UC15

unichannel 15 system software manual

digital

UC15

unichannel-15 system software manual

DEC-15-XUCMA-A-D

Order additional copies as directed on the Software Information page at the back of this document.

digital equipment corporation · maynard. massachusetts

First Printing, August 1974

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this manual.

The software described in this document is furnished to the purchaser under a license for use on a single computer system and can be copied (with inclusion of DIGITAL's copyright notice) only for use in such system, except as may otherwise be provided in writing by DIGITAL.

Digital Equipment Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by DIGITAL.

The HOW TO OBTAIN SOFTWARE INFORMATION page, located at the back of this document, explains the various services available to DIGITAL software users.

The postage prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

PREFACE

This manual describes the UNICHANNEL-15 (UC15) Software System and its primary component PIREX, the peripheral processor executive.

No attempt is made in this document to describe the various UC15 hardware instructions; those are explained in the UNICHANNEL-15 System Maintenance Manual (DEC-15-HUCMA-B-D). However, examples of instruction sequences will be used when necessary to clarify programming conventions or illustrate important aspects of the UNICHANNEL Software System.

It is recommended that the reader have a thorough understanding of the UC15 hardware components before attempting to proceed with this manual. The user who plans to use the UC15 Software System in conjunction with some operating system on the PDP-15, and not modify it, should gain a thorough understanding of Chapter 1 of this manual. Users who wish to modify the UNICHANNEL-15 Software System should read the UNICHANNEL-15 System Maintenance Manual (DEC-15-HUCMA-B-D). In addition, a knowledge of PDP-11 and its assembly language is necessary before attempting UC15 system modification.

A Glossary is included following the appendices, and should be used to clarify terms not familiar to the reader. Program flow charts are also included in this manual to aid the user in understanding the logic flow.

The following documents also pertain to the UC15 System:

MAC11 Assembler Programmer's Reference Manual DEC-15-LMCMA-A-D

DOS User's Manual DEC-15-ODUMA-B-D

DOS System Manual DEC-15-ODFFA-B-D

UNICHANNEL-15 System Maintenance Manual DEC-15-HUCMA-B-D

Instruction List for the PDP-15

PDP-15/76 Systems Reference Manual DEC-15-XSRMA-A-D

DOS-15 V3BØØØ Update Document DEC-15-OD3BA-A-D

CONTENTS

Page

FIGURES

TABLES

CHAPTER 1

INTRODUCTION

1.1 UNICHANNEL-15 SOFTWARE COMPONENTS

The UNICHANNEL-15 Software System consists of the following four components:

1. PIREX
2. SPOL11
3. MAC11
4. ABSL11

1.1.1 PIREX

PIREX (peripheral executive), a component of the UNICHANNEL-15 (UC15) Software System, is described in Chapters 3 and 4 of this manual. **PIREX is a multiprogramming peripheral processor executive executed by the PDP-11. It is designed to accept any number of requests from** programs on the PDP-15 or PDP-11 and process them on a priority basis while processing other tasks concurrently (e.g., spooling other I/O requests). PIREX services all input/output requests from the PDP-15 in parallel on a controlled priority basis. Requests to busy routines (tasks) are automatically entered (queued) onto a waiting list and processed whenever the task in reference is free. In a background environment, PIREX is also capable of supporting up to four priority-driven software tasks initiated by the PDP-15 or the PDP-11.

1.1.2 SPOL11

Spooling is a method by which data to and from slow peripherals is buffered on a high performance RK05 disk. Spooling allows the PDP-15 to access and output data at high speed, freeing more of its time to do computation. Programs that do a great deal of I/O, especially printing and plotting, are not required to be core resident to complete the entire job. This frees the computer to quickly advance to more jobs, dramatically increasing the throughput of the entire system.

The SPOL11 task permits simultaneous spooling of line printer and plotter output, and card reader input. The capacity of the spooler is user-defined with a possible maximum of over 1,000,000 characters allowed.

1.1.3 MAC11

MAC11 is a special version of the standard MACRO-11 assembler available on the traditional PDP-11 computer system. This program is executed as a task under the PIREX Executive. It is used to conditionally-assemble various components of the UNICHANNEL Software System. Since this assembler is a subset of MACRO-11, programs assembled under MACRO-11, will not necessarily assemble under MAC11. In addition, programs written and assembled under MAC11 will not necessarily operate correctly on other PDP-11 systems. MAC11 produces assembly listings and absolute binary paper tapes as outputs. Detailed information concerning MAC11 can be found in the MAC11 Assembler Programmers Reference Manual.

1.1.4 ABSL11

ABSL11 is a PDP-15 Hardware Read In Mode paper tape program used to bootstrap-load the UNICHANNEL peripheral processor with absolute binary paper tapes. While primarily designed to load the PIREX executive into the PDP-11 memory, ABSL11 may be used to load any absolute program into the PDP-11 and optionally start it. Additional information on ABSL11 may be found in Chapter 2 of this manual.

1.1.5 System Software Modification

The complete UC15 Software System may be modified or expanded by the user when running under the DOS-15, BOSS-15, or RSX-PLUS III programming systems. A common editor, called EDIT, allows source changes to the PDP-15 or PDP-11 software. MACRO-15, the PDP-15 MACRO Assembler, and MAC11, a PDP-11 MACRO Assembler allow new object code to be generated. Both the MACRO-15 and MAC11 assemblers are powerful MACRO assemblers that facilitate easy code generation and source readability.

1.2 UNICHANNEL-15 HARDWARE SYSTEM

The UC15 hardware (see Figure 1-1) consists of a PDP-11 minicomputer used as an intelligent peripheral controller for the larger PDP-15 main computer. The PDP-15 functions as the master processor by initiating and defining tasks while the PDP-11 peripheral processor functions as a slave in carrying out these tasks. In order to effectively operate, with a minimum of interference with the master processor, the peripheral processor uses its own local memory of between 4,096 and 12,288 16-bit words. Since peripheral control requires only a fraction of the peripheral processor resources, the remainder of the processor's resources can be used for parallel processing of background tasks.

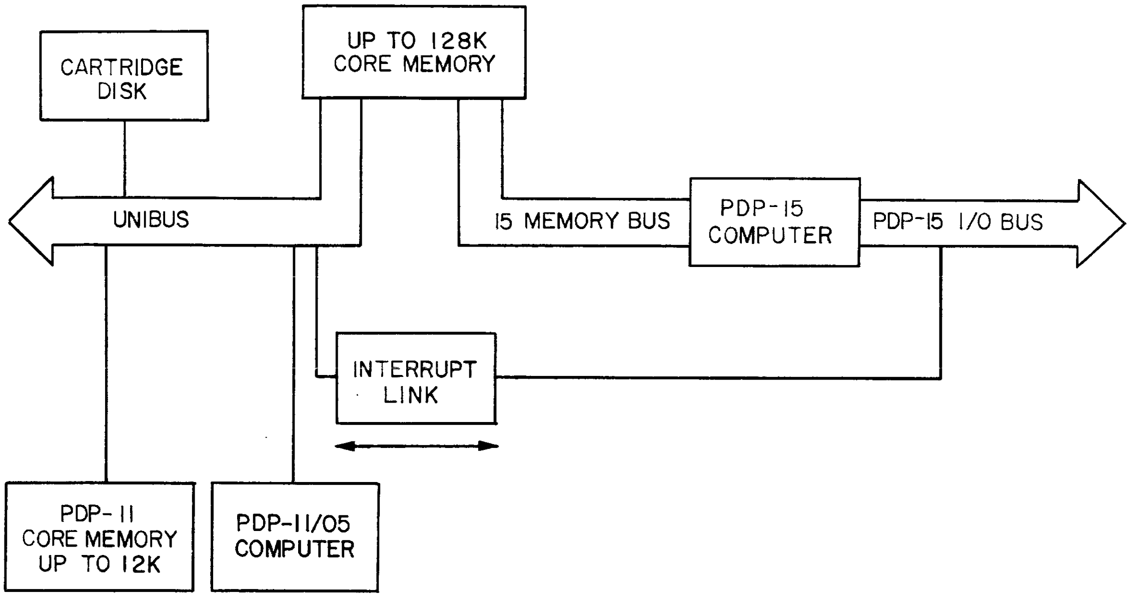


Figure 1-1
UNICHANNEL-15 Hardware System

1.2.1 Common Memory

Common memory is that memory directly accessable to both the master processor - the PDP-15, and the peripheral processor - the PDP-11. Thus common memory occupies the upper portion of the PDP-11 address space and at the same time the lower portion of the PDP-15 address space. The UC15 System allows any Non-Processor Request device on the UNIBUS to access PDP-15 memory so that data can be transferred between I/O devices and common memory.

The use of common memory allows ease of data transfer between PDP-15 memory and secondary storage (disk, magnetic tape, etc.). The PDP-11 peripheral processor can access a maximum of 28K of memory. Table 1-1 shows the amount of Common memory accessible to a PDP-11 processor with a given amount of Local memory.

Table 1-1
Common Memory Sizes

PDP-11 LOCAL MEMORY SIZE	COMMON MEMORY SIZE
4K[1]	24K
8K	20K
12K	16K

[1] NOT supported under DOS-15 V3BØØØ.

The UNIBUS can address the combined PDP-15/PDP-11 memory, which can extend to a maximum of 124K. For instance, the RK05 and its disk controller can transfer information to or from a location outside of the common memory region. Figure 1-2 outlines a typical memory map of the PDP-15 and PDP-11, illustrating the common shared memory address space and the PDP-11 local memory.

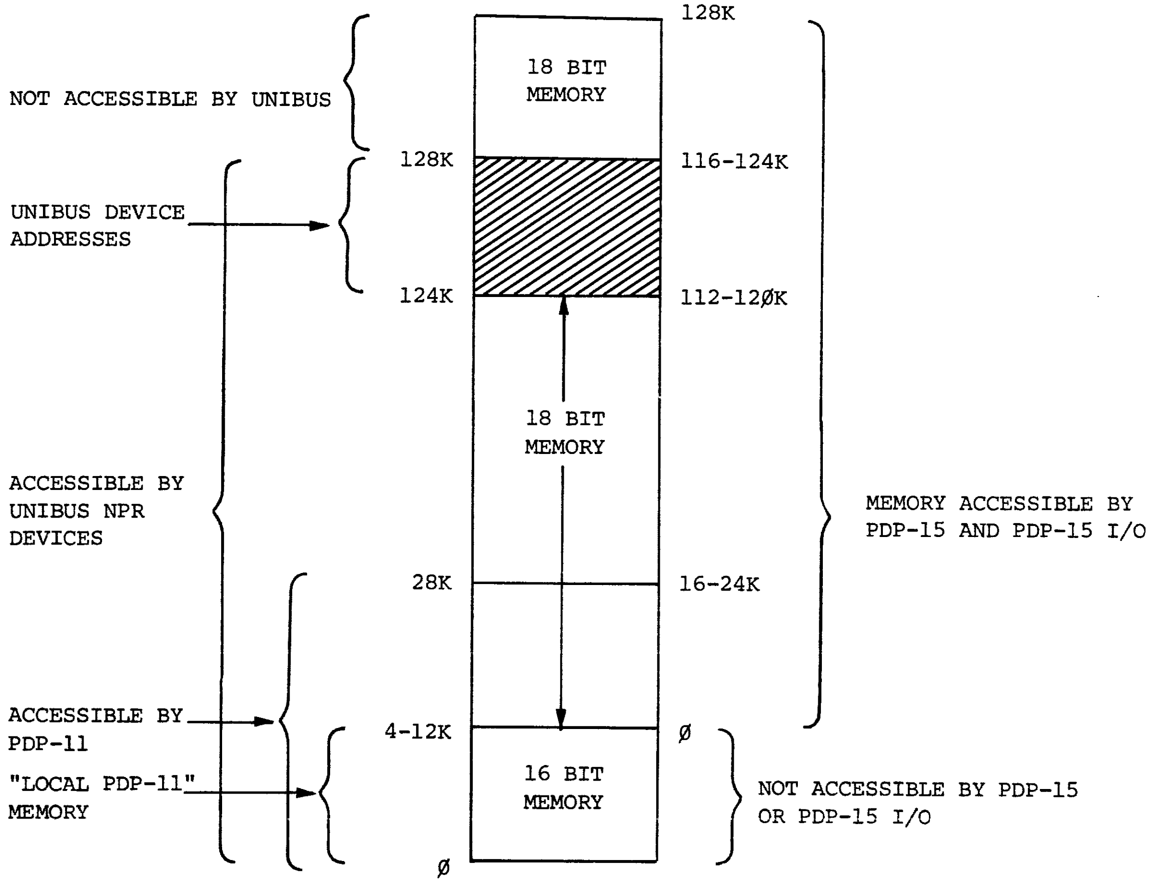


Figure 1-2
Memory Map of a UNICHANNEL System

1.2.2 Interrupt Link

The PDP-15 and the peripheral processor communicate with each other through device interfaces. When the PDP-15 initiates a new task, it interrupts the peripheral processor with a message. The message is designated as a Task Control Block Pointer (TCBP) and points to a table (Task Control Block) in common memory where the task is defined. The peripheral processor performs the task and can signify its completion by sending an optional interrupt back to the PDP-15.

1.2.3 Peripheral Processor Hardware

The UC15 System in its standard configuration consists of the following equipment (Figure 1-3):

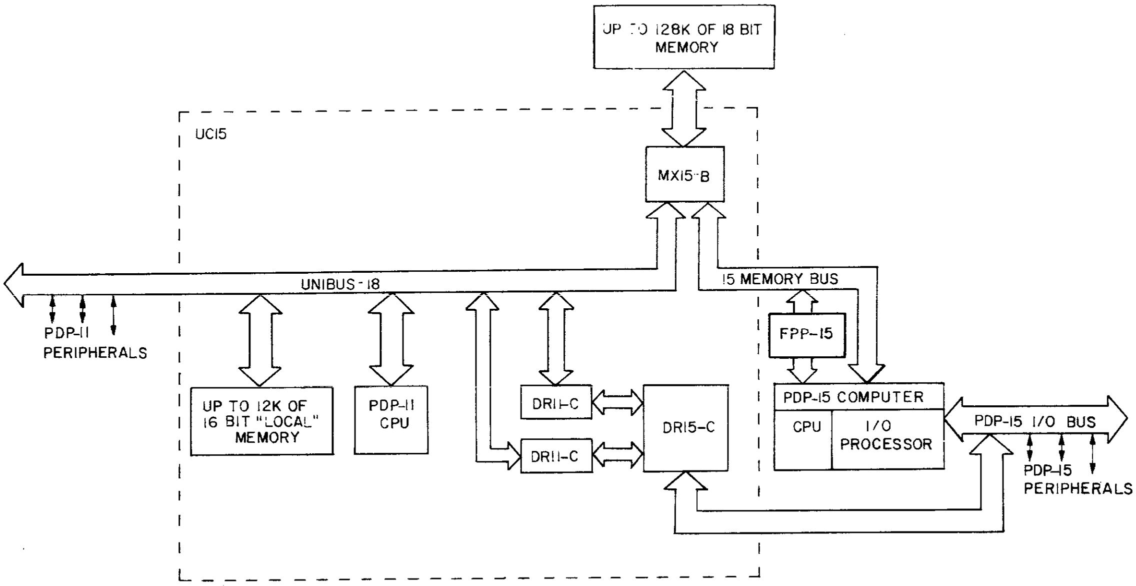


Figure 1-3
UNICHANNEL System

- PDP-11 Peripheral Processor
- DR15-C Device Interface
- Two DR11-C Device Interfaces
- MX15-B Memory Bus Multiplexer
- 8096 Words of 16-Bit Local Memory

The PDP-11, which functions as the peripheral processor, can itself only process 16-bit words but controls peripherals that can process 18-bit words to provide compatibility with the PDP-15. The DR15-C and the two DR11-C Device Interfaces provide the communication facility between the PDP-15 and the PDP-11. The PDP-15 can interrupt the PDP-11 and send a data word (TCBP) to the PDP-11; this interrupts the PDP-11 at priority level 7 (the highest priority level) and causes a trap thru location 310_8. The PDP-11, serving as a peripheral processor, can interrupt the PDP-15 to indicate an error condition or job completion at any one of 128 API vector locations at any one of four API priorities.[1]

[1] This applies to systems with the API option - systems without API can use four skip instructions, corresponding to the four hardware priority levels, to determine the nature of the interrupt.

The MX15-B Memory Bus Multiplexer functions as a memory bus switch to allow either the PDP-15 or the PDP-11 to communicate with the common memory. The MX15-B also provides the PDP-11 with the capability of performing byte instructions which reference PDP-15 memory.

CHAPTER 2

LOADING AND EXECUTION

2.1 INTRODUCTION

This chapter explains how to get the DEC-supplied UNICHANNEL-15 Software System up and running, how to tailor the system to a specific configuration, and how to maintain the system at a high level of performance. In addition, a list of the UC15 software components used in the various PDP-15 monitor systems is included.

2.2 LOADING THE SYSTEM[1,2]

The UC15 system is activated by using ABSL11 to load the PIREX executive into the PDP-11 UNICHANNEL local memory. DOS-15 is then bootstrapped from the RK05 cartridge and the system is ready to:

1. Continue running under DOS-15
2. Begin execution of BOSS-15
3. Begin execution of RSX-PLUS III

2.2.1 ABSL11

ABSL11 is a PDP-15 absolute binary paper tape program which is read into the PDP-15 at location 17700_8 via the Hardware Read In mode (HRM) on the PDP-15. It is used to load PDP-11 absolute binary paper tape on to the PDP-11. This self starting program is written in MACRO-15 and octal. (The PDP-11 code is written in octal and assembled with MACRO-15.) When ABSL11 is first loaded, PDP-15 halts and waits for the user to start the PDP-11. The starting address for a PDP-11 depends upon the size of its local memory. Table 2-1 lists the available options.

[1] Refer to the DOS SGEN Manual for the details of how to use DOSSAV to initially place a DOS System on the RK05 and prepare it for use.

[2] If the RK Disk is not going to be the system disk (e.g., the RP or RF disks would be the system disk), see Appendix D for details of the proper installation procedure.

Table 2-1
ABSL11 Starting Addresses

Local Memory Size	ABSL11 Starting Address[2]
4000 words	60000_8
8000 words	100000_8
12000 words	120000_8

When the PDP-11 is running, the user can place a PDP-11 absolute tape (in this case PIREX) in the PDP-15 High Speed Reader and depress the CONTINUE switch on the PDP-15. This reads the tape into the lower 8K[1] of the PDP-15 in identical relative positions as if it were loaded into the PDP-11's own local memory. When the tape is completely loaded, the PDP-15 signals the PDP-11 to relocate the program into the PDP-11's local memory and optionally start it, if a transfer address was specified on the tape (as on the PIREX tape). If not, the PDP-11 halts and waits for a manual start by the user. The PDP-15 halts once the tape has been loaded. The relocation of PDP-11 absolute programs into memory is done by copying the entire lower 8K[1] of the PDP-15 into the lower 8K addressing space of the PDP-11 (or the entire 4K, or, the entire 12K depending on local memory size) on a word by word transfer. This relocation, therefore, results in the entire PDP-11 memory being altered with all previous information overlaid.

If the first paper tape does not have a start address, additional tapes can be loaded by depressing the PDP-11 CONTINUE switch once and depressing the PDP-15 CONTINUE switch twice. Warning - the maximum PDP-11 program address that can be loaded by ABSL11 is the amount of PDP-11 local memory, which is a maximum of 12K for UNICHANNEL systems.

Checksum errors are detected by the PDP-15 and result in a halt with all 1's in the AC register. The checksum error may be ignored by depressing the CONTINUE switch on the PDP-15.

2.2.2 Loading ABSL11, PIREX, and DOS-15

The following is a step-by-step description of how ABSL11, PIREX, and DOS-15 are loaded.

1. Place the ABSL11 paper tape into the PDP-15 Paper Tape Reader. The Paper Tape Reader ON/OFF switch must be in the ON position.
2. Verify that the RK05 Disk Cartridge is loaded into drive and:
    a. The LOAD/RUN switch is in the RUN position.
    b. The write ENABLE/PROTECT switch is in the ENABLE position.

[1] This value depends upon the actual local memory size - 4K, 8K or 12K.

[2] This is the PDP-11 console address.

3. Press the HALT switch on the PDP-11 UNICHANNEL console.

4. On the PDP-15 console, set the address register switches to 177ØØ(octal), then press STOP and RESET simultaneously.

5. On the PDP-15 console, press READ IN. The ABSL11 paper tape should read in.

6. When the Paper Tape Reader stops, observe the PDP-15 accumulator (AC) using the proper setting of the rotary register selector and register select switch on the PDP-15 console.

   a. If the AC is 0, proceed to step 7.

   b. If the AC is not 0, retry starting at step 1. (If this fails consistently, you have either a bad ABSL11 paper tape or a hardware problem.)

7. On the PDP-11 UNICHANNEL console, load the starting address for the PDP-11 portion of ABSL11 into the switch registers:

   a. For a 4K local memory UNICHANNEL use 60000_8

   b. For an 8K local memory UNICHANNEL use 100000_8

   c. For a 12K local memory UNICHANNEL use 120000_8

   Then press the PDP-11 LOAD-ADR switch

8. On the PDP-11 UNICHANNEL console, raise the HALT/ENABLE switch to the ENABLE position and then press the START switch. The PDP-11 RUN light should now be lit.

9. Remove the ABSL11 paper tape from the reader and place the PIREX paper tape into it.

10. On the PDP-15 console, press the CONTINUE switch. PIREX paper tape should read in.

11. Remove the PIREX paper tape and verify that the bit 0 and RUN lights on the PDP-11 UNICHANNEL console are lit. This is an indication that PIREX is running.

12. Load RK Bootstrap tape (hardware read in mode tape) into the Paper Tape Reader.

13. Set Address Switches on the PDP-15 Console to

    a. 77637_8 for a 32K or more PDP-15

    b. 57637_8 for a 24K or 28K PDP-15

    c. 37637_8 for a 16K or 20K PDP-15

14. On the PDP-15 Console, press simultaneously STOP and RESET.

15. On the PDP-15 Console, press the READ IN switch. The RK Bootstrap tape should read in.

16. DOS-15 should announce itself. If not, check that the console terminal is powered up, is ONLINE and not out of paper. Also check that the correct disk cartridge was loaded into drive 0.

2.3 UNICHANNEL SOFTWARE RECONFIGURATION

The initial UC15 system supplied to the user may require modification to be effectively used. This system is configured as follows:

1. An 8K local memory MAC11 assembler
2. A PIREX Executive with RK and LP drivers
3. A SPOL11 spooler for LP only

2.3.1 MAC11

If your system does not have 8K local memory on the UNICHANNEL, you must first tailor the MAC11 assembler into a version compatible with your local memory size. The procedure to perform this under DOS-15 follows:

1. Assemble MACIMG XXX present under the PER UIC using MACRO-15 and one of the following assembly parameters.

a.	LM4K = 0	For a 4K local memory UNICHANNEL
b.	No parameter	For an 8K local memory UNICHANNEL
c.	LM12K = 0	For a 12K local memory UNICHANNEL

   This will produce the binary file MACIMG BIN

2. Load one of the following MAC11 paper tapes into the Paper Tape Reader:

a.	DEC-15-ODUFA-A-PB	For a 4K local memory UNICHANNEL
b.	DEC-15-ODUEA-A-PB	For an 8K local memory UNICHANNEL
c.	DEC-15-ODUTA-A-PB	For a 12K local memory UNICHANNEL

3. Issue the DOS-15 API OFF command (if you have API).
4. Issue the DOS-15 $GLOAD↲ command, then type >←MACIMG (ALT)
5. The paper tape should read in. When it stops a "DONE" message should be printed on the console terminal; at this point, the PDP-11 part of MAC11 is installed on disk.
6. Assemble the MACINT XXX under the PER UIC using the following assembly parameters:

a.	LM4K = 0	For a 4K local memory UNICHANNEL
b.	No parameter	For an 8K local memory UNICHANNEL
c.	LM12K = 0	For a 12K local memory UNICHANNEL

7. LOGIN under the MICLOG and assign DAT.-10 to the PER UIC.

   ```
   $A RK <PER>  -10↲
   ```

8. Using PATCH do the following:

```
    $PATCH ↵

    >MAC11 ↵

    > READ MACINT ↵

    >EXIT ↵
```

This installs the PDP-15 portion of MAC11 onto the disk.

9. A new MAC11 will now be available for use.

2.3.2 PIREX

The PIREX Executive should be configured to contain device drivers for only those peripherals actually present in the user's configuration. The DOS Assembly Parameters Document DEC-15-ODAPA-A-D describes the various assembly options available to the customer. The following procedure should be followed to produce a tailored version of PIREX.

1. Under the PER UIC, use EDIT to add or remove the various assembly parameters for PIREX. (Parameters for programs assembled by MAC11 must be included in the main source file.)

2. Assign DAT-12 to the listing device. (The absolute binary output device will always be paper tape.)

3. Run MAC11 and assemble PIREX XXX[1]:

```
    $MAC11 ↵

    >BL←PIREX XXX ALT
```

Where:

"B" causes the absolute binary paper tape to be punched

"L" causes the optional listing to be printed on DAT-12.

4. Load the new paper tape using the instructions in Section 2.2.2 of this chapter.

2.3.3 SPOL11[2]

The UNICHANNEL Spooler should be configured to provide spooling only for those devices present on the user's configuration. The spooler supplied with the system is configured to provide Line Printer spooling. If the user does not possess a UNICHANNEL Line Printer (LP11/LS11/LV11), or the user wishes to spool other UNICHANNEL devices, this spooler should not be used. The procedure for producing a spooler tailored to the user's configuration follows.

[1] XXX represents the latest version number, i.e., PIREX 118.

[2] This procedure applies only to DOS-15 V3AØØØ. See the DOS-15 V3BØØØ Update Document DEC-15-OD3BA-A-D for details of how to install the spooler on a DOS-15 V3BØØØ system.

1. Under the PER UIC use EDIT to add or delete the following assembly parameters in SPOL11 XXX:

   a. $LP = 40000 for Line Printer Spooling

   b. $CD = 20000 for Card Reader Spooling

   c. $PL = 10000 for Plotter Spooling

2. Assign DAT-12 to the listing device.

3. Assemble SPOL11 under MAC11 with <u>both</u> the B and L switches.

```
$MAC11↵

>BL ←SPOL11   XXX ALT
```

4. From the listing locate the definition of SPOLSZ and copy down the value.

5. Run PIP and type:

```
$ PIP↵

>L TT← RK (L)↵
```

   This will produce a symbolic listing. Using this listing, locate the column headed FB (first block) and find the first block of SPOOL.

6. Under the PER UIC assemble the SPOL15 XXX program with MACRO-15 using as assembly parameters:

   a. SPOLSZ = the value determined in 4 above.

   b. FB = the value determined in 5 above.

7. Under the PER UIC assemble the SPLIMG XXX program with MACRO-15 using the assembly parameter:

   a. SPOLSZ = the value determined in 4 above.

8. For API systems issue the DOS-15 command API OFF.

9. Place the SPOL11 absolute binary paper tape in the reader.

10. Issue the DOS-15 command GLOAD and type:

```
$ GLOAD↵

>← SPLIMG  ALT
```

11. The SPOL11 paper tape will be read in and a "DONE" message will be typed on the console terminal when completed.

12. Next MICLOG and assign DAT-10 to the PER UIC

```
$ A RK <PER> -10↵
```

13. Run PATCH and type:

```
$ PATCH↵

> SPOOL↵

> READ SPOL15↵

> EXIT↵
```

This will append the PDP-15 portion of the spooler to the previously loaded PDP-11 portion.

2.3.4 PDP-15 UNICHANNEL Handlers

PDP-15 UC15 Handlers that are not to be spooled must be assembled with the NOSPL = 0 assembly parameter. Those handlers that are to be spooled must be assembled without this parameter defined. The initial RK05 system supplied by DEC contains handler binaries under the< IOS > UFD that were assembled as follows:

1. LPA. was assembled to allow spooling
2. CDB. was assembled with NOSPL = 0 to not allow spooling.
3. XYA. was assembled with NOSPL = 0 to not allow spooling.

Any alteration of the mix of spooled devices requires reassembly of the handler sources. (Location under the <PER> UFD. See the DOS Assembly Parameter Manual, for additional assembly parameter options.) The resulting binaries must be renamed (see Section 2.7.2) and transferred to the <IOS> UFD.

2.3.5 SPOOLER Size Constraints

The following should be considered an absolute constraint on the number of devices spoolable on the UC15 system.

1. A 4K local memory system can have no spooled devices
2. An 8K local memory system can have up to 2 spooled devices
3. A 12K local memory system can have up to 4 spooled devices (DEC only provides spooler modules for 3 devices. Additional spooled device modules must be added by the user. Refer to chapters 5 and 6 for information on how to do this).

2.4 PERIPHERAL OPERATION

2.4.1 Disk Cartridge

On the front of the disk cartridge unit there are two (optionally a third, ON/OFF) toggle switches, RUN/LOAD, and WRITE/PROT. To load the disk, press ON (if present) and LOAD. Pull the door open. Pick up the cartridge by the molded hand-grip, metal side down, horizontal, and slide gently into the path between the wire guides. Shut the door. Put the LOAD/RUN switch into the RUN position. In about 10 seconds, the two lights, RDY and ON CYL will come on, indicating that the cartridge is ready. To unload the disk, place the toggle switch on LOAD. Wait for about 30 seconds until the LOAD light is on. At this time, the drive will release the cartridge with a noticeable 'clunk', only then open the door and pull the cartridge out.

WARNING

Do not turn off the drive while unloading (if drive has an OFF-ON toggle).

2.4.2 Plotter

Unlike the XY311, the XY11 does not have an offline switch. In order to be able to indicate the XY11 plotter off-line condition, provision is made in the software through the PDP-11 console switches. By setting bit '2' of the console data/address switches in the up/on position ('1' state) the plotter can be put in the off-line mode. This is made possible by the plotter device driver task in PIREX, which monitors this bit before initiating each plotter I/O requests. Once the plotter problem condition (e.g., out of paper) has been corrected, plotting will continue automatically when bit '2' of the console switches is reset to zero (down position).

The user is provided with the capability of halting the output on the plotter at the end of current file in the spooled mode. This is done through bit '3' of the PDP-11 console switches. By setting bit '3' of the console data/address switches in the up/on position ('1' state) output on the plotter can be halted at the end of current file. The plotter driver task in PIREX provides this facility by monitoring this bit before initiating each plotter I/O requests. After performing the necessary operations on the plotter, output can be resumed by setting bit '3' of the console switch in the down/off position ('0' state).

2.4.3 Card Reader

For the purposes of spooling, a card with ALT MODE, ALT MODE in columns 1 and 2 is used as an end-of-deck card. The handler throws away such cards, continuing on to the next card, so that the PDP-15 program using the handler never sees this card. This card is used to force data from a partially filled internal spooler buffer onto the disk where it can be despooled to the PDP-15.

2.4.4 Line Printer

Output to the Line Printer can be halted at the end of current file in the spooled mode. This is done through bit '1' of the PDP-11 console switches. By setting bit '1' of the console data/address switches in the up/on position ('1' state), outputs on the line printer can be halted at the end of current file. The Line Printer driver task in PIREX provides this facility by monitoring this bit before indicating completion of .CLOSE I/O request processing. After performing the necessary operations on the line printer, output can be resumed by setting bit '1' of the console switch in the down/off position ('0' state).

2.5 ERROR HANDLING

Within the PIREX system, the device drivers on the PDP-11 side handle errors by placing error condition indicators in a table in PIREX. On the PDP-15 side, a "poller" (part of the resident monitor of the operating system) periodically searches the table to see if any error messages are to be printed. In almost all cases the recovery is automatic when the error condition is rectified. See Appendix C for a list of UC15 related error messages.

2.5.1 Disk Cartridge Errors

Disk cartridges must be positioned properly during loading operations. Improper positioning of the cartridge can result in a drive not ready condition.

This condition can be eliminated in most instances by unloading the cartridge, repositioning it properly and reloading the cartridge.

The above operations should be repeated a few times before reporting the problem to your field service representative. Do not force the cartridge into or from position during the loading or unloading operation.

2.5.2 Card Reader Errors

The system divides card reader errors into two groups: hardware and software. A hardware error is a hardware read error (pick check, card jam, etc.) or an illegal punch combination. A software error is a supply error (hopper empty, stacker full) or an off-line condition.

For all hardware errors, the card causing the error will be on the top of the output stack. With most hardware errors, the card reader will stop, and a requisite light (i.e., pick check) will light on the reader. Remove the card, repair or replace it, and put it on the front of the input stack. Press the RESET button. The driver receives an interrupt when the device becomes ready again and will restart automatically.

For software errors, the card in the output hopper has already been read. It is merely necessary to fix the supply error and press the RESET button. Note that the card reader can be stopped by pressing the OFF-LINE button. To restart, press the RESET button.

Illegal punch combination (IOPSUC CDU 72) and card column lost (IOPSUC CDU 74) are exceptions to all other errors because in these cases alone, the card reader will stop, remain on line, and no diagnostic light will be lit. The card causing the error will be in the top of the output hopper. (Mangled cards may cause an illegal punch combination error.) Press the OFF-LINE button, repair or replace the faulty card, put it on the front of the input stack, and press the RESET button to restart.

2.6 SYSTEM CRASHES

During program development under PIREX on the PDP-11, system crashes may occur. Such crashes may not be apparent because PIREX keeps both the RUN light and bit 0 lit as if no problem existed. PIREX will then either not respond at all or return illegal event variable values. Under these circumstances, reload PIREX and reboot the operating system on the PDP-15.

2.7 UNICHANNEL RELATED SOFTWARE COMPONENTS

2.7.1 UC15 Components

NOMENCLATURE	SOURCE FILE NAME	BINARY FILE NAME
PIREX Executive	PIREX XXX	PIREX paper tape
SPOOLER	SPOL11 XXX	SPOOL ***
PDP-11 Absolute Loader	ABSL11 XXX *	ABSL11 paper tape
MAC11 Assembler	Special DOS-11 Tape**	MAC11 ***

2.7.2 DOS-15 Components

NOMENCLATURE	SOURCE FILE NAME	BINARY FILE NAME
PDP-15 SPOOLER Component	SPOL15 XXX	SPOOL ***
SPOOLER Disk Area Allocation	SPLGEN XXX	SPLGEN BIN*****
SPOOLER Image Loader	SPLIMG XXX	SPLOAD BIN*****
PDP-15 MAC11 Component	MACINT XXX	MAC11 ***
MACRO Image Loader	MACIMG XXX	MACIMG BIN
DOS Resident Monitor	RESMON XXX	RESMON ****
DOS Non-Resident Monitor	DOSNRM XXX	DOS15 ****

* ABSL11 requires a special assembler, that is not available as a supported product. Assembly of ABSL11 with the standard DOS-15 MACRO Assembler produces a paper tape with a load address of 1772∅.

** The MAC11 source is a PDP-11 tape that must be assembled and linked under DOS-11.

*** SPOL11 and MAC11 are combinations of PDP-15 and PDP-11 code segments.

**** These routines are versions of standard DOS-15 source files - created using special assembly parameters - see the DOS Monitor User's Manual.

***** DOS-15 V3B∅∅∅ components.

NOMENCLATURE	SOURCE FILE NAME	BINARY FILE NAME
PDP-15 LP11/LS11/LV11 Line Printer Handler	LPU. XXX	LPA. BIN
PDP-15 XY11/XY311 Plotter Handler	XYU. XXX	XYA. BIN
PDP-15 CR11 Card Reader Handler	CD.DOS XXX	CDB. BIN ****

**** These routines are versions of standard DOS-15 source files - created using special assembly characters - see the DOS Monitor User's Manual.

2.7.3 RSX-PLUS III Components

NOMENCLATURE	SOURCE FILE NAME	TASK NAME
Fixed-Head Disk File Handler	RFRES	RK ....
Disk File Handler Overlay	RFOPEN	RK ....
Disk File Handler Overlay	RFCLOS	RK ....
Disk File Handler Overlay	RFREAD	RK ....
Disk File Handler Overlay	RFDLET	RK ....
Disk File Handler Overlay	RFCREA	RK ....
Line Printer Handler	LP.30	LP ....
Card Reader Handler	CD	CD ....
UNICHANNEL Poller	POLLER	POLLER
Spooler	SPOOL	... SPO
Executive	RSX.P1 and RSX.P2	

These items are usually on DECTAPE or magnetic tape.

CHAPTER 3

SYSTEM DESIGN AND THEORY OF OPERATION--PIREX

This chapter describes the design and theory of operation of the UNICHANNEL-15 Peripheral Processor Executive. Knowledge of this information is necessary to successfully modify the UNICHANNEL-15 Software System. Chapter 4 will discuss techniques for modification of the PIREX system.

3.1 PIREX--PERIPHERAL EXECUTIVE

PIREX is a multiprogramming peripheral processor executive designed to provide device driver support to operating systems on the PDP-15 main-processor. PIREX is designed to be as independent of the particular PDP-15 operating system as possible, executing in conjunction with DOS-15, BOSS-15, or RSX-PLUS III. The PIREX Software System is designed to maximize flexibility and expandability and to minimize system overhead and complexity. To accomplish this, special software and hardware features are designed into the system.

3.1.1 PIREX-An Overview

PIREX is loaded from the PDP-15 high-speed reader into the PDP-11 local memory and automatically started. Once running, PIREX is capable of accepting multiple requests and directives from the PDP-15 or PDP-11 and processing them on a controlled-priority basis. Task requests are automatically queued (see Figure 3-1) and processed whenever the task in reference is free. When a particular device or routine completes the processing of a request, status information (e.g., parity or checksum errors, transfer OK, etc.) is passed back to the caller.

At the completion of a PDP-15 request, an optional hardware interrupt is initiated in the PDP-15 on any one of 128 possible API trap locations and at any one of 4 hardware API levels if requested. Since the software completely determines which interrupt vector and level to use when completing PDP-15 requests, the routines initiating the interrupts could actually be software routines used to simulate hardware conditions or just software tasks. If the request is issued from the PDP-11, the user may request an optional software interrupt after completion of the current request.

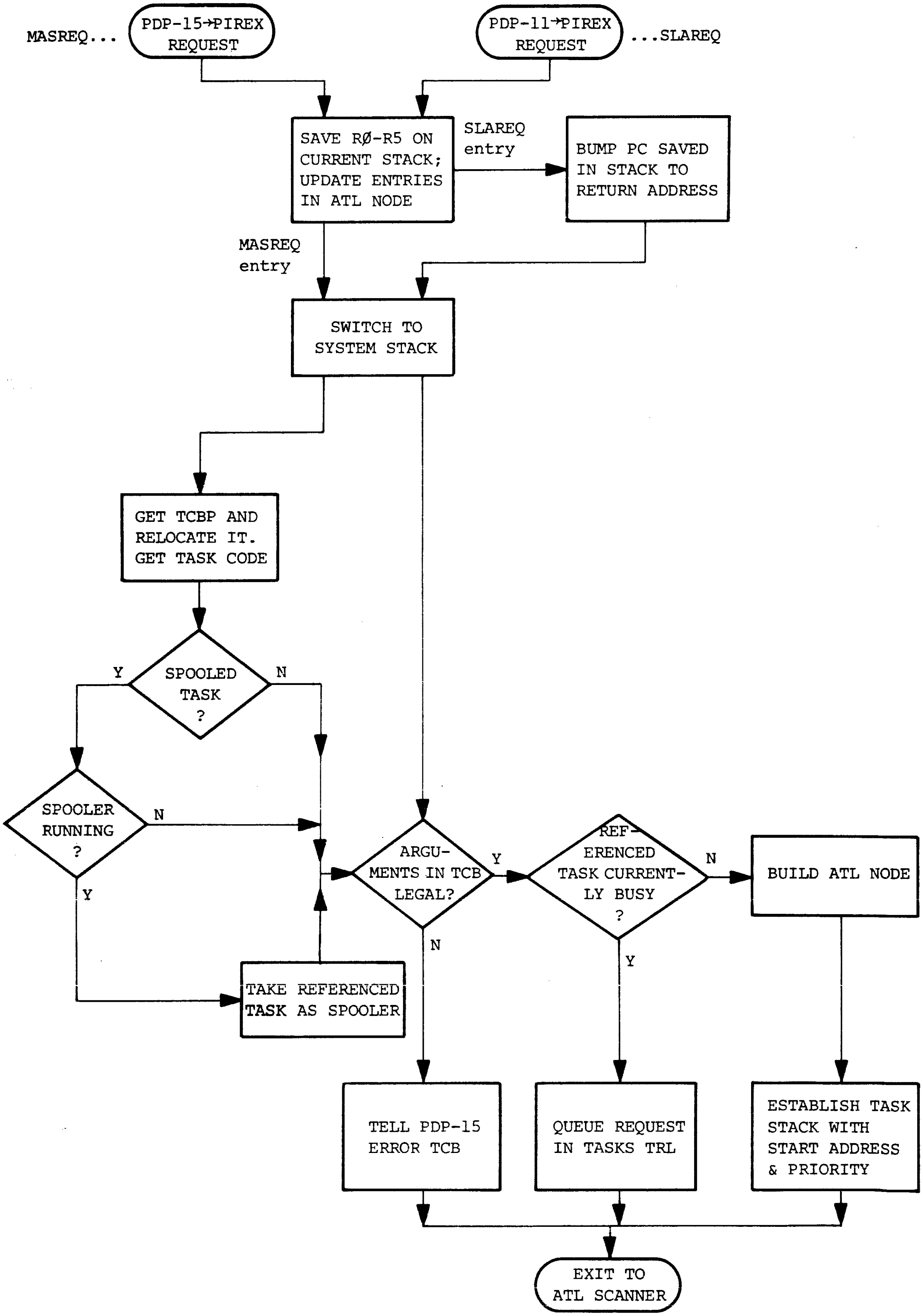

MASREQ...
PDP-15→PIREX REQUEST
PDP-11→PIREX REQUEST
...SLAREQ
SAVE RØ-R5 ON CURRENT STACK; UPDATE ENTRIES IN ATL NODE
SLAREQ entry
BUMP PC SAVED IN STACK TO RETURN ADDRESS
MASREQ entry
SWITCH TO SYSTEM STACK
GET TCBP AND RELOCATE IT. GET TASK CODE
Y
SPOOLED TASK ?
N
SPOOLER RUNNING ?
N
Y
TAKE REFERENCED TASK AS SPOOLER
ARGU-MENTS IN TCB LEGAL?
Y
N
REF-ERENCED TASK CURRENT-LY BUSY ?
N
Y
BUILD ATL NODE
TELL PDP-15 ERROR TCB
QUEUE REQUEST IN TASKS TRL
ESTABLISH TASK STACK WITH START ADDRESS & PRIORITY
EXIT TO ATL SCANNER

Figure 3-1
Basic Flow Chart of PDP-15/11 Request Processing

3.1.2 PIREX Components

The PIREX executive consists of modules that provide support for multiple I/O oriented tasks operating asynchronously with each other. In addition, support is provided for other background tasks such as MAC11. The services provided to tasks operating under PIREX include:

- Context switching - transferring control of the PDP-11 Central Processing Unit (CPU) from one task to another.
- Interprocessor communication - receiving requests for service from, and, sending results to the PDP-15 main processor.
- Intraprocessor communication - receiving requests for service from, and, sending results to tasks operating on the PDP-11 peripheral processor.
- Scheduling - determining which task is to execute next.
- Request Queuing - stacking requests for a busy task until it is able to process them.
- Timing - providing a timed wake-up service for requesting tasks.
- Error Reporting - providing a list of current device and task errors to the PDP-15 executive, on demand.
- Directive Processing - providing the PDP-15 monitor with specific services such as: notification of available memory space, connecting, disconnecting or stopping tasks and returning the status of certain tasks.

These services are provided to both device driver tasks and background tasks.

3.1.3 Device Drivers

Device Drivers are tasks that typically perform rudimentary device functions such as read, write, search, process, interrupt, etc. They can, however, be complete handlers, performing complex operations such as character generation and directory searching. PIREX provides each driver with requests for I/O actions and returns the results of the actions to the caller. Associated drivers are provided for the RK05 Disk Cartridge, the LP11/LS11/LV11 Line Printer, the CR11 Card Reader, and the XY11 Plotter.

3.1.4 Software Routines in Background Mode

The following are run as background tasks--executing only when I/O driver tasks are idle:

1. SPOL11 -- an input/output spooling processor
2. MAC11 -- A MACRO assembler for the PDP-11

3.1.5 Unsupported Tasks

All tasks supplied with the PIREX software system are fully supported by Digital Equipment Corp. except the DECtape Driver task (DT) and LV11 Plotter tasks. The DT task has not been completely tested, but is included in the system for illustrative purposes and for anyone who may desire to develop DECtape capability on the PDP-11. The LV11 task is designed to allow .TRAN operations to the LV11 when used as a plotter (instead of as a printer). This task was developed for the demonstration of vector scan plotting techniques. The task is unsupported because the vector scan routines are not currently available from DEC.

3.1.6 Power Fail Routine

A power fail section is present in PIREX. It is, however, not supported by DEC and currently only saves the general registers and does not attempt to handle I/O in progress. This routine could be expanded by the user into a complete power fail handler.

3.2 PIREX - SIMPLIFIED THEORY OF OPERATION

3.2.1 NUL Task

When the PIREX Software System is running, it is normally executing the NUL Task (a PDP-11 WAIT instruction). The NUL Task is executed whenever there are no other runnable tasks or while all other tasks are in the WAIT state waiting for previously initiated I/O. The NUL Task entry is a permanent element in the Active Task List. The Active Task List is a priority ordered list of tasks that is used to schedule the next task to be executed. The NUL task occupies the last position in the Active Task List (ATL).

3.2.2 Clock Task

One other permanent entry in the ATL is the Clock Task. The Clock Task is entered once every 16.6 milliseconds (for 60 hz machines). Its primary function is to provide other tasks with a wake up service. A typical use of the Clock Task would be to wake up the Line Printer Task every two seconds to check the Line Printer status for a change from OFF LINE to ON LINE. The Clock Task operates at the highest priority on the ATL.

3.2.3 Request Processing

When the PDP-15 issues a request to the PDP-11 to be carried out by PIREX, it does so by interrupting the PDP-11 at level 7 (the highest PDP-11 priority level) and simultaneously passing it the address of a Task Control Block (TCB) through the interrupt link. This address is called the Task Control Block Pointer (TCBP). A PDP-11 task can issue requests to other tasks via the IREQ macro. The IREQ macro simulates the PDP-15 request process and results in a TCBP being passed to PIREX. The contents of the Task Control Block completely describe the request (task addressed, function, optional interrupt return address and level, status words, etc.). The TCB will reside in the 'Common' Memory if the request is issued from the PDP-15 or in the 'Common' or 'Local' Memory if the request is issued from the PDP-11.

The flow chart in Figure 3-1 illustrates the basic processing of requests to PIREX from the PDP-15 or the PDP-11. Note that error conditions are passed back to either central processor in the TCB or via an error table to the PDP-15 monitor poller along with status information necessary for control and monitoring of a request. Usually the request is to a device on the PDP-11 but other types are allowed.

3.2.4 Task Structure

A task is a PDP-11 software routine capable of being requested by the PDP-15 or PDP-11 through the PIREX software system. The task may be a device driver, a directive processor, or just a software routine used to carry out a specified function. A task must have the format shown in Figure 3-2, TASK FORMAT.

```
                        ****   LOWER CORE
                        *  *
    task stack area     '  '
                        '  '
                        *  *
                        ****
    control register    *  *
                        ****
    busy/idle switch    *  *
                        *  *
                        ****
                        *  *
    task program        *  *
    code                '  '
                        *  *
                        *  *
                        ****   HIGHER CORE
```

Figure 3-2
Task Format

This structure consists of four sections; two are variable in size and two are fixed.

The "task program code" size is variable and contains the programming code necessary to carry out the task function.

The "busy/idle switch" consists of two words and is used by PIREX to determine if a task is busy or idle. The TCBP of the current request is stored in this section when the task is busy. This also enables a task to easily access the TCB.

The "control register" is either a dummy address (an address which points to an unused software variable) or the address of a device control register if the task is an I/O driver. This word is used only by the STOP TASKS (ST) task when shutting down I/O operations.

The "stack area" begins immediately below the control register and builds dynamically downwards. The purpose of the stack is to allow each task free use of a private space for temporary storage of data while it is executing and all its active registers during times when other higher priority tasks are being run. The stack area must be large enough to store the maximum number of temporary variables used at any one time plus one context register save. A context save requires 8 words of stack area plus an additional 3 words if the PDP-11 has an Extended Arithmetic Element (EAE). The stack size is fixed and determined at PIREX assembly time.

3.2.5 Task Control Block - TCB

Tasks, in PIREX, receive requests for action and return the results of their action in bundles of information called Task Control Blocks (TCB). The general format of a TCB consists of three words followed by task-specific optional words. The following information must be present in all TCBs since PIREX will honor requests in this format only.

	15 8	7 0	
TCB:	API TRAP ADDRESS	API LEVEL	WORD 0
	FUNCTION CODE	TASK CODE NUMBER	WORD 1
REV:	REQUEST EVENT VARIABLE		WORD 2
	OPTIONAL WORDS		WORD 3-N

3.2.5.1 API Trap Address and Level - The API trap address is a PDP-15 API trap vector and has a value between 0 and 177_8 when a hardware interrupt on the PDP-15 is required. Location 0 corresponds to location 0 in the PDP-15. The "API" level is the priority level at which the interrupt will occur in the PDP-15 and has a value between 0 and 3 when a hardware interrupt on the PDP-15 is required. A 0 signifies API level 0, a 1 for level 1, etc. The API trap address and level are used by tasks in the PDP-11 when informing the PDP-15 that the requested operation is complete (e.g., a disk block transferred or line printed). If the PDP-15 master computer doesn't have API or if API is not enabled, the PDP-11 issues an interrupt that when received is polled by the PDP-15 using 4 UC15 skips (one per level) on the traditional skip chain.

3.2.5.2 Function Code - The Function Code determines whether hardware interrupts on the PDP-15 or software interrupts on the PDP-11 are to be used at the completion of the request. If the code has a value of 0, a hardware interrupt is generated on the PDP-15 at the completion of the request; if a 1, an interrupt is not made. If the Function Code is a 3, a software interrupt is issued by PIREX. The task routine or program using this facility sets up the trap address in the SEND11 table in PIREX prior to issuing the request to the task. The task or route should return to PIREX after interrupt processing through an "RTS PC" instruction. All registers are available for use by tasks.

3.2.5.3 Task Code Number - The Task Code Number (TCN) is a positive $(1\text{-}177_8)$[1] or a negative $(200\text{-}377_8)$ 7-bit number plus a sign bit that informs PIREX which task is being referenced. The mnemonic TCN as used in this manual refers to the 7-bit portion of the Task Code Number. Tasks are addressed by a numeric value rather than by name. Tasks with positive code numbers are spooled tasks and tasks with negative code numbers are unspooled tasks. When the SPOOLER (see Chapter 5) is enabled and running, requests to spooled tasks are routed to the SPOOLER. When the SPOOLER is disabled, requests to spooled tasks are routed directly to device drivers.

Task Code Numbers are currently assigned as follows:

CODE[2]	TCN	TASK	
-1[3]	-1	CL task (Clock)	Driver task[3]
200	0	ST task (Stop Task)	Software task
201	1	SD task (Software Directive)	Directive task
202	2	RK task (Cartridge Disk)	Driver task
203	3	DT task (DECTAPE)	Driver task
4	4	LP task (Line Printer)	Driver task
5	5	CD task (Card Reader)	Driver task
6	6	PL task (Plotter)	Driver task
207	7	SP task (Spooler)	Background task
210	10	LV task (Printer/Plotter)	Driver task
211/11	11	Currently not used	-
212/12	12	Currently not used	-
213/13	13	Temporary Task Entry	Temporary task

[1] A task code of 0 indicates the STOP TASKS DIRECTIVE - See Section 3.5

[2] The code column corresponds to the typical task code in the TCB

[3] The minus 1 is represented internally as 377

PIREX is currently capable of handling these 13 tasks. Tasks 11-13 are spare task codes available for customer use.[1]

3.2.5.4 Request Event Variable - The REQUEST EVENT VARIABLE, commonly called REV, is initially cleared by PIREX (set to zero) when the TCB request is first received and later set to a value "n" (by the associated task) at the completion of the request. The values of "n" are:

0	=	request pending or not yet completed
1	=	request successfully completed
-200	=	(mod 2^{16}-1) nonexistent task referenced
-300	=	(mod 2^{16}-1) illegal API level given (illegal values are changed to level 3 and processed)
-400	=	(mod 2^{16}-1) illegal directive code given
-500	=	(mod 2^{16}-1) no free core in the PDP-11 local memory
-600	=	(mod 2^{16}-1) ATL node for this TCN missing
-777	=	(mod 2^{16}-1) request node was not available from the POOL (i.e., the node POOL was empty, and the referenced task was currently busy or the task did not have an ATL node in the Active Task List)

When an address is passed in a TCB as data, the receiver of the address must relocate it to correspond to the addressing structure in its memory space. For example, a PDP-15 address passed to the PDP-11 must first be multiplied by two to convert word to byte addressing and then the local memory size (LMS) of the PDP-11 must be added. For example,

PDP-11 address = (PDP-15 address *2) + LMS on PDP-11

The reverse is true for a PDP-11 address received by the PDP-15. For example,

PDP-15 address = (PDP-11 address - LMS)/2

3.3 SYSTEM TABLES AND LISTS

The PIREX system uses various tables, lists, and deques to control events within the system.

[1] See Section 4.4 for further information.

3.3.1 Active Tast List (ATL)

The selection of a task for execution by PIREX is accomplished by first scanning a priority-ordered linked list of all active tasks in the system called the Active Task List (ATL). An active task is one which satisfies one or more of the following conditions:

1. is currently executing
2. has a new request pending in its deque
3. is in a wait state, or
4. has been interrupted by a higher priority task

A task is inactive if there is no ATL node for it. A task can be in any one of the following states:

CODE	STATE	ACTIVITY
0	run	active
2	wait	active
4	exit	inactive

When a runnable task is found, the stack area and general purpose registers belonging to that task are restored and program control is transferred to it through an RTI instruction. Program execution normally begins at the first location of the task diagram code (see Figure 3-2) or at the point where the task was previously interrupted by a higher priority task, or in special cases at any desired location in the task using the 'PC' setting on the stack as in the RK task's error retry program logic. When a task is interrupted by other tasks, its general purpose registers are saved on its own stack. Control is returned to the interrupted task by restoring its stack pointer and then its active registers.

The ATL is rescanned when:

1. a new request is issued to a task
2. a previous request is completed
3. at the end of a clock interrupt
4. a task goes into a wait state

A task is said to be in a "wait" state when its ATL node exists and it is not runnable.

3.3.1.1 ATL Nodes - The Active Task List is a linked list containing 4 word entries called nodes.

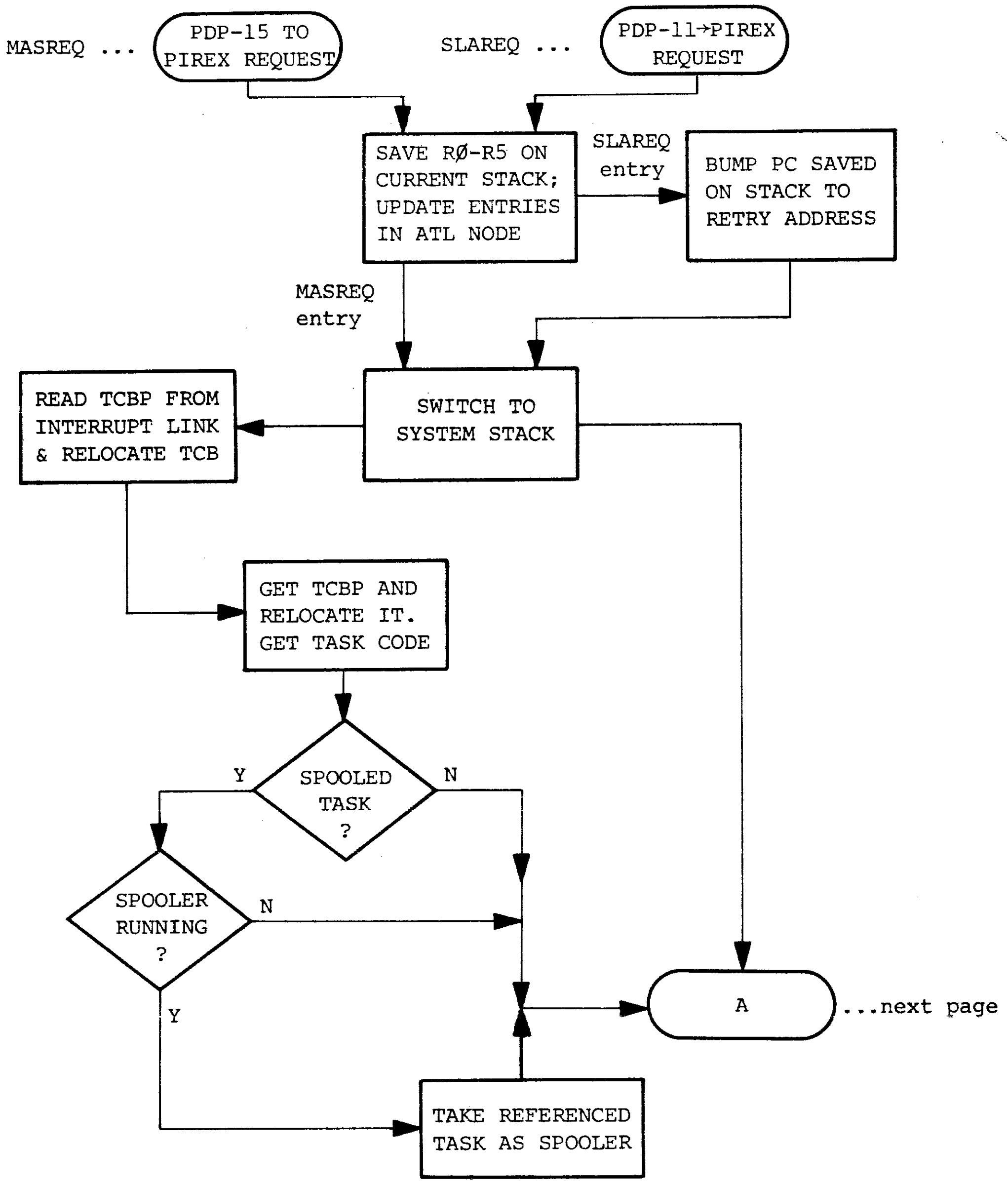

MASREQ ...
PDP-15 TO PIREX REQUEST
SLAREQ ...
PDP-11→PIREX REQUEST
SAVE RØ-R5 ON CURRENT STACK; UPDATE ENTRIES IN ATL NODE
SLAREQ entry
BUMP PC SAVED ON STACK TO RETRY ADDRESS
MASREQ entry
READ TCBP FROM INTERRUPT LINK & RELOCATE TCB
SWITCH TO SYSTEM STACK
GET TCBP AND RELOCATE IT. GET TASK CODE
Y
SPOOLED TASK ?
N
SPOOLER RUNNING ?
N
Y
A
...next page
TAKE REFERENCED TASK AS SPOOLER

Figure 3-3
Detailed Flow Chart of PDP-15/PDP-11 Request Processing

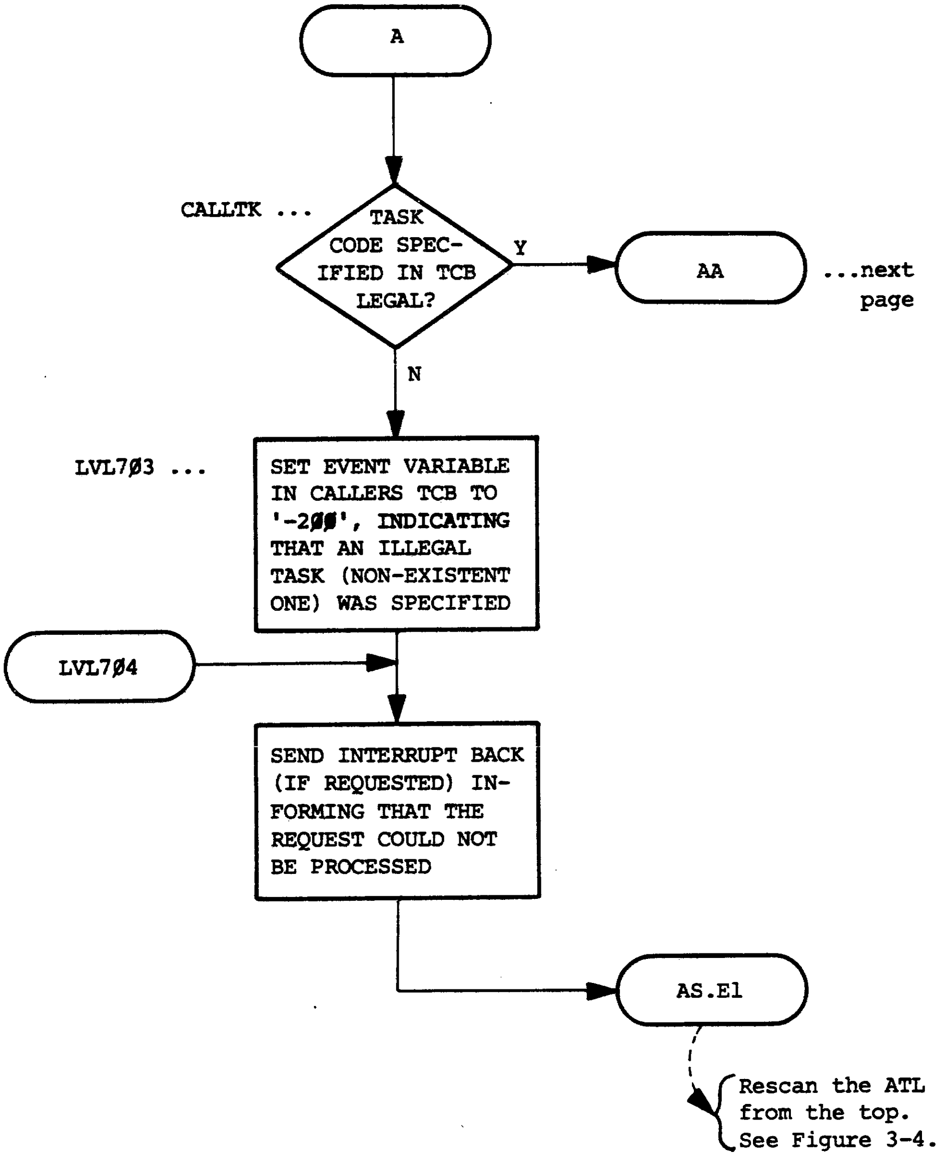

A
CALLTK ...
TASK CODE SPEC- IFIED IN TCB LEGAL?
Y
AA
...next page
N
LVL7Ø3 ...
SET EVENT VARIABLE IN CALLERS TCB TO '-2ØØ', INDICATING THAT AN ILLEGAL TASK (NON-EXISTENT ONE) WAS SPECIFIED
LVL7Ø4
SEND INTERRUPT BACK (IF REQUESTED) IN- FORMING THAT THE REQUEST COULD NOT BE PROCESSED
AS.E1
Rescan the ATL from the top. See Figure 3-4.

Figure 3-3 (Cont.)
Detailed Flow Chart of PDP-15/PDP-11 Request Processing

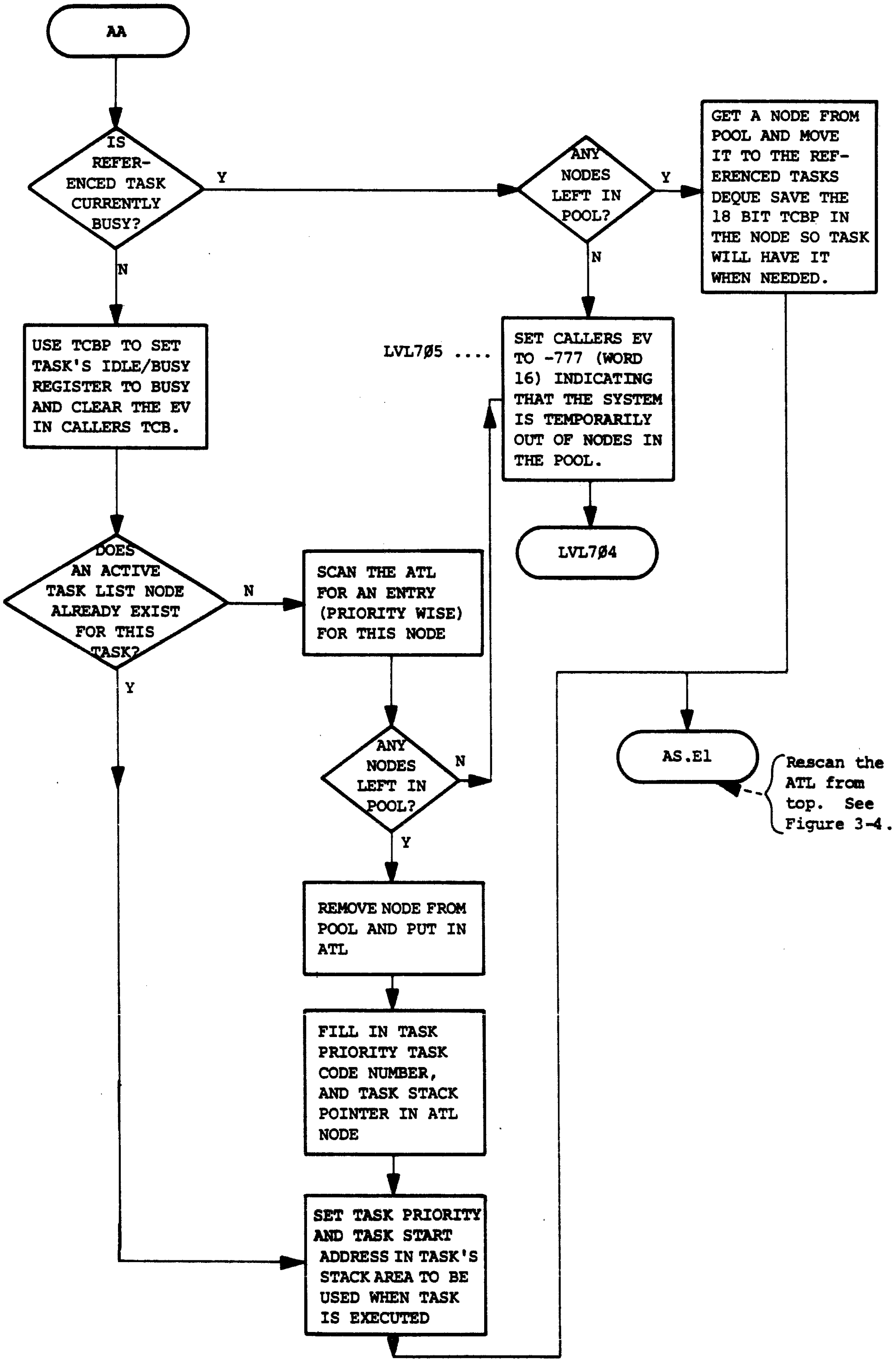

AA
IS REFERENCED TASK CURRENTLY BUSY?
Y
N
ANY NODES LEFT IN POOL?
Y
N
GET A NODE FROM POOL AND MOVE IT TO THE REFERENCED TASKS DEQUE SAVE THE 18 BIT TCBP IN THE NODE SO TASK WILL HAVE IT WHEN NEEDED.
USE TCBP TO SET TASK'S IDLE/BUSY REGISTER TO BUSY AND CLEAR THE EV IN CALLERS TCB.
LVL7Ø5 ....
SET CALLERS EV TO -777 (WORD 16) INDICATING THAT THE SYSTEM IS TEMPORARILY OUT OF NODES IN THE POOL.
LVL7Ø4
DOES AN ACTIVE TASK LIST NODE ALREADY EXIST FOR THIS TASK?
N
Y
SCAN THE ATL FOR AN ENTRY (PRIORITY WISE) FOR THIS NODE
ANY NODES LEFT IN POOL?
N
Y
AS.E1
Rescan the ATL from top. See Figure 3-4.
REMOVE NODE FROM POOL AND PUT IN ATL
FILL IN TASK PRIORITY TASK CODE NUMBER, AND TASK STACK POINTER IN ATL NODE
SET TASK PRIORITY AND TASK START ADDRESS IN TASK'S STACK AREA TO BE USED WHEN TASK IS EXECUTED

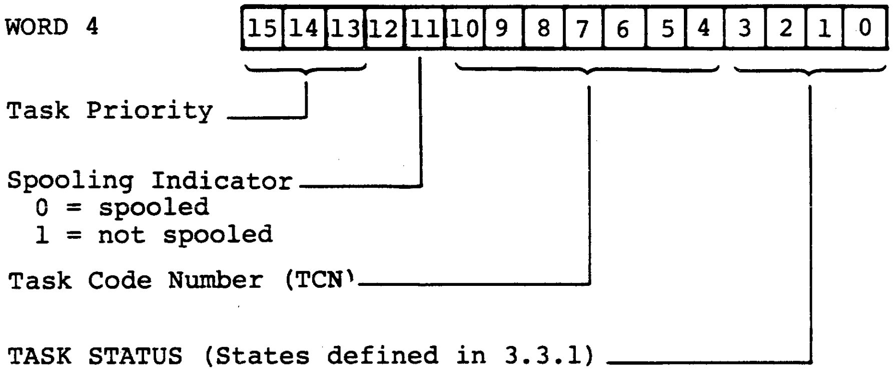
Figure 3-3 (Cont.)
Detailed Flow Chart of PDP-15/PDP-11 Request Processing

An ATL node has the following structure:

WORD 1 - Forward pointer to next node

WORD 2 - Backward pointer to previous node

WORD 3 - Stack pointer of task



The ATL is referenced by a 2-word listhead. The listhead contains backward and forward links pointing to the first and last nodes in the list. The ATL is a priority-ordered list.

3.3.1.2 ATL Node Pointer (ATLNP) - Each task has a pointer to its Active Task List Node (see Section 3.3.1.1) stored in the ATLNP table. This table is in TCN order. An entry is 0 if the task is inactive.

The format of an ATLNP entry is:

```
0 ; NAME task-code-number[1]
```

These entries are filled dynamically by PIREX with actual pointers.

3.3.2 Task Request List (TRL)

The Task Request Lists are doubly-linked, deque-structured lists of pending TCBs. If when a request arrives, the target task is busy, PIREX places the TCB pointer (TCBP) onto the busy task's deque for later processing. This deque is the Task Request List.

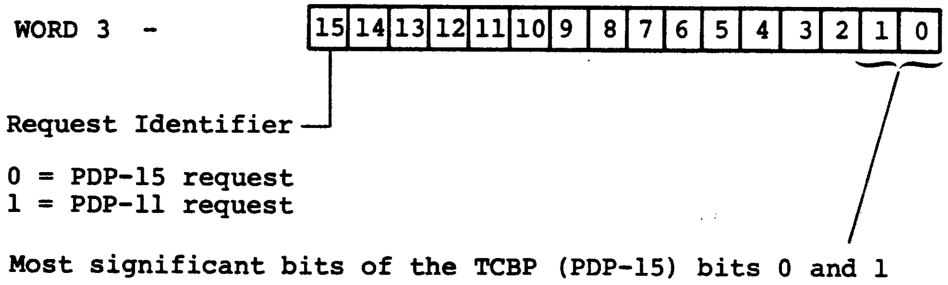
[1] The "NAME task-code-number" is a comment

A TRL node has the following structure:

WORD 1 - Forward pointer to next node.

WORD 2 - Backward pointer to previous node.

WORD 3 -



Request Identifier

0 = PDP-15 request
1 = PDP-11 request

Most significant bits of the TCBP (PDP-15) bits 0 and 1

WORD 4 - 16 least significant bits of TCBP (PDP-15 bits 2-17)

Each TRL is referenced by a two-word listhead. The listhead contains backward and forward links pointing to the last and first nodes of a given task's TRL. The TRL is built on a first come first serve basis.

3.3.3 TRL Listheads (LISTHD)

Each task has its own Task Request List, (TRL). Each LISTHD entry is a double-linked listhead used to point to a task's TRL. The LISTHD is a TCN ordered list.

The format for an entry is:

```
LISTHEAD  XX
```

where:

1. LISTHEAD is a system macro
2. XX is a two character task mnemonic (i.e., LP for Line Printer Task).

3.3.4 Clock Request Table (CLTABL)

The Clock Table (CLTABL) contains entries for one timing (wake up) request from each task. The format of a CLTABLE entry is:

XX[1].CL = .

.WORD 1 ; Time Word

.WORD 1 ; Address Word

[1] XX represents the task mnemonic (e.g., RK.CL)

Where the first word is remaining time before wakeup and the second word is the address for a JSR PC, XXX instruction. The JSR occurs at clock interrupt level (6). The user must do an RTS PC to return control to the clock routine. Time is measured in line frequency ticks: 16.6 milliseconds/tick for 60 Hertz Systems. A task may cancel a timing request by clearing the time word. A request for a wakeup is made by:

1. Placing the address of the routine to be called into word 2 - then

2. Placing the time delay (measured in 1/60 sec. increments) into the time word.

The above sequence must be exactly followed. See Chapter 4 for further details on the use of wakeup calls. CLTABL is a TCN ordered list.

3.3.5 Device Error Status Table (DEVST)

The DEVST table is used to store error status codes for delayed transfer to the PDP-15 monitor. The PDP-15 monitor contains a routine called the "Poller" which periodically requests error status codes from PIREX using a "get errors" software directive. This method of error transmission is useful for delayed error messages--such as those recognized on spooled devices. The specific PDP-15 I/O handler may no longer be present in the PDP-15's memory--thus the Request Event Variable (REV) method of returning error status would be useless. The "Poller" requests the entire DEVST table and reports those events on the system console terminal. A "Get Errors" directive clears the DEVST table upon completion. The reporting task may, for instance, correct the error condition before the "Get Error" directive is issued. When this happens, the task could simply clear its message from the DEVST table and thus eliminate a spurious message. DEVST is a TCN ordered table. The format of a DEVST entry is as follows:

```
WORD 1 - TASK (MNEMONIC IN SIXBIT/RAD50 RIGHT JUSTIFIED)

WORD 2 - SPARE (except for RK task where bad disk block is
               present)

WORD 3 - ERROR CODE:  SPOOLER ERROR CODE (HIGH BYTE)

                      TASK ERROR CODE (LOW BYTE)
```

3.3.6 LEVEL Table

The LEVEL table (task priority level) is used by the R.SAVE context switch routine to determine the priority level of the task about to begin execution. All interrupt vectors must specify a priority 7 entry into their respective interrupt routines. Upon entry, R.SAVE should be called to save the interrupt task state and return control to the interrupt processing routine at the proper priority--found in the LEVEL table. The LEVEL table is a TCN ordered task.

The LEVEL table entry format is:

```
.BYTE task priority *40
```

3.3.7 Task Starting Address (TEVADD)

The TEVADD Table contains the starting address of all defined tasks. The system currently has room for 13_8 tasks of which three are temporary entries used for tasks CONNECTED to and DISCONNECTED from PIREX. MAC11 is such a temporary task and uses the table entries of the currently unused highest task code. All PIREX systems must have at least one highest unused task entry to allow use of MAC11. The TEVADD table is TCN ordered.

The format of a TEVADD table entry is:

```
     .WORD START   ; task name
```

where START is either:

1. The starting address of the task, or,
2. 0 indicating that this entry is currently unoccupied.

where "Task name" is a comment.

3.3.8 Transfer Vector Table (SEND11)

The SEND11 table is used to store transfer vectors for use when issuing IREQ macro calls. The entry is the address at which the requesting routine receives control back from PIREX. This table is TCN ordered.

The format of a SEND11 entry is:

```
     0  ; task-name  task-code-number
```

where "task name task-code-number" is a comment.

3.3.9 System Interrupt Vectors

The device interrupt vector-pairs consist of interrupt routine address and priority level. The priority level of "all" devices should be Level-7 "only". This is to permit PIREX to do a context switch before processing the interrupt.

3.3.10 Internal Tables Accessible to All Tasks

All tasks in the PIREX system can easily access internal routines and tables through the use of the system registers. These registers begin at absolute location 1002_8 in the PDP-11 and contain either pointers to internal tables and listheads or entry points to commonly used subroutines. The following list summarizes these registers.

LOCATION	MNEMONIC			DESCRIPTION
01002		SEND11		INT. RETURN ADD. (ON 11) ON END OF I/O
01004	CURTSK:	000000		CURRENT TASK RUNNING
01006		POL.LH		ADDRESS OF POOL LISTHEAD
01010		LISTHD		ADDRESS OF TASK LISTHEADS
01012		R.SAVE		ENTRY POINT TO REGISTER SAVE
01014		R.REST		ENTRY POINT TO REGISTER RESTORE
01016		AS.E1		ENTRY POINT TO ATL RESCAN
01020		MOVEN		ENTRY POINT TO NODE MOVER
01022		DEQU		ENTRY POINT TO DEQUEUE
01024		SEND15		ENTRY POINT TO SEND INTERRUPT
01026		EMPTY		ENTRY POINT TO EMPTY A DEQUE
01030		ATLNP		ATL NODE POINTER TABLE
01032		RATLN		ENTRY POINT TO RETURN ATL NODE
01034		SPOLSW		SPOOLER SWITCHES ADDRESS
01036		RTURN		REUTURN INST. ADD. FOR PIC CODE
01040	NBRTEV:	NTEV		CURRENT NBR OF TASKS
01042	PWRDWN:	RTURN		ENTRY POINT TO PWR FAIL DOWN
01044	PWRUP:	RTURN		ENTRY POINT TO PWR FAIL UP
01046	SPOLSW:	000000		SPOOLER SWITCHES
01050		DEVST		DEVICE ERROR STATUS TABLE
01052		CLTABL		TABLE, A TIME-ADDR PAIR FOR EACH TASK
01054		DEQU1		ENTRY TO -SET TASK IN WAIT STATE-ROUTINE
01056		CEXIT		ENTRY TO -SET TASK IN RUN STATE-ROUTINE
01060		TEVADD		TABLE OF TASK START ADDRESSES
01062	DEVARE:	.WORD	DEVTYP	PIREX DEVICES SWITCH
01064	DEVSPL:	.WORD	0	DEVICES SPOOLED SWITCH
01066	CTLCNT:	.WORD	0	PDP-15 CTL C RUNNING COUNTER
01067	SPUNIT:	.WORD	0	DEVICE CURRENTLY BEING SPOOLED TO

```
;
;
```

These registers are accessed as absolute memory locations by various permanent and temporary tasks. NO CHANGE in the location or order of this table is permitted. New system registers may be added to the end of this table.

3.4 DETAILED THEORY OF OPERATION-PIREX

3.4.1 Request Procedure

The UC15 system allows the PDP-15 to initiate requests to the PDP-11 by interrupting at the highest PDP-11 hardware level and simultaneously passing to it an 18-bit Task Control Block address. Only the first 16 bits are used because PIREX does not support an external memory option[1] on the PDP-11. Requests from the PDP-15 or PDP-11 could be for:

(1) Memory management hardware support is not a feature of PIREX.

1. a directive-handing routine

2. a data transfer to or from a device driver task on the PDP-11

3. a background software routine (task)

3.4.2 Directive Handling[1]

Directive handling consists of such functions as:

1. Connecting and disconnecting tasks from the PIREX system

2. Reporting core status on the PDP-11 local memory to the calling routine

3. Stopping I/O on a particular device or all devices

4. Reporting UNIBUS device status to the calling routine

5. Stopping any or all tasks currently running[2]

6. Reporting spooler status to the caller

3.4.3 Logic Flow

The flow charts in Figures 3-3, 3-4, and 3-5 illustrate in detail the program logic flow when a request from the PDP-15 or PDP-11 is made to PIREX. Note that PIREX is capable of servicing requests in parallel on a priority basis.

3.4.4 Operating Sequence

PIREX is usually running the NUL task waiting for something to do. When a request is issued from the PDP-15 or PDP-11, PIREX immediately:

1. saves the general-purpose registers onto the stack belonging to the current task running

2. saves the stack pointer in the ATL nodes

3. sets the task in a RUN state

4. switches to the system stack (refer to Figure 3-5)

All of the preceding is done at level 7 (protected). The system stack is used when switching between tasks or rescanning the ATL.

In the case of a PDP-15 request, the TCBP (Task Control Block Pointer) register is now immediately read by the PDP-11 allowing additional requests to be made. PIREX corrects the TCBP by an amount equal to the PDP-11 local memory when a request comes from the PDP-15. The TCBP is present in R4 and R5 when the IREQ macrò is issued by a PDP-11 routine and the PDP-11 is able to address the TCB directly and retrieve information from it. The task code number is then obtained from the caller TCB and used to determine which task or directive that is being referenced.

[1] See Section 3.6 for additional information.
[2] See Section 3.5 for additional information.

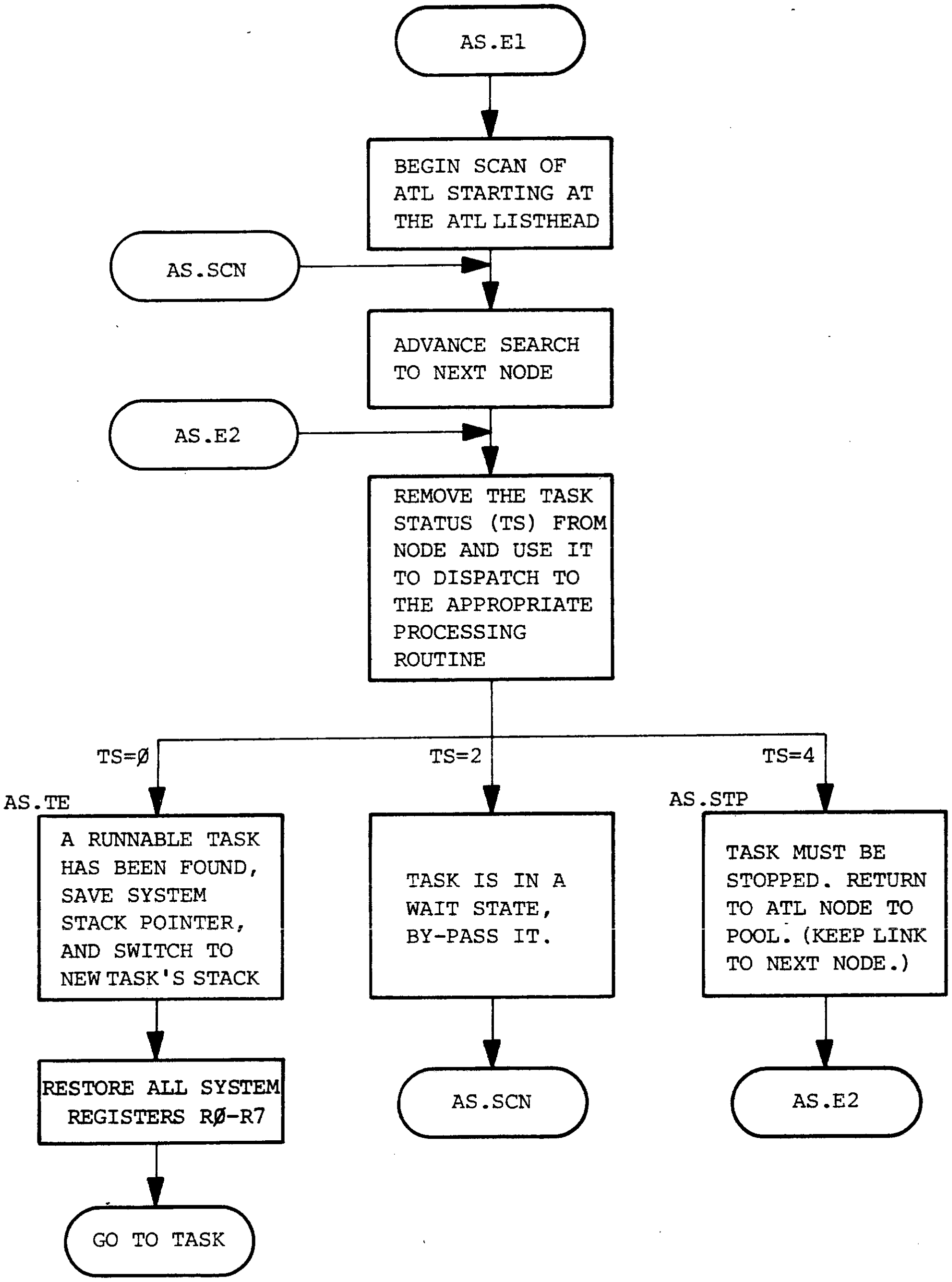

AS.E1
BEGIN SCAN OF ATL STARTING AT THE ATL LISTHEAD
AS.SCN
ADVANCE SEARCH TO NEXT NODE
AS.E2
REMOVE THE TASK STATUS (TS) FROM NODE AND USE IT TO DISPATCH TO THE APPROPRIATE PROCESSING ROUTINE
TS=Ø
TS=2
TS=4
AS.TE
A RUNNABLE TASK HAS BEEN FOUND, SAVE SYSTEM STACK POINTER, AND SWITCH TO NEW TASK'S STACK
TASK IS IN A WAIT STATE, BY-PASS IT.
AS.STP
TASK MUST BE STOPPED. RETURN TO ATL NODE TO POOL. (KEEP LINK TO NEXT NODE.)
RESTORE ALL SYSTEM REGISTERS RØ-R7
AS.SCN
AS.E2
GO TO TASK

Figure 3-4
Scan of Active Task List (ATL)

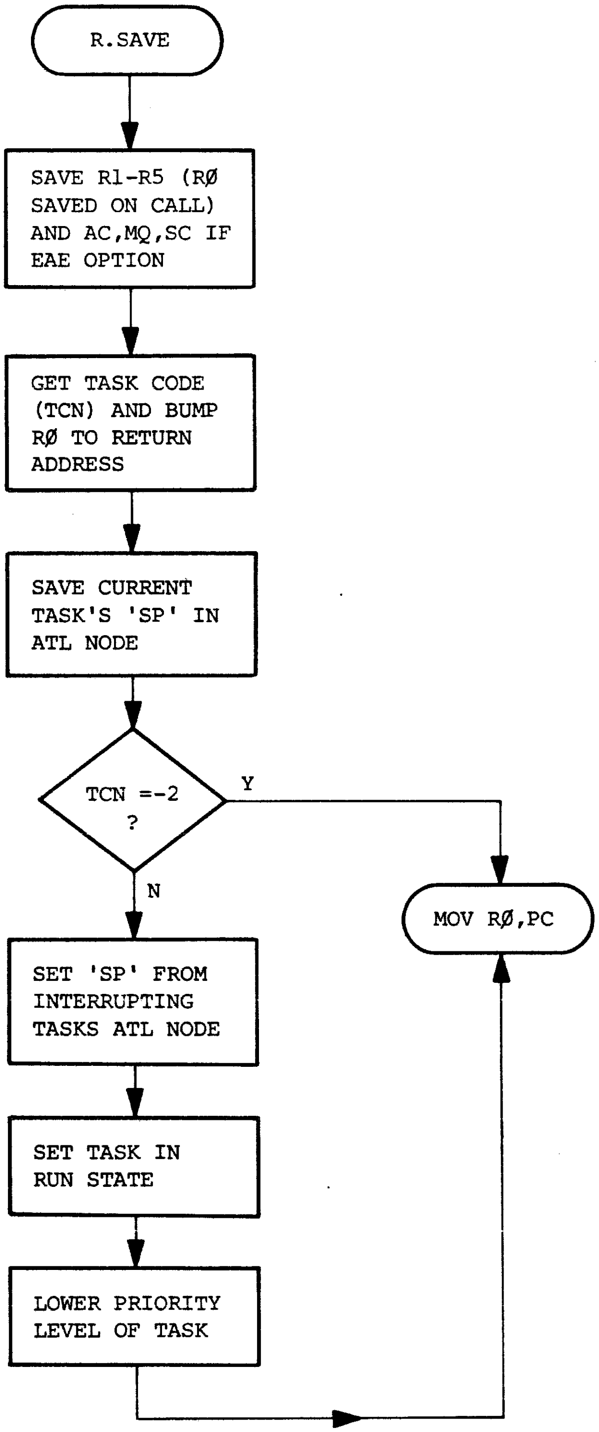

R.SAVE
SAVE R1-R5 (RØ SAVED ON CALL) AND AC,MQ,SC IF EAE OPTION
GET TASK CODE (TCN) AND BUMP RØ TO RETURN ADDRESS
SAVE CURRENT TASK'S 'SP' IN ATL NODE
TCN =-2 ?
Y
N
MOV RØ,PC
SET 'SP' FROM INTERRUPTING TASKS ATL NODE
SET TASK IN RUN STATE
LOWER PRIORITY LEVEL OF TASK

Figure 3-5
Context Switch or Save General Purpose Registers R0-R5.

A check is made to determine if the called task is a spooled task or not. If bit 7 = 0, it is a spooled task and if bit 7 = 1, it is an unspooled task. If the called task is a spooled task and if the SPOOLER is enabled, the request is processed by the SPOOLER. If the SPOOLER is not enabled, a check is made to determine if the task in reference is currently active and busy with a previous request. If so, the request is queued to the task's deque (TRL) on a first come, first serve basis. If the task in reference is currently inactive, an ATL node is built containing the appropriate entries, the address of the ATL node is set in the ATLNP table and the task's priority in the LEVEL table. In either case, the ATL is rescanned and the highest priority task is selected for execution (see Figure 3.4).

UC15 peripherals, controlled by PIREX, use a minimal driver to carry out requested functions and report the results back to the calling task via the TCB. When a driver finishes a request (whether an error occurred or not), it informs the requestor by placing the results (status and error register) in the TCB associated with that request and sends an optional hardware or software interrupt back to the requestor.

The request event variable (REV) is set prior to sending an interrupt to the PDP-15/PDP-11 and may be used by the PDP-15 or PDP-11 to determine if a request has been processed. This method is used during times when interrupts are not enabled or desired (as during the bootstrapping operation on the PDP-15). The hardware interrupt to the PDP-15 (see Figure 3-6) is optional and can be made at any of the PDP-15 API hardware levels and trap addresses. The API level and trap address are specified in the TCB associated with each request to allow complete flexibility in interrupt control.

3.4.5 Software Interrupt

A software interrupt return for the PDP-11 tasks is optional. This feature is available only if a hardware interrupt return to the PDP-15 is not required. To generate a software interrupt, the task using the request has to set the trap address before issuing the request. Each task running under PIREX has an entry in the SEND11 Transfer Vector Table. PIREX traps to this location on completion of a request by executing a JSR PC, SEND11 (Task Code *2). The task issuing the request specifies its task code in the TCB. All registers are free to be used when the control is transferred. Control is returned to PIREX through an RTS PC instruction.

3.4.6 Task Completion

When the PDP-15 has been notified (via interrupt) that its request has been completed, the task completing the request under PIREX becomes idle and calls DEQU (see Figure 3-7) to determine if any additional requests are pending. If no requests are pending, control is transferred to the ATL scanner (after saving the stack pointer and setting the current task in a wait state in its ATL node). If additional requests exist, the next request in the task's TRL is processed as if it were just received.

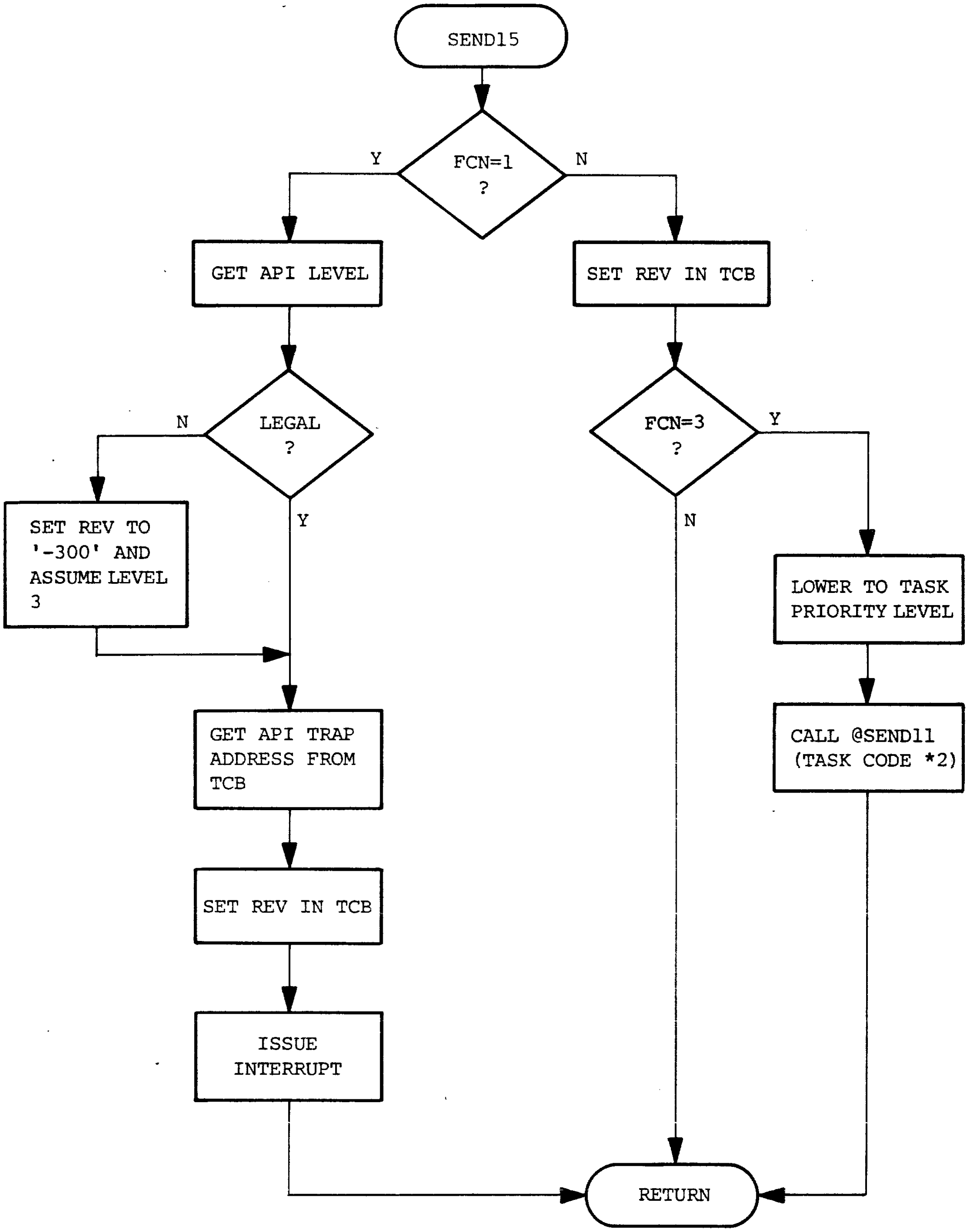

SEND15
FCN=1 ?
Y
N
GET API LEVEL
SET REV IN TCB
LEGAL ?
N
Y
FCN=3 ?
Y
N
SET REV TO '-300' AND ASSUME LEVEL 3
LOWER TO TASK PRIORITY LEVEL
GET API TRAP ADDRESS FROM TCB
CALL @SEND11 (TASK CODE *2)
SET REV IN TCB
ISSUE INTERRUPT
RETURN

Figure 3-6
Send Hardware Interrupt to PDP-15/Software Interrupt to PDP-11.

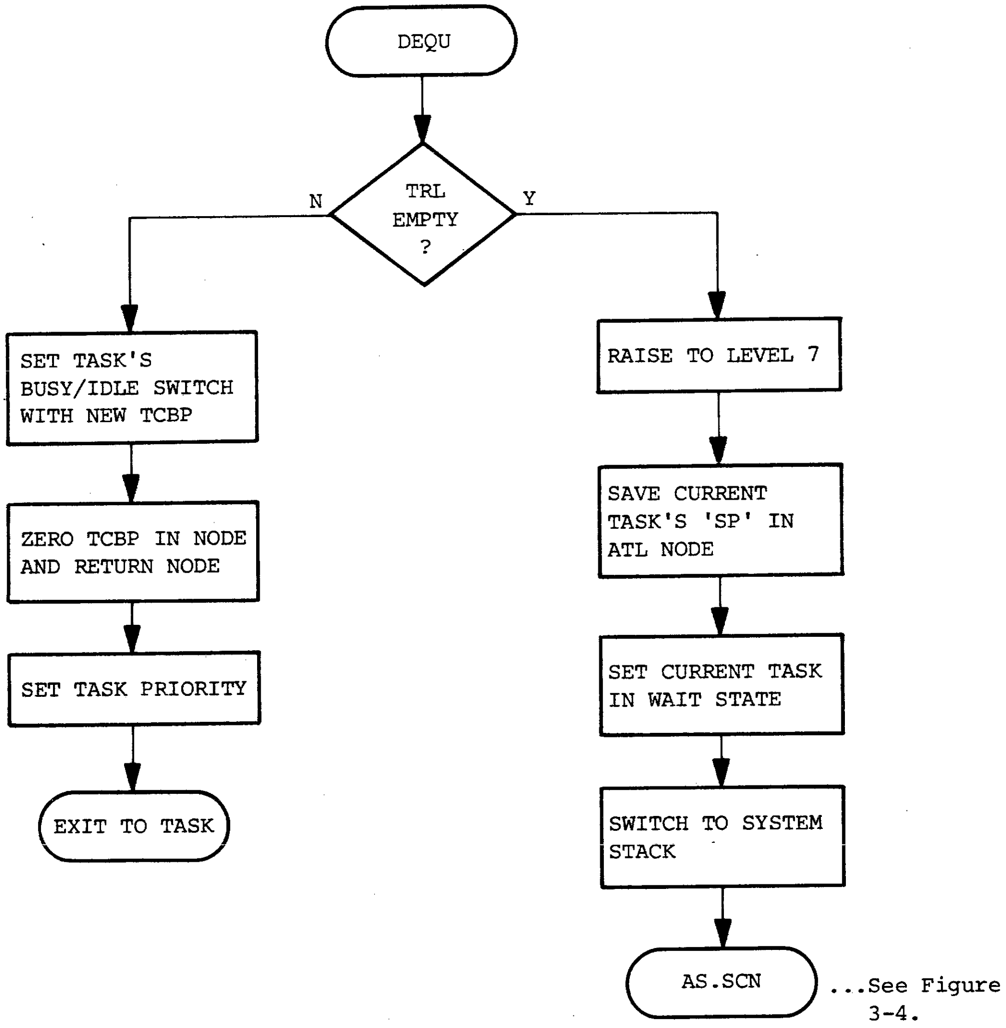

DEQU
TRL EMPTY ?
N
Y
SET TASK'S BUSY/IDLE SWITCH WITH NEW TCBP
ZERO TCBP IN NODE AND RETURN NODE
SET TASK PRIORITY
EXIT TO TASK
RAISE TO LEVEL 7
SAVE CURRENT TASK'S 'SP' IN ATL NODE
SET CURRENT TASK IN WAIT STATE
SWITCH TO SYSTEM STACK
AS.SCN
...See Figure 3-4.

Figure 3-7
Dequeue Node From Task's Deque.

3.5 STOP TASKS

The STOP TASKS Task is used to stop tasks and/or I/O currently underway for either all tasks or for a particular task. STOP TASKS can cancel all requests or only PDP-15 requests for the indicated task(s). There are four possibilities:

1. Stop all tasks unconditionally and cancel all pending PDP-15 requests

2. Stop a given task unconditionally and cancel all pending PDP-15 requests to that task

3. Cancel all PDP-15 requests to all tasks - this has no effect on PDP-11 requests

4. Cancel all PDP-15 requests to a given task - this has no effect on PDP-11 requests

The process of stopping a task includes (1 or 2 above):

1. Removal of all appropriate PDP-15 request nodes in the task(s) TRL(s)

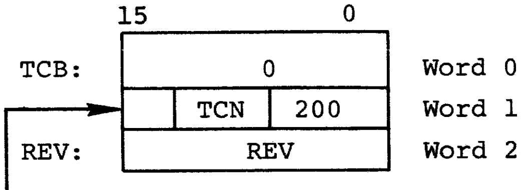
2. Zero the Busy Idle Switch for the task(s)

3. Clear the I/O device register(s) for the task(s)

4. Set the tasks status in the ATL to EXIT (for a temporary task) or WAIT (for a permanent task).

5. Indicate completion by setting the REV of the STOP TASKS requestor. (An interrupt return is <u>not</u> allowed.)

The Stop Tasks TCB has the following format:



bit 15 = 1 cancel PDP-15 requests and the current pending request unconditionally.

bit 15 = 0 cancel PDP-15 requests

TCN = 0 cancel all Tasks

TCN ≠ 0 cancel Task TCN only

REV = Return Event Variable

STOP TASKS is typically used by the PDP-15 operating system to quiet all interaction between the PDP-15 and the PDP-11.

3.6 SOFTWARE DIRECTIVE PROCESSING

The software directive task provides two main capabilities. These are:

1. The capability to connect and disconnect temporary tasks to PIREX (such as MACRO-11).

2. The capability to obtain various PIREX status information.

These capabilities are provided via five software directives, which are described later in this section.

The general format for software directive task control blocks is as follows:

15 8	7 Ø	
ATA	ALV	word Ø
FCN	2Ø1	word 1
REV		word 2
OPR		word 3
Contents Depend Upon Directive		word n

ATA PDP-15 API interrupt vector address

ALV PDP-15 API interrupt priority level. Must be 0, 1, 2, or 3 (unless FCN = 3).

FCN Function to perform upon completion of this software directive request. Valid values are:

000 Interrupt the PDP-15 at address ATA, priority ALV.

001 Do nothing (except set REV).

003 Cause a software interrupt to the PDP-11 task whose task code number is in ALV.

REV Request Event Variable. Initially zero, set to a non-zero value to indicate completion of the software directive request. The meaning of the various return values is described below.

OPR Indicates the exact operation (directive) to be performed. Must be one of the following values:

0 Disconnect Task

1 Connect Task

2 Core Status Report

3 Error Status Report

4 Spooler Status Report

5 MOVE

Returned REV values

1	Successful completion
-300	Invalid ALV value. The request may or may not have been performed - see individual directive descriptions. The PDP-15 will be interrupted at level 3.
-400	Invalid OPR (directive/operation code) value.
Other	See individual directive descriptions.

The following sections contain detailed descriptions of the individual software directives, their task control block (TCB) formats, and the REV values they may return.

3.6.1 Disconnect Task Directive

The disconnect task software directive instructs PIREX to delete a task from the active task list. Request should not be issued to a task after it has been disconnected. An attempt to issue a request to a disconnected task will result in a returned REV value of -200, implying that a non-existent task was referenced. The format of the task control block for the disconnect task software directive is as follows:

15 ... 8	7 ... ∅	
ATA	ALV	word ∅
FCN	2∅1	word 1
REV		word 2
∅∅∅	TCN	word 3
REL		word 4
First Address		word 5
unused		word 6
Length		word 7

TCN The task code number of the task to be disconnected.

REL 000000 if the task resides in PDP-15 memory
100000 if the task resides in PDP-11 memory

First Address PDP-11 byte address of the first location in memory occupied by this task (the lowest address of the task stack area). Only meaningful if the task resides in PDP-11 memory - if the task resides in PDP-15 memory this word is ignored.

Length Total size (in bytes) of this task, including stack area, control register, busy/idle switch, and program code. Only meaningful if the task resides in PDP-11 memory -- if the task resides in PDP-15 memory this word is ignored.

The disconnect task software directive verifies that the task to be disconnected is on the active task list. If present on the list, the task is disconnected - the active task list node is returned to the pool, the task's entry in the TEVADD table is cleared, and the task's task request list is cleared. If the task resides in PDP-11 memory, an attempt is made to free the memory space occupied by the task - if the first free local memory address is the address immediately following the storage area occupied by the task (as determined from the first address and length arguments), the task's first address becomes the new first free local memory address.

RESTRICTIONS:

1. If a task does not have an active task list node, it cannot be disconnected. Therefore, once a task has been connected, it cannot be disconnected until after a request has been issued to it.

2. All requests which are on the task request list of a task which is disconnected are forgotten. Such requests will never complete; their request event variables (REVs) will never be set to a non-zero value.

3. PDP-11 local memory resident tasks should only be disconnected if they are the last (highest address) task in local memory. If PDP-11 local memory resident tasks other than the last are disconnected first, the memory space occupied by these tasks will not be released. This will result in holes (of unusable memory) in the PDP-11's local memory.

4. Tasks should be disconnected in reverse sequential order by task code number. A task should not be disconnected if there are any connected tasks with higher task code numbers.

5. The high order bit of the task code number (TCN) must be clear.

Returned REV values:

1	Task successfully disconnected
2	Task successfully disconnected, but the (PDP-11 local) memory occupied by this task could not be released.
-300	Invalid ALV value, the task may or may not have been disconnected, its memory may or may not have been released.
-600	Task to be disconnected is not on the active task list (i.e., node not present)

3.6.2 Connect Task Directive

The connect task software directive instructs PIREX to add a new task to the system. Once a task has been connected to PIREX, the PDP-15 and/or other tasks may issue requests (task control blocks) to it. The format of the task control block for the connect task software directive is as follows:

15 8	7 Ø	
ATA	ALV	word Ø
FCN	2Ø1	word 1
REV		word 2
ØØ1	TCN	word 3
REL		word 4
unused		word 5
Entry Point		word 6
Length		word 7
unused	Priority	word 1Ø

TCN The new task's task code number (TCN)

REL 000000 if the new task resides in PDP-15 memory.
100000 if the new task resides in PDP-11 memory.

Entry Point Address of the new task's entry point - i.e., the first location of the task's program code. This address is a PDP-11 byte address if the new task resides in PDP-11 memory, a PDP-15 word address if the new task resides in PDP-15 memory.

Length Total size (in bytes) of the memory space occupied by this task, including stack area, control register, busy/idle switch, and program code. Only meaningful if the task resides in PDP-11 memory - if the task resides in PDP-15 memory this is ignored.

Priority The task's priority *40_8.

The connect task directive enters the new task start address (appropriately relocated if the new task resides in PDP-15 memory) into the TEVADD table. The directive does not actually create an active task list node for the new task; this occurs only when the first request is issued to the new task. The directive clears the new task's busy/idle switch (sets the task in idle state) and empties the new task's task request list. The new task priority is placed in the LEVEL table. If the new task resides in PDP-11 memory, PIREX updates its memory usage information by adding the size of the new task to the first free local memory address.

RESTRICTIONS:

1. The task code number must not be in use (correspond to any currently connected or permanently installed task) at the time this directive is issued.

2. The task code number must have been provided for when PIREX was assembled. As distributed by DEC, PIREX provides for task code numbers 0_8 through 13_8 inclusive.

3. The high order bit of the task code number must be clear.

4. If the task resides in PDP-11 memory, the first address it occupies must be the first free local memory address, as returned by the core status report software directive.

5. If the task resides in PDP-15 memory, it must reside entirely within the area addressable by the PDP-11's 28K addressing range.

6. Tasks should be connected in sequential order by task code numbers. Temporary tasks (tasks which will subsequently be disconnected) should always be connected to a task code number one higher than that obtained via the core status report software directive.

Returned REV values:

1	Task successfully connected
-300	Invalid ALV value. Task has been connected.

3.6.3 Core Status Report Directive

The core status report software directive returns information regarding PDP-11 local memory and task code number usage in PIREX. The format of the task control block for the core status report software directive is as follows:

15 8	7 Ø	
ATA	ALV	word Ø
FCN	2Ø1	word 1
REV		word 2
ØØ2	TCN	word 3
Local Memory Size		word 4
First Free Address		word 5
unused		word 6
Number of Free Words		word 7

TCN	Set to the highest currently connected task code number in PIREX.
Local Memory Size	The amount of local memory in the PDP-11 UNICHANNEL.
First Free Address	Set to the PDP-11 byte address of the first free (unoccupied) address in local memory.
Number of Free Words	Set to the number of unused words in PDP-11 local memory. Equal to ((Local memory size in bytes) - (First free address))/2.

RESTRICTIONS:

1. The core status report software directive has no restrictions. However, the restrictions (especially those regarding order of use of memory and task code numbers) on the connect and disconnect software directives must be adhered to in order to have valid information returned by core status report.

Returned REV values:

1	Successful completion
-300	Invalid ALV value. No information returned.
-500	No free PDP-11 memory. No information returned.

3.6.4 Error Status Report Directive

The error status report software directive returns information regarding device and/or spooler errors which have occurred since the last time this directive was issued. The format of the task control block for the error status software directive is as follows:

15 8	7 ∅	
ATA	ALV	word ∅
FCN	2∅1	word 1
REV		word 2
∅∅3	unused	word 3
Returned Error Information		word 4 word n

The error status report software directive copies error status information from the DEVST table onto the requestor's task control block, then clears the DEVST table to store new error information. The error information returned consists of a series of three word blocks, one per PIREX task. As distributed by DEC, eleven such blocks will be returned - one for each permanent task (excluding the clock task) plus two more for spare or temporary tasks. The number of these blocks returned may change, however, if users alter the number of tasks (especially permanent tasks) in PIREX. The format of each of these three word information blocks is as follows:

15 8,	7 ∅	
Task Name		word ∅
unused--zero		word 1
SPLERR	DEVERR	word 2

Task Name A three character (.SIXBT) mnemonic for the task to which the error information applies.

DEVERR Device error code for device associated with this task

SPLERR Spooler error-code for this task.

The mnemonics for the tasks and the order in which the blocks for the various tasks appear are as follows:

MNEMONIC	TASKS
EST	"Stop Task" task
ESD	Software directive task
DKU	RK (Cartridge) disk driver
DTU	DECTAPE driver
LPU	Line Printer driver
CDU	Card reader driver
GRU	XY (Plotter) driver
ESP	Spooler
LVU	LV11 printer/plotter driver
---	spare--no mnemonic
---	spare--no mnemonic

RESTRICTIONS: none

Returned REV values:

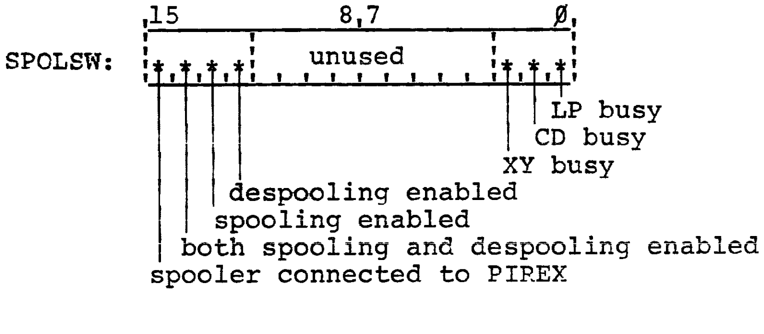
1	Successful completion
-300	Invalid ALV value. Information has been returned.

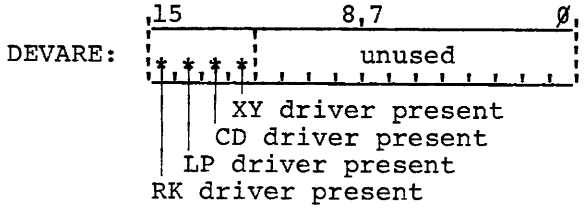
3.6.5 Spooler Status Report Directive

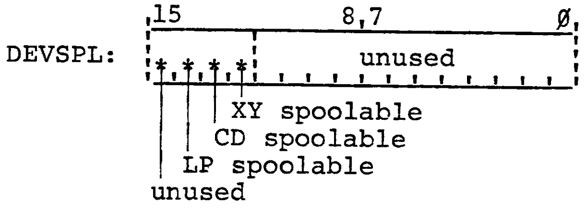
The spooler status report software directive returns information regarding spooler status and devices present in PIREX. The format of the task control block for the spooler status report software directive is as follows:

15 8	7 Ø	
ATA	ALV	word Ø
FCN	2Ø1	word 1
REV		word 2
ØØ4	unused	word 3
SPOLSW		word 4
DEVARE		word 5
DEVSPL		word 6
SPUNIT		word 7

SPOLSW, SPUNIT, DEVARE, and DEVSPL are four locations (within PIREX) in which information is kept concerning spooler status and which devices have been assembled into PIREX. The spooler status report software directive merely copies the contents of SPOLSW, SPUNIT, DEVARE, and DEVSPL into the task control block. Three of these words consist of a number of one-bit flags. If the bit is set (1) the corresponding condition is asserted: the device driver is present, spoolable, or busy; the activity is enabled. If the bit is clear (0) the opposite condition applies: the device driver is absent, non-spoolable, or idle, the activity is disabled. The exact format of these three words is as follows:







SPUNIT is the RK unit onto which the spooler is currently (or was previously) spooling data.

RESTRICTIONS:

1. DEVSPL and SPOLSW contain zero until after the first request has been issued to the spooler.

Returned REV value:

1	Successful completion
-300	Invalid ALV value. Information has been returned.

3.6.6 PIREX MOVE Directive

The PIREX MOVE directive moves information from one place in the PDP-11's address space to another place in its address space. (The address space is composed of both Local-11 and Common Memory.) The format of the task control block for the PIREX MOVE directive is as follows:

15 — 8	7 — Ø	
ATA	ALV	word Ø
FLN	2Ø1	word 1
REV		word 2
ØØ5		word 3
FROM LOCATION		word 4
TO LOCATION		word 5
WORDS TO MOVE		word 6

From Location	PDP-11 byte address of beginning of information to be moved.
To Location	PDP-11 byte address of a new starting location for information.
Words To Move	The number of words to move.

NOTE 1. This directive commonly is used to transfer information between common and local memory

CHAPTER 4

TASK DEVELOPMENT

4.1 INTRODUCTION

This chapter discusses in detail the procedure for developing a task and for installing it into the PIREX software system. The development of tasks in the UC15 system normally begins by the determination of the function to be performed by the task. Once the basic function of the task has been determined and designed, the user can integrate it into the UC15 system. The following summary describes the steps necessary to accomplish this:

1. Determine the priority level at which the task will execute.

2 Design one or more appropriate TCB formats

3. Assign a Task Code Number to the task

4. Enter appropriate information into the various PIREX lists and tables.

5. Design and code the requesting program. This is the program which issues requests to the task.

6. Design and code the task.

7. Assemble all programs and test.

The remaining sections describe these steps in detail.

4.2 PRIORITY LEVEL DETERMINATION

The selection of a priority level for a newly developed task must be based upon its function. If the task is a device driver, a device priority should be selected. If the task is a data manipulation routine, a background priority should be chosen.

4.2.1 Device Priorities

The device priorities are 7 (highest) through 4 (lowest)

- Priority 7 must be reserved for certain PIREX routines and should not be used as a task priority. (Certain short instructions sequences require priority level 7 protection but a general use of priority 7 must be avoided.)

- Priority 6 should be used only if interaction with the CR11 Card Reader can be avoided. If the CR11 is in use, excessive IOPSUC CDU 74 errors (card column lost) will occur if this level is used by another task executing in parallel.

- Priorities 4 and 5 can be used in an unrestricted manner.

There are three types of priorities to consider when selecting the priority of a device driver.

1. The actual device hardware priority N

2. The priority stored in the trap vector for the device (its new PS) must be priority 7 to allow an uninterrupted context switch.

3. The priority at which the task will execute after the context switch (R.SAVE). This should be N (the above constraints must be considered before deciding that it will be N). This priority is set in the LEVEL table (see Section 3.3.6).

4.2.2 Background Task Priorities

The standard UC15 PDP-11/05 computer does not differentiate between the software priorities 0 through 3. All software priorities are interruptable by any device operating at any device priority. These software priorities, while treated by the hardware as the same, are not treated by PIREX as identical. The background task's position in the Active Task List (the list to schedule the next task to run) is based upon its priority (as indicated in the LEVEL Table). Thus a priority 2 task is always selected for execution before a priority 1 task.

It should always be remembered that the ATL is built dynamically and is composed of only active tasks. Thus a task's actual ability to execute depends both on its priority and on what other tasks of equal or greater priority are actually available to execute (active). Tasks of the same priority are run on a first come-first serve basis.

4.3 TCB FORMAT AND LOCATION

The design of new Task Control Blocks (TCBs) must be governed by several constraints:

1. Certain "fixed" items of information must be present.

2. There may be a size constraint depending upon source of the TCB.

3. TCBs issued by the PDP-15 have a location constraint.

The first three TCB words have a fixed format (see Section 3.2.5). The remainder of the TCB should be as follows:

1. Control words should be allocated to fixed pre-defined locations.

2. Data words should be blocked into the location following the control words.

3. The TCB size should be kept constant for ease of core allocation.

Location and size constraints are interrelated:

1. If the TCB is for a task executing under PIREX in PDP-11 Local Memory, there is no location constraint. The TCB size must be kept small enough so that the TCB does not overflow into common memory.

2. If the TCB is for a PDP-11 task executing in Common Memory, it must be positioned so that it is:

   a. present entirely in Common memory (not PDP-15 Local Memory, and

   b. not overlaying any of the PDP-15 monitor resident code.

   These constraints actually apply to any PDP-11 Code or data located beyond PDP-11 Local Memory.

3. If the TCB is for an RSX-PLUS III routine, it must be located in a task partition or common area that is within the Common Memory.

4. Since the specification of absolute core location is difficult in DOS-15, the TCB placement problem is somewhat more complex. The standard DOS-15 system has seven TCBs assembled into the resident monitor. These include TCBs for RK Disk, XY11 Plotter, CR11 Card Reader and LP11/LV11/LS11 Printer. In addition there are three spare TCBs of various sizes. The user developing his own UNICHANNEL handler should take advantage of these spare TCBs. .SCOM + 100 (location 200_8 in PDP-15 memory) points to a table of pointers to each of these TCBs. The user should select the one closest to his size requirement. (See the DOS Systems Manual, DEC-15-ODFFA-B-D).

4.4 TASK CODE NUMBER DETERMINATION

Task code numbers are composed of two fields. Bits 6 through 0 are used to contain the actual task code number. This is the number used when searching tables and lists ordered by TCN. In the DEC-supplied system, these numbers range from 0 through 13_8. Bit 7 is used in TCBs to determine if the task is spooled. If bit 7 = 1, the task is not spooled. If bit 7 = 0, the TCBs for the task are routed to the spooler if the spooler is enabled. (There must then be a spooler module prepared to handle TCBs for that particular task (see Chapter 5)).

Task codes 11, 12, and 13 are spare task codes in the DEC-supplied system. They are used in increasing order. The highest task code position must not be used for a permanent task because MAC11 requires this slot for its use as a temporary task (a task that is connected and disconnected at run time.)

4.5 UPDATING LISTS AND TABLES

The installation of a new task requires placing entries into the various tables and lists. There are two cases:

1. the installation of a new task into a current spare task entry

2. the installation of a new task into a new entry (by expanding the tables)

For each of these two cases there are two types of task entries:

1. permanent tasks

2. temporary tasks

A permanent task is one that is assembled into the PIREX binary. Its actual starting address and priority level are known.

A temporary task is one that is dynamically connected to and disconnected from PIREX. Its starting address is dependent upon its placement in memory. (Temporary tasks must be written in Position Independent Code — see MAC11 Assembler Programmers Reference Manual DEC-15-LMCMA-A-D).

Chapter 3 describes the format of each table entry.

4.5.1 Temporary Task Installation — Existing Spare Entry

To install a Temporary Task into an Existing unused Task Entry, TCN 11_8, 12_8, or 13_8, simply use the CONNECT and DISCONNECT directives. No new table space and no new table entries are required.

4.5.2 Permanent Task Installation — Existing Spare Entry

To install a Permanent Task into an Existing unused Task Entry, TCN 11 or 12 perform the following:

1. Update the LEVEL table entry for that TCN with the task's priority (see Section 3.3.6).

2. Update the TEVADD Table entry for that TCN with the task's starting address (see Section 3.3.7).

4.5.3 Temporary Task — New Entry

To install a Temporary Task into a new Temporary Task Entry (i.e., to expand the table to accommodate a new Temporary Task) perform the following:

1. Add an entry to the ATLNP Table (see Section 3.3.1.2).
2. Add an entry to the LISTHD Table (see Section 3.3.3).
3. Add an entry to the LEVEL Table (use ".BYTE 0" as the priority value since this is a Temporary Task Entry and the actual task priority will be filled in by the connect directive).
4. Add an entry to the DEVST Table (see Section 3.3.5).[1]
5. Add an entry to the CLTABL (see Section 3.3.4).
6. Add an entry to the TEVADD Table (use ".WORD 0" as the entry, since this is a Temporary Task entry that will be filled in by the CONNECT directive).
7. Add an entry in the SEND11 Table (see Section 3.3.8).

4.5.4 Permanent Task Installation — New Entry

For a new Permanent Task, repeat the procedure in paragraph 4.5.3, for a new Temporary Task, with the following changes:

1. Step 3 is changed to: Place the task's priority in the new LEVEL Table entry (See Section 3.3.6).
2. Step 6 is changed to: Place the task's starting address in the new TEVADD entry (see Section 3.3.7).

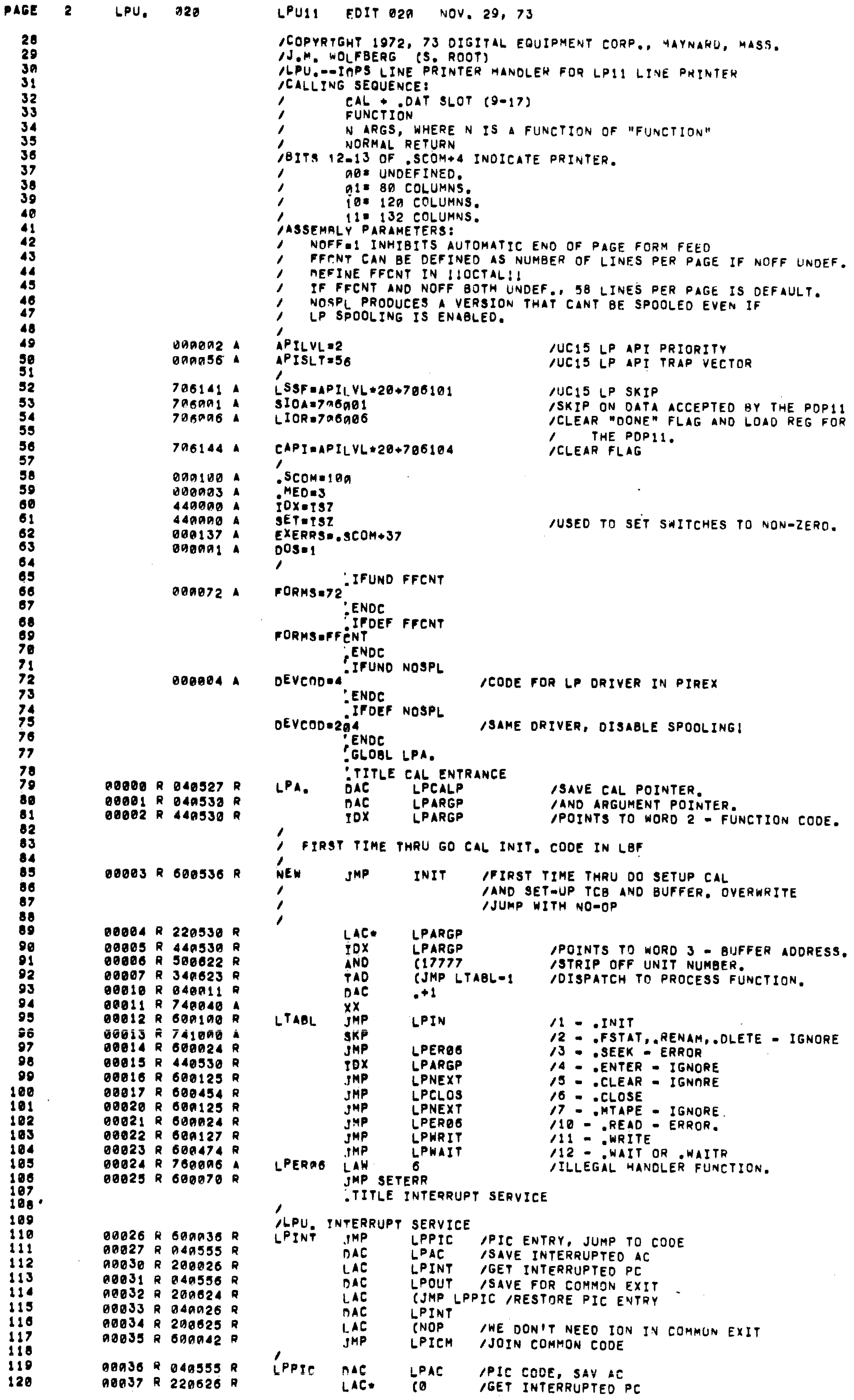
4.6 CONSTRUCTING DEVICE HANDLERS

This section describes how to construct device handlers for DOS-15 and RSX-PLUS III. Additional information on construction of a PDP-11 requesting task is provided.

4.6.1 Constructing a DOS UNICHANNEL Device Handler

The following description of how to construct a handler for the DOS-15 monitor does not discuss those topics related to all DOS-15 handlers both traditional and UNICHANNEL. General issues pertaining to all DOS-15 device handlers can be found in the DOS Systems Manual (DEC-15-ODFFA-B-D). The DOS-15 V3A000 UNICHANNEL Line Printer handler is usea as a descriptive example (see Figure 4-1). Several constants should be defined in a UNICHANNEL handler source file before the executable code (see Figure 4-1, lines 49-54, 72-75). These constants include:

[1] PIREX transfers, upon request, the entire DEVST Table to the PDP-15 monitor. The DOS resident monitor can accommodate a maximum of 5 additional DEVST entries beyond the current 13_8. Expansion beyond 20_8 entries would require reassembly of the DOS-15 resident monitor.



```
PAGE   2     LPU.    020            LPU11   EDIT 020   NOV. 29, 73

                                    /COPYRIGHT 1972, 73 DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
                                    /J.M. WOLFBERG  (S. ROOT)
                                    /LPU.--IOPS LINE PRINTER HANDLER FOR LP11 LINE PRINTER
                                    /CALLING SEQUENCE:
                                    /       CAL + .DAT SLOT (9-17)
                                    /       FUNCTION
                                    /       N ARGS, WHERE N IS A FUNCTION OF "FUNCTION"
                                    /       NORMAL RETURN
                                    /BITS 12-13 OF .SCOM+4 INDICATE PRINTER.
                                    /       00= UNDEFINED.
                                    /       01= 80 COLUMNS.
                                    /       10= 120 COLUMNS.
                                    /       11= 132 COLUMNS.
                                    /ASSEMBLY PARAMETERS:
                                    /   NOFF=1 INHIBITS AUTOMATIC END OF PAGE FORM FEED
                                    /   FFCNT CAN BE DEFINED AS NUMBER OF LINES PER PAGE IF NOFF UNDEF.
                                    /   DEFINE FFCNT IN !!OCTAL!!
                                    /   IF FFCNT AND NOFF BOTH UNDEF., 58 LINES PER PAGE IS DEFAULT.
                                    /   NOSPL PRODUCES A VERSION THAT CANT BE SPOOLED EVEN IF
                                    /   LP SPOOLING IS ENABLED.
                                    /
                   000002 A         APILVL=2                          /UC15 LP API PRIORITY
                   000056 A         APISLT=56                         /UC15 LP API TRAP VECTOR
                                    /
                   706141 A         LSSF=APILVL*20+706101             /UC15 LP SKIP
                   706001 A         SIOA=706001                       /SKIP ON DATA ACCEPTED BY THE PDP11
                   706006 A         LIOR=706006                       /CLEAR "DONE" FLAG AND LOAD REG FOR
                                                                      /    THE PDP11.
                   706144 A         CAPI=APILVL*20+706104             /CLEAR FLAG
                                    /
                   000100 A         .SCOM=100
                   000003 A         .MED=3
                   440000 A         IDX=IS7
                   440000 A         SET=ISZ                           /USED TO SET SWITCHES TO NON-ZERO.
                   000137 A         EXERRS=.SCOM+37
                   000001 A         DOS=1
                                    /
                                          .IFUND FFCNT
                   000072 A         FORMS=72
                                          .ENDC
                                          .IFDEF FFCNT
                                    FORMS=FFCNT
                                          .ENDC
                                          .IFUND NOSPL
                   000004 A         DEVCOD=4                /CODE FOR LP DRIVER IN PIREX
                                          .ENDC
                                          .IFDEF NOSPL
                                    DEVCOD=204              /SAME DRIVER, DISABLE SPOOLING!
                                          .ENDC
                                          .GLOBL LPA.
                                          .TITLE CAL ENTRANCE
          00000 R 040527 R          LPA.    DAC      LPCALP            /SAVE CAL POINTER.
          00001 R 040530 R                  DAC      LPARGP            /AND ARGUMENT POINTER.
          00002 R 440530 R                  IDX      LPARGP            /POINTS TO WORD 2 - FUNCTION CODE.
                                    /
                                    /  FIRST TIME THRU GO CAL INIT. CODE IN LBF
                                    /
          00003 R 600536 R          NEW     JMP      INIT    /FIRST TIME THRU DO SETUP CAL
                                    /                        /AND SET-UP TCB AND BUFFER. OVERWRITE
                                    /                        /JUMP WITH NO-OP
                                    /
          00004 R 220530 R                  LAC*     LPARGP
          00005 R 440530 R                  IDX      LPARGP            /POINTS TO WORD 3 - BUFFER ADDRESS.
          00006 R 500622 R                  AND      (17777            /STRIP OFF UNIT NUMBER.
          00007 R 340623 R                  TAD      (JMP LTABL-1      /DISPATCH TO PROCESS FUNCTION.
          00010 R 040011 R                  DAC      .+1
          00011 R 740040 A                  XX
          00012 R 600100 R          LTABL   JMP      LPIN              /1 - .INIT
          00013 R 741000 A                  SKP                        /2 - .FSTAT,.RENAM,.DLETE - IGNORE
          00014 R 600024 R                  JMP      LPER06            /3 - .SEEK - ERROR
          00015 R 440530 R                  IDX      LPARGP            /4 - .ENTER - IGNORE
          00016 R 600125 R                  JMP      LPNEXT            /5 - .CLEAR - IGNORE
          00017 R 600454 R                  JMP      LPCLOS            /6 - .CLOSE
          00020 R 600125 R                  JMP      LPNEXT            /7 - .MTAPE - IGNORE
          00021 R 600024 R                  JMP      LPER06            /10 - .READ - ERROR.
          00022 R 600127 R                  JMP      LPWRIT            /11 - .WRITE
          00023 R 600474 R                  JMP      LPWAIT            /12 - .WAIT OR .WAITR
          00024 R 760006 A          LPER06  LAW      6                 /ILLEGAL HANDLER FUNCTION.
          00025 R 600070 R                  JMP SETERR
                                            .TITLE INTERRUPT SERVICE
                                    /
                                    /LPU. INTERRUPT SERVICE
          00026 R 600036 R          LPINT   JMP      LPPIC   /PIC ENTRY, JUMP TO CODE
          00027 R 040555 R                  DAC      LPAC    /SAVE INTERRUPTED AC
          00030 R 200026 R                  LAC      LPINT   /GET INTERRUPTED PC
          00031 R 040556 R                  DAC      LPOUT   /SAVE FOR COMMON EXIT
          00032 R 200624 R                  LAC      (JMP LPPIC /RESTORE PIC ENTRY
          00033 R 040026 R                  DAC      LPINT
          00034 R 200625 R                  LAC      (NOP    /WE DON'T NEED ION IN COMMON EXIT
          00035 R 600042 R                  JMP      LPICM   /JOIN COMMON CODE
                                    /
          00036 R 040555 R          LPPIC   DAC      LPAC    /PIC CODE, SAV AC
          00037 R 220626 R                  LAC*     (0      /GET INTERRUPTED PC
```

Figure 4-1
PDP-15 LP11 DOS Handler

```
00040 R 040556 R            DAC      LPOUT   /SAVE
00041 R 200627 R            LAC      (ION    /NEED INTERRUPT ON INST. IN COMMON CODE
00042 R 040052 R    LPICM   DAC      LPISW
00043 R 706144 A            CAPI             /CLEAR FLAG, NOW IN COMMON CODE
00044 R 220542 R            LAC*     LPEV    /EVENT VARIABLE FROM PIREX
00045 R 742010 A            RTL              /PDP-11 (MINUS) BIT TO OUR AC0
00046 R 743120 A            SPA!RTR          /+ IS OK
00047 R 600055 R            JMP      LPIERR  /ERROR, GO LOOK
00050 R 140533 R    LPIRT   DZM      LPUND   /CLEAR UNDERWAY FLAG
00051 R 200555 R    LPIRT1  LAC      LPAC    /RESTORE AC
00052 R 740040 A    LPISW   HLT              /ION OR NOP
00053 R 703344 A            DBR
00054 R 620556 R            JMP*     LPOUT
                    /
                    /
00055 R 500630 R    LPIERR  AND      (177777 /KEEP REAL 16 BITS FROM PDP-11
00056 R 540631 R            SAD      (177001 /CODE FROM OUT OF NODES IN PIREX
00057 R 600062 R            JMP      RETRY   /JUST TRY AGAIN, LEAVING LPUND SET
00060 R 340632 R            TAD      (600000 /MAKE - NUMBER FOR IOPS
00061 R 600070 R            JMP      SETERR  /TREAT AS REGULAR IOPS ERROR
                    /                        /NOTE THAT THIS SHOULDN'T HAPPER.
                    /
                    /
00062 R 200537 R    RETRY   LAC      LPTCB   /TCB ADDRESS
00063 R 160542 R            DZM*     LPEV    /CLEAR EVENT VARIABLE
00064 R 706001 A            SIOA
00065 R 600064 R            JMP      .-1     /
00066 R 706006 A            LIOR             /THIS SENDS THE TCB ADDR. TO THE PDP-11
00067 R 600051 R            JMP      LPIRT1  /EXIT FROM INTERRUPT
                    /
                    /

                            .TITLE  ERROR ROUTINE
                    /
00070 R 040077 R    SETERR  DAC ERRNUM
00071 R 740000 A    ERLOOP  NOP                          /'JMP LPTRY' IF IOPS 4 ERROR.
00072 R 200077 R            LAC ERRNUM
00073 R 120633 R    EROUT   JMS* (EXERRS
00074 R 600071 R            JMP ERLOOP
00075 R 777777 A            LAW -1
00076 R 142025 A            .SIXBT 'LPU'
00077 R 000000 A    ERRNUM  0                            /HOLDS ERROR NUMBER FOR REPEAT.
                            .TITLE .INIT FUNCTION
                    /
                    /.INIT
                    /
00100 R 440530 R    LPIN    IDX      LPARGP
00101 R 200544 R            LAC BUFSIZ                   /36(10) FOR 80 COLS; 56(10) FOR 132 COLS.
00102 R 060530 R            DAC* LPARGP                  /RETURN TO USER.
00103 R 440530 R            IDX LPARGP                   /NOW POINTS TO RETURN.
00104 R 200531 R            LAC      PAGSIZ  /LF COUNTER
00105 R 040532 R            DAC      PAGCNT
00106 R 220527 R            LAC*     LPCALP  /DOES INIT INHIBIT AUTO FORMS FEED
00107 R 500634 R            AND      (4000   /THIS IS INHIBIT BIT
00110 R 340535 R            TAD      FFFF    /FFFF ASSEMBLED AS NOP FOR NOFF, ISZ IF NOT
00111 R 540535 R            SAD      FFFF    /SKIP IF INIT INHIBITS FF
00112 R 741000 A            SKP              /INIT DOESN'T INHIBIT, USE ASSEMBLED VALUE
00113 R 200625 R            LAC      (NOP    /INIT INHIBITS IT, USE NOP
00114 R 040534 R            DAC      FFSW    /THIS SWITCH XCT'ED BY FORMS CONTROL
                    /                        /SECTION IN PUTCH SUBROUTINE
00115 R 100443 R            JMS      RESETL  /RESET TAB AND LINE WIDTH COUNTERS
00116 R 100512 R            JMS      LPIOCK  /CHECK LP BUSY
00117 R 140551 R            DZM      COP     /SAY A FF OCCURRED
00120 R 750030 A            CLA!IAC          /COUNT OF ONE BYTE FOR HEADER
00121 R 060540 R            DAC*     LPBUF   /HEADER
00122 R 723013 A            AAC      13      /FORM FEED
00123 R 060541 R            DAC*     LPBUFD  /FOR BUFFER
                            .IFUND   NOFF    /DO ONLY IF NOFF NOT DEFINED
00124 R 100517 R            JMS      LPSET   /THIS SENDS REQ. TO PDP-11
                            .ENDC
                    /
                    /NORMAL CAL EXIT
                    /
00125 R 703344 A    LPNEXT  DBR
00126 R 620530 R            JMP*     LPARGP
                            .TITLE .WRITE FUNCTION
                    /
                    /.WRITE
                    /
00127 R 100512 R    LPWRIT  JMS LPIOCK                   /PRINTER BUSY?
00130 R 220527 R            LAC* LPCALP                  /GET THE DATA MODE FROM THE USER CAL.
00131 R 500635 R            AND      (1000   /MAKE SKP=NOP IN MIX
00132 R 240636 R            XOR      (SKP
00133 R 040554 R            DAC      MIX
00134 R 220530 R            LAC*     LPARGP              /USER BUFFER ADDRESS.
00135 R 440530 R            IDX LPARGP       /NOW POINTS TO WORD COUNT
00136 R 040550 R            DAC      TCHAR   /SAVE POINTER TO BUFFER HEADER
00137 R 723002 A            AAC      2       /MAKE X12 POINT TO DATA NOT HEADER
00140 R 040557 R            DAC      X12     /GETTER POINTER
                    /
                    /  SET-UP LIMIT OF INPUT BUFFER SIZE TO PREVENT DATA OVERRUN
                    /  FOR BOTH IOPS ASCII AND IMAGE ASCII
                    /
00141 R 777000 A            LAW      17000   /GET PAIR COUNT FROM LEFT HALF
00142 R 520550 R            AND*     TCHAR
00143 R 742030 A            SWHA             /BRING TO RIGHT. PAIR COUNT INCLUDES HEADER
                    /                        /PAIR COUNT. WE ISZ BEFORE LOOP SO THAT'S
```

Figure 4-1
PDP-15 LP11 DOS Handler (cont.)

```
                    /                       /OK. IOPS NOW SET XCPT CMA!IAC
00144 R 400554 R            XCT     MIX     /SKIP IF ASCII, NOT IF IMAGE
00145 R 751001 A            SKP!CLA!CMA     /IMAGE -1 IN AC, SKIP. -1 BECAUSE WE ISZ FIRST
00146 R 741031 A            SKP!CMA!IAC     /IOPS COMPLEMENTED TO CORRECT VALUE
00147 R 360530 R            TAD*    LPARGP  /IMAGE ADD IN TOTAL WORD COUNT, INCL
                    /                       /TWO WORDS FOR HEADER. WE ISZ BEFORE LOOP.
00150 R 040543 R            DAC     TEMP1   /INTO CONTROLLER, BOTH MODES
00151 R 440530 R            ISZ     LPARGP  /MOVE ARG POINTER TO EXIT
00152 R 200541 R            LAC     LPBUFD  /POINTER TO DATA PORTION OF BUFFER
00153 R 040560 R            DAC     PUTP    /LOAD TO CHARACTER PUTTER POINTER
00154 R 200335 R            LAC     GETIN   /INIT. CHAR GETTER
00155 R 040332 R            DAC     GETSW
00156 R 200431 R            LAC     PUTIN   /INIT CHAR PUTTER
00157 R 040427 R            DAC     PUTSW
00160 R 750000 A            CLA             /INIT OUTPUT BUFFER HEADER
00161 R 400554 R            XCT     MIX     /TO 0 IF IOPS, 400 FOR IMAGE
00162 R 200637 R            LAC     (400
00163 R 060540 R            DAC*    LPBUF
00164 R 750001 A            CLA!CMA         /COUNT OF 1 BLANK AS DEFUALT
                    /                       /FOR ZERO LENGTH IOPS LINE
00165 R 060541 R            DAC*    LPBUFD  /IN FIRST DATA CHAR
                    /
                    /  MAIN LOOP TO TRANSFER CHAR'S TO HANDLER BUFFER
                    /
00166 R 100320 R    MAIN    JMS     GETCH   /CHARACTER GETTER, LEAVES IT IN AC
00167 R 741200 A            SNA             /SKIP UNLESS NULL CHAR
00170 R 600166 R            JMP     MAIN    /NULL, IGNORE
00171 R 540640 R            SAD     (177    /IGNORE RUB-OUT
00172 R 600166 R            JMP     MAIN    /MAIN
00173 R 040550 R            DAC     TCHAR   /SAVE CHAR THROUGH TESTING
00174 R 723740 A            AAC     -40     /SEPARATE 'TEXT' CHAR'S FROM CONTROL CHAR'S
00175 R 741300 A            SNA!SPA         /SKIP ON REGULAR CHARS
00176 R 600235 R            JMP     MSPEC   /GO DO SPECIALS
00177 R 540641 R            SAD     (135    /ALT MODE
00200 R 600302 R            JMP     UCLP03  /END OF LINE ON ALT MODE
                    /
                    /  THE LOGIC AT PUTCH TO DO FORMS CONTROL DOESN'T DO IMPLIED
                    /  LINE FEEDS, I.E. THOSE LINES HAVING NO LEADING CONTROL CHAR.
                    /  WE MUST FAKE IT OUT BY PLACING A LINE FEED ON SUCH LINES!
                    /
00201 R 200547 R            LAC     FIRST   /DO ONLY IF FIRST CHAR OF LINE IS REGULAR
00202 R 740100 A            SMA             /SKIP IF FIRST CHAR
00203 R 600205 R            JMP     .+3     /NOT FIRST CHAR, JUST CONTINUE
00204 R 200642 R            LAC     (12     /HERE IS LINE FEED
00205 R 100366 R            JMS     PUTCH   /AND CALL TO DO FORMS CONTROL
                    /
00206 R 750030 A            CLA!IAC         /SET FLAG SAYING A REAL CHAR SINCE A FF
00207 R 040551 R            DAC     COP
                    /
00210 R 200552 R            LAC     BLANKC  /DO WE HAVE PENDING BLANKS/TABS TO SEND
                    /
                    /  NOTE BLANKC HAS MINUS COUNT OF CONSECTIVE BLANKS/TABS
                    /  SINCE PDP-11 CONTROLLER PRINTS ONLY BLANKS
                    /
00211 R 744100 A            SMA!CLL         /SKIP IF ANY COLLECTED, TO PUT OUT BEFORE
                    /                       /REAL CHAR'S
00212 R 600223 R            JMP     MAINC   /NONE, PENDING, GO PUT OUT THE CHAR
00213 R 340643 R            TAD     (200    /TOUGH, IF MORE THAN 127 COLLECTED, MUST
                    /                       /PUT OUT TWO COUNTS
00214 R 750100 A            SMA!CLA         /SKIP IF NEED TWO COUNTS
00215 R 600221 R            JMP     MAIND   /NO, JUST PUT OUT COLLECTED COUNT
00216 R 340643 R            TAD     (200    /TWO COUNTS, HERE IS FIRST
00217 R 100366 R            JMS     PUTCH
00220 R 200643 R            LAC     (200    /SET UP TO DO SECOND
00221 R 340552 R    MAIND   TAD     BLANKC  /COMMON CODE, LAST COUNT FOR EITHER CASE
00222 R 100366 R            JMS     PUTCH
00223 R 140552 R    MAINC   DZM     BLANKC  /CLEAR OUT BLANK COUNTER
00224 R 200550 R            LAC     TCHAR   /GET BACK ORIGINAL CHAR
00225 R 100366 R            JMS     PUTCH   /TO OUTPUT BUFFER
00226 R 440553 R    MAINK   ISZ     TABC    /INCREMENT TAB COUNTER
00227 R 600232 R            JMP     MAINE   /NOT OVERFLOW, GO CHECK LINE COUNTER
00230 R 777770 A            LAW     -10     /RESET TAB COUNTER
00231 R 040553 R            DAC     TABC
00232 R 440546 R    MAINE   ISZ     MAXC    /HAVE WE RUN OUT OF LINE
00233 R 600166 R            JMP     MAIN    /NO
00234 R 600302 R            JMP     UCLP03  /YES, GO FINISH UP, WITH END OF LINE
                    /
                    /  SPECIAL CHARACTERS
                    /
00235 R 750201 A    MSPEC   SZA!CLA!CMA     /SKIP IF IT IS A BLANK
00236 R 600242 R            JMP     MSPEC2  /NOPE, CHECK FOR OTHER THINGS
00237 R 340552 R            TAD     BLANKC  /ADD ONE TO BLANK COUNTER (IS MINUS COUNTER)
00240 R 040552 R            DAC     BLANKC
00241 R 600226 R            JMP     MAINK   /JOIN LINE AND TAB CONTROL SECTION
00242 R 200550 R    MSPEC2  LAC     TCHAR   /GET BACK ORIGINAL CHAR
00243 R 540644 R            SAD     (11     /IS IT A TAB
00244 R 600266 R            JMP     MTAB    /YUP, GO DO IT
00245 R 540645 R            SAD     (15     /CARRIAGE RETURN
00246 R 600302 R            JMP     UCLP03  /END OF LINE ON CARRIAGE RETURN
00247 R 540646 R            SAD     (20     /FORTRAN OTS OVERPRINT, DO AS CR
00250 R 600263 R            JMP     MCR
00251 R 540647 R            SAD     (14     /FORM FEED
00252 R 600256 R            JMP     MSPEC3  /JUST PUT IT OUT, FOR NOW
00253 R 540650 R            SAD     (21     /FORTRAN DOUBLE SPACE
00254 R 600260 R            JMP     MSPEC4  /DO AS TWO 12'S
```

Figure 4-1
PDP-15 LP11 DOS Handler (cont.)

```
00255 R 200642 R    MSPEC5  LAC     (12      /DEFAULT ON UNRECOGNIZED CONTROL CHAR. IS LINE FEED
00256 R 100366 R    MSPEC3  JMS     PUTCH    /PLACE IN BUFFER
00257 R 600166 R            JMP     MAIN     /GO DO NEXT
00260 R 200642 R    MSPEC4  LAC     (12      /FIRST OF TWO 12'S FOR THE 21
00261 R 100366 R            JMS     PUTCH
00262 R 600255 R            JMP     MSPEC5   /GO DO THE SECOND 112
00263 R 100443 R    MCR     JMS     RESETL   /NEW LINE, RESET VARIOUS GUYS
00264 R 200645 R            LAC     (15      /CARRIAGE RETURN
00265 R 600256 R            JMP     MSPEC3   /PUT CHAR AND LOOP
00266 R 200553 R    MTAB    LAC     TABC     /GET REMAINING COUNT FOR TAB
00267 R 340552 R            TAD     BLANKC   /AND ADD TO CUMULATIVE BLANK COUNT
00270 R 040552 R            DAC     BLANKC
00271 R 200553 R            LAC     TABC     /AND TO LINE CHECKER
00272 R 740031 A            CMA!IAC
00273 R 340546 R            TAD     MAXC
00274 R 040546 R            DAC     MAXC
00275 R 740100 A            SMA              /SKIP IF SOME LINE LEFT
00276 R 600302 R            JMP     UCLP03   /NONE LEFT, FINISH UP LINE
00277 R 777770 A            LAW     -10
00300 R 040553 R            DAC     TABC     /RESET TAB COUNTER
00301 R 600166 R            JMP     MAIN     /NEXT CHAR
                    /
00302 R 200645 R    UCLP03  LAC     (15      /CARRIAGE RETURN
00303 R 400554 R            XCT     MIX      /PLACE IN BUFFER ONLY ON IMAGE!!!
00304 R 100366 R            JMS     PUTCH
00305 R 100443 R            JMS     RESETL
00306 R 440551 R    UCLP04  ISZ     COP      /A BLANK LINE IS STILL A REAL CHAR SINCE FF
00307 R 220540 R            LAC*    LPBUF    /ZERO CHAR COUNT??
00310 R 500651 R            AND     (377     /COUNT ONLY IN LOW 8 BITS
00311 R 740200 A            SZA              /SKIP IF ZERO COUNT
00312 R 600316 R            JMP     UCLP05   /NON-ZERO, JUST GO DO REGULAR
00313 R 400554 R            XCT     MIX      /IMAGE OR IOPS
00314 R 600125 R            JMP     LPNEXT   /IMAGE DO NOTHING
00315 R 460540 R            ISZ*    LPBUF    /IOPS MAKE FAKE 1 COUNT
                    /                        /WE ARE DOING A BLANK LINE, AND 0
                    /                        /COUNT MAKES SPOOLER VERY ILL
00316 R 100517 R    UCLP05  JMS     LPSET    /SEND BUFFER TO PDP-11
00317 R 600125 R            JMP     LPNEXT   /CAL EXIT
                    /
                    /       CHARACTER UNPACKING ROUTINE
                    /
                    /
                    /  THIS ROUTINE 'OWNS' THE MQ
                    /
                    /
                    /  CHARACTERS ARE OBTAINED FROM X12 POINTER. EACH CHAR
                    /  IS RETURNED RIGHT JUSTIFIED IN THE AC
                    /  TEMP1 HAS A MINUS COUNT OF THE WORDS TO BE OBTAINED
                    /  FROM THE INPUT POINTER X12
                    /
00320 R 000000 A    GETCH   0
00321 R 400554 R            XCT     MIX      /SKIP IF IT IS ASCII
00322 R 741000 A            SKP
00323 R 620332 R            JMP*    GETSW    /GETSW IS POINTER TO CORRECT ACTION ON ONTHE
                    /                        /CORRECT ONE OF THE FIVE POSSIBLE CHAR'S
                    /
                    /  NOW DO IMAGE MODE
                    /
00324 R 440543 R            ISZ     TEMP1
00325 R 741000 A            SKP              /SKP ON NOT THRU YET
00326 R 600306 R            JMP     UCLP04   /DONE
00327 R 220557 R            LAC*    X12
00330 R 440557 R            ISZ     X12
00331 R 600333 R            JMP     GETCM    /FINISH UP IN COMMON
                    /
00332 R 000000 A    GETSW   0                /POINTER TO CORRECT ACTION. INIT'ED FROM GETIN
                    /                        /FILLED BY JMS GETSW AFTER EACH CHAR
00333 R 500640 R    GETCM   AND     (177     /COMMON FINISH UP, STRIP XTRA BITS
00334 R 620320 R            JMP*    GETCH    /OUT
                    /
00335 R 000337 R    GETIN   GET1             /INIT GETSW TO POINT TO FIRST CHAR ACTION
                    /
                    /  INDIVIDUAL CHARACTER ACTION
                    /
00336 R 100332 R    GETQ    JMS     GETSW    /AFTER 5TH CHAR, POINT BACK TO FIRST
                    /
00337 R 440543 R    GET1    ISZ     TEMP1    /OUT OF PAIRS?
00340 R 600343 R            JMP     .+3      /CONTINUE IF OK
00341 R 100443 R            JMS     RESETL   /END OF LINE RESET SONE STUFF
00342 R 600306 R            JMP     UCLP04
00343 R 220557 R            LAC*    X12      /FIRST WORD OF PAIR
00344 R 440557 R            ISZ     X12
00345 R 652000 A            LMQ              /INTO MQ FOR SHIFTING
00346 R 640607 A            LLS     7
00347 R 100332 R            JMS     GETSW    /DONE, LEAVE POINTER FOR SECOND CHAR
00350 R 640607 A    GET2    LLS     7        /SECOND CHAR
00351 R 100332 R            JMS     GETSW    /LEAVING POINTER FOR THIRD
00352 R 640604 A    GET3    LLS     4        /THE HALF-AND-HALF CHAR
00353 R 040332 R            DAC     GETSW    /VERY TEMPORARY
00354 R 220557 R            LAC*    X12      /CAN'T END IN MIDDLE OF PAIR
00355 R 440557 R            ISZ     X12
00356 R 652000 A            LMQ              /SECOND WORD TO SHIFTER
00357 R 200332 R            LAC     GETSW    /BRING BACK FIRST
00360 R 640603 A            LLS     3        /COMPLETE CHAR
00361 R 100332 R            JMS     GETSW    /LEAVING POINTER TO FOURTH ACTION
00362 R 640607 A    GET4    LLS     7
```

Figure 4-1
PDP-15 LP11 DOS Handler (cont.)

```
00363 R 100332 R            JMS     GETSW   /LEAVING FOR 5
00364 R 640607 A    GET5    LLS     7
00365 R 600336 R            JMP     GETQ    /BACK TO TOP FOR POINTER TO 1
                    /
                    /
                    /
                    /  CHARACTER PUTTER FOR PDP-11
                    /
                    /  TWO CHAR'S PER WORD FORMAT. FIRST CHAR IS RIGHT JUSTIFIED, SECOND
                    /  IS PLACED IMMEDIATELY ABOVE FIRST, LEAVING TOP TWO BITS OF WORD
                    /  UNUSED. CHAR IS DELEVERD TO US IN AC. INIT PUTSW BY DAC'ING CONTENTS
                    /  OF PUTIN INTO IT. ROUTINE COUNTS THE OUTPUT CHARS IN LBF
                    /
                    /  THIS ROUTINE ALSO HANDLES FORM FEED PAGE CONTROL
                    /  THE PDP-11 ASSUMES LINES HAVE A LF IN BEGINNING AND CR AT END
                    /  SO THIS ROUTINE REMOVES ANY LEADING LF.
                    /
                    /
00366 R 000000 A    PUTCH   0
00367 R 500651 R            AND     (377    /STRIP TO EIGHT BITS
00370 R 540642 R            SAD     (12     /SPECIAL CASE #1, LINE FEED
00371 R 600400 R            JMP     PUTLF   /GO DO IT
00372 R 540647 R            SAD     (14     /SPECIAL CASE #2, FORM FEED
00373 R 600415 R            JMP     PUTFF   /GO DO IT
00374 R 440547 R    PUTY    ISZ     FIRST   /BUMP FIRST TIME THRU SWTICH
00375 R 740000 A            NOP             /IN CASE SKIPS, WE DON'T NEED IT HERE
00376 R 460540 R    PUTZ    ISZ*    LPBUF   /COUNT AN OUTPUT CHAR
00377 R 620427 R            JMP*    PUTSW   /DISPATCH TO FIRST OR SECOND CHAR ACTION
                    /
00400 R 200551 R    PUTLF   LAC     COP     /HAS A REAL CHAR OCCURRED SINCE FF?
00401 R 740200 A            SZA             /SKIP IF NO REAL CHAR
00402 R 600412 R            JMP     PUTW    /GO DO REGULAR
00403 R 220541 R            LAC*    LPBUFD  /IF WE ALREADY HAVE A FF
00404 R 540647 R            SAD     (14     /IN BUFFER OUT, DON'T NEED A CR
00405 R 620366 R            JMP*    PUTCH
00406 R 200645 R            LAC     (15     /LEAD WITH CR, SO PDP-11 DOESN'T PUT ON AUTOMATIC LF
00407 R 400554 R            XCT     MIX     /BUT DO NOTHING FOR IMAGE MODE
00410 R 620366 R            JMP*    PUTCH
00411 R 600374 R            JMP     PUTY    /GO REAJOIN
00412 R 200642 R    PUTW    LAC     (12     /GET BACK LINE FEED
00413 R 400534 R            XCT     FFSW    /ISZ OR NOP FOR COUNT OF FF PER PAGE
00414 R 600422 R            JMP     PUTLFR  /NO FORM FEED NOW
00415 R 200531 R    PUTFF   LAC     PAGSIZ  /FORM FEED, RESET PAGE COUNTER
00416 R 040532 R            DAC     PAGCNT
00417 R 140551 R            DZM     COP     /FLAG SAYING FF OCCURRED.
00420 R 200647 R            LAC     (14     /FORM FEED CODE
00421 R 600376 R            JMP     PUTZ    /GO COUNT CHAR, AND PLACE IT
00422 R 400554 R    PUTLFR  XCT     MIX     /SKIP ON IOPS ASCII
00423 R 600374 R            JMP     PUTY    /IMAGE, ACTUALLY PLACE LF
00424 R 440547 R            ISZ     FIRST   /ASCII, IS IT FIRST THRU?
00425 R 600376 R            JMP     PUTZ    /NOT FIRST, DO LF
00426 R 620366 R            JMP*    PUTCH   /FIRST TIME, JUST RETURN
00427 R 000000 A    PUTSW   0               /INIT'ED AS PUT1. FILLED LATER BY JMS PUTSW
00430 R 620366 R            JMP*    PUTCH   /DONE, RETURN
                    /
00431 R 000433 R    PUTIN   PUT1            /START AT FIRST CHAR
                    /
00432 R 100427 R    PUTQ    JMS     PUTSW   /LEAVE POINTER FOR FIRST AFTER SECOND
00433 R 060560 R    PUT1    DAC*    PUTP    /FIRST CHARACTER ACTION, PLACE RIGHT JUSTIFIED
00434 R 100427 R            JMS     PUTSW   /LEAVING POINTER FOR SECOND
                    /
00435 R 746030 A    PUT2    CLL!SWHA        /PUT CHAR IN RIGHT PLACE
00436 R 740020 A            RAR
00437 R 260560 R            XOR*    PUTP    /PUT HALVES TOGETHER
00440 R 060560 R            DAC*    PUTP    /BOTH IN BUFFER
00441 R 440560 R            ISZ     PUTP    /MOVE POINTER
00442 R 600432 R            JMP     PUTQ    /GO TELL PUTSW  THAT PUT1 IS NEXT
                    /
                    /  OUTINE TO RESET LINE AND TAB COUNTRS
                    /
00443 R 000000 A    RESETL  0
00444 R 777777 A            LAW     -1      /SET FIRST CHAR OF LINE REMEMBERER
00445 R 040547 R            DAC     FIRST
00446 R 777770 A            LAW     -10     /SET TAB COUNTR
00447 R 040553 R            DAC     TABC
00450 R 200545 R            LAC     LINLIM  /SET UP MAX PER LINE COUNTER
00451 R 040546 R            DAC     MAXC
00452 R 140552 R            DZM     BLANKC  /RESET SPACE AND TAB COUNTER
00453 R 620443 R            JMP*    RESETL
                    /
                            .TITLE .CLOSE FUNCTION
                    /
                    /
                    /.CLOSE
                    /
00454 R 100512 R    LPCLOS  JMS     LPIOCK          /CHECK I/O UNDERWAY.
00455 R 140551 R            DZM     COP     /SAY A FF OCCURRED
00456 R 440470 R            ISZ     LPCLSW          /777777 IN AC IF HAVEN'T BEEN THRU CLOSE CODE.
00457 R 600471 R            JMP     LPCLDN          /DONE.
00460 R 750030 A            CLA!IAC         /SPOOLER REQUIRES FF,CR AS CLOSE
00461 R 060540 R            DAC*    LPBUF   /JUST GIVE FF TO DRIVER, HOWEVER
00462 R 200652 R            LAC     (6414   /THIS IS FF,CR IN PDP-11
00463 R 060541 R            DAC*    LPBUFD  /FIRST DATA WORD POINTER
                    /                       /THIS MEANS ALWAYS A FF ON CLOSE!!!
00464 R 100517 R            JMS     LPSET   /SEND BUFFER TO PDP-11
```

Figure 4-1
PDP-15 LP11 DOS Handler (cont.)

```
00465 R 10[illegible]63 R          JMS     RESETL  /RESET THE WORLD
00466 R 703344 A    LPCALX  DBR
00467 R 620527 R            JMP*    LPCALP          /HANG ON CAL.
00470 R 777777 A    LPCLSW  777777                  /-1 = .CLOSE NOT DONE.
00471 R 777777 A    LPCLDN  LAW -1
00472 R 040470 R            DAC     LPCLSW          /INITIALIZE .CLOSE INDICATOR
00473 R 600125 R            JMP     LPNEXT          /EXIT.
                            .TITLE .WAIT FUNCTION
                    /
                    /.WAIT OR .WAITR
                    /
00474 R 220527 R    LPWAIT  LAC*    LPCALP
00475 R 500635 R            AND     (1000
00476 R 741200 A            SNA                     /BIT 8 = 1 FOR .WAITR
00477 R 600510 R            JMP     LPWAT1          /.WAIT - GO HANG ON CAL.
00500 R 200653 R            LAC     (700000         /LINK, ETC.
00501 R 500527 R            AND     LPCALP
00502 R 040527 R            DAC     LPCALP
00503 R 220530 R            LAC*    LPARGP          /15-BIT BUSY ADDRESS.
00504 R 500654 R            AND     (77777
00505 R 240527 R            XOR     LPCALP
00506 R 040527 R            DAC     LPCALP
00507 R 440530 R            IDX     LPARGP
00510 R 100512 R    LPWAT1  JMS     LPIOCK          /CHECK I/O UNDERWAY.
00511 R 600125 R            JMP     LPNEXT          /OK - RETURN.
                    /
                    /CHECK FOR I/O UNDERWAY
                    /
                    /LPUND 0 WHEN FREE, NON0 WHEN BUSY
                    /
00512 R 000000 A    LPIOCK  0
00513 R 200533 R            LAC     LPUND           /0 = NO ACTIVITY.
00514 R 741200 A            SNA
00515 R 620512 R            JMP*    LPIOCK          /NO I/O UNDERWAY.
00516 R 600466 R            JMP     LPCALX          /HANG ON CAL TIL NOT BUSY.
                    /
                    / SETUP AND OUTPUT TO PRINTER.
                    /
00517 R 000000 A    LPSET   0
00520 R 200537 R            LAC     LPTCB   /SEND TCB POINTER TO PDP-11
00521 R 160542 R            DZM*    LPEV    /CLEAR THE EVENT VARIABLE
00522 R 706001 A            SIOA                    /MAKE SURE ITS ABLE TO GET IT
00523 R 600522 R            JMP     .-1             /NOTE THAT THIS IS PROTECTED SINCE
                                                    /    THE LIOR WILL BE ISSUED DIRECTLY
                                                    /    AFTER THE SIOA (FREE INSTRUCTION).
00524 R 706006 A            LIOR
00525 R 040533 R            DAC LPUND               /SET I/O BUSY FLAG.
00526 R 620517 R            JMP* LPSET
                            .TITLE INITIALIZATION CODE AND TEMPORARIES
                    /
00527 R 000000 A    LPCALP  0               /POINTER TO CAL ADDR
00530 R 000000 A    LPARGP  0               /POINTER ARGUMENTS OF CAL
00531 R 777706 A    PAGSIZ  -FORMS          /ASSEMBLED LINES PER PAGE
00532 R 777706 A    PAGCNT  -FORMS          /COUNT THE LINES HERE
00533 R 777772 A    LPUND   -6              /0=FREE,+=BUSY,-=ERROR
                    /                       /COUNTS UP TO INITAL 0 BELOW
                    /
                            .IFUND NOFF
00534 R 440532 R    FFSW    ISZ     PAGCNT  /ACTION FOR FORMS CONTROL, NEMORY
00535 R 440532 R    FFFF    ISZ     PAGCNT  /FFSW LOADED INTO HERE
                            .ENDC
                            .IFDEF NOFF
                    FFSW    NOP             /ACTION FOR FORMS, MEMORY
                    FFFF    NOP             /FFSW LOADED INTO HERE
                            .ENDC
00536 R 200625 R    INIT    LAC     (NOP    /WRITE OVER JUMP TO HERE
00537 R 040003 R    LPTCB   DAC     NEW     /PREVENT RE-ENTRY
00540 R 220655 R    LPBUF   LAC*    (.SCOM+4  /GT PRINTER LINE WIDTH
00541 R 742020 A    LPBUFD  RTR
00542 R 740020 A    LPEV    RAR             /MOVE TO '6' POSITION
00543 R 500656 R    TEMP1   AND     (6      /STRIP GARBAGE, LITERAL 6
00544 R 741200 A    BUFSIZ  SNA
00545 R 340656 R    LINLIM  TAD     (6      /TREAT 0 (UNDEFINED) AS 132 COLUMN!?!
00546 R 340613 R    MAXC    TAD     LBFTP   /POINTER TO CONSTANTS
00547 R 040613 R    FIRST   DAC     LBFTP
00550 R 220613 R    TCHAR   LAC*    LBFTP   /LINE WIDTH
00551 R 040545 R    COP     DAC     LINLIM
00552 R 440613 R    BLANKC  ISZ     LBFTP
00553 R 220613 R    TABC    LAC*    LBFTP   /BUFFER SIZE
00554 R 040544 R    MIX     DAC     BUFSIZ
                    /
                    /  NOW SET UP POINTERS TO BUFFER AND TCB LOC'S
                    /
00555 R 220643 R    LPAC    LAC*    (.SCOM+100  /POINTER TO TABLE OF POINTERS
00556 R 740030 A    LPOUT   IAC             /OUR POINTER IN TABLE +1
00557 R 040543 R    X12     DAC     TEMP1
00560 R 220543 R    PUTP    LAC*    TEMP1   /POINTER TO TCB
00561 R 040537 R            DAC     LPTCB
00562 R 040543 R            DAC     TEMP1   /POINTER TO FILL LOCATIONS
00563 R 723002 A            AAC     2       /MAKE POINTER TO EVENT VARIABLE
00564 R 040542 R            DAC     LPEV
00565 R 723002 A            AAC     2       /MAKE POINTER TO TCB POINTER
00566 R 040553 R            DAC     TABC    /TO BUFFER ADDR
00567 R 723005 A            AAC     5       /MAKE POINTER TO FIRST DATA WORD
00570 R 040541 R            DAC     LPBUFD
                    /
                    /  MAKE TCB
                    /
00571 R 200657 R            LAC     (APISLT*400+APILVL      /BUILD THE API RETURN
00572 R 060543 R            DAC*    TEMP1   /STORE IN TCB
```

Figure 4-1
PDP-15 LP11 DOS Handler (cont.)

```
00573 R 440543 R           ISZ     TEMP1   /INCRMT. POINTER TO TCB
00574 R 200660 R           LAC     (DEVCOD /PIREX CODE FOR LP DRIVER
00575 R 060543 R           DAC*    TEMP1   /STORE IN TCB
00576 R 440543 R   MKTCB   ISZ     TEMP1   /ZERO THRU FIRST BUFFER LOC
00577 R 160543 R           DZM*    TEMP1
00600 R 440533 R           ISZ     LPUND
00601 R 600576 R           JMP     MKTCB   /DONE YET ? - IF NOT THEN LOOP
00602 R 200543 R           LAC     TEMP1   /THIS POINTS TO BUFFER
00603 R 060553 R           DAC*    TABC    /TO LOCATION IN TCB THAT NEEDS
00604 R 040540 R           DAC     LPBUF   /AND A POINTER FOR US
00605 R 100443 R           JMS     RESETL  /RESET LINE AND TAB COUNTRS
00606 R 000056 A           CAL     APISLT  /ISSUE SETUP CAL TO ESTABLISH INTERRUPTS
00607 R 000016 A           16
00610 R 706141 A           LSSF
00611 R 000026 R           LPINT
00612 R 600003 R           JMP     NEW     / DONE
                   /
                           .DEC
00613 R 000612 R   LBFTP   .-1             /POINTER TO SIZE TABLE
00614 R 777660 A           -80
00615 R 000044 A           36
00616 R 777610 A           -120
00617 R 000064 A           52
00620 R 777574 A           -132
00621 R 000070 A           56
        000000 A           .END
00622 R 017777 A *L
00623 R 600011 R *L
00624 R 600036 R *L
00625 R 740000 A *L
00626 R 000000 A *L
00627 R 700042 A *L
00630 R 177777 A *L
00631 R 177001 A *L
00632 R 600000 A *L
00633 R 000137 A *L
00634 R 004000 A *L
00635 R 001000 A *L
00636 R 741000 A *L
00637 R 000400 A *L
00640 R 000177 A *L
00641 R 000135 A *L
00642 R 000012 A *L
00643 R 000200 A *L
00644 R 000011 A *L
00645 R 000015 A *L
00646 R 000020 A *L
00647 R 000014 A *L
00650 R 000021 A *L
00651 R 000377 A *L
00652 R 006414 A *L
00653 R 700000 A *L
00654 R 077777 A *L
00655 R 000104 A *L
00656 R 000006 A *L
00657 R 027002 A *L
00660 R 000004 A *L
    SIZE=00661      NO ERROR LINES
```

Figure 4-1
PDP-15 LP11 DOS Handler (cont.)

APILVL The API level at which PIREX should interrupt the PDP-15; this is used in TCBs and in the definition of CAPI. APILVL should indicate API level 0, 1, 2, or 3.[1]

APISLT The API slot to which PIREX should issue interrupts; used in TCBs and in the CONNECT/DISCONNECT software directives.

DEVICE SKIP In this case LSSF, one of the four possible UC15 skips. This skip is determined by which API level is chosen.
SKIP = APILVL*20 + 706101
The skip is used in the standard setup interrupts CAL (Figure 4-1, lines 614-618)

SIOA Skip if PDP-11 can accept a TCBP mnemonic; (706001).

LIOR Issue TCBP mnemonic; (706006).

CAPI Clear interrupt flag mnemonic; set to APILVL * 20 + 706104, used in interrupt service routine.

DEVCOD The device code as defined in PIREX: used in TCBs
NOTE: The conditional use of the spooled bit (PDP-11 bit 7) (Figure 4-1, lines 71-76).

4.6.1.1 Initialization - The CAL entry of a DOS-15 handler must have a once only section of code that:

1. Sets up a pointer to one of the reserved TCB areas in the DOS-15 monitor. This is done by locating a pointer to the TCB area in the table pointed to by .SCOM + 100 (Figure 4-1, lines 586, 590)

2. Computes pointers to the various locations within this TCB area, such as the event variable (Figure 4-1, lines 591-597).

3. Constructs the constant fields within the TCB such as the API RETURN and device code (Figure 4-1, lines 601-609).

4. Sets up a pointer to the data area in the TCB, which will be used as a buffer (Figure 4-1, lines 610-612).

4.6.1.2 Request Transmission - When issuing requests to a task from a PDP-15 program, the requesting program (e.g., a PDP-15 I/O handler) issues the following sequence of instructions.

```
DZM EV       /CLEAR EV IN TCB

LAC (TCB     /ADDRESS OF TCB IN AC

SIOA         /MAKE SURE PDP-11 CAN ACCEPT REQUEST

JMP .-1      /WAIT FOR IT IF NOT
```

[1] Level 0 may be used, but is not recommended because it could hang the PDP-15 system if the interrupt occured at the wrong time.

```
        LIOR        /ISSUE REQUEST TO THE PDP-11.  THIS CAUSES A LEVEL
                    /7 INTERRUPT TO THE PDP-11 and CONTROL TRANSFERRED
                    /TO THE LEVEL 7 HANDLER IN PIREX.
```

The instruction sequence which issues requests to tasks from the PDP-15 should have an identical format as shown above. These five instructions are ordered in a way which:

1. Clears the event variable (EV) before issuing the request.

2. Allows an interruptible sequence while waiting for the PDP-11.

3. Allows a non-interruptible sequence once the SIOA instruction skips and the LIOR is issued.

This occurs because the PDP-15 always allows a non-interruptible instruction following an IOT (in this case the SIOA). The SIOA and JMP .-1 sequence is interruptible immediately following the execution of JMP .-1.

The LPSET routine is used by the line printer handler to perform the request transmission and thus send data to the line printer (or line printer spooler) task (see Figure 4-1, lines 541-550).

4.6.1.3 Interrupt Section — Result Reception - After receipt of a request to PIREX, the PDP-11 will use the contents of the TCB to schedule the referenced task.

Meanwhile, the requesting program can either:

1. Give up control and wait for an interrupt from the PDP-11 as in the DOS-15 line printer handler case or

2. Test the EV until it goes non-zero. i.e.,

   ```
   LAC EV

   SNA

   JMP .-2
   ```

   to determine completion of the request. The EV is automatically set to a non-zero value by the referenced task when the request has been completed.[1]

Interrupts generated by the PDP-11 for the PDP-15 are serviced by the PDP-15 in a fashion identical to regular PDP-15 interrupts. As in a non-API environment, a SAPI N (N = 0, 1, 2, or 3 depending on what API level would have been used if the PDP-15 had API) instruction tests for the flag associated with the request. In an API environment, the appropriate API trap address must be set up before the interrupt occurs. When program control is transferred to the interrupt service routine, a CAPI N instruction must be issued to clear the hardware flag associated with the request.

[1] When interrupt returns are used, the EV is set to non-zero just prior to the issuing of the interrupt.

After clearing this flag, the event variable should be tested to detect an error condition (negative event variable). See Figure 4-1, lines 124-128).

If an error has occurred, the event variable should be tested for a possible PIREX out-of-node condition (PIREX ran out of space to store the request). If the error was an out-of-node error CR (EV = 177001) a retry of the request should be attempted (See Figure 4-1, lines 144-149).

If the error was not an out-of-node error, an error message should be sent to the user. The error code should be composed of the event variable and a handler mnemonic such as LPU (Figure 4-1, lines 136-139, 160)

4.6.1.4 .READ and .WRITE Requests - Actual input and output is accomplished by using typical DOS-15 handler code with the following exceptions:

1. The TCB is used as the data buffer[1]

2. The actual I/O is done by calls to the TCB transmission routine. In the example this is a call to LPSET (Figure 4-1, line 348)

4.6.1.5 .CLOSE Function - If PIREX provides spooling services for the device, there is a need to inform the device's spooler module that the current job has completed so that the spooler is forced to process any existing partially-filled buffers. The writer must insure that both the DOS-15 handler and the PIREX spooler module agree upon a convention to indicate this end-of-file. In the example, a form feed carriage return (6414) acts as an end-of-file (Figure 4-1, lines 497-502).

4.6.2 PDP-11 Requesting Task

Tasks such as MAC11 may execute under control of the PIREX executive in a background mode. Considerations such as TCB structure and event variable checking are similar to those of the DOS-15 handler.

When the requesting program is a PDP-11 task, it must issue the initiate request macro (IREQ) in lieu of the 5 instruction sequence shown for the PDP-15. (See Section 4.6.2). If the task being requested has a higher priority than the current one issuing the request, it will execute immediately; otherwise, control will return to the first instruction following the IREQ macro. IREQ is defined as follows:

```
.MACRO IREQ TCBP

MOV TCBP,R5

MOV #100000,R4
```

[1] Depending on Driver task design the TCB need not be used as a data buffer for NPR devices.

```
        IOT

        .BYTE 2,0

        .ENDM
```

The #100000 in R4 is used by PIREX to identify a PDP-11 request. A TCBP is a TCB pointer.

4.6.3 UNICHANNEL Device Handlers for RSX-PLUS III

The following description of how to write a UNICHANNEL device handler for RSX PLUS III does not discuss those topics pertaining to all RSX I/O handlers, see the chapter on Advanced Task Construction in the RSX-PLUS III Operating System Reference Manual (DEC-15-IROMA-A-D).

4.6.3.1 Definition of Constants - Several constants are defined in a UNICHANNEL handler's source file before any executable code (see Figure 4-2, lines 66-79). These constants include:

APISLT The API slot to which PIREX issues interrupts; this is used in TCBs and the CONNECT/DISCONNECT software directives.

APILVL The API level at which PIREX interrupts the PDP-15; this is used in the TCB and in definition of CAPI. APILVL should indicate API level 1, 2, or 3.

DEVICE SKIP UNICHANNEL device skip equated to APILVL*20+706101.

SIOA Mnemonic for "skip of PDP-11 can accept a TCBP"; 706001.

LIOR Mnemonic for "Issue TCBP"; 706006.

CAPI Clear interrupt flag mnemonic; set this to APILVL *20+706104. It is used in the interrupt service routine.

DEVCOD The device code as defined in PIREX; this is used in TCBs.

4.6.3.2 Initialization - The handler initialization is located immediately following these definitions (see Figure 4-2, lines 262-320). During handler initialization, the PIREX device driver status must be cleared and the event variable checked to see if the driver is functioning (see Figure 4-2, lines 287-304). Since it is not obvious to RSX whether or not the driver is operational, a message should be printed before the handler exits if the driver is not running under PIREX.

```
                        .TITLE  CD.... CR15/UC15 CARD READER EDIT #020
                 /
                 /
                 /                      FIRST PRINTING, FEBRUARY 1974
                 /
                 / THE INFORMATTON IN THIS DOCUMENT IS SUBJECT TO
                 / CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED
                 / AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.
                 / DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPON-
                 / SIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS
                 / DOCUMENT.
                 /
                 / THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FUR-
                 / NISHED TO THE PURCHASER UNDER A LICENSE FOR USE ON
                 / A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH
                 / INCLUSION OF DIGITAL'S COPYRIGHT NOTICE) ONLY FOR
                 / USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PRO-
                 / VIDED IN WRITING BY DIGITAL.
                 /
                 / DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY
                 / FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIP-
                 / MENT THAT IS NOT SUPPLIED BY DIGITAL.
                 /
                 / COPYRIGHT (C) 1974, BY DIGITAL EQUIPMENT CORPORATION
                 /
                 /
                        .EJECT
                 /
                 /EDIT #020      2/2/74  SCR  CLEANUP
                 /EDIT #019      SCR     CR15 ERROR HANDLING; RRN SWITCH!
                 /EDIT #018      SCR     FIX COON HANDLING CR15 VERSION
                 /EDIT #017      SCR     CLEANUP, !BOTH! DEVICES
                 /EDIT #016      SCR     MORE UC15 CODE
                 /EDIT #015      SCR  START TO PUT IN UC15 CODE
                 /EDIT #013      1-18-72
                 /EDIT #14       6-26-73
                 /COPYRIGHT 1973, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
                 /C.W. KEMP ---- W.A. DESIMONE. ---- G. M. COLE
                 /
                 /CR15 CARD READER CONTROL HANDLER TASK.  THIS CONTROL WILL
                 /SUPPORT SORBAN AND DOCUMATION READERS.
                 / CR15 CODE IS OBTAINED  WITH NO ASSEMBLY PPARAMETERS
                 /
                 / TO OBTAIN UC15 CODE DEFINE UC15=0.
                 /   ADDITIONAL UC15 PARAMETERS:
                 /  DEFINE NOSPL=0 TO DISABLE SPOOLING FOR CARD READER. FOR INSTANCE
                 /  IF SPOOLER PACKAGE DOESN'T HAVE CARD READER ASSEMBLED IN FOR SPACE REASONS.
                 / AN EQUATE FOR APILVL IS NECESSARY TO SET UP
                 / IOT'S FOR CORRECT PRIORITY LEVEL TO CLEAR PIREX REQUEST.
                 / PRESENTLY LEVEL 1 IS THE CARD READER ASSIGNMENT.
                 /
                 /     W   A   R   N   I   N   G   !   !
                 /
                 /  IN ORDER FOR THE UC15 HANDLER TO FUNCTION PROPERLY, THE
                 /  PDP11 MUST BE ABLE TO ACCESS OUR INTERNAL BUFFER
                 /  AND TCB'S. THIS MEANS THAT THEIR ADDRESS MUST BE LESS THAN
                 /   28K TO THE PDP11. THUS, IF THE PDP-11 LOCAL MEMORY IS 8K,
                 /  THIS HANDLER MUST RESIDE BELOW 20K IN PDP15 CORE!! THIS
                 /  IS EQUIVALENT TO 50000 OCTAL. SIMILARLY , IF THE LOCAL
                 /  PDP-11 MEMORY IS 12K, THE HANDLER MUST RESIDE BELOW
                 /  40000 OCTAL.
                 /
                        .IFDEF  UC15
                 /
                 /
        000055 A APISLT=55
        000001 A APILVL=1
        706121 A CRSI=APILVL*20+706101
        706001 A SIOA=706001
        706006 A LIOR=706006
        706124 A CAPI=APILVL*20+706104
                 /
                        .IFUND NOSPL
        000005 A DEVCOD=5
                        .ENDC
                        .IFDEF NOSPL
                 DEVCOD=205
                        .ENDC
                        .ENDC
                 /
                 /EDIT 14 ADDS ASSEMBLY PARAMETER ERRLUN TO SPECIFY LOGICAL UNIT
                 /       FOR ALL ERROR MESSAGES. THE IS SET TO 3 IF USED INTERACTIVELY
                 /       MOST OF THE TIME OR TO 100 WHEN USED WITH PHASE
                 /       III BATCH. LUN 100 IS DEFINED TO BE THE BATCH OPERATOR DEVICE.
                 /
                        .IFUND  ERRLUN
                 ERRLUN=100
                        .ENDC
                 /THIS IS AN IOPS ASCII ONLY HANDLER TASK.
                 /IT CAN BE ASSEMBLED TO READ 029 OR 026 IBM KEYPUNCHED CARDS.
                 /DEFINE DEC026 TO READ 026 PUNCHED CARDS.
                 /DEC026 UNDEFINED TO READ 029 PUNCHED CARDS.
                 /
                 /
                 /
                 / THE FOLLOWING QUEUE I/O DIRECTIVES ARE IMPLEMENTED
                 /
                 /       CPB     3600    HANDLER INFORMATION (HINF)
```

Figure 4-2
PDP-15 CR11 RSX-PLUS III Handler

```
            /               EVA
            /               LUN
            /
            / FOR HINF THE FOLLOWING INFORMATION IS RETURNED IN THE EV
            /
            /       BIT  0          UNUSED
            /       BIT  1 = 1      INPUT DEVICE
            /       BIT  2 = 0      NOT OUTPUT DEVICE
            /       BIT  3 = 0      NOT FILE-ORIENTED
            /       BITS 4-11       UNIT NUMBER 'ZERO'
            /       BITS 12-17      DEVICE CODE = 7  CARD READER
            /
            /
            /       CPB     2400    ATTACH CARD READER
            /               EVA
            /               LUN
            /
            /       CPB     2500    DETACH CARD READER
            /               EVA
            /               LUN
            /
            /       CPB     2600    READ CARD
            /  (1)          EVA
            /  (2)  LUN
            /  (3)          MODE
            /  (4)          BUFF
            /  (5)          SIZE
            /
            /IF A REQUEST CANNOT BE QUEUED, THE FOLLOWING EVENT VARIABLE
            /VALUES ARE RETURNED:
            /
            /       -101 -- INDICATED LUN DOES NOT EXITS.
            /       -102 -- INDICATED LUN IS NOT ASSIGNED TO PHYSICAL DEVICE.
            /       -103 -- HANDLER TASK IS NOT CORE RESIDENT.
            /       -777 -- NODE FOR REQUEST QUEUE NOT AVAILABLE.
            /
            /
            /IF THE QUEUED I/O REQUEST CANNOT BE SUCCESSFULLY DEQUEUED,
            /THE FOLLOWING EVENT VARIABLE VALUES ARE RETURNED:
            /
            /       -7  -- ILLEGAL DATA MODE.
            /       -6  -- UNIMPLEMENTED FUNCTION.
            /       -24 -- LUN REASSIGNED WHILE ATTACH/DETACH REQUEST IN QUEUE.
            /       -30 -- OUT OF PARTITION TRANSFER (NORMAL MODE).
            /       -203 -- CAL NOT TASK ISSUED.
            /
            /
                    .EJECT
            /
            /   ***** CONSTANTS *****
            /
000012 A    X12=12          /AUTO-INDEXREG. 12
000013 A    X13=13          /AUTO-INDEXREG. 13
000101 A    R1=101          /RE-ENTRANT REG. 1
000102 A    R2=102          /RE-ENTRANT REG. 2
000103 A    R3=103          /RE-ENTRANT REG. 3
000104 A    R4=104          /RE-ENTRANT REG. 4
000107 A    NADD=107        /NODE ADDITION ROUTINE ENTRY POINT
000123 A    SNAM=123        /NAME SCAN ROUTINE ENTRY POINT
000240 A    POOL=240        /LISTHEAD FOR POOL OF EMPTY NODES
000252 A    PDVL=252        /LISTHEAD FOR PHYSICAL DEVICE LIST
000325 A    ALAD=325        /ATTACH LUN & DEVICE ENTRY POINT
000332 A    DLAD=332        /DETACH LUN & DEVICE ENTRY POINT
000337 A    DQRQ=337        /DE-QUEUE REQUEST ENTRY POINT
000342 A    VAJX=342        /VERIFY AND ADJUST I/O PARAMS.
000345 A    IOCD=345        /DECREMENT TRANSFERS PENDING COUNT.
000361 A    DMTQ=361        /DE-QUEUE I/O REQUEST (FOR ABORTING).
000010 A    D.TG=10         /POSITION OF TRIGER EVENT VARIABLE IN PDVL NODE
            /
                    .IFUND  UC15
            /
            CWC=22          /WC DCH ADDRESS.
            CCA=23          /CA DCH ADDRESS.
            /
            /PSUEDO-INSTR. FOR WF.SW SUBR.
            /
            WFOFF=SNA       /WAITFOR CR15 NOT READY.
            WFON=SZA        /WAITFOR CR15 READY.
            /
            /
            /CONDITIONS FOR LOAD READER CONDITION IOT (CRLC).
            /
            CC1=20          /CLEAR STATUS,DISABLE INTERRUPT AND DATA CHANNEL.
            CC2=27          /CLEAR STATUS,START READ,ENABLE INTERRUPT AND DATA CHANNEL.
            CC3=26          /CLEAR STATUS,ENABLE INTERRUPT,ENABLE DATA CHANNEL.
            CC4=04          /ENABLE INTERRS. DISABLES DCH
            /
            / *****  IOT INSTRUCTIONS *****
            /
            CRPC=706724             /CLEAR STATUS EXCEPT CARD DONE.(ALSO DISABLES INTERR.)
            CRLC=706704             /LOAD READER CONDITIONS.
            CRRS=706732             /READ STATUS INTO AC.
            /
                    .ENDC
            /
705522 A    .INH=705522             /INHIBIT INTERRUPTS.
705521 A    .ENB=705521             /ENABLE INTERRUPTS.
            /
                    .EJECT
```

Figure 4-2
PDP-15 CR11 RSX-PLUS III Handler (cont.)

```
                          /----CR15 STATUS AND AC BIT ASSIGNMENTS.
                          /
                          /STATUS REGISTER BIT ASSIGNMENTS:
                          /
                          /       BIT       TRANSLATION
                          /
                          /       17        COLUMN READY
                          /       16        END OF CARD
                          /       15        DATA CHANNEL OVERFLOW
                          /       14        DATA CHANNEL ENABLED
                          /       13        READY TO READ
                          /       12        ON LINE
                          /       11        END OF FILE
                          /       1Ø        BUSY
                          /       Ø9        TROUBLE (= IOR OF BITS 4 - 8)
                          /       Ø8        DATA MISSED
                          /       Ø7        HOOPER EMPTY/STACKER FULL
                          /       Ø6        PICK ERROR
                          /       Ø5        MOTION ERROR
                          /       Ø4        PHOTO ERROR
                          /       Ø3-ØØ     UNUSED
                          /
                          /AC BIT ASSIGNMENTS FOR LOAD CONDITION FUNCTION (CRLC)
                          /
                          /       BIT       FUNCTION
                          /
                          /       17        START READ
                          /       16        DATA CHANNEL ENABLE
                          /       15        INTERRUPT ENABLE
                          /       14        OFFSET CARD
                          /       13        CLEAR STATUS REGISTER
                          /
                          /       STATUS REGISTER BITS CONNECTED TO FLAG AND INTERRUPT REQUEST:
                          /
                          /       17        DATA READY(ONLY IF DATA CHANNEL NOT ENABLED)
                          /       16        CARD DONE
                          /       15        DATA CHANNEL OVERFLOW
                          /       Ø9        ERROR CONDITION
                          /
                          /MACRO DEFINITIONS:
                          /
                          /CP MACRO FOR CARD COLUMN TO ASCII TRANSLATION TABLE Ø26/Ø29 CONDITIONALIZATION
                          /
                                  .IFDEF  DECØ26
                                  .DEFIN  CP,C26,C29
                                  C26&777777+1
                                  .ENDM
                                  .ENDC
                                  .IFUND  DECØ26
                                  .DEFIN  CP,C26,C29
                                  C29&777777+1
                                  .ENDM
                                  .ENDC
                          /
                          /
                                  .EJECT
                          /
                          /
                          / ***** HANDLER INITIALIZATION ***** (ONCE ONLY CODE)
                          /
                          /START                          /STORAGE FOR AC IN INTERR. SERVICE.
                          /IBUF                           /TOP OF INTERNAL BUFFER.
                          /
                          /
ØØØØØ R 2ØØ646 R          START   LAC     (PDVL)          /SCAN PDVL FOR THIS DEVICE'S NODE
ØØØØ1 R Ø6Ø647 R          IBUF    DAC*    (R1)
ØØØØ2 R 2ØØ65Ø R                  LAC     (HNAM)
ØØØØ3 R Ø6Ø651 R                  DAC*    (R2)
ØØØØ4 R 12Ø652 R                  JMS*    (SNAM)          /R, R2, R6, XR, & AC ARE ALTERED
                                                          /NODE FOUND?
ØØØØ5 R ØØØ653 R                  CAL     (1Ø)            /NO -- EXIT
ØØØØ6 R Ø4Ø567 R                  DAC     PDVNA           /YES -- PDVL NODE ADDRESS IN AC.
ØØØØ7 R 723Ø1Ø A                  AAC     D.TG            /SAVE NODE ADDRESS AND
ØØØ1Ø R Ø4Ø57Ø R                  DAC     PDVTA           /TRIGGER EVENT VARIABLE ADDRESS
ØØØ11 R ØØØ577 R                  CAL     CCPB            /CONNECT INTERRUPT LINE
ØØØ12 R 2ØØ561 R                  LAC     EV              /CONNECT OK?
ØØØ13 R 7411ØØ A                  SPA
ØØØ14 R ØØØ653 R                  CAL     (1Ø)            /NO -- EXIT
ØØØ15 R 2ØØ654 R                  LAC     (TG)            /YES -- SET TEV ADDRESS
ØØØ16 R Ø6Ø57Ø R                  DAC*    PDVTA
ØØØ17 R 5ØØ655 R                  AND     (7ØØØØ)         /DETERMINE 'XR-ADJ'
ØØØ2Ø R 74ØØ31 A                  TCA
ØØØ21 R Ø4Ø563 R                  DAC     XADJ
                          /
                                  .IFUND  UC15
                                  LAC     (CC1)           /CLEAR STATUS, DISABLE INTER, AND DCH.
                                  CRLC                    /LOAD FUNCTION.
                                  .ENDC
                                  .IFDEF  UC15
ØØØ22 R 1ØØ625 R                  JMS     CLEAR   /CLEAR OUT PIREX DEVICE, WAIT FOR COMPLETE
ØØØ23 R 2ØØ613 R                  LAC     EV11K   /FIND OUT IF OK
ØØØ24 R 742Ø1Ø A                  RTL             /PDP11 SIGN BIT TO OURS
ØØØ25 R 741ØØØ A                  SMA             /SKIP IF TROUBLE
ØØØ26 R 6ØØØ57 R                  JMP     WFTGR   /NOT, GO WAIT FOR WORK
ØØØ27 R ØØØØ34 R                  CAL     MSINIT  /PRINT PIREX HAS NO CD MESSAGE
ØØØ3Ø R ØØØØ32 R                  CAL     WFMS    /WAIT FOR MESSAGE COMPLETION
ØØØ31 R ØØØ653 R                  CAL     (1Ø     /EXIT
                          /
ØØØ32 R ØØØØ2Ø A          WFMS    2Ø
ØØØ33 R ØØØ561 R                  EV
```

Figure 4-2
PDP-15 CR11 RSX-PLUS III Handler (cont.)

```
00034 R 002700 A    MSINIT  2700
00035 R 000561 R            EV
00036 R 000100 A            ERRLUN
00037 R 000002 A            2
00040 R 000041 R            INITMS
00041 R 004002 A    INITMS  004002; 000001 .ASCII "*** NO CD IN PIREX"<15>
00042 R 000000 A
00043 R 251245 A
00044 R 220234 A
00045 R 475010 A
00046 R 342100 A
00047 R 446344 A
00050 R 250222 A
00051 R 512133 A
00052 R 006400 A
                            .ENDC
00053 R 600057 R            JMP     WFTGR           /WAIT FOR TRIGGER
                    /
00054 R 030400 A    HNAM    .SIXBT  'CD@@@@'        /HANDLER TASK NAME
00055 R 000000 A
                    /
                            .IFUND  UC15
                    /
                            .BLOCK  121+START-.
                    /
                            .ENDC
                    /
                            .IFDEF  UC15
                    /
00056 R 777775 A            .BLOCK  53+START-.
                    /
                            .ENDC
                    / ***** END OF INITIALIZATION CODE *****
                    /
                    /******** THE ABOVE CODE IS OVERLAYED BY THE INTERNAL BUFFER ******
                    /*****************************************************************
                    /
                    /  UC15  INTERRUPT-CAL INTERACTION WILL BE DIFFERENT
                    /    KEEP INITIAL PART SEPARATE
                    /
                            .IFUND  UC15
                    /
                    WFTGR   CAL     WFTCPB          /WAIT FOR TEV TO BE SET
                    /
                    / ***** THE TASK HAS BEEN TRIGGERED -- PICK A REQUEST FROM QUEUE
                    /
                            DZM     TG              /CLEAR TRIGGER
                    PQ      LAC     PDVNA   /DEQUE A REQUEST
                            DAC*    (R1)
                            JMS*    (DQRQ)          /R1, R2, R4, R5, R6, XR & AC ARE ALTERED
                                                    /WAS A REQUEST FOUND?
                            JMP     WFTGR           /NO -- WAIT FOR TRIGGER
                    /
                            .ENDC
                    /
                            .IFDEF  UC15
                    /  UC15 CODE
                    /
                    /    THE GENERAL IDEA IS THAT ALL WAITS ARE DONE THRU
                    /  THE TRIGGER, WE FIGURE OUT HERE WHO SET THE TRIGGER, THIS
                    /  ALLOWS US TO GET OUT OF HUNG DEVICE, SINCE WE WAIT HERE,
                    /  AND CAN SEE AN ABORT COMING THRU.
                    /
00057 R 000575 R    WFTGR   CAL     WFTCPB  /WAIT FOR EVENT VARIABLE TG
00060 R 200562 R    PQ      LAC     TG      /FIND OUT WHO IS CALLING
00061 R 140562 R            DZM     TG      /RESET
00062 R 742010 A            RTL             /ABORT BIT TO SIGN BIT
00063 R 751130 A            SPA!CLA!IAC     /SKIP IF NOT ABORT, 1 IN AC.
00064 R 600071 R            JMP     PQ1     /GO DO ABORT IN REGULAR WAY. THE HANGING
                    /                       /READ IS REMEMBERED IN RRN!
00065 R 540554 R            SAD     CDON    /HAS A CARD BEEN DECLARED DONE BY INTERRUPT
00066 R 600177 R            JMP     GOTCRD  /YEAH, GO TRANSLATE IT
00067 R 540407 R            SAD     POST    /ARE WE WAITING FOR INTERRUPT
00070 R 600057 R            JMP     WFTGR   /YES, AND IT HASN'T HAPPENED YET, SINCE
                    /                       /CDON NOT SET. WAIT ON THIS CAL REQ, TO BE
                    /                       /DONE AFTER THE INTERRUPT HAPPENS. IF ABORT
                    /                       /COMES IN THE MEANTIME, HE IS PUT AT HEAD
                    /                       /OF DEQUE OF WAITING REQ.'S SO WE DO HIM.
                    /
00071 R 200567 R    PQ1     LAC     PDVNA   /TRY TO DEQUE AFTER OPERATION BEFORE WAITING
00072 R 060047 R            DAC*    (R1     /IN CASE WAITING FOR INTERRUPT HAS HELD OFF
00073 R 120656 R            JMS*    (DQRQ   /A REQUEST.
00074 R 600057 R            JMP     WFTGR   /DIDN'T FIND ONE, GO WAIT
                    /
                            .ENDC
                    /
00075 R 040564 R            DAC     RN              /YES -- SAVE ADDRESS OF REQUEST NODE
00076 R 340563 R            TAD     XADJ            /SETUP XR TO ACCESS NODE
00077 R 721000 A            PAX
                    /
                    / ***** I/O REQUEST NODE FORMAT *****
                    /
                    /     (0) FORWARD LINK
                    /     (1) BACKWARD LINK
                    /     (2) STL PTR.
                    /     (3) PART. BLK PTR. (0 IF EXM TSK).
                    /     (4) TASK PRIORITY
                    /     (5) I/O FCN CODE IN BITS 9-17 AND LUN IN BITS 0-8
                    /     (5)  -- EVENT VARIABLE ADDRESS
                    /     (7) CTB PTR.
                    /     (10) EXTRA
```

Figure 4-2
PDP-15 CR11 RSX-PLUS III Handler (cont.)

```
                    /     (11) EXTRA
                    /
00100 R 210005 A            LAC     5,X             /FETCH I/O FCN CODF
00101 R 500657 R            AND     (777)
00102 R 540660 R            SAD     (024)           /ATTACH REQUEST?
00103 R 600120 R            JMP     ATTACH          /YES -- ATTACH TO TASK
00104 R 540661 R            SAD     (025)           /NO -- DETACH REQUEST?
00105 R 600127 R            JMP     DETACH          /YES -- DETACH FROM TASK
00106 R 540662 R            SAD     (026)           /NO -- READ REQUST?
00107 R 600140 R            JMP     READ            /YES -- READ CARD
00110 R 540663 R            SAD     (036)           /NO -- HANDLER INFO.?
00111 R 600136 R            JMP     HINF            /YES -- RETURN INFO IN EV
00112 R 540657 R            SAD     (777)           /NO -- EXIT (DEASSIGNED) REQUEST?
00113 R 600464 R            JMP     DAEX            /YES -- DEATTACH & EXIT
00114 R 540664 R            SAD     (017)           /ABORT REQUEST?
00115 R 600502 R            JMP     CDABRT          /YES.
00116 R 777772 A    EVM6    LAW     -6              /NO -- UNIMPLEMENTED FUNCTION -- SET
00117 R 600424 R            JMP     SEV             /EVENT VARIABLE TO -6
                    /
                    / ATTACH TO A TASK
                    /
00120 R 200567 R    ATTACH  LAC     PDVNA           /ATTACH LUN & DEVICE
00121 R 060647 R            DAC*    (R1)
00122 R 200564 R            LAC     RN
00123 R 060651 R            DAC*    (R2)
00124 R 120665 R            JMS*    (ALAD)          /R3, R4, R5, R6, X10, X11, XR & AC ARE ALTERED
                                                    /WAS LUN ATTACHED?
00125 R 600424 R            JMP     SEV             /NO -- SET REQUESTOR'S EV TO -24
00126 R 600423 R            JMP     REQCMP          /YES REQUEST COMPLETED
                    /
                    / DETACH FROM TASK
                    /
00127 R 200567 R    DETACH  LAC     PDVNA           /DETACH LUN & DEVICE
00130 R 060647 R            DAC*    (R1)
00131 R 200564 R            LAC     RN
00132 R 060651 R            DAC*    (R2)
00133 R 120666 R            JMS*    (DLAD)          /R3, R4, R5, R6, X10, X11, XR & AC ARE ALTERED
                                                    /WAS LUN ATTACHED
00134 R 600424 R            JMP     SEV             /NO -- SET REQUESTOR'S EV TO -24
00135 R 600423 R            JMP     REQCMP          /YES -- REQUEST COMPLETED
                    /
                            .EJECT
                    /
                    / RETURN HANDLER INFORMATION
                    /
00136 R 200667 R    HINF    LAC     (200007)
00137 R 600424 R            JMP     SEV
                    /
                    /READ CARD
                    /
00140 R 777776 A    READ    LAW     -2              /CHK. FOR IOPS ASCII DATA MODE.
00141 R 350007 A            TAD     7,X
00142 R 740200 A            SZA                     /IOPS ASCII?
00143 R 600460 R            JMP     EVM7            /NO, RETURN -5 EV.
00144 R 210002 A            LAC     2,X             /SAVE STL NODE PTR. FOR TASK IDENTIF.
00145 R 040556 R            DAC     STLA            /SAVE VALID STL PTR.
00146 R 210010 A            LAC     10,X            /YES.  VAL/ADJ. HEADER ADDRESS
00147 R 060670 R            DAC*    (R3)            /HEADER ADDRESS.
00150 R 210011 A            LAC     11,X            /WORD COUNT
00151 R 060671 R            DAC*    (R4)
00152 R 740031 A            TCA                     /SETUP COUNTER  SINCE
00153 R 723002 A            AAC     +2              /OFFSET FOR CR APPENDAGE.
00154 R 040566 R            DAC     COWDCT          /VAJX ALTERS THE XR.
00155 R 040574 R            DAC     TCWC            /SAVE IN CASE RETRY.
00156 R 200564 R            LAC     RN              /REQ. NODE ADDRESS.
00157 R 040571 R            DAC     RRN             /SAVE READ REQ. NODE ADDR. FOR ABORT.
00160 R 060651 R            DAC*    (R2)
00161 R 120672 R            JMS*    (VAJX)          /VAL/ADJ. (ALTERS XR,AC,R3,R5)
00162 R 600462 R            JMP     EVM30           /RETS. HERE IF ERROR (I/O PARAM. OUT
                                                    /OF PARTITION.
00163 R 220670 R            LAC*    (R3)            /ADJUSTED HEADER ADDRESS -1 TO X12 TEMP.
00164 R 723777 A            AAC     -1
00165 R 040572 R            DAC     TX12
00166 R 723002 A            AAC     +2              /TEXT ADDRESS-1 TO X13 TEMP.
00167 R 040573 R            DAC     TX13            /
00170 R 140565 R            DZM     CDRVAL          /INIT. VALID. BITS.
                            .IFUND  UC15
                            LAC     CDON            /HAS CARD DONE FLAG COME UP SINCE
                            SNA                     /LAST CARD READ?
                            CAL     WFCRCD          /NO. WAITFOR CARD DONE.
                            DZM     CDON            /YES. CLEAR CARD DONE FLAG.
                    RETRY   LAC     (IBUF-1)        /SET INTERN. BUFF ADDR-1 TO DCH CA.
                            DAC*    (CCA)
                            DZM*    (CWC)   /PREVENTS DOUBLE INTERRUPTS ON ERRORS!!!!
                            LAC     TCWC            /RESTORE REQ. WC.
                            DAC     COWDCT
                            DZM     EV1             /REINIT EV.  RETRY FROM ERROR.
                            CRRS                    /READ STATUS IN ORDER TO CHECK FOR READER READY
                            AND     (60)            /AND ON-LINE.
                            SAD     (60)            /STATUS BITS 12, 13 SET?
                            SKP                     /YES, ON-LINE AND READY FOR READ.
                            JMP     ERR1            /NO, NOT READY.  TYPE MSG1 AND WAIT FOR READY.
                            LAC     (CC2)           /CONDITION CODE 2 -- READ CARD.
                            CRLC                    /LOAD CONDITIONS.
                            CAL     WFCRCB          /WAIT FOR INTERRUPT.
                    /
                    /
                    /
                    /UPON RESUMPTION FOLLOWING WAITFOR, EXAMINE EV AND TAKE THE FOLLOWING
                    /ACTION:
```

Figure 4-2
PDP-15 CR11 RSX-PLUS III Handler (cont.)

```
/
/IF EV BIT 9 = 0 (TROUBLE BIT), NO ERRORS.  TRANSLATE CARD PUNCHES
/TO ASCII AND PASS TO USER AS 5/7 PACKED ASCII.
/IF BIT 9 = 1 (TROUBLE BIT), ERROR BITS 08 TO 04 ARE CHECKED IN
/DESCENDING NUMERICAL ORDER.  THE FOLLOWING ERROR MESSAGES FOR THE
/GIVEN ERROR CONDITIONS ARE OUTPUT:
/
/DATA MISSED OR PHOTO ERROR - '*** CD DATA MISSED/PHOTO ERROR'
/PICK OR MOTION ERROR - '*** CD PICK ERROR'
/HOPPER EMPTY OR STACKER FULL - IGNORED.  CAUGHT ON SUBSEQ.
/READ AS A READER NOT READY CONDITION.
/IN ALL CASES WHERE A MESSAGE IS TYPED,  THIS HANDLER TASK MARKS TIME
/UNTIL THE ERROR IS REMEDIED.  AT THIS POINT, THE CARD IS REREAD.
/
        LAC     EV1             /EV SET AT INTERR. LEVEL TO CONTENTS OF
        DAC     TST             /STATUS.  SAVE TEMP.
        SWHA                    /SWAP HALVES FOR TROUBLE BIT CHECK.
        SMA!RAR                 /IF NEG.,TROUBLE.
        JMP     TRANS           /NO TROUBLE.  GO TRANSLATE.
        SZL!RAR                 /DATA MISSED?
        JMP     ERR4            /YES.
        SZL!RAR                 /NO.  HOPPER EMPTY/STACK. FULL?
        JMP     TRANS           /YES. IGNORE.  WHEN NEXT CRD. READ CAUGHT AS NOT READY.
        SZL!RAR                 /PICK ERROR?
        JMP     ERR3            /YES.
        SZL!RAR                 /MOTION ERROR?
        JMP     ERR3            /YES.
        JMP     ERR4            /NO.  MUST BE PHOTO ERROR.
/
/
ERR4    ISZ     ERRPT
ERR3    ISZ     ERRPT
ERR2    ISZ     ERRPT
ERR1    LAC*    ERRPT           /ERRMSG. BUFFER ADDR. TO AC.
        JMS     TTYOUT          /TYPE MESSAAE.
        JMS     WF.SW           /WAITFOR READER READY.
                WFON
        LAC     (ERRPT+1)       /REINIT. ERRPT.
        DAC     ERRPT
        JMP     RETRY           /READ ANOTHER CARD.
/
        .EJECT
TRANS   LAC     TX12            /SET AUTO INDEX REG.
        DAC*    (X12)
        LAC     TX13
        DAC*    (X13)
/
/  NOW BRING BACK RN FROM RRN, IN CASE RN DESTROYED IN MEANTIME
/
        LAC     RRN
        DAC     RN
        LAC     (IBUF)          /TOP OF INTERNAL BUFFER
        DAC     ICA             /PTR TO BUFFER
        LAW     -20
        DAC     CDCOLC          /CARD COL COUNT
CDRM5   LAW     -5
        DAC     CDR5CT
CDML2   LAC*    ICA             /GET
        SAD     CDRALT          /ALT MODE (12,1,8 PUNCH)?
        JMP     CDGALT          /YES -- TERMINATE BUFFER
        SAD     (7777           /NO -- IS IT AN EOF?
        JMP     EOF             /YES.
        LAC     CDTABL          /NO -- TRANSLATE TO ASCII
        DAC     CDTPTR          /GET TOP OF TABLE AND SET PTR
        LAC     CDTLN1          /SET TABLE LENGTH
CDML4   DAC     CDTLEN          /CURRENT LENGTH/2
        ADD     CDTPTR          /CURRENT TABLE TOP + LENGTH/2
        DAC     CDCPTR
        LAC*    CDCPTR          /GET CURRENT ITEM
        AND     (7777
        SZA!CLL
        ADD     CD7700          /ADD IN REST OF 2'S COMPLEMENT WORD
        TAD*    ICA             /CURRENT COLUMN
        SNA!CLA                 /MATCH FOUND?
        JMP     CDCFND          /YES
        SAD     CDTLEN          /CURRENT TABLE LENGTH =0?
                                /THIS MEANS AN UNKNOWN CARD PUNCH
        JMP     ILLCP           /GO OUTPUT 'ILLEGAL CARD PUNCH'.
        SNL                     /L=0 JUMP UP, L=1 JUMP DOWN TABLE
        JMP     CDDPTR
        LAC     CDCPTR          /SET TABLE TOP TO LOWER HALF
        DAC     CDTPTR
CDDPTR  LAC     CDTLEN          /UPDATE TABLE LENGTH
        CLL!RAR
        JMP     CDML4
CDGALT  LAW     4000            /ALT MODE
        JMP     CDCPUT
/
EOF     LAC     (1005
        JMP     REQCMA          /SET HDR WDI TO EOF
                                /REQUEST COMPLETE
/
/COME HERE ON MATCH FOUND
/
CDCFND  LAC*    CDCPTR          /GET CURRENT ENTRY
        CMA!CLL                 /GEN. LEFTMOST BIT
        TAD     CDTABL+1        /ADD 4000000
        CMA
```

Figure 4-2
PDP-15 CR11 RSX-PLUS III Handler (cont.)

```
                                XOR     CDTABL+1        /RESTORE SIXTH BIT
                                RAR
                        CDCPUT  DAC     CDRWD3          /PUT IN TOP OF 3 WORD SHIFT BLOCK
                        CDCLAW  LAW     -7
                                DAC     CDR7CT
                        CDCPL1  LAC     CDRWD3          /CDEWD3,CDRWD2 & CDRWD1 SHIFT AS A UNIT USING
                                                        /THE LINK TO PASS BITS FROM WORD TO WORD
                                RAL
                                DAC     CDRWD3
                                LAC     CDRWD2
                                RAL
                                DAC     CDRWD2
                                LAC     CDRWD1
                                RAL
                                DAC     CDRWD1
                                ISZ     CDR7CT
                                JMP     CDCPL1
                                ISZ     ICA             /POINT TO NEXT CARD COL
                                ISZ     CDR5CT          /HAVE WE PROCESSED 5 WORDS?
                                JMP     CDML2           /NO GET ANOTHER ONE
                                LAC     CDWDCT          /YES -- UPDATE WORD COUNT AND
                                TAD     (2              /CHECK TO SEE IF WE HAVE OVERFLOWED THE
                                DAC     CDWDCT          /USER'S BUFFER
                                SMA
                                JMP     CDVER2          /YES -- WE HAVE OVERFLOWED
                                LAC     CDRWD2          /NO -- INSERT 5/7 WORDS IN USER'S BUFFER
                                CLL!RAL
                                DAC     CDRWD2
                                LAC     CDRWD1
                                RAL
                                DAC*    X13             /STORE FIRST WORD
                                LAC     CDRWD2
                                DAC*    X13             /STORE SECOND WORD
                                ISZ     CDCOLC
                                JMP     CDRM5
                        /
                                .ENDC
                        /
                                .IFDEF UC15
                        /
                        /   IN THE CASE OF THE UNICHANNEL, WE RECIEVE A 42(10) WORD
                        /  BUFFER. THE FIRST WORD IS A BYTE COUNT (NOW ALWAYS 80(10)).
                        /  NOTE THAT AN EOF CARD HAS A BYTE COUNT OF 1!!
                        /  SPOOLER DOES CHECKSUM CALCULATION, NOT US.
                        /  THE SECOND IS A CHECKSUM SO ENTIRE BUFFER ADDS TO 0
                        /  !!!###MODULO 2^16 THAT IS###!!!. THEN ARE 40(10) WORDS
                        /  OF 'COMPRESSED COLUMN'. (SEE CR-11 DRIVER MANUAL). EACH
                        /  WORD HAS TWO EXTRANEOUS BITS AT LEFT, THE !SECOND CHAR!
                        /  OF THE PAIR, AND FINALLY THE FIRST CHAR OF PAIR AT RIGHTMOST
                        /  OF WORD. THE PDP-11 HAS ALREADY CHECKED FOR VALID PUNCH
                        /  COMBINATIONS (64 VALID CARD ASCII, PLUS 12-1-8 FOR ALTMODE).
                        /
00171 R 750030 A        RETRY   CLA!IAC         /SET VARIABLE SAYBING WE'RE WAITING FOR
00172 R 040407 R                DAC     POST    /INTERRUPT
00173 R 140554 R                DZM     CDON    /AND SAY WE HAVEN'T GOTTEN IT YET
00174 R 200614 R                LAC     TCBP    /ADDR OF TABLE TELLING PDP-11 TO READ CARD
00175 R 100616 R                JMS     CDIU    /ROUTINE TO SEND REQUEST TO PDP-11
00176 R 600057 R                JMP     WFTGR   /WAIT FOR COMPLETION INTERRUPT
                        /
                        /  COME BACK HERE WHEN CARD IS READ
                        /
00177 R 200571 R        GOTCRD  LAC     RRN     /RESTORE RN NODE
00200 R 040564 R                DAC     RN
00201 R 140407 R                DZM     POST    /CLEAR INTERRUPT FLAGS
00202 R 140554 R                DZM     CDON    /BEST TO CLEAR POST FIRST!
00203 R 200605 R                LAC     EV11    /EVENT VARIABLE FROM PDP-11
00204 R 742010 A                RTL             /PDP-11 SIGN BIT TO OUR SIGN BIT
00205 R 745120 A                SPA!CLL!RAR     /SKIP IF OK, START CLEARING HIGH BITS
00206 R 600636 R                JMP     CDUCEC  /GO CHECK WHICH KIND OF PIREX ERROR
00207 R 220673 R                LAC*    (IBUF+2 /GET FIRST CHARACTER PAIR (2 WORD HDR)
00210 R 540674 R                SAD     (104611 /SPOOLER USES AN ALT-ALT CARD AS AN END
                        /                       /OF DECK CARD, WE SHOULD IGNORE IT!!
00211 R 600171 R                JMP     RETRY   /IT WAS ONE, JUST READ THE NEXT CARD
00212 R 500675 R                AND     (340    /12,11,0 PUNCHES IN FIRST COLM.=EOF
00213 R 340676 R                TAD     (445    /IF IT IS ONE, MAKE A 1005
00214 R 540677 R                SAD     (1005   /WELL, IF SO GO LACE 1005 AS HEADER
00215 R 600420 R                JMP     REQCMA  /EOF CARD, JUST SET HEADER.
00216 R 200572 R        TRANS   LAC     TX12    /SETUP X12,X13 FOR USER BUFFER
00217 R 060700 R                DAC*    (X12    /MANIPULATIONS.X12 HEADER POINTER
00220 R 200573 R                LAC     TX13    /X13 DATA POINTER
00221 R 060701 R                DAC*    (X13
                        /
00222 R 200673 R                LAC     (IBUF+2 /DATA STARTS AT BUFF+2
00223 R 744010 A                CLL!RAL         /TOP 17 BITS ADDRESS, LAST IS RIGHT-LEFT FLOP
00224 R 040405 R                DAC     CDIPTR  /TO GET INCOMING CHAR'S
00225 R 777660 A                LAW     -120    /80 CHAR'S
00226 R 040560 R                DAC     CDCOLC  /NOTE WE USE COUNTERS DIFERENT ALSO
00227 R 200331 R        PKINT   LAC     PAKI    /INIT 5/7 PACKER TO EXPECT
00230 R 040327 R                DAC     PAKSW   /1ST CHAR OF A BUNCH OF FIVE
00231 R 200566 R                LAC     CDWDCT  /WE USE AS COUNT OF PAIRS, NOT WORDS
00232 R 744020 A                CLL!RAR         /SO DIVIDE BY TWO
00233 R 040566 R                DAC     CDWDCT
00234 R 200405 R        CDRML2  LAC     CDIPTR  /WATCH IT! TOP 17 BITS ADDR, LOW BIT LEFT
00235 R 440405 R                ISZ     CDIPTR  /RIGHT FLIP-FLOP. AND!! POINTER POINTS TO
                        /                       /NEXT CHAR, NOT LAST ONE RETREIVED.
00236 R 744020 A                CLL!RAR         /FLIP-FLOP TO LINK, ADDR AC
00237 R 040406 R                DAC     CDT1    /HOLD POINTER IN TEMPORARY
00240 R 220406 R                LAC*    CDT1    /GET CHARACTER PAIR
00241 R 741410 A                SZL!RAL         /THESE THREE GET CORRECT CHAR
00242 R 743030 A                SWHA!SKP        /TO LOW ORDER 8 BITS OF WORD
00243 R 740020 A                RAR
```

Figure 4-2
PDP-15 CR11 RSX-PLUS III Handler (cont.)

```
00244 R 500702 R               AND     (377    /STRIP OTHER CHARACTER
                        /                      /AT THIS POINT HAVE CLOMNS 12,11,0,9,8,1-7
                        /                      /WHERF 1-7 COOFD IN THREE BITS
00245 R 040406 R               DAC     CDT1    /HOLD
00246 R 540404 R               SAD     CDALT   /ALT MODE SPECIAL CASE, NO REMAP
00247 R 600260 R               JMP     CDGALT  /REJOIN AS SPECIAL CASE
00250 R 500703 R               AND     (20     /IF NINE PUNCH, PECIAL CASE, REMAP TO 8,1 PUNCH
00251 R 740200 A               SZA             /COMBO FOR OUR TRANSLATE, SKIP IF NOT NINE
00252 R 777771 A               LAW     -7      /ADDED TO '9' GIVES '8' AND '1'
00253 R 340406 R               TAD     CDT1    /REMAPPED,
00254 R 040406 R               DAC     CDT1    /SAVE, NOW TO MOVE BOTTOM FOUR BITS LEFT ONE
00255 R 500664 R               AND     (17     /POSITION (9 POSITION NOW VACATED!)
00256 R 340406 R               TAD     CDT1    /THIS DOES TT, LEAVING LOW ORDER BIT ZERO
                        /                      /NOW COLUMNS 12,11,0,8,1-7,ZERO BIT!
00257 R 745000 A               SKP!CLL         /HIDF YOUR HEAD, CLL FOR COMING RTR,SKIP
                        /                      /OVER ALT-MODE RE-ENTRY
00260 R 200704 R        CDGALT LAC     (240    /INDEX TO ALT MODE
00261 R 742020 A               RTR             /RIGHT-LEFT TO LINK, INDEX TO AC
00262 R 340705 R               TAD     (CDTABL /TABLE ADDR
00263 R 040406 R               DAC     CDT1
00264 R 220406 R               LAC*    CDT1    /GET PAIR FROM TRANSLATE TABLE
00265 R 740400 A               SNL             /HERE 0 IS LEFT, IN NORMAL SENSE
00266 R 742030 A               SWHA
00267 R 100323 R               JMS     PAK57   /5/7/ PACKER (IT STRIPS XTRA BITS)
00270 R 440560 R               ISZ     CDCOLC  /807
00271 R 600234 R               JMP     CDRML2  /NO
00272 R 600410 R               JMP     CDCLOS  /YES
                        /
                        /  TRANSLATE TABLE 4 GROUPS OF 16 CHAR'S, TWO PER WORD. 8 WORD
                        /  SPACE BETWEEN LAST TWO GROUPS, IN WHICH WE PUT OTHER STUFF
                        /  CONDITIONALIZED FOR 026-029 OF COURSE. LEFT HAND CHAR IS FIRST.
                        /
                               .IFUND DEC026
00273 R 040061 A        CDTABL 040061  /BLANK, 1-PUNCH
00274 R 062063 A               062063  /2-PUNCH,3-PUNCH
00275 R 064065 A               064065  /4,5
00276 R 066067 A               066067  /6,7
00277 R 070071 A               070071  /8,9(ORDERED AS 8-1)
00300 R 072043 A               072043  /8-2,8-3
00301 R 100047 A               100047  /8-4,8-5
00302 R 075042 A               075042  /8-6,8-7
00303 R 060057 A               060057  /0,0-1
00304 R 123124 A               123124  /0-2,0-3
00305 R 125126 A               125126  /0-4,0-5
00306 R 127130 A               127130  /0-6,0-7
00307 R 131132 A               131132  /0-8,0-9(ORDERED AS 0-8-1)
00310 R 135054 A               135054  /0-8-2,0-8-3
00311 R 045137 A               045137  /0-8-4,0-8-5
00312 R 076077 A               076077  /0-8-6,0-8-7
00313 R 055112 A               055112  /11,11-1
00314 R 113114 A               113114  /11-2,11-3
00315 R 115116 A               115116  /11-4,11-5
00316 R 117120 A               117120  /11-6,11-7
00317 R 121122 A               121122  /11-8,11-9(ORDERED AS 11-8-1)
00320 R 041044 A               041044  /11-8-2,11-8-3
00321 R 052051 A               052051  /11-8-4,11-8-5
00322 R 073134 A               073134  /11-8-6,11-8-7
                               .ENDC
                               .IFDEF DEC026
                        CDTABL 040061
                               062063
                               064065
                               066067
                               070071
                               137075
                               100136
                               047134
                               060057
                               123124
                               125126
                               127130
                               131132
                               073054
                               050042
                               043045
                               055112
                               113114
                               115116
                               117120
                               121122
                               072044
                               052133
                               076046
                               .ENDC
                        /
                        /  NOW THE 8 LOC. BREAK IN THE TABLE
                        /
                        /  THE 5/7 PACKER, A LITTLE TRICKY PAKSW KEEPS A PC WHICH
                        /  'REMEMBERS' WHICH CHARQCTER OF 5 WE ARE AT. TO INIT PACKER,
                        /  SEE TWO LINFS OF CODE AT PAKINT. NORMAL 'FLUSH' OUT WOULD
                        /  BE TO SEND NUL CHAR'S UNTIL PAKSW=PAKI. IN THIS
                        /  HANDLER, PAST HISTORY SAYS WE TRUNCATE ALWAYS AT A WORD
                        /  PAIR BOUNDARY, EVEN FOR SHORT BUFFERS. I AM AFRAID TO
                        /  CHANGE THIS, EVEN THOUGH I DON'T LIKE IT.
                        /
00323 R 000000 A        PAK57  0               /CALL WITH CHAR IN AC, (DESTROYED)
                        /                      /PUSHES CHAR'S THRU X13. EARLY END CHECK
                        /                      /IN CDWDCT.
00324 R 500706 R               AND     (177    /STIP XTRA
00325 R 744000 A               CLL             /FOR ALL ROTATES AND SWAPS!
```

Figure 4-2
PDP-15 CR11 RSX-PLUS III Handler (cont.)

```
00326 R 620327 R                JMP*    PAKSW   /TO WHATEVER ACTION THIS CHAR. NEEDS.
00327 R 740040 A        PAKSW   HLT             /POINTER TO ACTINS FOR CHARACTER
00330 R 620323 R                JMP*    PAK57   /THAT'S ALL, OUT
00331 R 000345 R        PAKI    PAKST           /INIT PAKSW FOR FIRST CHAR.
00332 R 000000 A        PAKT    0               /TEMPORARY FOR PARTIAL WORDS
                        /
                        /  REST OF TRANSLATE TABLE
                        /
                                .IFUND DEC026
00333 R 046101 A                046101  /12,12-1
00334 R 102103 A                102103  /12-2,12-3
00335 R 104105 A                104105  /12-4,12-5
00336 R 106107 A                106107  /12-6,12-7
00337 R 110111 A                110111  /12-8,12-9(ORDERED AS 12-8-1)
00340 R 133056 A                133056  /12-8-2,12-8-3
00341 R 074050 A                074050  /12-8-4,12-8-5
00342 R 053136 A                053136  /12-8-6,12-8-7
                                .ENDC
                                .IFDEF DEC026
                                053101
                                102103
                                104105
                                106107
                                110111
                                077056
                                051135
                                074041
                                .ENDC
00343 R 175000 A                175000          /ALT MODE, FOR BOTH PUNCH SETS.
                        /
                        /  NOW REST OF 5/7 PACKER
                        /
00344 R 100327 R        PAKQ    JMS     PAKSW   /5TH CHAR WRAP BACK TO 1ST. JMS TO PAKSW
                        /                       /LEAVES ADDR OF ACTION FOR 1ST.!.
00345 R 742010 A        PAKST   RTL             /1ST CHARACTER ACTION, MOVE TO LEFT OF WORD
00346 R 742030 A                SWHA
00347 R 040332 R                DAC     PAKT    /HOLD AS PARTIALLY ASSEMBLED WORD
00350 R 100327 R                JMS     PAKSW   /LEAVE POINTER TO 2ND CHAR
                        /
00351 R 742010 A                RTL             /2ND CHAR ACTION
00352 R 742010 A                RTL
00353 R 240332 R                XOR     PAKT    /MARGE WITH FIRST
00354 R 040332 R                DAC     PAKT    /WAIT FOR PART OF 3RD TO FILL WORD
00355 R 100327 R                JMS     PAKSW   /LEAVE POINTER TO THIRD
                        /
00356 R 742020 A                RTR             /3RD, TWO PARTS, FIRST IS TOP 4 BITS
00357 R 740020 A                RAR             /RIGHT JUSTIFIED 1ST WORD OF PAIR
00360 R 040327 R                DAC     PAKSW   /VERY-TEMPORARY IN HERE
00361 R 500664 R                AND     (17     /ZAP OTHER BITS
00362 R 240332 R                XOR     PAKT    /COMPLETE 1ST WORD OF PAIR
00363 R 060013 A                DAC*    X13     /PLACE IN USER BUFFER
00364 R 200327 R                LAC     PAKSW   /GET BACK THIRD CHAR (LINK STILL OK!!!)
00365 R 740020 A                RAR             /2ND JOB, LOW THREE BITS OF CHAR TOP OF
00366 R 500707 R                AND     (700000 /2ND WORD OF PAIR
00367 R 040332 R                DAC     PAKT    /WHEW!, HOLD THAT IN PARTIAL WORD
00370 R 100327 R                JMS     PAKSW   /LEAVE POINTER FOR FOURTH
                        /
00371 R 742030 A                SWHA            /4TH, SNUG UP TO 3 BITS ON TOP
00372 R 740020 A                RAR
00373 R 240332 R                XOR     PAKT    /TOGETHER
00374 R 040332 R                DAC     PAKT
00375 R 100327 R                JMS     PAKSW   /LEAVE POINTER FOR 5TH
                        /
00376 R 440566 R                ISZ     CDWDCT  /OVERFLOW SHORT BUFFER?
00377 R 741010 A                SKP!RAL         /NO,  RAL LEAVE XTRA BIT OF PAIR ON RIGHT
00400 R 600452 R                JMP     CDVER2  /UH-OH, GO CORRECT
00401 R 240332 R                XOR     PAKT    /COMPLETE 2ND WORD OF PAIR
00402 R 060013 A                DAC*    X13     /PLACE
00403 R 600344 R                JMP     PAKQ    /GO PLACE PAKSW FOR FIRST CHAR OF FIVE
                        /
00404 R 000211 A        CDALT   211
00405 R 000000 A        CDIPTR  0               /POINTER TO INPUT DATA IN INPUT BUFFER
                        /                       /FRMAT. LOW BIT RIGHT-LEFT FLIPFLOP
                        /                       /TOP 17 BITS ADDRESS
00406 R 000000 A        CDT1    0               /TEMPORARY FOR TRANSLATION
00407 R 000000 A        POST    0               /0 WHEN NOT WAITING FOR INTERRUPT, 1 WHEN YES.
                                .ENDC
                        / THE BUFFER HAS BEEN REMAPPED -- STORE A 'CR' IN THE TRAILER
                        / WORD AND SET UP THE HEADER WORD
                        /
00410 R 200710 R        CDCLOS  LAC     (64000
00411 R 060013 A                DAC*    X13             /SET 'CR' IN USER BUFFER
00412 R 200560 R                LAC     CDCOLC          /CDCOLC IS NEGATIVE
00413 R 723022 A                AAC     22
00414 R 744000 A                CLL                     /ROTATE INTO PLACE
00415 R 640711 A                ALS     11              /SHIFT INTO POSITION
00416 R 340565 R                TAD     CDRVAL          /ADD IN BUFFER OVERFLOW IF ANY (BITS 12 & 13 =1)
00417 R 723002 A                AAC     2
00420 R 060012 A        REQCMA  DAC*    X12             /SET HEADER WORD ONE
00421 R 777777 A        REDCOM  LAW     -1      /SET RRN, SAYING NO MORE READ OUTSTANDING
00422 R 040571 R                DAC     RRN
00423 R 750030 A        REQCMP  CLA!IAC
00424 R 100426 R        SEV     JMS     SEVRN   /SUB. TO SET EV, RETURN NODE
00425 R 600060 R                JMP     PQ      /GO LOOK FOR MORE WORK
                        /
                        /
                        /    SEVRN
                        /
                        /
                        /  ROUTINE  IS CALLED WITH VALE FOR EV IN AC
```

Figure 4-2
PDP-15 CR11 RSX-PLUS III Handler (cont.)

```
                                  /  THE NODE ADDR. IS IN RN
                                  /
                                  /  EV IS SET, SIGNIFICANT EVENT DECLARED, IOCD ODOE, NODE RETURNED.
                                  /
        00426 0 000000 A          SEVRN   0
        00427 0 722000 A                  PAL                       /SAVE AC VALUE
        00430 0 200564 R                  LAC      RN       /NODE ADDR
        00431 0 060651 R                  DAC*     (R2      /SYSTEM ARGUMENT HOLDER
        00432 0 340563 R                  TAD      XADJ     /ADJUST FOR PREESENT PAGE
        00433 0 721000 A                  PAX               /FOR XR ADDRESSING
        00434 0 210006 A                  LAC      6,X      /EVENT VARIABLE ADDRESS
        00435 0 741200 A                  SNA               /SKIP IF REALLY ONE
        00436 0 600443 R                  JMP      NOSET    /NOPE, SO DON'T SET
        00437 0 340563 R                  TAD      XADJ     /MODIFY IT FOR ADDRESSING
        00440 0 721000 A                  PAX
        00441 0 730000 A                  PLA               /BRING BACK SETTING VALUE
        00442 0 050000 A                  DAC      0,X      /THERE IT GOES!
        00443 0 200711 R          NOSET   LAC      (401000 /DECLARE A SIGNIFICANT EVENT
        00444 0 705504 A                  ISA
        00445 0 200704 R                  LAC      (POOL    /GIVE NODE TO POOL
        00446 0 060647 R                  DAC*     (R1      /SYSTEM ARGUMENT REG
        00447 0 120712 R                  JMS*     (IOCD    /DECREMENT IO COUNT
        00450 0 120713 R                  JMS*     (NADD    /GIVE BACK NODE
        00451 0 620426 R                  JMP*     SEVRN    /THAT/S IT
                                  /
                                  /
                                  /
                                  / ***** BUFFER OVERFLOW
                                  /
        00452 0 777776 A          CDVER2  LAW      -2                /BACKUP USER BUFFER PTR
        00453 0 360701 R                  TAD*     (X13)
        00454 0 060701 R                  DAC*     (X13)
        00455 0 200714 R                  LAC      (60)              /SET OVERFLOW BITS FOR USE BY CDCLOS
        00456 0 040565 R                  DAC      CDRVAL
        00457 0 600410 R                  JMP      CDCLOS
                                  /
        00460 0 777771 A          EVM7    LAW      -7                /ILLEGAL DATA MODE.
        00461 0 600424 R                  JMP      SEV
        00462 0 777750 A          EVM30   LAW      -30               /I/O PARAM. OUT OF PARTITION.
        00463 0 600424 R                  JMP      SEV
                                  /
                                          .IFUND   UC15
                                  /
                                  AEVM6   LAW      -6                /ILLEGAL FUNCTION.
                                          JMP      SAEV              /SET ABORT EV.
                                  /
                                  /ON ILLEGAL CARD PUNCH, WAIT FOR READER NOT READY FOLLOWED BY
                                  /READER READY SEQUENCE BEFORE READING ANOTHER CARD.
                                  /
                                  ILLCP   LAC      (ERRMG2)          /TYPE 'ILLEGAL CARD PUNCH'.
                                          JMS      TTYOUT
                                          JMS      WF.SW             /WAIT FOR READER NOT READY.
                                                   WFOFF             /PSUEDO INSTR. FOR WF.SW.
                                          JMS      WF.SW             /WAIT FOR READER READY.
                                                   WFON              /PSUEDO INSTR. FOR WF.SW.
                                          JMP      RETRY             /READ ANOTHER CARD.
                                  /
                                  /       SUBR. TO WAIT FOR READER NOT READY OR READY FOR READ
                                  /       PER PSUEDO INSTR. IN CALLING SEQUENCE. AFTER MARK TIME REQS.,
                                  /       THE TRIG. EV. IS CHECKED FOR AN ABORT REQ. IN THE QUEUE.
                                  /       IF TASK REQ. READ IS TO BE ABORTED, THE SUBR. DOESN'T
                                  /       RETURN NORMALLY,BUT EVENTUALLY JUMPS TO CDABRT.
                                  /       CALLING SEQUENCE:
                                  /
                                  /       JMS      WF.SW
                                  /                PSUED. INSTR.   (WFOFF OR WFON)
                                  /       SUBR. RETURN ,IF NO INTERVENING ABORT FOR THIS TASK.
                                  /
                                  WF.SW   0
                                          LAC*     WF.SW             /GET PSUEDO INSTR.
                                          DAC      PV1
                                          ISZ      WF.SW             /BUMP EXIT.
                                  WF.SWA  CRRS                       /READ CARD READER STATUS.
                                          AND      (20)              /CHECK FOR READER READY FOR READ.
                                  PV1     XX                         /SNA OR SZA.  (READER READY IF NON-ZERO AC).
                                          JMP*     WF.SW             /EXIT.
                                          CAL      MTCPB             /MARK TIME FOR WAIT.
                                          CAL      WFECB             /WAIT FOR MARK TIME INTERVAL.
                                          DZM      EV
                                          LAC      TG                /CHECK FOR ABORT REQ. IN QUEUE.
                                          RTL
                                          SMA                        /ABORT REQ.?
                                          JMP      WF.SWA            /CHECK AGAIN.
                                          DZM      TG                /YES.  DEQUEUE ABORT REQ.
                                          LAC      PDVNA   /PDVL NODE ADDR.
                                          DAC*     (R1)
                                          JMS*     (DQRQ)            /DEQUEUE ABRT. REQ. R1,R2,R4,R5,R6,XR,AC
                                          NOP                        /ALTERED.  ASSUME ABRT. REQ. IN QUEUE.
                                          DAC      RN                /SAVE ABORT REQ. NODE ADDR.
                                          TAD      XADJ              /SET XR.
                                          PAX
                                          LAC      6,X               /GET ABRT. REQ. EV.
                                          DAC      ARE
                                          LAC      5,X               /CHECK FOR ZERO LUN.
                                          AND      (777000)          /BITS 0-8
                                          SZA
                                          JMP      AEVM6             /ERROR.  NON-ZERO LUN.
                                          LAC      2,X               /GET STL. NODE PTR. AND CHECK AGAINST
                                          SAD      STLA              /READ REQ. STL NODE PTR. SAME?
                                          JMP      CDABRT            /YES.  ABORT READ REQ. AND CLEAN UP.
```

Figure 4-2
PDP-15 CR11 RSX-PLUS III Handler (cont.)

```
                            LAC     PDVNA   /NO.  CLEAN UP QUEUE OF TASK TO BE ABRTED.
                            DAC*    (R1)            /ALSO RETR. ABRT. REQ. NODE TO POOL AND
                            LAC     RN              /DECR. TRANSF. PEND. CNT.  ABRT. REQ. NODE
                            DAC*    (R2)            /ADDR. TO R2.
                            JMS*    (DMTQ)          /EMPTY REQ. QUEUE OF ALL I/O
                                                    /REQ.'S MADE BY TASK BEING ABORTED.
                                                    /R1,R2,R3,R5,R6,X10,X11,X12,XR,AC ALTERED.
                            LAC     (1)             /SET ABRT. REQ. EV TO +1.
                    SAEV    PAL
                            LAC     ARE             /ABORT REQ. EV.
                            TAD     XADJ
                            PAX
                            PLA
                            DAC     0,X
                            LAC     (401000)
                            ISA                     /DECLARE SIGNIF. EVENT.
                            LAC     RN              /RETRN. ABRT. REQ. NODE TO POOL.
                            DAC*    (R2)
                            LAC     (POOL)
                            DAC*    (R1)
                            JMS*    (IOCD)          /DECR. TRANSF. PEND. CNT.
                            JMS*    (NADD)          /RETRN. NODE TO POOL.
                            JMP     WF.SWA          /CHECK AGAIN.
                    CDARD   CLA!IAC                 /SET CARD DONE FLAG.
                            DAC     CDON
                            JMP     CDABRT          /PROCEED WITH ABORT.
                    /
                            .ENDC
                            .EJECT
                    /
                    / EXIT REQUEST (FROM TASK "....REA")
                    /
00464 R 200704 R    DAEX    LAC     (POOL)          /RETURN REQUEST NODE TO POOL
00465 R 060647 R            DAC*    (R1)
00466 R 200564 R            LAC     RN
00467 R 060651 R            DAC*    (R2)
00470 R 120712 R            JMS*    (IOCD)          /DECREMENT TRANSF. PENDING COUNT
00471 R 120713 R            JMS*    (NADD)
                            .IFUND  UC15
                            LAC     (CC1)           /CONDITION CODE 1 -- CLEAR CONTROL.
                            CRLC
                            CAL     DCPB            /DISCONNECT
                            .ENDC
                            .IFDEF  UC15
00472 R 100625 R            JMS     CLEAR   /CLEAR DEVICE , WAIT FOR COMPLETION
00473 R 440577 R            ISZ     CCPB            /MAKE CONNECT A DISCONNECT (BURP)
00474 R 000577 R            CAL     CCPB            /DISCONNECT
                            .ENDC
00475 R 440570 R            ISZ     PDVTA           /POINT TO ASSIGN INHIBIT FLAG
00476 R 705522 A            .INH                    /INHIBIT INTERRUPTS.
00477 R 160570 R            DZM*    PDVTA           ///ZERO IT
00500 R 705521 A            .ENB                    ///ENABLE INTERRUPTS.
00501 R 000653 R            CAL     (10)            ///EXIT
                    /
                    /
                    /ABORT REQUEST.
                    /
00502 R 777000 A    CDABRT  LAW     17000   /MASK TO KEEP HALF WORD TO CHECK ABORT VALIDITY
00503 R 510005 A            AND     5,X     /HAS TO BE ZERO TO BE OK
00504 R 740200 A            SZA             /SO SKIP IF OK
00505 R 600116 R            JMP     EVM6    /ERROR RETURNED IF NOT
00506 R 200567 R            LAC     PDVNA   /MT THE DEQUE FOR THE ABORTED TASK
00507 R 060647 R            DAC*    (R1
00510 R 200564 R            LAC     RN      /ABORT NODE
00511 R 060651 R            DAC*    (R2
00512 R 120715 R            JMS*    (DMTQ   /THIS ROUTINE DOES ALL WORK
                    /
                    /  NOW WAS THIS ABORT FOR AN OUTSTANDING READ?
                    /
00513 R 200564 R            LAC     RN      /2+RN IS STL NODE ADDR
00514 R 340563 R            TAD     XADJ    /USE AS IDENTIFIER
00515 R 721000 A            PAX
00516 R 210002 A            LAC     2,X
00517 R 540556 R            SAD     STLA    /SAME ADDR FOR LAST READ DONE
00520 R 751001 A            SKP!CLA!CMA     /SKIP IF SAME, SET UP -1
00521 R 600423 R            JMP     REQCMP  /NOPE, WE'RE DONE, GO GIVE BACK NODE ETC.
00522 R 240571 R            XOR     RRN     /NASTY, MAKES 0 IF NO READ NOW! IN PROGRESS
00523 R 741201 A            SNA!CMA         /SKIP IF READ IN PROGRESS, RECREATE ITS NODE ADDR!
00524 R 600423 R            JMP     REQCMP  /NOPE, JUST COMPLETE
00525 R 060651 R            DAC*    (R2     /GIVE BACK NODE AND IOCD FOR SUSPENDED READ
00525 R 200704 R            LAC     (POOL
00527 R 060647 R            DAC*    (R1
00530 R 120712 R            JMS*    (IOCD
00531 R 120713 R            JMS*    (NADD
00532 R 750001 A            CLA!CMA         /SET READ NOT HERE SWITCH
00533 R 040571 R            DAC     RRN
                            .IFUND  UC15
                            LAC     (CC1    /CLEAR DEVICE
                            CRLC
                            .ENDC
                            .IFDEF  UC15
00534 R 100625 R            JMS     CLEAR   /AND CLEAR FOR UNICHANNEL
                            .ENDC
00535 R 600423 R            JMP     REQCMP          /DONE
                    /
                    /
                    /
                    /
                            .EJECT
```

Figure 4-2
PDP-15 CR11 RSX-PLUS III Handler (cont.)

```
                        /
                        / INTERRUPT SERVICE ROUTINE
                        /
00536 R 000000 A        INT     0
00537 R 707762 A                DBA
00540 R 040000 R                DAC     START           /SAVE AC
                                .IFUND  UC15
                                CRRS                    /READ STATUS INTO AC.
                                DAC     EV1             /SAVE FOR TASK LEVEL PROCESSING.
                                AND     (2)             /CARD DONE?  BIT 16.
                                SNA
                                JMP     INT1            /NO.  DON'T CLEAR CARD DONE.
                                DAC     CDON    /PLACE 2 INTO CDON TO SAY DONE
                                LAC     (CC3)           /YES.  CLEAR CARD DONE.  LEAVE
                                CRLC                    /INTERR. AND DCH ENABLED.
                        INT1    CRPC                    /CLEAR ALL BUT CARD DONE.
                                LAC     (CC4)           /ENABLE INTERRS.  DISABLE DCH
                                CRLC                    /NEEDED SINCE CRPC DISABLES INTERRS.
                                .ENDC
                        /
                                .IFDEF  UC15
00541 R 706124 A                CAPI                    /CLEAR FLAG FROM PDP-11
00542 R 200407 R                LAC     POST    /ARE WE WANTING AN INTERRUPT
00543 R 741200 A                SNA             /SKIP IF YES/USE VALUE TO SET
00544 R 600551 R                JMP     INTAC   /NO DO NOTHING
00545 R 040554 R                DAC     CDON    /AS FLAG TO DISTINGUISH CARD DONE FROM CAL
00546 R 040562 R                DAC     TG      /AND SET TG TO WAKE UP CAL LEVEL
                                .ENDC
00547 R 200711 R                LAC     (401000)        /DECLARE SIGNIF. EVENT.
00550 R 705504 A                ISA
00551 R 200000 R        INTAC   LAC     START           /RESTORE AC.
00552 R 703344 A                DBR
00553 R 620536 R                JMP*    INT
                                .EJECT
                        /
                                .IFUND  UC15
                        /SUBR. TO OUTPUT ERROR MESSAGES VIA ERRLUN.  AC SHOULD CONTAIN
                        /ADDRESS OF ERROR MESSAGE BUFFER.
                        /
                        TTYOUT  0
                                DAC     TECPB4          /SET CPB BUFFER ADDRESS.
                                CAL     TE              /TYPE ERROR MESSAGE.
                                CAL     WFECB           /WAITFOR EV.
                                JMP*    TTYOUT
                        /
                        /ERROR MESSAGE BUFFERS AND TABLE OF PTRS.:
                        /
                        ERRPT   .+1
                                ERRMG1
                                ERRMG2
                                ERRMG3
                                ERRMG4
                                ERRMG5
                        /
                        /
                        /
                        ERRMG1  ERRMG2-ERRMG1*1000/2+2
                                0
                                .ASCII '*** CD READER NOT READY'<15>
                        ERRMG2  ERRMG3-ERRMG2*1000/2+2
                                0
                                .ASCII '*** CD ILLEGAL PUNCH'<15>
                        ERRMG3  ERRMG4-ERRMG3*1000/2+2
                                0
                                .ASCII '*** CD PICK ERROR'<15>
                        ERRMG4  ERRMG5-ERRMG4*1000/2+2
                                0
                                .ASCII '*** CD DATA MISSED/PHOTO ERROR'<15>
                        ERRMG5=.
                                .EJECT
                        / ***** CARD COL TO ASCII TRANSLATION TABLE *****
                        /
                        /EACH TABLE ENTRY REPRESENTS VALID ASCII CARD PUNCHES WITH
                        /THE FOLLOWING FORMAT:
                        /
                        /BITS 0 - 5     SIXBIT ASCII CHARACTER.
                        /BITS 6 - 17    CARD PUNCHES WITH THE FOLLOWING MAPPING:
                        /
                        /BIT 6 = ZONE 12
                        /BIT 7 = ZONE 11
                        /BITS 8 - 17  = ZONES 0 - 9.
                        /THE ASSEMBLER BUILDS THE TWOS COMPLEMENT OF BITS 6-17 VIA THE
                        /7777&+1 OPERATION.  THE TABLE IS ORDERED ACCORDING TO INCREASING
                        /MAGNITUDE OF CARD PUNCHES(CONSIDERED AS 12 BIT RIGHT JUSTIFIED
                        /INTEGER VALUES).
                        /EXAMPLE:  ASCII '9' HAS FOLLOWING TABLE REPRESENTATION:

                        /
                        /       710001&7777+1
                        /
                        /WHERE 0001 INDICATES ZONE 9 PUNCHED AND 71 IS SIXBIT ASCII '9'.
                        /
                        /GRAPHIC CHARACTERS FOR 026 PUNCHES ARE IN PARENTHESES BELOW:
                        /
                        CDTABL  CDTABL+1
                                400000                  /BLANK
                                710001&7777+1           /9
                                700002&7777+1           /8
                                670004&7777+1           /7
                                CP 340006,420006        /" (0)
                                660010&7777+1           /6
```

Figure 4-2
PDP-15 CR11 RSX-PLUS III Handler (cont.)

```
                                 CP 470012,750012         /= (')
                                 65002067777+1            /5
                                 CP 360022,470022         /, (A)
                                 64004067777+1            /4
                                 00004267777+1            /@
                                 63010067777+1            /3
                                 CP 750102,430102         /# (=)
                                 62020067777+1            /2
                                 CP 370202,720202         /: (0)
                                 61040067777+1            /1
                                 60100067777+1            /0
                                 32100167777+1            /Z
                                 31100267777+1            /Y
                                 30100467777+1            /X
                                 CP 451006,771006         /? (%)
                                 27101067777+1            /W
                                 CP 431012,761012         /> (#)
                                 26102067777+1            /V
                                 CP 421022,371022         /RIGHT ARROW (")
                                 25104067777+1            /U
                                 CP 501042,451042         /% (()
                                 24110067777+1            /T
                                 54110267777+1            /'
                                 23120067777+1            /S
                                 CP 731202,351202         /](;)
                                 57140067777+1            //
                                 55200067777+1            /-
                                 22200167777+1            /R
                                 21200267777+1            /Q
                                 20200467777+1            /P
                                 CP 462006,342006         /6 (&)
                                 17201067777+1            /O
                                 CP 762012,732012         /; (>)
                                 16202067777+1            /N
                                 CP 332022,512022         /) ([)
                                 15204067777+1            /M
                                 52204267777+1            /*
                                 14210067777+1            /L
                                 44210267777+1            /$
                                 13220067777+1            /K
                                 CP 722202,412202         /↓ (:)
                                 12240067777+1            /J
                                 CP 534000,464000         /& (+)
                                 11400167777+1            /I
                                 10400267777+1            /H
                                 07400467777+1            /G
                                 CP 414006,364006         /^ (!)
                                 06401067777+1            /F
                                 CP 744012,534012         /+ (<)
                                 05402067777+1            /E
                                 CP 354022,504022         /( ())
                                 04404067777+1            /D
                                 CP 514042,744042         /< (])
                                 03410067777+1            /C
                                 56410267777+1            /.
                                 02420067777+1            /B
                                 CP 774202,334202         /[ (?)
                                 01440067777+1            /2
                       CDTLN1    .-1-CDTABL/2
                       CDRALT    4402
                                 .ENDC
                                 .EJECT
                       /
                       / ***** INTERNAL VARIABLES *****
                       /
00554 R 000001 A       CDON      1                  /CARD DONE FLAG.
00555 R 000000 A       TST       0                  /TEMP STORAGE FOR STATUS.
00556 R 000000 A       STLA      0                  /STL NODE. ADDR.
00557 R 000000 A       ARE       0                  /ABORT REQ. EV.
00560 R 000000 A       CDCOLC    0                  /CARD COL COUNT USED IN TRANSLATING CARDS
00561 R 000000 A       EV        0                  /INTERNAL EVENT VARIABLE
00562 R 000000 A       TG        0                  /TRIGGER EVENT VARIABLE
00563 R 000000 A       XADJ       0                 /XR ADJUST CONSTANT TO SUBTRACT PAGE BITS
00564 R 000000 A       RN        0                  /ADDRESS OF THE REQUEST NODE PICKED FROM AUEUE
00565 R 000000 A       CDRVAL    0                  /BUFFER OVERFLOW FLAG WORD
00566 R 000000 A       CDWDCT    0                  /WORD COUNT CHECK WORD SET FROM I/O REQUEST
                       /
                                 .IFUND  UC15
                       /
                       /  SAVE SOME ROOM FOR UC15, THESE ARE NOT NEEDED
                       /
                       ICA       0                  /INTERNAL BUFFER CURRENT ADDRESS POINTER
                       CDR7CT    0                  /SEVEN COUNTER USED BY THE 5/7 ASCII PACKING ROUTINE
                       CDR5CT    0                  /COUNTER FOR 5/7 ASCII PACKING
                       CDTPTR    0                  /POINTER TO TRANSLATION TABLE
                       CDTLEN    0                  /TRANSLATION TABLE LENGTH
                       CD7700    770000             /USED IN CARD TRANSLATION
                       CDCPTR    0                  /POINTER TO CURRENT INTEM IN TRANSLATION TABLE
                       CDRWD3    0                  //
                       CDRWD2    0                  // THREE WORD SHIFT REG. FOR 5/7 ASCII PACKING
                       CDRWD1    0                  //
                       EV1       0                  /CARD READER EV.
                       /
                                 .ENDC
                       /
00567 R 000000 A       PDVNA     0                  /PHYSICAL DEVICE NODE ADDRESS
00570 R 000000 A       PDVTA     0                  /ADDRESS OF ADDRESS OF TEV IN PHY DEV NODE
00571 R 777777 A       RRN       777777             /READ BEING PROC. FLAG.  -1 IF NOT BEING
                                                    /PROCESSED.  READ REQ. NODE ADDRESS IF BEING
                                                    /PROCESSED.
```

Figure 4-2
PDP-15 CR11 RSX-PLUS III Handler (cont.)

```
00572 R 000000 A    TX12    0                   /TEMP. FOR X12 STOR.
00573 R 000000 A    TX13    0                   /TEMP. FOR X13 STOR.
00574 R 000000 A    TCWC    0                   /TEMP. FOR REQ. WC.
                    /
                            .EJECT
                    /
                    / ***** CAL PARAMETER BLOCKS *****
                    /
                    /
00575 R 000020 A    WFTCPB  20                            /WAIT FOR TRIGGER CPB
00576 R 000562 R            TG
                    /
00577 R 000011 A    CCPB    11                            /CONNECT CPB
00600 R 000561 R            EV
00601 R 000015 A            15                            /LINENUMBER
00602 R 000536 R            INT                           /ENTRY ADDRESS OF INTERRUPT SERVICE ROUTINE
                    /
                            .IFUND  UC15
                    /
                    /  UC15 SAVE SPACE BY LEAVING OUT SOME CAL'S
                    /
                    /
                    /
                    WFECB   20                             /WAIT FOR EV CPB
                            EV
                    /
                    DCPB    12                            /DISCONNECT CPB
                            0                             /EV ADDRESS
                            15                            /INTERRUPT LINE NUMBER
                            INT                           /CURRENT INTERRUPT TRANSFER ADDRESS
                    /
                    TE      2700                          /WRITE TO ERRLUN.
                            EV
                            ERRLUN              /WRITE OUT THE ERROR MESSAG TO THE DESIRED
                                                /TELETYPE
                            2
                    TECPB4  XX
                    /
                    MTCPB   13                            /MARK TIME REQ.
                            EV
                            12                            /12 UNITS.
                            1                             /UNIT (TICK).
                    /
                    WFCRCB  20                  /WAIRFOR CR INTERRS.
                            EV1
                    /
                    WFCRCD  20                  /WAIT FOR CARD DONE FLAG TO BE SET.
                            CDON
                    /
                            .ENDC
                    /
                    /
                            .IFDEF  UC15
                    /
                    /  I/O INFORMATION , ROUTINES , ETC, FOR UC15
                    /
                    /  TCB (TASK CONTROL BLOCK) TELLING PDP-11 TO SEND US A CARD
                    /
00603 R 026401 A    TCB     APISLT*400+APILVL       /TELL PDP-11 WHERE TO COME BACK
00604 R 000005 A            DEVCOD                  /PIREX CODE FOR CD;THE 200 BIT SAYS
                    /                               /WE ARE NOT TO BE SPOOLED.
00605 R 000000 A    EV11    0                       /EVENT VARIABLE FROM PDP11 TO US
00606 R 000000 A            0                       /DUMMY, HIGH PORTION OF 18 BIT
                    /                               /ADRESS. NOT PRESENTLY USED
00607 R 000001 R            IBUF                    /POINTER TO BUFFER TO PUT CARD IN
00610 R 000000 A            0                       /UNIT #; FOR FUTURE GENERATIONS.
                    /
                    /  TCB TO TELL PDP11 TO CLEAR OUT CARD READER DEVICE
                    /
00611 R 000000 A    TCBK    0               /THIS WORKS, SEE PIREX FOR INFO.
00512 R 002600 A            DEVCOD&177*400+200
00613 R 000000 A    EV11K   0               /EVENT VARIABLE FOR CLEAR OPERTAION
                    /
                    /  POINTERS TO TCB, TCBK
                    /
00614 R 000603 R    TCBP    TCB
00615 R 000611 R    TCBKP   TCBK
                    /
                    /
                    /  CDIU IS THE SUBROUTINE TO SEND A TCB TO THE PDP-11
                    /
                    /  CAL WITH THE ADRESS OF THE TCB IN THE AC
                    /
00616 R 000000 A    CDIU    0
00617 R 140605 R            DZM     EV11    /CLEAR ONE COMING FROM PDP-11
00620 R 140613 R            DZM     EV11K   /AND THE OTHER ONE, IN CASE IT  USED
00621 R 706001 A            SIOA            /SKIP IF PDP-11 CAN TAKE REQUEST
00622 R 600621 R            JMP     .-1
00623 R 706006 A            LIOR            /TELL IT TO DO TCB WHOSE ADDRESS IN AC
00624 R 620616 R            JMP*    CDIU    /THAT'S ALL THERE IS TO IT.
                    /
                    /
                    /  CLEAR  CLEARS SWITCHES, AND CD IN PIREX, WAITS FOR COMPLETE
                    /
00625 R 000000 A    CLEAR   0
00626 R 140407 R            DZM     POST
00627 R 140554 R            DZM     CDON
00630 R 200615 R            LAC     TCBKP   /TCB FOR CLEAR
00631 R 100616 R            JMS     CDIU
00632 R 000634 R            CAL     WFCLER  /WAIT FOR CLEAROUT
```

Figure 4-2
PDP-15 CR11 RSX-PLUS III Handler (cont.)

```
00633 R 620625 R                JMP*    CLEAR
                        /
00634 R 000020 A        WFCLER  20
00635 R 000613 R                EV11K
                        /  COUCEC EXAMINES NEGATIVE EVENT VARIABLES FROM PIREX
                        /
00636 R 744020 A        COUCEC  CLL!RAR         /CLEAR OTHER TOP BIT
00637 R 340716 R                TAD     (600000 /SIGN EXTEND TO PDP-15 WORD
00640 R 540717 R                SAD     (777001 /THIS ONLY 'LEGAL' VALUE AT PRESENT
00641 R 600171 R                JMP     RETRY   /THAT SAYS PIREX IS OUT OF NODES,
                        /                       /WE SHOULD TRY AGAIN TO GET ONE
00642 R 100426 R                JMS     SEVRN   /OTHERS, RETURN NEG VARIABLE AS EV.
                        /                       /THIS IS SLIGHTLY FLAKEY, BUT WE
                        /                       /REALLY SHOULD NEVER GET HERE!?!?
00643 R 777777 A                LAW     -1      /SAY NO MORE READ OUTSTANDING
00644 R 040571 R                DAC     RRN
00645 R 600060 R                JMP     PQ      /BACK TO LOOK FOR MORE WORK
                        /
                        /
                                .ENDC
           000000 R             .END START
00646 R 200252 A *L
00647 R 000101 A *L
00650 R 000054 R *L
00651 R 000102 A *L
00652 R 000123 A *L
00653 R 000010 A *L
00654 R 000562 R *L
00655 R 070000 A *L
00656 R 000337 A *L
00657 R 000777 A *L
00660 R 000024 A *L
00661 R 000025 A *L
00662 R 000026 A *L
00663 R 000036 A *L
00664 R 000017 A *L
00665 R 000325 A *L
00666 R 000332 A *L
00667 R 200007 A *L
00670 R 000103 A *L
00671 R 000104 A *L
00672 R 000342 A *L
00673 R 000003 R *L
00674 R 104611 A *L
00675 R 000340 A *L
00676 R 200445 A *L
00677 R 001005 A *L
00700 R 000012 A *L
00701 R 000013 A *L
00702 R 000377 A *L
00703 R 000020 A *L
00704 R 000240 A *L
00705 R 000273 R *L
00706 R 000177 A *L
00707 R 700000 A *L
00710 R 064000 A *L
00711 R 401000 A *L
00712 R 000345 A *L
00713 R 000107 A *L
00714 R 000060 A *L
00715 R 000361 A *L
00716 R 600000 A *L
00717 R 777001 A *L
    SIZE=00720      NO ERROR LINES
```

Figure 4-2
PDP-15 CR11 RSX-PLUS III Handler (cont.)

4.6.3.3 Requests - Following handler initialization, requests can be processed. Note that the request de-queuing algorithm (see Figure 4-2 lines 351-406) is executed whenever Q-I/O places a request node in the list associated with the handler's PDVL node or whenever an interrupt for the device has occurred on the PDP-15. The latter condition implies that the handler's interrupt service routine (Figure 4-2, lines 1090-1119) will set the trigger event variable on each interrupt.

4.6.3.4 ABORT Requests - Because of the nature of the UNICHANNEL configuration, ABORT requests should be handled on a high priority basis. Hence, whenever the trigger event variable is set, the handler should first check to see if an ABORT request has been issued. (Figure 4-2, lines 352-356). This condition can be tested using the following algorithm:

```
LAC     TG        /GET THE TRIGGER EVENT VARIABLE INTO THE AC
RTL               /MOVE THE ABORT BIT INTO BIT ZERO OF THE AC
SPA               /SKIP IF ABORT BIT IS NOT SET
JMP     PICK      /ABORT REQUEST-DEQUEUE AND PROCESS IT
 .
 .                /NOT AN ABORT REQUEST--CHECK OTHER
 .                /REASONS FOR HAVING TRIGGER EVENT VARIABLE SET.
```

4.6.3.5 Interrupts - If the trigger event variable was not set due to an ABORT request, either PIREX has issued an interrupt or a new request for I/O is pending. Before checking for new requests, the handler should see if an interrupt occurred (see Figure 4-2, lines 358-361). If it did, the handler should check to see if an interrupt was requested. Unrequested interrupts should be ignored but the handler should finish processing the outstanding I/O request if the interrupt indicates that I/O is now complete.

If the trigger event variable was not set due to an interrupt and no I/O is being processed by PIREX, the handler can pick off the new I/O request and begin processing it (see Figure 4-2, lines 367-406).

On ABORT requests, the handler should determine if I/O is in progress on the PDP-11 for the task being aborted (see Figure 4-2, lines 1057-1066). If so, the handler should issue a "clear device directive" to PIREX to stop the I/O in progress (see Figure 4-2, lines 1072-1079).

The "clear device directive" must also be issued whenever a DISCONNECT and EXIT request from the MCR function REASSIGN is processed (see Figure 4-2, line 1032).

4.6.3.6 READ and WRITE Requests - READ and WRITE request processing usually involves the following procedures:

1. Checking the range of the issuing task's TCB and buffer.
2. Making data conform to PDP-11 standards for WRITE requests and PDP-15 standards for READ requests.
3. Sending a TCB directive to PIREX.

4. Waiting for PIREX to complete the operation initiated by sending the TCB directive.

5. Checking the event variable sent back to the handler by PIREX.

6. Setting data into the issuing task's request buffer for READ.

7. Sending an event variable to the task which initiated the request for I/O.

The following is a brief outline of the procedure used by the UNICHANNEL Card Reader handler when it processes a read request. (Refer to Figure 4-2).

1. Dequeue the I/O request node (lines 351-406)

2. Check the range of the task TCB and buffer (lines 439-464).

3. Clear the TCB event variable (line 1371)

4. Clear the "I/O Done" flag (line 641)

5. Set the "Interrupt Expected" flag (lines 639-640)

6. Issue the READ TCB to the Card Reader Driver in PIREX (lines 1373-1375)

7. Wait for the Trigger Event Variable (line 351)

8. When the Card Reader Driver has completed the request, the Card Reader handler interrupt service routine sets the Trigger Event Variable and the "I/O Done" flag (lines 112-113).

9. The handler then checks the Event Variable sent back by PIREX (lines 652-655).

10. Convert the data to PDP-15 card format and transfer it to the task's buffer (lines 664-878)

11. Set the task's Event Variable (lines 879-880).

12. Wait for the next request (line 351).

Note that in order for a UNICHANNEL handler to function properly, the PDP-11 must be able to access the handler's internal buffers and TCBs. Hence, all locations within these TCBs and buffers must be within the common memory accessible to the PDP-11.[1] Also, note that the RSX POLLER task should be modified to interrogate PIREX concerning the status of the new device.

4.7 BUILDING A PIREX DEVICE DRIVER

A device driver is a software routine that performs rudimentary I/O functions. PIREX device drivers typically operate in conjunction with more complex PDP-15 handlers. While a rudimentary device driver is typical, a PIREX task can be as complex as a full handler. The

(1) Depending on Driver task design the buffers for an NPR device may not have to be in common memory.

PIREX XY driver is a good example of a very complex driver. The PIREX line printer driver, a typical rudimentary driver, will be used to examine the construction of a device driver.

4.7.1 General Layout

The general layout of a driver task (see Figure 4-3) consists of:

1. A stack area which will be used when the task is executing

2. The address of a device control register. This is used to stop the device during STOP I/O requests. Dummy addresses are used for tasks which are not device drivers.

3. A 2-word busy/idle switch used to store the caller's 18-bit TCBP. When the busy/idle switch is zero, the routine is not busy.

4. The task request setup/processing section

5. The task interrupt processor section, if the task is a device driver.

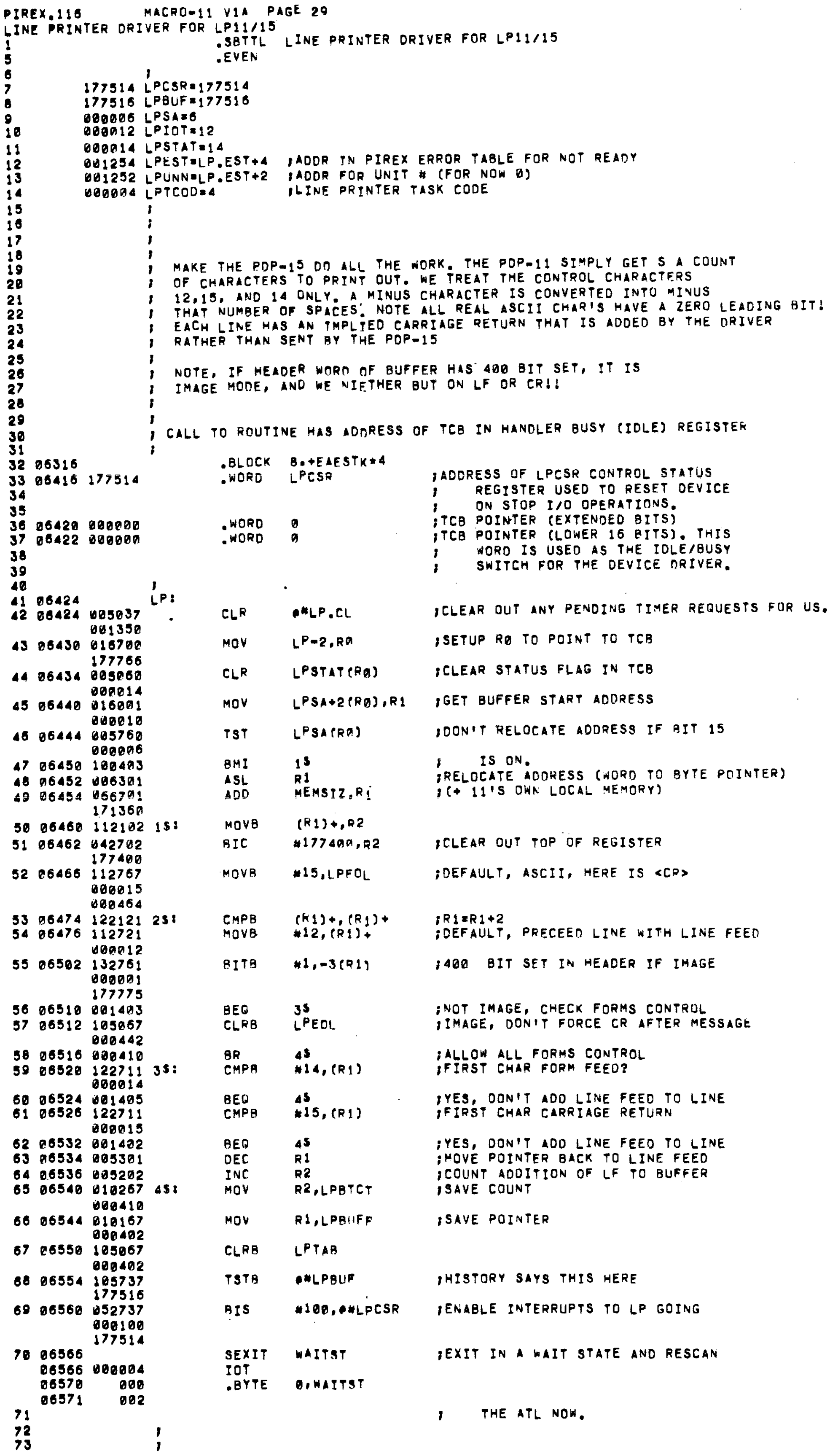
The task request setup/processing section obtains the parameters from the TCB and uses them to set up the referenced device or process the request. Entry into this section is made from the ATL scanner or DEQU with the current task stack area active at the priority level associated with that task. All general purpose registers are available for use by the current task at this time. The TCBP is stored in the busy/idle switch preceding the request section and signifying that the task is busy. Once some operation is underway or completed, **the task returns to the ATL scanner by issuing the 'SEXIT' macro instruction (refer to Section 4.7.2.4).**

If the task is a device driver, the interrupt section is called at the completion of an I/O request. All device interrupt priority vectors specify priority 7. This is done to save the general-purpose registers on the current task stack pointer and lower the system to the priority level of this task.

Control is transferred to the driver, which then checks for errors, stores status information into the TCB, clears the device busy switch (the driver becomes idle when the busy switch is cleared) and sends an optional interrupt (via SEND15, see Figure 3-6) to the system informing it that the request has been processed. The driver then transfers control to the routine DEQU (see Figure 3-7) to determine if more requests are in its TRL. If not, control is transferred to the ATL scanner, after saving the task stack pointer and setting the task status to the wait state in the ATL node.

4.7.2 Task Program Code

The task program code is necessary to carry out the task's function.

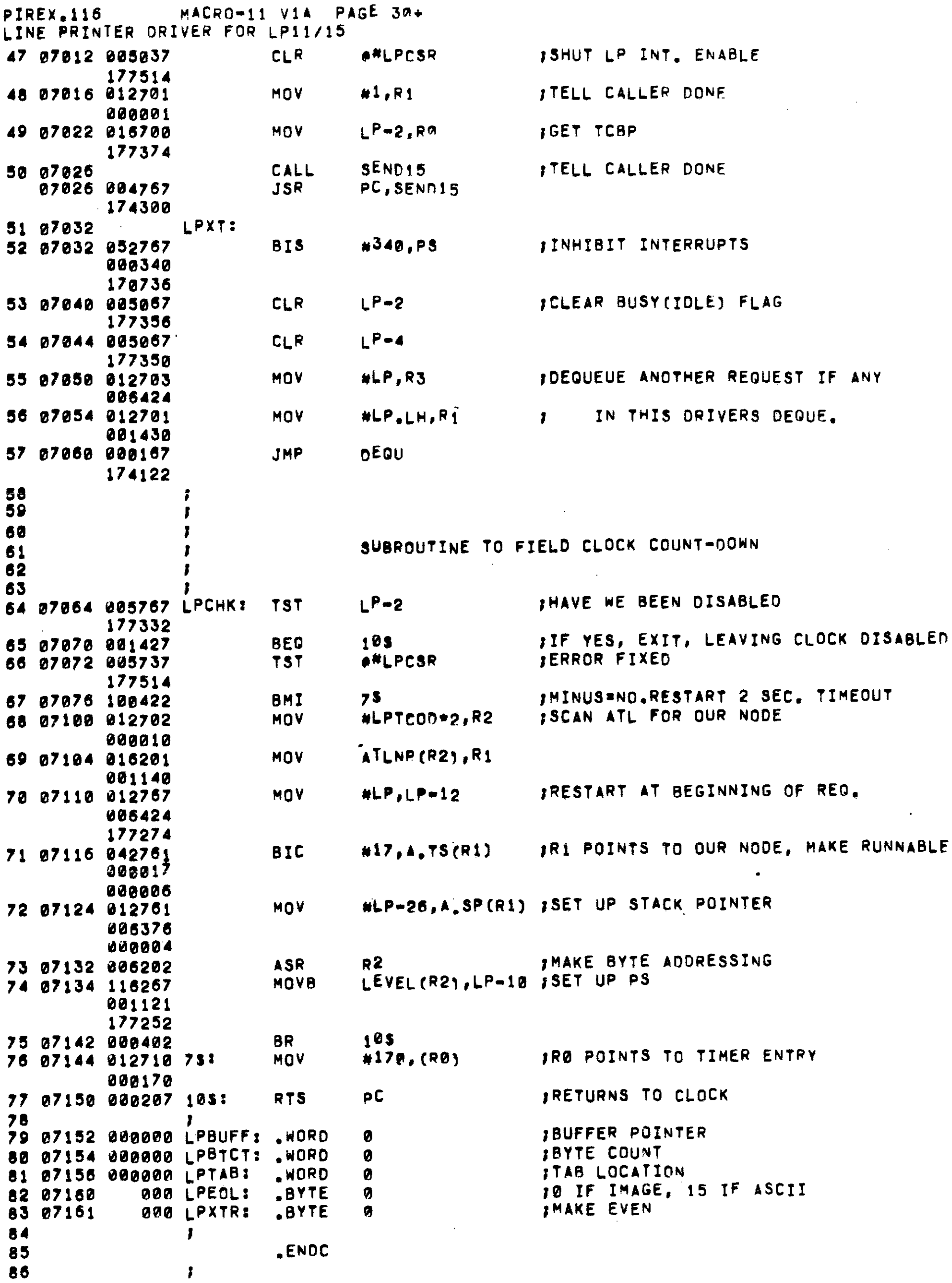


```
PIREX.116        MACRO-11 V1A  PAGE 29
LINE PRINTER DRIVER FOR LP11/15
                             .SBTTL  LINE PRINTER DRIVER FOR LP11/15
                             .EVEN
             ;
      177514 LPCSR=177514
      177516 LPBUF=177516
      000006 LPSA=6
      000012 LPIOT=12
      000014 LPSTAT=14
      001254 LPEST=LP.EST+4  ;ADDR IN PIREX ERROR TABLE FOR NOT READY
      001252 LPUNN=LP.EST+2  ;ADDR FOR UNIT # (FOR NOW 0)
      000004 LPTCOD=4        ;LINE PRINTER TASK CODE
             ;
             ;
             ;
             ;
             ;  MAKE THE PDP-15 DO ALL THE WORK. THE PDP-11 SIMPLY GET S A COUNT
             ;  OF CHARACTERS TO PRINT OUT. WE TREAT THE CONTROL CHARACTERS
             ;  12,15, AND 14 ONLY. A MINUS CHARACTER IS CONVERTED INTO MINUS
             ;  THAT NUMBER OF SPACES. NOTE ALL REAL ASCII CHAR'S HAVE A ZERO LEADING BIT!
             ;  EACH LINE HAS AN IMPLIED CARRIAGE RETURN THAT IS ADDED BY THE DRIVER
             ;  RATHER THAN SENT BY THE PDP-15
             ;
             ;  NOTE, IF HEADER WORD OF BUFFER HAS 400 BIT SET, IT IS
             ;  IMAGE MODE, AND WE NIETHER BUT ON LF OR CR!!
             ;
             ;
             ; CALL TO ROUTINE HAS ADDRESS OF TCB IN HANDLER BUSY (IDLE) REGISTER
             ;
06316                        .BLOCK  8.+EAESTK*4
06416 177514                 .WORD   LPCSR           ;ADDRESS OF LPCSR CONTROL STATUS
                                                     ;    REGISTER USED TO RESET DEVICE
                                                     ;    ON STOP I/O OPERATIONS.
06420 000000                 .WORD   0               ;TCB POINTER (EXTENDED BITS)
06422 000000                 .WORD   0               ;TCB POINTER (LOWER 16 BITS). THIS
                                                     ;    WORD IS USED AS THE IDLE/BUSY
                                                     ;    SWITCH FOR THE DEVICE DRIVER.
             ;
06424        LP:
06424 005037                 CLR     @#LP.CL         ;CLEAR OUT ANY PENDING TIMER REQUESTS FOR US.
      001350
06430 016700                 MOV     LP-2,R0         ;SETUP R0 TO POINT TO TCB
      177766
06434 005060                 CLR     LPSTAT(R0)      ;CLEAR STATUS FLAG IN TCB
      000014
06440 016001                 MOV     LPSA+2(R0),R1   ;GET BUFFER START ADDRESS
      000010
06444 005760                 TST     LPSA(R0)        ;DON'T RELOCATE ADDRESS IF BIT 15
      000006
06450 100403                 BMI     1$              ;    IS ON.
06452 006301                 ASL     R1              ;RELOCATE ADDRESS (WORD TO BYTE POINTER)
06454 066701                 ADD     MEMSIZ,R1       ;(+ 11'S OWN LOCAL MEMORY)
      171360
06460 112102 1$:             MOVB    (R1)+,R2
06462 042702                 BIC     #177400,R2      ;CLEAR OUT TOP OF REGISTER
      177400
06466 112767                 MOVB    #15,LPEOL       ;DEFAULT, ASCII, HERE IS <CR>
      000015
      000464
06474 122121 2$:             CMPB    (R1)+,(R1)+     ;R1=R1+2
06476 112721                 MOVB    #12,(R1)+       ;DEFAULT, PRECEED LINE WITH LINE FEED
      000012
06502 132761                 BITB    #1,-3(R1)       ;400  BIT SET IN HEADER IF IMAGE
      000001
      177775
06510 001403                 BEQ     3$              ;NOT IMAGE, CHECK FORMS CONTROL
06512 105067                 CLRB    LPEOL           ;IMAGE, DON'T FORCE CR AFTER MESSAGE
      000442
06516 000410                 BR      4$              ;ALLOW ALL FORMS CONTROL
06520 122711 3$:             CMPB    #14,(R1)        ;FIRST CHAR FORM FEED?
      000014
06524 001405                 BEQ     4$              ;YES, DON'T ADD LINE FEED TO LINE
06526 122711                 CMPB    #15,(R1)        ;FIRST CHAR CARRIAGE RETURN
      000015
06532 001402                 BEQ     4$              ;YES, DON'T ADD LINE FEED TO LINE
06534 005301                 DEC     R1              ;MOVE POINTER BACK TO LINE FEED
06536 005202                 INC     R2              ;COUNT ADDITION OF LF TO BUFFER
06540 010267 4$:             MOV     R2,LPBTCT       ;SAVE COUNT
      000410
06544 010167                 MOV     R1,LPBUFF       ;SAVE POINTER
      000402
06550 105067                 CLRB    LPTAB
      000402
06554 105737                 TSTB    @#LPBUF         ;HISTORY SAYS THIS HERE
      177516
06560 052737                 BIS     #100,@#LPCSR    ;ENABLE INTERRUPTS TO LP GOING
      000100
      177514
06566                        SEXIT   WAITST          ;EXIT IN A WAIT STATE AND RESCAN
06566 000004                 IOT
06570    000                 .BYTE   0,WAITST
06571    002
                                                     ;    THE ATL NOW.
             ;
             ;
```

Figure 4-3
UNICHANNEL LP Driver

```
PIREX.116       MACRO-11 V1A  PAGE 30
LINE PRINTER DRIVER FOR LP11/15
              ;       LP INTERRUPT ENTRANCE
              ;
006572        LPINT:
006572 042737         BIC     #100,@#LPCSR    ;DISABLE LP INTERRUPT
       000100
       177514
006600 004067         JSR     R0,R.SAVE       ;SAVE REGISTERS
       173154
006604 000004         4                       ;TASK CODE
006606 016700         MOV     LP-2,R0         ;GET TCB POINTER
       177610
006612 001507         BEQ     LPXT            ;IGNORE IF ITS ALREADY BEEN STOPPED BY
                                              ;    A STOP I/O REQUEST.
06614 005737          TST     @#LPCSR         ;CHECK FOR ERROR
      177514
06620 100454          BMI     LPERR           ;YES
06622 005037          CLR     @#LP.CL         ;CLEAR OUT ANY PENDING TIMER REQUEST FOR US.
      001350
06626         LPLOP:
06626 105737          TSTB    @#LPCSR         ;IS PRINTER CURRENTLY GOING?
      177514
06632 100043          BPL     LPSTIL          ;YES:  FORGET CHAR FOR NOW
06634 105767          TSTB    LPTAB           ;IN TAB EXPANSION TO SPACES?
      000316
06640 100421          BMI     4$              ;YES
06642 005367          DEC     LPBTCT          ;DECR CHAR COUNT
      000306
06646 100424          BMI     5$              ;WENT TO -1, MAKE CR TO FINISH LINE
06650 105777          TSTB    @LPBUFF         ;MINUS BYTE IS TAB EXPANSION COUNT
      000276
06654 100406          BMI     6$              ;IS ONE, GO SET UP
06656 117737          MOVB    @LPBUFF,@#LPBUF ;STICK CHAR INTO LINE PRINTER BUFFER
      000270
      177516
06664 005267          INC     LPBUFF          ;MOVE POINTER TO NEXT CHAR
      000262
06670 000756          BR      LPLOP           ;GO DO NEXT
             ;
06672 117767 6$:      MOVB    @LPBUFF,LPTAB   ;SET UP TAB COUNT (MINUS, A LA 15)
      000254
      000256
06700 005267          INC     LPBUFF
      000246
06704 105267 4$:      INCB    LPTAB           ;COUNT A SPACE FOR THIS TAB
      000246
06710 112737          MOVB    #40,@#LPBUF     ;SPACE TO LINE PRINTER
      000040
      177516
06716 000743          BR      LPLOP           ;GO DO NEXT
06720 105767 5$:      TSTB    LPEOL           ;IMAGE OR ASCII
      000234
06724 001403          BEQ     7$              ;IMAGE, DON'T FORCE <CR>
06726 116737          MOVB    LPEOL,@#LPBUF   ;ASCII, HERE IS <CARRIAGE RETURN>
      000226
      177516
06734 005260 7$:      INC     LPSTAT(R0)      ;SET REV TO GOOD COMPLETION
      000014
06740 000417          BR      LPXIT
             ;
06742 052737 LPSTIL:  BIS     #100,@#LPCSR    ;ENABLE INTERRUPT ON LP
      000100
      177514
06750 000411          BR      LPXIT1          ;RESTORE R0-R5 AND RETURN
             ;
06752 012737 LPERR:   MOV     #LPCHK,@#LP.CL+2;ADDR FOR TIMER REQ.
      007064
      001352
06760 012737          MOV     #170,@#LP.CL    ;TWO SECONDS IN TICKS (OCTAL)
      000170
      001350
06766 112737          MOVB    #4,@#LPEST      ;ERROR CODE 1, NOT READY TO TABLE
      000004
      001254
06774 000167 LPXIT1:  JMP     DEQU1           ;SCHEDULE NEXT TASK
      174270
             ;
07000 105037 LPXIT:   CLRB    @#LPEST         ;INDICATE SUCCESSFULL OPERATION
      001254
07004 052767          BIS     #340,PS         ;INHIBIT INT.
      000340
      170764
```

Figure 4-3
UNICHANNEL LP Driver (cont.)



```
PIREX.116       MACRO-11 V1A  PAGE 30+
LINE PRINTER DRIVER FOR LP11/15
47 07012 005037            CLR     @#LPCSR          ;SHUT LP INT. ENABLE
         177514
48 07016 012701            MOV     #1,R1            ;TELL CALLER DONE
         000001
49 07022 016700            MOV     LP-2,R0          ;GET TCBP
         177374
50 07026                   CALL    SEND15           ;TELL CALLER DONE
   07026 004767            JSR     PC,SEND15
         174300
51 07032            LPXT:
52 07032 052767            BIS     #340,PS          ;INHIBIT INTERRUPTS
         000340
         170736
53 07040 005067            CLR     LP-2             ;CLEAR BUSY(IDLE) FLAG
         177356
54 07044 005067            CLR     LP-4
         177350
55 07050 012703            MOV     #LP,R3           ;DEQUEUE ANOTHER REQUEST IF ANY
         006424
56 07054 012701            MOV     #LP.LH,R1        ;    IN THIS DRIVERS DEQUE.
         001430
57 07060 000167            JMP     DEQU
         174122
58                  ;
59                  ;
60                  ;
61                  ;              SUBROUTINE TO FIELD CLOCK COUNT-DOWN
62                  ;
63                  ;
64 07064 005767 LPCHK:     TST     LP-2             ;HAVE WE BEEN DISABLED
         177332
65 07070 001427            BEQ     10$              ;IF YES, EXIT, LEAVING CLOCK DISABLED
66 07072 005737            TST     @#LPCSR          ;ERROR FIXED
         177514
67 07076 100422            BMI     7$               ;MINUS=NO.RESTART 2 SEC. TIMEOUT
68 07100 012702            MOV     #LPTCOD*2,R2     ;SCAN ATL FOR OUR NODE
         000010
69 07104 016201            MOV     ATLNP(R2),R1
         001140
70 07110 012767            MOV     #LP,LP-12        ;RESTART AT BEGINNING OF REQ.
         006424
         177274
71 07116 042761            BIC     #17,A.TS(R1)     ;R1 POINTS TO OUR NODE, MAKE RUNNABLE
         000017
         000006
72 07124 012761            MOV     #LP-26,A.SP(R1) ;SET UP STACK POINTER
         006376
         000004
73 07132 006202            ASR     R2               ;MAKE BYTE ADDRESSING
74 07134 116267            MOVB    LEVEL(R2),LP-10 ;SET UP PS
         001121
         177252
75 07142 000402            BR      10$
76 07144 012710 7$:        MOV     #170,(R0)        ;R0 POINTS TO TIMER ENTRY
         000170
77 07150 000207 10$:       RTS     PC               ;RETURNS TO CLOCK
78                  ;
79 07152 000000 LPBUFF:  .WORD   0                ;BUFFER POINTER
80 07154 000000 LPBTCT:  .WORD   0                ;BYTE COUNT
81 07156 000000 LPTAB:   .WORD   0                ;TAB LOCATION
82 07160    000 LPEOL:   .BYTE   0                ;0 IF IMAGE, 15 IF ASCII
83 07161    000 LPXTR:   .BYTE   0                ;MAKE EVEN
84                  ;
85                         .ENDC
86                  ;
```

Figure 4-3
UNICHANNEL LP Driver (cont.)

4.7.2.1 Code Sections - The program code section of a device driver is composed of three or four of the following subsections (refer to Figure 4-3).[1]

1. Equates, device locations, etc. (Page 29, lines 7-14).
2. Initialization and I/O request section (Page 29, lines 1-73); used to set up and initiate a device operation.
3. Interrupt section, used to respond to the completion of a device operation and to check for errors (Page 30, lines 1-59).
4. An optional clock wake-up section; used to check the correction on an error condition and either retry the offending operation or set another wake-up call (Page 30, lines 60-86).

4.7.2.2 Task Entry--Initialization - When the task is initially called, the user stack area is reset. Execution normally begins at the first location of the program code. At this point, all general purpose registers are available for use by the task. If the task is interrupted by a higher priority task before completing the request, execution will resume at the point of interruption when program control is returned. Various steps in device driver (Figure 4-3) initialization include:[1]

1. Clearing out any pending timer requests (if the task uses wakeup services). (Page 29, line 42).
2. Setting up a pointer to the data buffer and relocating the pointer value if it comes from the PDP-15 (Page 29, lines 43-49).
3. Various device dependent operations (Page 29, lines 50-68).
4. Start up the device (Page 29, line 69).
5. Exit in a WAIT state (Page 29, line 70) until reawakened by an interrupt (see Section 4.7.2.4).

4.7.2.3 Interrupt Processing - An interrupt transfers control to the device driver interrupt section at priority 7. Interrupt processing (Figure 4-3) is composed of the following steps:

1. Disable the device interrupt (Page 30, line 4)
2. Save the interrupted task registers switch stacks and drop down to the task's actual priority as specified in the LEVEL table. This is all accomplished by a JSR R0, R.SAVE (Page 30, lines 5 and 6).
3. Test the task busy idle switch to see if the request has been cancelled (Page 30, lines 7 and 8). If it was cancelled, use the normal DEQU exit without sending a completion message to the caller (see Section 4.7.2.4).

[1] Page number refers to the page number at the top of the PIREX listing.

4. Perform task interrupt processing and error checking (Page 30, lines 10-36).

5. If a correctable error is detected, set the error code in the **DEVST table.** This error code should indicate a correctable error. The DEQU1 return should be used in conjunction with a clock wake up call to allow automatic retry of the operation (Page 30, lines 40-43). See Section 4.7.2.4 for information on **DEQU1** and Section 4.7.3 for information on the timed wake-up.

6. If a fatal error occurs, the event variable should be set to indicate this eror.

7. If the operation was successfully completed, use the normal exit procedure described in Section 4.7.2.4 (Page 30, lines 45-57).

4.7.2.4 Exit Techniques - When a task has finished execution, it can exit by issuing the SEXIT macro (exit and change state of task to "s").

```
.MACRO SEXIT s

IOT

.BYTE 0,s

.ENDM
```

The SEXIT macro allows a task to change status to state "s" after exiting. A task state of "0" indicates the task is runnable, a state of "2" indicates a wait state, and a state of "4" indicates a stop state with removal of the ATL node. Task states must always be an even number since they are used to compute a word index in the PDP-11.

There are actually three modes in which a task may exit. In the first **mode,** used on completion of a request, before a task exits. it must:

1. Zero the busy/idle switch.

2. Set the caller's Event·Variable to indicate the nature of task completion and send an optional interrupt to the PDP-15 or the PDP-11.

3. Dequeue a request from its deque and process it if found; otherwise exit.

Before a task can begin the three previously mentioned steps, it must be executing at level 7 (the highest priority level in the PDP-11). As an example, assuming a task name is "XR" (the first executable instruction of every task has the task name as its label), then the following program code would accomplish the three necessary steps:

```
BIS #340, @#PS;INHIBIT INTERRUPTS

MOV #?,R1      ;SET CALLER'S EV TO ? (APPROPRIATE VALUE)
```

```
  CALL SEND15   ;    AND SEND CALLER
                ;    AN OPTIONAL INTERRUPT
                ;    TELLING THE REQUESTOR THAT THE
                ;    REQUEST HAS BEEN PROCESSED.
                ;    (A COMPLETE LIST OF EVENT)
                ;    VARIABLE SETTINGS MAY BE
                ;    FOUND IN SECTION 3.2.5.4

 BIS #340, @#PS;INHIBIT INTERRUPTS,
 CLR XR-2      ;CLEAR THE BUSY/IDLE SWITCH ("XR" is the tag
                    associated with the first executable
                    instruction in the task program code.)
 CLR XR-4
 MOV #XR,R3    ;DEQUEUE ANOTHER REQUEST IF ANY
 MOV #XR,LH,R1
 JMP DEQU      ;    EXISTS IN THIS TASK'S DEQUE
               ;    IF A REQUEST EXISTS, NO RETURN
               ;    IS MADE FROM ROUTINE DEQU
               ;    AND THE REQUEST IS AUTOMATICALLY
               ;    REMOVED AND PROCESSED AS IF IT
               ;    WERE JUST RECEIVED WHEN THE
               ;    TASK WAS IDLE.
```

This first method is used in the interrupt section upon successful completion of a request. The second method is one where the task exits from the initialization section (Figure 4-3, Page 30, lines 46-57) in a wait state using the SEXIT macro, and an interrupt routine or other task will complete the previously mentioned three steps at a later time. A device driver is typically exited in this way (Figure 4-3, Page 30, line 75). The initial section of the device driver is used to set up the device controller and begin the I/O operation. The task will then exit in a wait state until the I/O is complete, the interrupt section is called, the device is shut down, and the previously mentioned three steps are done informing the requestor that the I/O operation has been completed.

The third method of exiting is one used either when a recoverable error is detected in the interrupt section of a driver and the intention is to exit and wait for an error recovery or when another I/O request is issued in the interrupt section and another interrupt is expected. This exit through DEQU1 does not cause the dequeuing of pending requests but simply places the task in a WAIT state. This method assumes that an R.SAVE has been performed upon entry to the interrupt process routine. The required code to use this exit is:

```
 JMP DEQU1
```

No registers are preserved by this exit. Control is returned to the interrupt section upon occurrence of an interrupt or via the clock routine wakeup, to a location chosen by the clock set up section. (Figure 4-3, Page 30, line 43).

4.7.3 Timed Wakeup

In the design of a device driver it is useful to include features that eliminate operator intervention whenever possible.

For instance, in the example of the PIREX Line Printer Task, an OFF Line condition is handled by retrying the printing every two seconds until successful. This is accomplished by using the wakeup feature of the Clock Task. This is done by simply placing the return address and the time delay into the Clock Table "CLTABL" (See Section 3.3.4) Figure 4-3, Page 30, lines 40-41) and the exits using the DEQU1 type exit.

When the wakeup call occurs, the clock wakeup subsection specified by the return address will be invoked. In this subsection:

1. Test the task IDLE/BUSY switch to see if the task has been shut down. If shut down, a RTS PC return to the Clock Task is in order. (Page 30, lines 64-65, 77)

2. Determine if the error has been corrected. If not, reset the timer and RTS PC to the Clock Task. (Page 30, lines 66-67, 76-77).

3. If the error has been corrected, reprocess the original TCB request and return to the Clock Task. (Page 30, lines 68-75). This will cause PIREX to retry the TCB.

4.7.4 Assembly and Testing

4.7.4.1 Assembly and Loading - New PIREX device dirver should be assembled as a part of the PIREX monitor. Background tasks may be assembled separately.

In the background task case, the user should construct a PDP-15 program to load the background task binary into PDP-15 memory. The PDP-15 program must then issue a CONNECT Directive (Section) To start the task, if the task is to execute in PDP-11 local memory, two additional steps are required:

1. Issue a local memory size directive to determine if there is enough local memory to accomodate the new task.

2. Issue a CONNECT directive (assuming there was enough room in local memory for the task).

3. After issuing the CONNECT directive, use the initial portion of the PDP-11 code to move the remainder of the task into the local memory starting at the first free location.

4.7.4.2 Testing - Since the typical UNICHANNEL system does not have a terminal device attached to the PDP-11 processor, the only debugging facility present is the console indicators on the PDP-11. An additional aid is the UDMP11 paper tape provided with all UC15 DOS-15 systems. This program provides a destructive dumping facility that recovers the entire state of the PDP-11 LOCAL memory and dumps it into the LP11/LS11/LV11 Printer. (Note: The UDMP11 program is an unsupported package that can only be used on systems with a printer device on the PDP-11 UNICHANNEL Processor). For tasks executing in the common memory, the traditional ↑Q-DUMP feature of the DOS-15 monitor should be used.

CHAPTER 5

SPOOLER DESIGN AND THEORY OF OPERATION

5.1 INTRODUCTION

This chapter discusses the design concepts of the UNICHANNEL-15 SPOOLER software and its theory of operation. This information is provided to enable the user to understand the SPOOLER software in order to add new SPOOLED tasks or to modify existing software. The actual modification process is described in Chapter 6. Flowcharts are provided whenever it is necessary.

5.2 OVERVIEW

5.2.1 SPOOLER

The word 'spool' and 'spooling' originated in the textile industry. During thread manufacture, the threads are wound on small spools by first storing them on large spindles and then transferring them onto small spools. This entire process is called spooling. In the computing industry, the term spooling is used to describe the process of collecting and storing data on a large high-speed medium and controlling the flow of this data to slow speed devices. The "SPOOLER" is a distinct piece of software that controls the entire spooling operations. Spooling permits data flow between a data source and a data sink to proceed at independent rates. This feature gives the user greater computing power and faster turn-around time because of better system resource utilization under an integrated operating system.

5.2.2 UNICHANNEL-15 Spooler

In the UNICHANNEL-15 system, spooling is achieved by using the dual processing capability of the system. The two processors, PDP-15 and PDP-11, operate in the Master and Slave mode respectively. The Slave processor (PDP-11) controls the entire spooling operation. Data to be spooled is supplied by either the master processor (PDP-15), or by tasks running under PIREX. Spooled data is stored on a disk cartridge. The Line Printer, Card Reader, and the Incremental Plotter, all being UNIBUS devices, are supported by the UNICHANNEL-15 spooler.

5.3 SPOOLER DESIGN

The UNICHANNEL-15 SPOOLER is based on a simple design. Spooling of data is done through the RK05 disk. A contiguous portion of disk is allocated via SPLGEN for this purpose by the operating system on the PDP-15. The starting block number and the size in terms of number of blocks is conveyed to the SPOOLER when it is issued the 'BEGIN' directive. The SPOOLER allocates and deallocates this space on the disk through a BITMAP it maintains. The spooling and despooling operations of every task are performed through a central "TABLE", in which every spooled task has a slot. Against each slot there are several entries used to keep track of the data during spooling and despooling. Provisions are made in the SPOOLER to permit spooling of data regardless of the number of blocks occupied in the spool space and the number of buffers in the SPOOLER provided despooling operations are going on. This prevents system lockout. All the data blocks on the disk belonging to a spooled task are linked together by forward pointers stored in the last word (377_8) of each data block. The end of data in a block is indicated by a zero word. Records are assumed to be less than 374_8 words in size. The last block in a spooled file has a pointer to the previous file's last block in word '1'$_8$ or a -1 if there is no active previous file, if the last spooled file has not yet been despooled. Also the last block in a spooled file contains an end of file indicator in word '376_8' of the data block. Sections 5.3 and 5.4 describe the static layout of the spooler. The dynamic layout is described in Section 5.5.

5.4 SPOOLER COMPONENTS

The following are the major components of the SPOOLER software:

1. request dispatcher
2. directive processing routine
3. task call service routine
4. device interrupt dispatcher
5. device interrupt service routine
6. utility routines
7. buffers, TABLE, BITMAP, TCBs

A brief description of each of the above components follows.

5.4.1 Request Dispatcher

This routine dispatches (routes) all requests made by the SPOOLER and requests to the spooled tasks. This is done by using the TCN in word '1' of the TCB. The dispatcher transfers control to the appropriate directive processing routines, in the case of spooler requests and to the task call service routine, in the case of requests to spooled tasks.

5.4.2 Directive Processing Routines

These routines process directives issued to the SPOOLER to control spooling operations. The basic operations are "BEGIN" spooling and "END" spooling. These routines may initialize switches, TABLE, BITMAP, pointers, buffers, set up TCB, start tasks, stop tasks, ... etc.

5.4.3 Task Call Service Routines

A task call service routine processes requests addressed to tasks running under PIREX. It spools data onto disk in case of output tasks, and for input tasks it despools the data from disk. Output tasks buffer data from several requests into blocks and transfer the blocks to disk when full. Input tasks read into core, data blocks stored on disk, and unpack the data into the requestor's buffer. Task Call Service Routines update the TABLE, pointers, and switches, and use the utility routines present in the SPOOLER to write or read a block onto or from the disk, get or give a buffer, get or give a TCB, etc. (Refer to Figure 5-2.)

5.4.4 Device Interrupt Dispatcher

All interrupts from devices interacting with the SPOOLER are dispatched by this routine to the appropriate service routines. This is done by using the TCN of the requestor for that task request present in word '13_8' of the TCB.

5.4.5 Device Interrupt Service Routines

These routines handle completion of I/O requests from devices. They supplement the driver routines present in PIREX as in the device handlers. Besides the disk interrupt service routine, each spooled task has its own interrupt service routine. The disk interrupt service routine is made up of the "read interrupt processor" and the "write interrupt processor." These are in turn made up of routines handling read/write operation for each specific spooled task. The interrupt service routine of a spooled task controls the despooling operation for output tasks and the spooling operation for input tasks. These operations are driven by the table entries which determine the end of the operation. Device interrupt service routines update the TABLE, pointers, switches and use the utility routines to write or read a block onto or from the disk, get or give a buffer, get or give a TCB, etc.

5.4.6 Utility Routines

Each SPOL11 utility routine performs a specific function. They are:

FINDBK	Find a free block on disk and set its bit in the BITMAP Table (protected).[1]

[1] Protected routines are those run at priority level 7.

FREEBK	Free the block indicated and reset its bit in the BITMAP Table.
GETBUF	Get an unused buffer from the buffer pool(protected).[1]
GIVBUF	Give the used buffer back to the buffer pool.
GETRKT	Get a disk TCB from the Disk TCB pool.
GIVRKT	Give back the TCB to the Disk TCB pool.
GETBLK	Read a block from disk.
PUTBLK	Put a block on disk.
GETPUT	Get or put a block on disk.
RESTRQ	Reissue a delayed request.
DEQREQ	Tell requestor that a request is done and dequeue the next request, if any.

5.4.7 Buffers, TABLE, BITMAP, TCBs

Buffers — The SPOOLER maintains a pool of buffers in a doubly linked list for general use. Buffers are used to pack data into blocks to be written onto disk (by output task call service routines) and to unpack data from data blocks read from disk into requestor buffers (by input task call service routines).

TABLE — The entire spooling and despooling operation of all tasks is controlled by entries in this table. Every spooled task has the following entries:

WORD 0:	DEV	device mnemonic (set by the BEGIN routine)
WORD 1:	CBN	current despooling block number (set by the despooler).
WORD 2:	CRP	current record pointer (set by the despooler).
WORD 3:	NBN	next despooling block number (set by the despooler).
WORD 4:	LSB	last spooled block number (set by the spooler).
WORD 5:	LFB	last spooled file block number (set by the spooler).

[1] Protected routines are those run at priority level 7.

BITMAP A record of availability of disk spooling space is maintained in the BITMAP. Corresponding to each disk block reserved for spooling is a bit which is 'ON' if the block is in use and 'OFF' if free.

TCBs Buffered blocks of data are read from disk and written onto disk using TCBs. Output spooled tasks despool data to devices using TCBs and input spooled task spool data from devices using TCBs.

5.5 THEORY OF OPERATION

This section will describe in detail the flow of control in the SPOOLER among the above components. To illustrate this process, the spooling and despooling operations of the Line Printer will be discussed. The routines in the SPOOLER listing (Figure 5-1) are broken up into logic boxes and referenced by line numbers.

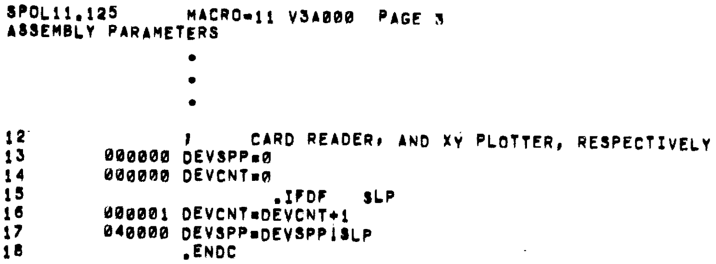
5.5.1 SPOOLER Startup

Spooling under an operating system on the PDP-15 is accomplished as follows. The SPOOLER task should be added to PIREX, by reading it into local memory and connecting it at run time via SPOOL (SPOL15). As supplied by DEC, the SPOOLER is a separate binary program from PIREX. A special PDP-15 program referred to as the system/SPOOLER interface (SPOL15) is responsible for loading the SPOOLER into PDP-11 local memory and then issuing requests to PIREX to connect the SPOOLER and then begin its operation.

Subsequently when PIREX schedules the SPOOLER task to run, the "BEGIN" request is processed. On gaining control, the 'request dispatcher' transfers control to the 'BEGIN' routine. The first time the SPOOLER processes a directive it also executes a once only section of code, which builds a central address table. This table contains addresses of frequently addressed locations in the SPOOLER and is necessary since the SPOOLER is coded in Position Independent Code (PIC) and thus can be loaded anywhere in the PDP-11 memory. SPOOLER is coded in PIC to permit additional tasks to be added to PIREX without necessitating SPOOLER changes. The BEGIN routine performs the following; general startup operations and the specific line printer startup operations (Refer to Figure 5-1):

GENERAL OPERATIONS - BEGIN DIRECTIVE:

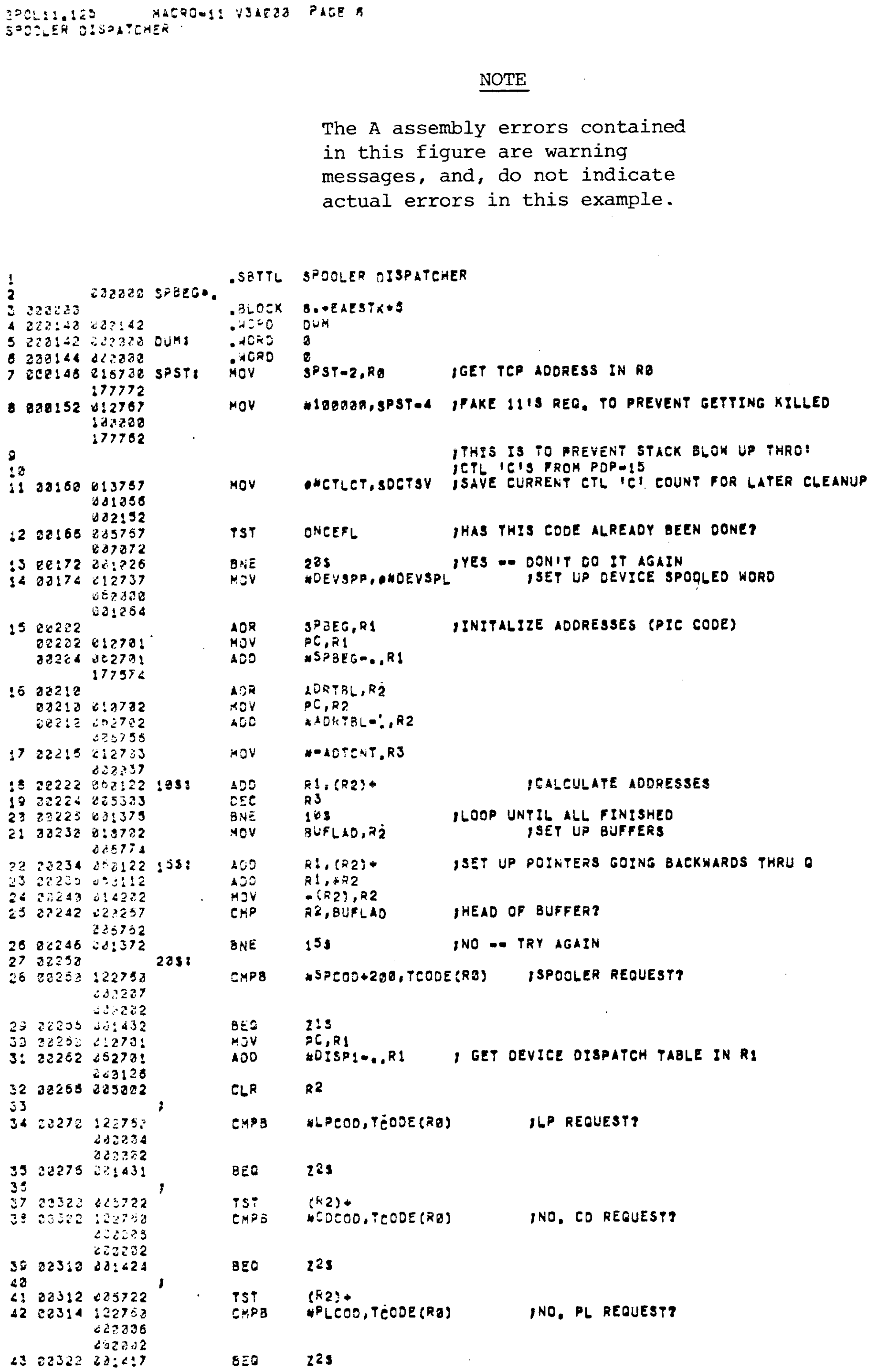
Set up the SOFTWARE INTERRUPT trap address in the PIREX SEND11 table	page 7, lines 9-12
Save the SPOOLER start address in the "disconnect SPOOLER" TCB	line 13
Initialize the FINDBK routine switches and pointers.	lines 15, 38



```
SPOL11.125      MACRO-11 V3A000  PAGE 3
ASSEMBLY PARAMETERS
                 •
                 •
                 •

                 ;      CARD READER, AND XY PLOTTER, RESPECTIVELY
         000000 DEVSPP=0
         000000 DEVCNT=0
                         .IFDF   $LP
         000001 DEVCNT=DEVCNT+1
         040000 DEVSPP=DEVSPP!$LP
                 .ENDC
```

Figure 5-1[1]
UNICHANNEL Spooler Components

[1] This listing is of the V3AØØØ version of SPOL11. V3BØØØ SPOL11 contains several differences. Refer to the DOS-15 V3BØØØ Update Document for a description of the significant new features.



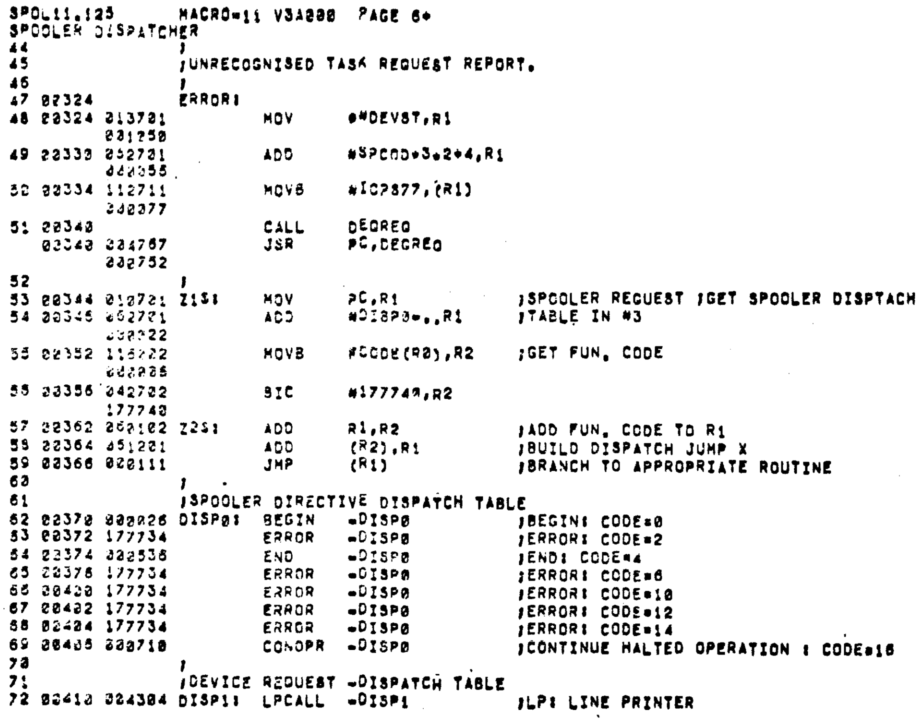
```
SPOL11.123      MACRO-11 V3A020  PAGE 6
SPOOLER DISPATCHER
```

NOTE

The A assembly errors contained in this figure are warning messages, and, do not indicate actual errors in this example.

```
                                .SBTTL  SPOOLER DISPATCHER
           000000 SPBEG=.
  000000                        .BLOCK  8.*EAESTK*5
  000140 000142                 .WORD   DUM
  000142 000000 DUM:            .WORD   0
  000144 000000                 .WORD   0
  000146 016700 SPST:           MOV     SPST-2,R0         ;GET TCP ADDRESS IN R0
         177772
  000152 012767                 MOV     #100000,SPST-4    ;FAKE 11'S REQ. TO PREVENT GETTING KILLED
         100000
         177762
                                                          ;THIS IS TO PREVENT STACK BLOW UP THRO'
                                                          ;CTL 'C'S FROM PDP-15
   00160 013767                 MOV     @#CTLCT,SDCTSV    ;SAVE CURRENT CTL 'C' COUNT FOR LATER CLEANUP
         001056
         002152
   00166 005767                 TST     ONCEFL            ;HAS THIS CODE ALREADY BEEN DONE?
         007072
   00172 001026                 BNE     20$               ;YES -- DON'T DO IT AGAIN
   00174 012737                 MOV     #DEVSPP,@#DEVSPL          ;SET UP DEVICE SPOOLED WORD
         062000
         001264
   00202                        ADR     SPBEG,R1          ;INITALIZE ADDRESSES (PIC CODE)
   00202 012701                 MOV     PC,R1
   00204 062701                 ADD     #SPBEG-.,R1
         177574
   00210                        ADR     ADRTBL,R2
   00210 010702                 MOV     PC,R2
   00212 062702                 ADD     #ADRTBL-.,R2
         005756
   00216 012703                 MOV     #-ADTCNT,R3
         000037
   00222 060122 10$:            ADD     R1,(R2)+                  ;CALCULATE ADDRESSES
   00224 005303                 DEC     R3
   00226 001375                 BNE     10$               ;LOOP UNTIL ALL FINISHED
   00230 016702                 MOV     BUFLAD,R2                 ;SET UP BUFFERS
         005774
   00234 060122 15$:            ADD     R1,(R2)+          ;SET UP POINTERS GOING BACKWARDS THRU Q
   00236 060112                 ADD     R1,@R2
   00240 014202                 MOV     -(R2),R2
   00242 020267                 CMP     R2,BUFLAD         ;HEAD OF BUFFER?
         005762
   00246 001372                 BNE     15$               ;NO -- TRY AGAIN
   00250               20$:
   00250 122760                 CMPB    #SPCOD+200,TCODE(R0)      ;SPOOLER REQUEST?
         000207
         000002
   00256 001432                 BEQ     21$
   00260 012701                 MOV     PC,R1
   00262 062701                 ADD     #DISP1-.,R1       ; GET DEVICE DISPATCH TABLE IN R1
         003126
   00266 005002                 CLR     R2
                       ;
   00270 122760                 CMPB    #LPCOD,TCODE(R0)          ;LP REQUEST?
         000004
         000002
   00276 001431                 BEQ     22$
                       ;
   00300 005722                 TST     (R2)+
   00302 122760                 CMPB    #CDCOD,TCODE(R0)          ;NO, CD REQUEST?
         000005
         000002
   00310 001424                 BEQ     22$
                       ;
   00312 005722                 TST     (R2)+
   00314 122760                 CMPB    #PLCOD,TCODE(R0)          ;NO, PL REQUEST?
         000006
         000002
   00322 001417                 BEQ     22$
```

Figure 5-1
UNICHANNEL Spooler Components (Cont.)

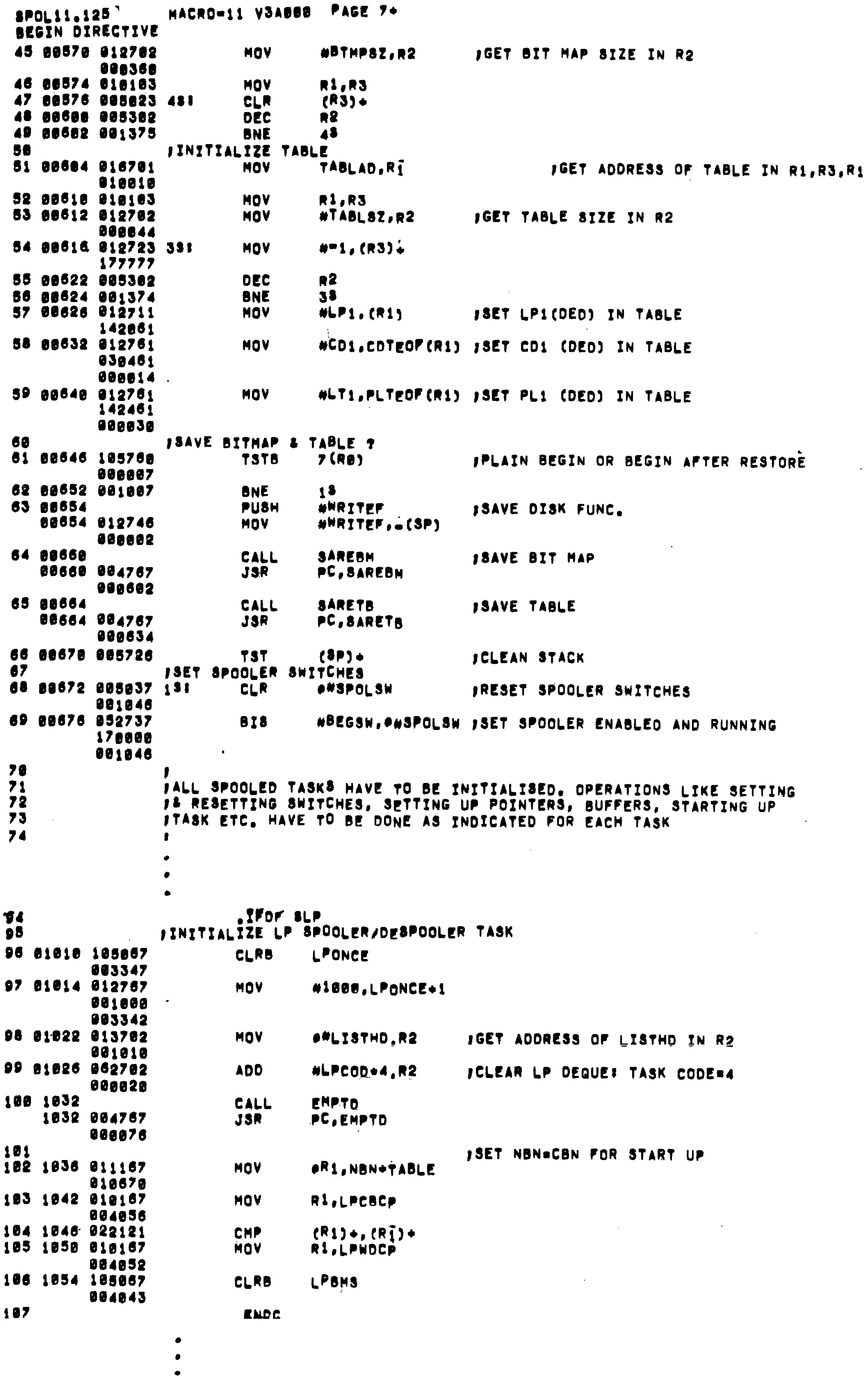


```
SPOL11.125      MACRO-11 V3A000  PAGE 6+
SPOOLER DISPATCHER
44                       ;
45                       ;UNRECOGNISED TASK REQUEST REPORT.
46                       ;
47 02324                 ERROR:
48 02324 013701                  MOV     @#DEVST,R1
         001250
49 02330 062701                  ADD     #SPCOD+3.2+4,R1
         000355
50 02334 112711                  MOVB    #ICP877,(R1)
         000377
51 02340                         CALL    DEQREQ
   02340 004767                  JSR     PC,DEQREQ
         002752
52                       ;
53 02344 010701 Z1$:             MOV     PC,R1              ;SPOOLER REQUEST ;GET SPOOLER DISPTACH
54 02346 062701                  ADD     #DISP0-.,R1        ;TABLE IN #3
         000022
55 02352 116022                  MOVB    #CODE(R0),R2       ;GET FUN. CODE
         000005
56 02356 042702                  BIC     #177740,R2
         177740
57 02362 060102 Z2$:             ADD     R1,R2              ;ADD FUN. CODE TO R1
58 02364 061201                  ADD     (R2),R1            ;BUILD DISPATCH JUMP X
59 02366 000111                  JMP     (R1)               ;BRANCH TO APPROPRIATE ROUTINE
60                       ;
61                       ;SPOOLER DIRECTIVE DISPATCH TABLE
62 02370 000026 DISP0:   BEGIN   -DISP0             ;BEGIN: CODE=0
63 02372 177734          ERROR   -DISP0             ;ERROR: CODE=2
64 02374 000536          END     -DISP0             ;END: CODE=4
65 02376 177734          ERROR   -DISP0             ;ERROR: CODE=6
66 02400 177734          ERROR   -DISP0             ;ERROR: CODE=10
67 02402 177734          ERROR   -DISP0             ;ERROR: CODE=12
68 02404 177734          ERROR   -DISP0             ;ERROR: CODE=14
69 02406 000710          CONOPR  -DISP0             ;CONTINUE HALTED OPERATION : CODE=16
70                       ;
71                       ;DEVICE REQUEST -DISPATCH TABLE
72 02410 004304 DISP1:   LPCALL  -DISP1             ;LP: LINE PRINTER
```

Figure 5-1
UNICHANNEL Spooler Components (Cont.)

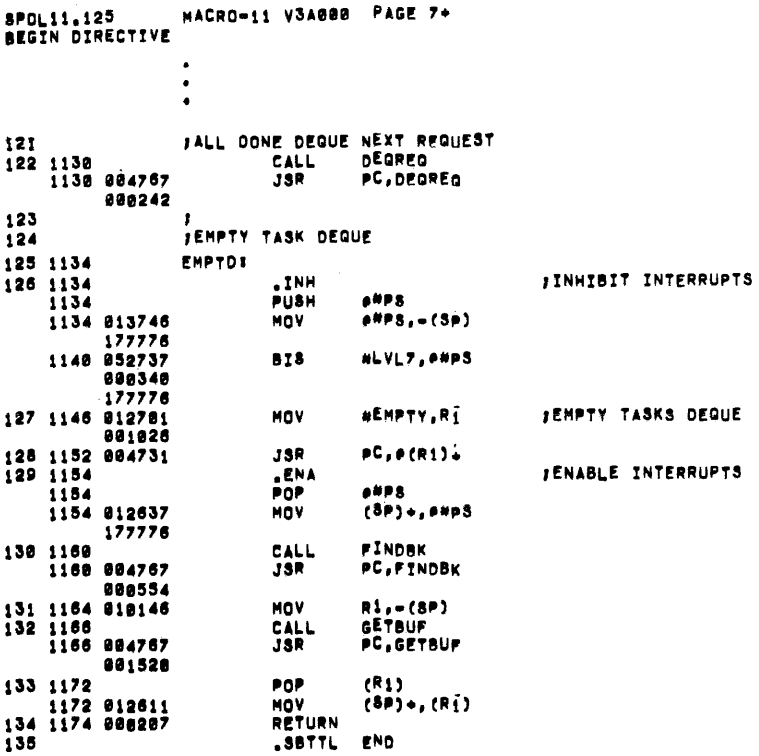
```
SPOL11.125       MACRO-11 V3A000  PAGE 7
BEGIN DIRECTIVE

1                               .SBTTL  BEGIN DIRECTIVE
2                       ;
3                       ;THIS ROUTINE STARTS ALL SPOOLING OPERATIONS. SWITCHES, CONTROL REGISTERS
4                       ;ETC. ARE SET . THE BUFFER POOL, TCB POINTERS, BITMAP, TABLE ETC. ARE
5                       ;SET UP;BITMAP & TABLE ARE SAVED ON DISK(FOR BACKUP OPERATIONS). EACH
6                       ;INDIVIDUAL SPOOLED TASK IS THEN INITIALIZED & STARTED UP IF NECESSARY
7                       ;
8                       ;
9 000416 010701 BEGIN:  MOV     PC,R1                   ;GET ADDRESS OF DEVINT IN R1
10 00420 062701         ADD     #DEVINT-.,R1
         002716
11 00424 013702         MOV     @#SEND11,R2
         001002
12 00430 010162         MOV     R1,SPCOD+2(R2)                  ;SET SEND11 ADDRESS IN PIREX
         000016
13 00434 016067         MOV     14(R0),TCBDSA+TCBDIS
         000014
         012240
14                      ; INITIALIZE ALL SWITCHES
15 00442 012767         MOV     #1,CBTPTR
         000001
         001742
16                      ;SET  CONTROL REGS.
17 00450 010701         MOV     PC,R1                   ;GET ADD. OF DUM IN R1
18 00452 062701         ADD     #DUM-.,R1
         177470
19 00456                PUSH    R1                      ;SAVE ON STACK
   00456 010146         MOV     R1,-(SP)
20 00460                POP     -(R1)                   ; SET SPOOLER CONTROL REG.!!
   00460 012641         MOV     (SP)+,-(R1)
21                      ;SETUP BUFFER POOL
22                      ;INITIALIZE RK TCB POINTERS
23 00462 016701         MOV     RKCAD,R1                        ;GET RKTCBP ADD. IN R1
         010116
24 00466 010702         MOV     PC,R2                   ;GET TCBR01 ADD. IN R2
25 00470 062702         ADD     #TCBST-.,R2
         011474
26 00474 012703         MOV     #TCBCT,R3               ;SETUP TCBCT TCB'S
         000014
27 00500 010221 2$:    MOV     R2,(R1)+                ;SET TCBRK1 POINTER
28 00502 062702         ADD     #30,R2                  ;BUMP R2 TO TCBRK2
         000030
29 00506 005303         DEC     R3
30 00510 001373         BNE     2$
31                      ;INITIALIZE BITMAP
32 00512                PUSH    NBK(R0)                 ;GET SIZE OF SPOOLER AREA NUMBER
   00512 016046         MOV     NBK(R0),-(SP)
         000012
33 00516 006216         ASR     (SP)                    ;COMPUTE SIZE OF BIT MAP
34 00520 006216         ASR     (SP)                    ;SIZE=NUMBK/8+2
35 00522 006216         ASR     (SP)
36 00524 042716         BIC     #1,(SP)                 ;GET EVEN NUMBER
         000001
37 00530 162716         SUB     #2,(SP)
         000002
38 00534 016767         MOV     BTMPAD,CWDPTR           ;RESET CWDPTR
         010054
         001652
39 00542 016701         MOV     BTMPAD,R1               ;(BR0112, TEMP FIX)
         010046
40 00546 062601         ADD     (SP)+,R1                ;ADD OFFSET TO END
41 00550 010167         MOV     R1,BTMPED               ;SET UP BTMPED
         011136
42 00554 016701         MOV     STBKNA,R1                       ;GET ADDRESS OF STBKNM-4 IS R1
         010036
43 00560 016021         MOV     SBN(R0),(R1)+           ;SET STARTING LOCK #
         000010
44 00564 016021         MOV     NBK(R0),(R1)+           ;SET NUMBER OF BLOCKS
         000012
```

Figure 5-1
UNICHANNEL Spooler Components (Cont.)



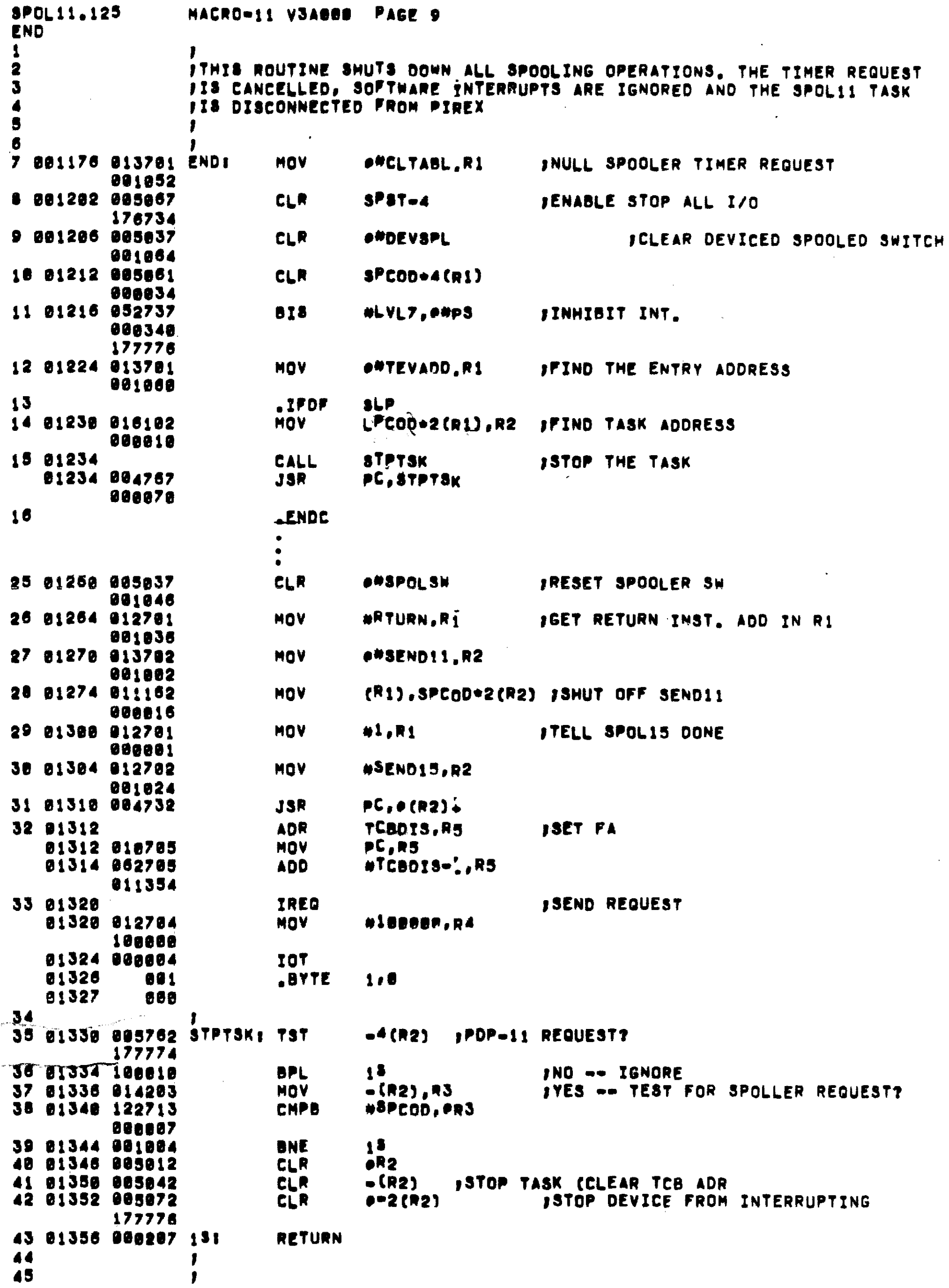
```
SPOL11.125      MACRO-11 V3A000  PAGE 7+
BEGIN DIRECTIVE
45 00570 012702         MOV     #BTMPSZ,R2        ;GET BIT MAP SIZE IN R2
         000360
46 00574 010103         MOV     R1,R3
47 00576 005023 4$:     CLR     (R3)+
48 00600 005302         DEC     R2
49 00602 001375         BNE     4$
50                ;INITIALIZE TABLE
51 00604 016701         MOV     TABLAD,R1                   ;GET ADDRESS OF TABLE IN R1,R3,R1
         010010
52 00610 010103         MOV     R1,R3
53 00612 012702         MOV     #TABLSZ,R2        ;GET TABLE SIZE IN R2
         000044
54 00616 012723 3$:     MOV     #-1,(R3)+
         177777
55 00622 005302         DEC     R2
56 00624 001374         BNE     3$
57 00626 012711         MOV     #LP1,(R1)         ;SET LP1(DED) IN TABLE
         142061
58 00632 012761         MOV     #CD1,CDTEOF(R1) ;SET CD1 (DED) IN TABLE
         030461
         000014
59 00640 012761         MOV     #LT1,PLTEOF(R1) ;SET PL1 (DED) IN TABLE
         142461
         000030
60                ;SAVE BITMAP & TABLE ?
61 00646 105760         TSTB    7(R0)             ;PLAIN BEGIN OR BEGIN AFTER RESTORE
         000007
62 00652 001007         BNE     1$
63 00654                PUSH    #WRITEF           ;SAVE DISK FUNC.
   00654 012746         MOV     #WRITEF,-(SP)
         000002
64 00660                CALL    SAREBM            ;SAVE BIT MAP
   00660 004767         JSR     PC,SAREBM
         000602
65 00664                CALL    SARETB            ;SAVE TABLE
   00664 004767         JSR     PC,SARETB
         000634
66 00670 005726         TST     (SP)+             ;CLEAN STACK
67                ;SET SPOOLER SWITCHES
68 00672 005037 1$:     CLR     @#SPOLSW          ;RESET SPOOLER SWITCHES
         001046
69 00676 052737         BIS     #BEGSW,@#SPOLSW ;SET SPOOLER ENABLED AND RUNNING
         170000
         001046
70                ;
71                ;ALL SPOOLED TASKS HAVE TO BE INITIALISED. OPERATIONS LIKE SETTING
72                ;& RESETTING SWITCHES, SETTING UP POINTERS, BUFFERS, STARTING UP
73                ;TASK ETC. HAVE TO BE DONE AS INDICATED FOR EACH TASK
74                ;
                  .
                  .
                  .

94                      .IFDF $LP
95                ;INITIALIZE LP SPOOLER/DESPOOLER TASK
96 01010 105067         CLRB    LPONCE
         003347
97 01014 012767         MOV     #1000,LPONCE+1
         001000
         003342
98 01022 013702         MOV     @#LISTHD,R2       ;GET ADDRESS OF LISTHD IN R2
         001010
99 01026 062702         ADD     #LPCOD*4,R2       ;CLEAR LP DEQUE; TASK CODE=4
         000020
100 1032                CALL    EMPTD
    1032 004767         JSR     PC,EMPTD
         000076
101                                               ;SET NBN=CBN FOR START UP
102 1036 011167         MOV     @R1,NBN+TABLE
         010670
103 1042 010167         MOV     R1,LPCBCP
         004056
104 1046 022121         CMP     (R1)+,(R1)+
105 1050 010167         MOV     R1,LPWDCP
         004052
106 1054 105067         CLRB    LPBMS
         004043
107                      .ENDC

                   .
                   .
                   .
```

Figure 5-1
UNICHANNEL Spooler Components (Cont.)



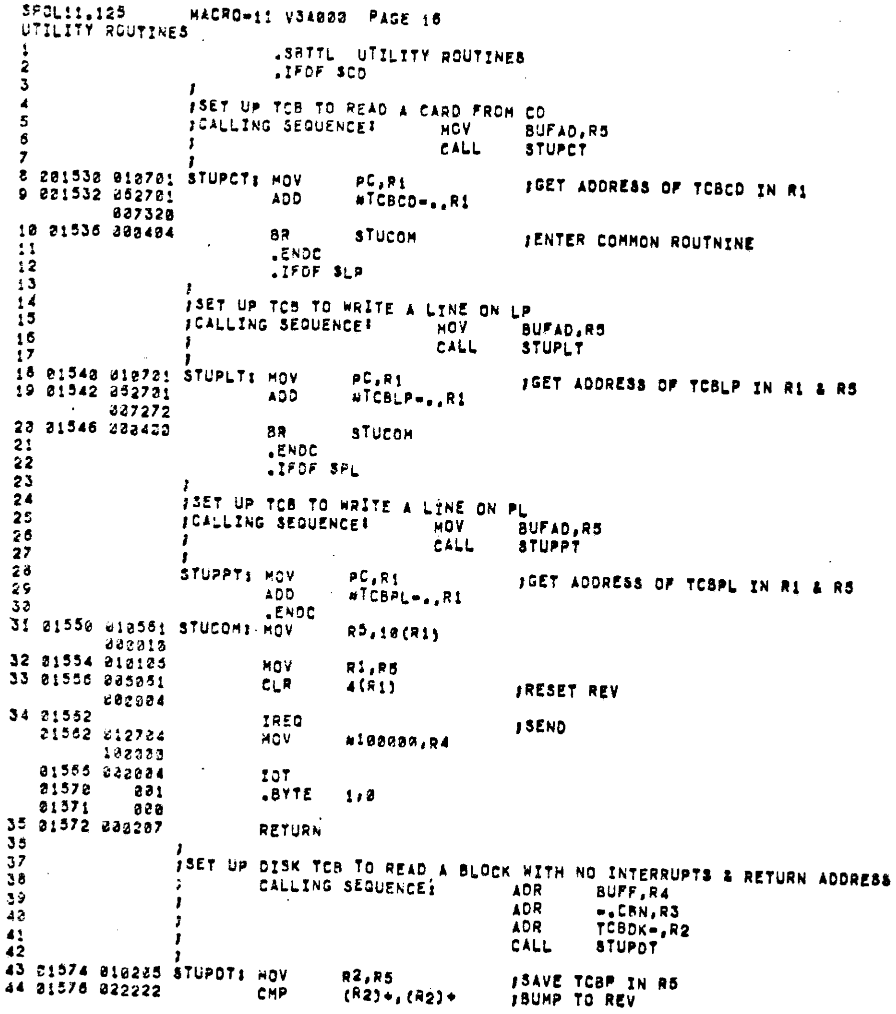
```
SPOL11.125        MACRO-11 V3A000  PAGE 7+
BEGIN DIRECTIVE
                  .
                  .
                  .

121               ;ALL DONE DEQUE NEXT REQUEST
122 1130                 CALL     DEQREQ
    1130 004767          JSR      PC,DEQREQ
         000242
123               ;
124               ;EMPTY TASK DEQUE
125 1134          EMPTD:
126 1134                 .INH                        ;INHIBIT INTERRUPTS
    1134                 PUSH     @#PS
    1134 013746          MOV      @#PS,-(SP)
         177776
    1140 052737          BIS      #LVL7,@#PS
         000340
         177776
127 1146 012701          MOV      #EMPTY,R1          ;EMPTY TASKS DEQUE
         001026
128 1152 004731          JSR      PC,@(R1)+
129 1154                 .ENA                        ;ENABLE INTERRUPTS
    1154                 POP      @#PS
    1154 012637          MOV      (SP)+,@#PS
         177776
130 1160                 CALL     FINDBK
    1160 004767          JSR      PC,FINDBK
         000554
131 1164 010146          MOV      R1,-(SP)
132 1166                 CALL     GETBUF
    1166 004767          JSR      PC,GETBUF
         001520
133 1172                 POP      (R1)
    1172 012611          MOV      (SP)+,(R1)
134 1174 000207          RETURN
135                      .SBTTL   END
```

Figure 5-1
UNICHANNEL Spooler Components (Cont.)



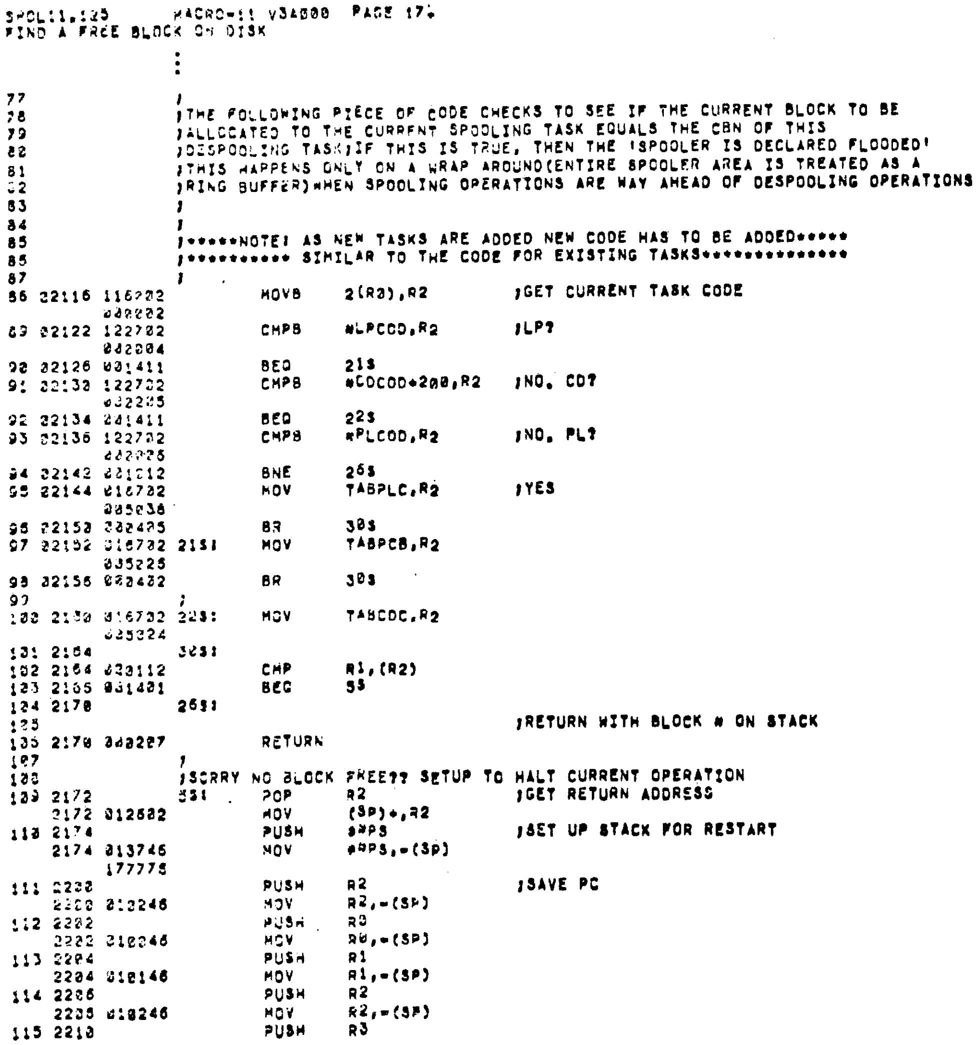
```
SPOL11.125        MACRO-11 V3A000  PAGE 9
END
1                       ;
2                       ;THIS ROUTINE SHUTS DOWN ALL SPOOLING OPERATIONS. THE TIMER REQUEST
3                       ;IS CANCELLED, SOFTWARE INTERRUPTS ARE IGNORED AND THE SPOL11 TASK
4                       ;IS DISCONNECTED FROM PIREX
5                       ;
6                       ;
7 001176 013701 END:    MOV     @#CLTABL,R1      ;NULL SPOOLER TIMER REQUEST
         001052
8 001202 005067         CLR     SPST-4           ;ENABLE STOP ALL I/O
         176734
9 001206 005037         CLR     @#DEVSPL                 ;CLEAR DEVICED SPOOLED SWITCH
         001064
10 01212 005061         CLR     SPCOD+4(R1)
         000034
11 01216 052737         BIS     #LVL7,@#PS       ;INHIBIT INT.
         000340
         177776
12 01224 013701         MOV     @#TEVADD,R1      ;FIND THE ENTRY ADDRESS
         001060
13                      .IFDF   $LP
14 01230 016102         MOV     LPCOD+2(R1),R2  ;FIND TASK ADDRESS
         000010
15 01234                CALL    STPTSK           ;STOP THE TASK
   01234 004767         JSR     PC,STPTSK
         000070
16                      .ENDC
                        .
                        .
                        .
25 01250 005037         CLR     @#SPOLSW         ;RESET SPOOLER SW
         001046
26 01264 012701         MOV     #RTURN,R1        ;GET RETURN INST. ADD IN R1
         001036
27 01270 013702         MOV     @#SEND11,R2
         001002
28 01274 011162         MOV     (R1),SPCOD+2(R2) ;SHUT OFF SEND11
         000016
29 01300 012701         MOV     #1,R1            ;TELL SPOL15 DONE
         000001
30 01304 012702         MOV     #SEND15,R2
         001024
31 01310 004732         JSR     PC,@(R2)+
32 01312                ADR     TCBDIS,R5        ;SET FA
   01312 010705         MOV     PC,R5
   01314 062705         ADD     #TCBDIS-.,R5
         011354
33 01320                IREQ                     ;SEND REQUEST
   01320 012704         MOV     #100000,R4
         100000
   01324 000004         IOT
   01326    001         .BYTE   1,0
   01327    000
34                      ;
35 01330 005762 STPTSK: TST     -4(R2)  ;PDP-11 REQUEST?
         177774
36 01334 100010         BPL     1$               ;NO -- IGNORE
37 01336 014203         MOV     -(R2),R3         ;YES -- TEST FOR SPOLLER REQUEST?
38 01340 122713         CMPB    #SPCOD,@R3
         000007
39 01344 001004         BNE     1$
40 01346 005012         CLR     @R2
41 01350 005042         CLR     -(R2)   ;STOP TASK (CLEAR TCB ADR
42 01352 005072         CLR     @-2(R2)          ;STOP DEVICE FROM INTERRUPTING
         177776
43 01356 000207 1$:     RETURN
44                      ;
45                      ;
```

Figure 5-1
UNICHANNEL Spooler Components (Cont.)



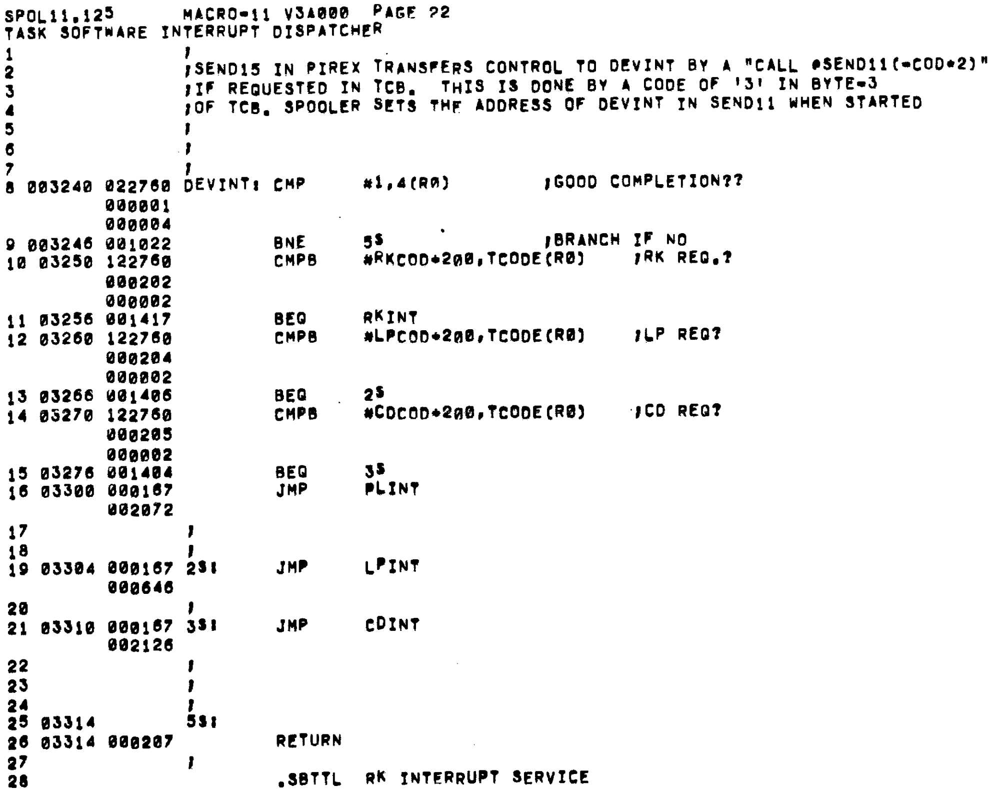
```
SPOL11.125        MACRO-11 V3A000  PAGE 16
UTILITY ROUTINES
                               .SBTTL  UTILITY ROUTINES
                               .IFDF $CD
                        ;
                        ;SET UP TCB TO READ A CARD FROM CD
                        ;CALLING SEQUENCE:       MOV      BUFAD,R5
                        ;                        CALL     STUPCT
                        ;
  001530 010701  STUPCT: MOV      PC,R1              ;GET ADDRESS OF TCBCD IN R1
  001532 062701          ADD      #TCBCD-.,R1
         007320
   01536 000404          BR       STUCOM             ;ENTER COMMON ROUTNINE
                         .ENDC
                         .IFDF $LP
                        ;
                        ;SET UP TCB TO WRITE A LINE ON LP
                        ;CALLING SEQUENCE:       MOV      BUFAD,R5
                        ;                        CALL     STUPLT
                        ;
   01540 010701  STUPLT: MOV      PC,R1              ;GET ADDRESS OF TCBLP IN R1 & R5
   01542 062701          ADD      #TCBLP-.,R1
         007272
   01546 000400          BR       STUCOM
                         .ENDC
                         .IFDF $PL
                        ;
                        ;SET UP TCB TO WRITE A LINE ON PL
                        ;CALLING SEQUENCE:       MOV      BUFAD,R5
                        ;                        CALL     STUPPT
                        ;
                 STUPPT: MOV      PC,R1              ;GET ADDRESS OF TCBPL IN R1 & R5
                         ADD      #TCBPL-.,R1
                         .ENDC
   01550 010561  STUCOM: MOV      R5,10(R1)
         000010
   01554 010105          MOV      R1,R5
   01556 005061          CLR      4(R1)              ;RESET REV
         000004
   01562                 IREQ                        ;SEND
   01562 012704          MOV      #100000,R4
         100000
   01566 000004          IOT
   01570    001          .BYTE    1,0
   01571    000
   01572 000207          RETURN
                        ;
                        ;SET UP DISK TCB TO READ A BLOCK WITH NO INTERRUPTS & RETURN ADDRESS
                        ;        CALLING SEQUENCE:       ADR      BUFF,R4
                        ;                                ADR      -.CBN,R3
                        ;                                ADR      TCBDK-,R2
                        ;                                CALL     STUPDT
                        ;
   01574 010205  STUPDT: MOV      R2,R5              ;SAVE TCBP IN R5
   01576 022222          CMP      (R2)+,(R2)+        ;BUMP TO REV
```

Figure 5-1
UNICHANNEL Spooler Components (Cont.)



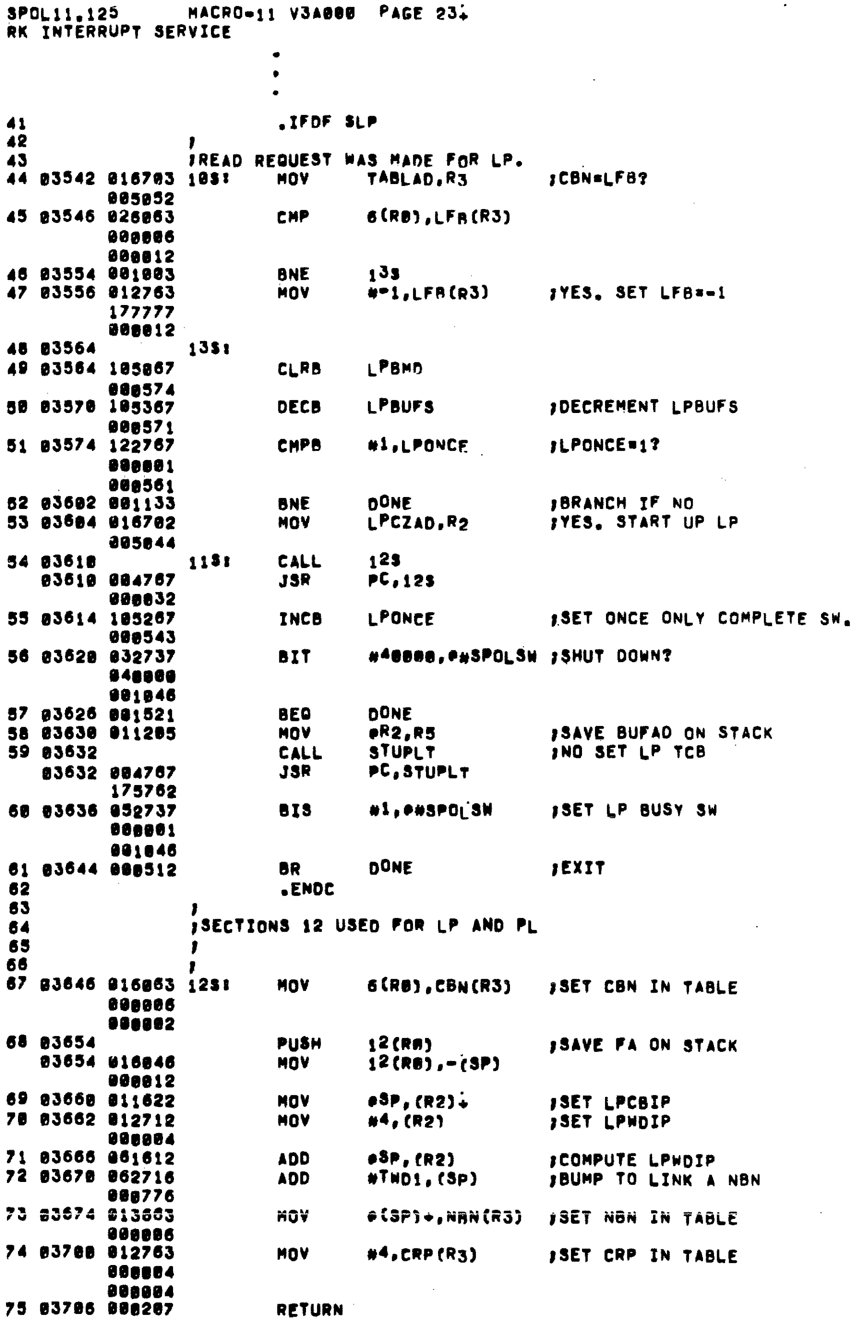
```
SPOL11.125      MACRO-11 V3A000  PAGE 17+
FIND A FREE BLOCK ON DISK
                :
                ;
                ;THE FOLLOWING PIECE OF CODE CHECKS TO SEE IF THE CURRENT BLOCK TO BE
                ;ALLOCATED TO THE CURRENT SPOOLING TASK EQUALS THE CBN OF THIS
                ;DESPOOLING TASK;IF THIS IS TRUE, THEN THE 'SPOOLER IS DECLARED FLOODED'
                ;THIS HAPPENS ONLY ON A WRAP AROUND(ENTIRE SPOOLER AREA IS TREATED AS A
                ;RING BUFFER)WHEN SPOOLING OPERATIONS ARE WAY AHEAD OF DESPOOLING OPERATIONS
                ;
                ;
                ;*****NOTE! AS NEW TASKS ARE ADDED NEW CODE HAS TO BE ADDED*****
                ;*********** SIMILAR TO THE CODE FOR EXISTING TASKS**************
                ;
 02116 116202           MOVB    2(R0),R2        ;GET CURRENT TASK CODE
       000002
 02122 122702           CMPB    #LPCOD,R2       ;LP?
       000004
 02126 001411           BEQ     21$
 02130 122702           CMPB    #CDCOD+200,R2   ;NO, CD?
       000205
 02134 001411           BEQ     22$
 02136 122702           CMPB    #PLCOD,R2       ;NO, PL?
       000025
 02142 001012           BNE     26$
 02144 016702           MOV     TABPLC,R2       ;YES
       005036
 02150 000405           BR      30$
 02152 016702 21$:      MOV     TABPCB,R2
       005026
 02156 000402           BR      30$
                ;
 02160 016702 22$:      MOV     TABCDC,R2
       005024
 02164          30$:
 02164 020112           CMP     R1,(R2)
 02166 001401           BEQ     5$
 02170          26$:
                                                ;RETURN WITH BLOCK # ON STACK
 02170 000207           RETURN
                ;
                ;SORRY NO BLOCK FREE?? SETUP TO HALT CURRENT OPERATION
 02172          5$:     POP     R2              ;GET RETURN ADDRESS
 02172 012602           MOV     (SP)+,R2
 02174                  PUSH    @#PS            ;SET UP STACK FOR RESTART
 02174 013746           MOV     @#PS,-(SP)
       177776
 02200                  PUSH    R2              ;SAVE PC
 02200 010246           MOV     R2,-(SP)
 02202                  PUSH    R0
 02202 010046           MOV     R0,-(SP)
 02204                  PUSH    R1
 02204 010146           MOV     R1,-(SP)
 02206                  PUSH    R2
 02206 010246           MOV     R2,-(SP)
 02210                  PUSH    R3
```

Figure 5-1
UNICHANNEL Spooler Components (Cont.)



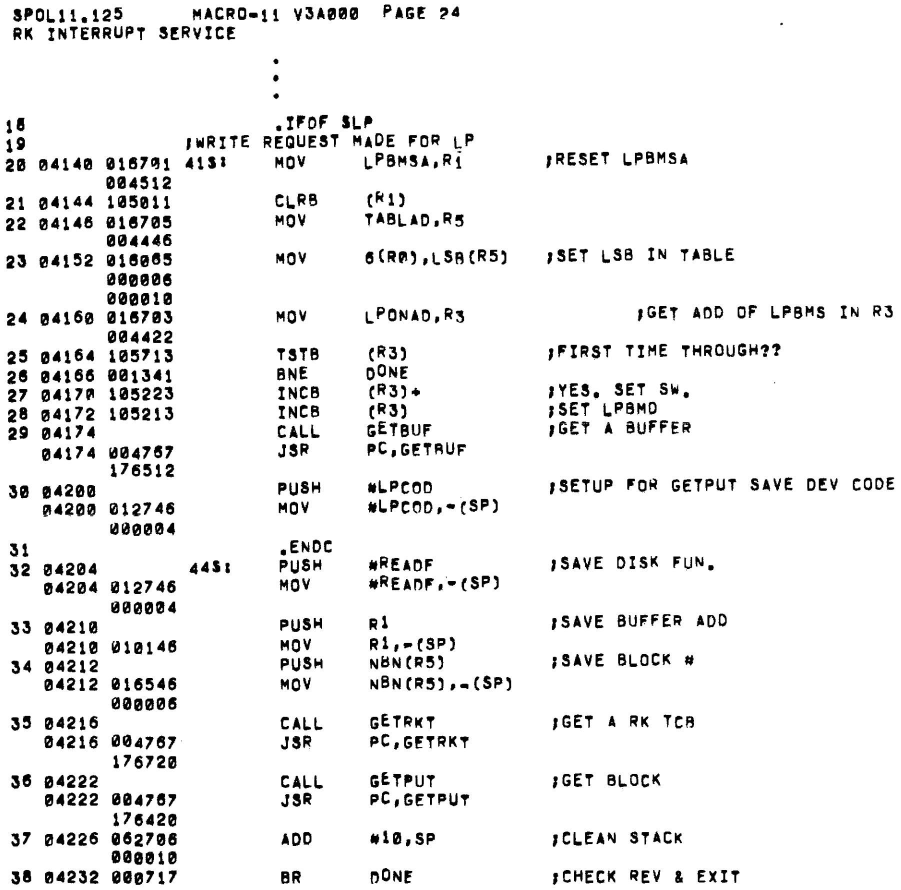
```
SPOL11.125       MACRO-11 V3A000  PAGE 22
TASK SOFTWARE INTERRUPT DISPATCHER
                       ;
                       ;SEND15 IN PIREX TRANSFERS CONTROL TO DEVINT BY A "CALL @SEND11(-COD*2)"
                       ;IF REQUESTED IN TCB.  THIS IS DONE BY A CODE OF '3' IN BYTE-3
                       ;OF TCB. SPOOLER SETS THE ADDRESS OF DEVINT IN SEND11 WHEN STARTED
                       ;
                       ;
                       ;
 003240 022760 DEVINT: CMP     #1,4(R0)          ;GOOD COMPLETION??
        000001
        000004
 003246 001022         BNE     5$                ;BRANCH IF NO
 03250 122760          CMPB    #RKCOD+200,TCODE(R0)     ;RK REQ.?
        000202
        000002
 03256 001417          BEQ     RKINT
 03260 122760          CMPB    #LPCOD+200,TCODE(R0)     ;LP REQ?
        000204
        000002
 03266 001406          BEQ     2$
 03270 122760          CMPB    #CDCOD+200,TCODE(R0)     ;CD REQ?
        000205
        000002
 03276 001404          BEQ     3$
 03300 000167          JMP     PLINT
        002072
                       ;
                       ;
 03304 000167 2$:      JMP     LPINT
        000646
                       ;
 03310 000167 3$:      JMP     CDINT
        002126
                       ;
                       ;
                       ;
 03314                 5$:
 03314 000207          RETURN
                       ;
                       .SBTTL  RK INTERRUPT SERVICE
```

Figure 5-1
UNICHANNEL Spooler Components (Cont.)



```
SPOL11.125      MACRO-11 V3A000  PAGE 23+
RK INTERRUPT SERVICE
                                .
                                .
                                .
41                              .IFDF SLP
42                    ;
43                    ;READ REQUEST WAS MADE FOR LP.
44 03542 016703 10$:  MOV       TABLAD,R3          ;CBN=LFB?
         005052
45 03546 026063       CMP       6(R0),LFB(R3)
         000006
         000012
46 03554 001003       BNE       13$
47 03556 012763       MOV       #-1,LFB(R3)        ;YES, SET LFB=-1
         177777
         000012
48 03564        13$:
49 03564 105067       CLRB      LPBMD
         000574
50 03570 105367       DECB      LPBUFS             ;DECREMENT LPBUFS
         000571
51 03574 122767       CMPB      #1,LPONCE          ;LPONCE=1?
         000001
         000561
52 03602 001133       BNE       DONE               ;BRANCH IF NO
53 03604 016702       MOV       LPCZAD,R2          ;YES, START UP LP
         005044
54 03610        11$:  CALL      12$
   03610 004767       JSR       PC,12$
         000032
55 03614 105267       INCB      LPONCE             ;SET ONCE ONLY COMPLETE SW.
         000543
56 03620 032737       BIT       #40000,@#SPOLSW ;SHUT DOWN?
         040000
         001046
57 03626 001521       BEQ       DONE
58 03630 011205       MOV       @R2,R5             ;SAVE BUFAD ON STACK
59 03632              CALL      STUPLT             ;NO SET LP TCB
   03632 004767       JSR       PC,STUPLT
         175762
60 03636 052737       BIS       #1,@#SPOLSW        ;SET LP BUSY SW
         000001
         001046
61 03644 000512       BR        DONE               ;EXIT
62                              .ENDC
63                    ;
64                    ;SECTIONS 12 USED FOR LP AND PL
65                    ;
66                    ;
67 03646 016063 12$:  MOV       6(R0),CBN(R3)      ;SET CBN IN TABLE
         000006
         000002
68 03654              PUSH      12(R0)             ;SAVE FA ON STACK
   03654 016046       MOV       12(R0),-(SP)
         000012
69 03660 011622       MOV       @SP,(R2)+          ;SET LPCBIP
70 03662 012712       MOV       #4,(R2)            ;SET LPWDIP
         000004
71 03666 061612       ADD       @SP,(R2)           ;COMPUTE LPWDIP
72 03670 062716       ADD       #TWD1,(SP)         ;BUMP TO LINK A NBN
         000776
73 03674 013663       MOV       @(SP)+,NBN(R3)     ;SET NBN IN TABLE
         000006
74 03700 012763       MOV       #4,CRP(R3)         ;SET CRP IN TABLE
         000004
         000004
75 03706 000207       RETURN
```

Figure 5-1
UNICHANNEL Spooler Components (Cont.)



```
SPOL11.125      MACRO-11 V3A000  PAGE 24
RK INTERRUPT SERVICE

                             .
                             .
                             .
18                           .IFDF $LP
19                     ;WRITE REQUEST MADE FOR LP
20 04140 016701 41$:         MOV     LPBMSA,R1          ;RESET LPBMSA
         004512
21 04144 105011              CLRB    (R1)
22 04146 016705              MOV     TABLAD,R5
         004446
23 04152 016065              MOV     6(R0),LSB(R5)      ;SET LSB IN TABLE
         000006
         000010
24 04160 016703              MOV     LPONAD,R3                  ;GET ADD OF LPBMS IN R3
         004422
25 04164 105713              TSTB    (R3)               ;FIRST TIME THROUGH??
26 04166 001341              BNE     DONE
27 04170 105223              INCB    (R3)+              ;YES. SET SW.
28 04172 105213              INCB    (R3)               ;SET LPBMD
29 04174                     CALL    GETBUF             ;GET A BUFFER
   04174 004767              JSR     PC,GETRUF
         176512
30 04200                     PUSH    #LPCOD             ;SETUP FOR GETPUT SAVE DEV CODE
   04200 012746              MOV     #LPCOD,-(SP)
         000004
31                           .ENDC
32 04204          44$:       PUSH    #READF             ;SAVE DISK FUN.
   04204 012746              MOV     #READF,-(SP)
         000004
33 04210                     PUSH    R1                 ;SAVE BUFFER ADD
   04210 010146              MOV     R1,-(SP)
34 04212                     PUSH    NBN(R5)            ;SAVE BLOCK #
   04212 016546              MOV     NBN(R5),-(SP)
         000006
35 04216                     CALL    GETRKT             ;GET A RK TCB
   04216 004767              JSR     PC,GETRKT
         176720
36 04222                     CALL    GETPUT             ;GET BLOCK
   04222 004767              JSR     PC,GETPUT
         176420
37 04226 062706              ADD     #10,SP             ;CLEAN STACK
         000010
38 04232 000717              BR      DONE               ;CHECK REV & EXIT
```

Figure 5-1
UNICHANNEL Spooler Components (Cont.)

```
SPOL11.125        MACRO-11 V3A000  PAGE 26
LP INTERRUPT SERVICE
                        ;
                        ;THIS ROUTINE HANDLES COMPLETION OF I/O SOFTWARE INTERRUPT FROM THE
                        ;DRIVER TASK IN PIREX. IT DESPOOLS THE SPOOLED DATA ONTO THE LP.
                        ;
                                .IFDF   $LP
 004362        000 LPDUMI:  .BYTE   0                       ;UNUSED
 004363        000 LPONCE:  .BYTE   0                       ;ONCE ONLY SW
 004364        000 LPBMD:   .BYTE   0                       ;BLOCK IN MOTION SW
 004365        000 LPBUFS:  .BYTE   0                       ;EMPTY BUFFER COUNT
  04366 000000 LPCBIP:  0                               ;CURRENT BUFFER POINTER
  04370 000000 LPWDIP:  0                               ;CURRENT WORD POINTER
  04372 000000 LPOBIP:  0                               ;NEXT BUFFER POINTER
                                .ENDC
                        ;
                        ;
                                .IFNDF $LP
                        LPINT:  MOV     @#DEVST,R1
                                MOVB    #IOPS77,LPSPER(R1)      ;REPORT TASK NOT SUPPORTED
                                RETURN
                                .ENDC
                                .IFDF $LP
                        ;
  04374 016701 LPINT:   MOV     TABCRT,R1
        004262
  04400 052737          BIS     #LVL5,@#PS      ;INHIBIT DISK INTERRUPTS
        000240
        177776
  04406 022711          CMP     #-1,(R1)        ;ANY MORE TO DO?
        177777
  04412 001014          BNE     1$
  04414 016703 11$:     MOV     LPONAD,R3               ;GET C(LPCBIP) IN R3
        004166
  04420 105023          CLRB    (R3)+           ;RESET SW.'S
  04422 105023          CLRB    (R3)+           ;BUMP TO LPBUFS
  04424 105223          INCB    (R3)+           ;RELEASE BUFF.
  04426 011303          MOV     (R3),R3
  04430                 CALL    GIVBUF          ;GIVE BACK BUFFER
  04430 004767          JSR     PC,GIVBUF
        176360
  04434 042737 2$:      BIC     #1,@#SPOLSW     ;NO. SET LP IDLE SW
        000001
        001046
  04442 000207 50$:     RETURN
  04444 005711 1$:      TST     (R1)            ;YES. BLOCK IN MOTION?
  04446 001040          BNE     3$
  04450 016704 15$:     MOV     LPCPAD,R4       ;SK-124 YES. GET ADD OF LLPCPADBIP IN R2
        004200
  04454 011403          MOV     (R4),R3         ;RELEASE BUFFER
  04456                 CALL    GIVBUF
  04456 004767          JSR     PC,GIVBUF
        176332
  04462 105244          INCB    -(R4)
  04464 105764 10$:     TSTB    -1(R4)          ;BLOCK READ IN?
        177777
  04470 001403          BEQ     4$
  04472                 CALL    WAITBK
  04472 004767          JSR     PC,WAITBK
        175222
  04476 000772          BR      10$
  04500          4$:
  04500 016767          MOV     TABLE+NBN,TABLE+CBN     ;SET CBN=NBN
        005226
        005220
  04506 012767          MOV     #4,TABLE+CRP            ;SET CRP
        000004
        005214
  04514 010703          MOV     PC,R3           ;GET LPOBIP ADD. IN R3
  04516 062703          ADD     #LPOBIP-.,R3
        177654
  04522 011304          MOV     (R3),R4         ;GET C(LPOBIP) IN R3 & BUMP TO TWD1
  04524 016467          MOV     TWD1(R4),TABLE+NBN      ;SET LP.NBN
        000776
        005200
  04532 016702          MOV     LPCPAD,R2               ;GET ADD. OF LLPCPADBIP IN R2
        004116
  04536 011322          MOV     (R3),(R2)+      ;SET LPCBIP
  04540 011312          MOV     (R3),(R2)       ;SET LPWDIP
  04542 062712          ADD     #4,(R2)
        000004
```

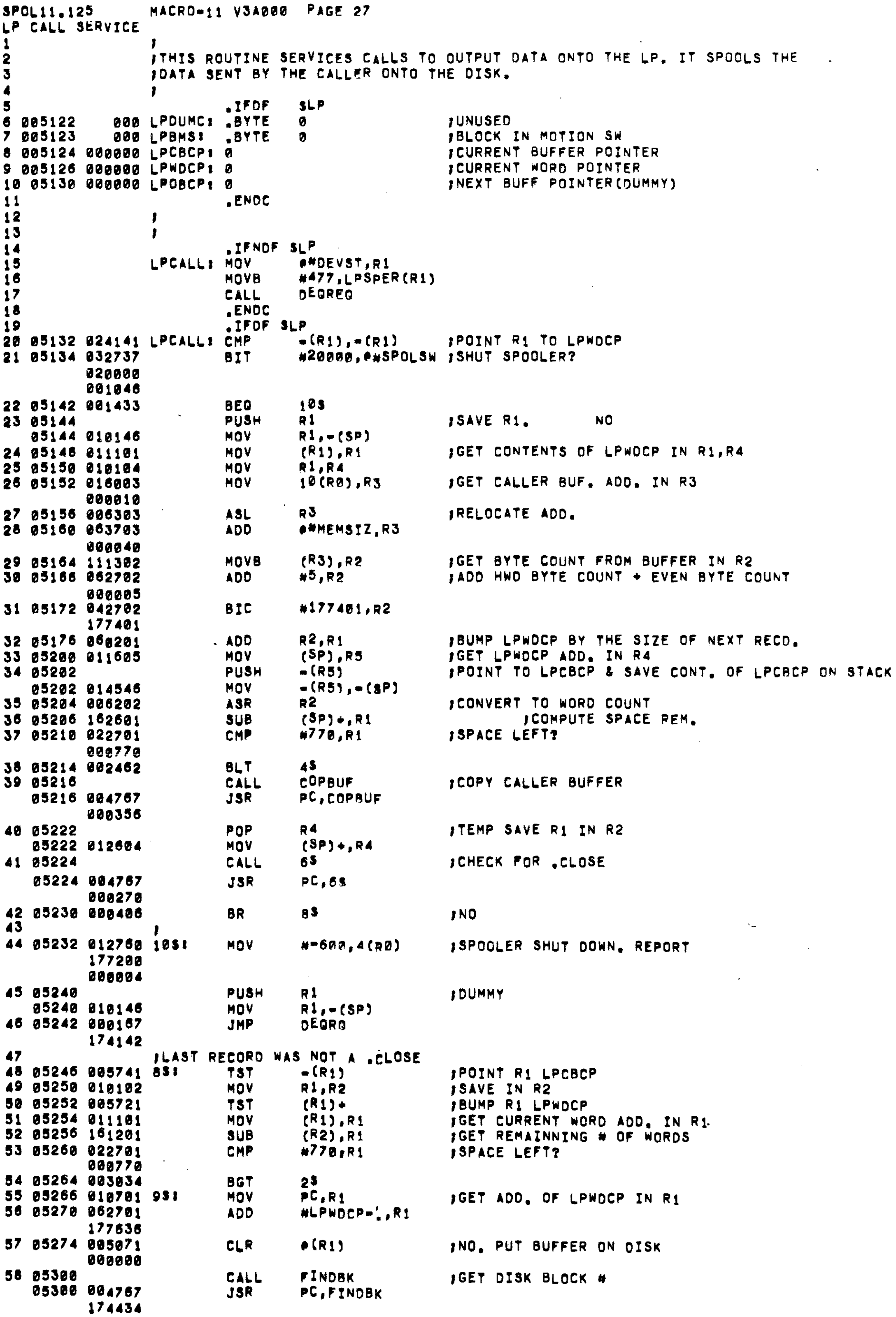
Figure 5-1
UNICHANNEL Spooler Components (Cont.)

```
SPOL11.125      MACRO-11 V3A000  PAGE 26+
LP INTERRUPT SERVICE
56 04546 000412              BR      5$                ;SEND WRITE REQ IF NOT SHUT DOWN
57 04550 016702 3$:          MOV     LPCWAD,R2                 ;GET ADD OF LPWDIP IN R2
         004064
58 04554 017246              MOV     @(R2),-(SP)
         000000
59 04560 062716              ADD     #5,(SP)           ;EVEN BYTE COUNT
         000005
60 04564 042716              BIC     #177401,(SP)
         177401
61 04570 061611              ADD     (SP),(R1)         ;BUMP CRP
62 04572 062612              ADD     (SP)+,(R2)        ;BUMP LPWDIP
63 04574 032737 5$:          BIT     #40000,@#SPOLSW ;SHUT DOWN?
         040000
         001046
64 04602 001714              BEQ     2$
65 04604 032737              BIT     #1,@#SPOLSW       ;SHUT LP?
         000001
         001046
66 04612 001710              BEQ     2$
67 04614 032737              BIT     #10000,@#SPOLSW ;SHUT DESPOOLER
         010000
         001046
68 04622 001704              BEQ     2$
69 04624 005772              TST     @(R2)             ;FIRST RECORD A .CLOSE?
         000000
70 04630 001024              BNE     13$
71 04632 026161              CMP     -2(R1),4(R1)      ;ANY MORE DATA?

         177776
         000004
72 04640 001003              BNE     14$
73 04642                     CALL    12$               ;NO, SET TABLE ENTRIES
   04642 004767              JSR     PC,12$
         000240
74 04646 000662              BR      11$               ;RESET SWITCHES & EXIT
75 04650 016704 14$:         MOV     LPONAD,R4         ;SK-124 GET LPBUFS ADRRESS
         003732
76 04654 062704              ADD     #2,R4             ;SK-124
         000002
77 04660 122714              CMPB    #1,(R4)           ;SK-124 ONE FREE BUFFER?
         000001
78 04664 001271              BNE     15$               ;SK-124
79 04666 105764              TSTB    -1(R4)            ;SK-124 YES, BLOCK IN MOTION?
         177777
80 04672 001266              BNE     15$               ;SK-124
81 04674                     CALL    9$                ;SK-124 NO, GET NEXT BLOCK
   04674 004767              JSR     PC,9$
         000146
82 04700 000663              BR      15$               ;SK-124 RELEASE BUFFER & WAIT FOR BLOCK TO COME IIN
83                    ;
84                    ;
85 04702 011205 13$:         MOV     @R2,R5            ;NO, SAVE BUFF ADD ON STACK
86 04704                     CALL    STUPLT            ;SET UP TCB TO UNTI A LINE
   04704 004767              JSR     PC,STUPLT
         174710
87 04710 016701              MOV     TABCRT,R1
         003746
88 04714 011204              MOV     (R2),R4           ;CHECK FOR BUFFER EMPTY
89 04716 017246              MOV     @(R2),-(SP)       ;GET BYTE COUNT
         000000
90 04722 062716              ADD     #5,(SP)           ;EVEN BYTE COUNT
         000005
91 04726 042716              BIC     #177401,(SP)
         177401
92 04732 062604              ADD     (SP)+,R4          ;BUMP R4 TO POINT TO PT WORD OF NEXT
93 04734 010702              MOV     PC,R2             ;NO, GET ADD OF LPBUFS IN R2
94 04736 062702              ADD     #LPBUFS-.,R2
         177427
95 04742 005714              TST     (R4)              ;LAST RECORD?
96 04744 001417              BEQ     6$
97 04746 022714              CMP     #-1,(R4)
         177777
98 04752 001414              BEQ     6$
99 04754 122712              CMPB    #1,(R2)           ;LPBUFS=1
         000001
100 4760 001230              BNE     50$
101 4762 105742              TSTB    -(R2)             ;YES, BLOCK IN NEXT?
102 4764 001226              BNE     50$
103 4766 026161              CMP     -2(R1),4(R1)      ;NO, MORE TO DOE (CBN=LSB)
         177776
         000004
```

Figure 5-1
UNICHANNEL Spooler Components (Cont.)

```
SPOL11.125      MACRO-11 V3A000  PAGE 26+
LP INTERRUPT SERVICE
104 4774 001622          BEQ     50$
105 4776                 CALL    9$                 ;SK-124 GET NEXT BLOCK
    4776 004767          JSR     PC,9$
         000044
106 5002 000617          BR      50$                ;SK-124 EXIT
107                ;
108                ;
109                ;BUFFER EMPTY; TEST IF MORE BLOCK TO DO?
****** A
110 5004 026161 6$:      CMP     -2(R1),4(R1)       ;MORE TO DO? (CBN=LSB)
         177776
         000004
111 5012 001412          BEQ     7$
112 5014 005011          CLR     (R1)               ;SK-124 SET CRP=0
113 5016 122712          CMPB    #1,(R2)            ;LPBUFS=1?
         000001
114 5022 001004          BNE     8$
115 5024 105742          TSTB    -(R2)              ;BLOCK IN TRANSIT?
116 5026 001002          BNE     8$                 ;SK-124
117 5030                 CALL    9$                 ;SK-124 GET NEXT BLOCK
    5030 004767          JSR     PC,9$
         000012
****** A
118 5034 000167 8$:      JMP     50$                ;SK-125
         177402
119                ;NO MORE BLOCKS TO DO
****** A
120 5040           7$:   CALL    12$                ;SET TABLE ENTRIES
    5040 004767          JSR     PC,12$
         000042
121 5044 000773          BR      8$
122                ;
123                ;
124                ;GET NEXT BLOCK
****** A
125 5046           9$:   PUSH    R1
    5046 010146          MOV     R1,-(SP)
126 5050                 PUSH    R2
    5050 010246          MOV     R2,-(SP)
127 5052                 CALL    GETBUF             ;YES. GET BUFFER & READ NEXT BLOCK
    5052 004767          JSR     PC,GETBUF
         175634
128 5056 010104          MOV     R1,R4              ;SAVE BUFAD IN R4
129 5060                 POP     R2
    5060 012602          MOV     (SP)+,R2
130 5062                 POP     R1
    5062 012601          MOV     (SP)+,R1
131 5064 010467          MOV     R4,LPOBIP                  ;SET LPOBIP
         177302
132 5070 105212          INCB    (R2)               ;SET LPBMS SW
133 5072 012703          MOV     #LPCOD,R3                  ;GET DEV.CODE IN R3. FOR GETBLK
         000004
134 5076 010102          MOV     R1,R2              ;GET LP.CRP ADD. IN R2
135 5100                 CALL    GETBLK             ;GET BLOCK FROM DISK
    5100 004767          JSR     PC,GETBLK
         003200
136 5104 000207          RETURN                     ;SK-124
137                ;
****** A
138 5106 012711 12$:     MOV     #-1,@R1            ;SET CRP=-1
         177777
139 5112 012761          MOV     #-1,6(R1)          ;SET LFB=-1
         177777
         000006
140 5120 000207          RETURN
141                ;
142                      .ENDC
143                      .SBTTL  LP CALL SERVICE
```

Figure 5-1
UNICHANNEL Spooler Components (Cont.)

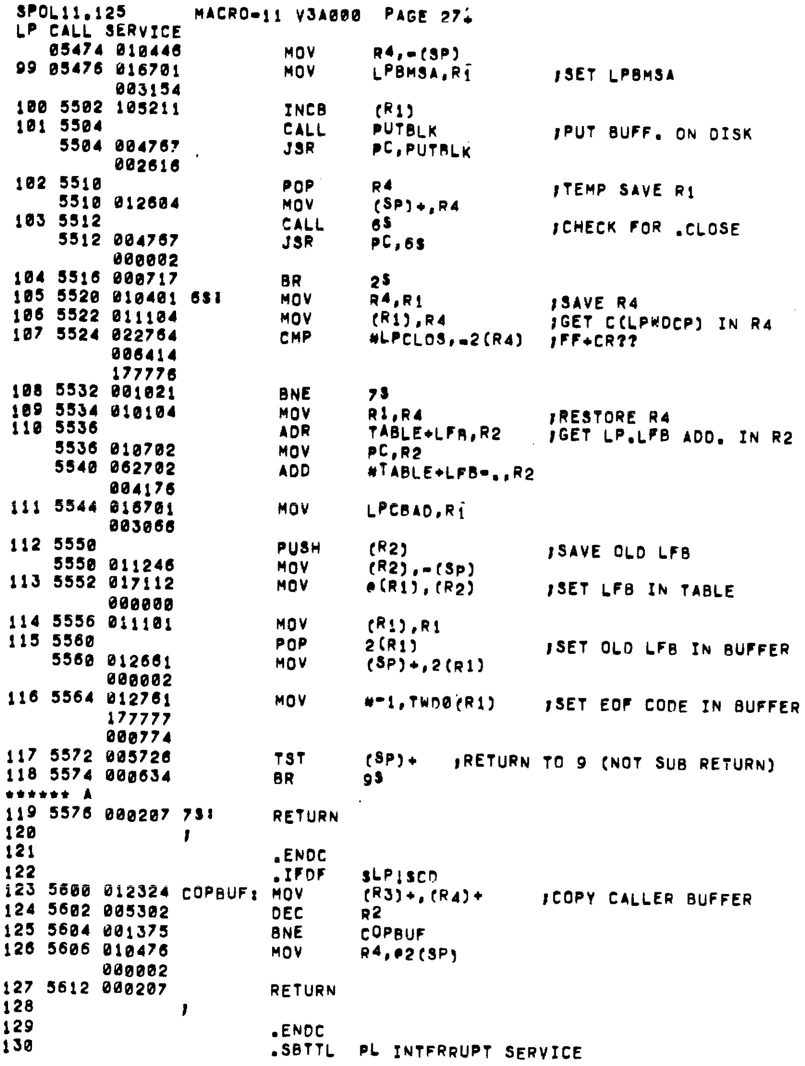


```
SPOL11.125          MACRO-11 V3A000  PAGE 27
LP CALL SERVICE
                     ;
                     ;THIS ROUTINE SERVICES CALLS TO OUTPUT DATA ONTO THE LP. IT SPOOLS THE
                     ;DATA SENT BY THE CALLER ONTO THE DISK.
                     ;
                             .IFDF   $LP
005122      000 LPDUMC: .BYTE   0                   ;UNUSED
005123      000 LPBMS:  .BYTE   0                   ;BLOCK IN MOTION SW
005124 000000 LPCBCP: 0                             ;CURRENT BUFFER POINTER
005126 000000 LPWDCP: 0                             ;CURRENT WORD POINTER
 05130 000000 LPOBCP: 0                             ;NEXT BUFF POINTER(DUMMY)
                        .ENDC
                     ;
                     ;
                        .IFNDF $LP
                LPCALL: MOV     @#DEVST,R1
                        MOVB    #477,LPSPER(R1)
                        CALL    DEQREQ
                        .ENDC
                        .IFDF $LP
 05132 024141 LPCALL: CMP     -(R1),-(R1)         ;POINT R1 TO LPWDCP
 05134 032737         BIT     #20000,@#SPOLSW ;SHUT SPOOLER?
       020000
       001046
 05142 001433         BEQ     10$
 05144                PUSH    R1                  ;SAVE R1.       NO
 05144 010146         MOV     R1,-(SP)
 05146 011101         MOV     (R1),R1             ;GET CONTENTS OF LPWDCP IN R1,R4
 05150 010104         MOV     R1,R4
 05152 016003         MOV     10(R0),R3           ;GET CALLER BUF. ADD. IN R3
       000010
 05156 006303         ASL     R3                  ;RELOCATE ADD.
 05160 063703         ADD     @#MEMSIZ,R3
       000040
 05164 111302         MOVB    (R3),R2             ;GET BYTE COUNT FROM BUFFER IN R2
 05166 062702         ADD     #5,R2               ;ADD HWD BYTE COUNT + EVEN BYTE COUNT
       000005
 05172 042702         BIC     #177401,R2
       177401
 05176 060201         ADD     R2,R1               ;BUMP LPWDCP BY THE SIZE OF NEXT RECD.
 05200 011605         MOV     (SP),R5             ;GET LPWDCP ADD. IN R4
 05202                PUSH    -(R5)               ;POINT TO LPCBCP & SAVE CONT. OF LPCBCP ON STACK
 05202 014546         MOV     -(R5),-(SP)
 05204 006202         ASR     R2                  ;CONVERT TO WORD COUNT
 05206 162601         SUB     (SP)+,R1                    ;COMPUTE SPACE REM.
 05210 022701         CMP     #770,R1             ;SPACE LEFT?
       000770
 05214 002462         BLT     4$
 05216                CALL    COPBUF              ;COPY CALLER BUFFER
 05216 004767         JSR     PC,COPBUF
       000356
 05222                POP     R4                  ;TEMP SAVE R1 IN R2
 05222 012604         MOV     (SP)+,R4
 05224                CALL    6$                  ;CHECK FOR .CLOSE
 05224 004767         JSR     PC,6$
       000270
 05230 000406         BR      8$                  ;NO
              ;
 05232 012760 10$:    MOV     #-600,4(R0)         ;SPOOLER SHUT DOWN, REPORT
       177200
       000004
 05240                PUSH    R1                  ;DUMMY
 05240 010146         MOV     R1,-(SP)
 05242 000167         JMP     DEQRQ
       174142
              ;LAST RECORD WAS NOT A .CLOSE
 05246 005741 8$:     TST     -(R1)               ;POINT R1 LPCBCP
 05250 010102         MOV     R1,R2               ;SAVE IN R2
 05252 005721         TST     (R1)+               ;BUMP R1 LPWDCP
 05254 011101         MOV     (R1),R1             ;GET CURRENT WORD ADD. IN R1.
 05256 161201         SUB     (R2),R1             ;GET REMAINNING # OF WORDS
 05260 022701         CMP     #770,R1             ;SPACE LEFT?
       000770
 05264 003034         BGT     2$
 05266 010701 9$:     MOV     PC,R1               ;GET ADD. OF LPWDCP IN R1
 05270 062701         ADD     #LPWDCP-.,R1
       177636
 05274 005071         CLR     @(R1)               ;NO. PUT BUFFER ON DISK
       000000
 05300                CALL    FINDBK              ;GET DISK BLOCK #
 05300 004767         JSR     PC,FINDBK
       174434
```

Figure 5-1
UNICHANNEL Spooler Components (Cont.)

```
SPOL11.125      MACRO-11 V3A000  PAGE 27+
LP CALL SERVICE
59 05304                  PUSH    R1              ;SAVE BLOCK # ON STACK
   05304 010146           MOV     R1,-(SP)
60 05306 016702           MOV     LPCBCP,R2              ;GET C(LPCBIP) IN R2
         177612
61 05312 011662           MOV     (SP),TWD1(R2)   ;SAVE BLOCK # IN TWD1
         000776
62 05316 012703           MOV     #LPCOD,R3              ;GET LP.DEV CODE IN R3
         000004
63 05322 016701           MOV     LPBMSA,R1       ;SET LPBMSA
         003330
64 05326 105211           INCB    (R1)
65 05330                  CALL    PUTBLK          ;PUT BUFF. ON DISK
   05330 004767           JSR     PC,PUTBLK
         002772
66 05334 016704           MOV     LPCBAD,R4              ;GET ADD. OF LLPCBADBCP IN R3&R4
         003276
67 05340          3$:     CALL    GETBUF          ;GET A NEW BUF
   05340 004767           JSR     PC,GETBUF
         175346
68 05344 010124           MOV     R1,(R4)+        ;SET LPCBCP=BUFAD
69 05346                  POP     (R1)            ;SET BLOCK # IN HWD0 OF NEW BUFF.
   05346 012611           MOV     (SP)+,(R1)
70 05350 062701           ADD     #4,R1           ;BUMP R2 TO WORD 2 OF BUF
         000004
71 05354 010114           MOV     R1,(R4)         ;SET LPWDCP
72 05356          2$:     CALL    DEQREQ          ;DEQUE REQUEST & EXIT IN WAIT STATE
   05356 004767           JSR     PC,DEQREQ
         174014
73 05362          4$:     POP     R1              ;RESTORE ADD. OF CURRENT WORD IN R1
   05362 012601           MOV     (SP)+,R1
74 05364                  PUSH    R3              ;SAVE R3,R2
   05364 010346           MOV     R3,-(SP)
75 05366                  PUSH    R2
   05366 010246           MOV     R2,-(SP)
76 05370 005071           CLR     @(R1)           ;SET BUFF. END SW
         000000
77 05374                  CALL    FINDBK          ;GET DISK BLOCK #
   05374 004767           JSR     PC,FINDBK
         174340
78 05400                  PUSH    R1              ;SAVE BLOCK #
   05400 010146           MOV     R1,-(SP)
79 05402                  CALL    GETBUF          ;GET A BUFF.
   05402 004767           JSR     PC,GETBUF
         175304
80 05406 011611           MOV     (SP),(R1)       ;SET BLOCK # IN HWD0 OF NEW BUFF.
81 05410 016704           MOV     LPCBAD,R4               ;GET ADD. OF LLPCBADBCP IN R4
         003222
82 05414                  PUSH    (R4)
   05414 011446           MOV     (R4),-(SP)
83 05416                  PUSH    (R4)            ;SAVE CONT. OF LPCBCP
   05416 011446           MOV     (R4),-(SP)
84 05420 062716           ADD     #TWD1,(SP)      ;BUMP TO TWD1
         000776
85 05424 016636           MOV     4(SP),@(SP)+    ;SET LINK IN OLD BUFF.
         000004
86 05430 010124           MOV     R1,(R4)+        ;SET LPCBCP & BUMP TO LPWDCP
87 05432 062701           ADD     #4,R1           ;POINT TO WORD 2 IN BUFF.
         000004
88 05436                  PUSH    R4              ;SAVE LPWDCP ADD. ON STACK
   05436 010446           MOV     R4,-(SP)
89 05440 010114           MOV     R1,(R4)         ;SET LPWDCP
90 05442 010104           MOV     R1,R4           ;GET CONT. OF LPWDCP
91 05444 016602           MOV     6(SP),R2        ;RESTORE R3,R2
         000006
92 05450 016603           MOV     10(SP),R3
         000010
93 05454                  CALL    COPBUF          ;COPY CALLER BUFFER
   05454 004767           JSR     PC,COPBUF
         000120
94 05460                  POP     R4              ;SAVE LPWDCP ADD. IN R4
   05460 012604           MOV     (SP)+,R4
95 05462                  POP     R2              ;CONT. OF LPCBCP ON STACK TOP???
   05462 012602           MOV     (SP)+,R2
96 05464 012703           MOV     #LPCOD,R3       ;GET DEV.CODE IN R3. FOR PUTBLK
         000004
97 05470 062706           ADD     #6,SP           ;CLEAN STACK
         000006
98 05474                  PUSH    R4              ;SAVE R5
```

Figure 5-1
UNICHANNDL Spooler Components (Cont.)



```
SPOL11.125      MACRO-11 V3A000  PAGE 27+
LP CALL SERVICE
   05474 010446            MOV     R4,-(SP)
99 05476 016701            MOV     LPBMSA,R1          ;SET LPBMSA
         003154
100 5502 105211            INCB    (R1)
101 5504                   CALL    PUTBLK             ;PUT BUFF. ON DISK
    5504 004767 .          JSR     PC,PUTBLK
         002616
102 5510                   POP     R4                 ;TEMP SAVE R1
    5510 012604            MOV     (SP)+,R4
103 5512                   CALL    6$                 ;CHECK FOR .CLOSE
    5512 004767            JSR     PC,6$
         000002
104 5516 000717            BR      2$
105 5520 010401 6$:        MOV     R4,R1              ;SAVE R4
106 5522 011104            MOV     (R1),R4            ;GET C(LPWDCP) IN R4
107 5524 022764            CMP     #LPCLOS,-2(R4)     ;FF+CR??
         006414
         177776
108 5532 001021            BNE     7$
109 5534 010104            MOV     R1,R4              ;RESTORE R4
110 5536                   ADR     TABLE+LFB,R2       ;GET LP.LFB ADD. IN R2
    5536 010702            MOV     PC,R2
    5540 062702            ADD     #TABLE+LFB-.,R2
         004176
111 5544 016701            MOV     LPCBAD,R1
         003066
112 5550                   PUSH    (R2)               ;SAVE OLD LFB
    5550 011246            MOV     (R2),-(Sp)
113 5552 017112            MOV     @(R1),(R2)         ;SET LFB IN TABLE
         000000
114 5556 011101            MOV     (R1),R1
115 5560                   POP     2(R1)              ;SET OLD LFB IN BUFFER
    5560 012661            MOV     (SP)+,2(R1)
         000002
116 5564 012761            MOV     #-1,TWD0(R1)       ;SET EOF CODE IN BUFFER
         177777
         000774
117 5572 005726            TST     (SP)+   ;RETURN TO 9 (NOT SUB RETURN)
118 5574 000634            BR      9$
****** A
119 5576 000207 7$:        RETURN
120             ;
121                        .ENDC
122                        .IFDF   $LP!SCD
123 5600 012324 COPBUF:    MOV     (R3)+,(R4)+        ;COPY CALLER BUFFER
124 5602 005302            DEC     R2
125 5604 001375            BNE     COPBUF
126 5606 010476            MOV     R4,@2(SP)
         000002
127 5612 000207            RETURN
128             ;
129                        .ENDC
130                        .SBTTL  PL INTFRRUPT SERVICE
```

Figure 5-1
UNICHANNEL Spooler Components (Cont.)

```
$POL11.125       MACRO-11 V3A000  PAGE 33
ADDRESS TABLE
                                .SBTTL  ADDRESS TABLE
                        ;
  007170                ADRTBL:
  007170 003024 RKCAD:  .WORD   RKTCBP
                        .IFDF   $LP
  007172 004145 LPONAD: .WORD   LPONCE
                        .ENDC
  007174 010324 TABPLA: .WORD   TABLE+PLTEOF
                        .IFDF   $PL
                PLONAD: .WORD   PLONCE
                        .ENDC
   07176 007322 BTMPAD: .WORD   BTMPST
   07200 007316 STBKNA: .WORD   STBKNM
   07202 010274 TABLAD: .WORD   TABLE
   07204 010276 TABPCB: .WORD   TABLE+CBN
   07206 010326 TABPLC: .WORD   TABLE+PLTEOF+CBN
   07210 010312 TABCDC: .WORD   TABLE+CDTEOF+CBN
   07212 010404 TCBK1A: .WORD   TCBDK1
                .IFDF   $CD
   07214 005434 CDCPAD: .WORD   CDCBIP
   07216 006002 CDCBAD: .WORD   CDCBCP
                        .ENDC
                        .IFDF   $LP
   07220 004706 LPCBAD: .WORD   LPCBCP
   07222 004152 LPCWAD: .WORD   LPWDIP
                        .ENDC
                        .IFDF   $PL
                PLCBAD: .WORD   PLCBCP
                PLWDAD: .WORD   PLWDIP
                        .ENDC
   07224 010432 TCBK3A: .WORD   TCBDK3
   07226 002322 ENDBAD: .WORD   ENDBSW
   07230 011116 BUFLAD: .WORD   BUFLHD
                        .IFDF   $LP
   07232                LPCPAD:
   07232 004150 LPCZAD: .WORD   LPCBIP
   07234 004705 LPBMSA: .WORD   LPBMS
                        .ENDC
   07236 010310 TABCDT: .WORD   TABLE+CDTEOF
   07240 010300 TABCRT: .WORD   TABLE+CRP
   07242 010330 TABPDT: .WORD   TABLE+PLTEOF+CRP
                        .IFDF   $PL
                PLCIAD: .WORD   PLCBIP
                PLOIAD: .WORD   PLOBIP
                PLBMSA: .WORD   PLBMS
                        .ENDC
                        .IFDF   $CD
   07244 005431 CDBMSA: .WORD   CDBMS
   07246 005442 CDINTA: .WORD   CDINT
                        .ENDC
   07250 010314 TABDCT: .WORD   TABLE+CDTEOF+CRP
   07252 006010 CDCAAD: .WORD   CDCALL
   07254 000146 SPSTAD: .WORD   SPST
                        .IFDF   $CD
   07256 006006 CDOBAD: .WORD   CDOBCP
   07260 006746 RESTAD: .WORD   RESTRO
   07262 005777 CDONAD: .WORD   CDONCE
                        .ENDC
   07264 000000 ONCEFL: .WORD   0
          177741 ADTCNT=ADRTBL-./2
                ;
                ;
                        .SBTTL  BITMAP & TABLE
                ;
   07266        BITMAP: .BLOCK  14                  ;SPOOLER ID INFO
   07316 000000 STBKNM: .WORD   0                   ;SPOOLER AREA FBN
   07320 000000         .WORD   0                   ;SPOOLER AREA SIZE
   07322        BTMPST: .BLOCK  360                 ;START OF BIT MAP
          000360 BTMPSZ=.-BTMPST/2
   10262 000000 BTMPED: 0                           ;POINTER TO END OF BIT MAP
                ;
   10264                .BLOCK  4                   ;HWD'S
   10274        TABLE:  .BLOCK  44                  ;3 DEVICES * 14(8) WORDS EACH
          000044 TABLSZ=.-TABLE/2
                ;
                ; TABLE ENTRIES ARE AS FOLLOWS FOR EACH TASK:
                ;       DEVCOD/CBN/CRP/NBN/LSB/LFB
                ;       0/2/4/6/10/12
                ;
```

Figure 5-1
UNICHANNEL Spooler Components (Cont.)

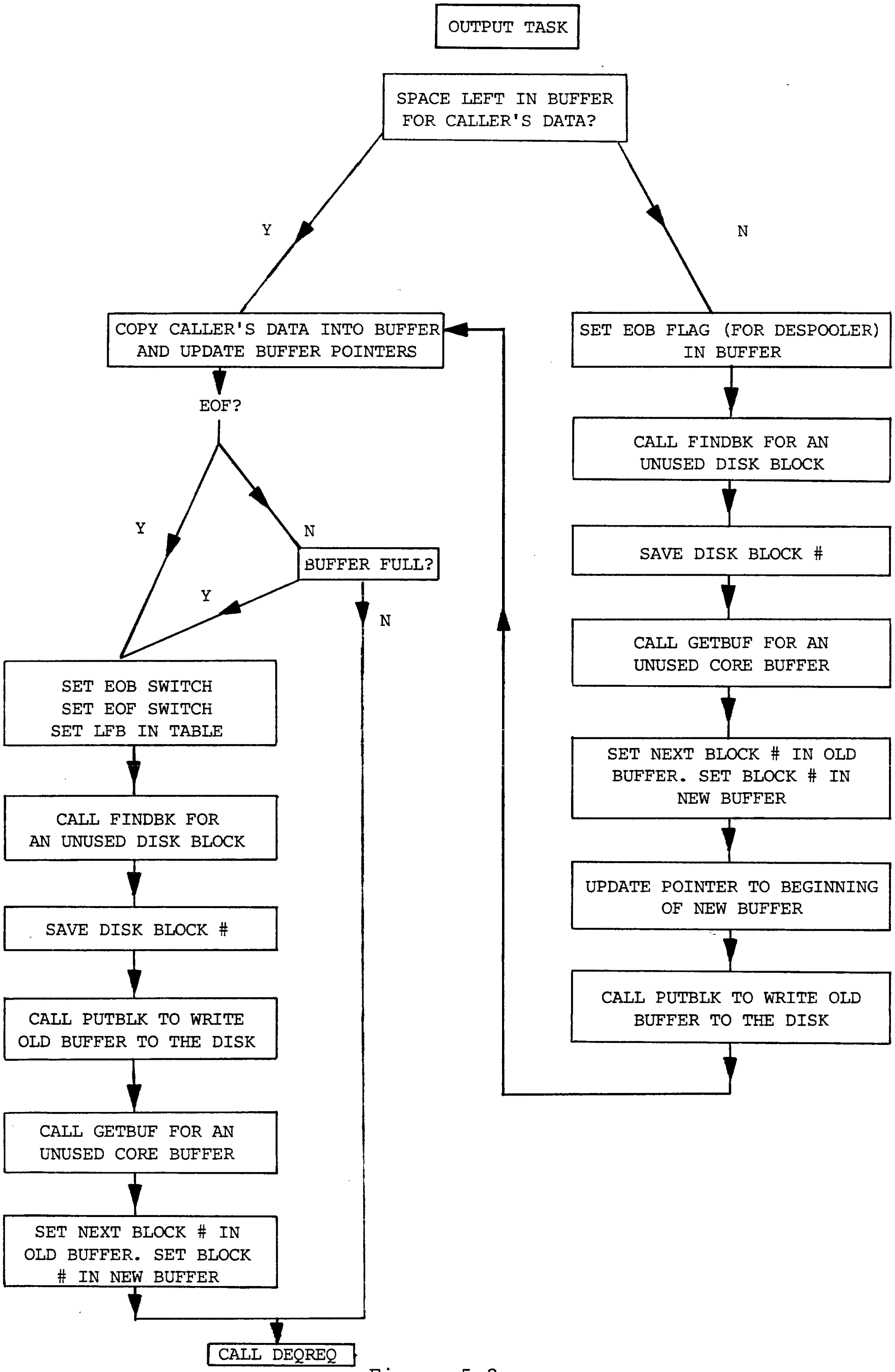

OUTPUT TASK
SPACE LEFT IN BUFFER FOR CALLER'S DATA?
Y
N
COPY CALLER'S DATA INTO BUFFER AND UPDATE BUFFER POINTERS
EOF?
Y
N
BUFFER FULL?
Y
N
SET EOB SWITCH SET EOF SWITCH SET LFB IN TABLE
CALL FINDBK FOR AN UNUSED DISK BLOCK
SAVE DISK BLOCK #
CALL PUTBLK TO WRITE OLD BUFFER TO THE DISK
CALL GETBUF FOR AN UNUSED CORE BUFFER
SET NEXT BLOCK # IN OLD BUFFER. SET BLOCK # IN NEW BUFFER
CALL DEQREQ
SET EOB FLAG (FOR DESPOOLER) IN BUFFER
CALL FINDBK FOR AN UNUSED DISK BLOCK
SAVE DISK BLOCK #
CALL GETBUF FOR AN UNUSED CORE BUFFER
SET NEXT BLOCK # IN OLD BUFFER. SET BLOCK # IN NEW BUFFER
UPDATE POINTER TO BEGINNING OF NEW BUFFER
CALL PUTBLK TO WRITE OLD BUFFER TO THE DISK

Figure 5-2
Task Call Service Routine

Set the SPOOLER task control registers	lines 17-20
Setup the disk TCB pointer table	lines 23-30
Setup and initialize BITMAP	lines 32-49
Initialize and setup TABLE	lines 51-59
Save BITMAP and TABLE on disk	lines 61-66
Set the SPOOLER switches	lines 68,69

LINE PRINTER OPERATIONS:

Initialize the LP call service routine switches and pointers	lines 96, 97, 103-106
Clear all pending LP task requests in PIREX get a free block on disk, get a buffer.	lines 98-100
Set the NBN entry in TABLE.	line 102
Process the next SPOOLER request	line 122

5.5.2 LP SPOOLING

All requests issued to spooled tasks (TCN = 0-177) after a 'BEGIN' directive to the SPOOLER, are processed by the SPOOLER. This is effected by PIREX. When the LP handler in the PDP-15 issues a request to the LP driver task in PIREX, the SPOOLER processes this request. The 'request dispatcher' transfers control to the 'LP call service routine' and the following operations are performed (Refer to Figure 5-1):

Get the current word pointer address	page 27, line 20
Check if spooling operations are disabled and, if disabled, exit	lines 21, 22
Point to the current word	lines 24, 25
Get the caller's buffer address and relocate that address	lines 26-28
Get the byte count of the current record, add the header word byte count, and make the byte count even	lines 29-31
Move ahead the current word pointer by the size of the current record	line 32

Compute the space remaining in the current buffer	line 33-36
Is the buffer full?	lines 37-38
Copy the caller's buffer	lines 39, 123-127
Check for a .CLOSE record	lines 41, 105-108
The record is not a .CLOSE; one more record can fit. Process the next request	lines 42, 48-54
The record is a .CLOSE record; save the old Last File Block (LFB) in TABLE	lines 109, 110, 112
Set the new LFB in TABLE	line 113
Set the old LFB in Header word 2 of the buffer	lines 114, 115
Set an end of file indicator in the buffer	line 116
Go to line 55	
The buffer is full. Set an indicator to this effect in the buffer	lines 55-57
Get a free block on disk (FINDBK)	line 58
Set a pointer to the next block in trailer word 1	lines 59-61
Set the "write block in motion" switch	lines 63, 64
Put the buffer on disk (PUTBLK)	lines 62, 65
Get another buffer (GETBUF)	line 67
Set the "current buffer" pointer for the new buffer	lines 66, 68
Set the block number in the current buffer	line 69
Set the current word pointer to word 2 in the buffer	lines 70, 71
Process the next request	line 72

As disk blocks are written on the disk the Last Spooled Block (LSB) entries in TABLE are updated when the completion of I/O interrupt is processed by the 'disk interrupt service routine' in the SPOOLER (RKINT).

5.5.3 LP Despooling

When the LP device is idle and the first spooled data block is written onto the disk the despooling operations are started in the RKINT routine as follows (Refer to Figure 5-1 and 5-3).

WRITE PROCESSOR:

Reset the "write block in motion" switch	Page 24, lines 20, 21
Set the LSB in TABLE	lines 22, 23
LPONCE = 0, first time through set LPONCE = 1	lines 24-27
Set the "read block in motion" switch	line 28
Get a buffer (GETBUF)	line 29
Get a disk TCB (GETRKT)	line 35
Read a block from disk (GETPUT)	lines 32-34, 36, 37
Return the disk TCB and then EXIT	line 38

READ PROCESSOR:

Is the block read = LFB?	page 23, lines 44-46
Yes, set LFB = 1	line 47
Reset the "read block in motion" switch	line 49
Decrement the LP free buffer count	line 50
LPONCE = 1, first time through, start up LP	lines 51-54
Set Current Block Number (CBN) in TABLE	line 67
Set the current despooling buffer pointer	lines 68, 69
Set the current despooling word pointer	lines 70, 71
Set the Next Block Number (NBN) in TABLE	lines 72, 73
Set Current Record Pointer (CRP) in TABLE	line 74

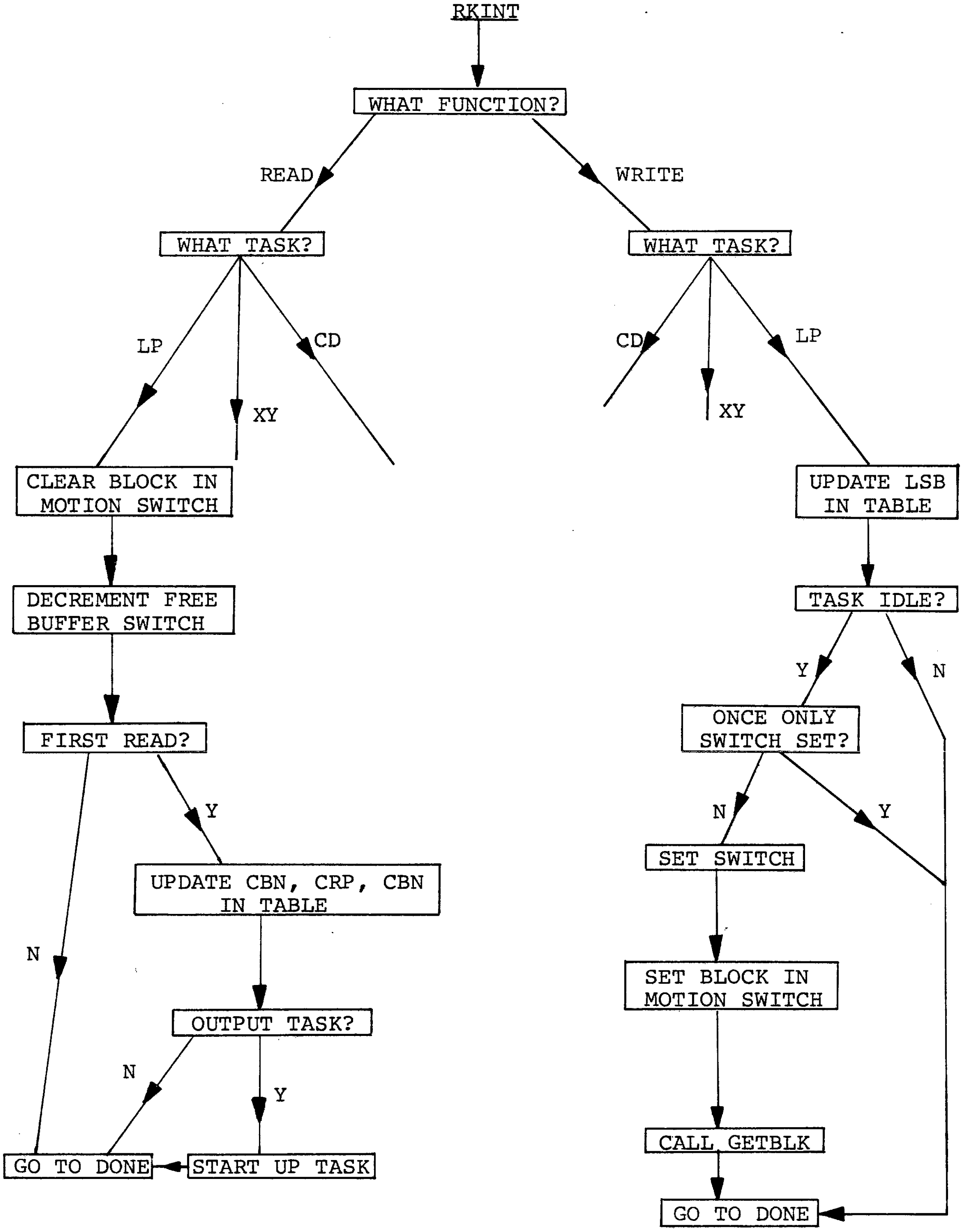


Figure 5-3
Device Interrupt Servicing Logic (For LP)

Set LPONCE = 2	line 55
LP despooling is not shut down; send the LP write request	lines 56-59
Set the LP busy switch	line 61
Return the disk TCB and then EXIT	

Once despooling operations are started the 'LP interrupt service routine' continues the despooling operations until there is no more data to be despooled.

The following operations are performed here (Refer to Figure 5-1):

Protect against a disk interrupt	page 26, line 24
There's nothing more to do; reset LPONCE	lines 25-28
Reset LPBMD and increment the free buffer count	lines 29, 30
Return the buffer (GIVBUF)	lines 31, 32
Set the LP idle switch and return	lines 33, 34
There's more to do; a block is in motion	lines 35, 36
Release the buffer (GIVBUF)	lines 37-39
Increment the free buffer count	line 40
Wait for a block to be read in	lines 41-44
Set CBN - NBN in TABLE	line 46
Set CRP in TABLE	line 47
Set NBN in TABLE	lines 48-51
Set the current despooling buffer and word pointer	lines 52-55
Shut down? Shut LP? Shut despooler?	lines 63-68
Current record in buffer is a .CLOSE record, check if more blocks to do	lines 69-71
There are no more blocks reset TABLE entries, switches and then exit	lines 73, 76, 120-122

One free buffer and no block in motion	lines 75-80
Get next block	line 81
Release buffer and wait to come in	lines 82, 37-44
The first record is not a .CLOSE; send an LP write request	lines 85-86
Point to the first word of the next record	lines 88-92
There are more records left and one free buffer	lines 95-100
There is no read block in motion and more blocks to do	lines 101-104
Get next block	lines 105, 125-136
Return from interrupt call	

5.5.4 SPOOLER Shutdown

All spooling operations can be terminated by issuing the 'END' directive to the SPOOLER. The following operations are performed (Refer to Figure 5-1):

Reset the spooler timer request in PIREX	page 9, line 10
Set the PDP-15's request indicator in the busy/idle switch	line 8
Clear the 'device spooled' switch	line 9
Inhibit interrupts	line 11
Stop the LP task	lines 15, 35-43
Reset the spooler switch	line 25
Shut off software interrupts	lines 26-28
Tell the caller that the 'END' is completed	lines 29-30
Send a request to disconnect the SPOOLER task	lines 32, 33

CHAPTER 6

SPOOLER TASK DEVELOPMENT

6.1 INTRODUCTION

This chapter discusses in detail the procedure for developing a spooled task, and, for integrating it into the SPOOLER software. The development of a spooled task[1] in the UC15 system begins with the development and installation of the task under the PIREX system, if not already present (see Chapters 4 and 5).

Once this has been done, the following summary describes the steps necessary to integrate it into the SPOOLER software:

1. Design and code the call service routine. (Refer to Figure 6-1.)

2. Design and code the interrupt service routine. (Refer to Figure 6-1.)

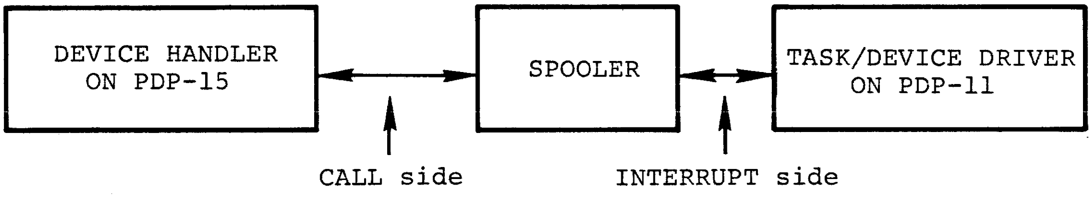


Figure 6-1
SPOOLER Schematic

NOTE

The logical structure of the 'task call service routine' and the 'task interrupt service routine' depends upon whether the task is an input or an output task. The 'task call service routine' is the despooler for an input task and it is the spooler for an output task. The 'task interrupt service routine' is the spooler for input tasks and it is the despooler for output tasks.

[1] There is no program logic or coding connections between the device driver tasks under PIREX and the spooler task. All communication to the device driver is through the TCB only.

3. Add code in the RKINT routine to handle the disk read or write operations for this task.

4. Code a routine to setup TCB and issue request.

5. Add a TCB for this task.

6. Add code to the BEGIN directive processing routine to initialize, and, (if necessary) startup this task.

7. Add code to the END directive processing routine to clear up this task.

8. Add code to the 'request dispatcher' to dispatch calls to this routine.

9. Add code to the 'device interrupt dispatcher' to dispatch interrupts from this device.

10. Increase the size of TABLE by 6 words if not sufficient.

11. Add entries of frequently addressed tags to the central address table.

12. Update DEVCNT and DEVSPP to ensure sufficient buffers and TCBs.

13. Update FINDBK routine.

The remaining sections describe the above steps in more detail. The Line Printer spooler task is used as a descriptive example.

6.1.1 Call Service Routine

This is the routine that normally processes calls from the handler on the PDP-15. For an output task this routine spools data onto the disk as indicated in Section 5.3.3. The operations performed by this routine are discussed in detail in Section 5.4.2.

Normally, data from records are copied into a buffer until it is full. As soon as a buffer is full, it is written onto the disk with a pointer to the next block; and then a new buffer is obtained. This process is continued until a special record that indicates the end of the file is received. For the Line Printer, this is a record with form feed and carriage return characters only. On receipt of this record, the call service routine copies this record into the current buffer and writes it out; regardless of whether the buffer is full or not. This is done to ensure complete processing of a distinct logical entity, a file. The call service routine sets only the LFB entry in the TABLE. It uses the utility routines GETBUF, FINDBK, PUTBLK, and DEQREQ.

6.1.2 Interrupt Service Routine

Completion of I/O interrupts from the device driver in PIREX is processed by this routine. For an output task, this routine despools the data onto the device as indicated in Section 5.3.5. The operations performed by this routine are discussed in detail in Section 5.4.3.

The interrupt service routine for the Line Printer despools data from the buffer onto the device by issuing requests to the task running under PIREX. This routine, like other despooling routines in the SPOOLER, is double buffered to increase throughput. Provision is made in the routine to wait for a block to be read into core during heavy disk utilization. This is done using the "block in motion" switch.

6.1.3 Code to Handle the Disk Read/Write Operations

All spooled tasks must perform certain functions on completion of a read/write block disk operation, as, Section 5.5.3 describes in detail.

On completion of a read disk block request the TABLE entries must be updated and the Line Printer started up if idle. If the Line Printer is busy, control is transferred to the "DONE" section of code where the disk TCB is returned to the pool and control is relinquished.

On completion of a "write block on disk" request, the buffer is returned and the LSB entry in TABLE is updated. If the Line Printer is idle, a request is issued for the Line Printer task to read in the next despooling block. This is done by supplying the NBN[1] entry in TABLE for the Line Printer. If the Line Printer is not busy or after issuing the read request as in read, control is transferred to the 'DONE' section of code.

6.1.4 Routine to Setup TCB and Issue Request

These operations are performed at several places in the SPOOLER. To optimize code this subroutine performs the TCB setup and request issuing functions.

The Line Printer routine performs the following operations (Figure 5-1) at tag STUPLT:

Get the address of the LP TCB	page 16, lines 18-19
Go to setup common	line 20
Set the buffer address specified in the TCB	line 31

(1) See Section 5.4.7.

Reset the REV in the TCB	lines 32-33
Issue the request	line 34
Return control	line 35

6.1.5 TCB

The format of the TCB used by spooler tasks is almost identical to the format of TCBs for tasks running under PIREX, except for the disk TCB which has an extra word. The extra word is used to store the TCN of the task for which the I/O transfer was requested. Another difference is that the TCN present in word '1' of all TCBs in the SPOOLER has the unspooled bit set, i.e., TCN' = 200_8 + TCN (0-177_8). This is to prevent the request from being queued to the SPOOLER. Also, word '0' of all TCBs contains the SPOOLER task code instead of the API information. This is to permit PIREX to transfer control to the 'device interrupt dispatcher' in the SPOOLER on receipt of an I/O completion interrupt from a SPOOLER request.

6.1.6 Initialization in the BEGIN Routine

All SPOOLER tasks have to be initialized before starting of spooling operations. The initialization normally consists of setting the pointers, switches and variables to the right value, obtaining buffers, block number on disk, etc. Section 5.5.1 explains these operations for the Line Printer in more detail.

6.1.7 Cleanup in the END Routine

All SPOOLER tasks have to be cleaned up before termination of spooling operations. The cleanup for the Line Printer consists of stopping the LP driver task in PIREX and clearing all pending requests in the task's TRL.

6.1.8 Updating the Request Dispatcher

The request dispatcher in the SPOOLER contains code to check the TCN of the current request being processed and to transfer control to the appropriate routine. For the Line Printer (Figure 5-1) this is done at:

Page 6, lines 34-36, 72

6.1.9 Updating the Device Interrupt Dispatcher

The SPOOLER is informed of completion of I/O requests through the PIREX Software Interrupt facility. PIREX calls the device interrupt dispatcher, which determines the task that issued the request and transfers control to the tasks interrupt service routine.

For the Line Printer this is done at:

Page 22, lines 12, 13, 19

6.1.10 Updating TABLE

The TABLE contains the complete record of the data being spooled and despooled. Each task has a 6 word entry in this TABLE. TABLE size must be increased (change the 'BLOCK XXX' statement at page 33, line 73) based upon the number of tasks in the SPOOLER. Currently there is sufficient space in the TABLE for 3 additional tasks.

6.1.11 Updating the Central Address TABLE

Code optimization in a PIC program is done by maintaining a table of addresses for frequently used tags. This table contains the unrelocated addresses of tags at assembly time. These are converted to absolute addresses (by adding the SPOOLER first address) by the once only section of code in the SPOOLER (Figure 5-1, page 6, lines 12-26).

For the Line Printer (Figure 5-1) the following tags are present in this table:

LPONCE	page 33,	line 6
TABPCB		line 15
LPCBCP		line 24
LPWDIP		line 25
LPCBIP		line 36
LPBMS		line 37

6.1.12 Update DEVCNT and DEVSPP

To facilitate automatic updating (increase or decrease) of buffers and disk TCBs in the SPOOLER based upon the number of tasks in it, a conditional parameter exists for each task.

DEVCNT and DEVSPP are modified for the Line Printer (Figure 5-1) at:

Page 3, line 13-14

Tasks are assembled into the SPOOLER by defining the conditional parameters of the form:

```
$XX = ZZZZ00
```

where

XX = mnemonic of the task (LP for Line Printer)

ZZZZ = a bit configuration (0400 for LP - there is a bit for each task)

6.1.13 Updating the FINDBK Routine

Code is present in this routine to prevent allocation of the disk block that is currently being despooled. This is necessary to insure proper operation of the spooler because despooling operations are halted when CBN = LSB. For the line printer task (Figure 5-1) this is done at:

page 17, lines 89, 90, 97, 98

6.2 ASSEMBLING THE SPOOLER

To assemble the SPOOLER with the required task in it, it may be necessary to edit the SPOL11 XXX source file to supply the appropriate assembly parameter. To assemble the SPOOLER with the Card Reader task also insert the line:

$CD = 20000 after the sub-title conditional assembly parameters.

(For Plotter insert: $PL = 10000)

An assembly of the above source (Figure 5-1) will produce a SPOOLER with Line Printer and Card Reader tasks.

APPENDIX A

ABBREVIATIONS

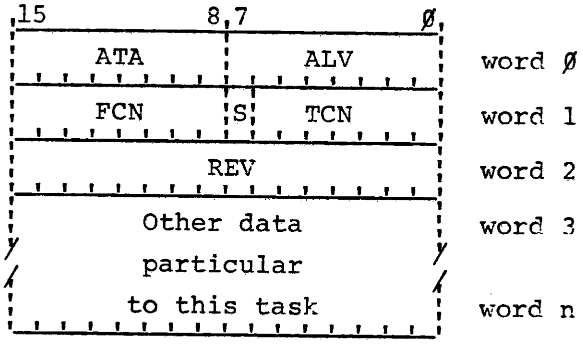
API	Automatic Priority Interrupt
ATL	Active Task List
CAF	Clear All Flags
CAPIn	Clear APIn flag in DR15-C (CAPI0 = 706104, CAPI1 = 706124, CAPI2 = 706144, CAPI3 = 706164)
CBN	Current Block Numbers
CIOD	Clear Input/Output done (706002)
CRP	Current Record Pointer
DOS-15	PDP-15 Disk Operating System
EV	Event Variable
LFB	Last File Block
LIOR	Load Input/Output Register (706006)
LSB	Last Spooled Block
PC	Program Counter
PIC	Position Independent Code (can be loaded anywhere in memory)
RDRS	Read Status Register (706112)
REV	Request Event Variable
RSX-15	PDP-15 Real Time System Executive
SAPIn	Skip on APIn flag in DR11-C (SAPI0 = 706101, SAPI1 = 706121, SAPI2 = 706141, SAPI3 = 706161)
SIOA	Skip on Input/Output data Accepted (706001)

TCB	Task Control Block
TCBP	Task Control Block Pointer
TRL	Task Request List
UC15	UNICHANNEL-15

APPENDIX B

CURRENTLY IMPLEMENTED TCBs

The general format for all task control blocks is as follows:



ATA PDP-15 API interrupt vector address

ALV PDP-15 API interrupt priority level. Must be 0, 1, 2, or 3 (unless FCN = 3).

FCN Function to perform upon completion of this request. Valid values are:

000 Interrupt PDP-15 at location ATA, priority ALV.

001 Do nothing (except set REV)

003 Cause software interrupt to the PDP-11 task whose task code number is in ALV.

S 0 if this request may be spooled.

1 if this request may not be spooled.

TCN Task code number of the task which is to process this request

REV Request Event Variable. Initially zero, set to a non-zero value to indicate completion of the request. The meaning of the various return values is described below.

Returned REV value:

1 Successful (normal) completion.

-200 Non-existent task. The task code number (TCN) does not correspond to any task currently in the PIREX system.

-300 Illegal ALV value. The request may or may not have been performed - see individual request descriptions. The PDP-15 is interrupted at API level 3.

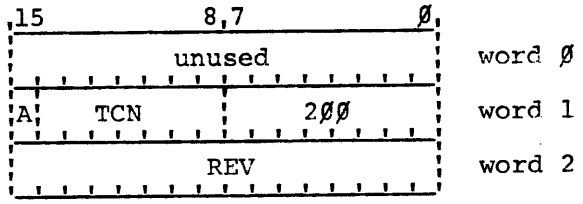
-777 Node Pool empty. PIREX is temporarily out of nodes, and therefore is unable to insert this request into the appropriate list. Reissue the request after a brief delay.

Other The meanings of other returned REV values are given with the descriptions of the task control blocks to which they apply.

In the sections that follow, many of the task control block diagrams show S and TCN combined into a single 8-bit quantity. This is done to indicate that the particular task may never be spooled, and thus S is always 1.

B.1 STOP TASK (ST)

This task provides the capability to stop one or all tasks in PIREX. Stopping a task may immediately abort processing of the request the task is currently processing, and also any PDP-15 originated requests on the task request list. The format of the task control block for the stop task is as follows (note that this is a non-standard task control block):



TCN If zero, this is a stop all tasks directive.

A If set unconditionally, abort the current request for this (or all) task(s). If clear, allow the request currently being processed by this (or each) task to complete if and only if the request originated from the PDP-11. Only PDP-15 requests on the task request list will be aborted regardless of the setting of this bit.

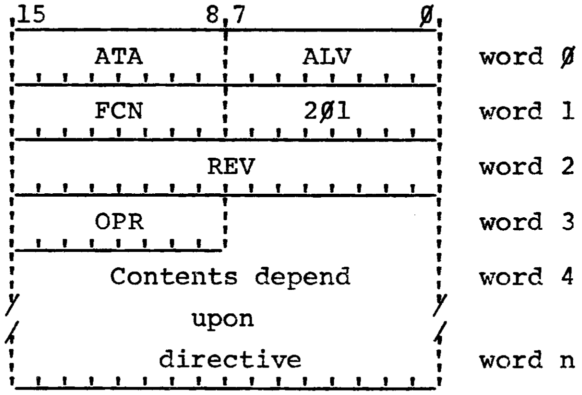
All requests which are aborted via this request will never complete; the request event variables (REVs) of such requests will never be set to a non-zero value. A permanent task which is stopped via this request will be placed in the wait state; a temporary task will be placed in the stopped state.

Returned REV values:

1	Successful completion
-600	Task to be stopped is not connected to PIREX. Only applicable when TCN ≠ 0.

B.2 SOFTWARE DIRECTIVE TASK (SD)

Descriptions of the software directives, including details of their task control block formats, are given in Section 3.6, Software Directive Processing. The general task control block format for all software directives is as follows:



OPR Indicate the exact operation (directive) to be performed. For details see Section 3.6.

Returned REV values:

1	Successful completion
-400	Invalid OPR (directive/operation code) values.
Other	See individual directive description in Section 3.6.

B.3 DISK DRIVER TASK (RK)

The disk driver task provides the capability of using the RK05 cartridge disk system. Task control blocks directed to this task have the following format:

15	8	7	Ø	
ATA		ALV		word Ø
FCN		2Ø2		word 1
REV				word 2
Block Number				word 3
REL + MSMA				word 4
LSMA				word 5
Word Count				word 6
unused	Unit	Function		word 7
RKCS				word 1Ø
RKER				word 11
RKDS				word 12

ATA	Usually 047_8
ALV	Usually 000
REV	Set to 1 upon completion regardless of errors.
Block Number	Disk block number to transfer
REL	000000 if request comes from PDP-15 100000 if request comes from PDP-11
MSMA	Core address at which to begin transfer - most significant bits
LSMA	Core address at which to begin transfer - least significant bits.
Word Count	Two's complement of the number of words to transfer
Unit	Disk drive (unit) number on which to perform the operation.
Function	Operation to be performed.

Valid values are:

002	Write
004	Read
006	Write check
012	Read Check
016	Write lock

For detailed descriptions of the functions, see the RK11-E Disk Drive Controller Manual (DEC-11-HRKDA-B-D).

RKCS
RKER
RKDS — Upon completion of the operation, these three words are loaded from the corresponding disk controller registers. See the RK11-E Disk Drive Controller Manual (DEC-11-HRKA-B-D) for a description of their meaning.

If the request originates from the PDP-11, LSMA is the 16-bit PDP-11 byte address at which the transfer is to begin. If the request originates from the PDP-15, MSMA and LSMA together are the 17-bit PDP-15 word address at which the transfer is to begin. Upon completion of the transfer, REV is always set to 1, regardless of whether or not the transfer succeeded. RKCS, RKER, and RKDS must be examined to determine whether the transfer succeeded or an error occurred.

Returned REV Values:

1	Request complete. Request may or may not have succeeded.
-300	Illegal ALV value. Request complete.

B.4 LINE PRINTER DRIVER TASK (LP)

The task control block format is as follows:

```
 15            8 7             Ø
|      ATA      |      ALV      |   word Ø
|      FCN      |S|    ØØ4      |   word 1
|              REV              |   word 2
|              REL              |   word 3
|         Buffer Address        |   word 4
|             unused            |   word 5
|           Status Flag         |   word 6
```

ATA Usually 056_8

ALV Usually 002

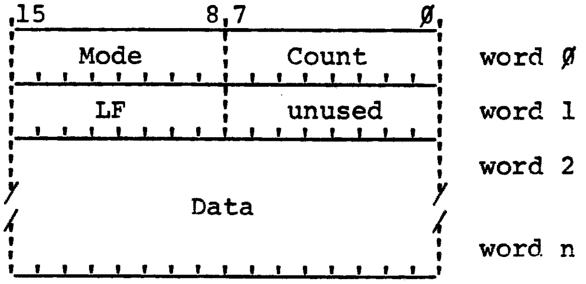
S Usually 0 (indicating spooled operation)

REL 000000 If request originates from PDP-15
100000 If request originates from PDP-11

Buffer Address PDP-11 byte address, if request is from PDP-11
PDP-15 word address, if request is from PDP-15

Status Flag Unused if request is spooled.
Cleared to zero at beginning of request processing and set to 000001 at completion if request is not spooled.

The buffer address argument refers to a line buffer of the following format:



Count The number of bytes of data in the buffer. Excludes the four byte header.

Mode Indicates transfer mode. Legal values are:

0 IOPS ASCII

1 Image

LF May be altered by the driver.

Data One line of output for the line printer.

The data sent to the line printer driver is a series of independent bytes. If a byte is positive, it represents a 7-bit ASCII character. If a byte is negative, it represents some number of spaces, the number of spaces being equal to the absolute value of the byte. If a line is in image mode, only the characters represented by the data bytes are output. If a line is in IOPS ASCII mode, a line feed is output before the beginning of the line unless the first character of the line is a carriage return or form feed. A carriage return is always output at the end of lines in IOPS ASCII mode. A line containing just the characters carriage return followed by form feed causes no output in either mode, but rather represents a .CLOSE (end of file) operation.

Line printer errors are not reported via returned REV values. The only line printer error which can occur is for the printer to go off line (become not ready). The line printer driver reports this by placing the value 4 in the device error byte of its entry in the DEVST table (see Section 3.6.4 on the Error Status Report Directive). When the printer comes back on line the driver clears the device error byte and outputs the line. Upon completion the REV is set to 1.

Returned REV Values:

1	Successful completion
-300	Illegal ALV value. Action may or may not have been taken.
-600	Spooler shut down. No action has been taken.

B.5 CARD READER DRIVER TASK (CD)

The task control block format is as follows:

15 … 8	7 … Ø	
ATA	ALV	word Ø
FCN	S ØØ5	word 1
REV		word 2
unused		word 3
Buffer Address		word 4

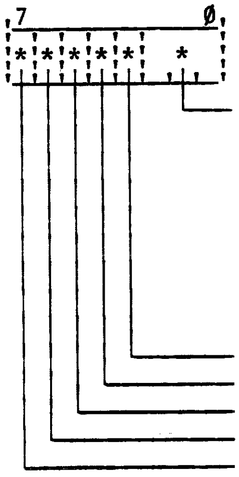
ATA	Usually 055_8
ALV	Usually 001
S	Usually 0 (Indicating spooled operation)
Buffer Address	PDP-11 byte address, if request is from PDP-11 PDP-15 word address, if request is from PDP-15

The buffer address argument refers to a card buffer of the following format:

15 … 8,7 … Ø	
Byte Count	word Ø
Checksum	word 1
Data	word 2 … word n

Byte Count	Always 80_{10}
Checksum	Word checksum of the buffer (including the byte count)
Data	80_{10} bytes (40_{10} words) of data

The card data is not in ASCII. Each card column occupies one byte in the following format:



bits Ø-2 Contents of rows 1-7 encoded as follows:

ØØØ	no punches in rows 1-7
ØØ1	row 1 punched
Ø1Ø	row 2 punched
Ø11	row 3 punched
1ØØ	row 4 punched
1Ø1	row 5 punched
11Ø	row 6 punched
111	row 7 punched

bit 3 Indicates row 8 punched

bit 4 Indicates row 9 punched

bit 5 Indicates zone Ø punched

bit 6 Indicates zone 11 punched

bit 7 Indicates zone 12 punched

NOTE

All combinations of punches which cannot be specified in this manner are illegal.

Any errors that occur are not reported by returned REV values. Instead the IOPSUC numeric error code is placed in the device error byte of the card reader's entry in the DEVST table (see Section 3.6.4, Error Status Report Directive). When the error condition is remedied, the driver clears the device error byte and the read operation continues. Ultimately the read completes and REV is set to 1.

Returned REV Values:

1	Successful completion
-300	Illegal ALV values. Action may or may not have been taken.
-700	Spooler shut down. (Despooling not enabled) No action taken.

B.6 PLOTTER DRIVER TASK (XY)

The task control block format is as follows:

15 8		7 Ø	
ATA		ALV	word Ø
FCN	S	ØØ6	word 1
REV			word 2
REL			word 3
Buffer Address			word 4

ATA Usually 065_8

ALV Usually 003

S Usually 0 (indicating spooled operation)

REL 000000 If request is from PDP-15
100000 If request is from PDP-11

Buffer Address PDP-11 byte address, if request is from PDP-11
PDP-15 word address, if request is from PDP-15.

The buffer address argument refers to a data buffer of the following format:

15 8	7 Ø	
Mode	Count	word Ø
unused		word 1
Data		word 2 word n

Count The number of bytes of data in the buffer. Excludes the four byte header.

Mode Indicates the function to perform and/or the mode in which the data should be interpreted. Valid modes are:

1	Line mode
2	Character mode
3	Initialize
4	Pen select[1]
377	End of file

Line mode data takes the following form. Each line is represented by a pair of data words. The first word is the incremental change in the X coordinate from the beginning to the end of the line, the second word the change in the Y coordinate. If this is to be an invisible line - i.e., it is to be drawn with the pen raised - 100000_8 should be added to the first word (change in X).

Character mode data is a series of ASCII characters to be drawn, one character per byte. Initialize requires 8 words of data which specify the character size and orientation for character mode plotting. The pen select operation[1] takes two words of data. The first is the pen number for the XY311 plotter (1, 2, or 3). The contents of this word are destroyed by the pen select operation. The second word must be zero. An end of file merely raises the pen. (It also forces the XY data through the spooler buffers if spooling is enabled.)

Returned REV Values:

1	Successful completion
-300	Illegal ALV value. Action may or may not have been taken.
-600	Spooler shut down. No action taken.

[1] This is used only by the XY311 plotter.

APPENDIX C

UC15 RELATED ERROR MESSAGES

```
IOPSUC    YYY     XXXX
```

Where YYY denotes one of the following:

EST	Stop all I/O	Task
ESD	Software Driver	"
RKU	Disk Cartridge	"
DTU	DECTAPE	"
LPU	Line Printer	"
CDU	Card Reader	"
PLU	Plotter	"
ESP	Spooler	"
EMA	MAC11	"

XXXX denotes one of the following:

3 -	ILLEGAL INTERRUPT TO DRIVER
4 -	DEVICE NOT READY
12 -	DEVICE FAILURE
15 -	SPOOLER FULL WARNING MESSAGE
20 -	SPOOLER DISK FAILURE - SPOOLING DISABLED
45 -	GREATER THAN 80 COLUMNS IN CARD
55 -	NO SPOOLER BUFFERS AVAILABLE
72 -	ILLEGAL PUNCH COMBINATION

74 - TIMING ERROR - CARD COLUMN
LOST - RETRY CARD

75 - HARDWARE BUSY - DRIVER NOT

76 - HARDWARE ERROR BETWEEN
CARDS

77 - UNRECOGNIZED TASK REQUEST -
DEVICE NOT PRESENT

400 - SPOOLER EMPTY - PDR-15 INPUT
REQUEST PENDING

Additional IOPS error messages:

Error Code	
25	XY plotter - value too large for plotting.
27	XY plotter - mode incorrect.
200	Non-existent task referenced.
300	Illegal API level given (illegal values are changed to level 3 and processed).
400	Illegal directive code given.
500	No free core in the PDP-11 local memory.
600	ALT node for this TCN missing.
777	Request node was not available from the POOL; i.e., the POOL was empty and the referenced task was currently busy or the task did not have an ATL node in the Active Task List.

APPENDIX D

UNICHANNEL-15 OPTION

NOTE

> The following applies ONLY to the construction of a DOS-15 V3AØØØ UNICHANNEL option system. This is required as a prerequisite to the construction of a DOS-15 V3BØØØ option system. See the DOS-15 V3BØØØ Update Document DEC-15-OD3BA-A-D for information on DOS-15 V3BØØØ option system construction.

WARNING

> When using SGEN with the UC15 option DO NOT reply yes to the "UC15 CONFIG?" question.

The UC15 OPTION system is designed to allow users with multiple types of disk devices to use the RF or RP disk as a systems device in conjunction with the UC15. The DOS-15 Vnn UC15 OPTION tape DEC-15-ODUCA-A-UC[1] must be used.

The following example sequence shows the installation of the UNICHANNEL software on an RP system. The installation on a RF disk system would be similar, as would the use of magtape instead of DECTAPE.

1. Load and start the DOSSAV paper tape. Restore the two DECTAPES onto the disk pack.

```
DOSSAV V3A000

INPUT DEVICE? DT

UNIT #? 0

OUTPUT DEVICE? DP

UNIT #? 0

DATE CREATED:  08-AUG-74

TAPE DONE.  MOUNT ANOTHER
1
DOSSAV V3A000

INPUT DEVICE?
```

2. Load and start the supplied RPBOOT tape.

(1) If the system has magtape, use magtape DEC-15-ODUCA-A-MC9 or DEC-15-ODUCA-A-MC7.

3. Assemble the RPBOOT XXX source with the assembly parameter UC15 = 0 with paper tape binary output. This special bootstrap is to be used whenever the PDP-11 monitor PIREX is running, and only then.

4. MICLOG SYS

5. Mount the UC15 OPTIONS tape on DT0 or MT0[1]

6. Patch the special RESMON, DOSNRM, DOSBCD, and SGNBLK, located on the UC15 OPTIONS tape onto the system.

```
$A ␣ {DT} ␣ -10↵
     {MT}

$PATCH↵

PATCH Vnn↵

>RESMON↵

>READ RESMON↵

>READ DOSBCD↵

>READ SGNBLK RPA↵ (for RF use 'SGNBLK RFA')

>DOS15↵

 READR  16077  DOSNRM↵

 EXIT↵
```

NOTE

The PDP-15 will halt on this EXIT.

7. Load ABSL11 XXX paper tape (see Section 2.2.2).

8. Load and start the supplied PIREX XXX PDP-11 MONITOR paper tape (see Section 2.2.2).

9. Reload the DOS System using the special RPBOOT tape produced in step 3. This tape will be used for all future boots while the UC15 option is being used.

10. MICLOG SYS

11. Run SGEN to install MAC11 as a systems program.

```
H. ADD SYS PROG? (N) Y

PROG NAME[ ] MAC11

# OF BLOCKS[ ] 40

OVERLAY NAME[ ]

BUFFS[0] 2
```

[1] For magtape use the MTA handler.

```
DAT SLOTS:

>-11, -12
>
```

12. Run PATCH to place proper values in SGNBLK for MAC11. The values typed by the system after the slash are current disk contents, and may not match the example typout given. Type the values after the >'s, i.e., 1, 17625, and 17500. Follow the typins with ALT-MODES.

```
DOS-15 V3A000

$PATCH↵

PATCH V3A000

>MAC11↵

>FA↵

>00237/001250>1

>PS↵

 00240/016331>17625

>SA↵

 00241/001415>17500

>EXIT↵
```

13. LOGIN PER

14. PIP the MAC11 components from DT0 or MT0 to disk.

```
DOS-15 V3A000

$PIP

DOSPIP V3A000

>T DP,←DT0 MACIMG 006,MACINT 014

>↑C
```

15. Assemble MAC1MG and MACINT. (See Section 2.3.1 for more details.)

The PDP-11 Peripheral Processor may have varying amounts of local memory. The default value is 8K, which requires no assembly parameters. For 12K define LM12K = 0, for 4K define LM4K = 0.

```
DOS-15 V3A000

$MACRO
```

```
BMACRO-15 V3A000

>BP←MACIMG 006

LM12K=0

↑D EOT

 END OF PASS 1

SIZE=00422      NO ERROR LINES

BMACRO-15 V3A000

>BP←MACINT  014

LM12K=0

↑D EOT

 END OF PASS 1

SIZE=17617      NO ERROR LINES

BMACRO-15 V3A000

 ↑C

DOS-15 V3A000
```

The system area on disk for MAC11 requires a PDP-15 core image, and a PDP-11 core image.

16. Load the PDP-11 image from paper tape by running the binary MACIMG. (See Section 2.3.1 for exact details of proper tape selection.) If the system has API - issue a DOS API OFF command first.

```
DOS-15 V3A000

$GLOAD

BLOADER V3A000

 >←MACIMG   ALT

DONE

DOS-15 V3A000
```

17. MICLOG SYS

18. Patch MACINT, the PDP-15 portion of MAC11, into the system in the normal manner

```
DOS-15 V3A000

$A DP  <PER>   -10
```

```
$PATCH↵

PATCH V3A000

>MAC11

>READ MACINT

>EXIT

DOS-15 V3A000
```

19. LOGIN PER

20. PIP the PIREX source onto the disk for editing.

```
DOS-15 V3A000

$PIP

DOSPIP V3A000

>T DP ←DT0 PIREX XXX

↑C
```

21. See Section 2.3.2 for the details of reconfiguring PIREX into a version specific to your exact configuration. Do this reconfiguration now.

22. PIP the sources for the UNICHANNEL handlers from DT0 or MT0 onto disk.

```
DOS-15 V3A000

$PIP

DOSPIP V3A000

>T DP ,←DT0 LPU. 020,XYU. 032

↑C

DOC-15 V3A000
```

Note that the card reader source CD.DOS is already on <PER>.

23. Assemble the sources to binaries. Note that the card reader source requires the assembly parameter UC15 = C.

```
$MACRO

BMACRO-15 V3A000

>B←LPU. 020

 END OF PASS 1

 SIZE=00657    NC ERROR LINES
```

```
BMACRO-15 V3A000

>B←XYU. 032

 END OF PASS 1

SIZE=01150    NO ERROR LINES

BMACRO-15 V3A000

>BP←CD.DOS 031

UC15=0

↑D EOT

 END OF PASS 1

SIZE=00613    NO ERROR LINES

BMACRO-15 V3A000

↑C

DOS-15 V3A000
```

24. MICLOG SYS

25. PIP the handler binaries to DP <IOS> . Note especially the name changes. The sources are called XXU for designating UNICHANNEL sources. The handlers, however, must be named XYA, CDB, LPA.

```
>T DP <IOS> XYA. BIN←DP <PER> XYU. BIN

>T DP <IOS> LPA. BIN←DP <PER> LPU. BIN

>T DP <IOS> CDB. BIN←DP <PER> CD.DOS BIN
```

26. Transfer the three RK handlers from the UC15 OPTIONS tape to the <IOS> UIC.

```
>T DP <IOS> , , ← DT0 RKA. BIN,RKB. BIN,RKC. BIN
```

27. It is now necessary to run SGEN to install new SKIP IOTS (all four devices) and new handler names (RK and XY) in the system.

```
B ALTER I/O DEVICES OR HANDLERS? (N) Y


DELETE DISCARDED HANDLERS? (Y) Y

TO BE KEPT

PR?($) $
    .
    .
    .
LP? ($) Y
```

```
LPA? (Y)

NEW HANDLERS:

>

LPSF=706501? (Y) Y

NEW SKIPS:

>LPSK=706141

>

CD? ($) Y

CDB? (Y)

NEW HANDLERS?

>

CRSI=706701? (Y) Y

CRSD=706721? (Y) Y

NEW SKIPS:

>CRSF=706121

>

C. ADD NEW DEVICE? (N) Y

DEVICE CODE[ ] RK

NEW HANDLERS:

>RKA

>RKB

>RKC

>

NEW SKIPS:

>RKSF=706101

>

C. ADD NEW DEVICE? (N) Y

DEVICE CODE[ ] XY
```

```
NEW HANDLERS:

>XYA

>

NEW SKIPS:

 XYSF=706161

>
```

28. Halt both machines.

29. Load ABSL11.

30. Load in the new PIREX tape (specific to your machine).

31. Bootstrap DOS with the modified RPBOOT.

The system is ready to use UNICHANNEL peripherals.

It should now be DOSSAVed. This system will operate only with the UNICHANNEL-15 peripheral processor. If PIREX is not executing, this system will not function.

GLOSSARY

Active Task

An Active Task is one which:

1. is currently executing
2. has a new request pending in its queue
3. is in a wait state
4. has been interrupted by a higher priority task.

Active Task List

A priority-ordered linked list of Active Tasks used for scheduling tables. The ATL is a queue consisting of one node for each Active Task in the system.

Busy/Idle Switch

A two-word storage area used to save TCBP's when processing a request. Every task has a two-word Busy/Idle Switch. If the two words are zero, the task is currently not busy and is able to accept and process a new request. Bit 15 of the first word is used by the system to determine if the TCB came from a PDP-15 or PDP-11 request. If zero, the request came from the PDP-15, otherwise it came from the PDP-11.

Call Side

All spoolers have a 'call side' where a set of data is passed by the caller to the spooler (for output spooled devices/tasks) or data is passed by the spooler to the caller (for input spooled devices/tasks). This is done only when a request is made to the spooler.

Context Save

The storing of all active registers, including the program counter (PC) and program status (PS), on the current task's stack. These saves are done when higher priority tasks interrupt lower priority ones and by device driver interrupt routines to allow them free use of the general purpose registers.

Context Switching

The process of saving the active registers belonging to the current task executing (a context save), determining a new task to execute, and finally restoring the registers belonging to it.

Deque

Deque, pronounced deck, is a double-ended queue consisting of a listhead and list elements, circularly linked by both forward and backward pointers. Deques (linked lists) are used, instead of tables, to store TCB pointers and ATL information. The list elements (commonly called nodes) are initially obtained from a pool of empty nodes called the POOL. Nodes consist of listhead and 2 words of data used to store the caller's TCB pointer or ATL information. When a node is needed, it is removed from the POOL and queued to the referenced task deque of the ATL. When a node is no longer needed, it is zeroed and returned to the POOL.

Dequeue

Remove a node from a queue.

Directive

A task which performs some specific operation under PIREX, e.g., connecting and disconnecting tasks.

Driver

A task which controls a hardware device. Drivers usually consist of necessary program only rudimentary operations (e.g., read, write or search). The more complex operations such as file manipulations and syntax checking are usually performed by handlers.

Event Variable

A word or variable used to determine the status of a request. The Event variable is set to indicate successful completion, rejection, status, or a request still pending condition.

Interrupt Side

All spoolers have an 'Interrupt Side' where data is passed by the spooler to the device/tasks (for output spooled device/tasks) or data is passed from the device/tasks to the spooler (for input spooler devices/tasks). This occurs whenever output of data is complete or input data is ready.

Linked List

A deque consisting of nodes and listhead used to store system information. An empty list consists of only a listhead.

Listhead

A two-word core block with forward and backward pointers pointing to the next and previous list node or to itself if empty. The listhead is a reference point in a circularly-linked list.

Local Memory

Core memory only addressable by the PDP-11. This is ordinary 16-bit PDP-11 core memory.

Node Manipulation

The process of transferring nodes from one deque structure to another.

Nodes

The list elements of a deque. All nodes consist of listhead, followed by 2 words of data (list elements).

Nul Task

The Nul Task is a task which runs when no other task can. It consists of only PDP-11 WAIT and BR Instruction to increase UNIBUS operations.

Permanent Task

A task in PIREX is said to be a permanent task if it is assembled into PIREX, has space in all PIREX system tables and has a fixed task code number.

POOL

A linked list of empty four-word nodes for use in any deque in the system. The POOL is generated at assembly time and currently has 20 decimal nodes available.

Pop

To remove an Item (word) from the current task's stack.

Push

To put an item (word) onto the current task stack.

Queue

To enter into a waiting list. Queues in PIREX consist only of deque structures.

Scheduling

The process of determining which task will be executed next. The operation is based on a priority ordered list of active tasks in the system (ATL).

Shared Memory

Core memory addressable by both the PDP-15 and PDP-11. The shared memory is ordinary 18-bit PDP-15 memory.

Spare Task

A task that runs under PIREX is said to be a temporary task if it is not assembled into PIREX, has space in all PIREX system tables, does not have a fixed task code number and its start address is not fixed.

The core occupied by the temporary tasks is not freed unless the tasks are disconnected in the order in which they were connected.

SPOLSW

This is a register in PIREX which contains the spooler control and status switches as indicated below.

BITS 0-7 Device busy Idle switch
'0' if idle and '1' busy

BIT 0	LP
1	CD
2	PL
3-7	UNUSED

BITS 8-15 Spooler State/Function switches
'0' if disabled and '1' if enabled

BIT 12	DESPOOLER
13	SPOOLER
14	SPOOLING
15=1	SPOL11 PROGRAM CONNECTED TO PIREX
=0	SPOL11 PROGRAM NOT CONNECTED TO PIREX

Task

A PDP-11 software routine capable of being requested by the PDP-15 or PDP-11 through the PIREX software system. The task may be a device driver, a Directive, or just a software routine used to carry out a specified function. A task must have the format shown in Figure 2-1.

Task Code Number

All tasks in the PIREX system are differentiated by a numbering system rather than by name. Task Code Numbers are used in TCBs and are currently assigned as follows:

CODE	
-1	CL task
200	ST task
201	SD task
202	RK Driver task
203	DT Driver task
4	LP Driver task
5	CD Driver task
6	PL Driver task
7	SPOOLER task
11	currently not used
12	currently not used
13	currently not used

TCB - Task Control Block

A set of contiguous memory locations (minimum of three) which contain all necessary information for a task to complete its request. The contents of the TCB must be defined prior to the request by the requesting program (e.g., a PDP-15 program).

A pointer to the TCB (called a TCBP) is then passed to the PDP-11 via the LIOR instruction in the PDP-15 or the IREQ macro in the PDP-11 to actually initiate the request.

TCBP - Task Control Block Pointer

A pointer to a TCB. This pointer is passed to the PDP-11 either via the LIOR instruction in the PDP-15 or the IREQ macro in the PDP-11 when initiating a request to PIREX.

INDEX

HOW TO OBTAIN SOFTWARE INFORMATION

SOFTWARE NEWSLETTERS, MAILING LIST

The Software Communications Group, located at corporate headquarters in Maynard, publishes newsletters and Software Performance Summaries (SPS) for the various Digital products. Newsletters are published monthly, and contain announcements of new and revised software, programming notes, software problems and solutions, and documentation corrections. Software Performance Summaries are a collection of existing problems and solutions for a given software system, and are published periodically. For information on the distribution of these documents and how to get on the software newsletter mailing list, write to:

Software Communications
P. O. Box F
Maynard, Massachusetts 01754

SOFTWARE PROBLEMS

Questions or problems relating to Digital's software should be reported to a Software Support Specialist. A specialist is located in each Digital Sales Office in the United States. In Europe, software problem reporting centers are in the following cities.

Reading, England
Paris, France
The Hague, Holland
Tel Aviv, Israel
Milan, Italy
Solna, Sweden
Geneva, Switzerland
Munich, West Germany

Software Problem Report (SPR) forms are available from the specialists or from the Software Distribution Centers cited below.

PROGRAMS AND MANUALS

Software and manuals should be ordered by title and order number. In the United States, send orders to the nearest distribution center.

Digital Equipment Corporation
Software Distribution Center
146 Main Street
Maynard, Massachusetts 01754

Digital Equipment Corporation
Software Distribution Center
1400 Terra Bella
Mountain View, California 94043

Outside of the United States, orders should be directed to the nearest Digital Field Sales Office or representative.

USERS SOCIETY

DECUS, Digital Equipment Computer Users Society, maintains a user exchange center for user-written programs and technical application information. A catalog of existing programs is available. The society publishes a periodical, DECUSCOPE, and holds technical seminars in the United States, Canada, Europe, and Australia. For information on the society and membership application forms, write to:

DECUS
Digital Equipment Corporation
146 Main Street
Maynard, Massachusetts 01754

DECUS
Digital Equipment Corporation
International (Europe)
P.O. Box 340
1211 Geneva 26
Switzerland

UNICHANNEL-15
SYSTEM SOFTWARE
MANUAL
DEC-15-XUCMA-A-D

READER'S COMMENTS

NOTE: This form is for document comments only. Problems with software should be reported on a Software Problem Report (SPR) form (see the HOW TO OBTAIN SOFTWARE INFORMATION page).

Did you find errors in this manual? If so, specify by page.

____
____
____
____
____
____

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

____
____
____
____
____
____

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

____
____
____
____
____
____

Please indicate the type of user/reader that you most nearly represent.

- □ Assembly language programmer
- □ Higher-level language programmer
- □ Occasional programmer (experienced)
- □ User with little programming experience
- □ Student programmer
- □ Non-programmer interested in computer concepts and capabilities

Name ______________________ Date ______________

Organization ______________________________

Street ______________________________

City ______________ State ________ Zip Code ________
or
Country

If you do not require a written reply, please check here. □

-------------------- **Fold Here** --------------------

-------------------- **Do Not Tear - Fold Here and Staple** --------------------

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

```
Software Communications
P. O. Box F
Maynard, Massachusetts  01754
```