XVM UNICHANNEL SOFTWARE MANUAL

DEC-XV-XUSMA-A-D

XVM Systems

digital

XVM UNICHANNEL SOFTWARE MANUAL

DEC-XV-XUSMA-A-D

digital equipment corporation · maynard. massachusetts

First Printing, December 1975

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.

CONTENTS

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

FIGURES

TABLE

LIST OF ALL XVM MANUALS

The following is a list of all XVM manuals and their DEC numbers, including the latest version available. Within this manual, other XVM manuals are referenced by title only. Refer to this list for the DEC numbers of these referenced manuals.

BOSS XVM USER'S MANUAL	DEC-XV-OBUAA-A-D
CHAIN XVM/EXECUTE XVM UTILITY MANUAL	DEC-XV-UCHNA-A-D
DDT XVM UTILITY MANUAL	DEC-XV-UDDTA-A-D
EDIT/EDITVP/EDITVT XVM UTILITY MANUAL	DEC-XV-UETUA-A-D
8TRAN XVM UTILITY MANUAL	DEC-XV-UTRNA-A-D
FOCAL XVM LANGUAGE MANUAL	DEC-XV-LFLGA-A-D
FORTRAN IV XVM LANGUAGE MANUAL	DEC-XV-LF4MA-A-D
FORTRAN IV XVM OPERATING ENVIRONMENT MANUAL	DEC-XV-LF4EA-A-D
LINKING LOADER XVM UTILITY MANUAL	DEC-XV-ULLUA-A-D
MAC11 XVM ASSEMBLER LANGUAGE MANUAL	DEC-XV-LMLAA-A-D
MACRO XVM ASSEMBLER LANGUAGE MANUAL	DEC-XV-LMALA-A-D
MTDUMP XVM UTILITY MANUAL	DEC-XV-UMTUA-A-D
PATCH XVM UTILITY MANUAL	DEC-XV-UPUMA-A-D
PIP XVM UTILITY MANUAL	DEC-XV-UPPUA-A-D
SGEN XVM UTILITY MANUAL	DEC-XV-USUTA-A-D
SRCCOM XVM UTILITY MANUAL	DEC-XV-USRCA-A-D
UPDATE XVM UTILITY MANUAL	DEC-XV-UUPDA-A-D
VP15A XVM GRAPHICS SOFTWARE MANUAL	DEC-XV-GVPAA-A-D
VT15 XVM GRAPHICS SOFTWARE MANUAL	DEC-XV-GVTAA-A-D
XVM/DOS KEYBOARD COMMAND GUIDE	DEC-XV-ODKBA-A-D
XVM/DOS READERS GUIDE AND MASTER INDEX	DEC-XV-ODGIA-A-D
XVM/DOS SYSTEM MANUAL	DEC-XV-ODSAA-A-D
XVM/DOS USERS MANUAL	DEC-XV-ODMAA-A-D
XVM/DOS V1A SYSTEM INSTALLATION GUIDE	DEC-XV-ODSIA-A-D
XVM/RSX SYSTEM MANUAL	DEC-XV-IRSMA-A-D
XVM UNICHANNEL SOFTWARE MANUAL	DEC-XV-XUSMA-A-D

PREFACE

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

No attempt is made in this document to describe the various UNICHANNEL hardware instructions; those are explained in the UNICHANNEL-15 SYStem Maintenance Manual. 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 UNICHANNEL hardware components before attempting to proceed with this manual. The user who plans to use the UNICHANNEL Software System in conjunction with some operating system on the XVM, and not modify it, should gain a thorough understanding of Chapter 1 of this manual. Users who wish to modify the UNICHANNEL XVM Software System should read the UNICHANNEL XVM System Maintenance Manual. In addition, a knowledge of PDP-11 and its assembly language is necessary before attempting UNICHANNEL 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 UNICHANNEL System:

MAC11 XVM Assembler Language Manual
XVM/DOS Users Manual
XVM/DOS System Manual
XVM UNICHANNEL Software Manual
Instruction List for the PDP-15
XVM Systems Reference Manual
XVM/DOS V1A System Installation Guide
RK11-E Controller Manual DEC-11-HRKA-B-D

CHAPTER 1

INTRODUCTION

1.1 XVM UNICHANNEL SOFTWARE COMPONENTS

The XVM UNICHANNEL Software System consists of the following four components:

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

1.1.1 XVM/PIREX

XVM/PIREX (peripheral executive), a component of the XVM UNICHANNEL (UC15) Software System, is described in Chapters 3 and 4 of this manual. XVM/PIREX (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 DIGITAL XVM (XVM) 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 XVM 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 XVM or the PDP-11.

1.1.2 SPOL11

Spooling is a method by which data to and from slow peripherals is buffered on an RK05 disk. Spooling allows the XVM 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 per-

mits 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 1,800,000 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 XVM Assembler Language Manual.

1.1.4 ABSL11

ABSL11 is a XVM Hardware Read In Mode (HRM) paper tape program used to bootstrap-load the UNICHANNEL peripheral processor with MAC11-generated 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 UNICHANNEL Support Programs

1.1.5.1 Spooler Disk Area Generation (SPLGEN) - SPLGEN allows the user to dynamically create or alter the RK disk area used by the UNICHANNEL spooler on any RK disk unit (0 through 7).

1.1.5.2 Spooler Installation Program (SPLOAD) - SPLOAD allows the user to install, on the system disk, the SPOL11 paper tape produced by MAC11.

1.1.5.3 XVM Spooler Control Program (SPOOL) - SPOOL (SPOL15) is used to initiate or terminate UNICHANNEL spooling using any RK disk unit which has been previously prepared for spooling by SPLGEN.

1.1.5.4 XVM MAC11 Control Program (MAC11) - MAC11 (MACINT) is used to initiate, perform Input/Output for, and terminate the MAC11 assembler.

1.1.5.5 MCLOAD - MCLOAD allows the user to install on the system disk, the MAC11 paper tape produced as a part of the XVM/DOS build process.

1.1.6 System Software Modification

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

1.2 UNICHANNEL HARDWARE SYSTEM

The UNICHANNEL hardware (see Figure 1-1) consists of a PDP-11 minicomputer used as an intelligent peripheral controller for the larger XVM main computer. The XVM 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 8,192 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.

1.2.1 Common Memory

Common memory is that memory directly accessible to both the master processor - the XVM, 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 XVM address space. The UNICHANNEL System allows any Non-Processor Request device on the UNIBUS to access XVM memory so that data can be transferred between I/O devices and common memory.

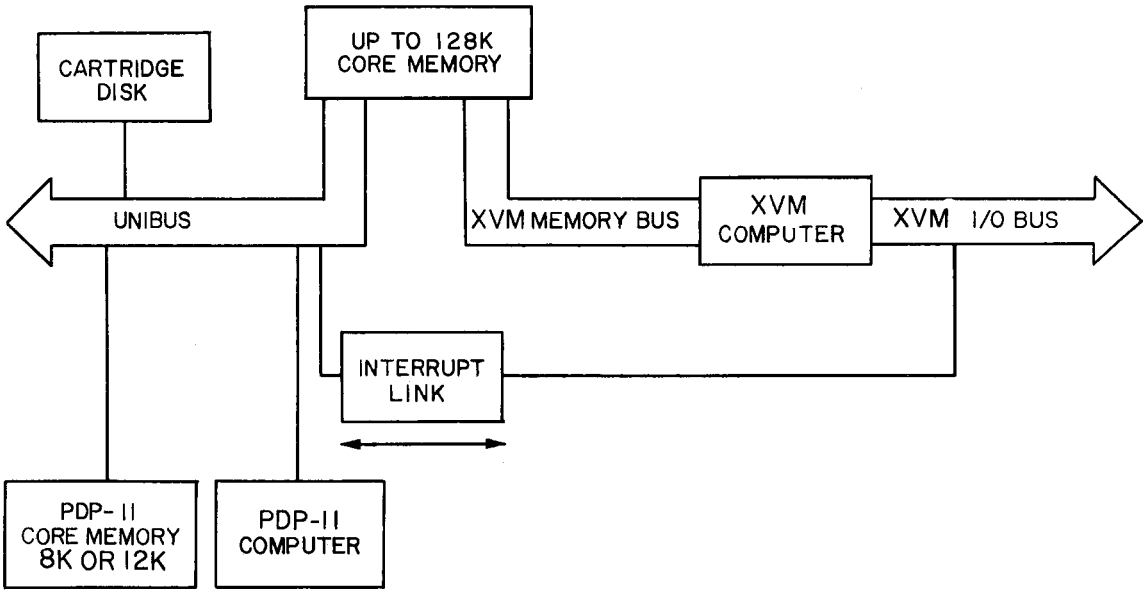


Figure 1-1
UNICHANNEL Hardware System

The use of common memory allows ease of data transfer between XVM 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
8K	20K
12K	16K

The UNIBUS can address the combined XVM/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 XVM and PDP-11, illustrating the common shared memory address space and the PDP-11 local memory.

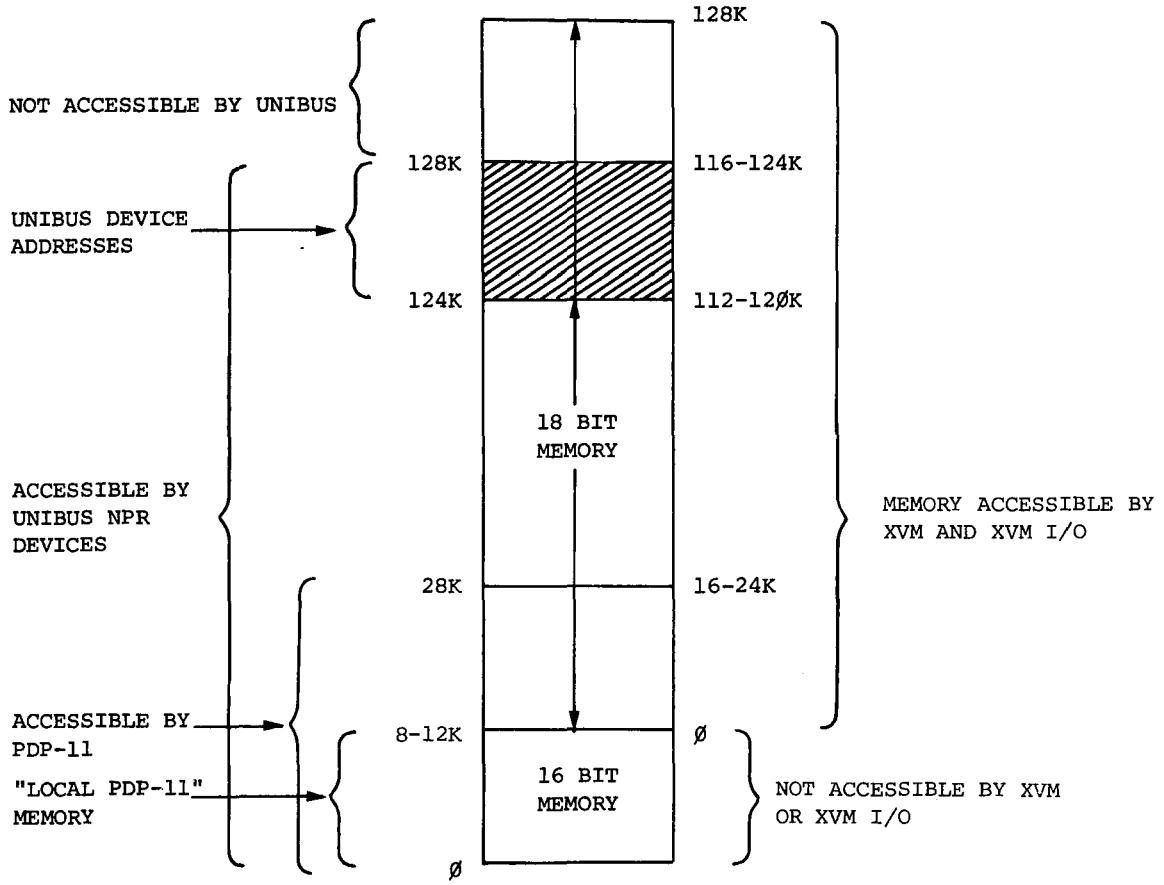


Figure 1-2
Memory Map of a UNICHANNEL System

1.2.2 Interrupt Link

The XVM central processor and the peripheral processor communicate with each other through device interfaces. When the XVM 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 XVM.

1.2.3 Peripheral Processor Hardware

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

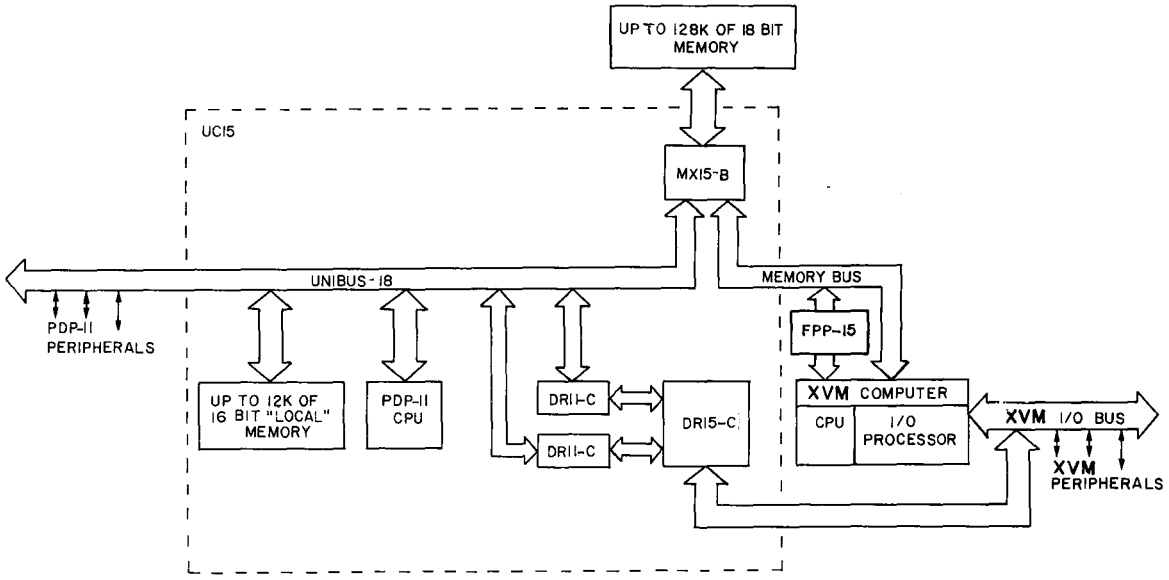


Figure 1-3
UNICHANNEL System

- PDP-11 Peripheral Processor
- DR15-C Device Interface
- Two DR11-C Device Interfaces
- XM15 Memory Bus Multiplexer
- 8192 or 12288 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 XVM. The DR15-C and the two DR11-C Device Interfaces provide the communication facility between the XVM and the PDP-11. The XVM 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 XVM to indicate an error condition or job completion at any one of 128 API vector locations at any one of four API priorities.[1] The XM15 Memory Bus Multiplexer functions as a memory bus switch to allow either the XVM or the PDP-11 to communicate with the common memory. The XM15 also provides the PDP-11 with the capability of performing byte instructions which reference XVM memory.

[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.

CHAPTER 2
LOADING AND EXECUTION

2.1 INTRODUCTION

This chapter explains how to get the DEC-supplied XVM UNICHANNEL Software System up and running. In addition, a list of the UNICHANNEL software components used in the various XVM monitor systems is included. For information on how to tailor the system to a specific configuration, see the XVM/DOS System Installation Guide.

2.2 LOADING THE SYSTEM

The UNICHANNEL system is activated by using ABSL11 to load the PIREX executive into the PDP-11 UNICHANNEL local memory. XVM/DOS is then bootstrapped and the system is ready to:

1. Continue running under XVM/DOS
2. Begin execution of BOSS XVM
3. Begin execution of XVM/RSX

2.2.1 ABSL11

ABSL11 is an XVM absolute binary paper tape program which is read into the XVM at location 17700_8 via the Hardware Read In Mode (HRM) on the XVM. It is used to load PDP-11 absolute binary paper tape on to the PDP-11. This self starting program is written in MACRO XVM and octal. (The PDP-11 code is written in octal and assembled with MACRO XVM.)

Load ABSL11 from the XVM High Speed Reader. XVM will then halt. Start the PDP-11 from its console switches at 140000. Note that the previous (DOS V3A) start addresses for ABSL11 can also be used. Once the PDP-11 is running, load the PDP-11 tape into the XVM High Speed Reader. Depress the Continue Switch on the XVM, and the paper tape will read in. Each data frame from the paper tape is transferred into the PDP-11 as soon as it is read. At the end of the tape, XVM will halt with the AC register equal to zero. If the paper tape has a start address, the

PDP-11 will begin execution at that address. If the paper tape does not have a start address, the PDP-11 will halt. To load another tape, place it in the XVM High Speed Reader, and continue both machines.

Checksum errors are detected by the XVM 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 XVM.

2.2.2 Loading ABSL11, XVM/PIREX, and XVM/DOS

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

1. Place the ABSL11 paper tape into the XVM 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.
3. Press the HALT switch on the PDP-11 UNICHANNEL console.
4. On the XVM console, set the address register switches to 17700 (octal), then press STOP and RESET simultaneously.
5. On the XVM console, press READ IN. The ABSL11 paper tape should read in.
6. When the paper tape reader stops, observe the XVM accumulator (AC) using the proper setting of the rotary register selector and register select switch on the XVM 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, there is either a bad ABSL11 paper tape or a hardware problem.)
7. On the PDP-11 UNICHANNEL console, load the starting address 140000 for the PDP-11 portion of ABSL11 into the switch registers:

   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 on.
9. Remove the ABSL11 paper tape from the reader and place the PIREX paper tape into it.
10. On the XVM 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 the XVM/DOS Bootstrap tape (hardware read in mode tape) into the Paper Tape Reader.

13. Set Address Switches on the XVM Console to

    a. 77637_8 for a 32K or more XVM

    b. 57637_8 for a 24K XVM

14. On the XVM Console, press simultaneously STOP and RESET.

15. On the XVM Console, press the READ IN switch. The XVM/DOS Bootstrap tape should read in.

16. XVM/DOS 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 RK unit 0.

2.3 PERIPHERAL OPERATION

2.3.1 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'click', 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).

2.3.2 Plotter

Unlike the XY311, the XY11 does <u>not</u> 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).

2.3.3 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 XVM 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 XVM.

2.3.4 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 ('1' state).

2.4 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 XVM 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.

2.4.1 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.

2.4.2 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.

2.4.3 Spooler Errors

During spooling operations, any unrecoverable disk error will result in the automatic termination of SPOOLing. Unrecoverable disk errors include:

> The attempt by the spooler to read/write a bad block on the disk cartridge.
>
> Setting the disk cartridge off line while SPOOLing is enabled. (This is detected even if no Input/Output to the disk cartridge is underway.)

The spooler is disconnected from PIREX upon the occurence of either of the above errors. The user may restart the spooler by issuing the XVM/DOS "SPOOL" command.

2.5 TASK CRASHES

During program development under PIREX on the PDP-11, the task under development may crash. Such crashes may not be apparent unless the PDP-11 halts, because PIREX keeps both the RUN light and bit 0 lit as if no problem existed.

2.6 UNICHANNEL RELATED SOFTWARE COMPONENTS

2.6.1 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 ***

2.6.2 XVM/DOS Components

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

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

* ABSL11 requires a special assembler, that is not available as a supported product. Assembly of ABSL11 with the standard MACRO XVM Assembler produces a paper tape with a load address of 17720.

** The MAC11 source is a PDP-11 DEC tape that must be assembled and linked under DOS/BATCH-11. This tape is not available as a part of the XVM/DOS kit.

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

**** These routines are versions of standard DOS-15 source files - created using special assembly parameters - see the XVM/DOS VIA System Installation Guide.

2.6.3 XVM/RSX Components

NOMENCLATURE	SOURCE FILE NAME	TASK NAME
RK05 Cartridge Disk File Handler	RFRES XXX	RK ....
Disk File Handler Overlay	RFOPEN XXX	RK ....
Disk File Handler Overlay	RFCLOS XXX	RK ....
Disk File Handler Overlay	RFREAD XXX	RK ....
Disk File Handler Overlay	RFDLET XXX	RK ....
Disk File Handler Overlay	RFCREA XXX	RK ....
Line Printer Handler	LP.XX SRC	LP ....
Card Reader Handler	CD.... XXX	CD ....
UNICHANNEL Poller	POLLER XXX	.POLLER
Plotter Handler	XY.XX SRC	XY ....
Executive	RSX.P1 XXX and RSX.P2 XXX	NA

CHAPTER 3
SYSTEM DESIGN AND THEORY OF OPERATION--PIREX

This chapter describes the design and theory of operation of the XVM UNICHANNEL Peripheral Processor Executive. Knowledge of this information is necessary to successfully modify the XVM UNICHANNEL 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 DIGITAL XVM main-processor. PIREX is designed to be as independent of the particular XVM operating system as possible, executing in conjunction with XVM/DOS, BOSS XVM, or XVM/RSX. 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 XVM 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 XVM 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 XVM request, an optional hardware interrupt is initiated in the XVM 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 XVM requests, the routines initiating the interrupts could actually be software routines used to simulate hardware conditions or

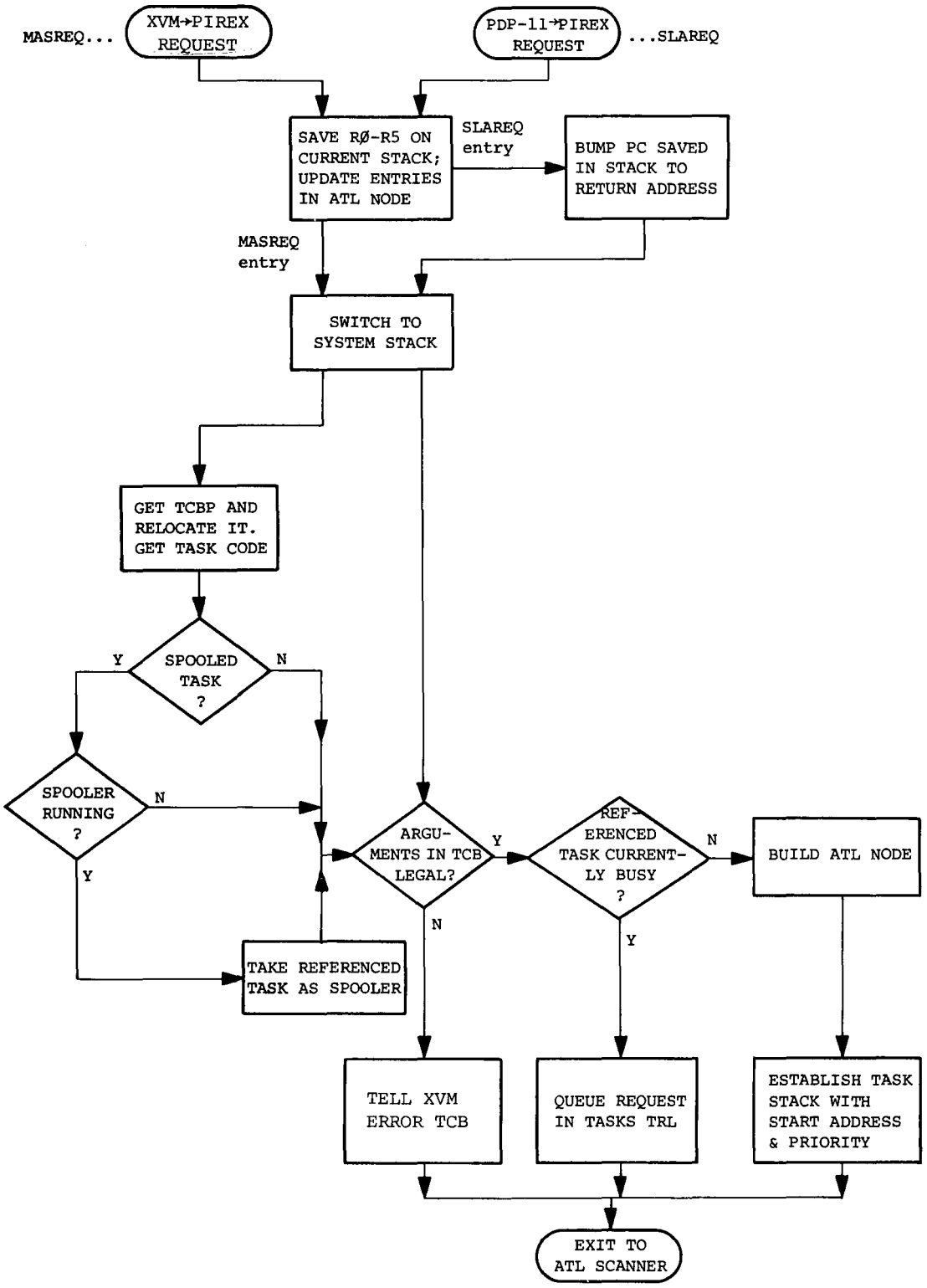


Figure 3-1
Basic Flow Chart of XVM/PDP-11 Request Processing

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.

3.1.2 PIREX Services

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 XVM 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 XVM executive, on demand.
- Directive Processing - providing the XVM 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 drive 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). 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.

3.1.6 Optional LV Support

For reasons of packaging optional LV support on a printer and a plotter is present in the standard PIREX ($LV=0). This support, however, is only at the device driver level. The PDP-15 side modules display-file-to-vector, vector-to-raster, and LV I/O handler may be purchased separately. Information is available through PDP-15 Marketing.

3.1.7 Optional DL Support

The DL-11 is supported as a communications protocal device between a DEC system-10 and a PDP-15. The code for this support is purchased separately and is available from the SDC. Information is available through PDP-15 Marketing.

3.1.8 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 (20.0 milliseconds for 50 Hz). 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 XVM 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 XVM 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 XVM 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 XVM 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 XVM 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 XVM 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.

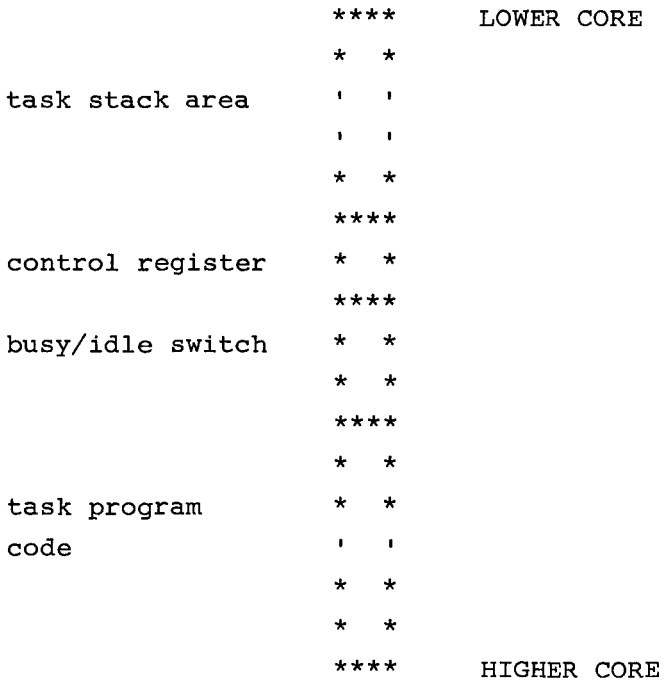


Figure 3-2
Task Format

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.

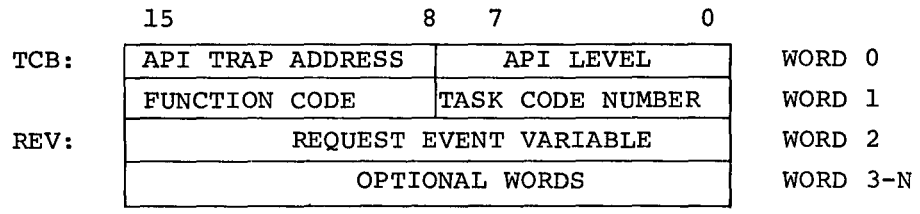
The "control register" is either a dummy address (an address which points to an unused software variable) or the address of a device

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 blocks 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.



3.2.5.1 API Trap Address and Level - The API trap address is a XVM API trap vector and has a value between 0 and 177_8 when a hardware interrupt on the XVM is required. Location 0 corresponds to location 0 in the XVM. The "API" level is the priority level at which the interrupt will occur in the XVM and has a value between 0 and 3 when a hardware interrupt on the XVM 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 XVM that the requested operation is complete (e.g., a disk block transferred or line printed). If the XVM 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 XVM using 4 UC15 skips (one per level) on the traditional skip chain.[1]

[1] API is optional on PDP-15's, standard on XVM's.

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 XVM 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	RF 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	DL task (Hurley protocal communication task)	Driver task
212	12	Currently not used	-
213	13	Currently not used	-
214	14	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 14 tasks. Tasks 11-14 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 XVM. For example,

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

[1]See Section 4.4 for further information.

3.3 SYSTEM TABLES AND LISTS

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

3.3.1 Active Task 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 3-3) 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.

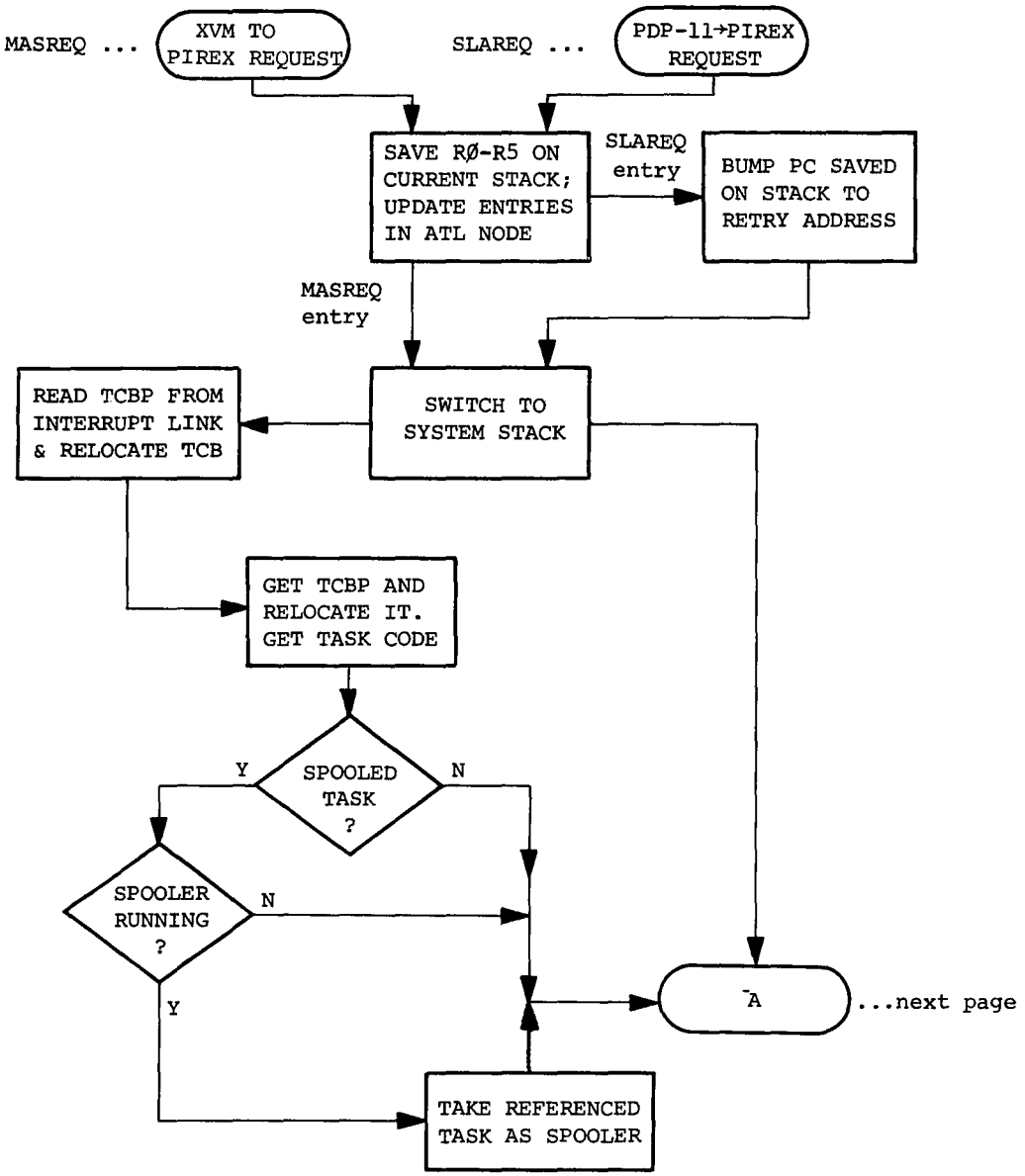


Figure 3-3
Detailed Flow Chart of XVM/PDP-11 Request Processing

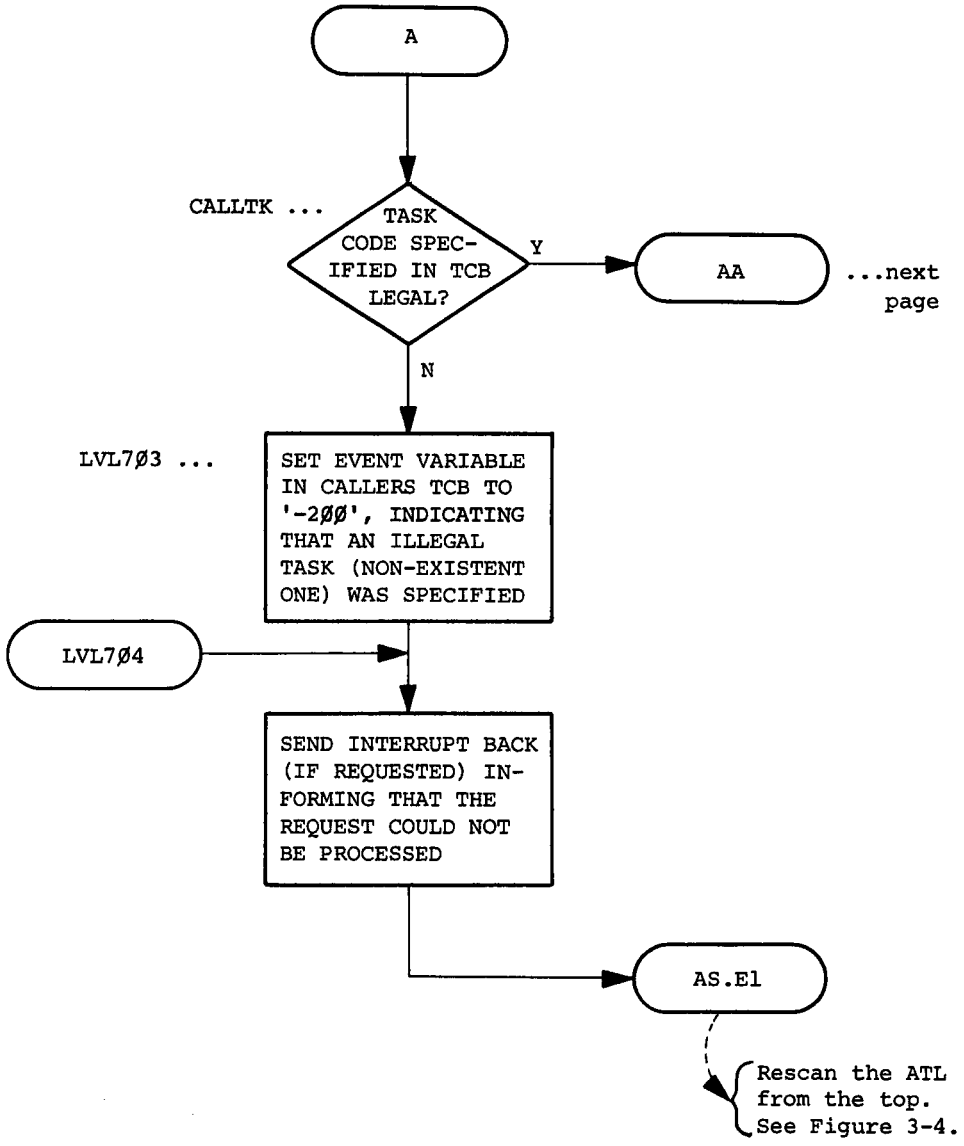


Figure 3-3 (Cont.)
Detailed Flow Chart of XVM/PDP-11 Request Processing

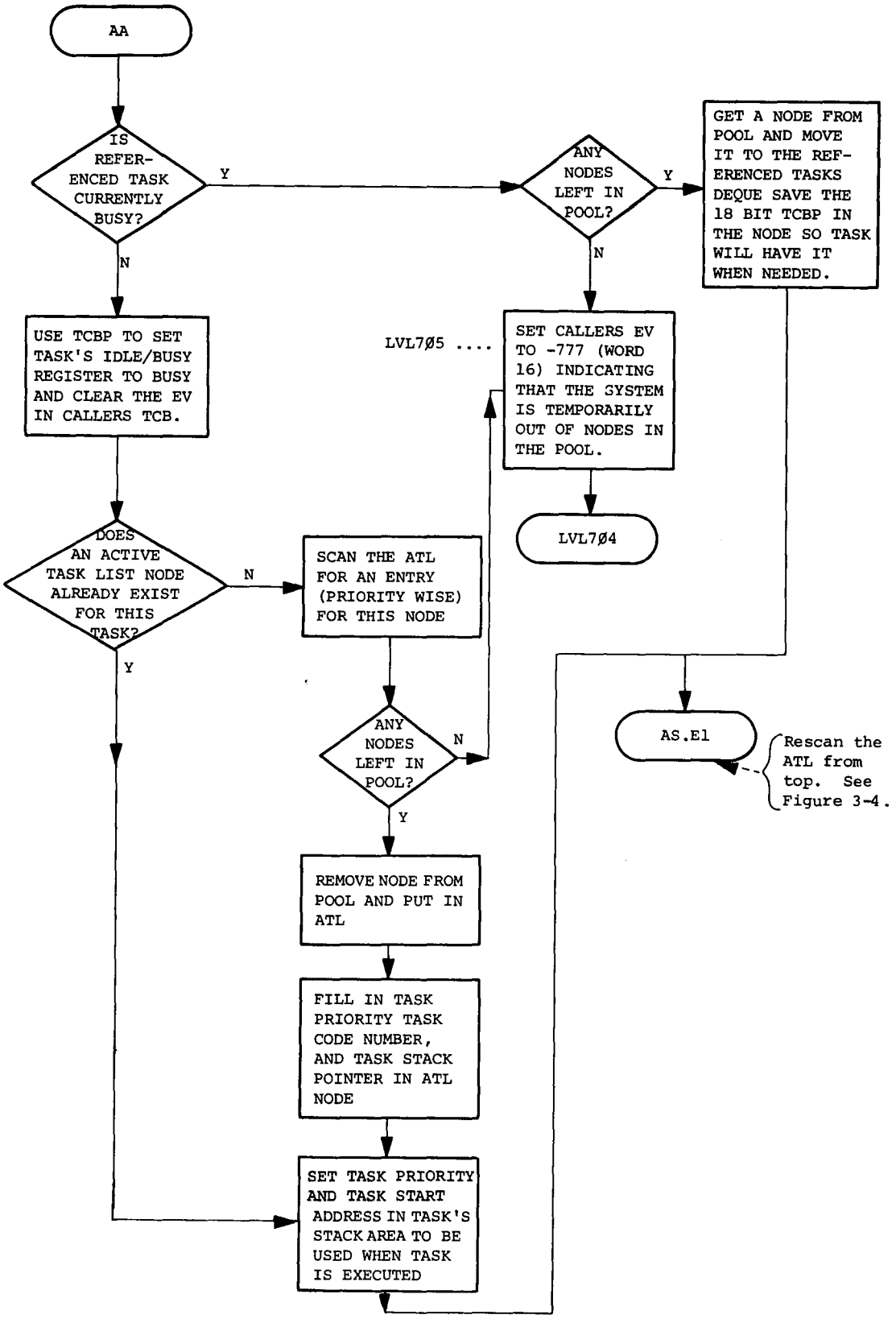


Figure 3-3 (Cont.)
Detailed Flow Chart of XVM/PDP-11 Request Processing

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.

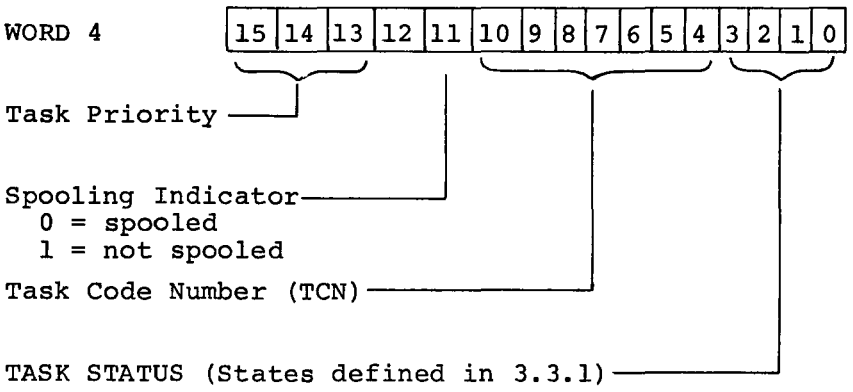
3.3.1.1 ATL Nodes - The Active Task List is a linked list containing 4 word entries called nodes.

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.

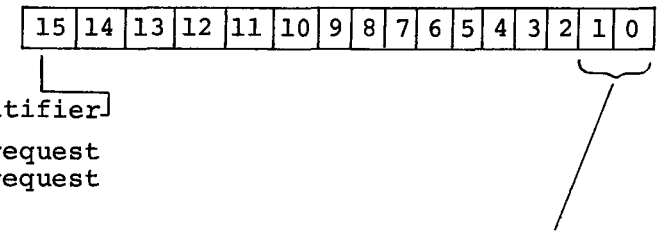
3.3.2 Task Request List (TRL)

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.

A TRL node has the following structure:

WORD 1 - Forward pointer to next node.
WORD 2 - Backward pointer to previous node.

WORD 3 -



WORD 4 - 16 least significant bits of TCBP (XVM 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.

---

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

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
```

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 Hz 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 XVM monitor. The XVM monitor contains a routine called the

---

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

"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 XVM 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 (used to report bad block numbers, and, to report disconnected spooler unit)

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 sub-routines. 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	XVM CTL C RUNNING COUNTER
01067	SPUNIT: .WORD	0	UNIT 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 XVM 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 a memory management option[1] on the PDP-11. Requests from the XVM 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 XVM 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 XVM 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.

---

[1] See Section 3.6 for additional information.

[2] See Section 3.5 for additional information.

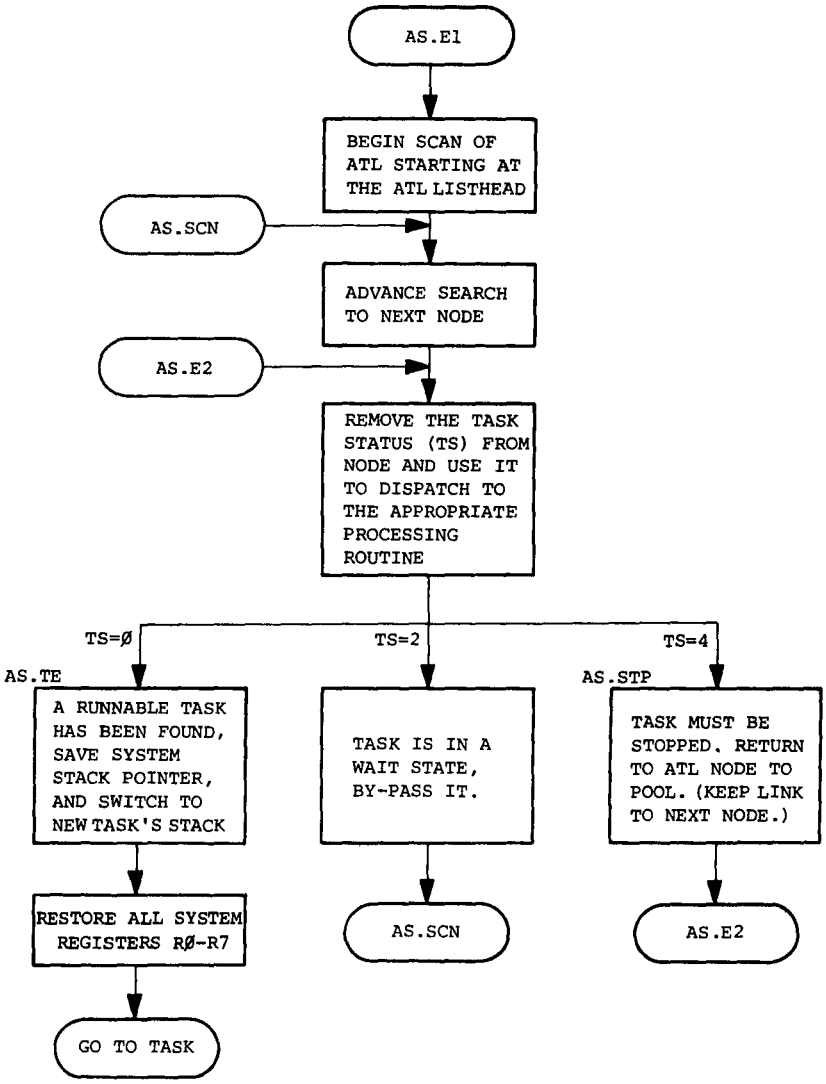


Figure 3-4
Scan of Active Task List (ATL)

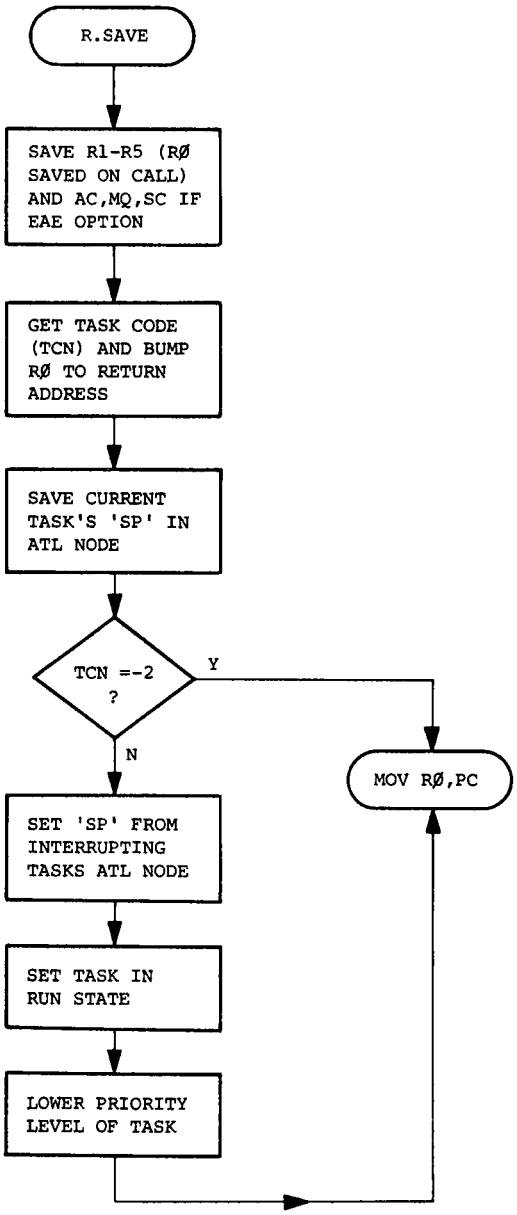


Figure 3-5
Context Switch or Save General Purpose Registers R0-R5

In the case of a XVM 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 XVM. The TCBP is present in R4 and R5 when the IREQ macro 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.

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 XVM/PDP-11 and may be used by the XVM 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 XVM). The hardware interrupt to the XVM (see Figure 3-6) is optional and can be made at any of the XVM 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.

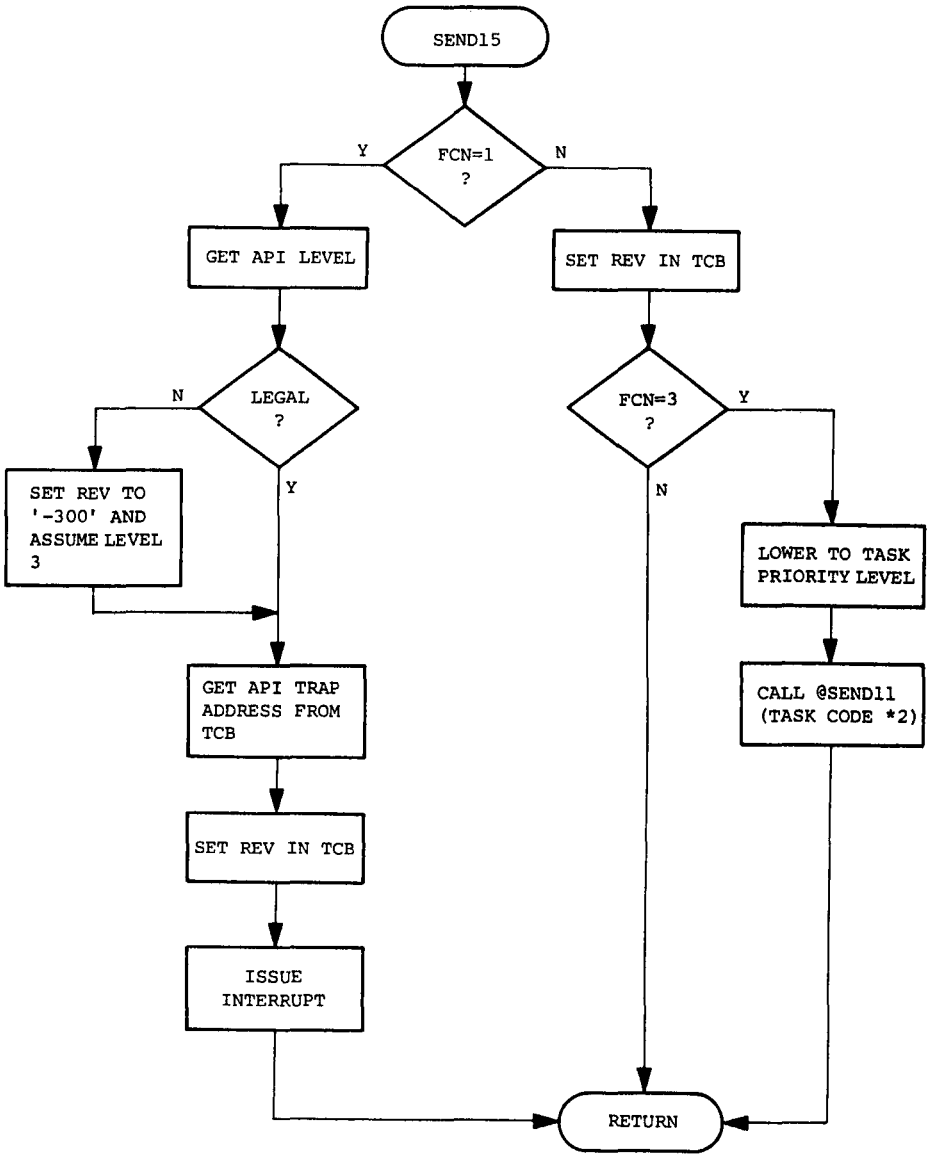


Figure 3-6
Send Hardware Interrupt to XVM/Software Interrupt to PDP-11

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 XVM 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 XVM 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.

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 XVM requests for the indicated task(s). There are four possibilities:

1. Stop all tasks unconditionally and cancel all pending XVM requests
2. Stop a given task unconditionally and cancel all pending XVM requests to that task
3. Cancel all XVM requests to all tasks - this has no effect on PDP-11 requests
4. Cancel all XVM requests to a given task - this has no effect on PDP-11 requests

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

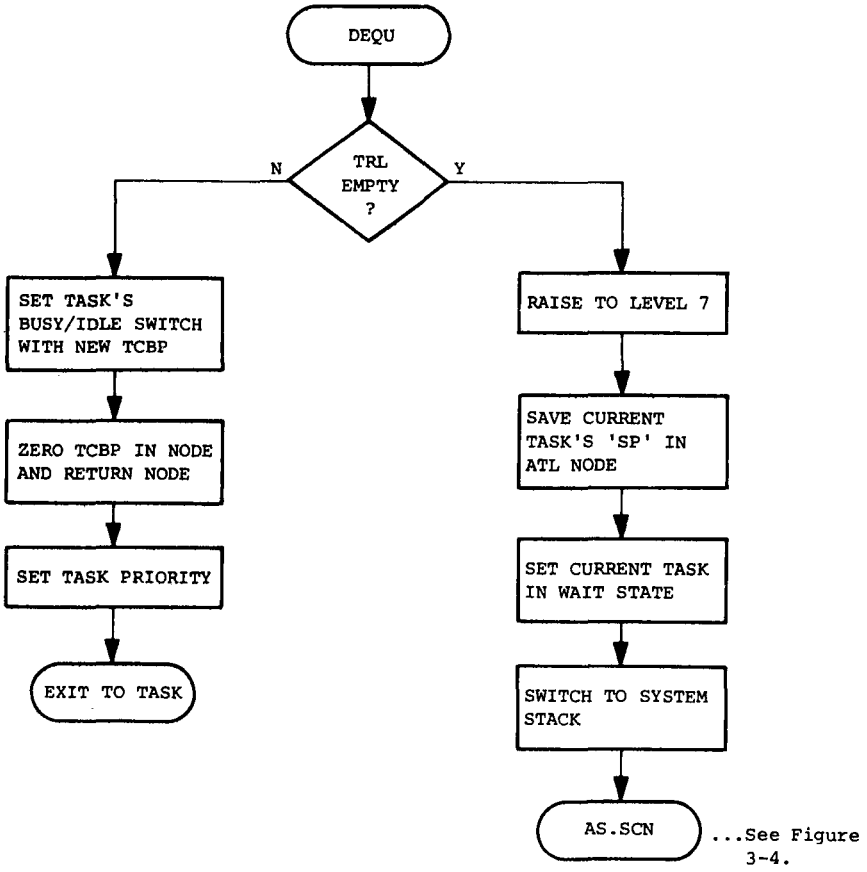
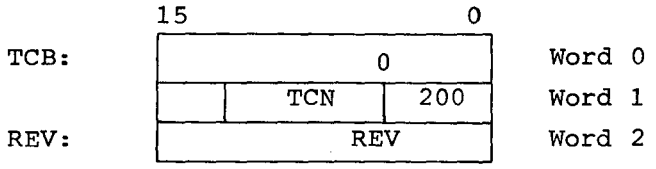


Figure 3-7
Dequeue Node From Task's Deque

1. Removal of all appropriate XVM 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 not allowed.)

The Stop Tasks TCB has the following format:



Word 1 bit 15 = 1 cancel XVM requests and the current pending request unconditionally.

bit 15 = 0 cancel XVM requests

TCN = 0 cancel all tasks

TCN ≠0 cancel Task TCN only

Word 2 REV = Return Event Variable

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

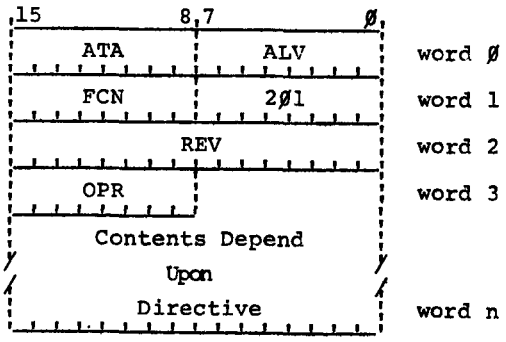
3.6 SOFTWARE DIRECTIVE PROCESSING

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

1. The capability to connect and disconnect temporary tasks to PIREX (such as MAC11
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:



ATA XVM API interrupt vector address

ALV XVM 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 XVM 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

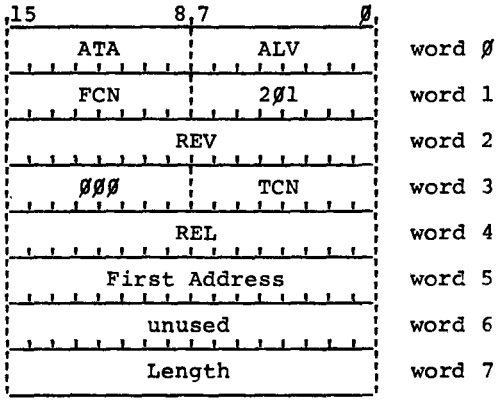
Returned REV values

- 1 Successful completion
- -300 Invalid ALV value. The request may or may not have been performed - see individual directive descriptions. The XVM 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:



TCN	The task code number of the task to be disconnected.
REL	000000 if the task resides in XVM 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 XVM 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 XVM 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 unusuable memory) in the PDP-11's local memory.

4. Tasks should be disconnected in a 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.

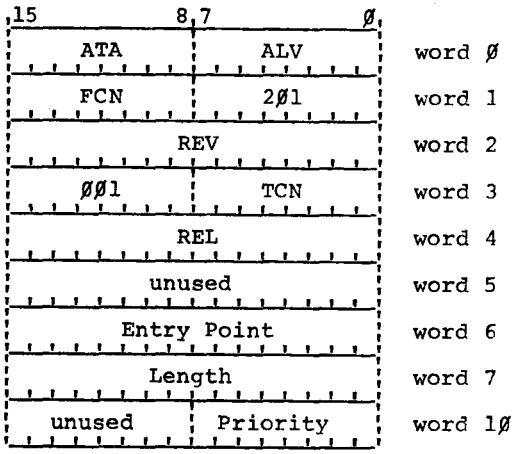
-300 Invalid ALV value, the task may or may not have been disconnected, its memory may or may not have been released.

-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 XVM 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:



TCN	The new task's task code number (TCN)
REL	000000 if the new task resides in XVM 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 XVM word address if the new task resides in XVM 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 XVM 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 XVM 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 XVM memory, it must reside entirely within the area addressable by the PDP-11's 28K addressing range.

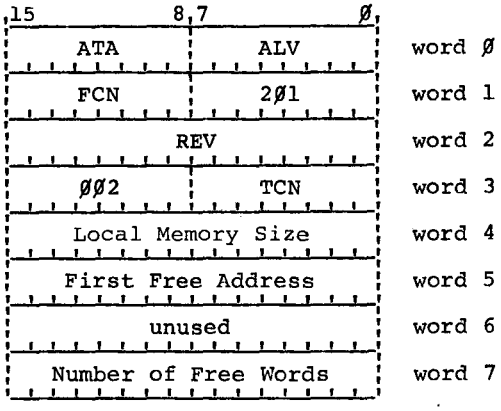
6. Tasks should be connected in sequential order by task code numbers. Temporary tasks (tasks which will subsequently be disconnected) should always be connected to a task code number one higher than that obtained via the core status report software directive.

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:



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:

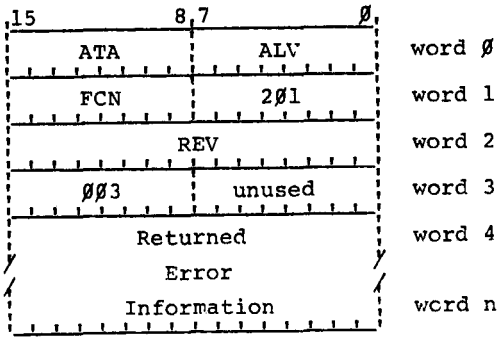
1. The core status report software directive has no restrictions. However, the restrictions (especially those regarding order of use of memory and task code numbers) on the connect and disconnect software directives must be adhered to in order to have valid information returned by the core status report.

Returned REV values:

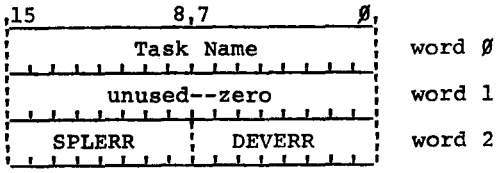
1	Successful completion
-300	Invalid ALV value. No information returned.
-500	No free PDP-11 memory. No information returned.

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:



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:



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

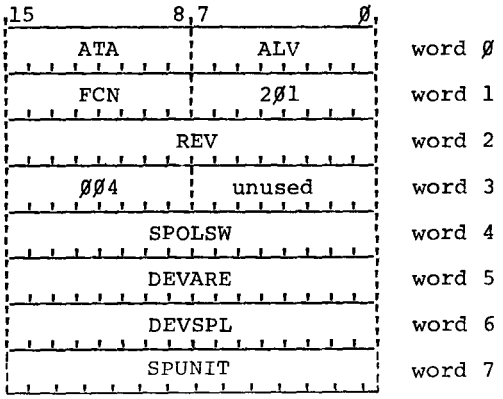
RESTRICTIONS: none

Returned REV values:

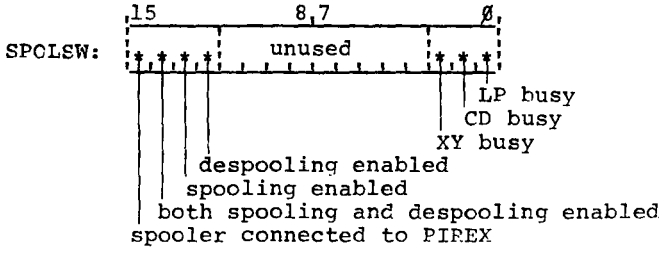
1	Successful completion.
-300	Invalid ALV value. Information has been returned.

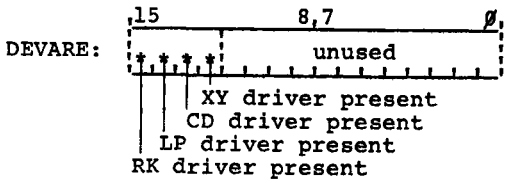
3.6.5 Spooler Status Report Directive

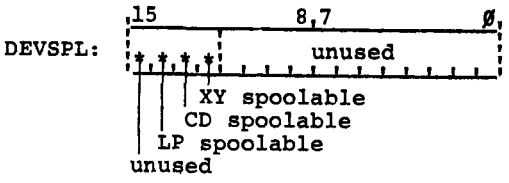
The spooler status report software directive returns information regarding spooler status and devices present in PIREX. The format of the task control block for the spooler status report software directive is as follows:



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:

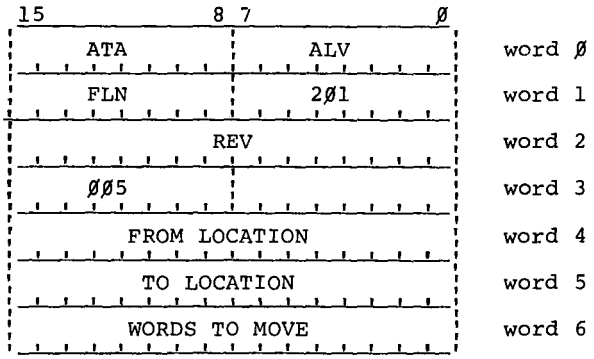
1	Successful completion.
-300	Invalid ALV value. Information has been returned.

3.6.6 PIREX MOVE Directive

NOTE

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

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:



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.

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 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 XVM 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 XVM Local Memory, and

   b. not overlaying any of the XVM 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 XVM/RSX 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 XVM/DOS, the TCB placement problem is somewhat more complex. The standard XVM/DOS 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 XVM 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 XVM/DOS Systems Manual.)

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 Language Manual.)

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).
3. Optionally update the interrupt vector table if the task is a device driver task (see Section 3.3.9).

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).

[1] PIREX transfers, upon request, the entire DEVST Table to the XVM/DOS monitor. The XVM/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 XVM/DOS resident monitor.

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).
3. Optionally update the interrupt vector table if the task is a device driver task (see Section 3.3.9).

4.6 CONSTRUCTING DEVICE HANDLERS

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

4.6.1 Constructing a XVM/DOS UNICHANNEL Device Handler

The following description of how to construct a handler for the XVM/DOS monitor does not discuss those topics related to all XVM/DOS handlers both traditional and UNICHANNEL. General issues pertaining to all XVM/DOS device handlers can be found in the XVM/DOS Systems Manual. The UNICHANNEL Line Printer handler is used 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 48-55, 71-73). These constants include:

```
  2     LPU. XVM V1A 122
  5     CAL ENTRANCE
  6     INTERRUPT SERVICE
  7     ERROR ROUTINE
  8     .INIT FUNCTION
  9     .WRITE FUNCTION
 15     .CLOSE FUNCTION
 16     .WAIT FUNCTION
 17     INITIALIZATION CODE AND TEMPORARIES


PAGE  1     LPU.  122

  1                             .SYSID <        .TITLE LPU. >,< 122>
                        *G      .DEFIN .SYSID,FR,BK
                        *G FR@XVM V1A@BK
                        *G      .ENDM
                        *G      .SYSID <        .TITLE LPU. >,< 122>


PAGE  2     LPU.  122     LPU. XVM V1A 122

                         *G         .TITLE LPU. XVM V1A 122
  2                         /
  3                         /COPYRIGHT (C) 1975
  4                         /DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
  5                         /
  6                         /THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY
  7                         /ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH
  8                         /THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS
  9                         /SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PRO-
 10                         /VIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
 11                         /EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO
 12                         /THESE LICENSE TERMS.  TITLE TO AND OWNERSHIP OF THE
 13                         /SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.
 14                         /
 15                         /THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE
 16                         /WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COM-
 17                         /MITMENT BY DIGITAL EQUIPMENT CORPORATION.
 18                         /
 19                         /DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY
 20                         /OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
 21                                .EJECT


PAGE  3     LPU.  122     LPU. XVM V1A 122

 22                         /
 23                         / EDIT LEGEND.
 24                         /
 25                         / 120   05-JUN-75 (RCHM)        MAKE XVM CHANGES.
 26                         / 121   05-JUN-75 (RCHM)        TAKE OUT NON-ESSENTIAL CONDITIONALS.
 27                         / 122   22-JUL-75 (RCHM)        TEST STATE OF UC15 ENABLED BIT.
 28                         /
 29                                .EJECT
```

Figure 4-1
XVM LP11 DOS Handler

```
PAGE   4     LPU.   122     LPU. XVM V1A 122

  30                          /J.M. WOLFBERG  (S. ROOT)
  31                          /LPU.--IOPS LINE PRINTER HANDLER FOR LP11 LINE PRINTER
  32                          /CALLING SEQUENCE:
  33                          /       CAL + .DAT SLOT (9-17)
  34                          /       FUNCTION
  35                          /       N ARGS, WHERE N IS A FUNCTION OF "FUNCTION"
  36                          /       NORMAL RETURN
  37                          /BITS 12-13 OF .SCOM+4 INDICATE PRINTER.
  38                          /       00= UNDEFINED.
  39                          /       01= 80 COLUMNS.
  40                          /       10= 120 COLUMNS.
  41                          /       11= 132 COLUMNS.
  42                          /ASSEMBLY PARAMETERS:
  43                          /   NOFF=1 INHIBITS AUTOMATIC END OF PAGE FORM FEED
  44                          /   FFCNT CAN BE DEFINED AS NUMBER OF LINES PER PAGE IF NOFF UNDEF.
  45                          /   DEFINE FFCNT IN !!OCTAL!!
  46                          /   IF FFCNT AND NOFF BOTH UNDEF., 58 LINES PER PAGE IS DEFAULT.
  47                          /
  48              000002 A    APILVL=2
  49              000056 A    APISLT=56
  50                          /
  51              706141 A    LSSF=APILVL*20+706101
  52              706001 A    S10A=706001                     /SKIP ON DATA ACCEPTED BY THE PDP11
  53              706006 A    LIOR=706006                     /CLEAR "DONE" FLAG AND LOAD REG FOR
  54                                                          /    THE PDP11.
  55              706144 A    CAP1=APILVL*20+706104           /CLEAR FLAG
  56                          /
  57              000100 A    .SCOM=100
  58              000104 A    SC.MOD=.SCOM+4                  /(RCHM=122) .SCOM MODE REGISTER.
  59              000002 A    SC.UC15=2                       /(RCHM=122) BIT WITHIN SC.MOD TO BE TESTED.
  60              000003 A    .MED=3
  61              440000 A    IDX=ISZ
  62              440000 A    SET=ISZ                         /USED TO SET SWITCHES TO NON-ZERO.
  63              000137 A    EXERRS=.SCOM+37
  64                          /
  65                          .IFUND FFCNT
  66              000072 A    FORMS=72
  67                          .ENDC
  71                          .IFUND NOSPL
  72              000004 A    DEVCOD=4                /CODE FOR LP DRIVER IN PIREX
  73                          .ENDC
  77                          .GLOBL LPA.


PAGE   5     LPU.   122     CAL ENTRANCE

  78                          .TITLE CAL ENTRANCE
  79        00000 R 040540 R  LPA.    DAC     LPCALP          /SAVE CAL POINTER.
  80        00001 R 040541 R          DAC     LPARGP          /AND ARGUMENT POINTER.
  81        00002 R 440541 R          IDX     LPARGP          /POINTS TO WORD 2 - FUNCTION CODE.
  82                          /
  83                          /  FIRST TIME THRU GO CAL INIT. CODE IN LBF
  84                          /
  85        00003 R 600547 R  NEW     JMP     INIT    /FIRST TIME THRU DO SETUP CAL
  86                          /                       /AND SET-UP TCB AND BUFFER, OVERWRITE
  87                          /                       /JUMP WITH NO-OP
  88                          /
  89        00004 R 220541 R          LAC*    LPARGP
  90        00005 R 440541 R          IDX     LPARGP          /POINTS TO WORD 3 - BUFFER ADDRESS.
  91        00006 R 500633 R          AND     (17777          /STRIP OFF UNIT NUMBER.
  92        00007 R 340634 R          TAD     (JMP LTABL-1    /DISPATCH TO PROCESS FUNCTION.
  93        00010 R 040011 R          DAC     .+1
  94        00011 R 740040 A          XX
  95        00012 R 600103 R  LTABL   JMP     LP1N            /1 - .INIT
  96        00013 R 741000 A          SKP                     /2 - .FSTAT,.RENAM,.DLETE - IGNORE
  97        00014 R 600024 R          JMP     LPERO6          /3 - .SEEK - ERROR
  98        00015 R 440541 R          IDX     LPARGP          /4 - .ENTER - IGNORE
  99        00016 R 600134 R          JMP     LPNEXT          /5 - .CLEAR - IGNORE
 100        00017 R 600466 R          JMP     LPCLOS          /6 - .CLOSE
 101        00020 R 600134 R          JMP     LPNEXT          /7 - .MTAPE - IGNORE
 102        00021 R 600024 R          JMP     LPERO6          /10 - .READ - ERROR.
 103        00022 R 600136 R          JMP     LPWRIT          /11 - .WRITE
 104        00023 R 600506 R          JMP     LPWAIT          /12 - .WAIT OR .WAITR
 105        00024 R 760006 A  LPERO6  LAW     6               /ILLEGAL HANDLER FUNCTION.
 106        00025 R 600073 R          JMP SETERR
 107        00026 R 760067 A  IOPS67  LAW 67                  /(RCHM=120) FETCH MEMORY BOUNDS ERROR MESSAGE.
 108        00027 R 600073 R          JMP SETERR              /(RCHM=120) GO PRINT ERROR.
 109        00030 R 760012 A  IOPS12  LAW 12                  /(RCHM=122) FETCH TERMINAL I/O ERROR MESSAGE.
 110        00031 R 600073 R          JMP SETERR              /(RCHM=122) GO PRINT ERROR.
```

Figure 4-1 (Cont.)
XVM LP11 DOS Handler

```
PAGE   6     LPU.   122      INTERRUPT SERVICE

 111                                .TITLE INTERRUPT SERVICE
 112                           /
 113                           /LPU, INTERRUPT SERVICE
 114       00032 R 600042 R    LPINT   JMP     LPPIC   /PIC ENTRY, JUMP TO CODE
 115       00033 R 040566 R            DAC     LPAC    /SAVE INTERRUPTED AC
 116       00034 R 200032 R            LAC     LPINT   /GET INTERRUPTED PC
 117       00035 R 040567 R            DAC     LPOUT   /SAVE FOR COMMON EXIT
 118       00036 R 200635 R            LAC     (JMP LPPIC /RESTORE PIC ENTRY
 119       00037 R 040032 R            DAC     LPINT
 120       00040 R 200636 R            LAC     (NOP    /WE DON'T NEED ION IN COMMON EXIT
 121       00041 R 600046 R            JMP     LPICM   /JOIN COMMON CODE
 122                           /
 123       00042 R 040566 R    LPPIC   DAC     LPAC    /PIC CODE, SAV AC
 124       00043 R 220637 R            LAC*    (0      /GET INTERRUPTED PC
 125       00044 R 040567 R            DAC     LPOUT   /SAVE
 126       00045 R 200640 R            LAC     (ION    /NEED INTERRUPT ON INST. IN COMMON CODE
 127       00046 R 040056 R    LPICM   DAC     LPISW
 128       00047 R 706144 A            CAPI            /CLEAR FLAG, NOW IN COMMON CODE
 129       00050 R 220553 R            LAC*    LPEV    /EVENT VARIABLE FROM PIREX
 130       00051 R 742010 A            RTL             /PDP-11 (MINUS) BIT TO OUR AC0
 131       00052 R 743120 A            SPA!RTR         /+ IS OK
 132       00053 R 600061 R            JMP     LPIERR  /ERROR, GO LOOK
 133       00054 R 140544 R    LPIRT   DZM     LPUND   /CLEAR UNDERWAY FLAG
 134       00055 R 200566 R    LPIRT1  LAC     LPAC    /RESTORE AC
 135       00056 R 740040 A    LPISW   HLT             /ION OR NOP
 136       00057 R 703344 A            DBR
 137       00060 R 620567 R            JMP*    LPOUT
 138                           /
 139                           /
 140       00061 R 500641 R    LPIERR  AND     (177777 /KEEP REAL 16 BITS FROM PDP-11
 141       00062 R 540642 R            SAD     (177001 /CODE FROM OUT OF NODES IN PIREX
 142       00063 R 600066 R            JMP     RETRY   /JUST TRY AGAIN, LEAVING UPUND SET
 143       00064 R 340643 R            TAD     (600000 /MAKE - NUMBER FOR IOPS
 144       00065 R 600073 R            JMP     SETERR  /TREAT AS REGULAR IOPS ERROR
 145                           /                       /NOTE THAT THIS SHOULDN'T HAPPEN.
 146                           /
 147                           /
 148       00066 R 200550 R    RETRY   LAC     LPTCB   /TCB ADDRESS
 149       00067 R 706001 A            SIOA
 150       00070 R 600067 R            JMP     .-1     /
 151       00071 R 706006 A            LIOR            /THIS MAGIC SHIPS TCB ADDR. TO PDP-11
 152       00072 R 600055 R            JMP     LPIRT1  /EXIT FROM INTERRUPT
 153                           /
 154                           /


PAGE   7     LPU.   122      ERROR ROUTINE

 155                                .TITLE  ERROR ROUTINE
 156                           /
 157       00073 R 040102 R    SETERR  DAC ERRNUM
 158       00074 R 740000 A    ERLOOP  NOP                     /'JMP LPTRY' IF IOPS 4 ERROR.
 159       00075 R 200102 R            LAC ERRNUM
 160       00076 R 120644 R    ERDUT   JMS* (EXERRS
 161       00077 R 600074 R            JMP ERLOOP
 162       00100 R 777777 A            LAW -1
 163       00101 R 142025 A            .SIXBT 'LPU'
 164       00102 R 000000 A    ERRNUM  0                       /HOLDS ERROR NUMBER FOR REPEAT.
```

Figure 4-1 (Cont.)
XVM LP11 DOS Handler

```
PAGE   8     LPU.   122     .INIT FUNCTION

 165                                .TITLE .INIT FUNCTION
 166                           /
 167                           /.INIT
 168                           /
 169        00103 R 220645 R    LPIN    LAC* (SC.MOD)           /(RCHM-122) CHECK MODE REGISTER FROM SCOM,
 170        00104 R 500646 R            AND (SC.UC15)           /(RCHM-122)  FOR UC15 ENABLED.
 171        00105 R 741200 A            SNA                     /(RCHM-122) IS IT?
 172        00106 R 600030 R            JMP IOPS12              /(RCHM-122) NO, GO PRINT ERROR.
 173        00107 R 440541 R            IDX LPARGP              /(RCHM-122)
 174        00110 R 200555 R            LAC BUFSIZ              /36(10) FOR 80 COLS; 56(10) FOR 132 COLS.
 175        00111 R 060541 R            DAC* LPARGP             /RETURN TO USER.
 176        00112 R 440541 R            IDX LPARGP              /NOW POINTS TO RETURN.
 177        00113 R 200542 R            LAC     PAGSIZ  /LF COUNTER
 178        00114 R 040543 R            DAC     PAGCNT
 179        00115 R 220540 R            LAC*    LPCALP  /DOES INIT INHIBIT AUTO FORMS FEED
 180        00116 R 500647 R            AND     (4000   /THIS IS INHIBIT BIT
 181        00117 R 340546 R            TAD     FFFF    /FFFF ASSEMBLED AS NOP FOR NOFF, ISZ IF NOT
 182        00120 R 540546 R            SAD     FFFF    /SKIP IF INIT INHIBITS FF
 183        00121 R 741000 A            SKP             /INIT DOESN'T INHIBIT, USE ASSEMBLED VALUE
 184        00122 R 200636 R            LAC     (NOP    /INIT INHIBITS IT, USE NOP
 185        00123 R 040545 R            DAC     FFSW    /THIS SWITCH XCT'ED BY FORMS CONTROL
 186                           /                         /SECTION IN PUTCH SUBROUTINE
 187        00124 R 100455 R            JMS     RESETL  /RESET TAB AND LINE WIDTH COUNTERS
 188        00125 R 100524 R            JMS     LP1OCK  /CHECK LP BUSY
 189        00126 R 140562 R            DZM     CUP     /SAY A FF OCCURRED
 190        00127 R 750030 A            CLA!IAC         /COUNT OF ONE BYTE FOR HEADER
 191        00130 R 060551 R            DAC*    LPBUF   /HEADER
 192        00131 R 723013 A            AAC     13      /FORM FEED
 193        00132 R 060552 R            DAC*    LPBUFD  /FOR BUFFER
 194                                    .IFUND  NOFF    /DO ONLY IF NOFF NOT DEFINED
 195        00133 R 100531 R            JMS     LPSET   /THIS SENDS REQ. TO PDP-11
 196                                    .ENDC
 197                           /
 198                           /NORMAL CAL EXIT
 199                           /
 200        00134 R 703344 A    LPNEXT  DBR
 201        00135 R 620541 R            JMP*    LPARGP
```

Figure 4-1 (Cont.)
XVM LP11 DOS Handler

```
PAGE   9     LPU.   122     .WRITE FUNCTION

 202                                .TITLE .WRITE FUNCTION
 203                           /
 204                           /.WRITE
 205                           /
 206        00136 R 100524 R    LPWRIT  JMS LPIOCK              /PRINTER BUSY?
 207        00137 R 220540 R            LAC* LPCALP             /GET THE DATA MODE FROM THE USER CAL.
 208        00140 R 500650 R            AND     (1000   /MAKE SKP-NOP IN MIX
 209        00141 R 240651 R            XOR     (SKP
 210        00142 R 040565 R            DAC     MIX
 211        00143 R 220541 R            LAC*    LPARGP          /USER BUFFER ADDRESS.
 212        00144 R 440541 R            IDX LPARGP      /NOW POINTS TO WORD COUNT
 213        00145 R 040561 R            DAC     TCHAR   /SAVE POINTER TO BUFFER HEADER
 214        00146 R 723002 A            AAC     2       /MAKE X12 POINT TO DATA NOT HEADER
 215        00147 R 040570 R            DAC     X12     /GETTER POINTER
 216        00150 R 500652 R            AND (700000)            /(RCHM-120) EXTRANC EXTEND ADDRESSING BITS FROM BUFFER ADDRESS.
 217        00151 R 740200 A            SZA                     /(RCHM-120) ARE ANY SET?
 218        00152 R 600026 R            JMP IOPS67              /(RCHM-120) YES, ISSUE IOPS67 ERROR MESSAGE.
 219                           /
 220                           /  SET UP LIMIT OF INPUT BUFFER SIZE TO PREVENT DATA OVERRUN
 221                           /  FOR BOTH IOPS ASCII AND IMAGE ASCII
 222                           /
 223        00153 R 777000 A            LAW     17000   /GET PAIR COUNT FROM LEFT HALF
 224        00154 R 520561 R            AND*    TCHAR
 225        00155 R 742030 A            SWHA            /BRING TO RIGHT. PAIR COUNT INCLUDES HEADER
 226                           /                         /PAIR COUNT. WE ISZ BEFORE LOOP SO THAT'S
 227                           /                         /OK. IOPS NOW SET XCPT CMA!IAC
 228        00156 R 400565 R            XCT     MIX     /SKIP IF ASCII, NOT IF IMAGE
 229        00157 R 751001 A            SKP!CLA!CMA     /IMAGE -1 IN AC, SKIP. -1 BECAUSE WE ISZ FIRST
 230        00160 R 741031 A            SKP!CMA!IAC     /IOPS COMPLEMENTED TO CORRECT VALUE
 231        00161 R 360541 R            TAD*    LPARGP  /IMAGE ADD IN TOTAL WORD COUNT, INCL
 232                           /                         /TWO WORDS FOR HEADER, WE ISZ BEFORE LOOP.
 233        00162 R 040554 R            DAC     TEMP1   /INTO CONTROLLER, BOTH MODES
 234        00163 R 440541 R            ISZ     LPARGP  /MOVE ARG POINTER TO EXIT
 235        00164 R 200552 R            LAC     LPBUFD  /POINTER TO DATA PORTION OF BUFFER
 236        00165 R 040571 R            DAC     PUTP    /LOAD TO CHARACTER PUTTER POINTER
 237        00166 R 200347 R            LAC     GETIN   /INIT. CHAR GETTER
 238        00167 R 040344 R            DAC     GETSW
 239        00170 R 200443 R            LAC     PUTIN   /INIT CHAR PUTTER
 240        00171 R 040441 R            DAC     PUTSW
 241        00172 R 750000 A            CLA             /INIT OUTPUT BUFFER HEADER
 242        00173 R 400565 R            XCT     MIX     /TO 0 IF IOPS, 400 FOR IMAGE
 243        00174 R 200653 R            LAC     (400
 244        00175 R 060551 R            DAC*    LPBUF
 245        00176 R 750001 A            CLA!CMA         /COUNT OF 1 BLANK AS DEFUALT
 246                           /                         /FOR ZERO LENGTH IOPS LINE
 247        00177 R 060552 R            DAC*    LPBUFD  /IN FIRST DATA CHAR
 248                           /
 249                           /  MAIN LOOP TO TRANSFER CHAR'S TO HANDLER BUFFER
 250                           /
 251        00200 R 100332 R    MAIN    JMS     GETCH   /CHARACTER GETTER, LEAVES IT IN AC
 252        00201 R 741200 A            SNA             /SKIP UNLESS NULL CHAR
 253        00202 R 600200 R            JMP     MAIN    /NULL, IGNORE
```

Figure 4-1 (Cont.)
XVM LP11 DOS Handler

```
PAGE  10     LPU.   122     .WRITE FUNCTION

 254        00203 R 540654 R              SAD     (177    /IGNORE RUB-OUT
 255        00204 R 600200 R              JMP     MAIN    /MAIN
 256        00205 R 040561 R              DAC     TCHAR   /SAVE CHAR THROUGH TESTING
 257        00206 R 723740 A              AAC     -40     /SEPARATE 'TEXT' CHAR'S FROM CONTROL CHAR'S
 258        00207 R 741300 A              SNA!SPA         /SKIP ON REGULAR CHARS
 259        00210 R 600247 R              JMP     MSPEC   /GO DO SPECIALS
 260        00211 R 540655 R              SAD     (135    /ALT MODE
 261        00212 R 600314 R              JMP     UCLP03  /END OF LINE ON ALT MODE
 262                                /
 263                                /  SORRY ABOUT NE XT FIVE L NES.
 264                                /  THE LOGIC AT PUTCH TO DO FORMS CONTROL DOESN'T DO IMPLIED
 265                                /  LINE FEEDS, I.E. THOSE LINES HAVING NO LEADING CONTROL CHAR.
 266                                /  WE HAVE TO FAKE IT OUT BY   LACING A LINE FEED ON SUCH LINES!?!
 267                                /
 268        00213 R 200560 R              LAC     FIRST   /DO ONLY IF FIRST CHAR OF LINE IS REGULAR
 269        00214 R 740100 A              SMA             /SKIP IF FIRST CHAR
 270        00215 R 600220 R              JMP     .+3     /NOT FIRST CHAR, JUST CONTINUE
 271        00216 R 200656 R              LAC     (12     /HERE IS LINE FEED
 272        00217 R 100400 R              JMS     PUTCH   /AND CALL TO DO FORMS CONTROL
 273                                /
 274        00220 R 750030 A              CLA!IAC         /SET FLAG SAYING A REAL CHAR SINCE A FF
 275        00221 R 040562 R              DAC     COP
 276                                /
 277        00222 R 200563 R              LAC     BLANKC  /DO WE HAVE PENDING BLANKS/TABS TO SEND
 278                                /
 279                                /  NOTE BLANKC HAS MINUS COUNT OF CONSECTIVE BLANKS/TABS
 280                                /  SINCE PDP-11 CONTROLLER PRINTS ONLY BLANKS
 281                                /
 282        00223 R 744100 A              SMA!CLL         /SKIP IF ANY COLLECTED, TO PUT OUT BEFORE
 283                                /                       /REAL CHAR'S
 284        00224 R 600235 R              JMP     MAINC   /NONE, PENDING, GO PUT OUT THE CHAR
 285        00225 R 340657 R              TAD     (200    /TOUGH, IF MORE THAN 127 COLLECTED, MUST
 286                                /                       /PUT OUT TWO COUNTS
 287        00226 R 750100 A              SMA!CLA         /SKIP IF NEED TWO COUNTS
 288        00227 R 600233 R              JMP     MAIND   /NO, JUST PUT OUT COLLECTED COUNT
 289        00230 R 340657 R              TAD     (200    /TWO COUNTS, HERE IS FIRST
 290        00231 R 100400 R              JMS     PUTCH
 291        00232 R 200657 R              LAC     (200    /SET UP TO DO SECOND
 292        00233 R 340563 R  MAIND       TAD     BLANKC  /COMMON CODE, LAST COUNT FOR EITHER CASE
 293        00234 R 100400 R              JMS     PUTCH
 294        00235 R 140563 R  MAINC       DZM     BLANKC  /CLEAR OUT BLANK COUNTER
 295        00236 R 200561 R              LAC     TCHAR   /GET BACK ORIGINAL CHAR
 296        00237 R 100400 R              JMS     PUTCH   /TO OUTPUT BUFFER
 297        00240 R 440564 R  MAINK       ISZ     TABC    /INCREMENT TAB COUNTER
 298        00241 R 600244 R              JMP     MAINE   /NOT OVERFLOW, GO CHECK LINE COUNTER
 299        00242 R 777770 A              LAW     -10     /RESET TAB COUNTER
 300        00243 R 040564 R              DAC     TABC
 301        00244 R 440557 R  MAINE       ISZ     MAXC    /HAVE WE RUN OUT OF LINE
 302        00245 R 600200 R              JMP     MAIN    /NO
 303        00246 R 600314 R              JMP     UCLP03  /YES, GO FINISH UP, WITH END OF LINE
 304                                /
 305                                /  SPECIAL CHARACTERS
```

Figure 4-1 (Cont.)
XVM LP11 DOS Handler

```
PAGE  11     LPU.   122     .WRITE FUNCTION

306                          /
307        00247 R 750201 A  MSPEC   SZA!CLA!CMA      /SKIP IF IT IS A BLANK
308        00250 R 600254 R          JMP     MSPEC2   /NOPE, CHECK FOR OTHER THINGS
309        00251 R 340563 R          TAD     BLANKC   /ADD ONE TO BLANK COUNTER (IS MINUS COUNTER)
310        00252 R 040563 R          DAC     BLANKC
311        00253 R 600240 R          JMP     MAINK    /JOIN LINE AND TAB CONTROL SECTION
312        00254 R 200561 R  MSPEC2  LAC     TCHAR    /GET BACK ORIGINAL CHAR
313        00255 R 540660 R          SAD     (11      /IS IT A TAB
314        00256 R 600300 R          JMP     MTAB     /YUP, GO DO IT
315        00257 R 540661 R          SAD     (15      /CARRIAGE RETURN
316        00260 R 600314 R          JMP     UCLP03   /END OF LINE ON CARRIAGE RETURN
317        00261 R 540662 R          SAD     (20      /FORTRAN OTS OVERPRINT, DO AS CR
318        00262 R 600275 R          JMP     MCR
319        00263 R 540663 R          SAD     (14      /FORM FEED
320        00264 R 600270 R          JMP     MSPEC3   /JUST PUT IT OUT, FOR NOW
321        00265 R 540664 R          SAD     (21      /FORTRAN DOUBLE SPACE
322        00266 R 600272 R          JMP     MSPEC4   /DO AS TWO 12'S
323        00267 R 200656 R  MSPEC5  LAC     (12      /DEFAULT ON UNRECOGNIZED CONTROL CHAR. IS LINE FEED
324        00270 R 100400 R  MSPEC3  JMS     PUTCH    /PLACE IN BUFFER
325        00271 R 600200 R          JMP     MAIN     /GO DO NEXT
326        00272 R 200656 R  MSPEC4  LAC     (12      /FIRST OF TWO 12'S FOR THE 21
327        00273 R 100400 R          JMS     PUTCH
328        00274 R 600267 R          JMP     MSPEC5   /GO DO THE SECOND 112
329        00275 R 100455 R  MCR     JMS     RESETL   /NEW LINE, RESET VARIOUS GUYS
330        00276 R 200661 R          LAC     (15      /CARRIAGE RETURN
331        00277 R 600270 R          JMP     MSPEC3   /PUT CHAR AND LOOP
332        00300 R 200564 R  MTAB    LAC     TABC     /GET REMAINING COUNT FOR TAB
333        00301 R 340563 R          TAD     BLANKC   /AND ADD TO CUMULATIVE BLANK COUNT
334        00302 R 040563 R          DAC     BLANKC
335        00303 R 200564 R          LAC     TABC     /AND TO LINE CHECKER
336        00304 R 740031 A          CMA!IAC
337        00305 R 340557 R          TAD     MAXC
338        00306 R 040557 R          DAC     MAXC
339        00307 R 740100 A          SMA              /SKIP IF SOME LINE LEFT
340        00310 R 600314 R          JMP     UCLP03   /NONE LEFT, FINISH UP LINE
341        00311 R 777770 A          LAW     -10
342        00312 R 040564 R          DAC     TABC     /RESET TAB COUNTER
343        00313 R 600200 R          JMP     MAIN     /NEXT CHAR
344                          /
345        00314 R 200661 R  UCLP03  LAC     (15      /CARRIAGE RETURN
346        00315 R 400565 R          XCT     MIX      /PLACE IN BUFFER ONLY ON IMAGE!!!
347        00316 R 100400 R          JMS     PUTCH
348        00317 R 100455 R          JMS     RESETL
349        00320 R 440562 R  UCLP04  ISZ     COP      /A BLANK LINE IS STILL A REAL CHAR SINCE FF
350        00321 R 220551 R          LAC*    LPBUF    /ZERO CHAR COUNT??
351        00322 R 500665 R          AND     (377     /COUNT ONLY IN LOW 8 BITS
352        00323 R 740200 A          SZA              /SKIP IF ZERO COUNT
353        00324 R 600330 R          JMP     UCLP05   /NON-ZERO, JUST GO DO REGULAR
354        00325 R 400565 R          XCT     MIX      /IMAGE OR IOPS
355        00326 R 600134 R          JMP     LPNEXT   /IMAGE DO NOTHING
356        00327 R 460551 R          ISZ*    LPBUF    /IOPS MAKE FAKE 1 COUNT
357                          /                         /WE ARE DOING A BLANK LINE, AND 0
```

Figure 4-1 (Cont.)
XVM LP11 DOS Handler

```
PAGE  12     LPU.   122     .WRITE FUNCTION

 358                          /                          /COUNT MAKES SPOOLER VERY ILL
 359        00330 R 100531 R  UCLP05  JMS      LPSET     /SEND BUFFER TO PDP-11
 360        00331 R 600134 R          JMP      LPNEXT    /CAL EXIT
 361                          /
 362                          /       CHARACTER UNPACKING ROUTINE
 363                          /
 364                          /
 365                          /  THIS ROUTINE 'OWNS' THE MQ
 366                          /
 367                          /
 368                          /  CHARACTERS ARE OBTAINED FROM X12 POINTER. EACH CHAR
 369                          /  IS RETURNED RIGHT JUSTIFIED IN THE AC
 370                          /  TEMP1 HAS A MINUS COUNT OF THE WORDS TO BE OBTAINED
 371                          /  FROM THE INPUT POINTER X12
 372                          /
 373        00332 R 000000 A  GETCH   0
 374        00333 R 400565 R          XCT      MIX       /SKIP IF IT IS ASCII
 375        00334 R 741000 A          SKP
 376        00335 R 620344 R          JMP*     GETSW     /GETSW IS POINTER TO CORRECT ACTION ON ONTHE
 377                          /                          /CORRECT ONE OF THE FIVE POSSIBLE CHAR'S
 378                          /
 379                          /  NOW DO IMAGE MODE
 380                          /
 381        00336 R 440554 R          ISZ      TEMP1
 382        00337 R 741000 A          SKP                /SKP ON NOT THRU YET
 383        00340 R 600320 R          JMP      UCLP04    /DONE
 384        00341 R 220570 R          LAC*     X12
 385        00342 R 440570 R          ISZ      X12
 386        00343 R 600345 R          JMP      GETCM     /FINISH UP IN COMMON
 387                          /
 388        00344 R 000000 A  GETSW   0                   /POINTER TO CORRECT ACTION. INIT'ED FROM GETIN
 389                          /                          /FILLED BY JMS GETSW AFTER EACH CHAR
 390        00345 R 500654 R  GETCM   AND      (177      /COMMON FINISH UP, STRIP XTRA BITS
 391        00346 R 620332 R          JMP*     GETCH     /OUT
 392                          /
 393        00347 R 000351 R  GETIN   GET1               /INIT GETSW TO POINT TO FIRST CHAR ACTION
 394                          /
 395                          /  INDIVIDUAL CHARACTER ACTION
 396                          /
 397        00350 R 100344 R  GETQ    JMS      GETSW     /AFTER 5TH CHAR, POINT BACK TO FIRST
 398                          /
 399        00351 R 440554 R  GET1    ISZ      TEMP1     /OUT OF PAIRS?
 400        00352 R 600355 R          JMP      .+3       /CONTINUE IF OK
 401        00353 R 100455 R          JMS      RESETL    /END OF LINE RESET SOME STUFF
 402        00354 R 600320 R          JMP      UCLP04
 403        00355 R 220570 R          LAC*     X12       /FIRST WORD OF PAIR
 404        00356 R 440570 R          ISZ      X12
 405        00357 R 652000 A          LMQ                /INTO MQ FOR SHIFTING
 406        00360 R 640607 A          LLS      7
 407        00361 R 100344 R          JMS      GETSW     /DONE, LEAVE POINTER FOR SECOND CHAR
 408        00362 R 640607 A  GET2    LLS      7         /SECOND CHAR
 409        00363 R 100344 R          JMS      GETSW     /LEAVING POINTER FOR THIRD
```

Figure 4-1 (Cont.)
XVM LP11 DOS Handler

```
PAGE  13     LPU.   122     .WRITE FUNCTION

 410         00364 R 640604 A     GET3    LLS     4       /THE HALF-AND-HALF CHAR
 411         00365 R 040344 R             DAC     GETSW   /VERY TEMPORARY
 412         00366 R 220570 R             LAC*    X12     /CAN'T END IN MIDDLE OF PAIR
 413         00367 R 440570 R             ISZ     X12
 414         00370 R 652000 A             LMQ             /SECOND WORD TO SHIFTER
 415         00371 R 200344 R             LAC     GETSW   /BRING BACK FIRST
 416         00372 R 640603 A             LLS     3       /COMPLETE CHAR
 417         00373 R 100344 R             JMS     GETSW   /LEAVING POINTER TO FOURTH ACTION
 418         00374 R 640607 A     GET4    LLS     7
 419         00375 R 100344 R             JMS     GETSW   /LEAVING FOR 5
 420         00376 R 640607 A     GET5    LLS     7
 421         00377 R 600350 R             JMP     GETQ    /BACK TO TOP FOR POINTER TO 1
 422                              /
 423                              /
 424                              /
 425                              /  CHARACTER PUTTER FOR PDP-11
 426                              /
 427                              /  TWO CHAR'S PER WORD FORMAT. FIRST CHAR IS RIGHT JUSTIFIED, SECOND
 428                              /  IS PLACED IMMEDIATELY ABOVE FIRST, LEAVING TOP TWO BITS OF WORD
 429                              /  UNUSED. CHAR IS DELEVERD TO US IN AC. INIT PUTSW BY DAC'ING CONTENTS
 430                              /  OF PUTIN INTO IT. ROUTINE COUNTS THE OUTPUT CHARS IN LBF
 431                              /
 432                              /  THIS ROUTINE ALSO HANDLES FORM FEED PAGE CONTROL
 433                              /  THE PDP-11 ASSUMES LINES HAVE A LF IN BEGINNING AND CR AT END
 434                              /  SO THIS ROUTINE REMOVES ANY LEADING LF.
 435                              /
 436                              /
 437         00400 R 000000 A     PUTCH   0
 438         00401 R 500665 R             AND     (377    /STRIP TO EIGHT BITS
 439         00402 R 540656 R             SAD     (12     /SPECIAL CASE #1, LINE FEED
 440         00403 R 600412 R             JMP     PUTLF   /GO DO IT
 441         00404 R 540663 R             SAD     (14     /SPECIAL CASE #2, FORM FEED
 442         00405 R 600427 R             JMP     PUTFF   /GO DO IT
 443         00406 R 440560 R     PUTY    ISZ     FIRST   /BUMP FIRST TIME THRU SWTICH
 444         00407 R 740000 A             NOP             /IN CASE SKIPS, WE DON'T NEED IT HERE
 445         00410 R 460551 R     PUTZ    ISZ*    LPBUF   /COUNT AN OUTPUT CHAR
 446         00411 R 620441 R             JMP*    PUTSW   /DISPATCH TO FIRST OR SECOND CHAR ACTION
 447                              /
 448         00412 R 200562 R     PUTLF   LAC     COP     /HAS A REAL CHAR OCCURRED SINCE FF?
 449         00413 R 740200 A             SZA             /SKIP IF NO REAL CHAR
 450         00414 R 600424 R             JMP     PUTW    /GO DO REGULAR
 451         00415 R 220552 R             LAC*    LPBUFD  /IF WE ALREADY HAVE A FF
 452         00416 R 540663 R             SAD     (14     /IN BUFFER OUT, DON'T NEED A CR
 453         00417 R 620400 R             JMP*    PUTCH
 454         00420 R 200661 R             LAC     (15     /LEAD WITH CR, SO PDP-11 DOESN'T PUT ON AUTOMATIC LF
 455         00421 R 400565 R             XCT     MIX     /BUT DO NOTHING FOR IMAGE MODE
 456         00422 R 620400 R             JMP*    PUTCH
 457         00423 R 600406 R             JMP     PUTY    /GO REAJOIN
 458         00424 R 200656 R     PUTW    LAC     (12     /GET BACK LINE FEED
 459         00425 R 400545 R             XCT     FFSW    /ISZ OR NOP FOR COUNT OF FF PER PAGE
 460         00426 R 600434 R             JMP     PUTLFR  /NO FORM FEED NOW
 461         00427 R 200542 R     PUTFF   LAC     PAGSIZ  /FORM FEED, RESET PAGE COUNTER
```

Figure 4-1 (Cont.)
XVM LP11 DOS Handler

```
PAGE  14     LPU.   122     .WRITE FUNCTION

 462        00430 R 040543 R             DAC     PAGCNT
 463        00431 R 140562 R             DZM     COP      /FLAG SAYING FF OCCURRED.
 464        00432 R 200663 R             LAC     (14      /FORM FEED CODE
 465        00433 R 600410 R             JMP     PUTZ     /GO COUNT CHAR, AND PLACE IT
 466        00434 R 400565 R     PUTLFR  XCT     MIX      /SKIP ON IOPS ASCII
 467        00435 R 600406 R             JMP     PUTY     /IMAGE, ACTUALLY PLACE LF
 468        00436 R 440560 R             ISZ     FIRST    /ASCII, IS IT FIRST THRU?
 469        00437 R 600410 R             JMP     PUTZ     /NOT FIRST, DO LF
 470        00440 R 620400 R             JMP*    PUTCH    /FIRST TIME, JUST RETURN
 471        00441 R 000000 A     PUTSW   0                /INIT'ED AS PUT1. FILLED LATER BY JMS PUTSW
 472        00442 R 620400 R             JMP*    PUTCH    /DONE, RETURN
 473                             /
 474        00443 R 000445 R     PUTIN   PUT1             /START AT FIRST CHAR
 475                             /
 476        00444 R 100441 R     PUTQ    JMS     PUTSW    /LEAVE POINTER FOR FIRST AFTER SECOND
 477        00445 R 060571 R     PUT1    DAC*    PUTP     /FIRST CHARACTER ACTION, PLACE RIGHT JUSTIFIED
 478        00446 R 100441 R             JMS     PUTSW    /LEAVING POINTER FOR SECOND
 479                             /
 480        00447 R 746030 A     PUT2    CLL!SWHA         /PUT CHAR IN RIGHT PLACE
 481        00450 R 740020 A             RAR
 482        00451 R 260571 R             XOR*    PUTP     /PUT HALVES TOGETHER
 483        00452 R 060571 R             DAC*    PUTP     /BOTH IN BUFFER
 484        00453 R 440571 R             ISZ     PUTP     /MOVE POINTER
 485        00454 R 600444 R             JMP     PUTQ     /GO TELL PUTSW  THAT PUT1 IS NEXT
 486                             /
 487                             /  OUTINE TO RESET LINE AND TAB COUNTRS
 488                             /
 489        00455 R 000000 A     RESETL  0
 490        00456 R 777777 A             LAW     -1       /SET FIRST CHAR OF LINE REMEMBERER
 491        00457 R 040560 R             DAC     FIRST
 492        00460 R 777770 A             LAW     -10      /SET TAB COUNTR
 493        00461 R 040564 R             DAC     TABC
 494        00462 R 200556 R             LAC     LINLIM   /SET UP MAX PER LINE COUNTER
 495        00463 R 040557 R             DAC     MAXC
 496        00464 R 140563 R             DZM     BLANKC   /RESET SPACE AND TAB COUNTER
 497        00465 R 620455 R             JMP*    RESETL
 498                             /


PAGE  15     LPU.   122     .CLOSE FUNCTION

 499                                     .TITLE .CLOSE FUNCTION
 500                             /
 501                             /
 502                             /.CLOSE
 503                             /
 504        00466 R 100524 R     LPCLOS  JMS     LP1OCK           /CHECK I/O UNDERWAY.
 505        00467 R 140562 R             DZM     COP      /SAY A FF OCCURRED
 506        00470 R 440502 R             ISZ     LPCLSW           /777777 IN AC IF HAVEN'T BEEN THRU CLOSE CODE.
 507        00471 R 600503 R             JMP     LPCLDN           /DONE.
 508        00472 R 750030 A             CLA!IAC          /SPOOLER REQUIRES FF,CR AS CLOSE
 509        00473 R 060551 R             DAC*    LPBUF    /JUST GIVE FF TO DRIVER, HOWEVER
 510        00474 R 200666 R             LAC     (6414    /THIS IS FF,CR IN PDP-11
 511        00475 R 060552 R             DAC*    LPBUFD   /FIRST DATA WORD POINTER
 512                             /                        /THIS MEANS ALWAYS A FF ON CLOSE!!!
 513        00476 R 100531 R             JMS     LPSET    /SEND BUFFER TO PDP-11
 514        00477 R 100455 R             JMS     RESETL   /RESET THE WORLD
 515        00500 R 703344 A     LPCALA  DBR
 516        00501 R 620540 R             JMP*    LPCALP           /HANG ON CAL.
 517        00502 R 777777 A     LPCLSW  /77777           /-1 = .CLOSE NOT DONE.
 518        00503 R 777777 A     LPCLDN  LAW -1
 519        00504 R 040502 R             DAC     LPCLSW           /INITIALIZE .CLOSE INDICATOR
 520        00505 R 600134 R             JMP     LPNEXT           /EXIT.
```

Figure 4-1 (Cont.)
XVM LP11 DOS Handler

```
PAGE  16     LPU.   122     .WAIT FUNCTION

 521                                .TITLE .WAIT FUNCTION
 522                          /
 523                          /.WAIT OR .WAITR
 524                          /
 525        00506 R 220540 R  LPWAIT  LAC*    LPCALP
 526        00507 R 500650 R          AND     (1000
 527        00510 R 741200 A          SNA                       /BIT 8 = 1 FOR .WAITR
 528        00511 R 600522 R          JMP     LPWAT1            /.WAIT - GO HANG ON CAL.
 529        00512 R 200652 R          LAC     (700000           /LINK, ETC.
 530        00513 R 500540 R          AND     LPCALP
 531        00514 R 040540 R          DAC     LPCALP
 532        00515 R 220541 R          LAC*    LPARGP            /15-BIT BUSY ADDRESS.
 533        00516 R 500667 R          AND     (77777
 534        00517 R 240540 R          XOR     LPCALP
 535        00520 R 040540 R          DAC     LPCALP
 536        00521 R 440541 R          IDX     LPARGP
 537        00522 R 100524 R  LPWAT1  JMS     LPIOCK            /CHECK I/O UNDERWAY.
 538        00523 R 600134 R          JMP     LPNEXT            /OK - RETURN.
 539                          /
 540                          /CHECK FOR I/O UNDERWAY
 541                          /
 542                          /LPUND 0 WHEN FREE, NONO WHEN BUSY
 543                          /
 544        00524 R 000000 A  LPIOCK  0
 545        00525 R 200544 R          LAC     LPUND             /0 = NO ACTIVITY.
 546        00526 R 741200 A          SNA
 547        00527 R 620524 R          JMP*    LPIOCK            /NO I/O UNDERWAY.
 548        00530 R 600500 R          JMP     LPCALX            /HANG ON CAL TIL NOT BUSY.
 549                          /
 550                          / SETUP AND OUTPUT TO PRINTER.
 551                          /
 552        00531 R 000000 A  LPSET   0
 553        00532 R 200550 R          LAC     LPTCB   /SEND TCB POINTER TO PDP-11
 554        00533 R 706001 A          SIOA                      /MAKE SURE ITS ABLE TO GET IT
 555        00534 R 600533 R          JMP     .-1               /NOTE THAT THIS IS PROTECTED SINCE
 556                                                            /    THE LIOR WILL BE ISSUED DIRECTLY
 557                                                            /    AFTER THE SIOA (FREE INSTRUCTION).
 558        00535 R 706006 A          LIOR
 559        00536 R 040544 R          DAC LPUND                 /SET I/O BUSY FLAG.
 560        00537 R 620531 R          JMP* LPSET
```

Figure 4-1 (Cont.)
XVM LP11 DOS Handler

```
PAGE  17     LPU.   122      INITIALIZATION CODE AND TEMPORARIES

 561                                .TITLE INITIALIZATION CODE AND TEMPORARIES
 562                          /
 563        00540 R 000000 A     LPCALP  0               /POINTER TO CAL ADDR
 564        00541 R 000000 A     LPARGP  0               /POINTER ARGUMENTS OF CAL
 565        00542 R 777706 A     PAGSIZ  -FORMS          /ASSEMBLED LINES PER PAGE
 566        00543 R 777706 A     PAGCNT  -FORMS          /COUNT THE LINES HERE
 567        00544 R 777772 A     LPUND   -6              /0=FREE,+=BUSY,-=ERROR
 568                          /                           /COUNTS UP TO INITAL 0 BELOW
 569                          /
 570                                .IFUND NOFF
 571        00545 R 440543 R     FFSW    ISZ     PAGCNT  /ACTION FOR FORMS CONTROL, MEMORY
 572        00546 R 440543 R     FFFF    ISZ     PAGCNT  /FFSW LOADED INTO HERE
 573                                .ENDC
 578        00547 R 200636 R     INIT    LAC     (NOP    /WRITE OVER JUMP TO HERE
 579        00550 R 040003 R     LPTCB   DAC     NEW     /PREVENT RE-ENTRY
 580        00551 R 220645 R     LPBUF   LAC*    (.SCOM+4  /GT PRINTER LINE WIDTH
 581        00552 R 742020 A     LPBUFD  RTR
 582        00553 R 740020 A     LPEV    RAR             /MOVE TO '6' POSITION
 583        00554 R 500670 R     TEMP1   AND     (6      /STRIP GARBAGE, LITERAL 6
 584        00555 R 741200 A     BUFSIZ  SNA
 585        00556 R 340670 R     LINLIM  TAD     (6      /TREAT 0 (UNDEFINED) AS 132 COLUMN!??!
 586        00557 R 340624 R     MAXC    TAD     LBFTP   /POINTER TO CONSTANTS
 587        00560 R 040624 R     FIRST   DAC     LBFTP
 588        00561 R 220624 R     ICHAR   LAC*    LBFTP   /LINE WIDTH
 589        00562 R 040556 R     COP     DAC     LINLIM
 590        00563 R 440624 R     BLANKC  ISZ     LBFTP
 591        00564 R 220624 R     TABC    LAC*    LBFTP   /BUFFER SIZE
 592        00565 R 040555 R     MIX     DAC     BUFSIZ
 593                          /
 594                          /  NOW SET UP POINTERS TO BUFFER AND TCB LOC'S
 595                          /
 596        00566 R 220657 R     LPAC    LAC*    (.SCOM+100  /POINTER TO TABLE OF POINTERS
 597        00567 R 740030 A     LPOUT   IAC             /OUR POINTER IN TABLE +1
 598        00570 R 040554 R     X12     DAC     TEMP1
 599        00571 R 220554 R     PUTP    LAC*    TEMP1   /POINTER TO TCB
 600        00572 R 040550 R             DAC     LPTCB
 601        00573 R 040554 R             DAC     TEMP1   /POINTER TO FILL LOCATIONS
 602        00574 R 723002 A             AAC     2       /MAKE POINTER TO EVENT VARIABLE
 603        00575 R 040553 R             DAC     LPEV
 604        00576 R 723002 A             AAC     2       /MAKE POINTER TO TCB POINTER
 605        00577 R 040564 R             DAC     TABC    /TO BUFFER ADDR
 606        00600 R 723005 A             AAC     5       /MAKE POINTER TO FIRST DATA WORD
 607        00601 R 040552 R             DAC     LPBUFD
 608                          /
 609                          /  MAKE TCB
 610                          /
 611        00602 R 200671 R             LAC     (APISLT*400+APILVL
 612        00603 R 060554 R             DAC*    TEMP1
 613        00604 R 440554 R             ISZ     TEMP1
 614        00605 R 200672 R             LAC     (DEVCOD /PIREX CODE FOR LP DRIEVER
 615        00606 R 060554 R             DAC*    TEMP1
 616        00607 R 440554 R             ISZ     TEMP1   /ZERO THRU FIRST BUFFER LOC
```

Figure 4-1 (Cont.)
XVM LP11 DOS Handler

```
PAGE  18     LPU.   122     INITIALIZATION CODE AND TEMPORARIES

 617        00610 R 160554 R            DZM*    TEMP1
 618        00611 R 440544 R            ISZ     LPUND
 619        00612 R 600607 R            JMP     .+3
 620        00613 R 200554 R            LAC     TEMP1   /THIS POINTS TO BUFFER
 621        00614 R 060564 R            DAC*    TABC    /TO LOCATION IN TCB THAT NEEDS
 622        00615 R 040551 R            DAC     LPBUF   /AND A POINTER FOR US
 623        00616 R 100455 R            JMS     RESETL  /RESET LINE AND TAB COUNTRS
 624        00617 R 000056 A            CAL     APISLT  /ISSUE SETUP CAL TO ESTABLISH INTERRUPTS
 625        00620 R 000016 A            16
 626        00621 R 706141 A            LSSF
 627        00622 R 000032 R            LPINT
 628        00623 R 600003 R            JMP     NEW     /WHEW, DONE
 629                                /
 630                                    .DEC
 631        00624 R 000623 R    LBFTP   .-1             /POINTER TO SIZE TABLE
 632        00625 R 777660 A            -80
 633        00626 R 000044 A            36
 634        00627 R 777610 A            -120
 635        00630 R 000064 A            52
 636        00631 R 777574 A            -132
 637        00632 R 000070 A            56
 638               000000 A            .END
            00633 R 017777 A *L
            00634 R 600011 R *L
            00635 R 600042 R *L
            00636 R 740000 A *L
            00637 R 000000 A *L
            00640 R 700042 A *L
            00641 R 177777 A *L
            00642 R 177001 A *L
            00643 R 600000 A *L
            00644 R 000137 A *L
            00645 R 000104 A *L
            00646 R 000002 A *L
            00647 R 004000 A *L
            00650 R 001000 A *L
            00651 R 741000 A *L
            00652 R 700000 A *L
            00653 R 000400 A *L
            00654 R 000177 A *L
            00655 R 000135 A *L
            00656 R 000012 A *L
            00657 R 000200 A *L
            00660 R 000011 A *L
            00661 R 000015 A *L
            00662 R 000020 A *L
            00663 R 000014 A *L
            00664 R 000021 A *L
            00665 R 000377 A *L
            00666 R 006414 A *L
            00667 R 077777 A *L
            00670 R 000006 A *L


PAGE  19     LPU.   122     INITIALIZATION CODE AND TEMPORARIES

            00671 R 027002 A *L
            00672 R 000004 A *L
               SIZE=00673     NO ERROR LINES
```

Figure 4-1 (Cont.)
XVM LP11 DOS Handler

```
PAGE   20   LPU.   CROSS REFERENCE

APILVL 000002    48*    51     55    611
APISLT 000056    49*   611    624
BLANKC  00563   277    292    294    309    310    333    334    496
                590*
BUFSIZ  00555   174    584*   592
CAPI   706144    55*   128
COP     00562   189    275    349    448    463    505    589*
DEVCOD 000004    72*    75*   614
ERLOOP  00074   158*   161
EROUT   00076   160*
ERRNUM  00102   157    159    164*
EXERRS 000137    63*   160
FFFF    00546   181    182    572*   576*
FFSW    00545   185    459    571*   575*
FIRST   00560   268    443    468    491    587*
FORMS  000072    66*    69*   565    566
GETCH   00332   251    373*   391
GETCM   00345   386    390*
GETIN   00347   237    393*
GETQ    00350   397*   421
GETSW   00344   238    376    388*   397    407    409    411    415
                417    419
GET1    00351   393    399*
GET2    00362   408*
GET3    00364   410*
GET4    00374   418*
GET5    00376   420*
IDX    440000    61*    81     90     98    173    176    212    536
INIT    00547    85    578*
IOPS12  00030   109*   172
IOPS67  00026   107*   218
LBFTP   00624   586    587    588    590    591    631*
LINLIM  00556   494    585*   589
LIOR   706006    53*   151    558
LPAC    00566   115    123    134    596*
LPARGP  00541    80     81     89     90     98    173    175    176
                201    211    212    231    234    532    536    564*
LPA.    00000    77     79*
LPBUF   00551   191    244    350    356    445    509    580*   622
LPBUFD  00552   193    235    247    451    511    581*   607
LPCALP  00540    79    179    207    516    525    530    531    534
                535    563*
LPCALX  00500   515*   548
LPCLDN  00503   507    518*
LPCLOS  00466   100    504*
LPCLSW  00502   506    517*   519
LPERR6  00024    97    102    105*
LPEV    00553   129    582*   603
LPICM   00046   121    127*
LPIERR  00061   132    140*
LPIN    00103    95    169*
LPINT   00032   114*   116    119    627
LPIOCK  00524   188    206    504    537    544*   547
```

Figure 4-1 (Cont.)
XVM LP11 DOS Handler

```
PAGE   21   LPU.   CROSS REFERENCE

LPIRT   00054   133*
LPIRT1  00055   134*   152
LPISW   00056   127    135*
LPNEXT  00134    99    101    200*   355    360    520    538
LPOUT   00567   117    125    137    597*
LPPIC   00042   114    118    123*
LPSET   00531   195    359    513    552*   560
LPTCB   00550   148    553    579*   600
LPUND   00544   133    545    559    567*   618
LPWAIT  00506   104    525*
LPWAT1  00522   528    537*
LPWRIT  00136   103    206*
LSSF   706141    51*   626
LTABL   00012    92     95*
MAIN    00200   251*   253    255    302    325    343
MAINC   00235   284    294*
MAIND   00233   288    292*
MAINE   00244   298    301*
MAINK   00240   297*   311
MAXC    00557   301    337    338    495    586*
MCR     00275   318    329*
MIX     00565   210    228    242    346    354    374    455    466
                592*
MSPEC   00247   259    307*
MSPEC2  00254   308    312*
MSPEC3  00270   320    324*   331
MSPEC4  00272   322    326*
MSPEC5  00267   323*   328
MTAB    00300   314    332*
NEW     00003    85*   579    628
PAGCNT  00543   178    462    566*   571    572
PAGSIZ  00542   177    461    565*
PUTCH   00400   272    290    293    296    324    327    347    437*
                453    456    470    472
PUTFF   00427   442    461*
PUTIN   00443   239    474*
PUTLF   00412   440    448*
PUTLFR  00434   460    466*
PUTP    00571   236    477    482    483    484    599*
PUTQ    00444   476*   485
PUTSW   00441   240    446    471*   476    478
PUTW    00424   450    458*
PUTY    00406   443*   457    467
PUTZ    00410   445*   465    469
PUT1    00445   474    477*
PUT2    00447   480*
RESETL  00455   187    329    348    401    489*   497    514    623
RETRY   00066   142    148*
SC.MOD 000104    58*   169
SC.UC1 000002    59*   170
SET    440000    62*
SETERR  00073   106    108    110    144    157*
SIOA   706001    52*   149    554
```

Figure 4-1 (Cont.)
XVM LP11 DOS Handler

```
PAGE   22   LPU.   CROSS REFERENCE

TABC     00564   297     300     332     335     342     493     591*    605
                 621
TCHAR    00561   213     224     256     295     312     588*
TEMP1    00554   233     381     399     583*    598     599     601     612
                 613     615     616     617     620
UCLP03   00314   261     303     316     340     345*
UCLP04   00320   349*    383     402
UCLP05   00330   353     359*
X12      00570   215     384     385     403     404     412     413     598*
%DOS     000001
%RELES   000001
%VERSN   000001
%XVM     000001
.CLEAR   MACRO
.CLOSE   MACRO   499
.DLETE   MACRO
.ENTER   MACRO
.EXIT    MACRO
.FSTAT   MACRO
.GET     MACRO
.GTBUF   MACRO
.GVBUF   MACRO
.INIT    MACRO   165
.MED     000003   60*
.MTAPE   MACRO
.OVRLA   MACRO
.PUT     MACRO
.RAND    MACRO
.READ    MACRO
.RENAM   MACRO
.RTRAN   MACRO
.SCOM    000100   57*    58      63      580     596
.SEEK    MACRO
.SYSID   MACRO     1
.TIMER   MACRO
.TRAN    MACRO
.USER    MACRO
.WAIT    MACRO   521
.WAITR   MACRO
.WRITE   MACRO   202
```

Figure 4-1 (Cont.)
XVM LP11 DOS Handler

APILVL — The API level at which PIREX should interrupt the XVM; 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 624-628).

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 an XVM/DOS handler must have a once only section of code that:

1. Sets up a pointer to one of the reserved TCB areas in the XVM/DOS monitor. This is done by locating a pointer to the TCB area in the table pointed to by .SCOM+100 (Figure 4-1, lines 596-600).
2. Computes pointers to the various locations within this TCB area, such as the event variable (Figure 4-1, lines 601-607).
3. Constructs the constant fields within the TCB such as the API RETURN and device code (Figure 4-1, lines 611-619).
4. Sets up a pointer to the data area in the TCB, which will be used as a buffer (Figure 4-1, lines 620-622).

4.6.1.2 .INIT Function - The .INIT function of any XVM UNICHANNEL handler should check to see if the UNICHANNEL is enabled by testing bit 16 of .SCOM+4. If bit 16 is set, the UNICHANNEL is enabled, or else if bit 16 is not set, IOPS 12 (device error) should be issued. (Figure 4-1, lines 169-172.)

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

4.6.1.3 Request Transmission - When issuing requests to a task from a XVM program, the requesting program (e.g., a XVM 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
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 XVM 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 XVM 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 551-560).

4.6.1.4 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 XVM/DOS 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 XVM are serviced by the XVM in a fashion identical to regular XVM 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 XVM 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.

After clearing this flag, the event variable should be tested to detect an error condition (negative event variable). See Figure 4-1, lines 129-132.

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 (EV = 177001) a retry of the request should be attempted (see Figure 4-1, lines 148-152).

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 155-164).

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

4.6.1.5 .READ and .WRITE Requests - Actual input and output is accomplished by using typical XVM/DOS 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 359)

4.6.1.6 .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 XVM/DOS 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 499-513).

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 XVM/DOS 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 XVM. (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
IOT
.BYTE 2,0
.ENDM
```

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

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

A TCBP is a TCB pointer. If the requesting task desires a software interrupt it should place the interrupt return address in the proper entry of the "SEND 11" Table (see Section 3.3.8).

4.6.3 UNICHANNEL Device Handlers for XVM/RSX

The following description of how to write a UNICHANNEL device handler for XVM/RSX does not discuss those topics pertaining to all XVM/RSX I/O handlers, see the chapter on Advanced Task Construction in the XVM/ RSX System Manual.

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 67-80). 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 XVM; 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 263-321). 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 288-305). Since it is not obvious to XVM/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.

```
PAGE   2     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

  28                          /
  29                          /EDIT #021      4/22/75 SCR UC15 EOF CARD FIX
  30                          /EDIT #020      2/2/74  SCR  CLEANUP
  31                          /EDIT #019      SCR     CR15 ERROR HANDLING; RRN SWITCH!
  32                          /EDIT #018      SCR     FIX CDON HANDLING CR15 VERSION
  33                          /EDIT #017      SCR     CLEANUP, !BOTH! DEVICES
  34                          /EDIT #016      SCR     MORE UC15 CODE
  35                          /EDIT #015      SCR  START TO PUT IN UC15 CODE
  36                          /EDIT #013      1-18-72
  37                          /EDIT #14       6-26-73
  38                          /COPYRIGHT 1973, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
  39                          /C.W. KEMP ---- W.A. DESIMONE, ---- G. M. COLE
  40                          /
  41                          /CR15 CARD READER CONTROL HANDLER TASK.  THIS CONTROL WILL
  42                          /SUPPORT SORBAN AND DOCUMATION READERS.
  43                          / CR15 CODE IS OBTAINED  WITH NO ASSEMBLY PPARAMETERS
  44                          /
  45                          / TO OBTAIN UC15 CODE DEFINE UC15=0.
  46                          /   ADDITIONAL UC15 PARAMETERS:
  47                          /  DEFINE NOSPL=0 TO DISABLE SPOOLING FOR CARD READER. FOR INSTANCE
  48                          /  IF SPOOLER PACKAGE DOESN'T HAVE CARD READER ASSEMBLED IN FOR SPACE REASONS.
  49                          / AN EQUATE FOR APILVL IS NECESSARY TO SET UP
  50                          / IOT'S FOR CORRECT PRIORITY LEVEL TO CLEAR PIREX REQUEST.
  51                          / PRESENTLY LEVEL 1 IS THE CARD READER ASSIGNMENT.
  52                          /
  53                          /     W   A   R   N   I   N   G   !   !
  54                          /
  55                          /  IN ORDER FOR THE UC15 HANDLER TO FUNCTION PROPERLY, THE
  56                          /  PDP11 MUST BE ABLE TO ACCESS OUR INTERNAL BUFFER
  57                          /  AND TCB'S, THIS MEANS THAT THEIR ADDRESS MUST BE LESS THAN
  58                          /   28K TO THE PDP11. THUS, IF THE PDP-11 LOCAL MEMORY IS 8K,
  59                          /  THIS HANDLER MUST RESIDE BELOW 20K IN PDP15 CORE!! THIS
  60                          /  IS EQUIVALENT TO 50000 OCTAL. SIMILARLY , IF THE LOCAL
  61                          /  PDP-11 MEMORY IS 12K, THE HANDLER MUST RESIDE BELOW
  62                          /  40000 OCTAL.
  63                          /
  64                                  .IFDEF  UC15
  65                          /
  66                          /
  67              000055 A    APISLT=55
  68              000001 A    APILVL=1
  69              706121 A    CRSI=APILVL*20+706101
  70              706001 A    SIOA=706001
  71              706006 A    LIOR=706006
  72              706124 A    CAPI=APILVL*20+706104
  73                          /
  74                                  .IFUND NOSPL
  75              000005 A    DEVCOD=5
  76                                  .ENDC
  77                                  .IFDEF NOSPL
  78                          DEVCOD=205
  79                                  .ENDC
```

Figure 4-2
XVM CR11 XVM/RSX Handler

```
PAGE   3     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

                                    .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)
                               /               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.


PAGE   4     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

                               /       -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
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

```
PAGE   5     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

 148                          /
 149                          /   ***** CONSTANTS *****
 150                          /
 151              000012 A    X12=12          /AUTO-INDEXREG. 12
 152              000013 A    X13=13          /AUTO-INDEXREG. 13
 153              000101 A    R1=101          /RE-ENTRANT REG. 1
 154              000102 A    R2=102          /RE-ENTRANT REG. 2
 155              000103 A    R3=103          /RE-ENTRANT REG. 3
 156              000104 A    R4=104          /RE-ENTRANT REG. 4
 157              000107 A    NADD=107        /NODE ADDITION ROUTINE ENTRY POINT
 158              000123 A    SNAM=123        /NAME SCAN ROUTINE ENTRY POINT
 159              000240 A    POOL=240        /LISTHEAD FOR POOL OF EMPTY NODES
 160              000252 A    PDVL=252        /LISTHEAD FOR PHYSICAL DEVICE LIST
 161              000325 A    ALAD=325        /ATTACH LUN & DEVICE ENTRY POINT
 162              000332 A    DLAD=332        /DETACH LUN & DEVICE ENTRY POINT
 163              000337 A    DQRQ=337        /DE-QUEUE REQUEST ENTRY POINT
 164              000342 A    VAJX=342        /VERIFY AND ADJUST I/O PARAMS.
 165              000345 A    IOCD=345        /DECREMENT TRANSFERS PENDING COUNT.
 166              000361 A    DMTQ=361        /DE-QUEUE I/O REQUEST (FOR ABORTING).
 167              000010 A    D.TG=10         /POSITION OF TRIGER EVENT VARIABLE IN PDVL NODE
 168                          /
 169                                 .IFUND  UC15
 170                          /
 171                          CWC=22          /WC DCH ADDRESS.
 172                          CCA=23          /CA DCH ADDRESS.
 173                          /
 174                          /PSUEDO-INSTR. FOR WF.SW SUBR.
 175                          /
 176                          WFOFF=SNA       /WAITFOR CR15 NOT READY.
 177                          WFON=SZA        /WAITFOR CR15 READY.
 178                          /
 179                          /
 180                          /CONDITIONS FOR LOAD READER CONDITION IOT (CRLC).
 181                          /
 182                          CC1=20          /CLEAR STATUS,DISABLE INTERRUPT AND DATA CHANNEL.
 183                          CC2=27          /CLEAR STATUS,START READ,ENABLE INTERRUPT AND DATA CHANNEL.
 184                          CC3=26          /CLEAR STATUS,ENABLE INTERRUPT,ENABLE DATA CHANNEL.
 185                          CC4=04          /ENABLE INTERRS. DISABLES DCH
 186                          /
 187                          / *****  IOT INSTRUCTIONS *****
 188                          /
 189                          CRPC=706724             /CLEAR STATUS EXCEPT CARD DONE.(ALSO DISABLES INTERR.)
 190                          CRLC=706704             /LOAD READER CONDITIONS.
 191                          CRRS=706732             /READ STATUS INTO AC.
 192                          /
 193                                 .ENDC
 194                          /
 195              705522 A    .INH=705522             /INHIBIT INTERRUPTS.
 196              705521 A    .ENB=705521             /ENABLE INTERRUPTS.
 197                          /
 198                                 .EJECT
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

```
PAGE   6     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

 199                          /----CR15 STATUS AND AC BIT ASSIGNMENTS.
 200                          /
 201                          /STATUS REGISTER BIT ASSIGNMENTS:
 202                          /
 203                          /       BIT     TRANSLATION
 204                          /
 205                          /       17      COLUMN READY
 206                          /       16      END OF CARD
 207                          /       15      DATA CHANNEL OVERFLOW
 208                          /       14      DATA CHANNEL ENABLED
 209                          /       13      READY TO READ
 210                          /       12      ON LINE
 211                          /       11      END OF FILE
 212                          /       10      BUSY
 213                          /       09      TROUBLE (= IOR OF BITS 4 - 8)
 214                          /       08      DATA MISSED
 215                          /       07      HOOPER EMPTY/STACKER FULL
 216                          /       06      PICK ERROR
 217                          /       05      MOTION ERROR
 218                          /       04      PHOTO ERROR
 219                          /       03-00   UNUSED
 220                          /
 221                          /AC BIT ASSIGNMENTS FOR LOAD CONDITION FUNCTION (CRLC)
 222                          /
 223                          /       BIT     FUNCTION
 224                          /
 225                          /       17      START READ
 226                          /       16      DATA CHANNEL ENABLE
 227                          /       15      INTERRUPT ENABLE
 228                          /       14      OFFSET CARD
 229                          /       13      CLEAR STATUS REGISTER
 230                          /
 231                          /       STATUS REGISTER BITS CONNECTED TO FLAG AND INTERRUPT REQUEST:
 232                          /
 233                          /       17      DATA READY(ONLY IF DATA CHANNEL NOT ENABLED)
 234                          /       16      CARD DONE
 235                          /       15      DATA CHANNEL OVERFLOW
 236                          /       09      ERROR CONDITION
 237                          /
 238                          /MACRO DEFINITIONS:
 239                          /
 240                          /CP MACRO FOR CARD COLUMN TO ASCII TRANSLATION TABLE 026/029 CONDITIONALIZATION
 241                          /
 242                                  .IFDEF  DEC026
 243                                  .DEFIN  CP,C26,C29
 244                                  C26\7777+1
 245                                  .ENDM
 246                                  .ENDC
 247                                  .IFUND  DEC026
 248                                  .DEFIN  CP,C26,C29
 249                                  C29\7777+1
 250                                  .ENDM


PAGE   7     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

 251                                  .ENDC
 252                          /
 253                          /
 254                                  .EJECT
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

```
PAGE   8     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

 255                          /
 256                          /
 257                          / ***** HANDLER INITIALIZATION ***** (ONCE ONLY CODE)
 258                          /
 259                          /START                          /STORAGE FOR AC IN INTERR. SERVICE.
 260                          /IBUF                           /TOP OF INTERNAL BUFFER.
 261                          /
 262                          /
 263        00000 R 200646 R  START   LAC     (PDVL)          /SCAN PDVL FOR THIS DEVICE'S NODE
 264        00001 R 060647 R  IBUF    DAC*    (R1)
 265        00002 R 200650 R          LAC     (HNAM)
 266        00003 R 060651 R          DAC*    (R2)
 267        00004 R 120652 R          JMS*    (SNAM)          /R, R2, R6, XR, & AC ARE ALTERED
 268                                                          /NODE FOUND?
 269        00005 R 000653 R          CAL     (10)            /NO -- EXIT
 270        00006 R 040567 R          DAC     PDVNA           /YES -- PDVL NODE ADDRESS IN AC.
 271        00007 R 723010 A          AAC     D.TG            /SAVE NODE ADDRESS AND
 272        00010 R 040570 R          DAC     PDVTA           /TRIGGER EVENT VARIABLE ADDRESS
 273        00011 R 000577 R          CAL     CCPB            /CONNECT INTERRUPT LINE
 274        00012 R 200561 R          LAC     EV              /CONNECT OK?
 275        00013 R 741100 A          SPA
 276        00014 R 000653 R          CAL     (10)            /NO -- EXIT
 277        00015 R 200654 R          LAC     (TG)            /YES -- SET TEV ADDRESS
 278        00016 R 060570 R          DAC*    PDVTA
 279        00017 R 500655 R          AND     (70000)         /DETERMINE 'XR-ADJ'
 280        00020 R 740031 A          TCA
 281        00021 R 040563 R          DAC     XADJ
 282                          /
 283                                   .IFUND  UC15
 284                                   LAC     (CC1)           /CLEAR STATUS, DISABLE INTER, AND DCH.
 285                                   CRLC                    /LOAD FUNCTION.
 286                                   .ENDC
 287                                   .IFDEF  UC15
 288        00022 R 100625 R          JMS     CLEAR   /CLEAR OUT PIREX DEVICE, WAIT FOR COMPLETE
 289        00023 R 200613 R          LAC     EV11K   /FIND OUT IF OK
 290        00024 R 742010 A          RTL             /PDP11 SIGN BIT TO OURS
 291        00025 R 740100 A          SMA             /SKIP IF TROUBLE
 292        00026 R 600057 R          JMP     WFTGR   /NOT, GO WAIT FOR WORK
 293        00027 R 000034 R          CAL     MSINIT  /PRINT PIREX HAS NO CD MESSAGE
 294        00030 R 000032 R          CAL     WFMS    /WAIT FOR MESSAGE COMPLETION
 295        00031 R 000653 R          CAL     (10     /EXIT
 296                          /
 297        00032 R 000020 A  WFMS    20
 298        00033 R 000561 R          EV
 299        00034 R 002700 A  MSINIT  2700
 300        00035 R 000561 R          EV
 301        00036 R 000100 A          ERRLUN
 302        00037 R 000002 A          2
 303        00040 R 000041 R          INITMS
 304        00041 R 004002 A  INITMS  004002; 000000; .ASCII "*** NO CD IN PIREX"<15>
            00042 R 000000 A
            00043 R 251245 A
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

```
PAGE   9     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

          00044 R 220234 A
          00045 R 475010 A
          00046 R 342100 A
          00047 R 446344 A
          00050 R 050222 A
          00051 R 512133 A
          00052 R 006400 A
305                               .ENDC
306       00053 R 600057 R        JMP     WFTGR           /WAIT FOR TRIGGER
307                       /
308       00054 R 030400 A  HNAM    .SIXBT  'CD@@@@'        /HANDLER TASK NAME
          00055 R 000000 A
309                       /
310                               .IFUND  UC15
311                       /
312                               .BLOCK  121+START-.
313                       /
314                               .ENDC
315                       /
316                               .IFDEF  UC15
317                       /
318       00056 R 777775 A        .BLOCK  53+START-.
319                       /
320                               .ENDC
321                       / ***** END OF INITIALIZATION CODE *****
322                       /
323                       /******** THE ABOVE CODE IS OVERLAYED BY THE INTERNAL BUFFER ******
324                       /*************************************************************
325                       /
326                       /  UC15  INTERRUPT-CAL INTERACTION WILL BE DIFFERENT
327                       /    KEEP INITIAL PART SEPARATE
328                       /
329                               .IFUND  UC15
330                       /
331                       WFTGR   CAL     WFTCPB          /WAIT FOR IEV TO BE SET
332                       /
333                       / ***** THE TASK HAS BEEN TRIGGERED -- PICK A REQUEST FROM QUEUE
334                       /
335                               DZM     TG              /CLEAR TRIGGER
336                       PQ      LAC     PDVNA   /DEQUE A REQUEST
337                               DAC*    (R1)
338                               JMS*    (DQRQ)          /R1, R2, R4, R5, R6, XR & AC ARE ALTERED
339                                                       /WAS A REQUEST FOUND?
340                               JMP     WFTGR           /NO -- WAIT FOR TRIGGER
341                       /
342                               .ENDC
343                       /
344                               .IFDEF  UC15
345                       /  UC15 CODE
346                       /
347                       /    THE GENERAL IDEA IS THAT ALL WAITS ARE DONE THRU
348                       /  THE TRIGGER, WE FIGURE OUT HERE WHO SET THE TRIGGER. THIS
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

```
PAGE  10     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

 349                          /  ALLOWS US TO GET OUT OF HUNG DEVICE, SINCE WE WAIT HERE,
 350                          /  AND CAN SEE AN ABORT COMING THRU.
 351                          /
 352        00057 R 000575 R  WFTGR   CAL     WFTCPB  /WAIT FOR EVENT VARIABLE TO
 353        00060 R 200562 R  PQ      LAC     TG      /FIND OUT WHO IS CALLING
 354        00061 R 140562 R          DZM     TG      /RESET
 355        00062 R 742010 A          RTL             /ABORT BIT TO SIGN BIT
 356        00063 R 751130 A          SPA!CLA!IAC     /SKIP IF NOT ABORT, 1 IN AC.
 357        00064 R 600071 R          JMP     PQ1     /GO DO ABORT IN REGULAR WAY. THE HANGING
 358                          /                       /READ IS REMEMBERED IN RRN!
 359        00065 R 540554 R          SAD     CDON    /HAS A CARD BEEN DECLARED DONE BY INTERRUPT
 360        00066 R 600177 R          JMP     GOTCRD  /YEAH, GO TRANSLATE IT
 361        00067 R 540407 R          SAD     POST    /ARE WE WAITING FOR INTERRUPT
 362        00070 R 600057 R          JMP     WFTGR   /YES, AND IT HASN'T HAPPENED YET, SINCE
 363                          /                       /CDON NOT SET. WAIT ON THIS CAL REQ, TO BE
 364                          /                       /DONE AFTER THE INTERRUPT HAPPENS. IF ABORT
 365                          /                       /COMES IN THE MEANTIME, HE IS PUT AT HEAD
 366                          /                       /OF DEQUE OF WAITING REQ.'S SO WE DO HIM.
 367                          /
 368        00071 R 200567 R  PQ1     LAC     PDVNA   /TRY TO DEQUE AFTER OPERATION BEFORE WAITING
 369        00072 R 060647 R          DAC*    (R1     /IN CASE WAITING FOR INTERRUPT HAS HELD OFF
 370        00073 R 120656 R          JMS*    (DQRQ   /A REQUEST.
 371        00074 R 600057 R          JMP     WFTGR   /DIDN'T FIND ONE, GO WAIT
 372                          /
 373                                  .ENDC
 374                          /
 375        00075 R 040564 R          DAC     RN              /YES -- SAVE ADDRESS OF REQUEST NODE
 376        00076 R 340563 R          TAD     XADJ            /SETUP XR TO ACCESS NODE
 377        00077 R 721000 A          PAX
 378                          /
 379                          / ***** I/O REQUEST NODE FORMAT *****
 380                          /
 381                          /    (0) FORWARD LINK
 382                          /    (1) BACKWARD LINK
 383                          /    (2) STL PTR.
 384                          /    (3) PART. BLK PTR. (0 IF EXM TSK).
 385                          /    (4) TASK PRIORITY
 386                          /    (5) I/O FCN CODE IN BITS 9-17 AND LUN IN BITS 0-8
 387                          /    (6)  -- EVENT VARIABLE ADDRESS
 388                          /    (7) CTB PTR.
 389                          /    (10) EXTRA
 390                          /    (11) EXTRA
 391                          /
 392        00100 R 210005 A          LAC     5,X             /FETCH I/O FCN CODE
 393        00101 R 500657 R          AND     (777)
 394        00102 R 540660 R          SAD     (024)           /ATTACH REQUEST?
 395        00103 R 600120 R          JMP     ATTACH          /YES -- ATTACH TO TASK
 396        00104 R 540661 R          SAD     (025)           /NO -- DETACH REQUEST?
 397        00105 R 600127 R          JMP     DETACH          /YES -- DETACH FROM TASK
 398        00106 R 540662 R          SAD     (026)           /NO -- READ REQUST?
 399        00107 R 600140 R          JMP     READ            /YES -- READ CARD
 400        00110 R 540663 R          SAD     (036)           /NO -- HANDLER INFO.?


PAGE  11     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

 401        00111 R 600136 R          JMP     HINF            /YES -- RETURN INFO IN EV
 402        00112 R 540657 R          SAD     (777)           /NO -- EXIT (DEASSIGNED) REQUEST?
 403        00113 R 600464 R          JMP     DAEX            /YES -- DEATTACH & EXIT
 404        00114 R 540664 R          SAD     (017)           /ABORT REQUEST?
 405        00115 R 600502 R          JMP     CDABRT          /YES.
 406        00116 R 777772 A  EVM6    LAW     -6              /NO -- UNIMPLEMENTED FUNCTION -- SET
 407        00117 R 600424 R          JMP     SEV             /EVENT VARIABLE TO -6
 408                          /
 409                          / ATTACH TO A TASK
 410                          /
 411        00120 R 200567 R  ATTACH  LAC     PDVNA           /ATTACH LUN & DEVICE
 412        00121 R 060647 R          DAC*    (R1)
 413        00122 R 200564 R          LAC     RN
 414        00123 R 060651 R          DAC*    (R2)
 415        00124 R 120665 R          JMS*    (ALAD)          /R3, R4, R5, R6, X10, X11, XR & AC ARE ALTERED
 416                                                          /WAS LUN ATTACHED?
 417        00125 R 600424 R          JMP     SEV             /NO -- SET REQUESTOR'S EV TO -24
 418        00126 R 600423 R          JMP     REQCMP          /YES REQUEST COMPLETED
 419                          /
 420                          / DETACH FROM TASK
 421                          /
 422        00127 R 200567 R  DETACH  LAC     PDVNA           /DETACH LUN & DEVICE
 423        00130 R 060647 R          DAC*    (R1)
 424        00131 R 200564 R          LAC     RN
 425        00132 R 060651 R          DAC*    (R2)
 426        00133 R 120666 R          JMS*    (DLAD)          /R3, R4, R5, R6, X10, X11, XR & AC ARE ALTERED
 427                                                          /WAS LUN ATTACHED
 428        00134 R 600424 R          JMP     SEV             /NO -- SET REQUESTOR'S EV TO -24
 429        00135 R 600423 R          JMP     REQCMP          /YES -- REQUEST COMPLETED
 430                          /
 431                                  .EJECT
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

```
PAGE  12     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

 432                          /
 433                          / RETURN HANDLER INFORMATION
 434                          /
 435        00136 R 200667 R  HINF    LAC     (200007)
 436        00137 R 600424 R          JMP     SEV
 437                          /
 438                          /READ CARD
 439                          /
 440        00140 R 777776 A  READ    LAW     -2              /CHK. FOR IOPS ASCII DATA MODE.
 441        00141 R 350007 A          TAD     7,X
 442        00142 R 740200 A          SZA                     /IOPS ASCII?
 443        00143 R 600460 R          JMP     EVM7            /NO, RETURN -5 EV.
 444        00144 R 210002 A          LAC     2,X             /SAVE STL NODE PTR. FOR TASK IDENTIF.
 445        00145 R 040556 R          DAC     STLA            /SAVE VALID STL PTR.
 446        00146 R 210010 A          LAC     10,X            /YES.  VAL/ADJ. HEADER ADDRESS
 447        00147 R 060670 R          DAC*    (R3)            /HEADER ADDRESS.
 448        00150 R 210011 A          LAC     11,X            /WORD COUNT
 449        00151 R 060671 R          DAC*    (R4)
 450        00152 R 740031 A          TCA                     /SETUP COUNTER  SINCE
 451        00153 R 723002 A          AAC     +2              /OFFSET FOR CR APPENDAGE.
 452        00154 R 040566 R          DAC     CDWDCT          /VAJX ALTERS THE XR.
 453        00155 R 040574 R          DAC     TCWC            /SAVE IN CASE RETRY.
 454        00156 R 200564 R          LAC     RN              /REQ. NODE ADDRESS.
 455        00157 R 040571 R          DAC     RRN             /SAVE READ REQ. NODE ADDR. FOR ABORT.
 456        00160 R 060651 R          DAC*    (R2)
 457        00161 R 120672 R          JMS*    (VAJX)          /VAL/ADJ. (ALTERS XR,AC,R3,R5)
 458        00162 R 600462 R          JMP     EVM30           /RETS. HERE IF ERROR (I/O PARAM. OUT
 459                                                          /OF PARTITION.
 460        00163 R 220670 R          LAC*    (R3)            /ADJUSTED HEADER ADDRESS -1 TO X12 TEMP.
 461        00164 R 723777 A          AAC     -1
 462        00165 R 040572 R          DAC     TX12
 463        00166 R 723002 A          AAC     +2              /TEXT ADDRESS-1 TO X13 TEMP.
 464        00167 R 040573 R          DAC     TX13            /
 465        00170 R 140565 R          DZM     CDRVAL          /INIT. VALID. BITS.
 466                                  .IFUND  UC15
 467                                  LAC     CDON            /HAS CARD DONE FLAG COME UP SINCE
 468                                  SNA                     /LAST CARD READ?
 469                                  CAL     WFCRCD          /NO. WAITFOR CARD DONE.
 470                                  DZM     CDON            /YES. CLEAR CARD DONE FLAG.
 471                           RETRY  LAC     (IBUF-1)        /SET INTERN. BUFF ADDR-1 TO DCH CA.
 472                                  DAC*    (CCA)
 473                                  DZM*    (CWC)   /PREVENTS DOUBLE INTERRUPTS ON ERRORS!!!!
 474                                  LAC     TCWC            /RESTORE REQ. WC.
 475                                  DAC     CDWDCT
 476                                  DZM     EV1             /REINIT EV.  RETRY FROM ERROR.
 477                                  CRRS                    /READ STATUS IN ORDER TO CHECK FOR READER READY
 478                                  AND     (60)            /AND ON-LINE.
 479                                  SAD     (60)            /STATUS BITS 12, 13 SET?
 480                                  SKP                     /YES, ON-LINE AND READY FOR READ.
 481                                  JMP     ERR1            /NO, NOT READY.  TYPE MSG1 AND WAIT FOR READY.
 482                                  LAC     (CC2)           /CONDITION CODE 2 -- READ CARD.
 483                                  CRLC                    /LOAD CONDITIONS.
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

```
PAGE  13     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

                             CAL     WFCRCB          /WAIT FOR INTERRUPT.
                        /
                        /
                        /
                        /UPON RESUMPTION FOLLOWING WAITFOR, EXAMINE EV AND TAKE THE FOLLOWING
                        /ACTION:
                        /
                        /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 MESSAGE.
                               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)
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

```
PAGE  14     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

536                          /
537                          /  NOW BRING BACK RN FROM RRN, IN CASE RN DESTROYED IN MEANTIME
538                          /
539                                 LAC     RRN
540                                 DAC     RN
541                                 LAC     (IBUF)          /TOP OF INTERNAL BUFFER
542                                 DAC     ICA             /PTR TO BUFFER
543                                 LAW     -20
544                                 DAC     CDCOLC          /CARD COL COUNT
545                          CDRM5  LAW     -5
546                                 DAC     CDR5CT
547                          CDML2  LAC*    ICA             /GET
548                                 SAD     CDRALT          /ALT MODE (12,1,8 PUNCH)?
549                                 JMP     CDGALT          /YES -- TERMINATE BUFFER
550                                 SAD     (7777           /NO -- IS IT AN EOF?
551                                 JMP     EOF             /YES.
552                                 LAC     CDTABL          /NO -- TRANSLATE TO ASCII
553                                 DAC     CDTPTR          /GET TOP OF TABLE AND SET PTR
554                                 LAC     CDTLN1          /SET TABLE LENGTH
555                          CDML4  DAC     CDTLEN          /CURRENT LENGTH/2
556                                 ADD     CDTPTR          /CURRENT TABLE TOP + LENGTH/2
557                                 DAC     CDCPTR
558                                 LAC*    CDCPTR          /GET CURRENT ITEM
559                                 AND     (7777
560                                 SZA!CLL
561                                 ADD     CD7700          /ADD IN REST OF 2'S COMPLEMENT WORD
562                                 TAD*    ICA             /CURRENT COLUMN
563                                 SNA!CLA                 /MATCH FOUND?
564                                 JMP     CDCFND          /YES
565                                 SAD     CDTLEN          /CURRENT TABLE LENGTH =0?
566                                                         /THIS MEANS AN UNKNOWN CARD PUNCH
567                                 JMP     ILLCP           /GO OUTPUT 'ILLEGAL CARD PUNCH'.
568                                 SNL                     /L=0 JUMP UP, L=1 JUMP DOWN TABLE
569                                 JMP     CDDPTR
570                                 LAC     CDCPTR          /SET TABLE TOP TO LOWER HALF
571                                 DAC     CDTPTR
572                          CDDPTR LAC     CDTLEN          /UPDATE TABLE LENGTH
573                                 CLL!RAR
574                                 JMP     CDML4
575                          CDGALT LAW     4000            /ALT MODE
576                                 JMP     CDCPUT
577                          /
578                          EOF    LAC     (1005
579                                 JMP     REQCMA          /SET HDR WDI TO EOF
580                                                         /REQUEST COMPLETE
581                          /
582                          /COME HERE ON MATCH FOUND
583                          /
584                          CDCFND LAC*    CDCPTR          /GET CURRENT ENTRY
585                                 CMA!CLL                 /GEN. LEFTMOST BIT
586                                 TAD     CDTABL+1        /ADD 4000000
587                                 CMA
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

```
PAGE  15     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

 588                            XOR     CDTABL+1        /RESTORE SIXTH BIT
 589                            RAR
 590                 CDCPUT     DAC     CDRWD3          /PUT IN TOP OF 3 WORD SHIFT BLOCK
 591                 CDCLAW     LAW     -7
 592                            DAC     CDR7CT
 593                 CDCPL1     LAC     CDRWD3          /CDEWD3,CDRWD2 & CDRWD1 SHIFT AS A UNIT USING
 594                                                    /THE LINK TO PASS BITS FROM WORD TO WORD
 595                            RAL
 596                            DAC     CDRWD3
 597                            LAC     CDRWD2
 598                            RAL
 599                            DAC     CDRWD2
 600                            LAC     CDRWD1
 601                            RAL
 602                            DAC     CDRWD1
 603                            ISZ     CDR7CT
 604                            JMP     CDCPL1
 605                            ISZ     ICA             /POINT TO NEXT CARD COL
 606                            ISZ     CDR5CT          /HAVE WE PROCESSED 5 WORDS?
 607                            JMP     CDML2           /NO GET ANOTHER ONE
 608                            LAC     CDWDCT          /YES -- UPDATE WORD COUNT AND
 609                            TAD     (2              /CHECK TO SEE IF WE HAVE OVERFLOWED THE
 610                            DAC     CDWDCT          /USER'S BUFFER
 611                            SMA
 612                            JMP     CDVER2          /YES -- WE HAVE OVERFLOWED
 613                            LAC     CDRWD2          /NO -- INSERT 5/7 WORDS IN USER'S BUFFER
 614                            CLL!RAL
 615                            DAC     CDRWD2
 616                            LAC     CDRWD1
 617                            RAL
 618                            DAC*    X13             /STORE FIRST WORD
 619                            LAC     CDRWD2
 620                            DAC*    X13             /STORE SECOND WORD
 621                            ISZ     CDCOLC
 622                            JMP     CDRM5
 623                 /
 624                            .ENDC
 625                 /
 626                            .IFDEF UC15
 627                 /
 628                 /   IN THE CASE OF THE UNICHANNEL, WE RECIEVE A 42(10) WORD
 629                 /  BUFFER. THE FIRST WORD IS A BYTE COUNT (NOW ALWAYS 80(10)).
 630                 /  NOTE THAT AN EOF CARD HAS A BYTE COUNT OF 1!!
 631                 /  SPOOLER DOES CHECKSUM CALCULATION, NOT US.
 632                 /  THE SECOND IS A CHECKSUM SO ENTIRE BUFFER ADDS TO 0
 633                 /  !!!###MODULO 2^16 THAT IS###!!!. THEN ARE 40(10) WORDS
 634                 /  OF 'COMPRESSED COLUMN'. (SEE CR-11 DRIVER MANUAL). EACH
 635                 /  WORD HAS TWO EXTRANEOUS BITS AT LEFT, THE !SECOND CHAR!
 636                 /  OF THE PAIR, AND FINALLY THE FIRST CHAR OF PAIR AT RIGHTMOST
 637                 /  OF WORD. THE PDP-11 HAS ALREADY CHECKED FOR VALID PUNCH
 638                 /  COMBINATIONS (64 VALID CARD ASCII, PLUS 12-1-8 FOR ALTMODE).
 639                 /
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

```
PAGE  16     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

  640        00171 R 750030 A    RETRY   CLA!IAC          /SET VARIABLE SAYBING WE'RE WAITING FOR
  641        00172 R 040407 R            DAC      POST     /INTERRUPT
  642        00173 R 140554 R            DZM      CDON     /AND SAY WE HAVEN'T GOTTEN IT YET
  643        00174 R 200614 R            LAC      TCBP     /ADDR OF TABLE TELLING PDP-11 TO READ CARD
  644        00175 R 100616 R            JMS      CDIU     /ROUTINE TO SEND REQUEST TO PDP-11
  645        00176 R 600057 R            JMP      WFTGR    /WAIT FOR COMPLETION INTERRUPT
  646                               /
  647                               /  COME BACK HERE WHEN CARD IS READ
  648                               /
  649        00177 R 200571 R    GOTCRD  LAC      RRN      /RESTORE RN MODE
  650        00200 R 040564 R            DAC      RN
  651        00201 R 140407 R            DZM      POST     /CLEAR INTERRUPT FLAGS
  652        00202 R 140554 R            DZM      CDON     /BEST TO CLEAR POST FIRST!
  653        00203 R 200605 R            LAC      EV11     /EVENT VARIABLE FROM PDP-11
  654        00204 R 742010 A            RTL               /PDP-11 SIGN BIT TO OUR SIGN BIT
  655        00205 R 745120 A            SPA!CLL!RAR       /SKIP IF OK, START CLEARING HIGH BITS
  656        00206 R 600636 R            JMP      CDUCEC   /GO CHECK WHICH KIND OF PIREX ERROR
  657        00207 R 200572 R            LAC      TX12     /SETUP X12,X13 FOR USER BUFFER
  658        00210 R 060673 R            DAC*     (X12     /MANIPULATIONS. X12 HEADER POINTER
  659        00211 R 200573 R            LAC      TX13     /X13 DATA POINTER
  660        00212 R 060674 R            DAC*     (X13
  661        00213 R 220675 R            LAC*     (IBUF+2 /GET FIRST CHARACTER PAIR (2 WORD HDR)
  662        00214 R 540676 R            SAD      (104611 /SPOOLER USES AN ALT-ALT CARD AS AN END
  663                               /                      /OF DECK CARD, WE SHOULD IGNORE IT!!
  664        00215 R 600171 R            JMP      RETRY    /IT WAS ONE, JUST READ THE NEXT CARD
  665        00216 R 500677 R            AND      (340     /12,11,0 PUNCHES IN FIRST COLM.=EOF
  666        00217 R 340700 R            TAD      (445     /IF IT IS ONE, MAKE A 1005
  667        00220 R 540701 R            SAD      (1005    /WELL, IF SO GO LACE 1005 AS HEADER
  668        00221 R 600420 R            JMP      REQCMA   /EOF CARD, JUST SET HEADER.
  669                               /
  670        00222 R 200675 R            LAC      (IBUF+2 /DATA STARTS AT BUFF+2
  671        00223 R 744010 A            CLL!RAL           /TOP 17 BITS ADDRESS, LAST IS RIGHT-LEFT FLOP
  672        00224 R 040405 R            DAC      CDIPTR   /TO GET INCOMING CHAR'S
  673        00225 R 777660 A            LAW      -120     /80 CHAR'S
  674        00226 R 040560 R            DAC      CDCOLC   /NOTE WE USE COUNTERS DIFERENT ALSO
  675        00227 R 200331 R    PKINI   LAC      PAKI     /INIT 5/7 PACKER TO EXPECT
  676        00230 R 040327 R            DAC      PAKSW    /1ST CHAR OF A BUNCH OF FIVE
  677        00231 R 200566 R            LAC      CDWDCT   /WE USE AS COUNT OF PAIRS, NOT WORDS
  678        00232 R 744020 A            CLL!RAR           /SO DIVIDE BY TWO
  679        00233 R 040566 R            DAC      CDWDCT
  680        00234 R 200405 R    CDRML2  LAC      CDIPTR   /WATCH IT! TOP 17 BITS ADDR, LOW BIT LEFT
  681        00235 R 440405 R            ISZ      CDIPTR   /RIGHT FLIP-FLOP, AND!! POINTER POINTS TO
  682                               /                      /NEXT CHAR, NOT LAST ONE RETREIVED.
  683        00236 R 744020 A            CLL!RAR           /FLIP-FLOP TO LINK, ADDR AC
  684        00237 R 040406 R            DAC      CDT1     /HOLD POINTER IN TEMPORARY
  685        00240 R 220406 R            LAC*     CDT1     /GET CHARACTER PAIR
  686        00241 R 741410 A            SZL!RAL           /THESE THREE GET CORRECT CHAR
  687        00242 R 743030 A            SWHA!SKP          /TO LOW ORDER 8 BITS OF WORD
  688        00243 R 740020 A            RAR
  689        00244 R 500702 R            AND      (377     /STRIP OTHER CHARACTER
  690                               /                      /AT THIS POINT HAVE COLMNS 12,11,0,9,8,1-7
  691                               /                      /WHERE 1-7 CODED IN THREE BITS
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

```
PAGE  17     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

  692        00245 R 040406 R            DAC     CDT1    /HOLD
  693        00246 R 540404 R            SAD     CDALT   /ALT MODE SPECIAL CASE, NO REMAP
  694        00247 R 600260 R            JMP     CDGALT  /REJOIN AS SPECIAL CASE
  695        00250 R 500703 R            AND     (20     /IF NINE PUNCH, PECIAL CASE, REMAP TO 8,1 PUNCH
  696        00251 R 740200 A            SZA             /COMBO FOR OUR TRANSLATE. SKIP IF NOT NINE
  697        00252 R 777771 A            LAW     -7      /ADDED TO '9' GIVES '8' AND '1'
  698        00253 R 340406 R            TAD     CDT1    /REMAPPED,
  699        00254 R 040406 R            DAC     CDT1    /SAVE, NOW TO MOVE BOTTOM FOUR BITS LEFT ONE
  700        00255 R 500664 R            AND     (17     /POSITION (9 POSITION NOW VACATED!)
  701        00256 R 340406 R            TAD     CDT1    /THIS DOES IT, LEAVING LOW ORDER BIT ZERO
  702                           /                         /NOW COLUMNS 12,11,0,8,1-7,ZERO BIT!
  703        00257 R 745000 A            SKP!CLL         /HIDE YOUR HEAD. CLL FOR COMING RTR.SKIP
  704                           /                         /OVER ALT-MODE RE-ENTRY
  705        00260 R 200704 R    CDGALT  LAC     (240    /INDEX TO ALT MODE
  706        00261 R 742020 A            RTR             /RIGHT-LEFT TO LINK, INDEX TO AC
  707        00262 R 340705 R            TAD     (CDTABL /TABLE ADDR
  708        00263 R 040406 R            DAC     CDT1
  709        00264 R 220406 R            LAC*    CDT1    /GET PAIR FROM TRANSLATE TABLE
  710        00265 R 740400 A            SNL             /HERE 0 IS LEFT, IN NORMAL SENSE
  711        00266 R 742030 A            SWHA
  712        00267 R 100323 R            JMS     PAK57   /5/7/ PACKER (IT STRIPS XTRA BITS)
  713        00270 R 440560 R            ISZ     CDCOLC  /80?
  714        00271 R 600234 R            JMP     CDRML2  /NO
  715        00272 R 600410 R            JMP     CDCLOS  /YES
  716                           /
  717                           /  TRANSLATE TABLE 4 GROUPS OF 16 CHAR'S, TWO PER WORD. 8 WORD
  718                           /  SPACE BETWEEN LAST TWO GROUPS, IN WHICH WE PUT OTHER STUFF
  719                           /  CONDITIONALIZED FOR 026-029 OF COURSE. LEFT HAND CHAR IS FIRST.
  720                           /
  721                                   .IFUND DEC026
  722        00273 R 040061 A    CDTABL  040061  /BLANK, 1-PUNCH
  723        00274 R 062063 A            062063  /2-PUNCH,3-PUNCH
  724        00275 R 064065 A            064065  /4,5
  725        00276 R 066067 A            066067  /6,7
  726        00277 R 070071 A            070071  /8,9(ORDERED AS 8-1)
  727        00300 R 072043 A            072043  /8-2,8-3
  728        00301 R 100047 A            100047  /8-4,8-5
  729        00302 R 075042 A            075042  /8-6,8-7
  730        00303 R 060057 A            060057  /0,0-1
  731        00304 R 123124 A            123124  /0-2,0-3
  732        00305 R 125126 A            125126  /0-4,0-5
  733        00306 R 127130 A            127130  /0-6,0-7
  734        00307 R 131132 A            131132  /0-8,0-9(ORDERED AS 0-8-1)
  735        00310 R 135054 A            135054  /0-8-2,0-8-3
  736        00311 R 045137 A            045137  /0-8-4,0-8-5
  737        00312 R 076077 A            076077  /0-8-6,0-8-7
  738        00313 R 055112 A            055112  /11,11-1
  739        00314 R 113114 A            113114  /11-2,11-3
  740        00315 R 115116 A            115116  /11-4,11-5
  741        00316 R 117120 A            117120  /11-6,11-7
  742        00317 R 121122 A            121122  /11-8,11-9(ORDERED AS 11-8-1)
  743        00320 R 041044 A            041044  /11-8-2,11-8-3
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

```
PAGE  18     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

 744        00321 R 052051 A            052051  /11-8-4,11-8-5
 745        00322 R 073134 A            073134  /11-8-6,11-8-7
 746                                    .ENDC
 747                                    .IFDEF DEC026
 748                              CDTABL 040061
 749                                     062063
 750                                     064065
 751                                     066067
 752                                     070071
 753                                     137075
 754                                     100136
 755                                     047134
 756                                     060057
 757                                     123124
 758                                     125126
 759                                     127130
 760                                     131132
 761                                     073054
 762                                     050042
 763                                     043045
 764                                     055112
 765                                     113114
 766                                     115116
 767                                     117120
 768                                     121122
 769                                     072044
 770                                     052133
 771                                     076046
 772                                     .ENDC
 773                                /
 774                                /  NOW THE 8 LOC, BREAK IN THE TABLE
 775                                /
 776                                /  THE 5/7 PACKER, A LITTLE TRICKY PAKSW KEEPS A PC WHICH
 777                                /  'REMEMBERS' WHICH CHARACTER OF 5 WE ARE AT. TO INIT PACKER,
 778                                /  SEE TWO LINES OF CODE AT PAKINT. NORMAL 'FLUSH' OUT WOULD
 779                                /  BE TO SEND NUL CHAR'S UNTIL PAKSW=PAK1. IN THIS
 780                                /  HANDLER, PAST HISTORY SAYS WE TRUNCATE ALWAYS AT A WORD
 781                                /  PAIR BOUNDARY, EVEN FOR SHORT BUFFERS. I AM AFRAID TO
 782                                /  CHANGE THIS, EVEN THOUGH I DON'T LIKE IT.
 783                                /
 784        00323 R 000000 A     PAK57  0                    /CALL WITH CHAR IN AC, (DESTROYED)
 785                                /                         /PUSHES CHAR'S THRU X13. EARLY END CHECK
 786                                /                         /IN CDWDCI.
 787        00324 R 500706 R            AND     (177         /STIP XTRA
 788        00325 R 744000 A            CLL                  /FOR ALL ROTATES AND SWAPS!
 789        00326 R 620327 R            JMP*    PAKSW        /TO WHATEVER ACTION THIS CHAR. NEEDS.
 790        00327 R 740040 A     PAKSW  HLT                  /POINTER TO ACTINS FOR CHARACTER
 791        00330 R 620323 R            JMP*    PAK57        /THAT'S ALL, OUT
 792        00331 R 000345 R     PAK1   PAKST                /INIT PAKSW FOR FIRST CHAR.
 793        00332 R 000000 A     PAKT   0                    /TEMPORARY FOR PARTIAL WORDS
 794                                /
 795                                /  REST OF TRANSLATE TABLE
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

```
PAGE  19     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

 796                            /
 797                                    .IFUND DEC026
 798        00333 R 046101 A            046101  /12,12-1
 799        00334 R 102103 A            102103  /12-2,12-3
 800        00335 R 104105 A            104105  /12-4,12-5
 801        00336 R 106107 A            106107  /12-6,12-7
 802        00337 R 110111 A            110111  /12-8,12-9(ORDERED AS 12-8-1)
 803        00340 R 133056 A            133056  /12-8-2,12-8-3
 804        00341 R 074050 A            074050  /12-8-4,12-8-5
 805        00342 R 053136 A            053136  /12-8-6,12-8-7
 806                                    .ENDC
 807                                    .IFDEF DEC026
 808                                    053101
 809                                    102103
 810                                    104105
 811                                    106107
 812                                    110111
 813                                    077056
 814                                    051135
 815                                    074041
 816                                    .ENDC
 817        00343 R 175000 A            175000          /ALT MODE, FOR BOTH PUNCH SETS.
 818                            /
 819                            /  NOW REST OF 5/7 PACKER
 820                            /
 821        00344 R 100327 R    PAKQ    JMS     PAKSW   /5TH CHAR WRAP BACK TO 1ST. JMS TO PAKSW
 822                            /                       /LEAVES ADDR OF ACTION FOR 1ST.!.
 823        00345 R 742010 A    PAKST   RTL             /1ST CHARACTER ACTION, MOVE TO LEFT OF WORD
 824        00346 R 742030 A            SWHA
 825        00347 R 040332 R            DAC     PAKT    /HOLD AS PARTIALLY ASSEMBLED WORD
 826        00350 R 100327 R            JMS     PAKSW   /LEAVE POINTER TO 2ND CHAR
 827                            /
 828        00351 R 742010 A            RTL             /2ND CHAR ACTION
 829        00352 R 742010 A            RTL
 830        00353 R 240332 R            XOR     PAKT    /MARGE WITH FIRST
 831        00354 R 040332 R            DAC     PAKT    /WAIT FOR PART OF 3RD TO FILL WORD
 832        00355 R 100327 R            JMS     PAKSW   /LEAVE POINTER TO THIRD
 833                            /
 834        00356 R 742020 A            RTR             /3RD, TWO PARTS, FIRST IS TOP 4 BITS
 835        00357 R 740020 A            RAR             /RIGHT JUSTIFIED 1ST WORD OF PAIR
 836        00360 R 040327 R            DAC     PAKSW   /VERY-TEMPORARY IN HERE
 837        00361 R 500664 R            AND     (17     /ZAP OTHER BITS
 838        00362 R 240332 R            XOR     PAKT    /COMPLETE 1ST WORD OF PAIR
 839        00363 R 060013 A            DAC*    X13     /PLACE IN USER BUFFER
 840        00364 R 200327 R            LAC     PAKSW   /GET BACK THIRD CHAR (LINK STILL OK!!!)
 841        00365 R 740020 A            RAR             /2ND JOB, LOW THREE BITS OF CHAR TOP OF
 842        00366 R 500707 R            AND     (700000 /2ND WORD OF PAIR
 843        00367 R 040332 R            DAC     PAKT    /WHEW!, HOLD THAT IN PARTIAL WORD
 844        00370 R 100327 R            JMS     PAKSW   /LEAVE POINTER FOR FOURTH
 845                            /
 846        00371 R 742030 A            SWHA            /4TH, SNUG UP TO 3 BITS ON TOP
 847        00372 R 740020 A            RAR
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

```
PAGE  20     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

 848        00373 R 240332 R            XOR     PAKT    /TUGETHER
 849        00374 R 040332 R            DAC     PAKT
 850        00375 R 100327 R            JMS     PAKSW   /LEAVE POINTER FOR 5TH
 851                              /
 852        00376 R 440566 R            ISZ     CDWDCT  /OVERFLOW SHORT BUFFER?
 853        00377 R 741010 A            SKP!RAL         /NO,  RAL LEAVE XTRA BIT OF PAIR ON RIGHT
 854        00400 R 600452 R            JMP     CDVER2  /UH-OH, GO CORRECT
 855        00401 R 240332 R            XOR     PAKT    /COMPLETE 2ND WORD OF PAIR
 856        00402 R 060013 A            DAC*    X13     /PLACE
 857        00403 R 600344 R            JMP     PAKQ    /GO PLACE PAKSW FOR FIRST CHAR OF FIVE
 858                              /
 859        00404 R 000211 A    CDALT   211
 860        00405 R 000000 A    CDIPTR  0               /POINTER TO INPUT DATA IN INPUT BUFFER
 861                              /                      /FRMAT. LOW BIT RIGHT-LEFT FLIPFLOP
 862                              /                      /TOP 17 BITS ADDRESS
 863        00406 R 000000 A    CDT1    0               /TEMPORARY FOR TRANSLATION
 864        00407 R 000000 A    POST    0               /0 WHEN NOT WAITING FOR INTERRUPT, 1 WHEN YES.
 865                              .ENDC
 866                             / THE BUFFER HAS BEEN REMAPPED -- STORE A 'CR' IN THE TRAILER
 867                             / WORD AND SET UP THE HEADER WORD
 868                             /
 869        00410 R 200710 R    CDCLOS  LAC     (64000
 870        00411 R 060013 A            DAC*    X13             /SET 'CR' IN USER BUFFER
 871        00412 R 200560 R            LAC     CDCOLC          /CDCOLC IS NEGATIVE
 872        00413 R 723022 A            AAC     22
 873        00414 R 744000 A            CLL                     /ROTATE INTO PLACE
 874        00415 R 640711 A            ALS     11              /SHIFT INTO POSITION
 875        00416 R 340565 R            TAD     CDRVAL          /ADD IN BUFFER OVERFLOW IF ANY (BITS 12 & 13 =1)
 876        00417 R 723002 A            AAC     2
 877        00420 R 060012 A    REQCMA  DAC*    X12             /SET HEADER WORD ONE
 878        00421 R 777777 A    REDCOM  LAW     -1      /SET RRN, SAYING NO MORE READ OUTSTANDING
 879        00422 R 040571 R            DAC     RRN
 880        00423 R 750030 A    REQCMP  CLA!IAC
 881        00424 R 100426 R    SEV     JMS     SEVRN   /SUB. TO SET EV, RETURN NODE
 882        00425 R 600060 R            JMP     PQ      /GO LOOK FOR MORE WORK
 883                             /
 884                             /
 885                             /   SEVRN
 886                             /
 887                             /
 888                             /  ROUTINE  IS CALLED WITH VALE FOR EV IN AC
 889                             /  THE NODE ADDR. IS IN RN
 890                             /
 891                             /  EV IS SET, SIGNIFICANT EVENT DECLARED, IOCD ODDE, NODE RETURNED.
 892                             /
 893        00426 R 000000 A    SEVRN   0
 894        00427 R 722000 A            PAL                     /SAVE AC VALUE
 895        00430 R 200564 R            LAC     RN      /NODE ADDR
 896        00431 R 060651 R            DAC*    (R2     /SYSTEM ARGUMENT HOLDER
 897        00432 R 340563 R            TAD     XADJ    /ADJUST FOR PREESENT PAGE
 898        00433 R 721000 A            PAX             /FOR XR ADDRESSING
 899        00434 R 210006 A            LAC     6,X     /EVENT VARIABLE ADDRESS
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

```
PAGE  21     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

 900        00435 R 741200 A            SNA             /SKIP IF REALLY ONE
 901        00436 R 600443 R            JMP     NOSET   /NOPE, SO DON'T SET
 902        00437 R 340563 R            TAD     XADJ    /MODIFY IT FOR ADDRESSING
 903        00440 R 721000 A            PAX
 904        00441 R 730000 A            PLA             /BRING BACK SETTING VALUE
 905        00442 R 050000 A            DAC     0,X     /THERE IT GOES!
 906        00443 R 200711 R    NOSET   LAC     (401000 /DECLARE A SIGNIFICANT EVENT
 907        00444 R 705504 A            ISA
 908        00445 R 200704 R            LAC     (POOL   /GIVE NODE TO POOL
 909        00446 R 060647 R            DAC*    (R1     /SYSTEM ARGUMENT REG
 910        00447 R 120712 R            JMS*    (IOCD   /DECREMENT IO COUNT
 911        00450 R 120713 R            JMS*    (NADD   /GIVE BACK NODE
 912        00451 R 620426 R            JMP*    SEVRN   /THAT/S IT
 913                            /
 914                            /
 915                            /
 916                            / ***** BUFFER OVERFLOW
 917                            /
 918        00452 R 777776 A    CDVER2  LAW     -2              /BACKUP USER BUFFER PTR
 919        00453 R 360674 R            TAD*    (X13)
 920        00454 R 060674 R            DAC*    (X13)
 921        00455 R 200714 R            LAC     (60)            /SET OVERFLOW BITS FOR USE BY CDCLOS
 922        00456 R 040565 R            DAC     CDRVAL
 923        00457 R 600410 R            JMP     CDCLOS
 924                            /
 925        00460 R 777771 A    EVM7    LAW     -7              /ILLEGAL DATA MODE.
 926        00461 R 600424 R            JMP     SEV
 927        00462 R 777750 A    EVM30   LAW     -30             /I/O PARAM. OUT OF PARTITION.
 928        00463 R 600424 R            JMP     SEV
 929                            /
 930                                    .IFUND  UC15
 931                            /
 932                            AEVM6   LAW     -6              /ILLEGAL FUNCTION.
 933                                    JMP     SAEV            /SET ABORT EV.
 934                            /
 935                            /ON ILLEGAL CARD PUNCH, WAIT FOR READER NOT READY FOLLOWED BY
 936                            /READER READY SEQUENCE BEFORE READING ANOTHER CARD.
 937                            /
 938                            ILLCP   LAC     (ERRMG2)        /TYPE 'ILLEGAL CARD PUNCH'.
 939                                    JMS     TTYOUT
 940                                    JMS     WF.SW           /WAIT FOR READER NOT READY.
 941                                            WFOFF           /PSUEDO INSTR. FOR WF.SW.
 942                                    JMS     WF.SW           /WAIT FOR READER READY.
 943                                            WFON            /PSUEDO INSTR. FOR WF.SW.
 944                                    JMP     RETRY           /READ ANOTHER CARD.
 945                            /
 946                            /       SUBR. TO WAIT FOR READER NOT READY OR READY FOR READ
 947                            /       PER PSUEDO INSTR. IN CALLING SEQUENCE. AFTER MARK TIME REQS.,
 948                            /       THE TRIG. EV. IS CHECKED FOR AN ABORT REQ. IN THE QUEUE.
 949                            /       IF TASK REQ. READ IS TO BE ABORTED, THE SUBR. DOESN'T
 950                            /       RETURN NORMALLY,BUT EVENTUALLY JUMPS TO CDABRT.
 951                            /       CALLING SEQUENCE:
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

```
PAGE  22     CD.... 021      CD,... CR15/UC15 CARD READER EDIT #020

 952                          /
 953                          /       JMS     WF.SW
 954                          /               PSUED. INSTR.  (WFOFF OR WFON)
 955                          /       SUBR. RETURN ,IF NO INTERVENING ABORT FOR THIS TASK.
 956                          /
 957                          WF.SW   0
 958                                  LAC*    WF.SW           /GET PSUEDO INSTR.
 959                                  DAC     PV1
 960                                  ISZ     WF.SW           /BUMP EXIT.
 961                          WF.SWA  CRRS                    /READ CARD READER STATUS.
 962                                  AND     (20)            /CHECK FOR READER READY FOR READ.
 963                          PV1     XX                      /SNA OR SZA.  (READER READY IF NON-ZERO AC).
 964                                  JMP*    WF.SW           /EXIT.
 965                                  CAL     MTCPB           /MARK TIME FOR WAIT.
 966                                  CAL     WFECB           /WAIT FOR MARK TIME INTERVAL.
 967                                  DZM     EV
 968                                  LAC     TG              /CHECK FOR ABORT REQ. IN QUEUE.
 969                                  RTL
 970                                  SMA                     /ABORT REQ.?
 971                                  JMP     WF.SWA          /CHECK AGAIN.
 972                                  DZM     TG              /YES.  DEQUEUE ABORT REQ.
 973                                  LAC     PDVNA   /PDVL NODE ADDR.
 974                                  DAC*    (R1)
 975                                  JMS*    (DQRQ)          /DEQUEUE ABRT. REQ. R1,R2,R4,R5,R6,XR,AC
 976                                  NOP                     /ALTERED.  ASSUME ABRT. REQ. IN QUEUE.
 977                                  DAC     RN              /SAVE ABORT REQ. NODE ADDR.
 978                                  TAD     XADJ            /SET XR.
 979                                  PAX
 980                                  LAC     6,X             /GET ABRT. REQ. EV.
 981                                  DAC     ARE
 982                                  LAC     5,X             /CHECK FOR ZERO LUN.
 983                                  AND     (777000)        /BITS 0-8
 984                                  SZA
 985                                  JMP     AEVM6           /ERROR.  NON-ZERO LUN.
 986                                  LAC     2,X             /GET STL. NODE PTR. AND CHECK AGAINST
 987                                  SAD     STLA            /READ REQ. STL NODE PTR. SAME?
 988                                  JMP     CDARD           /YES.  ABORT READ REQ. AND CLEAN UP.
 989                                  LAC     PDVNA   /NO.  CLEAN UP QUEUE OF TASK TO BE ABRTED.
 990                                  DAC*    (R1)            /ALSO RETR. ABRT. REQ. NODE TO POOL AND
 991                                  LAC     RN              /DECR. TRANSF. PEND. CNT.  ABRT. REQ. NODE
 992                                  DAC*    (R2)            /ADDR. TO R2.
 993                                  JMS*    (DMTQ)          /EMPTY REQ. QUEUE OF ALL I/O
 994                                                          /REQ.'S MADE BY TASK BEING ABORTED.
 995                                                          /R1,R2,R3,R5,R6,X10,X11,X12,XR,AC ALTERED.
 996                                  LAC     (1)             /SET ABRT. REQ. EV TO +1.
 997                          SAEV    PAL
 998                                  LAC     ARE             /ABORT REQ. EV.
 999                                  TAD     XADJ
1000                                  PAX
1001                                  PLA
1002                                  DAC     0,X
1003                                  LAC     (401000)



PAGE  23     CD.... 021      CD,... CR15/UC15 CARD READER EDIT #020

1004                                  ISA                     /DECLARE SIGNIF. EVENT.
1005                                  LAC     RN              /RETRN. ABRT. REQ. NODE TO POOL.
1006                                  DAC*    (R2)
1007                                  LAC     (POOL)
1008                                  DAC*    (R1)
1009                                  JMS*    (IOCD)          /DECR. TRANSF. PEND. CNT.
1010                                  JMS*    (NADD)          /RETRN. NODE TO POOL.
1011                                  JMP     WF.SWA          /CHECK AGAIN.
1012                          CDARD   CLAIIAC                 /SET CARD DONE FLAG.
1013                                  DAC     CDON
1014                                  JMP     CDABRT          /PROCEED WITH ABORT.
1015                          /
1016                                  .ENDC
1017                                  .EJECT
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

```
PAGE  24     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

1018                         /
1019                         / EXIT REQUEST (FROM TASK "....REA")
1020                         /
1021        00464 R 200704 R DAEX    LAC     (POOL)          /RETURN REQUEST NODE TO POOL
1022        00465 R 060647 R         DAC*    (R1)
1023        00466 R 200564 R         LAC     RN
1024        00467 R 060651 R         DAC*    (R2)
1025        00470 R 120712 R         JMS*    (IOCD)          /DECREMENT TRANSF. PENDING COUNT
1026        00471 R 120713 R         JMS*    (NADD)
1027                                 .IFUND  UC15
1028                                 LAC     (CC1)           /CONDITION CODE 1 -- CLEAR CONTROL.
1029                                 CRLC
1030                                 CAL     DCPB            /DISCONNECT
1031                                 .ENDC
1032                                 .IFDEF  UC15
1033        00472 R 100625 R         JMS     CLEAR   /CLEAR DEVICE , WAIT FOR COMPLETION
1034        00473 R 440577 R         ISZ     CCPB            /MAKE CONNECT A DISCONNECT (BURP)
1035        00474 R 000577 R         CAL     CCPB            /DISCONNECT
1036                                 .ENDC
1037        00475 R 440570 R         ISZ     PDVTA           /POINT TO ASSIGN INHIBIT FLAG
1038        00476 R 705522 A         .INH                    /INHIBIT INTERRUPTS.
1039        00477 R 160570 R         DZM*    PDVTA           ///ZERO IT
1040        00500 R 705521 A         .ENB                    ///ENABLE INTERRUPTS.
1041        00501 R 000653 R         CAL     (10)            ///EXIT
1042                         /
1043                         /
1044                         /ABORT REQUEST.
1045                         /
1046        00502 R 777000 A CDABRT  LAW     17000   /MASK TO KEEP HALF WORD TO CHECK ABORT VALIDITY
1047        00503 R 510005 A         AND     5,X     /HAS TO BE ZERO TO BE OK
1048        00504 R 740200 A         SZA             /SO SKIP IF OK
1049        00505 R 600116 R         JMP     EVM6    /ERROR RETURNED IF NOT
1050        00506 R 200567 R         LAC     PDVNA   /MT THE DEQUE FOR THE ABORTED TASK
1051        00507 R 060647 R         DAC*    (R1
1052        00510 R 200564 R         LAC     RN      /ABORT NODE
1053        00511 R 060651 R         DAC*    (R2
1054        00512 R 120715 R         JMS*    (DMTQ   /THIS ROUTINE DOES ALL WORK
1055                         /
1056                         /  NOW WAS THIS ABORT FOR AN OUTSTANDING READ?
1057                         /
1058        00513 R 200564 R         LAC     RN      /2+RN IS STL NODE ADDR
1059        00514 R 340563 R         TAD     XADJ    /USE AS IDENTIFIER
1060        00515 R 721000 A         PAX
1061        00516 R 210002 A         LAC     2,X
1062        00517 R 540556 R         SAD     STLA    /SAME ADDR FOR LAST READ DONE
1063        00520 R 751001 A         SKP!CLA!CMA     /SKIP IF SAME, SET UP -1
1064        00521 R 600423 R         JMP     REQCMP  /NOPE, WE'RE DONE, GO GIVE BACK NODE ETC.
1065        00522 R 240571 R         XOR     RRN     /NASTY, MAKES 0 IF NO READ NOW! IN PROGRESS
1066        00523 R 741201 A         SNA!CMA         /SKIP IF READ IN PROGRESS, RECREATE ITS NODE ADDR!
1067        00524 R 600423 R         JMP     REQCMP  /NOPE, JUST COMPLETE
1068        00525 R 060651 R         DAC*    (R2     /GIVE BACK NODE AND IOCD FOR SUSPENDED READ
1069        00526 R 200704 R         LAC     (POOL


PAGE  25     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

1070        00527 R 060647 R         DAC*    (R1
1071        00530 R 120712 R         JMS*    (IOCD
1072        00531 R 120713 R         JMS*    (NADD
1073        00532 R 750001 A         CLA!CMA         /SET READ NOT HERE SWITCH
1074        00533 R 040571 R         DAC     RRN
1075                                 .IFUND  UC15
1076                                 LAC     (CC1    /CLEAR DEVICE
1077                                 CRLC
1078                                 .ENDC
1079                                 .IFDEF  UC15
1080        00534 R 100625 R         JMS     CLEAR   /AND CLEAR FOR UNICHANNEL
1081                                 .ENDC
1082        00535 R 600423 R         JMP     REQCMP          /DONE
1083                         /
1084                         /
1085                         /
1086                         /
1087                                 .EJECT
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

```
PAGE  26     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

1088                          /
1089                          / INTERRUPT SERVICE ROUTINE
1090                          /
1091        00536 R 000000 A  INT     0
1092        00537 R 707762 A          DBA
1093        00540 R 040000 R          DAC     START           /SAVE AC
1094                          .IFUND  UC15
1095                          CRRS                      /READ STATUS INTO AC.
1096                          DAC     EV1               /SAVE FOR TASK LEVEL PROCESSING.
1097                          AND     (2)               /CARD DONE?  BIT 16.
1098                          SNA
1099                          JMP     INT1              /NO.  DON'T CLEAR CARD DONE.
1100                          DAC     CDON     /PLACE 2 INTO CDON TO SAY DONE
1101                          LAC     (CC3)             /YES.  CLEAR CARD DONE.  LEAVE
1102                          CRLC                      /INTERR. AND DCH ENABLED.
1103                      INT1    CRPC                  /CLEAR ALL BUT CARD DONE.
1104                          LAC     (CC4)             /ENABLE INTERRS.  DISABLE DCH
1105                          CRLC                      /NEEDED SINCE CRPC DISABLES INTERRS.
1106                          .ENDC
1107                          /
1108                          .IFDEF  UC15
1109        00541 R 706124 A          CAPI                      /CLEAR FLAG FROM PDP-11
1110        00542 R 200407 R          LAC     POST     /ARE WE WANTING AN INTERRUPT
1111        00543 R 741200 A          SNA              /SKIP IF YES/USE VALUE TO SET
1112        00544 R 600551 R          JMP     INTAC    /NO DO NOTHING
1113        00545 R 040554 R          DAC     CDON     /AS FLAG TO DISTINGUISH CARD DONE FROM CAL
1114        00546 R 040562 R          DAC     TG       /AND SET TG TO WAKE UP CAL LEVEL
1115                          .ENDC
1116        00547 R 200711 R          LAC     (401000)          /DECLARE SIGNIF. EVENT.
1117        00550 R 705504 A          ISA
1118        00551 R 200000 R      INTAC   LAC     START             /RESTORE AC.
1119        00552 R 703344 A          DBR
1120        00553 R 620536 R          JMP*    INT
1121                          .EJECT


PAGE  27     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

1122                          /
1123                              .IFUND  UC15
1124                          /SUBR. TO OUTPUT ERROR MESSAGES VIA ERRLUN.  AC SHOULD CONTAIN
1125                          /ADDRESS OF ERROR MESSAGE BUFFER.
1126                          /
1127                          TTYOUT  0
1128                                  DAC     TECPB4            /SET CPB BUFFER ADDRESS.
1129                                  CAL     TE                /TYPE ERROR MESSAGE.
1130                                  CAL     WFECB             /WAITFOR EV.
1131                                  JMP*    TTYOUT
1132                          /
1133                          /ERROR MESSAGE BUFFERS AND TABLE OF PTRS.:
1134                          /
1135                          ERRPT   .+1
1136                                  ERRMG1
1137                                  ERRMG2
1138                                  ERRMG3
1139                                  ERRMG4
1140                                  ERRMG5
1141                          /
1142                          /
1143                          /
1144                          ERRMG1  ERRMG2-ERRMG1*1000/2+2
1145                                  0
1146                                  .ASCII '*** CD READER NOT READY'<15>
1147                          ERRMG2  ERRMG3-ERRMG2*1000/2+2
1148                                  0
1149                                  .ASCII '*** CD ILLEGAL PUNCH'<15>
1150                          ERRMG3  ERRMG4-ERRMG3*1000/2+2
1151                                  0
1152                                  .ASCII '*** CD PICK ERROR'<15>
1153                          ERRMG4  ERRMG5-ERRMG4*1000/2+2
1154                                  0
1155                                  .ASCII '*** CD DATA MISSED/PHOTO ERROR'<15>
1156                          ERRMG5=.
1157                                  .EJECT
1158                          / ***** CARD COL TO ASCII TRANSLATION TABLE *****
1159                          /
1160                          /EACH TABLE ENTRY REPRESENTS VALID ASCII CARD PUNCHES WITH
1161                          /THE FOLLOWING FORMAT:
1162                          /
1163                          /BITS 0 - 5      SIXBIT ASCII CHARACTER.
1164                          /BITS 6 - 17     CARD PUNCHES WITH THE FOLLOWING MAPPING:
1165                          /
1166                          /BIT 6 = ZONE 12
1167                          /BIT 7 = ZONE 11
1168                          /BITS 8 - 17  = ZONES 0 - 9.
1169                          /THE ASSEMBLER BUILDS THE TWOS COMPLEMENT OF BITS 6-17 VIA THE
1170                          /7777\+1 OPERATION.  THE TABLE IS ORDERED ACCORDING TO INCREASING
1171                          /MAGNITUDE OF CARD PUNCHES(CONSIDERED AS 12 BIT RIGHT JUSTIFIED
1172                          /INTEGER VALUES).
1173                          /EXAMPLE:  ASCII '9' HAS FOLLOWING TABLE REPRESENTATION:
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

```
PAGE  28     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

1174                          /
1175                          /       710001\7777+1
1176                          /
1177                          /WHERE 0001 INDICATES ZONE 9 PUNCHED AND 71 IS SIXBIT ASCII '9'.
1178                          /
1179                          /GRAPHIC CHARACTERS FOR 026 PUNCHES ARE IN PARENTHESES BELOW:
1180                          /
1181                          CDTABL  CDTABL+1
1182                                  400000                  /BLANK
1183                                  710001\7777+1           /9
1184                                  700002\7777+1           /8
1185                                  670004\7777+1           /7
1186                                  CP 340006,420006        /" (\)
1187                                  660010\7777+1           /6
1188                                  CP 470012,750012        /= (')
1189                                  650020\7777+1           /5
1190                                  CP 360022,470022        /, (^)
1191                                  640040\7777+1           /4
1192                                  000042\7777+1           /@
1193                                  630100\7777+1           /3
1194                                  CP 750102,430102        /# (=)
1195                                  620200\7777+1           /2
1196                                  CP 370202,720202        /: (_)
1197                                  610400\7777+1           /1
1198                                  601000\7777+1           /0
1199                                  321001\7777+1           /Z
1200                                  311002\7777+1           /Y
1201                                  301004\7777+1           /X
1202                                  CP 451006,771006        /? (%)
1203                                  271010\7777+1           /W
1204                                  CP 431012,761012        /> (#)
1205                                  261020\7777+1           /V
1206                                  CP 421022,371022        /RIGHT ARROW (")
1207                                  251040\7777+1           /U
1208                                  CP 501042,451042        /% (()
1209                                  241100\7777+1           /T
1210                                  541102\7777+1           /'
1211                                  231200\7777+1           /S
1212                                  CP 731202,351202        /)(;)
1213                                  571400\7777+1           //
1214                                  552000\7777+1           /-
1215                                  222001\7777+1           /R
1216                                  212002\7777+1           /Q
1217                                  202004\7777+1           /P
1218                                  CP 462006,342006        /\ (&)
1219                                  172010\7777+1           /O
1220                                  CP 762012,732012        /; (>)
1221                                  162020\7777+1           /N
1222                                  CP 332022,512022        /) ([)
1223                                  152040\7777+1           /M
1224                                  522042\7777+1           /*
1225                                  142100\7777+1           /L


PAGE  29     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

1226                                  442102\7777+1           /S
1227                                  132200\7777+1           /K
1228                                  CP 722202,412202        /! (:)
1229                                  122400\7777+1           /J
1230                                  CP 534000,464000        /& (+)
1231                                  114001\7777+1           /I
1232                                  104002\7777+1           /H
1233                                  074004\7777+1           /G
1234                                  CP 414006,364006        /^ (!)
1235                                  064010\7777+1           /F
1236                                  CP 744012,534012        /+ (<)
1237                                  054020\7777+1           /E
1238                                  CP 354022,504022        /( (])
1239                                  044040\7777+1           /D
1240                                  CP 514042,744042        /< (})
1241                                  034100\7777+1           /C
1242                                  564102\7777+1           /.
1243                                  024200\7777+1           /B
1244                                  CP 774202,334202        /[ (?)
1245                                  014400\7777+1           /2
1246                          CDTLN1  .-1-CDTABL/2
1247                          CDRALT  4402
1248                                  .ENDC
1249                                  .EJECT
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

```
PAGE  30     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

1250                          /
1251                          / ***** INTERNAL VARIABLES *****
1252                          /
1253        00554 R 000001 A  CDON    1               /CARD DONE FLAG.
1254        00555 R 000000 A  TST     0               /TEMP STORAGE FOR STATUS.
1255        00556 R 000000 A  STLA    0               /STL NODE, ADDR.
1256        00557 R 000000 A  ARE     0               /ABORT REQ. EV.
1257        00560 R 000000 A  CDCOLC  0               /CARD COL COUNT USED IN TRANSLATING CARDS
1258        00561 R 000000 A  EV      0               /INTERNAL EVENT VARIABLE
1259        00562 R 000000 A  TG      0               /TRIGGER EVENT VARIABLE
1260        00563 R 000000 A  XADJ     0              /XR ADJUST CONSTANT TO SUBTRACT PAGE BITS
1261        00564 R 000000 A  RN      0               /ADDRESS OF THE REQUEST NODE PICKED FROM AUEUE
1262        00565 R 000000 A  CDRVAL  0               /BUFFER OVERFLOW FLAG WORD
1263        00566 R 000000 A  CDWDCT  0               /WORD COUNT CHECK WORD SET FROM I/O REQUEST
1264                          /
1265                                .IFUND  UC15
1266                          /
1267                          /  SAVE SOME ROOM FOR UC15, THESE ARE NOT NEEDED
1268                          /
1269                          ICA     0               /INTERNAL BUFFER CURRENT ADDRESS POINTER
1270                          CDR7CT  0               /SEVEN COUNTER USED BY THE 5/7 ASCII PACKING ROUTINE
1271                          CDR5CT  0               /COUNTER FOR 5/7 ASCII PACKING
1272                          CDTPTR  0               /POINTER TO TRANSLATION TABLE
1273                          CDTLEN  0               /TRANSLATION TABLE LENGTH
1274                          CD7700  770000          /USED IN CARD TRANSLATION
1275                          CDCPTR  0               /POINTER TO CURRENT INTEM IN TRANSLATION TABLE
1276                          CDRWD3  0               //
1277                          CDRWD2  0               // THREE WORD SHIFT REG. FOR 5/7 ASCII PACKING
1278                          CDRWD1  0               //
1279                          EV1     0               /CARD READER EV.
1280                          /
1281                                .ENDC
1282                          /
1283        00567 R 000000 A  PDVNA   0               /PHYSICAL DEVICE NODE ADDRESS
1284        00570 R 000000 A  PDVTA   0               /ADDRESS OF ADDRESS OF IEV IN PHY DEV NODE
1285        00571 R 777777 A  RRN     777777          /READ BEING PROC. FLAG.  -1 IF NOT BEING
1286                                                  /PROCESSED.  READ REQ. NODE ADDRESS IF BEING
1287                                                  /PROCESSED.
1288        00572 R 000000 A  TX12    0               /TEMP. FOR X12 STOR.
1289        00573 R 000000 A  TX13    0               /TEMP. FOR X13 STOR.
1290        00574 R 000000 A  TCWC    0               /TEMP. FOR REQ. WC.
1291                          /
1292                                .EJECT
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

```
PAGE  31     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

1293                          /
1294                          / ***** CAL PARAMETER BLOCKS *****
1295                          /
1296                          /
1297        00575 R 000020 A  WFTCPB  20                      /WAIT FOR TRIGGER CPB
1298        00576 R 000562 R          TG
1299                          /
1300        00577 R 000011 A  CCPB    11                      /CONNECT CPB
1301        00600 R 000561 R          EV
1302        00601 R 000015 A          15                      /LINENUMBER
1303        00602 R 000536 R          INT                     /ENTRY ADDRESS OF INTERRUPT SERVICE ROUTINE
1304                          /
1305                                  .IFUND  UC15
1306                          /
1307                          /  UC15 SAVE SPACE BY LEAVING OUT SOME CAL'S
1308                          /
1309                          /
1310                          /
1311                          WFECB   20                       /WAIT FOR EV CPB
1312                                  EV
1313                          /
1314                          DCPB    12                      /DISCONNECT CPB
1315                                  0                       /EV ADDRESS
1316                                  15                      /INTERRUPT LINE NUMBER
1317                                  INT                     /CURRENT INTERRUPT TRANSFER ADDRESS
1318                          /
1319                          TE      2700                    /WRITE TO ERRLUN.
1320                                  EV
1321                                  ERRLUN          /WRITE OUT THE ERROR MESSAG TO THE DESIRED
1322                                                  /TELETYPE
1323                                  2
1324                          TECPB4  XX
1325                          /
1326                          MTCPB   13                      /MARK TIME REQ.
1327                                  EV
1328                                  12                      /12 UNITS.
1329                                  1                       /UNIT (TICK).
1330                          /
1331                          WFCRCB  20              /WAIRFOR CR INTERRS.
1332                                  EV1
1333                          /
1334                          WFCRCD  20              /WAIT FOR CARD DONE FLAG TO BE SET.
1335                                  CDON
1336                          /
1337                                  .ENDC
1338                          /
1339                          /
1340                                  .IFDEF  UC15
1341                          /
1342                          /  I/O INFORMATION , ROUTINES , ETC. FOR UC15
1343                          /
1344                          /  TCB (TASK CONTROL BLOCK) TELLING PDP-11 TO SEND US A CARD
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

```
PAGE  32     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

1345                         /
1346        00603 R 026401 A    TCB     APISLT*400+APILVL       /TELL PDP-11 WHERE TO COME BACK
1347        00604 R 000005 A            DEVCOD                  /PIREX CODE FOR CD;THE 200 BIT SAYS
1348                         /                                  /WE ARE NOT TO BE SPOOLED.
1349        00605 R 000000 A    EV11    0                       /EVENT VARIABLE FROM PDP11 TO US
1350        00606 R 000000 A            0                       /DUMMY, HIGH PORTION OF 18 BIT
1351                         /                                  /ADRESS. NOT PRESENTLY USED
1352        00607 R 000001 R            IBUF                    /POINTER TO BUFFER TO PUT CARD IN
1353        00610 R 000000 A            0                       /UNIT #; FOR FUTURE GENERATIONS.
1354                         /
1355                         /  TCB TO TELL PDP11 TO CLEAR OUT CARD READER DEVICE
1356                         /
1357        00611 R 000000 A    TCBK    0               /THIS WORKS, SEE PIREX FOR INFO.
1358        00612 R 002600 A            DEVCOD&177*400+200
1359        00613 R 000000 A    EV11K   0               /EVENT VARIABLE FOR CLEAR OPERTAION
1360                         /
1361                         /  POINTERS TO TCB, TCBK
1362                         /
1363        00614 R 000603 R    TCBP    TCB
1364        00615 R 000611 R    TCBKP   TCBK
1365                         /
1366                         /
1367                         /  CDIU IS THE SUBROUTINE TO SEND A TCB TO THE PDP-11
1368                         /
1369                         /  CAL WITH THE ADRESS OF THE TCB IN THE AC
1370                         /
1371        00616 R 000000 A    CDIU    0
1372        00617 R 140605 R            DZM     EV11    /CLEAR ONE COMING FROM PDP-11
1373        00620 R 140613 R            DZM     EV11K   /AND THE OTHER ONE, IN CASE IT  USED
1374        00621 R 706001 A            SIOA            /SKIP IF PDP-11 CAN TAKE REQUEST
1375        00622 R 600621 R            JMP     .-1
1376        00623 R 706006 A            LIOR            /TELL IT TO DO TCB WHOSE ADDRESS IN AC
1377        00624 R 620616 R            JMP*    CDIU    /THAT'S ALL THERE IS TO IT.
1378                         /
1379                         /
1380                         /  CLEAR  CLEARS SWITCHES, AND CD IN PIREX, WAITS FOR COMPLETE
1381                         /
1382        00625 R 000000 A    CLEAR   0
1383        00626 R 140407 R            DZM     POST
1384        00627 R 140554 R            DZM     CDON
1385        00630 R 200615 R            LAC     TCBKP   /TCB FOR CLEAR
1386        00631 R 100616 R            JMS     CDIU
1387        00632 R 000634 R            CAL     WFCLER  /WAIT FOR CLEAROUT
1388        00633 R 620625 R            JMP*    CLEAR
1389                         /
1390        00634 R 000020 A    WFCLER  20
1391        00635 R 000613 R            EV11K
1392                         /  CDUCEC EXAMINES NEGATIVE EVENT VARIABLES FROM PIREX
1393                         /
1394        00636 R 744020 A    CDUCEC  CLL!RAR         /CLEAR OTHER TOP BIT
1395        00637 R 340716 R            TAD     (600000 /SIGN EXTEND TO PDP-15 WORD
1396        00640 R 540717 R            SAD     (777001 /THIS ONLY 'LEGAL' VALUE AT PRESENT
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

```
PAGE  33     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

1397        00641 R 600171 R            JMP     RETRY   /THAT SAYS PIREX IS OUT OF NODES,
1398                              /                       /WE SHOULD TRY AGAIN TO GET ONE
1399        00642 R 100426 R            JMS     SEVRN   /OTHERS, RETURN NEG VARIABLE AS EV.
1400                              /                       /THIS IS SLIGHTLY FLAKEY, BUT WE
1401                              /                       /REALLY SHOULD NEVER GET HERE!?!?
1402        00643 R 777777 A            LAW     -1      /SAY NO MORE READ OUTSTANDING
1403        00644 R 040571 R            DAC     RRN
1404        00645 R 600060 R            JMP     PQ      /BACK TO LOOK FOR MORE WORK
1405                              /
1406                              /
1407                                    .ENDC
1408                 000000 R           .END START
            00646 R 000252 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 000012 A *L
            00674 R 000013 A *L
            00675 R 000003 R *L
            00676 R 104611 A *L
            00677 R 000340 A *L
            00700 R 000445 A *L
            00701 R 001005 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


PAGE  34     CD.... 021      CD.... CR15/UC15 CARD READER EDIT #020

            00716 R 600000 A *L
            00717 R 777001 A *L
               SIZE=00720     NO ERROR LINES
```

Figure 4-2 (Cont.)
XVM CR11 XVM/RSX Handler

4.6.3.3 Requests - Following handler initialization, requests can be processed. Note that the request dequeuing algorithm (see Figure 4-2 lines 352-407) 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 XVM. The latter condition implies that the handler's interrupt service routine (Figure 4-2, lines 1091-1120) 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 353-357). 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 359-362). 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 368-407).

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 1058-1067). If so, the handler should issue a "clear device directive" to PIREX to stop the I/O in progress (see Figure 4-2, lines 1073-1080).

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 1033).

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 XVM 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 UNI-CHANNEL Card Reader handler when it processes a read request. (Refer to Figure 4-2).

1. Dequeue the I/O request node (lines 352-407)
2. Check the range of the task TCB and buffer (lines 440-465).
3. Clear the TCB event variable (line 1372).
4. Clear the "I/O Done" flage (line 642).
5. Set the "Interrupt Expected" flage (lines 640-641).
6. Issue the READ TCB to the Card Reader Driver in PIREX (lines 1374-1376).
7. Wait for the Trigger Event Variable (line 352).
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" flage (lines 113-114).
9. The handler then checks the Event Variable sent back by PIREX (lines 653-656).
10. Convert the data to XVM card format and transfer it to the task's buffer (lines 670-879).

11. Set the task's Event Variable (lines 880-881).

12. Wait for the next request (line 352).

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 XVM/RSX POLLER task should be modified to interrogate PIREX concerning the status of the new device.

4.7 BUILDING A XVM/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 XVM handlers. While a rudimentary device driver is typical, a PIREX task can be as complex as a full handler. The 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 and Section 4.5) consists of:

1. Entries on PIREX internal lists.

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

3. 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.

4. 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.

5. The task request setup/processing section.

6. The task interrupt processing section, if the task is a device driver.

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

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 allow the interrupt routine to save the general-purpose registers on the current task stack pointer and lower the system to the priority level of this task. (The interrupt section accomplishes this by calling R.SAVE.)

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.

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

```
PIREX,142       MAC11 XVM V1A000  PAGE 28
LINE PRINTER DRIVER FOR LP11/15
4                       .SBTTL  LINE PRINTER DRIVER FOR LP11/15
5                       .EVEN
6                ;
7         177514 LPCSR=177514
8         177516 LPBUF=177516
9         000006 LPSA=6
10        000012 LPTOT=12
11        000014 LPSTAT=14
12        001264 LPEST=LP.EST+4   ;ADDR IN PIREX ERROR TABLE FOR NOT READY
13        001262 LPUNN=LP.EST+2   ;ADDR FOR UNIT # (FOR NOW 0)
14        000004 LPTCOD=4         ;LINE PRINTER TASK CODE
15        006414 LPEOF=6414               ;EOF CODE(DATA) FOR SPOOLING
16               ;
17               ;
18               ;
19               ;
20               ;  MAKE THE PDP-15 DO ALL THE WORK. THE PDP-11 SIMPLY GET S A COUNT
21               ;  OF CHARACTERS TO PRINT OUT. WE TREAT THE CONTROL CHARACTERS
22               ;  12,15, AND 14 ONLY. A MINUS CHARACTER IS CONVERTED INTO MINUS
23               ;  THAT NUMBER OF SPACES. NOTE ALL REAL ASCII CHAR'S HAVE A ZERO LEADING BIT!
24               ;  EACH LINE HAS AN IMPLIED CARRIAGE RETURN THAT IS ADDED BY THE DRIVER
25               ;  RATHER THAN SENT BY THE PDP-15
26               ;
27               ;  NOTE, IF HEADER WORD OF BUFFER HAS 400 BIT SET, IT IS
28               ;  IMAGE MODE, AND WE NIETHER BUT ON LF OR CR!!
29               ;
30               ;
31               ; CALL TO ROUTINE HAS ADDRESS OF TCB IN HANDLER BUSY (IDLE) REGISTER
32               ;
33 06754                .BLOCK  8.+EAESTK*4
34 07054 177514         .WORD   LPCSR           ;ADDRESS OF LPCSR CONTROL STATUS
35                                              ;    REGISTER USED TO RESET DEVICE
36                                              ;    ON STOP I/O OPERATIONS.
37 07056 000000         .WORD   0               ;TCB POINTER (EXTENDED BITS)
38 07060 000000         .WORD   0               ;TCB POINTER (LOWER 16 BITS). THIS
39                                              ;    WORD IS USED AS THE IDLE/BUSY
40                                              ;    SWITCH FOR THE DEVICE DRIVER.
41               ;
42 07062         LP:
43 07062 005067          CLR     LP.CL          ;CLEAR OUT ANY PENDING TIMER REQUESTS FOR US.
         172300
44 07066 016700          MOV     LP-2,R0        ;SETUP R0 TO POINT TO TCB
         177766
45 07072 005060          CLR     LPSTAT(R0)     ;CLEAR STATUS FLAG IN TCB
         000014
46 07076 016001          MOV     LPSA+2(R0),R1  ;GET BUFFER START ADDRESS
         000010
47 07102 005760          TST     LPSA(R0)       ;DONIT RELOCATE ADDRESS IF BIT 15
         000006
48 07106 100403          BMI     1$             ;    IS ON.
49 07110 006301          ASL     R1             ;RELOCATE ADDRESS (WORD TO BYTE POINTER)
50 07112 066701          ADD     MEMSIZ,R1      ;(+ 11'S OWN LOCAL MEMORY)
         170722
51 07116 112102 1$:      MOVB    (R1)+,R2
52 07120 042702          BIC     #177400,R2     ;CLEAR OUT TOP OF REGISTER
         177400
53 07124 112767          MOVB    #15,LPEOL      ;DEFAULT, ASCII, HERE IS <CR>
         000015
         000610
54 07132 062701          ADD     #2,R1   ;INC R1 BY 2 (BR=134)
         000002
55 07136 112721          MOVB    #12,(R1)+      ;DEFAULT, PRECEED LINE WITH LINE FEED
         000012
56 07142 105067          CLRB    LPERWT         ;RESET ERROR WAIT SWITCH
         000575
57                       .IFNDF  $NOSW          ;##124##IF $NOSW, DISABLE ALL SWITCH INTERACT
58 07146 032767          BIT     #140000,SPOLSW ;SPOOLER ENABLED & RUNNNG
         140000
         171672
59 07154 001427          BEQ     6$             ;GO TO DISABLE HALT AT EOF (BR=135)
```

Figure 4-3
UNICHANNEL LP Driver

```
PIREX,142       MAC11 XVM V1A000  PAGE 28+
LINE PRINTER DRIVER FOR LP11/15
60 07156 022711          CMP     #LPEOF,(R1)      ;EOF RECORD?
         000414
61 07162 001421          BEQ     5$               ;CURRENT TCB CONTAINS EOF (BR=135)
62 07164 105767          TSTB    LPEFWT           ;WAS LAST RECORD AN EOF ? (BR=135)
         000554
63 07170 001423          BEQ     2$               ;NO - BRANCH TO NORMAL CODE (BR=135)
64 07172 105067          CLRB    LPEFWT           ;YES - CLEAR SWITCH FOR NEXT USE (BR=135)
         000546
65 07176 032767          BIT     #2,SW            ;IS SWITCH 2 UP ON 11 CONSOLE ? (BR=135)
         000002
         170364
66 07204 001415          BEQ     2$               ;NO - RESUME NORMAL CODE (BR=135)
67 07206 012767          MOV     #LPECHK,LP,CL+2  ;YES - SET UP CLOCK (BR=135)
         007626
         172154
68 07214 012767          MOV     #170,LP,CL       ;TWO SECOND RETRY (BR=135)
         000170
         172144
69 07222                 SEXIT   WAITST           ;EXIT TO SYSTEM
   07222 000004          IOT
   07224    000          .BYTE   0,WAITST
   07225    002
70 07226 105267 5$:      INCB    LPEFWT           ;SET EOF FLAG FOR NEXT TCB (BR=135)
         000512
71 07232 000402          BR      2$               ;RESUME NORMAL CODE (BR=135)
72 07234 105067 6$:      CLRB    LPEFWT           ;CLEAR FLAG - IN CASE SPOOLER JUST TURNED OFF (BR=135)
         000504
73                       .ENDC
74 07240 132761 2$:      BITB    #1,-3(R1)        ;400  BIT SET IN HEADER IF IMAGE
         000001
         177775
75 07246 001403          BEQ     3$               ;NOT IMAGE, CHECK FORMS CONTROL
76 07250 105067          CLRB    LPEOL            ;IMAGE, DON'T FORCE CR AFTER MESSAGE
         000466
77 07254 000410          BR      4$               ;ALLOW ALL FORMS CONTROL
78 07256 122711 3$:      CMPB    #14,(R1)         ;FIRST CHAR FORM FEED?
         000014
79 07262 001405          BEQ     4$               ;YES, DON'T ADD LINE FEED TO LINE
80 07264 122711          CMPB    #15,(R1)         ;FIRST CHAR CARRIAGE RETURN
         000015
81 07270 001402          BEQ     4$               ;YES, DON'T ADD LINE FEED TO LINE
82 07272 005301          DEC     R1               ;MOVE POINTER BACK TO LINE FEED
83 07274 005202          INC     R2               ;COUNT ADDITION OF LF TO BUFFER
84 07276 010267 4$:      MOV     R2,LPBTCT        ;SAVE COUNT
         000434
85 07302 010167          MOV     R1,LPBUFF        ;SAVE POINTER
         000426
86 07306 105067          CLRB    LPTAB
         000426
87 07312 105767          TSTB    LPBUF            ;HISTORY SAYS THIS HERE
         170200
88 07316 052767          BIS     #100,LPCSR       ;ENABLE INTERRUPTS TO LP GOING
         000100
         170170
89 07324                 SEXIT   WAITST           ;EXIT IN A WAIT STATE AND RESCAN
   07324 000004          IOT
   07326    000          .BYTE   0,WAITST
   07327    002
90                                                ;    THE ATL NOW,
91                ;
92                ;
```

Figure 4-3 (cont)
UNICHANNEL LP Driver

```
PIREX.142       MAC11 XVM V1A000  PAGE 29
LINE PRINTER DRIVER FOR LP11/15
1                       ;       LP INTERRUPT ENTRANCE
2                       ;
3  007330               LPINT:
4  007330 042767                BIC     #100,LPCSR      ;DISABLE LP INTERRUPT
          000100
          170156
5  007336 004067                JSR     R0,R.SAVE       ;SAVE REGISTERS
          172444
6  007342 000004                4                       ;TASK CODE
7  007344 016700                MOV     LP-2,R0         ;GET TCB POINTER
          177510
8  007350 001511                BEQ     LPXT            ;IGNORE IF ITS ALREADY BEEN STOPPED BY
9                                                       ;     A STOP I/O REQUEST.
10 07352 005767                 TST     LPCSR           ;CHECK FOR ERROR
          170136
11 07356 100454                 BMI     LPERR           ;YES
12 07360 005067                 CLR     LP.CL           ;CLEAR OUT ANY PENDING TIMER REQUEST FOR US.
          172002
13 07364               LPLOP:
14 07364 105767                 TSTB    LPCSR           ;IS PRINTER CURRENTLY GOING?
          170124
15 07370 100043                 BPL     LPSTIL          ;YES:  FORGET CHAR FOR NOW
16 07372 105767                 TSTB    LPTAB           ;IN TAB EXPANSION TO SPACES?
          000342
17 07376 100421                 BMI     4$              ;YES
18 07400 005367                 DEC     LPBTCT          ;DECR CHAR COUNT
          000332
19 07404 100424                 BMI     5$              ;WENT TO -1, MAKE CR TO FINISH LINE
20 07406 105777                 TSTB    @LPBUFF         ;MINUS BYTE IS TAB EXPANSION COUNT
          000322
21 07412 100406                 BMI     6$              ;IS ONE, GO SET UP
22 07414 117767                 MOVB    @LPBUFF,LPBUF   ;STICK CHAR INTO LINE PRINTER BUFFER
          000314
          170074
23 07422 005267                 INC     LPBUFF          ;MOVE POINTER TO NEXT CHAR
          000306
24 07426 000756                 BR      LPLOP           ;GO DO NEXT
25                      ;
26 07430 117767         6$:     MOVB    @LPBUFF,LPTAB   ;SET UP TAB COUNT (MINUS, A LA 15)
          000300
          000302
27 07436 005267                 INC     LPBUFF
          000272
28 07442 105267         4$:     INCB    LPTAB           ;COUNT A SPACE FOR THIS TAB
          000272
29 07446 112767                 MOVB    #40,LPBUF       ;SPACE TO LINE PRINTER
          000040
          170042
30 07454 000743                 BR      LPLOP           ;GO DO NEXT
31 07456 105767         5$:     TSTB    LPEOL           ;IMAGE OR ASCII
          000260
32 07462 001403                 BEQ     7$              ;IMAGE, DON'T FORCE <CR>
33 07464 116767                 MOVB    LPEOL,LPBUF     ;ASCII, HERE IS <CARRIAGE RETURN>
          000252
          170024
34 07472 005260         7$:     INC     LPSTAT(R0)      ;SET REV TO GOOD COMPLETION
          000014
35 07476 000421                 BR      LPXIT
36                      ;
37 07500 052767         LPSTIL: BIS     #100,LPCSR      ;ENABLE INTERRUPT ON LP
          000100
          170006
38 07506 000413                 BR      LPXIT1          ;RESTORE R0-R5 AND RETURN
39                      ;
40 07510 105267         LPERR:  INCB    LPERWT          ;SET ERROR WAIT SW.
          000227
41 07514 112767                 MOVB    #4,LPEST        ;ERROR CODE 1,NOT READY TO TABLE
          000004
          171542
```

Figure 4-3 (cont)
UNICHANNEL LP Driver

```
PIREX.142       MAC11 XVM V1A000  PAGE 29+
LINE PRINTER DRIVER FOR LP11/15
42 07522 012767 LPERR1: MOV     #LPCHK,LP,CL+2  ;ADDR. FOR TIMER REQ.
         007646
         171640
43 07530 012767         MOV     #170,LP,CL      ;2 SECS. IN TICKS(OCTAL)
         000170
         171630
44 07536 000167 LPXIT1: JMP     DEQU1           ;SCHEDULE NEXT TASK
         173616
45                      ;
46 07542 105067 LPXIT:  CLRB    LPEST           ;INDICATE SUCCESSFULL OPERATION
         171516
47 07546 052767         BIS     #340,PS         ;INHIBIT INTERRUPTS
         000340
         170222
48 07554 005067         CLR     LPCSR           ;SHUT DOWN DEVICE
         167734
49 07560 012701         MOV     #1,R1           ;TELL CALLER DONE
         000001
50 07564 016700         MOV     LP-2,R0         ;GET TCBP
         177270
51 07570                CALL    SEND15          ;TELL CALLER DONE
   07570 004767         JSR     PC,SEND15
         173626
52 07574        LPXT:
53 07574 052767         BIS     #340,PS         ;INHIBIT INTERRUPTS
         000340
         170174
54 07602 005067         CLR     LP-2            ;CLEAR BUSY(IDLE) FLAG
         177252
55 07606 005067         CLR     LP-4
         177244
56 07612 012703         MOV     #LP,R3          ;DEQUEUE ANOTHER REQUEST IF ANY
         007662
57 07616 012701         MOV     #LP,LH,R1       ;     IN THIS DRIVERS DEQUE.
         001452
58 07622 000167         JMP     DEQU
         173450
59                      ;
60                      ;
61                      ;
62                      ;               SUBROUTINE TO FIELD CLOCK COUNT-DOWN
63                      ;
64                      ;
65 07626 005767 LPFCHK: TST     LP-2            ;HAVE WE BEEN DISABLED  ? (BR=135)
         177226
66 07632 001437         BEQ     LPCX            ;YES = RETURN TO CLOCK = NO RETRY (BR=135)
67 07634 032767         BIT     #2,SW           ;NO = IS SWITCH 2 STILL UP ? (BR=135)
         000002
         167726
```

Figure 4-3 (cont)
UNICHANNEL LP Driver

```
PIREX,142       MAC11 XVM V1A000  PAGE 29*
LINE PRINTER DRIVER FOR LP11/15
 68 07642 001033          BNE     LPCXIT          ;YES - SET UP CLOCK RETRY (BR=135)
 69 07644 000406          BR      LPCLK           ;NO - SET UP RETRY OF TCB (BR=135)
 70 07646 005767 LPCHK:   TST     LP+2            ;HAVE WE BEEN DISABLED
          177206
 71 07652 001427          BEQ     LPCX            ;IF YES, EXIT, LEAVING CLOCK DISABLED (BR=135)
 72 07654 005767          TST     LPCSR           ;DOES ERROR STILL EXIST ? (BR=135)
          167634
 73 07660 100422          BMI     LPCXIT          ;YES - SET UP CLOCK RETRY (BR=135)
 74 07662 012702 LPCLK:   MOV     #LPTCOD*2,R2    ;SCAN ATL FOR OUR NODE (BR=135)
          000010
 75 07666 016201          MOV     ATLNP(R2),R1
          001144
 76 07672 012767          MOV     #LP,LP+12       ;RESTART AT BEGINNING OF REQ.
          007062
          177150
 77 07700 042761          BIC     #17,A.TS(R1)    ;R1 POINTS TO OUR NODE, MAKE RUNNABLE
          000017
          000006
 78 07706 012761          MOV     #LP+26,A.SP(R1) ;SET UP STACK POINTER
          007034
          000004
 79 07714 006202          ASR     R2              ;MAKE BYTE ADDRESSING
 80 07716 116267          MOVB    LEVEL(R2),LP+10 ;SET UP PS
          001125
          177126
 81 07724 000207          RTS     PC              ;RETURN TO CLOCK (BR=135)
 82 07726 012710 LPCXIT:  MOV     #170,(R0)       ;R0 POINTS TO TIMER ENTRY
          000170
 83 07732 000207 LPCX:    RTS     PC              ;RETURNS TO CLOCK
 84                       ;
 85 07734 000000 LPBUFF:  .WORD   0               ;BUFFER POINTER
 86 07736 000000 LPATCT:  .WORD   0               ;BYTE COUNT
 87 07740 000000 LPTAB:   .WORD   0               ;TAB LOCATION
 88 07742     000 LPEOL:  .BYTE   0               ;0 IF IMAGE, 15 IF ASCII
 89 07743     000 LPERWT: .BYTE   0               ;MAKE EVEN
 90 07744     000 LPEFWT: .BYTE   0               ;EOF WAS LAST RECORD FLAG (BR=135)
 91                       .EVEN                   ;MAKE EVEN (BR=135)
 92                       ;
```

Figure 4-3 (cont)
UNICHANNEL LP Driver

1. Equates, device locations, etc. (Page 28, lines 7-15).

2. Initialization and I/O request section (Page 28, lines 1-90); 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 on the clearing of a wait-at-end of file condition and either retry the offending operation or set another wake-up call (Page 29, lines 61-91).

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 28, line 43).

2. Setting up a pointer to the data buffer and relocating the pointer value if it comes from the XVM (Page 28, lines 44-50, 74-87).

3. Various device dependent operations (Page 28, lines 51-56).

4. Detect and initiate halt at end of file procedure (Page 28, line 57-73).

5. Start up the device (Page 28, line 88).

6. Exit in a WAIT state (Page 28, line 89) 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 29, 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 29, lines 5 and 6). R.SAVE is called the task's "TCN" as a parameter and passed.

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

3. Test the task busy idle switch to see if the request has been cancelled (Page 27, 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).

4. Perform task interrupt processing and error checking (Page 29, 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 29, lines 40-44). 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 error.

7. If the operation was successfully completed, use the normal exit procedure described in Section 4.7.2.4 (Page 29, lines 46-58).

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. A SEXIT in state "0" causes the system to rescan the ATL list for the highest priority task.

There are actually three modes in which a task may exit. In the first mode, is 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 XVM 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 DEQUE
             ;  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 task 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 29, lines 46-58) 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 29, line 57). 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 wake-up, to a location chosen by the clock set up section. (Figure 4-3, Page 29, line 44).

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 dealy into the Clock Table "CLTABL" (See Section 3.3.4) Figure 4-3, Page 29, lines 42-43) 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 29, lines 65, 70-71, 83.)

2. Determine if the error has been corrected. If not, reset the timer and RTS PC to the Clock Task. (Page 29, lines 72, 73, 82, 83.)

3. If the error has been corrected, reprocess the original TCB request and return to the Clock Task. (Page 29, lines 74-81.) This will cause PIREX to retry the TCB.

4.7.4 Assembly and Testing

4.7.4.1 Assembly and Loading - New PIREX device driver 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 an XVM program to load the background task binary into XVM memory. (See SPOL15 for an example of the required technique.) The XVM program must then issue a CONNECT Directive. 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 accommodate 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 XVM/DOS 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 XVM/DOS monitor should be used.

CHAPTER 5
SPOOLER DESIGN AND THEORY OF OPERATION

5.1 INTRODUCTION

This chapter discusses the design concepts of the XVM UNICHANNEL 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 XVM UNICHANNEL Spooler

In the XVM UNICHANNEL system, spooling is achieved by using the dual processing capability of the system. The two processors, XVM 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 (XVM), 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 XVM UNICHANNEL spooler.

5.3 SPOOLER DESIGN

The XVM UNICHANNEL 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 XVM. 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, BIT-MAP, 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]
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.

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

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).

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 XVM 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 XVM 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.

SPOL15 (SPOOL) determines if the spooler is running. If so, SPOL15 asks "END?". If the reply is yes, a terminate spooling directive is sent to PIREX and the SPOOLER is disabled. If the SPOOLER is not running, SPOL15 asks on which RK drive the user wishes to begin spooling. Spooling may be done on any RK unit that has a cartridge that has been initialized with a SPOOLER area by the SPLGEN program. If the cartridge has a SPOOLER area and if there is room in the PDP-11 local memory, the SPOOLER is read from the system disk (DP0, DK, or RK0) and transferred to local PDP-11 memory and started. Note that the questions "RK UNIT#" and "BEGIN?" must be answered in this process.

All questions have default replies displayed. These replies may be selected by entering a carriage return. The options on YES/NO questions are "Y" or "N". The default valve for the RK unit is the unit upon which spooling was done previously (or unit 0 if PIREX was just loaded).

Example:

```
XVM/DOS Vnxnnn
$SPOOL
SPOOL XVM Vnxnnn
    RK UNIT # [1] 1
    BEGIN? (Y) Y
    SPOOLING ENABLED
XVM/DOS Vnxnnn
$SPOOL
SPOOL XVM Vnxnnn
    END?  (Y)  Y
    SPOOLING DISABLED
XVM/DOS Vnxnnn
$
```

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 15-18, 40

```
SPOL11.141      MAC11 XVM V1A000  PAGE 3
ASSEMBLY PARAMETERS
1                       .SBTTL  ASSEMBLY PARAMETERS
2               ;
3               ;  CONDITIONAL ASSEMBLY, $LP, $CD, $PL, FOR LINEPRINTER
4               ; FOR LP USE 40000
5               ; FOR PL USE 10000
6               ; FOR CD USE 20000
7       040000  $LP=40000
8               ;$PL=10000
9               ;
10              ;     CARD READER, AND XY PLOTTER, RESPECTIVELY
11      000000  DEVSPP=0
12      000000  DEVCNT=0
13                      .IFDF   $LP
14      000001  DEVCNT=DEVCNT+1
15      040000  DEVSPP=DEVSPP!$LP
16              .ENDC
17              .IFDF   $CD
18              DEVCNT=DEVCNT+1
19              DEVSPP=DEVSPP!$CD
20              .ENDC
21              .IFDF   $PL
22              DEVCNT=DEVCNT+1
23              DEVSPP=DEVSPP!$PL
24              .ENDC
25              ;
26              ;
27              ;
28              ;
29                      .SBTTL  SYMBOLIC EQUATES
```

Figure 5-1
UNICHANNEL Spooler Components

```
SPOL11.141      MAC11 XVM V1A000  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.

```
1                               .SBTTL  SPOOLER DISPATCHER
2           000000 SPBEG=.
3  000000 000576                .WORD   SPEND-SPBEG/2           ;SIZE OF SPOOLER (BR=127)
4  000002 000146                .WORD   SPST                    ;STARTING BYTE OFFSET (BR=128)
5  000004                       .BLOCK  8.+EAESTK*6-2   ;(BR=128)
6  000140 000142                .WORD   DUM
7  000142 000000 DUM:           .WORD   0
8  000144 000000                .WORD   0
9  000146 016700 SPST:          MOV     SPST-2,R0       ;GET TCP ADDRESS IN R0
           177772
10 00152 012767                 MOV     #100000,SPST-4  ;FAKE 11'S REQ. TO PREVENT GETTING KILLED
           100000
           177762
11                                                      ;THIS IS TO PREVENT STACK BLOW UP THRO'
12                                                      ;CTL 'C'S FROM PDP-15
13 00160 013767                 MOV     @#CTLCT,SDCTSV  ;SAVE CURRENT CTL 'C' COUNT FOR LATER CLEANUP
           001066
           001740
14 00166 005767                 TST     ONCEFL          ;HAS THIS CODE ALREADY BEEN DONE?
           005046
15 00172 001026                 BNE     20$             ;YES -- DON'T DO IT AGAIN
16 00174 012737                 MOV     #DEVSPP,@#DEVSPL        ;SET UP DEVICE SPOOLED WORD
           040000
           001064
17 00202                        ADR     SPBEG,R1        ;INITALIZE ADDRESSES (PIC CODE)
   00202 010701                 MOV     PC,R1
   00204 062701                 ADD     #SPBEG-.,R1
           177574
18 00210                        ADR     ADRTBL,R2
   00210 010702                 MOV     PC,R2
   00212 062702                 ADD     #ADRTBL-.,R2
           004746
19 00216 012703                 MOV     #-ADTCNT,R3
           000031
20 00222 060122 10$:            ADD     R1,(R2)+                ;CALCULATE ADDRESSES
21 00224 005303                 DEC     R3
22 00226 001375                 BNE     10$             ;LOOP UNTIL ALL FINISHED
23 00230 016702                 MOV     BUFLAD,R2               ;SET UP BUFFERS
           004762
24 00234 060122 15$:            ADD     R1,(R2)+        ;SET UP POINTERS GOING BACKWARDS THRU Q
25 00236 060112                 ADD     R1,@R2
26 00240 014202                 MOV     -(R2),R2
27 00242 020267                 CMP     R2,BUFLAD       ;HEAD OF BUFFER?
           004750
28 00246 001372                 BNE     15$             ;NO -- TRY AGAIN
29 00250            20$:
30 00250 122760                 CMPB    #SPCOD+200,TCODE(R0)    ;SPOOLER REQUEST?
           000207
           000002
31 00256 001432                 BEQ     Z1$
32 00260 010701                 MOV     PC,R1
33 00262 062701                 ADD     #DISP1-.,R1     ; GET DEVICE DISPATCH TABLE IN R1
           000124
```

Figure 5-1 (Cont.)
UNICHANNEL Spooler Components

```
SPOL11,141      MAC11 XVM V1A000  PAGE 6+
SPOOLER DISPATCHER
34 00266 005002         CLR     R2
35                      ;
36 00270 122760         CMPB    #LPCOD,TCODE(R0)        ;LP REQUEST?
         000004
         000002
37 00276 001431         BEQ     Z2$
38                      ;
39 00300 005722         TST     (R2)+
40 00302 122760         CMPB    #CDCOD,TCODE(R0)        ;NO. CD REQUEST?
         000005
         000002
41 00310 001424         BEQ     Z2$
42                      ;
43 00312 005722         TST     (R2)+
44 00314 122760         CMPB    #PLCOD,TCODE(R0)        ;NO. PL REQUEST?
         000006
         000002
45 00322 001417         BEQ     Z2$
46                      ;
47                      ;UNRECOGNISED TASK REQUEST REPORT.
48                      ;
49 00324                ERROR:
50 00324 013701         MOV     @#DEVST,R1
         001056
51 00330 062701         ADD     #SPCOD*3*2+4,R1
         000056
52 00334 112711         MOVB    #IOPS77,(R1)
         000077
53 00340                CALL    DEQREQ
   00340 004767         JSR     PC,DEQREQ
         000664
54                      ;
55 00344 010701 71$:    MOV     PC,R1           ;SPOOLER REQUEST ;GET SPOOLER DISPTACH
56 00346 062701         ADD     #DISP0-.,R1     ;TABLE IN #3
         000022
57 00352 116002         MOVB    FCODE(R0),R2    ;GET FUN. CODE
         000006
58 00356 042702         BIC     #177740,R2
         177740
59 00362 060102 Z2$:    ADD     R1,R2           ;ADD FUN. CODE TO R1
60 00364 061201         ADD     (R2),R1         ;BUILD DISPATCH JUMP X
61 00366 000111         JMP     (R1)            ;BRANCH TO APPROPRIATE ROUTINE
62                      ;
63                      ;SPOOLER DIRECTIVE DISPATCH TABLE
64 00370 000024 DISP0:  BEGIN   =DISP0          ;BEGIN: CODE=0
65 00372 177734         ERROR   =DISP0          ;ERROR: CODE=2
66 00374 000434         END     =DISP0          ;END: CODE=4
67 00376 177734         ERROR   =DISP0          ;ERROR: CODE=6
68 00400 177734         ERROR   =DISP0          ;ERROR: CODE=10
69 00402 177734         ERROR   =DISP0          ;ERROR: CODE=12
70 00404 177734         ERROR   =DISP0          ;ERROR: CODE=14
71                      ;
72                      ;DEVICE REQUEST =DISPATCH TABLE
73 00406 003722 DISP1:  LPCALL  =DISP1          ;LP: LINE PRINTER
74 00410 004462         CDCALL  =DISP1          ;CD: CARD READER
75 00412 004430         PLCALL  =DISP1          ;PL: XY PLOTTER
76                      ;
```

Figure 5-1 (Cont.)
UNICHANNEL Spooler Components

```
SPOL11.141      MAC11 XVM V1A000  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 000414 010701 BEGIN1: MOV     PC,R1           ;GET ADDRESS OF DEVINT IN R1
10 00416 062701         ADD     #DEVINT-.,R1
         002346
11 00422 013702         MOV     @#SEND11,R2
         001002
12 00426 010162         MOV     R1,SPCOD+2(R2)          ;SET SEND11 ADDRESS IN PIREX
         000016
13 00432 016067         MOV     14(R0),TCBDSA+TCBDIS
         000014
         006274
14                      ; INITIALIZE ALL SWITCHES
15 00440 012767         MOV     #1,CBTPTR       ;START BIT MAP SEARCH
         000001
         001440
16 00446 016701         MOV     ASPLFU,R1       ;##139##SETUP TASK CODE STACK FOR FINDBK
         004542
17 00452 010167         MOV     R1,TCDINI       ;##139##WHEN MORE THAN ONE GUY FINDS OUT
         001432
18 00456 010167         MOV     R1,TCDPNT       ;##139##THERE ARE NO BLOCKS
         001430
19                      ;SET  CONTROL REGS.
20 00462 010701         MOV     PC,R1           ;GET ADD. OF DUM IN R1
21 00464 062701         ADD     #DUM-.,R1
         177456
22 00470                PUSH    R1              ;SAVE ON STACK
   00470 010146         MOV     R1,-(SP)
23 00472                POP     -(R1)           ; SET SPOOLER CONTROL REG.!!
   00472 012641         MOV     (SP)+,-(R1)
24                      ;SETUP BUFFER POOL
25                      ;INITIALIZE RK TCB POINTERS
26 00474 016701         MOV     RKCAD,R1                ;GET RKTCBP ADD. IN R1
         004460
27 00500 010702         MOV     PC,R2           ;GET TCBR01 ADD. IN R2
28 00502 062702         ADD     #TCBST-.,R2
         006012
29 00506 012703         MOV     #TCBCT,R3       ;SETUP TCBCT TCB'S
         000005
30 00512 010221 2$:     MOV     R2,(R1)+        ;SET TCBRK1 POINTER
31 00514 062702         ADD     #30,R2          ;BUMP R2 TO TCBRK2
         000030
32 00520 005303         DEC     R3
33 00522 001373         BNE     2$
34                      ;INITIALIZE BITMAP
35 00524                PUSH    NBK(R0)         ;GET SIZE OF SPOOLER AREA NUMBER
   00524 016046         MOV     NBK(R0),-(SP)



SPOL11.141      MAC11 XVM V1A000  PAGE 7+
BEGIN DIRECTIVE
         000012
36 00530 006216         ASR     (SP)            ;COMPUTE SIZE OF BIT MAP
37 00532 006216         ASR     (SP)            ;SIZE=NUMBK/8+2
38 00534 006216         ASR     (SP)
39 00536 042716         BIC     #1,(SP)         ;GET EVEN NUMBER
         000001
40 00542 016767         MOV     BTMPAD,CWDPTR   ;RESET CWDPTR
         004420
         001334
41 00550 016701         MOV     BTMPAD,R1       ;(BR0112, TEMP FIX)
         004412
42 00554 062601         ADD     (SP)+,R1        ;ADD OFFSET TO END
43 00556 010167         MOV     R1,BTMPED       ;SET UP BTMPED
         005460
```

Figure 5-1 (Cont.)
UNICHANNEL Spooler Components

```
SPOL11.141      MAC11 XVM V1A000  PAGE 7+
BEGIN DIRECTIVE
44 00562 016701          MOV     STBKNA,R1               ;GET ADDRESS OF STBKNM+4 IS R1
         004402
45 00566 016021          MOV     SBN(R0),(R1)+   ;SET STARTING LOCK #
         000010
46 00572 016021          MOV     NBK(R0),(R1)+   ;SET NUMBER OF BLOCKS
         000012
47 00576 016037          MOV     UNIT(R0),@#SPUNIT       ;TELL PIREX SPOOLING UNIT (BR=126)
         000016
         001070
48 00604 016067          MOV     UNIT(R0),UNITSP         ;COPY INTO LOCAL MEM, (BR=126)
         000016
         001550
49 00612 000367          SWAB    UNITSP                  ;SET UP FOR TCB USE (BR=126)
         001544
50 00616 012702          MOV     #BTMPSZ,R2      ;GET BIT MAP SIZE IN R2
         000362
51 00622 010103          MOV     R1,R3
52 00624 005023 4$:      CLR     (R3)+
53 00626 005302          DEC     R2
54 00630 001375          BNE     4$
55                       ;INITIALIZE TABLE
56 00632 016701          MOV     TABLAD,R1               ;GET ADDRESS OF TABLE IN R1,R3,R1
         004334
57 00636 010103          MOV     R1,R3
58 00640 012702          MOV     #TABLSZ,R2      ;GET TABLE SIZE IN R2
         000044
59 00644 012723 3$:      MOV     #-1,(R3)+
         177777
60 00650 005302          DEC     R2
61 00652 001374          BNE     3$
62 00654 012711          MOV     #LP1,(R1)       ;SET LP1(DED) IN TABLE
         142061
63 00660 012761          MOV     #CD1,CDTEOF(R1) ;SET CD1 (DED) IN TABLE
         030461
         000014
64 00666 012761          MOV     #LT1,PLTEOF(R1) ;SET PL1 (DED) IN TABLE
         142461
         000030
65                       ;SET SPOOLER SWITCHES
66 00674 005037 1$:      CLR     @#SPOLSW        ;RESET SPOOLER SWITCHES
         001046
67 00700 052737          BIS     #BEGSW,@#SPOLSW ;SET SPOOLER ENABLED AND RUNNING
         170000
         001046
68                       ;
69                       ;ALL SPOOLED TASKS HAVE TO BE INITIALISED, OPERATIONS LIKE SETTING
70                       ;& RESETTING SWITCHES, SETTING UP POINTERS, BUFFERS, STARTING UP
71                       ;TASK ETC. HAVE TO BE DONE AS INDICATED FOR EACH TASK
72                       ;
73                               .IFDF $CD
74                               BIS     #2,@#SPOLSW     ;SET CD ON ONLY IF PRESENT
75                       ;INITIALIZE CD SPOOLER/DESPOOLER TASK
76                               CLRB    CDONCE
77                               MOV     #1000,CDONCE+1
78                               MOV     @#LISTHD,R2     ;GET ADDRESS OF LISTHD IN R2
79                               ADD     #CDCOD*4,R2             ;CLEAR CD DEQUE TASK CODE=5
80                               CALL    EMPTD
81                               MOV     @R1,NBN+TABLE+CDTEOF
82                               MOVB    #1,CDCNTI       ;INITIALIZE CDCNTI
83                               CLRB    CDBMS           ;RESET CDBMS
84                               CLRB    CDBFS
85                               MOV     R1,CDCBIP
86                               CMP     (R1)+,(R1)+
87                               MOV     R1,CDWDIP
88                               ADD     #CDSIZE,CDWDIP  ;BUMP TO NEXT CARD
89                               MOV     R1,R5           ;SAVE BUFFER ADDRESS ON DTA H
90                               CALL    STUPCT          ;SET UP TCB TO READ A CARD
91                               .ENDC
92                               .IFDF $LP
93                       ;INITIALIZE LP SPOOLER/DESPOOLER TASK
94 00706 105067          CLRB    LPONCE
         002643
```

Figure 5-1 (Cont.)
UNICHANNEL Spooler Components

```
SPOL11.141      MAC11 XVM V1A000  PAGE 7+
BEGIN DIRECTIVE
95 00712 012767         MOV     #1000,LPONCE+1
         001000
         002638
96 00720 013700         MOV     @#LISTHD,R2     ;GET ADDRESS OF LISTHD IN R2
         001010
97 00724 062700         ADD     #LPCOD*4,R2     ;CLEAR LP DEQUE: TASK CODE=4
         000020
98 00730                CALL    EMPTD
   00730 004767         JSR     PC,EMPTD
         000026
99                                              ;SET NBN=CBN FOR START UP
100 0734 011167         MOV     @R1,NBN+TABLE
         005320
101 0740 010167         MOV     R1,LPCBCP
         003356
102 0744 022121         CMP     (R1)+,(R1)+
103 0746 010167         MOV     R1,LPWDCP
         003352
104 0752 105067         CLRB    LPBMS
         003343
105                     .ENDC
106                     .IFDF $PL
107             ;INITIALIZE PL SPOOLER/DESPOOLER TASK
108                     CLRB    PLONCE
109                     MOV     #1000,PLONCE+1
110                     MOV     @#LISTHD,R2     ;GET ADDRESS OF LISTHD IN R2
111                     ADD     #PLCOD*4,R2     ;CLEAR PL DEQUE: TASK CODE=6
112                     CALL    EMPTD
113                     MOV     @R1,NBN+TABLE+PLTEOF
114                     MOV     R1,PLCBCP               ;SET PLCBCP
115                     CMP     (R1)+,(R1)+
116                     MOV     R1,PLWDCP       ;SET PLWDCP
117                     CLRB    PLBMS           ;RESET PLBMS
118                     .ENDC
119             ;ALL DONE DEQUE NEXT REQUEST
120 0756                CALL    DEQREQ
    0756 004767         JSR     PC,DEQREQ
         000246
121             ;
122             ;EMPTY TASK DEQUE
123 0762        EMPTD:
124 0762                .INH                    ;INHIBIT INTERRUPTS
    0762                PUSH    @#PS
    0762 013746         MOV     @#PS,-(SP)
         177776
    0766 052737         BIS     #LVL7,@#PS
         000340
         177776
125 0774 012701         MOV     #EMPTY,R1       ;EMPTY TASKS DEQUE
         001026
126 1000 004731         JSR     PC,@(R1)+
127 1002                .ENA                    ;ENABLE INTERRUPTS
    1002                POP     @#PS
    1002 012637         MOV     (SP)+,@#PS
         177776
128 1006                CALL    FINDBK
    1006 004767         JSR     PC,FINDBK
         000426
129 1012 010146         MOV     R1,-(SP)
130 1014                CALL    GETBUF
    1014 004767         JSR     PC,GETBUF
         001344
131 1020                POP     (R1)
    1020 012611         MOV     (SP)+,(R1)
132 1022 000207         RETURN
133                     .SBTTL  END
```

Figure 5-1 (Cont.)
UNICHANNEL Spooler Components

```
SPOL11.141      MAC11 XVM V1A000  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 001024 052737 END:    BIS     #LVL7,@#PS      ;PROTECT ROUTINE (BR=138)
         000340
         177776
8 001032 013701         MOV     @#CLTABL,R1     ;NULL SPOOLER TIMER REQUEST
         001052
9 001036 005067         CLR     SPST=4          ;ENABLE STOP ALL I/O
         177100
10 01042 005037         CLR     @#DEVSPL                ;CLEAR DEVICED SPOOLED SWITCH
         001064
11 01046 005061         CLR     SPCOD*4(R1)
         000034
12 01052 005037         CLR     @#SPOLSW        ;RESET SPOOLER SWITCH
         001046
13 01056 042737         BIC     #LVL7,@#PS      ;UNPROTECT TO ALLOW INTS. TO RUN DOWN (BR=138)
         000340
         177776
14 01064 012705         MOV     #20,R5          ;ALLOW 20 INTERRUPTS (CLOCK OR DEVICE) (BR=138)
         000020
15 01070 000001 1$:     WAIT                    ;WAIT FOR THEM (BR=138)
16 01072 005305         DEC     R5              ;COUNT 20 INTS. (BR=138)
17 01074 001375         BNE     1$              ;BRANCH IF NOT 20 (BR=138)
18 01076 052737         BIS     #LVL7,@#PS      ;INHIBIT INT.
         000340
         177776
19 01104 013701         MOV     @#TEVADD,R1     ;FIND THE ENTRY ADDRESS
         001060
20                      .IFDF   $LP
21 01110 016102         MOV     LPCOD*2(R1),R2  ;FIND TASK ADDRESS
         000010
22 01114                CALL    STPTSK          ;STOP THE TASK
   01114 004767         JSR     PC,STPTSK
         000054
23                      .ENDC
24                      .IFDF   $CD
25                      MOV     CDCOD*2(R1),R2  ;STOP THE CARD READER TASK
26                      CALL    STPTSK          ;STOP  THE TASK
27                      .ENDC
28                      .IFDF   $PL
29                      MOV     PLCOD*2(R1),R2  ;STOP THE PLOTTER TASK
30                      CALL    STPTSK
31                      .ENDC
32 01120 012701         MOV     #RTURN,R1       ;GET RETURN INST. ADD IN R1
         001036
33 01124 013702         MOV     @#SEND11,R2
         001002
34 01130 011162         MOV     (R1),SPCOD*2(R2) ;SHUT OFF SEND11
         000016
```

Figure 5-1 (Cont.)
UNICHANNEL Spooler Components

```
SPOL11.141      MAC11 XVM V1A000  PAGE 9+
END
35 01134 026027         CMP     FCODE(R0),#4    ;SEE IF THIS WAS  "END" OR IOPSUC 20 (BR=138)
         000006
         000004
36 01142 001005         BNE     2$              ;BRANCH IF IOPSUC 20 (BR=138)
37 01144 012701         MOV     #1,R1           ;TELL SPOL15 DONE
         000001
38 01150 012702         MOV     #$END15,R2
         001424
39 01154 004732         JSR     PC,@(R2)+
40 01156         2$:    ADR     TCBDIS,R5       ;SET FA
   01156 010705         MOV     PC,R5
   01160 062705         ADD     #TCBDIS-.,R5
         005542
41 01164                IREQ                    ;SEND REQUEST
   01164 012704         MOV     #100000,R4
         100000
   01170 000004         IOT
   01172     001        .BYTE   1,0
   01173     000
42                      ;
43 01174 005702 STPTSK: TST     R2              ;(GAR-141) IS TASK IN EXISTENCE?
44 01176 001413         BEQ     1$              ;(GAR-141) BRANCH IF NOT.
45 01200 005762         TST     -4(R2)  ;PDP-11 REQUEST?
         177774
46 01204 100010         BPL     1$              ;NO -- IGNORE
47 01206 014203         MOV     -(R2),R3        ;YES -- TEST FOR SPOLLER REQUEST?
48 01210 122713         CMPB    #SPCOD,@R3
         000007
49 01214 001004         BNE     1$
50 01216 005012         CLR     @R2
51 01220 005042         CLR     -(R2)   ;STOP TASK (CLEAR TCB ADR
52 01222 005072         CLR     @-2(R2)         ;STOP DEVICE FROM INTERRUPTING
         177776
53 01226 000207 1$:     RETURN
54                      ;
55                      ;
```

Figure 5-1 (Cont.)
UNICHANNEL Spooler Components

```
SPOL11.141      MAC11 XVM V1A000  PAGE 11
UTILITY ROUTINES
1                       .SBTTL  UTILITY ROUTINES
2                       .IFDF $CD
3                       ;
4                       ;SET UP TCB TO READ A CARD FROM CD
5                       ;CALLING SEQUENCE:      MOV     BUFAD,R5
6                       ;                       CALL    STUPCT
7                       ;                               /
8                       STUPCT: MOV     PC,R1           ;GET ADDRESS OF TCBCD IN R1
9                               ADD     #TCBCD-.,R1
10                              BR      STUCOM          ;ENTER COMMON ROUTNINE
11                              .ENDC
12                              .IFDF $LP
13                      ;
14                      ;SET UP TCB TO WRITE A LINE ON LP
15                      ;CALLING SEQUENCE:      MOV     BUFAD,R5
16                      ;                       CALL    STUPLT
17                      ;
18 01320 010701 STUPLT: MOV     PC,R1           ;GET ADDRESS OF TCBLP IN R1 & R5
19 01322 062701         ADD     #TCBLP-.,R1
         005362
20 01326 000400         BR      STUCOM
21                              .ENDC
22                              .IFDF $PL
23                      ;
24                      ;SET UP TCB TO WRITE A LINE ON PL
25                      ;CALLING SEQUENCE:      MOV     BUFAD,R5
26                      ;                       CALL    STUPPT
27                      ;
28                      STUPPT: MOV     PC,R1           ;GET ADDRESS OF TCBPL IN R1 & R5
29                              ADD     #TCBPL-.,R1
30                              .ENDC
31 01330 010561 STUCOM: MOV     R5,10(R1)
         000010
32 01334 010105         MOV     R1,R5
33 01336 005061         CLR     4(R1)           ;RESET REV
         000004
34 01342                IREQ                    ;SEND
   01342 012704         MOV     #100000,R4
         100000
   01346 000004         IOT
   01350     001        .BYTE   1,0
   01351     000
35 01352 000207         RETURN
36                      ;
37                      ;SET UP DISK TCB TO READ A BLOCK WITH NO INTERRUPTS & RETURN ADDRESS
38                      ;       CALLING SEQUENCE:       ADR     BUFF,R4
39                      ;                               ADR     -.CBN,R3
40                      ;                               ADR     TCBDK-,R2
41                      ;                               CALL    STUPDT
42                      ;
43 01354 010205 STUPDT: MOV     R2,R5           ;SAVE TCBP IN R5
44 01356 022222         CMP     (R2)+,(R2)+     ;BUMP TO REV
45 01360 005022         CLR     (R2)+           ;RESET REV
```

Figure 5-1 (Cont.)
UNICHANNEL Spooler Components

```
SPOL11,141      MAP11 XVM V1A000  PAGE 12*
FIND A FREE BLOCK ON DISK
42 01556 020103          CMP     R1,R3           ;DID WE GET TO BEGINNING WORD
43 01560 101066          BHI     55$             ;YES, NO BITS, SET UP FOR 'ERROR'
44                       ;
45 01562 014102 7$:      MOV     -(R1),R2        ;BACK UP TO GET COPY OF MAP WORD
46                       ; < > < > < > < > < > < > < > < > < > < > < > END OF EDIT #135
47 01564 010167          MOV     R1,CWDPTR       ;SAVE FIND POSITION FOR NEXT TIME CALLED
         000314
48 01570 005202          INC     R2              ;SETS FIRST ZERO BIT IN WORD!!
49 01572 041102 6$:      BIC     (R1),R2         ;CLEAR ALL REST,LEAVING BIT FOR OUR BLOK
50 01574 050211          BIS     R2,(R1)         ;SET BIT IN MAP
51 01576 010267          MOV     R2,CBTPTR       ;REMEMBER BIT FOR NEXT TIME
         000304
52 01602 166701          SUB     BTMPAD,R1       ;BYTE INDEX FOR FOUND BLOCK #
         003360
53 01606 105702          TSTB    R2              ;IS BIT IN LOW HALF OF WORD
54 01610 001001          BNE     8$              ;YUP, NO CHANGE
55 01612 005201          INC     R1              ;IN HIGH HALF, INC BYTE COUNT
56 01614 006301 8$:      ASL     R1              ;NIBBLE (4 BIT) INDEX FOR FIND
57 01616 032702          BIT     #170360,R2      ;IS BIT IN HIGH NIBBLE OF BYTE
         170360
58 01622 001401          BEQ     9$              ;NO, NOCHANGE
59 01624 005201          INC     R1              ;YES, SO INCR NIBBLE COUNT
60 01626 006301 9$:      ASL     R1              ;CRUMB (2 BIT) INDEX FOR FOUND BLOCK
61 01630 032702          BIT     #146314,R2      ; IS BIT IN HIGH CRUMB OF NIBBLE
         146314
62 01634 001401          BEQ     10$             ;NO, NO CHANGE
63 01636 005201          INC     R1              ;YES, SO INCR CRUMB COUNT
64 01640 006301 10$:     ASL     R1              ;NOW HAVE BIT COUNT FOR BLOCK
65 01642 032702          BIT     #125252,R2      ;IS BIT IN HIGH BIT OF CRUMP
         125252
66 01646 001401          BEQ     11$             ;NO, NO CHANGE
67 01650 005201          INC     R1              ;YES, SO ADD ONE
68 01652 066701 11$:     ADD     STBKNM,R1       ;AND FINALLY ADD #OF FIRST MAPPED BLOCK
         003414
69                       ;
70                       ; < > < > < > < > < > < > < > < > < > < > < > END OF EDIT #133
71                       ;
72                       ;THE FOLLOWING PIECE OF CODE CHECKS TO SEE IF THE CURRENT BLOCK TO BE
73                       ;ALLOCATED TO THE CURRENT SPOOLING TASK EQUALS THE CBN OF THIS
74                       ;DESPOOLING TASK;IF THIS IS TRUE, THEN THE 'SPOOLER IS DECLARED FLOODED'
75                       ;THIS HAPPENS ONLY ON A WRAP AROUND(ENTIRE SPOOLER AREA IS TREATED AS A
76                       ;RING BUFFER)WHEN SPOOLING OPERATIONS ARE WAY AHEAD OF DESPOOLING OPERATIONS
77                       ;
78                       ;
79                       ;*****NOTE: AS NEW TASKS ARE ADDED NEW CODE HAS TO BE ADDED*****
80                       ;********** SIMILAR TO THE CODE FOR EXISTING TASKS**************
81                       ;
82 01656 116002          MOVB    2(R0),R2        ;GET CURRENT TASK CODE
         000002
83 01662 122702          CMPB    #LPCOD,R2       ;LP?
         000004
84 01666 001411          BEQ     21$
85 01670 122702          CMPB    #CDCOD+200,R2   ;NO, CD?
         000205
86 01674 001411          BEQ     22$
87 01676 122702          CMPB    #PLCOD,R2       ;NO, PL?
         000006
88 01702 001012          BNE     26$
89 01704 016702          MOV     TABPLC,R2       ;YES
         003268
90 01710 000405          BR      30$
91 01712 016702 21$:     MOV     TABPCB,R2
         003256
92 01716 000402          BR      30$
93                       ;
94 01720 016702 22$:     MOV     TABCDC,R2
         003254
95 01724                 30$:
96 01724 020112          CMP     R1,(R2)
97 01726 001410          BEQ     55
98 01730                 26$:
99 01730                 POP     @#PS            ;DEBUG;UNPROTECT
   01730 012637          MOV     (SP)+,@#PS
         177776
```

Figure 5-1 (Cont.)
UNICHANNEL Spooler Components

```
SPOL11.141      MAC11 XVM V1A000  PAGE 12+
FIND A FREE BLOCK ON DISK
100                                             ;RETURN WITH BLOCK # ON STACK
101 1734 000207         RETURN
102                     ;
103                     ;SORRY NO BLOCK FREE?? SETUP TO HALT CURRENT OPERATION
104                     ; < > < > < > < > < > < > < > < > < > < > START OF EDIT #135
105 1736 016703 55$:    MOV     AFNDBK,R3       ;ADDR 'FINDBK' ; ENTER WHEN NO BLOCK
         003250
106 1742                POP     R2              ;STACK NOW   /ENTER PS/CALL PC/
    1742 012602         MOV     (SP)+,R2
107 1744                PUSH    R3              ;MAKE IT  /ENTER PS/ADDR FINDBK/CALL PC
    1744 010346         MOV     R3,-(SP)
108 1746                PUSH    R2              ;AND HOPE IT FALLS THRU 5 OK
    1746 010246         MOV     R2,-(SP)
109                     ; < > < > < > < > < > < > < > < > < > < > END OF EDIT #135
110 1750 011602 5$:     MOV     (SP),R2         ;DEBUG;GET OLD PS;BR HERE 1 BLK LEFT
111 1752 016616         MOV     2(SP),(SP)      ;DEBUG;SET UP PC
         000002
112 1756 010266         MOV     R2,2(SP)        ;DEBUG;SET PS
         000002
113 1762                PUSH    R0
    1762 010046         MOV     R0,-(SP)
114 1764                PUSH    R1
    1764 010146         MOV     R1,-(SP)
115 1766                PUSH    R2
    1766 010246         MOV     R2,-(SP)
116 1770                PUSH    R3
    1770 010346         MOV     R3,-(SP)
117 1772                PUSH    R4
    1772 010446         MOV     R4,-(SP)
118 1774                PUSH    R5
    1774 010546         MOV     R5,-(SP)
119 1776 013767         MOV     @#CTLCT,SDCTSV  ;SAVE CURRENT COUNT OF PDP-11 CTL 'C'S


SPOL11.141      MAC11 XVM V1A000  PAGE 17
TASK SOFTWARE INTERRUPT DISPATCHER
1                       ;
2                       ;SEND15 IN PIREX TRANSFERS CONTROL TO DEVINT BY A "CALL @SEND11(=COD*2)"
3                       ;IF REQUESTED IN TCB.  THIS IS DONE BY A CODE OF '3' IN BYTE=3
4                       ;OF TCB. SPOOLER SETS THE ADDRESS OF DEVINT IN SEND11 WHEN STARTED
5                       ;
6                       ;
7                       ;
8 002764 022760 DEVINT: CMP     #1,4(R0)        ;GOOD COMPLETION??
         000001
         000004
9 002772 001022         BNE     5$              ;BRANCH IF NO
10 02774 122760         CMPB    #RKCOD+200,TCODE(R0)    ;RK REQ.?
         000202
         000002
11 03002 001417         BEQ     RKINT
12 03004 122760         CMPB    #LPCOD+200,TCODE(R0)    ;LP REQ?
         000204
         000002
13 03012 001406         BEQ     2$
14 03014 122760         CMPB    #CDCOD+200,TCODE(R0)    ;CD REQ?
         000205
         000002
15 03022 001404         BEQ     3$
16 03024 000167         JMP     PLINT
         001772
17                      ;
18                      ;
19 03030 000167 2$:     JMP     LPINT
         000532
20                      ;
21 03034 000167 3$:     JMP     CDINT
         002014
22                      ;
23                      ;
24                      ;
25 03040                5$:
26 03040 000207         RETURN
27                      ;
28                      .SBTTL  RK INTERRUPT SERVICE
```

Figure 5-1 (Cont.)
UNICHANNEL Spooler Components

```
SPOL11,141      MAC11 XVM V1A000  PAGE 19
RK INTERRUPT SERVICE
1                       ;
2                       ;DISK WRITE REQUEST WAS MADE FOR A SPOOLED DEVICE
3                       ;
4 003372 016001 WRITE:  MOV     12(R0),R1       ;GET BUFFER ADDRESS IN R1,
         000012
5 003376 010103         MOV     R1,R3
6 003400 005021         CLR     (R1)+           ;RESET HWDS
7 003402 005011         CLR     (R1)
8 003404                CALL    GIVBUF
  003404 004767         JSR     PC,GIVBUF
         177056
9 003410 122760         CMPB    #PLCOD,DTCODE(R0)       ;REQ MADE FOR PL DEV?
         000006
         000026
10 03416 001450         BEQ     43$
11 03420 122760         CMPB    #CDCOD,DTCODE(R0)       ;REQ MADE FOR CD DEV?
         000005
         000026
12 03426 001436         BEQ     42$
13                      .IFNDF $LP
14                 41$: MOV     @#DEVST,R1
15                      MOVB    #IOPS77,LPSPER(R1)      ;REPORT TASK NOT SUPPORTED
16                      RETURN
17                      .ENDC
18                      .IFDF $LP
19                      ;WRITE REQUEST MADE FOR LP
20 03430 016701 41$:    MOV     LPBMSA,R1       ;RESET LPBMSA
         001566
21 03434 105011         CLRB    (R1)
22 03436 016705         MOV     TABLAD,R5
         00153@
23 03442 016065         MOV     6(R0),LSB(R5)   ;SET LSB IN TABLE
         000006
         000010
24 03450 016703         MOV     LPONAD,R3               ;GET ADD OF LPBMS IN R3
         001506
25 03454 105713         TSTB    (R3)            ;FIRST TIME THROUGH??
26 03456 001341         BNE     DONE
27 03460 105223         INCB    (R3)+           ;YES, SET SW,
28 03462 105213         INCB    (R3)            ;SET LPBMD
29 03464                CALL    GETBUF          ;GET A BUFFER
   03464 004767         JSR     PC,GETBUF
         176674
30 03470                PUSH    #LPCOD          ;SETUP FOR GETPUT SAVE DEV CODE
   03470 012746         MOV     #LPCOD,-(SP)
         000004
31                      .ENDC
32 03474         44$:   PUSH    #READF          ;SAVE DISK FUN,
   03474 012746         MOV     #READF,-(SP)
         000004
33 03500                PUSH    R1              ;SAVE BUFFER ADD
   03500 010146         MOV     R1,-(SP)
34 03502                PUSH    NBN(R5)         ;SAVE BLOCK #
```

Figure 5-1 (Cont.)
UNICHANNEL Spooler Components

```
SPOL11.141      MAC11 XVM V1A000  PAGE 19+
RK INTERRUPT SERVICE
   03502 016546          MOV     NBN(R5),-(SP)
         000006
35 03506                 CALL    GETRKT          ;GET A RK TCB
   03506 004767          JSR     PC,GETRKT
         177056
36 03512                 CALL    GETPUT          ;GET BLOCK
   03512 004767          JSR     PC,GETPUT
         176574
37 03516 062706          ADD     #10,SP          ;CLEAN STACK
         000010
38 03522 000717          BR      DONE            ;CHECK REV & EXIT
39                       .IFNDF $CD
40 03524 013701 42$:     MOV     @#DEVST,R1
         001056
41 03530 112761          MOVB    #IOPS77,CDSPER(R1)      ;REPORT TASK NOT SUPPORTED
         000077
         000043
42 03536 000207          RETURN
43                       .ENDC
44                       .IFDF $CD
45                ;WRITE REQUEST MADE FOR CD
46                42$:   MOV     CDBMSA,R1       ;SET CDBMD
47                       CLRB    (R1)
48                       MOV     TABCDT,R5
49                       MOV     6(R0),LSB(R5)   ;SET LSB IN TABLE
50                       MOV     CDONAD,R4       ;YES, CDONCE=0?
51                       TSTB    (R4)
52                       BNE     DONE
53                       INCB    (R4)            ;SET CDONCE
54                       INCB    1(R4)           ;SET CDBMS
55                       CALL    GETBUF          ;GET A BUFFER
56                       MOV     R1,7(R4)        ;SET CDOBCP
57                       CALL    GETBUF
58                       PUSH    #CDCOD          ;SAVE DEV.CODE FOR GETPUT
59                       BR      44$             ;ISSUE READ REQUEST
60                       .ENDC
61                       .IFNDF $PL
62 03540 013701 43$:     MOV     @#DEVST,R1
         001056
63 03544 112761          MOVB    #IOPS77,PLSPER(R1)      ;REPORT TASK NOT SUPPORTED
         000077
         000051
64 03552 000207          RETURN
65                       .ENDC
66                       .IFDF $PL
67                ;WRITE REQUEST MADE FOR PL
68                43$:   MOV     PLBMSA,R1       ;RESET PLBMSA
69                       CLRB    (R1)
70                       MOV     TABPLA,R5
71                       MOV     6(R0),LSB(R5)   ;SET LSB IN TABLE
72                       MOV     PLONAD,R3               ;GET ADD OF PLBMS IN R3
73                       TSTB    (R3)            ;FIRST TIME THROUGH??
74                       BNE     DONE
```

Figure 5-1 (Cont.)
UNICHANNEL Spooler Components

```
SPOL11.141      MAC11 XVM V1A000  PAGE 21
LP INTERRUPT SERVICE
1                       ;
2                       ;THIS ROUTINE HANDLES COMPLETION OF I/O SOFTWARE INTERRUPT FROM THE
3                       ;DRIVER TASK IN PIREX, IT DESPOOLS THE SPOOLED DATA ONTO THE LP.
4                       ;
5                               .IFDF   $LP
6  003554     000  LPNUMI:  .BYTE   0               ;UNUSED
7  003555     000  LPNNCE:  .BYTE   0               ;ONCE ONLY SW
8  003556     000  LPBMO:   .BYTE   0               ;BLOCK IN MOTION SW
9  003557     000  LPBUFS:  .BYTE   0               ;EMPTY BUFFER COUNT
10 003560 000000  LPCBIP:  0                       ;CURRENT BUFFER POINTER
11 003562 000000  LPWDIP:  0                       ;CURRENT WORD POINTER
12 003564 000000  LPNBIP:  0                       ;NEXT BUFFER POINTER
13                              .ENDC
14                      ;
15                      ;
16                              .IFNDF $LP
17                      LPINT:  MOV     @#DEVST,R1
18                              MOVB    #IOPS77,LPSPER(R1)      ;REPORT TASK NOT SUPPORTED
19                              RETURN
20                              .ENDC
21                              .IFDF $LP
22                      ;
23 003566 016701  LPINT:  MOV     TABCRT,R1
          001434
24 003572 052737          BIS     #LVL5,@#PS      ;INHIBIT DISK INTERRUPTS
          000240
          177776
25 003600 022711          CMP     #-1,(R1)        ;ANY MORE TO DO?
          177777
26 003604 001014          BNE     1$
27 003606 016703  11$:    MOV     LPONAD,R3               ;GET C(LPCBIP) IN R3
          001350
28 003612 105023          CLRB    (R3)+           ;RESET SW,!S
29 003614 105023          CLRB    (R3)+           ;BUMP TO LPBUFS
30 003616 105223          INCB    (R3)+           ;RELEASE BUFF,
31 003620 011303          MOV     (R3),R3
32 003622                 CALL    GIVBUF          ;GIVE BACK BUFFER
   003622 004767          JSR     PC,GIVBUF
          176640
33 003626 042737  2$:     BIC     #1,@#SPOLSW     ;NO, SET LP IDLE SW
          000001
          001046
34 003634 000207  50$:    RETURN
35 003636 005711  1$:     TST     (R1)            ;YES, BLOCK IN MOTION?
36 003640 001042          BNE     3$
37 003642 016704  15$:    MOV     LPCPAD,R4       ;SK=124 YES, GET ADD OF LLPCPADBIP IN R2
          001352
38 003646 011403          MOV     (R4),R3         ;RELEASE BUFFER
39 003650                 CALL    GIVBUF
   003650 004767          JSR     PC,GIVBUF
          176612
40 003654 105244          INCB    -(R4)
41 003656 105764  10$:    TSTB    -1(R4)          ;BLOCK READ IN?
```

Figure 5-1 (Cont.)
UNICHANNEL Spooler Components

```
SPOL11.141      MACY11 XVM V1A000  PAGE 21+
LP INTERRUPT SERVICE
          177777
42 03662 001403          BEQ     4$
43 03664                 CALL    WAITBK
   03664 004767          JSR     PC,WAITBK
          175530
44 03670 000772          BR      10$
45 03672          4$:
46 03672 016701          MOV     TABCRT,R1       ;DEBUG
          001330
47 03676 016767          MOV     TABLE+NBN,TABLE+CBN     ;SET CBN=NBN
          002360
          002352
48 03704 012767          MOV     #4,TABLE+CRP            ;SET CRP
          000004
          002346
49 03712 010703          MOV     PC,R3           ;GET LPOBIP ADD. IN R3
50 03714 062703          ADD     #LPOBIP-.,R3
          177650
51 03720 011304          MOV     (R3),R4         ;GET C(LPOBIP) IN R3 & BUMP TO TWD1
52 03722 016467          MOV     TWD1(R4),TABLE+NBN      ;SET LP.NBN
          000776
          002332
53 03730 016702          MOV     LPCPAD,R2               ;GET ADD. OF LLPCPADBIP IN R2
          001264
54 03734 011322          MOV     (R3),(R2)+      ;SET LPCBIP
55 03736 011312          MOV     (R3),(R2)       ;SET LPWDIP
56 03740 062712          ADD     #4,(R2)
          000004
57 03744 000412          BR      5$              ;SEND WRITE REQ IF NOT SHUT DOWN
58 03746 016702 3$:      MOV     LPCWAD,R2               ;GET ADD OF LPWDIP IN R2
          001234
59 03752 017246          MOV     @(R2),-(SP)
          000000
60 03756 062716          ADD     #5,(SP)         ;EVEN BYTE COUNT
          000005
61 03762 042716          BIC     #177401,(SP)
          177401
62 03766 061611          ADD     (SP),(R1)       ;BUMP CRP
63 03770 062612          ADD     (SP)+,(R2)      ;BUMP LPWDIP
64 03772 032737 5$:      BIT     #40000,@#SPOLSW ;SHUT DOWN?
          040000
          001046
65 04000 001719          BEQ     2$
66 04002 032737          BIT     #1,@#SPOLSW     ;SHUT LP?
          000001
          001046
67 04010 001706          BEQ     2$
68 04012 032737          BIT     #10000,@#SPOLSW ;SHUT DESPOOLER
          010000
          001046
69 04020 001702          BEQ     2$
70 04022 005772          TST     @(R2)           ;FIRST RECORD A .CLOSE?
          000000
71 04026 001024          BNE     13$
72 04030 026161          CMP     -2(R1),4(R1)    ;ANY MORE DATA?
          177776
          000004
73 04036 001003          BNE     14$
74 04040                 CALL    12$             ;NO, SET TABLE ENTRIES
   04040 004767          JSR     PC,12$
          000240
75 04044 000660          BR      11$             ;RESET SWITCHES & EXIT
76 04046 016702 14$:     MOV     LPONAD,R2       ;DEBUG;SK=124 GET LPBUFS ADRHESS
          001110
77 04052 062702          ADD     #2,R2           ;DEBUG;SK=124
          000002
78 04056 122712          CMPB    #1,(R2)         ;DEBUG;SK=124 ONE FREE BUFFER?
          000001
79 04062 001267          BNE     15$             ;SK=124
80 04064 105762          TSTB    -1(R2)          ;DEBUG;SK=124 YES, BLOCK IN MOTION?
          177777
81 04070 001264          BNE     15$             ;SK=124
```

Figure 5-1 (Cont.)
UNICHANNEL Spooler Components

```
SPOL11.141      MAC11 XVM V1A000  PAGE 21+
LP INTERRUPT SERVICE
82 04072                CALL    9$              ;SK=124 NO, GET NEXT BLOCK
   04072 004767         JSR     PC,9$
         000146
83 04076 000661         BR      15$             ;SK=124 RELEASE BUFFER & WAIT FOR BLOCK TO COME IIN
84                ;
85                ;
86 04100 011205 13$:    MOV     @R2,R5          ;NO, SAVE BUFF ADD ON STACK
87 04102                CALL    STUPLT          ;SET UP TCB TO UNTI A LINE
   04102 004767         JSR     PC,STUPLT
         175212
88 04106 016701         MOV     TABCRT,R1
         001114
89 04112 011204         MOV     (R2),R4         ;CHECK FOR BUFFER EMPTY
90 04114 017246         MOV     @(R2),-(SP)     ;GET BYTE COUNT
         000000
91 04120 062716         ADD     #5,(SP)         ;EVEN BYTE COUNT
         000005
92 04124 042716         BIC     #177401,(SP)
         177401
93 04130 062604         ADD     (SP)+,R4        ;BUMP R4 TO POINT TO PT WORD OF NEXT
94 04132 010702         MOV     PC,R2           ;NO, GET ADD OF LPBUFS IN R2
95 04134 062702         ADD     #LPBUFS-.,R2
         177423
96 04140 005714         TST     (R4)            ;LAST RECORD?
97 04142 001417         BEQ     6$
98 04144 022714         CMP     #-1,(R4)
         177777
99 04150 001414         BEQ     6$
100 4152 122712         CMPB    #1,(R2)         ;LPBUFS=1
         000001
101 4156 001226         BNE     50$
102 4160 105742         TSTB    -(R2)           ;YES, BLOCK IN NEXT?
103 4162 001224         BNE     50$
104 4164 026161         CMP     -2(R1),4(R1)    ;NO, MORE TO DOE (CBN=LSB)
         177776
         000004
105 4172 001620         BEQ     50$
106 4174                CALL    9$              ;SK=124 GET NEXT BLOCK
    4174 004767         JSR     PC,9$
         000044
107 4200 000615         BR      50$             ;SK=124 EXIT
108               ;
109               ;
110               ;BUFFER EMPTY; TEST IF MORE BLOCK TO DO?

111 4202 026161 6$:     CMP     -2(R1),4(R1)    ;MORE TO DO? (CBN=LSB)
         177776
         000004
112 4210 001412         BEQ     7$
113 4212 005011         CLR     (R1)            ;SK=124 SET CRP=0
114 4214 122712         CMPB    #1,(R2)         ;LPBUFS=1?
         000001
115 4220 001004         BNE     8$
116 4222 105742         TSTB    -(R2)           ;BLOCK IN TRANSIT?
117 4224 001002         BNE     8$              ;SK=124
118 4226                CALL    9$              ;SK=124 GET NEXT BLOCK
    4226 004767         JSR     PC,9$
         000012
```

Figure 5-1 (Cont.)
UNICHANNEL Spooler Components

```
SPOL11.141      MAC11 XVM V1A000  PAGE 21+
LP INTERRUPT SERVICE
119 4232 000167 8$:     JMP     50$             ;SK=125
         177376
120             ;NO MORE BLOCKS TO DO

121 4236        7$:     CALL    12$             ;SET TABLE ENTRIES
    4236 004767         JSR     PC,12$
         000042
122 4242 000773         BR      8$
123             ;
124             ;
125             ;GET NEXT BLOCK

126 4244        9$:     PUSH    R1
    4244 010146         MOV     R1,-(SP)
127 4246                PUSH    R2
    4246 010246         MOV     R2,-(SP)
128 4250                CALL    GETBUF          ;YES, GET BUFFER & READ NEXT BLOCK
    4250 004767         JSR     PC,GETBUF
         176110
129 4254 010104         MOV     R1,R4           ;SAVE BUFAD IN R4
130 4256                POP     R2
    4256 012602         MOV     (SP)+,R2
131 4260                POP     R1
    4260 012601         MOV     (SP)+,R1
132 4262 010467         MOV     R4,LPOBIP               ;SET LPOBIP
         177276
133 4266 105219         INCB    (R2)            ;SET LPBMS SW
134 4270 012703         MOV     #LPCOD,R3               ;GET DEV,CODE IN R3, FOR GETBLK
         000004
135 4274 010102         MOV     R1,R2           ;GET LP,CRP ADD, IN R2
136 4276                CALL    GETBLK          ;GET BLOCK FROM DISK
    4276 004767         JSR     PC,GETBLK
         000004
137 4302 000207         RETURN                  ;SK=124
138

139 4304        12$:
140 4304 012711         MOV     #-1,@R1         ;SET CRP=-1
         177777
141 4310 012761         MOV     #-1,6(R1)       ;SET LFB=-1
         177777
         000006
142 4316 000207         RETURN
143
144                     .ENDC
145                     .SBTTL  LP CALL SERVICE
```

Figure 5-1 (Cont.)
UNICHANNEL Spooler Components

```
SPOL11.141      MAC11 XVM V1A000  PAGE 22
LP CALL SERVICE
1                       ;
2                       ;THIS ROUTINE SERVICES CALLS TO OUTPUT DATA ONTO THE LP. IT SPOOLS THE
3                       ;DATA SENT BY THE CALLER ONTO THE DISK.
4                       ;
5                               .IFDF   $LP
6  004320    000   LPNUMC: .BYTE   0               ;UNUSED
7  004321    000   LPBMS:  .BYTE   0               ;BLOCK IN MOTION SW
8  004322 000000  LPCBCP: 0                       ;CURRENT BUFFER POINTER
9  004324 000000  LPWDCP: 0                       ;CURRENT WORD POINTER
10 004326 000000  LPQBCP: 0                       ;NEXT BUFF POINTER(DUMMY)
11                              .ENDC
12                      ;
13                      ;
14                              .IFNDF $LP
15                      LPCALL: MOV     @#DEVST,R1
16                              MOVB    #477,LPSPER(R1)
17                              CALL    DEQREQ
18                              .ENDC
19                              .IFDF $LP
20 04330 024141 LPCALL: CMP     -(R1),-(R1)     ;POINT R1 TO LPWDCP
21 04332 032737         BIT     #20000,@#SPOLSW ;SHUT SPOOLER?
         020000
         001046
22 04340 001435         BEQ     10$
23 04342                PUSH    R1              ;SAVE R1.       NO
   04342 010146         MOV     R1,-(SP)
24 04344 011101         MOV     (R1),R1         ;GET CONTENTS OF LPWDCP IN R1,R4
25 04346 010104         MOV     R1,R4
26 04350 016003         MOV     10(R0),R3       ;GET CALLER BUF. ADD. IN R3
         000010
27 04354 006303         ASL     R3              ;RELOCATE ADD.
28 04356 063703         ADD     @#MEMSIZ,R3
         000042
29 04362 111302         MOVB    (R3),R2         ;GET BYTE COUNT FROM BUFFER IN R2
30 04364 062702         ADD     #5,R2           ;ADD HWD BYTE COUNT + EVEN BYTE COUNT
         000005
31 04370 042702         BIC     #177401,R2
         177401
32 04374 060201         ADD     R2,R1           ;BUMP LPWDCP BY THE SIZE OF NEXT RECD.
33 04376 011605         MOV     (SP),R5         ;GET LPWDCP ADD. IN R4
34 04400                PUSH    -(R5)           ;POINT TO LPCBCP & SAVE CONT. OF LPCBCP ON STACK
   04400 014546         MOV     -(R5),-(SP)
35 04402 006202         ASR     R2              ;CONVERT TO WORD COUNT
36 04404 162601         SUB     (SP)+,R1                ;COMPUTE SPACE REM.
37 04406 022701         CMP     #770,R1         ;SPACE LEFT?
         000770
38 04412 002402         BLT     4$
39 04414                CALL    COPBUF          ;COPY CALLER BUFFER
   04414 004767         JSR     PC,COPBUF
         000356
40 04420                POP     R4              ;TEMP SAVE R1 IN R2
   04420 012604         MOV     (SP)+,R4
41 04422                CALL    6$              ;CHECK FOR .CLOSE



SPOL11.141      MAC11 XVM V1A000  PAGE 22+
LP CALL SERVICE
   04422 004767         JSR     PC,6$
         000270
42 04426 000406         BR      8$              ;NO
43                      ;
44 04430 012760 10$:    MOV     #-600,4(R0)     ;SPOOLER SHUT DOWN. REPORT
         177200
         000004
45 04436                PUSH    R1              ;DUMMY
   04436 010146         MOV     R1,-(SP)
46 04440 000167         JMP     DEQRQ
         174576
47                      ;LAST RECORD WAS NOT A .CLOSE
48 04444 005741 8$:     TST     -(R1)           ;POINT R1 LPCBCP
```

Figure 5-1 (Cont.)
UNICHANNEL Spooler Components

```
SPOL11.141      MAC11 XVM V1A000  PAGE 22+
LP CALL SERVICE
49 04446 010102          MOV     R1,R2           ;SAVE IN R2
50 04450 005721          TST     (R1)+           ;BUMP R1 LPWDCP
51 04452 011101          MOV     (R1),R1         ;GET CURRENT WORD ADD. IN R1
52 04454 161201          SUB     (R2),R1         ;GET REMAINNING # OF WORDS
53 04456 022701          CMP     #770,R1         ;SPACE LEFT?
         000770
54 04462 003034          BGT     2$
55 04464 010701  9$:     MOV     PC,R1           ;GET ADD. OF LPWDCP IN R1
56 04466 062701          ADD     #LPWDCP-.,R1
         177634
57 04472 005071          CLR     @(R1)           ;NO. PUT BUFFER ON DISK
         000000
58 04476                 CALL    FINDBK          ;GET DISK BLOCK #
   04476 004767          JSR     PC,FINDBK
         174736
59 04502                 PUSH    R1              ;SAVE BLOCK # ON STACK
   04502 010146          MOV     R1,-(SP)
60 04504 016702          MOV     LPCBCP,R2               ;GET C(LPCBIP) IN R2
         177612
61 04510 011662          MOV     (SP),TWD1(R2)   ;SAVE BLOCK # IN TWD1
         000776
62 04514 012703          MOV     #LPCOD,R3               ;GET LP.DEV CODE IN R3
         000004
63 04520 016701          MOV     LPBMSA,R1       ;SET LPBMSA
         000476
64 04524 105211          INCB    (R1)
65 04526                 CALL    PUTBLK          ;PUT BUFF. ON DISK
   04526 004767          JSR     PC,PUTBLK
         000376
66 04532 016704          MOV     LPCBAD,R4               ;GET ADD. OF LLPCBADBCP IN R3&R4
         000446
67 04536         3$:     CALL    GETBUF          ;GET A NEW BUF
   04536 004767          JSR     PC,GETBUF
         175622
68 04542 010124          MOV     R1,(R4)+        ;SET LPCBCP=BUFAD
69 04544                 POP     (R1)            ;SET BLOCK # IN HWD0 OF NEW BUFF.
   04544 012611          MOV     (SP)+,(R1)
70 04546 062701          ADD     #4,R1           ;BUMP R2 TO WORD 2 OF BUF
         000004
71 04552 010114          MOV     R1,(R4)         ;SET LPWDCP
72 04554         2$:     CALL    DEQREQ          ;DEQUE REQUEST & EXIT IN WAIT STATE
   04554 004767          JSR     PC,DEQREQ
         174450
73 04560         4$:     POP     R1              ;RESTORE ADD. OF CURRENT WORD IN R1
   04560 012601          MOV     (SP)+,R1
74 04562                 PUSH    R3              ;SAVE R3,R2
   04562 010346          MOV     R3,-(SP)
75 04564                 PUSH    R2
   04564 010246          MOV     R2,-(SP)
76 04566 005071          CLR     @(R1)           ;SET BUFF. END SW
         000000
77 04572                 CALL    FINDBK          ;GET DISK BLOCK #
   04572 004767          JSR     PC,FINDBK
         174642
78 04576                 PUSH    R1              ;SAVE BLOCK #
   04576 010146          MOV     R1,-(SP)
79 04600                 CALL    GETBUF          ;GET A BUFF.
   04600 004767          JSR     PC,GETBUF
         175560
80 04604 011611          MOV     (SP),(R1)       ;SET BLOCK # IN HWD0 OF NEW BUFF.
81 04606 016704          MOV     LPCBAD,R4               ;GET ADD. OF LLPCBADBCP IN R4
         000372
82 04612                 PUSH    (R4)
   04612 011446          MOV     (R4),-(SP)
83 04614                 PUSH    (R4)            ;SAVE CONT. OF LPCBCP
   04614 011446          MOV     (R4),-(SP)
84 04616 062716          ADD     #TWD1,(SP)      ;BUMP TO TWD1
         000776
85 04622 016636          MOV     4(SP),@(SP)+    ;SET LINK IN OLD BUFF.
         000004
86 04626 010124          MOV     R1,(R4)+        ;SET LPCBCP & BUMP TO LPWDCP
87 04630 062701          ADD     #4,R1           ;POINT TO WORD 2 IN BUFF.
         000004
```

Figure 5-1 (Cont.)
UNICHANNEL Spooler Components

```
SPOL11.141      MAC11 XVM V1A000  PAGE 22+
LP CALL SERVICE
88 04634                PUSH    R4              ;SAVE LPWDCP ADD. ON STACK
   04634 010446         MOV     R4,-(SP)
89 04636 010114         MOV     R1,(R4)         ;SET LPWDCP
90 04640 010104         MOV     R1,R4           ;GET CONT. OF LPWDCP
91 04642 016602         MOV     6(SP),R2        ;RESTORE R3,R2
          000006
92 04646 016603         MOV     10(SP),R3
          000010
93 04652                CALL    COPBUF          ;COPY CALLER BUFFER
   04652 004767         JSR     PC,COPBUF
          000120
94 04656                POP     R4              ;SAVE LPWDCP ADD. IN R4
   04656 012604         MOV     (SP)+,R4
95 04660                POP     R2              ;CONT. OF LPCBCP ON STACK TOP???
   04660 012602         MOV     (SP)+,R2
96 04662 012703         MOV     #LPCOD,R3       ;GET DEV.CODE IN R3. FOR PUTBLK
          000004
97 04666 062706         ADD     #6,SP           ;CLEAN STACK
          000006
98 04672                PUSH    R4              ;SAVE R5
   04672 010446         MOV     R4,-(SP)
99 04674 016701         MOV     LPBMSA,R1       ;SET LPBMSA
          000322
100 4700 105211         INCB    (R1)
101 4702                CALL    PUTBLK          ;PUT BUFF. ON DISK
    4702 004767         JSR     PC,PUTBLK
          000222
102 4706                POP     R4              ;TEMP SAVE R1
    4706 012604         MOV     (SP)+,R4
103 4710                CALL    6$              ;CHECK FOR .CLOSE
    4710 004767         JSR     PC,6$
          000002
104 4714 000717         BR      2$
105 4716 010401 6$:     MOV     R4,R1           ;SAVE R4
106 4720 011104         MOV     (R1),R4         ;GET C(LPWDCP) IN R4
107 4722 022764         CMP     #LPCLOS,-2(R4)  ;FF+CR??
          000414
          177776
108 4730 001021         BNE     7$
109 4732 010104         MOV     R1,R4           ;RESTORE R4
110 4734                ADR     TABLE+LFB,R2    ;GET LP.LFB ADD. IN R2
    4734 010702         MOV     PC,R2
    4736 062702         ADD     #TABLE+LFB-.,R2
          001330
111 4742 016701         MOV     LPCBAD,R1
          000236
112 4746                PUSH    (R2)            ;SAVE OLD LFB
    4746 011246         MOV     (R2),-(SP)
113 4750 017112         MOV     @(R1),(R2)      ;SET LFB IN TABLE
          000000
114 4754 011101         MOV     (R1),R1
115 4756                POP     2(R1)           ;SET OLD LFB IN BUFFER
    4756 012661         MOV     (SP)+,2(R1)
          000002
116 4762 012761         MOV     #-1,TWD0(R1)    ;SET EOF CODE IN BUFFER
          177777
          000774
117 4770 005726         TST     (SP)+   ;RETURN TO 9 (NOT SUB RETURN)
118 4772 000634         BR      9$

119 4774 000207 7$:     RETURN
120                     ;
121                     .ENDC
122 4776            COPBUF:
123 4776 026737         CMP     SDCTSV,@#CTLCT  ;DEBUG
          175124
          001066
124 5004 001005         BNE     1$
125 5006 012324         MOV     (R3)+,(R4)+     ;COPY CALLER BUFFER
126 5010 005302         DEC     R2
127 5012 001371         BNE     COPBUF
128 5014 010476         MOV     R4,@2(SP)
          000002
```

Figure 5-1 (Cont.)
UNICHANNEL Spooler Components

```
SPOL11.141      MAC11 XVM V1A000  PAGE 23
PL INTERRUPT SERVICE
1                       ;
2                       ;THIS ROUTINE HANDLES COMPLETION OF I/O SOFTWARE INTERRUPT FROM THE
3                       ;DRIVER TASK IN PIREX. IT DESPOOLS THE SPOOLED DATA ONTO THE XY PLOTTER.
4                       ;
5                               .IFDF   SPL
6                       PLNUMI: .BYTE   0               ;UNUSED
7                       PLNNCE: .BYTE   0               ;ONCE ONLY SW
8                       PLBMD:  .BYTE   0               ;BLOCK IN MOTION SW
9                       PLBUFS: .BYTE   0               ;EMPTY BUFFER COUNT
10                      PLCBIP: 0                       ;CURRENT BUFFER POINTER
11                      PLWDIP: 0                       ;CURRENT WORD POINTER
12                      PLNBIP: 0                       ;NEXT BUFFER POINTER
13                              .ENDC
14                      ;
15                      ;
16                              .IFNDF SPL
17 05022 013701 001058  PLINT:  MOV     @#DEVST,R1
18 05026 112761 000077  000051  MOVB    #IOPS77,PLSPER(R1)      ;REPORT TASK NOT SUPPORTED
19 05034 000207                 RETURN
20                              .ENDC
21                              .IFDF SPL
22                      ;
23                      PLINT:  MOV     TABPDT,R1
24                              BIS     #LVL5,@#PS      ;INHIBIT DISK INT.
25                              CMP     #-1,(R1)        ;ANY MORE TO DO?
26                              BNE     1$
27                      11$:    MOV     PLONAD,R3               ;GET C(PLCBIP) IN R3
28                              CLRB    (R3)+           ;RESET SW.'S
29                              CLRB    (R3)+           ;BUMP TO PLBUFS
30                              INCB    (R3)+           ;RELEASE BUFF.
31                              MOV     (R3),R3
32                              CALL    GIVBUF          ;GIVE BACK BUFFER
33                      2$:     BIC     #4,@#SPOLSW     ;NO, SET PL IDLE SW
34                      50$:    RETURN
35                      1$:     TST     (R1)            ;YES, BLOCK IN MOTION?
36                              BNE     3$
37                      15$:    MOV     PLCIAD,R4       ;SK=124 YES, GET ADD OF PLCBIP IN R2
38                              MOV     (R4),R3         ;RELEASE BUFFER
39                              CALL    GIVBUF
40                              INCB    -(R4)
41                      10$:    TSTB    -1(R4)          ;BLOCK READ IN?
42                              BEQ     4$
43                              CALL    WAITBK          ;NO
44                              BR      10$
45                      4$:     MOV     TABPDT,R2
46                              MOV     2(R2),-2(R2)    ;SET CBN=NBN
47                              MOV     #4,(R2)         ;SET CRP
48                              MOV     PLOIAD,R3               ;GET PLOBIP ADD. IN R3
49                              MOV     (R3),R4         ;GET C(PLOBIP) IN R3 & BUMP TO TWO1
50                              MOV     TWO1(R4),2(R2)  ;SET PL.NBN
```

Figure 5-1 (Cont.)
UNICHANNEL Spooler Components

```
SPOL11.141      MAC11 XVM V1A000  PAGE 23+
PL INTERRUPT SERVICE
51                      MOV     R2,R1           ;SAVE PL.CRP ADD. IN R1
52                      MOV     PLC1AD,R2               ;GET ADD. OF PLCBIP IN R2
53                      MOV     (R3),(R2)+      ;SET PLCBIP
54                      MOV     (R3),(R2)       ;SET PLWDIP
55                      ADD     #4,(R2)
56                      BR      5$              ;SEND WRITE REQ IF NOT SHUT DOWN
57              3$:     MOV     PLWDAD,R2               ;GET ADD OF PLWDIP IN R2
58                      MOV     @(R2),-(SP)
59                      ADD     #5,(SP)         ;EVEN BYTE COUNT
60                      BIC     #177401,(SP)
61                      ADD     (SP),(R1)       ;BUMP CRP
62                      ADD     (SP)+,(R2)      ;BUMP LPWDIP
63              5$:     BIT     #40000,@#SPOLSW ;SHUT DOWN?
64                      BEQ     2$
65                      BIT     #4,@#SPOLSW     ;SHUT PL?
66                      BEQ     2$
67                      BIT     #10000,@#SPOLSW ;SHUT DESPOOLER
68                      BEQ     2$
69                      TST     @(R2)           ;LAST RECORS?
70                      BNE     13$
71                      CMP     -2(R1),4(R1)    ;YES, ANY MORE DATA?
72                      BNE     14$
73                      CALL    12$             ;NO, SET TABLE ENTRIES
74                      BR      11$
75              14$:    MOV     PLONAD,R2       ;SK=124 GET PLBUFS ADDRESS
76                      ADD     #2,R2           ;SK=124
77                      CMPB    #1,(R2)         ;SK=124 ONE FREE BUFFER?
78                      BNE     15$             ;SK=124
79                      TSTB    -1(R2)          ;SK=124 YES, BLOCK IN MOTION
80                      BNE     15$             ;SK=124
81                      CALL    9$              ;SK=124 NO, GET NEXT BLOCK
82                      BR      15$             ;SK=124 WAIT FOR BLOCK TO COME IN
83
84              13$:    MOV     @R2,R5          ;NO, SAVE BUFF ADD ON STACK
85                      CALL    STUPPT          ;SET UP TCB TO UNTI A LINE
86                      MOV     PC,R1           ;GET PL.CRP ADD. IN R1
87                      ADD     #TABLE+PLTEOF-.+4,R1
88                      MOV     (R2),R4         ;CHECK FOR BUFFER EMPTY
89                      MOV     @(R2),-(SP)     ;GET BYTE COUNT
90                      ADD     #5,(SP)         ;EVEN BYTE COUNT
91                      BIC     #177401,(SP)
92                      ADD     (SP)+,R4        ;BUMP R4 TO POINT TO PT WORD OF NEXT
93                      MOV     PC,R2           ;NO, GET ADD OF PLBUFS IN R2
94                      ADD     #PLBUFS-.,R2
95                      TST     (R4)            ;LAST RECORD?
96                      BEQ     6$
97                      CMP     #-1,(R4)
98                      BEQ     6$
99                      CMPB    #1,(R2)         ;PLBUFS=1
100                     BNE     50$
101                     TSTB    -(R2)           ;YES, BLOCK IN NEXT?
102                     BNE     50$
103                     CMP     -2(R1),4(R1)    ;NO, MORE TO DOE (CBN=LSB)
```

Figure 5-1 (Cont.)
UNICHANNEL Spooler Components

```
SPOL11.141      MAC11 XVM V1A000  PAGE 28
ADDRESS TABLE
1                               .SBTTL  ADDRESS TABLE
2                       ;
3  005160               ADRTBL:
4  005160 002556 RKCADR:        .WORD   RKTCBP
5                               .IFDF   $LP
6  005162 003555 LPONAD:        .WORD   LPONCE
7                               .ENDC
8  005164 006304 TABPLA:        .WORD   TABLE+PLTEOF
9                               .IFDF   $PL
10                      PLONAD:      .WORD   PLONCE
11                              .ENDC
12 05166 005276 BTMPAD:         .WORD   BTMPST
13 05170 005272 STBKNA:         .WORD   STBKNM
14 05172 006254 TABLAD:         .WORD   TABLE
15 05174 006256 TABCBI:         .WORD   TABLE+CBN
16 05176 006306 TABPLC:         .WORD   TABLE+PLTEOF+CBN
17 05200 006272 TABCDC:         .WORD   TABLE+CDTEOF+CBN
18 05202 006364 TCBK1A:         .WORD   TCBDK1
19                              .IFDF   $CD
20                      CDCPAD:      .WORD   CDCBIP
21                      CDCBAD:      .WORD   CDCBCP
22                              .ENDC
23                              .IFDF   $LP
24 05204 004322 LPCBAD:         .WORD   LPCBCP
25 05206 003562 LPCWAD:         .WORD   LPWDIP
26                              .ENDC
27                              .IFDF   $PL
28                      PLCBAD:      .WORD   PLCBCP
29                      PLWDAD:      .WORD   PLWDIP
30                              .ENDC
31 05210 006412 TCBK3A:         .WORD   TCBDK3
32 05212 001440 AFNDBK:         .WORD   FINDBK
33 05214 002124 ASPLFU:         .WORD   SPLFUL          ;##139##
34 05216 006742 BUFLAD:         .WORD   BUFLHD
35                              .IFDF   $LP
36 05220                LPCPAD:
37 05220 003560 LPCZAD:         .WORD   LPCBIP
38 05222 004321 LPBMSA:         .WORD   LPBMS
39                              .ENDC
40 05224 006270 TABCDT:         .WORD   TABLE+CDTEOF
41 05226 006260 TABCRT:         .WORD   TABLE+CRP
42 05230 006310 TABPDT:         .WORD   TABLE+PLTEOF+CRP
43                              .IFDF   $PL
44                      PLCIAD:      .WORD   PLCBIP
45                      PLOIAD:      .WORD   PLOBIP
46                      PLBMSA:      .WORD   PLBMS
47                              .ENDC
48                              .IFDF   $CD
49                      CDBMSA:      .WORD   CDBMS
50                      CDINTA:      .WORD   CDINT
51                              .ENDC
52 05232 006274 TABDCT:         .WORD   TABLE+CDTEOF+CRP
53 05234 005070 CDCAAD:         .WORD   CDCALL
```

Figure 5-1 (Cont.)
UNICHANNEL Spooler Components

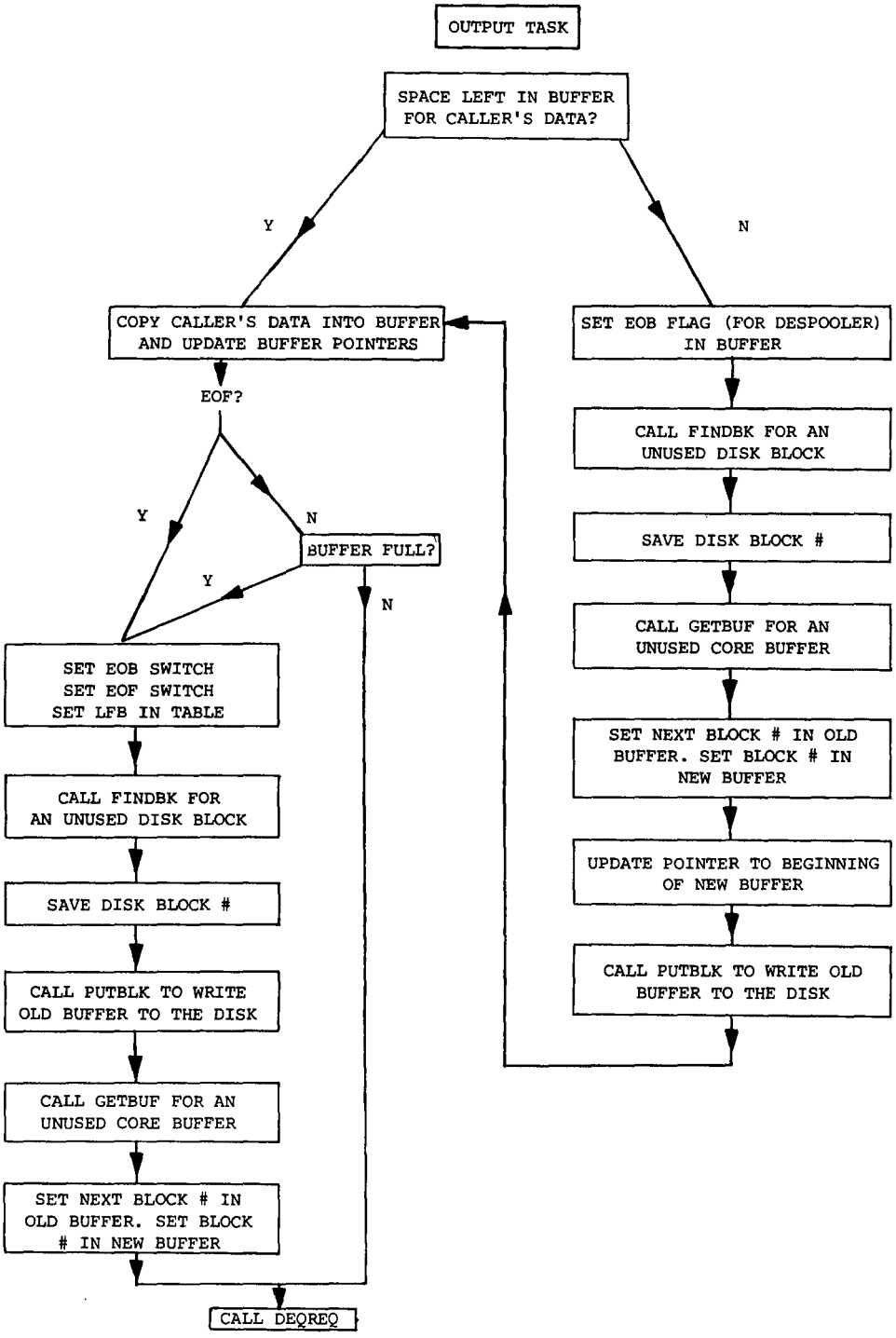


Figure 5-2
Task Call Service Routine

Set the SPOOLER task control registers	lines 19-23
Setup the disk TCB pointer table	lines 25-33
Setup and initialize BITMAP	lines 35-54
Initialize and setup TABLE	lines 55-64
Set the SPOOLER switches	lines 65-67

LINE PRINTER OPERATIONS:

Initialize the LP call service routine switches and pointers	lines 94-95, 101-104
Clear all pending LP task requests in PIREX get a free block on disk, get a buffer.	lines 96-98
Set the NBN entry in TABLE.	line 100
Process the next SPOOLER request	line 120

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 XVM 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 22, line 20
Check if spooling operations are disabled and, if disabled, exit	lines 26, 22
Point to the current word	lines 26, 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 Figures 5-1 and 5-3).

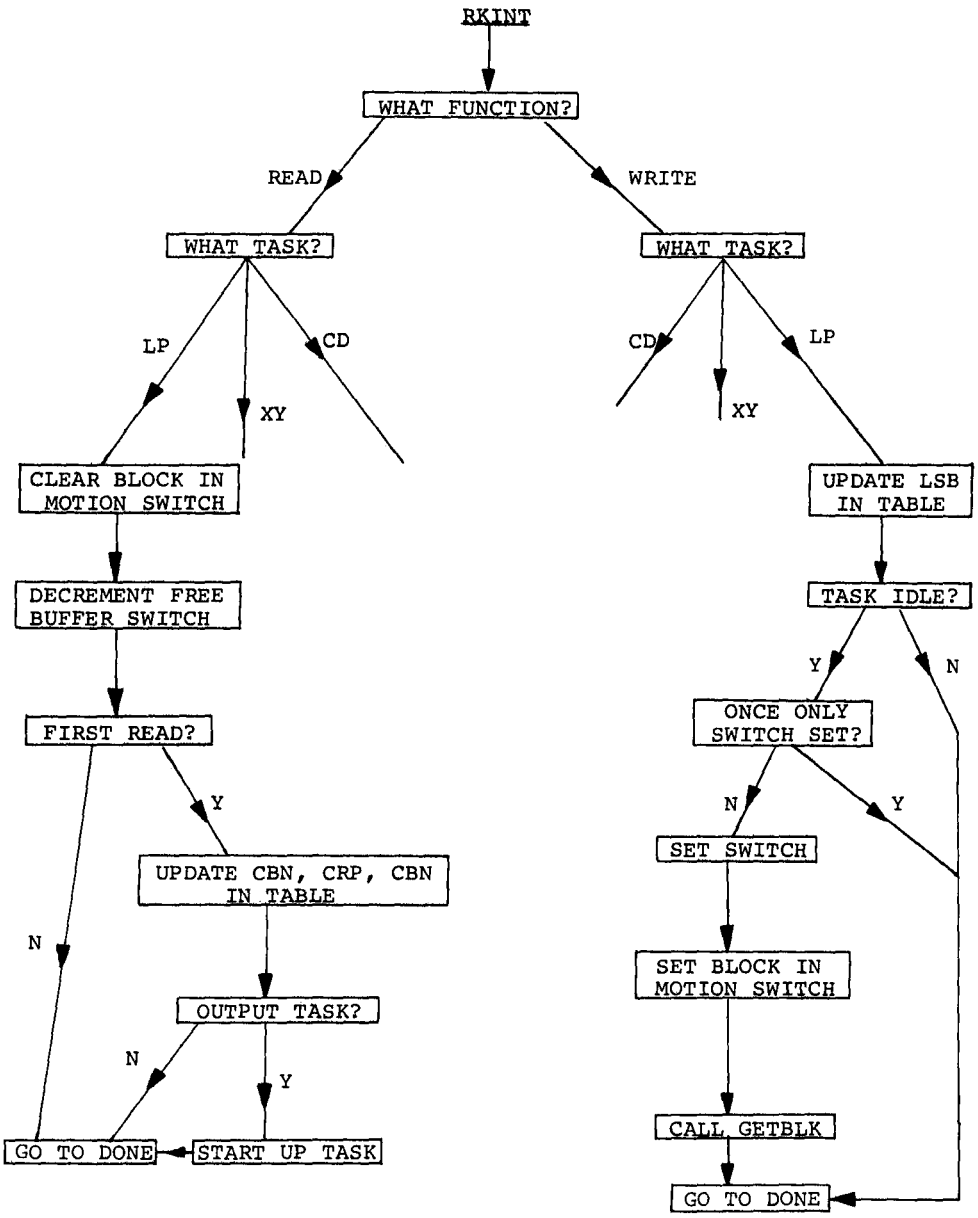


Figure 5-3
Device Interrupt Servicing Logic (For LP)

WRITE PROCESSOR:

Reset the "write block in motion" switch	page 19, 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 43-45
Yes, set LFB = 1	line 46
Reset the "read block in motion" switch	line 48
Decrement the LP free buffer count	line 49
LPONCE = 1, first time through, start up LP	lines 50-53
Set Current Block Number (CBN) in TABLE	line 66
Set the current despooling buffer pointer	lines 67-68
Set the current despooling word pointer	lines 69-70
Set the Next Block Number (NBN) in TABLE	lines 71-72
Set Current Record Pointer (CRP) in TABLE	line 73
Set LPONCE = 2	line 54
LP despooling is not shut down; send the LP write request	lines 55-58
Set the LP busy switch	line 60
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 21, 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 47
Set CRP in TABLE	line 48
Set NBN in TABLE	lines 49-52
Set the current despooling buffer and word pointer	lines 53-56
Shut down? Shut LP? Shut despooler?	lines 64-69
Current record in buffer is a .CLOSE record, check if more blocks to do	lines 70-72
There are no more blocks reset TABLE entries, switches and then exit	lines 74, 77, 121-123
One free buffer and no block in motion	lines 76-81
Get next block	line 82
Release buffer and wait to come in	lines 83, 37-44
The first record is not a .CLOSE, send an LP write request	lines 86-87
Point to the first word of the next record	lines 89-93
There are more records left and one free buffer	lines 96-101
There is no read block in motion and more blocks to do	lines 102-105
Get next block	lines 106, 126-137
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):

Protect shutdown routine	page 9, line 7
Clear any pending SPOOLER wakeup requests	line 8
Allow devices to run down	lines 13-18
Shut down LP task	lines 20-23
Turn off SEND11	lines 32-34
Test if shut down due to disk error	lines 35-36
If "END" shutdown, tell "SPOL15" that it has occurred	lines 37-39
Disconnect SPOOLER	lines 40-41

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.)

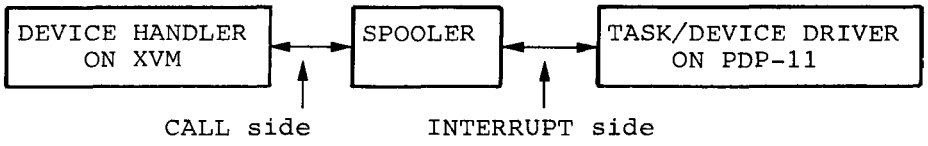


Figure 6-1
SPOOLER Schematic

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 XVM. 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 indication 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.

---

[1] See Section 5.4.7.

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

Get the address of the LP TCB	page 11, lines 18-19
Go to setup common	line 20
Set the buffer address specified in the TCB	line 31
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 36-38, 73

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 17, 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 28, line 6
TABPCB	line 15
LPCBCP	line 24
LPWDIP	line 25
LPCBIP	line 37
LPBMS	line 38

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 13-16

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

$XX = ZZZZOO

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 12, lines 83-84, 91-92

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.

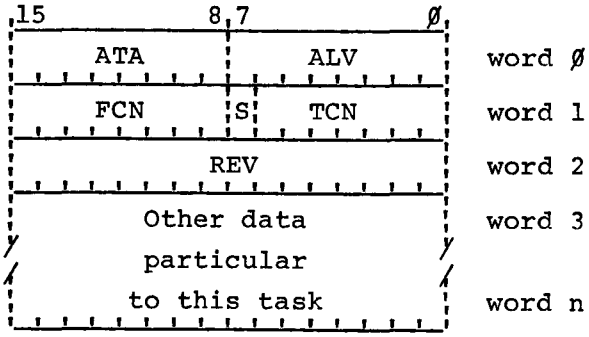
APPENDIX A
ABBREVIATIONS

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
XVM/DOS	XVM 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
XVM/RSX	XVM 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	PDP-11 Front End Processor and Interlace to XVM

APPENDIX B

CURRENTLY IMPLEMENTED TCBs

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



ATA XVM API interrupt vector address

ALV XVM 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 XVM 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.

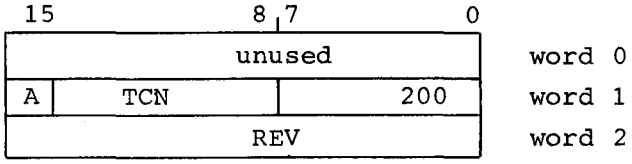
Returned REV value:

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 XVM 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 XVM 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 XVM requests on the task request list will be aborted regardless of the setting of this bit.

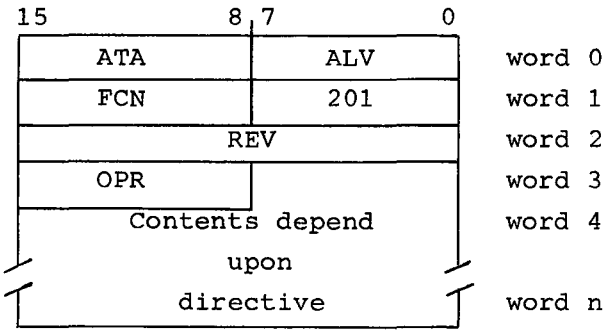
All requests which are aborted via this request will never complete; the request event variables (REVs) of such requests will never be set to a non-zero value. A permanent task which is stopped via this request will be placed in the wait state; a temporary task will be placed in the stopped state.

Returned REV values:

1	Successful completion
-600	Task to be stopped is not connected to PIREX. Only applicable when TCN $\neq$ 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.

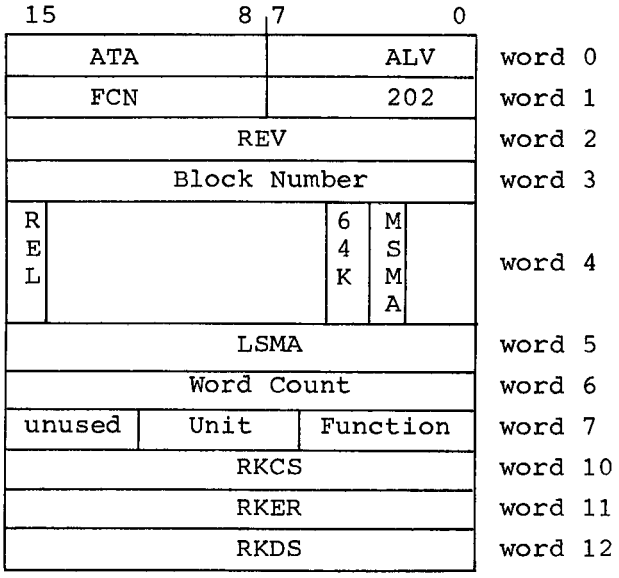
Returned REV values:

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:

Currently Implemented TCBs



ATA	Usually 047_8
ALV	Usually 000
REV	Set to 1 upon completion regardless of errors.
Block Number	Disk block number to transfer.
REL	0 if request comes from XVM 1 if request comes from PDP-11
64K[1]	When 1 causes an additional 64K words to be transferred.
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.

[1] A zero in the word count field (word 6) causes a 64K word transfer. The "64K" field (word 4) is used in conjunction with the word count to specify transfers greater than 64K words. Thus to transfer 65K words, the user would set the "64K" bit and place a minus -1024_{10} in the word count field.

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-HRKD-B-D) for a description of their meaning.

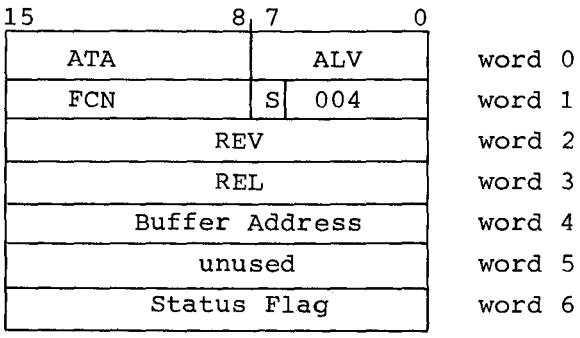
If the request originates from the PDP-11, LSMA is the 16-bit PDP-11 byte address at which the transfer is to begin. If the request originates from the XVM, MSMA and LSMA together are the 17-bit XVM word address at which the transfer is to begin. Upon completion of the transfer, REV is always set to 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)

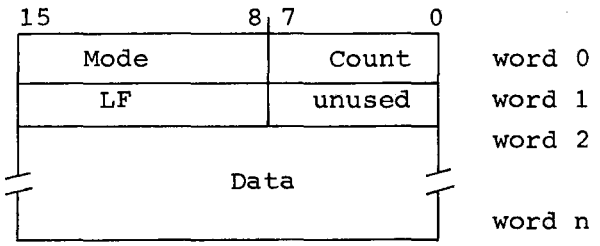
The task control block format is as follows:



Currently Implemented TCBs

ATA	Usually 056_8
ALV	Usually 002
S	Usually 0 (indicating spooled operation)
REL	0 if request originates from XVM 1 if request originates from PDP-11
Buffer Address	PDP-11 byte address, if request is from PDP-11 XVM word address, if request is from XVM
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.

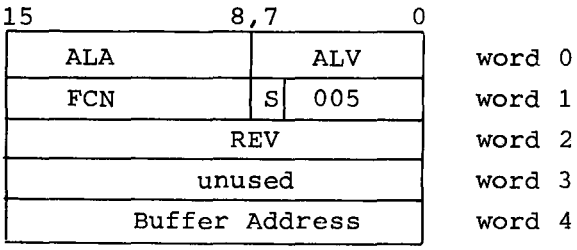
Line printer errors are not reported via returned REV values. The only line printer error which can occur is for the printer to go off line (become not ready). The line printer driver reports this by placing the value 4 in the device error byte of its entry in the DEVST table (see Section 3.6.4 on the 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.

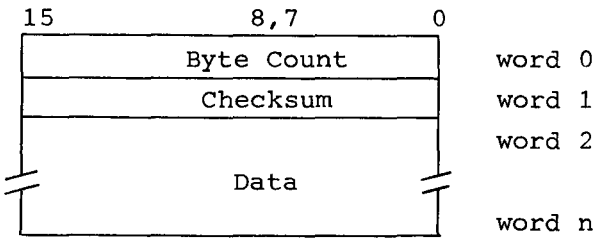
B.5 CARD READER DRIVER TASK (CD)

The task control block format is as follows:



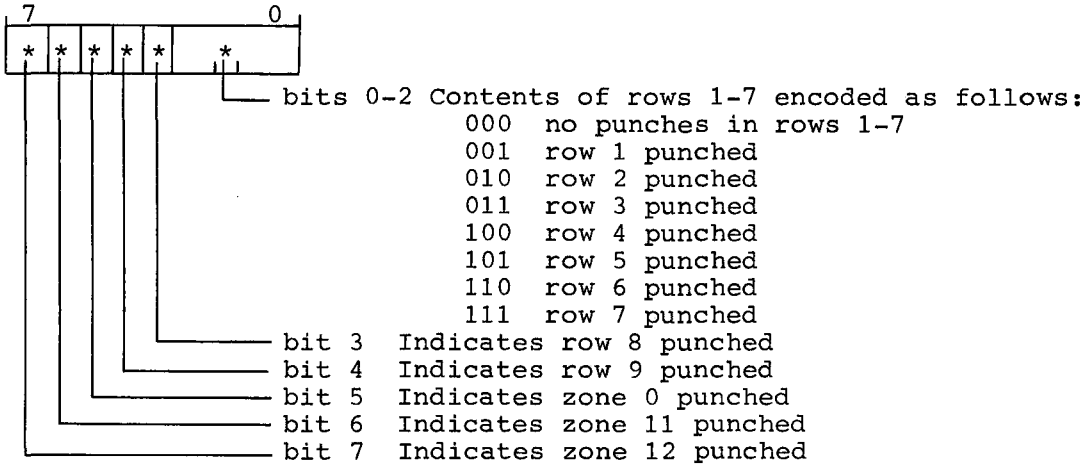
ATA	Usually 055_8
ALV	Usually 001
S	Usually 0 (Indicating spooled operation)
Buffer Address	PDP-11 byte address, if request is from PDP-11 XVM word address, if request is from XVM

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



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:



NOTE

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

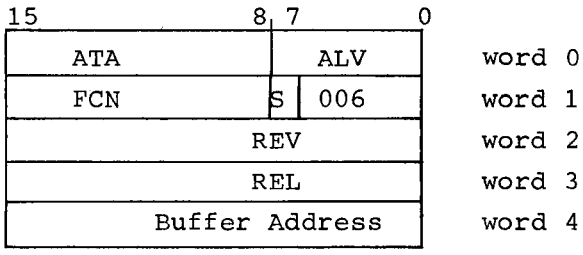
Any errors that occur are not reported by returned REV values. Instead the IOPSUC numeric error code is placed in the device error byte of the card reader's entry in the DEVST table (see Section 3.6.4, Error Status Report Directive). When the error condition is remedied, the driver clears the device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

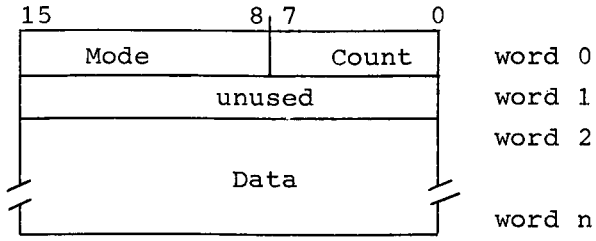
B.6 PLOTTER DRIVER TASK (XY)

The task control block format is as follows:



ATA	Usually 065_8
ALV	Usually 003
S	Usually 0 (indicating spooled operation)
REL	000000 If request is from XVM If request is from PDP-11
Buffer Address	PDP-11 byte address, if request is from PDP-11. XVM word address, if request is from XVM.

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



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:

Currently Implemented TCBs

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

Standard format 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	ATL 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.
601	System Memory Map Invalid This indicates that the memory map used by CONNECT/DISCONNECT is invalid. PIREX should be rebooted before any CONNECT/DISCONNECT attempt.
602	TCB Out of Range This indicates that the TCB address is not within the 28K word addressing range of the UNICHANNEL.

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 an XVM or PDP-11 request. If zero, the request came from the XVM, 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 determing 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 XVM and PDP-11. The shared memory is ordinary 18-bit XVM 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' is 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 XVM 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 continguous 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 XVM program).

A pointer to the TCB (called a TCBP) is then passed to the PDP-11 via the LIOR instruction in the XVM 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 XVM or the IREQ macro in the PDP-11 when initiating a request to PIREX.

INDEX

Please cut along this line.

READER'S COMMENTS

NOTE: This form is for document comments only. Problems with software should be reported on a Software Problem Report (SPR) form.

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 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

digital

digital equipment corporation

PRINTED IN U.S.A.