digital

pdp9

digital equipment corporation

94-003/011/16

PDP-9

BACKGROUND / FOREGROUND MONITOR SYSTEM

PROGRAMMER'S REFERENCE MANUAL

CONTENTS

CONTENTS (Cont.)

CONTENTS (Cont.)

CONTENTS (Cont.)

SECTION 1
BACKGROUND/FOREGROUND MONITOR

1.1 INTRODUCTION

The reader is assumed to be familiar with the Keyboard Monitor
environment as described in the Advanced Software Monitors Manual,
DEC-9A-MADO-D. It should also be noted that all material presented
herein supersedes the information given in the Monitor Manual.

1.2 BACKGROUND/FOREGROUND MONITOR FUNCTIONS

The Background/Foreground Monitor is designed to control proces-
sing and I/O operations in a real-time or time-shared environment.
It is, essentially, an extension of the Keyboard Monitor and allows
for time-shared use of a PDP-9 by a protected, priority, user FORE-
GROUND program and an unprotected system or user BACKGROUND program.

The Background/Foreground Monitor greatly expands the capabili-
ties of PDP-9 ADVANCED Software and makes optimum use of all avail-
able 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 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 require-

ments, the user's FOREGROUND program could be an Executive capable of handling many real-time programs or subprograms at four levels of priority (with API present).

BACKGROUND processing is essentially the same as the processing normally accomplished under control of the Keyboard 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.

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 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-9 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-unit 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.

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

Since multi-unit device handlers can be shared by FOREGROUND
and BACKGROUND programs, there is a mechanism by which a FORE-
GROUND I/O request will 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 .IDLE or .IDLEC command (Section 6.3 - 6.4).

1-3

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 6.1 for description of real time read .REALR command).

```
.REALR 1, 0,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 FORE-GROUND 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
0	=	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
0, 4, 5, 6, or 7, but is converted to 0 since
the BACKGROUND job can run only on the main-
stream 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 KX09A). 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 the software
since the Monitor and IOPS screen 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-Unit Device Handlers

The Background/Foreground Monitor allows sharing of multi-unit,
mass-storage device handlers (such as, DECtape, Magnetic Tape,
and Disk between BACKGROUND and FOREGROUND jobs). Using these

1-5

multi-unit 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 (because of the .DAT slots not
being used simultaneously), the keyboard command FILES (Sec-
tion 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-unit handlers are capable of stacking one BACKGROUND
I/O request. This provision is made to simulate exactly program
operation as it would occur under Keyboard 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 pre-
occupied 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 re-
quests 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

In hardware configurations which include the Automatic Priority
Interrupt (API) option, the Background/Foreground Monitor allows
convenient use of software priority levels of the API by the FORE-
GROUND 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 FORE-
GROUND 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.

1.3 HARDWARE REQUIREMENTS AND OPTIONS

The following hardware is required to operate the Background/
Foreground Monitor System:

a. Basic PDP-9 with Teletype,

b. Memory Extension Control, Type KG09A,

c. Additional 8192-Word Core Memory Module, Type MM09A,

d. Memory Protection Option, Type KX09A,

e. External Teletype System, including at least*:

(1) One Teletype Control, Type LT09A or LT19A,

(2) One Teletype Line Unit, Type LT09B or LT19B,

(3) One Teletype, Model KSR33, KSR35 or equivalent**,

---

*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. If the
API option is available, a Type LT19A Teletype Control and a Type
LT19B Line Unit are recommended.

**Model 37 Teletypes are not supported. Model 33 or 35ASR Teletypes
are supported only to the extent that they operate as KSR's only;
their paper tape input and output facility cannot be used. LT09's
and LT19's may not both appear in the same configuration.

f. Bulk Storage System, comprising either:

   (1) One DECtape Control, Type TC02, and two DECtape

        Transports, Type TU55 (three recommended), or

   (2) One Disk System, Type RB09 (and one DECtape

        Control, Type TC02, and at least one DECtape

        Transport, Type TU55), or

   (3) One Disk System, Type RF09/RS09 (and one

        DECtape Control, Type TC02, and at least

        one DECtape Transport, Type TU55)

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 MM09B and MM09C (to a maximum of 32,768 words)	Increase the maximum size of both BACKGROUND and FOREGROUND programs that can be handled by the system.
Automatic Priority Interrupt, Type KF09A	Allows for quicker recognition of requests for service by I/O devices.
Extended Arithmetic Element, Type KE09A	Increases speed of arithmetic calculations.
Additional DECtape Transports, Type TU55, or IBM-compatible Magnetic Tape Transports, Type TU20 or TU20A and Tape Control Type TC59	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 TU20 or TU20A transports may be desirable for some applications.

Automatic Line Printer, Type 647	Provides greater listing capabilities.
200 CPM Card Reader, Type CR03B[1]	Allows card input and control cards for BACKGROUND Batch processing.
Additional Teletype Line Units, Type LT09B, (or LT19B) and Teletypes, Type KSR33, KSR35 or equivalent** (up to a maximum of 16[10] LT09B or LT19B units, requiring four LT09A or LT19A controls).	Provides additional output devices if multiple FOREGROUND jobs may require simultaneous output or BACKGROUND jobs wish to use multiple devices.

---

[1] The Type CR01E and Type CR02B Card Readers, although no longer sold by the Company, are supported by software in the BACKGROUND/FOREGROUND System.

Note: The 339 Programmed Buffered Display is supported by software.

SECTION 2

BFKM9 - NON-RESIDENT BACKGROUND/FOREGROUND MONITOR

2.1 INTRODUCTION

BFKM9 is the title of the non-resident portion of the
Background/Foreground Monitor. It is identical in nature to
the Keyboard listening section of the Keyboard Monitor, with
which the reader is assumed to be familiar. BFKM9 reads and
interprets commands typed by the user at a control teletype
(there is one Background control teletype and one 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 $\emptyset$ of the system device;

    b. Allocation parameters, such as, core size,
       number of open files, and I/O devices to be used;

    c. Load a system or user program.

2.2 LOCATION AND WHEN CALLED

BFKM9 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 re-
loaded to start a new Foreground job, the Resident Monitor is
first loaded into lower core from unit $\emptyset$ of the system device,
either by use of the paper tape bootstrap or by typing CTRL C
at the Foreground control teletype. The Resident Monitor then
brings the Non-resident Monitor into the top of memory. When

Operating in the Foreground, BFKM9 runs with memory protect disabled.

After the Foreground user program has been loaded and has started to run, the Non-resident Monitor is re-loaded, with memory protect enabled, to converse with the user at the Background control teletype. BFKM9 is also re-loaded whenever the Background job exits or the user types CTRL C at the Background control teletype.

In both the Foreground and 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.

2.3 INITIAL OPERATION

When BFKM9 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 2. This sets 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 user may assign more free core by issuing the FCORE command, described in section 2.5.3.

b. BFKM9 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

             FKM9 V1A
             $

             The Non-resident Monitor identifies itself
             to the Foreground user by printing FKM9 V1A
             and types $ whenever it is ready to accept
             a command.

When BFKM9 is started for the Background job, it performs

initialization of which the following is of interest:

    a.   It builds the initial configuration of the Back-
         ground .DAT table (.DATB). Any .DATB slots which
         request a single user version of a device handler
         (for example, DTB or DTC) will be changed to the
         multi-unit 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.   BFKM9 will check all Background .DAT slots to
         make certain that they do not conflict with Fore-
         ground I/O. The Resident Monitor contains, for
         this purpose, a table (.IOIN) which lists all
         I/O handlers and unit numbers in use. The follow-
         ing occurs:

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

         (2)   If a single user handler for this device
               is already in core for use by the Fore-
               ground 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-unit 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-unit 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.
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 clear-
ing .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 LPA (line
printer handler - not shareable). The
initial Background .DAT table contains
conflicting requests as follows:

.DAT SLOT	CONTENTS
-15	DTA1
-12	LPAØ
-4	DTA2
3	TTA2

The following will be printed on the Back-
ground control teletype when BFKM9 is first
loaded:

BGD .DATS CLEARED BECAUSE OF FGD I/O:

  -15 -12 -4 3

FCONTROL = TTA1

```
FGD DEV-UNITS:

 TTA2
 DTA1
 DTA2

 BKM9 V1A
 $
```

> FCONTROL indicates which unit is the Foreground
> control teletype. The remainder of the message
> indicates what I/O is being used by the Fore-
> ground job. The Monitor identifies itself to
> the Background job user as BKM9 V1A and signals
> that it is ready to accept a command by printing
> $.

2.4 INFORMATION COMMANDS

The following information commands exist in Background/Fore-

ground:

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 Fore-
ground 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
Keyboard 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
Keyboard 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 Keyboard
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_u.DAT$_u$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 LPA∅
    $

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 (∅ through 7) on the system device.
Directory listings have been altered in BFKM9 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 abbre-
viated 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 = 32301

    FCONTROL = TTA2

    FGD DEV-UNITS:
            DTA1
            LPA0

    $
```

2.5 ALLOCATION COMMANDS

The following commands assign parameters, control 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
NOSHAPE	To nullify effect of SHARE
7CHAN	To specify 7-channel MAGtape operation
9CHAN	To specify 9-channel MAGtape operation
VC38	To load the VC38 character table
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

Keyboard 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, LP for line printer, etc.

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

N is the unit number to go with the device handler and is
also optional. If 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) DTB, DTC, etc., when assigned,
will automatically be changed to DTA. This applies as well to
handler assignments made in the Background whenever the multi-
unit version of the handler is in core for Foreground use.

Background .DAT slot assignments are tested to insure 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 insure that no conflict can occur when assigning the core-
to-core handler, COA., the unit number, independent of what
the user typed, is set to $\emptyset$ for Foreground and 1 for Background.
The core-to-core handler disregards the unit number.

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-
unit device handlers.

The format of the FILES command is:

FILES⌴DD⌴N)

where DD stands for the multi-unit handler or device name (e.g.,
DTA or DT) and N 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 1$\emptyset$.
             (.IODEV 1, 2, 3, ..., 1$\emptyset$). Further, assume that
             all 1$\emptyset$ slots were assigned to DECtape, DTAn (the
             unit numbers are unimportant to this discussion).

Most multi-unit 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 Link-
ing 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-unit handler based on the
maximum number of files that the user programs will
have open simultaneously. Each multi-unit 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-unit 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-unit 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, permanently assembled into the Resident

Monitor, is always available to the Background job. This buffer

is assumed to be as large as the largest I/O device buffer

(6ØØ octal words). In the event that the Background job were to

.IODEV only one .DAT slot which is linked to a multi-unit 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-unit handlers may require a
smaller buffer size than others. If the user were only to use
the former function type, 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 Keyboard 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. It is possible for all of core to be assigned to Foreground. This means, however, that there will be nc room for Background to run, which is perfectly legal. If this is the case, the message:

    SORRY, NO ROOM FOR BGD

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 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
            ABORT

    b. The system is reloaded from the system device.

    c. The Monitor prints

            FKM9 V1A
            $

        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 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 unit N belongs to the Foreground job. Even though unit N may have been assigned to a Foreground .DAT slot, it does not belong to the Foreground job unless it happens to be the Foreground control teletype or unless the Foreground user programs contained an .IODEV to that .DAT slot. This command is also ignored if unit N is already the Background control teletype.

If BCONTROL is issued in the Foreground or if the Background
control teletype is changed because of an FCONTROL command,
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 fol-
lowing 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 (BFKM9) is reloaded for
        Background from the system device

    d.   The Monitor prints

        BKM9 V1A
        $

        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 Keyboard

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 ab-

breviated 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 bulk storage devices (DECtape, Disk, Magtape, etc.) and

that the device handlers are the multi-unit 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 slots 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)

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 VC38 Command (V)

This command is legal in both Foreground and Background and may
be abbreviated by the single letter V. No action is taken until
a command has been given to load a program. At that time, if
the VC38 command was given, the Non-resident Monitor will seek
and load the file VC38TB DMP from unit Ø of the system device.
The VC38 character table is used in conjunction with the 339
display handler, DYA., when the system does not have a VC38
hardware character generator.

The table is loaded into core such that its base address is a
multiple of 1ØØØ octal. The base address is stored as the
first word in the 339 Pushdown List. The address of the Pushdown
List is in .SCOM+12.

The VC38 command given in the Background will be accepted but
ignored if the 339 display handler is assigned to the Foreground.

The format of the VC38 command is:

    VC38)

2.5.12   The MPOFF Command

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

Format:

    MPOFF)

Normally, Background may not modify nor transfer to registers
within the Resident Monitor and the Foreground job; it also
cannot issue IOT's. The MPOFF command signals the Resident
Monitor to set the hardware protect bound to zero and also
allows Background IOT's to be issued.

2.5.13   The MPON Command (M)

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

Format:

    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 Foregound, 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 Keyboard Monitor System.

The following program load commands exist in the Background:

PATCH**)**	LOAD**)**
CHAIN**)**	GLOAD**)**
F4**)**	DDT**)**
F4A**)**	DDTNS **)**
EDIT**)**	SGEN**)**
PIP**)**	DUMP**)**
EXECUTE⊔XXX**)**	UPDATE**)**
	CONV **)**

2.7   FINAL OPERATION

After BFKM9 has received a program load command from either the Foreground or Background, it will bring the System Loader (.SYSLD) into the top of core overlaying BFKM9.   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 BFKM9, 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.

2.9 SUMMARY OF COMMANDS

LEGAL IN F B	ABBREVIATION	COMMAND EXAMPLE
F B	A	ASSIGN␣DTA1␣2, 3/TT1␣1, 4/DT␣-4**)**
F B		BCONTROL␣2**)**
F B	D	DIRECT␣Ø**)**
F		FCONTROL␣1**)**
F		FCORE␣14ØØ**)**
F B	F	FILES␣DT␣3**)**
B	I	INUSE**)**
F B	L	LOG␣....... (ALTMODE)
F B	N	NEWDIR␣5**)**
F B		NOSHARE**)**
F B	R	REQUEST␣XXX**)** or REQUEST␣USER**)** or REQUEST␣.DAT j,k,l**)** or REQUEST**)** .
F	S	SHARE**)**
F	7	7CHAN**)**
F	9	9CHAN**)**
F		MPOFF**)**
F B	M	MPON**)**
F B	V	VC38**)**
B		CHAIN**)**
B		CONV**)**
B		DDT**)**
B		DDTNS**)**
B		DUMP**)**
B		EDIT**)**
F B	E	EXECUTE␣XXX**)**
B		F4**)**
B		F4A**)**
F B		GLOAD**)**
F B		LOAD**)**
B		MACRO**)**
B		MACROA**)**
B		PATCH**)**
F B		PIP**)**
B		SGEN**)**
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 interrrupt 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 tele-
type input.

3.2 CONTROL TELETYPE

In the Background/Foreground System, which may accommodate
up to 17 (decimal) teletype units, 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.

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 initi-
    ate the loading and execution of programs.

b. Certain control characters are recognized only
    at control teletypes; i.e., are ignored if they
    are typed on the other teletype units (see sec-
    tion 3.4 and following).

3.3 TELETYPE HANDLER

The multi-unit 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 descrip-
tion 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 tele-
type. It will be echoed to the teleprinter as ↑C.

If CTRL C is typed at the Background control teletype, the
Background job will be aborted and the Non-resident Monitor will
be loaded to start a new Background job. Foreground is not affec-
ted.

CTRL C typed at the Foreground control teletype aborts both
the Foreground and the Background jobs. In this case, the entire

system is restarted; that is, the Resident Monitor and the Non-resident Monitor are reloaded to start a new Foreground job and the message ABORT is printed on the Background control teletype.

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, extend memory,

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 para-
graphs 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:

3-4

```
LOC            CAL M_8+A_{9-17}
LOC+1          1
LOC+2          P+ADDR_{9-17}
LOC+3          0
```

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

$$M = \text{transfer mode} \quad \begin{matrix} 0 = & \text{Input} \\ 1 = & \text{Output} \end{matrix}$$

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

$$P = \text{priority code} \quad \begin{matrix} 0 & = \text{Normal CTRL P} \\ 100000 & = \text{Mainstream (REAL-TIME)} \\ 200000 & = \text{No change to CTRL P} \\ 300000 & = \text{Priority level of the .INIT} \\ 400000 & = \text{API level 5} \\ 500000 & = \text{API level 5} \\ 600000 & = \text{API level 6} \\ 700000 & = \text{API level 7} \end{matrix}$$

Background requests to an API level (400000 - 700000) 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 0) 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 Keyboard 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 Fore-ground). The Monitor will, when control is at Mainstream, save the status of the Link, extend memory, 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 = $\emptyset$, CTRL P to the given teletype will be disabled; i.e., ignored if typed (except if P = 3$\emptyset\emptyset\emptyset\emptyset\emptyset$).

3.7.2 <u>No Change</u>

If .INIT for a given teletype unit contains the priority code 3$\emptyset\emptyset\emptyset\emptyset\emptyset$, the restart address is ignored and the status of CTRL P to that unit is not changed.

3.7.3 <u>REAL-TIME CTRL P</u>

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 tele-
type 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 Ø Ø4 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 insure that CTRL Q will not overlay useful data, the user
must employ the system program PIP to write a new file direc-
tory 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, Back-
ground 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.

3.1Ø   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,
BFLOAD BIN and EXECUT BIN.

.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 Fore-
ground 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 V1A

>

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

The command string format is nearly the same as for the
Linking Loader in the Keyboard Monitor System. The only

4-1

change is the addition of memory map options, which must pre-
cede the list of user program names. The format is as follows:

>options←mainprog, others,... ALTMODE

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.

4.2.3 Sequence of Operation

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

a. Load all user programs, 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 bank boundaries.

b. If a library search is necessary and the contents of .DATF -5 is non-0, the Loader will seek the user library, .LIBR BIN, via that .DAT slot, and will load all requested library routines 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, .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 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 core banks.

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-unit 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 cf 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 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.

.SYSLD contains a table which lists the .DAT slots used by each system program. Information about the load address, start address, size and initial block number on the system device for each system program is available in block 1Ø1 (SYSBLK).

4-4

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

a. For each .DAT slot (with non-0 contents) required by a system program, it determines which device handlers 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 4.2g. 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), a relocatable file, is loaded starting just above the new hardware protect bound.

d. For all other system programs, .SYSLD builds a short routine just above the hardware protect bound to bring in the program overlaying the System Loader.

e. Finally, .SYSLD exits to the Resident Monitor 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 Loader insures that the relocatable programs do not overlay any of the absolute programs.

The Foreground Linking Loader is also responsible for load-
ing the system program PIP 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.

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 V1A
>
```

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 (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 all user programs 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 defini-
    tions, 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 core boundaries.

4-6

b. Same action as described in 4.2b, using .DATB -5.

c. Same action as described in 4.2c, 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 core starting just above the Foreground job.
    They may, however, be loaded above the Loader
    at this point in time because 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 built by the system program

CHAIN and run by the system program EXECUTE. 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, BFKM9, stores the filename (.SIXBT

format) in .SCOM+1Ø7, 11Ø 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, 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:

     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, load the file via
         .DAT -7. If "XCT", load via .DAT -4. Set the
         extension to "XCT".

     c.   Read the XCT file and check that it was indeed
         built to be run in the Foreground of a PDP-9.

     d.   Save the upper and lower core limits of the over-
         lay structure and check that it does not overlay
         the Resident Monitor.

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

f. Load in EXECUTE.

g. Same action as described in 4.2g 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 4.5.1.b.

c. Read the XCT file and check that it was built to
    be run in the Background of a PDP-9.

d. Save the lower core limit of the overlay structure
    and test, when EXECUTE has been loaded, that they
    do not overlap.

e. Decode the .IODEV bit map in the XCT file and then
    load any I/O handlers needed by the file.

f. Same action as described in 4.2g.

g. Set the hardware memory protect bound above the
    I/O buffers and then load EXECUTE starting above
    this bound.

h. Same action as described in 4.3.e.

4.6 ERROR CONDITIONS

The number of different error messages in the Loaders has been

expanded in Background/Foreground. These are tabulated in Appendix

2, Section A2.5. 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

Memory Map A

```
16K ┌────────────┐
    │ ////////// │ ◄──── System Bootstrap
    ├────────────┤
    │            │
    │            │
    │            │
 8K │─ ─ ─ ─ ─ ─ │
    │            │
    │            │
    │            │
    │            │
  0 └────────────┘
```
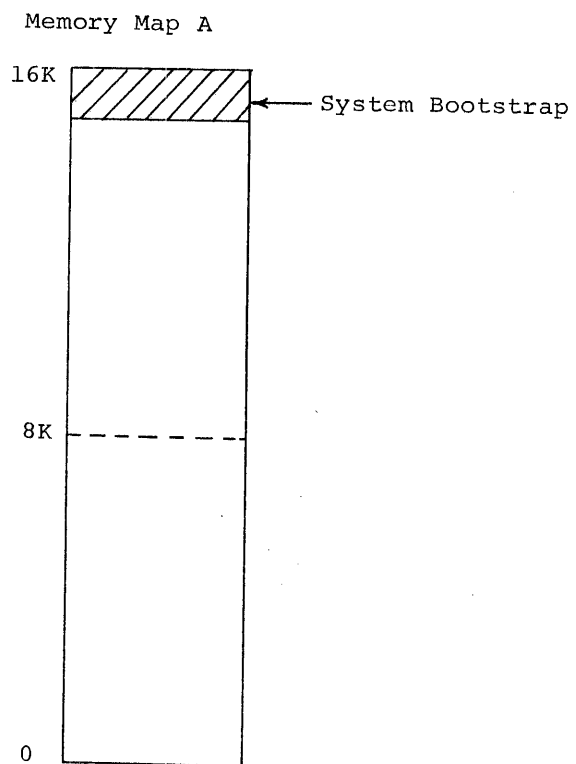
The System Bootstrap is loaded at the top of
core via the paper tape reader in HRM format.

Memory Map B
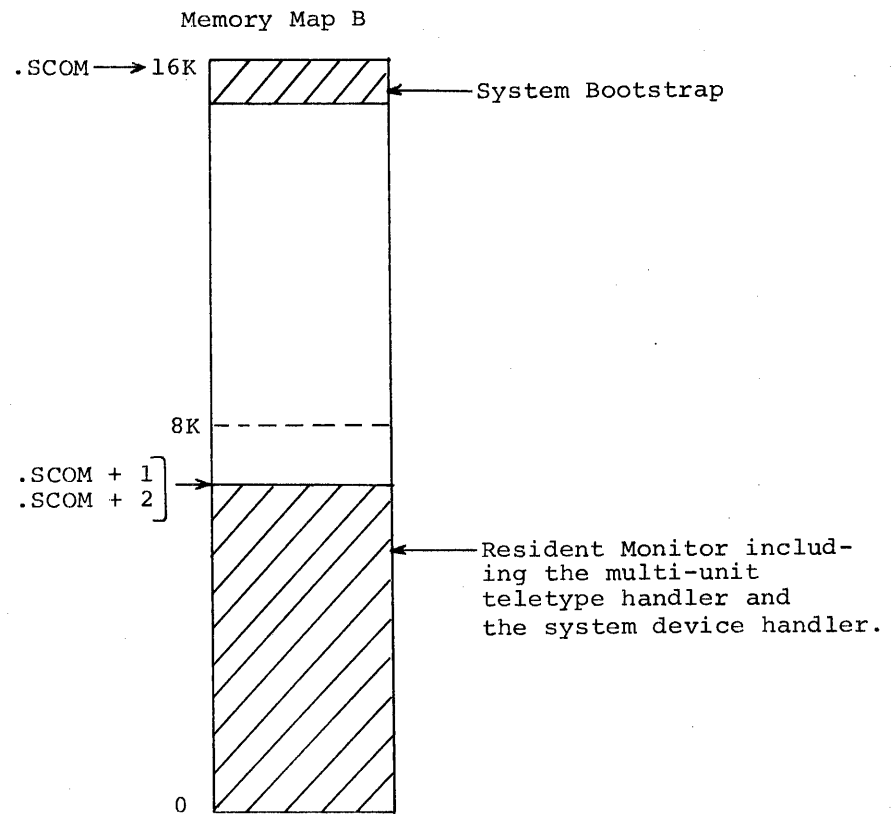
The System Bootstrap automatically loads the Resident
Monitor from the system device into lower core.
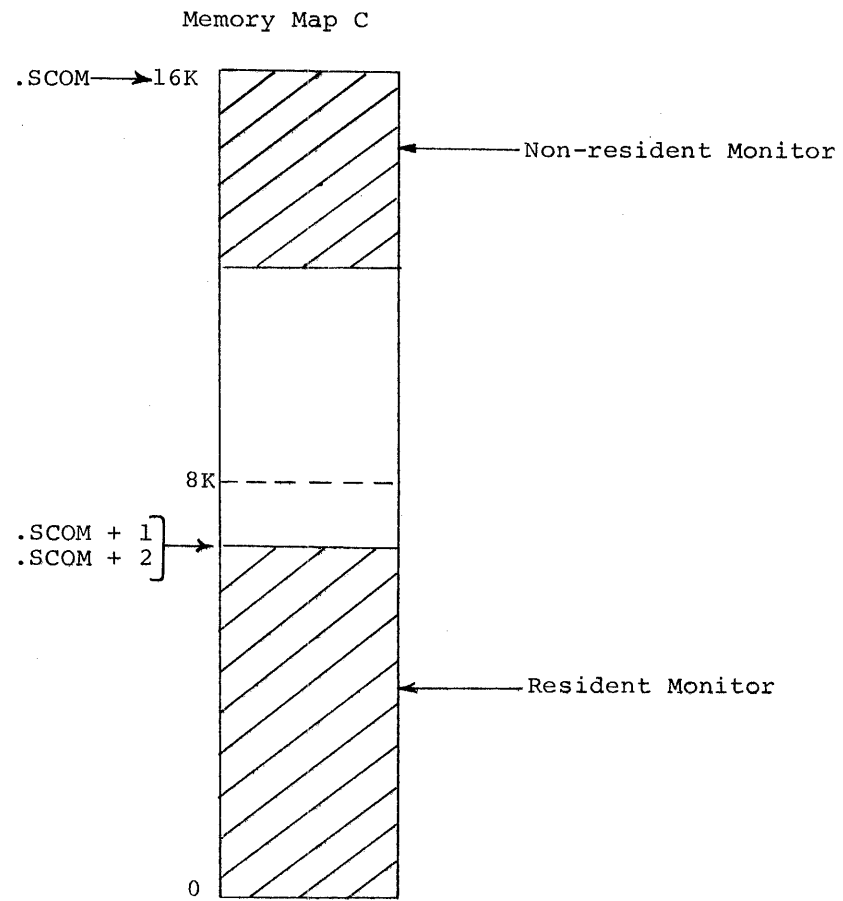
Memory Map C

```
.SCOM ──────▶16K  ┌──────────┐
                   │ ///////  │
                   │ ///////  │◀────── Non-resident Monitor
                   │ ///////  │
                   ├──────────┤
                   │          │
                   │          │
                   │          │
              8K   ├ ─ ─ ─ ─ ─┤
.SCOM + 1  ⎫       │          │
.SCOM + 2  ⎭ ─────▶├──────────┤
                   │ ///////  │
                   │ ///////  │◀────── Resident Monitor
                   │ ///////  │
                   │ ///////  │
               0   └──────────┘
```

The Resident Monitor loads the Non-resident Monitor
(via the resident system device handler) into upper
core, overlaying the System Bootstrap.

Memory Map D


```
.SCOM ——→ 16K   ┌────────────┐
                │ ////////// │ ←———— Foreground Linking
.SCOM + 3 ——→   ├────────────┤        Loader (.SYSLD)
                │            │
                │            │
                │            │
                │            │
           8K   │- - - - - - │
.SCOM + 1 ⎤     │            │
.SCOM + 2 ⎦ ——→ ├────────────┤
                │ ////////// │
                │ ////////// │
                │ ////////// │ ←———— Resident Monitor
                │ ////////// │
                │ ////////// │
           0    └────────────┘
```
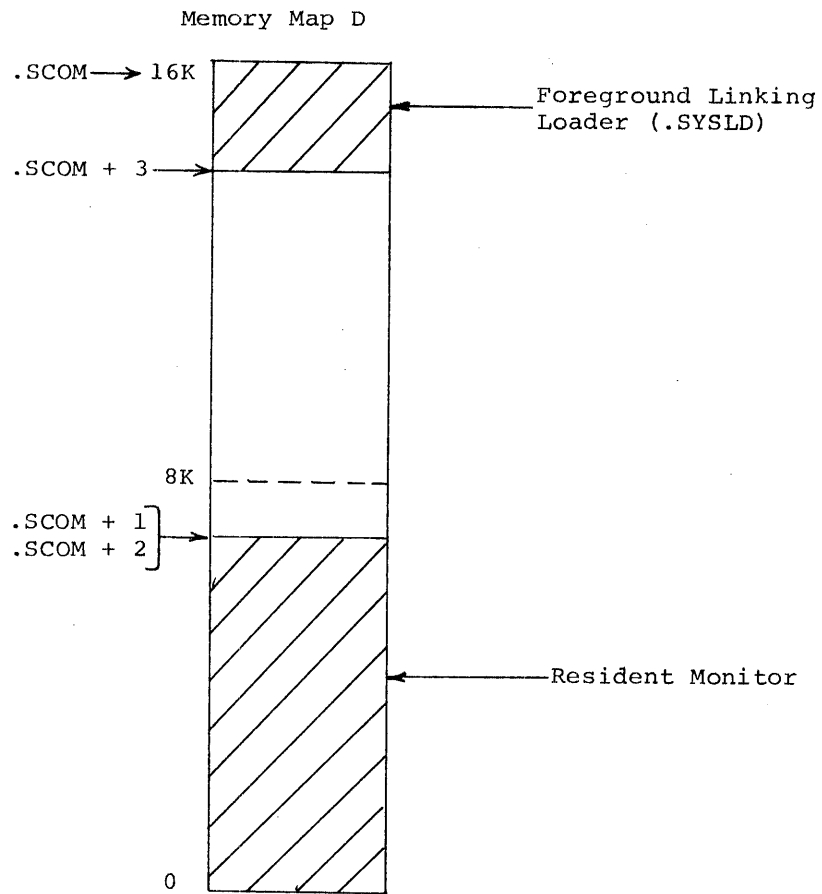
To load a user FOREGROUND program, the Non-resident
Monitor brings in the Foreground Linking Loader
(.SYSLD), overlaying itself.

4-14

Memory Map E

```
.SCOM ──────► 16K  ┌──────────┐
                   │//////////│
                   │//////////│◄─Foreground Linking Loader
                   │//////////│   (.SYSLD)
                   ├──────────┤
                   │          │◄─Loader's I/O Handlers
                   ├──────────┤
                   │//////////│◄─Loader's Symbol Table
                   │          │
                   │    ↓     │
                   │          │
.SCOM + 32 ─────►  ├──────────┤◄─Hardware protect bound
.SCOM + 31 ┐       │          │◄─Software protect bound
.SCOM + 3  ┘─────► │XXXXXXXXXX│◄─Foreground free core
.SCOM + 2 ──────►  ├──────────┤◄─User's I/O Handlers and
                   │          │   I/O Buffers
 Foreground     ⎫  │/─/─/─/─/─│◄─Foreground user programs
    Job       8K⎬  │/         │
                ⎭  │/////     │
                   │//////////│
                   │//////////│◄─Resident Monitor
                   │//////////│
                   │//////////│
                   │//////////│
                 0 └──────────┘
```
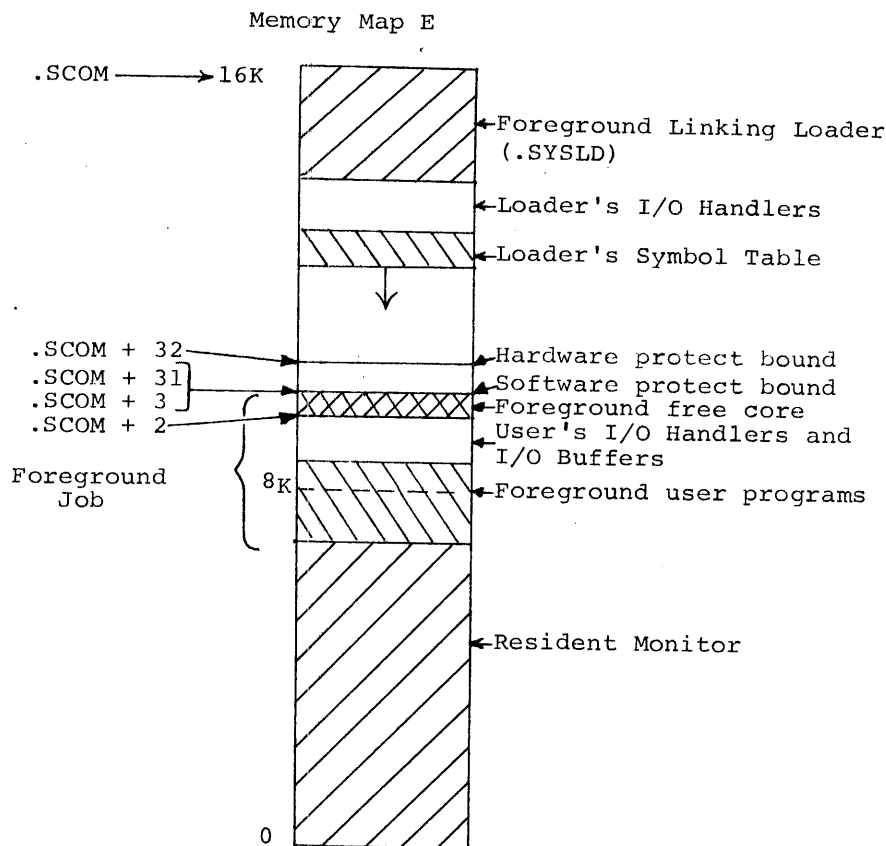
The Foreground Linking Loader first brings in any additional
I/O handlers required for loading. Then it loads the user
program(s), library routines, user I/O handlers and I/O
buffers, and allocates Foreground free core. The software
memory protect bound is established just above the Foreground
job. The hardware memory protect bound, because it can be set
only in increments of 1K 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.

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

Memory Map F

```
.SCOM ────────▶16K ┌──────────┐
                   │//////////│
                   │//////////│◀─Non-resident Monitor
                   │//////////│
                   │//////////│
                   ├──────────┤
.SCOM + 32╲        │          │
.SCOM + 1  ┐───▶   ├─ ─ ─ ─ ─ │◀─Hardware protect bound
.SCOM + 2  ┘───▶   ├──────────┤◀─Software protect bound
.SCOM + 25 ┐       │//////////│  ╲
.SCOM + 31 ┘    8K │/─ ─ ─ ─ /│  │ Foreground job
                   │//////////│  ╱
                   ├──────────┤
                   │//////////│
                   │//////////│
                   │//////////│◀─Resident Monitor
                   │//////////│
                   │//////////│
                 0 └──────────┘
```
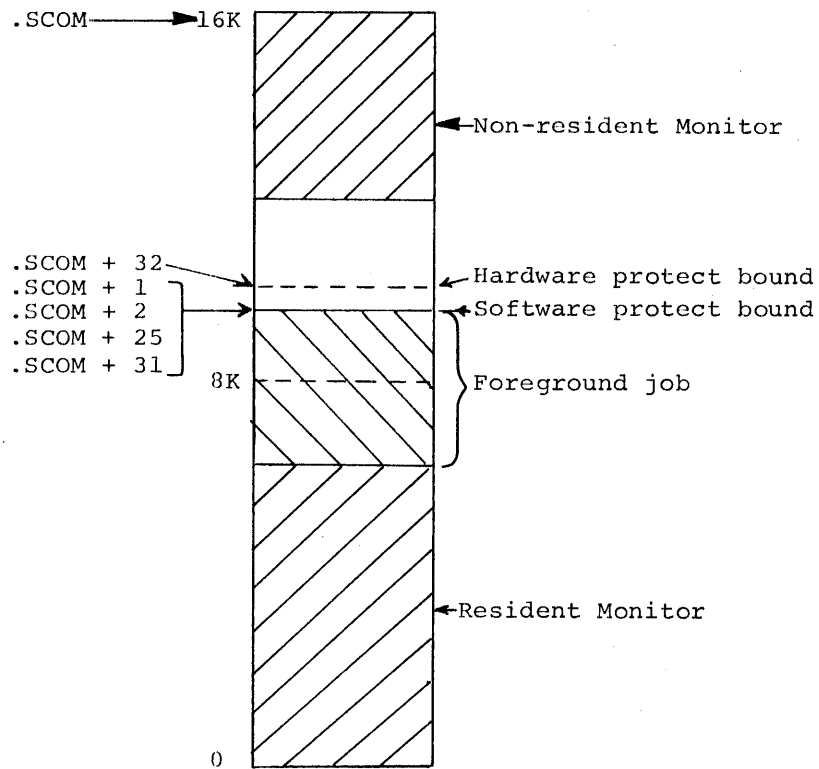
When the FOREGROUND job becomes I/O bound, control is transfer-
red to the BACKGROUND job. The Resident Monitor loads the Non-
resident Monitor (via the resident system device handler) into
upper core. It then gives control to the Keyboard Listener
(within the Non-resident Monitor) to await a BACKGROUND keyboard
command. Memory protect is enabled while the Background job is
running.

Memory Map G

```
.SCOM ─────────► 16K ┌─────────────┐
                     │/////////////│ ◄──── Background System
                     │/////////////│       Loader (.SYSLD)
.SCOM + 3 ──────────►├─────────────┤
                     │             │
                     │             │
.SCOM + 32          │             │
.SCOM + 1   ───────►│─ ─ ─ ─ ─ ─ ─│ ◄──── Hardware protect bound
.SCOM + 2   ───────►├─────────────┤ ◄──── Software protect bound
.SCOM + 25          │/////////////│    ┐
.SCOM + 31          │/////////////│    │
                     │/////////////│    ├─ Foreground Job
                8K ──│─ ─ ─/////////│    │
                     │/////////////│    ┘
                     ├─────────────┤
                     │/////////////│
                     │/////////////│ ◄──── Resident Monitor
                     │/////////////│
                     │/////////////│
                 0 ──└─────────────┘
```
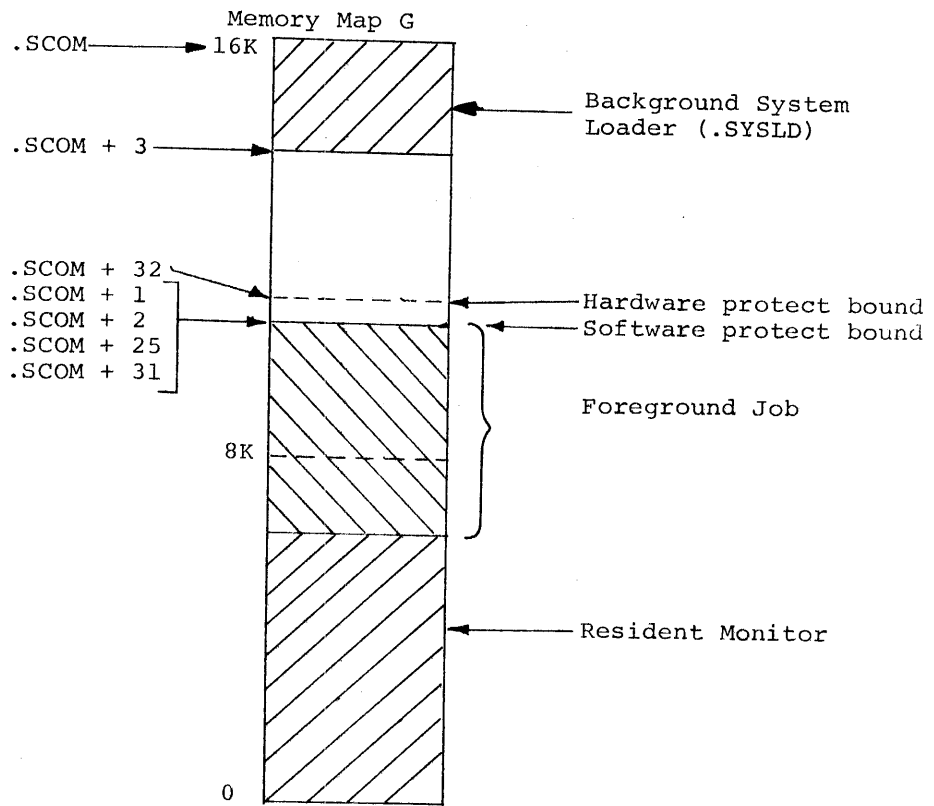
When a BACKGROUND keyboard command requests loading
of a system or user program, the Non-resident Monitor
brings in the System Loader, overlaying itself. Note
that the BACKGROUND System Loader and the FOREGROUND
Linking Loader are physically the same program.

4-17

Memory Map H


```
.SCOM ─────► 16K                                    Background
                                                    System Program

.SCOM + 3 ───►

                                                    Background Free Core

.SCOM + 32 ─►                            ┌────────  Hardware Protect Bound

.SCOM + 2  ─►                            ────────── Software Protect Bound
.SCOM + 31 ─►                            ────────── Background I/O Buffers
.SCOM + 1  ─►                            ────────── Background I/O Handlers
.SCOM + 25 ─►
                                                    Foreground Job

           8K

                                         ────────── Resident Monitor

            0
```
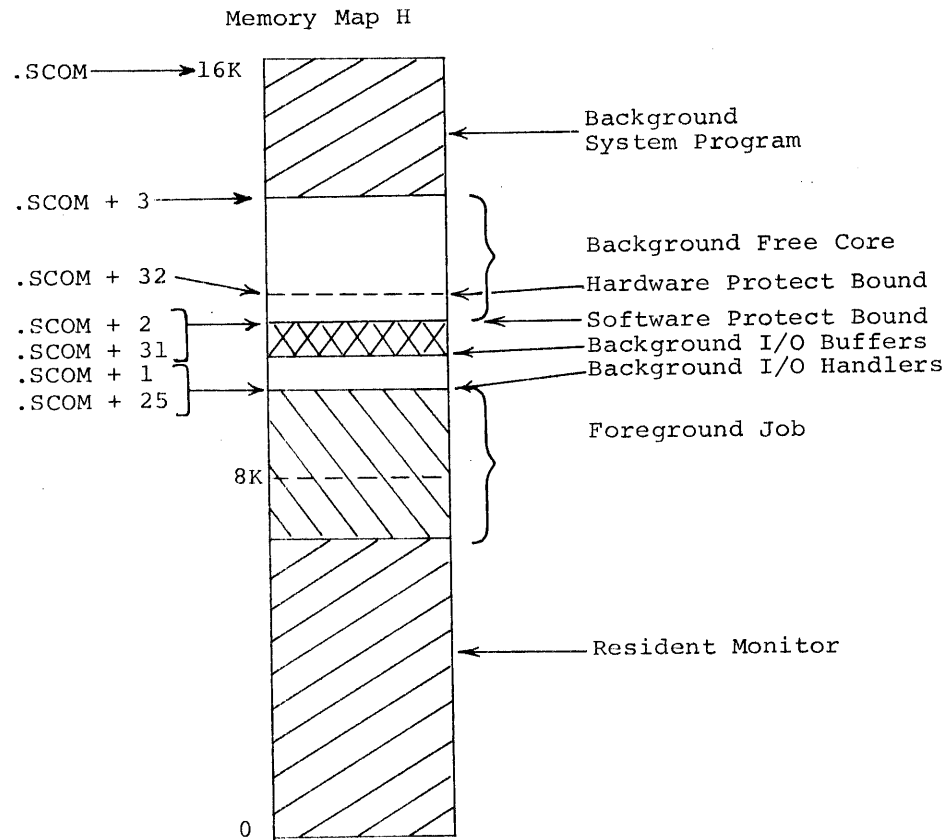
If the BACKGROUND request is for a system program, the
System Loader loads the system program I/O handlers up
from the top of the FOREGROUND job, allocates I/O buffer
space, and loads the system program at the top of core
(overlaying the System Loader). Control is returned to
the Resident Monitor, which sets the memory protect bound
above the buffer space and given control to the system
program.

Memory Map I

```
.SCOM
.SCOM + 3  }────►16K    ┌─────────────┐
                        │/////////////│
                        │/////////////│◄── Background System
                        │/////////////│    Loader (.SYSLD)
                        ├─────────────┤
                        │             │
                        │             │
.SCOM + 2 ──────────►   ├─────────────┤
                        ││││││││││││││││
                        ││││││││││││││││◄──Background Linking Loader
                        ││││││││││││││││
.SCOM + 32 ─────────►   ├─────────────┤◄──Hardware Protect Bound
.SCOM + 1  ⎫            │XXXXXXXXXXXXX│◄──Link Loader's I/O Handlers
.SCOM + 25 ⎬──────►     ├─────────────┤◄──Software Protect Bound
.SCOM + 31 ⎭            │/////////////│⎫
                        │/////////////│⎬ Foreground Job
                 8K ─ ─ │/─ ─ ─ ─ ─ ──│
                        │/////////////│⎭
                        ├─────────────┤
                        │/////////////│
                        │/////////////│
                        │/////////////│◄──Resident Monitor
                        │/////////////│
                        │/////////////│
                  0     └─────────────┘
```
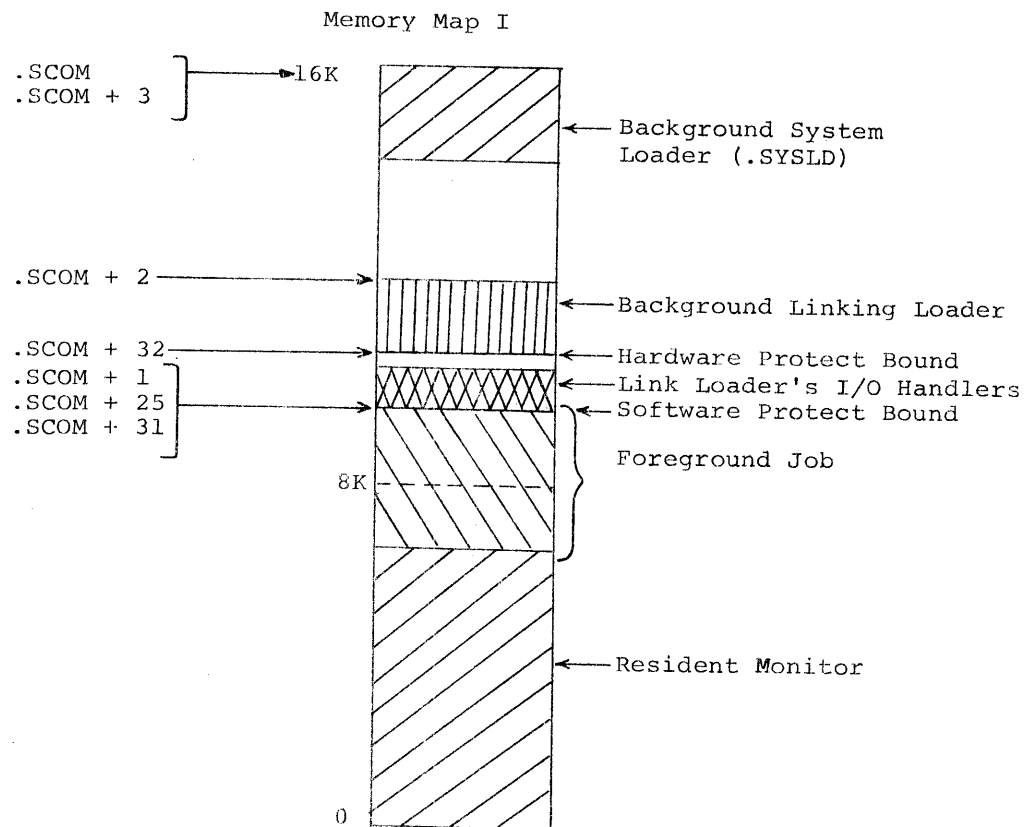
If the BACKGROUND program is a user program*, the
System Loader loads the Linking Loader I/O handlers
up from the top of the FOREGROUND job and loads the
Linking Loader such that the memory protect bound
can be set just below it.

*User programs may be loaded along with the system
program DDT.

Memory Map J

```
.SCOM      ⎫        ►16K                    Background user Programs
.SCOM + 3  ⎭                                Library Routines

                                            Background User I/O
                                            Handlers

.SCOM + 2  ──────────────►                  Loader's Symbol Table

                                            Background Linking Loader

.SCOM + 32 ──────────────►                  Hardware Protect Bound

.SCOM + 1  ⎫
           ⎬ - - - - - - -►                 Software Protect Bound
.SCOM + 25 ⎥
           ⎥
.SCOM + 31 ⎭
                                            Foreground Job
                   8K

                                            Resident Monitor


                    0
```
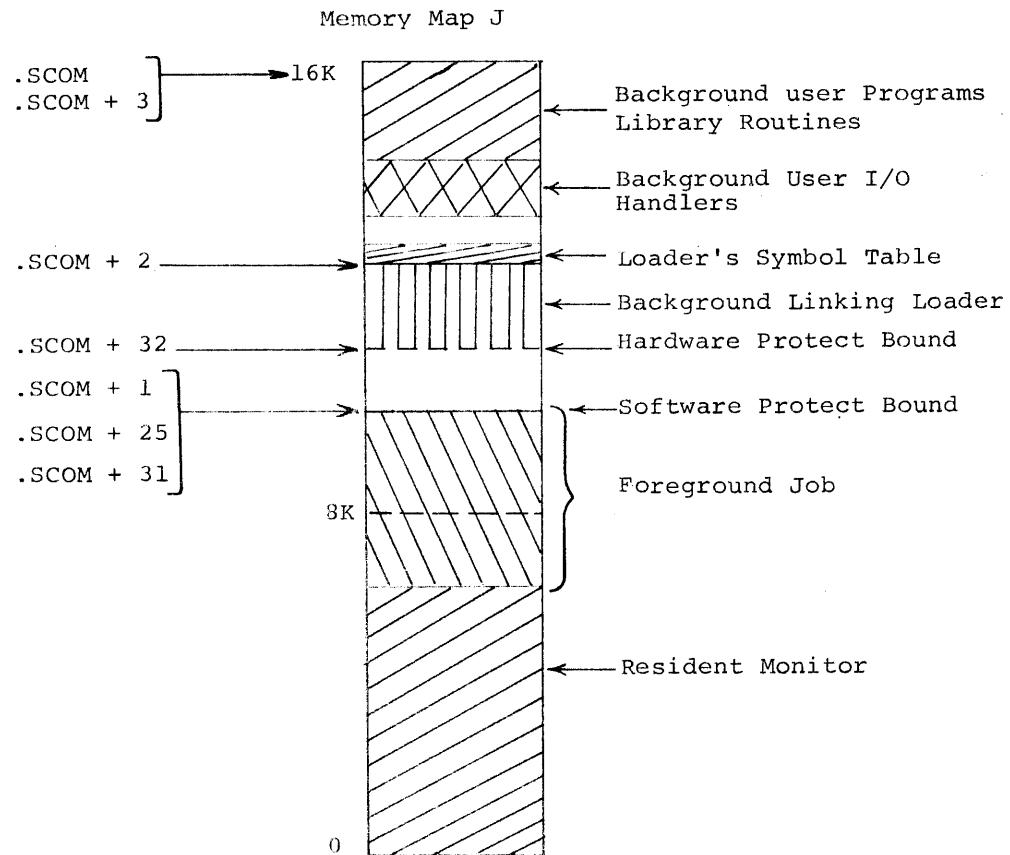
The BACKGROUND Linking Loader overlays the System Loader
by loading user programs down from the top of core. User
I/O handlers, presuming that they cannot fit in core
between the FOREGROUND job and the bottom of the Loader,
are loaded into upper core but relocated to run just above
the FOREGROUND job so that they memory protect bound can
be set above them. Common blocks and I/O buffers are not
shown in this memory map.

4-20

Memory Map K

.SCOM ──────→ 16K

Background User Programs and Library routines

User's Common Blocks

.SCOM + 3 ──────→

Background Free core

.SCOM + 32 ──────→
.SCOM + 2 ──────→ Hardware Protect Bound
.SCOM + 31 ──────→ Software Protect Bound

Background User I/O handlers

.SCOM + 1* ──────→
.SCOM + 25 ──────→

Foreground Job

8K

Resident Monitor

0
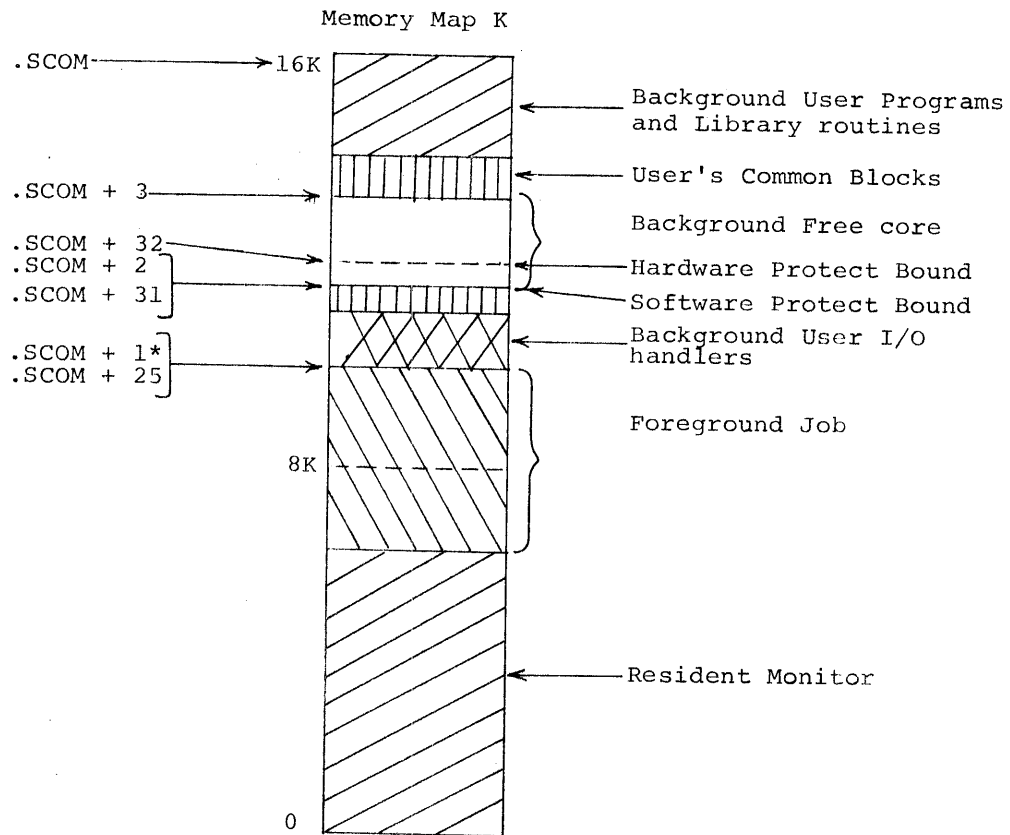
The .EXIT from the Linking Loader causes the user program
I/O handlers to be block transferred to their running
position, the memory protect bound to be set just above
the I/O buffer space, and control given to the user program.
if DDT was also loaded, it resides at the top of core, above
the user programs. Its symbol table, built by the Loader,
is block transferred by the Monitor to start at the soft-
ware protect bound.

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

4-21

Memory Map L


```
.SCOM ──────────► 16K          ◄──── System Loader

                               ◄──── Loader's symbol table

.SCOM + 32 ──────────►         ◄──── Hardware protect bound
.SCOM + 25 and 31 ═══╗         ◄──── Software protect bound
.SCOM + 3 ───────────╫───►     ◄──── Foreground free core
.SCOM + 2 ───────────╨───►
                               ◄──── EXECUTE
                               ┌ I/O Handlers + I/O buffers
                               ◄─┤ including 2nd copy of
                               └ EXECUTE's handler

Foreground ──────────┐           Core occupied by
    Job              }           Overlay structure
                               ┌ 1st copy of EXECUTE's
           8K                  └ I/O handler



                               ◄──── Resident Monitor




            0
```
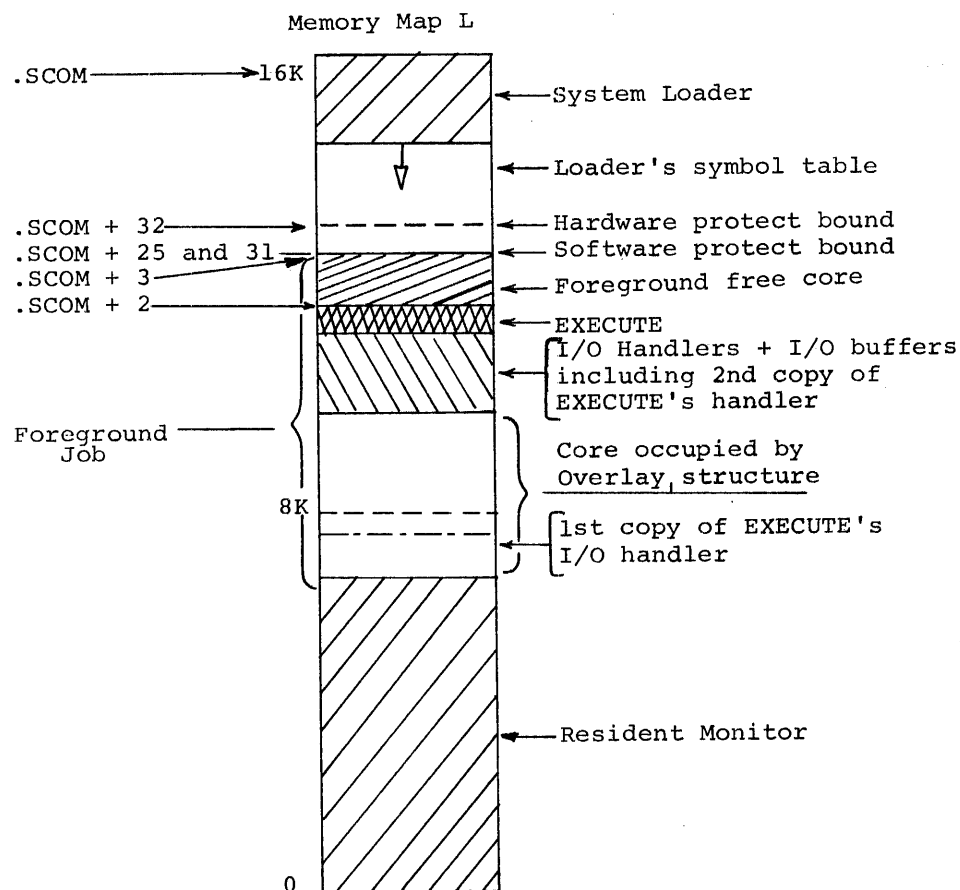
The System Loader first loads EXECUTE's I/O handler (if not the resident handler) in order to read the XCT file. The core limits of the overlay structure are read from the file as well as the request for I/O from its .IODEV bit map. The requested handlers, including a second copy of EXECUTE's handler, 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.

4-22

.SCOM ──────────────────►16K

Core occupied by
Overlay structure

System Loader

Loader's symbol table

.SCOM + 3 ──────────────►
.SCOM + 2 ──────────────►

Free core

EXECUTE

.SCOM + 32 ─────────────►

Hardware protect bound

.SCOM + 31 ─────────────►

Software Protect Bound

.SCOM + 25 ─────────────►

Background I/O handlers
and I/O buffers

Foreground job

8K ── ── ── ──

Resident Monitor

0
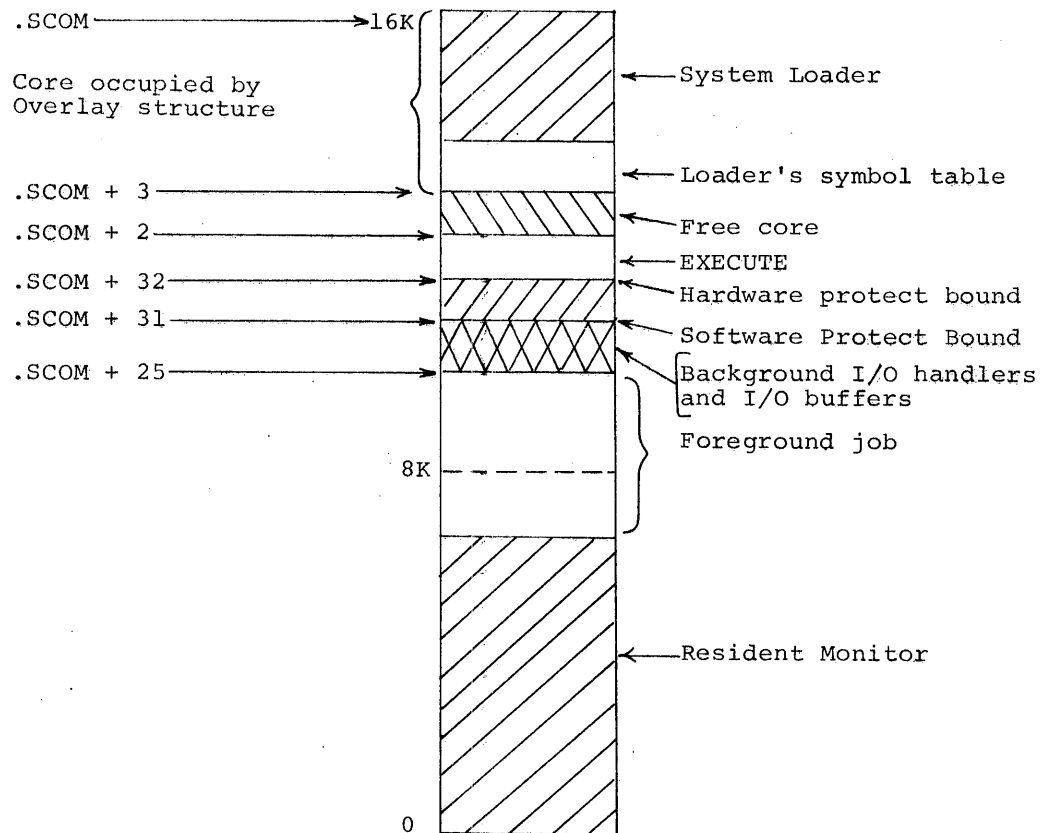
The System Loader loads EXECUTE's I/O handler (if not in core) in order to read the XCT file. The core limits of the overlay structure and the I/O requests in the .IODEV bit map are read from the XCT file. The user's I/O handlers and I/O buffers are then laoded above EXECUTE's handler, and the hardware 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

BACKGROUND/FOREGROUND START-UP PROCEDURE

5.1 LOADING THE B/F MONITOR

Before startup procedures can be carried out, the user must generate a working system (using .SGEN) from the master tape supplied with the system. Refer to Section 8 for .SGEN procedures.

In Disk systems, the Monitor and system programs are assumed to be on Unit Ø.

In DECtape systems, mount the working system tape onto DECtape unit 8 (i.e., Ø) and perform the following:

1) Load appropriate paper tape Bootstrap in the reader.

2) Momentarily depress Reader TAPE FEED pushbutton to clear end-of-tape flag.

3) Set console address switches as follows:

If you have a –	Set switches to –
16K System	37637
24K System	57637
32K System	77637

4) Press and release in sequence the console I/O RESET and READ IN switches.

When loaded, the monitor identifies itself and indicates its
readiness by outputting the following on the Foreground control
Teletype (i.e., normally unit 1)

        FKM9 V1A
        $

The paper tape bootstraps used to load the Background/Foreground
monitor are identical to those used in the PDP-9 Keyboard Monitor
system. The bootstrap restart address, however, is different
(i.e., .SCOM+11 = register 111_8) because the resident Monitor
contains a copy of the bootstrap.

    The three examples given in paragraphs 5.2, 5.3 and 5.4
are intended to get the user "on the air". Note that the symbol
$ is output by the Monitor to indicate that it is ready to accept
commands, whereas">"and "*" are used by system programs to
denote readiness.

    The symbol Ⓢ in the text below indicates the typing of
an ALTMODE terminator.

5.2 .IDLE LOADED AS THE FOREGROUND JOB

    An Idle job is loaded in the Foreground to allow immediate
use of the Background. Refer to section 6.3 for a discussion
of .IDLE.

```
FKM9 V1A
$A  [DTAØ]    -4      /The program "IDLE" is on unit
    [DKAØ]            /Ø of the system device
$GLOAD               /Call the Loader to Load-and-Go

FGLOAD V1A           /The Loader is in core
>←IDLE Ⓢ             /Load "IDLE BIN".
```

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

```
BKM9 V1A          /The Monitor is now ready to
$                 /accept background commands.
```

5.3   SINGLE-USER FOCAL LOADED INTO THE FOREGROUND

```
FKM9 V1A
$A ⎡DTØ⎤ -4         /FOCAL is on the "system tape".
   ⎣DKØ⎦
$A DTI   3         /Library input to FOCAL.
$A TT1   5         /Library output to FOCAL.
$FCORE 1ØØØ        /Free core for FOCAL buffer area.
$GLOAD             /Call Loader to Load-and-Go.

FGLOAD V1A         /The Loader is in core.
>←FOCAL, FNEW ($)  /Load FOCAL and its library, FNEW

FOCAL V3A          /FOCAL is in core and
*                  /is ready to accept commands.
```

Once FOCAL is running in the Foreground, the Non-resident
Monitor will be loaded into core as explained in 5.2.

5.4   TWO-USER FOCAL LOADED IN THE FOREGROUND

```
FKM9 V1A
$A ⎡DTØ⎤
   ⎣DKØ⎦ -4         /FOCAL2  is on the "system tape".
$A  TT1   1        /Teletype for user #1.
$A  DT1   2        /Library In/Out for user #1.
$A  TT2   3        /Teletype for user #2.
$A  DT2   4        /Library In/Out for user #2.
$FCORE 3ØØØ        /Assign 14ØØ (Octal) locations of free
                   /core for each user.
$GLOAD             /Call the Loader to Load-and-Go.

FGLOAD V1A         /The Loader is in core.
>←FOCAL2,FNEW ($)  /Load two-user FOCAL and its
                   /library, FNEW.
FOCAL V3A          /FOCAL is in core and will
*          ·       /identify itself on each user's teletype.
```

5-3

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 are available in addition to the Macros provided in the Advanced Keyboard 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.

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.

----
[1]

See Section 3.7.

6.2   .REALR

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

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

$$M^{1} = \text{Data Mode} \begin{cases} \emptyset = \text{IOPS binary} \\ 1 = \text{Image binary} \\ 2 = \text{IOPS ASCII} \\ 3 = \text{Image Alphanumeric} \\ 4 = \text{Dump Mode} \end{cases}$$

L = 15-bit 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 .REALR is completed.

P = API priority level at which to go to ADDR

P	Priority Level
$\emptyset$	Mainstream
4	Level of .REALR
5	API software level 5
6	API software level 6
7	API software level 7

EXPANSION: LOC $CAL+1\emptyset\emptyset\emptyset\emptyset+M_{6-8}+A_{9-17}$
             LOC+1 $1\emptyset$
             LOC+2   L
                       .DEC   /Decimal Radix
             LOC+3   -W
                       .OCT   /Octal Radix
             LOC+4 $ADDR+P_{0-2}$

---

[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 thus cannot be protected against re-entry.

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 modes of the data to be
transferred, L is the address of the line buffer, W is
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 the
following example.

EXAMPLE 1: STRUCTURE OF A REAL-TIME SUBROUTINE

```
ADDR     0                              /Entry point

         DAC        SAVEAC              /Save AC

         .                             /Any system Macro may
         .                             /be issued at this point.
         .

         LAC        SAVEAC              /Restore AC

         .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)

M^1 = Data Mode

$$\left\{ \begin{array}{l} \emptyset = \text{IOPS binary} \\ 1 = \text{Image binary} \\ 2 = \text{IOPS ASCII} \\ 3 = \text{Image Alphanumeric} \\ 4 = \text{Dump Mode} \end{array} \right.$$

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.

----

[1]Data modes 5, 6 and 7 are passed to all I/O handlers.
[2]See footnote 2, page 6-2.

6-3

P = API priority level at which to go to ADDR

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

EXPANSION: LOC $CAL+10000+M_{6-8}+A_{9-17}$

LOC+1	11
LOC+2	L
	.DEC /Decimal Radix
LOC+3	-W
	.OCT /Octal Radix
LOC+4	$ADDR+P_{0-2}$

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 transfer-
red, 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 sub-
routine 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 proces-
sing; 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+1$\emptyset\emptyset\emptyset$
             LOC+1

DESCRIPTION: 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 sub-
                  routine to handle interrupt at end of
                  interval

            P = API priority level at which to go to ADDR

---

[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 thus cannot be protected against re-entry.

P	Priority Level
$\emptyset$	Mainstream
4	Level of .TIMER
5	API software level 5
6	API software level 6
7	API software level 7

EXPANSION:

LOC	CAL[1]
LOC+1	14
LOC+2	$ADDR+P_0-_2$
	.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 1/60s for 60 Hz systems and 1/50s 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.

6.7 .RLXIT

FORM: .RLXIT ADDR

VARIABLES: ADDR = 13-bit entry point address of the real-time subroutine from which an exit is to be made.

EXPANSION:

LOC	CAL	ADDR
LOC+1	$2\emptyset$	

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 $\emptyset$ of the contents of ADDR.

.RLXIT protects against re-entrance to BACKGROUND or FORE-GROUND Mainstream real-time subroutines. When the contents of ADDR is non-zero, the subroutine is assumed active; .RLXIT sets the contents of ADDR to $\emptyset$, thus making it avail-

---

[1]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.

able 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.

SECTION 7

WRITING DEVICE HANDLERS FOR THE BACKGROUND/FOREGROUND MONITOR SYSTEM

7.1 INTRODUCTION

Writing a handler which will run in the Background/Foreground Monitor environment requires adherence to certain established conventions which differ from those in the Keyboard Monitor environment. 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. There are three types of I/O device handlers that one may wish to develop to operate under 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 they 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 will be primarily devoted to describing the development of single-user handlers. After having done this, it will show the transition from single use to sequential multi-use.

7-1

I/O handler type 3 (Multi-user) is too involved to be presented without example listings (such as our Multi-user DECtape handler) and personal consultation regarding the philosophies of the Background/Foreground Monitor System. Consultation is available to customers whose applications require type 3 handlers.

7.2 FORMAT OF DEVICE HANDLER'S CAL PROCESSOR

The first 37 (octal) words of the I/O handler must have the format described in the following pages. An assembly listing of the Background/Foreground line printer handler (LPA) is appended to this section for reference.

```
WORD Ø:   JMS     SWAP        /SWAP is in the I/O handler
```

The SWAP subroutine must execute WORD5 which restores the state of the program interrupt and DBK from level Ø 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 CAL handler).

```
WORD 1:   Foreground Busy Register --- must be assembled
                                         with Ø contents
              Ø=Not Busy
              NonØ=Busy (Current .DAT slot number, 18 bits
                        if negative)


WORD 2:   Background Busy Register --- must be assembled with
                                          Ø contents
              Ø=Not Busy
              NonØ=Busy (Current .DAT slot number, 18 bits
                        if negative)
```

The CAL handler 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 and proceeds as follows:

1) If the flag indicates that the handler is already busy, the job becomes I/O bound at this level.

2) If the flag indicates not busy, it is set to busy and the CAL handler processes the function and passes the request on to the device 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. If the corresponding flag indicates:

1) BUSY

   a) For .WAIT in the Foreground, control is given to a lower Foreground level or the Background.

   b) For .WAIT in the Background, hang on the CAL.

   c) For .WAITR in either the Background or Foreground, control goes to the address in LOC+2 (which must be above the hardware memory protect bound if in the Background).

2) NOT BUSY - Fall through.

WORD 3: Foreground .CLOSE register -- must be assembled
                                       with ∅ contents
            ∅=.CLOSE not in progress
         NON∅=.CLOSE in progress

WORD 4:   Background .CLOSE register --- must be assembled
                                          with $\emptyset$ contents
                  $\emptyset$=.CLOSE not in progress
              Non$\emptyset$=.CLOSE in progress

WORD 5[1]: ION or IOF (state of PIC on INTERRUPT or CAL entries).

WORD 6[1]: Same as WORD 5 on CAL entries; DBR on INTERRUPT
          entries.

WORD 7[1]: Return pointer. The CAL handler places the
          address of CALXIT in this register.

WORD 1$\emptyset$:   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 2$\emptyset$-32) of the I/O handler
in this register.

WORD 11: User CAL in progress. The CAL handler sets this
          register.
                      $\emptyset$=Foreground
                      1=Background

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

WORD 13: Unit number for Multi-unit devices in Bits $\emptyset$-2,
          with bits 3-17 containing the address of the
          CAL. The CAL handler sets this register.

---

[1]When a hardware interrupt occurs to this I/O device handler, the
interrupt processor must:
    a)   Save the state of the program interrupt in Word 5
    b)   Place a DBR in Word 6
    c)   and place the interrupted program counter (with link,
         extend mode and memory protect bits) in Word 7.

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) All 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 $\emptyset$ 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 is 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                   Ø=IOPS binary
                                            1=image binary
            .WRITE, .REALW - Data Mode 2    2=IOPS ASCII
                                            3=Image ALPHA
                                            4=DUMP
                                            5=DUMP ALPHA
                                            6=are passed on
                                            7=by the CAL handler

            .MTAPE    --MAGtape function    (Ø-17₈)
            .TRAN     --Transfer direction  (Ø-3
            .OPER     --Subfunction code    (1-3)
```

$(\emptyset\text{-}17_8)$
$(\emptyset\text{-}3$

WORD 15:   .INIT --- User restart address plus code bits ($\emptyset$-2)

.READ, .REALR
.WRITE, .REALW --- Line buffer address (checked
        for memory violation on software protect
        bound (SCOM+31) if Background job.

.DLETE, .RENAM
.FSTAT, .ENTER, .SEEK --- Address of Directory
        entry block (checked for memory violation
        on .SCOM+31 if Background job).

.TRAN --- Core starting address (checked for
        memory violation on .SCOM+31 if Background
        job.

WORD 16:   .INIT --- Address of Register which is to have
        standard buffer size placed in it (checked
        for memory violation on .SCOM+31 if Back-
        ground job).

.TRAN, .READ, .REALR -- Line buffer word count
        (from CAL ARG. LIST). Counts are checked
        for core fit and negative value if Back-
        ground job.

.WRITE, .REALW --- Line buffer word count (from
        line buffer word pair ct., except for
        dump mode and Mode 5 which use counts
        from CAL argument list.) Counts are
        checked for core fit and negative value
        if Background job.

WORD 17:   .TRAN --- Device address (Block number)

.FSTAT --- Address of register which will have
        the device code put in bits $\emptyset$-2, (checked
        for memory violation on. .SCOM+31 if
        Background job).

.REALR, .REALW --- Address to give control to on
        completion of I/O request and priority
        level in bits $\emptyset$-2[1], (checked for memory
        violation on .SCOM+32, the hardware protect
        bound, if Background job).

---

[1]If it is a Background CAL, bits $\emptyset$-2 of this register will always
contain $\emptyset$, which indicates Background Mainstream. If it is a
Foreground CAL and there is no API, bits $\emptyset$-2 contain 1, the Fore-
ground Mainstream code.

Function JMP Table

Ignored functions, functions that do not issue IOT's at the
CAL level, and error functions must set up to have the Fore-
ground or Background busy flag (Words 1 and 2, respectively)
cleared during the protected exit routine (which begins at
LPTIO in the line printer handler).

```
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  (.REALR)  /Function 1Ø
WORD 30:      JMP  WRITE (.REALW)  /Function 11
WORD 31:      NOP                  /.WAIT or .WAITR never get to
                                        I/O handler
WORD 32:      JMP  TRAN            /Function 13

WORD 33:      Ø                    /Storage for .SCOM+35, the "in
                                        an interrupt service" flag.
WORD 34:      SUBRF                /Stop FGRD I/O subroutine
```

When the Foreground job terminates (.EXIT, ↑C, terminal error, etc.)
this routine in every Foreground device is called at Mainstream
level to effect the controlled shutting down of the device (see 7.4).

```
WORD 35:      SUBRF                /Stop BGRD I/O subroutine

        For single user device handlers (devices that
        cannot be shared by Foreground and Background),
        the same subroutine can be used for FGRD and
        BGRD STOP I/O.

WORD 36:      Ø                    /Handler I.D. code

        This word has other values (Non-Ø) for devices
        that require special consideration from the CAL
        handler.
```

7.2.1 .SETUP

On the first (and only on the first) .INIT to a device handler,
the device handler must call .SETUP to connect the device handler's
interrupt service routine to the appropriate API channel register
or program interrupt skip chain entries. The address of .SETUP
can be found in .SCOM+55 (155_8).

Calling sequence:

```
LAC*        (.SCOM+55
DAC         LPTEMP
JMS*        LPTEMP
LSDF                    /SKIP IOT
LPINT                   /ADDRESS OF INTERRUPT SERVICE
```

If this is not done, the first hardware interrupt for this device
will be deemed an illegal interrupt and processed accordingly.

7.2.2 Initiating I/O

It is imperative that all IOT's that initiate hardware opera-
tions be executed during protected (API level $\emptyset$, IOF) exit from
the handler to assure that the exit takes place prior to the
hardware operation completing and causing re-entry to the handler
at the interrupt level for servicing.

CAL function requests that require more than one hardware opera-
tion should cause the 2nd through Nth operations to be initiated
at the interrupt level during protected exit. A handler should

not cause sitting on a CAL until the entire function is com-
pleted because this prevents optimum usage of central processor
time for the duration of the function. The user cannot do
other things while the hardware operations proceed.

7.2.3   .OPER Functions

.OPER functions (.FSTAT, .RENAM and .DLETE) are unique in
that they return information in the AC. For device handlers
that wish to utilize this function, the method is as follows:

> On completion of .OPER operation, the interrupt
> service level of the handler sets the appropriate
> close register (Word 3 if Foreground, Word 4 if
> Background) to:
>> 1=File not present
>>
>> INFORMATION+1=File present (where information
>> is the device block number).
>
> The information must not = -1

As at the completion of other I/O requests, it sets up to have
the appropriate busy flag (Word 1 if Foreground, Word 2 if Back-
ground) cleared during protected exit.

7.3   FORMAT OF DEVICE HANDLER'S INTERRUPT PROCESSOR

Figure 7-1 contains a detailed flow chart of the interrupt
service routine of a single-user handler. This is the actual
flow chart of the LPA. handler whose listing is appended to
this section for reference.

Interrupt Processor of Device
Handler in BACKGROUND/FOREGROUND
MONITOR Environment

LPINT

```
┌──────────────────────────────┐
│ Save AC, PC, L, XM, MP       │      ┌─────────────────────────┐
│ in WORD7 of handler.         │─ ─ ─ │entered with memory      │
│ Save status of PIC in WORD5. │      │protect disabled and     │
└──────────────────────────────┘      │at hardware level (API   │
                                       │or PIC) of device        │
                                       └─────────────────────────┘
```

```
┌──────────────────────────────┐
│ Save contents of .SCOM+35    │
│ in WORD33 of handler         │
└──────────────────────────────┘
```
                                       ┌─────────────────────────┐
                                       │When .SCOM+35 is non Ø   │
```
┌──────────────────────────────┐      │at least one interrupt   │
│ non Ø to .SCOM+35            │─ ─ ─ │service is in progress    │
│ (on exit, .SCOM+35 is        │      └─────────────────────────┘
│ returned to its saved state) │
└──────────────────────────────┘
```
                                       ┌─────────────────────────┐
                                       │When location Ø is non Ø │
                                       │at least one interrupt   │
                                       │service is in progress   │
                                       └─────────────────────────┘

```
         ╱─────────╲          ┌───────────────┐
        ╱   PIC     ╲    Y    │ Set location  │
        ╲ interrupt ╱─────────│ Ø to Ø        │
         ╲─────────╱          └───────────────┘
              │
              │ N
              ▼
┌──────────┐  ┌──────────────────────┐
│ Clear    │◄─│ Clear flag, enable PIC,│◄────────┘
│ STOP I/O │  │ DBR to WORD6 of handler│
│ Switch   │  └──────────────────────┘
└──────────┘
     │
     ▼
 ╱─────────╲         ┌───────────┐
╱    Is     ╲   N    │ Process   │
╲ busy flag ╱────────│ Interrupt │
╲    Ø     ╱         └───────────┘
 ╲─────────╱               │
     │                     │
     │ Y                   ▼
     │              ╱──────────────╲
     │             ╱     does       ╲   N    ┌─────────────────────┐
     │             ╲ this complete  ╱────────│ Set up to issue     │      LPNOR
     │             ╲  I/O request  ╱         │ next IOT in pro-    │     ┌───┐
     │              ╲──────────────╱         │ tected (API and     │────►│ 1 │
     │                     │                 │ PIC) exit routine.  │     └───┘
     │        LPEMPT       │ Y               └─────────────────────┘
     │                     ▼                 ┌─────────────────────┐
     │       ┌──────────────────────────┐    │WORD11 of handler    │
     └──────►│ Set up to have BGRD or FGRD│─ ─│indicates BGRD or    │
             │ busy flag cleared during   │    │FGRD ownership of    │
             │ protected exit routine.    │    │I/O request.         │
             └──────────────────────────┘    └─────────────────────┘
                          │
                          ▼
                        ┌───┐
                        │31 │
                        └───┘
```

Figure 7-1   Interrupt Service Routine, Flow Chart

7-10

NOTE:
The address of
CALL4 is in
.SCOM+54 (154₈)
CALL4 initiates
an API (or
pseudo API)
level 4 inter-
rupt with the
level 4 interrupt
processor
controlling
BGRD to FGRD
transitions
and real-
time requests.

LPNOR

Check for device ready
prior to initiating I/O

this is the common
protected exit for
interrupt and CAL
entries

LPTIO

Raise to API level Ø, IOF

this was an
interrupt
service entry

does WORD6
contain a DBR   Y →   JMS CALL 4
                       WORD7
                       WORD33

N

LPT·6

restore .SCOM+35 from
WORD33 of handler

Clear BGRD and/or FGRD busy
flag as previously set up.

LPIOT

execute setup IOT (IOF if
I/O completed)

restore AC DBK from level Ø

execute WORD5 which is the
exit status of PIC

execute WORD6 (DBR if inter-
rupt entry, same as WORD5 if
CAL entry)

exit via WORD7

Figure 7-1   (Cont.)

7-11

```
                                    ┌────┐
                                    │ 31 │
                                    └────┘
                          LPT31        │
                                    ┌───────────────────────┐
                                    │ Raise to API level Ø  │
                                    │ IOF                   │
                                    └───────────────────────┘
                                              │
  NOTE:                                       │        ┌ ─ ─ ─ ─ ─ ─ ─ ─ ┐
  The address of                             │        │ Check if this   │
  IOBUSY is in                               │        │ device involved │
  .SCOM+52 (152₈).              ◇───────────────◇     │ in I/O busy     │
  If the FGRD job              < LAC (WORDØ JMS IOBUSY >─ ─│ situation       │
  became I/O bound              ◇───────────────◇     └ ─ ─ ─ ─ ─ ─ ─ ─ ┘
  on this handler,                            │
  IOBUSY will prime            ┌───────────────────────┐
  the Monitor to               │ ION                   │
  continue the                 │ DBK from API level Ø  │
  FGRD job on the              └───────────────────────┘
  busy CAL.                                 │
                                            │
                                        ◇ real time ◇  N      ╭───╮
                                        < request¹   >───────→│ 1 │  LPNOR
                                        ◇           ◇         ╰───╯
                                            │ Y
                               ┌───────────────────────┐
                               │ Raise to API level Ø  │
  ┌ ─ ─ ─ ─ ─ ─ ─ ┐           │ IOF                   │
  │ Call real-    │           └───────────────────────┘
  │ time processor│                        │
  │ with level/   │                        │
  │ subroutine    │           <───────────────────>
  │ address in AC │─ ─ ─ ─ ─ < LAC WORD17 JMS REALTP >
  └ ─ ─ ─ ─ ─ ─ ─ ┘           <───────────────────>
                                            │
  NOTE:                        ┌───────────────────────┐
  The address of               │ ION                   │
  REALTP is in                 │ DBK from API level Ø  │
  .SCOM+51 (151₈).             └───────────────────────┘
  REALTP primes
  the Monitor to
  honor real-time
  requests.
```

¹To determine whether the completed operation is real-time:

   a)   WORD1Ø must contain a JMP to WORD27 (READ or .REALR) or WORD3Ø (.WRITE or .REALW).

   b)   and WORD17 must be non-Ø.
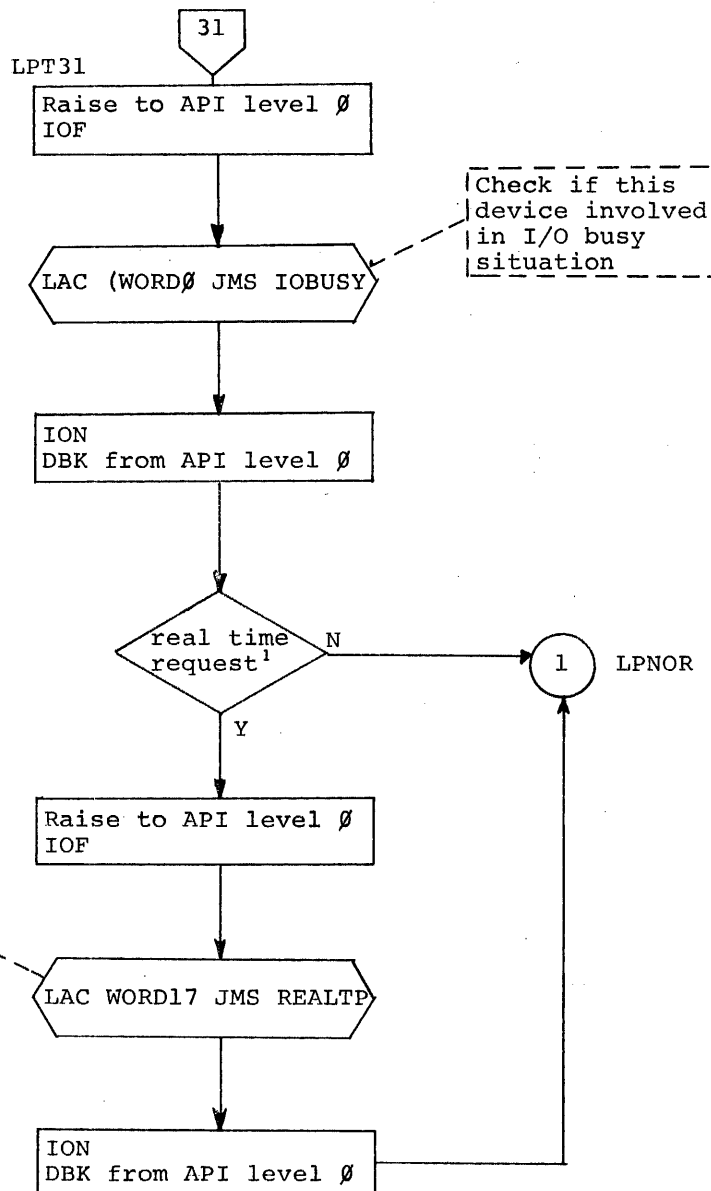
<center>Figure 7-1   (Cont.)</center>

Please note that interrupt service routines (the
code beginning at LPINT in the LPA. handler) must be set up
to operate with or without API.

7.4 SYSTEM ANNOUNCEMENTS

7.4.1 Errors

All device handler error messages should be terminal; that is,
should 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).

Device handlers that wish to set up an error condition should
use the following coding sequence:

```
        LAC*    (.SCOM+66               /POINTER TO ERRORQ
        DAC     TEMP                    /SUBROUTINE.
        LAC     (400200                 /RAISE TO API
        ISA                             /LEVEL 0.
        IOF
        LAW     CODE                    /SEE BELOW.
        JMS*    TEMP                    /CALL ERRORQ.
AUXARG  XX                              /AUXILIARY ARGUMENT.
        DBK                             /RETURN HERE.
        ION
  /CLEAR BUSY FLAG (WORD 1 or WORD 2)
  /EXIT HANDLER VIA PROTECTED EXIT.
```

The first argument, given in the AC to ERRORQ, may be loaded
either by LAW code or by LAC code in the following format:

<u>Code</u>

```
Bits 0-5 are ignored
Bit 6=1 means terminal error
Bit 7=1 means Background error[1]
Bit 8=1 means Foreground error[1]
Bits 9-17 is a 3-digit error code
```

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.

7.4.2 Recovery from I/O Device Not Ready Condition

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 non-terminal error and in their continuation once the condition has been corrected.

---

[1]Bits 7 and 8 may both be set if the error applies to both the Foreground and the Background jobs.

Since all I/O in B/F handlers must be initiated in the common,
protected exit routine of the handler (the code beginning at
LPNOR in the line printer example), it is there and only there
where not-ready conditions should be checked and handled.[1]

Prior to executing the desired IOT, check for not-ready with
the code at LPNOR which is as follows:

```
     LPNOR              /DO WHATEVER IS NECESSARY
                        /TO DETERMINE WHETHER
                        /DEVICE (UNIT) IS READY.
     JMP LPRDY          /DEVICE READY.
     /WITH THE DEVICE (UNIT) NOT READY IT
     /IS NOW NECESSARY TO DEFER THE
     /DESIRED IOT, ANNOUNCE THE NON-
     /TERMINAL ERROR, AND EXIT FROM THE
     /HANDLER SET UP TO CONTINUE WHEN
     /↑R IS TYPED ON THE USER'S CONTROL
     /TELETYPE.
                 LAC      LPIOT        /SAVE DESIRED IOT.
                 DAC      LPIOTB       /
                 LAC      (IOF         /EXECUTE IOF IN
                 DAC      LPIOT        /PLACE OF IOT.
                 JMS      LPMSG        /INITIATE NOT READY MSG.
     LPRDY       .
                 .
                 .
```

Where the code at LPMSG is as follows:

```
/SUBROUTINE TO CALL A ROUTINE IN THE RESIDENT
/MONITOR TO INITIATE A NOT READY MESSAGE.
/CALLING SEQUENCE:
/       JMS LPMSG
/       RETURN WITH LPCTLR NON-Ø IF
/       REQUEST HONORED, OTHERWISE,
/       LPCTLR IS SET TO Ø AND A
/       TERMINAL ERROR WILL RESULT
/
```

---

[1]The exception to this is when a handler can only determine not
ready at the interrupt level; that is, after it has issued the
desired IOT and an error flag results.

```
LPMSG              Ø
       /DETERMINE WHICH JOB (FGRD OR BGRD)
       /IS CURRENTLY MANIPULATING THIS DEVICE.
              LAC       LPA.+11         /Ø=FGRD,1=BGRD
              DAC       LPARG1
              LAC       LPCTLR          /EXIT IF MESSAGE
              SZA!CMA                   /ALREADY REQUESTED
              JMP*      LPMSG           /FOR THIS DEVICE.
              DAC       LPCTLR          /SET↑R FLAG.
              LAC       UNITNO          /UNIT NUMBER (BITS
              DAC       LPARG3          /Ø-2) IF APPLICABLE.
              LAC*      (.SCOM+64       /POINTER TO ↑R
              DAC       LPTMP1          /QUEUER IN MONITOR.
              LAC       (4ØØ2ØØ         /RAISE TO API
              ISA                       /LEVEL Ø AND
              IOF                       /TURN OFF PIC.
              JMS*      LPTMP1          /GO TO ↑R QUEUER.
LPARG1        XX                        /Ø=FGRD, 1=BGRD.
              .ASCII    /LP/            /DEVICE NAME
              .LOC      .-1
LPARG3        Ø                      /UNIT NUMBER (BITS Ø-2) IF APPLICABLE
              LPFRA +   2ØØØØØ /FGRD↑R SUB AND DEVICE'S API LEVEL
              LPFRA +   2ØØØØØ /BGRD↑R SUB AND DEVICE'S API LEVEL
              DZM       LPCTLR /REQUEST NOT HONORED-TERMINAL ERROR.
              ION              /PIC ON (RETURN HERE IF HONORED).
              DBK              /DEBREAK FROM LVØ
              JMP*      LPMSG
```

Where the code at LPFRA is as follows:

```
/SUBROUTINE ENTERED AT API LEVEL 2, PIC OFF.
/WHEN ↑R IS INPUT FROM KEYBOARD, EVEN IF DEVICE IS
/ONLY CONNECTED TO PIC, AN API HARDWARE
/LEVEL (Ø,1,2,OR 3) MUST HAVE BEEN SPECIFIED.
LPFRA              Ø
              DZM       LPCTLR          /CLEAR ↑R FLAG
       /DO WHATEVER IS NECESSARY
       /TO DETERMINE WHETHER DEVICE (UNIT)
       /IS READY
/IF STILL NOT READY, CALL LPMSG
/TO CAUSE NOT READY MESSAGE TO BE
/OUTPUT AGAIN.
              JMS       LPMSG
              JMP       LPFOUT          /BYPASS IOT.
              LAC       LPIOAC          /AC FOR IOT IF APPLICABLE.
LPIOTB        XX                        /DEFERRED IOT.
LPFOUT        ION                       /PIC ON.
              DBR                       /DEBREAK FROM LEVEL Ø.
              JMP*      LPFRA
/THE CONTROL R(↑R)IN PROGRESS FLAG
/MUST INITIALLY BE CLEARED IN THE
/STOP I/O ROUTINE.
LPCTLR             Ø                    /↑R FLAG
```

7.5 STOP I/O TECHNIQUE

In the Background/Foreground Monitor environment, it is
necessary to have some orderly means of stopping I/O that is
in progress. When a job terminates (.EXIT, ↑C, terminal error,
etc.), the Monitor System must assure that all I/O for that job
is shut down before it removes the associated device handlers
from core. This is accomplished via the following method:

    a)   Word 34 of every device handler points to the
          Foreground STOP I/O subroutine which is
          internal to the handler.

    b)   Word 35 of every device handler points to the
          Background STOP I/O subroutine which for
          single-user handlers can be the same as the
          Foreground STOP I/O routine.

    c)   When a job terminates, the Monitor calls the
          appropriate STOP I/O subroutine at Mainstream
          level which actually accomplishes the orderly
          shut down of I/O.

    d)   For devices that can stop I/O hardwarewise,
          via an IOT, this plus steps 4, 5, 6, and 9
          must be done.

For devices that cannot stop I/O hardwarewise via an IOT,
the following procedure can be used:

    1)   Raise to level ∅ of the API and turn off the
          program interrupt to protect against getting
          interrupted in mid-decision.

    2)   Check the ↑R flag. If it is set, clear the
          software flag that will be tested in Step 8
          and bypass Step 3. This is done because no
          I/O is under way if this handler is waiting
          on a ↑R.

3) Check the appropriate busy register (WORD1 or WORD2). If it is not set, no I/O is in progress; therefore, we do not have to wait for its completion. If it is set, set a software flag that will be tested in Step 8.

4) Clear the appropriate busy register (WORD1 or WORD2).

5) Clear the appropriate .CLOSE register (WORD3 or WORD4).

6) Clear the ↑R flag (see Section 7.4.2).

7) Debreak (DBK) from API level Ø and turn on the program interrupt to allow servicing of hardware flags that may have or will occur.

8) If the appropriate busy register had been set, sit in a tight loop testing the software flag that was set in Step 3 above,

```
I.E.    LAC        FLAG
        SZA
        JMP        .-2
```

FLAG is the STOP I/O switch that must be cleared (SET=Ø) by the interrupt service routine on all interrupts that are final. Final means that no other flags will occur without more I/O being initiated via an IOT.

The interrupt service routine must also make a decision whether or not to initiate more I/O. When the appropriate busy register (WORD1 or WORD2) has been cleared, (Step 4) this should indicate that no new I/O should be started. (See the flow chart in Section 7.3).

9) Exit from the STOP I/O subroutine.

7.6   SEQUENTIAL MULTI-USER DEVICE 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:

a)   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 at generation time.

b)   The SWAP subroutine (pointed to by WORD0 of the handler) must set both busy registers (WORD1 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 7.2.

c)   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 job. This is done by testing WORD11, (0=Foreground I/O, 1=Background I/O). They should do nothing if the other job is in control.

In Step 2 of the stop I/O Routine, if the ↑R flag is set, the I/O busy routine in the Monitor (pointed to by .SCOM+52) must be called in case the Foreground job is I/O bound on this device.

d)   Because the SWAP subroutine sets both busy registers (WORD1 and WORD2), the CLEAR BUSY FLAG routine that sets up to have the flags cleared during protected exit from the device handler must always set up to have both flags cleared. The STOP I/O subroutines should also clear both busy registers.

7.6.1 .WAITR

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.

7.7   DEVICE HANDLER LISTING

A listing of the Background/Foreground line printer device
handler (LPA) is given on pages 7-21 through 7-34 of this
section.

```
                                    .TITLE LPA.
                          / .... EDIT #4 .... 2 DEC 69
                          /LPA.---BACKGROUND/FOREGROUND MONITOR SYSTEM.
                          /LPA.=LINE PRINTER (647) HANDLER.
                          /CALLING SEQUENCE:
                                    /CAL+.DAT SLOT (9-17)
                                    /FUNCTION
                                    /N ARGUMENTS, WHERE N IS A FUNCTION OF FUNCTION.
                                    /NORMAL RETURN
              706501  A    LSDF=706501          /SKIP ON DONE FLAG -CONNECTED TO INTERRUPT
              706502  A    LPCB=706502          /CLEAR DONE FLAG, CLEAR PRINTER BUFFER,
                                                /SET DONE FLAG
              706566  A    LPL1=706566          /LOAD PRINTER BUFFER 1 CHAR (AC 12-17)
              706526  A    LPL2=706526          /LOAD PRINTER BUFFER 2 CHAR (AC 6-11, 12-17)
              706546  A    LPLD=706546          /LOAD PRINTER BUFFER 3 CHAR (AC 0-5,6-11,12-8)
              706601  A    LSEF=706601          /SKIP ON ERROR FLAG -NOT CONNECTED TO
                                                /INTERRUPT
              706602  A    LPCF=706602          /CLEAR DONE FLAG
              706606  A    LPPB=706606          /CLEAR DONE FLAG, SELECT PRINTER,
                                                /PRINT BUFFER, CLEAR BUFFER, SET DONE FLAG
              706626  A    LPLS=706626          /LOAD SPACING BUFFER (AC 15-17), SPACE
                                                /SET DONE FLAG

              000003  A    .MED=3
              000100  A    .SCOM=100
                                      .GLOBL LPA.
00000  R   100525  R    LPA.     JMS         SWAP
00001  R   000000  A             0                      /FOREGROUND BUSY REGISTER.
00002  R   000000  A             0                      /BACKGROUND BUSY REGISTER
00003  R   000000  A             0                      /FOREGROUND .CLOSE REGISTER.
00004  R   000000  A             0                      /BACKGROUND .CLOSE REGISTER.
00005  R   740040  A    LPSWCH   XX                     /ION OR IOF
00006  R   740040  A    LPWRD6   XX                     /ION OR IOF OR DBR
00007  R   740040  A    LPTOUT   XX                     /RETURN POINTER
                        /START OF DATA REGISTERS.
                        /FOR SINGLE-USER DEVICES.
00010  R   740040  A    LPWD10   XX                     /JMP FUNCTB
00011  R   740040  A             XX                     /CAL OWNER (0=F,1=B)
00012  R   740040  A    LV2WC    XX                     /.DAT SLOT NUMBER
00013  R   740040  A    LPSVAC   XX                     /UNIT NUMBER (BITS 0-2) CAL ADDRESS (BITS 3-17)
00014  R   740040  A    LPWPC    XX                     /W14
00015  R   740040  A    LPLBHP   XX                     /W15 - LINE BUFFER ADDRESS.
00016  R   740040  A    LP8CT    XX                     /W16
00017  R   740040  A    LPWD17   XX                     /W17
                        /END OF DATA REGISTERS.
                        /BEGINNING OF FUNCTION DISPATCH TABLE.
00020  R   600046  R             JMP  LPIN              /.INIT
00021  R   600523  R             JMP  LPIGN            /.OPER - IGNORED.
00022  R   600512  R             JMP  LPERR            /.SEEK - ERROR.
00023  R   600523  R             JMP  LPIGN            /.ENTER - IGNORED.
00024  R   600523  R             JMP  LPIGN            /.CLEAR - IGNORED.
00025  R   600040  R             JMP  LPCLOS           /.CLOSE
00026  R   600523  R             JMP  LPIGN            /.MTAPE - IGNORED
00027  R   600512  R             JMP  LPERR            /.READ (.REALR) - ERROR
00030  R   600167  R    LWRITE   JMP  LPWRT            /.WRITE (.REALW)
00031  R   740040  A    LV2FC    XX                    /.WAIT (.WAITR) PROCESSED COMPLETELY BY CAL HANDL
```

```
00032  R  620512  R              JMP LPERR              /.TRAN - ERROR
00033  R  740040  A    SCOM35    XX                     /STORAGE FOR .SCOM+35
00034  R  000531  R              LPSTP                  /FGRD STOP I/O SUBROUTINE
00035  R  000531  R              LPSTP                  /BGRD STOP I/O SUBROUTINE
00036  R  000000  A    LPZERO    0                      /HANDLER ID
00037  R  000000  A    LPTMP1    0
                       /.CLOSE LPT ROUTINE
00040  R  140012  R    LPCLOS    DZM LV2WC              /FORM FEED.
00041  R  140434  R              DZM PGECNT             /INITIALIZE #LINES/PAGE COUNTER
00042  R  750001  A              CLA!CMA
00043  R  040435  R              DAC LPIOAC
00044  R  200602  R    LPSPCE    LAC (LPLS              /SET UP TO DO I/O DURING
00045  R  600064  R              JMP LPCOMD             /PROTECTED EXIT.
                       /INIT LPT ROUTINE.
00046  R  750001  A    LPIN      CLA!CMA
00047  R  040012  R              DAC LV2WC
00050  R  040031  R              DAC LV2FC              /SET UP FOR FORM FEED ON INTERRUPT FROM LPCB.
00051  R  200603  R              LAC (64               /52 (DECIMAL)---RETURN
00052  R  060016  R              DAC* LPA.+16                /STANDARD LINE BUFFER SIZE TO USER.
00053  R  220604  R    LPIN2     LAC* (.SCOM+55         / - ONCE ONLY CODE.
00054  R  040053  R              DAC LPIN2
00055  R  120053  R    LP57T     JMS* LPIN2             /.SETUP - THESE 6 REGISTERS ARE OVERLAYED.
00056  R  706501  A              LSDF
00057  R  000350  R              LPINT
00060  R  200062  R    LPWORD    LAC .+2
00061  R  040053  R    LPHRTB    DAC LPIN2
00062  R  600063  R    LP3CHR    JMP .+1
00063  R  200605  R              LAC (LPCB
00064  R  040110  R    LPCOMD    DAC LPIOT
                       /COMMON EXIT SEQUENCE FOR CAL LEVEL
                       /AND INTERRUPT LEVEL ENTRIES.
                       /
                       /
00065  R  100120  R    LPNOR     JMS       LPNRDY       /CHECK IF DEVICE READY.
00066  R  220606  R              LAC*      (.SCOM+54    /ADDRESS
00067  R  040037  R              DAC       LPTMP1       /OF CALL4
00070  R  100571  R    LPTIO     JMS       LPRAIS       /RAISE TO API LEVEL 0, TURN PIC OFF.
00071  R  200006  R              LAC       LPWRD6
00072  R  540165  R              SAD       LPDBR
00073  R  741000  A              SKP                    /INTERRUPT
00074  R  600100  R              JMP       LPT.6        /CAL
00075  R  120037  R              JMS*      LPTMP1
00076  R  000007  R              LPA.+7                 /PC
00077  R  000033  R              SCOM35
00100  R  200033  R    LPT.6     LAC       SCOM35       /RESTORE IN INTERRUPT
00101  R  060607  R              DAC*      (.SCOM+35    /HANDLER FLAG.
00102  R  740000  A    LPFCLR    NOP                    /NOP IF FGRD BUSY FLAG NOT TO BE CLEARED
00103  R  740000  A    LPBCLR    NOP                    /NOP IF BGRD BUSY FLAG NOT TO BE CLEARED
00104  R  200610  R              LAC       (NOP         /RESET
00105  R  040102  R              DAC       LPFCLR       /SWITCHES.
00106  R  040103  R              DAC       LPBCLR
00107  R  200435  R              LAC       LPIOAC       /AC FOR IOT
00110  R  740040  A    LPIOT     XX                     /IOF OR IOT
00111  R  200013  R              LAC LPSVAC             /RESTORE AC.
00112  R  703304  A              DBK                    /FROM LEVEL 0.
```

```
00113  R   420005  R                    XCT  LPA.+5              /ION OR IOF
00114  R   400006  R                    XCT  LPA.+6              /ION OR IOF OR DBR
00115  R   400116  R                    XCT  .+1
00116  R   400117  R                    XCT  .+1
00117  R   620207  R                    JMP* LPA.+7              /RETURN POINTER
00120  R   000000  A       LPNRDY       0
00121  R   706601  A                    LSEF
00122  R   620120  R                    JMP* LPNRDY             /DEVICE READY.
00123  R   200110  R                    LAC  LPIOT       /SAVE IOT.
00124  R   040163  R                    DAC  LPIOTB
00125  R   200574  R                    LAC  LPIOF         /EFFECTIVELY DEFER IOT.
00126  R   040110  R                    DAC  LPIOT
00127  R   100131  R                    JMS  LPMSG        /INITIATE NOT READY REQUEST.
00130  R   620120  R                    JMP* LPNRDY
                                /SUBROUTINE TO CALL A ROUTINE IN THE RESIDENT MONITOR TO
                                /INITIATE A NOT READY REQUEST.
                                /CALLING SEQUENCE:
                                /       JMS LPMSG
                                /       RETURN WITH LPCTLR NON-0 IF REQUEST
                                /       HONORED; 0 OTHERWISE.
                                /
00131  R   000000  A       LPMSG        0
00132  R   200556  R                    LAC       LPCTLR         /+R FLAG.
00133  R   740201  A                    SZA!CMA                  /AVOID DUPLICATE CALL.
00134  R   620131  R                    JMP*      LPMSG          /.ERR 4 ALREADY REQUESTED.
00135  R   040556  R                    DAC       LPCTLR         /SET +R IN PROGRESS FLAG.
00136  R   200011  R                    LAC       LPA.+11        /0=FGRD, 1=BGRD
00137  R   040144  R                    DAC       LPARG1
00140  R   220611  R                    LAC*      (.SCOM+64      /POINTER TO +R QUEUER
00141  R   040037  R                    DAC       LPTMP1
00142  R   100571  R                    JMS       LPRAIS         /TO API LEVEL 0, PIC OFF
00143  R   120037  R                    JMS*      LPTMP1         /GO TO +R QUEUER
00144  R   740040  A       LPARG1       XX                       /0=FGRD, 1=BGRD
00145  R   462400  A                    .ASCII /LP/
00146  R   000000  A
00146  R                                .LOC      .-1
00146  R   000000  A                    0                        /UNIT NUMBER
00147  R   200154  R                    LPFRA+200000             /RETURN ADDRESS AT LEVEL 2
00150  R   200154  R                    LPFRA+200000             /SAME FOR BGRD
                            /
00151  R   140556  R                    DZM       LPCTLR         /TERMINAL ERROR. REQUEST NOT HONORED
00152  R   100576  R                    JMS       LPLOWR         /PIC ON, DEBREAK FROM LV 0
00153  R   620131  R                    JMP*      LPMSG
                            /
                            /SUBROUTINE ENTERED AT API LEVEL 2, PIC OFF, WHEN +R FROM KEYBOARD
                            /IS ASSOCIATED WITH LINE PRINTER.
                            /
00154  R   000000  A       LPFRA        0
00155  R   140556  R                    DZM       LPCTLR         /CLEAR +R FLAG
00156  R   706601  A                    LSEF
00157  R   600162  R                    JMP       .+3
00160  R   100131  R                    JMS       LPMSG          /NOT READY CONDITION
00161  R   600164  R                    JMP       LPIOTB+1       /NOT CORRECTED.
00162  R   200435  R                    LAC       LPIOAC         /AC FOR IOT
00163  R   740040  A       LPIOTB       XX                       /EXECUTE SAVED IOT.
```

```
00164   R   700242   A                     ION
00165   R   703344   A   LPDBR     DBR                        /FROM LEVEL 2
00166   R   620154   R             JMP*       LPFRA
                          /WRITE LPT ROUTINE.
00167   R   750001   A   LPWRT     CLA!CMA
00170   R   040450   R             DAC LP5CH                  /INIT BEFORE CALL TO .LPCONV
00171   R   200014   R             LAC LPA.+14                     /DATA MODE BITS 15-17
00172   R   540612   R             SAD (2
00173   R   600176   R             JMP LPOK                   /IOPS ASCII
00174   R   765007   A             LAW 5007                   /ILLEGAL DATA MODE.
00175   R   600513   R             JMP        LPER06+1
00176   R   220015   R   LPOK      LAC* LPLBHP                /WPC
00177   R   500613   R             AND (377000
00200   R   040014   R             DAC LPWPC
00201   R   140012   R             DZM LV2WC
00202   R   440015   R             ISZ LPLBHP                 /MOVE L.B. POINTER (IN USER'S
00203   R   440015   R             ISZ LPLBHP                 /AREA) TO 1ST DATA WORD.
00204   R   140031   R             DZM LV2FC                  /IOPS ASCII MODE NO FORM CONTROL
00205   R   777770   A             LAW 17770                  /INITIALIZE SWITCH AT LPCONV
00206   R   040212   R             DAC LPCONV
00207   R   777775   A   LPTSTR    LAW 17775                  /3 CHAR. COUNTER FOR
00210   R   040062   R             DAC LP3CHR                 /3/6 WORD.
00211   R   140060   R             DZM LPWORD                 /CLEAR DATA WORD.
                          /THIS ROUTINE GETS THE
                          /NEXT 7-BIT ASCII CHAR.
                          /FROM THE 5/7 LINE BUFFER (USER'S AREA)
                          /IT RETURNS WITH IT RIGHT
                          /JUSTIFIED IN AN OTHERWISE
                          /CLEAR AC.
                          /LP5CH MUST BE SET TO
                          /777777 BEFORE THE INITIAL
                          /CALL TO LPCONV, LPWPC TO WPC INCLUDING HEADER.
                          /LPLBHP TO 1ST DATA WORD IN L.B.(USER'S AREA)
00212   R   777770   A   LPCONV    LAW 17770                  /INITIALIZATION
00213   R   040016   R             DAC LP8CT                  /FOR
00214   R   200614   R             LAC (SAD LPCTAB            /CONTROL CHAR.
00215   R   040304   R             DAC LPVTST                       /SCANNING.
00216   R   440450   R             ISZ LP5CH                  /MODIFIED FOR HOR. TAB.
00217   R   600235   R             JMP LPGET5                 /THIS 5/7 PAIR NOT EXHAUSTED.
00220   R   777000   A             LAW 17000
00221   R   340014   R             TAD LPWPC
00222   R   040014   R             DAC LPWPC                  /SKIP ON NON 0.
00223   R   741300   A             SNA!SPA
00224   R   600301   R             JMP LPEND                  /WORD PAIRS EXHAUSTED
00225   R   220015   R             LAC* LPLBHP                /PICK UP NEXT
00226   R   040055   R             DAC LP57T                  /WORD PAIR
00227   R   440015   R             ISZ LPLBHP
00230   R   220015   R             LAC* LPLBHP
00231   R   040056   R             DAC LP57T+1
00232   R   440015   R             ISZ LPLBHP
00233   R   777773   A             LAW 17773                  /RESET CHAR. COUNTER
00234   R   040450   R             DAC LP5CH                  /FOR THIS WORD PAIR
00235   R   777770   A   LPGET5    LAW 17770
00236   R   040057   R             DAC LP57T+2                /GO THROUGH SHIFT LOOP 7 1/2 TIMES.
00237   R   200056   R   LPGET6    LAC LP57T+1
```

```
00240   R   740212  A               RAL
00241   R   440057  R               ISZ LP57T+2
00242   R   600245  R               JMP .+3
00243   R   500615  R               AND (177              /GOT CHARACTER.
00244   R   600252  R               JMP LPCON1
00245   R   040056  R               DAC LP57T+1
00246   R   200255  R               LAC LP57T
00247   R   740212  A               RAL
00250   R   040255  R               DAC LP57T
00251   R   600237  R               JMP LPGET6
00252   R   540615  R     LPCON1    SAD (177
00253   R   600212  R               JMP LPCONV            /DELETE RUBOUTS.
00254   R   340357  R               TAD LPM40            /-40
00255   R   741100  A               SPA
00256   R   600304  R               JMP LPVTST            /CHAR. <40---CONTROL CHAR.
00257   R   340616  R               TAD (777700          /-100
00260   R   741100  A               SPA
00261   R   340617  R               TAD (40
00262   R   340620  R               TAD (100             /140-176 MAPPED INTO 100-136
00263   R   500621  R     LPCON2    AND (77              /6-BIT TRIMMED.
00264   R   240060  R               XOR LPWORD           /CONSTRUCT 3/6 WORD.
00265   R   440062  R               ISZ LP3CHR           /3 CHARACTER COUNTER.
00266   R   600273  R               JMP LPCON3
                                                         /DATA WORD COMPLETE.
00267   R   440012  R               ISZ LV2WC            /INDEX DATA WORD COUNT
00270   R   040435  R               DAC LPIOAC
00271   R   200622  R               LAC (LPLD
00272   R   600064  R               JMP LPCOMD
00273   R   742010  A     LPCON3    RTL                  /SHIFT CHARS. LEFT
00274   R   742010  A               RTL
00275   R   742010  A               RTL
00276   R   500616  R               AND (777700          /IN CASE LINK WAS ON.
00277   R   040060  R               DAC LPWORD
00300   R   600212  R               JMP LPCONV           /GET NEXT CHAR.
                      /END OF CHARACTER STRING OR CARR. RETURN (IOPS ASCII)
00301   R   200623  R     LPEND     LAC (47
00302   R   040012  R               DAC LV2WC
00303   R   600332  R               JMP LPHT3            /PAD LAST WORD WITH SPACES.
                      /CONTROL CHARACTER ROUTINE. - CHAR. IN AC.
00304   R   740040  A     LPVTST    XX                   /SAD LPCTAB-SAD LPCTAB+7
00305   R   600316  R               JMP LPFORM           /VERTICAL FORM CONTROL CHAR.
00306   R   440304  R               ISZ LPVTST           /SAD LPCTAB+N-SAD LPCTAB+N+1
00307   R   440016  R               ISZ LP8CT
00310   R   600304  R               JMP LPVTST
00311   R   540446  R               SAD LPCTAB+10
00312   R   600301  R               JMP LPEND            /CARRIAGE RETURN.
00313   R   540447  R               SAD LPCTAB+11
00314   R   600321  R               JMP LPHT             /HORIZONTAL TAB.
00315   R   600212  R               JMP LPCONV           /DELETE MEANINGLESS CHAR.
                      /COMPUTE FORM CONTROL CODE AND PLACE
                      /IN LV2FC
00316   R   200016  R     LPFORM    LAC LP8CT
00317   R   040031  R               DAC LV2FC            /L.B.H. IN D.B.
00320   R   600212  R               JMP LPCONV
                      /CONVERT HOR. TAB TO N SPACES, WHERE N IS THE NUMBER
```

```
                                    /NECESSARY TO HAVE THE NEXT CHAR. IN COLUMN 11,21,31,41,51...,111.
       00321   R   220012   R     LPHT      LAC LV2WC               /DATA WORD COUNT
       00322   R   744210   A               CLL!RAL                 /X2
       00323   R   340012   R               TAD LV2WC               /X3
       00324   R   340624   R               TAD (4
       00325   R   340062   R               TAD LP3CHR              /CURRENT WORD CHAR. COUNTER.
       00326   R   340625   R               TAD (777766             /-10 (DECIMAL)
       00327   R   740300   A               SMA!SZA
       00330   R   600326   R               JMP .-2
       00331   R   340425   R               TAD LPOVRP
       00332   R   040061   R     LPHT3     DAC LPHRTB
       00333   R   200626   R               LAC (JMP LPHT2
       00334   R   040212   R               DAC LPCONV
       00335   R   760040   A     LPHT2     LAW 40
       00336   R   440061   R               ISZ LPHRTB
       00337   R   600263   R               JMP LPCON2
       00340   R   777770   A               LAW 17770
       00341   R   040212   R               DAC LPCONV
       00342   R   600212   R               JMP LPCONV
                                    /INTERRUPT HANDLER.
       00343   R   040013   R     LPPIC     DAC LPSVAC              /SAVE AC
       00344   R   220036   R               LAC* LPZERO
       00345   R   040007   R               DAC LPTOUT              /SAVE PC, L, EM, MP
       00346   R   200372   R               LAC LPION
       00347   R   600361   R               JMP LPSTON
       00350   R   600343   R     LPINT     JMP LPPIC               /PIC ENTRY.
       00351   R   040013   R               DAC LPSVAC              /API ENTRY, SAVE AC.
       00352   R   200350   R               LAC LPINT               /PC, L, EM, MP
       00353   R   040007   R               DAC LPTOUT
       00354   R   140350   R               DZM       LPINT         /0=API ENTRY
       00355   R   700314   A               IORS                    /READ I/O STATUS
       00356   R   750100   A               SMA!CLA
       00357   R   777740   A     LPM40     LAW 17740               /PIC OFF -- BUILD IOF
       00360   R   340372   R               TAD LPION               /PIC ON -- ION
       00361   R   040005   R     LPSTON    DAC LPSWCH
       00362   R   220607   R               LAC* (.SCOM+35
       00363   R   040033   R               DAC SCOM35
       00364   R   750001   A               CLA!CMA
       00365   R   060607   R               DAC* (.SCOM+35
       00366   R   200350   R               LAC       LPINT
       00367   R   740200   A               SZA
       00370   R   160036   R               DZM*      LPZERO        /PIC ENTRY
       00371   R   706602   A               LPCF                    /CLEAR LPT DONE FLAG
       00372   R   700042   A     LPION     ION                     /ENABLE PIC
       00373   R   200165   R               LAC LPDBR
       00374   R   040006   R               DAC LPWRD6
       00375   R   140555   R               DZM       LPSTPS                    /CLEAR STOP I/O SWITCH
       00376   R   200001   R               LAC       LPA.+1
       00377   R   340002   R               TAD       LPA.+2
       00400   R   741200   A               SNA                               /DO NOT CONTINUE I/O IF
       00401   R   600451   R               JMP       LPEMPT                  /BOTH BUSY FLAGS ARE  0
                                    /INITIATE MORE OUTPUT IF APPROPRIATE.
       00402   R   200012   R     LPTOK     LAC LV2WC
       00403   R   741200   A               SNA
       00404   R   600451   R               JMP LPEMPT
```

```
00405  R  741100  A              SPA
00406  R  600412  R              JMP LPCLSE
00407  R  540627  R              SAD (50         /BUFFER FULL.
00410  R  600413  R              JMP LPSPPR
00411  R  600207  R              JMP LPTSTR
00412  R  140012  R      LPCLSE  DZM LV2WC
00413  R  777726  A      LPSPPR  LAW 17706       /IS PAGE FULL?
00414  R  340434  R              TAD PGECNT
00415  R  740201  A              SZA!CMA         /YES - FORM FEED
00416  R  200031  R              LAC LV2FC
00417  R  540425  R              SAD LPOVRP
00420  R  600430  R              JMP LPOVER      /OVERPRINT.
00421  R  500630  R              AND (7
00422  R  540630  R              SAD (7
00423  R  140434  R              DZM PGECNT      /INIT #LINES/PAGE CNT
00424  R  040435  R      LPVMOV  DAC LPIOAC      /VERT. SPACING BEFORE PRINTING
00425  R  777776  A      LPOVRP  LAW 17776
00426  R  040031  R              DAC LV2FC
00427  R  600044  R              JMP LPSPCE
                        /WORD COUNT EXHAUSTED.
00430  R  140012  R      LPOVER  DZM LV2WC
00431  R  440434  R              ISZ PGECNT      /INCREMENT #LINES/PAGE CNT
00432  R  200631  R              LAC (LPPB       /PRINT BUFFER.
00433  R  600064  R              JMP LPCOMD
00434  R  000000  A      PGECNT  0               /INITIALIZED TO 0 WHEN AT TOP OF FORM
                                                 /INCREMENTED BY 1 UNTIL 58(10) LINES
                                                 /  HAVE BEEN OUTPUT OR FORM FEED IS ENCOUNTERED
00435  R  000000  A      LPIOAC  0
                                                 /TABLE OF ASCII CONTROL CHAR'S SCANNED BY LPT IN IOPS ASCII MODE
00436  R  777752  A      LPCTAB  777752          /LF-EVERY LINE 0   ---12
00437  R  777761  A              777761          /DC1-EVERY 2ND LINE 1          ---21
00440  R  777762  A              777762          /DC2-EVERY 3RD LINE 2          ---22
00441  R  777753  A              777753          /VT-EVERY 6TH LINE 3           ---13
00442  R  777763  A              777763          /DC3-EVERY 10TH LINE 4         ---23
00443  R  777764  A              777764          /DC4-EVERY 20TH LINE 5         ---24
00444  R  777760  A              777760          /DLE-OVER PRINT 6   ---20
00445  R  777754  A              777754          /FF-TOP ON NEXT FORM 7         ---14
00446  R  777755  A              777755          /CR                 ---15
00447  R  777751  A              777751          /HT                 ---11
00450  R  000000  A      LP5CH   0               /5/7 COUNTER
                        /SET UP SWITCH IN EXIT ROUTINE TO
                        /CLEAR FOREGROUND OR BACKGROUND BUSY REGISTER AS
                        /A FUNCTION OF WORD11, AND PLACE IOF IN LPT
                        /IOT REGISTER SO THAT NO NEW I/O WILL BE STARTED.
00451  R  100476  R      LPEMPT  JMS       CLFLAG
                        /IS THIS DEVICE INVOLVED IN I/O BUSY SITUATION.
00452  R  220632  R      LPT31   LAC*      (.SCOM+52         /ADDRESS OF
00453  R  040037  R              DAC       LPTMP1            /I/O BUSY TESTER
00454  R  100571  R              JMS       LPRAIS            /RAISE TO LEVEL 0 AND TURN OFF PIC
00455  R  200633  R              LAC       (LPA.
00456  R  120037  R              JMS*      LPTMP1
00457  R  100576  R              JMS       LPLOWR
                        /
                        /ROUTINE TO DETERMINE IF THIS I/O
                        /WAS A REAL TIME REQUEST OR NOT.
```

```
00460  R  200017  R              LAC       LPWD17          /NOW 0 IF REAL TIME.
00461  R  741200  A              SNA
00462  P  600065  R              JMP       LPNOR           /NOT .REALW
00463  R  200634  R              LAC       (JMP LWRITE
00464  R  540010  R              SAD       LPWD10          /JMP FUNCTION
00465  R  741000  A              SKP                       /.REALW
00466  R  600065  R              JMP       LPNOR           /NOT REAL TIME REQUEST.
00467  R  220635  R              LAC*      (.SCOM+51       /ADDR. OF
00470  R  040037  R              DAC       LPTMP1          /REAL TIME PROCESSOR
00471  R  100571  R              JMS       LPRAIS
00472  R  200017  R              LAC       LPWD17
00473  R  120037  R              JMS*      LPTMP1
00474  R  100576  R              JMS       LPLOWR
00475  R  600065  R              JMP       LPNOR
                       /SUBROUTINE TO SET UP CLEARING OF THE
                       /APPROPRIATE BUSY FLAG (AT PROTECTED EXIT TIME)
                       /AND NULL (IOF) LINE PRINTER IOT
                       /REGISTER.
00476  P  000000  A    CLFLAG    0
00477  R  200574  R              LAC       LPIOF           /IOF
00500  R  040110  R              DAC       LPIOT
00501  R  200011  R              LAC       LPA.+11         /WORD 11 OF LIVE REGS.
00502  R  740200  A              SZA                       /0=FGRD, 1=BGRD
00503  R  600507  R              JMP       .+4
00504  R  200543  R              LAC       LPFBSY          /FOREGROUND
00505  R  040102  R              DAC       LPFCLR
00506  R  620476  R              JMP*      CLFLAG
00507  R  200544  R              LAC       LPBBSY          /BACKGROUND
00510  R  040103  R              DAC       LPBCLR
00511  R  620476  R              JMP*      CLFLAG
                       /
       000512  R       LPERR=.
00512  R  765006  A    LPER06    LAW       5006            /ILLEGAL FUNCTIONS
00513  R  040037  R              DAC       LPTMP1
00514  R  200013  R              LAC       LPSVAC
00515  R  040566  R              DAC       LPTAUX
00516  R  200011  R              LAC       LPA.+11
00517  R  740200  A              SZA
00520  R  200636  R              LAC       (3000           /BGRD
00521  R  240037  R              XOR       LPTMP1
00522  R  100557  R              JMS       LPTERR
                       /
00523  R  100476  R    LPIGN     JMS       CLFLAG          /CLEAR BUSY FLAG
00524  R  600065  R              JMP       LPNOR
                       /THIS SUBROUTINE IS EXECUTED BY THE
                       /CAL HANDLER VIA WORD 0 OF THIS I/O
                       /HANDLER JUST PRIOR TO GIVING CONTROL
                       /TO THE HANDLER AT THE APPROPRIATE
                       /ENTRY IN THE FUNCTION DISPATCH TABLE.
00525  R  000000  A    SWAP      0
00526  R  400005  R              XCT       LPA.+5          /ION OR IOF
00527  R  703304  A              DBK                       /FROM LEVEL 0
00530  R  620525  R              JMP*      SWAP
                       /STOP I/O SUBROUTINE
00531  R  000000  A    LPSTP     0
```

```
00532  R  100571  R              JMS     LPRAIS            /PROTECT
00533  R  200556  R              LAC     LPCTLR            /DO NOT HANG IF
00534  R  750200  A              SZA!CLA                   /+R IN PROGRESS.
00535  R  600542  R              JMP     LPCLER
00536  R  200001  R              LAC     LPA.+1            /IF I/O IS UNDER WAY SET
00537  R  340002  R              TAD     LPA.+2            /STOP SWITCH.
00540  R  750200  A              SZA!CLA
00541  R  740001  A              CMA
00542  R  040555  R      LPCLER  DAC     LPSTPS
00543  R  140001  R      LPFBSY  DZM     LPA.+1            /CLEAR I/O BUSY SWITCHES.
00544  R  140002  R      LPBBSY  DZM     LPA.+2
00545  R  140003  R              DZM     LPA.+3            /CLEAR CLOSE SWITCHES.
00546  R  140004  R              DZM     LPA.+4
00547  R  140556  R              DZM     LPCTLR            /+R FLAG
00550  R  100576  R              JMS     LPLOWR            /ALLOW INTERRUPTS.
00551  R  200555  R              LAC     LPSTPS            /WAIT UNTILL I/O IS DONE.
00552  R  740200  A              SZA
00553  R  600551  R              JMP     .-2
00554  R  620531  R              JMP*    LPSTP
00555  R  000000  A      LPSTPS  0                         /STOP I/O SWITCH.
00556  R  000000  A      LPCTLR  0                         /+R IN PROGRESS IF NON-0.
                                 /
                                 /
                                 /SUBROUTINE TO CAUSE OUTPUTTING OF ERROR MESSAGE.
00557  R  000000  A      LPTERR  0
00560  R  040564  R              DAC     LPTLAW            /LAW CODE
00561  R  220637  R              LAC*    (.SCOM+66
00562  R  040037  R              DAC     LPTMP1
00563  R  100571  R              JMS     LPRAIS            /RAISE TO API LEVEL 0 AND TURN OF PIC.
00564  R  740040  A      LPTLAW  XX                        /LAW CODE
00565  R  120037  R              JMS*    LPTMP1
00566  R  740040  A      LPTAUX  XX
00567  R  100576  R              JMS     LPLOWR
00570  R  620557  R              JMP*    LPTERR
                                 /
                                 /SUBROUTINE TO RAISE TO API LEVEL 0
                                 /AND TURN OFF PIC.
00571  R  000000  A      LPRAIS  0
00572  R  200640  R              LAC     (400200
00573  R  705504  A              ISA
00574  R  700002  A      LPIOF   IOF
00575  R  620571  R              JMP*    LPRAIS
                                 /SUBROUTINE TO DEBREAK FROM API LEVEL 0
                                 /AND TURN ON PIC.
00576  R  000000  A      LPLOWR  0
00577  R  703304  A              DBK
00600  R  700042  A              ION
00601  R  620576  R              JMP*    LPLOWR
       000000  A              .END
00602  R  706626  A *LIT
00603  R  000064  A *LIT
00604  R  000155  A *LIT
00605  R  706502  A *LIT
00606  R  000154  A *LIT
00607  R  000135  A *LIT
```

```
00610   R   740220   A  *LIT
00611   R   000164   A  *LIT
00612   R   000002   A  *LIT
00613   R   377000   A  *LIT
00614   R   540436   R  *LIT
00615   R   000177   A  *LIT
00616   R   777700   A  *LIT
00617   R   000040   A  *LIT
00620   R   000100   A  *LIT
00621   R   000077   A  *LIT
00622   R   706546   A  *LIT
00623   R   000047   A  *LIT
00624   R   000004   A  *LIT
00625   R   777766   A  *LIT
00626   R   600335   R  *LIT
00627   R   000050   A  *LIT
00630   R   000007   A  *LIT
00631   R   706606   A  *LIT
00632   R   000152   A  *LIT
00633   R   000000   R  *LIT
00634   R   600030   R  *LIT
00635   R   000151   A  *LIT
00636   R   003000   A  *LIT
00637   R   000166   A  *LIT
00640   R   400200   A  *LIT
```

NO ERROR LINES

```
CLFLAG        20476      P
LPARG1        00144      R
LPA.          00000      R
LPBBSY        00544      R
LPBCLR        00103      R
LPCB          706502     A
LPCF          706602     A
LPCLER        00542      R
LPCLOS        00040      R
LPCLSE        00412      R
LPCOMD        00064      R
LPCONV        00212      R
LPCON1        00252      R
LPCON2        00263      R
LPCON3        00273      R
LPCTAB        00436      R
LPCTLR        00556      R
LPDBR         00165      R
LPEMPT        00451      R
LPEND         00301      R
LPERR         000512     R
LPER06        00512      R
LPFBSY        00543      R
LPFCLR        00102      R
LPFORM        00316      R
LPFRA         00154      R
LPGET5        00235      R
LPGET6        00237      R
LPHRTB        00061      R
LPHT          00321      R
LPHT2         00335      R
LPHT3         00332      R
LPIGN         00523      R
LPIN          00046      R
LPINT         00350      R
LPIN2         00053      R
LPIOAC        00435      R
LPIOF         00574      R
LPION         00372      R
LPIOT         00110      R
LPIOTB        00163      R
LPLBHP        00015      R
LPLD          706546     A
LPLOWR        00576      R
LPLS          706626     A
LPL1          706566     A
LPL2          706526     A
LPMSG         00131      R
LPM40         00357      R
LPNOR         00065      R
LPNRDY        00120      R
LPOK          00176      R
LPOVER        00430      R
LPOVRP        00425      R
LPPB          706606     A
```

LPPIC	00343	R
LPRAIS	00571	R
LPSPCE	00044	R
LPSPPR	00413	R
LPSTON	00361	R
LPSTP	00531	R
LPSTPS	00555	R
LPSVAC	00013	R
LPSWCH	00005	R
LPTAUX	00566	R
LPTERR	00557	R
LPTIO	00070	R
LPTLAW	00564	R
LPTMP1	00037	R
LPTOK	00402	R
LPTOUT	00007	R
LPTSTR	00207	R
LPT.6	00100	R
LPT31	00452	R
LPVMOV	00424	R
LPVTST	00304	R
LPWD10	00010	R
LPWD17	00017	R
LPWORD	00060	R
LPWPC	00014	R
LPWRD6	00006	R
LPWRT	00167	R
LPZERO	00036	R
LP3CHR	00062	R
LP5CH	00450	R
LP57T	00055	R
LP8CT	00016	R
LSDF	706501	A
LSEF	706601	A
LV2FC	00031	R
LV2WC	00012	R
LWRITE	00030	R
PGECNT	00434	R
SCOM35	00033	R
SWAP	00525	R
.MED	000003	A
.SCOM	000100	A

```
LPA.        00000     R
.MED        000003     A
LPSWCH      00005     R
LPWRD6      00006     R
LPTOUT      00007     R
LPWD10      00010     R
LV2WC       00012     R
LPSVAC      00013     R
LPWPC       00014     R
LPLBHP      00015     R
LP8CT       00016     R
LPWD17      00017     R
LWRITE      00030     R
LV2FC       00031     R
SCOM35      00033     R
LPZERO      00036     R
LPTMP1      00037     R
LPCLOS      00040     R
LPSPCE      00044     R
LPIN        00046     R
LPIN2       00053     R
LP57T       00055     R
LPWORD      00060     R
LPHRTB      00061     R
LP3CHR      00062     R
LPCOMD      00064     R
LPNOR       00065     R
LPTIO       00070     R
LPT.6       00100     R
.SCOM       000100     A
LPFCLR      00102     R
LPBCLR      00103     R
LPIOT       00110     R
LPNRDY      00120     R
LPMSG       00131     R
LPARG1      00144     R
LPFRA       00154     R
LPIOTB      00163     R
LPDBR       00165     R
LPWRT       00167     R
LPOK        00176     R
LPTSTR      00207     R
LPCONV      00212     R
LPGET5      00235     R
LPGET6      00237     R
LPCON1      00252     R
LPCON2      00263     R
LPCON3      00273     R
LPEND       00301     R
LPVTST      00304     R
LPFORM      00316     R
LPHT        00321     R
LPHT3       00332     R
LPHT2       00335     R
LPPIC       00343     R
```

```
LPINT      00350      R
LPM42      00357      R
LPSTON     00361      R
LPION      00372      R
LPTOK      00402      R
LPCLSE     00412      R
LPSPPR     00413      R
LPVMOV     00424      R
LPOVRP     00425      R
LPOVER     00430      R
PGECNT     00434      R
LPIOAC     00435      R
LPCTAB     00436      R
LP5CH      00450      R
LPEMPT     00451      R
LPT31.     00452      R
CLFLAG     00476      R
LPERR      000512     R
LPER06     00512      R
LPIGN      00523      R
SWAP       00525      R
LPSTP      00531      R
LPCLER     00542      R
LPFBSY     00543      R
LPBBSY     00544      R
LPSTPS     00555      R
LPCTLR     00556      R
LPTERR     00557      R
LPTLAW     00564      R
LPTAUX     00566      R
LPRAIS     00571      R
LPIOF      00574      R
LPLOWR     00576      R
LSDF       706501     A
LPCB       706502     A
LPL2       706526     A
LPLD       706546     A
LPL1       706566     A
LSEF       706601     A
LPCF       706602     A
LPPB       706606     A
LPLS       706626     A
```

SECTION 8

SYSTEM GENERATION

The system utility program .SGEN, used to tailor a
Background/Foreground tape to operate in different hard-
ware configurations, is not available at this time.

.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 + $\emptyset$
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 about 115_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 + $\emptyset$ (F)   Pointer to the highest register in core
                                 (37777, 57777, or 77777).

.SCOM + 1

    (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

Bits indicate machine configuration:

(F)	Bit $\emptyset$	$\emptyset$=No API; 1=API
	Bit 1[1]	$\emptyset$=No EAE; 1=EAE
(F)	Bits 2-5	$\emptyset$ (Reserved and unused)
	Bit 6[2]	$\emptyset$=7-channel MAGtape
		1=9-channel MAGtape
(F)	Bit 7	$\emptyset$=Bank Mode (PDP-9)
(U)	Bits 8-13	Unassigned
(F)	Bit 14	1=Background/Foreground
(F)	Bits 15-17	Drum size for RM$\emptyset$9=
		$\emptyset$=No drum
		1=32K (RM$\emptyset$9A)
		2=65K (RM$\emptyset$9B)
		3=131K(RM$\emptyset$9C)
		4=262K(RM$\emptyset$9D)
		5=524K(RM$\emptyset$9E)

---

[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 + 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 ∅ - 2 indicate which command was given to the Monitor:

$LOAD, $GLOAD, $DDT, or $DDTNS.
Bit ∅ = 1 if $DDT or $DDTNS (DDT load)
Bit 1 = ∅ if $LOAD; Bit 1 = 1 if $GLOAD
Bit 2 = ∅ 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 (∅-2) remain as in (a).

.SCOM + 7

The interrupted PC plus L,XM,MP are saved here for DDT in the Background when CTRL T has been typed.

.SCOM + 1∅ (S)

The interrupted PC plus L,XM,MP are saved here after a NORMAL CTRL P has been typed and honored.

.SCOM + 11 (F)

Bootstrap restart instruction.

.SCOM + 12 (F)

Pointer to the 339 Pushdown list within the Resident Monitor.

.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 $\emptyset$, in the Resident Monitor.

.SCOM + 17 (F)   Pointer to .DATB[4], Background .DAT slot $\emptyset$, in the Resident Monitor.

.SCOM + 2$\emptyset$ (U)   Reserved for PDP-15.

.SCOM + 21   MAGtape status register.

.SCOM + 22   Reserved for MAGtape handler.

.SCOM + 23 (F)   Twos 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).

---

[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-unit 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 + 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 2 registers.

(b) After the Foreground job has been loaded,
this register contains a pointer to the regis-
ter 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.

.SCOM + 31   (a) The Software Memory Protect Bound 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 Back-
ground job has been loaded.

.SCOM + 32   (a) Pointer to the Hardware Memory Protect
Bound (or where it should be set). Contents
(.SCOM + 32) $\geq$ contents (.SCOM + 31).

.SCOM + 33   Background Program Counter, including L,XM,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; and so on.

.SCOM + 37 (F)   Pointer to CALER. Used to detect attempt
                        to re-enter CAL handler and to trap CAL*
                        instructions.

.SCOM + 40   CAL Flag. Non-∅ if control is in the CAL
                        handler (indication necessary for interrupt
                        servicing).

.SCOM + 41   "Who's running in the Background" Flag.
                        Bit ∅ = 1 if a Loader is running.
                        Bits 1-17:
                        17777 = Non-resident Monitor
                            ∅ = user program or DDT
                            1 = EDIT
                            2 = MACRO
                            3 = PIP
                            4 = F4
                            5 = SGEN
                            6 = DUMP
                            7 = UPDATE
                           10 = CONV
                           11 = MACROA
                           12 = F4A
                           13 = EXECUTE
                           14 = CHAIN
                           15 = PATCH

.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-∅
                        with the initial address of the device handler
                        doing the I/O. If the device is teletype,
                        the unit number + 400000 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 + 5∅   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.

---

[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.

.SCOM + 57   If non-$\emptyset$, 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 con-
                    trol.

.SCOM + 6$\emptyset$ Pointer to the entry point + 1 of the first
                    subroutine in the chain of Foreground Main-
                    stream real-time routines to be run when
                    Foreground Mainstream gets control.

.SCOM + 61   Same as .SCOM + 57 for Background.

.SCOM + 62   Same as .SCOM + 6$\emptyset$ for Background.

.SCOM + 63   Argument for API instruction ISA when inter-
                    rupts at API software levels are to be re-
                    quested.

.SCOM + 64 (F)   Pointer to CR.QR[1] in the Resident Monitor.

.SCOM + 65   Set non-$\emptyset$, while a Foreground user program is
                    running, to indicate that the resident buffer
                    may not be used by the Foreground. The resi-
                    dent 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[2] in the Resident Monitor.

.SCOM + 67 (F)   Pointer to Foreground control character table
                    in TTA.

.SCOM + 7$\emptyset$ (F)   Pointer to Background control character table
                    in TTA.

----

[1]CR.QR is a routine called by I/O handlers to initiate a device-
not-ready request.

[2]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.

.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 turn on
                                    extend memory after a DBR.

.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 inter-
                                    rupt service routine, used during error proces-
                                    sing.

.SCOM + 76 (F)   Twos complement count of the number of tele-
                                    types 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.

.SCOM + 1ØØ (F)   Pointer to ENTERQ[1] in the Resident Monitor.
                                    Will contain Ø, instead, if ENTERQ routine
                                    not assembled into the Monitor.

.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.

_____

[1]ENTERQ is a subroutine which makes entries in the API queue.

.SCOM + 1Ø2 (U)   Unused

.SCOM + 1Ø3 (U)   Unused

.SCOM + 1Ø4 (U)   Unused

.SCOM + 1Ø5 (F)   Twos complement size of the PI skip 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).

.SCOM + 1Ø7   Used to store the file directory entry
.SCOM + 11Ø   block of the XCT file to be EXECUTE'd
.SCOM + 111   in the Foreground.

.SCOM + 112   Used to store the file directory entry
.SCOM + 113   block of the XCT file to be EXECUTE'd
.SCOM + 114   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).

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 has been changed in the Background/Foreground System from the way they are treated in the Keyboard 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 is one that 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 2-4,-5,-6 and -7.

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 2-8.

CONTINUATION AFTER ERROR

.OTS errors are terminal errors. After OTS has printed

the error message, it exits to the Monitor. Therefore, after

an .OTS error the user does not have the option of restarting

his program.

Non-terminal .ERR errors do not terminate the operation of

user programs. Continuation, following the printing of the error

message, is automatic.

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 occurred while control was
in the Non-resident Monitor or in a Loader, the user does not
have the options indicated above. The Monitor will automatically
be reloaded.

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     (400200            /RAISE TO API
        ISA                        /LEVEL 0.
        IOF
        LAW     CODE               /SEE BELOW.
        JMS*    TEMP               /CALL ERRORQ.
AUXARG  XX                         /AUXILIARY ARGUMENT.
        DBK                        /RETURN HERE.
        ION
```

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 0-5 are ignored
        Bit 6 = 0 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
```

To avoid the possibility of future conflicts, user programs
and device handlers should utilize codes 600 - 777.

II-3

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 0 is the status of the link
XM- bit 1 is the status of extend memory
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
000	ILLEGAL CAL FUNCTION	L, XM, MP, CAL ADDR	YES
001	CAL[1] ILLEGAL	L, XM, MP, CAL ADDR	YES
002	.DAT SLOT ERROR (erroneous .DAT slot number or .DAT slot not tied to an I/O handler)	L, XM, MP, CAL ADDR	YES
003	ILLEGAL INTERRUPT	L, XM, MP, PC	YES

---

[1]The auxiliary information, depending on the source of the error, is sometimes UNIT #, CAL ADDR.

ERROR NO.	ERROR	AUXILIARY INFORMATION	TERMINAL
Ø04	MORE THAN ONE DEVICE NOT READY	.ASCII /XX/ ; XX = DEVICE NAME	YES
Ø05	ILLEGAL .SETUP	RETURN ADDRESS FROM .SETUP (ADDRESS IN CALLING DEVICE HANDLER)	YES
Ø06	ILLEGAL HANDLER FUNCTION	L, XM, MP, CAL ADDR[1]	YES
Ø07	ILLEGAL DATA MODE or SUBFUNCTION CODE	L, XM, MP, CAL ADDR[1]	YES
Ø10	FILE STILL ACTIVE	UNIT #, CAL ADDR	YES
Ø11	SEEK/ENTER NOT EXECUTED	UNIT #, CAL ADDR	YES
Ø12	UNRECOVERABLE DECTAPE ERROR	STATUS REGISTER B (Bits 0-1) 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
Ø20	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 COUNT (Either the word count was positive or the starting address plus the absolute value of the word count exceeded existing memory)	L, XM, MP, 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
Ø27	ILLEGAL DISK UNIT	UNIT #, CAL ADDR	YES
Ø31	NON-EXISTENT MEMORY REFERENCE	L, XM, MP, PC	YES
Ø32	MEMORY PROTECT VIOLA-TION	L, XM, MP, PC[1]	YES
Ø33	MEMORY PARITY ERROR	L, XM, MP, PC	YES
Ø36	BACKGROUND MEMORY PRO-TECT VIOLATION ATTEMPT VIA CAL ARGUMENT	L, XM, MP, CAL ADDR	YES
Ø5Ø	.TIMER REQUEST CANNOT FIT IN CLOCK QUEUE OR BACKGROUND REQUEST REMOVED TO MAKE ROOM FOR FGRD REQUEST	ADDRESS OF REAL TIME SUBROUTINE THAT WAS TO GET CONTROL ON COM-PLETION OF INTERVAL	NO
Ø52	MAINSTREAM REAL TIME REQUEST IGNORED BECAUSE ROUTINE IS ALREADY ENTERED	PRIORITY LEVEL/SUB-ROUTINE ENTRY POINT	NO
Ø53	APIQ OVERFLOW	ENTRY THAT WOULD NOT FIT (PRIORITY LEVEL/ SUBROUTINE ENTRY POINT)	NO
Ø55	NO BUFFERS AVAILABLE	RETURN ADDRESS FROM GETBUF (ADDRESS IN CALLING DEVICE HAND-LER)	YES
Ø56	ILLEGAL .ERROR CAL[2]	L, XM, MP, CAL ADDR	YES
Ø57	ILLEGAL .EXIT CAL	L, XM, MP, CAL ADDR	YES
Ø6Ø	.INIT NOT EXECUTED	CAL ADDR	YES
Ø61	TOO MANY NON-TERMINAL ERRORS	NUMBER OF ERRORS DISCARDED	NO
2ØØ	ILLEGAL TELETYPE UNIT	L, XM ,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 loca-tion of the JMP.

[2]A special error call to the Monitor (CAL code 16) is available for use only by the Loaders.

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 (Bad input data)
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 on a PDP-9 or PDP-15 and Background or Foreground.

OBJECT TIME SYSTEM ERRORS (.OTS)

All .OTS errors 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
12	ILLEGAL FORMAT STATEMENT
13	ILLEGAL REAL LOGARITHMIC ARGUMENT
14	ILLEGAL DOUBLE LOGARITHMIC ARGUMENT

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 LT19A or LT09A multi-station teletype controls, and

c. from 1 to 16_{10} Model 33 or Model 35KSR teletypes interfaced to the LT19A or LT09A controls.

The console teletype has its own set of IOTs, operates as half-duplex and is connected to the PIC (Program Interrupt Control). It cannot be connected to the API.

LT09A multi-station teletype controls can handle from 1 to 5 teletype lines and is connected only to the PIC. The LT19A is identical to the LT09A except that if a machine has API it will operate at API level 3, using channel registers 74 and 75.

Teletypes connected to LT09 or LT19 controls are operated in full-duplex mode, which requires the software to echo characters input from the Keyboard back to the teleprinter.

LT09/LT19 IOTs

Whether LT09 or LT19 is used, the IOT's associated with a particular teletype unit are the same. 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 LT09/LT19 controls, is defined to
be logical unit ∅.

TABLE 1:   1 to 5 units;   1 LT09A or LT19A

	UNIT #	PRINTER CODE	KEYBOARD CODE	LOGICAL UNIT #
LT09A/	1	XX400X	XX410X	1
LT19A	2	XX402X	XX412X	2
#1	3	XX404X	XX414X	3
	4	XX406X	XX416X	4
	5	XX420X	XX430X	5

TABLE 2:   6 to 10 units;   2 LT09A's or LT19A's

	UNIT #	PRINTER CODE	KEYBOARD CODE	LOGICAL UNIT #
LT09A/	1	XX400X	XX410X	1
LT19A	2	XX402X	XX412X	2
#1	3	XX404X	XX414X	3
	4	XX406X	XX416X	4
	5	XX440X	XX450X	11
LT09A/	1	XX420X	XX430X	5
LT19A	2	XX422X	XX432X	6
#2	3	XX424X	XX434X	7
	4	XX426X	XX436X	10
	5	XX442X	XX452X	12

TABLE 3:   11 to 15 Units;   3 LT09A's or LT19A's

	UNIT #	PRINTER CODE	KEYBOARD CODE	LOGICAL UNIT #
LT09A/	1	XX400X	XX410X	1
LT19A	2	XX402X	XX412X	2
#1	3	XX404X	XX414X	3
	4	XX406X	XX416X	4
	5	XX460X	XX470X	15

TABLE 3:   11 to 15 units; 3 LT09A's or LT19A's
(cont'd)

	UNIT #	PRINTER CODE	KEYBOARD CODE	LOGICAL UNIT #
LT09A/	1	XX420X	XX430X	5
LT19A	2	XX422X	XX432X	6
#2	3	XX424X	XX434X	7
	4	XX426X	XX436X	10
	5	XX462X	XX472X	16
LT09A/	1	XX440X	XX450X	11
LT19A	2	XX442X	XX452X	12
#3	3	XX444X	XX454X	13
	4	XX446X	XX456X	14
	5	XX464X	XX474X	17

TABLE 4:   16 units;   4 LT09A's or LT19A's

	UNIT #	PRINTER CODE	KEYBOARD CODE	LOGICAL UNIT #
LT09A/	1	Unused	Unused	—
LT19A	2	Unused	Unused	—
#4	3	Unused	Unused	—
	4	Unused	Unused	—
	5	XX466X	XX476X	20

(The setup for the first three controls would be as in Table 3).

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

        Model 35:   Has all three of the above.

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.