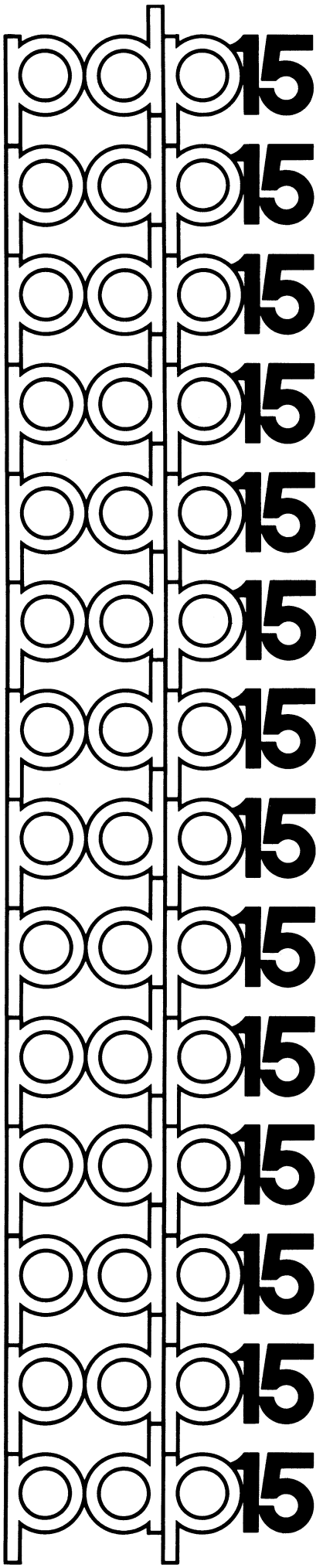
digital

background foreground monitor

programmers reference manual



digital equipment corporation

November 73

DEC-15-MR3A-D

updated with DEC-15-MR3A-DN2 5/2/74 CDC

PDP-15/30 AND PDP-15/40
BACKGROUND/FOREGROUND MONITOR

SOFTWARE SYSTEM

PROGRAMMERS REFERENCE MANUAL

To obtain additional copies of this manual, order number DEC-15-MR3A-D from the Program Library, Digital Equipment Corporation, Maynard, Massachusetts, 01754. Price $6.50

Printed December, 1970
Second Printing, April, 1972

The material in this manual is intended for information purposes and is subject to change without notice.

The following are trademarks of Digital Equipment Corporation, Maynard, Massachusetts

DEC	PDP
FLIP CHIP	FOCAL
DIGITAL	COMPUTER LAB

CONTENTS

Page

SECTION 1 BACKGROUND/FOREGROUND MONITOR

1.1 Introduction 1-1
1.2 Background/Foreground Monitor Functions 1-1
1.2.1 Scheduling of Processing Time 1-2
1.2.2 Protection of Foreground Core and I/O 1-3
1.2.3 Sharing of Multi-User Device Handlers 1-4
1.2.4 Use of Software Priority Levels 1-4
1.2.5 Use of Real-Time Clock 1-4
1.2.6 Communication Between Background and Foreground Jobs 1-4
1.2.7 Use of CPU Registers 1-5
1.3 Hardware Requirements and Options 1-5

SECTION 2 BFKM15 - NON-RESIDENT BACKGROUND/FOREGROUND MONITOR

2.1 Introduction 2-1
2.2 Location and When Called 2-1
2.3 Initial Operation 2-2
2.4 Information Commands 2-3
2.4.1 The LOG Command (L) 2-4
2.4.2 The REQUEST Command (R) 2-4
2.4.3 The DIRECT Command (D) 2-5
2.4.4 The INUSE Command (I) 2-5
2.5 Allocation Commands 2-5
2.5.1 The ASSIGN Command (A) 2-6
2.5.2 The FILES Command (F) 2-7
2.5.3 The FCORE Command 2-8
2.5.4 The FCONTROL Command 2-9
2.5.5 The BCONTROL Command 2-9
2.5.6 The NEWDIR Command (N) 2-10
2.5.7 The SHARE Command (S) 2-11
2.5.8 The NOSHARE Command 2-11
2.5.9 The 7CHAN Command (7) 2-11
2.5.10 The 9CHAN Command (9) 2-12
2.5.11 The MPOFF Command 2-12
2.5.12 The MPON Command (M) 2-12
2.6 Program Load Commands 2-13
2.7 Final Operation 2-13
2.8 Control Characters 2-13
2.9 Summary of Commands 2-14

SECTION 3 CONTROL CHARACTERS

3.1 Purpose 3-1
3.2 Control Teletype 3-1
3.3 Teletype Handler 3-2
3.4 CTRL C (↑C) 3-2
3.5 CTRL S (↑S) 3-3
3.6 CTRL T (↑T) 3-3

		Page
3.7	CTRL P (↑P)	3-3
3.7.1	NORMAL CTRL P	3-4
3.7.2	No Change	3-5
3.7.3	REAL-TIME CTRL P	3-5
3.8	CTRL R (↑R)	3-5
3.9	CTRL Q (↑Q)	3-5
3.10	CTRL U (@)	3-6
3.11	RUBOUT (\)	3-7
3.12	CTRL D (↑D)	3-7

SECTION 4 LOADERS

4.1	Introduction	4-1
4.2	Foreground Linking Loader	4-1
4.2.1	Option Characters and Their Meanings	4-2
4.2.2	Use of ← Terminator	4-2
4.2.3	Sequence of Operation	4-2
4.3	Background System Loader	4-3
4.4	Background Linking Loader	4-5
4.5	Loading XCT Files	4-6
4.5.1	EXECUTE in the Foreground	4-6
4.5.2	EXECUTE in the Background	4-7
4.6	Error Conditions	4-8
4.7	System Memory Maps	4-9

SECTION 5 EXAMPLES OF BACKGROUND/FOREGROUND OPERATIONS

5.1	Introduction	5-1
5.2	Startup Procedures	5-1
5.2.1	Loading Master B/F Monitor System	5-1
5.2.2	System Generation	5-3
5.3	Examples	5-3
5.3.1	IDLE Loaded as the Foreground Job	5-3
5.3.2	Single-user FOCAL Loaded (Foreground)	5-4
5.3.3	Two-user FOCAL Loaded (Foreground)	5-4

SECTION 6 BACKGROUND FOREGROUND MONITOR COMMANDS (SYSTEM MACROS)

6.1	Introduction	6-1
6.2	.REALR	6-1
6.3	.REALW	6-2
6.4	.IDLE	6-3
6.5	.IDLEC	6-4
6.6	.TIMER	6-4
6.7	.RLXIT	6-5
6.8	Mainstream Real-Time Subroutines	6-5
6.9	API Software Levels -- Programming Note	6-5

Page

SECTION 7 WRITING DEVICE HANDLERS FOR THE PDP-15 BACKGROUND/FOREGROUND MONITOR SYSTEM

7.1 Introduction 7-1
7.2 I/O Service Routine 7-1
7.3 I/O Device Handler 7-5
7.3.1 Types of Device Handlers 7-6
7.3.2 General Structure of Device Handlers 7-7
7.4 Reentrancy Protection 7-7
7.5 Device Handler's CAL Processor 7-8
7.5.1 Arguments of the CAL 7-8
7.5.2 .SETUP 7-14
7.5.3 Initiating I/O 7-15
7.6 Device Handler's Interrupt Processor 7-15
7.7 Error Processing 7-20
7.8 Stop I/O Routines 7-21
7.9 Recovery From I/O Device Not Ready Condition 7-23
7.9.1 CTRL R Mechanism 7-23
7.9.2 .INIT Consideration 7-21
7.10 The .INIT Function 7-27
7.11 Sequential Multi-user Device Handler 7-28
7.11.1 Transition from Single-user Handler 7-28
7.11.2 Peculiarities 7-29
7.11.3 Use of the .WAITR Function 7-29
7.12 External I/O Buffers 7-29
7.12.1 Calling for a Buffer 7-30
7.12.2 Releasing a Buffer 7-31
7.13 PDP-9/PDP-15 Compatibility 7-31
7.13.1 Page Mode 7-31
7.13.2 Bank Mode 7-31
7.14 Device Handler Listing 7-31

SECTION 8 SYSTEM GENERATION

8.1 Introduction 8-1
8.1.1 BFSGEN, Generation and Update Features 8-1
8.2 BFSGEN Device Requirements 8-1
8.2.1 DECtape Masters 8-1
8.2.2 Loading BFSGEN 8-2
8.3 System Generation Procedures 8-3
8.3.1 Section A -- Initialization 8-3
8.3.2 Section B -- System Selection & Read-in 8-4
8.3.3 Section C -- System Parameters 8-5
8.3.4 Section D -- Existing I/O Devices 8-6
8.3.4.1 Expendable (Deletable) Devices 8-9
8.3.5 Section E -- Additional I/O Devices 8-11
8.3.6 Section F -- PI Skip Chain 8-13
8.3.7 Section G -- .IOTAB 8-13
8.3.8 Section H -- .DAT Slots 8-15
8.3.9 Section I -- Re-write System Information 8-16
8.4 Post-Generation Procedures 8-16
8.5 Error Detection 8-19

APPENDICES

I .SCOM Registers I-1
II Errors II-1
III Teletype Hardware Characteristics III-1
IV Monitor System Tables IV-1

SECTION 1

BACKGROUND/FOREGROUND MONITOR

1.1 INTRODUCTION

In the preparation of this manual, it was assumed that the reader is familiar with the PDP-15 ADVANCED Monitor Software System as described in DEC-15-MR2A-D. A complete description of the Background/Foreground Monitor System is given in this manual; however, where redundancy occurs, the reader has been referenced to the ADVANCED Monitor manual.

1.2 BACKGROUND/FOREGROUND MONITOR FUNCTIONS

The Background/Foreground Monitor is designed to control processing and I/O operations in a real-time or time-shared environment. It is, essentially, an extension of the ADVANCED Monitor and allows for time-shared use of a PDP-15 by a protected, priority, user FOREGROUND program[1] and an unprotected system or user BACKGROUND program.

The Background/Foreground Monitor greatly expands the capabilities of PDP-15/20 ADVANCED Software and makes optimum use of all available hardware. It permits recovery of the free time (or dead time) that occurs between input/output operations, thus promoting 100% utilization of central processor time.

FOREGROUND programs are defined as the higher-priority, debugged user programs[1] that interface with the real-time environment. They normally operate under Program Interrupt (PI) or Automatic Priority Interrupt (API) control, and are memory protected. At load time they have top priority in selection of core memory and I/O devices, and at execution time they have priority (according to the assigned priority levels) over processing time. Depending upon system requirements, the user's Foreground program could be an Executive capable of handling many real-time programs or subprograms at four levels of priority.

BACKGROUND processing is essentially the same as the processing normally accomplished under control of the ADVANCED Monitor. That is, it could be an assembly, compilation, debugging run, production run, editing task, etc. Background programs may use any facilities (for example, core, I/O and processing time) that are available and not simultaneously required by the Foreground job. Under certain circumstances, I/O devices may be shared by both the Foreground and the Background jobs.

---

[1] It may be feasible in the future to provide system programs which will operate in the FOREGROUND.

The Background/Foreground Monitor system is externally a keyboard-oriented system; that is, Foreground and Background requests for systems information, core, I/O devices, programs to be run, etc., are made via the Teletype[1] keyboards. At run time, the Monitor internally controls scheduling and processing of I/O requests, while protecting the two resident users.

The Background/Foreground Monitor performs the following functions as it controls the time-shared use of the PDP-15 central processor by two co-resident programs:

a. Schedules processing time.
b. Protects the Foreground job's core and I/O devices.
c. Provides for the sharing of multi-user device handlers, such as DECtape, by both Foreground and Background jobs.
d. Allows convenient use of API software levels by Foreground jobs.
e. Provides for convenient and shared use of the system Real Time Clock.
f. Allows communication between the Background and Foreground jobs via core-to-core transfers or by the shared use of bulk storage devices.
g. Allows concurrent use of the CPU's active registers, such as the AC and Index Register.

1.2.1 Scheduling of Processing Time

At run time, the Foreground job retains control except when it is I/O bound; that is, when completion of an I/O request must occur before it can proceed any further. In the following example, if the .WAIT is reached before the input requested by the .READ has been completed, control is transferred to a lower priority Foreground segment or to the Background job until the input for the Foreground job is completed.

```
.READ 3, Ø, LNBUF, 48          /READ TO .DAT SLOT 3
    .
    .
    .
.WAIT 3                        /WAIT ON .DAT SLOT 3
```

Since multi-user device handlers can be shared by Foreground and Background programs, there is a mechanism by which a Foreground I/O request may cause a Background I/O operation to be stopped immediately so that the Foreground operation can be honored. On completion of the Foreground I/O, the Background I/O is resumed with no adverse effects on the Background job.

The Foreground program can also indicate that it is I/O bound by means of the

[1]Teletype is a registered trademark of the Teletype Corporation.

.IDLE or .IDLEC command (Sections 6.4 and 6.5). This is useful when the Foreground job is waiting for real-time input from any one of a number of input devices. Consider the following example (see Section 6.2 for description of real-time read .REALR command).

```
.REALR 1, Ø,LNBUF1, 32, CTRL1, N1        /REAL
.REALR 2, 2,LNBUF2, 42, CTRL2, N2        /TIME
.REALR 3, 3,LNBUF3, 36, CTRL3, N3        /READS
    .
    .
    .
.IDLE
```

If .IDLE is reached before any of the input requests have been satisfied, control is transferred to a lower priority Foreground segment or to the Background job. The lower priority job retains control until one of the Foreground input requests is satisfied. Control is then returned to the Foreground job by executing the subroutine at the specified completion address (CTRL1, CTRL2, CTRL3) and at the priority level specified by N1, N2, N3 which may be:

Value of N		Level
Ø	=	Mainstream (lowest level)
4	=	Current level (level of .REALR)
5	=	Software level 5
6	=	Software level 6
7	=	Software level 7

NOTE

If real-time reads (.REALR), real-time writes (.REALW), or interval timer (.TIMER) requests are employed in the Background, N may be set to Ø, 4, 5, 6, or 7, but is converted to Ø since the Background job can run only on the mainstream level. This allows the value of N to be preset in cases where a Background program is to be subsequently run in the Foreground.

1.2.2 Protection of Foreground Core and I/O

The Foreground job's core is protected by the Memory Protection Option (Type KM15). The Background job runs with memory protect enabled; the Foreground job runs with memory protect disabled.

Protection of the Foreground job's I/O devices is accomplished via the hardware by the Memory Protect Option (which prohibits IOT and Halt instructions in the Background area) and by the software since the Monitor screens all I/O requests made via I/O macros. Also, the Monitor and the Background Loaders prevent the Background job from requesting I/O which would conflict with that of the Foreground job (for example, they would not honor a Background request for a paper

tape handler being used by the Foreground job).

1.2.3 Sharing of Multi-User Device Handlers

The Background/Foreground Monitor permits sharing of multi-user device handlers (such as DECtape, Magnetic Tape and Disk) between Background and Foreground jobs. Using these multi-user handlers, n files can be open simultaneously, where n equals the number of .DAT slots associated with the particular bulk storage device. Some multi-unit handlers require external data buffers (assigned at load time), one for each open file. These buffers are acquired from and released to a pool by the handler as needed.

When this count is not accurate (when the .DAT slots are not used simultaneously), the keyboard command FILES (Section 2.5.2) can be used to specify the actual number of files simultaneously open. Both the Foreground and Background jobs can indicate their file requirements by means of the FILES keyboard command.

The multi-user handlers are capable of stacking one Background I/O request. This provision is made to exactly simulate program operation as it would occur under ADVANCED or I/O Monitor (i.e., single user) control. Thus, control is returned to the Background job to allow non-I/O related processing when the handler is preoccupied with an I/O request from the Foreground job. For example, if the Foreground job has requested DECtape I/O with a .READ, and is waiting for its completion on a .WAIT, control is returned to the Background job. If the Background job then requests DECtape I/O with a .READ, the handler will stack the request and return control to the Background job following the .READ. The Background job can then continue with non-I/O related processing as though the .READ were being honored.

1.2.4 Use of Software Priority Levels

The Background/Foreground Monitor allows convenient use of software priority levels of the API by the Foreground job. The Background job is permitted to use only the mainstream level.

1.2.5 Use of Real-Time Clock

The Background/Foreground Monitor provides for convenient and shared use of the system real-time clock. It will effectively handle many intervals at the same time; thus, the real-time clock can be used simultaneously by both Background and Foreground jobs.

1.2.6 Communication Between Background and Foreground Jobs

The Background/Foreground Monitor allows communication between Background and

Foreground jobs via core-to-core transfers. This is accomplished by means of a special "Core I/O device" handler within IOPS. Complementing I/O requests are required for a core-to-core transfer to be effected; for example, a Foreground .READ (.REALR) from core must be matched with a Background .WRITE (.REALW) to core.

Two possible uses of this feature are:

a. The Background job could be related to the Foreground job and, as a result of its processing, pass on information that would affect Foreground processing, or vice-versa.

b. The Background job could be a future Foreground job and the current Foreground job, being its predecessor, could pass on real-time data to create a true test environment.

Communication between two jobs can also be done by storing and retrieving data on shared bulk storage devices.

1.2.7 Use of CPU Registers

Whenever control passes from one API software level to another, or to Foreground mainstream or to Background, the following CPU registers are saved and restored.

XR	Index Register [1]
LR	Limit Register [1]
MQ	Multiplier-Quotient Register
AIX	The Autoincrement Registers — ?
L	The Link
PC	The Program Counter (including bits to indicate the state of memory protect and page/bank mode)

The Step Counter and the Accumulator are saved and restored only for the Background job. The Foreground job, because it runs with memory protect disabled, can save the contents of the Step Counter in the two free (non-interruptible) instructions following a Normalize instruction by saving the AC (DAC) and then loading the AC with the SC (LACS). The AC is not saved for any level of the Foreground job because a level can give up control only by issuing a Monitor call (CAL) (either .IDLE, .WAIT, or an implied .WAIT). The contents of the AC are not saved and restored by the CAL handler. In addition to these hardware registers, .SCOM+1,+2,+3,+4, and +1Ø are swapped whenever control changes from Foreground to Background or vice versa.

1.3 HARDWARE REQUIREMENTS AND OPTIONS

The following PDP-15 System hardware configurations are required to run the

[1] In the bank mode system, the XR and LR registers are not saved and restored; all other registers are handled as stated.

Background/Foreground Monitor Software System.

PDP-15/30 DECtape System	PDP-15/40 DECdisk System
PDP-15 CPU with a minimum of 16K core memory	PDP-15 CPU with a minimum of 24K of core
KE15 (EAE)	KE15 (EAE)
KSR35 Console Teletype[1]	KSR35 Console Teletype[1]
PC15 (High Speed Reader/Punch)	PC15 (High Speed Reader/Punch)
KA15 (API)	KA15 (API)
KW15 (Real Time Clock)	KW15 (Real Time Clock)
KM15 (Memory Protect)	KM15 (Memory Protect)
TCØ2D[2] or TC15 (DECtape Control)	TCØ2D[2] or TC15 (DECtape Control)
3 TU55 (DECtape Transports) or 2 TU56 (Dual-DECtape Transports) as a minimum	2 TU55 (DECtape Transports) or 1 TU56 (Dual-DECtape Transport) as a minimum
An LT15 or an LT19[2] Teletype Control Unit with at least one additional KSR33 or KSR35 Teletype	An LT15 or an LT19[2] Teletype Control Unit with at least one additional KSR33 or KSR35 Teletype
	RF15 (DECdisk control)
	2 RSØ4 (Disk platters) minimum; 4 maximum at present.

[1]The basic system Teletype is normally assigned to the Background environment. One Teletype of the external Teletype system must be reserved for the Foreground job; additional Teletypes may be assigned to either Background or Foreground functions.

Model 37 Teletypes are not supported. Models 33 or 35 ASR are supported only to the extent that they operate as KSR's; their paper tape input and output facility cannot be used. Detailed information concerning Teletype units is given in Appendix III.

[2]The TC02D DECtape control and the LT19 Teletype Control require the DW15.

In addition to the 15/30 and 15/40 configurations shown, the following PDP-9 configurations may also be used when running the bank mode system.

PDP-9 DECtape System	PDP-9 DECdisk or RBØ9 Disk System
PDP-9 with a minimum of 16K core memory	PDP-9 with minimum of 24K core memory
Real Time Clock	Real Time Clock
KX09A (Memory Protect)	KX09A (Memory Protect)
KE09 (EAE)	KE09 (EAE)
2KSR 33/35 (Teletypes)	2KSR 33/35 (Teletypes)
PC02 (high speed paper tape reader/punch)	PC02 (high speed paper tape reader/punch)
KF09A (API)	KF09A (API)
TC02 (DECtape control)	TC02 (DECtape control)
3 TU55 (DECtape transports)	2 TU55 (DECtape transports)
or	or
2 TU56 (Dual-DECtape transports)	1 TU56 (Dual-DECtape transport)
LT19A (Teletype control)	LT19A (Teletype control) RF09 (DECdisk control) and RS09 (DECdisk plotter) or RB09 disk and control

The following options currently supported by software may be added to improve system performance (as noted):

Options	Effect
Additional 8192-word Core Memory Modules, Type MM15-A plus MK15A[1] (to a maximum of 32,768 words)	Increases the maximum size of both Background and Foreground programs that can be handled by the system.
Additional DECtape Transports, Type TU56, or IBM-compatible Magnetic Tape Transports, Type TU20A or TU20B and Tape Control Type TC59D	Allows greater bulk storage capability, simultaneous use of storage media by more programs. Since only one file may be open at a time on IBM-compatible magnetic tape transports, more than two Type TU20A or TU20B transports may be desirable for some applications
Automatic Line Printer, Type LP15F or C[2]	Provides greater listing capabilities.
200 CPM Card Reader, Type CR03B	Provides another form of data input to the machine.
Additional Teletype Line Units, Type LT19E[3], and Teletypes, Type KSR33, KSR35 or equivalent (standard system is configured to handle up to 6 Teletype units including the console unit. The system may be expanded to handle up up to 17 units including the console unit).	Provides additional control terminals useful for multi-user programs.

[1] MM09 B and C core memory modules on the PDP-9.

[2] Two line printers supported on the PDP-9 are designated Type 647 and LPØ9.

[3] LT19B on the PDP-9.

SECTION 2

BFKM15 - NON-RESIDENT BACKGROUND/FOREGROUND MONITOR

2.1 INTRODUCTION

The non-resident portion of the Background/Foreground Monitor, entitled BFKM15, is identical in nature to the Keyboard listening section of the ADVANCED Monitor. BFKM15 reads and interprets commands typed by the user at either the Background control Teletype or the Foreground control Teletype.

There are three kinds of commands which the user may type:

a. Requests for information, such as a directory listing of unit Ø of the system device;
b. Allocation parameters, such as free core required, number of open files, and I/O devices to be used;
c. Requests to load a system or user program.

2.2 LOCATION AND WHEN CALLED

BFKM15 is loaded from register 12000 of the highest core bank to the top of memory and is transparent to the user since it is always overlayed.

When the Background/Foreground system is loaded or reloaded to start a new Foreground job, the Resident Monitor is first loaded into lower core from unit Ø of the system device, either by use of the paper tape bootstrap or by typing CTRL C[1] at the Foreground control Teletype. The Resident Monitor then brings the Non-resident Monitor into the top of memory. When operating in the Foreground, BFKM15 runs with memory protect disabled.

After the Foreground user program has been loaded and has started to run, the Non-resident Monitor is reloaded with memory protect enabled, to converse with the user at the Background control Teletype. BFKM15 is also reloaded whenever the Background job exits or the user types CTRL C at the Background control Teletype.

In both the Foreground and the Background, after the user has given a command to load a program, the Non-resident Monitor brings the System Loader into memory from the system device, overlaying the Non-resident Monitor.

---

[1]Refer to Section 3.4 for a discussion of CTRL C.

2.3 INITIAL OPERATION

When BFKM15 is started for the Foreground job, it must perform some initialization of which the following is of interest:

a. Set the contents of .SCOM+25 to the initial size of free core to be allotted to the Foreground job, in addition to the space required by the Foreground user programs. The initial value of .SCOM+25 is set during system generation. This value must take into consideration the initial size of free core to be allotted to the Foreground job plus the space required by the Foreground user program. The user may change free core allotted by issuing the FCORE command, described in Section 2.5.3.

b. BFKM15 checks the entire Foreground Device Assignment Table (.DATF) to see if any of those .DAT slots request the Teletype handler and the unit number currently assigned to the Background control Teletype. If so, those slots are changed to the Foreground control Teletype and a message is output as in the following example.

EXAMPLE 1: The Foreground control Teletype is TT1, the Background control Teletype is TTØ, and the initial contents of .DATF slots 1 and 3 refer to TTAØ. .DATF slots 1 and 3 will be changed to refer to TTA1 and the following message will be printed on the Foreground control Teletype:

```
FGD .DATS CHANGED TO TTA1:

 1  3

FKM15 V3A[1]
$
```

The Non-resident Monitor identifies itself to the Foreground user by printing FKM15 V3A and types $ whenever it is ready to accept a command.

When BFKM15 is started for the Background job, it performs initialization, of which the following is of interest:

a. It builds the initial configuration of the Background .DAT table (.DATB). Any .DATB slots which request a single user version of a device handler (for example, DTF) will be changed to the multi-user handler (DTA in this case) if it is already in core for the Foreground job or if it is the resident system device handler.

b. BFKM15 will check all Background .DAT slots to make certain that they do not conflict with Foreground I/O. The Resident Monitor contains, for this purpose, a table (.IOIN) which lists all I/O handlers and unit numbers in use. The following occurs:

(1) If a handler for this I/O device is not already in core, the Background .DAT slot is left untouched.

[1] FKM15 is the page mode monitor printout. F9/15 is the bank mode monitor printout. Bank mode users should substitute the correct monitor printout in further references.

(2) If a single user handler for this device is already in core for use by the Foreground job, by definition the Background job may not use this device. Therefore the Background .DAT slot is cleared (set to zero).

(3) If the multi-user handler for this device is in core, but the device unit number in question is not assigned to the Foreground job, Background is allowed to share that handler. Unit Ø of the system device may always be used by the Background job.

(4) If the Background .DAT slot requests a multi-user handler and unit number already assigned to the Foreground, normally this is illegal and that .DAT slot will be cleared. However, some users may wish to allow both jobs to access the same unit. Normally, this is permitted only for bulk storage devices (DECtape, Disk, etc.) provided that the Foreground user typed the command SHARE, explained in Section 2.5.7.

If the initial Background .DAT table was altered by clearing .DAT slots for the reasons given above, a message will be output to the Teletype as in the following example.

EXAMPLE 2: The Foreground job is running and has been assigned device handlers and unit numbers DTA1, DTA2, TTA1, TTA2, and PPA (paper tape punch handler - not shareable). The initial Background .DAT table contains conflicting requests as follows:

.DAT SLOT	CONTENTS
-15	DTA1
-4	DTA2
3	TTA2
7	PPAØ

The following will be printed on the Background control Teletype when BFKM15 is first loaded:

```
BGD .DATS CLEARED BECAUSE OF FGD I/O:

 -15 -4 3 7

FCONTROL  = TTA1

FGD DEV-UNITS:

  TTA2
  DTA1
  DTA2
  PPAØ

BKM15 V3A[1]
$
```

[1] BKM15 is the page mode monitor printout. B9/15 is the bank mode monitor printout. Bank mode users should substitute the correct monitor printout in further references.

FCONTROL indicates which unit is the Foreground control Teletype. The remainder of the message indicates what I/O is being used by the Foreground job. The Monitor identifies itself to the Background job user as BKM15 V3A and signals that it is ready to accept a command by printing $.

2.4 INFORMATION COMMANDS

The following information commands exist in Background/Foreground:

COMMAND	USE
LOG	To print a comment
REQUEST	To examine .DAT slots
DIRECT	To obtain a directory listing
INUSE	To list information about core and I/O in use by the Foreground.

2.4.1 The LOG Command (L)

This command is legal in both Foreground and Background and may be abbreviated by the single letter L. It is used to record comments on the Teletype. Unlike all other commands, LOG is terminated only by the character ALTMODE, so that multiple comment lines may be typed.

EXAMPLE 3:

```
$LOG      THIS LINE)
    AS WELL AS THIS ONE)
    AND THIS ONE ARE IGNORED (ALTMODE)
$
```

2.4.2 The REQUEST Command (R)

This command is legal in both Foreground and Background and may be abbreviated by the single letter R. It is used to examine the contents of all or part of the user's .DAT table. The Foreground user may examine only the Foreground .DAT table and the Background user, only the Background .DAT table.

FORM 1: R)

This requests a printout of the entire .DAT table. No example is given since R is essentially the same request as in the ADVANCED Monitor System.

FORM 2: R␣USER↵

This requests a printout of the contents of all the positive numbered .DAT slots. The result, again, is the same as in the ADVANCED Monitor System.

FORM 3: R␣XYZ↵

Here, XYZ stands for the name of a system program; e.g., MACRO, PIP, F4, LOAD, etc. The names given must be identical to those used to load the programs. The information printed, as in the ADVANCED Monitor System, is those .DAT slots used by the given system program. Since, at present, the only system program load commands allowed in the Foreground are LOAD, GLOAD, PIP and EXECUTE, only these four may be used in Foreground REQUEST commands.

FORM 4: R␣.DAT␣j, k, l, ... , r, s↵

Here, j, k, l, etc., are .DAT slot numbers.

EXAMPLE 4:

```
$R␣.DAT␣-3, -1, 4, 7↵
TTA1 DTA2 NONE LPAØ
$
```

2.4.3 The DIRECT Command (D)

This command is legal in both Foreground and Background and may be abbreviated as D. The format is:

D␣n↵

where n = a unit number (Ø through 7) on the system device. Directory listings have been altered in BFKM15 to print the number of free blocks before the file names. The Background user may not request directory listings of any units owned by the Foreground job unless the Foreground user typed the SHARE command (see below).

2.4.4 The INUSE Command (I)

This command is legal only in the Background and may be abbreviated by the single letter I. It causes the Monitor to print the first free core location above the Foreground job, the Foreground control Teletype unit number, and any other I/O used by Foreground.

EXAMPLE 5:

```
$I↵
1ST REG ABOVE FGD = 323Ø1
FCONTROL = TTA2
FGD DEV-UNITS:
     DTA1
     LPAØ
$
```

2.5 ALLOCATION COMMANDS

The following commands assign parameters, controls, and conditions:

COMMAND	PURPOSE
ASSIGN	To assign I/O handlers to .DAT slots
FILES	To specify handler file capacity
FCORE	To set up Foreground free core
FCONTROL	To select Foreground control Teletype
BCONTROL	To select Background control Teletype
NEWDIR	To write a new file directory
SHARE	To allow jobs to share same I/O units
NOSHARE	To nullify effect of SHARE
7CHAN	To specify 7-channel MAGtape operation
9CHAN	To specify 9-channel MAGtape operation
MPOFF	To let Background access all of core
MPON	To nullify effect of MPOFF

2.5.1 The ASSIGN Command (A)

This command is legal in both Foreground and Background and may be abbreviated by the single letter A. Its format and function are, with a few exceptions, identical to the same command in the ADVANCED Monitor System.

The format is:

A␣␣DDLN␣␣m, n, ..., p/ .../DDLN␣␣m, n, ..., p↵

where DD stands for the two letter device name; e.g., DT for DECtape, PP for paper tape punch, etc.

L represents the third letter of a device handler name and is optional. If not given, the third letter is assumed to be A; e.g., DT1 = DTA1. The "A" version of a handler is the multi-user, shareable handler, provided that one exists. PPA, for example, is not a multi-user handler.

N is the unit number to go with the device handler and is also optional. If the unit number is missing, N is assumed to be Ø, e.g., DTA = DTAØ. Therefore, DT = DTØ = DTA = DTAØ. The letters m, n, ..., p stand for .DAT slot numbers. The slash (/) separates handlers.

To clear out a .DAT slot, assign NONE to it. If any error is detected in the command, none of the assignments will be made.

The Foreground and Background users may make assignments only to their respective .DAT tables. Foreground may not assign TTAØ if, for example, that is the Background control Teletype. Since DTA is permanently in core with the Resident Monitor (assuming that DECtape is the system device) DTE, DTF, etc., when assigned, will automatically be changed to DTA. This applies as well to handler assignments made in the Background whenever the multi-user version of the handler is in core for Foreground use.

Background .DAT slot assignments are tested to ensure that they do not conflict with Foreground I/O, as explained in section 2.3. Whenever the Monitor detects such a conflict, it will print the message:

OTHER JOB'S DEV-UNIT

To ensure that no conflict can occur when assigning the core-to-core handler, COA., the unit number is preset to Ø for Foreground and 1 for Background. The core-to-core handler disregards the unit number anyway.

2.5.2 The FILES Command (F)

This command is legal in both Foreground and Background and may be abbreviated as F. The purpose of this command is to save core space by limiting the number of I/O buffers assigned to multi-user device handlers.

The format of the FILES command is:

FILES␣DD␣N)

where: DD stands for the multi-user handler or device name (e.g., DTA or DT).
N stands for an octal file count.

EXAMPLE 6: Assume that the Foreground user programs are being loaded into core by the Foreground Linking Loader and that these programs use .DAT slots 1 through 1Ø. (.IODEV 1, 2, 3, ..., 1Ø). Further, assume that all 1Ø slots were assigned to DECtape, DTAn (the unit numbers are unimportant to this discussion).

Most multi-user handlers, DTA being one of them, require that I/O buffers be assigned to them externally. This is done by the various loaders. In this example, the Foreground Linking Loader, seeing that no FILES command was given for the handler DTA, must assume that the user wants 1Ø files open simultaneously. This will require 1Ø buffers, each 6ØØ octal words in size.

The FILES command is used to tell the loaders to assign a given number of buffers for a particular multi-user handler based on the maximum number of files that the user programs will have open simultaneously. Each multi-user handler has a maximum open file capacity; for example, DTA may have up to 2Ø octal. If 1Ø I/O buffers are assigned for DTA in the Foreground, then only up to 1Ø may be assigned for Background. The FILES command issued in the Foreground specifies only Foreground I/O buffers. Thus, to limit the number of I/O buffers assigned to the Background, the FILES command, for the same multi-user device, must also be issued in the Background.

At load-time, I/O buffers are set aside in core by the Loaders. The buffers are recorded in a table within the Resident Monitor, .BFTAB, but are not flagged for the exclusive use of particular device handlers. At run-time, each multi-user

handler which needs a buffer must request a buffer from the Monitor. The handler must also release the buffer to the pool when it is no longer needed.

The resident buffer[1], permanently assembled into the Resident Monitor, is always available to the Background job. In the event that the Background job were to .IODEV only one .DAT slot which is linked to a multi-user handler that requires external buffers, (DTA. for example) the user could save 6ØØ registers by typing:

```
$FILES␣DT␣Ø
```

that is, assign one less buffer than is needed.
In the FILES command, the pseudo-device .. is recognized. The size of the external buffer for this pseudo-device is 1ØØ octal. Some functions in multi-user handlers may require a smaller buffer size than others. If the user were only to use such function types, he could type, for example, $FILES␣DT␣Ø and $FILES␣..␣N. In DTA., .TRAN and .MTAPE commands only require the smaller buffer.

2.5.3 The FCORE Command

This command is legal only in the Foreground and may not be abbreviated.

The format of the FCORE command is:

```
FCORE␣N)
```

where N is the amount (in octal) of free core requested for the Foreground job.

As in the ADVANCED Monitor System, unused (free) core is defined by the address pointers in the registers .SCOM+2 and .SCOM+3, the lowest and the highest free core location, respectively. Since both the Foreground and the Background jobs have their own separate free core areas, the values in .SCOM+2 and .SCOM+3 are changed appropriately whenever control passes from one job to the other.

The FCORE command allows the Foreground user to specify how much free core his program will need, in addition to that required to load his program. The default value for FCORE is specified during system generation. It is possible for all of core to be assigned to Foreground. This means, however, that there will be no room for Background to run, which is perfectly legal. If this is the case, the message:

```
SORRY, NO ROOM FOR BGD
```

[1] The resident buffer ($6ØØ_8$ words) is assumed to be large enough to be used by any multi-user handler which might be used by the loaders.

is printed on the control Teletype.

2.5.4 The FCONTROL Command

This command is legal only in the Foreground and may not be abbreviated. It is used to transfer control from the control Teletype to some other Teletype unit.

The format of the FCONTROL command is:

```
FCONTROL␣N)
```

where: N is the number (octal) of any Teletype on the system.

If N is already the Foreground control Teletype, the command is ignored. If N is the current Background control Teletype, the two Teletypes are swapped but no message will be printed to this effect. Changing the Background control Teletype may affect Foreground .DAT slots and an appropriate message will be printed on the Foreground control Teletype. This is fully explained in the next section on the BCONTROL command.

When FCONTROL changes the Foreground control Teletype, the following action takes place:

a. The following message is printed on the old control unit:

```
CONTROL RELINQUISHED
```

b. The system is reloaded from the system device.

c. The Monitor prints

```
FKM15 V3A
$
```

on the new Foreground control unit and is ready to accept commands there.

2.5.5 The BCONTROL Command

This command is legal both in the Foreground and in the Background and may not be abbreviated. It is used to transfer control from the Background control Teletype to some other Teletype unit.

The format of the BCONTROL command is:

```
BCONTROL␣N)
```

where N is the number (octal) of any Teletype on the system. This command is

illegal and is ignored if

a. N is the Foreground control Teletype
b. N has been .IODEVed by a Foreground user program
c. N is already the Background control Teletype

If the Background control Teletype is changed by either a BCONTROL or FCONTROL command in the Foreground, all Foreground .DAT slots which now refer to the new Background control unit will be changed to the Foreground control unit to avoid I/O conflict. Should that situation occur, the following example shows what would be printed on the Foreground control unit:

```
FGD .DATS CHANGED TO TTA1
 -6 2 7 1Ø
```

If BCONTROL is issued in the Background, the following action takes place:

a. The following message is printed on the old control unit:

```
CONTROL  RELINQUISHED
```

b. ↑C is printed on the new unit
c. The Non-resident Monitor (BFKM15) is reloaded for Background from the system device
d. The Monitor prints

```
BKM15 V3A
$
```

on the new Background control Teletype and is ready to accept commands there.

2.5.6 The NEWDIR Command (N)

This command is legal in both Foreground and Background and may be abbreviated by the single letter N. Just as in the ADVANCED Monitor System, this command allows the user to write a new file directory on some unit of the system device.. However, space will not be reserved for a ↑Q (CTRL Q) area.

The format of the NEWDIR command is:

```
N␣M)
```

where M is some unit number (octal) on the system device. Unit Ø may not be used. The Background may not write a new file directory on a unit that belongs to the Foreground unless the Foreground has issued the SHARE command (see below).

2.5.7. The SHARE Command (S)

This command is legal only in the Foreground and may be abbreviated by the single letter S. Its purpose is to allow the Background job to assign and to use the same units of any I/O devices that belong to the Foreground job, provided that they are unit-shareable devices[1] (DECtape, Disk, MAGtape, etc.) and that the device handlers are the multi-user versions. The user must be careful when allowing this condition to occur. The "tape" could be fouled if both jobs were to try to use the same unit for output at the same time.

The SHARE command also removes the restriction that the Foreground user program may not use unit Ø on the system device. Normally, this unit is reserved for the Background.

The format for this command is:

SHARE)

2.5.8 The NOSHARE Command

This command is legal both in Foreground and in Background and may not be abbreviated. It nullifies the effect of any previous SHARE command; i.e., does not allow the Background to share device units with the Foreground.

When NOSHARE is issued in the Background, it may cause some Background .DAT slot to be cleared. A message, as in Example 2, will be printed to that effect.

The command format is:

NOSHARE)

2.5.9 The 7CHAN Command (7)

This command is legal only in the Foreground and may be abbreviated by the single character 7. The effect of this command is to clear bit 6 in .SCOM+4 to inform the Magtape device handlers that the default assumption is 7-channel operation.

The format of the 7CHAN Command is:

7CHAN)

[1]Normally, only mass storage devices are unit-shareable.

2.5.10 The 9CHAN Command (9)

This command is legal only in the Foreground and may be abbreviated by the single character 9. It sets bit 6 in .SCOM+4 to inform the Magtape device handlers that the default assumption is 9-channel operation.

The format of the 9CHAN command is:

```
9CHAN)
```

2.5.11 The MPOFF Command

This command is legal only in the Foreground and may not be abbreviated.

The format is:

```
MPOFF)
```

Under normal circumstances, the Background job operates in user mode (memory protect enabled) with the memory protect boundary register set from the contents of .SCOM+32. The MPOFF Command does not disable memory protect for Background; it causes the contents of the boundary register to be set to zero, independent of .SCOM+32.

The effect this has is to allow the Background job to reference, modify, and transfer to any location in core memory. Any attempt to do so via a system macro call (CAL sequence, such as .WAITR) will not result in a terminal error, .ERR Ø36. Normally, the Monitor's CAL handler would validate Background arguments by comparison with .SCOM+31 or .SCOM+32, as appropriate.

Since the Background still runs with memory protect on, IOT instructions, non-existent memory references, double XCT instructions, HLT, and OAS will trap to the Monitor. OAS[1] is executed by the Monitor whether or not the MPOFF command was issued. IOT instructions are executed by the Monitor for the Background job (this includes IOT's that cause a skip) when MPOFF is in effect. The reader is cautioned to avoid the use of instructions, such as CAF, EBA, DBA, ISA, which could play havoc with the system if executed in the Background. The MPOFF facility was provided to allow a limited amount of Foreground debugging by using DDT in the Background (strictly for examination and modification--no breakpoints).

2.5.12 The MPON Command (M)

This command is legal in both Foreground and Background and may be abbreviated by the letter M.

---

[1]OAS must not be microcoded with any skip instruction.

The format is:

MPON↵

The MPON command nullifies the effect of MPOFF, thereby protecting the Foreground job from the Background job in the normal manner.

2.6 PROGRAM LOAD COMMANDS

In the Foreground, only four load commands are legal: LOAD↵, GLOAD↵, PIP↵, and EXECUTE␣XXX↵. EXECUTE may be abbreviated by the single letter E. LOAD and GLOAD have the same meaning and effect as in the ADVANCED Monitor System.

The following program load commands exist in the Background:

PATCH↵	MACROA↵
CHAIN↵	LOAD↵
F4↵	GLOAD↵
F4A↵	DDT↵
EDIT↵	DDTNS↵
PIP↵	DUMP↵
EXECUTE␣XXX↵	UPDATE↵
MACRO↵	BFSGEN↵
DTCOPY↵	SRCCOM↵

2.7 FINAL OPERATION

After BFKM15 has received a program load command from either the Foreground or the Background, it will bring the System Loader (.SYSLD) into the top of core overlaying BFKM15. In the Foreground, .SYSLD is actually the Foreground Linking Loader. In the Background, .SYSLD loads Background System Programs, including the Background Linking Loader.

2.8 CONTROL CHARACTERS

While control is in BFKM15, the user may type CTRL P to terminate execution of the current command and to restart. Restart in this manner does not nullify the effect of previously executed commands; e.g., will not reset the .DAT table to its initial configuration. To reload the Monitor for the current job, the user may type CTRL C.[1]

[1]Refer to section 3.4 for a discussion of CTRL C.

2.9 SUMMARY OF COMMANDS

LEGAL IN		ABBREVIATION	COMMAND EXAMPLE
F	B	A	ASSIGN␣DTA1␣2, 3/TT1␣1, 4/DT␣-4↵
F	B		BCONTROL␣2↵
	B		BFSGEN↵
	B		CHAIN↵
F		7	7CHAN↵
F		9	9CHAN↵
	B		DDT↵
	B		DDTNS↵
F	B	D	DIRECT Ø↵
	B		DTCOPY↵
	B		DUMP↵
	B		EDIT↵
F	B	E	EXECUTE␣XXX↵
	B		F4↵
	B		F4A↵
F			FCONTROL␣1↵
F			FCORE␣14ØØ↵
F	B	F	FILES␣DT␣3↵
F	B		GLOAD↵
	B	I	INUSE↵
F	B		LOAD↵
F	B	L	LOG␣.....(ALTMODE)
	B		MACRO↵
	B		MACROA↵
F			MPOFF↵
F	B	M	MPON↵
F	B	N	NEWDIR␣5↵
F	B		NOSHARE↵
	B		PATCH↵
F	B		PIP↵
F	B	R	REQUEST␣XXX↵ or REQUEST␣USER↵ or REQUEST␣.DAT j,k,l↵ or REQUEST↵
F		S	SHARE↵
	B		SRCCOM↵
	B		UPDATE↵

SECTION 3

CONTROL CHARACTERS

3.1 PURPOSE

Control characters are single characters, typed by the user at a Teletype, which request special action by the Monitor. Except for the character, RUBOUT, all control characters are formed by holding down the control key, CTRL, while striking the appropriate letter key.

The characters CTRL U and RUBOUT are used as "erase" characters during Teletype input or output. CTRL C, CTRL P, CTRL S, and CTRL T are used to interrupt the operation of the current program and to transfer control elsewhere. CTRL R is used to restart I/O after a not-ready condition has been detected for some device. CTRL Q stops the current job and dumps memory onto a specified area of some unit of the system device. CTRL D effects an end-of-file condition during Teletype input.

3.2 CONTROL TELETYPE

In the Background/Foreground System, which may accommodate up to 17 (decimal) Teletype units[1], two Teletypes are designated as control Teletypes (one for Background and one for Foreground). Initially, it is assumed that unit Ø (the console Teletype) is the control Teletype for Background and unit 1 is the control unit for Foreground[2].

Control Teletypes differ from the other units in two ways:

a. They are used to converse with the Non-resident Monitor and system programs in order to set up parameters and conditions for a job and to initiate the loading and execution of programs.
b. Certain control functions are honored only at control Teletypes; i.e., they are ignored if they are typed on the other Teletype units (see Section 3.4 and following).

[1]The system as shipped to customers will handle a maximum of 6 Teletypes. Expansion requires a simple reassembly of the code for the Resident Monitor.

[2]The initial control Teletypes are specified during system generation.

3.3 TELETYPE HANDLER

The multi-user Teletype handler (TTA) which is imbedded in the Resident Monitor makes special tests for control characters when it receives typed input. Normally, when no .READ request has been issued to a Teletype, characters received from that unit are ignored unless they are control characters. A description of the action taken in each case is given in the following paragraphs.

3.4 CTRL C (↑C)

This character is ignored unless typed at a control Teletype. It is echoed to the teleprinter as ↑C.

If a Background job is not in core and the user types CTRL C at the Foreground control Teletype, ↑C is echoed to it and the Resident Monitor is reloaded by the resident bootstrap.

If a Background job is in core when CTRL C is typed on the Foreground control Teletype, ↑B is echoed to it to indicate that a Background job exists, a "bell" is sent to the Background control Teletype, and a flag is set indicating that CTRL C has been typed in the Foreground. What happens thereafter depends on which job is the "confirmer", a parameter set by the System Generator. Once CTRL C has been entered on the Foreground control Teletype, the Foreground job is terminated.

When Foreground is the "confirmer", the second time CTRL C is typed on the Foreground control Teletype ↑C is echoed to it and the Resident Monitor is reloaded.

When Background is the "confirmer", CTRL C typed on the Foreground control Teletype causes ↑B to be printed on the Foreground control Teletype and a "bell" to be sent to the Background control Teletype. Thus Foreground cannot abort Background. When CTRL C is typed on the Background control Teletype, ↑C is echoed to it and then the Resident Monitor is reloaded by the resident bootstrap.

In the normal case where Foreground is running and CTRL C is typed on the Background control Teletype but not on the Foreground control Teletype, the Foreground job is not affected. The Background job is aborted and the Non-resident Monitor is reloaded to start up a new Background job.

The "confirmer" flag is .SCOM+1Ø4.
Ø = Foreground.
1 (nonzero) = Background.

3.5 CTRL S (↑S)

CTRL S is recognized only at a control Teletype and, specifically, only after the Monitor has printed ↑S. This is the result of loading a user program by giving the command $LOAD (instead of $GLOAD) to the Non-resident Monitor. Both commands bring in the Linking Loader to load user programs. $GLOAD means LOAD-AND-GO. $LOAD means load the user programs, signal the user that this has been done (by printing ↑S), and then wait for the go-ahead signal (when the user types CTRL S).

This feature allows the user to set up I/O devices before starting his program. When CTRL S is typed by the user and is accepted by the Monitor, ↑S is echoed back to the teleprinter.

3.6 CTRL T (↑T)

This character is recognized only at the Background control Teletype when the user has called in the system program DDT. When CTRL T is typed and accepted, it is echoed to the teleprinter as ↑T.

CTRL T provides a means of interrupting the execution of a user program and transferring control to DDT. When CTRL T is typed, the Monitor saves the status of the Link, page/bank mode, and memory protect along with the interrupted PC in .SCOM+7 so that DDT will be able to return control to the user program at the point at which it was interrupted. The contents of the AC at the time of interruption is returned in the AC and saved by DDT.

3.7 CTRL P (↑P)

CTRL P is the interrupt and restart character available to user and system programs. When it is typed on some Teletype and is accepted by the Monitor, ↑P is echoed to the teleprinter on that unit.

In the Background/Foreground system there are two types of CTRL P functions:

1. NORMAL CTRL P and
2. REAL TIME CTRL P.

The two CTRL P functions are described, individually, in paragraphs 3.7.1 and 3.7.3.

Setting a CTRL P restart address (ADDR) is accomplished by issuing the I/O MACRO .INIT to any .DAT slot linked to the Teletype handler.

The format of the .INIT macro is:

```
.INIT A,M,P+ADDR
```

which is expanded by the MACRO assembler into the following machine code:

```
LOC      CAL M8+A9-17
LOC+1    1
LOC+2    P+ADDR9-17
LOC+3    Ø
```

where A = a .DAT slot number (octal radix)

M = transfer mode
- Ø = Input
- 1 = Output

ADDR = a 15-bit address (octal) of a restart point in the program or of the entry point of a closed real-time subroutine.

P = priority code
- Ø = Normal CTRL P
- 1ØØØØØ or 2ØØØØØ = Mainstream (REAL-TIME)
- 3ØØØØØ = No change to CTRL P
- 4ØØØØØ = Priority level of the .INIT
- 5ØØØØØ = API level 5
- 6ØØØØØ = API level 6
- 7ØØØØØ = API level 7

Background requests to an API level (4ØØØØØ - 7ØØØØØ) will be converted to Mainstream since Background programs cannot use the API software levels.

3.7.1 NORMAL CTRL P

A .INIT to set up a NORMAL CTRL P (priority code Ø) may be done only to a control Teletype. NORMAL CTRL P was so named because the action taken when the user types CTRL P is nearly the same as in the ADVANCED Monitor System.

When a control Teletype has been set up for a NORMAL CTRL P and that character is typed by the user, the Teletype handler will abort all Teletype I/O for that job (Background or Foreground). The Monitor will, when control is at Mainstream, save the status of the Link, page/bank mode, and memory protect with the interrupted PC in .SCOM+10 (whose contents are swapped in and out for Background and Foreground), return the interrupted AC to the AC, and transfer control to the restart address ADDR as specified by the last .INIT.

NOTE

When the Monitor processes a CTRL T or a NORMAL CTRL P, it kills any pending mainstream real-time routines to be run by zeroing the contents of .SCOM+57 (Foreground) or .SCOM+61 (Background). The user's program (if NORMAL CTRL P) or the user (if CTRL T) must zero the entry points of all his mainstream real-time routines. CTRL P and CTRL T do not affect API level real-time requests.

If the restart address ADDR = Ø, CTRL P to the given Teletype will be disabled; i.e., ignored if typed (except if P = 3ØØØØØ).

3.7.2 No Change

If .INIT for a given Teletype unit contains the priority code 3ØØØØØ, the CTRL P restart address for that unit is not changed. DDT uses this so that it can .INIT to abort a .READ to the Teletype without altering the CTRL P address set up by the user's program.

3.7.3 REAL-TIME CTRL P

A .INIT to set up a REAL-TIME CTRL P may be done to any Teletype unit. When so set up and the user types CTRL P, I/O to that Teletype is aborted. Control eventually goes to a closed real-time subroutine, ADDR, at the priority level defined by P, in the same manner as for a .REALR, .REALW, or .TIMER request.

If the restart address ADDR = Ø, CTRL P to the given Teletype will be disabled, i.e., ignored if typed.

REAL-TIME CTRL P is useful for multi-user programs, for instance multi-user FOCAL, where each Teletype has the ability to interrupt and restart.

3.8 CTRL R (↑R)

In the Background/Foreground system, I/O device handlers which detect a not-ready condition will request the Monitor to print a message on the appropriate control Teletype. The line printer handler message, for instance, would be:

```
LPØ NOT READY
```

The unit number has no significance for the line printer. Some single-unit handlers, such as the card reader handler, use the unit number designation to indicate the cause of the not-ready condition. After the message has been printed, the user should ready the device and then type CTRL R, which is echoed as ↑R. I/O for that device is then resumed.

While the Monitor is waiting for the user to type CTRL R, the user's program continues execution provided that it is not hung up waiting for completion of I/O from the not-ready device. The Monitor can handle one not-ready condition per job. Should a second not-ready request occur while another is being processed, job execution will be aborted with a .ERR ØØ4 terminal error.

3.9 CTRL Q (↑Q)

CTRL Q may be typed at any time, but it is ignored if it is not issued at a control Teletype.

The purpose of typing CTRL Q is to stop program execution and to dump all of core memory onto a specified area of some unit on the system device. The dump starts with block 1Ø1 octal on the given unit and overlays any data that may have existed in that area on the output device. A 16K system will dump 1ØØ octal blocks (1Ø1-2ØØ); a 24K system, 14Ø octal blocks (1Ø1 - 24Ø); a 32K system, 2ØØ octal blocks (1Ø1 - 3ØØ).

To ensure that CTRL Q will not overlay useful data, the user must employ the system program PIP to write a new file directory on that unit, using the (S) switch to reserve space for CTRL Q. For example:

```
>N␣XXu␣␣(S)
```

where XX is the device name and u the unit number. Note that the size of the CTRL Q area reserved is based on the amount of core existing in the system in which the new directory is written. The area reserved on a DECtape in a 16K system is not sufficient to do a protected CTRL Q in a 24K or 32K system.

When the Monitor accepts CTRL Q, it first terminates execution of the job (Foreground if Foreground CTRL Q, Background if Background CTRL Q). This involves calling all device handlers tied to that job to stop I/O, clearing all Monitor queues of entries for that job and disabling all control characters for that job except CTRL C.

The Monitor then prints ↑Q on the appropriate control Teletype and reads one character. The user must then type the number of the unit on which the dump is to occur. Unit zero may not be used. If the SHARE command is not in effect, a dump may not be done to a unit which belongs to the other job. If the Monitor rejects the typed character, it prints ↑Q again and waits for another character.

When the unit number is accepted, the dump takes place; then the Monitor is automatically reloaded. A Background CTRL Q does not affect Foreground. A Foreground CTRL Q, on the other hand, aborts the Background job. It is not possible to load and restart a core dump in Background/Foreground.

3.10 CTRL U (@)

CTRL U may be typed at any Teletype unit. If a .READ or .REALR was issued to some Teletype and the user decides he wants to "erase" everything he has typed for that read request, he may type CTRL U, which will be echoed to the teleprinter as @. The .READ or .REALR will still be in effect and he may then retype the input.

While output to a Teletype is being done as a result of a .WRITE or .REALW,

the user may type CTRL U to terminate the write. In this case nothing is echoed to the teleprinter.

3.11 RUBOUT (\)

This character is recognized only while the user is typing input to satisfy a .READ or .REALR request. When typed, RUBOUT deletes the last input character. For example, if the user has typed ABC and then RUBOUT, the C will be "erased". If he now types another RUBOUT, the B will be "erased". Every time a character is so removed, the character \ is echoed to the teleprinter.

3.12 CTRL D (↑D)

The character CTRL D is recognized at all Teletypes and is echoed back as ↑D. When typing input, CTRL D effects an end-of-file condition by terminating the .READ or .REALR request and storing the end-of-file, ØØ1ØØ5, in the input line buffer header. Since the word pair count returned is a 1, any characters typed prior to the CTRL D for the same read request will be lost.

SECTION 4

LOADERS

4.1 INTRODUCTION

There are three program Loaders in the Background/Foreground system. On the system file directory they are listed as .SYSLD SYS[1], BFLOAD BIN[2] and EXECUT BIN[2].

.SYSLD is an absolute system program that functions as two loaders: when it is called in for Foreground loading, it is the Foreground Linking Loader; when it is called in for Background loading, it is the Background System Program Loader. BFLOAD is the Background Linking Loader.

EXECUTE operates in both Foreground and Background as a loader of overlay programs (XCT files) built by the CHAIN system program. A description of CHAIN and EXECUTE is given in the utility manual.

4.2 FOREGROUND LINKING LOADER

Link loading of the Foreground job is initiated by typing GLOAD (Load-and-Go) or LOAD (Load-and-Pause) to the Monitor at the Foreground control Teletype. The Foreground Link Loader (.SYSLD) is then brought into the top of memory, overlaying the Non-resident Monitor. The following message will then be printed:

```
FGLOAD V2A
>
```

The > signals the user that he may now type in his command string.

The command string format is the same as for the Linking Loader in the ADVANCED Monitor System:

```
>options←mainprog, others,... ALTMODE
```

[1] Operates in bank mode.

[2] Operates in page mode; operates in bank mode only when running in bank mode.

4.2.1 Option Characters and their Meanings

Character	Meaning
P	Print program names and their assigned relocation factors
C	Print common block names and their assigned locations
G	Print global symbol names and their definitions

4.2.2 Use of ← Terminator

Prior to the terminator ← all characters except option characters are ignored. Carriage return preceding the ← starts a continuation line headed by >. ALTMODE preceding the ← restarts the Loader; therefore, no loading is done unless the character ← appears in the command string.

If no option characters precede the ←, the default assumption is that no memory map is to be printed.

After the ←, type the program names (main program first - no extensions) separated by comma or carriage return. Terminate the command string with ALTMODE. Before the terminating ALTMODE has been typed, the Loader may be restarted by typing CTRL P. All files named in the command string may contain 1 or more program units, and all program units will be loaded in each file named.

4.2.3 Sequence of Operation

Once the command string has been accepted, the Loader will perform the following sequence of operations:

a. Load to end of file all user programs[1] specified in the command string, from .DATF -4. These programs are loaded from the bottom of core up, starting at the top of the Resident Monitor. Calls to external library routines via .GLOBL, common block definitions, and .IODEV requests are saved in the Loader's symbol table, built from the bottom of the Loader down. Programs containing executable code (which excludes BLOCKDATA subprograms) are relocated such that they do not overlap core page boundaries in the page mode system or core bank boundaries in the bank mode system.

[1] These programs will operate in page mode and must not execute the EBA instruction which would change operation to bank mode. All programs in the bank mode system operate in bank mode only. Avoid EBA instructions in the bank mode system. Although EBA instructions have no effect on the PDP-15, they are equivalent to a LEM (leave extend mode) on the PDP-9. LEM has disasterous results during a background/foreground system run.

b. If a library search is necessary and the contents of .DATF -5 is non-zero, the Loader will seek the user library, .LIBR5 BIN, via that .DAT slot, and will load all requested library routines[1] which it finds. I/O device handlers must not be in the user library.

c. If a library search is still necessary for non-I/O routines, the Loader will search the system arithmetic library[1], .F4LIB BIN, via .DATF -7 in the same manner as above. I/O device handlers must not be in .F4LIB.

d. If any I/O handlers[1] must be loaded, the Loader searches through the system I/O Library, .IOLIB BIN, via .DATF -7. After this has been done, program loading has terminated.

e. At this point, all undefined common blocks are defined and assigned core space. Common blocks are allowed to overlap page boundaries.

f. If there are still some undefined global symbols, they will be matched with common block names and, if a match is found, defined as the base address of the matching common block.

g. For all multi-user device handlers in use for the user's programs, external I/O buffers are assigned core space (if necessary) and recorded in .BFTAB within the Resident Monitor. The number of such buffers depends on the $FILES counts given by the user to the non-resident Monitor or, if no counts given, the number of .IODEV'ed .DAT slots calling those handlers. I/O buffers are allowed to overlap core boundaries.

h. The amount of free core assigned to the Foreground job (contents of .SCOM+25) is added to the current size of assigned Foreground core to determine the upper limit of the Foreground job. Pointers to the first and last registers in Foreground free core are then stored in .SCOM+2 and .SCOM+3, respectively.

i. The Loader now exits to the Resident Monitor. The Resident Monitor prints ↑S and waits for the user to type CTRL S, if the Loader is called by the LOAD command. Control then is given to the start address of the user's main program, which was stored in .SCOM+6 by the Loader.

4.3 BACKGROUND SYSTEM LOADER

Loading of all system programs is done by the System Loader (.SYSLD), which

[1]These programs will operate in page mode and must not execute the EBA instruction which would change operation to bank mode. All programs in the bank mode system operate in bank mode only.

also performs link loading for the Foreground. Initiation of the loading cycle is done when the user, in the Background, types a request to the Non-resident Monitor to load a system program; e.g., $PIP, $EDIT, etc.

The Non-resident Monitor puts a code number in .SCOM+5 to tell the System Loader which program to load. The System Loader is then loaded into upper core overlaying the Non-resident Monitor. When loading a Background program other than the Linking Loader or EXECUTE, .SYSLD contains a SYSBLK which lists the .DAT slots used by each system program and information about the load address, start address, size and initial block number on the system device for each system program. SYSBLK exists as block 4Ø on the system device and is also used by PATCH.

To load a system program in the Background, .SYSLD performs the following operations:

a. For each .DAT slot (with non-zero contents) required by a system program, it determines which device handlers[1] are needed; and, if a library search is necessary, it brings in the handlers from the file .IOLIB BIN on the system device through .DATB -7. They are loaded starting immediately above the top of the Foreground job.

b. I/O buffers are then assigned core space immediately above the handlers as in the description in paragraph 4.2.3g. The hardware memory protect bound is set above the handlers and buffers.

c. If the load command was $LOAD, $GLOAD, $DDT, or $DDTNS, the Background Link Loader (BFLOAD)[1], a relocatable file, is loaded starting just above the new hardware protect bound.

d. For all other system programs (excluding EXECUTE)[1], .SYSLD builds a short routine just above the hardware protect bound to bring in the program[2] overlaying the System Loader.

e. Finally, .SYSLD exits to the Resident Monitor[2], which establishes the new hardware protect bound and then passes control to the system program via the address stored by .SYSLD in .SCOM+5.

The Loader allows the loading of absolute .LOC programs prior to loading any relocatable files. This permits the user to load programs which may overlay parts of the Resident Monitor. Mixing of absolute and relocatable .LOC's in the same program file is not allowed and will be flagged as an error. The

[1]Operate(s) in page mode; operates in bank mode only when using bank mode system.
[2]Operates in bank mode.

Loader ensures that the relocatable programs do not overlay any of the absolute programs.

The Foreground Linking Loader is also responsible for loading the system program PIP[1] in the Foreground. The Foreground version of PIP exists in the system as the relocatable file PIP BIN. It is loaded by typing PIP as a command to the Non-resident Monitor[2].

4.4 BACKGROUND LINKING LOADER

Externally, the Background Linking Loader (BFLOAD) looks nearly the same to the user as the Foreground Linking Loader. When it has been loaded, it prints the following message on the Background control Teletype:

```
BGLOAD V2A
>
```

The command string processing is identical with that of the Foreground Linking Loader (see 4.2).

If the Load command was $DDT or $DDTNS, the system program DDT[1] (a relocatable file) has already been loaded into the top of core via .DATB -1, prior to reading in the command string.

Once the command string has been accepted, the Loader will perform the following sequence of operations:

a. Load to end of file all user programs[1] specified in the command string from .DATB -4. These programs are loaded from the top of core down. Calls to external library routines via .GLOBL, common block definitions, and .IODEV requests are saved in the Loader's symbol table, built from the top of the Loader upwards in core. Programs containing executable code (which excludes BLOCKDATA subprograms) are relocated such that they do not overlap page boundaries.

b. Same action as described in 4.2.3b, using .DATB -5.

c. Same action as described in 4.2.3c, using .DATB -7.

d. If any I/O handlers must be loaded, the Loader searches through .IOLIB BIN via .DATB -7. The handlers are relocated to run in lower core, that is, as if they were being loaded upwards in

[1]Operate(s) in page mode; operates in bank mode only when using bank mode system.
[2]Operates in bank mode.

core, starting just above the Foreground job. They may, however, be loaded above the Loader if the Loader is in the way.

e. Same action as described in 4.2 e,f,g. Common blocks are assigned space in upper core; I/O buffers, in lower core.

f. The hardware memory protect bound is established above the I/O handlers and buffers. Common blocks may go below the hardware protect bound.

g. If DDT was loaded and a symbol table was requested (not $DDTNS), the symbol table is compacted to delete entries not needed by DDT. The Loader determines where the symbol table should be moved; and, along with the I/O handlers which were loaded into upper core, builds a special .EXIT list which tells the Resident Monitor where to block transfer each segment. The DDT symbol table may be loaded below the hardware protect bound.

h. The Loader then exits to the Resident Monitor, which performs the block transfers, sets the new hardware memory protect bound, and transfers control to DDT (via .SCOM+5) or to the user program (via .SCOM+6), pausing to print ^S and waiting for the user to type CTRL S if the Load command was $LOAD.

4.5 LOADING XCT FILES

XCT files are overlay programs[1] built by the system program CHAIN and run by the system program EXECUTE.[1] Loading of an XCT file in either the Foreground or the Background is initiated by typing E␣XXX or EXECUTE␣XXX to the Monitor (where XXX is the file name without the extension XCT).

The Non-resident Monitor, BFKM15, stores the filename (.SIXBT format) in .SCOM+107, 110, and 111 for the Foreground, or .SCOM+112, 113, and 114 for the Background. If EXECUTE's .DAT slot requests the resident system device handler[2], the Monitor stores "XCS" as the extension. If EXECUTE's handler is different from the resident handler, the Monitor stores the extension "XCT".

The System Loader is then called in, overlaying the Non-resident Monitor in upper core.

4.5.1 EXECUTE in the Foreground

The following operations are carried out when EXECUTE is used in the Foreground:

---

[1] These programs will operate in page mode and must not execute the EBA instruction which would change operation to bank mode. All programs in bank mode system operate in bank mode only.

[2] Runs in bank mode, unlike most I/O handlers; in the bank mode system, all I/O handlers run in bank mode only.

a. EXECUTE's handler, if different from the resident handler, is loaded immediately above the Monitor.

b. The System Loader, which must open the XCT file, checks the extension. If "XCS", meaning EXECUTE's handler is the resident handler, the file is loaded via .DAT -7. If "XCT", it is loaded via .DAT -4. The extension is then set to "XCT".

c. The XCT file is read and checked that it was indeed built to be run in the Foreground of a PDP-15 in page mode. In the bank mode system, The XCT file is checked to ensure that it was built to run in bank mode.

d. The upper and lower core limits of the overlay structure are saved and a check is made that it does not overlay the Resident Monitor.

e. The .IODEV bit map in the XCT file is decoded. The loading bound is set immediately above the area of core to be occupied by the overlay structure and then all I/O handlers required by the XCT file are loaded. Also, another copy of EXECUTE's handler is loaded (the first copy will be overlayed).

f. EXECUTE is loaded.

g. Same action as described in 4.3.4g and h.

h. The Loader exits to the Resident Monitor. The Monitor gives control to EXECUTE, whose start address is stored in .SCOM+6 by the Loader.

4.5.2 EXECUTE in the Background

The following operations are carried out when EXECUTE is used in the Background:

a. EXECUTE's handler, if different from the resident handler, is loaded immediately above the Foreground job.

b. Same action as described in 4.5.1b.

c. The XCT file is read and checked that it was built to be run in the Background of a PDP-15 in page mode. In the bank mode system, the XCT file is checked to ensure that it was built to run in the bank mode.

d. The lower core limit of the overlay structure is saved and, when EXECUTE has been loaded, a test is made to ensure that they do not overlap.

e. The .IODEV bit map in the XCT file is decoded and then any I/O handlers needed by the file are loaded.

f. Same action as described in 4.3.4g.

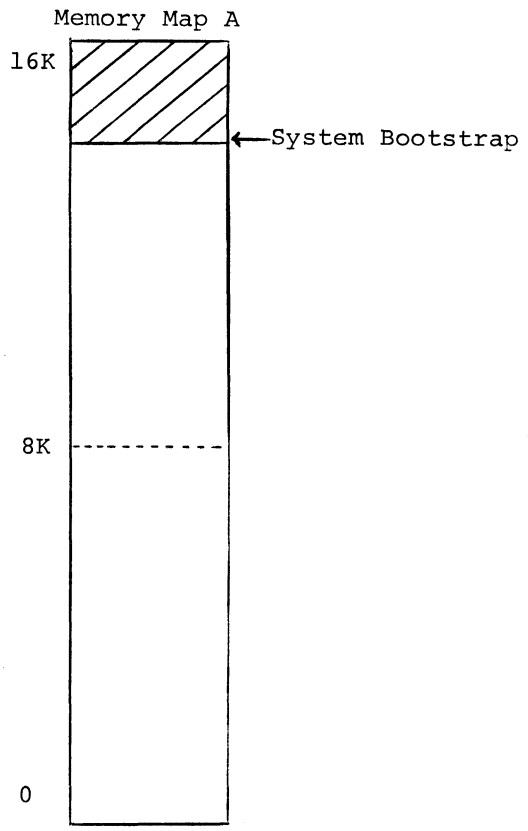
g. The hardware memory protect bound is set above the I/O buffers and EXECUTE is loaded starting above this bound.

h. Same action as described in 4.3e.

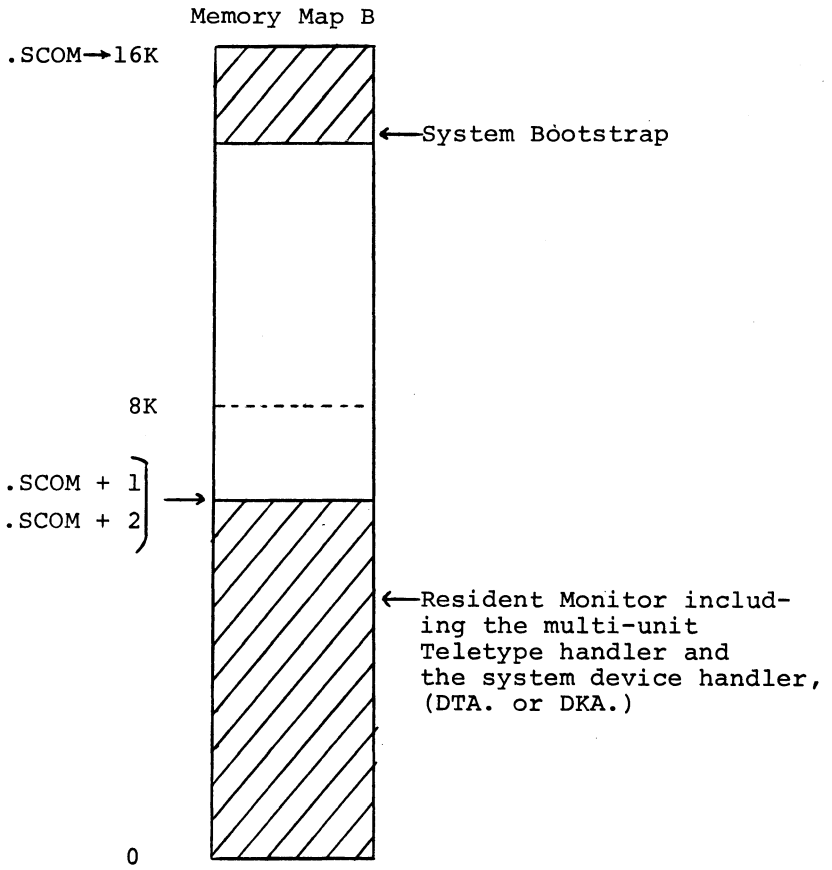
4.6 ERROR CONDITIONS

The number of different error messages in the Loaders has been expanded in Background/Foreground. These are tabulated in Appendix II. The error number is passed on to the Resident Monitor by a special error .EXIT macro (CAL sequence). Loader errors are non-recoverable After the error message is printed, the Monitor will automatically be reloaded to start another job.

4.7 SYSTEM MEMORY MAPS

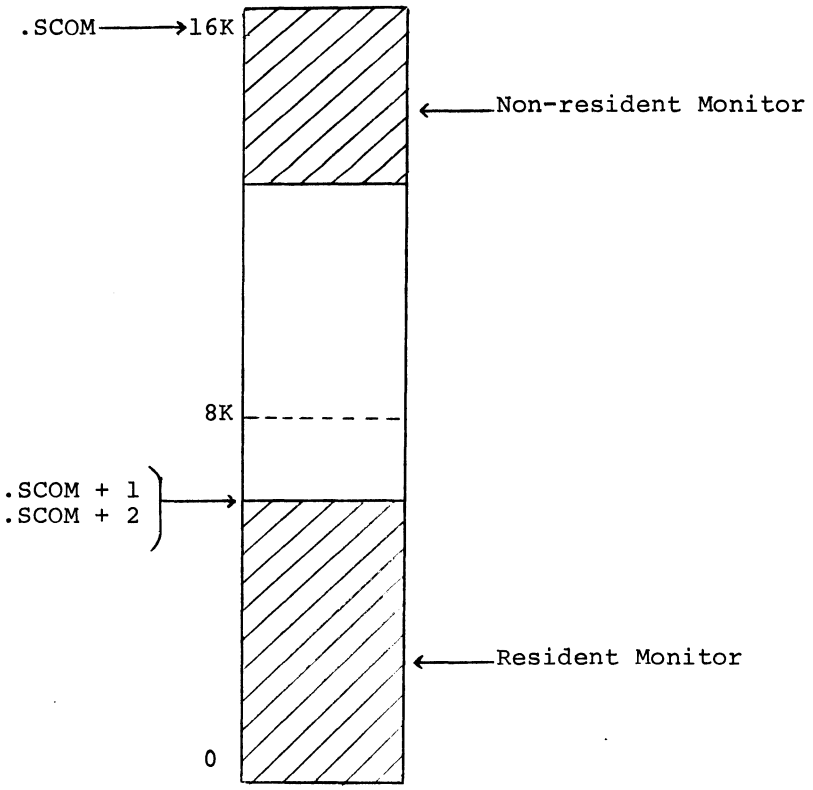


The System Bootstrap is loaded at the top of core via the paper tape reader in HRM format.



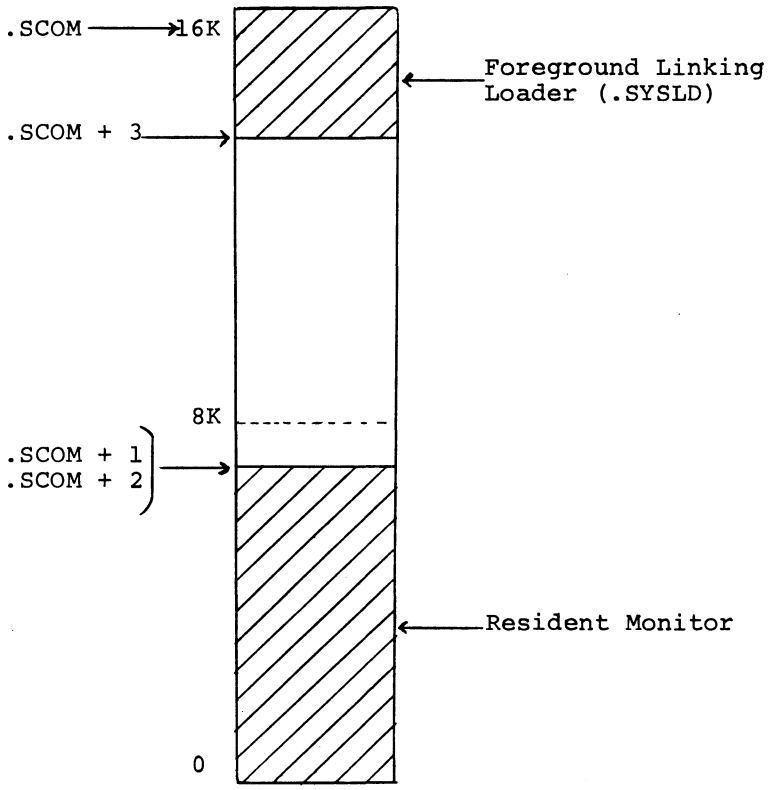
The System Bootstrap automatically loads the Resident Monitor from the system device into lower core.

Memory Map C



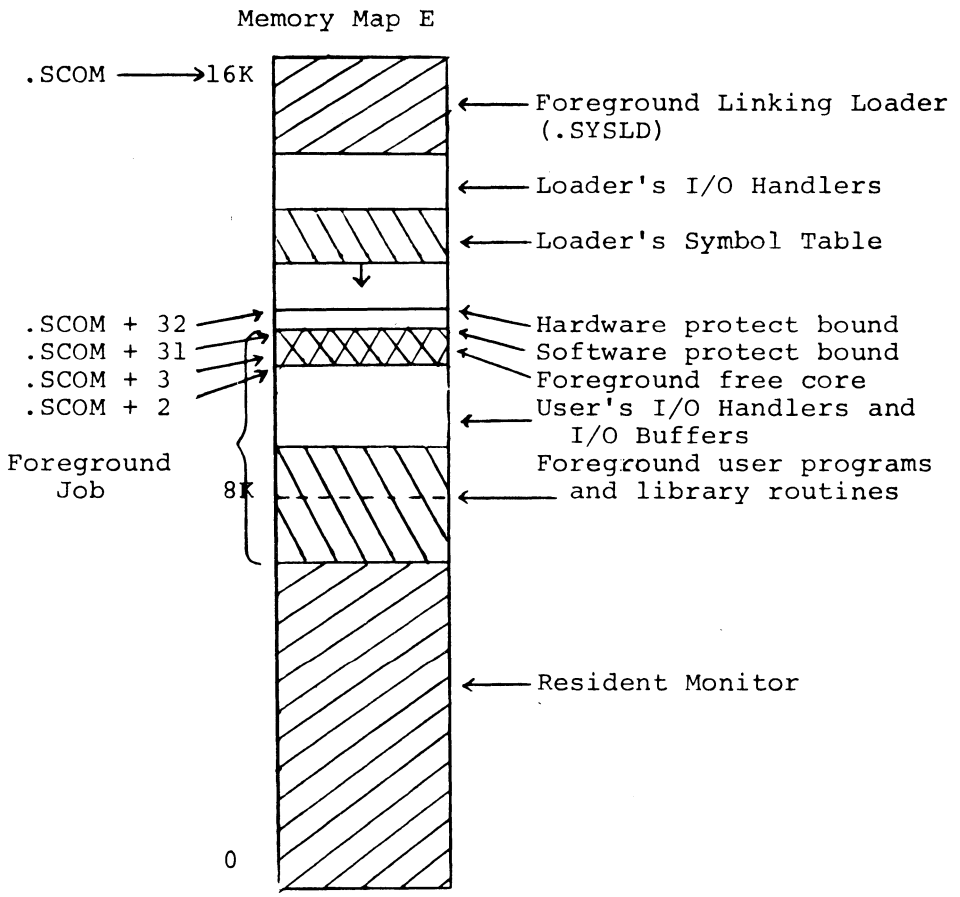
The Resident Monitor loads the Non-resident Monitor (via the resident system device handler) into upper core, overlaying the System Bootstrap. Within itself the Resident Monitor contains a simpler copy of the bootstrap which is used whenever the Resident Monitor is to be reloaded. The bootstrap restart address is location 111_8.

Memory Map D



To load a user FOREGROUND program, the Non-resident Monitor brings in the Foreground Linking Loader (.SYSLD), overlaying itself.

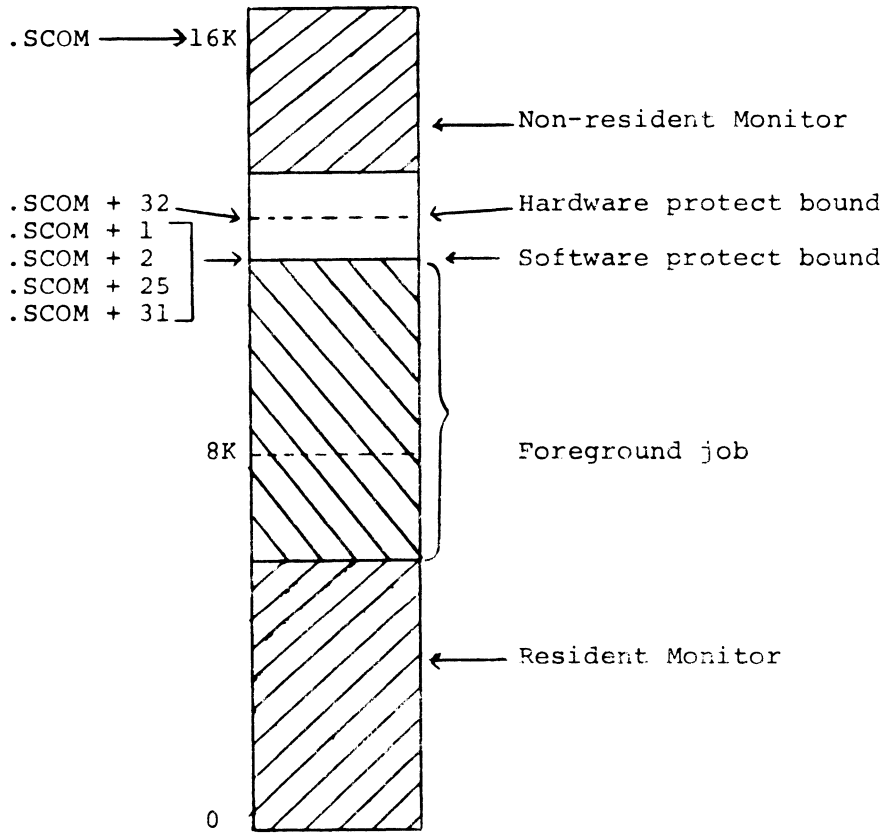
Memory Map E



The Foreground Linking Loader first brings in any additional I/O handlers required for loading. Then it loads the user program(s), library routines, user I/O handlers and I/O buffers, and allocates Foreground free core. The software memory protect bound is established just above the Foreground job. The hardware memory protect bound, because it can be set only in increments of 256 decimal, will leave some unused space between it and the Foreground job. The software protect bound allows this space to be used for dynamic data storage by the Background job. On the PDP-9 the memory protect bound can only be set at 1024 (10) word intervals, so the bank mode system sets the bound at 1024 word increments, not 256, even on a PDP-15.

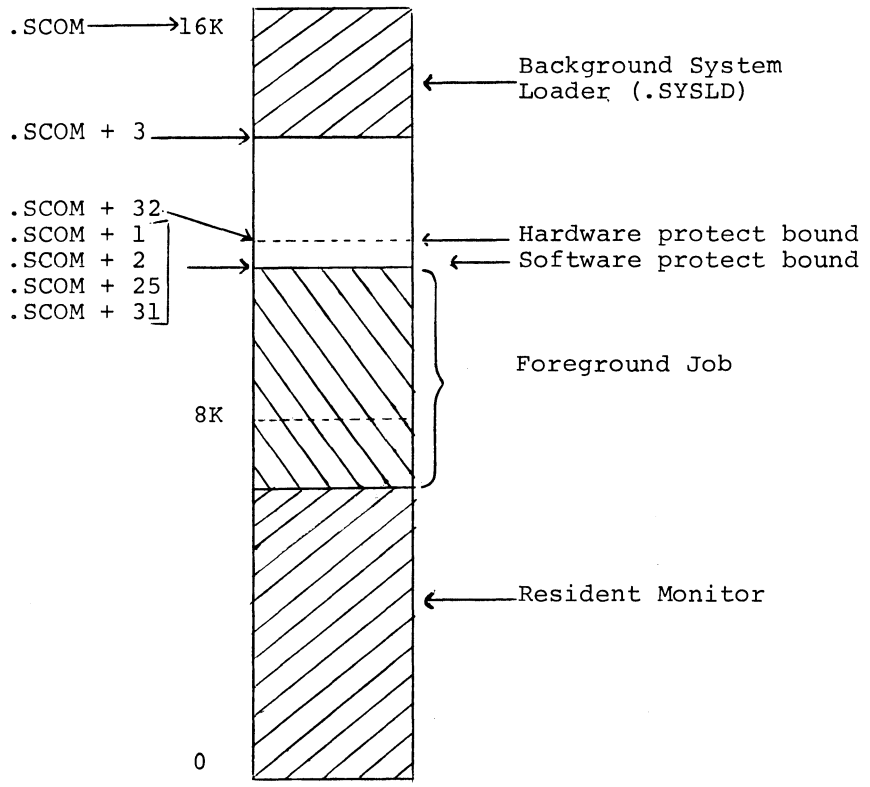
For a description of loading of Foreground XCT files, see Memory Map L.

Memory Map F



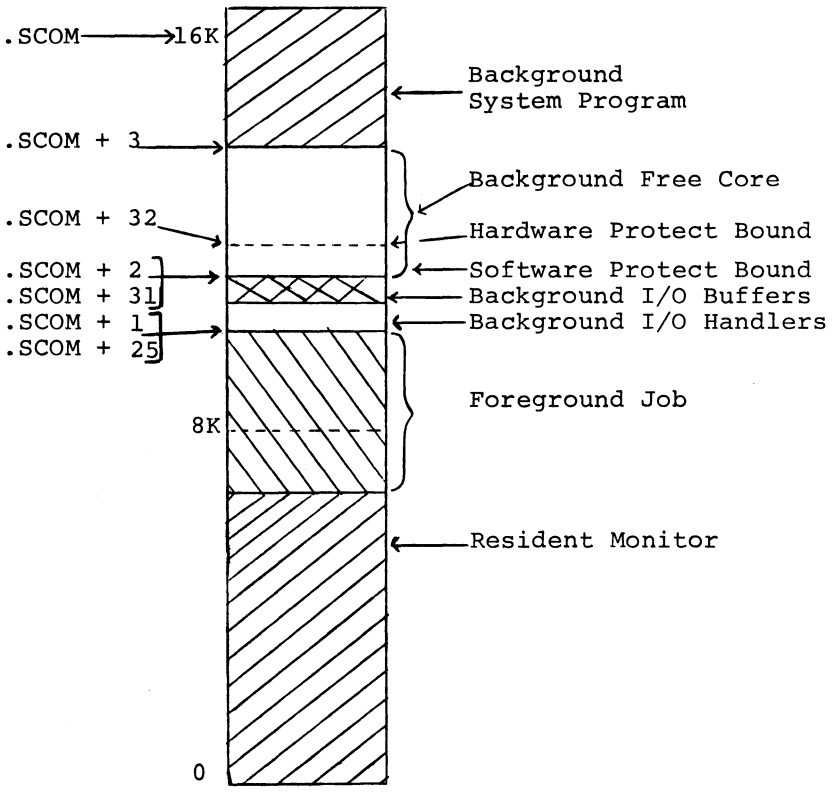
When the FOREGROUND job becomes I/O bound, control is transferred to the BACKGROUND job. The Resident Monitor loads the Non-resident Monitor (via the resident system device handler) into upper core. It then gives control to the Keyboard Listener (within the Non-resident Monitor) to await a BACKGROUND keyboard command. Memory protect is enabled while the Background job is running.

Memory Map G

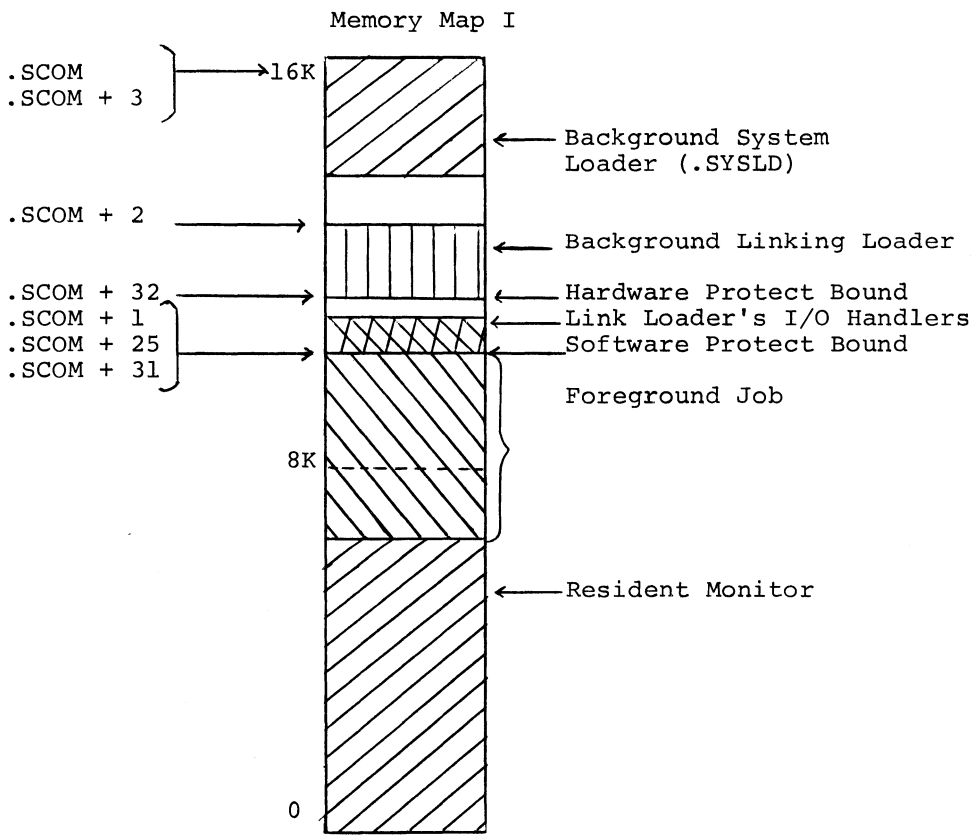


When a BACKGROUND keyboard command requests loading of a system or user program, the Non-resident Monitor brings in the System Loader, overlaying itself. Note that the BACKGROUND System Loader and the FOREGROUND Linking Loader are physically the same program, except that SYSBLK is also read into core when the BACKGROUND system program to be loaded is other than the Linking Loader or Execute.

Memory Map H



If the BACKGROUND request is for a system program, the System Loader loads the system program I/O handlers up from the top of the FOREGROUND job, allocates I/O buffer space, and loads the system program at the top of core (overlaying the System Loader). Control is returned to the Resident Monitor, which sets the memory protect bound above the buffer space and gives control to the system program.

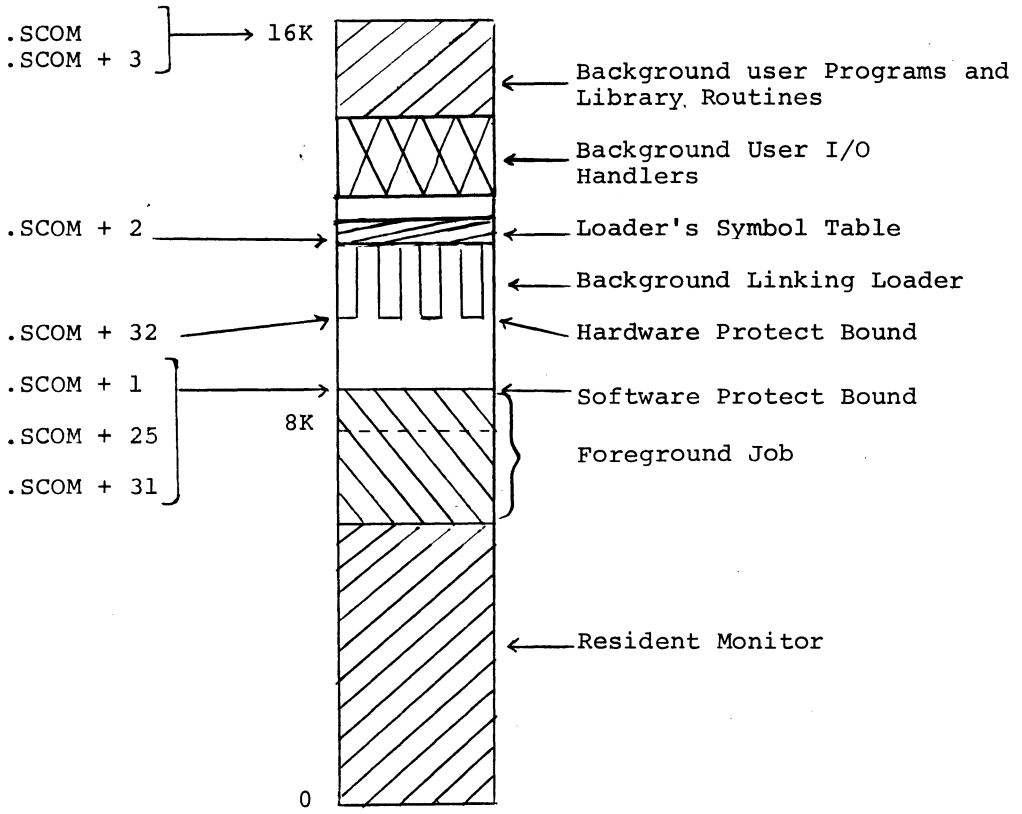


If the BACKGROUND program is a user program[1], the System Loader loads the Linking Loader I/O handlers up from the top of the FOREGROUND job and loads the Linking Loader such that the memory protect bound can be set just below it.

---

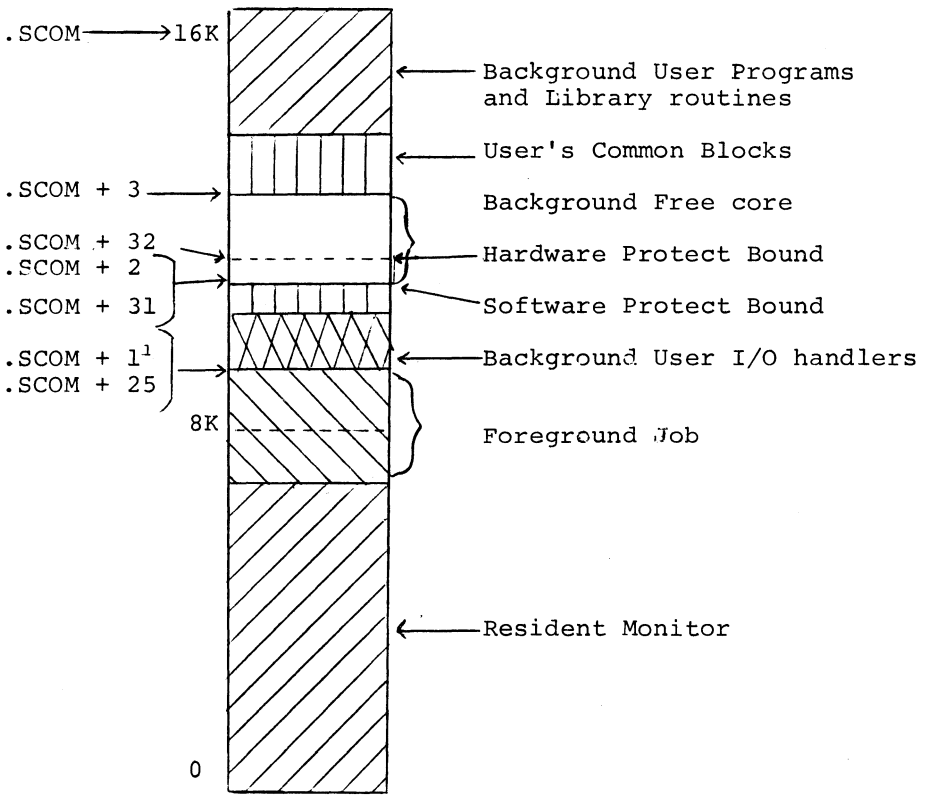
[1]User programs may be loaded along with the system program DDT.

Memory Map J



The BACKGROUND Linking Loader overlays the System Loader by loading user programs down from the top of core. User I/O handlers, presuming that they cannot fit in core between the FOREGROUND job and the bottom of the Loader, are loaded into upper core but relocated to run just above the FOREGROUND job so that the memory protect bound can be set above them. Common blocks and I/O buffers are not shown in this memory map.

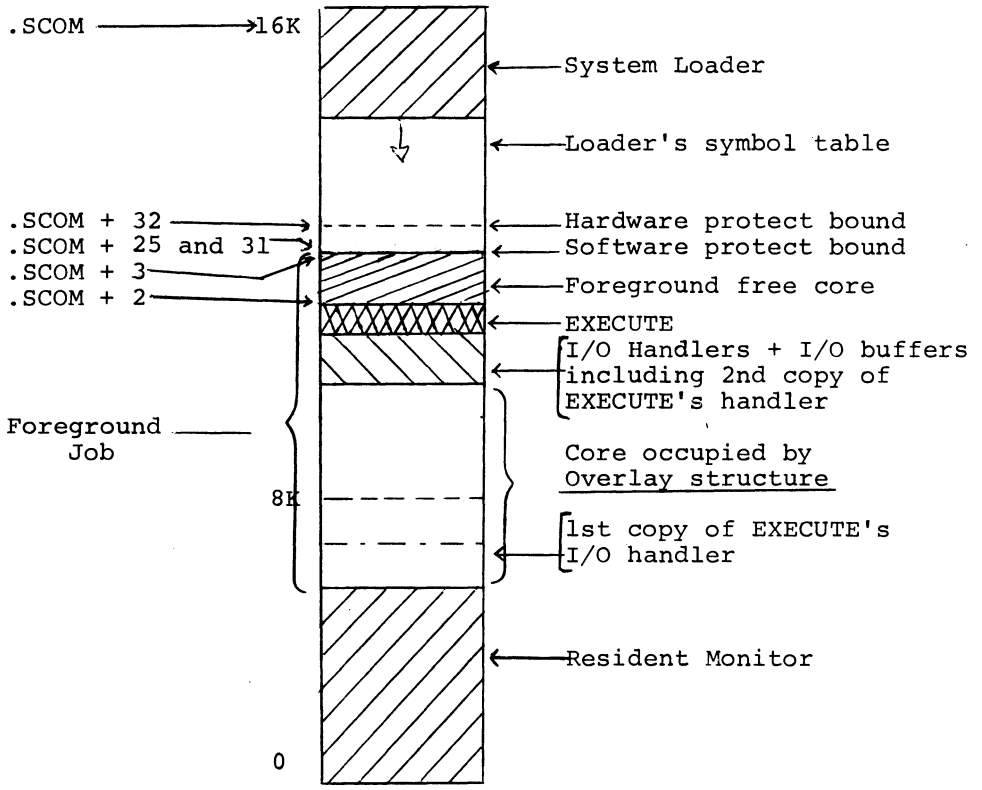
Memory Map K



The .EXIT from the Linking Loader causes the user program I/O handlers to be block transferred to their running position, the memory protect bound to be set just above the I/O buffer space, and control given to the user program. If DDT was also loaded, it resides at the top of core, above the user programs. Its symbol table, built by the Loader, is block transferred by the Monitor to start at the software protect bound.

[1]If DDT is loaded, .SCOM + 1 will be set to point at the start of the DDT symbol table.

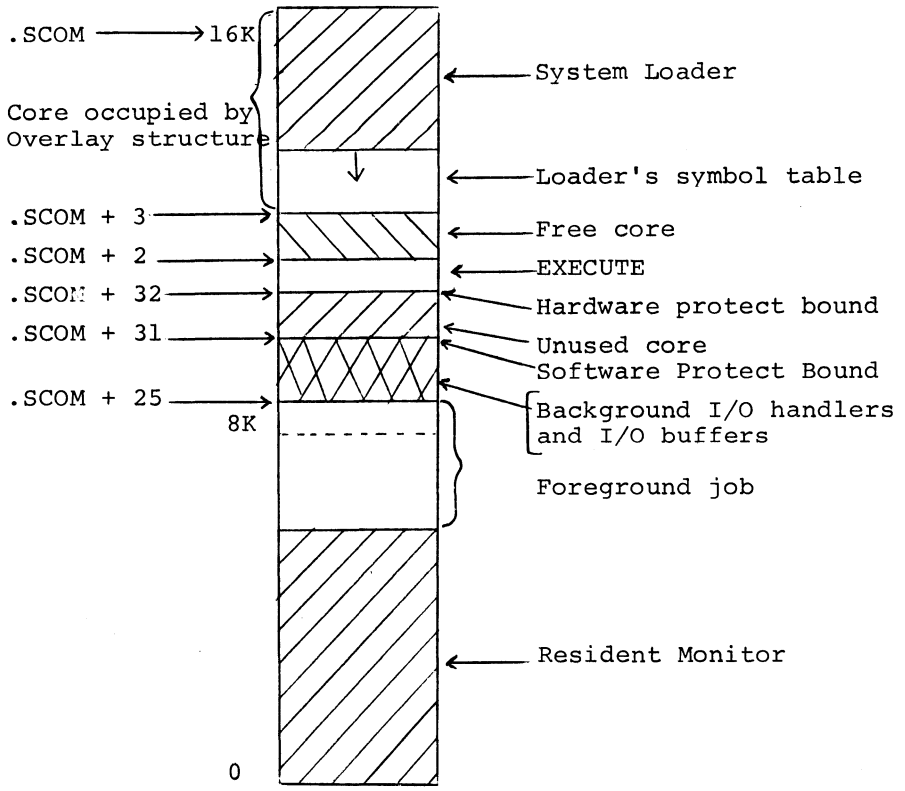
Memory Map L



EXECUTE in the Foreground:

The System Loader first loads EXECUTE's I/O handler (if not the resident handler) in order to read the XCT file. The core limits of the overlay structure are read from the file as well as the request for I/O from its .IODEV bit map. The requested handlers, including a second copy of EXECUTE's handler, are loaded above the core area to be occupied by the overlay structure. Then I/O buffers are created, if necessary, and EXECUTE is loaded above them. Finally, Foreground free core, the software protect bound, and the hardware protect bound are established.

Memory Map M



EXECUTE in the Background:

The System Loader loads EXECUTE's I/O handler (if not in core) in order to read the XCT file. The core limits of the overlay structure and the I/O requests in the .IODEV bit map are read from the XCT file. The user's I/O handlers and I/O buffers are then loaded above EXECUTE's handler, and the hardware protect bound is established above them. EXECUTE is loaded above the bound and Background free core is set up from the top of EXECUTE to the bottom of the overlay area.

SECTION 5

EXAMPLES OF BACKGROUND/FOREGROUND OPERATIONS

5.1 INTRODUCTION

The initial system startup procedure and three examples of operating within the Background/Foreground environment are described in this Section. The procedure and examples are intended to get the programmer "on the air" and to demonstrate loading programs in the Foreground.

5.2 STARTUP PROCEDURES

During initial system startup, the user normally loads the master system supplied (on DECtape) and utilizing system program BFSGEN generates a "working system". The user may run using the master system, but it is usually more desirable to generate a working system which is optimized to meet the user's needs and particular equipment configuration.

5.2.1 Loading Master B/F Monitor System

The master system for both the DECtape and DECdisk B/F systems is supplied on DECtape. To load the master system into a PDP-15/30 (DECtape system):

1. Mount the master DECtape onto a transport (TU-55 or -56) and set its unit number to that of the system device; that is, Ø on a TU-56, 8 on a TU-55.
2. Load the paper tape Bootstrap; B/F V3A uses the multi-core bootstrap.
3. Set the console address switches as follows:

If you have a -	Set Switches to -
16K system	37637
24K system	57637
32K system	77637

4. Check to ensure that the MEMORY PROTECT/RELOCATE switch is in the PROTECT position. This switch is located at the rear of the memory protect cabinet. P-mode and R-mode indicator lights are mounted on a panel located at the top front of the cabinet. The PDP-9 does not have a MEMORY PROTECT/RELOCATE switch. A PDP-9 equipped with the memory protect feature will always be in the protect position.

5. Press and release, in sequence, the console STOP, RESET, and READIN switches.

When loaded, the Monitor identifies itself and indicates its readiness by outputting the following message on the Foreground control Teletype (normally unit 1):

```
FKM15 V3A
$
```

To load the master system into a PDP-15/40 DECdisk system:

1. Mount the master disk system DECtape onto a transport and set its unit number to that of the system device; that is, Ø on a TU-56; 8 on a TU-55.

2. Load the RFSAV paper tape (supplied with the system) into the paper tape reader.

3. Set the console address switches to 1772Ø.

4. Set the DECdisk WRITE LOCKOUT switches for disk unit Ø to the WRITE ENABLE position.

5. Press and release, in sequence, the console STOP, RESET, and READIN switches. When loaded, the RFSAV program outputs the following message:

```
RFSAV V2A
SET: ACSØ= Ø DECTAPE TO DISK (LOAD)
     ACSØ= 1 DISK TO DECTAPE (SAVE)
     ACS15-17= UNIT#Ø,1,2,3,4,5,6,7
```

6. Set all console AC switches to the Ø position.

7. Press and release the console CONTINUE switch. This action causes the disk system contained by the DECtape on unit Ø to be copied onto disk unit Ø.

8. Set the DECdisk WRITE LOCKOUT switches to the WRITE DISABLE position.

9. Load the disk multicore bootstrap, RF15BT, into the paper tape reader.

10. Set the console address switches as follows:

If you Have a -	Set Switches to -
16K system	37637
24K system	57637
32K system	77637

11. Press and release, in sequence, the console STOP, RESET, and READIN switches. When loaded, the Monitor identifies itself and indicates its readiness by outputting the following message on the Foreground control Teletype:

```
FKM15 V3A
$
```

5.2.2 System Generation

A step-by-step procedure for the generation of a working system from a master system is given in Section 8 of this manual.

5.3 EXAMPLES

Three example procedures are described in paragraphs 5.3.1, 5.3.2, and 5.3.3. These procedures are used to demonstrate the loading of IDLE, single-user FOCAL, and two-user FOCAL in the Foreground.

The following conventions are used for the examples given:

1. All user inputs are underlined.

2. Readiness to accept commands is indicated by the symbol $ for the Monitor and the symbols > and * for system programs.

3. The entry of an ALTMODE character is indicated by the symbol (ALT).

5.3.1 IDLE Loaded as the Foreground Job

An Idle job is loaded in the Foreground to allow immediate use of the Background. Refer to section 6.4 for a discussion of the .IDLE system macro.

```
FKM15 V3A
$A  DTAØ    -4   (DECtape) /The program "IDLE" is on unit
or
$A  DKAØ    -4   (DECdisk) /Ø of the system device.
$GLOAD

FGLOAD V2A                 /The Loader is in core.
>←IDLE (ALT)               /Load "IDLE BIN".
```

When IDLE is loaded, no indication is given on the Foreground control Teletype. Control passes to the Background and the Non-resident Monitor is then loaded into core. The Monitor identifies itself on the Background control Teletype as:

```
BKM15 V3A                  /The Monitor is now ready to
$                          /accept Background commands.
```

5.3.2 Single-user FOCAL Loaded (Foreground)

The following illustrates a step-by-step procedure to load single-user FOCAL in the Foreground:

```
      FKM15 V3A
or    $A DTØ  -4   (DECtape)     /FOCAL is on unit Ø of
      $A DKØ  -4   (DECdisk)     /the system device.
      $A DT1  3,5                /Library input-output to FOCAL.
      $A DT3  7,1Ø               /User's data input-output.
      $FCORE  14ØØ               /Free core for FOCAL buffer.
      $GLOAD                     /Call loader to LOAD-and-Go.

      FGLOAD V2A                 /Loader is in core.
      >←FOCAL (S)                /Load FOCAL.
      FOCAL V9A                  /FOCAL is in core and is ready to
      *                          /accept commands.
                                 /User can begin to run FOCAL commands.
```

5.3.3 Two-user FOCAL Loaded (Foreground)

```
      FKM15 V3A
or    $A DTØ  -4   (DECtape)     /FOCAL is on unit Ø of
      $A DKØ  -4   (DECdisk)     /the system device.
      $A  TT1  1                 /Teletype for User #1.
      $A  DT1  2                 /Library input-output for User #1.
      $A  TT2  3                 /Teletype for User #2.
      $A  DT2  4                 /Library input-output for User #2.
      $FCORE  3ØØØ               /Assign 14ØØ  (octal) locations
      $GLOAD                     /for each user.
                                 /Call Loader to LOAD-and-Go.

      FGLOAD V2A                 /Loader is in core.
      >←FOCAL2 (S)               /Load two-user FOCAL'
      FOCAL V9A                  /FOCAL is in core and will identify
      *                          /itself on each user's Teletype.
                                 /User can begin to run FOCAL programs.
```

NOTE

Two-user FOCAL is not available on the bank mode B/F V3B system.

SECTION 6

BACKGROUND/FOREGROUND MONITOR COMMANDS (SYSTEM MACROS)

6.1 INTRODUCTION

The system MACROS unique to the Background/Foreground Monitor are listed and described briefly in Table 6-1. The Monitor Macros listed below are available in addition to those provided in the PDP-15/2Ø Monitor System for use in programs that are to be run in the Background/Foreground environment. Detailed descriptions of the macros are given in the remainder of this Section.

The .INIT macro has been altered for Background/Foreground to handle the CTRL P restart address in a manner different from the Advanced Monitor. Refer to Section 3 for an explanation.

TABLE 6-1

Background/Foreground System Macros

Name	Purpose
.REALR	Real-time transfer of data from I/O device to line buffer (real-time READ).
.REALW	Real-time transfer of data from line buffer to I/O device (real-time WRITE).
.IDLE	Allows Foreground job to indicate that control can be given to lower levels of the Foreground job or to the Background job until completion of any Foreground real-time transfer or clock interval.
.IDLEC	Allows Foreground Mainstream to give control to Background job with Foreground continuing after the .IDLEC on completion of any Foreground real-time transfer or clock interval.
.TIMER	Calls and uses real-time clock and allows priority level to be established.
.RLXIT	Accomplishes the exit from all real-time subroutines that were entered via .REALR, .REALW, .TIMER, or real-time CTRL P[1] requests.

6.2 .REALR

FORM: .REALR A, M, L, W, ADDR, P

VARIABLES:

A = .DAT slot number (octal radix)

M[2] = Data Mode
- Ø = IOPS binary
- 1 = Image binary
- 2 = IOPS ASCII
- 3 = Image Alphanumeric
- 4 = Dump Mode

L = 15-bit buffer address (octal radix)

---

[1] See Section 3.7.

[2] Data modes 5, 6, and 7 are passed to all I/O handlers.

W = Line buffer word count (decimal radix), including the two-word header

ADDR[1] = 15-bit address of closed subroutine that is given control when the request made by the .REALR is completed.

P = API priority level at which to go to ADDR

P	Priority Level
Ø	Mainstream
4	Level of .REALR
5	API software level 5
6	API software level 6
7	API software level 7

EXPANSION:

```
LOC     CAL+1ØØØØ+M6-8+A9-17
LOC+1   1Ø
LOC+2   L
        .DEC            /Decimal Radix
LOC+3   -W
        .OCT            /Octal Radix
LOC+4   ADDR+PØ-2
```

DESCRIPTION: The .REALR command is used to transfer the next line of data from the device assigned to .DAT slot A to the line buffer in the user's program. In this operation, M defines the mode of the data to be transferred, L is the address of the line buffer (including the two-word header), and ADDR is the address of a closed subroutine which should be constructed as shown in the following example.

EXAMPLE 1: STRUCTURE OF A REAL-TIME SUBROUTINE

```
ADDR    Ø                           /Entry point

        DAC         SAVEAC          /SAVE AC and all other
        .                           /live registers used.
        .                           /Any system Macro may be
        .                           /issued at this point.
        .
        LAC         SAVEAC          /Restore AC and all other
                                    /registers saved.
        .RLXIT      ADDR            /Return to interrupted
                                    /point via Monitor CAL.
```

6.3 .REALW

FORM: .REALW A, M, L, W, ADDR, P

VARIABLES: A = .DAT slot number (octal radix)

---

[1]The subroutine specified by a .REALR, .REALW, .TIMER, or real-time CTRL P should not be used at more than one priority level. The subroutine is entered via a JMS and normally cannot be protected against re-entry.

M[1]= Data Mode
- Ø = IOPS binary
- 1 = Image binary
- 2 = IOPS ASCII
- 3 = Image Alphanumeric
- 4 = Dump Mode

L = 15-bit Line buffer address (octal radix).

W = Line buffer word count (decimal radix), including the two-word header

ADDR[2]= 15-bit address of closed subroutine that is given control when the request made by the .REALW is completed.

P = API priority level at which to go to ADDR

P	Priority Level
Ø	Mainstream
4	Level of .REALW
5	API software level 5
6	API software level 6
7	API software level 7

EXPANSION:

```
LOC     CAL+1ØØØØ+M6-8+A9-17
LOC+1   11
LOC+2   L
        .DEC        /Decimal Radix
LOC+3   -W
        .OCT        /Octal Radix
LOC+4   ADDR+PØ-2
```

DESCRIPTION: The .REALW command is used to transfer the next line of data from the line buffer in the user's program to the device assigned to .DAT slot A. In this operation, M defines the mode of the data to be transferred, L is the address of the line buffer, W is the count of the number of words in the line buffer (including the two-word header), and ADDR is the address of a closed subroutine which should be constructed as shown in EXAMPLE 1 above.

6.4 .IDLE

FORM: .IDLE

EXPANSION:

```
LOC     CAL
LOC+1   17
```

DESCRIPTION: The Foreground job in a Background/Foreground environment can indicate that it wishes to relinquish control to lower levels of the Foreground job or to the Background job by executing this command. This is useful when the Foreground job is waiting for the completion of real-time I/O from any one of a number of I/O requests that it has initiated or for completion of .TIMER requests.

The .IDLE is the logical end of the current level's processing;

---

[1]Data modes 5, 6, and 7 are passed to all I/O handlers.

[2]The subroutine specified by a .REALR, .REALW, .TIMER, or real-time CTRL P should not be used at more than one priority level. The subroutine is entered via a JMS and normally cannot be protected against re-entry.

that is, control never returns to LOC+2. If the .IDLE is issued at a Foreground API software level, it effects a debreak (DBR) from that level so that pending real-time routines at that level will not be executed until the level is requested again. If the .IDLE is issued at Foreground Mainstream, control goes to the Background job. If the .IDLE is issued at Background Mainstream, control is returned to the .IDLE CAL.

6.5 .IDLEC

FORM: .IDLEC

EXPANSION:

```
LOC     CAL+1ØØØ
LOC+1   17
```

DESCRIPTION: .IDLEC is identical to .IDLE except when issued at the Foreground Mainstream level. In this case, control goes to the Background job, and LOC+2 is saved as the Foreground Mainstream return pointer. The next time control returns to Foreground (at any priority level), Foreground Mainstream processing will resume at LOC+2 when Mainstream becomes the highest active Foreground level.

6.6 .TIMER

FORM: .TIMER N, ADDR, P

VARIABLES:

N[1] = Number of clock increments (decimal radix)

ADDR[2] = 15-bit address of closed real-time subroutine to handle interrupt at end of interval

P = API priority level at which to go to ADDR

P	Priority Level
Ø	Mainstream
4	Level of .TIMER
5	API software level 5
6	API software level 6
7	API software level 7

EXPANSION:

```
LOC       CAL[3]
LOC+1     14
LOC+2     ADDR+PØ-2
          .DEC        /Decimal Radix
LOC+3     -N
```

DESCRIPTION: .TIMER is used to set the real-time clock to N increments and to start it. Each clock increment represents 1/60 second for 60 Hz systems and 1/50 second for 50 Hz systems. When the Monitor services the clock interrupt, it passes control to location ADDR+1 with the priority level set to P. The coding at ADDR should be in closed subroutine form, as in EXAMPLE 1.

---

[1]To transfer control to subroutine ADDR at priority level P immediately, N should be set equal to zero.

[2]The subroutine specified should not be used at more than one priority level. The subroutine is entered via a JMS and normally cannot be protected against re-entry.

[3]When bit 8 of CAL is set to 1, an abort .TIMER is effected. All intervals having the same address and priority level (LOC+2) will be aborted.

6.7 .RLXIT

FORM: .RLXIT ADDR

VARIABLES: ADDR = 12-bit[1] entry point address of the real-time subroutine from which an exit is to be made.

EXPANSION:

```
LOC     CAL   ADDR
LOC+1   2Ø
```

DESCRIPTION: .RLXIT is used to exit from all real-time subroutines that were entered via .REALR, .REALW, .TIMER, or real-time CTRL P requests. The instruction just preceding the .RLXIT call should restore the AC with the value of the AC on entrance to this subroutine. .RLXIT will restore the link from bit Ø and page/bank mode from bit 1 of the contents of ADDR.

.RLXIT protects against re-entrance to Background or Foreground Mainstream real-time subroutines. When the contents of ADDR is non-zero, the subroutine is assumed active; .RLXIT sets the contents of ADDR to Ø, thus making it available again. Note: Real-time subroutines should initially have their entry point register set to Ø; and restart procedures, entered via CTRL P or after CTRL T, should reset all entry points to Ø.

6.8 MAINSTREAM REAL-TIME SUBROUTINES

Mainstream real-time subroutines in the Foreground are not equivalent to those in the Background due to the manner in which I/O busy situations are handled. If the Background becomes I/O busy, the Monitor "sits on" the Background CAL instruction (while Background is in control) until it can be processed. Therefore, Background Mainstream real-time routines can be executed despite the fact that Background Mainstream is I/O busy. If Foreground Mainstream is I/O busy, Foreground Mainstream real-time routines cannot be executed until the busy situation is terminated. This is due to the fact that control is given to the Background whenever Foreground Mainstream becomes I/O busy. The device handler responsible for the busy situation is remembered in the Foreground Mainstream busy flag. Mainstream real-time routines cannot then be run because they too could become busy.

This situation can be avoided either by using .REALR or .REALW in conjunction with .IDLE or .IDLEC, or by using .WAITR to prevent Foreground Mainstream from becoming I/O bound.

6.9 API SOFTWARE LEVELS -- PROGRAMMING NOTE

On configurations that have API, elements of the Foreground job may run at four

[1]The Resident Monitor, which operates in bank addressing mode, uses .RLXIT with a 13-bit entry point address. In the bank mode system, all addresses have 13-bit values.

priority levels (levels 5, 6, and 7 of the API and Mainstream). It is important to understand that as Foreground becomes I/O busy at a given level, the Monitor drops to the Foreground's next highest active level.

The lower level may be dependent upon the completion of the I/O that caused the higher level to become busy. The following coding method is incorrect because the lower level will receive control as a result of the I/O not being done.

Level 5 Subroutine

```
.READ n,2,BUFFER,52
.WAIT n
```

When the Monitor processes the .READ and encounters the unsatisfied .WAIT, it recognizes this as an I/O busy situation on level 5 and drops control to the next lower active level. Suppose at level 7 there is a user subroutine dependent upon the contents of BUFFER.

Level 7 Subroutine

```
.WRITE x,2,BUFFER,52
```

In the above case, the .WRITE will be executed independent of whether the level 5 I/O call to fill BUFFER has been completed.

Two proper coding methods would be:

(1) to perform the .WRITE within the level 5 subroutine after the .WAIT n;

(2) to use a .REALR at level 5 which would specify the level 7 subroutine to be called upon completion of the .REALR. This would eliminate the need for .WAIT n in the level 5 subroutine.

SECTION 7

WRITING DEVICE HANDLERS FOR THE PDP-15 BACKGROUND/FOREGROUND MONITOR SYSTEM

> WARNING:
>
> I/O device handlers and service routines written according to this section will operate on a PDP-15 in page mode or, with the modifications noted, on a PDP-9 or PDP-15 in bank mode. For further information, read section 7.13.

7.1 INTRODUCTION

The reader is assumed to be acquainted with the concept of an I/O device handler from experience using the Keyboard Monitor system. I/O handlers are a convenience because they interface to user programs by accepting a small set of standard commands (Monitor calls), e.g., .READ and .WRITE. Within reason, programs can be written to function without regard to specific I/O devices. They refer to logical devices (.DAT slots) and the assignment of real devices is made at program load time. Device handlers, because they interface with the Monitor, must conform to certain established conventions (which differ from those in the Keyboard Monitor environment) and are more difficult to write and to understand than stand-alone I/O service routines.

An I/O service routine[1], unlike a device handler, is coded into the user program or is loaded as a user subprogram. It interfaces directly with the user program and does not use system macros (.READ, etc.), does not use .DAT slots, and is not loaded from the system's I/O library. Such a routine cannot normally[2] operate in the Background because it employs IOT instructions.

7.2 I/O SERVICE ROUTINE

The coding of an I/O service routine is most easily explained by example. Consider a device which consists of two pushbuttons. Each sets a hardware flag which can be tested by skip IOT, and either flag being set requests a hardware interrupt. The device has the following IOT instructions:

---

[1]The term I/O service routine is used in this section to distinguish a simple, direct interface user I/O routine from a standard, full-blown I/O device handler.

[2]Refer to the MPOFF command in Section 2.5.12.

```
PBSF1          /Skip if button 1 flag is set.
PBCF1          /Clear button 1 flag.
PBSF2          /Skip if button 2 flag is set.
PBCF2          /Clear Button 2 flag.
```

If the device were connected to the API assume that it would interrupt at API level 3 and via API channel 2Ø (Register 6Ø). If the device were connected to the PIC it would interrupt at API level 3[2] and via Register Ø (as all PI devices do).

At system generation time one would have to add this as a new device to the system. The following illustrates the conversation with the System Generator (read Section 8):

API CASE:

```
MORE I/O? Y
DEVICE NAME > PB
NUMBER OF INTERRUPTS SETUP > 1
API ? Y
SKIP IOT > 7Ø66Ø1
API CHNL > 2Ø
```

PI CASE:

```
MORE I/O ? Y
DEVICE NAME > PB
NUMBER OF INTERRUPTS SETUP > 2
API ? N
SKIP IOT > 7Ø66Ø1
MNEMONIC > PBSF1
SKIP IOT > 7Ø6621
MNEMONIC > PBSF2
```

In the API case, note that both device flags interrupt via the same API channel; hence, only one .SETUP call is needed.

Since PB is added as a new device, the System Generator assumes the existence of a "PBA" handler. To be safe, change the handler to "PBW" so that this non-existent handler is not inadvertently assigned to some .DAT slot.

```
.TITLE FOREGROUND JOB

/THE PUSHBUTTON SERVICE ROUTINE COULD BE A SEPARATELY LOADED SUBPROGRAM;
/HOWEVER, HERE IT IS SHOWN AS IN-LINE CODE WITHIN A LARGER PROGRAM.
/IN THE NORMAL MODE OF SYSTEM OPERATION THIS CODE IS ILLEGAL IN THE
/BACKGROUND BECAUSE IT USES IOT INSTRUCTIONS[2]. SINCE THE MONITOR HAS NO
```

[1] True only of the PDP-15.
[2] See Section 2.5.12.

```
/CONNECTION TO THIS SERVICE ROUTINE, THERE IS NO WAY TO GUARANTEE THAT THIS
/DEVICE HAS STOPPED I/O BEFORE RELOADING THE MONITOR[1], E.G., FOLLOWING CTRL C.

BEGIN .                                     /THIS IS MAIN PROGRAM CODE AND
      .                                     /NEED HAVE NOTHING TO DO WITH
      .                                     /THE "PB" SERVICE ROUTINE.
      .

/"PB" (PUSHBUTTON) SERVICE ROUTINE.  THE FOLLOWING IS ONCE-ONLY
/INITIALIZATION CODE.  THESE LOCATIONS MAY BE USED LATER ON FOR TEMPORARY
/STORAGE (AS SHOWN).

ACØ     LAC*      (.SCOM+55           /ADDRESS OF THE MONITOR'S
.SETUP  DAC       .SETUP              /.SETUP ROUTINE.
TEMP1   LAC*      (.SCOM+51           /ADDRESS OF THE MONITOR'S
REALTP  DAC       REALTP              /REALTP ROUTINE.

/RAISE TO API LEVEL 4 FROM THE MAINSTREAM LEVEL.  THE MONITOR'S .SETUP
/ROUTINE IS CALLED FROM THE CAL LEVEL AND IS NOT REENTRANT CODE.

AC1     LAC       (4ØØØ1Ø
AC2     ISA

/CALL THE MONITOR'S .SETUP ROUTINE TO LINK HARDWARE INTERRUPTS FROM
/THE DEVICE TO THE SERVICE ROUTINE NOW THAT IT IS IN CORE[2].  AT SYSTEM
/GENERATION TIME, IT IS ASSUMED, BFSGEN RESERVED API CHANNEL 2Ø
/(REGISTER 6Ø) FOR THIS DEVICE BY PLACING THERE A "JMS* (ERROR"
/INSTRUCTION AND ASSOCIATING IT WITH THE SKIP IOT "PBSF1".  THE .SETUP
/ROUTINE WILL CHANGE THE INSTRUCTION TO "JMS* (PBINT".

        JMS*   .SETUP                   /CALL .SETUP WITH 2 ARGUMENTS:
        PBSF1                           /THE SKIP IOT AND THE ADDRESS
        PBINT                           /OF THE INTERRUPT SERVICE ROUTINE.

/IF THIS DEVICE IS ON PI, A SECOND .SETUP CALL IS NECESSARY BECAUSE
/THERE WILL BE TWO SKIP IOT'S IN THE SKIP CHAIN.  FOR PI DEVICES,
/THE ENTRY INSTRUCTIONS ARE "JMP* (PBINT".

        JMS*    .SETUP                  /CALL .SETUP WITH 2 ARGUMENTS:
        PBSF2                           /SKIP IOT AND THE ADDRESS
        PBINT                           /OF THE INTERRUPT SERVICE ROUTINE.

/DEBREAK FROM LEVEL 4 BACK TO MAINSTREAM.

        DBK

/END OF ONCE-ONLY CODE.

/MAIN PROGRAM PROCESSING MAY NOW CONTINUE UNTIL IT IS INTERRUPTED BY ONE
/OF THE PUSHBUTTON FLAGS.
        .
        .
        .
        .

/THE FOLLOWING IS THE INTERRUPT SERVICE ROUTINE FOR THE PUSHBUTTONS.  IT
/IS ENTERED AT API LEVEL 3.

/IN THE CASE WHERE THIS DEVICE IS ON API, THIS ROUTINE IS ENTERED VIA A
/JMS INSTRUCTION.  THE STATE OF THE PROGRAM INTERRUPT CONTROL (ION OR IOF)
/WILL NOT BE ALTERED.
```

---

[1] See Section 7.8.

[2] See Section 8.3.4, Note 3.

```
PBINT       Ø                           /LINK + PAGE/BANK + MEM.PROT. + PC.
            DBA[1]                      /ENTER PAGE MODE.
            DAC       ACØ               /SAVE THE ACCUMULATOR.

/IF, INSTEAD, THE DEVICE IS CONNECTED TO THE PIC, THE ROUTINE IS ENTERED
/BY A JMP INSTRUCTION AND THE FOLLOWING CODE SHOULD BE SUBSTITUTED FOR
/THE ABOVE.  THE PIC IS OFF (IOF).

PBINT       DBA                         /ENTER PAGE MODE.
            DAC       ACØ               /SAVE THE ACCUMULATOR.
            LAC*      (Ø                /SAVE THE INTERRUPT POINT:
            DAC       PC                /LINK + PAGE/BANK + MEM.PROT. + PC.
            DZM*      (Ø                /NECESSARY ON THE PDP-9; GOOD
            ION[2]                      /PRACTICE ON THE PDP-15.

/FROM HERE ON, THE CODE IS COMMON TO BOTH API AND PIC DEVICES.

            PBSF1                       /SKIP IF BUTTON 1 FLAG SET.
            JMP       PB2               /NO. MUST BE BUTTON 2.
            PBCF1                       /CLEAR BUTTON 1 FLAG.

/BUTTON 1 IS INTERPRETED TO MEAN:  REQUEST REAL-TIME SUBROUTINE "SUBR1"
/AT API SOFTWARE PRIORITY LEVEL 5.

            LAC       (SUBR1+5ØØØØØ
            JMP       RUN.IT

/BUTTON 2 MEANS:  REQUEST "SUBR2" AT API LEVEL 6.

PB2         PBCF2                       /CLEAR BUTTON 2 FLAG.
            LAC       (SUBR2+6ØØØØØ

/CALL THE MONITOR'S REALTP SUBROUTINE TO PLACE THE REAL-TIME REQUEST IN
/THE API QUEUE.  AS SOON AS THE API LEVEL AT WHICH THE SUBROUTINE IS TO RUN
/BECOMES THE HIGHEST ACTIVE LEVEL, THAT SUBROUTINE WILL BE CALLED.

RUN.IT      DAC       TEMP1
            LAC*      (.SCOM+1Ø2        /RAISE TO API LEVEL Ø OR LEVEL 1.[3]
            ISA
            LAC       TEMP1             /SUBR+API LEVEL CODE.
            JMS*      REALTP[4]
            DBK                         /TO LEVEL 3.

/NOW THAT THE PUSHBUTTON HAS BEEN SERVICED AND A SPECIFIC ACTIVITY
/(SUBR1 OR SUBR2) HAS BEEN SCHEDULED, EXIT FROM THE HARDWARE LEVEL TO
/THE API LEVEL 4 INTERRUPT HANDLER IN THE MONITOR.  LEVEL 4 IS THE SYSTEM
/DISPATCHER WHICH DECIDES WHAT IS TO BE RUN NEXT BASED UPON CONDITIONS SET
/BY HARDWARE INTERRUPT ROUTINES.
```

[1] Omit DBA instructions in background /foreground bank mode.

[2] The ION instruction may precede the clearing of the device flags on the PDP-15 so long as operation continues at API level 3. On a PDP-9, however, the ION must come after the flags have been cleared; otherwise, an immediate interrupt would occur, unless the ION was executed after a raise to API level 3.

[3] Refer to section 7.4 which explains why .SCOM+1Ø2 is used to protect common Monitor routines from reentrancy.

[4] If the highest Monitor API level is defined to be level one, according to the contents of .SCOM+1Ø2, then REALTP must not be called at API level zero. This statement holds true for all Monitor routines called at the highest Monitor level.

```
                LAC         (4Ø4ØØØ       /REQUEST AN API INTERRUPT
                ISA                       /AT SOFTWARE LEVEL 4.
                LAC         ACØ           /RESTORE THE ACCUMULATOR.
                DBR                       /DEBREAK AND RESTORE FROM LEVEL 3.

/IF THE DEVICE IS ON API, THE INTERRUPTED PC IS STORED IN "PBINT".

                JMP*        PBINT

/IF THE DEVICE IS ON THE PIC, THE INTERRUPTED PC IS STORED IN "PC".

                JMP*        PC

/WHICHEVER JMP* IS USED, IT MUST IMMEDIATELY FOLLOW THE DBR.  ONCE THE
/DBR HAS BEEN EXECUTED, THERE MUST BE NO POSSIBILITY OF INTERRUPTING BEFORE
/THE JMP* IS DONE.  ALL DECISION MAKING MUST THEREFORE PRECEDE THE DBR.

/END OF PUSHBUTTON SERVICE ROUTINE.

        .
        .
        .
/API LEVEL 5 REAL-TIME SUBROUTINE -- DOES SOMETHING AS A RESULT OF BUTTON 1
/HAVING BEEN PRESSED.

SUBR1           Ø
                DAC         AC1
                .
                .
                .
                LAC         AC1
                .RLXIT      SUBR1

/API LEVEL 6 REAL-TIME SUBROUTINE -- DOES SOMETHING AS A RESULT OF
/BUTTON 2 HAVING BEEN PRESSED.

SUBR2           Ø
                DAC         AC2
                .
                .
                .
                LAC         AC2
                .RLXIT      SUBR2
                .
                .
                .
```

This is a very simple I/O service routine. It does not perform data manipulation and does not issue any IOT's that could cause further interrupts.

7.3 I/O DEVICE HANDLER

A device handler written to operate in the Background/Foreground Monitor environment must conform to the rules outlined in the remainder of this

section. Handlers differ from I/O service routines in the following ways:

1. They interface to user programs via Monitor calls, e.g., .READ.
2. Because they are referenced by .DAT slot number, they can be used by device independent programs.
3. Except for TTA. and the resident system device handler, all handlers are part of the system's I/O library.
4. Handlers can be used by the Background job when there is no conflict with Foreground needs.
5. I/O handlers all have STOPIO routines which allow the Monitor to shut down I/O in an orderly fashion. This is absolutely necessary when a handler is to be used by the Background job.

7.3.1 Types of Device Handlers

There are three types of I/O device handlers that can operate within the Background/Foreground Monitor System:

1. Single user -- This handler can be used by either the Foreground job or the Background job but not both during the same core load; that is, it is dedicated to one job and the Monitor System will not permit the other job to be connected to it.
2. Sequential Multi-user -- This handler can be connected to both the Foreground and the Background job and both can utilize it on a sequential first-come-first-served basis.
3. Multi-user -- This handler can be connected to both the Foreground and the Background jobs with the Foreground job having priority on usage. If the Background job is using the handler and Foreground requires it, the Background I/O will be deferred until the Foreground I/O has been completed.

This section is primarily devoted to describing the development of single-user handlers. Thereafter, the transition to a sequential multi-user handler is described.

I/O handler type 3 (Multi-user) is not described because it is unlikely that a customer will need to write one and because the description would overly complicate this section of the manual. Should the need to write such a handler arise, it is recommended that a listing be obtained of the Multi-user DECtape (DTA.) or Disk (DKA.) handler to be used as a guide.

All device handlers, except for TTA. and the resident system device handler, are loaded to run in page mode and therefore may use indexed instructions. Where they do so, however, they must save and restore the Index Register (and

Limit Register, if used). All device handlers in the bank mode system run only in bank mode; no indexed instructions are allowed in the bank mode system.

7.3.2 General Structure of Device Handlers

User program commands to device handlers are initiated by CAL instructions, which trap to absolute location 2Ø octal in the Monitor. The CAL handler in the Monitor, operating at API level 4, transfers control to the CAL processing section of the device handler at level 4.

All devices which perform I/O should be interrupt driven[1], i.e., should rely on a hardware interrupt condition to signal I/O completion. Without an interrupt, the device would have to be polled after elapsed clock intervals (which is possible for slow devices) or tested continuously (which defeats the purpose of a real-time system). Handlers which perform I/O will, in general, have an interrupt service routine which operates at the API level of the hardware.

For some devices, all I/O must be solicited. For example, no interrupt from the papertape punch can occur until the handler, PPA., has initiated I/O as a result of some Monitor call. This fact allows the CAL and interrupt portions of PPA. to share common storage registers and common code.

Some devices, such as Teletype, generate unsolicited interrupts which can occur while the Teletype handler is processing a CAL command. Therefore, the CAL and interrupt portions of TTA. cannot use common code and common registers except where the CAL code raises to API level 3 (the hardware level for Teletypes) to prevent Teletype interrupts.

Besides CAL and interrupt processors, a handler must have subroutines for stopping I/O. The STOPIO code is called at Mainstream (all API levels inactive) but is guaranteed not to be called while CAL processing is in progress.

7.4 REENTRANCY PROTECTION

There are common routines in the Monitor which are called by all device handlers. Since these routines cannot be reentered[2], all calling programs must raise to the highest commonly used API level in order to avoid being interrupted. Normally, API level Ø is the highest commonly used level; and the instructions to raise to level Ø would be:

```
LAC     (4ØØ2ØØ
ISA
```

---

[1] The core-to-core handler, COA., is an example of a handler which is not interrupt driven; and therefore all its processing is done at the CAL level (API level 4).

[2] Interrupted, entered at a higher API level, and then resumed at the lower level.

However, some Monitor routines which operate at this highest commonly used level may take nearly 2ØØ microseconds to complete. Some devices may require faster service than this allows; hence, it is desirable to reserve API level Ø for them. The highest commonly used Monitor level would be defined to be API level 1, and devices on level Ø would only have to compete with one another. They could not, however, use common Monitor routines[1] due to the problem of reentrancy. Such routines would best be I/O service routines rather than I/O device handlers.

The highest commonly used API level (Ø or 1) is established during system generation and the value 4ØØ2ØØ (for level Ø) or 4ØØ1ØØ (for level 1) is placed in .SCOM+1Ø2. The instructions that must be executed to protect against reentrancy are:

```
LAC*     (.SCOM+1Ø2
ISA
```

For devices which operate at the highest commonly used API level, a raise has no effect. Therefore, a check for this case must be made so that the corresponding debreak (DBK) instruction is not executed. Where a DBR (debreak and restore) instruction would have been used, an RES (restore) should be executed instead. The PDP-9 does not have an RES instruction as does the PDP-15; however, there has never yet been a need for a user to replace a DBR with an RES.

As a side issue to reentrancy protection, the reader is cautioned not to share subroutines within a handler at the CAL and interrupt levels unless CAL's and interrupts cannot coincide.

7.5 DEVICE HANDLER'S CAL PROCESSOR

7.5.1 Arguments of the CAL

The first 37 (octal) words of an I/O handler must have the format described in the following pages. The CAL handler in the Monitor has been implemented to do as much of the function processing as possible. In giving control to the I/O handler, the CAL handler will have set up registers in the I/O handler with all pertinent information (arguments) of the CAL in the most accessible state, and will then transfer control to the appropriate function processor via the JMP table in the I/O handler which begins at word 20_8 relative to the first location in the handler. Since CAL is not a reentrant process, CAL instructions should not be executed while at the CAL level or at a hardware interrupt level[2].

---

[1] Such as, REALTP and IOBUSY.

[2] As a special case, the Monitor allows the MAGtape handler to do so.

WORD Ø: JMS SWAP

The SWAP subroutine is in the device handler. The JMS instruction will be simulated from within the Monitor so that the SWAP routine will return to the Monitor and not to WORDl of the handler. The SWAP subroutine must execute WORD5 which restores the state of the program interrupt[1] and DBK from level Ø or 1 of the API. The presence of this routine becomes functionally necessary for type 3 (Multi-user) handlers to accomplish swapping from Background to Foreground usage. The I/O device independence of the system requires that all handlers look alike to the outside world (namely, the Monitor's CAL handler).

WORD 1:	Ø	/Foreground Busy Register[2]
WORD 2:	Ø	/Background Busy Register[2]

For both busy registers:

Ø = Not Busy	
Non-Ø = Busy	(the CAL handler in the Monitor places the current .DAT slot number here -- full 18 bit value if negative.)

When the Monitor's CAL handler receives an I/O call, it checks the validity of the .DAT slot number for this job (Foreground or Background), checking for its existence, whether or not a device has been assigned to it, and if the appropriate handler was loaded.

The CAL handler then checks the appropriate busy register[3] and proceeds as follows:

1. If the flag indicates that the handler is already busy, the job becomes I/O bound at this level. Foreground can become I/O bound at 4 levels, which means it gives up control to lower levels or to the Background until the I/O operation is completed.
2. If the flag indicates not busy, it is set to busy[4] and the CAL handler processes the function and passes the request on to the device handler.

---

[1]This is a vestige from PDP-9 code.

[2]Must be assembled with contents = Ø. The Teletype handler is a special case.

[3]The Teletype handler is an exception.

[4]Actually, there is also a test on the CLOSE flag (Word 3 or 4) which is described on the next few pages. As a result, the function might not be passed on to the handler.

Note that .WAIT's and .WAITR's are completely processed by the CAL handler and are not passed on to the I/O handler[1].

If the corresponding busy register indicates busy:

1. For .WAIT in the Foreground, control is given to a lower Foreground level or to the Background. The .WAIT command is not reexecuted; instead, the WAIT condition is recorded for the specific Foreground level in a .SCOM register. When the I/O completes, the device handler will call the IOBUSY routine in the Monitor, which will clear the WAIT condition and prepare to resume processing at that level following the .WAIT.
2. For .WAIT in the Background, since there is no further processing that can be done, control is returned to the .WAIT.
3. For .WAITR in either the Background or Foreground, control goes to the address specified in LOC+2 (which must be above the hardware memory protect bound if in the Background)[2].

If the corresponding busy register indicates not busy, the WAIT condition has been satisfied and control is returned to LOC+2 (if .WAIT) or LOC+3 (if .WAITR).

```
WORD 3:   Ø                          /Foreground .CLOSE register[3].
WORD 4:   Ø                          /Background .CLOSE register[3].
```

For both .CLOSE registers:

Ø = .CLOSE or .OPER not in progress
Non-Ø = .CLOSE or .OPER in progress

.CLOSE and .OPER functions have a built-in WAIT condition. When the .CLOSE or .OPER is first executed, the busy register and .CLOSE register for the appropriate job contain zero. The CAL handler in the Monitor sets the return PC so that the function will be reexecuted. The busy register is set with the .DAT slot number and the .CLOSE register is set non-Ø (-1).

At completion of the .CLOSE or .OPER function, the device handler must clear only the appropriate busy register. When the function is reexecuted with the busy register cleared but the .CLOSE register set, the contents minus 1 of the .CLOSE register are returned in the AC to the calling program following the .CLOSE or .OPER command and the handler's .CLOSE flag is cleared by the

---

[1]The Teletype handler is an exception.

[2]The CAL handler validates Background arguments in Monitor calls. The test based on the setting of the hardware memory protect bound uses the contents of .SCOM+32, which is not set to zero by the $MPOFF command.

[3]Must be assembled with contents = Ø.

Monitor. The handler is not entered a second time. The contents minus 1 of the .CLOSE register are returned in the AC specifically for .OPER functions (.FSTAT, .RENAM, .DLETE). Device handlers that utilize this capability should set the appropriate .CLOSE register (WORD 3 if Foreground; WORD 4 if Background) as follows:

```
1 = File not present
INFORMATION +1 = File is present (where INFORMATION is the
                 device block number, which must not = -1)
```

Either Ø or INFORMATION is returned in the AC.

```
WORD 5:   ION                    /The CAL handler will store an ION
                                 /instruction here. This is vesti-
                                 /gial code from the PDP-9.
WORD 6:   ION[1]                 /The CAL handler will also store an
                                 /ION here.
WORD 7:[1]                       /Return Pointer. The CAL handler
                                 /places the address of the Monitor's
                                 /CALXIT routine in this register.
```

Words 1Ø through 17 are the BACKUP DATA REGISTERS. The CAL handler sets up these registers prior to entering the device handler. For multi-user handlers, a set of backup registers must be available to queue one Background I/O request when the handler is processing a Foreground request. The handler's SWAP routine is called to swap the contents of the backup registers with that of the live registers (elsewhere in the handler). For single user handlers, the SWAP routine does not perform a swap since the backup registers <u>are</u> the live registers.

```
WORD 10:  JMP FUNC               /After checking the validity of
                                 /function and subfunction codes, the
                                 /CAL handler places a JMP to the
                                 /appropriate entry in the function
                                 /JMP table (words 2Ø-32) of the I/O
                                 /handler in this register.

WORD 11:                         /The CAL handler sets this register
                                 /to indicate which job executed
                                 /this CAL:
                                 /          Ø = Foreground
                                 /          1 = Background

WORD 12:                         /.DAT slot number (18-bits if
                                 /negative). The CAL handler sets
                                 /this register.
```

[1]If it is guaranteed that the device cannot cause an interrupt while processing is at the CAL level, then the handler's CAL and interrupt processors can use common exit code as described in 7.6. If so, the interrupt service routine must store a DBR instruction in WORD6 and the interrupted PC (with Link, Page/Bank Mode and Memory Protect bits) in WORD7.

WORD 13: /Unit number for multi-unit devices
/in bits Ø-2 with bits 3-17 contain-
/ing the address of the CAL. The CAL
/handler sets this register.

The CAL handler makes a general check for validity on:

a. File type
b. Data Mode
c. MAGtape subfunction code
d. Transfer directions
e. .OPER subfunction code
f. Addresses
g. Word counts

and will pass on what appears to be legitimate values. Each handler must then make its own validity determination with respect to the device it controls. For example, the CAL handler will always accept data modes Ø through 7; however, the device handler may only accept a subset of these.

The contents of words WORD 14 through WORD 17 vary with the function being processed. Adjacent to what will appear in each of these words are the limits on the values that will be accepted and passed on by the CAL handler.

WORD 14:	.INIT	File type	Ø = input 1 = output
	.READ .REALR .WRITE .REALW	Data mode	Ø = IOPS binary 1 = Image binary 2 = IOPS ASCII 3 = Image ALPHA 4 = DUMP 5 = DUMP ALPHA 6 and 7 are undefined but are passed on by the CAL handler.
	.MTAPE	MAGtape function	Ø thru 17_8
	.TRAN	Transfer direction	Ø thru 3
	.OPER	Subfunction code	1 thru 3
WORD 15:	.INIT[2]	User restart address plus code bits (Ø - 2)	
	.READ[1]	Line buffer address	
	.REALR[1]		
	.WRITE[1]		
	.REALW[1]		

[1]Checked for non-existent memory. If this is a Background CAL and if the Background is operating in normal protect mode ($MPON) this address is also compared with the contents of .SCOM+31, the software boundary, to signal an error if the address points below the bound.

[2]Same as for footnote 1, except that the check is made on .SCOM+32, the hardware bound, and only if the function is to be executed by the Teletype handler.

	.FSTAT[1]	Address of the Directory entry block
	.DLETE[1]	
	.RENAM[1]	
	.ENTER[2]	
	.SEEK[2]	
	.TRAN[1]	Core starting address.
WORD 16:	.INIT[1]	Address of register which is to have standard buffer size placed in it.
	.READ[3]	Linebuffer word count (from the CAL argument list).
	.REALR[3]	
	.TRAN[3]	
	.WRITE[3]	Line buffer word count (from the linebuffer word pair count, except for dump mode and mode 5 which use counts from the CAL argument list.
	.REALW[3]	
WORD 17:	.FSTAT[1]	Address of the register which will have the device code placed in bits Ø - 2.
	.REALR[4]	Address to which control will be passed on completion of the real-time I/O request. Bits 0 - 2 will contain the priority code: Ø = Background Mainstream 1 = Foreground Mainstream 5 = Foreground API Level 5 6 = Foreground API Level 6 7 = Foreground API Level 7 The CAL handler always changes the Background code to Ø since Background cannot use the API software levels.
	.REALW[4]	
	.TRAN	The device address (block number).

Words 20 through 32 make up the Function JMP Table. Those functions which are ignored, those which are illegal, and those which do not issue IOT's at the CAL level must prepare to have the Foreground or Background busy flag (WORD 1 and WORD 2, respectively) cleared during the protected exit routine. Because the CAL handler in the Monitor has set the busy flag prior to entering the handler, the handler must clear the busy register since no further processing will be done.

---

[1]Checked for non-existent memory. If this is a Background CAL and if the Background is operating in normal protect mode ($MPON) this address is also compared with the contents of .SCOM+31, the software boundary, to signal an error if the address points below the bound.

[2] No address check is made.

[3]Counts are validated for the Background job to ensure a negative value and to ensure that the count added to the start address does not reference non-existent memory.

[4]Same as for footnote 1, except that the check is made on .SCOM+32, the hardware bound.

```
WORD 2Ø:     JMP INIT                  /Function 1
WORD 21:     JMP OPER                  /Function 2
WORD 22:     JMP SEEK                  /Function 3
WORD 23:     JMP ENTER                 /Function 4
WORD 24:     JMP CLEAR                 /Function 5
WORD 25:     JMP CLOSE                 /Function 6
WORD 26:     JMP MTAPE                 /Function 7
WORD 27:     JMP READ[1]               /Function 1Ø
WORD 30:     JMP WRITE[2]              /Function 11
WORD 31:     XX[3]                     /Function 12
WORD 32:     JMP TRAN                  /Function 13

WORD 33:     Ø                         /Storage for .SCOM+35, the "in an
                                       /interrupt service" flag -- set by
                                       /the CAL handler.  This is a vestige
                                       /from the PDP-9.

WORD 34:     SUBRF                     /Address of the STOP-FOREGROUND-I/O
                                       /subroutine.
```

When the Foreground job terminates as a result of a terminal error, CTRL C, or a .EXIT command, the Foreground STOPIO routine in every device handler assigned to the Foreground job is called[4] at the Mainstream level to effect the controlled shutdown of the device (see 7.8).

```
WORD 35:     SUBRF                     /Address of the STOP-BACKGROUND-I/O
                                       /subroutine.
```

For single user device handlers (devices that cannot be shared by Foreground and Background), the same subroutine can be used for Foreground and Background STOPIO (as noted above).

```
WORD 36:      Ø                        /Handler I.D. code (normally Ø).
```

This word has other values (non-Ø) for devices that require special consideration from the CAL handler. The Monitor will not allow .INIT to force its way into a busy handler unless the I.D. code is -1 and the .INIT is to the same .DAT slot on which the handler is busy.

7.5.2 .SETUP

If a device generates hardware interrupts they must be routed to the proper interrupt service routines. When the appropriate handler is not in core, the interrupts are routed to an error processor which treats the interrupts as illegal.

---

[1] Includes .REALR.

[2] Includes .REALW.

[3] .WAIT and .WAITR never get to the handler; they are processed by the Monitor's CAL handler. Word 31 can be used for data storage.

[4] The call is made only if the handler's Foreground busy register is set.

When a handler has been loaded in core it must call the Monitor's .SETUP routine (at the CAL level, API level 4) to connect the interrupt line(s) to the handler's interrupt service routine(s). As a rule, this is executed as once-only code the first time a .INIT call is processed. To ensure that .SETUP has been done for a device which has IOT's that can cause interrupts, all CAL functions that could perform such IOT's must test that a .INIT was performed at least once. If .INIT was not performed, the offending job should be terminated with an .ERR Ø6Ø (.INIT not executed).

The .SETUP routine is typically called only once for API devices; but, for devices on the Program Interrupt Control, one call per IOT in the Monitor's skip chain is required. Read section 8.3.4, Note 3, for a clearer explanation.

Calling sequence:

```
LAC*      (.SCOM+55          /Get address of .SETUP
DAC       TEMP
JMS*      TEMP               /Call .SETUP
SKPIOT                       /Argument 1:  IOT skip
INTSVC                       /Argument 2:  Address of the
(return here)                /interrupt service subroutine
```

7.5.3 Initiating I/O

For interrupt driven devices it is imperative that all IOT's that initiate hardware operations be executed during the protected exit from the handler to ensure that the exit takes place prior to the completion of the hardware operation (which could cause re-entry to the handler at the interrupt level).

CAL function requests that require more than one hardware operation should cause the 2nd through Nth operations to be initiated at the interrupt level during protected exit. A handler should not cause an implicit .WAIT for the duration of the function processing because this prevents optimum usage of the central processor time. The .CLOSE and .OPER functions are exceptions and the implied .WAIT in those functions is handled automatically by the CAL handler.

7.6 DEVICE HANDLER'S INTERRUPT PROCESSOR

The following steps detail the logic necessary for interrupt processing in a single-user handler. References to a common exit routine for both CAL's and interrupts presupposes that interrupts for the device cannot occur while CAL's are being processed.

Both page mode and bank mode systems now require API hardware, consequently tne interrupt service routine can rely on its existence.

1. If this is an API interrupt, the PC, Link (bit Ø), Page/Bank (bit 1), and Memory Protect (bit 2) are stored at the entry point to this routine. If this is a PIC interrupt, the PC et al. are stored in location zero in memory. This routine is entered at the API level of the device (level 3 with PIC off if a PIC interrupt), with memory protect disabled.

2. DBA -- Disable Bank (Enter Page) Address Mode.

3. Save the AC -- to be restored on exit from interrupt service. Also, save hardware registers such as XR, LR, if they are to be used herein.

4. Save the PC, Link, Page/Bank Mode, and Memory Protect in WORD 7 of the handler. WORD 7 is so used if the CAL and interrupt code can use common exit logic.

5. Store a DBR instruction in WORD 6 of the handler. This is done only if the CAL and interrupt code can use common exit logic.

6. Turn off (clear) the device's hardware flag so that it will not cause another interrupt unless reset.

7. If this is a PIC interrupt, set location zero to zero and execute the ION instruction to turn the Program Interrupt Control on. Location zero with non-zero contents indicates a PIC interrupt in progress (this was significant in PDP-9 logic. On a PDP-9, the device flag(s) must be cleared before the PIC is turned on again.)

8. If this is the type of device for which I/O in progress cannot be stopped, test if this is the last interrupt expected from the device.[1] If it is, clear (set to zero) the IOSTOP flag (read section 7.8 describing STOPIO procedures). Only one IOSTOP flag is needed for a single user or sequential multi-user handler.

9. Is the busy flag (WORD 1 if Foreground; WORD 2 if Background) zero? If it is (meaning that the handler was not busy and that the interrupt was unsolicited, or that the STOPIO routine was called), ignore the interrupt by going to step 13.

1Ø. Process the interrupt, e.g., store data or prepare to transmit more data.

11. Is the I/O request (the I/O call issued by the user program) complete as a result of this last interrupt? If so, continue at step 14.

[1]The CRØ3B card reader handler must test if this is last interrupt since, for one read operation, 8Ø column interrupts will occur and there is no way to prevent them.

12. If more I/O must be done before the CAL request is satisfied, set up the protected exit routine to issue the next IOT(s) to the device. Go to step 22, the protected exit routine.

13. If the interrupt was unsolicited, set up to have the Foreground or Background busy flag (WORD 1 or 2 as appropriate) cleared during the protected exit code. Go to step 22, the protected exit routine. Note that WORD 11 in the handler indicates Background or Foreground ownership of the I/O request.

14. I/O is complete as a result of this interrupt; therefore, set up to have the Foreground or Background busy flag (WORD 1 or 2 as appropriate) cleared during the protected exit code. Note that WORD 11 in the handler indicates Background or Foreground ownership of the I/O request.

15.
```
LAC*     (.SCOM+1Ø2
ISA
```
This will raise to API level zero or one, depending upon the contents of .SCOM+1Ø2. If this device interrupts at level one or zero, reread section 7.4.

16.
```
LAC*     (.SCOM+52
DAC      TEMP
LAC      (WORD Ø
JMS*     TEMP
```
.SCOM+52 contains the address of the Monitor's IOBUSY subroutine, which must be protected against reentrancy. It is passed an argument in the AC -- the address of WORD Ø of the device handler. IOBUSY will compare this with the contents of the Foreground busy registers (.SCOM+42 through .SCOM+45). If a match is found, it indicates that the Foreground level was waiting for I/O completion by this device. Since I/O has completed, the Foreground level is made not busy and is set up to resume operation when that level becomes the highest active level in the system. The level-busy .SCOM registers are set originally by the Monitor's CAL handler. Note that the call to IOBUSY must be done for Background as well as Foreground I/O completion.

17.
```
DBK
```
Debreak from API level zero or one (reread section 7.4 if this device interrupts at level zero or one). This is done, as is step 19, simply to allow interrupts to occur for higher level devices if their flags come up while the IOBUSY code is being executed.

18. The handler must check if this is a real-time I/O request. It is if WORD 1Ø contains a JMP to WORD 27 (.READ or .REALR) or to WORD 3Ø

(.WRITE or .REALW) and WORD 17 is non-zero. If this is not a real-time request, go to step 22.

19. LAC* (.SCOM+1Ø2
ISA

Raise to API level zero or one to protect the following Monitor routine from being reentered. If this device operates at level one or zero, reread section 7.4.

2Ø. LAC* (.SCOM+51
DAC TEMP
LAC WORD 17
JMS* TEMP

.SCOM+51 contains the address of the Monitor's REALTP subroutine. It is passed an argument in the AC -- the contents of WORD 17 which is the priority level code plus real-time subroutine address. REALTP will prime the Monitor to run the real-time subroutine.

21. DBK

From API level zero or one, as in step 17.

22. This is the beginning of the protected exit routine. Note that it is coded as a common exit for both interrupt and CAL code. This is explained in section 7.3.2. If I/O is to be performed now to the device, check the device to ensure that it is ready to accept I/O commands. If the device is not ready, a message must be output to denote this fact (see section 7.9) and the I/O must be deferred. Therefore, set a program flag, call it IOTFLG, for example, so that the IOT(s) will not be issued in step 27. If the device is ready, clear IOTFLG.

23. LAC* (.SCOM+1Ø2
ISA

Raise to API level zero or one. Reread section 7.4 for level zero or level one devices.

24. If WORD 6 contains a DBR instruction (recall step 6), this is an interrupt service exit. If so, request an API level 4 interrupt as follows:

```
LAC     (4Ø4ØØØ
ISA
```

The API level 4 handler is the Monitor's dispatch routine. It controls transitions from Background to Foreground, real-time requests, errors, and control character functions.

25. If previously set up to do so, as in steps 13 and 14, clear the Background and/or Foreground busy flags (WORDS 2 and 1, respectively). The busy flag is cleared on ignored functions, completed functions, and aborted functions.

26. Is I/O supposed to be done now? If IOTFLG (set or cleared in step 22) is non-zero, if the appropriate busy flag is zero, or if the IOSTOP flag is non-zero (see step 8) go to step 28 to bypass execution of the IOT(s).

27. Execute the IOT(s) to the device. This may involve several instructions. If this is an exit from the CAL level of the handler and if no I/O is to be done (e.g., an ignored function), this code will be bypassed since the busy flag was cleared.

28. Restore the AC plus any other hardware registers used (refer to step 3).

29. DBK
    Debreak from API level zero or one (reread section 7.4 if this is a level zero or level one device).

3Ø. XCT WORD 6
    This will be a DBR instruction if this is an interrupt exit (recall step 6) and ION if this is a CAL exit. This assumes that the device handler can have common interrupt and CAL exit code.

31. JMP* WORD 7
    Again, this assumes common CAL and interrupt exit code. The DBR in step 3Ø for an interrupt exit will debreak out of the device's hardware level and will prime the machine to restore the state of the Link, Page/Bank Mode, and Memory Protect (in this case from bits Ø - 2 of WORD 7). The ION instruction in step 3Ø for a CAL exit is effectively a NOP. Return will be at API level 4 to CALXIT, the common CAL exit routine in the Monitor.

Note that once the DBK in step 29 is executed, the sequence of code leading to step 31, the JMP*, must be non-interruptible, i.e., a string of IOT instructions.

If the exit is done with the device not ready, note that the device's busy flags are still set. The job will continue execution without knowledge of the not-ready situation. Of course, if the job attempts to perform more I/O to the device (.INIT being a special case; see section 7.9.2), it will become I/O bound.

7.7 ERROR PROCESSING

All device handler error conditions should be terminal; that is, they should terminate the operation of user programs. Whether errors are detected during CAL processing or during interrupt processing the following coding sequence will set up an error condition and cause an error message to be printed on the appropriate job control Teletype. This coding sequence cannot be common to both the CAL and interrupt levels unless it is known that the device cannot cause interrupts while the handler is processing a CAL.

1.
```
LAC*    (.SCOM+66
DAC     TEMP                    /This TEMP cannot be common.
                                /(See preceding paragraph.)
```
.SCOM+66 contains the address of the error queuer subroutine in the Monitor.

2.
```
LAC*    (.SCOM+1Ø2
ISA
```
Raise to API level zero or one to protect the error queuer from being reentered. Reread section 7.4 for level zero and level one devices.

3.
```
LAW code or LAC (code
JMS*  TEMP
auxarg
```
Go to the error queuer with one argument in the AC and one following the JMS instruction. The argument in the AC, loaded either by LAW code or LAC (code, is formatted as follows:

Bits Ø - 5 are ignored
Bit 6 = 1 means a terminal error
Bit 7 = 1 means a Background error
Bit 8 = 1 means a Foreground error
Bits 9 - 17 form a 3-digit error code

If the error pertains to both jobs then both bits 7 and 8 may be set.

The auxiliary argument, auxarg, is simply a 6-digit quantity to be printed with the error message, which is of the form:

.ERR NNN XXXXXX)

NNN is a 3-digit error code
XXXXXX is a 6-digit auxiliary argument

4. DBK
   Debreak from API level zero or one. Reread section 7.4 if this is a level zero or level one device.

5. If no further interrupts are expected, set up to have the appropriate job busy flag (WORD 1 or WORD 2) cleared during protected exit from the handler. However, if more interrupts are expected, the busy flag must remain set to signal the STOPIO routine that interrupts are pending. Instead, set the IOSTOP flag so that I/O set up to be executed in the protected exit routine will be bypassed.

The actual printing of the error message will not be done until all interrupt and CAL processing is complete. Background error messages are not printed until the Background job is given control.

As a result of a terminal error, the handler can be certain that its STOPIO routine will be called by the Monitor if the handler's busy flag is set for the job in error.

7.8 STOP I/O ROUTINES

In Background/Foreground it is necessary to have some orderly means of stopping I/O that is in progress. When a job terminates (.EXIT, terminal error, etc.), the Monitor must ensure that all I/O for that job is shut down. This is particularly necessary in the Background where I/O must be stopped before the associated device handlers are removed from core.

WORD 34 of each handler must contain the address of its Foreground STOPIO subroutine.

WORD 35 of each handler must contain the address of its Background STOPIO subroutine, which for single-user handlers can be the same as the Foreground STOPIO routine.

Whenever a job terminates execution, the Monitor calls the appropriate STOPIO subroutine at the Mainstream level. The following steps should be followed:

1. Is this the type of device for which I/O can be terminated by issuing an IOT instruction? Terminating I/O means ensuring that no further interrupts will occur. If so, do so, and continue at step 6.

2. LAC* (.SCOM+1Ø2
   ISA
   Raise to API level zero or one to protect against getting interrupted in mid-decision.

3. If this device does not generate Not-Ready conditions (read section 7.9), go to step 5.

4. Check the "CTRL R in progress" flag (see CTRLR in section 7.9). If it is set, clear the STOPIO flag (which is tested in step 11) and go to step 6. This is done because I/O cannot be under way if the handler is waiting for ↑R. Otherwise, go to step 5.

5. Check the appropriate job busy flag (WORD 1 or WORD 2). If it is set, set the IOSTOP flag (a flag internal to the handler) non-zero. Otherwise, clear it. This flag is tested in step 11.

6. Clear the appropriate job busy register (WORD 1 or WORD 2).

7. Clear the appropriate .CLOSE register (WORD 3 or WORD 4).

8. If the handler has one, clear the "CTRL R in progress" flag (see step 18 in section 7.9).

9. If this is the type of device which can terminate I/O by IOT (refer back to step 1), go to step 12.

1Ø. DBK
    Debreak from API level zero or one back to Mainstream to allow hardware flags that may have or will occur to be serviced.

11. LAC IOSTOP
    SZA
    JMP .-2
    If the appropriate busy register had been set and I/O is under way, control will stay here until the IOSTOP flag is set to zero. That will happen only when the device's final interrupt occurs (refer to section 7.6, step 8). This loop is executed at Mainstream, in case the handler is being used by Background, so that the Foreground job may resume execution when ready.

12. JMP* STOPIO
    Exit back to the Monitor. I/O by the device has stopped and, if the job is restarted, the handler flags have been reset so that the handler can accept more I/O commands.

7.9 RECOVERY FROM I/O DEVICE NOT READY CONDITION

7.9.1 CTRL R Mechanism

The Background/Foreground Monitor system is designed to handle simultaneously one not-ready condition per job. This is a limitation but a reasonable one based on Keyboard Monitor (single user) experience.

I/O handlers that can encounter and detect not-ready conditions must adhere to the following ground rules in their announcement of the not-ready condition and in their continuation once the condition has been corrected.

Some devices are designed so that they can be tested at any time for a state of readiness; therefore, the test can be made at the CAL level prior to starting I/O. Other devices will not generate a not-ready condition until after an IOT has been issued and an error flag results. In such cases, the not-ready condition is detected at the interrupt level.

The reader is assumed to understand the mechanism whereby a device interrupt transfers control to a handler's interrupt service routine at a hardware API level and the process called .SETUP whereby the handler connects itself to the device's interrupt line(s). When the handler is not in memory, interrupts from the device are shunted to the illegal interrupt handler. When the handler is in core and has performed the .SETUP, device interrupts will transfer control to the handler. The processing of a device-not-ready condition involves a pseudo-.SETUP and a simulated API interrupt, which will be explained at the end of this section.

It is best to check for device ready in only one location, the beginning of the protected exit routine in the handler. This starts at step 22 in section 7.6, where it is assumed that the CAL and interrupt portions of the handler share a common exit logic.

1. Test for device ready or not. This is the same as step 22 in section 7.6. If the device is ready, set IOTFLG to zero so that the IOT(s) in step 27 may be executed, and go to step 23 in section 7.6.

2. With the device not ready, it is necessary to defer the IOT(s), announce the not-ready condition, and exit from the handler set up to continue after CTRL R is typed on the user's control Teletype. For a single-user or sequential multi-user handler, the IOT(s) that were to be executed may remain where they are. Set IOTFLG non-zero so that they will not be executed in step 27 of the protected exit logic (section 7.6). (For multi-user handlers, the IOT(s) must be physically moved in case I/O

for the other job is started up.)

3. JMS NRMSG
   Call a subroutine to initiate the printing of the not-ready message. Then go to step 23 in the protected exit routine (section 7.6).

Steps 4 through 11 contain the code for subroutine NRMSG.

4. NRMSG Ø /Entry point.
```
LAC   CTRLR
SZA!CLC
JMP*  NRMSG
DAC   CTRLR
```
   Register CTRLR is a program flag internal to the handler. If it contains zero, the handler has not already initiated a not-ready request. If it is non-zero, exit from the subroutine, since a not-ready condition has been announced. If CTRLR was zero, set it non-zero.

5.
```
LAC   WORD11
DAC   ARG1
```
   WORD 11 in the handler contains zero if Foreground and one if Background. This is passed on as argument one in the call to the Monitor's CTRL R setup routine.

6.
```
LAC   UNITNO
DAC   ARG3
```
   Bits Ø-2 of argument three are considered to be the device unit number, which is printed as part of the device-not-ready message. Some devices have only one unit, for example, the papertape punch; and this code is, therefore, unnecessary. The card reader handler uses the unit number in the printout to indicate the cause of the not-ready condition.

7.
```
LAC*  (.SCOM+64
DAC   TEMP                  /Beware -- TEMP probably cannot be used
                            /by both the CAL and interrupt levels.
```
   .SCOM+64 contains the address of the CTRL R setup subroutine, which is part of the Teletype handler in the Monitor. Store this address in a temporary register.

8.
```
LAC*  (.SCOM+1Ø2
ISA
```
   Raise to API level zero or one. Reread section 7.4 if this device is already operating at level one or zero.

9. JMS* TEMP

```
ARG1   XX
ARG2   .ASCII /DV/
       .LOC   .-1
ARG3   XX
ARG4   F.CTLR+?ØØØØØ
ARG5   B.CTLR+?ØØØØØ
```

Call the CTRL R subroutine in the Monitor. Argument 1 contains zero if Foreground and one if Background. Argument 2 is the two-letter device name in .ASCII, e.g., LP for Line Printer. Argument 3 is the device's unit number, in bits Ø-2. Argument 4 is the address (F.CTLR) and API level code (?ØØØØØ) of the subroutine which is to be entered when a Foreground not-ready condition for this device exists and CTRL R is typed on the Foreground control Teletype. Argument 5 is similar to argument 4, but is used for Background. (?ØØØØØ) for API level 2 would be 2ØØØØØ, for example. Only levels Ø, 1, 2, or 3 are allowed.

For device DV with unit number Ø, the not-ready message would be printed as follows:

DVØ NOT READY)

Return from the CTRL R subroutine will either be normal (step 1Ø) or skip one location (to step 11).

1Ø. DZM CTRLR

If the Monitor's CTRL R subroutine does not skip on return (returns here), it is because the request to set up a not-ready condition was not honored. This would happen if, for this job, a not-ready condition had been established for some other device. No queueing mechanism exists; thus, two simultaneous not-ready conditions for a job will result in a job terminal error, .ERR ØØ4. When return is to step 1Ø, the Monitor has already posted the .ERR ØØ4 printout request.

11. DBK
JMP* NRMSG

If the Monitor has honored the request to set up a device-not-ready condition, step 1Ø will have been bypassed. Debreak from API level zero or one. Reread section 7.4 if this device operates at level zero or one.

Steps 12 through 18 contain the code for subroutine F.CTLR and B.CTLR.

12. F.CTLR ø /Entry point
B.CTLR=F.CTLR

For a single-user handler, the same subroutine can be used for Foreground and Background, as indicated by the equivalence statement.

Prior to entering this routine, the Monitor was called by the handler to set up a not-ready condition for this device. A not-ready message was printed on the appropriate job control Teletype and the CTRL R function for that job was primed by storing ARG4 or ARG5, as appropriate, (see step 9), in the Monitor's Foreground CTRL R or Background CTRL R register.

Note that this has the effect of a pseudo-.SETUP call. If CTRL R is now typed on the appropriate job control Teletype (the user's way of posting a "done" flag), the corresponding CTRL R register in the Monitor acts like an API channel register. The Teletype handler raises to the designated API priority level and then performs a JMS to this subroutine. The subroutine must be entered at API level zero, one, two, or three. Prior to entering this subroutine, the Teletype handler will clear the relevant Monitor CTRL R register to disable CTRL R until another not-ready condition is established.

13. DZM CTRLR
Clear the handler's not-ready-condition-in-progress flag.

14. Test for device ready or not. If ready, go to step 16.

15. JMS NRMSG
The device still isn't ready. Reestablish the not-ready condition and then exit by going to step 17.

16. Execute the IOT(s) that were deferred for this device, i.e., start I/O up again.

17. DBR
JMP* F.CTLR
Debreak and return to the Teletype handler. Note that the AC need not be restored.

18. CTRLR ø
The "CTRL R in progress" flag must initially be cleared. It must also be cleared in the STOPIO subroutine and in the .INIT code. For a single-user handler, only one CTRLR register is needed.

7.9.2 .INIT Consideration

WORD 36 of the I/O handler is an identification code which is zero for most handlers. Only if the code is -1 (777777)[1] will the Monitor allow .INIT to be processed by a busy handler.

In the latter case, the handler must test its CTRLR flag (and then clear it) to see if a not-ready condition existed for the device. If so, the handler must also clear the appropriate CTRL R register in the Monitor as follows:

```
        LAC     CTRLR
        SNA
        JMP     OVER
        LAC     (.SCOM+67
        TAD     WORD11          /Ø = FGD; 1 = BGD.
        DAC     TEMP            /.SCOM+67 = FGD; .SCOM+7Ø = BGD.
        LAC*    TEMP            /Address of FGD or BGD CTRL char. table.
        TAD     (4
        DAC     TEMP
        DZM*    TEMP            /Ø Monitor's B or F CTRL R.
OVER    DZM     CTRLR           /Ø Handler's own CTRL R.
```

Note that .SCOM+67 points to the Foreground control character table in the Teletype handler and that .SCOM+7Ø points to the Background control character table. WORD 11 in the handler contains zero if Foreground and one if Background. The CTRL R register for each job is the fifth entry in each of these tables.

7.10 THE .INIT FUNCTION

In order to satisfy the requirements of the disk and DECtape handlers, the FIOPS routine in the FORTRAN Object Time System operates somewhat differently in Background/Foreground from the way it does in the Keyboard Monitor. In Background/Foreground, FIOPS will perform a .INIT to a given .DAT slot only the first time the slot is referenced, not each time the direction of data transfer changes.

The .INIT-only-once change is necessary because DTA. and DKA. can perform non-file-oriented I/O, that is, treat the DECtape or Disk as if it were MAGtape. .INIT always resets the device to file-oriented mode and is, therefore, avoided.

This change has ramifications for user-written device handlers, even if they are strictly file-oriented handlers. .INIT can no longer be relied upon to signal a change in the direction of data transfer. However, if .CLOSE is followed by .SEEK, .ENTER, .FSTAT, .DLETE, .RENAM, .CLEAR, or REWIND (.MTAPE Ø), the transfer direction is obvious anyway. Therefore, .INIT need only be used to tie the

---

[1]This is a necessary but not sufficient condition.

handler to its interrupt lines (.SETUP) and to abort I/O (after CTRL P, for example).

7.11 SEQUENTIAL MULTI-USER DEVICE HANDLER

7.11.1 Transition from Single-user Handler

To accomplish the transition from a single-user device handler to a sequential multi-user device handler, the following procedures must be adhered to:

1. The device handler must be the "A" version; that is, LPA., MTA., etc., as the Background/Foreground Monitor System will only allow "A" versions to be connected to both jobs simultaneously. Also, this shareability must be specified to the B/F System Generator.

2. The SWAP subroutine (pointed to by WORDØ of the handler) must set both busy registers (WORDl and WORD2) to prevent the Foreground job from forcing itself in before the Background job has completed its operation. This is in addition to and prior to its normal duties as outlined in 7.5.1.

3. The handler's identification code, WORD36, should not be -1. If it were, it would be possible for one job to abort the I/O operation of the other.

4. There must be two unique STOP I/O subroutines, one for Foreground (pointed to by WORD34) and one for Background (pointed to by WORD35). Before executing the STOP I/O procedures, both subroutines must first determine if the I/O belongs to their respective jobs. This is done by testing WORDll, (Ø=Foreground I/O, l=Background I/O). They should do nothing if the other job is in control.

   In step 8 of the STOPIO routine, section 7.8, check the CTRLR flag before clearing it. If the flag was set, call the I/O BUSY routine in the Monitor (as in steps 15, 16, 17 of section 7.6) in case some level of the Foreground job is I/O bound on this device.

5. Because the SWAP subroutine sets both busy registers (WORDl and WORD2), the CLEAR BUSY FLAG routine that sets up to have the flags cleared during protected exit from the device handler (refer to steps 13 and 14 in section 7.6) must always setup to have both flags cleared. The STOP I/O subroutines should also clear both busy registers.

7.11.2 Peculiarities

It is understood that in multi-user handlers, such as DTA., the Foreground has a built-in priority. Therefore, it comes as no surprise that the Foreground job can completely prevent the Background from performing DECtape I/O. For sequential multi-user handlers, one might assume that Foreground and Background I/O operations would compete on an equal basis. This may not be the case.

It is possible, given the right set of circumstances, that Foreground never gets a chance to manipulate the device, that Background never gets a chance to manipulate the device, or that one program (not necessarily Foreground) does more actual I/O to the device than an identical program running as the other job.

This situation should only become a problem when one or both jobs attempt continuous operation of the device. The "right set of circumstances" depends upon where processing is in the Monitor's CAL handler when the current I/O operation for the device completes (interrupts).

7.11.3 Use of the .WAITR Function

When a sequential multi-user device handler is being used by the Background job, the Foreground job will become I/O bound if it attempts to use the same handler.

The .WAITR monitor function affords both the Foreground job and the Background job a means of determining that the handler is available before requesting I/O from and to it. This feature is only useful when the job has other things which can be performed while it is waiting for the handler to free up.

The use of .WAITR in this manner is foolproof when executed in the Foreground. This is not so in the Background because the Foreground job can regain control after the Background .WAITR has been executed and before the ensuing Background I/O command.

7.12 EXTERNAL I/O BUFFERS

Device handlers which might require a great deal of buffer space may do well to use the system's capability of setting aside I/O buffers at load time. Only multi-user or sequential multi-user handlers (the shareable "A" versions) may utilize external buffers.

Buffer sizes required by each shareable handler are specified during system generation. Buffers are set aside at load time by the Loaders either as a result of a $FILES Keyboard command or, in lieu thereof, one per .DAT slot which references the multi-user device handler.

Typically, the handler would test to see if it had a buffer for a given .DAT slot before performing the I/O request. If not, it would call the GETBUF routine in the Monitor to scan the buffer table, .BFTAB, for a usable free buffer. At the end of the I/O sequence, usually .CLOSE, the handler must relinquish the buffer so that other handlers might use it.

7.12.1 Calling for a Buffer

At run time, the handler may obtain an external I/O buffer as follows:

1.
```
LAC*    (.SCOM+56
DAC     TEMP                    /Beware -- TEMP probably cannot be
                                /used by both the CAL and interrupt
                                /levels.
```
The address of the GETBUF subroutine in the Resident Monitor is in .SCOM+56.

2.
```
LAC*    (.SCOM+1Ø2
ISA
```
Raise to API level zero or one. Reread section 7.4 if the handler is already at level zero or one.

3.
```
JMS*    TEMP
argument
```

Call GETBUF with one argument:

```
Bit Ø = Ø if Foreground
Bit Ø = 1 if Background
Bits 1-5 = Ø
Bits 6-17 = Buffer size.
```

GETBUF will search .BFTAB for a free Foreground or Background buffer, as specified, of a size equal to (or greater than, if necessary) that indicated in the argument.

4. If a buffer is found, the address of the first word of the .BFTAB entry is returned in the AC and the entry is flagged busy by the GETBUF routine. If no buffer can be found, zero is returned in the AC and GETBUF initiates a terminal error (.ERR Ø55) for the job.

5.
```
DBK
```
Debreak -- Reread section 7.4 if this is an API level one or zero device.

7.12.2 Releasing a Buffer

The format of .BFTAB is given in Appendix IV. When the handler wishes to relinquish a buffer, it does so by clearing the busy bit of the entry in .BFTAB. Note that the address of the first word of this entry in .BFTAB is returned in the AC by the Monitor subroutine GETBUF.

7.13 PDP-9/PDP-15 COMPATIBILITY

7.13.1 Page Mode

The I/O handler description in this manual was written for page-mode operation, which is valid only on the PDP-15.

Two coding requirements which are necessary for PDP-9 hardware may be omitted for handlers that are to run in page-mode-only systems: (1) raising to API level 3 and (2) double XCT .+1 following DBR (see 7.13.2). For page mode operation on a PDP-15, add a DBA (Disable Bank Addressing = Enter Page Mode) instruction as the first instruction in the handler's interrupt service routine.

7.13.2 Bank Mode

Since handlers that operate in the bank mode system must be able to run on both a PDP-9 and a PDP-15 (assuming that the device exists on both machines), the following PDP-9 requirements must be followed:

1. Do not insert a DBA instruction at the beginning of the interrupt service routine.
2. If a device on the PDP-9 is connected to the PIC (Program Interrupt Control) but not to API, then the interrupt service routine must raise to API level 3 before executing the ION instruction. On the PDP-15, this raise to level 3 is done automatically by the hardware. Formerly, when API hardware was optional on PDP-9 Background/Foreground, an "in interrupt service routine" flag (.SCOM 35) was needed to signal that state after the ION instruction was executed.
3. To allow API synchronization following a DBR instruction on a PDP-9, the following exit sequence must be used:

```
DBR
XCT .+1
XCT .+1
JMP*
```

7.14 DEVICE HANDLER LISTING

The following pages contain the assembly listing of a paper tape reader handler (PRA.) for PDP-15 Background/Foreground operation.

```
                        .TITLE -- PRA.

                       /COPYRIGHT 1970, DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

                       /PAPER TAPE READER I/O DEVICE HANDLER FOR THE
                       /PDP-15 BACKGROUND/FOREGROUND MONITOR SYSTEM.

                       /THIS IS NOT THE VERSION OF PRA, SUPPLIED WITH
                       /THE BACKGOUND/FOREGROUND SYSTEMS, IT IS ADAPTED
                       /(BY C, PROTEAU) FROM PRA, (WRITTEN BY D, LENEY
                       /AND M, SIFNAS) TO CONFORM TO THE DESCRIPTION
                       /OF A DEVICE HANDLER IN SECTION 7 OF THE PDP-15/30
                       /AND PDP-15/40 MONITOR MANUAL, THE MAJOR DIFFERENCE
                       /BETWEEN THESE TWO VERSIONS IS THAT THIS ONE IS
                       /NOT PARTICULARLY AMENABLE TO CONVERSION FOR
                       /OPERATION IN PDP-9 BACKGROUND/FOREGROUND.


                       /CHARACTERISTICS:
                       /
                       /       1, SINGLE-USER (NON-SHAREABLE) HANDLER
                       /       2, NON-FILE-ORIENTED (A TERM SOMETIMES
                       /               TAKEN TO MEAN NON-RANDOM-ACCESS)
                       /       3, HANDLES DATA MODES 0 THROUGH 4
                       /       4, CAN USE CODE AND REGISTERS IN COMMON
                       /               BY CAL AND INTERRUPT SERVICE
                       /       5, OPERATES ENTIRELY IN PAGE ADDRESSING MODE

                        .EJECT
```

```
                700101 A    RSF=700101      /SKIP IF READER FLAG IS SET.
                700112 A    RRB=700112      /READ READER BUFFER INTO THE AC
                                            /AND CLEAR THE READER FLAG.
                700104 A    RSA=700104      /SELECT READER IN ALPHANUMERIC MODE.
                                            /CLEAR THE READER FLAG AND THEN
                                            /READ ONE 8-BIT CHARACTER (RIGHT
                                            /JUSTIFIED) INTO THE READER BUFFER.
                700144 A    RSB=700144      /SELECT READER IN BINARY MODE.
                                            /CLEAR READER FLAG AND THEN READ
                                            /THREE 6-BIT CHARACTERS AND ASSEMBLE
                                            /THEM INTO THE READER BUFFER TO FORM
                                            /ONE 18-BIT BINARY WORD. IN THIS MODE,
                                            /BIT 7 OF A TAPE LINE IS IGNORED; AND
                                            /THE LINE IS IGNORED IF BIT 8 IS
                                            /NOT SET.
                002100 A    .SCOM=100       /BASE ADDRESS OF THE MONITOR'S
                                            /SYSTEM COMMUNICATION REGISTERS.
                440000 A    IDX=ISZ         /IDX IS USED INSTEAD OF ISZ WHEN THE
                                            /INTENT IS TO ADD 1 TO A REGISTER BUT
                                            /NOT TO SKIP.

                                    .EJECT
```

```
                                /THIS AND THE NEXT PAGE CONTAIN THE FIRST 37 OCTAL WORDS WHICH
                                /ARE STRUCTURED ACCORDING TO BGD/FGD HANDLER CONVENTION.

                                        .GLOBL PRA.

                                /NOTE -- A HANDLER MAY HAVE ONLY 1 GLOBAL SYMBOL.

00000 R 100037 R        PRA.    JMS PRSWAP      /WD 0 - JMS TO SWAP SUBROUTINE
00001 R 000000 A                0               /WD 1 - FGD BUSY REGISTER
00002 R 000000 A                0               /WD 2 - BGD BUSY REGISTER
00003 R 000000 A                0               /WD 3 - FGD CLOSE REGISTER
00004 R 000000 A                0               /WD 4 - BGD CLOSE REGISTER
00005 R 740040 A                XX              /WD 5 - ION (SET BY CAL HANDLER)
                                                /       THIS IS AN UNUSED VESTIGE FROM PDP-9.
00006 R 740040 A                XX              /WD 6 - ION OR IOF OR DBR
00007 R 740040 A                XX              /WD 7 - RETURN POINTER

                                /START OF LIVE/BACKUP DATA REGISTERS

00010 R 740040 A                XX              /WD 10 - JMP TO FUNCTION
00011 R 740040 A                XX              /WD 11 - WHOSE CAL (0=FGD,1=BGD)
00012 R 740040 A        PR8CT   XX              /WD 12 - .DAT SLOT# - 8TH BIT COUNT
00013 R 740040 A        PARER   XX              /WD 13 - UNIT # AND CAL ADDRESS - PARITY COUNT
00014 R 740040 A        PTRDM   XX              /WD 14 - DATA MODE
00015 R 740040 A        PRLBHP  XX              /WD 15 - LINE BUFFER ADDRESS
00016 R 740040 A        PTRWC   XX              /WD 16 - -WC OR BUFFER SIZE POINTER
00017 R 740040 A                XX              /WD 17 - REAL TIME REQUEST

                                /END OF DATA REGISTERS

                                        .EJECT
```

```
PAGE   4     PRA.   008     -- PRA.

  83                              /START OF FUNCTION DISPATCH TABLE
  84
  85        00021 R 600043 R              JMP PRIN        /WD 20 - 1=,INIT
  86        00021 R 600517 R              JMP PRIGN       /WD 21 - 2=,OPER-IGNORED
  87        00022 R 600517 R              JMP PRIGN       /WD 22 - 3=,SEEK-IGNORED
  88        00023 R 600621 R              JMP PRER6       /WD 23 - 4=,ENTER-ERROR
  89        00024 R 600621 R              JMP PRER6       /WD 24 - 5=,CLEAR-ERROR
  90        00025 R 600517 R              JMP PRIGN       /WD 25 - 6=,CLOSE-IGNORED
  91        00026 R 600517 R    PRJIGN    JMP PRIGN       /WD 26 - 7=,MTAPE-IGNORED
  92        00027 R 600065 R    PREAD     JMP PRRED       /WD 27 - 10=,READ,,REALR
  93        00030 R 600621 R              JMP PRER6       /WD 30 - 11=,WRITE,,REALW = ERROR
  94        00031 R 000000 A    PTR57     0               /WD 31 - 12=,WAIT,,WAITR
  95                                                      /        SINCE THIS FUNCTION IS PROCESSED
  96                                                      /        ENTIRELY BY THE MONITOR, PRA, USES
  97                                                      /        THIS REGISTER AS A VARIABLE STOR-
  98                                                      /        AGE -- 5/7 ASCII CHARACTER POSITION.
  99        00032 R 600621 R              JMP PRER6       /WD 32 - 13=,TRAN-ERROR
 100        00033 R 000000 A              0               /WD 33 - SAVED ,SCOM+35 (SET BY CAL HANDLER)
 101                                                      /        THIS IS AN UNUSED VESTIGE FROM PDP-9,
 102        00034 R 000644 R              PRSTOP          /WD 34 - STOP FGD I/O
 103        00035 R 000644 R              PRSTOP          /WD 35 - STOP BGD I/O
 104        00036 R 000000 A              0               /WD 36 - HANDLER ID = 0


PAGE   5     PRA.   008     -- SWAP SUBROUTINE -- API LEVEL 0 OR 1

 105                                      .TITLE -- SWAP SUBROUTINE -- API LEVEL 0 OR 1
 106
 107                              /THIS SUBROUTINE IS ENTERED BY THE
 108                              /CAL HANDLER VIA WORD 0 OF THIS I/O
 109                              /HANDLER JUST PRIOR TO GIVING CONTROL
 110                              /TO THE HANDLER AT THE APPROPRIATE
 111                              /ENTRY IN THE FUNCTION DISPATCH TABLE.
 112
 113        00037 R 000000 A    PRSWAP    0
 114        00040 R 400005 R              XCT PRA.+5      /ION.
 115        00041 R 703304 A              DBK             /FROM LEVEL 0 OR 1 TO LEVEL 4.
 116        00042 R 620037 R              JMP* PRSWAP     /RETURN TO CAL HANDLER.
```

```
                                .TITLE -- CAL PROCESSING -- API LEVEL 4

                        /.INIT -- CAL FUNCTION

                        /NOTE -- SINCE THE HANDLER I.D. CODE (WORD 36) IS NOT =1, .INIT
                        /CANNOT OVERRIDE A BUSY CONDITION, THEREFORE, NO CAL WILL BE
                        /ALLOWED INTO PRA, UNTIL PRA, HAS CONCLUDED ITS INTERRUPT PROCESSING.

00043 R 200673 R    PRIN    LAC (64
00044 R 060016 R            DAC* PRA.+16    /STANDARD BUFFER SIZE=52(10)

                        /NOTE -- THE TEST ON .SETUP HAVING BEEN DONE AND, CONSEQUENTLY,
                        /AT LEAST ONE .INIT, IS BASED ON PRNIT BEING SET TO "JMP PRIGN".
                        /THE FOLLOWING IS ONCE-ONLY CODE; HENCE, THESE REGISTERS ARE USED
                        /LATER ON TO STORE DATA VARIABLES.

00045 R 220674 R    PRNIT   LAC* (.SCOM+55  /ADDRESS OF MONITOR'S .SETUP ROUTINE.
00046 R 040012 R    PRDTCT  DAC PR8CT       /DATA WORD COUNT
00047 R 120012 R    PRCCT   JMS* PR8CT      /.SETUP - CHAR COUNT
00050 R 700101 A    PRDBP   RSF             /LATER DATA WORD POINTER
00051 R 000160 R.   PRCKSM  PTRINT          /CHECKSUM (IOPS ASCII)
                    000051 R    PRDVS2=PRCKSM           /TEMPORARY STORAGE FOR PRDVS SUBROUTINE.
00052 R 200054 R    PRCNT   LAC .+2         /PARITY CHECK COUNTER
00053 R 040045 R    PTRAC   DAC PRNIT       /SAVED AC
00054 R 600517 R    PRCHAR  JMP PRIGN       /CHARACTER PROCESSED

                        /END OF ONCE-ONLY CODE.
                        /TABLE OF IOT'S USED WITH THE VARIOUS DATA MODES. NOTE THAT
                        /5, 6 AND 7, BEING POSITIVE VALUES, INDICATE ILLEGAL DATA MODES.

00055 R 700104 A    PTRIOT  RSA     /IOPS BINARY    /MODE 0
00056 R 700144 A            RSB     /IMAGE BINARY   /MODE 1
00057 R 700104 A            RSA     /IOPS ASCII     /MODE 2
00060 R 700104 A            RSA     /IMAGE ALPHA    /MODE 3
00061 R 700144 A            RSB     /DUMP MODE      /MODE 4
00062 R 000617 R            PRER7   /ILLEGAL MODE   /MODE 5
00063 R 000617 R            PRER7   /ILLEGAL MODE   /MODE 6
00064 R 000617 R            PRER7   /ILLEGAL MODE   /MODE 7
                            .EJECT
```

```
                                /.READ OR .REALR -- CAL FUNCTION

00165 R 200245 R    PRRED   LAC PRNIT       /IF PRNIT HASN'T BEEN CHANGED
00166 R 540226 R            SAD PRJIGN      /TO "JMP PRIGN", THEN .INIT
00167 R 741000 A            SKP
00170 R 600615 R            JMP PRER60      /WAS NEVER DONE.

                            /THE PRECEDING TEST WAS MADE TO INSURE THAT THE HANDLER IS
                            /CONNECTED TO ITS INTERRUPT LINE BEFORE IT ISSUES IOT'S,
                            /OTHERWISE, AN ILLEGAL INTERRUPT MIGHT RESULT THAT WOULD ABORT
                            /BOTH BGD AND FGD OPERATION.

00171 R 777777 A            LAW -1
00172 R 040663 R            DAC PRIOWC      /IOPS BINARY WORD COUNT
00173 R 200215 R            LAC PRLBHP      /LINE BUFFER HEADER
00174 R 040650 R            DAC PRDBP       /DATA
00175 R 140246 R            DZM PRDTCT      /CLEAR DATA COUNT

                            /THE DATA POINTER IS NOW POINTING TO THE LINE BUFFER HEADER ADDRESS.
                            /FOR DUMP MODE (MODE 4) THERE IS NO HEADER, SO DON'T CHANGE THE DATA
                            /POINTER. FOR IOPS BINARY MODE (MODE 0) THE HEADER WORD PAIR MUST
                            /FIRST BE READ IN FROM THE TAPE, SO DON'T CHANGE THE DATA POINTER.
                            /FOR IMAGE MODES AND FOR IOPS ASCII THERE IS NO HEADER ON THE TAPE;
                            /THEREFORE, THE DATA POINTER MUST BE MOVED PAST THE HEADER WORD PAIR.

00176 R 200014 R            LAC PTRDM       /DATA MODE
00177 R 500675 R            AND (3          /CHECK FOR DUMP AND IOPS BINARY MODES.
00200 R 741200 A            SNA
00201 R 600204 R            JMP PRNXR1      /IOPS BINARY AND DUMP MODES -- DO NOT
                                            /INDEX DATA AREA POINTER.
00202 R 100562 R            JMS PRNXWD      /INDEX PAST LINE BUFFER HEADER
00203 R 100562 R            JMS PRNXWD      /FOR IOPS ASCII OR IMAGE MODES.

                            .EJECT
```

```
PAGE   8     PRA.   008     -- CAL PROCESSING -- API LEVEL 4

 190        00104 R 200014 R    PRNXR1  LAC PTRDM       /DATA MODE
 191        00105 R 340676 R            TAD (LAC PTRIOT
 192        00106 R 040107 R            DAC .+1
 193        00107 R 740040 A            XX
 194        00110 R 040153 R            DAC PRIOT
 195        00111 R 740100 A            SMA
 196        00112 R 600617 R            JMP  PRER7      /ILLEGAL DATA MODE.
 197        00113 R 777773 A            LAW -5          /5/7 CHARACTER
 198        00114 R 040031 R            DAC PTR57       /COUNTER.
 199        00115 R 140047 R            DZM PRCCT       /CLEAR CHARACTER COUNT.
 200        00116 R 140051 R            DZM PRCKSM      /CLEAR CHECKSUM
 201        00117 R 140012 R            DZM PR8CT       /CLEAR ASCII 8TH BIT SET COUNTER
 202        00120 R 140013 R            DZM PARER       /CLEAR PARITY ERROR SWITCH
 203        00121 R 140672 R            DZM PRXCES      /CLEAR "EXCESS IOPS BINARY DATA" FLAG.


PAGE   9     PRA.   008     -- COMMON EXIT -- API LEVEL 4 OR 2

 204                                .TITLE -- COMMON EXIT -- API LEVEL 4 OR 2
 205
 206                            /THE IOPS BINARY INTERRUPT SERVICE ROUTINE MAY ENTER HERE --
 207                            /CODE IN COMMON WITH THE READ CAL.
 208
 209        00122 R 160050 R    PRIOB6  DZM* PRDBP      /CLEAR NEXT DATA WORD.
 210        00123 R 777775 A    PRIOB7  LAW -3          /IOPS BINARY 3 BYTE COUNT.
 211        00124 R 040664 R            DAC PRIOBN
 212
 213                            /IF THIS WERE THE TYPE OF DEVICE THAT CAN BECOME NOT READY, THE
 214                            /RECOMMENDED CHECK POINT WOULD BE HERE. THE END-OF-MEDIUM TEST BELOW
 215                            /IS LIKE A NOT READY CHECK. WHEN A DEVICE IS DETECTED NOT READY,
 216                            /THE HANDLER REMAINS BUSY AND THE CAL FUNCTION IS NOT COMPLETED. ON
 217                            /END-OF-MEDIUM, HOWEVER, THE NOT READY INFORMATION IS RETURNED TO THE
 218                            /CALLER AND THE CAL FUNCTION IS COMPLETED. END-OF-MEDIUM IS ALSO
 219                            /CHECKED IN THE INTERRUPT SERVICE CODE.
 220
 221        00125 R 700314 A    PROUT2  IORS            /BIT 8 = 1 IN THE IORS
 222        00126 R 500677 R            AND (1000       /STATUS WORD MEANS
 223        00127 R 740200 A            SZA
 224        00130 R 600441 R            JMP PREOM       /NO-TAPE-IN-READER.
 225
 226                                .EJECT
```

```
                                /COMMON PROTECTED EXIT FOR CAL LEVEL (API LEVEL 4)
                                /AND INTERRUPT LEVEL (API LEVEL 2).

00131 R 220700 R    PRNOR   LAC* (.SCOM+102 /RAISE TO HIGHEST LEVEL ALLOWED
00132 R 705504 A            ISA             /(API 0 OR 1).
00133 R 200006 R            LAC PRA.+6      /DOES WORD 6 CONTAIN A DBR?
00134 R 540701 R            SAD (DBR
00135 R 741000 A            SKP             /YES, THIS IS AN INTERRUPT EXIT.
00136 R 600141 R            JMP .+3         /NO, THIS IS A CAL EXIT.
00137 R 200702 R            LAC (404000     /REQUEST A LEVEL 4 INTERRUPT TO
00140 R 705504 A            ISA             /ACTIVATE THE MONITOR'S DISPATCHER.
00141 R 600144 R    PRCFLG  JMP .+3         /SET TO "NOP" IF FLAGS ARE TO BE CLEARED.
00142 R 140001 R            DZM PRA.+1      /CLEAR BOTH THE FGD AND BGD
00143 R 140002 R            DZM PRA.+2      /BUSY REGISTERS -- ONLY 1 WAS SET.
00144 R 200703 R            LAC (JMP .      /RESET PRCFLG TO SKIP THE DZM'S.
00145 R 040141 R            DAC PRCFLG
00146 R 200001 R            LAC PRA.+1      /IF THE APPROPRIATE BUSY FLAG
00147 R 340002 R            TAD PRA.+2      /IS 0 (THE OTHER ALWAYS IS 0)
00150 R 751200 A            SNA!CLA         /OR IF THE STOP I/O FLAG IS
00151 R 600154 R            JMP .+3         /SET (NON-0), BYPASS THE IOT.
00152 R 540670 R            SAD PRSTPS
00153 R 740040 A    PRIOT   XX              /READER IOT.
00154 R 200053 R            LAC PTRAC       /RESTORE AC.
00155 R 703304 A            DBK             /FROM LEVEL 0 OR 1.
00156 R 400006 R            XCT PRA.+6      /ION (IF CAL EXIT) OR
                                            /DBR (IF INTERRUPT EXIT).
00157 R 620007 R            JMP* PRA.+7     /RETURN POINTER, FOR CAL EXITS,
                                            /THIS WILL RETURN TO A POINT
                                            /CALLED "CALXIT" IN THE MONITOR.
```

```
                                .TITLE -- INTERRUPT SERVICE -- API LEVEL 2

                        /READER INTERRUPT SERVICE SUBROUTINE -- ENTERED AT API LEVEL 2
                        /BY THE JMS INSTRUCTION SET UP IN API CHANNEL 10 (CORE REGISTER 50).

00160 R 000000 A    PTRINT  0               /L*P/B*MP*RETURN ADDRESS.

00161 R 707762 A            DBA             /ENABLE PAGE MODE ADDRESSING.
00162 R 040053 R            DAC PTRAC       /SAVE THE AC.
00163 R 200160 R            LAC PTRINT      /SAVE THE L*P/B*MP*PC IN
00164 R 040007 R            DAC PRA.+7      /WORD 7 (THE COMMON EXIT REGISTER).
00165 R 200701 R            LAC (DBR        /STORE A DBR INSTRUCTION
00166 R 040006 R            DAC PRA.+6      /IN WORD 6.
00167 R 700112 A            RRB             /READ READER BUFFER INTO THE
                                            /AC AND CLEAR THE READER FLAG.
00170 R 040054 R            DAC PRCHAR      /SAVE THE CHARACTER (BITS 10-17).

                        /THIS IS THE FINAL INTERRUPT EXPECTED FROM THE READER FOR THIS
                        /I/O OPERATION. THEREFORE, I/O HAS STOPPED. IN CASE IT WAS SET, CLEAR
                        /THE STOP I/O FLAG TO SIGNAL THE STOP I/O ROUTINE THAT NO FURTHER
                        /INTERRUPTS WILL OCCUR.

00171 R 140670 R            DZM PRSTPS      /CLEAR STOP SWITCH.

                        /IF THE APPROPRIATE BUSY FLAG IS ZERO (THE OTHER ONE MUST BE 0),
                        /THEN THE STOP I/O ROUTINE CLEARED IT. IF SO, IGNORE THE INTERRUPT.

00172 R 200001 R            LAC PRA.+1      /CHECK FOR STOP SINCE LAST SELECT.
00173 R 340002 R            TAD PRA.+2
00174 R 741200 A            SNA             /OK?
00175 R 600517 R            JMP PRIGN       /NO, IGNORE LAST READ.

                                .EJECT
```

```
                                /PROCESS THE INTERRUPT. IF AFTER DOING SO, THE I/O CALL IS COMPLETE.
                                /CONTROL WILL GO TO THE I/O DONE LOGIC. IF NOT, MORE I/O WILL BE
                                /INITIATED BY IOT IN THE COMMON EXIT ROUTINE.

00176 R 720314 A                        IORS            /READ IORS STATUS INTO THE AC.
00177 R 500677 R                        AND (1000       /READER OUT OF TAPE? (IORS8=1)
00200 R 740200 A                        SZA
00201 R 600441 R                        JMP PREOM       /SET EOM IN DATA BUFFER AND STOP READING.

                                /DISPATCH ON THE DATA MODE AND PROCESS THE INTERRUPT.

00202 R 200014 R                        LAC PTRDM
00203 R 340704 R                        TAD (JMP PRDIS1
00204 R 040206 R                        DAC .+2
00205 R 200054 R                        LAC PRCHAR
00206 R 740040 A                        XX              /SERVICE ACCORDING TO DATA MODE.
00207 R 600216 R        PRDIS1  JMP PRIOB       /IOPS BINARY
00210 R 600213 R                JMP PRIMB       /IMAGE BINARY
00211 R 600327 R                JMP PRIOA       /IOPS ASCII
00212 R 740000 A                NOP             /IMAGE ASCII

                                /PROCESS IMAGE BINARY OR IMAGE ASCII OR DUMP MODE.

00213 R 060050 R        PRIMB   DAC* PRDBP      /STORE CHARACTER OR BINARY
                                                /WORD IN DATA BUFFER.
00214 R 100562 R                JMS PRNXWD      /WORD COMPLETE SUBROUTINE
00215 R 600125 R                JMP PROUT2      /NEXT CHARACTER

                                .EJECT
```

```
                                /PROCESS IOPS BINARY.
                                
00216 R 100312 R    PRIOB   JMS PRPAR       /COMPUTE PARITY AND EXIT IF NULL
00217 R 744410 A            SNL!RAL!CLL     /BIT 8=1 (BINARY FRAME?)
00220 R 600125 R            JMP PROUT2      /NON-BINARY FRAME (IGNORE)
00221 R 040054 R            DAC PRCHAR      /CHARACTER IN 0-5
                                
                                /NEXT 6-BIT BYTE ROUTINE
                                
00222 R 200672 R            LAC PRXCES      /IS THIS EXCESS DATA?
00223 R 740200 A            SZA
00224 R 600305 R            JMP PRIORS      /YES, IGNORE IT.
00225 R 220050 R            LAC* PRDBP      /SHIFT EARLIER CHARACTERS LEFT
00226 R 744020 A            RAR!CLL
00227 R 240054 R            XOR PRCHAR      /ADD THIS CHARACTER TO OTHER 1 OR 2
00230 R 742010 A            RTL
00231 R 742010 A            RTL
00232 R 742010 A            RTL
00233 R 740010 A            RAL
00234 R 060050 R            DAC* PRDBP
00235 R 200665 R            LAC PRCNT1      /CHECK CHARACTER PARITY
00236 R 740020 A            RAR             /ODD IF BIT 17=1
00237 R 740400 A            SNL             /O.K.
00240 R 440013 R            IDX PARER       /PARITY ERROR-SAVE BUT PASS CHARACTER
00241 R 440012 R            IDX PRBCT       /TO FORCE A PARITY CHECK.
00242 R 440664 R            ISZ PRIOBN      /INDEX 3 BYTE IOPS BINARY COUNT.
00243 R 600125 R            JMP PROUT2      /STILL WORKING ON CURRENT DATA WORD.
00244 R 440663 R            ISZ PRIOHC      /INDEX HEADER WORD 0 COUNT.
00245 R 600264 R            JMP PRIOB4      /CURRENT DATA WORD COMPLETE.
00246 R 220050 R            LAC* PRDBP      /CHECK IF THE WORD PAIR COUNT (WPC)
00247 R 742030 A            SWHA            /IS LESS THAN THE WORD COUNT (WC).
00250 R 740010 A            RAL
00251 R 500705 R            AND (776
00252 R 040663 R            DAC PRIOHC      /TEMPORARILY STORE THE WPC.
00253 R 340016 R            TAD PTRWC
00254 R 741300 A            SPA!SNA         /WPC > WC, SHORT LINE.
00255 R 600300 R            JMP PRIOB9      /WPC INTO PTRWC, NO SHORT LINE.
                                
                                .EJECT
```

```
00256 R 200706 R            LAC (60         /WC STAYS IN PTRWC.
00257 R 100606 R            JMS PRDVS       /SET DATA VALIDITY BITS=SHORT LINE.
00260 R 777776 A            LAW -2
00261 R 340663 R            TAD PRIOWC      /SET UP PRIOWC TO SKIP EXCESS DATA
00262 R 740001 A            CMA
00263 R 040663 R            DAC PRIOWC
00264 R 220050 R    PRIOB4  LAC* PRDBP      /DATA WORD=THIS INSTRUCTION IS USED
                                            /AS A LITERAL.
00265 R 340051 R            TAD PRCKSM      /ADD TO CHECKSUM
00266 R 040051 R            DAC PRCKSM
00267 R 440050 R            IDX PRDBP       /INDEX DATA WORD POINTER
00270 R 440046 R            IDX PRDTCT      /INDEX DATA WORD COUNT
00271 R 440016 R            ISZ PTRWC       /INDEX WORD COUNT
00272 R 600122 R            JMP PRIOB6      /NEXT DATA WORD
00273 R 200663 R            LAC PRIOWC      /EXCESS DATA?
00274 R 750101 A            SMA!CLC
00275 R 600452 R            JMP PRIOBE      /NO.
00276 R 040672 R            DAC PRXCFS      /SET "EXCESS DATA" FLAG.
00277 R 600123 R            JMP PRIOB7      /RESET 3 BYTE COUNT.
00300 R 777777 A    PRIOB9  LAW -1
00301 R 340663 R            TAD PRIOWC      /WPC INTO PTRWC
00302 R 740001 A            CMA
00303 R 040016 R            DAC PTRWC
00304 R 600264 R            JMP PRIOB4

                    /COME HERE TO BYPASS EXCESS BINARY DATA WHEN THE USER LINE
                    /BUFFER IS SHORT.

00305 R 440664 R    PRIOBS  ISZ PRIOBN      /INDEX 3 BYTE COUNT
00306 R 600125 R            JMP PROUT2
00307 R 440663 R            ISZ PRIOWC      /INDEX EXCESS DATA WORD COUNT.
00310 R 600123 R            JMP PRIOB7
00311 R 600452 R            JMP PRIOBE      /END OF BINARY BLOCK

                            .EJECT
```

```
                                /SUBROUTINE PRPAR -- EXIT TO PROUT2 IF THE 8-BIT CHARACTER PRCHAR IS
                                /NULL (ZERO). IF NOT, COUNT THE NUMBER OF 1 BITS. THE SIGNIFICANT
                                /RESULT IS THE EVEN OR ODD PARITY (BIT 17) IN PRCNT1.
                                /
                                /CALLING SEQUENCE:
                                /
                                /       JMS PRPAR
                                /       (RETURN IF CHARACTER IS NON-0)

00312 R 000000 A    PRPAR   0
00313 R 777770 A            LAW -10         /PARITY COUNTER (-8)
00314 R 040052 R            DAC PRCNT
00315 R 140665 R            DZM PRCNT1
00316 R 200054 R            LAC PRCHAR
00317 R 741200 A            SNA
00320 R 600125 R            JMP PROUT2      /NULL
00321 R 740020 A            RAR
00322 R 741400 A            SZL
00323 R 440665 R            IDX PRCNT1      /1 BIT COUNTER
00324 R 440052 R            ISZ PRCNT
00325 R 600321 R            JMP .-4
00326 R 620312 R            JMP* PRPAR

                            .EJECT
```

```
                                /PROCESS IOPS ASCII
                
00327 R 200016 R    PRIOA   LAC PTRWC       /SEE IF EXCESS DATA
00330 R 740100 A            SMA
00331 R 600405 R            JMP PRASE3      /YES
00332 R 200054 R            LAC PRCHAR
00333 R 500707 R            AND (177
00334 R 540710 R            SAD (12
00335 R 600125 R            JMP PROUT2      /IGNORE LINE FEED
00336 R 540711 R            SAD (13
00337 R 600125 R            JMP PROUT2      /IGNORE VERTICAL TAB
00340 R 540712 R            SAD (14
00341 R 600125 R            JMP PROUT2      /IGNORE FORM FEED
00342 R 100312 R            JMS PRPAR       /COMPUTE PARITY AND EXIT IF NULL
00343 R 741400 A            SZL
00344 R 440012 R            IDX PR8CT       /8TH BIT=1, ADD TO COUNT
00345 R 440047 R            IDX PRCCT
00346 R 200665 R            LAC PRCNT1      /PARITY COUNT=SHOULD BE EVEN
00347 R 740020 A            RAR
00350 R 741400 A            SZL
00351 R 440013 R            IDX PARER       /NOT EVEN PARITY
00352 R 100425 R            JMS PRENDT      /CONVERT ALTMODES
00353 R 500707 R            AND (177        /DROP ALL BUT 7 BITS
00354 R 540707 R            SAD (177        /DELETE CODE (RUBOUT)=IGNORE
00355 R 600125 R            JMP PROUT2
00356 R 100522 R            JMS PRPK57      /PACK INTO LINE BUFFER IN 5/7
00357 R 100425 R            JMS PRENDT
00360 R 740100 A            SMA
00361 R 600125 R            JMP PROUT2      /NEXT ASCII CHARACTER
00362 R 200047 R            LAC PRCCT
00363 R 540713 R            SAD (1
00364 R 600124 R            JMP PRNXR1      /IGNORE SINGLE CARRIAGE RETURN LINE
00365 R 100570 R            JMS PRPAD       /PAD LAST DATA WORD PAIR

                            .EJECT
```

```
                                /END OF IOPS ASCII, IMAGE ASCII, IMAGE BINARY, AND
                                /DUMP MODE LINE.
                                
00366 R 200014 R    PRASE   LAC PTRDM
00367 R 540714 R            SAD (4
00370 R 600472 R            JMP PRIORB          /DUMP MODE.
00371 R 100577 R            JMS PRHEAD          /SET UP L.B.H 0, W.P.C.+D.M.
00372 R 220012 R            LAC PR8CT           /DID ALL CHAR'S HAVE BIT 8
00373 R 540047 R            SAD PRCCT           /NO - IOPS ASCII CHECK PARITY
00374 R 600401 R            JMP PRASE4          /YES - ASSUME NON IOPS ASCII
00375 R 220013 R            LAC PARER           /PARITY ERROR
00376 R 740200 A            SZA                 /NO
00377 R 200715 R            LAC (20             /YES
00400 R 100606 R            JMS PRDVS           /PARITY ERROR SET VALIDITY BITS
00401 R 200014 R    PRASE4  LAC PTRDM
00402 R 240716 R            XOR (2              /IOPS ASCII
00403 R 740200 A            SZA
00404 R 600472 R            JMP PRIORB          /END LINE
                                
                                /IOPS ASCII EXCESS DATA TO BE IGNORED.
                                
00405 R 100425 R    PRASE3  JMS PRENDT          /SKIP TO END LINE
00406 R 751101 A            SPA!CLA!CMA
00407 R 600472 R            JMP PRIORB
00410 R 340050 R            TAD PRDBP
00411 R 040050 R            DAC PRDBP           /POINTS TO LAST CHARACTER
00412 R 777400 A            LAW 17400
00413 R 520050 R            AND* PRDBP
00414 R 240717 R            XOR (33             /PUT CARRIAGE RETURN IN LAST WORD PAIR
00415 R 060050 R            DAC* PRDBP
00416 R 440050 R            IDX PRDBP           /IN CASE MORE BEFORE CARRIAGE RETURN
00417 R 220015 R            LAC* PRLBHP
00420 R 500706 R            AND (60             /IF DATA VALIDITY BITS HAVEN'T ALREADY
00421 R 240706 R            XOR (60             /BEEN SET, SET TO INDICATE SHORT LINE.
00422 R 540706 R            SAD (60
00423 R 100606 R            JMS PRDVS
00424 R 600125 R            JMP PROUT2
                                
                                .EJECT
```

```
                        /SUBROUTINE PRENDT -- CALLED FOR IOPS ASCII. IF AN ALTMODE IS IN
                        /PRCHAR, IT IS CHANGED (IN THE AC) FROM 33 OR 176 TO THE STANDARD
                        /175. IN ADDITION, IF THE CHARACTER IS CARRIAGE RETURN OR ALTMODE,
                        /BOTH LINE TERMINATORS, THE NUMBER IS SET NEGATIVE SO THAT IT CAN
                        /BE TESTED ON RETURN.
                        /
                        /CALLING SEQUENCE:
                        /
                        /       JMS PRENDT
                        /       (RETURN)

00425 R 000000 A    PRENDT  0
00426 R 200054 R            LAC PRCHAR
00427 R 500707 R            AND (177
00430 R 540720 R            SAD (15         /RETURN
00431 R 760015 A            LAW 15
00432 R 540721 R            SAD (175        /ALTMODE
00433 R 760175 A            LAW 175
00434 R 540722 R            SAD (176        /ALTMODE
00435 R 760175 A            LAW 175
00436 R 540717 R            SAD (33         /ESCAPE
00437 R 760175 A            LAW 175
00440 R 620425 R            JMP* PRENDT
```

```
513                                     .TITLE -- I/O DONE LOGIC -- API LEVEL 4 OR 2
514
515                             /END OF PAPER TAPE ROUTINE.
516
517     00441 R 200014 R    PREOM   LAC PTRDM       /CHECK DATA MODE
518     00442 R 540714 R            SAD (4          /DUMP MODE HAS NO HEADER;
519     00443 R 600472 R            JMP PRIOBB      /THEREFORE, IGNORE END-OF-MEDIUM.
520     00444 R 540716 R            SAD (2
521     00445 R 100570 R            JMS PRPAD       /IOPS ASCII-PAD LAST WORD PAIR.
522     00446 R 100577 R            JMS PRHEAD      /SET UP WORD 0 OF HEADER.
523     00447 R 500723 R            AND (777760     /MASK ALL BUT MODE BITS,
524     00450 R 240724 R            XOR (6          /AND SET THE MODE BITS TO
525     00451 R 060015 R            DAC* PRLBHP     /INDICATE END-OF-MEDIUM.
526     00452 R 200012 R    PRIOBE  LAC PR8CT       /IOPS BINARY WILL
527     00453 R 540047 R            SAD PRCCT       /ALWAYS CHECK
528     00454 R 600460 R            JMP PRIOB5      /PARITY.
529     00455 R 200013 R            LAC PARER
530     00456 R 750200 A            SZA!CLA
531     00457 R 600470 R            JMP PRIOBP      /PARITY ERROR(S)
532     00460 R 220015 R    PRIOB5  LAC* PRLBHP     /WORD 0 OF LINE BUFFER HEADER.
533     00461 R 500706 R            AND (60
534     00462 R 740200 A            SZA
535     00463 R 600472 R            JMP PRIOBB      /VALIDITY BITS ALREADY SET
536     00464 R 200051 R            LAC PRCKSM
537     00465 R 741200 A            SNA
538     00466 R 600472 R            JMP PRIOBB      /NORMAL END OF LINE
539     00467 R 200715 R            LAC (20         /CHECKSUM ERROR -- SET DATA
540                                                 /VALIDITY BITS (12-13) = 10.
541     00470 R 340715 R    PRIOBP  TAD (20         /PARITY ERROR -- SET DATA
542                                                 /VALIDITY BITS (12-13) = 01.
543     00471 R 100606 R            JMS PRDVS       /SET DATA VALIDITY BITS.
544
545                                     .EJECT
```

```
                                /IS THIS DEVICE INVOLVED IN AN I/O BUSY SITUATION? THE IOBUSY
                                /ROUTINE IN THE MONITOR WILL ANSWER THAT QUESTION AND WILL CLEAR
                                /THE CONDITION BY REACTIVATING THE FOREGROUND LEVELS THAT WERE BUSY.

00472 R 220725 R    PRIOBB  LAC* (.SCOM+52  /ADDRESS OF THE MONITOR'S
00473 R 040671 R            DAC PRTMP1      /I/O BUSY TESTER.
00474 R 220700 R            LAC* (.SCOM+102 /RAISE TO LEVEL 0 OR 1.
00475 R 705504 A            ISA
00476 R 200726 R            LAC (PRA.       /(WORD 0.
00477 R 120671 R            JMS* PRTMP1
00500 R 703304 A            DBK             /FROM LEVEL 0 OR 1.

                            /CHECK IF THE COMPLETED I/O CAL IS A REAL-TIME REQUEST.

00501 R 200017 R            LAC PRA.+17     /NON-0 IF REAL-TIME.
00502 R 741200 A            SNA
00503 R 600517 R            JMP PRIGN       /NOT .REALR.
00504 R 200727 R            LAC (JMP PREAD
00505 R 540010 R            SAD PRA.+10     /JMP FUNCTION
00506 R 741000 A            SKP             /.REALR.
00507 R 600517 R            JMP PRIGN       /NOT .REALR.
00510 R 220730 R            LAC* (.SCOM+51  /ADDRESS OF THE MONITOR'S
00511 R 040671 R            DAC PRTMP1      /REAL TIME PROCESSOR.
00512 R 220700 R            LAC* (.SCOM+102 /RAISE TO LEVEL 0 OR 1.
00513 R 705504 A            ISA
00514 R 200017 R            LAC PRA.+17     /CALL REALTP TO QUEUE THE
00515 R 120671 R            JMS* PRTMP1     /REAL-TIME SUBROUTINE.
00516 R 703304 A            DBK

                            /SET UP TO CLEAR THE BUSY FLAG DURING PROTECTED EXIT. THIS IS
                            /DONE BECAUSE I/O HAS COMPLETED, BECAUSE I/O HAS BEEN ABORTED
                            /(STOPIO), BECAUSE THIS IS AN IGNORED CAL FUNCTION, OR BECAUSE
                            /THIS CAL FUNCTION WAS IGNORED DUE TO AN ERROR.

00517 R 200731 R    PRIGN   LAC (NOP
00520 R 040141 R            DAC PRCFIG

00521 R 600131 R            JMP PRNOR
```

```
                                        .TITLE -- MISCELLANEOUS SUBROUTINES -- API LEVEL 4 OR 2

                                /SUBROUTINE PRPK57 -- PACKS 5/7 IOPS ASCII DATA, PRIOR TO THE FIRST
                                /CALL, PTR57 MUST BE INITIALIZED TO -5.
                                /
                                /CALLING SEQUENCE:
                                /
                                /       CHARACTER IN THE AC
                                /       IN BITS 11 THROUGH 17
                                /       JMS PRPK57
                                /       (CONDITIONAL RETURN)
                                /
                                /RETURN WILL NOT OCCUR IF THE WORD COUNT IS EXHAUSTED.

00522 R 000000 A    PRPK57  0               /CHARACTER IN AC BITS 11-17.
00523 R 742030 A            SWHA            /MOVE TO AC BITS 0-6
00524 R 742010 A            RTL
00525 R 040052 R            DAC PRTMP
00526 R 777771 A            LAW -7
00527 R 040665 R            DAC PRLPCT
00530 R 200052 R    PRPKBK  LAC PRTMP       /ROTATE CHAR LEFT
00531 R 740010 A            RAL             /7 BITS THROUGH
00532 R 040052 R            DAC PRTMP       /THE DOUBLE WORD
00533 R 200667 R    PRBCK2  LAC PRRTHF      /ACCUMULATOR
00534 R 740010 A            RAL             /PRLFHF /PRRTHF.
00535 R 040667 R            DAC PRRTHF
00536 R 200666 R            LAC PRLFHF
00537 R 740010 A            RAL
00540 R 040666 R            DAC PRLFHF
00541 R 200665 R            LAC PRLPCT
00542 R 745200 A            SNA!CLL
00543 R 600551 R            JMP PRPDNE      /2 WORDS ALL SET.
00544 R 440665 R            ISZ PRLPCT      /IS 7 TIMES COUNT EXHAUSTED?
00545 R 600530 R            JMP PRPKBK      /NO.
00546 R 440031 R            ISZ PTR57       /DO WE HAVE 5 CHARACTERS?
00547 R 620522 R            JMP* PRPK57     /NO, EXIT
00550 R 600533 R            JMP PRBCK2      /YES, SHIFT LEFT ONCE MORE.

                                        .EJECT
```

```
00551 R 200666 R     PRPONE  LAC PRLFWF       /PLACE ACCUMULATED
00552 R 060050 R             DAC* PRDBP       /2 WORDS INTO
00553 R 100562 R             JMS PRNXWD       /USERS LINE BUFFER,
00554 R 200667 R             LAC PRRTWF       /UPDATING POINTERS,
00555 R 060050 R             DAC* PRDBP
00556 R 100562 R             JMS PRNXWD
00557 R 777773 A             LAW -5           /RESET 5 CHARACTER COUNTER,
00560 R 040031 R             DAC PTR57
00561 R 620522 R             JMP* PRPK57
                             .EJECT
```

```
                                /SUBROUTINE PRNXWD -- INDEXES DATA POINTER AND WORD COUNTS.
                                /
                                /CALLING SEQUENCE:
                                /
                                /       JMS PRNXWD
                                /       (CONDITIONAL RETURN)
                                /
                                /RETURN WILL NOT OCCUR IF THE WORD COUNT IS EXHAUSTED.

00562 R 000000 A    PRNXWD  0
00563 R 440050 R            IDX PRDBP       /INDEX TO NEXT DATA WORD.
00564 R 440046 R            IDX PRDTCT      /ADD TO DATA WORD COUNT.
00565 R 440016 R            ISZ PTRWC       /WORD COUNT EXHAUSTED?
00566 R 620562 R            JMP* PRNXWD     /NO, EXIT FOR NEXT CHARACTER.
00567 R 620366 R            JMP PRASE       /EXIT TO END OF IOPS ASCII LINE ROUTINE.

                                /SUBROUTINE PRPAD -- PADS LAST WORD PAIR (IN IOPS ASCII) WITH NULL
                                /(ZERO) CHARACTERS.
                                /
                                /CALLING SEQUENCE:
                                /
                                /       JMS PRPAD
                                /       (RETURN)

00570 R 000000 A    PRPAD   0
00571 R 777773 A    PRPAGN  LAW -5          /IS LAST WORD
00572 R 540031 R            SAD PTR57       /PAIR FULL?
00573 R 620570 R            JMP* PRPAD      /YES, EXIT.
00574 R 752000 A            CLA             /NO, INSERT ANOTHER
00575 R 100522 R            JMS PRPK57      /NULL CHARACTER.
00576 R 620571 R            JMP PRPAGN

                                .EJECT
```

```
                                /SUBROUTINE PRHEAD -- SETS UP THE LINE BUFFER HEADER (L,B,H,) WORD 0
                                /WITH THE WORD PAIR COUNT AND THE DATA MODE.
                                /
                                /CALLING SEQUENCE:
                                /
                                /       JMS PRHEAD
                                /       (RETURN)

00577 R 000000 A    PRHEAD  0
00600 R 200046 R            LAC PRDTCT      /WORD COUNT (INCLUDES L,B,H,)
00601 R 742030 A            SWHA
00602 R 744020 A            RCR             /WORD PAIR COUNT (BITS 1-8),
00603 R 240014 R            XOR PTRDM       /DATA MODE.
00604 R 060015 R            DAC* PRLBHP
00605 R 620577 R            JMP* PRHEAD

                                /SUBROUTINE PRDVS -- SETS DATA VALIDITY (D.V.) BITS IN THE
                                /LINE BUFFER HEADER WORD 0.
                                /
                                /CALLING SEQUENCE:
                                /
                                /       DATA VALIDITY BITS (12,13)
                                /       SET TO DESIRED VALUE IN THE AC
                                /       JMS PRDVS
                                /       (RETURN)

00606 R 000000 A    PRDVS   0
00607 R 040051 R            DAC PRDVS2      /SAVE DATA VALIDITY BITS.
00610 R 777717 A            LAW 17717       /MASK ALL BITS IN THE LINE BUFFER HEADER WORD
00611 R 520015 R            AND* PRLBHP     /0, EXCEPT IGNORE CHECKSUM BIT (0) AND D.V.
00612 R 240051 R            XOR PRDVS2      /BITS (12,13), ADD IN PARITY ERROR, CHECKSUM
00613 R 060015 R            DAC* PRLBHP     /ERROR, OR SHORT LINE.
00614 R 620606 R            JMP* PRDVS
```

```
700                                     .TITLE -- ERRORS -- API LEVEL 4
701
702                                 /ERROR ROUTINE -- ALL READER ERRORS ARE DETECTED AT THE CAL LEVEL.
703
704     00615 R 766060 A    PRER60  LAW 6060        /.INIT NOT PERFORMED.
705     00616 R 741000 A            SKP
706     00617 R 766007 A    PRER7   LAW 6007        /ILLEGAL DATA MODE.
707     00620 R 741000 A            SKP
708     00621 R 766006 A    PRER6   LAW 6006        /ILLEGAL CAL FUNCTION.
709     00622 R 040637 R            DAC PRERoR      /TEMPORARY SAVE.
710     00623 R 200011 R            LAC PRA,+11     /WHOSE CAL: 0=FGD: 1=BGD.
711     00624 R 740020 A            RAR
712     00625 R 200637 R            LAC PRERoR
713     00626 R 7404 0 A            SNL             /IS IT FGD?
714     00627 R 240732 R            XOR (3000       /YES. CHANGE BGD ERROR TO FGD ERROR.
715     00630 R 040637 R            DAC PRERoR
716     00631 R 200013 R            LAC PARER       /PUT CAL ADDRESS IN THE AUXILIARY
717     00632 R 040641 R            DAC PRAUX       /ARGUMENT FOR PRINTOUT.
718     00633 R 220733 R            LAC* (,SCOM+66  /ADDRESS OF THE MONITOR'S
719     00634 R 040671 R            DAC PRTMP1      /ERROR QUEUER.
720     00635 R 220700 R            LAC* (,SCOM+102
721     00636 R 705504 A            ISA
722     00637 R 740040 A    PRERoR  XX              /LAW ERRoR # + JOB AT FAULT.
723     00640 R 120671 R            JMS* PRTMP1
724     00641 R 740040 A    PRAUX   XX              /AUXILIARY ARGUMENT.
725     00642 R 703304 A            DBK             /BACK TO LEVEL 4.
726     00643 R 600517 R            JMP PRIGN       /IGNORE THE CAL BY CLEARING
727                                                 /THE BUSY REGISTER.
```

```
                                .TITLE -- STOPIO SUBROUTINE -- MAINSTREAM

                        /STOP I/O SUBROUTINE -- OPERATING AT MAINSTREAM BY VIRTUE OF A CALL
                        /FROM WITHIN THE FOREGROUND OR BACKGROUND ERROR ROUTINE IN THE
                        /MONITOR. THE MONITOR WILL CALL THIS ROUTINE ONLY IF THE APPRO-
                        /PRIATE JOB BUSY FLAG (WORD 1 OR 2) INDICATES THAT I/O IS UNDER
                        /WAY. I/O CANNOT BE STOPPED BY ISSUING AN IOT; THEREFORE, A WAIT
                        /LOOP MAY NEED TO BE EXECUTED UNTIL THE FINAL INTERRUPT HAS OCCURRED.

00644 R 000000 A    PRSTOP  0
00645 R 220700 R            LAC* (.SCOM+102 /RAISE TO LEVEL 0 OR 1.
00646 R 705004 A            ISA
00647 R 200001 R            LAC PRA,+1
00650 R 340002 R            TAD PRA,+2
00651 R 040670 R            DAC PRSTPS      /SET STOP SWITCH ACCORDING TO BUSY REGISTER.
00652 R 140001 R            DZM PRA,+1      /CLEAR BUSY FLAGS.
00653 R 140002 R            DZM PRA,+2
00654 R 140003 R            DZM PRA,+3      /CLEAR CLOSE SWITCHES.
00655 R 140004 R            DZM PRA,+4
00656 R 703304 A            DBK             /DEBREAK BACK TO MAINSTREAM.
00657 R 200670 R            LAC PRSTPS      /WAIT LOOP. FALL THROUGH
00660 R 740200 A            SZA             /WHEN NO FURTHER INTERRUPTS
00661 R 600657 R            JMP .-2         /WILL OCCUR.
00662 R 620644 R            JMP* PRSTOP     /RETURN TO THE MONITOR'S ERROR ROUTINE.

                        /TEMPORARY REGISTERS.

00663 R 000000 A    PRIOWC  0               /IOPS BIN, WD1 COUNT
00664 R 000000 A    PRIOBN  0               /IOPS BIN 3 BYTE COUNT
00665 R 000000 A    PRCNT1  0               /1 BIT COUNTER FOR PARITY CHECK
        000052 R    PRTMP=PRCNT             /TEMP. STORAGE FOR 5/7 CHAR.
        000665 R    PRLPCT=PRCNT1           /ROTATE 7 BITS COUNTER.
00666 R 000000 A    PRLFHF  0               /2 WORD ACCUMULATOR FOR
00667 R 000000 A    PRRTHF  0               /5/7 WORD PAIR.
00670 R 000000 A    PRSTPS  0               /STOP UNDERWAY SWITCH
00671 R 000000 A    PRTMP1  0               /GENERAL TEMPORARY REGISTER.
00672 R 000000 A    PRXCES  0               /IOPS BINARY EXCESS DATA FLAG, 0 = DATA
                                            /O.K.; NON-0 = IGNORE EXCESS DATA.
```

```
 766                                            .TITLE -- LITERALS
 767
 768                    000000 A                .END
            00673 R 000064 A *L
            00674 R 000155 A *L
            00675 R 000003 A *L
            00676 R 200055 R *L
            00677 R 001000 A *L
            00700 R 000202 A *L
            00701 R 703344 A *L
            00702 R 404000 A *L
            00703 R 600144 R *L
            00704 R 600207 R *L
            00705 R 000776 A *L
            00706 R 000060 A *L
            00707 R 000177 A *L
            00710 R 000012 A *L
            00711 R 000013 A *L
            00712 R 000014 A *L
            00713 R 000001 A *L
            00714 R 000004 A *L
            00715 R 000020 A *L
            00716 R 000002 A *L
            00717 R 000033 A *L
            00720 R 000015 A *L
            00721 R 000175 A *L
            00722 R 000176 A *L
            00723 R 777760 A *L
            00724 R 000006 A *L
            00725 R 000152 A *L
            00726 R 000000 R *L
            00727 R 600027 R *L
            00730 R 000151 A *L
            00731 R 740000 A *L
            00732 R 003000 A *L
            00733 R 000166 A *L
                SIZE=00734     NO ERROR LINES
```

IDX	440000	A	PARER	00013	R	PRASE	00366	R	PRASE3	00405	R
PRASE4	00401	R	PRAUX	00641	R	PRA.	00000	R	PRBCK2	00533	R
PRCCT	00047	R	PRCFLG	00141	R	PRCHAR	00054	R	PRCKSM	00051	R
PRCNT	00052	R	PRCNT1	00665	R	PRDBP	00050	R	PRDIS1	00207	R
PRDTCT	00046	R	PRDVS	00606	R	PRDVS2	000051	R	PREAD	00027	R
PRENDT	00425	R	PREOM	00441	R	PREROR	00637	R	PRER6	00621	R
PRER60	00615	R	PRER7	00617	R	PRHEAD	00577	R	PRIGN	00517	R
PRIMB	00213	R	PRIN	00043	R	PRIOA	00327	R	PRIOB	00216	R
PRIOBB	00472	R	PRIOBE	00452	R	PRIOBN	00664	R	PRIOBP	00470	R
PRIOBS	00305	R	PRIOB4	00264	R	PRIOB5	00460	R	PRIOB6	00122	R
PRIOB7	00123	R	PRIOB9	00300	R	PRIOHC	00663	R	PRIOT	00153	R
PRJIGN	00026	R	PRLBHP	00015	R	PRLFHF	00666	R	PRLPCT	000665	R
PRNIT	00045	R	PRNOR	00131	R	PRNXR1	00104	R	PRNXWD	00562	R
PROUT2	00125	R	PRPAD	00570	R	PRPAGN	00571	R	PRPAR	00312	R
PRPDNE	00551	R	PRPKBK	00530	R	PRPK57	00522	R	PRRED	00065	R
PRRTHF	00667	R	PRSTOP	00644	R	PRSTPS	00670	R	PRSWAP	00037	R
PRTMP	000052	R	PRTMP1	00671	R	PRXCES	00672	R	PR8CT	00012	R
PTRAC	00053	R	PTRDM	00014	R	PTRINT	00160	R	PTRIOT	00055	R
PTRWC	00014	R	PTR57	00031	R	RRB	700112	A	RSA	700104	A
RSB	700144	A	RSF	700101	A	.SCOM	000100	A			

PRA.	00000	R	PR8CT	00012	R	PARER	00013	R	PTRDM	00014	R
PRLBHP	00015	R	PTRWC	00016	R	PRJIGN	00026	R	PREAD	00027	R
PTR57	00031	R	PRSWAP	00037	R	PRIN	00043	R	PRNIT	00045	R
PRDTCT	00046	R	PRCCT	00047	R	PRDBP	00050	R	PRCKSM	00051	R
PRDVS2	000051	R	PRCNT	00052	R	PRTMP	000052	R	PTRAC	00053	R
PRCHAR	00054	R	PTRIOT	00055	R	PRRED	00065	R	.SCOM	000100	A
PRNXR1	00104	R	PRIOB6	00122	R	PRIOB7	00123	R	PROUT2	00125	R
PRNOR	00131	R	PRCFLG	00141	R	PRIOT	00153	R	PTRINT	00160	R
PRDIS1	00207	R	PRIMB	00213	R	PRIOB	00216	R	PRIOB4	00264	R
PRIOB9	00300	R	PRIOB5	00305	R	PRPAR	00312	R	PRIOA	00327	R
PRASE	00366	R	PRASE4	00401	R	PRASE3	00405	R	PRENDT	00425	R
PREOM	00441	R	PRIOBE	00452	R	PRIOB5	00460	R	PRIOBP	00470	R
PRIOBB	00472	R	PRIGN	00517	R	PRPK57	00522	R	PRPKBK	00530	R
PRBCK2	00533	R	PRPDNE	00551	R	PRNXWD	00562	R	PRPAD	00570	R
PRPAGN	00571	R	PRHEAD	00577	R	PRDVS	00606	R	PRER60	00615	R
PRER7	00617	R	PRER6	00621	R	PREROR	00637	R	PRAUX	00641	R
PRSTOP	00644	R	PRIOHC	00663	R	PRIOBN	00664	R	PRCNT1	00665	R
PRLPCT	000665	R	PRLFHF	00666	R	PRRTHF	00667	R	PRSTPS	00670	R
PRTMP1	00671	R	PRXCES	00672	R	IDX	440000	A	RSF	700101	A
RSA	700104	A	RRB	700112	A	RSB	700144	A			

```
IDX     440000   47*  341   342   367   368   410   431   432   436
                 481   644   645
PARER    00013    74*  202   341   436   461   529   716
PRASE    00366   454*  648
PRASE3   00405   420   472*
PRASE4   00401   460   465*
PRAUX    00641   717   724*
PRA.     00000    55    59*  114   126   232   239   240   243   244
                 251   253   266   268   283   284   554   560   564
                 571   710   740   741   743   744   745   746
PRBCK2   00533   607*  620
PRCCT    00047   135*  199   432   445   459   527
PRCFLG   00141   238*  242   581
PRCHAR   00054   141*  271   303   323   332   405   421   502
PRCKSM   00051   137*  138   200   365   366   536
PRCNT    00052   139*  403   411   758
PRCNT1   00665   338   404   410   433   757*  759
PRDBP    00050   136*  171   209   312   330   337   347   363   367
                 475   476   478   480   481   624   627   644
PRDIS1   00207   301   305*
PRDTCT   00046   134*  172   368   645   676
PRDVS    00606   358   464   486   543   693*  699
PRDVS2   000051  138*  694   697
PREAD    00027    92*  563
PRENDT   00425   437   442   472   501*  512
PREOM    00441   224   296   517*
PRERDR   00637   709   712   715   722*
PRER6    00621    88    89    93    99   708*
PRER60   00615   161   704*
PRER7    00617   152   153   154   196   706*
PRHEAD   00577   457   522   675*  681
PRIGN    00517    86    87    90    91   141   286   562   566   580*
```

Continued on next page.

		726								
PRIMB	00213	306	312*							
PRIN	00043	85	125*							
PRIOA	00327	307	418*							
PRIOB	00216	305	320*							
PRIOBB	00472	456	468	474	519	535	538	550*		
PRIOBE	00452	373	389	526*						
PRIOBN	00664	211	343	385	756*					
PRIOBP	00470	531	541*							
PRIOBS	00305	329	385*							
PRIOB4	00264	346	363*	380						
PRIOB5	00460	528	532*							
PRIOB6	00122	209*	370							
PRIOB7	00123	210*	375	388						
PRIOB9	00300	354	376*							
PRIOHC	00663	169	345	351	360	362	371	377	387	755*
PRIOT	00153	194	248*							
PRJIGN	00026	91*	159							
PRLBHP	00015	76*	170	482	525	532	680	696	698	
PRLFHF	00666	610	612	623	760*					
PRLPCT	00665	603	613	616	759*					
PRNIT	00045	133*	140	158						
PRNOR	00131	230*	583							

```
PRNXR1   00104   184   190*  447
PRNXWD   00562   186   187   314   625   628   643*  647
PROUT2   00125   221*  315   322   344   386   407   424   426   428
                 440   444   487
PRPAD    00570   448   521   658*  661
PRPAGN   00571   659*  664
PRPAR    00312   320   401*  413   429
PRPDNE   00551   615   623*
PRPKBK   00530   604*  617
PRPK57   00522   441   598*  619   631   663
PRRED    00065    92   158*
PRRTHF   00667   607   609   626   761*
PRSTOP   00644   102   103   737*  751
PRSTPS   00670   247   278   742   748   762*
PRSWAP   00037    59   113*  116
PRTMP    000052  601   604   606   758*
PRTMP1   00671   551   555   568   572   719   723   763*
PRXCES   00672   203   327   374   764*
PR8CT    00012    73*  134   135   201   342   431   458   526
PTRAC    00053   143*  249   264
PTRDM    00014    75*  181   190   300   454   465   517   679
PTRINT   00163   137   261*  265
PTRIOT   00055   147*  191
PTRWC    00016    77*  352   369   379   418   646
PTR57    00031    94*  198   618   630   660
RRB      700112   31*  269
RSA      700104   33*  147   149   150
RSB      700144   37*  148   151
RSF      700101   30*  136
.SCOM    000100   45*  133   230   550   552   567   569   718   720
                738
```

SECTION 8

SYSTEM GENERATION

8.1 INTRODUCTION

Master PDP-15 Background/Foreground Monitor systems are supplied to customers in an executable binary form on a single DECtape. The Master system represents an immediately usable system; however, it is recommended that the user perform a system generation procedure using the Master system, a system generator program, and command inputs to produce a working Background/Foreground Monitor system conforming to the user's specific equipment and software needs. The required system generator program is listed on the Master system under the file name "BFSGEN SYS".

8.1.1 BFSGEN, Generation and Update Features

The program BFSGEN can perform two optional functions:

1) SYSTEM GENERATION. - This function enables the transfer of a system of selected software from the Master system to some intermediate, file structured, mass storage device (DECtape or Disk). Normally, the transferred system will not conform to the requirements of the installation on which it is to be used; however, it is in a form which is easily modified (i.e., updated).
2) SYSTEM UPDATE. - The update function of BFSGEN enables the user to modify both hardware and software parameters of a generated system. This feature must be used when building the initial system. Update may also be used to modify a previously generated system to meet new hardware or software requirements.

8.2 BFSGEN DEVICE REQUIREMENTS

System generation operations must be performed between two similar I/O devices (e.g., DECtape to DECtape); dissimilar devices (e.g., DECtape to DISK) cannot be used.

8.2.1 DECtape Masters

Both Disk and DECtape system Masters are supplied on DECtape. For both types of Masters, system generation may be performed directly from the Master DECtape to a system DECtape.

For an installation with a Disk, the user may first transfer the Master system from DECtape onto a Disk unit, using utility program PIP or RFSAV (refer to PDP-15 Utility Manual DEC-15-YWZA-D). An example of the required PIP command string follows:

```
C DK4 ←DT3 (H)↵
```

System generation is then performed between Disk units. The resulting system should be copied onto DECtape for backup purposes, using either PIP or RFSAV (refer to Utility manual).

8.2.2 Loading BFSGEN

BFSGEN uses the following .DAT slots:

.DAT slot	Used for...
-14	> Input from Master tape
-15	a) Output of a generated system b) Input/Output of system to be updated.
-3	> Teletype output
-2	> Teletype input

Initially, one must run the system supplied on the Master tape since this is the only way the System Generator program can be loaded and run. The following two steps must first be performed:

1. If the tape is a Disk Master, the system must be installed onto Disk unit zero from the Master tape by using the utility program RFSAV (refer to the PDP-15 Utility Manual, DEC-15-YWZA-D) or by performing a COPY using PIP running under the Advanced Monitor System.
2. Then, whether the operating system resides on DECtape or on Disk, follow the procedures in paragraphs 5.2.1 and 5.3.1 to bring in the Monitor and to load IDLE in the Foreground.

The remaining steps indicate how to load the System Generator in the Background. This is similar to Advanced Monitor System operation.

3. When the Monitor is ready to accept commands on the Background control Teletype, it prints:

```
BKM15 V3A
$
```

4. If the Master tape is on DECtape transport 4 and a scratch tape (onto which a generated system is to be output) is on DECtape transport 5, for example, perform the following:

```
$ASSIGN DT4  -14
$ASSIGN DT5  -15
$BFSGEN
```

When the System Generator is loaded and running, it will print out:

```
B/F-15 SGEN Vxx
```

8.3 SYSTEM GENERATION PROCEDURES

The System Generator consists of modular sections of code. The operation of each is discussed individually in the following paragraphs.

8.3.1 Section A -- Initialization

Summary: The user is asked if he would like to run in brief mode. If he answers no, a summary of System Generator rules is typed out.

Operation: When the System Generator has been loaded into core and is started, it will type:

```
BF-15 SGEN Vnn
BRIEF MODE? (Y/N)
```

In the brief mode the System Generator will abridge the messages it types out. The user must respond by typing Y (meaning YES) or N (meaning NO). If YES, the remainder of Section A is bypassed and processing goes to Section B. If NO, the System Generator types out a summary of operating rules and from then on all messages are printed in expanded form.

The summary of operating rules is typed out as follows:

```
.DAT SLOT ASSIGNMENTS:

WHEN GENERATING A NEW SYSTEM, A B/F MASTER MUST BE ON
THE DEVICE ASSIGNED TO .DAT SLOT -14.  A NEW SYSTEM WILL
BE BUILT ON THE DEVICE ASSIGNED TO .DAT SLOT -15.

WHEN UPDATING AN OLD SYSTEM, THAT SYSTEM MUST BE ON THE
DEVICE ASSIGNED TO .DAT SLOT -15.

KEY-IN CONVENTIONS:

STATEMENTS THAT REQUIRE A RESPONSE END IN EITHER A
QUESTION MARK (?), AN ANGLE BRACKET (>), OR A SQUARE
BRACKET (]).

QUESTION MARK (?)-- A YES OR NO ANSWER IS REQUIRED.  TYPE
'Y' FOR YES OR 'N' FOR NO.  IN BRIEF MODE, A QUESTION MARK
MAY BE TYPED TO CAUSE THE QUESTION TO BE RESTATED IN AN
EXPANDED FORM.

ANGLE BRACKET (>)-- A PARAMETER (NAME, SET OF NAMES,
OR A VALUE) IS REQUIRED.  TYPE THE PARAMETER AND TERMINATE
WITH A CARRIAGE RETURN. **

SQUARE BRACKET (]) --A PARAMETER HAS BEEN TYPED OUT
ENCLOSED IN SQUARE BRACKETS ([ ]).  THE PARAMETER MAY BE
ACCEPTED AS IS BY TYPING A CARRIAGE RETURN, OR IT MAY BE
ALTERED BY RETYPING THE ENTIRE PARAMETER TERMINATED BY A
CARRIAGE RETURN. **
```

RESTART:

TYPING A ↑P WILL RESTART THE SYSTEM GENERATOR. THIS SHOULD ONLY BE DONE WHILE BFSGEN IS WAITING FOR A TYPE IN.

** THE PARAMETER IS READ IN IOPS ASCII. I.E., RUBOUT & ↑U EDITING AND ALTMODE TERMINATION ARE POSSIBLE.

8.3.2 Section B -- System Selection & Read-in

Summary: The user is asked if this is to be a system generation or an update of a previously generated system. If a new system (generation) is wanted, the requested system is transferred to the output device from the Master system.

Operation: The following question is typed:

IS THIS AN UPDATE OF A PREVIOUSLY GENERATED SYSTEM?

If the user types Y (for YES), most of Section B is bypassed and processing continues with the reading of information from a previously generated tape (see below).

If the user types N (for NO), the System Generator copies the Master system onto the output device. As this is being done, the System Generator informs the user

SYSTEM BEING TRANSFERRED

When the transfer is complete, BFSGEN reads selected blocks of information from the generated system and enters its update phase. While this is being done, BFSGEN types:

SYSTEM INFORMATION BEING READ

This information is then presented to the user during the update process as assumed parameter values. For instance, if the system was built for a machine with three Teletypes, one of the parameters will be:

NUMBER OF TTY'S [3]

which the user may either accept or change.

Once the system parameters have been read in, BFSGEN will type:

A ↑P TYPEIN WILL RETURN CONTROL AT THIS POINT

8.3.3 Section C -- System Parameters

Summary: The user is asked if he would like to change or see the system parameters. If he answers yes, each parameter is typed out with the parameter value enclosed in square brackets. The user may accept or retype each value.

Operation: At the start of Section C, BFSGEN types:

DO YOU WISH TO CHANGE (OR SEE) SYSTEM PARAMETERS?

expecting the user to type Y (if YES) or N (if NO). If NO, the old parameter values are retained and operation continues at Section D.

If YES, each parameter is listed with its assumed value and a response of acceptance (carriage return) or correction (typing new value) is expected. The following table lists the parameters in the order in which they are typed out and also the legal values that may be typed in.

NOTE

System core size information is obtained from the "multicore" paper tape bootstrap.

PARAMETER	ACCEPTED VALUES
NUMBER OF TTY'S [XX]	2,..., N (Note 1)
TTY #N IS A MODEL KSR[XX]	33 or 35 (Note 2)
BCONTROL [XX]	Ø,1,...,N-1 (Note 3)
FCONTROL [XX]	Ø,1,...,N-1 (Note 4)
RAISE TO LEVEL [X]	Ø,1 (Note 5)
$SHARE [X]	Y,N (Note 6)
FCORE [XXXXX]	2,3..., core size (Note 7)
FGD ↑C CONFIRMER [X]	Y,N (Note 8)

NOTE 1:

The maximum number of Teletypes the system will allow is a function of a parameter assignment made during assembly of the Resident Monitor. For the initial system generation, the maximum number is assumed and will appear within the brackets.

NOTE 2:

Teletypes are logically numbered Ø to N-1, where N is the value of the first parameter (number of Teletypes). For each Teletype, the parameter is printed in order to verify or determine that it is a model KSR33 or a model KSR35. ASR33 Teletypes are considered to be KSR33 and ASR35 to be KSR35.

NOTE 3:

The Background control Teletype may be any logical unit from Ø to N-1, where N is the value of the first parameter (number of Teletypes).

NOTE 4:

The Foreground control Teletype may be any logical unit from Ø to N-1, excluding the unit number assigned to the Background control Teletype.

NOTE 5:

All system software will raise to API level Ø for reentrancy protection[1] unless the user wants complete control over level Ø. If such is the case, he must specify that all system software will RAISE TO LEVEL 1.

NOTE 6:

The initial setting of the SHARE flag is determined at system generation time.

NOTE 7:

The initial setting of FREE CORE is determined at system generation time. 2 is the lowest value required.

NOTE 8:

One of the two jobs, Background or Foreground, can cause the Monitor to be reloaded once CTRL C has been typed the first time on the Foreground control Teletype. That job is designated as the ↑C Confirmer[2]. This parameter is phrased in the form of a question and asks is the Foreground job the ↑C Confirmer.

8.3.4 Section D -- Existing I/O Devices

Summary: The user is asked if he would like to change or see the known I/O devices. If he answers yes, the devices which must exist are listed followed by the devices which exist but may be deleted from the system. A question mark is typed following the listing of

[1] See section 7.4.

[2] See section 3.4.

each device which may be deleted, and the user must indicate whether or not each shall be kept by typing "Y" or "N".

Operation: BFSGEN begins Section D by typing:

DO YOU WISH TO CHANGE (OR SEE) KNOWN I/O DEVICES?

expecting the user to respond with Y (if YES) or N (if NO). If NO, the existing devices are retained and operation continues at Section E. If YES, the following question is typed out:

DO YOU WISH TO SEE MANDATORY I/O DEVICES?

expecting a response of Y or N.

If NO, operation continues with the expendable devices (see below).

If YES, the mandatory devices are listed along with relevant information:

```
--  (CLOCK)                     (Note 1)
API                             (Note 2)
SKIP IOT  7ØØØØ1                (Note 3)
API CHNL  11                    (Note 4)

-- (MEMORY PROTECT)
PI
SKIP IOT  7Ø17Ø1
MNEMONIC  MPSK                (Note 4)

TT  (TTY CONTROL)             (Note 2)
API
SKIP IOT  7ØØ4Ø1
MNEMONIC  TSFØ
SKIP IOT  7ØØ3Ø1
MNEMONIC  KSFØ

CO  (CORE-CORE)
NO  (NONE)

DT  (SYSTEM DEV)              (Note 5)
API
```

SKIP IOT 7Ø76Ø1
API CHNL Ø4
SKIP IOT 7Ø7561
API CHNL Ø4 (Note 6)

NOTE 1:

The system clock and the memory protect hardware must exist and are, therefore, listed. They are devices which do not have assignable handlers and do not have device names. Their names, therefore, are shown as --.

NOTE 2:

The designation API or PI is printed to indicate how the device is attached to the interrupt hardware of the machine. The Teletype device (TT) always includes the console Teletype which is not attached to the LT19 and operates always on PI. Teletype is the only device in the system which is allowed to receive interrupts on both PI (console) and API (LT19 Teletypes).[1] The core to core device (CO) and the device NONE (NO) do not utilize IOT instructions and do not generate interrupts.

NOTE 3:

SKIP IOT's for each device are listed, whether the device is on PI or API, for two reasons: First, if the device is on PI, the skip IOT's must be placed in the IOT skip chain located in the Resident Monitor so that interrupts for that device can be detected. That, however, is insufficient because the Monitor must then transfer to some core location where the interrupt will be processed. Since all but the resident device handlers are loaded relocatably from the system's I/O library, the linkage between the device handler and its interrupt lines must be made after the handler has been loaded into core. The handler does this by calling the .SETUP routine in the Resident Monitor. For each call it passes on two arguments: a skip IOT and the entry point address of the routine to service the interrupt.

Figure 8-1 illustrates how the linkage is made. If the device is on API, the System Generator places a JMS* TV instruction in the channel register for the device. The address, TV, is some fixed slot in the Monitor's transfer vector table. In the same relative position as the transfer vector, the System Generator places the associated skip IOT in a register in the SKIP IOT table. If the device is on PI, the skip IOT is placed in the skip chain (as well as the skip IOT table) and is followed by the instructions SKP and JMP* TV.

When the .SETUP routine in the Monitor is called, it searches the skip IOT table for the first argument (the skip IOT) and then stores the second argument (the address of the entry point of the interrupt service routine) in the corresponding transfer vector. Transfer vectors initially contain the address of an error routine which traps illegal (unserviceable) interrupts.

[1]The LT15 Teletype control is equivalent, from a software point of view, to the LT19.

NOTE 4:

If a device operates on API, the API channel number (channel address-40) is given following the skip IOT. If the device operates on PI, the skip IOT mnemonic is printed, later to be used to reorder the skip chain.

NOTE 5:

In a DISK system, the system device is DK (instead of DT for DECtape).

NOTE 6:

Sometimes more than one skip IOT is listed for a given API channel because DEC-supplied handlers are written to operate with or without API. If two skips must appear in the skip chain, the handler makes two calls to .SETUP. This sets up two transfer vectors, only one of which is referenced by the instruction in the channel register.

8.3.4.1 EXPENDABLE (DELETABLE) DEVICES - Once the mandatory devices have been (optionally) listed, a check is made to see if any other devices exist. If not, the following is typed out:

```
ALL EXISTING DEVICES ARE MANDATORY
```

and then processing goes on to Section E.

If deletable devices exist, the following is typed out:

```
THE FOLLOWING DEVICES EXIST.  INDICATE (Y/N) TO KEEP
```

Then, for each expendable device, BFSGEN types a 2-letter device mnemonic followed by a question mark, e.g.,

```
LP?
```

The user must then type Y (meaning YES) in order to keep the device or N (meaning NO) to delete it. If YES, then the device parameters are typed out in the same format as the mandatory devices, e.g.,

```
LP? Y
API
SKIP IOT  7Ø65Ø1
API CHNL  16
```

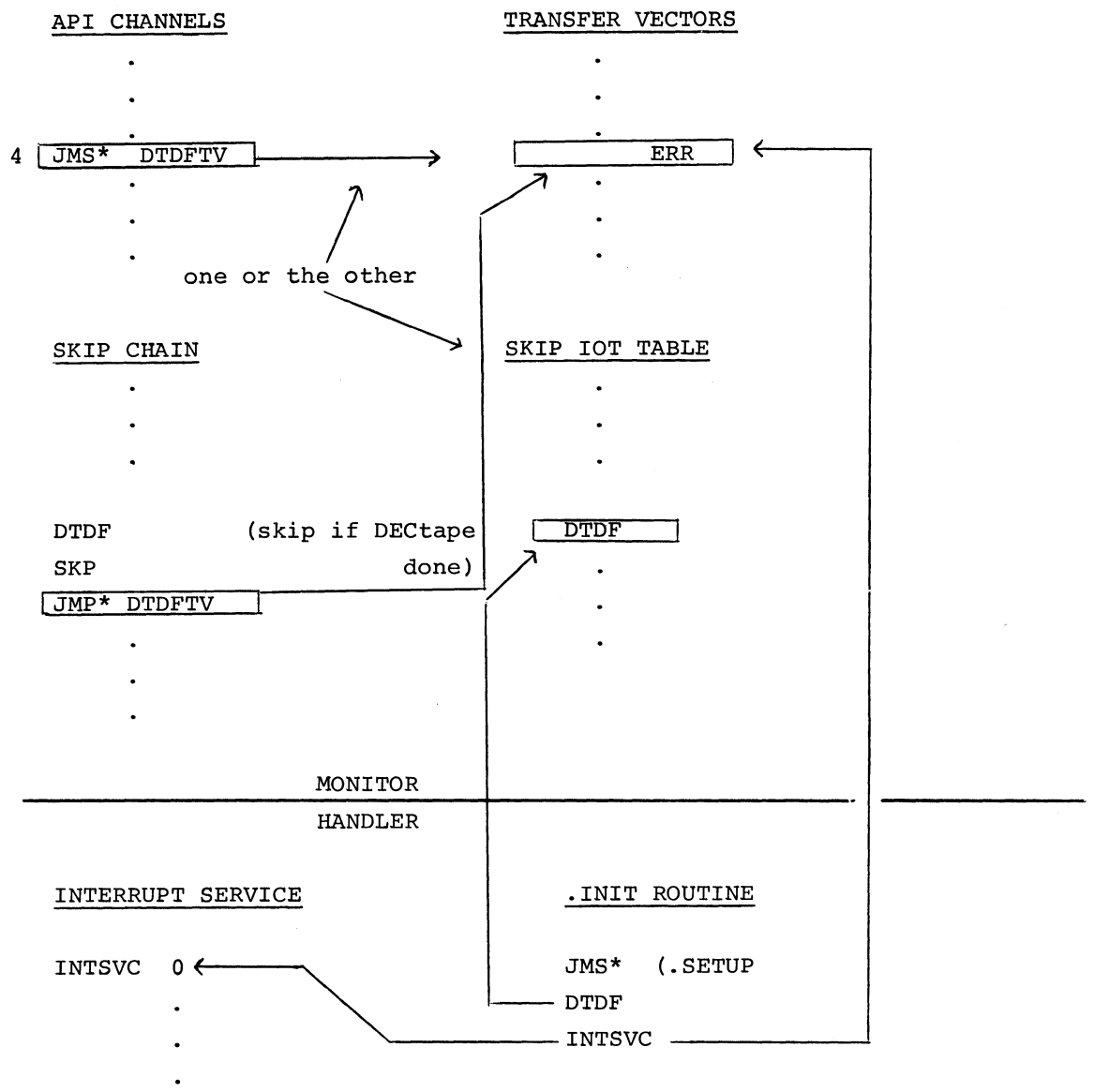


FIGURE 8-1 INTERRUPT LINKAGES FOR I/O DEVICES

It is possible that a user may want to keep an existing device but change its parameters (e.g., the skip IOT). In such a case, the existing device should be deleted and then reentered as a new device (refer to 8.3.5).

8.3.5 Section E -- Additional I/O Devices

Summary: The user is asked if I/O devices are to be added to the system. If he answers yes, the device name, skip IOT's, and skip mnemonics (or API channel numbers) are requested.

Operation: Section E starts with the following typeout:

```
ARE I/O DEVICES TO BE ADDED TO THE SYSTEM?
```

to which the user must respond Y (for YES) or N (for NO). If NO, processing continues at Section F. If YES, the following instruction is typed:

```
PROVIDE PARAMETERS AND ANSWER SPECIFIC QUESTIONS
```

A series of requests are then made to obtain information about each new device. The requests will include several of the following typeouts:

a. DEVICE NAME >
b. NUMBER OF INTERRUPTS SETUP >
c. API ?
d. SKIP IOT >
e. API CHNL >
f MNEMONIC >

The requests and types of responses required are as follows:

a. First, the device name is requested. The user must type a 2-letter name followed by a carriage return. The name must be unique; that is, it must not be identical to that of an existing device.

b. The number of interrupt lines that will be set up is asked. This, (referring to Note 3, Section D)is the number of calls the device handler(s) will make to the .SETUP routine in the Monitor; this is also the number of transfer vector table slots which must be reserved, the number of skips which will appear in the skip chain (for a PI device) and the number of skip IOT's for which slots must be reserved in the skip table.

Specifying zero interrupts setup is legitimate; this indicates that the device is not interrupt driven, (e.g., the CO, core-to-core, device). If the reply is 0, no more parameters are requested for such a device.

c. The user is asked if the device is connected to the API. The reply must be either Y↵ (for YES) or N↵ (for NO).

d. A pair of questions are asked pertaining to skip IOT's and are repeated for as many IOT's as were specified. The user is first requested to type in the octal value of the skip IOT instruction. Six digits are required in the answer and the IOT must be one not previously used.

e. The second request, if an API device, is the API channel number associated with the preceding skip IOT. The channel number may be any number from 4 through 37 (octal) but it must not belong to another existing device.

   The same channel number may be used several times for a given device in order to accommodate device handlers which set up several skip IOT's and which are written to operate with or without API[1].

f. The second request, for a PI device, is to type in a mnemonic for the skip IOT, to be used later if reordering the skip chain. The mnemonic may be 1 to 6 printing characters and must be unique.

   The mnemonic may be preceded by a minus sign to indicate a negative skip IOT (skip if device flag is not set) and will result in only a 2-word entry in the skip chain as opposed to the normal entry of 3 (for a positive skip).

When all the parameters have been entered for a new device, the following question is typed:

DO YOU WISH TO ADD ANOTHER DEVICE?

If the user replies N (for NO), BFSGEN will go on to Section F. If Y (for YES), requests for parameters, as above, will be made for another device.

[1]A hold-over from the PDP-9 where systems without API were supported at one time.

8.3.6 Section F -- PI Skip Chain

Summary: The user is asked if he would like to see or change the skip chain. If yes, the old skip chain order is typed out and then the user may retype the skip IOT's in whatever order he chooses.

Operation: BFSGEN types the following question:

```
DO YOU WISH TO SEE OR CHANGE THE SKIP CHAIN ORDER?
```

If N (for NO), operation proceeds to Section G. If Y (for YES), the skip IOT mnemonics are listed in the old order, minus those skips for deleted devices and with new device skips appended to the end of the list. Then the user is asked:

```
DO YOU WISH TO CHANGE THE SKIP CHAIN ORDER?
```

If N, operation will continue at Section G. If Y, BFSGEN types:

```
RETYPE MNEMONICS IN DESIRED ORDER
>
```

The angle bracket (>) signals the user to type a skip mnemonic followed by a carriage return. If he types an ALTMODE instead, the first unused skip in the old skip chain will be typed out. BFSGEN will continue to type an angle bracket until all the skips have been retyped.

8.3.7 Section G -- .IOTAB

Summary: The user is asked if he wishes to change (or see) the parameters for I/O device handlers. These are stored in the I/O table, .IOTAB, within the Non-resident Monitor. If he answers YES, the name of each existing device is typed out and the user is requested to provide information about available device handlers.

Operation: The user is asked:

```
DO YOU WISH TO CHANGE (OR SEE) I/O HANDLER PARAMETERS?
```

If N is the reply, processing will continue at Section H. If Y, the following is typed:

```
ACCEPT OR RETYPE THE FOLLOWING HANDLER PARAMETERS
```

Then, for each existing device, excluding NO (the null device), CO (core), TT (Teletype), and the System Device (DT or DK), some of the following information will be typed out for acceptance or revision:

a. DEVICE -- ZZ
   HANDLER NAME(S) [ZZA,ZZB,ZZC]

b. DEVICE-SHAREABLE ZZA? [Y]

c. EXT BUF SIZE [XXXX]

d. MAX OPEN FILES [XX]

e. UNIT-SHAREABLE ZZA? [N]

In the preceding list, the characters ZZ represent the 2-letter device name and XX and XXXX represent octal quantities.

The order of requests and the type of responses required are as follows:

a. First, the device name is typed out with the existing handler names enclosed in brackets. For a new device BFSGEN assumes at first that there exists an "A" handler. The user may accept the list of handlers by typing a carriage return. Otherwise, he must type in a new list of names, each separated by a comma, and terminate the list by a carriage return. Each handler name must be three letters long. The first two letters must be the device name.

b. If ZZA, namely, the "A" version of the handler, is present, it is asked if it is a multi-user (shareable) handler. An assumption is enclosed in brackets. The user must either type a carriage return to confirm the assumption or type Y↵ (for YES) or N↵ (for NO). A shareable ZZA handler is one that may be used by both the Foreground and the Background jobs simultaneously.

c. If ZZA is shareable, the external buffer size must be specified. The old value or a default value will be enclosed in brackets. Typing carriage return signals acceptance. Otherwise, the user must type in a buffer size (in octal) ranging in value from 0 to 7777. A zero buffer size is legal because some multi-user handlers may not require external I/O buffers.

d. If ZZA is shareable, the open file capacity of the handler must also be established. The value may range from 1 to 77 octal. For certain I/O handlers, particularly the ones which do not require external buffers, this count is meaningless. However, a value of at least 1 must be entered.

e. Finally, if ZZA is device-shareable, it is asked if it is also unit-shareable (with an assumption in brackets). The system device, for example, DECtape, is unit-shareable. This means that if the keyboard command $SHARE is typed to the Monitor, Background will be allowed to assign and use any DECtape units which the Foreground may be using. The responses which the user may give are the same as for the question "DEVICE-SHAREABLE ZZA?"

Once all the device handlers have been specified, the System Generator will check if the devices MT (MagTape) or LP (Line Printer) are present. If so, it will type out a system parameter related to the device and wait for the user to accept the parameter or retype it as follows:

```
DEFAULT MT TRACKS [7]

#LP COLUMNS  [XXX]
```

The Magtape is assumed to operate either as 7-track or 9-track so the user may type 7↵ or 9↵. This "default" assumption is entered in one bit in .SCOM+4 in the Resident Monitor, and that bit is interrogated by the Magtape handlers in the absence of a command from the user's program specifying the mode of operation.

The number of columns on the Line Printer can be 80, 120, or 132. Two bits in .SCOM+4 can be interrogated to determine the above information.

8.3.8 Section H -- .DAT Slots

Summary: The user is asked if he would like to change (or see) the permanent .DAT table assignments for Background (.DATB) and Foreground (.DATF). If he answers yes, he must accept or retype the contents of each reassignable slot.

Operation: The following question is typed:

```
DO YOU WISH TO CHANGE (OR SEE) BGD .DAT SLOTS?
```

After the Background .DAT table has been disposed of, similar questions will be posed for the Foreground .DAT table. If the user answers N (NO) for the Background table, processing continues with the Foreground table. If he doesn't wish to see or modify the Foreground table, operation continues at

Section I. In each case, if the answer is Y (YES), the following is typed:

```
ACCEPT OR RETYPE .DAT SLOT CONTENTS
```

The number and the contents of each .DAT slot will be typed out in the following form:

```
-10   [DTA3]
```

The assignment (DECtape "A" handler-unit 3) is enclosed in brackets to indicate that it may be accepted or changed. The assignments in .DAT slots -7, -3, and -2 will not be enclosed in brackets because reassignment is not permitted.

Typing carriage return indicates acceptance of the assignment. Otherwise, a new handler/unit must be typed using the same rules as for the Background/Foreground Monitor. For example,

```
-13   [DTA2]   DT↵          (meaning DTAØ)
-12   [LPAØ]   TT5↵         (meaning TTA5)
-11   [DTA1]   CDB↵         (meaning CDBØ)
-1Ø   [PRAØ]   TTA2↵
```

The System Generator will initially assign NONE to Foreground .DAT slots which refer to the Background control Teletype and vice versa and will reject user-typed assignments of the same nature.

8.3.9 Section I -- Re-write System Information

At this point, all system modifications have been made in core. BFSGEN thus informs the user:

```
SYSTEM INFORMATION BEING WRITTEN
```

and proceeds to transfer the modifications to the output device. When this has been done, the message:

```
SYSTEM UPDATE COMPLETED
DON'T FORGET POST-GENERATION PROCEDURES
```

is typed and then the System Generator exits to the Monitor.

8.4 POST-GENERATION PROCEDURES

Once the System Generator has completed its task, the user may need to modify the new system depending upon his particular hardware configuration:

a. 50 Cycle Machines

Customers whose hardware operates on a line frequency of 50 Hz should replace the 60 Hz versions of the subroutines TIME and TIME10 in the library, .F4LIB BIN, with the 50 Hz versions. .F4LIB BIN as well as the 50 Hz versions of TIME BIN and TIME10 BIN are on the generated Background/Foreground System. To do this, return to the Monitor and assign .DAT slots -10 and -14 to access files from the generated Background/Foreground system. Also, assign a scratch device to .DAT slot -15 and call in the system program UPDATE. When UPDATE is ready for a command, type the following:

```
> U ← .F4LIB)
> R  TIME)
> R  TIME10)
> CLOSE (ALTMODE)
```

When UPDATE has returned to the Monitor, call in PIP. Using PIP, delete the old .F4LIB BIN on the generated system and then transfer to it the new .F4LIB BIN from the scratch device.

Note that the versions of TIME and TIME10 distributed with the Background/Foreground System will not operate properly in any other Monitor System nor will the other Monitor versions of TIME and TIME10 work in Background/Foreground. This is also true of the FORTRAN OTS routine called FIOPS, the Background/Foreground version of which is in the library .F4LIB BIN.

b. Card Reader

The device handler CDB. BIN which is supplied in the system's I/O library, .IOLIB BIN, has been assembled for the CR03B card reader using the DEC029 character set. For customers who wish to use the DEC026 character set, the source file of the card reader handler, CDB. SRC, is present both on the Master system and on the generated system. The source file may be conditionally assembled by defining at assembly time the following parameters:

```
No parameters    --  CR03B and DEC029
DEC026 = 0       --  CR03B and DEC026
```

The new binary of the card reader handler should replace the old one in .IOLIB BIN on the generated tape. The commands to the system program UPDATE are as follows:

```
> US ← .IOLIB)
> R  CDB.)
> CLOSE (ALTMODE)
```

Note, the S command is a feature available in UPDATE V6A which strips the internal symbol definitions from binary files and thereby shortens the library. If an earlier version of UPDATE is used, omit the S in the first command. The library may be shortened at some later date.

After UPDATE has created the new I/O library, use PIP, as in part a. to replace the old library on the generated system.

c. Line Printers

The device handler LPA. BIN which is supplied in the system's I/O library, .IOLIB BIN, is for the LP15 line printer on the DECtape and RF/RS disk systems and for the 647 line printer on the RB09 system. The following binary files appear on the system tapes in case the library does not have the desired handler:

LP.15 BIN	(for the LP15)
LP.Ø9 BIN	(for the LPØ9)
LP.647 BIN	(for the 647)

The commands to the system program UPDATE to replace the line printer handler are equivalent to those shown in section 8.4b above, with the appropriate line printer handler used in place of the card reader handler.

d. The I/O Library

In the process of updating a Background/Foreground System, the user may have deleted some previously existing devices and device handlers and also added new ones. Since the System Generator does not modify the I/O Library, .IOLIB BIN, in the generated or updated system, the user should use the system program UPDATE to delete unwanted device handlers and to insert new ones.

e. Recouping Tape Space

Provided that the user never modifies the Master of the Background/ Foreground System, space may be retrieved on the system "tape" by deleting the following files using the Delete command in PIP:

TIME BIN	(50 Hz version)
TIME10 BIN	(50 Hz version)
CDB. SRC	(Card Reader Handler)

These files are all present on the Master "tape".

f. Disk Systems

Once a system generation and update has been performed and a useable Background/Foreground System has been built onto some unit on the Disk, that Disk unit should be copied onto a DECtape (or paper tapes) to provide a backup medium for restoring the system. To dump a disk unit onto DECtape, use the Copy command in PIP:

```
>C DTx←DKy (H)❯
```

The paper tape utility program RFSAV[1] may also be used to perform the same function.

g. All Systems

To avoid possible grief due to loss or damage of tapes, the user is urged to copy Master tapes and generated tapes to be kept as backups.

8.5 ERROR DETECTION

The System Generator is a locative program, and any mistakes made by the user in answering queries will result in a descriptive error message. The incorrect answer will be ignored and the question will be repeated.

Within the System Generator there is a safeguard function which should never be noticed by the user. If a malfunction is detected, which could be the result of a bad system tape or of a programming bug in the System Generator itself, the following will be printed on the console Teletype:

```
SYSTEM CRASH
PLEASE ↑Q &
SAVE TTY LOG

nnnnnn
oooooo
pppppp
```

If such a message should occur, the user is requested to perform a dump of core memory by executing a CTRL Q (↑Q). The dump can then be printed for diagnostic purposes. The TTY Log is simply the entire printout that led to the error. The six-digit octal numbers that are printed in the preceding message are:

nnnnnn	=	unrelocated address within BFSGEN where the error was detected.
oooooo	=	runtime (relocated) address within BFSGEN where the error was detected. Bits Ø-2 indicate the state of the Link, Page/Bank Mode, and Memory Protect.
pppppp	=	the contents of the accumulator when the error was detected.

This information would be an aid in tracking down the cause of the error.

---

[1]See PDP-15 Utility Programs Manual, DEC-15-YWZA-D

APPENDIX I

.SCOM REGISTERS

The function of the .SCOM (System COMmunication) Registers is to provide, among the various program elements of the Background/Foreground Monitor System, an easily accessible set of registers which contain communication pointers, data words, and program flags indicating the state of the system.

The .SCOM table begins at location 100_8 within the Resident Monitor. Location 100 is referred to as .SCOM or .SCOM+0 and the $(N+1)^{th}$ register is referred to as .SCOM+N.

Each .SCOM register has a special meaning and format. At present, there are 117_8 such registers. Slots at the end will be allotted for future expansion as needed.

REGISTER DEFINITIONS: The following list indicates the contents of each .SCOM register. Those which are fixed at assembly or system generation time and never changed are marked by (F). Some .SCOM registers must have a Foreground value and a Background value. Therefore, their contents must be swapped from one to the other, depending upon which job has control. They are flagged by (S). Some .SCOM registers have been reserved for future software. If their contents (format) are as yet unspecified, they will be flagged with (U).

.SCOM + 0	(F)	Pointer to the highest register in core (37777, 57777, or 77777). This value is established from the location of the bootstrap loader the first time the system is loaded.
.SCOM + 1	(S)	(a) Address just above the Resident Monitor when the Non-resident Monitor has been loaded for Foreground. (b) Address just above the Foreground job when the Resident Monitor has loaded the Non-resident Monitor in the Background. If the system program PIP is called, this will be the first location of its .DEV table. (c) For DDT in the Background this points to the start of its symbol table.
.SCOM + 2	(S)	(a) Same as (a) for .SCOM + 1. (b) Normally used by user and system programs to indicate the first (lowest) location in free core. (c) For DDT in the Background this points to the first location after the symbol table, which is also the first location of free core.

.SCOM + 3	(S)	Normally used by user and system programs to indicate the last (highest) location in free core. For the Foreground, this is also the highest location allocated to the Foreground job.
.SCOM + 4	(S)	Bits indicate machine configuration:
	(F)	Bit Ø Ø=No API; 1=API
		Bit 1[1] Ø=No EAE; 1=EAE
	(F)	Bits 2-5 Ø (Reserved and unused)
		Bit 6[2] Ø = 7-channel MAGtape
		1 = 9-channel MAGtape
	(S)	Bit 7 Ø=Bank Mode Addressing
		1=Page Mode Addressing
	(F)	Bit 8 1 = no ↑Q area on system tape unit Ø
	(U)	Bit 9 Unassigned
	(F)	Bits 1Ø-11 Ø (Reserved and unused)
	(F)	Bits 12-13 Ø = No Line Printer
		1 = 8Ø column printer
		2 = 12Ø column printer
		3 = 132 column printer
	(F)	Bit 14 1 = Background/Foreground System
	(F)	Bits 15-17 Ø (Reserved and unused)
.SCOM + 5		(a) Initially this points to RESINT, the address of the initialization section in RESMON. The paper tape bootstrap loader transfers control indirectly through this location. (b) When calling the System Loader to bring in a system program, the Non-resident Monitor stores here the code number of the program to be loaded. (c) When running a system program, its start address is stored here.
.SCOM + 6		(a) When the Non-resident Monitor calls the System Loader to load user programs, bits Ø - 2 indicate which command was given to the Monitor: $LOAD, $GLOAD, $DDT, or $DDTNS. Bit Ø = 1 if $DDT or $DDTNS (DDT load) Bit 1 = Ø if $LOAD; Bit 1 = 1 if $GLOAD Bit 2 = Ø if $DDT; Bit 2 = 1 if $DDTNS (b) When the user programs have been loaded, the start address of the main program is stored here. The load command code bits (Ø - 2) remain as in (a).
.SCOM + 7		The interrupted PC plus L,P/B,MP are saved here for DDT in the Background when CTRL T has been typed.
.SCOM + 1Ø	(S)	The interrupted PC plus L,P/B,MP are saved here after a NORMAL CTRL P has been typed and honored.
.SCOM + 11	(F)	Bootstrap restart instruction.
.SCOM + 12	(F)	Ø (Reserved for PDP-9 use).

[1]The presence or lack of EAE is determined dynamically by the Resident Monitor.

[2]7/9-channel default operation may be set by Foreground Keyboard command.

.SCOM + 13	(F)	Pointer to the .IOIN[1] table in the Resident Monitor.
.SCOM + 14	(F)	Pointer to the .MUD[2] table in the Resident Monitor.
.SCOM + 15	(F)	Pointer to the .BFTAB[3] table in RESMON.
.SCOM + 16	(F)	Pointer to .DATF[4], Foreground .DAT slot Ø in the Resident Monitor.
.SCOM + 17	(F)	Pointer to .DATB[4], Background .DAT slot Ø in the Resident Monitor.
.SCOM + 2Ø	(F)	Ø indicates that the computer does not have an extra 4K page segment (which rules out 20K and 28K).
.SCOM + 21	(F)	Default value of $FCORE (Foreground free core) established at system generation.
.SCOM + 22		Reserved for MAGtape handler
.SCOM + 23	(F)	Two's complement size of the Monitor's transfer vector table (used by System Generator).
.SCOM + 24	(F)	Pointer to the Monitor's transfer vector table (used by System Generator).
.SCOM + 25		(a) Prior to loading the Foreground job, the amount of free core requested by the $FCORE command is stored here. If no $FCORE command is given, the default assumption is taken from .SCOM + 21. (b) After the Foreground job has been loaded, this register contains a pointer to the register immediately above the Foreground core area.
.SCOM + 26	(S)	Contains Ø if Foreground is in control and 1 if Background is in control.
.SCOM + 27	(F)	Pointer to IOT Skip literal table in the Monitor (used by System Generator).
.SCOM + 3Ø	(F)	Pointer to PI Skip Chain.

[1].IOIN is the table which indicates which I/O devices are in core, which units on each device are spoken for, and which job (Background or Foreground) owns them.

[2].MUD is a table listing all available multi-user device handlers, with pertinent information about those handlers.

[3].BFTAB is a buffer table containing pointers to and the sizes of all external I/O buffers that were set up by the loaders.

[4].DATF is the Device Assignment Table for Foreground.
.DATB is the Device Assignment Table for Background.

.SCOM + 31		Software Memory Protect Bound (a) Set from .SCOM + 25 after the System Loader has loaded the Foreground job. (b) Set to point just above the Background I/O handlers and I/O buffers after the Background job has been loaded.
.SCOM + 32		(a) Pointer to the Hardware Memory Protect Bound (or where it should be set). Contents (SCOM + 32) ≥ contents (.SCOM + 31).
.SCOM + 33		Background Program Counter, including L,P/B,MP.
.SCOM + 34	(F)	Address of the resident Teletype handler (TTA).
.SCOM + 35		Interrupt Service Flag. Non-Ø indicates that control is in some interrupt service routine.
.SCOM + 36	(F)	Bits to tell the Teletype handler which units are model 33 (specific bit = Ø) and which model 35 (specific bit = 1). Bit Ø corresponds to unit Ø, bit 1 to unit 1; etc.
.SCOM + 37	(F)	Pointer to CALER. Used to detect attempt to re-enter CAL handler and to trap CAL* instructions.
.SCOM + 4Ø		CAL flag. Non-Ø if control is in the CAL handler (indication necessary for interrupt servicing).
.SCOM + 41		"Who's running in the Background" Flag. Bit Ø = 1 if a Loader is running. Bits 1-17: 17777 = Non-resident Monitor Ø = user program or DDT 1 = EDIT 2 = MACRO 3 = PIP 4 = F4 5 = SRCCOM 6 = DUMP 7 = UPDATE 1Ø unused 11 = MACROA 12 = F4A 13 = EXECUTE 14 = CHAIN 15 = PATCH 16 = DTCOPY
.SCOM + 42		Level 5 (API,Foreground) busy register. Zero indicates level 5 non-busy. Non-zero indicates that Foreground level 5 is idle waiting for some I/O to complete. Set non-Ø with the initial address of the device handler doing the I/O. If the device is Teletype, the unit number + 4ØØØØØ is stored here instead.
.SCOM + 43		Same as .SCOM + 42 for Foreground level 6.
.SCOM + 44		Same as .SCOM + 42 for Foreground level 7.

.SCOM + 45		Same as .SCOM + 42 for Foreground Mainstream level.
.SCOM + 46		Foreground level 5 I/O satisfied flag. Zero indicates that level 5, which was I/O bound, can be started up again.
.SCOM + 47		Same as .SCOM + 46 for level 6.
.SCOM + 5Ø		Same as .SCOM + 46 for level 7.
.SCOM + 51	(F)	Pointer to REALTP[1] in the Resident Monitor.
.SCOM + 52	(F)	Pointer to IOBUSY[2] in the Resident Monitor.
.SCOM + 53	(F)	Pointer to LV4Q[3] in the Resident Monitor.
.SCOM + 54	(F)	Pointer to CALL4[4] in the Resident Monitor.
.SCOM + 55	(F)	Pointer to .SETUP[5] in the Resident Monitor.
.SCOM + 56	(F)	Pointer to GETBUF[6] in the Resident Monitor.
.SCOM + 57		If non-Ø, a pointer to the entry point of the last Mainstream Foreground real-time subroutine in the chain of subroutines to be run when Foreground Mainstream gets control.
.SCOM + 6Ø		Pointer to the entry point +1 of the first subroutine in the chain of Foreground Mainstream real-time routines to be run when Foreground Mainstream gets control.
.SCOM + 61		Same as .SCOM + 57 for Background.
.SCOM + 62		Same as .SCOM + 6Ø for Background.
.SCOM + 63		Argument for API instruction ISA when interrupts at API software levels are to be requested.
.SCOM + 64	(F)	Pointer to CR.QR[7] in the Resident Monitor.

[1] REALTP is a subroutine to process real-time requests.

[2] IOBUSY is a subroutine to check for I/O busy termination.

[3] LV4Q queue is a list of I/O handlers which are waiting to complete their interrupt service processing at API level 4.

[4] CALL4 is a subroutine to initiate an API level 4 request.

[5] SETUP is the routine initially called by all I/O handlers to set up skips in the PI skip chain or API channel registers.

[6] GETBUF is a routine called by the I/O handlers which assigns buffer areas to the handlers via .BFTAB.

[7] CR.QR is a routine called by I/O handlers to initiate a device-not-ready request.

.SCOM + 65		Set non-Ø, while a Foreground user program is running, to indicate that the resident buffer may not be used by the Foreground. The resident buffer must be available to the Background, which presumably changes jobs more often, for use by the Monitor and the Loaders.
.SCOM + 66	(F)	Pointer to ERRORQ[1] in the Resident Monitor.
.SCOM + 67	(F)	Pointer to Foreground control character table in TTA.
.SCOM + 7Ø	(F)	Pointer to Background control character table in TTA.
.SCOM + 71		Error flag. The following conditions exist if the respective bit = 1: Ø - Background error 1 - Foreground error 2 - Background terminal error 3 - Foreground terminal error
.SCOM + 72	(F)	Pointer to the Foreground error processing subroutine plus the 2ØØØØØ bit to enter bank mode.
.SCOM + 73	(F)	Same as .SCOM + 72 for Background error subroutine.
.SCOM + 74		Saved argument for Foreground error routine ISA instruction.
.SCOM + 75	(F)	Contains JMS IGNORE, a call to a dummy interrupt service routine, used during error processing.
.SCOM + 76	(F)	Two's complement count of the number of Teletypes on the machine.
.SCOM + 77		$SHARE Flag (to allow Background to share Foreground I/O bulk storage units). Non-zero indicates that SHARING is allowed. Initial value is set at System Generation.
.SCOM + 1ØØ	(F)	Pointer to ENTERQ[2] in the Resident Monitor. Will contain Ø, instead, if ENTERQ routine not assembled into the Monitor.

[1]ERRORQ is a routine called to enter information in the Foreground and/or Background error queues and to set the error flags in .SCOM + 71.

[2]ENTERQ is a subroutine which makes entries in the API queue.

.SCOM + 1Ø1		If set non-zero by the Foreground keyboard command, $MPOFF, Background enters EXEC mode. The memory protect boundary register is zeroed to allow Background to modify and transfer to any location in core. Background IOT's will still trap to the Monitor but the IOT's will be executed.
.SCOM + 1Ø2	(F)	Argument for ISA instructions which will raise either to API level Ø or level 1 (as the highest used Monitor level) to protect common Monitor routines from being reentered. Value established at System Generation.
.SCOM + 1Ø3	(F)	Monitor version number. For FKM15 V3A printout, for example, this register will contain: .ASCII "3A".
.SCOM + 1Ø4	(F)	Flag to indicate which job (Ø = Foreground; 1 = Background) confirms Foreground CTRL C see Section 3.4).
.SCOM + 1Ø5	(F)	Two's complement size of the PI ship chain. (Used by System Generator).
.SCOM + 1Ø6	(F)	Pointer to the register immediately above the Resident Monitor (set by the Non-resident Monitor after it has built the .MUD table). (Used by CHAIN program.)
.SCOM + 107 .SCOM + 11Ø .SCOM + 111		Used to store the file directory entry block of the XCT file to be EXECUTEd in the Foreground.
.SCOM + 112 .SCOM + 113 .SCOM + 114		Used to store the file directory entry block of the XCT file to be EXECUTEd in the Background.
.SCOM + 115	(F)	Maximum number of Teletypes allowed, which is a function of an assembly parameter in the Monitor (Used by System Generator).
.SCOM + 116		Foreground MAGtape status.
.SCOM + 117		Background MAGtape status.

APPENDIX II

ERRORS

ERROR HANDLING IN BACKGROUND/FOREGROUND

The processing of errors detected by the Resident Monitor, I/O handlers, the Linking Loader, and the System Loader in the Background/Foreground System has been changed from the manner of error processing in the ADVANCED and I/O Monitor Systems.

The most significant change is the introduction of terminal and non-terminal errors. A terminal error stops execution of the job associated with the error. This causes all I/O handlers assigned to that job to be called to stop I/O that may be in progress and all Monitor queues to be cleared of entries for that job (Background, Foreground, or both).

A non-terminal error does not necessarily warrant aborting the operation of the offending job. A non-terminal error message is entered into a queue for the appropriate job and is printed on the appropriate control Teletype when that unit is free. While the printing of non-terminal error messages is pending or in progress, operation of the offending job is suspended. This restriction does not apply to I/O handlers, which may continue interrupt processing.

The format for error messages generated by the Resident Monitor, I/O handlers, and the Loaders is:

```
.ERR  NNN  XXXXXX
```

where NNN = error code
XXXXXX = auxiliary information

These errors are tabulated on pages II-3, -4, and -5.

Errors detected by the FORTRAN Object Time System (OTS) are formatted as follows:

```
OTS NN
```

where NN = error code.

OTS errors are listed on page II-6.

CONTINUATION AFTER ERROR

All .OTS errors, except 7 and 15 (see list on page II-6), are terminal errors. After OTS has printed the error message, it exits to the Monitor. Therefore, after a terminal .OTS error the user does not have the option of restarting his program.

Terminal .ERR errors terminate the operation of user programs. After the printing of the error message, the user has the option of typing CTRL P (to restart his program at the CTRL P restart address), CTRL T (to return to DDT), CTRL Q (to take a dump of memory), or CTRL C (to return to the Monitor to load another job). If the error occurs while control is in the Non-resident Monitor or in a Loader, the user does not have the options indicated above. The Monitor will automatically be reloaded.

Non-terminal .ERR errors do not terminate the operation of user programs. Continuation, following the printing of the error message, is automatic.

ERROR CALL

Routines that wish to set up an error condition, I/O device handlers for example, should use the following coding sequence:

```
        LAC*    (.SCOM+66           /POINTER TO ERRORQ
        DAC     TEMP                /SUBROUTINE.
        LAC*    (.SCOM+1Ø2          /RAISE TO API
        ISA                         /LEVEL Ø (OR 1).[1]

        LAW     CODE                /SEE BELOW.
        JMS*    TEMP                /CALL ERRORQ.
AUXARG  XX                          /AUXILIARY ARGUMENT.
        DBK                         /RETURN HERE.[1]
```

The calling program must be operating with memory protect disabled in order to be able to issue IOT's.

The first argument, given in the AC to ERRORQ, may be loaded either by LAW code or by LAC (code in the following format:

```
                Bits Ø-5 are ignored
                Bit 6 = Ø means non-terminal error
                Bit 6 = 1 means terminal error
        Code    Bit 7 = 1 means Background error        Both bits (7 and 8)
                Bit 8 = 1 means Foreground error        may be set to 1
                Bits 9-17 is a 3-digit error code
```

[1]For a routine that operates at API level 1 or Ø, read the discussion on Reentrancy Protection, 7.4.

To avoid the possibility of future conflicts, user programs and device handlers should utilize codes 6ØØ - 777.

The auxiliary argument, following the JMS to ERRORQ, will be printed in the error message as a 6-digit octal number. The error message will be printed in the form:

```
.ERR  NNN  XXXXXX
```

where NNN = the 3 digit error code
XXXXXX = the 6-digit auxiliary information

The actual printing of the error message and processing of the error will be done only after all interrupt processing has ceased and when control is no longer in the CAL handler.

BACKGROUND/FOREGROUND MONITOR ERRORS (.ERR)

The following abbreviations are used below in describing the auxiliary information:

L - bit Ø is the status of the link
PB - bit 1 is the status of page/bank addressing mode
MP - bit 2 is the status of memory protect
CAL ADDR - bits 3-17 contain the address of the CAL in error.

ERROR NO.	ERROR	AUXILIARY INFORMATION	TERMINAL
ØØØ	ILLEGAL CAL FUNCTION	L, PB, MP, CAL ADDR	YES
ØØ1	CAL* ILLEGAL	L, PB, MP, CAL ADDR[1]	YES
ØØ2	.DAT SLOT ERROR (erroneous .DAT slot number or .DAT slot not tied to an I/O handler)	L, PB, MP, CAL ADDR	YES
ØØ3	ILLEGAL INTERRUPT	L, PB, MP, PC	YES
ØØ4	MORE THAN ONE DEVICE NOT READY	.ASCII /XX/ ; XX = DEVICE NAME	YES
ØØ5	ILLEGAL .SETUP	RETURN ADDRESS FROM .SETUP (ADDRESS IN CALLING DEVICE HANDLER)	YES
ØØ6	ILLEGAL HANDLER FUNCTION	L, PB, MP, CAL ADDR[1]	YES
ØØ7	ILLEGAL DATA MODE OR SUBFUNCTION CODE	L, PB, MP, CAL ADDR[1]	YES
Ø1Ø	FILE STILL ACTIVE	UNIT #, CAL ADDR	YES

[1]The auxiliary information, depending on the source of the error, is sometimes UNIT #, CAL ADDR.

ERROR NO.	ERROR	AUXILIARY INFORMATION	TERMINAL
Ø11	SEEK/ENTER/REWIND NOT EXECUTED	UNIT #, CAL ADDR	YES
Ø12	UNRECOVERABLE DECTAPE ERROR	STATUS REGISTER B (Bits 0-11) AND UNIT (Bits 15-17)	YES
Ø13	FILE NOT FOUND	UNIT #, CAL ADDR	YES
Ø14	DIRECTORY FULL	UNIT #, CAL ADDR	YES
Ø15	DECTAPE FULL	UNIT #, CAL ADDR	YES
Ø16	OUTPUT BUFFER OVERFLOW	UNIT #, CAL ADDR	YES
Ø17	TOO MANY FILES FOR HANDLER	UNIT #, CAL ADDR	YES
Ø2Ø	DISK FAILURE	DISK STATUS REGISTER	YES
Ø21	ILLEGAL DISK ADDRESS	UNIT #, CAL ADDR	YES
Ø22	TWO OUTPUT FILES ON ONE UNIT	UNIT #, CAL ADDR	YES
Ø23	ILLEGAL WORD PAIR COUNT	UNIT #, BLOCK #	YES
Ø27	ILLEGAL DISK UNIT	UNIT #, CAL ADDR	YES
Ø31	NON-EXISTENT MEMORY REFERENCE	L, PB, MP, PC	YES
Ø32	MEMORY PROTECT VIOLATION	L, PB, MP, PC[1]	YES
Ø36	BACKGROUND MEMORY PROTECT VIOLATION ATTEMPT VIA CAL ARGUMENT	L, PB, MP, CAL ADDR	YES
Ø37	LINE OVERFLOW	L, PB, MP, CAL ADDR	YES
Ø47	ILLEGAL HORIZONTAL TAB	L, PB, MP, CAL ADDR	YES
Ø5Ø	.TIMER REQUEST CANNOT FIT IN CLOCK QUEUE OR BACKGROUND REQUEST REMOVED TO MAKE ROOM FOR FOREGROUND REQUEST	ADDRESS OF REAL TIME SUBROUTINE THAT WAS TO GET CONTROL ON COMPLETION OF INTERVAL	NO
Ø52	MAINSTREAM REAL TIME REQUEST IGNORED BECAUSE ROUTINE IS ALREADY ENTERED	PRIORITY LEVEL/SUBROUTINE ENTRY POINT	NO
Ø53	APIQ OVERFLOW	ENTRY THAT WOULD NOT FIT (PRIORITY LEVEL/ SUBROUTINE ENTRY POINT)	NO
Ø54	ILLEGAL WORD OR WORD PAIR COUNT (Either the word count was positive or the starting address plus the absolute value of the word count exceeded existing memory)	L, PB, MP, CAL ADDR	YES
Ø55	NO BUFFERS AVAILABLE	RETURN ADDRESS FROM GETBUF (ADDRESS IN CALLING DEVICE HANDLER)	YES
Ø56	ILLEGAL .ERROR CAL[2]	L, PB, MP, CAL ADDR	YES

[1]If a memory protect violation occurs because of a Background JMP instruction, the PC is the effective address rather than the location of the JMP.

[2]A special error call to the Monitor (CAL code 16) is available for use only by the Loaders.

ERROR NO.	ERROR	AUXILIARY INFORMATION	TERMINAL
Ø57	ILLEGAL .EXIT CAL	L, PB, MP, CAL ADDR	YES
Ø6Ø	.INIT NOT EXECUTED	CAL ADDR	YES
Ø61	PARITY ERROR IN DIRECTORY BLOCK (1ØØ) OR FILE BIT MAP BLOCK (71-77)	UNIT #, BLOCK #	YES
Ø62	TOO MANY NON-TERMINAL ERRORS	NUMBER OF ERRORS DISCARDED	NO
2ØØ	ILLEGAL TELETYPE UNIT	L, PB, MP, CAL ADDR	YES

LOADER ERRORS (.ERR)

All Loader errors are terminal. The auxiliary information which is printed is irrelevant.

1ØØ NO ROOM IN CORE FOR PROGRAM SEGMENT
1Ø1 PROGRAM AND SYMBOL TABLE OVERLAP
1Ø2 .BFTAB OVERFLOW
1Ø3 .IOIN TABLE OVERFLOW
1Ø4 $FILES COUNT OVERFLOW
1Ø5 PARITY ERROR, CHECKSUM ERROR, OR BUFFER OVERFLOW
1Ø6 ILLEGAL LOADER CODE
1Ø7 COMMON BLOCK SIZE ERROR[1]
11Ø MISSING GLOBAL(S)
111 ILLEGAL .DAT SLOT NUMBER
112 .DAT SLOT CONTENTS = Ø
113 SAME DEVICE - DIFFERENT HANDLERS[2]
114 ILLEGAL HANDLER CODE (Illegal .DAT slot contents)
115 ABSOLUTE PROGRAM ERROR[3]
116 FOREGROUND CAN'T USE UNIT Ø ON SYSTEM DEVICE[4]
117 NO ROOM TO BUILD .EXIT LIST
12Ø XCT FILE OVERLAYS EXECUTE
121 XCT FILE OVERLAYS THE MONITOR
122 XCT FILE OVERLAYS THE SYMBOL TABLE
123 XCT FILE NOT BUILT FOR THIS CONFIGURATION[5]

---

[1] COMMON Block size declared differently when Block size previously fixed in BLOCKDATA subprogram.

[2] Only one version of a device handler may be in core. .DAT slot requested a different handler for a device when another handler for that device was already in core.

[3] An absolute .LOC program may not be loaded once relocatable programs have been loaded. Absolute and relocatable .LOC in same program is illegal.

[4] $SHARE command was not given.

[5] Configuration word in "XCT" file indicates if it was built to run in bank or page mode and Background or Foreground.

OBJECT TIME SYSTEM ERRORS (.OTS)

All .OTS errors (except 7 and 15) are terminal and no auxiliary information is printed:

Ø-4	UNUSED
5	ILLEGAL REAL SQUARE ROOT ARGUMENT
6	ILLEGAL DOUBLE SQUARE ROOT ARGUMENT
7	ILLEGAL INDEX IN COMPUTED GOTO
1Ø	ILLEGAL I/O DEVICE NUMBER
11	ILLEGAL INPUT DATA OR INCORRECT DATA MODE
12	ILLEGAL FORMAT STATEMENT
13	ILLEGAL REAL LOGARITHMIC ARGUMENT
14	ILLEGAL DOUBLE LOGARITHMIC ARGUMENT
15	RAISE ZERO TO A POWER LESS THAN OR EQUAL TO ZERO

APPENDIX III

TELETYPE HARDWARE CHARACTERISTICS

SYSTEM REQUIREMENTS AND OPTIONS

The multi-unit Teletype handler assumes that the Teletype configuration consists of:

a. A model 33 or Model 35KSR console Teletype,
b. from 1 to 4 LT19[1] multi-station Teletype controls, and
c. from 1 to 16_{10} Model 33 or Model 35KSR Teletypes interfaced to the LT19 controls[2].

The console Teletype has its own set of IOT's, operates as half-duplex, and is connected to the PIC (Program Interrupt Control).

The LT19 can handle from 1 to 5 Teletype lines and will operate at API level 3, using channel registers 74 and 75.

Teletypes connected to LT19 controls are operated in full-duplex mode, which requires the software to echo characters input from the Keyboard back to the teleprinter.

LT19 IOT's

The following tables list the device and subdevice codes associated with each teleprinter and keyboard and indicate the logical unit numbers which the Teletype handler associates with them. The console Teletype, which is not connected to the LT19 controls, is defined to be logical unit Ø

TABLE 1: 1 to 5 units; 1 LT19

	UNIT #	PRINTER CODE	KEYBOARD CODE	LOGICAL UNIT #
LT19 #1	1	XX400X	XX410X	1
	2	XX402X	XX412X	2
	3	XX404X	XX414X	3
	4	XX406X	XX416X	4
	5	XX420X	XX430X	5

[1]For two-Teletype systems, an LT15 control may be used rather than LT19. The LT15 is operationally identical to line 1 on the LT19; thus no further mention will be made of the LT15.

[2]As standardly supplied, the Background/Foreground system will support a maximum of six Teletypes. Expansion beyond six requires a simple reassembly of the Resident Monitor.

TABLE 2: 6 to 10 units; 2 LT19's

	UNIT #	PRINTER CODE	KEYBOARD CODE	LOGICAL UNIT #
LT19 #1	1	XX400X	XX410X	1
	2	XX402X	XX412X	2
	3	XX404X	XX414X	3
	4	XX406X	XX416X	4
	5	XX440X	XX450X	11
LT19 #2	1	XX420X	XX430X	5
	2	XX422X	XX432X	6
	3	XX424X	XX434X	7
	4	XX426X	XX436X	10
	5	XX442X	XX452X	12

TABLE 3: 11 to 15 units; 3 LT19's

	UNIT #	PRINTER CODE	KEYBOARD CODE	LOGICAL UNIT #
LT19 #1	1	XX400X	XX410X	1
	2	XX402X	XX412X	2
	3	XX404X	XX414X	3
	4	XX406X	XX416X	4
	5	XX460X	XX470X	15
LT19 #2	1	XX420X	XX430X	5
	2	XX422X	XX432X	6
	3	XX424X	XX434X	7
	4	XX426X	XX436X	10
	5	XX462X	XX472X	16
LT19 #3	1	XX440X	XX450X	11
	2	XX442X	XX452X	12
	3	XX444X	XX454X	13
	4	XX446X	XX456X	14
	5	XX464X	XX474X	17

TABLE 4: 16 units; 4 LT19's

(The setup for the first three controls is in Table 3).

	UNIT #	PRINTER CODE	KEYBOARD CODE	LOGICAL UNIT #
LT19 #4	1	Unused	Unused	-
	2	Unused	Unused	-
	3	Unused	Unused	-
	4	Unused	Unused	-
	5	XX466X	XX476X	20

TELETYPES

In the Background/Foreground System, Teletype models are presumed to have certain hardware characteristics:

Model 33: No horizontal tabbing mechanism
No vertical tabbing mechanism
No form feed mechanism

(Note, the lack of vertical tab or form feed does not affect the software.)

Model 35: Has horizontal tabbing mechanism
Has vertical tabbing mechanism
Has form feed mechanism

The Teletypes are assumed to be KSR (Keyboard Send/Receive) units. ASR (Automatic Send/Receive) Teletypes may be used; however, their paper tape input and output capability cannot be used. The system will not support Model 37 Teletypes.

The Teletype handler will simulate horizontal tab both on input and on output, on model 33 Teletypes, but will not simulate either vertical tab or form feed. Tab stops are assumed to be 8 spaces apart.

APPENDIX IV

MONITOR SYSTEM TABLES

1. CONTENTS

Some of the most commonly used Monitor system tables are described in this appendix. These descriptions are intended for individuals interested in the detailed structural aspects of the Monitor; the information they contain is not required to operate the system. The following tables are described:

	MNEMONIC	DERIVED FROM
a)	SYSBLK	System Block
b)	.DAT	Device Assignment Table
c)	.IOIN	Input/Output Handlers In Core Table
d)	.MUD	Multi-user Device Table
e)	.BFTAB	Buffers Table

2. SYSBLK

The following paragraphs describe in detail the overall aspects and components of a system SYSBLK.

2.1 Function

SYSBLK acts as a central source of information about system programs. For example, the system loader (.SYSLD) uses SYSBLK information to determine how to load each system program. Utility program PATCH also uses SYSBLK for loading information and can also modify SYSBLK to permit system programs to be loaded or patched onto the system tape.

2.2 Size and Location

SYSBLK is a 256-word block located on the system DECtape. In Background/Foreground system DECtapes, SYSBLK is located in block 40_8; in ADVANCED Monitor systems, SYSBLK is located in block 61_8 of the system DECtape.

When loaded, SYSBLK resides in core immediately below the system loader (.SYSLD).

2.3 Overall Structure

The following table illustrates the overall structure and organization of SYSBLK.

(1^{st} word) pointer to .DAT slot pointer table.	DATTAB
(7 words)	System parameters for Program Ø
(7 words)	System parameters for Program 1
. . .	. . .
(7 words)	System parameters for Program M
(DATTAB = 7*(N+1)+1)[1]	Program Ø
.DAT slot pointers table	Program 1
. . .	. . .
	Program N
Pointer to the END.	END
(PROGØ)	List of .DAT slots used by Program Ø.
(PROG1)	List of .DAT slots used by Program 1.
. . .	. . .
(PROGM)[1]	List of .DAT slots used by Program M.
(END)	

2.4 Entry Structure

2.4.1 System Program Parameter Table

The System Program Parameter Table is so arranged that it can be used unaltered in the command table for the system program PATCH, as all entries for Program I (I is greater than or equal to Ø and smaller than or equal to N) can be described in general without knowing which value I takes on. The following discussion will cover all the entries (7*(N+1) of them) with just 7 descriptions.

[1]Note that M need not equal N.

Example: 7-Register System Parameter Block for .SYSLD (an .ABS Program with Values Taken from PDP-15/20 System)

WORD	CONTENTS	DESCRIPTION
1. 2.	.SIXBT /.SYSLD/	Name of system program for use by PATCH (0 if program is relocatable).
3.	56_8	First block occupied by .SYSLD on system device.
4.	11_8	Number of blocks allotted to .SYSLD on system device.
5.	13400_8	First address in .SYSLD (13 bits).
6.	4237_8	Program size of .SYSLD.
7.	15400_8	Starting address of .SYSLD (13 bits).

(1.) WORD I*7+1
(2.) WORD I*7+2

If program I is an .ABS system program, these words contain the .SIXBT name in the PATCH command to select program I for patching; otherwise they must be 0.

(3.) WORD I*7+3

This word contains the first system device block in which .ABS system program I is stored (ignored for relocatable system programs).

(4.) WORD I*7+4

This word contains the number of system device blocks allotted to .ABS system program I (ignored for relocatable system programs).

(5.) WORD I*7+5

This word contains the 13 bit address of the first core location that .ABS system program I occupies (ignored for relocatable system programs).

(6.) WORD I*7+6

This word contains the size of the .ABS system program (ignored for relocatable system programs). The size of a program is defined as the last core location occupied by the program minus the first core location occupied by the program plus 1.

(7.) WORD I*7+7

This word contains the 13 bit starting address of the .ABS system program I (ignored for relocatable system programs).

2.4.2 .DAT Slot Pointer Table

The next M words after the system program parameter table (starting with address DATTAB) contain pointers to lists of .DAT slots used by the system programs loaded by .SYSLD.

(1) WORD DATTAB+I

This word points to the first word in SYSBLK containing the first member of the list of .DAT slots used by system program I (either .ABS or relocatable system programs need this table).

(2) WORD DATTAB+I+1

Points to the last word plus 1 in SYSBLK of the list of .DAT slots used by program I. The .DAT slot numbers are contained in bits 9-17 of the entry words with bits Ø-8 always Ø (bit 9 is the sign bit and negative numbers are expressed in 2's complement notation). As can be seen, this word also represents the first word in SYSBLK containing the list of .DAT slots used by system program I+1.

2.4.3 Table of .DAT Slot Lists

From the end of the .DAT slot pointer table until the end of SYSBLK is the space reserved for .DAT slot lists for each system program as divided by the .DAT slot pointer table.

2.5 SYSBLK for PDP-15/3Ø and PDP-15/4Ø

The following are excerpts from a listing of the SYSBLK used for PDP-15/3Ø and PDP-15/4Ø systems. The listing is used here for illustration only; the numerical values of this "version" may have been altered and cannot be relied upon.

```
        .TITLE B/F SYSTEM BLOCK
/
/ EDIT #Ø       8-18-7Ø
/
/ COPYRIGHT 197Ø, DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
/
/ PDP-9/15 BACKGROUND FOREGROUND SYSTEM
/
/ SPECIAL SYSTEM PROGRAM BLOCK (4Ø) WITH 7-WORD ENTRY PER PROGRAM
/
/ ORDERING OF PROGRAMS IN SYSBLK MUST NOT CHANGE.
/ LOGICAL CODES FOR PROGRAM NAMES ARE THE NUMBER OF THE ENTRY IN SYSBLK
/ THIS BLOCK IS USED BY BFKM9/15 (NON-RESIDENT MONITOR), .SYSLD (SYSTEM
/ LOADER), AND PATCH (SYSTEM PATCHER).
/
/       WD1,2   .SIXBT  'PGNAME'
/       WD3     LOGICAL BLOCK #
/       WD4     # OF BLOCKS RESERVED
/       WD5     LOAD ADDRESS (13 BITS)
/       WD6     PROGRAM SIZE
/       WD7     START ADDRESS (13 BITS)
/
```

```
        .ABS
        .LOC    Ø
COMTAB  DATTAB
/
/ Ø. THIS NUMBER CORRESPONDS TO THE LINKING LOADER OR DDT, BUT BECAUSE
/       THESE PROGRAMS ARE RELOCATABLE THEY ARE NOT REPRESENTED HERE.
/       THIS SPOT IS USED, AS FAR AS PATCH IS CONCERNED, FOR THE RESIDENT
/       MONITOR.
/
        .SIXBT  'RESMON'

        Ø
        4Ø
        1ØØ
        177ØØ
        Ø
/
/ 1.  THE SYSTEM EDITOR PROGRAM
/
        .SIXBT  'EDIT'

        514
        13
        12577
        52Ø1
        13ØØØ
          .
          .
          .
          .
          .
/
/ 24.  THE CROSS REFERENCE PROGRAM SEGMENT OVERLAY TO THE ASSEMBLER
/
        .SIXBT  'CREF'

        .BLOCK  5
/
/ 25.  THE PI VERSION OF THE RESIDENT MONITOR (PDP9 ONLY) [1]
/PARAMETERS UNKNOWN AT THIS TIME
        .IFDEF  PDP9
        .SIXBT  'PIRESM'
        .BLOCK  5
        .ENDC
        .IFUND  PDP9
        Ø
        Ø
        .BLOCK  5
        .ENDC
/
/ TABLE WHICH CONTAINS POINTERS TO .IODEV INFORMATION FOR SYSTEM PROG.
/
DATTAB  LOAD.           /Ø - $LOAD,$GLOAD,$DDT
        EDIT.           /1 - $EDIT
        MACRO.          /2 - $MACRO
        PIP.            /3 - $PIP
        F4.             /4 - $F4
        SRCOM.          /5 - $SRCCOM
        DUMP.           /6 - $DUMP
        UPDTE.          /7 - $UPDATE
        CONV.           /1Ø - $CONV
        MACA.           /11 - $MACROA
        F4A.            /12 - $F4A
        EXEC.           /13 - $EXECUTE
        CHAIN.          /14 - $CHAIN
```

[1] Not applicable to the bank mode system.

```
        PATCH.          /15 - $PATCH
        DTCOP.          /16 - $DTCOPY
        F17.            /17 - NOT USED
        F2Ø.            /2Ø - NOT USED
        F21.            /21 - NOT USED
        F22.            /22 - NOT USED
        END.            /TO TELL WHERE END IS
/
/ .IODEV INFORMATION FOR SYSTEM PROGRAMS
/
LOAD.   -7&777
        -5&777
        -4&777
        -1&777
EDIT.   -15&777
        -14&777
        -1Ø&777
MACRO.  -14&777
        -13&777
        -12&777
        -11&777
        -1Ø&777
PIP.    -1
F4.     -13&777
        -12&777
        -11&777
          .
          .
          .
          .
          .
PATCH.  -14&777
        -10&777
DTCOP.  -15&777
        -14&777
F17.=.
F2Ø.=.
F21.=.
F22.=.
END.=.
```

---

3. .DAT TABLES

3.1 Function

There are 2 .DAT tables: .DATF for Foreground (FGD) and .DATB for Background (BGD). The function of these device assignment tables, as in the ADVANCED Monitor system, is to provide the basis for device-independent programming. Programs which issued monitor calls to perform some I/O function, such as a READ, refer to a slot in the .DAT table instead of a specific I/O device. At program load time, the user has the option of assigning that program's .DAT slots to any I/O device he wishes to use.

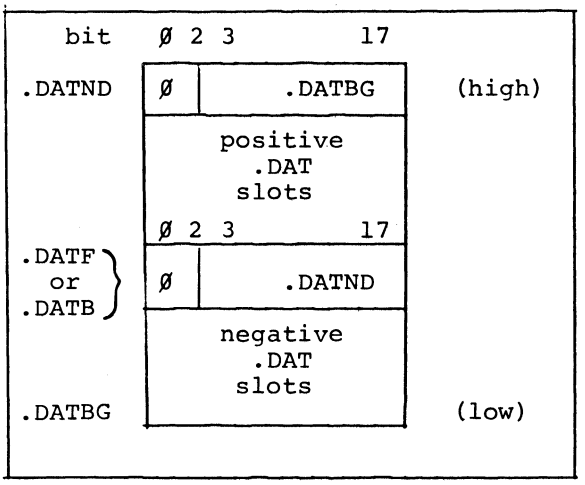
3.2 Location

Both .DAT tables reside in the resident portion of the monitor (RESMON). The .SCOM registers which point to them are:

C(.SCOM+16) = .DATF
C(.SCOM+17) = .DATB

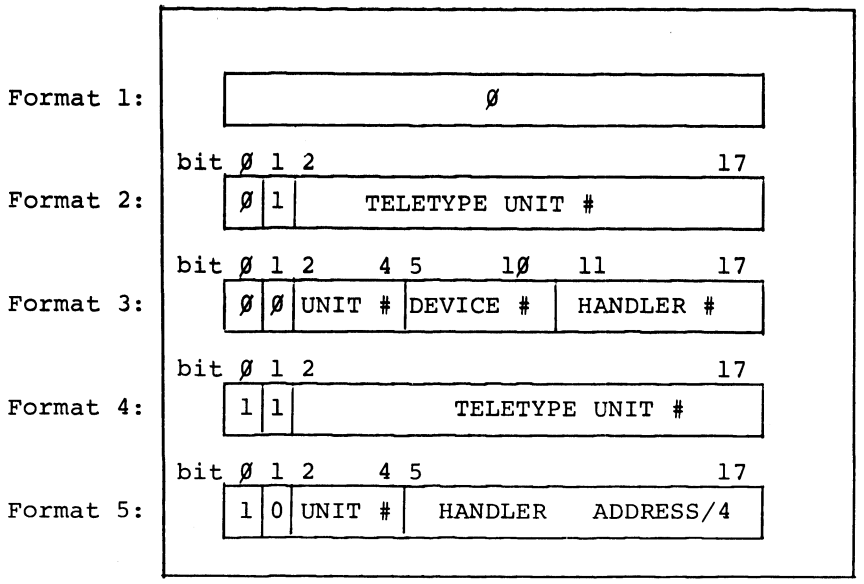
3.3 Table Structure

The FGD and BGD tables are similarly structured. The entry point, either .DATF or .DATB, is equivalent to .DAT slot Ø. Slot Ø contains a pointer to the highest register in the table (.DATND). .DATND, in turn, contains a pointer to the first register in the table (.DATBG). This permits future expansion of table size. Each .DAT slot is a one-word entry. The slots are referenced relative to slot Ø. The most negative slot is at .DATBG; the most positive is at .DATND-1.



3.4 Entry Structure

Depending on what phase of the loading process the system is in and on which I/O device is to be selected, a .DAT slot entry may contain information in any of 5 formats. Background/Foreground .DAT tables do not resemble those of the ADVANCED Monitor system.



Formats 1 through 3 are the pre-setup stage (I/O handler linkage not established); Formats 4 and 5 are setup (I/O handler linkage established, handler is in core and I/O calls to such .DAT slots will be honored by the Monitor). Bit Ø distinguishes Formats 1 through 3 from 4 and 5.

3.4.1 Format 1: - When a .DAT slot contains zero, no device is assigned to it.

3.4.2 Format 2: - Bit 1 set to 1 indicates that this is for the resident Teletype handler (TTA.), which handles the console Teletype (unit Ø) and up to 16_{10} Teletypes on the LT15 or LT19 controls (units 1 - $2Ø_8$). The unit number appears in bits 2 through 17.

3.4.3 Format 3: - Used for all devices except Teletype. The device unit number is in bits 2 - 4 (up to 8 units), the device code is in bits 5 - 1Ø (up to 64_{10} distinct devices), and the device handler code is in bits 11 - 17 (up to 128 handlers for the entire system, not 128 per device). The I/O handler codes are the same as are used to position entries in the .IOC table[1]. The logical device and handler codes bear no relationship to one another. It is up to the System Generator and the Non-resident Monitor to ensure that unit number, device code, and handler code correspond to a legal combination.

3.4.4 Format 4: - Same as Format 2 except that bit Ø is set to 1. In this form it is legal to issue I/O calls from a user program which refers to such a .DAT slot. When set up in this way, the .DAT slot does not contain the address of the Teletype handler. That address (full 15 bit) is permanently stored in .SCOM+34.

3.4.5 Format 5: - Differs from Format 2 as follows: Bit 0=1 to indicate the .DAT slot is set up and ready for use. The unit number remains in bits 2 - 4.

[1]The .IOC table, built by the Non-resident Monitor for the loaders, contains the names of all I/O device handlers coded in radix 5Ø form.

The device and handler codes (bits 5 - 17) are replaced by the address of the I/O handler divided by 4. This means that all I/O handlers, save TTA., must start at a core address evenly divisible by 4. The loaders account for this fact.

4. .IOIN TABLE

4.1 Function

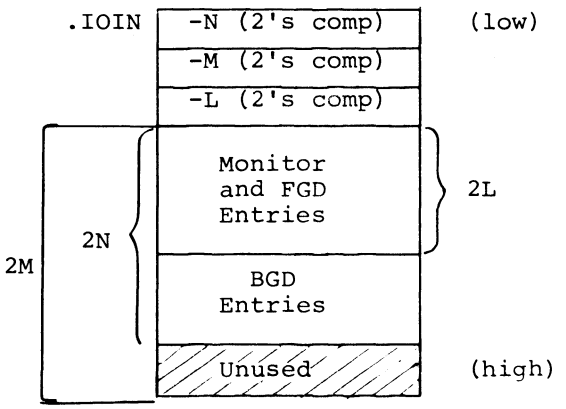
The .IOIN Table maintains a list of the I/O handlers which are in core, showing which device units are in use and which job owns them. This allows the Non-resident Monitor and the loaders to prevent conflicts between Foreground and Background I/O and ensures that only one handler is loaded per device.

4.2 Location

.IOIN is part of the Resident Monitor and is pointed to by .SCOM+13.

4.3 Table Structure

The first three words are two's complement counts.



Legend: N = total # of entries
M = maximum # of entries
L = # of Monitor and FGD entries

The first three entries in .IOIN are the following Monitor entries:

1. The system device handler and unit
2. The Foreground control Teletype
3. The Background control Teletype

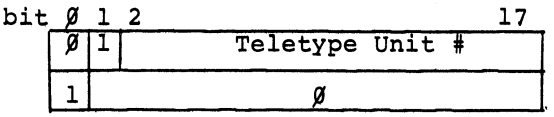
4.4 Entry Structure

Each entry consists of two word registers. The first word contains exactly the

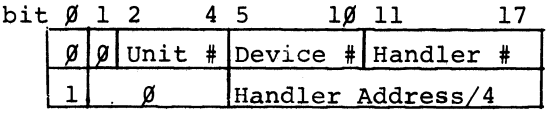
the same information as a .DAT slot before it was set up (i.e., .DAT slot Formats 2 and 3).

EXAMPLE:

Format 1:



Format 2:



4.4.1 Format 1: - This is used for the resident Teletype handler only. Word 2 conveys no information. Bit Ø is set to 1.

4.4.2 Format 2: - Word 2 contains the same information as the .DAT slot would when set up (.DAT slot Format 5), except that the unit number is removed.

5. .MUD TABLE

5.1 Function

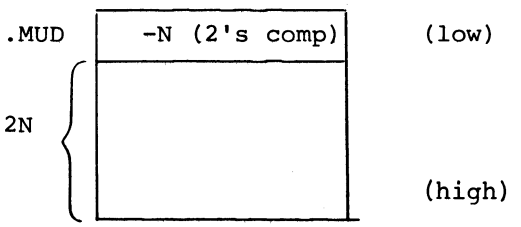
The .MUD table provides information about the multi-user (shareable) device handlers in the system. It indicates, for each handler, the size of external buffer used, the maximum file handling capacity, and the number of open files the user expects to have.

5.2 Location

.MUD is positioned at the very top of the Resident Monitor and is pointed to by .SCOM+14.

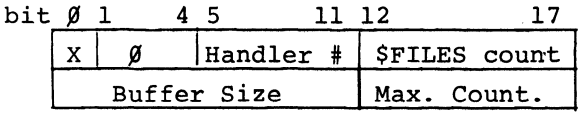
5.3 Table Structure

The first word contains the two's complement count of the number of table entries. Each entry consists of two registers.



5.4 Entry Structure

All entries are similarly structured. The first word contains the logical handler code in bits 5 - 11. There is no assumed relationship between handler code and table position.



If, prior to loading his job, the user issues the $FILES command for a given device (thereby indicating the maximum number of simultaneously open files he expects to have for that device), the count will be entered in bits 12 - 17 of word 1 and bit Ø will be set to 1. Otherwise, those bits are zero.

Word 2 contains the buffer size in bits Ø - 11. A size of zero is legal because some multi-user handlers do not need external buffers. One buffer is needed for each open file. The maximum count, in bits 12 - 17, is the maximum open file handling capacity of the handler. In the absence of a $FILES count declared by the user, the Loaders will compute the count.

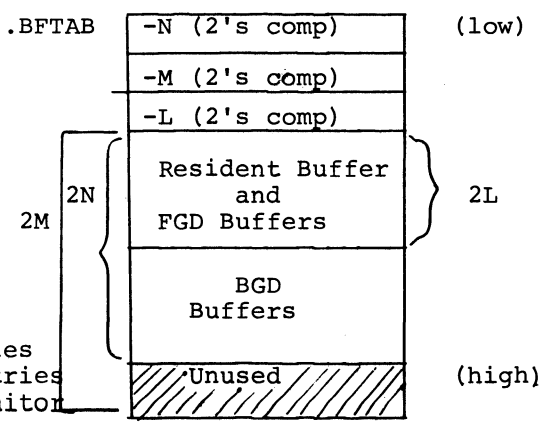
6. .BFTAB TABLE

6.1 Function

.BFTAB is a pool of I/O buffers that are assigned to multi-user device handlers as needed. The table consists of pointers to the actual buffer locations assigned by the loaders.

6.3 Table Structure

The first three words are 2's complement counts.



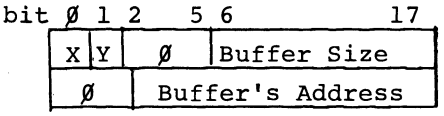
Legend:

N = Total # of entries
M = Maximum # of entries
L = # of FGD and Monitor entries

The first entry is for the resident buffer located within the Resident Monitor.

6.4 Entry Structure

Each entry consists of two words. In word 1 the size of the buffer is in bits 6 - 17. Bit Ø is set to Ø if the buffer belongs to Foreground and set to 1 if the buffer belongs to Background. Bit 1 is set to Ø if the buffer is free and to 1 if the buffer is in use. Word 2 contains a pointer to the buffer's location in core.



HOW TO OBTAIN SOFTWARE INFORMATION

Announcements of new and revised software, as well as programming notes, software problems, and documentation corrections are published by Software Information Service in the following newsletters:

Digital Software News for the PDP-8 and PDP-12

Digital Software News for the PDP-9/15 Family

Digital Software News for the PDP-11

These newsletters contain information to update the cumulative

Software Performance Summary for the PDP-8 and PDP-12

Software Performance Summary for the PDP-9/15 Family

Software Performance Summary for the PDP-11

The appropriate edition of the Software Performance Summary is included in each basic software kit for new customers. Additional copies may be requested without charge.

Any questions or problems on the articles contained in these publications or concerning the use of Digital's software should be reported to the Software Specialist or Sales Engineer at the nearest Digital office.

New and revised software and manuals, and current issues of the Software Performance Summary are available from the Program Library. To place an order, write to:

Program Library
Digital Equipment Corporation
146 Main Street, Building 1-2
Maynard, Massachusetts 01754

When ordering, include the code number and a brief description of the program or manual requested.

Digital Equipment Computer Users Society (DECUS) maintains a user library and publishes a catalog of available programs as well as the DECUSCOPE magazine for its members and non-members who request it. For further information, please write to:

DECUS
Digital Equipment Corporation
146 Main Street, Building 3-5
Maynard, Massachusetts 01754

BACKGROUND/FOREGROUND MONITOR
Programmers Reference Manual
DEC-15-MR3A-D

READER'S COMMENTS

Digital Equipment Corporation maintains a continuous effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback -- your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability, and readability.

Did you find errors in this manual? If so, specify by page.

How can this manual be improved?

Other comments?

Please state your position. _____ Date: _____

Name: _____ Organization: _____

Street: _____ Department: _____

City: _____ State: _____ Zip or Country _____

Fold Here

Do Not Tear - Fold Here and Staple

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Digital Equipment Corporation
Software Information Services
146 Main Street, Bldg. 3-5
Maynard, Massachusetts 01754