$# COMPILER OPTIONS
$ SET MERGE LINEINFO
$ SET LIST STACK XREF
$ SET NEW
$ SET B7000 B7800ONLY
$ SET READLOCK READLOCKTIMEOUT
   insert patches or "$. PATCHDECK <filename>" card at this point
```

```
<I> DATA MDUTILITY/PATCH
$. SET MARK COMPARE
$# COMPILER OPTIONS
$ SET MERGE LINEINFO CHECK
$ SET $ LIST LISTINCL FORMAT
$ SET NOXREFLIST STACK XREF
   insert patches or "$. PATCHDECK <filename>" card at this point

<I> DATA MINILOADER/PATCH
$. SET MARK COMPARE
$# COMPILER OPTIONS
$ SET MERGE LINEINFO CHECK
$ SET $ LIST LISTINCL FORMAT
$ SET NOXREFLIST STACK XREF
$ SET LISTO
$ RESET LINEINFO
$ SET POOL
   insert patches or "$. PATCHDECK <filename>" card at this point

<I> DATA SCTABLEGEN/PATCH
$. SET MARK COMPARE
$# COMPILER OPTIONS
$ SET MERGE
$ SET NEW
$ SET LIST STACK SINGLE XREF
$ SET LINEINFO
   insert patches or "$. PATCHDECK <filename>" card at this point

<I> DATA WFL/PATCH
$. SET MARK COMPARE
$# COMPILER OPTIONS
$ SET MERGE
$ SET LIST STACK SINGLE XREF
$ SET LINEINFO LIBRARY
   insert patches or "$. PATCHDECK <filename>" card at this point

<I>
```

```
%%%%%%%%%%%%%%%% START OF WFL COMPILE DECK  %%%%%%%%%%%%%%%%%%%%%%

BOOLEAN
      BADCOMPILE,
      BADMCPCOMPILE,
      SKIPCONTROLLER;
TASK
      TCOPY,
      TMCPBIND;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE COMPILEIT (
      STRING SOURCENAME VALUE,
      STRING OBJNAME VALUE,
      STRING COMPILERNAME VALUE,
      BOOLEAN BADCOMP);
BEGIN
STRING
      COMPILEDNAME;
TASK
      TCOMP,
      TPATCH;

% MERGE THE PATCHES

RUN SYSTEM/PATCH [TPATCH];
  FILE CARD (TITLE = #SOURCENAME/PATCH);
  FILE PATCH (TITLE = PATCHES/#SOURCENAME);
  FILE TAPE (TITLE = SYMBOL/#SOURCENAME);

% CHECK FOR GOOD COMPLETION OF THE PATCH MERGE

IF TPATCH (VALUE) = 1 THEN
     DISPLAY SOURCENAME & " PATCHES OK"
   ELSE
     BEGIN
     DISPLAY "BAD " & SOURCENAME & " PATCH";
     BADCOMP := TRUE;
     GO XIT;
     END;

% SET UP THE NAME OF THE COMPILED PROGRAM

IF OBJNAME = "MCP" THEN
     COMPILEDNAME := "SYSTEM/MCPHOST"
   ELSE IF COMPILEDNAME = "INTR" THEN
     COMPILEDNAME := OBJNAME
   ELSE COMPILEDNAME := "SYSTEM/" & OBJNAME;
```

```
% COMPILE THE PROGRAM

COMPILE #COMPILEDNAME WITH #COMPILERNAME [TCOMP] TO LIBRARY;
  COMPILER FILE CARD (KIND = DISK, TITLE = PATCHES/#SOURCENAME);
  COMPILER FILE TAPE (TITLE = SYMBOL/#SOURCENAME);
  COMPILER FILE NEWTAPE (TITLE = NEWSYMBOL/#SOURCENAME);
  COMPILER FILE MCPSYMBOLIC (TITLE = NEWSYMBOL/MCP);
  COMPILER FILE SCT (TITLE = NEWSYMBOL/SCTABLEGEN);
  COMPILER FILE JOBFORMATTER(TITLE = NEWSYMBOL/JOBFORMATTER);

% CHECK FOR GOOD COMPLETION OF THE COMPILE

IF TCOMP IS COMPILEDOK THEN
    DISPLAY SOURCENAME & " COMPILED OK"
  ELSE
    BEGIN
    DISPLAY "BAD " & SOURCENAME & " COMPILE";
    BADCOMP := TRUE;
    END;

XIT:

END;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% START OF COMPILES

% MCP HOST

    CHANGE NEWSYMBOL/MCP TO OLDSYMBOL/MCP;
    COMPILEIT ("MCP","MCP","NEWP",BADMCPCOMPILE);
    IF BADMCPCOMPILE THEN ABORT "MCP HOST DID NOT COMPILE";

% MAKE SURE NEWSYMBOL WAS CLOSED

    IF FILE NEWSYMBOL/MCP ISNT RESIDENT THEN
        ABORT "NEWSYMBOL/MCP ISNT AVAILABLE";

% SCTABLEGEN

    COMPILEIT ("SCTABLEGEN","SCTABLEGEN","DCALGOL",SKIPCONTROLLER);

% ALGOLSUPPLEMENT

    COMPILEIT ("ALGOLSUPPLEMENT","ALGOLSUPPLEMENT","DCALGOL",
            SKIPMCPBIND);

% CONTROLLER

    IF SKIPCONTROLLER THEN
        BEGIN
        DISPLAY "CONTROLLER WILL NOT BE COMPILED";
        SKIPMCPBIND := TRUE;
        END
      ELSE
        COMPILEIT ("CONTROLLER","CONTROLLER","DCALGOL",SKIPMCPBIND);
```

```
% JOBFORMATTER

     COMPILEIT ("JOBFORMATTER","JOBFORMATTER","DCALGOL",SKIPMCPBIND);

% WFL

     COMPILEIT ("WFL","WFL","DCALGOL",SKIPMCPBIND);

% BIND THE MCP

     IF SKIPMCPBIND THEN
          DISPLAY "MCP BIND WILL NOT BE PERFORMED"
        ELSE
          BEGIN
          BIND NEW/MCP BINDER [TMCPBIND] LIBRARY;
            COMPILER DATA
$SET TOUGH
EXTERNAL RSANALYZER;
HOST IS SYSTEM/MCPHOST;
BIND CONTROLLER   FROM SYSTEM/CONTROLLER;
BIND JOBFORMATTER FROM SYSTEM/JOBFORMATTER;
BIND =            FROM SYSTEM/ALGOLSUPPLEMENT,
                       SYSTEM/WFL;

<I>
          IF TMCPBIND IS COMPILEDOK THEN
               DISPLAY "NEW/MCP CREATED"
             ELSE
               DISPLAY "BAD MCP BIND";
          END;

% ESPOL

     COMPILEIT ("B7X/ESPOL","B7X/ESPOL","ALGOL",BADCOMPILE);

% LOADER

     COMPILEIT ("B7X/LOADER","B7X/LOADER","ESPOL",BADCOMPILE);

% MINILOADER

     COMPILEIT ("MINILOADER","MINILOADER","ESPOL",BADCOMPILE);

% MDUTILITY

     COMPILEIT ("MDUTILITY","MDUTILITY","ALGOL",BADCOMPILE);

<I> END JOB
```

# 5. B 7000 SOFTWARE GENERATION

## INTRODUCTION

The following chapter describes procedures that permit the MCP and INTRINSICS to be compiled. The MCP is normally compiled either to change compile-time options or when changes are made to the MCP. The INSTRINICS should be compiled whenever the MCP is compiled.

The SYMBOL files required for the compilations should be copied and compared from the base-level symbolic tapes. The object files for ALGOL, BINDER, DCALGOL, ESPOL, NEWP, PATCH, and USERSTRUCTURE should be copied and compared from the current SYSTEM tape.

## FILE DEPENDENCIES

The following tables describe all programs on the appropriate SYMBOL1, SYMBOL2, SYMBOL3, and SYSTEST tapes which require updated program symbolics because of an INCLUDE:

| FILE NAME | REQUIRED DURING COMPILE OF: |
|---|---|
| MCPSYMBOLIC | CONTROLLER |
| | LISTVOLUMELIB |
| | WFL |
| JOBFORMATTER | LOGANALYZER |
| | LOGGER |
| | SYMTEST/UTIL/DKERRANAL |
| | SYMTEST/UTIL/LOGSTRIPPER |
| SCTABLEGEN | CONTROLLER |

## MCP COMPILE-TIME OPTIONS

The MCP can be characteristically altered by setting or resetting the various compile-time options. These options affect portions of the symbolic to be compiled, the NEWP code file generated, or both.

Some (but not all) of the options are set or reset in the MCP symbolic near the beginning of the file. The states of these options should be examined prior to compilation to ensure that they meet the necessary requirements.

The following are the compile-time options that may be set by the user for the MCP host. The default settings are shown in parentheses.

**DEBUG (RESET)**

When this option is set, the MCP contains the debugging procedure CONDITIONALDUMPER, which stores pertinent information in a trace table each time certain MCP functions are executed. Because of the overhead introduced, this option is usually set only for system troubleshooting.

**DEBUGMCP (RESET)**

This option causes the following debugging options to be set: DIAGNOSTICS, TRACE7000, READLOCK, and SWAPTRACE.

**DIAGNOSTICS (RESET)**

When this option is set, extra debugging code is included in the MCP. Because of the additional overhead caused by this option, it is usually set only for troubleshooting purposes.

**LINEINFO (SET)**

When this option is set, the MCP code file contains line-number information. Memory and program dump analysis use this information to associate MCP symbolic line numbers with RCWs in a stack.

**LOCKTRACE (RESET)**

When this option is set, debugging code is included to provide a trace of directory locks. This option is normally set only for troubleshooting.

**MTBF (SET)**

When this option is set, the resultant MCP performs collection and display of peripheral unit and channel-error statistics. The run-time option DIAGNOSTICS is meaningful only if this option is set.

**NODUMP (RESET)**

When this option is set, the resulting MCP contains no memory dump facility. Conditions that normally cause a memory dump generate either fatal or non-fatal ODT messages instead.

**OVERHEADCHARGED (RESET)**

When this option is set, the processor time incurred by processor-to-processor communication (ANSWER) is charged to the job stack in which the action occurred. This option is meaningful only for a multiple-processor system.

**PRESENCEBITCHARGED (RESET)**

When this option is set, the resultant MCP causes the processor time required to handle a presence-bit interrupt to be charged to the stack that caused the interrupt. In the default case, this stack is not charged with this processor time. The option is normally reset because it causes the total processor time for a job to be highly mix-dependent.

**READLOCK (SET)**

When this option is set, the BUZZ intrinsics insert diagnostic information (SNR, SDI, and PIR) into a lock to identify where it was BUZZed. Whether or not this option is set, SNR of the locking process is inserted if the READLOCKTIMEOUT option is set.

**READLOCKTIMEOUT (SET)**

When this option is set, the waiting time of BUZZ and BUZZ47 intrinsics is monitored. As part of this monitoring, the locking SNR is inserted into a lock as it is BUZZed. The MCP procedure READLOCKTIMEOUT is called if a lock is unavailable on the first attempt to BUZZ it. Appropriate action is taken if ownership of the lock has not changed within a specified time limit.

**REVERSEPAPERTAPE (SET)**

When this option is set, the resultant MCP allows tape-parity-retry action on paper-tape readers. This option must be reset when compiling an MCP that is intended for use at installations using the Facit paper tape reader, which cannot be reversed.

# COMPILE DECK EXAMPLE

### Introduction

This section contains an example of a WFL job deck that causes the MCP and INTRINSICS to be compiled. The deck processes SYSTEM/PATCH and the related compiles sequentially for each program to permit its use on a system with limited resources. (Many other options exist in WFL which could be used to permit the simultaneous compilation of several programs once the MCP is completed.)

In the deck, the MCP host is compiled first. The file NEWSYMBOL/MCP produced during this compilation is required to compile the remaining programs bound into the MCP.

The compilation of SCTABLEGEN causes the file NEWSYMBOL/SCTABLEGEN to be produced. This file is required during compilation of CONTROLLER; thus, SCTABLEGEN must precede CONTROLLER.

Successful completion of the job results in the creation of the files NEW/MCP and NEW/INT. (The names were chosen so as not to conflict with the MCP and INTRINSICS files currently in use.)

Files created during the compilation and used later in the job are:

| FILE NAME | PRODUCED DURING COMPILE OF | FILE TITLE ACTUALLY USED |
|---|---|---|
| MCPSYMBOLIC | MCP host | NEWSYMBOL/MCP |
| JOBFORMATTER | JOBFORMATTER | NEWSYMBOL/JOBFORMATTER |
| SCTABLEGEN | SCTABLEGEN | NEWSYMBOL/SCTABLEGEN |

**Compilation Job Deck**

```
<I> BEGIN JOB MCP/COMPILES;

<I> DATA ALGOLSUPPLEMENT/PATCH
$. SET MARK COMPARE
$# COMPILER OPTIONS
$ SET MERGE
$ SET LIST STACK SINGLE XREF
$ SET LINEINFO
   insert patches or "$. PATCHDECK <filename>" card at this point

<I> DATA ALGOLINTRINSICS/PATCH
$. SET MARK COMPARE
$# COMPILER OPTIONS
$ SET MERGE
$ SET LIBRARY
$ SET INTRINSICS
$ SET LIST STACK SINGLE XREF
$ SET LINEINFO
   insert patches or "$. PATCHDECK <filename>" card at this point

<I> DATA ALGOLPLINTRINSICS/PATCH
$. SET MARK COMPARE
$# COMPILER OPTIONS
$ SET MERGE
$ SET LIBRARY
$ SET INTRINSICS
$ SET LIST STACK SINGLE XREF
$ SET LINEINFO
   insert patches or "$. PATCHDECK <filename>" card at this point

<I> DATA CONTROLLER/PATCH
$. SET MARK COMPARE
$# COMPILER OPTIONS
$ SET MERGE
$ SET LIST STACK SINGLE XREF
$ SET LINEINFO
   insert patches or "$. PATCHDECK <filename>" card at this point

<I> DATA DCALGOLINTRINSICS/PATCH
$. SET MARK COMPARE
$# COMPILER OPTIONS
$ SET MERGE
$ SET LIBRARY
$ SET INTRINSICS
$ SET LIST STACK SINGLE XREF
$ SET LINEINFO
   insert patches or "$. PATCHDECK <filename>" card at this point

<I> DATA ESPOLINTRINSICS/PATCH
$. SET MARK COMPARE
$# COMPILER OPTIONS
$ SET MERGE
$ SET LIBRARY
$ SET LIST STACK SINGLE XREF
$ SET LINEINFO
   insert patches or "$. PATCHDECK <filename>" card at this point
```

```
<I> DATA JOBFORMATTER/PATCH
$. SET MARK COMPARE
$# COMPILER OPTIONS
$ SET MERGE
$ SET NEW
$ SET LIST STACK SINGLE XREF
$ SET LINEINFO
   insert patches or "$. PATCHDECK <filename>" card at this point

<I> DATA MCP/PATCH
$. SET MARK COMPARE
$# COMPILER OPTIONS
$ SET MERGE LINEINFO
$ SET LIST STACK XREF
$ SET NEW
   insert patches or "$. PATCHDECK <filename>" card at this point

<I> DATA PLINTRINSICS/PATCH
$. SET MARK COMPARE
$# COMPILER OPTIONS
$ SET MERGE
$ SET LIBRARY
$ SET INTRINSICS
$ SET LIST STACK SINGLE XREF
$ SET LINEINFO
   insert patches or "$. PATCHDECK <filename>" card at this point

<I> DATA SCTABLEGEN/PATCH
$. SET MARK COMPARE
$# COMPILER OPTIONS
$ SET MERGE
$ SET NEW
$ SET LIST STACK SINGLE XREF
$ SET LINEINFO
   insert patches or "$. PATCHDECK <filename>" card at this point

<I> DATA WFL/PATCH
$. SET MARK COMPARE
$# COMPILER OPTIONS
$ SET MERGE
$ SET LIST STACK SINGLE XREF
$ SET LINEINFO
   insert patches or "$. PATCHDECK <filename>" card at this point

<I>
```

```
%%%%%%%%%%%%%%%% START OF WFL COMPILE DECK  %%%%%%%%%%%%%%%%%%%%%%%%

BOOLEAN
       BADMCPCOMPILE,
       SKIPCONTROLLER,
       SKIPINTBIND,
       SKIPMCPBIND;
TASK
       TCOPY,
       TINTBIND,
       TMCPBIND,
       TUDSTRUCTURE;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE COMPILEIT (
       STRING SOURCENAME VALUE,
       STRING OBJNAME VALUE,
       STRING COMPILERNAME VALUE,
       BOOLEAN BADCOMP);
BEGIN
STRING
       COMPILEDNAME;
TASK
       TCOMP,
       TPATCH;

% MERGE THE PATCHES

RUN SYSTEM/PATCH [TPATCH];
  FILE CARD (TITLE = #SOURCENAME/PATCH);
  FILE PATCH (TITLE = PATCHES/#SOURCENAME);
  FILE TAPE (TITLE = SYMBOL/#SOURCENAME);

% CHECK FOR GOOD COMPLETION OF THE PATCH MERGE

IF TPATCH (VALUE) = 1 THEN
     DISPLAY SOURCENAME & " PATCHES OK"
   ELSE
     BEGIN
     DISPLAY "BAD " & SOURCENAME & " PATCH";
     BADCOMP := TRUE;
     GO XIT;
     END;

% SET UP THE NAME OF THE COMPILED PROGRAM

IF OBJNAME = "MCP" THEN
     COMPILEDNAME := "SYSTEM/MCPHOST"
   ELSE IF COMPILEDNAME = "INTR" THEN
     COMPILEDNAME := OBJNAME
   ELSE COMPILEDNAME := "SYSTEM/" & OBJNAME;
```

```
% COMPILE THE PROGRAM

COMPILE #COMPILEDNAME WITH #COMPILERNAME [TCOMP] TO LIBRARY;
  COMPILER FILE CARD (KIND = DISK, TITLE = PATCHES/#SOURCENAME);
  COMPILER FILE TAPE (TITLE = SYMBOL/#SOURCENAME);
  COMPILER FILE NEWTAPE (TITLE = NEWSYMBOL/#SOURCENAME);
  COMPILER FILE MCPSYMBOLIC (TITLE = NEWSYMBOL/MCP);
  COMPILER FILE SET (TITLE = NEWSYMBOL/SCTABLEGEN);

% CHECK FOR GOOD COMPLETION OF THE COMPILE

IF TCOMP IS COMPILEDOK THEN
     DISPLAY SOURCENAME & " COMPILED OK"
   ELSE
     BEGIN
     DISPLAY "BAD " & SOURCENAME & " COMPILE";
     BADCOMP := TRUE;
     END;

XIT:

END;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% START OF COMPILES

% MCP HOST

      CHANGE NEWSYMBOL/MCP TO OLDSYMBOL/MCP;
      COMPILEIT ("MCP","MCP","NEWP",BADMCPCOMPILE);
      IF BADMCPCOMPILE THEN ABORT "MCP HOST DID NOT COMPILE";

% MAKE SURE NEWSYMBOL WAS CLOSED

      IF FILE NEWSYMBOL/MCP ISNT RESIDENT THEN
           ABORT "NEWSYMBOL/MCP ISNT AVAILABLE";

% SCTABLEGEN

      COMPILEIT ("SCTABLEGEN","SCTABLEGEN","DCALGOL",SKIPCONTROLLER);

% ALGOLSUPPLEMENT

      COMPILEIT ("ALGOLSUPPLEMENT","ALGOLSUPPLEMENT","DCALGOL",
              SKIPMCPBIND);

% CONTROLLER

      IF SKIPCONTROLLER THEN
           BEGIN
           DISPLAY "CONTROLLER WILL NOT BE COMPILED";
           SKIPMCPBIND := TRUE;
           END
         ELSE
           COMPILEIT ("CONTROLLER","CONTROLLER","DCALGOL",SKIPMCPBIND);

% ALGOLINTRINSICS

      COMPILEIT ("ALGOLPLINTRINSICS","INTR/ALGOL","ALGOL","SKIPINTBIND");
```

```
% ALGOLPLINTRINSICS

     COMPILEIT ("ALGOLPLINTRINSICS","INTR/ALGPLI","ALGOL",
             "SKIPINTBIND");

% DCALGOLINTRINSICS

     COMPILEIT ("DCALGOLINTRINSICS","INTR/DCALGOL","DCALGOL",
             SKIPINTBIND);

% ESPOLINTRINSICS

     COMPILEIT ("ESPOLINTRINSICS","INTR/ESPOL","ESPOL",SKIPINTBIND);

% JOBFORMATTER

     COMPILEIT ("JOBFORMATTER","JOBFORMATTER","DCALGOL",SKIPMCPBIND);

% PLINTRINSICS

     COMPILEIT ("PLINTRINSICS","INTR/PLI","ESPOL",SKIPINTBIND);

% UDSTRUCTURETABLE

     COMPILE INTR/UDSTRUCTURETABLE WITH USERSTRUCTURE [TUDSTRUCTURE]
             TO LIBRARY;
       COMPILER FILE CARD (KIND = DISK,
             TITLE = SYMBOL/UDSTRUCTURETABLE);

     IF TUDSTRUCTURE ISNT COMPILEDOK THEN
          BEGIN
          DISPLAY "BAD UDSTRUCTURETABLE COMPILE";
          SKIPINTBIND := TRUE;
          END;

% WFL

     COMPILEIT ("WFL","WFL","DCALGOL",SKIPMCPBIND);

% BIND THE INTRINSICS FILE

     IF SKIPINTBIND THEN
          DISPLAY "INTRINSICS BIND WILL NOT BE PERFORMED"
        ELSE
          BEGIN
          BIND NEW/INT BINDER [TINTBIND] LIBRARY;
            COMPILER DATA
$ SET STACK SEGS LIST SINGLE
$ SET INTRINSICS
BIND = FROM INTR/ALGOL, INTR/ALGPLI, INTR/ESPOL, INTR/DCALGOL,
             INTR/PLI, INTR/=;

<I>
          IF TINTBIND IS COMPILEDOK THEN
               DISPLAY "NEW/INT CREATED"
             ELSE
               DISPLAY "BAD INTRINSICS BIND";
          END;
```

```
% BIND THE MCP

      IF SKIPMCPBIND THEN
           DISPLAY "MCP BIND WILL NOT BE PERFORMED"
         ELSE
           BEGIN
           BIND NEW/MCP BINDER [TMCPBIND] LIBRARY;
             COMPILER DATA
HOST IS SYSTEM/MCPHOST;
BIND CONTROLLER    FROM SYSTEM/CONTROLLER;
BIND JOBFORMATTER FROM SYSTEM/JOBFORMATTER;
BIND =             FROM SYSTEM/ALGOLSUPPLEMENT;
                        SYSTEM/WFL;

<I>
           IF TMCPBIND IS COMPILEDOK THEN
                DISPLAY "NEW/MCP CREATED"
              ELSE
                DISPLAY "BAD MCP BIND";
           END;

<I> END JOB
```

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

SUMLOG

TABLE OF CONTENTS


1. INTRODUCTION . . . . . 15-1- 1

2. LOG STRUCTURE . . . . . 15-2- 1

3. LOG ENTRIES . . . . . 15-3- 1

GENERAL FORMAT . . . . . 15-3- 1

Table 15-3-1. Format of LINK to Variable Items . . . . . 15-3- 3

4. THE MCP WRITELOG PROCEDURE . . . . . 15-4- 1

5. LOG ENTRY TYPES . . . . . 15-5- 1

Table 15-5-1. Log Entry Classes . . . . . 15-5- 2

Table 15-5-2. Format of First Four Log-Entry Words . . . . . 15-5- 4

Table 15-5-4. Log Entry Formats . . . . . 15-5- 5

MAJOR TYPE = 1 Job or Task Entry (LOGMAJJOB) . . . . . 15-5- 5

Minor Type = 1 BOJ Entry (LOGBOJTYP) . . . . . 15-5- 5

Minor Type = 3 BOT Entry (LOGBOTYP) . . . . . 15-5- 5

Minor Type = 2 EOJ Entry (LOGEOJ) . . . . . 15-5- 8

Minor Type = 4 EOT Entry (LOGEOTYP) . . . . . 15-5- 8

Minor Type = 5 File Open Entry (LOGOPENTYPE) . . . . . 15-5- 14

Minor Type = 6 File Close Entry (LOGCLOSETYPE) . . . . . 15-5- 16

Minor Type = 7 Job Rejected Entry (LOGABORTTYPE) . . . . . 15-5- 17

Minor Type = 8 Job/Task Abort History (LOGABORTHISTORY) . . 15-5- 18

Minor Type = 10 File Interval Record (LOGINTERVALTYPE) . . . 15-5- 18A

MAJOR TYPE = 2 Maintenance Entry (LOGMAJMAINT) . . . . . 15-5- 19

Minor Type = 1 Mainframe Configuration (MLMFCONFIG) . . . . 15-5- 19

Minor Type = 2 Peripheral Configuration (MLPERIPHCONFIG) . . 15-5- 21

Minor Type = 3 Maintenance Log Comment Entry (MLCOMMENT) . . 15-5- 22

Minor Type = 4 Memory Error Entry (MLMEMACCESS) . . . . . 15-5- 23

Minor Type = 5 SCR Maintenance Records (MLSCRRECORD) . . . . 15-5- 24

Minor Type = 9 I/O Condition Entry (MLIOCONDITION) . . . . 15-5- 25

Minor Type = 10 I/O Error Entry (MLIOERROR) . . . . . 15-5- 25

Minor Type = 11 Library Maintenance Compare (MLLIBERR) . . . 15-5- 30

Minor Type = 12 Mainframe Error Entry (MLMAINFRAMERR) . . . 15-5- 31

Minor Type = 13 Disk Pack Error Entry . . . . . 15-5- 33
Minor Type = 14 DFO Error Entry (MLDFOERROR). . . . . 15-5- 34
Minor Type = 15 B6800 Abort and Diagnostics (MLCPUERROR). . 15-5- 35
Minor Type = 16 MLIP I/O Error Entry. . . . . 15-5- 36A
Minor Type = 17 Hardware Configuration (HARDWARE-CONFIG). . 15-5- 36E
Minor Type = 18 Software Configuration (SOFTWARE-CONFIG). . 15-5- 36M
MAJOR TYPE = 3 String Entry (LOGMAJSTR). . . . . 15-5- 37
Minor Type = 1 RSVP Message Entry (RSVP). . . . . 15-5- 37
Minor Type = 2 Fatal Error Message Entry (FATAL). . . . . 15-5- 37
Minor Type = 3 Non-fatal Error Message Entry (NONFATAL) . . 15-5- 37
Minor Type = 4 System Message Entry (SYSTEM). . . . . 15-5- 37
Minor Type = 5 Unit Message Entry (UNIT). . . . . 15-5- 37
Minor Type = 6 Directory Message Entry (DIRECTORY). . . . . 15-5- 37
Minor Type = 7 Display Message Entry (DISPLAY). . . . . 15-5- 37
Minor Type = 8 Status Message Entry (STATUS). . . . . 15-5- 37
Minor Type = 9 Unit RSVP Message Entry (UNITRSVP) . . . . . 15-5- 37
MAJOR TYPE = 4 MCS Entry (LOGMAJMCS) . . . . . 15-5- 38
Minor Type = 1 Log-on Entry (MCSLOGON). . . . . 15-5- 38
Minor Type = 2 Log-off Entry (MCSLOGOFF). . . . . 15-5- 39
Minor Type = 3 RJE Control Card Entry (MCSRJECC). . . . . 15-5- 41
Minor Type = 4 MCS Message Entry (MCSMESSAGE) . . . . . 15-5- 42
Minor Type = 5 Session Accrual Entry (MCSACCRUAL) . . . . . 15-5- 43
Minor Type = 6 MCS Security Violation Entry . . . . . 15-5- 43A
MAJOR TYPE = 5 DCP Entry (LOGMAJDCP) . . . . . 15-5- 44
Minor Type = 1 DCP Initiate Entry (LOGDCPINIT). . . . . 15-5- 44
Minor Type = 2 MCS Initiate Entry (LOGMCSINIT). . . . . 15-5- 45
Minor Type = 3 DCP Fault Entry (LOGDCPFAULT). . . . . 15-5- 46
Minor Type = 4 MCS Error Entry (LOGMCSMSG). . . . . 15-5- 48
MAJOR TYPE = 6 Miscellaneous Entry (LOGMAJMISC). . . . . 15-5- 49
Minor Type = 1 Halt/Load Entry (LOGHL). . . . . 15-5- 49
Minor Type = 2 Log Release Entry (LOGLR). . . . . 15-5- 50
Minor Type = 3 SETSTATUS Call Entry (LOGSETSTAT). . . . . 15-5- 51
Minor Type = 4 Security Violation Entry (LOGSECURITY) . . . 15-5- 52

MAJOR TYPE = 7 Installation Entry (LOGMAJINST) . . . . . 15-5- 53
MAJOR TYPE = 9 Reader/Sorter Entry (LOGMAJRS). . . . . 15-5- 54
Minor Type = 1 RSMONITOR Initiation (LOGRSMONITORINIT). . . 15-5- 54
Minor Type = 2 RSMONITOR Faults (LOGRSMONITORFAULT) . . . . 15-5- 55
Minor Type = 3 RSMONITOR Termination (LOGRSMONITORTERM) . . 15-5- 56
Minor Type = 4 RSP Initiation (LOGRSPINITIATION). . . . . 15-5- 57
Minor Type = 5 RSP Termination (LOGRSPTERM) . . . . . 15-5- 58
Minor Type = 6 RSP Dump Reason (LOGRSPDUMPHISTORY). . . . . 15-5- 60
Minor Type = 7 RSP Non-fatal Faults (LOGRSPFAULT) . . . . . 15-5- 61
Minor Type = 8 RSP Disk Errors (LOGRSPDISKERR). . . . . 15-5- 62
Minor Type = 9 RSP BIC Errors (LOGRSPBICERR). . . . . 15-5- 63
Minor Type = 10 RSP UA or UR (LOGRSPUAUR) . . . . . 15-5- 65
Minor Type = 11 RSLABEL (LOGRSLABEL). . . . . 15-5- 66
Minor Type = 12 RS Statistics (LOGRSSTATISTICS) . . . . . 15-5- 67
Minor Type = 13 RS UA or UR (LOGRSUAUR) . . . . . 15-5- 69
Minor Type = 14 RS SUBSYSTEM MESSAGES (LOGRSMSGS) . . . . . 15-5- 70
Minor Type = 15 UCR Initiation (LOGUCRBOJ). . . . . 15-5- 71
Minor Type = 16 UCR Termination (LOGUCREOJ) . . . . . 15-5- 72
Minor Type = 17 BIC UA or UR (LOGBICUAUR) . . . . . 15-5- 73
6. IAD LOGS. . . . . 15-6- 1

# 1. INTRODUCTION

This chapter serves as a reference to users of the system summary log "SUMLOG" information on Burroughs B 5000/B 6000/B 7000 series large-scale computer systems. This log contains records concerning jobs previously run, MCP activity, and other associated information regarding the past status of the machine environment.

The formats of the records and words associated with this chapter are taken from the 3.3 release of the MCP, JOBFORMATTER, and LOGANALYZER. Therefore, because of the volatility of SUMLOG (the ability for easily modifying and adding structures), care should be taken when attempting to isolate and correlate data from this chapter and various log outputs produced from other releases.

The information in this chapter applies equally to the B 5000, B 6000, and B 7000 series systems unless specifically noted otherwise for that data item.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# 2. LOG STRUCTURE

The system summary log, SYSTEM/SUMLOG, is designed by default as a protected disk file of up to 50 rows of 2000 segments per row, consisting of a varying number of 30-word physical records. The number of words of information in each record differs not only for the different log entry types but also between specific entries of the same type (because they include program names, file names, and other variable-length information). The exact number of records in an entry is entered in the first word (word 0, field [39: 8]) of the 30-word record of the entry, and the length of the entry in words is contained in field [47:16] of word 3.

When a log release (LR) or transfer log (TL) system input message is entered by the operator, the current SYSTEM/SUMLOG is renamed SUMLOG/<system serial #>/<date>/nnnnnn, where nnnnnn is a six-digit integer that is one greater than that used at the previous LR or TL. The MCP maintains this number in a disk table, so that the numbers increase until the next cold start.

The SUMLOG file is also changed when the MCP determines that the existing file is full (95% filled) and an automatic log release is initiated. Because the SUMLOG file holds up to 100,000 records, an automatic log release takes place at approximately 95,000 records.

Two system procedures can be used as models for log analysis - JOBFORMATTER and SYSTEM/LOGANALYZER. JOBFORMATTER, a DCALGOL program bound into the MCP, prints the job wrap-up summaries as part of the AUTOPRINT option. SYSTEM/LOGANALYZER is the log analysis utility program. It includes log format defines from JOBFORMATTER, which can also be done by an installation ALGOL or DCALGOL program. These defines may be a useful adjunct to this document; therefore, the log format details include the names of the log classifications.

**NOTE**

Although every effort is made to keep the fields as described in the section on "LOG ENTRY TYPES", words may have to be added to a record or new log types may have to be added. Words are added, if necessary, at the end of the fixed part of a record. For example, in the BOJ record, additions would be made after the eighteenth word. Because all log records contain their length in word 3, changes made in that manner should be invisible to existing log programs.

Addition of types is more of a problem for programs. Therefore, programs should be written to expect major and minor types not recognized.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# 3. LOG ENTRIES

## GENERAL FORMAT

The log is organized so that bad information in one record or a disk error does not hamper retrieval of other records. According to this organization, all logical log entries are split into contiguous, fixed-size physical records of 30 words each. The first word of every physical record contains a record group description that describes its position in the logical log entry. The MCP procedure WRITELOG breaks up the logical log entry into physical log records.

When analyzing the log, a program must reassemble the logical entry from the physical records found in the log before using the word numbers used to describe the log entries in section 5. Those word numbers refer to the words as contained in the logical entry, not as they occur after the entry has been broken up into physical records.

The relationship between the words of the logical entry and their location in the physical log record is shown in the following table:

LOG ENTRY:

| Physical Rec # | Physical Record Word 0 | Logical Entry Data Words |
|---|---|---|
| 1 | Rec 1 of 3 | 0-29 |
| 2 | Rec 2 of 3 | 30-58 |
| 3 | Rec 3 of 3 | 59-87 |

The first four words (0-3) of every logical log entry contain the following information (see Table 15-5-2 for the exact format). The record group description is found in word 0 of every physical log record as well as in word 0 of the logical log entry.

1. Word 0 contains a record group description with the following four fields:

   a. The cardinality of this physical record within the logical log entry ("this is record m of n") beginning with 1.

   b. The total number of records in the logical log entry.

   c. The job number of the associated job.

   d. The task number of the task causing the log entry.

2. Word 1 contains the current date in binary Julian format.

3. Word 2 contains the time the log entry was made, in units of 2.4 usec.

4. Word 3 contains the log entry type. This word is divided into a length field, plus major and minor classification fields. The length field contains the number of words in the log entry. The major field gives a rough classification according to originator, and the minor field completes the classification of the entry.

The physical records of each log entry are always contiguous in the SUMLOG. However, these entries are not always logged in strict chronological order. Aside from the effects of the TR and DR ODT commands, unusual timestamps may occur when a Halt/Load is done after part of the system has been powered off and on. Furthermore, when contention occurs over which task is to write to the log next, the entries are made in the order of the task's priorities, not in the time sequence of their requests. The physical records of a single log entry are never split across two different logs, even if the TL or LR ODT commands were entered while the entry was being written. Entries are always made in the current SUMLOG file. Therefore, entries for a single job or task may occur in all the SUMLOG files that were active while the job or task was running.

Each log entry is divided into a fixed part and a variable part. All items of information are given a reserved location in the fixed part. However, this location is not adequate for items such as task names and file titles, whose size may vary. In these cases, the location in the fixed part is used to contain a LINK to the actual information, which is placed in the variable part of the log entry. Because all the fixed information precedes the variable information, items may always be found at the same index, and for any given level of the log, all records of a single type have the same fixed-part size.

This index helps preserve upward compatability, because new items of log information are always added at the end of the fixed part and are thus invisible to log programs that do not expect to find them. Downward compatability is somewhat more difficult to achieve because analysis programs must first check the length of the fixed part in order to determine if the information they seek is present in the version of the log under analysis.

Two formats for the LINK word used to refer to variable-length information exist. The first format, which is given in the following table, is used for all log records except the Hardware and Software Configuration records (Major Type = 2, Minor Types = 17 and 18, respectively). This simple format for the LINK provides a word index to the start of the variable-length information and is the format referred to whenever the word LINK occurs in a log-entry description, except for the Hardware and Software Configuration entries.

Table 15-3-1. Format of LINK to Variable Items

| | |
|---|---|
| word <n> | Fixed part entry for an item of variable length. |
| | [39:20] length of this item in words. |
| | [19:20] index of word in log entry where this item starts. It is found in the variable part of the log entry. |

The Hardware and Software Configuration records require a link containing more information, such as the type of variable-length information referred to and the amount of information available. To accommodate this information, a new format for a LINK word and a new structure for a log record using these LINKs has been defined. The layout of the new LINK word is as follows:

| Field | Meaning |
|---|---|
| [47:08] | LINK-TYPE |
| [39:04] | LINK-VERSION |
| [35:16] | START-INDEX |
| [19:12] | ITEM-COUNT |
| [07:08] | ITEM-WIDTH |

The new LINK describes a contiguous group of array entries known as a "region". LINKs may appear within a region that is described by a LINK; this recursive use creates a tree structure. This LINK structure is described in the following paragraphs.

The LINK-TYPE field of the LINK specifies the kind of information encoded in this region. LINK-TYPE values are assigned by the encoding routines.

The LINK-VERSION field identifies the version of the format of the information. The LINK-VERSION is changed only when the offset or format of a previously existing item has changed.

The START-INDEX field of the LINK specifies the array index where the region described by the LINK starts. All items residing in this area have a fixed offset from the START-INDEX.

The ITEM-COUNT field contains the number of items found in the region described by the LINK.

Regions may be either homogeneous or heterogeneous. If a region is homogeneous, all items that it contains must have the same size and format. This size is stored in the ITEM-WIDTH field of the LINK. If the region is heterogeneous, the ITEM-WIDTH field is 0, and the data in the region may be formatted in the most convenient manner.

The units for the ITEM-WIDTH field are not specified, thereby allowing one type of LINK to describe items with a width specified in characters, and a different type of LINK to describe items with a width specified in words. A nonzero ITEM-WIDTH is ambiguous to a routine attempting to skip over data that it does not understand because the unit associated with the value is not specified. An ITEM-WIDTH of 0 is defined to mean that the ITEM-COUNT is given in words. Consequently, an ITEM-WIDTH of 0 is used for heterogeneous regions.

The direct part of a structure contains entries with a known fixed offset. The indirect part of a structure must be reached by following LINKs from a direct part. An indirect part may become a direct part when the LINK describing the indirect part is followed. This direct part has all the properties of any other direct part.

Every user program should always verify that the LINK-TYPE and LINK-VERSION fields contain the expected values. If they do not, the program should not attempt to decode the information. The ITEM-COUNT field should be checked to verify that the expected number of items is present. The ITEM-WIDTH field should also be checked to verify that the item size conforms to the initial specifications. As long as the LINK-TYPE and LINK-VERSION match the values that the decoding program expects, the data are in a format that the program understands. The ITEM-COUNT or ITEM-WIDTH, or both, may be larger or smaller than expected. The program should be coded to ignore any extra data in an item or unexpected items in a region.

# 4. THE MCP WRITELOG PROCEDURE

The MCP WRITELOG procedure provides the mechanism to enter the log records, time stamp them, send them to the SUMLOG, and if the other parameters and stack status permit, to send them to the jobfile. WRITELOG suppresses logging to either of these destinations, based on two MCP arrays: SUMLOGOMIT for the system log and JOBLOGOMIT for the jobfile. When a log request is made, WRITELOG indexes the appropriate array by the log major type and selects a bit, using the log minor type. If the bit is on, logging is suppressed on the respective file. If the major type is greater than the array length or if the minor type is greater than 47, the entry is logged. By default, open- and close-records are suppressed from the jobfiles, as is aborted history from the SUMLOG.

A log entry type has been provided for the installation use. To get records of this type into the log, an installation must add appropriate WRITELOG calls to the MCP. (The first four words of these records must agree in format with the standard entry types.)

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# 5. LOG ENTRY TYPES

Ten major classes of log entry records (identified by major type numbers) have been established; each major class is divided into subclasses that are identified by minor type numbers. The log entry classes are listed in Table 15-5-1, and a detailed description of the format of each type is presented in Table 15-5-4. The entries are grouped by function.

Major and Minor Type = 0 are conditions reserved to flag null records. No such records are currently being generated, but future MCPs may write them.

Table 15-5-1. Log Entry Classes

| MAJOR TYPE | MINOR TYPE | RECORD CLASS |
|---|---|---|
| 1 | | JOB RECORD |
| | 1 | Beginning-of-job (BOJ) record. |
| | 2 | End-of-job (EOJ) record. |
| | 3 | Beginning-of-task (BOT) record. |
| | 4 | End-of-task (EOT) record. |
| | 5 | File-open record. |
| | 6 | File-close record. |
| | 7 | Controller-rejected job record. |
| | 8 | Aborted-history record. |
| | 9 | Job or task continuation. (B 7000 only) |
| | 10 | File interval record. |
| 2 | | MAINTENANCE RECORD |
| | 1 | Mainframe configuration record (B 7000 only). |
| | 2 | Peripheral configuration record (B 7000 only). |
| | 3 | Maintenance comment record. |
| | 4 | Memory access error accumulations. |
| | 5 | SCR maintenance entries. |
| | 9 | I/O condition. |
| | 10 | I/O error record. |
| | 11 | Library maintenance error record. |
| | 12 | B 7000 mainframe error. |
| | 13 | B 7000 disk pack error. |
| | 14 | Bad disk file optimizer scanin. |
| | 15 | B 6800 mainframe error. |
| | 16 | MLIP I/O error. |
| | 17 | Hardware configuration record. |
| | 18 | Software configuration record. |
| 3 | | STRING RECORD |
| | 1 | RSVP message. |
| | 2 | FATAL error message. |
| | 3 | NONFATAL error message. |
| | 4 | SYSTEM message. |
| | 5 | UNIT error message. |
| | 6 | DIRECTORY error message. |
| | 7 | DISPLAY message. |
| | 8 | STATUS message. |
| | 9 | UNIT RSVP message. |

Table 15-5-1. Log-Entry Classes (Continued)

| MAJOR TYPE | MINOR TYPE | RECORD CLASS |
|---|---|---|
| 4 | | MCS RECORD |
| | 1 | Session log-on record. |
| | 2 | Session log-off record. |
| | 3 | RJE control card record. |
| | 4 | MCS message record. |
| | 5 | MCS time accrued. |
| | 6 | MCS security violation entry. |
| 5 | | DCP MAINTENANCE RECORD |
| | 1 | DCP initialization record. |
| | 2 | MCS initialization record. |
| | 3 | DCP fault error record. |
| | 4 | MCS result message record. |
| 6 | | MISCELLANEOUS RECORD |
| | 1 | Halt/Load record. |
| | 2 | Log release record. |
| | 3 | SETSTATUS record. |
| | 4 | Security violation. |
| 7 | | INSTALLATION RECORD |
| 8 | | PROCESSOR AND CPU ERROR RECORD |
| | 1 | Miscellaneous parameters. |
| | 2 | Stack image, minus tags. |
| | 3 | Stack image tags. |
| 9 | | READER/SORTER ENTRIES |
| | 1 | Initiation of independent runner. |
| | 2 | Fault information. |
| | 3 | Program termination records. |
| | 4 | B 6800/B 1700 communications initiation. |
| | 5 | Termination records. |
| | 6 | Dump information. |
| | 7 | Fault records. |
| | 8 | Disk errors. |
| | 9 | Bus interface control (BIC) errors. |
| | 10 | Reserve units (B 1700). |
| | 11 | Label R/S subsystems. |
| | 12 | Statistics records. |
| | 13 | Reserve units (B 6800). |
| | 14 | Communication messages. |
| | 15 | BOJ of B 1700 R/S control program. |
| | 16 | EOJ of B 1700 R/S control program. |
| | 17 | BIC unit reserve. |

Table 15-5-2. Format of First Four Log-Entry Words

| | |
|---|---|
| Word 0 | Record group description.<br><br>[47: 8] cardinality of record in the block.<br>[39: 8] total number of records in this entry.<br>[31:16] job number.<br>[15:16] task number. |
| Word 1 | Julian date in binary. |
| Word 2 | Time log entry was made using the 2.4 usec clock. |
| Word 3 | Log entry type code.<br><br>[47: 6] length of fixed part of record.<br><br>If equal to 0, then it is a pre-3.1 record format.<br>If it equals 3, then the fixed length is not available.<br><br>[41:10] log record size.<br>[31:16] major type.<br>[15:16] minor type (subtype). |

Table 15-5-4. Log Entry Formats

**MAJOR TYPE = 1 Job or Task Entry (LOGMAJJOB)**

Minor Type = 1 BOJ Entry (LOGBOJTYP)
Minor Type = 3 BOT Entry (LOGBOTYP)

Words 0-3 See Table 15-5-2.

Word 4 Task priority information.

[47:24] job queue.
[23:24] priority.

Word 5 Task type.

0 = Dependent task (PROCESS).
1 = Coroutine (CALL).
2 = Independent task (RUN).
3 = Job stack.

Word 6 LINK to external name.

Word 7 LINK to usercode.

Word 8 LINK to charge code.

Word 9 LINK to compiled name. Contains name of object code file if task was COMPILE or BIND, otherwise value is zero.

Word 10 Task source and destination information.

[45: 6] destination MCS.
[39: 1] destination is REMOTE.
[38:15] destination UNIT.
[21: 6] originating MCS.
[15: 1] origin is REMOTE.
[14:15] originating UNIT.

Word 11 LINK to access code.

Word 12 Job entry time.

[47:16] contains the date.
[31:32] holds the entry time.

The job entry time is the time stamp of the code file. Thus, for a job it is the time the WFL compiler finished with it and it entered the system. For a task, it is the timestamp of the object code file being run. It is zero if the code stack was being shared with an already running task. The date format is: [47:16] (Julian date - 70000). The entry time format is: [31:32] (Time of day in ticks DIV 16).

Table 15-5-4. Log-Entry Formats (Continued)

Word 13 Compiler information.

[47: 1] IPC capable.
[46: 1] sort capable.
[45: 1] control program (CP).
[44: 1] DMS capable.
[43: 1] no uplevel pointers.
[42: 1] privileged program (PP).
[41: 1] library capable.
[31: 8] language type.

0 = ALGOL
1 = COBOL
2 = FORTRAN
3 = XALGOL
4 = PL/I
5 = JOVIAL
6 = NEWP
7 = ESPOL
8 = DCALGOL
9 = BASIC
10 = WFL
11 = VFORTRAN
12 = VLINKER
254 = INTRINSICS ($INTRINSICS set)
255 = MCP ($MCP set)

[23: 8] compiler mark number and level number.
[ 9:10] compiler cycle number.

Word 14 Swapper information.

[39:10] number of swap space core slots in use.
[15: 8] SUBSPACE attribute requested.

0 = do not run in swap space.
1 = data (D2) stack only in swap space.
2 = data in swap space. Code (D1) stack in swap space if code file is in my directory.
3 = data and code in swap space.

[ 7: 8] SUBSPACE actually granted.

(Same as [15: 8] but 2 is not valid.)

Word 15 Box numbers of task.

[39:20] box number of code (D1) stack.
[19:20] box number of data (D2) stack.

Table 15-5-4. Log-Entry Formats (Continued)

| | |
|---|---|
| Word 16 | LINK to value of SUBSYSTEM attribute. |
| Word 17 | LINK to the value of the ITINERARY chain. |
| | The LENGTH field is in characters rather than words. |
| Word 18 | Amount of time spent in the schedule. |
| Word 19 to end. | Variable length data pointed to by words 6, 7, 8, 9, 11, 16, and 17. |

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 1 Job or Task Entry (LOGMAJJOB)**

Minor Type = 2 EOJ Entry (LOGEOJ)
Minor Type = 4 EOT Entry (LOGEOTYP)

Words 0-3 See Table 15-5-2.

Words 4-10 Same as major type 1, minor type 1 and 3.

Word 11 Processor time.

Word 12 I/O time.

Word 13 Number of lines printed.

Word 14 Number of cards read.

Word 15 Number of cards punched.

Word 16 Memory integral code, D1 stack.

Word 17 Memory integral data, D2 stack.

Word 18 Termination condition. Contains the value of the history task attribute at the time of termination. The values of these fields are as follows:

[ 7: 8] gives the type of termination.

4 = DSed
5 = Normal EOT
6 = Syntax Error

[15: 8] gives the cause of a DS.

1 = OPERATORCAUSE
2 = PROGRAMCAUSE
3 = RESOURCECAUSE
4 = FAULTCAUSE
5 = SYSTEMCAUSE
6 = DCERR
7 = IOERR
8 = SOFTIOERR
9 = NEWIOERR
10 = UNIMPLEMENTED
11 = UNSPECIFIEDCAUSE
12 = EBDMSERR
13 = NETWORKCAUSE

[23: 8] gives the detailed reason for DS. (Reasons for a DS for DCERR, IOERR, SOFTIOERR, and NEWIOERR are given in messages generated by these procedures.) The reasons, by cause, are as follows:

Table 15-5-4. Log-Entry Formats (Continued)

For OPERATORCAUSE:

0 = RSVP
1 = CLEARUNIT
2 = JUSTDSED

For PROGRAMCAUSE:

0 = MISSINGCODEFILENAME
1 = MISSINGCODEFILE
3 = INITACTIVETASK
4 = NONEXTERNALRUN
14 = INVALIDACCESSCODE
15 = INCOMPATIBLEBOXES
18 = DEATHINFAMILY
19 = CRITICALBLOCK
20 = BADGOTO
23 = INVALIDPARAM (B 7000 only)
26 = NOTEXECUTABLE
27 = UNMATCHEDPARAMS
28 = INVCOMPILER
29 = SECURITYERROR
30 = LIBMAINT
31 = ILLEGALTASKXFER
32 = BADRECSIZEDEALLOOC
37 = MISSINGINTRINSIC
38 = INCOMPATIBLEVEL
39 = INFANTICIDE
40 = NOTBOUND
41 = ILLEGALOWNARRAY
42 = DIMSIZERROR
43 = UPLEVELATTACH
44 = ILLEGALSWAP
45 = SWAPDOESNTALLOW
46 = BADTASKATTRIBUTE
47 = MISSINGCARDDECK
48 = BADRESTART
49 = BADEVENTUSAGE
50 = BADGIVELOCK
51 = BADGETLOCK
57 = COMPILERSONLY
58 = TASKLIMEXCEEDED
59 = AXBADARRAY
60 = RUNTIMEWFL
61 = COMPILERERROR
62 = XSPARAMS
62 = FORTRANERR (B 7000 only)
63 = PLIRUNTIMEERR (B 7000 only)
64 = MATHERR (B 7000 only)
65 = INTRINSICSERR (B 7000 only)
66 = FORMATERR (B 7000 only)
67 = SORTKILL (B 7000 only)
66 = LIBMISSINGNAME
67 = LIBTYPEMISMATCH
68 = LIBNOTINITIATED
69 = CYCLICPROVISIONSPECIFICATION
70 = PREVIOUSLYFROZENLIB
71 = LIBIMPLEMENTATIONERROR
72 = BADPTRLIB
73 = NONUNIQLIB
74 = SAVELIBTASKNEVERCALLED

Table 15-5-4. Log-Entry Formats (Continued)

```
75 = SWAPJOBCANTBELIB
76 = GLOBALLIBSONLY
77 = BADLIBTASK
78 = LIBFEATURENOTIMPLEMENTED
79 = BADCOMPILERINDEX
80 = LIBNOTPROCESSEDORRUN
81 = LIBMUSTBESEPERATESTACK
82 = INVALIDPARAM
83 = FORTRANERR
84 = PLIRUNTIMEERR
85 = INTRINSICSERR
86 = MATHERR
87 = FORMATERR
88 = LIBDEIMPLEMENTATIONERROR
89 = LIBLEVELINCOMPATIBLE
90 = BADLIBTITLE

For RESOURCECAUSE:

 0 = PROCESSEXCEED
 1 = IOEXCEED
 2 = STACKEXCEED
 3 = PRINTEXCEED
 4 = PUNCHEXCEED
 5 = CARDREADEXCEED
 6 = MEMORYEXCEED
 7 = DIRECTORYEXCEEDED
 8 = TAPEEXCEEDED
 9 = WAITEXCEEDED
10 = ELAPSEDEXCEEDED
11 = DISKLIMITEXCEEDED
12 = STRINGPOOLEXCEEDED
13 = PARENTMEMORYEXCEEDED

For FAULTCAUSE:

 1 = DIVIDEBYZERO
 2 = EXPOVERFLOW
 3 = EXPUNDERFLOW
 4 = INVALIDINDEX
 5 = INTEGEROVERFLOW
 6 = INACTIVEQ
 7 = MEMORYPROTECT
 8 = INVALIDOP
 9 = LOOP
10 = MEMORYPARITY
11 = SCANPARITY
12 = INVALIDADDRESS
13 = STACKOVERFLOW
14 = STRINGPROTECT
15 = PROGRAMMEDOP
16 = BOTTOMOFSTACK
17 = SEQUENCEERROR
18 = INVALIDPCW
19 = STACKUNDERFLOW
20 = ZAPPED            (B 6000 only)
```

Table 15-5-4. Log-Entry Formats (Continued)

```
   20 = PROCINTERNAL      (B 7000 only)
   21 = INVALIDINTERRUPT(B 7000 only)
   22 = SCANINPARITY      (B 7000 only)
   23 = MEMFAIL1          (B 7000 only)
   24 = PRIVILEDGEDINST   (B 7000 only)
   25 = MEMPARANDFAIL1    (B 7000 only)
   26 = MEMORYFAIL2       (B 7000 only)
   27 = IOMERR            (B 7000 only)
   28 = PINTANDFAIL1      (B 7000 only)
   29 = PINTANDINVADDR    (B 7000 only)
   35 = PROCDIED          (B 7000 only)
   36 = PROCFAILED        (B 7000 only)
   40 = DISKPARITY

For SYSTEMCAUSE:

    1 = NOMEM
    2 = PARITYONPBIT
    3 = ARRAYTOOLARGE
    4 = INCOMPATIBLEWFLJOBFILE
    4 = NOUSERDIRECTORY (B 6000 only)
    4 = SYSTEMRECOVERY  (B 7000 only)
    5 = HUNGSWAPJOB
    6 = TCMABORT        (B 7000 only)

For SORT/DISK parity errors.

    1 = DYNAMICOWNARRAY

For UNIMPLEMENTEDCAUSE:

    1 = DYNAMICOWNARRAY

For NETWORKCAUSE:

    2 = NOHOST
    3 = HOSTHALTLOADED
    5 = HOSTNOTREACHABLE
   10 = USERPORTERROR
   11 = USERSIGNALERROR
```

Table 15-5-4. Log-Entry Formats (Continued)

Word 19 Swapper information.

[39:10] number of swap space core slots in use.
[15: 8] SUBSPACE attribute requested.

0 = do not run in swap space.
1 = data (D2) stack only in swap space.
2 = data in swap space. Code (D1) stack in swap space if code file is in my directory.
3 = data and code in swap space.

[ 7: 8] SUBSPACE actually granted.

(Same as [15: 8] but 2 is not valid.)

Word 20 Value of ACTIVETIME. (B 6000 only)

(The MCP must be compiled with ACTIVETIME set to log this value.)

Word 21 Average code space.

Word 22 Average data space.

Word 23 Elapsed time.

Word 24 LINK to access code.

Word 25 Compiler information.

[47: 1] IPC capable.
[46: 1] sort capable.
[45: 1] control program (CP).
[44: 1] DMS capable.
[43: 1] no uplevel pointers.
[42: 1] privileged program (PP).
[41: 1] library capable.
[31: 8] language type.

0 = ALGOL
1 = COBOL
2 = FORTRAN
3 = XALGOL
4 = PL/I
5 = JOVIAL
6 = NEWP
7 = ESPOL
8 = DCALGOL
9 = BASIC
10 = WFL
11 = VFORTRAN
12 = VLINKER
254 = INTRINSICS ($INTRINSICS set)
255 = MCP ($MCP set)

[23: 8] compiler mark number and level number.
[ 9:10] compiler cycle number.

Table 15-5-4. Log-Entry Formats (Continued)

| | |
|---|---|
| Word 26 | Task box number with the bit I-1 turned on in each field to specify box I is valid.<br><br>[31: 8] data core allowed.<br>[23: 8] code core allowed.<br>[15: 8] data core occupied.<br>[ 7: 8] code core occupied. |
| Word 27 | LINK to the value of the SUBSYSTEM attribute. |
| Word 28 | LINK to the value of the ITINERARY chain.<br><br>The LENGTH field is in characters rather than words. |
| Word 29 | Amount of time spent in the ready queue. |
| Word 30 | Amount of processor time spent handling initial P-bits for this task. |
| Word 31 | Number of initial P-bits for this task. |
| Word 32 | Amount of processor time spent handling all other P-bits for this task. |
| Word 33 | Number of other P-bits for this task. |
| Word 34 | Time at which the task was initiated.<br><br>[47:16] date (Julian date - 70000).<br>[31:32] entry time (time of day in ticks DIV 16). |
| Word 35 to end. | Variable-length data for the LINKs in this record. |

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 1 Job or Task Entry (LOGMAJJOB)**

Minor Type = 5 File Open Entry (LOGOPENTYPE)

| | |
|---|---|
| Words 0-3 | See Table 15-5-2. |
| Word 4 | File size in units of words. (Valid for disk and pack only.) |
| Word 5 | LINK to internal name. |
| Word 6 | LINK to external name. |
| Word 7 | Open description word.<br><br>[39: 8] unit number.<br>[31: 8] unit type.<br>[23: 8] value of the FILEKIND file attribute.<br>[15: 8] open error number.<br>[ 7: 7] value of the MYUSE file attribute.<br><br>1 = INPUT<br>2 = OUTPUT<br>3 = I/O<br><br>[ 0: 1] true if [15: 8] contains a valid error. |
| Word 8 | LINK to access code. |

Table 15-5-4. Log-Entry Formats (Continued)

Word 9 OPEN parameter information.

[46: 1] valid subfile field.
[39: 8] relative integer subfile index.
[ 7: 8] open type.

1 = OPEN WAIT
2 = AVAILABLE
3 = OFFER
4 = RESIDENT
5 = PRESENT
6 = PBT REEL SWITCH OPEN
7 = STACK ASSIGNED OPEN REVERSE

[15: 8] position.

1 = AT FRONT
2 = AT END

[23: 8] motion.

1 = MOVE IT
2 = DON'T BOTHER

Word 10 to end. Variable-length data pointed to by words 5, 6, and 8.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 1 Job or Task Entry (LOGMAJJOB)**

Minor Type = 6 File Close Entry (LOGCLOSETYPE)

Words 0-3 See Table 15-5-2.

Word 4 File size in units of words; valid for disk files only.

Word 5 LINK to internal name (standard form).

[12:20] Word of the log entry where the item starts. The word number refers to the log entry after it is rebuilt into a single entry; that is, the record group description words of the second and succeeding physical records are skipped in this count.

[39:20] Length of the item in units of words.

Word 6 LINK to external name (standard form).

Word 7 Close description word.

[39: 8] Unit number.
[31: 8] Unit type.
[23: 8] Value of FILEKIND file attribute.
[15: 8] Close error number.
[ 7: 7] Value of MYUSE file attribute.

1 = INPUT
2 = OUTPUT
3 = I/O

[ 0: 1] True if [15: 8] is valid.

Word 8 Name qualification information.

[42:15] USASI generation number.
[27:11] SAVEFACTOR.
[16:17] Creation date in binary Julian format.

Word 9 File blocking information.

[47:16] BLOCKSIZE.
[27:11] MINRECSIZE.
[15:16] MAXRECSIZE.

Word 10 Transaction count (number of logical I/Os).

Word 11 Elapsed I/O time.

Table 15-5-4. Log-Entry Formats (Continued)

Word 12 Miscellaneous information.

For a disk file:

[47: 1] IAD file.
[46: 1] Permanent file.
[44: 1] Updated file.
[43: 1] Crunched file.
[42: 1] Protected file.
[41: 1] Multipack disk-pack file.
[40: 1] Interchange disk-pack file.
[32: 1] Duplicated file.
[31: 4] Number of copies.
[27: 4] Speed.
[23:24] Value of AREASIZE attribute. For units other than tape or disk, this word contains zeros.

For a tape file:

[15: 1] ON if a multi-reel file is present.
[14:15] Reel number.

For units other than tape or disk, this word contains zeros.

Word 13 Serial number (six EBCDIC characters) for tape file. This word is zero for other units.

Word 14 Unit information.

[25:14] Unit cycle.
[11: 8] Unit version number.

Table 15-5-4. Log-Entry Formats (Continued)

Word 15 Close parameters.

[46: 1] Indicates if the file is a valid subfile.
[39:16] Indicates subfile index, if file is a valid subfile.
[23: 8] Disposition.

1 = REWIND
2 = NOREWIND
3 = SAVE
4 = LOCK
5 = PURGE
6 = CRUNCH
7 = HERE
8 = BLOCKEXIT

[15: 8] Association.

1 = RELEASE
2 = RETAIN
3 = RESERVE
4 = DISABLE

[ 7: 8] Close type.

1 = REGULAR CLOSE
2 = REEL CLOSE
3 = DONT WAIT

Word 16 Physical I/O count.

[47:24] Read count.
[23:24] Write count.

Word 17 LINK to FORMMESSAGE (standard form).

Word 18 LINK to port file information.

This LINK is zero if this entry is not for a port file. If the LINK is non-zero, it points to an area used to contain a block of additional information for port files. The information contained in that block is listed in the following table. The location of each item is described by an offset from the start of this block of information.

Table 15-5-4. Log-Entry Formats (Continued)

The LINKs within this block of information point to strings of characters. The length of each string is given in characters (not words) in the length field ([39:20]) of the LINK. The index field ([19:20]) of the LINK provides the word index in the log entry where the information starts.

| Offset | Contents |
|---|---|
| 0 | LINK to YOURHOSTNAME value. |
| 1 | LINK to YOURNAME value. |
| 2 | Value of MAXSEGSIZE attribute. |
| 3 | LINK to YOURUSERCODE value. |
| 4 | Number of messages sent. |
| 5 | Number of messages received. |
| 6 | Number of segments sent. |
| 7 | Number of segments received. |
| 8 | Number of retransmissions. |
| 9 | Number of control frames sent. |

Word 19 to end. Variable-length data for the LINKs in this record.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 1 Job or Task Entry (LOGMAJJOB)**

Minor Type = 7 Job Rejected Entry (LOGABORTTYPE)

Words 0-3 See Table 15-5-2.

Word 4 Reason for the DS action. See SYMBOL/JOBFORMATTER for a more complete description of the DS cause.

[31: 8] reason for controller DS:

1 = DSed out of the queue.

2 = DSed by attribute error on job start.

[23:16] attribute error number.

8 = Invalid user code.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 1 Job or Task Entry (LOGMAJJOB)**

Minor Type = 8 Job/Task Abort History (LOGABORTHISTORY)

(These are aborted TASK history entries and are in string format [see Major type = 3]. SUMLOGOMIT normally inhibits these from being logged.)

Table 15-5-4. Log Entry Formats (Continued)

**MAJOR TYPE = 1 Job/ or Task Entry (LOGMAJJOB)**

Minor Type = 10 File Interval Record (LOGINTERVALTYPE)

The format and contents of the File Interval log entry are identical to those of the File Close log entry with the following exceptions:

| | |
|---|---|
| Word 3 | Field [15:16] minor type = 10 |
| Word 7 | Field [47:08] is undefined. |
| Word 7 | Field [15:08] is undefined. |
| Word 7 | Field [00:01] is undefined. |
| Word 15 | Field [23:08] is undefined. |
| Word 15 | Field [15:08] is undefined. |
| Word 15 | Field [07:08] is undefined. |

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 2 Maintenance Entry (LOGMAJMAINT)**

Minor Type = 1 Mainframe Configuration (MLMFCONFIG)
(B 7000 only)

Words 0-3 See Table 15-5-2.

Word 4 MCP and system identification.

[47:12] system serial number.
[35:12] MCP mark number.
[23:12] MCP level number.
[11:12] MCP patch number.

Word 5 Pointer to MCP name.

[47: 1] =1
[39:20] length of MCP name in words.
[19:20] =6 index to beginning of MCP name.

Words 6-N MCP name (standard form name)
N= 5 + length of MCP name in words.

Word N+1 PROCMASK
bit mask of all processors in this partition.

Word N+2 IOMMASK
bit mask of all IOMs in this partition.

Word N+3 DCPMASK
bit mask of all DCPs configured in current NDL.

Word N+4 MCMMASK
bit mask of all MCMs in this partition.

Words N+5 to N+12 MCMINFO (one word for each MCM):

| | | |
|---|---|---|
| [46: 2] | MSUAVF | MSU availability. |
| [44: 4] | FRSTATUSF | MCM status. |
| [35: 8] | REQINHIBITF | requestor inhibit. |
| [27: 6] | LOWERF | lower address. |
| [21: 6] | UPPERF | upper address. |
| [15: 4] | MSUSTATUS | MSU status. |
| [ 3: 4] | MCMF | MCM number. |

Word N+13 Number of DFO entries.

Table 15-5-4. Log-Entry Formats (Continued)

| | | | |
|---|---|---|---|
| Words N+14 to end | DFO entry (one word for each DFO entry): | | |
| | [39: 8] | | unit number from UNITCARDS [unit number]. |
| | [47: 1] | INFOPARTBIT | unit is in this partition. |
| | [31: 8] | INFOFUDF | first unit on exchange. |
| | [23: 8] | INFONUDF | next unit on exchange. |
| | [15: 1] | INFOENDBIT | no longer used. |
| | [14: 1] | INFODFOBIT | disk is serviced by a DFO. |
| | [13: 1] | INFOEXCHBIT | unit is on an exchange. |
| | [12: 5] | INFOCHANNELF | base channel number. |
| | [ 7: 8] | INFOIOMMASK | mask of IOMs servicing this unit. |

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 2 Maintenance Entry (LOGMAJMAINT) (B 7000 only)**

Minor Type = 2 Peripheral Configuration (MLPERIPHCONFIG)

| | |
|---|---|
| Words 0-3 | See Table 15-5-2. |
| Words 4-N | See Table 15-5-4.<br>Maintenance entry (major type = 2).<br>Mainframe configuration entry (minor type = 1). |
| Words N+1 to end | Unit Entry (one word for each unit)<br>constructed from UNITCARDS array. |

If this unit is a DFO

[47:48] = 0

All other units:

| | | |
|---|---|---|
| [47: 1] | INFOPARTBIT | unit is in this partition. |
| [39: 2] | INFOLASTCHANF | low-order two bits of the last channel on exchange. |
| [37: 6] | INFOUNITTYPE | hardware unit type. |
| [31: 8] | INFOFUDF | first unit on exchange. |
| [23: 8] | INFONUDF | next unit on exchange. |
| [15: 1] | INFOENDBIT | no longer used. |
| [14: 1] | INFODFOBIT | unit is serviced by a DFO. |
| [13: 1] | INFOEXCHBIT | unit is on an exchange. |
| [12: 5] | INFOCHANNELF | base channel number. |
| [ 7: 8] | INFOIOMMASKF | mask of IOMs servicing this unit. |

If this unit is serviced by a DFO, [14: 1] = 1.

| | |
|---|---|
| [31: 8] | unit number of primary DFO for this unit. |
| [46: 8] | unit number of secondary DFO for this unit. |

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 2 Maintenance Entry (LOGMAJMAINT)**

Minor Type = 3 Maintenance Log Comment Entry (MLCOMMENT)
(B 7000 only)

This log-entry type is not currently used.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 2 Maintenance Entry (LOGMAJMAINT)**

Minor Type = 4 Memory Error Entry (MLMEMACCESS)
(B 7000 only)

| | |
|---|---|
| Words 0-3 | See Table 15-5-2. |
| Word 4 | MCP and system identification.<br><br>[47:12] system number.<br>[35:12] MCP mark number.<br>[23:12] MCP level number.<br>[11:12] MCP patch number. |
| Word 5 | Time, in units of 2.4 usec, when first error occurred. (Memory access errors are accumulated internally until some maximum error count is reached, at which time a log record is written.) |
| Word 6 | Date, in binary Julian, when first error occurred. |
| Word 7 | Time, in units of 2.4 usec, when last error occurred. |
| Word 8 | Date, in binary Julian, when last error occurred. |
| Word 9 | Memory access error count for IOM 0. |
| Word 10 | Memory access error count for IOM 1. |
| Word 11 | Memory access error count for IOM 2. |
| Word 12 | Memory access error count for IOM 3. |
| Word 13 | Memory access error count for IOM 4. |
| Word 14 | Memory access error count for IOM 5. |
| Word 15 | Memory access error count for IOM 6. |
| Word 16 | Memory access error count for IOM 7. |

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 2 Maintenance Entry (LOGMAJMAINT)**

Minor Type = 5 SCR Maintenance Records (MLSCRRECORD)

| | |
|---|---|
| Words 0-3 | See Table 15-5-2. |
| Word 4 | Number of fixed words in the entry.<br>[39:20] number of fixed words in entry. |
| Word 5 | Fixed word.<br>[39:20] length of area in EBCDIC characters.<br>[19:20] index (in words) into log record of first word of specific record. |

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 2 Maintenance Entry (LOGMAJMAINT)**

Minor Type = 9 I/O Condition Entry (MLIOCONDITION)
Minor Type = 10 I/O Error Entry (MLIOERROR)

Words 0-3 See Table 15-5-2.

Word 4 MCP and system identification.

[47:12] system number.
[35:12] MCP mark number.
[23:12] MCP level number.
[11:12] MCP patch number.

Word 5 Time I/O initiated.

Word 6 Mix number of task.
(Negative stack number if not visible.)

Word 7 Unit table entry. (B 6000 only)

[47: 6] unit type

1 = Diskfile
2 = SPO
3 = Remote
4 = Paper tape reader
5 = Paper tape punch
6 = Buffered line printer
7 = Un-buffered line printer
8 = Readersorter
9 = Card reader
10 = Pseudo (not used)
11 = Card punch
13 = NRZ7 magnetic tape
14 = NRZ9 magnetic tape
15 = PE magnetic tape
17 = Pack
19 = Portfile
25 = Diskette
29 = MTCLUSTER 1
30 = MTCLUSTER 2
31 = MTCLUSTER 3
38 = EBCBUFRDLP
39 = EBCUNBUFRDLP

[41: 1] abnormal power off.
[40: 1] normal power off.
[39: 1] unit initializing.
[35: 1] unit assigned to autobackup.
[34: 1] label has been read.
[33: 1] MTR is doing something to unit.
[32: 1] unit marked marginal by MTBF.
[31: 1] usercode is required for this unit.
[30: 1] unit is assigned to a stack.
[29: 4] the disk queuer # + 1.

Table 15-5-4. Log-Entry Formats (Continued)

| | |
|---|---|
| Word 7 cont. | [25: 1] unit not ready.<br>[24: 1] unit interrupt.<br>[23: 1] unit waiting for path.<br>[22: 1] unit busy.<br>[21: 1] unit is in process.<br>[20: 1] unit in retry.<br>[19: 1] unit locked.<br>[18: 1] unit saved.<br>[17: 3] unit density code (disk, pack, and tape).<br><br>For TAPE<br><br>0 = 800bpi NRZ<br>1 = 556bpi NRZ<br>2 = 200bpi NRZ<br>3 = 1600bpi PE<br>4 = 6250bpi PE<br><br>For PACK<br><br>0 = 215<br>1 = N215<br>2 = 225<br>3 = 235<br>4 = 206 sequential<br>5 = 206 interlaced<br>6 = 207<br><br>For DISK<br><br>0 = IA-2<br>1 = IC-3<br>2 = 5N<br>3 = IC-4<br>4 = IIB-2<br>5 = IIB-6<br>6 = IIB-4<br><br>[17: 1] traintable of unit loaded.<br>[16: 2] train printer type.<br><br>0 = 450/750 lpm.<br>1 = 1100 lpm.<br><br>[14: 1] marked scratch.<br>[13: 1] unit assigned to a file.<br>[12: 1] unit is labelled.<br>[11: 1] unit is in rewind.<br>[10: 1] closed with no rewind.<br>[ 9: 1] unit is to be purged.<br>[ 8: 1] write ring is present.<br>[ 7: 1] unit in use by control card.<br>[ 6: 1] closed with error outstanding.<br>[ 5: 5] unit path group. |

Table 15-5-4. Log-Entry Formats (Continued)

Word 7 Unit table entry. (B 7000 only)

[47: 6] unit type values.

(Format is the same as the B 6000 entry)

[41: 1] halt/load unit.
[40: 1] unit using new firmware.
[39: 1] unit uninitialized (disk or pack).
[38: 1] seek ordering (pack).
[38: 1] DFO used (disk).
[35: 1] unit assigned to autobackup (printer or punch).
[35: 1] unit looked at by verify family (disk or pack).
[34: 1] label has been read.
[33: 1] MTR using the unit.
[33: 1] unit volumed (disk or tape).
[32: 1] unit marked marginal by MTR.
[31: 1] usercode is required for this unit.
[30: 1] unit assigned to a stack.
[29: 1] unit assigned to DFO (disk).
[27: 2] secondary DFO number (disk).
[26: 1] unit closed with no rewind (tape).
[25: 1] unit is not ready.
[24: 1] error on unit.
[23: 1] unit to be blasted.
[22: 1] unit marked hung.
[20: 1] unit undergoing retry (all units except disk and halt/load pack).
[19: 1] unit is locked.
[18: 1] unit is saved.
[17: 3] unit density code (disk,pack,tape).
[17: 1] train table of unit loaded.
[16: 2] train printer type (speed).
[14: 1] unit is scratch.
[13: 1] unit is assigned to file.
[12: 1] unit has a label.
[11: 1] unit in rewind.
[11: 1] unit has duplicate serial number (pack).
[10: 1] unit is closed with no rewind.
[10: 1] unit has been complemented (disk or pack).
[ 9: 1] unit was closed with purge.
[ 8: 1] unit has write ring (disk,pack,tape).
[ 7: 1] unit in use by controlcard.
[ 6: 1] unit closed with error.
[ 5: 1] unit is closed (pack).
[ 4: 1] unit buffer locking disabled (pack).
[ 3: 1] I/O queue of unit to be restarted.
[ 2: 1] use in line retry action.
[ 0: 1] unit had path at halt/load.

Table 15-5-4. Log Entry Formats (Continued)

Word 8 User IOCB (contains unit number). The RD.RDUNITNO contains the MPX which did the job. The IOSTUFF. UNITNUMBERF contains the unit number. (B 6000 only)

```
[47:10] SNR of owner.
[37: 1] counting user I/Os.
[36: 1] log I/O exception errors
[31: 8] hardware unit number.
[19: 5] designated MPX route.
[13: 1] library maintenance bit.
[12: 1] used for read check after each line.
[10: 1] buffer one used in parity retry.
[ 9: 1] on, if I/O is in process.
[ 8: 1] badly blocked disk file.
[ 7: 1] user special I/O.
[ 6: 1] I/O bit.
[ 4: 1] direct I/O bit.
[ 3: 1] I/O error recovery in process.
[ 2: 1] error recovery failed.
[ 1: 1] I/O ran into EOF or EOT.
[ 0: 1] I/O finished.
```

Word 8 User word of IOCB. (B 7000 only)

```
[47:10] SNR of owner.
[37: 1] counting user I/Os.
[36: 1] log I/O exception errors.
[35: 1] bypass retry.
[34: 1] this I/O is for writing to SUMLOG.
[31: 8] physical unit number.
[23: 1] IOCB [IOMAREADESC] word is valid.
[22: 1] IOCB [IOMIOCW] word is valid.
[21: 1] IOCB [IOMCDL] word is valid.
[20: 1] change in direction pending for pack I/O
        queue or software attention is set to
        denote I/O queue empty condition.
[19: 1] user has requested software attention.
[18: 1] this I/O is for read extended status.
[17: 1] alternate path retry in effect.
[16: 1] this I/O is an MCP special I/O;
        no exception handling required.
[ 9: 1] this I/O is in process.
[ 8: 1] badly blocked disk.
[ 7: 1] over the row/truncated direct I/O.
[ 6: 1] this is user I/O.
[ 5: 1] unit was "blasted".
[ 4: 1] this is direct I/O.
[ 3: 1] this I/O is for I/O error recovery.
[ 2: 1] last I/O was fatal
[ 1: 1] end of file or tape.
[ 0: 1] I/O not in process.
```

Word 9 I/O Area Descriptor.

Word 10 IOCW (last for disk, disk pack, and tape file).

Word 11 I/O result descriptor (last for disk, disk pack, and tape file) bearing MPX number in unit number field [24: 8].

Table 15-5-4. Log Entry Formats (Continued)

Words 12-14 Twelve 12-bit pointers (entry-relative) to unit dependent information. (See next word.)

Word 15 Detailed result descriptor. (B 6000 only)

[39:20] length, including typeword.
[19:20] subtype field.

Word 16 to end. Miscellaneous file information. The first key word in each block of information contains two fields which give the type (field [19:20]) and length (field [39:20]) of the block in words (including the key word). The type field indicates the sort of information following the key word of the block. This information is pointed to by words 12-14.

1 = Key word is followed by four words of LABEL information including serial number. Words 1 through 4 broken down respectively:

Word 1 [47: 9] label level.
[38:15] cycle number.
[23: 8] generation number.
[15: 1] reel is specified.
[14:15] reel number.

Word 2 [27:11] save factor.
[16:17] creation date (tape only).

Word 3 UNUSED.

Word 4 SERIAL NO. for six ebcdic characters.

2 = Key word is followed by file title in standard form.

3 = Key word is followed by the job name, in standard form.

4 = Key word is followed by the tape file information in word triples of IOCW, result descriptor, and I/O start time.

5 = Key word is followed by the disk file information in word triples of IOCW, result descriptor, and I/O start time.

6 = Buffer contents: reader check.

7 = INTNAME.

8 = New tape RDs.

9 = Extended RDs.

10 = B 6800 channel memory.

11 = Diskette retry information.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 2 Maintenance Entry (LOGMAJMAINT)**

Minor Type = 11 Library Maintenance Compare (MLLIBERR)

| | |
|---|---|
| Words 0-3 | See Table 15-5-2. |
| Word 4 | Source information word, consisting of:<br>[41: 1] disk flag (0 = disk).<br>[40: 1] tape flag (1 = tape).<br>[39: 8] unit number.<br>[31:12] first word in source on which compare failed.<br>[19:20] disk address. |
| Word 5 | First source word on which the compare failed. |
| Word 6 | Destination information word. (See word 4.) |
| Word 7 | First destination word on which compare failed. |

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 2 Maintenance Entry (LOGMAJMAINT)**

Minor Type = 12 Mainframe Error Entry (MLMAINFRAMERR)
(B 7000 only)

Words 0-3 See Table 15-5-2.

Word 4 MCP and system identification.

[47:12] system number.
[35:12] MCP mark number.
[23:12] MCP level number.
[11:12] MCP patch number.

Word 5 Time of error.

Word 6 System configuration.

[31: 8] CPM box numbers present.
[23: 8] IOM box numbers present.
[15: 8] MCM box numbers present.
[ 7: 8] DFO box numbers present.

Word 7 MCMS 0-2

This word contains three 16 bit entries, that is, the entry for MCM 0 is contained in [47:16]. The data contained in that entry would be:

[47: 6] MCM lower address limit.
[41: 6] MCM upper address limit.
[40: 4] MSU status.
[40: 1] = 1 then MSU #4 is available.
[39: 1] = 1 then MSU #3 is available.
[38: 1] = 1 then MSU #2 is available.
[37: 1] = 1 then MSU #1 is available.

Word 8 MCMS 3-5

(Same format as word 7.)

Word 9 MCMS 6-7

(Same format as word 7.)

Word 10 Mix number and SNR. (Processor interrupts only.)

[45:10] SNR.
[31:16] job number.
[15:16] task number.

Word 11 Fail register.

Word 12 Unused.

Table 15-5-4. Log-Entry Formats (Continued)

Word 13 Return control word.

Word 14 Fault entry information.

[15: 4] Box type.

1 = CPM
2 = IOM
3 = MCM

[11: 4] Box number.
[ 7: 8] Fault type.

Word 15 P1 interrupt word.

Word 16 P2 interrupt word.

Word 17 P2 interrupt word tag.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 2 Maintenance Entry (LOGMAJMAINT)**

Minor Type = 13 Disk Pack Error Entry
(B 7000 only)

The format of this record is unavailable at this time.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 2 Maintenance Entry (LOGMAJMAINT)**

Minor Type = 14 DFO Error Entry (MLDFOERROR)

| | |
|---|---|
| Words 0-3 | See Table 15-5-2. |
| Word 4 | Number of DFO scanouts. |
| Word 5 | Bad scanin result. |
| Word 6 | Scanout words. |

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 2 Maintenance Entry (LOGMAJMAINT)**

Minor Type = 15 B6800 Abort and Diagonostics (MLCPUERROR)

Words 0-3 See Table 15-5-2.

Word 4 MCP and system identification.

[47:12] system serial number.
[35:12] MCP mark number.
[23:12] MCP level number.
[11:12] MCP patch number.

Word 5 Processor number (WHOI number).

Word 6 P1 interrupt parameter with tag O.

[25: 1] alarm interrupt.

[21: 1] error using memory port B.
[10: 1] scan in information error.
[11: 1] scan out information error.
[ 7: 1] multiple data bit error.
[ 6: 1] store address residue.
[ 5: 1] invalid program word.
[ 3: 1] invalid address local.
[ 2: 1] scan bus parity.
[ 1: 1] store address parity.

[26: 1] hardware error.

[ 4: 1] compare residue.
[ 3: 1] address residue.
[ 2: 1] bus residue.
[ 1: 1] ram card parity.
[ 0: 1] prom card parity.

[22: 1] general control interrupt.

[ 5: 1] address retry.
[ 4: 1] read data recheck bit.
[ 3: 1] read data retry.
[ 0: 1] single bit error.

Table 15-5-4. Log-Entry Formats (Continued)

| | |
|---|---|
| Word 7 | P3 interrupt parameter with tag 0.<br><br>[47:20] memory address.<br>[31: 8] card number if hardware error.<br>(memory address and residue unused).<br>[26: 3] residue of address<br>[23: 6] check bits for general control interrupt.<br>(following fields unused).<br>[23: 1] vector mode interrupt.<br>[22: 1] in table mode.<br>[21: 1] in edit mode.<br>[20: 1] in variant op.<br>[19: 4] family strobe.<br>[15: 4] opcode within family strobe.<br>[11: 1] 1 = j-count, 0 = micro address.<br>[10:11] j-count or micro address. |
| Word 8 | P2 word tag. |
| Word 9 | P2 interrupt parameter with tag 0. |

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 2 Maintenance Entry (LOGMAJMAINT)**

Minor Type = 16 MLIP I/O Error Entry

This minor type is a MLIP I/O error record and is the MLIP counterpart to the MPX I/O error record with a minor type of 10. The first four words of the MLIP I/O error record contain the standard information found in any log record. The rest of the MLIP log record is different from the MPX log record.

Words 0-3 See Table 15-5-2.

Word 4 MLIP LOG TYPE

[47: 8] MLIP LOG TYPE

1 = this is an error-recovery log record.
2 = this is an error-IOCB log record.

For MLIP LOG TYPE = 1 (error recovery record):

Word 4 Retry information.

[39: 8] MLIP LOG VARIANT

1 = Log record is for normal error recovery and all information for a retry is present.

2 = Log record is for limited error recovery and some of the information for each retry is not present. The RETRY DLP COMMAND and the RETRY MLIP CONTROL are not present.

[27:12] NUMBER OF RETRIES which were logged.

[15:01] IRRECOVERED.

0 = the I/O recovered.
1 = the I/O was irrecoverable.

Word 5 UNIT information.

[47:12] logical unit number.
[35:12] physical unit number.
[23: 6] logical unit type.
[17: 6] unit subtype.
[11:12] physical unit type.

Table 15-5-4. Log-Entry Formats (Continued)

Word 6 File and job information.

[47: 4] Number of items of file information and job information logged. File information includes the file title, the internal name, and the serial number. Job information is the job name.

[11:12] File and job information location - the number of the word in the log where the file and job information starts.

The next eighteen words of the log record contain information about the original I/O.

Word 7 MLIP control word.

[19:20] MLIP CONTROL.

Word 8 DLP address.

Word 9

Word 10

Word 11

Word 12 MLIP state and result.

Word 13 I/O START time.

Word 14 STACK numbers.

[45:10] STACK number of IOCB owner.
[33:10] STACK number of IO initiator.

Word 15

Word 16 I/O control word.

Word 17 AREA descriptor.

[47:20] memory address of start of area.
[27:20] length of area.

Word 18

Word 19 Logical result descriptor.

Word 20

Word 21 DLP command.

Word 22 DLP result (Word 1).

Word 23 DLP result (Word 2).

Table 15-5-4. Log-Entry Formats (Continued)

Word 24-33 Extended result descriptor area. 10-word results of reading extended result descriptor.

Word 34 I/O statistics (Word 1).

[47:24] total read I/Os to this unit since Halt/Load.

[23:24] total write I/Os to this unit since Halt/Load.

Word 35 I/O statistics (Word 2).

[47:24] total read errors for unit since Halt/Load.

[23:24] total write errors for unit since Halt/Load.

Word 36-N RETRY entries.

This section contains retry entries of five words each. The number of retries (Word 4) which were logged is variable, so this section is of variable length. The format of each entry is as follows:

Entry Word 0: Retry DLP command (not present if LOG VARIANT=2).

Entry Word 1: Retry DLP result (Word 1).

Entry Word 2: Retry DLP result (Word 2).

Entry Word 3:

[47: 1] actual retry.

0 = Entry is not actual attempt to retry I/O (for example, repositioning a tape).
1 = Entry is an actual attempt to retry I/O.

[19:20] RETRY MLIPCONTROL (not present if LOG VARIANT is 2).

Entry Word 4: Retry MLIP STATE and RESULT.

Table 15-5-4. Log-Entry Formats (Continued)

Word N+1 to end — File and job information.

This section contains file and job information. Because the length of the previous section varies, the starting word of this section is also variable and is the value of file and job information location in the control section (Word 6). The number of file and job information items (Word 6) is variable, as is the length of each item, so this section is of variable length. The format of each item is identical to the file and job information items of the MPX I/O error log records. The first word of each item specifies the type and length of the item.

Item Word 0 - Item descriptor.

[39:20] length of item (including this word).

[19:20] TYPE of item.

1 = Item is SERIAL NUMBER information.
2 = Item is FILE TITLE information.
3 = Item is JOB NAME information.
7 = Item is INTERNAL NAME information.

If TYPE = 1 (SERIAL NUMBER):

Item Word 1: LEBCONTROL.
Item Word 2: GENEALOGY1.
Item Word 3: GENEALOGY2.
Item Word 4: SERIAL NUMBER in EBCDIC. (Present only if the unit is a pack, mag tape, or diskette.)

If TYPE = 2 (FILE TITLE):

Item Word 1: ROW INDEX of the file on pack or disk. This word is all ones if no valid row index is present or if the unit is not a pack or disk.

Item Word 2-n: If the length of the FILE TITLE item is greater than two, the rest of the item is the file title in standard form.

If TYPE = 3 (JOB NAME):

Item Word 1-n: The job name in standard form.

If TYPE = 7 (INTERNAL NAME):

Item Word 1-n: The internal name in standard form.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 2 Maintenance Entry (LOGMAJMAINT)**

Minor Type = 17 Hardware Configuration (HARDWARE-CONFIG)

The hardware configuration record is a tree-structured record that uses the LINK structure. One LINK is at the top of the tree; this LINK describes a region that contains five LINKs. Each of these LINKs describes a region that contains information about data processors, I/O processors, memory, or peripherals.

The direct portion of the hardware configuration record consists of the following:

Words 0-3 See Table 15-5-2.

Word 4 General information.

[47: 8] Number of data processors in system.
[39: 8] Number of I/O processors in system.
[31: 8] System type.

1 = B 6700
2 = B 6800
3 = B 6900
4 = B 5900
11 = B 7700
12 = B 7800
13 = Mixture of B 7700s and B 7800s

[23:24] Reserved.

Word 5 LINK to the hardware information.

All of the actual hardware configuration information resides in the indirect portion of the configuration record. Accessing any of this information requires that the LINK at word 5 be examined.

The LINK at word 5 contains the following values:

LINK-TYPE = 1
LINK-VERSION = 0
ITEM-COUNT = 5 (LINKs)
ITEM-WIDTH = 1 (word)

The region described by the LINK at word 5 has the following format:

| Offset | Contents | LINK-TYPE |
|---|---|---|
| 0 | LINK to data processor information | 2 |
| 1 | LINK to I/O processor information | 3 |
| 2 | LINK to memory information | 4 |
| 3 | LINK to peripheral information | 5 |
| 4 | LINK to extended pack information | 6 |

The value contained in each LINK in the preceding table and the format of the information corresponding to each LINK are given in the following paragraphs. The array indices given for the preceding LINKs are not absolute indices; they are relative to the START-INDEX of the LINK at word 5. The notation [i] is defined to mean the START-INDEX of the LINK at array index "i". For example, the LINK to memory information is found at array entry [5] + 2.

LINK [5] + 0: DATA PROCESSOR INFORMATION

This LINK has the following values:

```
LINK-VERSION = 0.
LINK-TYPE    = 2.
ITEM-COUNT   = number of data processors.
ITEM-WIDTH   = 5 (words).
```

Each data processor entry has the following format:

```
Entry Word 0 : PROCESSOR-NUMBER.

Entry Word 1 : PROCESSOR-TYPE.

     1 = B 6700
     2 = B 6800
     3 = B 6900
     4 = B 5900
    11 = B 7700
    12 = B 7800
    13 = Either B 7700 or B 7800

Entry Word 2 : PROCESSOR-GMM-PORT-ADDRESS
               (four characters, left-justified).

Entry Word 3 : PROCESSOR-FIRMWARE-LEVEL.

Entry Word 4 : PROCESSOR-IOP-MASK.

               I/O processors that this data
               processor can see; bit "i" is ON
               if this data processor can see
               I/O processor number "i".
```

LINK [5] + 1 : I/O PROCESSOR INFORMATION

This LINK has the following values:

```
LINK-VERSION = 0.
LINK-TYPE    = 3.
ITEM-COUNT   = number of I/O processors.
ITEM-WIDTH   = 4 (words).
```

Each I/O processor entry has the following format:

Entry Word 0 : IOP-NUMBER.

Entry Word 1 : IOP-TYPE.

```
1 = Type 1 or 2 MPX
2 = Type 3 or 4 MPX
3 = IOM
4 = MLIP
```

Entry Word 2 : IOP-FIRMWARE-LEVEL.

Entry Word 3 : IOP-PROCESSOR-MASK.

Data processors that this I/O processor can see; bit "i" is ON if this I/O processor can see data processor number "i".

LINK [5] + 2 : MEMORY INFORMATION

This LINK has the following values:

```
LINK-VERSION = 0.
LINK-TYPE    = 4.
ITEM-COUNT   = number of memory chunks.
ITEM-WIDTH   = 5 (words).
```

Each memory entry has the following format:

Entry Word 0 :

```
[01:01]  GLOBAL-MEMORY.
         All processors can see this memory.
[00:01]  WRITE-OK.
```

Entry Word 1 : MEMORY-START-ADDRESS.

Entry Word 2 : MEMORY-END-ADDRESS.

Entry Word 3 : MEMORY-PROCESSOR-MASK.

Data processors that can see this memory; bit "i" is ON if I/O processor number "i" can see this memory.

In some cases, two entries may be made for the same chunk of Global (tm) Memory*. Two entries are made if one data processor has read-only access to some part of Global (tm) Memory. One entry has GLOBAL-MEMORY and WRITE-OK SET, and the data processor mask is SET to show all processors except the one with read-only access. Another entry has GLOBAL-MEMORY SET, WRITE-OK RESET, and the data processor mask SET to show the processor with read-only access to that area of memory.

LINK [5] + 3 : PERIPHERAL INFORMATION

This LINK has the following values:

```
LINK-VERSION = 0.
LINK-TYPE    = 5.
ITEM-COUNT   = Number of units.
ITEM-WIDTH   = 2 (words).
```

Each peripheral information entry has the following format:

Entry Word 0 : Peripheral unit information.

```
[47:12]  PHYSICAL-UNIT-NUMBER
[35:08]  PERIPHERAL-TYPE
[27:08]  PERIPHERAL-SUBTYPE
[19:04]  PERIPHERAL-DENSITY
[15:01]  PERIPHERAL-SAVED
[14:01]  PERIPHERAL-URED
         Unit is reserved.
[13:01]  PERIPHERAL-SEEN-RY
         Unit has been seen ready since the
         last Halt/Load.
[12:01]  PERIPHERAL-AVAIL
         Unit is available to group.
[11:12]  Reserved.
```

Entry Word 1 : LINK to path information.

---

*Global (tm) memory is a trade mark of the Burroughs Corporation.

PERIPHERAL-SEEN-RY is SET if the unit has been ready at any time since the last Halt/Load, even if the device is not currently ready.

PERIPHERAL-AVAIL is SET if the peripheral is currently available to the group. In a loosely-coupled system, paths to a device from two or more groups may be present. In this case, one of these groups "acquires" the device, and the device becomes available to that group. The other group(s) may have a valid path to the device, but that device is not available to those groups.

The density and subtype information reported for printer, pack, and disk devices is based on the information in the unit table of the MCP. The unit table is built from information in the configuration file. If a printer, pack, or disk has not been seen as ready, its density and subtype often may not be correct.

Any one of three different types of LINK may appear at offset 1 in a peripheral entry. These three different LINK-TYPES correspond to the three different types of paths that exist on B 5000/B 6000/B 7000 series systems.

Multiplexor Path

Multiplexor path information is described by a LINK with the following values:

```
START-INDEX  = 0.
LINK-VERSION = 0.
LINK-TYPE    = 7.
ITEM-COUNT   = Number of paths.
ITEM-WIDTH   = 2 (words).
```

Each Multiplexor path entry has the following format:

Entry Word 0 :

```
[47:08]  PATH-ID.
[39:08]  MPX-NUMBER.
[31:06]  PATH.
           Multiplexor path number (0-3).
[25:01]  PATH-RESERVED.
[24:25]  Reserved.
```

Entry Word 1 : FIRMWARE-LEVEL
(4 hex digits, right-justified).

I/O Module Path

IOM path information is described by a LINK with the following values:

```
START-INDEX  = 0.
LINK-VERSION = 0.
LINK-TYPE    = 8.
ITEM-COUNT   = Number of paths.
ITEM-WIDTH   = 2 (words).
```

Each IOM path has the following format:

Entry Word 0 :

```
[47:08]  PATH-ID.
[39:08]  IOM-NUMBER.
[31:06]  CHANNEL-NUMBER.
[25:01]  CHANNEL-RESERVED.
[24:25]  Reserved.
```

Entry Word 1: FIRMWARE-LEVEL
(4 hex digits, right-justified).

Message Level Interface Port Path

MLIP path information is described by a LINK with the following values:

```
START-INDEX  = 0.
LINK-VERSION = 0.
LINK-TYPE    = 9.
ITEM-COUNT   = Number of paths.
ITEM-WIDTH   = 2 (words).
```

Each MLIP path entry has the following format:

Entry Word 0 :

| | |
|---|---|
| [47:08] | PATH-ID. |
| [39:08] | MLIP-NUMBER. |
| [31:04] | MLIP-PORT-NUMBER. |
| [27:04] | LEM-PORT-NUMBER. |
| [23:04] | RELATIVE-DLP (behind the base). |
| [19:04] | RELATIVE-UNIT (behind the DLP). |
| [15:12] | DLP-IO. |
| [03:01] | PATH-RESERVED. |
| [02:03] | Reserved. |

Entry Word 1 : FIRMWARE-LEVEL (4 hex digits, right-justified).

LINK [5] + 4 : EXTENDED PACK INFORMATION

This LINK has the following values:

| | |
|---|---|
| LINK-VERSION | = 0. |
| LINK-TYPE | = 6. |
| ITEM-COUNT | = Number of packs on-line. |
| ITEM-WIDTH | = 5 (words). |

Each extended pack entry has the following format:

Entry word 0 :

| | |
|---|---|
| [47:12] | PHYSICAL-UNIT-NUMBER. |
| [35:08] | FAMILY-INDEX-NUMBER. |
| [27:01] | MODE (IN = 0, I/O = 1). |
| [26:27] | Reserved. |

Entry Word 1 : SERIAL-NUMBER (in EBCDIC characters).

Entry Word 2 :

| | |
|---|---|
| [47:08] | Number of characters in the pack label. |
| [39:40] | First five characters of the pack label. |

Entry Word 3 : Next six characters of the pack label.

Entry Word 4 : Last six characters of the pack label.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 2 Maintenance Entry (LOGMAJMAINT)**

Minor Type = 18 Software Configuration (SOFTWARE-CONFIG)

The software configuration record is a tree-structured record that uses the LINK structure. One LINK is at the top of the tree; this LINK describes a region that contains eight LINKs. Each of these LINKs describes a region that contains information about the software configuration. In some cases, these regions contain LINKs to other regions.

The direct portion of the software configuration record consists of the following:

Words 0-3 See Table 15-5-2.

Word 4 General information.

[47: 8] Number of data processors in system.
[39: 8] Number of I/O processors in system.
[31: 8] System type.

1 = B 6700
2 = B 6800
3 = B 6900
4 = B 5900
11 = B 7700
12 = B 7800
13 = Mixture of B 7700s and B 7800s

[23:24] Reserved.

Word 5 LINK to the software information.

All the actual software configuration information resides in the indirect portion of the software configuration record. Accessing any of this information requires that the LINK at word 5 be examined.

The LINK at word 5 contains the following values:

LINK-VERSION = 0.
LINK-TYPE = 1.
ITEM-WIDTH = 1.
ITEM-COUNT = 8 (LINKs).

The region described by the LINK at word 5 has the following format:

| Offset | Contents | LINK-TYPE |
|---|---|---|
| 0 | LINK to general MCP information | 2 |
| 1 | LINK to support library information | 3 |
| 2 | LINK to disk location information | 4 |
| 3 | LINK to substitute backup information | 5 |
| 4 | LINK to memory management parameter information | 6 |
| 5 | LINK to subsystem information | 7 |
| 6 | LINK to queue information | 8 |
| 7 | LINK to SWAPPER parameter information | 9 |

The values contained in each LINK in the preceding table and the format of the information corresponding to each LINK are given in the following paragraphs.

All LINKs that describe an item in length-string form have the following values:

```
LINK-VERSION = 0.
LINK-TYPE    = 254.
ITEM-COUNT   = 1 (name).
ITEM-WIDTH   = number of characters used to
               store this string.
```

All LINKs that describe an item in standard form have the following values:

```
LINK-VERSION = 0.
LINK-TYPE    = 255.
ITEM-COUNT   = 1 (name).
ITEM-WIDTH   = number of characters used to
               store this name.
```

LINK [5] + 0 : GENERAL MCP INFORMATION

This LINK has the following values:

```
LINK-VERSION = 0.
LINK-TYPE    = 2.
ITEM-COUNT   = 14.
ITEM-WIDTH   = 0 (heterogeneous).
```

Each entry in this region has the following format:

```
Entry Word 0 : LINK to Halt/Load and backup MCP
               unit information.

Entry Word 1 : MCP version.

  [47:16]  MARK-LEVEL.
  [31:16]  CYCLE.
  [15:16]  PATCH-NUMBER.
```

```
Entry Word 2 : MCP compilation date and time
               (timestamp).

  [47:16] date (Julian - 700000).
  [35:32] time (2.4 microsecond
          increments) DIV 16.

Entry Word 3 : MCP "compiled by" information.

  [23:08]  MARK-LEVEL (of compiler).
  [09:10]  CYCLE (of compiler).

Entry Word 4 : System serial number.

Entry Word 5 : LINK to MCP compile-time options.

    LINK-VERSION = 0.
    LINK-TYPE    = 14.
    ITEM-COUNT   = 1.
    ITEM-WIDTH   = Number of characters.

    Each option is separated from the next by a
    null byte (48 "00"), and the list is
    terminated by two null bytes in succession.

Entry Word  6 : LINK to GROUP ID
                (length-string form).

Entry Word  7 : CATALOG-LEVEL.

Entry Word  8 : OPTIONS word.

                The option numbers correspond
                to the bit numbers in this
                word.

Entry Word  9 : LINK to NIF name (standard form).

Entry Word 10 : LINK to CONFIGURATION FILE name
                (standard form).

Entry Word 11 : LINK to SUPERVISOR name
                (standard form).

Entry Word 12 : LINK to SYSTEM INTRINSICS name
                (standard form).

Entry Word 13 : Mask specifying condition under
                which IOMs interrupt CPMs
                (B 7000 series systems only).

  [00:01]  IDLE.
  [01:01]  IOFINISH.
  [02:01]  WAITING.
  [03:01]  QUEUEEMPTY.
```

HALT/LOAD AND BACKUP MCP UNIT INFORMATION

This region is used to list each unit that is either a Halt/Load unit or an MCP backup unit. For each unit listed, the name of the MCP code file on that unit is given. In addition, a mask of all processors that use this unit as a Halt/Load unit is included. If this mask is non-zero, the unit is a Halt/Load unit and may have up to two MCP backup units listed in the appropriate fields. If this mask is zero, the unit is a backup MCP unit, and the backup unit fields have no meaning.

The LINK to HALT/LOAD and backup MCP unit information has the following values:

```
LINK-VERSION =  0.
LINK-TYPE    = 10.
ITEM-COUNT   = Number of Halt/Load units +
               number of backup MCP units.
ITEM-WIDTH   = 3 (words).
```

Each entry in this region has the following format:

Entry Word 0 : Unit word.

```
[47:12]  Physical unit number.
[35:12]  Physical unit number of BACKUP-UNIT-1
         (not applicable if word 1 is 0).
[23:12]  Physical unit number of BACKUP-UNIT-2
         (not applicable if word 1 is 0).
```

Entry Word 1 : Halt/Load processor mask
(bit SET for each processor that uses this unit as its Halt/Load unit).

Entry Word 2 : LINK to name of MCP on this unit
(standard form).

LINK [5] + 1 : SUPPORT LIBRARY INFORMATION

This LINK has the following values:

```
LINK-VERSION = 0.
LINK-TYPE    = 3.
ITEM-COUNT   = Number of libraries.
ITEM-WIDTH   = 4 (words).
```

Each entry in this region has the following format:

Entry Words 0-2 : LIBRARY-ID (function name, length-string form).

Entry Word 3 : LINK to library code file name
(standard form).

LINK [5] + 2 : DISK LOCATION INFORMATION

The LINK to this region has the following format:

```
LINK-VERSION = 0.
LINK-TYPE    = 4.
ITEM-COUNT   = 7.
ITEM-WIDTH   = 1 (words).
```

Each entry in this region has the following format:

```
Entry Word 0 : LINK to CATALOG family
               (length-string form).

Entry Word 1 : LINK to JOBS family
               (length-string form).

Entry Word 2 : LINK to USERDATA family
               (length-string form).

Entry Word 3 : LINK to BACKUP family
               (length-string form).

Entry Word 4 : LINK to LOG family
               (length-string form).

Entry Word 5 : LINK to GLOBAL family
               (length-string form).

Entry Word 6 : LINK to processor overlay
               families.
```

The LINK to this region has the following values:

```
LINK-VERSION = 0.
LINK-TYPE    = 11.
ITEM-COUNT   = Number of boxes.
ITEM-WIDTH   = 4 (words).
```

Each entry for an indiviual queue in this region has the following format:

```
Entry Word 0 : Box number.

  0 = GLOBAL box.
  1 = LOCAL 1.
  2 = LOCAL 2.
  .
  .
  .

Entry Word 1-3 : Overlay family
                 (length-string form).
```

LINK [5] + 3 : SUBSTITUE BACKUP INFORMATION

This LINK has the following values:

```
LINK-VERSION = 0.
LINK-TYPE    = 5.
ITEM-COUNT   = 6.
ITEM-WIDTH   = 1 (word).
```

Each entry in this region has the following format:

Entry Word 0 : SB information for DISK.

Entry Word 1 : SB information for PACK.

Entry Word 2 : SB information for TAPE.

Entry Word 3 : SB information for PETAPE.

Entry Word 4 : SB information for TAPE9.

Entry Word 5 : SB information for TAPE7.

Each of the preceding six entry words has the following format:

```
[47:04]  FIRST-DEVICE.
[43:04]  SECOND-DEVICE.
[39:04]  THIRD-DEVICE.
[35:04]  FOURTH-DEVICE.
[31:04]  FIFTH-DEVICE.
[27:04]  SIXTH-DEVICE.
[23:04]  SEVENTH-DEVICE.
[19:20]  Reserved.
```

The devices are given the following numbers:

```
1 = DISK
2 = PACK
3 = TAPE
4 = PETAPE
5 = TAPE9
6 = TAPE7
7 = DLBACKUP
```

LINK [5] + 4 : MEMORY MANAGEMENT PARAMETERS

This LINK has the following values:

```
LINK-VERSION = 0.
LINK-TYPE    = 6.
ITEM-COUNT   = Number of groups of boxes with
               memory management parameters.
ITEM-WIDTH   = 5 (words).
```

Each entry in this region has the following format:

Entry Word 0 : Mask of boxes to which these parameters apply.

```
0 = GLOBAL box.
1 = LOCAL 1.
2 = LOCAL 2.
.
.
.
```

Entry Word 1 : OVERLAY GOAL (percent).

Entry Word 2 : AVAILMIN (percent).

Entry Word 3 : FACTOR (percent).

Entry Word 4 : MEMORY PRIORITY FACTOR (percent).

LINK [5] + 5 : SUBSYSTEM INFORMATION

This LINK has the following values:

```
LINK-VERSION = 0.
LINK-TYPE    = 7.
ITEM-COUNT   = Number of subsystems.
ITEM-WIDTH   = 4 (words).
```

Each entry in this region has the following format:

Entry Words 0-2 : Subsystem name, length-string.

Entry Word 3 : Mask of boxes that are in this subsystem.

```
0 = GLOBAL box.
1 = LOCAL 1.
2 = LOCAL 2.
.
.
.
```

LINK [5] + 6 : QUEUE INFORMATION

```
LINK-VERSION = 0.
LINK-TYPE    = 8.
ITEM-COUNT   = 3.
ITEM-WIDTH   = 0 (heterogeneous).
```

Each entry in this region has the following format:

```
Entry Word 0 : GLOBAL-MIXLIMIT.

Entry Word 1 : DEFAULT-QUEUE.

Entry Word 2 : LINK to individual queue
               information.
```

The LINK to the individual queue information region has the following values:

```
LINK-VERSION = 0.
LINK-TYPE    = 12.
ITEM-COUNT   = Number of queues.
ITEM-WIDTH   = 37 (words).
```

Each entry for an individual queue in this region has the following format:

```
Entry Word     0 : Queue number.

Entry Word     1 : MIXLIMIT.

Entry Word     2 : TASKLIMIT

Entry Word     3 : TURNAROUND (seconds).

Entry Word     4 : SUBSPACES.

Entry Words  5-7 : FAMILY to substitute
                   (for example, DISK)
                   length-string form.

Entry Words 8-10 : FAMILY to substitute
                   (for example, "= MCPMAST")
                   length-string form.

Entry Words 11-13 : FAMILY OTHERWISE
                    (for example, "OTHERWISE
                    SYS33") length-string form.

Entry Words 14-16: SUBSYSTEM
                   (length-string form).

Entry Word    17 : DEFAULT PRIORITY.

Entry Word    18 : DEFAULT IOTIME (seconds).

Entry Word    19 : DEFAULT PROCESSTIME (seconds).
```

Entry Word 20 : DEFAULT LINES.

Entry Word 21 : DEFAULT CARDS.

Entry Word 22 : DEFAULT WAITLIMIT (seconds).

Entry Word 23 : DEFAULT DISKLIMIT (segments).

Entry Word 24 : DEFAULT ELAPSEDLIMIT (seconds).

Entry Word 25 : LIMIT PRIORITY.

Entry Word 26 : LIMIT IOTIME (seconds).

Entry Word 27 : LIMIT PROCESSTIME (seconds).

Entry Word 28 : LIMIT LINES.

Entry Word 29 : LIMIT CARDS.

Entry Word 30 : LIMIT WAITLIMIT (seconds).

Entry Word 31 : LIMIT DISKLIMIT (segments).

Entry Word 32 : LIMIT ELAPSEDLIMIT (seconds).

Entry Word 33 : LIMIT TAPE.

Entry Word 34 : LIMIT TAPE7.

Entry Word 35 : LIMIT TAPE9.

Entry Word 36 : LIMIT TAPEPE.

LINK [5] + 7 : SWAPPER PARAMETERS

LINK-VERSION = 0.
LINK-TYPE = 9.
ITEM-COUNT = Number of boxes with SWAPPER parameters.
ITEM-WIDTH = 18 (words).

Each entry in this region has the following format:

Entry Word 0 : Box number.

0 = GLOBAL box
1 = LOCAL 1
2 = LOCAL 2
.
.
.

Entry Word 1 : SWAPPERSTATUS.

0 = NOT RUNNING
1 = RUNNING

Entry Word 2 : CORESIZE (990-word slots).

```
Entry Word  3 : EXPMAXCORE (990-word slots).
Entry Word  4 : EXPRESSRESERVE (990-word slots).
Entry Word  5 : EXPMAXTIME (seconds).
Entry Word  6 : IOBIAS (percent).
Entry Word  7 : MAXCORE (990-word slots).
Entry Word  8 : MAXIOSIZE (990-word slots).
Entry Word  9 : MAXSLICENUMBER.
Entry Word 10 : MEMORYBIAS (percent).
Entry Word 11 : MINCHUNKSIZE (990-word slots).
Entry Word 12 : MINTIMESLICE (seconds).
Entry Word 13 : PRIORITYBIAS (percent).
Entry Word 14 : RATIO.
Entry Word 15 : UTILIZATIONBIAS (percent).
Entry Word 16 :

  [47:47]  Reserved.
  [00:01]

    0 = NOSWAPTRANSTATE
    1 = SWAPTRANSTATE

Entry Word 17 : LINK to SWAPPER family names used
                by the SWAPPER running in  this
                box.
```

The LINK to the SWAPPER family names region has the following format:

```
LINK-VERSION = 0.
LINK-TYPE    = 13.
ITEM-COUNT   = Number of SWAPPER families.
ITEM-WIDTH   = 3 (words).
```

Each entry in this region has the following format:

```
Entry Words 0-2 : SWAPPER family name
                  (length-string form).
```

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 3 String Entry (LOGMAJSTR)**

Minor Type = 1 RSVP Message Entry (RSVP)
Minor Type = 2 Fatal Error Message Entry (FATAL)
Minor Type = 3 Non-fatal Error Message Entry (NONFATAL)
Minor Type = 4 System Message Entry (SYSTEM)
Minor Type = 5 Unit Message Entry (UNIT)
Minor Type = 6 Directory Message Entry (DIRECTORY)
Minor Type = 7 Display Message Entry (DISPLAY)
Minor Type = 8 Status Message Entry (STATUS)
Minor Type = 9 Unit RSVP Message Entry (UNITRSVP)

| | |
|---|---|
| Words 0-3 | See Table 15-5-2. |
| Word 4 | Length of string, in EBCDIC characters. |
| Word 5 to end. | Character string in EBCDIC characters. |

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 4 MCS Entry (LOGMAJMCS)**

Minor Type = 1 Log-on Entry (MCSLOGON)

| | |
|---|---|
| Words 0-3 | See Table 15-5-2. |
| Word 4 | Log-on priority information. |
| Word 5 | Sign-on class.<br>0 = Unswitched.<br>1 = New log-on.<br>2 = HELLO.<br>3 = New charge code.<br>4 = SPLIT (Session number change).<br>5 = New access code. |
| Word 6 | LINK to station name. |
| Word 7 | LINK to usercode. |
| Word 8 | LINK to charge code. |
| Word 9 | Represents the time of day when last connected. If log-on is a new connect and the word is valid, bit 47 is on. |
| Word 10 | Originator information.<br>[23: 8] MCS number.<br>[15:16] LSN. |
| Word 11 | LINK to access code. |
| Word 12 to end. | Variable-length data pointed to by words 6, 7, 8, and 11. |

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 4 MCS Entry (LOGMAJMCS)**

Minor Type = 2 Log-off Entry (MCSLOGOFF)

| | |
|---|---|
| Words 0-3 | See Table 15-5-2. |
| Word 4 | Log-off priority information. |
| Word 5 | Log-off class. |
| Word 6 | LINK to station name. |
| Word 7 | LINK to user code. |
| Word 8 | LINK to charge code. |
| Word 9 | Unused. |
| Word 10 | Originator information.<br>[23: 8] MCS number.<br>[15:16] LSN. |
| Word 11 | Processor time. |
| Word 12 | I/O time. |
| Word 13 | Number of lines printed. |
| Word 14 | Number of cards read. |
| Word 15 | Number of cards punched. |
| Words 16-17 | Unused. |
| Word 18 | Termination condition.<br>0-199 = Unknown termination condition.<br>200 = Normal log-off.<br>201 = MCS forced to EOJ (QUITted).<br>202 = DCP died.<br>203 = Disconnected.<br>204 = New log-on sequence required. (Log-on status changed by system operator.)<br>205 = Station cleared by operator or program.<br>206 = Station released to another MCS.<br>207 = Error abort.<br>208 = MCS restart.<br>209 = Change of charge code.<br>210 = Session number returned.<br>211 = Split Session.<br>212 = Log-off by HELLO.<br>213 = New access code.<br>214 = Reserved for expansion.<br>215 = Reserved for expansion.<br>216 = Security cleared. |

Table 15-5-4. Log-Entry Formats (Continued)

| | |
|---|---|
| Words 19-22 | Unused. |
| Word 23 | Session elapsed time (since log-on, split or hello). |
| Word 24 | LINK to access code. |
| Word 25 to end. | Variable-length data pointed to by words 6, 7, 8 and 23. |

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 4 MCS Entry (LOGMAJMCS)**

Minor Type = 3 RJE Control Card Entry (MCSRJECC)

| | |
|---|---|
| Words 0-3 | See Table 15-5-2. |
| Word 4 | Length of control card image (in units of characters). |
| Word 5 to end. | Control card image as a string of EBCDIC characters. |

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 4 MCS Entry (LOGMAJMCS)**

Minor Type = 4 MCS Message Entry (MCSMESSAGE)

| | |
|---|---|
| Words 0-3 | See Table 15-5-2. |
| Word 4 | Length of message (in units of characters). |
| Word 5 to end. | MCS message as a string of EBCDIC characters. |

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 4 MCS Entry (LOGMAJMCS)**

Minor Type = 5 Session Accrual Entry (MCSACCRUAL)
(B 7000 only)

This record is identical in format to the log-off record. The times and counts it includes are those accumulated since the last session time-accumulated, session log-on, or session log-off entry. This information is entered under several circumstances: it may be entered as a checkpoint (so that if no true log-off occurs, the resource accounting is not lost); or it may be entered when the line has been hung up but the system still has work to do for that user (as with RJE callback).

Table 15-5-4. Log-Entry Formats (Continued)

MAJOR TYPE = 4 MCS Entry (LOGMAJMCS)

Minor Type = 6 MCS Security Violation Entry

Words 0-3 See Table 15-5-2.

Word 4 Error information.

[34: 1] Line was disconnected.
[33: 1] Station was saved.
[32: 1] Station was cleared (and logged off if necessary).
[31:16] Installation-determined error code.
[15:16] MCS error code.

0 = Installation-detected error at log-on.
1 = Installation-detected error at chargecode change.
2 = Installation-detected error at accesscode change.
3 = Invalid usercode/password at log-on.
4 = Invalid chargecode at log-on.
5 = Invalid accesscode/password at log-on.
6 = Invalid station name at log-on (RJE only).
7 = Invalid chargecode while changing chargecodes.
8 = Invalid accesscode while changing accesscodes.
9 = Invalid old password while changing password of usercode.
10 = Invalid old password while changing password of accesscode.

Word 5 Identification.

[39:16] Session number if already logged on.
[23: 8] MCS number.
[15:16] LSN.

Word 6 LINK to station identifier.

Word 7 LINK to usercode.

Word 8 LINK to chargecode.

Word 9 LINK to accesscode.

**NOTE**

The usercode, chargecode, or accesscode is supplied only if it has already been successfully validated. For example, if the usercode/password failed validation on a new log-on, words 7, 8, and 9 would be all zeros. The old chargecode or accesscode is supplied in error codes 1, 2, 7, and 8. In the case of error code 6, which is detected only by RJE with the STATIONID option set, the station identifier is the default station for that line.

Word 10 LINK to input string that failed validation.

Word 11 to end. Variable-length data for the LINKs in this record.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 5 DCP Entry (LOGMAJDCP)**

Minor Type = 1 DCP Initiate Entry (LOGDCPINIT)

| | |
|---|---|
| Words 0-3 | See Table 15-5-2. |
| Word 4-5 | Unused. |
| Word 6 | DCP number in [22: 7] with bit 23 on. |
| Word 7 to end. | DCP prefix name in display form. |

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 5 DCP Entry (LOGMAJDCP)**

Minor Type = 2 MCS Initiate Entry (LOGMCSINIT)

| | |
|---|---|
| Words 0-3 | See Table 15-5-2. |
| Word 4 | MCS number. |
| Words 5-6 | Unused. |
| Word 7 to end. | MCS name in standard form. |

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 5 DCP Entry (LOGMAJDCP)**

Minor Type = 3 DCP Fault Entry (LOGDCPFAULT)

| | |
|---|---|
| Words 0-3 | See Table 15-5-2. |
| Word 4 | MCS number. |
| Word 5 | Unused. |
| Word 6 | DCP number.<br>[22: 7] DCP number. |
| Words 7-11 | Unused. |
| Word 12 | Tags.<br>[43:36] is divided into nine 4 bit tags for scratchpads 0 through 7 and the word register. |
| Word 13 | Unused. |
| Word 14 | ACIDYX registers.<br>[47: 8] AA register.<br>[47: 4] cluster number field of AA reg.<br>[43: 4] adapter number field of AA reg.<br>[39: 8] AC register.<br>[36: 5] AI index.<br>[31: 8] AI register.<br>[23: 8] D register.<br>[15: 8] Y register.<br>[ 7: 8] X register. |
| Word 15 | Scratchpad memory (SPM) number 7. |
| Word 16 | SPM 6. |
| Word 17 | SPM 5. |
| Word 18 | SPM 4. |
| Word 19 | SPM 3. |
| Word 20 | SPM 2. |

Table 15-5-4. Log-Entry Formats (Continued)

Word 21 SPM 1.

[47:24] IA register.
[47: 8] fault number if faulted.
[39: 8] IA1 register.
[39: 1] memory type (1 if local memory).
[31: 8] IA0 register.
[24: 1] half of word DCP was in.

Word 22 SPM 0.

[23:24] is the MA register.

Word 23 Word register.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 5 DCP Entry (LOGMAJDCP)**

Minor Type = 4 MCS Error Entry (LOGMCSMSG)

Words 0-3 See Table 15-5-2.

Word 4 MCS number.

Word 5 Unused.

Word 6 Word 0 of error result message (result Class = 99).

[39: 8] AC register.
[31: 8] AI register.
[23: 1] 1 is DLS, 0 is LSN.
[22:23] LSN.
[22: 7] DCP for DLS.
[15: 8] line for DLS.
[ 7: 8] station for DLS.

Word 7 Word 1 of the error result message.

[47: 8] RSLT byte index. Values and meanings for the errors are listed below:

1 = TERMINATE ERROR
2 = NO LABEL
3 = NOT APPLICABLE TO STATION
4 = TRANSMIT ABORT
5 = RECEIVE ABORT
6 = ADAPTER FAULT ABORT
7 = NO SPACE ABORT
13 = UNABLE TO INITIATE BECAUSE OF STATUS
14 = UNABLE TO COMPLETE

[39: 8] line status.
[31: 8] last flag.
[23:24] error flag.

Word 8 Word 2 of the error result message.

[47: 8] character register.
[15:16] DCP sleep address.

Word 9 Word 3 of the error result message.

Word 10 Word 4 of the error result message.

[23: 8] original DCWRITE type.
[ 7: 8] variant field.

Word 11 Error count.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 6 Miscellaneous Entry (LOGMAJMISC)**

Minor Type = 1 Halt/Load Entry (LOGHL)

| | |
|---|---|
| Words 0-3 | See Table 15-5-2. |
| Word 4 | MCP and system identification.<br>[47:12] system number.<br>[35:12] MCP mark number.<br>[23:12] MCP level number.<br>[11:12] MCP patch number. |
| Word 5 | LINK to MCP name. |
| Words 6-8 | Halt/Load cause as a string of EBCDIC characters. |
| Word 9 to end. | Variable length data pointed to by word 5. |

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 6 Miscellaneous Entry (LOGMAJMISC)**

Minor Type = 2 Log Release Entry (LOGLR)

(To be specified.)

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 6 Miscellaneous Entry (LOGMAJMISC)**

Minor Type = 3 SETSTATUS Call Entry (LOGSETSTAT)

| | |
|---|---|
| Words 0-3 | See Table 15-5-2. |
| Word 4 | Information regarding parameters to SETSTATUS call.<br>[47: 8] TYPE, the first parameter.<br>[39: 8] SUBTYPE, the second parameter.<br>[31: 8] V, the third parameter.<br>[15:16] number of words to follow in the entry. |
| Word 5 to end. | Array A is the fourth parameter to SETSTATUS. |

**NOTE**

This entry records a call to SETSTATUS, which is the MCP procedure called to carry out operator input commands. Thus, these records show most operator input that changes the status of the system, such as time and date changes, tapes purged, and tasks DSed.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 6 Miscellaneous Entry (LOGMAJMISC)**

Minor Type = 4 Security Violation Entry (LOGSECURITY)

| | |
|---|---|
| Words 0-3 | See Table 15-5-2. |
| Word 4 | LINK to usercode of violator. |
| Word 5 | Violation code as follows:<br>1 = Attempted to open a PRIVATE file.<br>2 = Attempted to open INPUT on a write only file.<br>3 = Attempted to open OUTPUT on a read only file.<br>4 = Attempted to open I/O on a read or write only file.<br>5 = Attempted to rename someone elses file.<br>6 = Attempted to create a permanent file not under your usercode.<br>7 = Attempted to execute a PRIVATE code file.<br>8 = Use of a usercode when no usercodes are defined (no SYSTEM/USERDATAFILE).<br>9 = Use of an incorrect usercode.<br>10 = Use of an incorrect password.<br>11 = Attempted to remove someone elses file.<br>12 = Illegal use of file name *USERCODE.<br>13 = Illegal use of security file attributes.<br>14 = Attempted to COPY a PRIVATE file.<br>15 = Non-privileged user attempted to modify SYSTEM/USERDATAFILE.<br>16 = Not a viable usercode (no SYSTEM node in SYSTEM/USERDATAFILE entry).<br>17 = Suspended usercode.<br>18 = Old password required but not supplied (changing passwords).<br>19 = Old password incorrect (changing passwords).<br>20 = Invalid chargecode. |
| Word 6 | LINK to name. (Can be file name, password and so on, depending upon violation involved.) |
| Word 7 | LINK to access code. |
| Word 8 | Violation origin information.<br>[47: 1] Task originated from a WFL queue.<br>[46:10] Queue from which task originated.<br>[36:15] LSN from task's STATION attribute.<br>[21: 6] MCS from which task originated.<br>[15: 1] Task originated from Data Comm.<br>[14:15] If task originated from Data Comm, LSN of originating terminal; otherwise, unit number of originating peripheral. |
| Word 9 to end. | Variable-length data for the LINKs in this record. |

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 6 Miscellaneous Entry (LOGMAJMISC)**

Minor Type = 3 SETSTATUS Call Entry (LOGSETSTAT)

| | |
|---|---|
| Words 0-3 | See Table 15-5-2. |
| Word 4 | Information regarding parameters to SETSTATUS call.<br>[47: 8] TYPE, the first parameter.<br>[39: 8] SUBTYPE, the second parameter.<br>[31: 8] V, the third parameter.<br>[15:16] number of words to follow in the entry. |
| Word 5 to end. | Array A is the fourth parameter to SETSTATUS. |

**NOTE**

This entry records a call to SETSTATUS, which is the MCP procedure called to carry out operator input commands. Thus, these records show most operator input that changes the status of the system, such as time and date changes, tapes purged, and tasks DSed.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 6 Miscellaneous Entry (LOGMAJMISC)**

Minor Type = 4 Security Violation Entry (LOGSECURITY)

| | |
|---|---|
| Words 0-3 | See Table 15-5-2. |
| Word 4 | LINK to usercode of violator. |
| Word 5 | Violation code as follows:<br>1 = Attempted to open a PRIVATE file.<br>2 = Attempted to open INPUT on a write only file.<br>3 = Attempted to open OUTPUT on a read only file.<br>4 = Attempted to open I/O on a read or write only file.<br>5 = Attempted to rename someone elses file.<br>6 = Attempted to create a permanent file not under your usercode.<br>7 = Attempted to execute a PRIVATE code file.<br>8 = Use of a usercode when no usercodes are defined (no SYSTEM/USERDATAFILE).<br>9 = Use of an incorrect usercode.<br>10 = Use of an incorrect password.<br>11 = Attempted to remove someone elses file.<br>12 = Illegal use of file name *USERCODE.<br>13 = Illegal use of security file attributes.<br>14 = Attempted to COPY a PRIVATE file.<br>15 = Non-privileged user attempted to modify SYSTEM/USERDATAFILE.<br>16 = Not a viable usercode (no SYSTEM node in SYSTEM/USERDATAFILE entry).<br>17 = Suspended usercode.<br>18 = Old password required but not supplied (changing passwords).<br>19 = Old password incorrect (changing passwords).<br>20 = Invalid chargecode (B 7000 only). |
| Word 6 | LINK to name. (Can be file name, password and so on, depending upon violation involved.) |
| Word 7 | LINK to access code. |
| Word 8 to end. | Variable length data pointed to by words 4, 6, and 7. |

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 7 Installation Entry (LOGMAJINST)**

(Reserved for installation use.)

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 9 Reader/Sorter Entry (LOGMAJRS)**

Minor Type = 1 RSMONITOR Initiation (LOGRSMONITORINIT)

| | |
|---|---|
| Words 0-3 | See Table 15-5-2. |
| Word 4 | Parameter passed to RSMONITOR for initiation. |
| Word 5 | Prefix used for all software files (in standard form format). |

**NOTE**

This record contains the prefix for all software files used by the Reader/Sorter subystem. This prefix is used until RSMONITOR terminates because of non-use of Reader/Sorters, operator DS, or fatal RSMONITOR fault.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 9 Reader/Sorter Entry (LOGMAJRS)**

Minor Type = 2 RSMONITOR Faults (LOGRSMONITORFAULT)

Words 0-3 See Table 15-5-2.

Word 4 Type of fault.

Word 5 Pertinent data.

**NOTE**

This record is used to log Reader/Sorter subsystem software control errors detected by RSMONITOR. These include, but are not limited to:

1. Message queue link errors.
2. Message queue sequence errors.
3. Message queue "lock" errors.
4. Inconsistent messages.

    a. Bad RSP number.
    b. Bad message type.
    c. Bad RS number.

5. Input queue recovery performed.
6. Inconsistent sequence of RS software control messages.

These errors may be indicative of either software or hardware malfunction.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 9 Reader/Sorter Entry (LOGMAJRS)**

Minor Type = 3 RSMONITOR Termination (LOGRSMONITORTERM)

| | |
|---|---|
| Words 0-3 | See Table 15-5-2. |
| Word 4 | Reason for termination. |
| Word 5 | Number of messages handled. |

**NOTE**

This records the reason for termination of RSMONITOR which include:

1. Normal termination due to no users.
2. Operator DS and no users.
3. RSMONITOR irrecoverable fault.
4. R/S subsystem files not available.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 9 Reader/Sorter Entry (LOGMAJRS)**

Minor Type = 4 RSP Initiation (LOGRSPINITIATION)

Words 0-3 See Table 15-5-2.

Word 4 RSP number.

Word 5 Reason for initiation.

**NOTE**

This records an attempt to initialize an RSP. A corresponding RSP termination log record exists. This record is generated as a result of either:

1. A call on RSMONITOR to initiate all RSPs. This is a result of a R/S file OPEN when RSMONITOR is not running.

2. An explicit operator request to initiate (or re-initiate) a specific RSP.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 9 Reader/Sorter Entry (LOGMAJRS)**

Minor Type = 5 RSP Termination (LOGRSPTERM)

Words 0-3 See Table 15-5-2.

Word 4 RSP number.

Word 5 RSPTABLE entry for this RSP.

[23: 8] Reason for termination.

During intialization.

1 = Code file disk error by a mainframe.
2 = Scan fault.
3 = Timeout - failed to respond.
4 = Code file disk error by RSP.
5 = Code file not visible to RSP.
6 = RSP software load error.

During normal operation.

32 = No handshake.
33 = Input message sequence error.
34 = Killed by MEMDUMP (no response).
35 = No response to pause request.
36 = Invalid message buffer.
37 = Operator request.

During queue recovery.

64 = No response to pause request.
65 = Message link word left locked.
66 = Input message sequence error.
67 = Extraneous messages.

**NOTE**

This records the reason for removal of an RSP from an ON-LINE environment as determined by the mainframe. This may be for several reasons:

1. Operator request ("RSP n-").
2. RSMONITOR being terminated.

Table 15-5-4. Log Entry Formats (Continued)

**NOTE (Continued)**

3. RSMONITOR detected malfunction.

   a. Failure to respond within time interval indicating an RSP or BIC failure.
   b. Message queue malfunction.
   c. Initiation failure.

      1) No BIC path.
      2) No response on initial load.
      3) Disk errors in system software code files being loaded into the RSP. This may be due to mainframe or RSP detected hardware failures reported to the mainframe.
      4) System software code files are on disk units that are not accessable to the RSP.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 9 Reader/Sorter Entry (LOGMAJRS)**

Minor Type = 6 RSP Dump Reason (LOGRSPDUMPHISTORY)

Words 0-3 See Table 15-5-2.

Word 4 RSP number.

Word 5 Dump information word.

[47:24] Return address for call on ERR.
[23:24] Cause of death.

Word 6 X and Y registers.

[47:16] X register.
[23:24] Y register.

Word 7 T and FA registers.

[47:24] T register.
[23:24] FA register.

**NOTE**

This records the reason for abnormal termination as determined by the RSP. The information for this record is retrieved from a fixed location in the RSP memory. In the event the mainframe cannot retrieve this information via the BIC, then it can be obtained from an RSP memory dump forced to disk, or as a last resort it may be read directly from RSP memory. In addition to the reason, important registers are recorded to allow determination of the point of error detection in the RSP software.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 9 Reader/Sorter Entry (LOGMAJRS)**

Minor Type = 7 RSP Non-fatal Faults (LOGRSPFAULT)

Words 0-3 See Table 15-5-2.

Word 4 RSP number (if applicable).

[27: 8] R/S number.
[19: 4] R/S channel number (within RSP).
[ 3: 4] RSP number.

Word 5 To be defined.

**NOTE**

This records non-fatal RSP software detected faults. This is normally associated with a specific UCR or RS. Examples of these are:

1. UCR invalid S-op.
2. UCR invalid index.
3. UCR disk file EOF-no label.
4. UCR disk file PARITY-no label.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 9 Reader/Sorter Entry (LOGMAJRS)**

Minor Type = 8 RSP Disk Errors (LOGRSPDISKERR)

Words 0-3 See Table 15-5-2.

Word 4 RSP number, and so forth.

[27: 8] RS number(if applicable).
[19: 4] RS channel number(within RSP).
[ 3: 4] RSP number.

Word 5 IO information word.

[47: 4] Number of retries.
[43: 1] Success flag.
[42: 1] Type of operation (Read/Write).
[39: 8] Result Descriptor.
[31:32] Disk address.
[31: 4] RSP Disk Channel.
[27: 4] RSP EU number.
[23:24] Disk address within the EU.

Word 6 IO error information.

[47: 4] File number within UCR (or 0).
[43:20] Block length.
[23:24] Number of accesses prior to failure.

Word 7 Mainframe Unit number corresponding to RSP unit.

**NOTE**

This records a disk error encountered by an RSP. Sufficient information is recorded to identify the failing unit number, area, failure type, the corresponding mainframe unit number, and UCR disk file index.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 9 Reader/Sorter Entry (LOGMAJRS)**

Minor Type = 9 RSP BIC Errors (LOGRSPBICERR)

| | |
|---|---|
| Words 0-3 | See Table 15-5-2. |
| | **** detected by RSSCANIN |
| Word 4 | Record type (0 = RSSCANIN). |
| Word 5 | P1 word returned to HARDWAREINTERRUPT. |
| Word 6 | Scan Function used with the operation. |
| Word 7 | (Not used at this time.) |
| Word 8 | RSP number.<br>[ 7: 8] RSP number (or Port No. if [46: 1]=1).<br>[46: 1] If = 1, RSP number is BIC port. |
| Word 9 | BICTABLE word #1. |
| Word 10 | BICTABLE word #2.<br>[39:20] Current Upper Bound.<br>[19:20] Current Lower Bound.<br>[43: 4] BIC State (see MCP). |
| Word 11 | RSPTABLE word (unless specific BIC port requested, the 0). |
| Word 12 | STACK HISTORY (4 levels). |
| Word 4 | Record type (1 = RSSCANOUT). |
| Word 5 | P1 word returned to HARDWAREINTERRUPT. |
| Word 6 | Scan Function word used with the operation. |
| Word 7 | Data word (if any) used with the Scan operation. |
| Word 8 | The rest of the record is the same as that for the RSSCANIN record. |

Table 15-5-4. Log-Entry Formats (Continued)

Word 4 Record type (2 = RSP).

Word 5 RSP number, and so forth.

[ 3: 4] RSP number.

Word 6 RSP IO information.

[47:24] RSP memory address.
[23:24] Result descriptor from BIC.

Word 7 Type and origin of operation.

[45: 2] Port Interchange Port Number.
[43: 4] Type of operation.
[39:20] Length (if applicable).
[19:20] B 6700 address (if applicable).

Word 4 Record type (3 = BIC).

Word 5 BIC number.

Word 6 BICTABLE word no. 1 for the BIC.

**NOTE**

This records a BIC error encountered by either the RSP, the HOST system, or the BIC (reported in Scan In Status). The errors fall into one of four categories:

1. Detected by RSSCANIN.
2. Detected by RSSCANOUT.
3. Detected (and reported) by the RSP.
4. Detected by the BIC.

The records for each of the 4 types vary somewhat, but word four indicates the error type and the record format can then be determined as below.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 9 Reader/Sorter Entry (LOGMAJRS)**

Minor Type = 10 RSP UA or UR (LOGRSPUAUR)

Words 0-3 See Table 15-5-2.

Word 4 RSP number and UA or UR flag.

[32: 1] UA/UR flag (1 = UR).

**NOTE**

This records the removal of an RSP for maintenance or the subsequent return of that unit to the system. Errors detected during this type of maintenence are not logged by the Reader/Sorter subsystem.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 9 Reader/Sorter Entry (LOGMAJRS)**

Minor Type = 11 RSLABEL (LOGRSLABEL)

(Format not available at this time.)

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 9 Reader/Sorter Entry (LOGMAJRS)**

Minor Type = 12 RS Statistics (LOGRSSTATISTICS)

Words 0-3 See Table 15-5-2.

Word 4 RSP number, and so forth.

[35: 1] Logged at EOR.
[34: 1] Logged because of JAM.
[33: 1] Logged because of MISSORT.
[32: 1] Logged because of BLACKBAND item.
[27: 8] RS number.
[ 3: 4] RSP number.

Word 5 RS not ready time (0.1 seconds).

Word 6 RS flow time (0.1 seconds).

Word 7 Number of items.

Word 8 Number of "can't read" items.

Word 9 Number of halts.

Word 10 Number of double documents.

**NOTE**

This records the performance statistics of a Reader/Sorter. Statistics are gathered during the time a UCR is loaded in the RSP for the Reader/Sorter. The information provided can be useful in several ways:

1. The user can determine the efficiency of the application and the utilization of the equipment.
2. Maintenance personnel can use these records to evaluate the reliability of the Reader/Sorter (and associated control). The data provided here indicate several specific problem areas:

   a. JAMS
   b. Missorts
   c. Can't reads - A continuously high can't read rate over a large quantity of mixed items would probably imply a character recognization problem, whereas a fluctuating can't read rate would probably indicate document problems.
   d. Double documents

Table 15-5-4. Log-Entry Formats (Continued)

**NOTE (Continued)**

Reader/Sorter statistics are logged whenever a Jam, Missort, Black-band, or End-Of-Run occurs. The times and totals are for the period following the last previous logging of RSSTATISTICS for the Reader/Sorter. Using this technique provides summary data for segments of the total run rather than an overall average, thus providing a more complete picture of the operation.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 9 Reader/Sorter Entry (LOGMAJRS)**

Minor Type = 13 RS UA or UR (LOGRSUAUR)

Words 0-3 See Table 15-5-2.

Word 4 RS number, and so forth.

[27: 8] RS number.
[19: 4] RS channel number.
[ 3: 4] RSP number.

**NOTE**

This records the removal of the Reader/Sorter from the system for maintenance and the subsequent return of the unit to the system.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 9 Reader/Sorter Entry (LOGMAJRS)**

Minor Type = 14 RS SUBSYSTEM MESSAGES (LOGRSMSGS)

(Format not available at this time.)

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 9 Reader/Sorter Entry (LOGMAJRS)**

Minor Type = 15 UCR Initiation (LOGUCRBOJ)

Words 0-3 See Table 15-5-2.

Word 4 RS number, and so forth.

[27: 8] RS number.
[19: 4] RS channel number.
[ 3: 4] RSP number.

**NOTE**

This record provides the user with the starting time for a UCR. This can help the user determine the utilization of the RSP.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 9 Reader/Sorter Entry (LOGMAJRS)**

Minor Type = 16 UCR Termination (LOGUCREOJ)

Words 0-3 See Table 15-5-2.

Word 4 RS number, and so forth.

[27: 8] RS number.
[19: 4] RS channel number
[ 3: 4] RSP number.

Word 5 Final Result Descriptor sent to RS file.

**NOTE**

This record provides the user with the ending time for a UCR and the reason for the termination of the UCR.

Table 15-5-4. Log-Entry Formats (Continued)

**MAJOR TYPE = 9 Reader/Sorter Entry (LOGMAJRS)**

Minor Type = 17 BIC UA or UR (LOGBICUAUR)

Words 0-3 See Table 15-5-2.

Word 4 BIC number, and so forth.

[32: 1] UA/UR flag (1 = UR).
[27: 8] BIC number.

**NOTE**

This records the removal of the BIC from the system for maintenance and the subsequent return of the unit to the system.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# 6. IAD LOGS

To preserve a system log through a cold start, an installation must place it in an installation-allocated disk (IAD) file. This placement is done by including a log card in the system loader deck. The presence of this card causes a cold start. The log card must specify two rows for the system log, giving the EU and starting segment for each row. Optionally, it may first specify the row size; if this option is absent, 1000 segments are allocated per row.

**Examples:**

```
LOG ROW EU1 SEGMENT 10000 ROW EU2 SEGMENT 10000
LOG ROWSIZE 2000 ROW EU 1 SEGMENT 2000
    ROW EU 1 SEGMENT 22000
```

This card is in addition to the card specifying these rows as IAD. The log card may be used to specify non-IAD areas, but these areas are not recoverable over a cold start.

When an IAD-type log is initialized, the end-of-file pointer is set to the start of the second row; thus log entries are written into row two. When that row is filled, the two row descriptors are exchanged so that the first row becomes the second. Writing continues at the start of the second row (which was formerly the first). A program wishing to analyze the LRed section of the log must read it starting at the first row (which was initially the second row) until it detects a "binary end" pattern.

To read the log file to the current record (as LOGANALYZER does), a program should begin with the first row. When it encounters the binary end pattern, it should skip over the pattern to the start of the second row and continue until end-of-file.

Because, unlike the standard system log, an IAD log wraps around on a row-by-row basis, the installation should make the row size large enough to contain a logging interval (the number of records likely to be written between log analysis runs.)

If the system must be cold started, the following action must be taken to preserve the log. First, the log card in the loader deck must be changed to specify a different area of IAD. Second, after the cold start, IADMAPPER must be run to create a header to the first log, giving it an appropriate name.

**NOTE**

To properly recover the system log after a cold start, an installation must keep track of the absolute disk addresses of the current log and update that information whenever a cold start reassigns the log addresses.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

SWAPPER

TABLE OF CONTENTS


1. GENERAL INFORMATION . . . . . 16-1- 1
INTRODUCTION. . . . . 16-1- 1
SWAPPING ON MULTIPLE FAMILIES . . . . . 16-1- 1
NON-MONOLITHIC SYSTEMS. . . . . 16-1- 1
SWAPPER EXPRESS FACILITY. . . . . 16-1- 2
INTER-PROGRAM COMMUNICATION (IPC) . . . . . 16-1- 2
2. SWAPPER PARAMETERS. . . . . 16-2- 1
INTRODUCTION. . . . . 16-2- 1
PARAMETER DISPLAY . . . . . 16-2- 4
TIME-SLICE COMPUTATION. . . . . 16-2- 5
TIMESLICING PARAMETERS. . . . . 16-2- 5
QUEUE PRIORITY. . . . . 16-2- 6
PRAGMATICS OF PARAMETERS. . . . . 16-2- 7
Changing the Swapspace Size via the CORESIZE Parameter . . . 16-2- 8
EXPANDING THE SWAPSPACE . . . . . 16-2- 8
SHRINKING THE SWAPSPACE . . . . . 16-2- 8
SWAPPER Initialization and Swapspace Size. . . . . 16-2- 8
Swapping on Multiple Families. . . . . 16-2- 9
Multiple Families on a Non-Monolithic System . . . . . 16-2- 9
Multiple Swapdisk Families with Different Row Sizes. . . . . 16-2- 9
Out of Swapdisk Mechanism. . . . . 16-2- 9
I/O ERROR HANDLING MECHANISM. . . . . 16-2- 11
I/O Errors on Write Operations . . . . . 16-2- 11
I/O Errors on Read Operations. . . . . 16-2- 11
3. SWAPPING. . . . . 16-3- 1
SWAPPING OUT. . . . . 16-3- 1
SWAPPING IN . . . . . 16-3- 1
EXPRESS TASKS . . . . . 16-3- 1
4. EXECUTING TASKS IN SWAPSPACE. . . . . 16-4- 1
5. SYSTEM/SWAPDISK . . . . . 16-5- 1

APPENDIX 16A. SWAPPER ERROR MESSAGES . . . . . 16-A- 1

APPENDIX 16B. SWAPPER MECHANISM. . . . . 16-B- 1

APPENDIX 16C. SYSTEMSTATUS INFORMATION . . . . . 16-C- 1

STATISTICS. . . . . 16-C- 1

APPENDIX 16D. QUEUE PRIORITY . . . . . 16-D- 1

QUEUE PRIORITY EFFECT . . . . . 16-D- 1

QUEUE PRIORITY VALUE. . . . . 16-D- 3

APPENDIX 16E. B 5000/B 6000 TIGHTLY-COUPLED. . . . . 16-E- 1

The SUBSYSTEM Task Attribute. . . . . 16-E- 1

SUBSYSTEM vs SUBSPACES: Conflicting Specifications. . . . . 16-E- 1

Process Family Migration on a Non-Monolithic System . . . . . 16-E- 2

Swap Tasks and Nonexchanged Units . . . . . 16-E- 2

APPENDIX 16F. B 7000 TIGHTLY COUPLED . . . . . 16-F- 1

Limitations . . . . . 16-F- 1

Swap Tasks and Nonexchanged Units . . . . . 16-F- 1

# 1. GENERAL INFORMATION

## INTRODUCTION

Job swapping or time-slicing is a facility that allows installations to service users in a time-sharing environment. In such an environment many tasks are competing for a relatively small amount of memory. To service all users adequately, tasks are swapped from memory to disk when they can no longer run or have exceeded a specified time-slice. Previously swapped out tasks that are ready to run are swapped in from disk to memory in order to utilize the free memory space. Time-slicing provides all tasks with an opportunity to use the system resources.

Memory utilized for job swapping consists of one or more areas or "chunks", collectively known as the "swapspace", whose size is specified by the installation. This swapspace may be subdivided among several tasks. The memory areas in the swapspaces which are occupied by a given task are contiguous in order to enable the task to be read from or written to disk with a single input/output operation.

For tasks run in swapspace, memory management functions almost the same as "normal" memory management. Tasks have areas overlayed and request more memory space via presence-bit interrupts. However, since memory areas for tasks are contiguous, the allocation algorithm views each task as a subset of memory. Memory allocation is performed within the boundaries established for each task.

Because each task possesses a unique set of memory links, overlay occurs in the area of the task so that one task never overlays memory areas that belong to another task.

Requests for memory which cannot be satisfied (because no available or overlayable memory is present) cause a swap-out to occur for the task. When the swapspace area of the task is swapped in, it is enlarged in increments of 990 words. Tasks that exceed the installation-specified, swap job size (a system maximum of 130,680 words) are terminated by the system.

An important aspect of the swap-in mechanism is the relocation of the task in swapspace. During relocation, all memory addresses are examined and appropriately modified. The swap-in mechanism attempts to place the task at the end of its subspace in order to minimize "checkerboarding".

## SWAPPING ON MULTIPLE FAMILIES

SWAPPER has the ability to use more than one family and more than one SYSTEM/SWAPDISK file on which to swap. The list of families being used by SWAPPER can be dynamically adjusted.

## NON-MONOLITHIC SYSTEMS

In a non-monolithic environment, a SWAPPER independent runner and its associated swapspace may be initiated in each memory subsystem (except Global Memory). Each of these SWAPPERs is independent in the sense that each has its own set of parameters and can be initiated and terminated independently of other SWAPPERs running in other memory subsystems. However, all SWAPPERs in such an environment must share at least one *SYSTEM/SWAPDISK file; each row of this *SYSTEM/SWAPDISK file must, therefore, reside on a unit that is exchanged to each processor in which a SWAPPER is to be run.

A more detailed discussion of the implications of this capability of running multiple, independent SWAPPERs on a non-monolithic system can be found in Appendix 16E.

## SWAPPER EXPRESS FACILITY

An "express" facility improves the response for small interactive tasks, by preventing large or non-interactive tasks from monopolizing the swapspace. "Small" is defined by an operator parameter, according to the resource-allocation and user-service strategy of an installation. "Interactive" is determined by program behavior; the determination depends, to some extent, on the SWAPPER time-slicing parameters.

The express feature works by reserving part of the swapspace for express tasks. When memory must be found for a task, an express task is placed anywhere space is available, with the search starting in the reserved area. A non-express task may not occupy the reserved area. Thus, express tasks compete mainly with other express tasks; and because they are interactive, each one spends only a short time in central memory at each interaction.

## INTER-PROGRAM COMMUNICATION (IPC)

With some restrictions, any Inter-program Communication (IPC) tasking structure ("family") may be run in swapspace. The principal rule governing swappable IPC families is that if the "critical block" for a dependent process is in a swappable stack, then that process must also be swapped.

Families in which up-level pointers may potentially exist are prohibited from running in swapspace. IPC-capable tasks with codefiles compiled prior to the 3.0 release are not run in swapspace unless the IPCOVERRIDE bit of the OPTION word of the task is set. This bit, which is reset by default, may be set in WFL or CANDE in the same manner as the other OPTION word bits (for example, RUN T; OPTION=IPCOVERRIDE; SUBSPACES=2). All other codefiles run in swapspace if the SUBSPACES attribute is non-zero, regardless of whether or not the task is IPC-capable.

Libraries, databases, and jobs may all run in swapspace. The general rules governing such usage are as follows:

1. Library stacks. If a library executes in swapspace, any program using the library, and any task processed out of the library, must also run in swapspace. On a non-monolithic system, all tasks associated with the library (users and processes) must also run in the same subsystem. Swapping a user of a library is functionally the same as swapping an offspring of the library; that is, whenever the user is in memory, the library must also be in memory. However, the reverse is not true; when the library is in memory, any of the users of the library may or may not be in memory.

2. Database stacks (DBS). The DMS ACCESSROUTINES are initiated as an ordinary task with certain privileges. As with libraries, if the DBS runs in swapspace, so must all users of the DBS.

3. Job stacks. A WFL job and its offspring, when run in swapspace, follow the same rules as regular process families run in swapspace. However, any external process of a WFL job (RUN or PROCESS of a separate code file) need not run in swapspace. Such a task is not visible to the parent job stack and is defined as a "semi-dependent task". Any parameters from WFL are passed by value, including arrays and strings; thus, the task does not access memory belonging to the job stack.

The swap queue priority of a task that is a member of an IPC family is affected by the resources (number of I/O operations and amount of swapspace) required to ensure that all swappable parents of the task are brought into the swapspace. Two parameters, IOBIAS and MEMORYBIAS, affect the degree to which the swap queue priority of the task is affected. These parameters are discussed in Section 2 of this chapter.

Only a "leaf" of an IPC family "tree" is eligible to be placed in the express area of swapspace if other criteria of express status are satisfied. Processes with swappable offspring are never considered express tasks.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# 2. SWAPPER PARAMETERS

## INTRODUCTION

The SWAPPER parameters are used to control swapdisk size, time-slice computation, express tasks, and queue placement. These parameters are stored in the directory of the Halt/Load unit.

The following descriptions provide an introductory explanation of the parameters. Subsequent sections give a more detailed explanation of the parameters and how they are used.

SWAPPER parameters that affect subspace sizes are specified in 990-word slots.

The system default values for the parameters are assigned to a parameter if its value is zero when SWAPPER is initiated.

The following parameters specify the sizes of the swap area:

CORESIZE — The total amount of memory, in 990-word slots, allocated as swapspace.

MAXCORE — The maximum amount of memory that may be allocated to any swap task. System default is the minimum of CORESIZE, the largest AREASIZE of any swapdisk file, and 132 slots.

MINCHUNKSIZE — The smallest amount of memory that a single chunk of swapspace may contain; must be greater than or equal to MAXCORE. By setting MINCHUNKSIZE equal to the value of CORESIZE, one contiguous area of memory is allocated.

The following parameters affect time-slice computation for swap jobs:

MINTIMESLICE — The amount of processor time in seconds that a non-express task may accumulate in each time-slice before being swapped out. This value is specified in units of seconds and may be a fractional number. System default for non-express tasks is 3.

EXPTIME — The amount of processor time (in seconds) that an express task may accumulate in each time-slice before being swapped out. This value is specified in units of seconds and may be a fractional number. System default for express tasks in .75 seconds.

RATIO — RATIO is used to specify the maximum elapsed time that a task may stay in swapspace before being swapped out. The system default is 2. The maximum elapsed time is specified by the following formula:

ET = RATIO * MINTIMESLICE

MAXSLICENUMBER — The maximum value of the slice number of a task. System default is 7; this value may range from 0 to 255.

Each time a task is swapped out because it was sliced (when MINTIMESLICE or elapsed time is exceeded), its slicenumber is incremented by one until it reaches MAXSLICENUMBER. When a task is demand-swapped (for example, for remote I/O), its slicenumber is set to zero. When the slicenumber is zero, the task is considered to be interactive; when the slicenumber is MAXSLICENUMBER, it is non-interactive. Any value between zero and MAXSLICENUMBER causes the task to look partially interactive.

SWAPTRANSTATE — Tasks are time-sliced regardless of whether or not they are in transaction state. This is the default state.

NOSWAPTRANSTATE — Any task that is in transaction state and exceeds its time slice is given another time slice.

The express facility is controlled by:

EXPRESERVE — The amount of memory in 990-word slots reserved in swapspace for express tasks. System default is 0.

EXPMAXCORE — The maximum size of an express task in 990-word slots. An interactive task whose memory requirements do not exceed EXPMAXCORE is express. System default is zero.

The queueing mechanism is affected by:

PRIORITYBIAS — This value controls the effect of priority on the queueing mechanism. System default is 0; this value may range from 0 to 100.

UTILIZATIONBIAS — This value controls the effect of interactivity on the queueing mechanism. System default is 6; this value may range from 0 to 50.

IOBIAS — This value affects the relative priority of a task that is a member of an IPC family running in swapspace. The number of I/O operations necessary to bring into the swapspace all swappable parents of the task whose priority is being calculated is multiplied by the value of this parameter; the larger the value, the lower the relative priority of the task. System default is 0.

MEMORYBIAS — This value affects the relative priority of a task that is a member of an IPC family running in swapspace. The amount of memory (in slots) required by all swappable parents of the task whose priority is being calculated is multiplied by the value of this parameter; the larger the value, the lower the relative priority of the task. System default is 0.

The following parameters apply to swapdisk:

| | |
|---|---|
| FAMILY | The families in use by SWAPPER. Each in-use family must contain a SYSTEM/SWAPDISK file. |
| MAXIOSIZE | The maximum size, in words, of any SWAPPER I/O operation. When a single subspace exceeds the value of the MAXIOSIZE parameter, multiple I/O operations are done to transfer the subspace to or from disk. The I/O operations are calculated to result in approximately equal length transfers. |

The following swapdisk file attributes may be set using the CREATE option of the SW ODT message.

| | |
|---|---|
| AREASIZE | The swapdisk row size in units of 44-segment disk slots (30 words each). |
| AREAS | The number of swapdisk rows. |

**NOTE**

When AREAS and AREASIZE are specified, care must be taken to ensure that enough space is reserved on the swapdisk(s) for all swap tasks.

The attributes of different swapdisk files need not be the same; however, SWAPPER requires that each I/O operation fit entirely within a single row of a swapdisk. The maximum value of the MAXIOSIZE parameter is the size of the largest row of swapdisk available.

When initialized, SWAPPER performs several consistency checks on the swapdisk files and produces error messages if all the declared rows are not allocated or if the file is crunched. SWAPPER sets the following attributes in the swapdisk files:

```
SECURITYTYPE = PRIVATE
SECURITYUSE  = SECURED
SENSITIVEDATA= TRUE
EXCLUSIVE    = TRUE
```

## PARAMETER DISPLAY

Most SWAPPER parameters may be changed dynamically by use of the SW ODT command. The parameters are changed by entering SW followed by at least the first four characters of the parameter names. The current values of these parameters are interrogated using an SW message followed by an ETX; the system responds by displaying the current values on the display screen.

Any SW message causes the SWAPPER parameters to be displayed. The display shows the current status of SWAPPER and the value of all the parameters. The memory size specifications are displayed in both slots and words.

**Example**

The following lines would be part of the response to the SW ODT message:

```
SW ** RUNNING ** CORESIZE=90 SLOTS (89100 WORDS)
SW ACTUALCORESIZE=90 SLOTS (89100 WORDS) MINTIME=3 SECONDS
SW MAXSLICENR=7 RATIO=2 MAXCORE=50 SLOTS (49500 WORDS)
SW MAXIOSIZE=   50 SLOTS (49500 WORDS)
SW MINCHUNKSIZE=50 SLOTS (49500 WORDS) EXPRESERVE=0 SLOTS (0 WORDS)
SW EXPMAXCORE=   0 SLOTS (0 WORDS)     EXPMAXTIME=0.75 SECONDS
SW PRIORITYBIAS= 0 UTILIZATIONBIAS=6   IOBIAS=0 MEMORYBIAS=0
SW NOSWAPTRANSTATE ON SWAPPACK ON KSID
```

The initial item "** RUNNING **" is an indicator of the current status of SWAPPER; the possible values are NOT RUNNING, INITIALIZING, and SHUTTING DOWN.

ACTUALCORESIZE is an indicator of the amount of swapcore actually being used by SWAPPER (see Changing the Swapspace Size via the CORESIZE Parameter for semantics).

NOSWAPTRANSTATE indicates that the NOSWAPTRANSTATE option is set; if no reference to this option appears, SWAPTRANSTATE is set.

ON SWAPPACK ON KSID indicates that SWAPPER is using the SYSTEM/SWAPDISK files on the packs SWAPPACK and KSID.

Any parameters may be modified at any time, but some parameters (for example, CORESIZE and FAMILY) may not take effect immediately.

**Example**

The following message sets EXPRESERVE to 20 slots (of 990 words each) and sets EXPMAXCORE to 15 slots:

```
SW EXPR 20 EXPMAXCORE 15
```

## TIME-SLICE COMPUTATION

The length of a time-slice is determined by the SWAPPER parameters MINTIMESLICE (non-express tasks) or EXPTIME (express tasks) and RATIO. The processor time-slice is the amount of processor time (in seconds) which a task may accrue during one period of residency in memory, after which it may be swapped to SWAPDISK to make room for other tasks. The processor time-slice allocated to a task is MINTIMESLICE for non-express tasks and EXPTIME for express tasks. An express task that remains in memory long enough for its time-slice to elapse loses express status; in that case, the processor time-slice allocated to that task on its next swapin is MINTIMESLICE.

The elapsed time-slice is the amount of time (in seconds) which may pass while a task remains resident, after which it may be sliced out regardless of its processor time accrual. Any time the task was ready for a processor, but no processor was available, is not counted in elapsed time. The elapsed time-slice is MINTIMESLICE * RATIO for non-express tasks and EXPTIME * RATIO for express tasks.

## TIMESLICING PARAMETERS

MINTIMESLICE has very important effects on response time and system efficiency. A short time-slice increases the frequency of commutation through non-interactive tasks, thereby increasing overhead. A long time-slice slows response time by extending the time an interactive task must wait for a non-interactive task to complete when no swapspace is available. When the express mechanism is active, large or non-interactive tasks no longer compete for the reserved part of the swapspace; thus, the response time of express tasks is minimally affected by increasing the time-slice. On the other hand, interactive tasks that are too large to be express suffer poor response when the system is busy.

A very short MINTIMESLICE limits the amount of work a task can do at each interaction while remaining classified as interactive. However, if the express parameters are set to accommodate most interactive jobs at an installation, the time-slicing parameters may be set large enough to be efficient for non-interactive tasks; thus, MINTIMESLICE has little effect in determining which jobs are express. (Typically, a clear dichotomy exists: a job is interactive with output responses very quickly following input, or it is non-interactive with little or no remote input.)

A very long MINTIMESLICE may reduce throughput. SWAPPER verifies that the task has sufficient memory only when a job is swapped out; hence, jobs that require more memory are only allocated more slots at swap-out time. These jobs thrash until the elapsed time-slice is exceeded, at which time more slots can be allocated.

Whenever a task is time-sliced, its "slice number" is incremented until it reaches the value specified by the MAXSLICENUMBER parameter. The slice number is reset to zero when a task is swapped out for any reason other than a time-slice.

## QUEUE PRIORITY

SWAPPER uses a single, priority-ordered queue. Selection from the queue is always done from the highest priority end. The priority of a task in the queue is based on a function of the declared priority and the interactiveness of the task and on the number of times the task has been passed over in the queue. The effect of these parameters is individually controlled by the PRIORITYBIAS, UTILIZATIONBIAS, IOBIAS, and MEMORYBIAS SWAPPER parameters.

PRIORITYBIAS controls the effect of the declared priority of a task on placement in the queue. UTILIZATIONBIAS controls the extent to which interactiveness of a task affects the placement in the queue. Appendix 16D contains a more detailed explanation of PRIORITYBIAS and UTILIZATIONBIAS.

The interactiveness of a task is based on the percentage of the allocated time-slice which is not used on a demand swap, or the percentage of slices (with respect to MAXSLICENUMBER) used by a time-sliced task. Thus, various degrees of interactiveness are possible. For example, a task that uses only a small amount of its time-slice is highly interactive, while a task that uses most of its time-slice is less so. Also, a task whose slice number is smaller than the MAXSLICENUMBER can be thought of as slightly interactive, while a task that has reached MAXSLICENUMBER is no longer interactive. The time-slice used in these calculations is the normal job time-slice, so express jobs (which actually use small slices) appear highly interactive.

To prevent total starvation of lower priority, non-interactive tasks, the queue priorities of the tasks are incremented (by one) each time the tasks are passed over by higher priority tasks. The effects of this aging are dependent on both PRIORITYBIAS and UTILIZATIONBIAS parameters.

For IPC families running in swapspace, the queue priority is further affected by the parameters IOBIAS and MEMORYBIAS. The calculated queue priority of the IPC family member is first decremented by the value of IOBIAS multiplied by the number of I/O operations required to bring in all swappable parents of the member. The calculated queue priority of the IPC family member is further decremented by the value of MEMORYBIAS multiplied by the ratio of the number of memory slots required to hold all swappable parents to the value of MAXCORE.

Thus, the larger the values of IOBIAS and MEMORYBIAS, the lower the queue priority of IPC families.

## PRAGMATICS OF PARAMETERS

CORESIZE determines how much of the central memory is devoted to swap tasks. Enough space must be left outside to accommodate data and code for the MCP, its independent runners (both invisible ones like CONTROLCARD and visible ones like AUTOBACKUP), any MCSs or DBSs, and any user jobs to be run in batch space. (Many installations prefer to run virtually all user work in swapspace.)

The swapspace may be made up of any number of pieces of memory. These "chunks" may be located anywhere in memory. This feature allows dynamic growth of the swapspace and prevents "holes" in memory from severely affecting site operations.

Even though the swapspace can be divided into several different pieces, each individual subspace must still reside in a contiguous area of memory. A segmented swapspace may not be able to handle the same number of tasks as a nonsegmented swapspace with the same number of slots. For example, a swapspace with one chunk of 90 slots can run three tasks of 30 slots each simultaneously; a swapspace with a 50-slot chunk and a 40-slot chunk can run, at most, two 30-slot tasks simultaneously. The MINCHUNKSIZE parameter controls this situation. This parameter specifies the smallest amount of memory that a single chunk of swapspace may contain (MAXCORE <= MINCHUNKSIZE). By setting the MINCHUNKSIZE attribute to the value of the CORESIZE attribute, a single contiguous area of memory is allocated.

MAXCORE determines how much of the swapspace can be used by any single task; it may be used to prevent one task from using an excessive amount of the swapspace.

EXPRESERVE determines how much of the swapspace is set aside for express service. Rational choice of this value depends on the size of the typical express job, on its duty cycle (how much time in memory followed by how much time waiting on SWAPDISK for another input), and on the number of such jobs to be accommodated.

EXPMAXCORE determines the largest memory requirement necessary for a task to hold express status. It should be set to accommodate the largest tasks for which express handling is intended.

The following inequalities must hold:

MAXCORE <= MIN(largest AREASIZE,CORESIZE,MINCHUNKSIZE)

EXPRESERVE <= CORESIZE-MAXCORE

EXPMAXCORE <= EXPRESERVE

An attempt to change one of these parameters is rejected if its inequality would be violated. When one parameter is changed, those dependent on it are changed, if necessary, to preserve their inequalities. Thus, specifying EXPRESERVE may change EXPMAXCORE; specifying MAXCORE may change both EXPRESERVE and EXPMAXCORE.

## Changing the Swapspace Size via the CORESIZE Parameter

The swapspace size may be changed in order to cause SWAPPER to accept the new value of the parameter and attempt to bring the swapspace into conformance with this value. The new value is preserved so that SWAPPER can get the exact amount of swapspace desired the next time it is initialized. The amount of swapspace actually in use can be determined at any time from the value displayed next to ACTUALCORESIZE in response to the SW keyin.

### EXPANDING THE SWAPSPACE

When the value of the CORESIZE parameter is set greater than the current value of ACTUALCORESIZE, SWAPPER attempts to expand the amount of swapspace available by getting one or more chunks of memory and inserting these chunks into SWAPPER's tables. These new chunks of memory must be at least as large as the value of the MINCHUNKSIZE parameter. If the difference between the new value of CORESIZE and the current value of ACTUALCORESIZE is less than MINCHUNKSIZE, no increase in swapspace can be performed.

### SHRINKING THE SWAPSPACE

When the value of the CORESIZE parameter is set smaller than the current value of ACTUALCORESIZE, SWAPPER attempts to shrink the amount of swapspace available by shrinking or removing individual chunks of memory. No chunks of memory may be smaller than the MINCHUNKSIZE parameter. In some situations, SWAPPER is unable to shrink the swapspace to a size as small as that desired (SWAPPER never shrinks the swapspace to a size smaller than CORESIZE). The current algorithm specifies that the smallest chunks of swapspace are shrunk or discarded first.

## SWAPPER Initialization and Swapspace Size

At initialization time, if SWAPPER is unable to obtain the amount of swapspace specified by the CORESIZE parameter, an RSVP is issued notifying the operator of the situation. The operator may:

1. Reply DS to the RSVP, aborting SWAPPER initialization.

2. Reply OK, causing SWAPPER to again attempt to find the required memory. (SWAPPER makes this search automatically in the absence of operator action.)

3. Reply NOTOK, causing SWAPPER to continue initialization using whatever memory it has already obtained.

4. Enter a new value for the CORESIZE parameter using the SW ODT message. SWAPPER then attempts to obtain the new amount of memory.

AREASIZE and AREAS determine the amount and subdivision of each swapdisk file; enough space must be present for all swap tasks that are currently active.

## Swapping on Multiple Families

SWAPPER has the ability to use more than one family and more than one SYSTEM/SWAPDISK file on which to swap. The SW CREATE ODT message is used to create a swapdisk file on the desired family. The FAMILY option on the SW message is used to dynamically add to or subtract from the list of packs SWAPPER uses, as follows:

SW FAMILY + KSID

This message causes SWAPPER to find the file SYSTEM/SWAPDISK on KSID. This file is opened and made available to SWAPPER.

SW FAMILY - KSID

This message causes SWAPPER to cease using the family KSID. Also, all tasks that are currently swapped out on the family KSID are swapped in (from KSID) and then swapped out again (to a family other than KSID). When no more tasks are resident on KSID, the file SYSTEM/SWAPDISK on KSID is closed and the operator is notified.

## Multiple Families on a Non-Monolithic System

On a non-monolithic system, each processor running SWAPPER maintains its own list of families available to SWAPPER; thus, swapping is possible on a family that is not exchanged to every processor running SWAPPER. To support process family migration, at least one family of swapdisk must be usable by all processors in which the migratory task will run.

## Multiple Swapdisk Families with Different Row Sizes

The row sizes for each swapdisk family need not be the same; however, SWAPPER requires that each I/O operation fit entirely within a single row of a swapdisk. The maximum possible value of the MAXIOSIZE parameter is the size of the largest row of swapdisk available. Thus, an installation can segregate some classes of SWAPPER I/Os by setting the row size of some swapdisk families smaller, preventing subspaces with I/Os larger than the row size from being swapped on that family. This capability can cause an "out of swapdisk" condition to occur whenever insufficient space remains on swapdisk(s) with large enough row sizes, even though space may be available on other swapdisk families. This situation can be relieved by either adding another family on which to swap (with a large enough row size) or decreasing the MAXIOSIZE parameter.

## Out of Swapdisk Mechanism

When SWAPPER runs out of swapdisk, an independent runner is forked which then waits on an RSVP notifying the operator that more swapdisk is needed. The operator can either wait for the situation to repair itself (for a job in memory to complete) or add a new family available to SWAPPER via the SW ODT message. When running on a non-monolithic system, amounts of certain kinds of swapdisk may be exhausted. Because a migratory task requires swapdisk that is used by each of the processors in which the task can run, this task may be out of swapdisk even though space may remain on another family that is not being used by all of the processors. When this situation arises, the OUTOFSWAPDISK independent runner displays the list of processor visibility swapdisks that are needed.

**Example**

```
--- 1 WAITING ENTRY ---
3496/3496 OUTOFSWAPDISK
3496 ** OUT OF SWAPDISK FOR PROC(S) (1),(2,3) **
```

The above example notifies the operator that some swapdisk is required that is usable by processor 1 and some swapdisk is required that is usable by processors 2 and 3.

## I/O ERROR HANDLING MECHANISM

SWAPPER has two distinct algorithms for dealing with I/O errors.

### I/O Errors on Write Operations

After receiving an I/O error on a write operation, SWAPPER marks the area of swapdisk in question as bad and attempts to retry the write operation to a different area of swapdisk (possibly on a different family). The table of areas of bad swapdisk is stored in the SYSTEM/SWAPDISK file itself and is referenced any time SWAPPER opens this file. To clear the table, the swapdisk file is removed and then recreated.

### I/O Errors on Read Operations

If an I/O error occurs on a read operation, the affected task is stuck on disk and cannot be DSed. When this situation occurs, SWAPPER writes a message into the job log for the affected task and notifies the operator via an RSVP. If the operator wishes to retry the I/O operation (for example, if a pack goes not ready and is then on a different spindle), an appropriate reply to the RSVP causes a retry. If the situation is not corrected (another I/O error occurs), the above process is repeated.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# 3. SWAPPING

## SWAPPING OUT

Tasks are swapped from memory to disk for the following reasons:

1. An attempt is made to obtain input from a Data Comm system when no input is available.
2. The Data Comm output buffers for a file are full and more OUTPUT is attempted.
3. The task has been suspended.
4. The allocated time-slice has been exceeded.
5. The swapspace size of the task is too small.
6. An event wait exceeds two seconds.
7. An ALGOL WHEN (or equivalent statement) exceeding two seconds is executed.
8. An ALGOL CALL or CONTINUE (or equivalent statement) is executed, and control does not return to the task within two seconds.

## SWAPPING IN

Tasks are candidates to be swapped from disk to memory when the conditions that caused the swap-out have been satisfied and sufficient memory exists in the swap area.

The queue priority is used to select tasks for swap-in. New jobs are given artificially high priorities to get started; DSed tasks get artificially high priorities in order to remove them.

## EXPRESS TASKS

An express task is swapped out when its process time-slice of EXPTIME seconds is exceeded. Any task whose memory requirement does not exceed EXPMAXCORE is considered an express task whenever it is interactive. A task may switch from interactive to non-interactive and back. If the memory allocation of a task grows beyond EXPMAXCORE, the task ceases to be express.

Any new task is considered interactive and remains interactive as long as it has not been time-sliced. A task becomes interactive if it is swapped out on a demand basis (that is, if it attempts a Data Comm write but its buffer allocation is in use, or it is waiting for time or for an external event). A task ceases to be interactive if its time-slice expires; at this point, it is swapped out if any other tasks are requesting memory to be swapped in.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# 4. EXECUTING TASKS IN SWAPSPACE

The specification required to execute tasks in swapspace is the SUBSPACES control statement. The variants of the SUBSPACES statement are listed below:

| | |
|---|---|
| SUBSPACES = 0 | Do not run in SWAPSPACE |
| SUBSPACES = 1 | Place only the task data in swapspace. |
| SUBSPACES = 2 | Data is in SWAPSPACE; code also if this is a user program. |
| SUBSPACES = 3 | Data and code are placed in SWAPSPACE. |

If SUBSPACES=2 was specified and the program code file is under the user's own USERCODE, then SUBSPACES is set to three. If the program code file is under a different (or no) usercode, SUBSPACES is set to one.

SUBSPACE statements may be specified at compile time via WFL. Tasks may not be executed in swapspace for the following reasons:

1. SWAPPER is not running or is in the process of terminating.

2. The code file is compiled prior to 3.0, might have up-level pointers, is marked IPC-capable, and the task OPTION word does not have the IPCOVERRIDE bit set. This bit may be set in WFL or CANDE (that is, OPTION=IPCOVERRIDE); it is reset by default.

3. The SUBSYSTEM attribute for the task specifies a set of processors in which no SWAPPER is currently running.

Denial of a request to run in swapspace causes the task attribute SUBSPACES to be set to zero.

For IPC families, if the critical block for a dependent process is in a swappable stack, that process must also be swapped.

The interpretation of SUBSPACES for libraries, database stacks, and job stacks is as follows:

1. Libraries. The SUBSPACES attribute can be specified at compile time through WFL. If no value is specified at compile time, a library with a sharing class of PRIVATE inherits the value of the SUBSPACES attribute from the tasks that caused the library to be initiated.

2. Databases. The SUBSPACES attribute can only be specified at compile time in the ACCESSROUTINES code file. The SUBSPACES attribute for a database is never inherited.

3. Jobs. Only SUBSPACES values 0 and 3 are meaningful for a job stack. The attribute can be specified on the queue through which the job runs or on the job header. If either of these methods is used, the attribute is inherited by all tasks run out of the job. Operational overhead can be reduced by specifying the option SWAPALLJOBS (Option 32), which causes all job stacks to run in swapspace without forcing inheritance of the SUBSPACES attribute on the tasks run out of a job stack; thus, an installation could choose to run all job stacks, but not other stacks, in swapspace.

## SWAPPER

Tasks run by CANDE are run in swapspace if SWAPPER is running. By default, compilers and system utility programs are run with SUBSPACES=1, with the D1 stacks in main memory and the D2 stacks in subspace. User programs are run with SUBSPACES=2. If the code file is in a system directory, the code is re-entrant and only data are swapped. The execution part of a compile and go RUN command is SUBSPACES=3 where both code and data are swapped. If the CANDE option SWAPALL(12) is set, CANDE sets SUBSPACES=3 for all tasks.

# 5. SYSTEM/SWAPDISK

THIS SECTION HAS BEEN DELETED

THIS PAGE INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES

# APPENDIX 16A. SWAPPER ERROR MESSAGES

| | |
|---|---|
| CORE NOT AVAILABLE | During SWAPPER initialization, SWAPPER could not get MINCHUNKSIZE slots of contiguous memory. SWAPPER terminates; SW must be input again, and another attempt to gain the needed space must be initiated. |
| EXPRESS MAX CORE NOT VALID | Indicates that the value specified for EXPMAXCORE is not valid. EXPMAXCORE must be less than, or equal to, the value of EXPRESERVE. |
| EXPRESS RESERVED CORE NOT VALID | Indicates that the value specified for EXPRESERVE is not valid. EXPRESERVE must be less than, or equal to, the difference between CORESIZE and MAXCORE. |
| GOING AWAY | Indicates that SWAPPER has been terminated (SW-) and will go away when the last swap job goes away. |
| UNRECOGNIZED REQUEST | Indicates that SWAPPER was not able to recognize the message entered. |
| PRIORITYBIAS NOT VALID | Indicates that the value of PRIORITYBIAS is greater than 100 or less than zero. |
| UTILIZATIONBIAS NOT VALID | Indicates that the value of UTILIZTIONBIAS is greater than 50 or less than zero. |
| MAX CORE SIZE NOT VALID | Indicates that the value specified for MAXCORE is not valid. MAXCORE may not be less than two, or greater than the number of memory slots in the total SUBSPACE (CORESIZE). |
| MAX SLICE NUMBER NOT VALID | Indicates that MAXSLICENUMBER is outside the range 0-255. |
| MIN TIME SLICE NOT VALID | Indicates that MINTIMESLICE is less than zero. |
| | Text Deleted. |
| OUT OF SWAPDISK | Indicates that SWAPPER is out of swapdisk for the current mix situation. The size of swapdisk should be increased if this problem occurs often. |
| SLICING RATIO NOT VALID | Indicates that the value specified for RATIO is a non-zero value less than one. |
| SWAPCORE NOT VALID (TOO SMALL) | Indicates that the CORESIZE parameter is too small. |

Text Deleted.

SYSTEM/SWAPDISK IS CRUNCHED
: During SWAPPER initialization, the SYSTEM/SWAPDISK file was found to be a CRUNCHed file.

SYSTEM/SWAPDISK NOT THERE
: Indicates that SWAPPER has checked the directory, and no file is present on disk with the name SYSTEM/SWAPDISK.

SYSTEM/SWAPDISK VALIDITY CHECK FAILURE
: Indicates that the first three words of record 0 of SYSTEM/SWAPDISK do not contain the EBCDIC string "SYSTEM/SWAPDISK. ".

# APPENDIX 16B. SWAPPER MECHANISM

This appendix contains a simplified overview of the SWAPPER mechanism.

SWAPPER is an independent runner of the MCP initiated by the operator input message SW+. SWAPPER is queue-driven and performs the actual swapping of tasks to and from memory.

The following steps occur after selecting a task to be swapped into memory:

1. The task is read from disk to memory.

2. Absolute memory addresses are relocated for:
   a. Logical I/O buffers.
   b. Direct I/O buffers.
   c. data and segment descriptions.
   d. SORT work areas if task was swapped out while sorting.
   e. data management areas.
   f. memory links.

3. At least one 990-word slot is added to the task memory areas if task was swapped out because of insufficient memory, or if overlay activity had been so high as to indicate that more space was needed.

4. The swapdisk space to which this task was written is deallocated.

The following MCP and SWAPPER actions occur automatically when swapping a task from memory to disk:

1. If the independent runner OLAYSCOUT is running for this stack in order to add overlay space for this task, the stack is marked as not needing overlay space, and OLAYSCOUT completes.

2. If the task requires more memory to run effectively, the task is marked for memory expansion on the next swap-in.

3. If the task is being time-sliced, the slice number is incremented.

4. The next time-slice of the task is computed.

5. If a task is not being swapped out for a time-slice, the slice-number counter of the task is zeroed. The task is now considered an express task if it is small enough.

6. The swapdisk space in which the task is to be written is allocated.

7. The present COPY descriptors whose memory areas reside outside swapspace are sought. The memory areas are marked as absent and pointing to the MOM descriptor.

8. The task is written to swapdisk.

# APPENDIX 16C. SYSTEMSTATUS INFORMATION

## STATISTICS

The SWAPPER mechanism records several times and counts that may be monitored via the SYSTEMSTATUS intrinsic. This feature is provided as a development tool and is subject to major change in any subsequent release, without prior notice.

The contents of this section have been deleted. Refer to the B 5000/B 6000/B 7000 Series SYSTEMSTATUS Reference Manual for details on SYSTEMSTATUS information.

THIS PAGE INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES

A sequence of three-word entries follows the transition matrix. One entry is present for each swap task in the mix. For K=0,1,...A[2]-1, let X=3+A[3]+3*K. The data returned are as follows:

| Word | Field | Type* | Name and Meaning |
|---|---|---|---|
| X | 47: 1 | | Task was time-sliced the last time it was swapped out. |
| | 46: 1 | | Task is an express task. |
| | 45: 1 | | D1 stack and code are in the subspace. |
| | 31: 5 | I | LOCATION (state) index. |
| | 23: 4 | I | Box number of swap task. |
| | 15:16 | I | Mix number of swap task. |
| X+1 | 47: 8 | I | Contains the number of additional slots needed. |
| | 39:10 | I | Contains the current length of the subspace in slots. |
| | 29:10 | I | Contains the slot index+1 where the subspace begins in swapspace; zero if no memory is allocated for this task. |
| | 19:20 | I | Contains the swapdisk address for the subspace; zero if no disk is allocated for this task.<br>[19:10] is the disk row index.<br>[ 9:10] is the slot index within that row.<br>(Disk slot 0,0 is never allocated; it holds the SWAPPER parameters.) |
| X+2 | 47: 5 | I | Contains the composite state index (0 thru 17) of the state last entered by this swap task. |
| | 42:43 | I | Contains the time (CLOCK value) when the task entered that state. |

*I = Integer

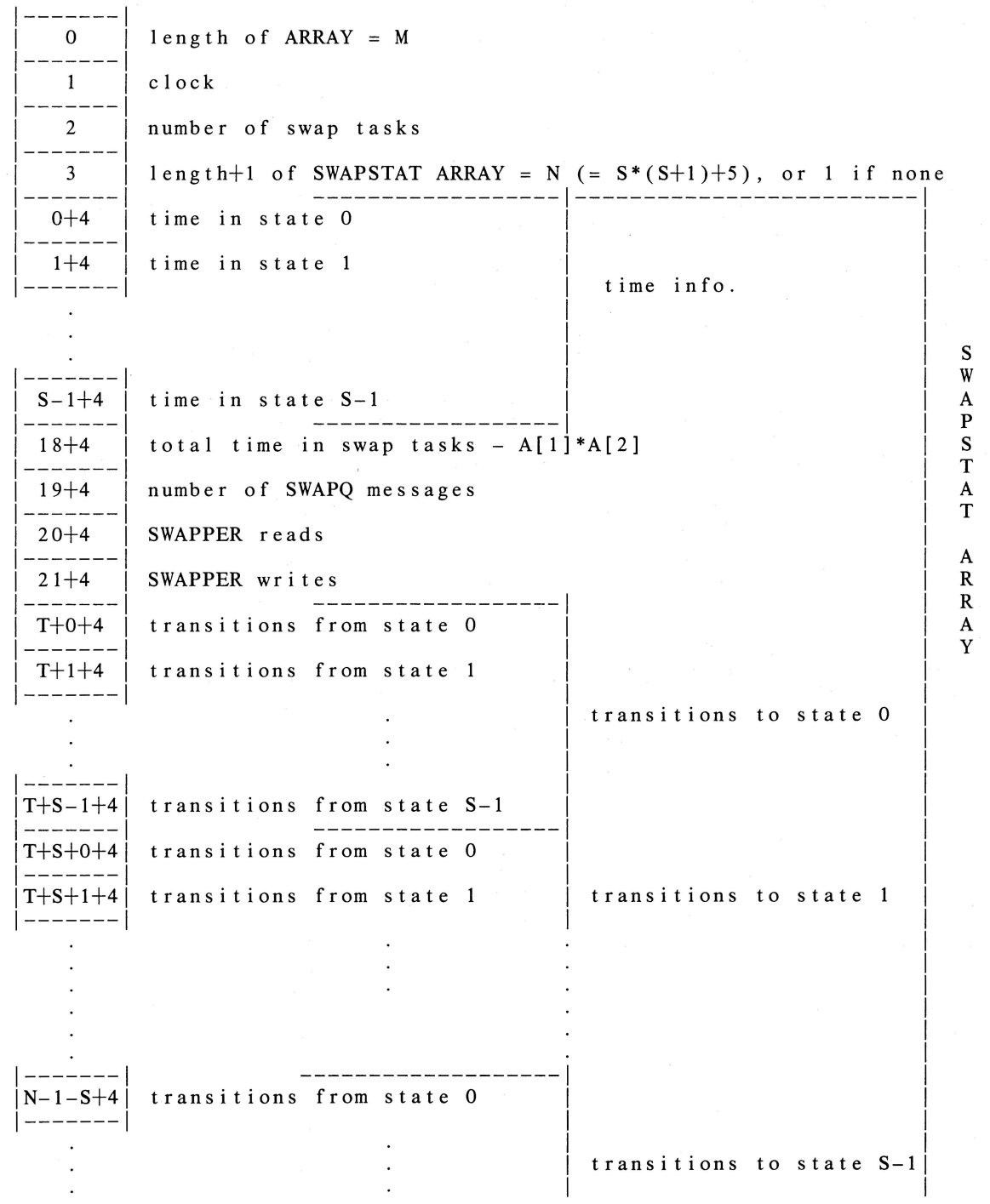
The swap task is in state I=A[X+2].[47: 5]. The length of time it has been in this state is A[1]-A[X+2].[42:43]. This value may be added to A[4+I] to accumulate the total time swap tasks have spent in this state.

The organization of the SYSTEMSTATUS array is illustrated by Figure 16C-1.

S=18 is the number of composite swap states.

T=S+4 is the number of accumulators before the transition matrix.

**Figure 16C-1. SYSTEMSTATUS ARRAY**



SWAPPER

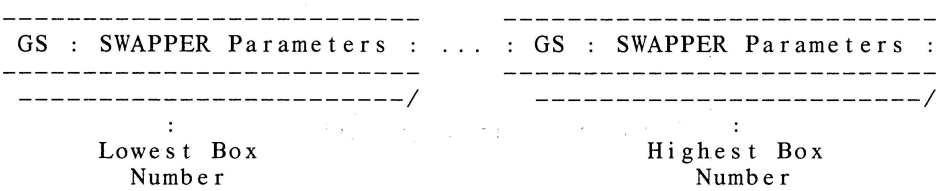
| | | |
|---|---|---|
| N-1-1+4 | transitions from state S-1 | |
| N+3 | | |
| N+3+1 | | info. for 1st swap task |
| N+3+2 | | |
| N+6 | | |
| N+6+1 | | info. for 2nd swap task |
| . | . | |
| . | . | |
| . | . | |
| M-3 | | |
| M-2 | | info. for last swap task |
| M-1 | | |

The number of swap tasks is also returned by SYSTEMSTATUS case 2 in word 31 of the array (and word 31 of the "operational" group of case 0).

The count includes all swap tasks, even those which are being staged and those which are being terminated. Thus, not all swap tasks are counted in mixcount (SYSTEMSTATUS case 2 word 6), which includes only tasks that have completed INITIATE and have not yet completed NORMALEOJ.

## PARAMETERS

A Type 11 SYSTEMSTATUS call returns SWAPPER parameters. The structure of the information returned by this call is as follows:




Type 11 is not affected by the bitmask in V1; it returns data from all boxes in which SWAPPER is active. M is a bitmask of these boxes.

The following parameters are returned with a Type 11 call:

| Word | Type* | Name and Meaning |
|---|---|---|
| 0 | | Group-size word |
| 1 | I | SWAPCOREMAX in words |
| 2 | R | MINTIMESLICE in seconds |
| 3 | I | MAXSLICENR |
| 4 | R | RATIO |
| 5 | I | MAXCORE in words |
| 6 | I | EXPRESERVE in words |
| 7 | I | EXPMAXCORE in words |
| 8 | I | PRIORITYBIAS 0-99 |
| 9 | I | UTILIZATIONBIAS 0-99 |
| 10 | R | EXPRESSMAXTIME in seconds |
| 11 | R | IOBIAS |
| 12 | R | MEMORYBIAS |

*I = Integer
R = Real

# APPENDIX 16D. QUEUE PRIORITY

## QUEUE PRIORITY EFFECT

To measure the effect of queue priority, a mix consisting of three identical job streams is run at priorities of 30, 50 and 70. Each job consists of an ALGOL compile, a COBOL compile, a FORTRAN compile, and a bind and runs in swapspace. The elapsed times of the jobs are given in Table 16D-1 in terms of the priority bias value.

Table 16D-1. Queue Priority Effect

| priority bias | Elapsed times: job priority 30 | 50 | 70 |
|---|---|---|---|
| 50 | 47:11 | 27:15 | 11:45 |
| 25 | 26:23 | 21:52 | 13:58 |
| 10 | 25:25 | 23:38 | 16:54 |

To measure the effect of interactivity, a mix of tasks of various degrees of interactivity is run. The total process time accumulated by the task during the test (14 minutes elapsed) is measured with UTILIZATIONBIAS at zero (no bias for interactive tasks) and compared to a run with UTILIZATIONBIAS set to 100 (maximum bias to interactive tasks). The interactiveness is determined by the specification of process time to be used before requesting a swap-out, using an ALGOL WHEN. The effect of interactivity is given in Table 16D-2.

Table 16D-2. The Interactivity Effect

| Process time before swap-out | Total process time utilization bias 0 | Total process time utilization bias 100 |
|---|---|---|
| 0 | 2.1 | 4.0 |
| .25 | 61 | 83 |
| .7 | 107 | 84 |
| 2 | 136 | 98 |
| 4 | 144 | 136 |
| 6 | 150 | 90 |

## QUEUE PRIORITY VALUE

```
INTERACTIVITY:=IF DEMANDSWAP
                  THEN MAX(PTNOTUSED/PTSLICE,ETNOTUSED/ETSLICE)
                  ELSE -SLICENR/MAXSLICENUMBER

SWAPPRIORITY:=DECLAREDPRIORITY / 100

QPRIORITY:= ( (INTERACTIVITY + 1) / 2 * UTILIZATIONBIAS )
            + ( SWAPPRIORITY * PRIORITYBIAS )

        where  0 <= PRIORITYBIAS <= 100
               0 <= UTILIZATIONBIAS <= 100

        so     0 <= QPRIORITY <= 200
```

If the task whose queue priority is being calculated is a member of an IPC family, its QPRIORITY is further adjusted by considering the values of the parameters IOBIAS and MEMORYBIAS, as follows:

```
QPRIORITY:= MIN(255, MAX(0,QPRIORITY - ((IOBIAS * N)
                        + (MEMORYBIAS * (M/MAXCORE)))
                        )
                )
```

In the above values, N equals the number of I/O operations required to be in all of the swappable parents of the task, and M equals the number of memory slots required to hold all the swappable parents of the task.

SWAPPER performance previous to 2.9 may be reasonably approximated by using the default parameter values of:

```
PRIORITYBIAS = 0
UTILIZATIONBIAS = 6
MAXSLICENUMBER = 7
MINTIMESLICE = 3
IOBIAS = 0
MEMORYBIAS = 0
```

These values are offered to permit continuity with previous performance; they are not recommended for any particular mix.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# APPENDIX 16E. B 5000/B 6000 TIGHTLY-COUPLED

On a B 5000/B 6000 multiprocessor system, a SWAPPER independent runner and its associated swapspace may be initiated in each memory subsystem ("box") except Global Memory. Each SWAPPER is independent in the sense that it maintains its own set of parameters in the Halt/Load unit directory and may be initiated and terminated independently of SWAPPERs running in other subsystems. However, all such SWAPPER independent runners must be visible to at least one family containing a swapdisk file; therefore, each row of the file must reside on a unit that is exchanged to each processor in which a SWAPPER is to be run.

## The SUBSYSTEM Task Attribute

The SUBSYSTEM task attribute is used to specify a subset of the memory subsystems where this task may best be run. For batch tasks, one of the subsystems is chosen from the specified set and then permanently placed in that subsystem. (No capability for moving batch tasks between different subsystems exists.) For swap tasks, the meaning of this attribute is not changed, but the capability of moving the task is readily available. (The task is swapped out of one subsystem and into another. This capability requires a *SYSTEM/SWAPDISK to be exchanged to all processors running SWAPPER.) If a task requests to be run in swapspace (task SUBSPACES NEQ 0), this task is run in any of the swapspaces that are in the subset of memory subsystems specified by the SUBSYSTEM attribute.

### Example

Task A is started with SUBSPACES=3 and SUBSYSTEM=BOTH, where BOTH is setup to processors 1 and 3. At task initiation time, a SWAPPER is present only in processor 3. This task would be run as a swap task constrained to processor 3. (Even if there were a SWAPPER in processor 2, A would never run there.) Subsequently, if a SWAPPER is started in processor 1, task A can be swapped into either processor 1 or 3. The SWAPPER in processor 3 might even be shut down; in this case, A would only run in processor 1.

## SUBSYSTEM vs SUBSPACES: Conflicting Specifications

Whenever a conflict arises between the specifications of the SUBSYSTEM and SUBSPACES task attributes, the SUBSYSTEM specification is considered more binding than SUBSPACES. In case of conflict, the SUBSPACES attribute is modified in order to make it consistent. Some examples of conflicts and their resolution are as follows:

1. SUBSPACES NEQ 0 but no SWAPPER is running in the specified subsystem. In this case, remedial action is to assume SUBSPACES=0.

2. SUBSYSTEM specifies more than one processor and SUBSPACES=1. The inconsistency in this situation is that the user has requested that the object code be placed in non-swap memory but has said that the task can run in more than one processor. In this case SUBSPACES=3 is assumed, and a warning message is logged for the task. This situation can also occur if SUBSPACES=2 is in effect.

## Process Family Migration on a Non-Monolithic System

Just as a single swap task may migrate between different swapspaces (on different processors), an IPC family of swap tasks may also migrate. As with a single task, migration is controlled by the SUBSYSTEM task attribute. However, the full generality of the subsystem specification mechanism is not supported. The following restriction is made:

> Any task spawned by a swappable task runs in the same set of subsystems as that in which the parent runs.

An offspring of a swappable task inherits the subsystem specification of the parent regardless of any user specification that might be present.

## Swap Tasks and Nonexchanged Units

On a B 5000/B 6000 multiprocessor system, whenever a task attempts to use a peripheral that is not exchanged to the processor running the task, the buffer for the I/O operation is moved into Global Memory and the I/O operation is performed by one of the processors to which the unit is exchanged. For swappable tasks, the buffer is moved into Global Memory if the peripheral is not exchanged to every processor in which this task might run. (Processors in which SWAPPER is not currently running are also included in this test because a SWAPPER might be started in that processor at some point in the future.) In order to reduce the amount of Global Memory used, any swap tasks that use nonexchanged units should have a subsystem specified which includes only those processors to which the peripherals are exchanged.

# APPENDIX 16F. B 7000 TIGHTLY COUPLED

## Limitations

The SWAPPER independent runner can only be initiated on a single local box due to IOM/unit visibility on Tightly Coupled configurations. The SW+(n,m) is valid and is accepted, but the copy of SWAPPER which is initiated in the local box(es) which has no visibility to all family members containing swapdisk file rows will terminate itself with:

SYSTEM/SWAPDISK ON NON-EXCHANGED UNIT

As only one SWAPPER can exist in a Tightly Coupled environment, it is not possible for a swap task to migrate from one local box to another as is the case on B 5000/B 6000 systems.

All other pragmatics as to possible SUBSYSTEM/SUBSPACES conflicts match those noted in Appendix 16E.

## Swap Tasks and Nonexchanged Units

The same rules/actions apply to B7000 as do on the B 5000/B 6000 series, see Appendix E.

THIS PAGE LEFT INTENTIONALLY BLANK FOR FORMATTING PURPOSES

SUPERIV (B 5000/B 6000 series Systems Only)

TABLE OF CONTENTS

1. INTRODUCTION . . . . . 17-1- 1

2. OPERATING INSTRUCTIONS . . . . . 17-2- 1

# 1. INTRODUCTION

SUPERIV is a system maintenance routine used to initialize and verify (IV) model 215 and 225 disk packs.

This section of the SOG docments the operational procedures for using this maintenance tool.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# 2. OPERATING INSTRUCTIONS

1. The SYSTEST/SCR/SUPERIV file is COPY&COMPAREd from the SYSTESTS tape to the Halt/Load unit using LIBRARY/MAINTENANCE.

2. The pack to be IVed is closed using the ODT CLOSE command.

3. The pack unit to be IVed is reserved using the ODT UR command.

4. The following statement is entered on the ODT to execute the SUPERIV program:

   SCR; RUN "SYSTEST/SCR/SUPERIV"

   The following list contains the questions asked by the program and the appropriate responses. The items contained in brackets < > are symbolic and must be replaced with the correct data.

| Question | Response |
|---|---|
| ENTER <MIX>HI<MPX#> | The multiplexor number through which the IV is to be routed is entered. |
| ENTER <MIX>HI<PK#> | The pack unit number to be IVed is entered. |
| ENTER <MIX>HI<SER #> | The serial number to be assigned to the IVed pack (up to six digits) is entered. |
| SET HEADS CENTER | Entry of 1 causes the program to request 12 display entries for head offset for each verify. Entry of 2 causes the program to run without the head offset requests. |
| ENTER <MIX>HI<1 OR 2> | Entry of 2 allows the program to run in an unattended mode. Entry of 1 causes the program to request ODT input at the end of each pass. Program execution is functionally the same in both cases. |

During program execution, entering <MIX>HI causes the program to display SUPERIV PASS <N> CYL <NNN> REL <NNN> on the ODT. Thus, the progress of the program (PASS <N> CYLINDER <NNN>) may be monitored and the number of relocated sectors (REL <NNN>) determined.

5. When the SUPERIV program has completed, the pack may be made available to the system as follows:

   a. The ODT UA command is entered to terminate the reserved status of the unit.

   b. The ODT RC command is entered to assign a name to the pack and make the pack available for system use.

# RECONFIGURATION

## TABLE OF CONTENTS


MODULE SELECTION . . . . . 18-1- 1

INTRODUCTION . . . . . 18-1- 1

SYSTEM INITIALIZATION . . . . . 18-1- 2

MEMORY CONTROL MODULE (MCM) . . . . . 18-1- 2

CENTRAL PROCESSOR MODULE (CPM) . . . . . 18-1- 4

INPUT OUTPUT MODULE (IOM) . . . . . 18-1- 6

RECONFIGURATION . . . . . 18-2- 1

ON-LINE MODULE DECOMMIT/RECOMMIT . . . . . 18-2- 1

CPM DECOMMIT/RECOMMIT . . . . . 18-2- 1

IOM DECOMMIT/RECOMMIT . . . . . 18-2- 3

MCM DECOMMIT/RECOMMIT . . . . . 18-2- 5

SYSTEM HALTED RECONFIGURATION . . . . . 18-2- 8

PERIPHERAL RECONFIGURATION . . . . . 18-2- 9

UNIT DELETION . . . . . 18-2- 9

PATH DELETION . . . . . 18-2- 9

UNIT ADDITION . . . . . 18-2- 10

PATH ADDITION . . . . . 18-2- 10

DCP RECONFIGURATION . . . . . 18-2- 11

# MODULE SELECTION

## INTRODUCTION

The following describes where the various switches are found. These switches must be set to something, regardless of whether you are partitioning or not. In general:

ON THE MCM

The inhibits are set to allow access by CPMs and IOMs in the system or partition.

The availability mask reflects the real status of the MCM and is usually set by the Site Engineer.

The limits are usually set such that all MCMs in the partition or system form a continuum of memory. There must be an MCM with a lower limit of zero.

The MCM is usually "ON-LINE".

The partition switch must be set.

ON THE CPM

The CPM is "ON-LINE", the memory address switches are down (zero).

The partition switch is set correctly.

ON THE IOM

The IOM is "ON-LINE", the memory address switches are down (zero).

The partition is set correctly.

Only one IOM in the system or partition has the select switch set.

The CPM set on the switches must be in the partition and "ON-LINE".

The remaining switches specify a reader and an EU on this IOM to be used if this IOM is "selected".

## SYSTEM INITIALIZATION

All mainframe components should be examined to insure that their various switches are in the correct positions.

All ODTs must be in local or receive mode, not in transmit.

The parameter deck for the loader should be checked to insure that the unit table and partition cards reflect the system desired.

The loader deck, followed by its parameter deck, is then placed in the "SYSTEM" card reader and a "CARD LOAD" is done from the console.

The loader will find the MCP specified in the deck and load it to "SYSTEM" family. It will build a unit table and partition word and load these on "SYSTEM" family. Termination of system loader is via a "SOFT" halt load of the MCP just loaded. Subsequent halt loads may be done from the console.

Each mainframe module (MCM, CPM, IOM) has various switches that control its relationship to the system environment. Those which are of direct concern to the configuration are described in the following paragraphs.

## MEMORY CONTROL MODULE (MCM)

On the MCM panel, there are five sets of switches which are of direct concern to the configurer. These switches are as follows:

REQUESTER INHIBITS

This is a set of eight switches, numbered 0 thru 7, which provide a means of allowing or disallowing access to an MCM by CPMs and IOMs on the system. Each switch controls access by a particular IOM or CPM. When the switch is up, the CPM or IOM does not have access. When the switch is down, it does have access. (The switches themselves do not control the access -- see LOCAL/REMOTE.)

AVAILABILITY MASK

This is a set of switches that provide a means of specifying which of the possible memory stacks in an MCM are to actually be used. Controlling the number of stacks affects two things, multiple word phasing and the maximum memory available.

When the switch is down, the stack will not be used. When it is up, the stack will be used. (The switches themselves do not cause access, see LOCAL/REMOTE, etc.)

# MODULE SELECTION

## ADDRESS LIMITS

These are two sets of six switches each of which provides a means of specifying what memory addresses are to be used for the memory available in that MCM. The six switches represent the six most significant bits of the 20-bit memory address.

An MCM, regardless of its MCM designation, may be used for any contiguous slice of memory. The maximum amount of memory selectable is that which is available (see Availability Mask). When it is down, the bit is reset. (The switches themselves do not cause the limits to be used, see LOCAL/REMOTE, etc.).

A convenient way of determining the correct switch settings for any memory configuration is as follows:

First MCM

Lower Address: 0
Upper Address: ((No. of stacks) x 4) - 1

Additional MCMs

Lower Address: (Upper Address Limit from Last MCM) + 1
Upper Address: ((No. of stacks) x 4) - 1) + (Lower Address Limit of this MCM)

Examples are as follows:

| | LOWER | UPPER | STACKS |
|---|---|---|---|
| 1st MCM | 000000 | 001111 | 4 |
| 2nd MCM | 010000 | 010111 | 2 |
| 3rd MCM | 011000 | 100111 | 4 |

| | LOWER | UPPER | STACKS |
|---|---|---|---|
| 1st MCM | 000000 | 000111 | 2 |
| 2nd MCM | 001000 | 001111 | 2 |

## ON-LINE/TEST

When an MCM is on-line, the panel is disabled; the switches do not do anything. The MCM will respond to signals from the console.

When it is in test, it is not necessarily off-line; however, some of the other switches are enabled. The MCM will not respond to signals from the console.

## LOCAL/REMOTE

This switch works when the MCM is in "Test".

When it is in "Remote", the MCM will respond to IOMs or CPMs, but not to console signals.

When it is in "Local", the MCM will not respond to any other components; it is "off-line". In addition, all panel switches are operable.

CONTROL CLEAR

This push button is operable when the MCM is in "Local". Depressing this button causes two things -- the MCM is cleared and the inhibit, Availability and Address Limit registers are loaded from their respective switches. This second part causes the various switch settings discussed above to be actually used by the MCM. The contents of the registers are displayed on the panel.

A-B PARTITION SWITCH

This switch is located to the right of the MCM panel. It selects partition A or B. It allows or disallows general clear from the console, depending on whether A or B is selected at the console. A clear signal from the console operates similarly to the "Control Clear" button in that the Inhibit, Availability and Address Limits are loaded.

NOTE When an MCM is not "On-Line" or when the switch is in the center position, it will not respond to the console.

## CENTRAL PROCESSOR MODULE (CPM)

Switches must be correctly set on panel 1, panel 2 and on the cold start/halt load selection card which is located on the backplane of the CPM.

COLD START/HALT LOAD SELECTION CARD

The selection card is a circuit board which is located on the back of the CPM near the floor on the right-hand end of the cables.

A/B This toggle switch determines which portion of the system (System A or System B) this module belongs to. The switch selects partition A or B. It allows or disallows the Halt, Clear and Load signals from the console, depending on whether A or B is selected.

MEMORY NO This switch indicates the 16K memory segment to be used for initialization (most significant six bits of the memory address). The switches must be down (address zero).

No other switches on the CPM selection card are used.

PANEL 2 SWITCHES

The following switches are located on panel 2 (the left-hand panel) on the front of the CPM.

MDU COMP CONTROL, PROGRAM, EU STORAGE, STACK, ASSOCIATIVE MEMORY, DISPLAY and BUFFER CONTROL

These toggle switches should be left in the down position.

The following switches must be reset (off):

MDU COMPARE OR
MDU COMPARE AND
MDU ENABLE
INHIBIT ERROR
REPEAT INST
SINGLE ROUTINE
SINGLE PULSE

The ERROR STOP switch must be off for the halt load CPM during the system initialization routine. It may be turned on later for troubleshooting, if desired.

PANEL 1 SWITCHES

The following switches are located on panel 1 (the right-hand panel) on the front of the CPM:

ON LINE/TEST — When this switch is in the ON-LINE position, the CPM will respond to the HALT, CLEAR and START signals from the control console. When this switch is in the TEST position, the CPM will not respond to console signals. Additionally, if this switch is in the TEST position, a loop timeout error will not cause an interrupt.

The following switches must be reset (off):

SERIAL MODE
SINGLE INST
REPEAT PROGRAM
SINGLE PULSE
4 AM DISABLE

The following switches are used for debugging purposes and will normally be reset (off):

CONDITIONAL HALT
HALT ON SDI
HALT ON PIR

# INPUT OUTPUT MODULE (IOM)

Switches must be correctly set both on panel 1 and on the cold start/halt load selection card, which is located on the backplane of the IOM.

COLD START/HALT LOAD SELECTION CARD

The selection card is a circuit board which is located on the back of the IOM, near the floor, on the right-hand end of the cable connectors.

A/B

This switch allows or disallows a signal from the console for HALT, CLEAR and LOAD, depending on console selection of A or B.

SELECT

This switch selects the load IOM by allowing the load signal from the console; the IOM which has SELECT up will initiate a card or disk load operation. If more than one IOM is included by the partition switches, only one must have SELECT up; all other IOMs must have SELECT down.

MEMORY NO.

These switches indicate the 16K memory segment to be used for initialization (most significant six bits of the memory address). The switches must be down (address 000000).

CPM INTERRUPT NO.

These switches indicate the channel number (binary) of the load CPM; i.e., the CPM which is to be interrupted following the load operation.

CARD READER CHANNEL NO.

The number of the channel (internal to the load IOM) to which the load card reader is connected.

DISK FILE CHANNEL NO.

The number of the channel (internal to the load IOM) to which the desired disk file exchange is connected.

EU UNIT DESIGNATE NO.

The unit designate number of the desired electronics unit. The transfer from the EU will start at SU 0, zone 0, track 0, segment 0 of the specified EU. Note that the HALTLOADEU in the cold start parameter deck (the first card following the ENDUNITS card ) must specify the same disk EU.

CARD LOAD/DISK LOAD

This switch governs a halt/load from the front panel of the IOM in the same way that the corresponding console CARD/DISK switch governs a console H/L. Its setting is ignored unless a H/L from the IOM panel is being done. To accomplish a H/L from the IOM front panel:

1. Place all other IOMs and CPMs in the states that would enable them to be readied into a running system.

2. Place the IOM to be used for the H/L in TEST. Push CONTROL CLEAR, COLDSTART and START. The IOM should then begin the load operation selected on the CARD/DISK switch.

An IOM other than the load IOM uses the SELECT switch (must be OFF), the MEMORY NO., A/B and CPM INTERRUPT NO. switches. The load IOM uses all switches.

The following switches are located on the panel on the front of the IOM:

ON-LINE/TEST

This switch must be in the ON-LINE position for the IOM to respond to console signals.

ERROR STOP

This switch is used for troubleshooting and is usable only if the IOM is in TEST. It should be off when the IOM is ON-LINE.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# RECONFIGURATION

## ON-LINE MODULE DECOMMIT/RECOMMIT

## CPM DECOMMIT/RECOMMIT

The following procedure will enable a running processor to be decommitted from the system:

-- Enter the SAVE CPM message (refer to the ODT manual for the syntax). This will cause the following:

   The fault mask registers of the remaining processors will be set so that interrupts from the "saved" processor are ignored.

   The "saved" processor will enter a loop loaded in the program buffer.

   The requestor inhibit bit will be set for the removed processor on all MCMs in the partition.

   The processor will be deleted from the partition in the MCP INFO table.

-- Place the requestor inhibit switch for the removed processor up on all MCMs to ensure that the processor is not re-admitted to memory following subsequent halt loads.

-- If the removed processor was the halt load processor, designate another halt load processor.

The following procedure will enable a processor to be recommitted:

-- Initialize the processor to be recommitted. This is accomplished as follows:

   1. Place the processor in TEST.

   2. Press control CLEAR.

   3. Press PAUSE.

   4. Place the processor in ON LINE.

      This sequence places the processor in the pause state, ready to execute instructions from word 8 of memory following receipt of an interrupt.

-- Enter the READY CPM message (refer to the ODT manual for the syntax). This will cause the following:

   The processor will be allowed access to all memories in the partition.

   The fault mask registers of all CPMs on the system will allow interrupts from the new processor.

   The processor will be included in the partition in the MCP INFO table.

   The processor will be used by the MCP.

-- Reset the requestor inhibit switch for the new processor on all MCMs in the partition so that the processor will have access to memory following subsequent halt loads.

-- Ensure that the new processor is in the correct partition.

## IOM DECOMMIT/RECOMMIT

The following procedure will cause an IOM to be decommitted from the system:

-- Enter the SAVE IOM message (refer to the ODT manual for the syntax). This will cause the following:

If there are no other IOMs on the system, the request will not be honored.

If there are no ODTs on the remaining IOMs, the request will not be honored.

If any disk or disk pack unit would be made inaccessible by removal of IOM <n>, the request will not be honored.

Devices which can only be accessed through IOM <n> will be saved. This part of the SAVE IOM function may take some indeterminate amount of time. The operator will always know which units are being waited for by the SAVE function and he always has the option of DSing the function or entering <mix no.> RM which will cause all the units which are blocking the SAVE IOM function to be cleared, thereby DSing the jobs using those units. This then allows the IOM to be removed.

DSing the SAVE IOM function implies undoing everything that has been taken care of up to that point.

If the master ODT is on IOM <n>, it will be changed to the lowest numbered ODT left on the system.

Devices which are on shared exchanges will be made accessible only through the other IOMs.

If there is a DCP on IOM <n> that cannot be handled with a cluster exchange, an RSVP message will come out to the operator indicating that he must DS or swap the DCP in order for the SAVE IOM function to continue. See DCP Reconfiguration in this section.

All CPMs will change their interrupt register such that they will no longer accept interrupts from IOM <n>.

The inhibits on all MCMs will be changed to lock out IOM <n>.

Configuration tables on disk will be modified to indicate the absence of IOM <n>. If IOM <n> was the H/L IOM, an RSVP message will appear indicating to the operator that he has lost the H/L IOM and requesting that he change the module switches and key in the new H/L IOM number. The format of this message from the operator is <mix no.> AX IOM <number>.

The saved IOM is left inhibited with its HOMEADDRESS, STATUSQ and IOQUE set to zero.

The following procedure will enable an IOM to be recommitted:

-- Initialize the IOM as follows:

1. Place the IOM in TEST. Ensure that the following switches are set as follows:

   a) All toggle switches at the bottom of panel 2 except the start (SRT) switches - down.

   b) TEST OPERATION - SUB-SYS.

   c) XLTR TEST - OFF.

   d) ERROR STOP - OFF.

   e) All toggle switches at the bottom of panel 1 - down.

2. Press CONTROL CLEAR.

3. Clear the DFI and PCI sections using the general clear toggle switches.

4. Press START.

5. Place the IOM ON-LINE.

-- Enter the READY IOM message (refer to the ODT manual for the syntax).

A home address command will be placed at word zero of memory indicating that IOM <n> should change his home address to the standard address. Subsequently, all of the IOM's registers will be initialized properly.

CPMs and MCMs will change their masks so that they now recognize IOM <n>.

The channel mask for all those devices on IOM <n> which are on a shared exchange will be changed to include IOM <n>.

All devices which are only accessible through IOM <n> will be readied.

IOM <n> will be included in the partition in the MCP INFO table.

NOTE The peripherals that will be used by the new IOM are those which were described in the unit deck at the last cool start, whether or not IOM <n> was included in the partition. Additional peripherals may be added using the READY input command (refer to the ODT manual for the syntax).

Reset the requestor inhibit switches for the IOM on all MCMs in the partition to ensure that the IOM is allowed access to memory following subsequent halt loads.

Check the module switches on the Cold Start/Halt Load selection cards to ensure that this IOM is in the proper partition, that only one IOM in the partition is selected, and if this IOM is to be the halt load IOM, that the other switches are set correctly.

## MCM DECOMMIT/RECOMMIT

An MCM can be decommitted by the SAVE MCM message. (Refer to the ODT manual for syntax). Any MCM can be saved, including the MCM containing address 0, although the address limits of other MCMs may be changed.

The saving of MCMs is done by a visible independent runner called MCMHANDLER. To understand his operation, one should first know that save core is allocated from the low end of memory and overlayable areas are allocated from the high end. Thus, at any time, there exists a "save-fence" above which there is no save core. If MCM <n> (the one to be saved) is above the fence, we overlay all of its areas and the MCM can be removed. Otherwise, we look for another MCM (as big or bigger than n) above the fence, overlay its areas and swap MCM <n> into the one just saved. Thus, MCM <n> is removed. If the fence falls in MCM <n> and there is an MCM above the fence bit enough to hold the save core of MCM <n>, then the MCM above the fence is saved, as is the part of MCM <n> which is above the fence. After this, the remainder of MCM <n> is swapped into the other MCM and MCM <n> is removed (inhibited). If none of these alternatives is possible, MCMHANDLER quits and puts out the message:

"ABORTED -- TOO MUCH NON-OVERLAYABLE CORE"

In the saving process, MCMHANDLER may come across some temporarily save areas which he cannot move or overlay. He will put out the message:

"SAVE CORE OWNED BY MIX # <number>"

Where <number> is the mix number of the stack which is using that area, or

"SAVE CORE OWNED BY MCP"

MCMHANDLER will check several times, waiting for this area to be released or to become overlayable. After four tries, he enters the waiting entries with the message:

"ENTER OK TO KEEP TRYING MEMORY SAVE"

and awaits operator intervention. Three replies are allowed:

1. <mix no.> OK - continue checking for the save areas to be released or become overlayable.

2. <mix no.> DS - give up trying to save an MCM and restore all MCMs.

3. <mix no.> QT - give up trying to save this MCM, but look for another one to save and then swap with the desired MCM.

An MCM SAVE can be cancelled by DSing MCMHANDLER and all core will be restored.

If the SAVE is completed, MCMHANDLER puts out the message:

"SAVED: MCM # <number>"

After the SAVE is completed, set the requestor inhibit and address switches on all MCMs, including the one that was saved, to correspond to the values that were set into the indicator lights by the MCP. It is necessary to check all MCMs because the SAVE may have changed address limits on other MCMs.

An MCM can also be decommitted by use of the SWAP MCM message provided that a substitute MCM, not in use by the system and at least as large as the MCM being decommitted, is available. (Refer to the ODT manual for the syntax).

The following procedure is used to swap MCMs:

-- Place the new MCM on line with all requestors inhibited.

-- Enter the appropriate SWAP MCM message (refer to the ODT manual for syntax). The swap will be performed, the requestor inhibits and limits will be set in the new MCM, and the removed MCM will be left inhibited. The MCP INFO table will be updated to reflect the configuration change.

-- Set the requestor inhibit and address switches on the new MCM to correspond to the values which were set into the indicator lights by the MCP. Place the partition switch (A/B) in the proper position.

-- Set all requestor inhibit switches on the decommitted MCM or place it in local to prevent an address conflict following a subsequent halt load.

The SWAP MCM command allows an MCM to be physically removed from a running system by having its contents transferred to another MCM. The destination MCM must not have been in the partition before the swap and must have sufficient MSUs available to hold the contents of the source MCM. If the destination MCM has more memory than the source, the swap may be used to add memory to the system. The memory added must immediately follow the mods to be swapped. In the following examples, assume the original memory configuration is:

```
MCM 0       Mods   0-15
MCM 1       Mods  16-23
MCM 2       Mods  32-39
MCM 3       Off-line, 4 MSUs available
```

Example 1: SWAP MCM 2 INTO 3 MOD 32-39 MSU 1-2

The contents of MCM 2 will be transferred into the specified stacks of MCM 3. MCM 3 will be added to the partition; MCM 2 will be inhibited to all requestors and deleted from the partition.

Example 2: SWAP MCM 2 INTO 3 MOD 32-47 MSU 1-4

The contents of MCM 2 are transferred into the low 8 mods of MCM 3. The high 8 mods (40-47), not accessible before the swap, are added to the system.

Example 3: SWAP MCM 1 INTO 3 MOD 16-31 MSU 1-4

Similar to example 2, but the added mods are 24-31, filling the "hole" in memory that existed before the swap.

Example 4: SWAP MCM 2 INTO 3 MOD 24-39 MSU 1-4

This is an invalid swap command. Current implementation requires that the starting addresses of the source and destination MCMs agree.

An MCM can be recommitted by use of the READY input message. (Refer to the ODT manual for the syntax).

The following procedure is used to ready an MCM:

-- Place the MCM on-line with all requestors inhibited.

-- Enter the appropriate READY MCM message (refer to the ODT manual for the syntax). The requester inhibits and limits will be set in the MCM and the additional memory will be used by the system. The MCP INFO table will be updated to reflect the additional MCM.

-- If the INFO option (of the READY MCM message) is used, the additional memory will not be used by the system. This is intended for running on-line MCM CONFIDENCE. A halt/load will remove the MCM.

-- Set the requestor inhibit and address switches on the MCM to correspond to the values which were set into the indicator lights by the MCP. Place the partition switch (A/B) in the proper position.

Memory can also be saved or readied using the SAVE MOD or READY MOD input command (refer to the ODT manual for the syntax). It is possible to save or ready only one mod (16K) at a time, thus multiple save or ready messages are necessary to decommit or recommit an MCM. The processing of either a save or ready request must be complete before the next request can be entered.

NOTE Memory configuration changes made in this manner do not result in a change in the list of MCMs considered "in use" by the system. Therefore, such changes are not "remembered" across halt loads and an MCM saved in this manner cannot be used in a later READY or SWAP MCM request.

## SYSTEM HALTED RECONFIGURATION

A mainframe requestor module (CPM or IOM) may be decommitted with the system halted by placing the module in TEST mode and doing a halt load. Note that if the module is either the load CPM or the load IOM, the switch settings must be changed to designate the new load IOM or CPM and that the new load IOM must have access to the system disk EU. An MCM may be decommitted when the system is halted by placing it in LOCAL or by changing the requestor inhibit register to exclude all requestors in the operational system. If the decommitted MCM contained address zero, the address limits on at least one other MCM must be changed so that an MCM which contains address zero is available to the system.

Modules that have been configured out of the system in this manner can be recommitted by doing a halt load with the module available to the system or, alternatively, by using the methods described under ON-LINE RECOMMIT/DECOMMIT if the system is running.

A requestor (CPM or IOM) that is not in the partition can be added at halt load time simply by causing it to be one of the halt load units. This unit will be a member of the system only for that halt load period.

A requestor that has been added in the above manner or which was deleted over a halt load may have its new status made permanent by using the appropriate SAVE or READY message (refer to the ODT manual for the syntax).

Since the MCP will attempt to acquire all modules that are in the partition at each halt load, interference between partitions may result if a module that was decommitted but remained in the partition is to be used as part of another partition or for maintenance testing. This interference can be avoided by ensuring that the partition for an operational system includes only the desired modules.

In order to change the partition with the system halted, a cool start or warm start operation is required. Partition cards should be included for the desired modules and excluded for other modules.

Whenever the system is reconfigured, the switches on all modules should be checked to ensure that each module is in the proper partition, all desired requestors are enabled into the proper MCMs and inhibited from any other MCMs, and that all other requestors are inhibited from the MCMs that are to be used.

## PERIPHERAL RECONFIGURATION

The MCP will attempt to use only those peripheral units and paths described in the Peripheral Configuration Table. The Peripheral Configuration Table is built on disk by the LOADER from the information supplied in the LOADER's parameter deck each time the LOADER is run. The MCP, during system initialization, uses the Peripheral Configuration Table information to build the I/O Map structures that define the available units and paths.

The Peripheral Configuration Table and I/O Map structures can be changed dynamically by ODT input messages, providing the capability to add or delete peripheral units and paths without requiring a halt load or cool start.

## UNIT DELETION

Unit deletion is done with the SAVE input command. (Refer to the ODT manual for the syntax).

The unit or units involved will be deleted from the system. HPT disk units must not be deleted unless they are first reserved. If the unit is a disk pack, it will be deleted only if no files are open on that pack family. Any other unit will be deleted immediately and will be cleared if necessary. (Any job using that unit will be DSed).

## PATH DELETION

Path deletion is done with the SAVE input command. (Refer to the ODT manual for the syntax).

The designated channel will be deleted from use by the system and will be "set reserved" in the IOM. The RESERVED CHANNELS table on disk will be updated so that the reserved status of the channel is preserved across halt loads. If the channel is not on an exchange or is the only channel on the exchange, all of its units will be deleted in the same manner as that described in UNIT DELETION.

If the channel represents the last remaining path to one or more disk file units, the SAVE is disallowed.

If the channel represents the last remaining path to a disk pack exchange, only those packs that can be closed will be deleted. If any packs cannot be closed because of open files, the SAVE operation is terminated.

If the last remaining unit on a channel is deleted using the SAVE command, the channel will also be saved.

## UNIT ADDITION

Unit addition is done with the READY input command. (Refer to the ODT manual for the syntax).

The units will become immediately accessible to the system, using the IOM(s) and channel(s) indicated. Also, HPT disk units that are added in this manner must be RCed into the disk family before they will be used.

The "KIND" specification is necessary only when the proper kind cannot be determined from the unit type, as in the case of magnetic tapes, or when it is desired to override the default unit kind for a particular unit type.

NOTE The IOM and channel "map" specified must correspond to the physical configuration of the system; otherwise, the results are unpredictable.

If the specified map contains no IOMs in the current system partition, a "NO IO PATH" error message is given and the attempt to add the units is ignored.

## PATH ADDITION

Path addition is done with the READY input command. (Refer to the ODT manual for the syntax).

The indicated channels or paths will be added to the system. If the channel is being added to an existing exchange, the EXCHANGE specification (of the READY message) must be used. If no "IOM" specification is made, "all IOMs" is assumed.

# DCP RECONFIGURATION

DCP RECONFIGURAION ON SAVING AN IOM

When an IOM that has one or more DCPs running is saved, either through an operator input message or through the MCP FAILUREANALYSIS routine, an attempt is made to continue the datacomm operation via another IOM. The action taken by the MCP is dependent on the cluster exchange capability of the datacomm subsystem.

The MCP will attempt a transparent transfer of all users of DCP(s) on an IOM which is being saved to other DCP(s) if the following requirements are met:

1. For each DCP to be removed, the DCP section of the NDL description specifies an exchange DCP.

2. The exchange DCP for the DCP being removed is running on an IOM in the same partition which is not being saved.

Provided these requirements are met, the MCP will perform an unsolicited DCRECON. If the reconfiguration request (cluster exchange) is successfully completed, a message will be displayed indicating that the DCP was saved.

If the requirements are not met, one of the following messages will be displayed dependent on the cause of the missing requirement:

1. DCP <number> NOT SAVED (NOT EXCHANGED)

2. DCP <number> NOT SAVED (EXCH DCP NOT RUNNING)

3. DCP <number> NOT SAVED (EXCH DCP ON SAME IOM)

If the automatic cluster exchange cannot be successfully performed, the MCP will cause the DCP(s) to be "idled" and will display the following RSVP message:

IOM BEING SAVED, DS OR SWAP DCP <number>

The state of the DCP(s) may be confirmed by examining the DCP panel which will show both RUN and EXECUTE with what appears to be an "unconditional jump" in the instruction register.

The possible operator action at this point is as follows:

1. The SAVE IOM execution may be terminated by DSing the save.

2. The running DCP with its users may be discarded by DSing the DCP and responding to the save with a <mix no.> OK.

3. The DCP may be reconfigured as follows:

   A. The plugboard in the reconfiguration cabinet may be changed.

   B. The MCP is notified of the new configuration through the following message:

      DC <number> : IOM <number> : <scan address>

      Example: DC4:IOM 1:0

   C. The save is allowed to continue by responding to the save with a <mix no.> OK.

EXPLICIT DCP RECONFIGURATION

A DCP may be "swapped" from one I/O location to another (different scan buss port or different IOM) without "saving" the IOM through the following message:

DC <number> : FORCE <IOM number> : <scan address>

The MCP will initiate an independent runner named DCSWAPMESS which will idle the DCP and display the following RSVP message:

SWAP IO REQUEST FOR DCP/<number>, RECONFIGURE

The possible operator action at this point is as follows:

1. The reconfiguration may be halted by DSing DCSWAPMESS.

2. The DCP may be reconfigured as follows:

   A. The plugboard in the reconfiguration cabinet may be changed.

   B. DCSWAPMESS may be continued by OKing the task.

If the reconfiguration is not successful, one of the following messages may be output:

1. NO IO PATH

   The requested IOM or scan address does not exist. Another "DC<number>:FORCE" message may be entered to specify the current configuration.

2. ALREADY RUNNING

   A DCP was found to be already running in the location specified. Another "DC<number>:FORCE" message may be entered to specify the correct location.

3. DATACOM INACTIVE

   The DCP specified in the "DC<number>:FORCE" message is not running. No exchange is possible.

4. REQUEST DENIED

   The operator erroneously entered the "DC<number>:IOM" type message rather than the "DC<number>:FORCE" type message. The correct "DC<number>:FORCE" type message may be entered.

5. IT CAN'T BE DONE

   The "DC<number> : IOM<number> : <scan-address>" message, specified during a save of an IOM, specified the same IOM as was being saved. A "DC<number>:FORCE" type message may be used to specify the correct information.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

B 7000 SOFTWARE FEATURES

TABLE OF CONTENTS


FAILURE ANALYSIS . . . . . 19-1- 1

DESCRIPTION . . . . . 19-1- 1

FA AND ITS COUNTER MATRIX. . . . . 19-1- 1

CPM CONTROL MODES. . . . . 19-1- 2

MCM FAIL REGISTERS . . . . . 19-1- 2

CPM INSTRUCTION RETRY. . . . . 19-1- 2

LOGGING. . . . . 19-1- 3

FA MESSAGES. . . . . 19-1- 3

FA THRESHOLD ACTIONS. . . . . 19-1- 4

FA CHARGEABLE ERRORS. . . . . 19-1- 5

MAINFRAME LOG ENTRIES . . . . . 19-1- 6

SYSTEM BALANCING . . . . . 19-2- 1

SYSTEM MONITORING . . . . . 19-2- 1

UTILIZATION OF PROCESSING. . . . . 19-2- 1

UTILIZATION OF INPUT/OUTPUT. . . . . 19-2- 2

IMPLEMENTATION . . . . . 19-2- 3

DYNAMIC VARIATION OF SYSTEM BALANCING PARAMETERS. . . . . 19-2- 4

INTERVAL . . . . . 19-2- 4

IOINTERRUPT. . . . . 19-2- 4

IMPLEMENTATION . . . . . 19-2- 5

SYSTEM BALANCING USAGE. . . . . 19-2- 5

# FAILURE ANALYSIS

## DESCRIPTION

The goals of the Failure Analysis (FA) feature of the B7000 system are simplicity in operation and implementation, reliable error and action reporting, and above all, FA actions (such as box removal) which would best preserve system continuity.

## FA AND ITS COUNTER MATRIX

The prime technique utilized by FA to monitor the condition of the various components of the system is a series of counters which are incremented each time a chargeable hardware error occurs. When an installation determined threshold is achieved on a component, action will be taken to attempt to eliminate its disruptive influence. The setting of this action threshold is given to the installation, since it is a determination which is subject to operational considerations.

There are four categories of counters:

| | | |
|---|---|---|
| 1. | REQUESTOR (8) | For charges against CPMs and IOMs. There is one per box. |
| 2. | MCM (8) | For charges against MCMs. There is one per MCM. |
| 3. | INTERFACE (64) | For the interface each requestor has to each MCM. |
| 4. | MF2 (8) | For counting one bit errors on an MCM basis. |

When a counter's threshold is exceeded, actions taken will vary depending upon counter type and system configuration. Table 1-1 (FA THRESHOLD ACTIONS) summarizes these actions.

Incidents which draw the attention of FA and cause counter action are as shown in table 1-2 (FA CHARGEABLE ERRORS).

Simply stated, action against a requestor will involve the offender being removed without a halt/load. Action against an MCM will involve a halt/load. For interface errors (those involving an MCM and requestor), the total of all interface errors for each of the involved modules will be compared, with action being directed at the module with the greater number of interface errors.

In all cases, FA actions will be logged; also, an MCP independent runner named "CASUALTYREPORT" will appear in the mix, displaying the action periodically and requesting acknowledgement. Once acknowledged, CASUALTYREPORT will go to EOJ.

## CPM CONTROL MODES

Should a CPM advance to control mode 2 and the interrupt causing the control mode advance be an acceptable type, the MCP will put the CPM back onto the original stack and perform normal error handling and counter action. Failing this criteria, the CPM will be removed from the system regardless of its FA counter value. Should this be the only CPM in the system, a halt/load will be invoked via a fatal dump titled "ONLY CPM IN CM2".

If a CPM should advance to control mode 3, it will delay momentarily upon entry to determine if it is the only CPM which has done so. This being the case, it will go to a coded "DEAD" stop. System clean-up for this CPM will be handled by the remainder of the system when the "DEAD" CPM fails to check in.

All CPMs entering control mode 3 are indicative of a catastrophic memory failure. The action taken here is to determine the failing memory and, if necessary, reconfigure, then perform a halt/load to get the system going again. Control mode 3 MCM reconfiguration occurs whenever the MCM containing memory address zero is the memory which has had the major failure. If the entire MCM is lost, then the MCM which contains the control mode 3 logic will be reconfigured to fill in for the lost MCM. Should an MSU be lost, the MCM will be configured to exclude this lost storage unit. The action following reconfiguration is a programmatic halt/load. Following this halt/load, CASUALTYREPORT will advise as to what action or actions were taken.

## MCM FAIL REGISTERS

MCM fail registers are handled by FA whenever they are found to contain error information. Generally, this is the result of a memory fail 1 interrupt to an IOM or CPM. To ensure that all MCM failure data is kept current, the registers of all MCMs in the system are polled once every second. When a loaded fail register is found, it will be analyzed and the appropriate counter incremented. Table 1-2 (FA CHARGEABLE ERRORS) indicates the criteria for counter action.

Single bit errors will be logged and counted on an MCM by MCM basis for a limit of 25. Once an MCM achieves this limit, one bit errors from it will thereafter be ignored. One bit logging for a given MCM may be restored by using the ZFA primitive message (refer to the ODT manual).

## CPM INSTRUCTION RETRY

The B7000 CPM is capable of performing instruction retry in most cases of processor internal error. It is the responsibility of FA to invoke this feature. Before doing so, FA will verify that certain hardware and software criteria are satisfied. Failing this, normal error handling is performed.

Should the failure reoccur on the retry, FA will determine again that retry is in order, and if another CPM is available, the next retry will be done by it. Failure to find another CPM or not meeting retry criteria will invoke normal error handling.

The criteria for instruction retry on the B7800 CPM for a PROCESSOR INTERNAL interrupt and a LOOP interrupt are: the CPM is in normal state, it is not in vectormode, and the MES bit in the processor fail register is reset. MES indicates the processor is in a non-recoverable state. For the B7200 CPM instruction retry will occur under the same CPM conditions only for a PROCESSOR INTERNAL interrupt.

## LOGGING

The mainframe section of the system log will receive entries concerning:

1. All CPM alarm interrupts.
2. All IOM errors.
3. All loaded MCM fail registers found (excluding one bits)
4. Limited single bit error entries (as previously described).
5. All FA actions.

These log entries are grouped and presented by the LOGANALYZER as one of four types. These four entry types and their associated information are as shown in table 1-3 (MAINFRAME LOG ENTRIES).

## FA MESSAGES

Messages used to monitor or set FA parameters are as follows:

| | |
|---|---|
| LOG MAINFRAME | Will cause LOGANALYZER to output mainframe errors. |
| FAS | Causes display of the FA counter matrix. |
| ??SFL <value> | Sets action threshold used by FA. |
| ??ZFA <box type> <box no> | Clears the FA counter for the indicated box and its associated interface counters. If <box type> and <box no> are empty, all FA counters are cleared. |

## TABLE 1-1

## FA THRESHOLD ACTIONS

| COUNTER EXCEEDING THRESHOLD | INVOLVED MODULE IS REDUNDANT | INVOLVED MODULE IS NOT REDUNDANT | H-L REQUIRED |
|---|---|---|---|
| REQUESTOR | Module Removed | Module Removal Aborted | No |
| MCM | MCM Removed | MCM Removal Aborted | Yes |
| INTERFACE (Req Err Total Lss MCM Err Total) | MCM Removed | MCM Removal Aborted | Yes |
| INTERFACE (Req Err Total Gtr MCM Err Total) | Requestor Removed | Requestor Removal Aborted | No |
| MF2 (Count Eql 25) | Ignore Subject MCM's One Bit Errs | Ignore Subject MCM's One Bit Errs | No |

# TABLE 1-2

## FA CHARGEABLE ERRORS

| MODULE REPORTING ERROR | ERROR | COUNTER CHARGED WITH ERROR | EXCEPTIONS |
|---|---|---|---|
| CPM | Memory Parity | MCM | None |
| CPM | Processor Internal | REQ | First Retry Successful |
| CPM | Loop | REQ | Interrupted operator is linked list up. |
| CPM | Invalid Address | REQ | Referenced address is not in system. |
| IOM | Initiate Busy Channel | REQ | None |
| IOM | Illegal Home Address | REQ | None |
| IOM | Buffer Register Parity | REQ | None |
| IOM | Residue Error | REQ | None |
| IOM | Address Residue | REQ | None |
| IOM | Store Disparity | REQ | None |
| IOM | No Access | REQ | Referenced address is not in system. |
| IOM | Fetch Disparity | MCM | None |
| MCM | Internal | MCM | None |
| MCM | 2 Bit | MCM | None |
| MCM | Control Word Parity | INTERFACE | None |
| MCM | Illegal Operation | INTERFACE | None |
| MCM | Wrong Address | INTERFACE | None |
| MCM | Data Word Parity | INTERFACE | None |
| MCM | Data Strobe Error | INTERFACE | None |
| MCM | One Bit Error | MF2 | Limited to 25 |

## TABLE 1-3

## MAINFRAME LOG ENTRIES

| REPT. MODULE | TYPE NO. | TYPE TITLE & SUB-HEADINGS | COMMENT |
|---|---|---|---|
| CPM | 1 | Type = CPM X Interrupt | "X" is the interrupted CPM's ID. |
| | | Cause | The reason the log entry was made. |
| | | System Configuration | Mainframe configuration. |
| | | Program Interrupt Point | Where the interrupt occurred. Included are PIR, PSR, SDI, Stack No., Job No.,Task No., LL and NCSFF (State). |
| | | Error Report | The interrupt P1,P2 and processor fail register are given in raw and analyzed form. |
| IOM | 2 | Type - IOM X Error | "X" is the number of the reporting IOM. |
| | | Cause | Tells how the fail RD was found. |
| | | System Configuration | Mainframe configuration. |
| | | Error Report | The fail RD is given in raw and analyzed form. |
| MCM | 3 | Type = MCM X Error | "X" is the reporting MCM. |
| | | Cause | How the fail register was located. |
| | | System Configuration | Mainframe configuration. |
| | | Error Report | The fail register is given in raw and analyzed form. |

| | | | |
|---|---|---|---|
| MCP | 4 | Type = System Error Action | FA is reporting some action it performed. |
| | | Cause | What FA did (or could not do). |
| | | Reason | Why the action was initiated. |
| | | System Configuration | Mainframe configuration following action. |
| MCP | 5 | Type = Mainframe Event | FA is reporting a system event. |
| | | Cause | Event being recorded (dump or halt-load). |
| | | System Configuration | Mainframe configuration. |
| | | Report | Halt-load or dump reason. For dumps, includes calling SNR and mix number and whether fatal or non-fatal. |
| MCP | 6 | Type = System Reconfiguration Action | FA is reporting a voluntary change in mainframe configuration. |
| | | Cause | Box involved and whether SAVEed, READYed or SWAPed (if an MCM). |
| | | System Configuration | Mainframe configuration after action. |
| | | Report | Whether change was operator requested or the aftermath of on-line CPM testing. |

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# SYSTEM BALANCING

SYSTEM BALANCING is a feature of the MCP which gives the user a set of functions by which he can "balance" or "tune" his system to a set of operations. In general, these functions provide the user with the ability to monitor the overall system utilization and the ability to change various software parameters which affect the system utilization.

## SYSTEM MONITORING

Mechanisms have been installed in the MCP to gather statistics which define SYSTEM UTILIZATION in two categories, utilization of processing and utilization of input/output, over a user settable time interval. The description and purpose of the system utilization components follow.

## UTILIZATION OF PROCESSING

The utilization of processing is subdivided into four components, each described in percentage of the last time interval.

1. User time - this component is the time logged directly to user jobs (less the time spent for handling physical I/O). User time includes all user jobs and "visible" MCP independent runners, such as library/maintenance.

2. I/O time - I/O time is the percentage of processing time over the last interval which is spent handling physical I/O functions. This component is critical to system utilization because the amount of I/O activity is directly dependent upon the amount of processing given to the I/O subsystem.. The user can observe this component and vary certain "system balancing parameters" to achieve a desired amount of I/O activity.

3. MCP time - this component is the percentage of time over the last interval spent by the operating system (MCP) to "manage" the system.

4. Idle time - this is simply the percentage of time that the processor(s) was (were) not used for user, I/O or MCP functions above.

## UTILIZATION OF INPUT/OUTPUT

The utilization of I/O is subdivided into several components, each described in rates calculated over the last time interval.

1. User I/O - this component consists of two rates, the average number of user I/O(s) per second and the number of kilobytes of data transferred per second by user I/O(s).

2. MCP I/O - this component is expressed in the same terms as user I/O, but applies to I/O(s) performed by the MCP. MCP I/O(s) include such things as overlays and logging.

3. Total I/O - this component represents the total system I/O activity and is the sum of user I/O(s) and MCP I/O(s).

4. I/O interrupts - this component is simply the average number of I/O interrupts per second calculated over the time interval.

The system utilization can be displayed on the console via invocation of the system input message "UTILIZATION" or simply "U". This option has also been added to the AUTOMATIC DISPLAY MODE or ADM console display. An example of this ADM option is as follows:

```
ADM (UTILIZATION 7, MSG) DELAY 10.
```

In this case, the SYSTEM UTILIZATION followed by MESSAGES would be displayed on the console every ten seconds. An example of the output of the utilization message is as follows:

```
-----SYSTEM UTILIZATION-----
CPM: USER = 68%      MCP =  5%
     I/O  = 10%      IDLE = 17%
I/O: USER = 60  IO/SEC  (220 KB/SEC)
     MCP  = 12  IO/SEC  ( 40 KB/SEC)
     TOTAL= 72  IO/SEC  (260 KB/SEC)
     43 IO-INTERRUPTS/SEC
```

## IMPLEMENTATION

The system utilization statistics are kept within the MCP. Since the nature of the statistics require that they be placed in the main stream of the MCP, a slight overhead will be encountered.

Since these statistics are only meaningful if considered over a time interval, the MCP resets the statistic counters at the end of every interval. This resetting action is done in procedure ONESECONDBURDEN of the MCP, which is called approximately once every second. Therefore, the time interval is at least one second and is an integral multiple of seconds. The system utilization statistics are stored in two arrays. One array contains the counts from the previous time interval. The other array contains the counts from the end of the previous time interval to the present.

A global MCP procedure "SYSTEMSTATISTICS" has been added to retrieve these statistics. This procedure, when called, calculates an interval's worth of values by an interpolation of the previous time interval and the current time interval values. This method is used to allow the statistics to be interrogated at any point during the time interval. Procedure SYSTEMSTATISTICS has one parameter, an array. It is a real procedure which returns a value of zero if no errors were encountered and a one if the input array is too short. The procedure returns the following statistics in the array:

| WORD # | DESCRIPTION |
|---|---|
| 1 | Percentage of idle time |
| 2 | Percentage of I-O time |
| 3 | Percentage of user time |
| 4 | Percentage of MCP time |
| 5 | User I-O(s) per second |
| 6 | User I-O kilobytes per second |
| 7 | MCP I-O(s) per second |
| 8 | MCP I-O kilobytes per second |
| 9 | Total I-O(s) per second |
| 10 | Total I-O kilobytes per second |
| 11 | I-O interrupts per second |

A user program, written in DCALGOL, can obtain these system utilization statistics by calling GETSTATUS, using type 2 (MISCREQUEST) and subtype 3. This subtype simply causes the CONTROLLER to make a call on the MCP procedure SYSTEMSTATISTICS. The system utilization statistics in the format described above are returned in the array passed to GETSTATUS. The following call would obtain the current utilization statistics:

```
B := GETSTATUS (0 & 2 GSTYPEF & 3 GSSUBTYPEF, 0, 0, A);
```

Where: "B" is a boolean, "GSTYPEF" is the field 7:8, "GSSUBTYPEF" is the field 15:8 and "A" is an array. Before the call on GETSTATUS, A[0] must contain the value 2.

# DYNAMIC VARIATION OF SYSTEM BALANCING PARAMETERS

"Dynamic Variation of System Balancing Parameters" is the ability to change, while the MCP is running, certain software parameters which control the flow of the MCP. In particular, the user is given the ability to specify the MCP's I/O interrupt scheme and the ability to change the system utilization time interval. Both of these parameters can be altered or interrogated by the system input message "SBP", (System Balancing Parameters - refer to the ODT manual). SBP may have one of two parameters following: "INTERVAL" (used to change the system utilization time interval) and "IOINTERRUPT" (used to change the MCP I/O interrupt schemes). SBP with no parameter following will display the current values of INTERVAL and IOINTERRUPT. The INTERVAL is expressed in terms of seconds and IOINTERRUPT is a choice of four interrupt strategies. The SBP message may be one of the following:

```
SBP (request display of current balancing parameters)
SBP INTERVAL <seconds> (e.g., SBP INTERVAL 300)
SBP IOINTERRUPT QEMPTY
SBP IOINTERRUPT IOFINISH
SBP IOINTERRUPT IDLE
SBP IOINTERRUPT WAITING
SBP IOINTERRUPT WAITING QEMPTY
```

## INTERVAL

INTERVAL is the system utilization time interval as discussed in System Monitoring. A small interval gives an immediate picture of system utilization and a large interval gives a more overall, averaged picture. With a small interval, radical changes will be accurately reflected. With a large interval, radical changes will be "smoothed out". The default interval is ten seconds, which can be termed a small interval. The user may desire a larger interval for an overall picture but a smaller one is not recommended because the statistics tend to fluctuate too much.

## IOINTERRUPT

IOINTERRUPT is the system balancing feature that allows a choice of four I/O interrupt strategies. The differences among the four involve the way various types of jobs interact and include the amount of CPM time devoted to handling I/O completes. The four interrupt strategies are:

1. QEMPTY - It specifies that the IOM(s) interrupt a CPM when the last I/O request for any unit is completed and on every I/O completion if a CPM is idle.

2. IOFINISH - specifies that IOM(s) interrupt a CPM upon handling every I/O completion. This causes I/O bound jobs to run in a shorter elapsed time, but requires more CPM time to handle I/O completions.

3. IDLE - provides for no I/O interrupts except when a CPM is idle. The effect of this is to bias CPM usage toward CPM bound jobs. This requires the least amount of CPM time for handling I/O completions.

4. WAITING - specifies that I/O interrupts occur upon completion of an I/O operation only if a process is waiting for that I/O or a CPM is idle. As long as the progress of no process is dependent on the completion of a particular I/O operation, the IOM generates no interrupt. The effect of this strategy is to use just enough CPM time to keep I/O bound jobs running.

This WAITING interrupt strategy may be specified along with QEMPTY via the message:

SBP IOINTERRUPT WAITING QEMPTY

IOINTERRUPT WAITING QEMPTY is the default strategy when none is specified.

## IMPLEMENTATION

Procedure SETSTATUS of the MCP will handle the SBP message. Within SETSTATUS, MCP global variables are altered to change the interval or IOINTERRUPT strategy. If the user wishes to change the system balancing parameters from a user program, he can use the DCKEYIN function of DCALGOL or he can invoke SETSTATUS, in which case he should refer to procedure "SYSTEMBALANCINGPARAMETERS" in the controller for specific details of input formats, etc.

## SYSTEM BALANCING USAGE

System balancing can be used in a number of ways. By observing the system utilization and changing the system balancing parameters, the user can "tune" the system to help achieve maximum throughput. He can "bias" the system toward a specific type of application; for instance, IOINTERRUPT on idle will bias the system towards process-bound jobs and IOINTERRUPT on every I/O FINISH will bias the system towards IO-bound jobs. Another use is to record and graph programs throughout the day, enabling the user to distribute the "system load" on an equitable basis. In general, "system balancing" should enhance the user's knowledge and control of his system.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# LIBRARIES

## TABLE OF CONTENTS


1. INTRODUCTION . . . . . 20-1- 1

2. FUNCTIONAL DESCRIPTION OF LIBRARIES . . . . . 20-2- 1

Library Programs . . . . . 20-2- 1

User Programs . . . . . 20-2- 1

Library Directories and Templates . . . . . 20-2- 1

Library Initiation . . . . . 20-2- 1

Linkage Provisions . . . . . 20-2- 2

Error Handling . . . . . 20-2- 3

3. CREATING LIBRARIES . . . . . 20-3- 1

Library Sharing Specifications . . . . . 20-3- 1

4. REFERENCING LIBRARIES . . . . . 20-4- 1

Library Attributes . . . . . 20-4- 1

Type Matching . . . . . 20-4- 2

Parameter Passing . . . . . 20-4- 6

# 1. INTRODUCTION

Libraries provide a new process-structuring technique. Specifically, a library is a program which provides a set of procedural "entry points" which can be called by other programs known as "user programs". A call on a procedure in a library is equivalent to a call on a procedure in the user program.

Collecting functions that are logically related into a library makes programming easier and program structure more visible.

Any programmer can create and use libraries, and more than one program can share a given library.

Libraries offer the following improvements over binding:

1. Interlanguage communication is significantly improved.
2. Sharing of globals is prevented.
3. Standard packages of routines (such as plotting routines and statistical routines) need not be copied into any user programs.

Libraries offer the following improvements over Installation Intrinsics:

1. A CI by the system operator is not required.
2. More than one version of a library can be in use at a time.
3. Libraries can be written in more languages than the Installation Intrinsics.
4. Individual users can create their own libraries without requiring special privileges.
5. A library can have its own global files, databases, transaction bases, and so on.
6. Libraries can contain initialization and termination code.
7. Libraries may themselves become user programs and call other libraries.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# 2. FUNCTIONAL DESCRIPTION OF LIBRARIES

## Library Programs

A library program specifies entry points to be used by user programs and is distinguished by the appearance of <export declaration>s and <freeze statement>s. (COBOL and COBOL 74 differ from other languages in this respect; see Appendix H in the B 7000/B 6000 Series COBOL Reference Manual and Section 12 of the B 5000/B 6000/B 7000 Series COBOL ANSI-74 Reference Manual.) A library program becomes a library after execution of a <freeze statement>.

## User Programs

A user program uses entry points provided by libraries and is distinguished by the appearance of <library declaration>s and <procedure declaration>s that use the <library entry point declaration>s specified in each of the declared libraries. (COBOL and COBOL 74 differ from other languages in this respect; see Appendix H of the B 7000/B 6000 Series COBOL Reference Manual and Section 12 of the B 5000/B 6000/B 7000 Series COBOL ANSI-74 Reference Manual.)

A library may itself function as a user program and call other libraries. The only restriction is that a chain of library linkages must never be circular. That is, a library cannot reference itself, either directly or indirectly; through a chain of library references.

## Library Directories and Templates

The information used by the MCP to match entry points in the library and the user program is contained in a pair of data structures called the "directory" and the "template" which are built by the compilers. The code file for a library program contains exactly one directory, which describes all the entry points, the entry-point parameters and result types, and ways to link to the entry points. The code file for a user program contains one template for each library declared in the user program. The template describes the library attributes and contains the entry point descriptions for the library procedures being used.

## Library Initiation

When a library entry point is first called, the user program is suspended, and the the MCP automatically initiates the library program (if it has not already been initiated). The library program runs normally until it executes a <freeze statement>; this statement makes the exported entry points available. All of the entry points that the user program declared for the library are then linked to the entry points provided by the library, and the user program resumes execution.

The PERMANENT and TEMPORARY specifications for the <freeze statement> control the duration of the library. A permanent library remains available until it is DSed. A temporary library remains available as long as users of the library remain. A temporary library that is no longer in use unfreezes and resumes running as a regular program until "end-of-task". When a library unfreezes, it

may not execute a <freeze statement> in order to become a library again.

Any running program that executes a <freeze statement> becomes a library. When a library is initiated in this way, the <freeze statement> should specify PERMANENT. (If TEMPORARY is specified, the library immediately unfreezes because the active user count is zero.)

Because a library program initially runs as a regular program, the flow of execution can be such that the execution of a <freeze statement> is conditional and can occur anywhere in the outer block of the program.

If a user program causes a library program to be initiated and this library program does not execute a <freeze statement> (for example, because it was not actually a library program and, thus, had no <freeze statement>), the attempted linkage to the library entry points cannot be made, and the user program is DSed.

## Linkage Provisions

Entry points declared in a user program may be linked to entry points provided by a library in one of three ways: directly, indirectly, or dynamically. The library program specifies the form of linkage. Indirect and dynamic linkages allow linkage to be established to libraries other than the library specified by the user program. The user program can only control the library invocation to which it is linked by specifying the code file title of the library (using the TITLE library attribute or in ALGOL, using the LIBACCESS attribute) or, for dynamic linkage, by specifying the LIBPARAMETER library attribute. (Refer to Library Attributes in Section 3.)

Direct linkage occurs when the library contains the procedure that is named in the <export declaration> of the library.

Indirect linkage occurs when the library exports a procedure that is declared as an entry point of another library. The MCP then attempts to link the user program to this second library, which can provide the entry point directly, indirectly, or dynamically.

Dynamic linkage allows a library to specify another library to which linkage is made. For an entry point that a library provides dynamically, the library also specifies a "selection" procedure, which the MCP invokes when the initial linkage between the user program and the library program is established for that entry point. An EBCDIC string whose value is the LIBPARAMETER attribute value specified in the user program for this library is passed to the selection procedure. The selection procedure returns information to the MCP indicating which task (which must be a subtask of the library) provides the entry point. This library task can provide the entry point directly, indirectly, or dynamically. Only libraries written in ALGOL can provide entry points dynamically.

The only restrictions on the complexity of indirect and dynamic linkages are: (1) eventually, some library must provide the entry point directly, and (2) the chain of referenced libraries must never become circular.

## Error Handling

Any fault caused (and ignored) by a procedure in a library which is invoked by a user program is treated as a fault in the user program. If ignored by the user program, this fault causes the user program to be DSed but has no effect on the status of the library.

If a library program faults (or is otherwise DSed) before executing a <freeze statement>, then all user programs that are waiting to link to that library program are also DSed.

If a library is DSed while user programs are linked to it, those user programs are also DSed.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# 3. CREATING LIBRARIES

Programmers can create libraries within the limitations that the language used imposes on the library facility. The programmer of a library need only define a relationship in which one program functions as a library program by declaring procedures to be exported as entry points and another program functions as a user program by invoking these entry points. A library program differs from other programs only because it declares these procedural entry points and uses the <freeze statement> to make itself available as a library. (COBOL and COBOL 74 differ from the other languages in that all programs written in these languages can be invoked either as normal programs or as library programs without requiring the use of a <freeze statement>.)

## Library Sharing Specifications

The programmer specifies the allowable simultaneous uses of a library at compile time using the dollar option SHARING. This option takes one of the forms SHARING=PRIVATE, SHARING=SHAREDBYALL, or SHARING=DONTCARE. For SHARING=PRIVATE, a separate library stack is initiated for each user program that calls the library, so that any changes made by each user program to global items in the library apply only to that particular user program. For SHARING=SHAREDBYALL, one library stack is initiated for all user programs that call the library, so that any changes made by any user program to global items in the library apply to all the user programs that call that library. For SHARING=DONTCARE, the MCP determines the type of sharing to be used, and this determination is unknown to the programmers of the library and user programs. User program calls on a library can be restricted both through different library sharing specifications and through the normal file access restrictions of the system.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES

# 4. REFERENCING LIBRARIES

To use a library, the user program declares a library, giving the attributes of the library and the entry points in the library used by the user program. When an entry point is invoked, the MCP automatically creates the library linkage. If the library program has not already been initiated, the MCP initiates it; then, when the library is frozen, the MCP links the library to the user program. The MCP attempts to make linkages to all entry points in a library at the time that an entry point in the library is first invoked. The CANCEL statement is used to terminate the linkage between the user program and the library. Only PRIVATE libraries or those with a sharing class of SHAREDBYRUNUNIT (the default for COBOL) can be CANCELed. An attempt to CANCEL another type of library results in a warning message and the CANCEL is not done.

## Library Attributes

Libraries, like files, have attributes that can be set and tested programmatically. Libraries have the following attributes:

| | |
|---|---|
| FUNCTIONNAME | FUNCTIONNAME is an EBCDIC string that specifies the system function name used to find the target code file for a library. (See LIBACCESS below.) This attribute applies only to ALGOL libraries. |
| INTNAME | INTNAME is an EBCDIC string that specifies the internal identifier for the library. |
| LIBACCESS | LIBACCESS specifies the way in which a library code file is accessed when a library is called. LIBACCESS has one of the values BYTITLE or BYFUNCTION. If the value is BYTITLE (the default), the TITLE attribute of the library is used to locate the code file. If the value is BYFUNCTION, the FUNCTIONNAME attribute of the library is looked up in the MCP library function table, and the code file associated with the FUNCTIONNAME is used. This attribute applies only to ALGOL libraries. |
| LIBPARAMETER | LIBPARAMETER is an EBCDIC string used to transmit information from the user program to the selection procedures of libraries that provide dynamic entry point linkage. |
| TITLE | TITLE is an EBCDIC string that specifies the code file name of the library. |

The user program can change library attributes dynamically; however, these changes must be made before the first attempt to link to the library. The MCP ignores any changes made to library attributes in the user program which are made after the user program is linked to the library.

## Type Matching

Library procedural entry points may be either typed or untyped. Type matching is performed on entry points during the library linkage. If the type specified in the template does not match the type specified in the directory, the linkage is not made, and the referencing program is DSed. Matching is based on several factors: the procedure type, the number of parameters, the parameter types, and the ways in which the procedure and parameters are provided. The procedure and parameters can be provided by value, by reference, or by name.

Type matching may also be performed when the library program and the user program are written in different languages.

An ALGOL library entry point may be any of the following:

Untyped procedure
Boolean procedure
DOUBLE procedure
Real procedure
Integer procedure
Complex procedure
EBCDIC string procedure
ASCII string procedure
Hex string procedure

The parameters to an ALGOL library entry point may be any of the following types:

Boolean variable or array
DOUBLE variable or array
Real variable or array
Integer variable or array
Complex variable or array
EBCDIC string variable or array
ASCII string variable or array
Hex string variable or array
Event variable or array
Task variable or array
EBCDIC character array
ASCII character array
Hex character array
File
Pointer
Queue
Transaction record or record array

A parameter may also be a fully specified (using FORMAL) procedure with the above restrictions on its possible parameters and return type.

002

# 4. REFERENCING LIBRARIES

To use a library, the user program declares a library, giving the attributes of the library and the entry points in the library used by the user program. When an entry point is invoked, the MCP automatically creates the library linkage. If the library program has not already been initiated, the MCP initiates it; then, when the library is frozen, the MCP links the library to the user program. The MCP attempts to make linkages to all entry points in a library at the time that an entry point in the library is first invoked.

## Library Attributes

Libraries, like files, have attributes that can be set and tested programmatically. Libraries have the following attributes:

| | |
|---|---|
| FUNCTIONNAME | FUNCTIONNAME is an EBCDIC string that specifies the system function name used to find the target code file for a library. (See LIBACCESS below.) This attribute applies only to ALGOL libraries. |
| INTNAME | INTNAME is an EBCDIC string that specifies the internal identifier for the library. |
| LIBACCESS | LIBACCESS specifies the way in which a library code file is accessed when a library is called. LIBACCESS has one of the values BYTITLE or BYFUNCTION. If the value is BYTITLE (the default), the TITLE attribute of the library is used to locate the code file. If the value is BYFUNCTION, the FUNCTIONNAME attribute of the library is looked up in the MCP library function table, and the code file associated with the FUNCTIONNAME is used. This attribute applies only to ALGOL libraries. |
| LIBPARAMETER | LIBPARAMETER is an EBCDIC string used to transmit information from the user program to the selection procedures of libraries that provide dynamic entry point linkage. |
| TITLE | TITLE is an EBCDIC string that specifies the code file name of the library. |

The user program can change library attributes dynamically; however, these changes must be made before the first attempt to link to the library. The MCP ignores any changes made to library attributes in the user program which are made after the user program is linked to the library.

## Type Matching

Library procedural entry points may be either typed or untyped. Type matching is performed on entry points during the library linkage. If the type specified in the template does not match the type specified in the directory, the linkage is not made, and the referencing program is DSed. Matching is based on several factors: the procedure type, the number of parameters, the parameter types, and the ways in which the procedure and parameters are provided. The procedure and parameters can be provided by value, by reference, or by name.

Type matching may also be performed when the library program and the user program are written in different languages.

An ALGOL library entry point may be any of the following:

Untyped procedure
Boolean procedure
DOUBLE procedure
Real procedure
Integer procedure
Complex procedure
EBCDIC string procedure
ASCII string procedure
Hex string procedure

The parameters to an ALGOL library entry point may be any of the following types:

Boolean variable or array
DOUBLE variable or array
Real variable or array
Integer variable or array
Complex variable or array
EBCDIC string variable or array
ASCII string variable or array
Hex string variable or array
Event variable or array
Task variable or array
EBCDIC character array
ASCII character array
Hex character array
File
Pointer
Queue
Transaction record or record array

A parameter may also be a fully specified (using FORMAL) procedure with the above restrictions on its possible parameters and return type.

The following attributes are used to return types for COBOL library entries and are listed with their corresponding ALGOL equivalents. (Refer to Appendix H of the B 7000/B 6000 Series COBOL Reference Manual.)

| COBOL Parameter | Corresponding ALGOL Parameter |
|---|---|
| COMP, 77 | Integer variable |
| COMP, 01 | Real array [0] |
| COMP, 01 RECORD AREA | Direct real array [0] |
| COMP-4, 77 | Real variable |
| COMP-5, 77 | DOUBLE variable |
| COMP-1, 77 single | Integer variable |
| COMP-1, 77 double | DOUBLE variable |
| PIC X, 77 | EBCDIC string variable |
| COMP-2, 01 no OCCURS | Hex character array [0] |
| COMP-2, 01 RECORD AREA | Direct hex character array [0] |
| ASCII, 01 no OCCURS | ASCII character array [0] |
| DISPLAY, 01 no OCCURS | EBCDIC character array [0] |
| DISPLAY, 01 RECORD AREA | Direct EBCDIC character array |
| EVENT, 77 | Event variable |
| EVENT, 01 | Event array [0] |
| FILE | File |
| FILE, ASSIGN TO DIRECT | Direct file |
| Transaction record | Transaction record |
| Transaction record array | Transaction record array [0] |

**NOTE**

The COBOL construct equivalent to the ALGOL construct that uses an "*" as an array parameter can be constructed using the WITH LOWER BOUNDS clause.

The following attributes are given to return types for COBOL 74 library entries and are listed with their corresponding ALGOL equivalents. (Refer to Section 12 of the B 5000/B 6000/B 7000 Series COBOL ANSI-74 Reference Manual.)

| COBOL 74 Parameter | Corresponding ALGOL Parameter |
|---|---|
| COMP, 01 and 77 and INDEX USAGE | Hex character array [0] |
| ASCII, 01 and 77 | ASCII character array [0] |
| DISPLAY, 01 | EBCDIC character array [0] |
| Transaction record | Transaction record |
| Transaction record array | Transaction record array [0] |
| Transaction procedure | Transaction procedure |
| String (DISPLAY) | String (This string is not RESIZEable). |
| Integer (COMP) | Integer |

The following attributes are given to return types for FORTRAN library entries and are listed with their corresponding ALGOL equivalents. (Refer to Section 21 of the B 7000/B 6000 Series FORTRAN Reference Manual.)

| FORTRAN Parameter | Corresponding ALGOL Parameter |
|---|---|
| Integer | Integer variable |
| Real | Real variable |
| Logical | Boolean variable |
| Double-precision | DOUBLE variable |
| Complex | Complex variable |
| Integer array | Integer array [*] |
| Real array | Real array [*] |
| Logical array | Boolean array [*] |
| Double-precision array | DOUBLE array [*] |
| Complex array | Complex array [*] |

The following attributes are given to return types for PL/I library entries and are listed with their corresponding ALGOL equivalents.

| PL/I Parameter | Corresponding ALGOL Parameter |
|---|---|
| Binary Fixed(p, q), | |
| ...q = 0, p <= 39 | Integer variable |
| ...q = 0, 39 < p <= 78 | Unique to PL/I |
| ...q neq 0, p <= 39 | Unique to PL/I |
| ...q neq 0, 39 < p <= 78 | Unique to PL/I |
| Decimal Fixed(p, q), | |
| ...q = 0, p <= 11 | Integer variable |
| ...q = 0, 11 < p <= 23 | Unique to PL/I |
| ...q neq 0, p <= 11 | Unique to PL/I |
| ...q neq 0, 11 < p <= 23 | Unique to PL/I |
| Binary Float(p), | |
| ...p <= 39 | Real variable |
| ...39 < p <= 78 | DOUBLE variable |
| Decimal Float(p) | |
| ...p <= 11 | Real variable |
| ...11 < p <= 23 | DOUBLE variable |
| Character(n) | EBCDIC string variable |
| Character(*) | EBCDIC string variable |
| Character(n) Varying | EBCDIC string variable |
| Character(*) Varying | EBCDIC string variable |
| Bit(n), n <= 48 | Boolean variable |
| Transaction record | Transaction record |
| Transaction record array | Transaction record array [*] |

## Parameter Passing

If a library program declares a parameter to be provided by name, the user program can pass the parameter by name, by reference, or by value. If the library program declares a parameter to be provided by reference, the user program can pass the parameter by reference or by value. If the library program declares a parameter to be provided by value, the user program can only pass the parameter by value.

In ALGOL, parameters are usually passed or specified by name or by value; files, events, tasks, strings, arrays, and formal procedures are passed by reference. In COBOL, FORTRAN, and PL/I, parameters are passed by reference unless specified otherwise.

A formal parameter array in ALGOL that is declared using the "[*]" lower bound construct is passed as two parameters: (1) the array and (2) the lower bound of the array as an integer parameter. In COBOL (when an array parameter is declared WITH LOWER BOUNDS) and FORTRAN, array parameters are also passed in this fashion. In ALGOL, an identical relationship exists between passing an array with "[*]" lower bounds and passing an array row with constant lower bounds (for example, [0]) and an integer by value.

For multidimensional ALGOL and COBOL arrays, if any lower bound is an asterisk, then all the lower bounds are passed, one integer for each bound. All FORTRAN arrays (regardless of the number of dimensions) are passed as a one-dimensional ALGOL array with a lower bound.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

# 1. INTRODUCTION

This chapter serves as a reference to users of the system summary log "SUMLOG" information on Burroughs B 5000/B 6000/B 7000 series large-scale computer systems. This log contains records concerning jobs previously run, MCP activity, and other associated information regarding the past status of the machine environment.

The constructs of the records and words associated with this chapter are from the 3.2 release of the MCP, JOBFORMATTER, and LOGANALYZER code. Therefore, because of the volatility of SUMLOG (the ability for easily modifying and adding structures), care should be taken when attempting to isolate and correlate data from this chapter and various log outputs produced from other releases.

The information in this chapter applies equally to the B 5000, B 6000, and B 7000 series systems unless specifically noted otherwise for that data item.

NOTE

The SUMLOG chapter has not been updated for the 3.3 release. An additional PCN will be issued later this year.