PRELIMINARY

PROM/RT-11
System User's Guide

Order NO: AA-D848A-TC

October 1978

This document is intended to be used by system-level programmers responsible for writing application programs that use PROM and EPROM memory for the application. Further it provides instructions for operators to program (blast) PROM or EPROM chips in a production operation. This manual and the RT-11 operating system documents provide the necessary information to write programs for and program PROM and EPROM memories.

PRELIMINARY

SUPERSESSION/UPDATE INFORMATION: This is a new manual.

OPERATING SYSTEM AND VERSION: RT-11 V03B

First printing, October 1978

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.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-10
DECCOMM	DECSYSTEM-20	TYPESET-11

PRELIMINARY

CONTENTS

CONTENTS (CONT.)

CONTENTS (CONT.)

CONTENTS (CONT.)

CONTENTS (CONT.)

PREFACE

HOW TO USE THIS MANUAL

This document provides the system level programmer the necessary information to write application programs that run in PROM or EPROM and RAM memory and to program (blast) the PROM or EPROM memory. It also provides the information for an operator to program PROM or EPROM memories in a production environment. It also provides the information required to install the programmer hardware and software. This manual is not intended to teach RT-11 programming.

Chapter 1 describes the system hardware and software components.

Chapter 2 summarizes the types of RAM, PROM, and EPROM memory components and explains how to select memory components for your application.

Chapter 3 provides information required for the system level programmer to write MACRO programs, FORTRAN programs, and programs containing both MACRO and FORTRAN instructions that run in PROM or EPROM and RAM memory.

Chapter 4 explains installation, acceptance test, and startup procedures for the PROM/RT-11 system.

Chapter 5 explains operating procedures, including system start-up in foreground and background, and all the commands and procedures used to program and check PROM or EPROM memory chips.

Chapter 6 provides information required to troubleshoot the hardware and software components.

Appendixes provide a summary of information presented in the manual, including error messages, and detailed examples of PROM and EPROM programming sessions.

DOCUMENTATION CONVENTIONS

The following list describes the symbolic conventions used throughout this manual. Familiarize yourself with these conventions before you continue reading the manual.

1. Examples consist of actual computer output wherever possible. In these examples, user input is underlined where necessary to differentiate it from computer output.

2. Unless the manual indicates otherwise, terminate all commands and command strings and with a carriage return. The symbol <RETURN> represents a carriage return, <LF> represents a line feed, <SP> a space, <ESC> an ESCAPE or ALTMODE, and <TAB> a tab.

3. To produce several characters in system commands, you must type a combination of keys concurrently. For example, hold down the CTRL key and type C at the same time to produce the <CTRL/C> character. Key combinations such as this one are documented as <CTRL/C>, <CTRL/O>, <SHIFT/N>, etc.

4. In descriptions of command syntax, capital letters represent the command name that you must type. Lower case letters represent a variable for which you must supply a value.

5. The sample terminal dialogue provided in this document contains version numbers where they would normally appear. While the version numbers given include xx's in those fields that may vary from installation to installation, the exact contents of these fields are not of interest, as long as appropriate digits appear in the area indicated. The same is true for the VIRTUAL MEMORY USED and DYNAMIC MEMORY AVAILABLE messages printed by any of the system programs and for the FREE BLOCKS messages included in device directories.

6. Square brackets enclose optional arguments and characters but are not part of the syntax.

7. Parentheses are required where shown, following the standard MACRO and FORTRAN syntax conventions.

8. An ellipsis indicates the possible repetition of the argument that precedes the ellipsis.

9. A bit with a value of one (1) is referred to as being "on" or "set". A bit with a value of zero (0) is referred to as being "cleared" or "reset". For inverted bits, addresses, or data, a bit that is set has a value of 0 and a cleared bit has a value of 1.

RELATED DOCUMENTS

The documents listed below contain information relating specifically to development techniques for PROM based microcomputer application

programs. Specific areas of interest within each document are also listed.

Applicable documents are as follows:

Manual	Area of Interest
PDP-11 MACRO-11 Programmer's Guide	Description of the program section (.PSECT) directives.
RT-11/RSTS/E FORTRAN IV User's Guide	Description of program sectioning and assembly language interface.
RT-11 System User's Guide	Description of the LINK utility, LINK maps, and .PSECT handling.
RT-11 Advanced Programmer's Guide	Description of the System Library (SYSLIB)
Microcomputer Handbook Series-Memories And Peripherals (EB-1514)	Presents information on the serial-line-unit interfaces available to interface the PROM programmer to the development system. It also presents information about the memory modules available for the LSI-11.
Microcomputer Handbook Series-Microcomputer Processors (EB-1515)	Presents information about the LSI-11 processors.
PDP-11/03 Configurator Manual	Presents information about microcomputer configurations.

See the RT-11 Documentation Directory for other information concerning related documents in the RT-11 library.

CHAPTER 1

INTRODUCTION

The PROM/RT-11 system consists of a Universal Programmer and a software utility to control the PROM programmer. The utility is supplied as an addition to the RT-11 Operating System (version V03B or later). The computer used in the development system is the PDP-11 (11V03, 11T03L, 11/34, etc.).

The PROM programming hardware is connected to the development system over a serial line that is not controlled by a modem. The serial line and interface must be dedicated to the programming hardware (it cannot be used as a programming hardware interface and a terminal interface).

The PROM/RT-11 system allows you to directly program PROM and EPROM chips for LSI-11 microcomputer applications. The programmed chips are used on LSI-11 memory option boards. The LSI-11 memory options that accept these chips are MRV11-A, MRV11-BA, BDV11, etc.

Application programs that will use PROM and EPROM memory are created by: first writing the program in MACRO or FORTRAN, and then using the program sections (.PSECTS) to divide the program into read-only and read-write segments.

After you code the application program and test it in the development system, a final link is performed to generate the following:

1. A .SAV binary image file of the program that is to be used to program PROM or EPROM chips.

2. A link map of the .SAV file contents, indicating the low and high memory limits for the read-only area of the application program.

The software operates in either the foreground or background of the RT-11 operating system. When the utility is operated in the foreground, program development and PROM or EPROM programming can take place at the same time.

The information in this manual assumes you have a working knowledge of the RT-11 operating system, the MACRO and FORTRAN languages, and the LSI-11 microcomputer. The system description presents only the aspects of the operating system and the hardware that are required to

use this system effectively, If questions arise relating to RT-11 or LSI-11 you should consult the documentation supporting that hardware or software,

1,1 OVERVIEW OF PROM BASED APPLICATION DEVELOPMENT

Development of a PROM based application is a multi step process; that is from the time the application is coded through to the point where it is executed in an LSI-11 computer, The table below defines the individual steps in application development, This table also indicates the areas of this manual where information is given on each development step, This description assumes that the installation procedures given in Chapter 4 have been fully carried out,

Table 1-1
PROM Based Application Development Steps

	Development Step	Coverage Area
1,	Application programs destined for PROMs is initially generated on any PDP-11 family computer have a disk based RT-11 operating system, Coding can be in MACRO-11 assembly language or FORTRAN IV combined with MACRO-11 assembly language	Chapter 3 presents information on PROM based application program development techniques, Appendix A supplies examples of PROM based applications, Further pertinent data is presented in Appendix F,
2,	Application is assembled/compiled	Examples of assembled/compiled application programs are given in Appendix A,
3,	Application program is LINKed using the LINKage editor, LINK maps are obtained as a requirement of the PROM blasting session	Chapter 3 describes procedures for invoking the LINKage editor and obtaining LINK maps, Examples of LINK maps are also presented in Chapter 5 and Appendix A,
4,	The PROM/RT-11 utility is invoked preparatory to blasting the PROMs	Chapter 5 describes the procedures for invoking the PROM/RT-11 utility in either background or foreground modes,
5,	PROMs/EPROMs are blasted to contain	The blasting process is achieved primarily using the PROM/RT-11

	the application program	PROGRAM command described in Chapter 5. Considerations governing selection of PROMs/EPROM types for your application are presented in Chapter 2 and Appendix D. If any problems are encountered during the PROM blasting session, the troubleshooting procedures of Chapter 6 and error message summary of Appendix B will prove helpful. Chapter 5 and Appendix A present examples of a complete PROM blasting session.
6.	The blasted PROMs (now containing the application) are inserted into the LSI-11 memory module option	LSI-11 memory module options (such as the MRV11-AA and MRV11-BA) are well marked to indicate the required locations of individual PROMs.
7.	Memory module option is inserted in LSI-11 backplane	Refer to the PDP-11/03 Configurator Manual.
8.	PROM based application program is executed	Application program now executes on power-up of the LSI-11 computer. Appendix A presents an example of executing a PROM based application.

1.2 SYSTEM DESCRIPTION

The PROM/RT-11 system (see Figure 1-1) allows you as a PDP-11 user to create PROM or EPROM-based microcomputer application programs from a program created on a PDP-11 computer operating under RT-11 operating system. The programs created on a PDP-11 development system are compiled or assembled and linked using the RT-11 LINK utility. The object file output by the linker is then used to program (blast) PROM or EPROM chips that can be used on PROM or EPROM option modules (such as the MRV11-AA, MRV11-BA, BDV11, etc.).

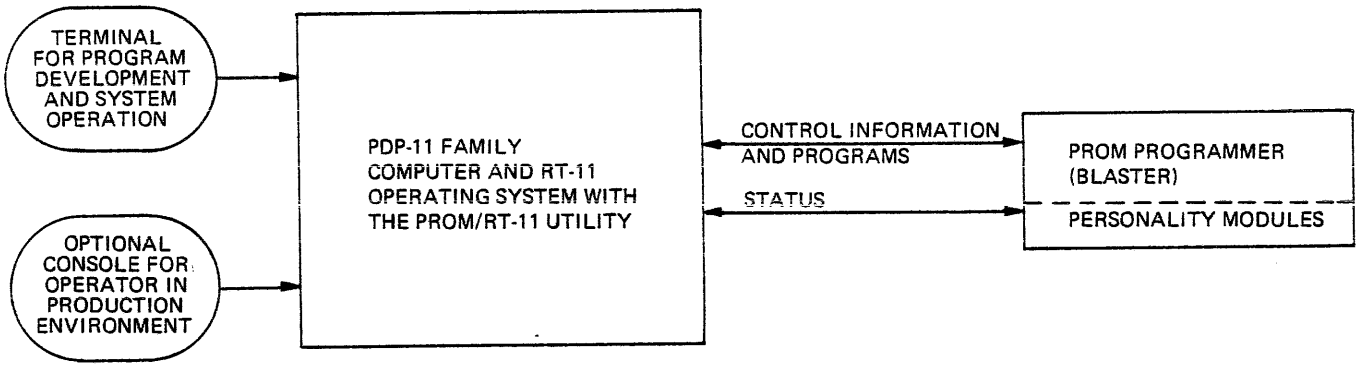


Figure 1-1 PROM/RT-11 System Block Diagram

The PROM/RT-11 system (see Figure 1-1) requires five hardware and software components:

1. A PDP-11 family computer system with the following:

   a. A terminal device to operate the system, write programs, and control the PROM and EPROM chip programming.

   b. A mass storage device. Either the RX01 or RL01 disk must be part of the system, since PROM/RT-11 is distributed on an RX01 floppy disk or RL01 disk cartridge.

2. The RT-11 operating system to provide for program development, assembly and/or compilation, and program linking.

3. The PB-11 system with the following:

   a. The PROM/RT-11 utility to control the PROM programmer.

   b. A PROM/RT-11 programmer to program PROM and EPROM chips. The programmer requires a set of personality cards and mounting sockets for the chips you are going to program (one set is supplied with the programmer).

   c. A serial-line-unit-interface (see Section 1.1.5).

4. A RS232 cable and adapter.

If you plan to blast chips in a production environment (an operator programs chips at a terminal running PROM/RT-11 in the foreground mode), you should have at least two terminals.

1.2.1 PDP-11 Family Computer

Any of the PDP-11 computers manufactured by DIGITAL can be used in a PROM/RT-11 development system. For information about the PDP-11 computers see the PDP-11 Processor Handbook, the Microcomputer Handbook, or the hardware documentation for any of the PDP-11 Processors.

1.2.2 RT-11 Operating System

The RT-11 operating system (version V03B or later versions) must be used as the software development system. For more information about RT-11 see the PDP-11 Software Handbook and the RT-11 software documentation.

1.2.3 Mass Storage Device

A RX01 floppy disk or RL01 cartridge disk manufactured by DIGITAL must be included in the system that is used to develop programs for PROM or EPROM applications and program the chips. For more information about these devices refer to the PDP-11 peripherals Handbook, the Microcomputer Handbook series-Memories and Peripherals Handbook, or the hardware manual for the RX01 or RL01.

1.2.4 PROM/RT-11 Utility

The PROM/RT-11 utility operates under the RT-11 operating system. The utility is invoked as a foreground or background job (see Chapter 5). PROM/RT-11 controls the programming of PROM and EPROM chips via commands from the operator's console.

The following commands are used to control the PROM programmer (blaster):

COPY	Read the contents of a master PROM chip and program one or more chips with its contents.
DIAGNOSE	Run PROM programmer and interface diagnostics to isolate hardware problems.
EXIT	Terminate PROM/RT-11 utility operations and return

control to the RT-11 monitor,

HELP — Print a list of PROM/RT-11 commands at the terminal,

INTERFACE — Change the CSR (Control and Status Register) and vector addresses for the serial-line interface to the PROM programmer hardware,

LIST — Print the contents of a PROM or EPROM chip on the terminal or line printer,

MODIFY — Modify the contents of one or more chips to reflect changes in the application program, This may not always be possible, (see Chapter 5 for details),

PROGRAM — Program (blast) a set of PROM or EPROM chips from a data file on the RT-11 Operating System,

SEQUENTIAL — Alter the operation of the MODIFY, PROGRAM, and VERIFY commands for programming of PROM chips with non-PDP-11 programs or data,

VERIFY — Compare the contents of a PROM or EPROM chip with the contents of a file or a master PROM, and verify they are the same,

The PROM/utility is also used to troubleshoot the hardware used to program PROM and EPROM chips, The DIAGNOSE command is used for this operation (see Chapter 6),

1,2,5 PROM/RT-11 Programmer (Blaster)

The PB-11 programmer is a remotely controlled universal PROM programmer (blaster), It is universal because it uses program Personality cards sets and socket adapters to program a variety of PROM and EPROM chips,

Many PROM types are of a generic family, which require identical specifications for reading and programming, Socket adapters are used to configure pinout, word size, and word length of a memory chip to a card set for that chip,

The PROM programmer is interfaced to the system by a serial-line-unit interface and connected to the interface by a 25-foot (7,6m) cable,

The power supplies and controls are contained within the unit, The operator need only connect the unit to a power source and to the PDP-11 computer (see Chapter 4), install the correct personality card and socket, and turn the unit power on,

1.3 USING RT-11 TO PROGRAM PROM AND EPROM MEMORY CHIPS

The PROM/RT-11 utility operates in the background or foreground of the RT-11 operating system. If PROM/RT-11 operates in the foreground, the RT-11 system supports program development and PROM programming operations at the same time. This also allows an operator to program chips in a production environment. The operator instructions are summarized in Appendix E.

The PROM/RT-11 system can be used to blast all chips if personality modules are available with the following restrictions:

1. The bit width of the PROM or EPROM chips must be either four or eight bits.

2. The number of words in each PROM or EPROM chip must be 32 and be a power of 2. (For example, 256, 512, 1024, 2048 words).

3. The length of the chip must not exceed 4045 words.

1.4 SOFTWARE REQUIREMENTS

The PROM/RT-11 system can be added to any standard disk-based LSI-11 development system and an RT-11 operating system (V03B or later version).

The software can operate under RT-11 with multi-terminal support, but the serial-line-unit interface used to interface the PROM programmer is controlled by routines in the PROM/RT-11 software. The system generation for a RT-11 multiterminal monitor should not include support for the programmer's serial-line-unit interface.

1.5 HARDWARE REQUIREMENTS

The PROM/RT-11 development system must provide at least the following hardware:

1. A 11V03, 11V03L, 11T03L, or 11/34 system.

2. At least 16K of memory for background operation or at least 28K words of memory for foreground development operation.

3. One additional serial-line-unit interface to connect the PROM programmer to the development system computer. Interfaces supported are.

   a. DLV11

   b. DLV11-F

c, DLV11-J (one port)

d, DLV11-E

e, DL11-W

f, DL11-E

The 20 MA current loop devices and multiplexers (such as, DZV11, DZ11, DH11, DJ11) are not supported by the PROM/RT-11 system.

NOTE

Maximum line-speed for the interface is 9600 baud.

4. A cable for the serial-line-unit interface that has a RS232 plug. The PROM/RT-11 programmer hardware is supplied with a 25-foot (7,6m) null-modem cable.

The PROM/RT-11 hardware is supplied with one of the following personality card sets and socket adapters:

1. Card set/socket adapter for the 2708 EPROM chips.

2. Card set/socket adapter for the 2716 and 2732 EPROM chips.

3. Card set/socket adapter for the 82S131 and 82S129 PROM chips.

4. Card set/socket adapter for the 82S181 and 82S191 PROM chips.

1.6 SUPPORT

The PROM/RT-11 software is Category C; no Software Performance Report servicing is provided. All software problems will be reported using the problem report form printed in this manual. The form allows you to submit information to DIGITAL, but DIGITAL does not guarantee that any action is expressed or implied as a result of the receipt of the form.

1.7 TROUBLESHOOTING

The PROM/RT-11 system has built-in hardware diagnostic capability. You can use the DIAGNOSE command (see Chapter 6) to determine what hardware component is malfunctioning when a problem occurs with the system hardware.

CHAPTER 2

SEMICONDUCTOR MEMORY

This chapter provides you with an overview of semiconductor memory that can be used to store PDP-11 computer programs and data. The overview includes: types of memory devices, a short description of some of the PDP-11 memory modules, and information required to select the memory devices for your application program.

The types of memory devices used in PDP-11 application programs are as follows:

1. Random Access Memory (RAM) is a read/write semiconductor memory device for application programs that require space for storage of variables and buffers.

2. Read only Memory (ROM) is a read only memory device that are manufactured with the binary values in each addressable location. This memory is also referred to as masked ROM (see Section 2.3.1). ROM not used during program development.

3. Programmable Read Only Memory (PROM) is a type of read only memory device that is manufactured in the blank state (all 0's or all 1's) and programmed (blasted) by you with the desired bit pattern for your application program. The alterations made in the device for storage of the program are permanent (see Section 2.3.2).

4. Erasable Programmable Read Only Memory (EPROM) is the same as PROM, except you can erase the memory device (return the device to the blank state) if you expose the device to an intense ultra-violet light (see Sections 2.3.3 and 2.6).

2.1 MEMORY STORAGE ELEMENTS

Memory storage elements are divided into two major groups. The two groups are based on the volatility of the data in the storage element when dc power is lost. The two groups are:

1. Volatile; data or programs stored in memory are lost during a power failure. Semiconductor RAM memory chips fall into this group (see Section 2.2).

2. Non-volatile; data or programs are not lost during a power failure. The ROM, PROM, and EPROM memory chips fall into this group (see Section 2.3).

2.2 RANDOM ACCESS MEMORY (RAM)

Random Access Memory (RAM) may be semiconductor or core read/write memory. This manual refers to RAM as semiconductor read/write memory that is used with PROM (see Section 2.3) or EPROM (see Section 2.3.3) to store variables and provide buffers for the application programs stored in PROM or EPROM memory.

LSI-11 RAM memory chips are volatile (the contents of memory are lost when power is removed). However the LSI-11 can be configured to have a battery supply for short power failures.

The RAM memory used on the LSI-11 is also dynamic which means that it must be refreshed periodically. A refresh operation cycles the contents of each memory location, to ensure the integrity of the contents of memory. You should not concern yourself with the refresh operation if you use LSI-11 memory modules.

2.3 ROM, PROM, AND EPROM MEMORY: SUMMARY

ROM, PROM, and EPROM chips are non-volatile memory (the chips do not lose the data stored in the memory elements when power is lost).

ROM, PROM, and EPROM memory chips are considered to be in a blank state (the chips contain all binary zeros or ones) prior to programming. The blank state varies from chip to chip and between different IC manufacturers.

Programming of PROM chips consists of using a high current or voltage to permanently alter the internal structure of each chip, providing a new current path or closing an existing one. The process of altering the chip is called "blasting" the memory chip.

Programming of EPROM chips consist of charging a gate isolated in silicon dioxide, which opens a path for current flow. Unprogrammed bits appear as open circuits to a source voltage. EPROM chips may be erased (discharged) using an ultraviolet light (see Section 2.3.3 and 2.6).

Changes or additions to the program in memory can be made to bits that were not altered when the chip was originally programmed. However, if a bit is in the non-blank state and must be set to the blank state,

you must erase and reprogram the entire chip. PROM/RT-11 provides the facilities to check the chip to determine if the change can be made. If the change can be made PROM/RT-11 makes the necessary change or addition (see Section 5.).

PROM memory is used for applications that require security and the data integrity of hard-wired solid state logic and relay controllers, and in addition the requirement to provide flexibility, power, and low cost of a computer.

2.3.1 Read Only Memory (ROM)

Read Only Memory (ROM) chips (often referred as masked ROM) are purchased from an integrated circuit manufacturer with a program supplied by the user blasted into the chip. The user cannot make any changes to the program after he receives the chip. In any case the program in the chip(s) is read only and cannot contain variables or buffers for storage of data.

The one-time setup charge for manufacture of ROMs makes them more expensive than PROM or EPROM chips in low volume applications. Because of the frequent software changes required by program development ROM's are generally not used in low-volume applications.

2.3.2 Programmable Read Only Memory (PROM)

Programmable Read Only Memory (PROM) chips are BIPOLAR devices that are permanently altered by fusing or shorting links in the chip when the device is programmed (blasted). As the name implies you can program the chip(s) only once and the chip must be discarded if an instruction must be changed that requires a change back to the blank state on any bit.

2.3.3 Erasable Programmable Read Only Memory (EPROM)

Erasable Programmable Read Only Memory (EPROM) devices are MOS chips that retain each data bit as a stored charge in the memory cell. The memory cell for each bit is a Field Effect Transistor (FET) with a gate isolated in silicon dioxide. During programming, the gate is charged through avalanche injection and can be discharged (erased) using an ultra-violet light source (see Section 2.6).

2.4 LSI-11 SEMICONDUCTOR MEMORY: SUMMARY

You can select from a number of different LSI-11 memory modules. The PROM or EPROM modules provide sockets for you to mount your memory chips.

NOTE

The memory chips are soldered to the MSV11-B, MSV11-CD and MSV11-D-E modules at the factory. You cannot install or replace the memory chips.

MSV11-B 4k By 16-Bit Mos Read/Write Memory

The MSV11-B is a 4k by 16-bit dynamic Mos read/write memory module which you can use to store programs and data. Access time is 550 ns.

The MSV11-B can be installed at any 4k address boundary (for example location 00000, 40000, 60000, etc.).

MSV11-CD 16k By 16-Bit Mos Read/Write Memory

The MSV11-CD is a 16k dynamic Mos read/write memory module which you can use to store programs and data.

The MSV11-CD on any 4k boundary (for example location 00000, 40000, 60000, etc.).

MSV11-D-E Memory

There are eight versions of this module. They are as follows:

Model	Memory Capacity
MSV11-DA	4K by 16 bits
msv11-db	8K by 16 bits
msv11-dc	16K by 16 bits
msv11-dd	32K by 16 bits
msv11-ea	4K by 18 bits
msv11-eb	8K by 18 bits
msv11-ec	16K by 18 bits
msv11-ed	32K by 18 bits

The module can be installed at any 4K address boundary in the 0 to 128K address range.

NOTE

The PROM modules provide sockets for the

memory chips so that you can remove and replace the chips.

2.4.1 MRV11-AA 4K By 16-Bit Read-Only Memory

The MRV11-AA module provides space (as sockets) for up to 32 PROM chips. It accepts either 512x4 chips (for a total of 4K 16-bit words) or 256x4 chips (for a total of 2K 16-bit words). Each MRV11-AA can be installed at any 4K word address boundary for 4K configurations and any 2K boundary for 2K configurations in the LSI-11 address space (for example, location 0000, 20000, 40000, 60000, etc. for 4K configuration or 0, 10000, 20000, 30000, etc, for 2K configurations.)

2.4.2 MRV11-BA LSI-11UV PROM/RAM

The MRV11-BA module provides space (as sockets) for eight 1024x8 EPROM chips (for a total of 4K 16-bit words of PROM) and 256 16-bit words of static RAM memory.

Each MRV11-BA can be installed at any 4K address boundary (location 0000, 20000, 40000, 60000, etc.) and the RAM section can be independently configured to start at any 256 word memory address boundary (for example location 0000, 1000, 2000, 3000, 4000, etc.).

2.4.3 BDV11 Diagnostic Bootstrap Terminator

The BDV11 module provides ROM device bootstraps and system diagnostics and in addition to space for up to 18K 16-bit words of ROM or EPROM memory.

The BDV11 PROM/EPROM memory chips do not appear in the standard LSI-11 memory map, but instead is window mapped into the bootstrap area of the I/O page (173000 to 173777). The BDV11 provides an automatic means of loading the contents of user PROMs into RAM memory when system power is turned on.

2.5 MEMORY CHIP CHARACTERISTICS

Table 2-1 lists the characteristics of the memory chips available for LSI-11 PROM and EPROM memory. It also lists the memory option module that you can use the chip.

The trade-offs which influence your selection of memory chip types are as follows:

Access Time — Access time determines the speed that the processor can process program instructions.

EPROM devices with typical access time in the 0,5 microsecond range are slower than bipolar PROM devices which take approximately 0,1 microsecond to respond,

Development Cost — Because of frequent software changes, during the application development and debugging process, EPROM chips are often used, EPROMs may be erased and reused if the software is modified,

Production Cost — PROM chips are most often used for production operations because they are less expensive,

2,6 SELECTING MEMORY FOR YOUR APPLICATION

To select the type of memory for your application program, you should read the information presented in Figure 2-1 and the following sections,

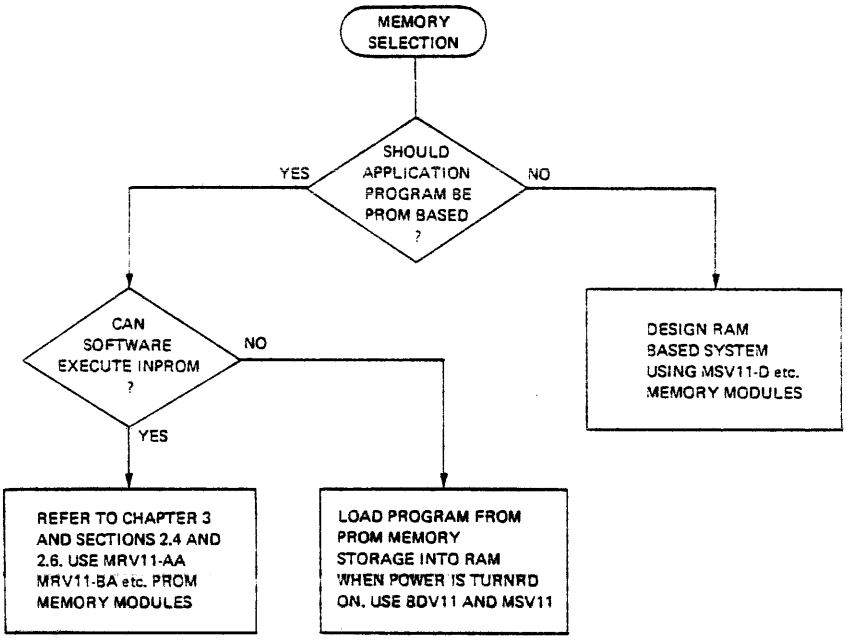


Figure 2-1 LSI-11 Memory Selection Flow Diagram

2,6,1 PROM Based Applications

Applications that require security and the data integrity of hardwired solid state logic and relay controllers and in addition the requirement to provide flexibility, power, and low cost of a computer use PROM or EPROM memory,

The reliability and non-volatile characteristics of read-only memory protect the system against program loss by electrical noise, AC cycle dropouts, power brownouts, operator errors, or power failures,

The fact that PROM and EPROM memories are "blasted" (the internal structure of the memory device is permanently altered) ensures the stability of the program. But, on the other hand inhibits the use of instructions or operations that are self modifying. However, if you are using a PDP-11 family computer, you can store the main program in non-volatile read-only memory and use a RAM memory to store and retrieve data. Or you can store your program in PROM and transfer it to RAM for execution (see Section 2.6.3).

2.6.2 Disadvantages of Semiconductor RAM Memory

Semiconductor RAM memory is volatile. Therefore, when you remove power from the system the programs and data stored in memory are lost. This means that the program must be stored on an external device and transferred to memory when you turn power on.

On small systems, the peripheral devices that are cost effective for you to store the program may be one of the following:

1. A floppy disk
2. A tape cartridge
3. A paper-tape reader
4. A serial line connected to another computer that stores the program.

The devices all have a common characteristic, they do not operate efficiently and reliably in hostile environments. The operation of peripheral devices is affected by temperature changes, dirt, electrical problems, and operator errors.

Adding peripheral devices to the system also increases the cost of a system and for some applications the cost could be prohibitive.

Therefore PROM or EPROM memory must be used for these applications, because the program remains in PROM when you remove power, PROM memory is reliable, and it is cost effective.

2.6.3 Application Programs That Will Not Execute Out of PROM

Programs that you write in the MACRO and/or FORTRAN languages will execute out of PROM; if they are designed properly (see Chapter 3) and if they will fit into memory. Thus the problem is with software that exists already. You have two options; you can rewrite the software so that it will operate out of PROM (see Chapter 3) or you can store it in PROM and transfer it to RAM for execution.

For the first option you must analyze the software and determine if it is possible to rewrite the software. If you decide to rewrite the

software you should follow the procedures in Chapter 3.

The second option is to store the program in PROM and transfer it to RAM for execution. You can use the BDV11 memory option for this operation (see Section 2.4.?).

The BDV11 PROM/EPROM chips do not appear in the standard LSI-11 memory map, but is window mapped into the bootstrap area of the I/O page (173000 to 173777). The BDV11 automatically loads the contents of your PROMs or EPROMs into RAM when the power is turned on.

Table 2-1
Semiconductor Memory Chip Information

Memory Chip Designation	Type of Memory	Chip Size (number of words x bit width)	Access Time	Memory Module
2708	EPROM	1Kx8	350ns	MRV11-BA
2716	EPROM	2Kx8	450ns	BDV11-A
2732	EPROM	4Kx8	450ns	Contact your local DIGITAL Sales Representative
82S129	PROM	256x4	70ns	MRV11-AA
82S131	PROM	512x4	70ns	MRV11-AA
82S181	PROM	1024x8	70ns	Contact your local DIGITAL Sales Representative
82S191	PROM	2048x8	80ns	Contact your local DIGITAL Sales Representative

2.7 EPROM ERASING PROCEDURE

The EPROM chips may be erased by exposure to high intensity, ultraviolet light at a wave length of 2537A (Angstrom).

The recommended integrated dose (UV intensity x exposure time) is 6W sec/cm2. The ultraviolet lamps should be used without shortwave filters, and the EPROM should be placed about one inch away from the lamp tube to be erased. This operation has the effect of writing all

1's or 0's (depending on the blank state of the chip) into the EPROM.

WARNING

Short wave ultraviolet light can cause "sunburning" of the eyes and skin. Eyes should be protected from exposure.

CHAPTER 3

DESIGN CONSIDERATIONS FOR PROM-BASED APPLICATION PROGRAMS

3.1 INTRODUCTION

Information presented in this chapter is directed toward two dissimilar user groups. The differences between the two groups stem from their degree of exposure to PROM/EPROM hardware on the one hand and PDP-11 software on the other. In general, the two groups and their backgrounds are:

1. Chip level hardware designers who have used PROMs for firmware driven controller design but have little knowledge of PDP-11 software. Such persons may need additional exposure to techniques for using PROM/RAM memory for an entire application program.

2. Traditional programmers who have used PDP-11 software (though not necessarily LSI-11) but are unfamiliar with ROMs, PROMs, and EPROMs. Persons in this category should thoroughly read Chapter 2.

3.1.1 Application Program Development Sequence

An application program is initially generated on a development system having a disk based operating system. After development and preliminary listing, the application is assembled/compiled and then LINKed preparatory to blasting the program bit patterns into PROM/EPROM chips. The next step in the sequence is invoking the PROM/RT-11 utility program (Chapter 5) which is used to enter the application program into PROM chips while they are mounted in the PROM blaster. After blasting, the PROM chips are mounted into host printed circuit boards such as the MRV11-AA or MRV11-BA modules.*1 Lastly, the host printed circuit module is mounted in an LSI-11 backplane. The application program is now contained in PROM and is ready for execution on LSI-11 power up.

---

1. Other host printed circuit modules are available. Consult your DIGITAL Sales Representative for further information.

3.1.2 Languages Used During Application Program Development

When coding an application program for the disk based development system, two programming languages are generally used. These languages and considerations governing when they might be most favorably used are

1. Assembly Language (MACRO-11) Only

   When speed is a major consideration for a PROM-destined application, you should code the program in assembly language only. This is because assembly language has less overhead than higher level languages. When using assembly language only certain extra steps are required of the programmer. These are discussed in the next sub section. An example of a simple application developed using MACRO-11 only is given in Appendix A.

2. FORTRAN IV combined with Assembly Language

   For large PROM based application programs, particularly those requiring floating point computation, you may wish to develop the application using a combination of FORTRAN IV and MACRO-11 assembly language. In such cases, main sections of the application can be developed using FORTRAN IV while MACRO-11 can be used to generate device handler subroutines. An example of such a program is given in Appendix A.

3.2 DEVELOPMENT CONSIDERATIONS WHEN CODING IN ASSEMBLY LANGUAGE ONLY

Characteristically, as an assembly language programmer using RT-11 you generate a program in three steps:

1. Assemble
2. Link
3. Run

Features built into the RT-11 operating system free you from such concerns as absolute memory address assignments or how the program is to be started. In the latter case you simply give the RUN command. Table 3-1 contrasts automatic features of the RT-11 operating system with corresponding actions that you must take into account for a PROM based application program.

Other assembly language coding considerations involve the positioning of certain categories of code in PROM or RAM. This is discussed in Section 3.3.

Table 3-1

RT-11 Operating System Features Contrasted with
PROM Based Application Requirements
When Coding with Assembly Language

RT-11 Operating System Features	PROM Based Application Program Requirements
1. Loads program into Random Access Memory (RAM) automatically	1. Application programmer must blast absolute and pure code sections of program into PROM/EPROM and assign variables handled by the program to RAM. See Section 3.3.
2. Automatically sets up stack pointer	2. Application programmer must assign low order area of RAM as the stack and set stack pointer. See Section 3.3.2 and Appendix F.
3. Starts execution of program when RUN command is given	3. Auto start on power up by using startup vector in PROM locations 24(8) and 26(8). An alternative method of power up occurs when the user is creating his own bootstrap at location 173000(8). In such a case he can use the LSI-11 power up mode which starts execution at this location. See Section 3.3.2 and Appendix F.

3.3 STRUCTURING PROM/RAM MEMORY FOR APPLICATION PROGRAMS

The prime consideration in developing an application program, whether using assembly language only or FORTRAN IV plus assembly language, is the separation of PROM based data from RAM based data. This is in contrast to programs written for execution in read/write memory. When generating an application program on a development system, you must conform with the following fundamental structure.

1. Absolute address section of application. Code for this section is to be PROM based and located in the low order absolute memory addresses (i.e., the vector area).

2. Machine instructions and pure data (such as constants, messages, etc.)

   Code in this category is also to be PROM based and located above the absolute address section

3. Scratch data section. This section is to be RAM based and contains the variables and buffers of the application program

Interrupt vectors have absolute addresses. Since you want to start your PROM based application using the power-up vector at locations 24(8) and 26(8), the absolute address section of the application must be located in the low order address area of PROM. Given this consideration, machine instructions and pure code are best placed in PROM immediately above the absolute section since this represents the most efficient and economical use of your printed circuit memory modules.

3.3.1 Using Program Sections (PSECTS) for Structuring PROM Based Application Programs

When developing a PROM based application program the program section (.PSECT) feature of RT-11 provides a convenient means of ordering like data. This is true whether the data is intended for PROM or RAM. The protocol for using PSECTs differs when using assembly language only versus FORTRAN IV. Rules for using PSECTs when coding in assembly language only are given in Section 3.3.2 and Appendix F. Program sections can be used to define

1. Absolute section destined for PROM

2. Pure code section destined for PROM

3. Scratch data section destined for RAM

After you have used the .PSECTs to group and define like data, you must execute the LINK program to generate a memory image file. (The LINK maps described later depict the ordering of like data within the memory image file.

The LINKage editor treats all .PSECTs equally since it has no mechanism for distinguishing one from the other. However, the LINKage editor will always assign the .PSECTs in the order of their presentation. Therefore when generating an application in assembly language the .PSECTs to be used and the order of their presentation (to the LINKage editor) are:

1. .ASECT

2. .PSECT ROM

3. .PSECT RAM

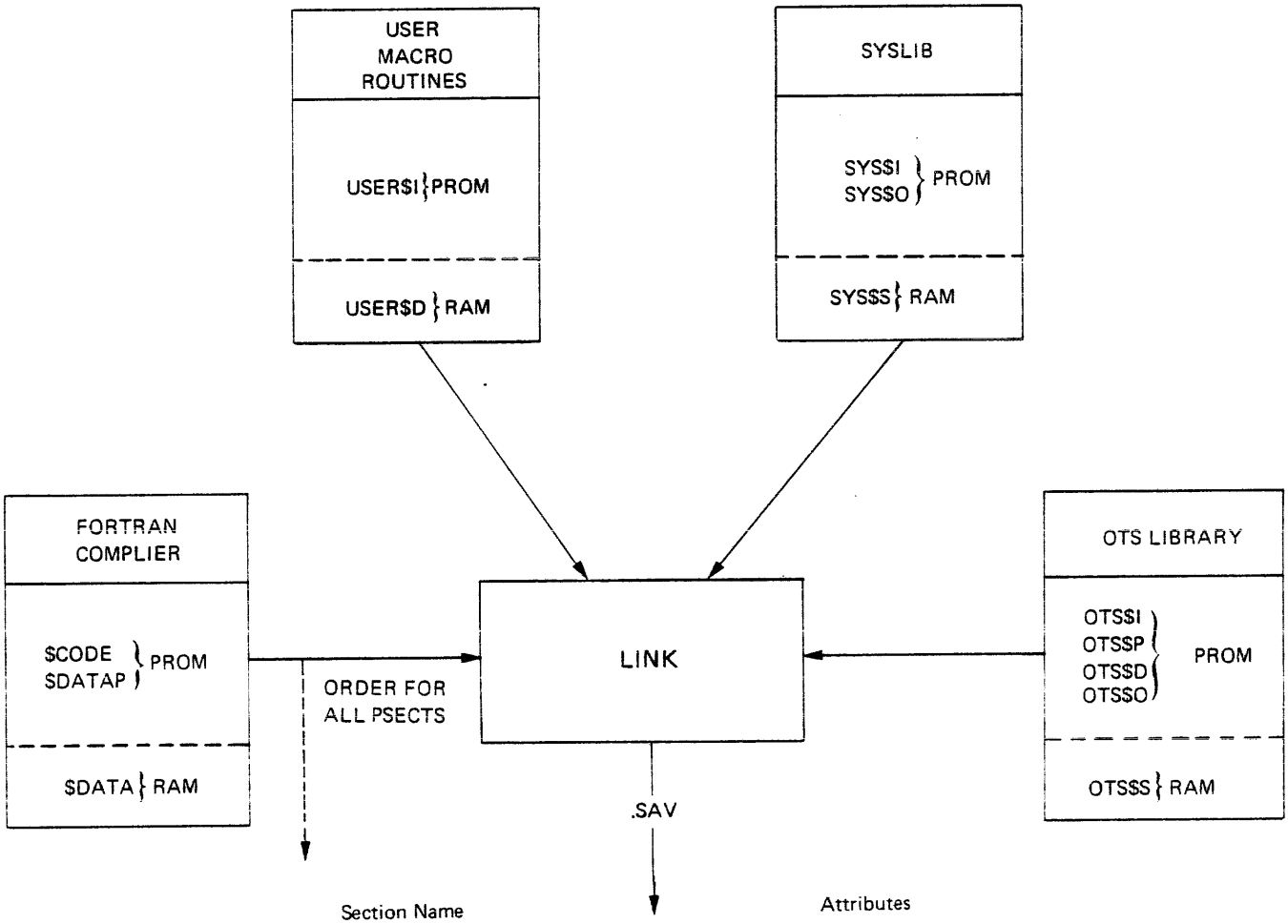
When using FORTRAN IV and MACRO-11, PSECTs are identified by unique

names as segments of the object program. Attributes associated with each .PSECT cause the LINK utility to combine program units from four different sources as follows:

1. FORTRAN compiler
2. User MACRO routines
3. OTS Library
4. System Library (SYSLIB)

Figure 3-1 shows the PSECTs produced from each of these sources and the resultant ordering produced by the LINK utility.

More information on using .PSECTs is available in the PDP-11 MACRO-11 Programmer's Guide and the LINK section of the RT-11 System User's Guide.



Section Name	Attributes				
OTS$I	RW,	I,	LCL,	REL,	CON
OTS$P	RW,	D,	GBL,	REL,	CON
SYS$I	RW,	I,	LCL,	REL,	CON
USER$I	RW,	I,	LCL,	REL,	CON
$CODE	RW,	I,	LCL,	REL,	CON
OTS$O	RW,	I,	LCL,	REL,	CON
SYS$O	RW,	I,	LCL,	REL,	CON
$DATAP	RW,	D,	LCL,	REL,	CON
OTS$D	RW,	D,	LCL,	REL,	CON
OTS$S	RW,	D,	LCL,	REL,	CON
SYS$S	RW,	D,	LCL,	REL,	CON
$DATA	RW,	D,	LCL,	REL,	CON
USER$D	RW,	D,	LCL,	REL,	CON
.$$$$.	RW,	D,	GBL,	REL,	CON
OTHER COMMON BLOCKS	RW,	D,	GBL,	REL,	CON

Figure 3-1 PSECTs Presented to LINK Utility and Resultant Ordering Sequence

3.3.1.1 Typical PROM/RAM Structure for Application Program Generated in Assembly Language Only - Use of .PSECT directives when developing an application in assembly language only calls for the following .PSECT sequence.

1. .ASECT for absolute address section

2. .PSECT ROM, I, LCL, REL, CON
   For pure code and data section

3. .PSECT RAM, D, LCL, REL, CON
   For scratch data (variables)

The above defines the general ordering of .PSECTs to accommodate the PROM based and RAM based data. An assembly language definition module that can be used to achieve this ordering sequence is given in Appendix F. The example shown below indicates how the data is organized in a sample application program. The example is taken from the first sample application in Appendix A.

Notice that the following three program sections are used exclusively:

1. .ASECT — to initialize low-memory areas (vectors)
2. .PSECT ROM — to define all instructions and data to be placed in PROM memory
3. .PSECT RAM,D — to define all RAM (scratchpad) locations

```
        .GLOBAL INIT, HOURS, MINS, SECS ;External variables from other module

        .ASECT                          ;Go to absolute section to define vectors
.=24                                    ;Origin to the power-up vector
        .WORD   PWRUP                   ;Power-up PC = routine labeled "PWRUP"
        .WORD   340                     ;Power-up PS indicates PRIO 7 (no interrupts)

        .PSECT  ROM                     ;Origin to ROM section
PWRUP:  MOV     #SSTACK,SP              ;Set up a valid hardware stack pointer
        JSR     PC,INIT                 ;And initialize all required RAM locations
                                        ; defined in the clock module
        CLR     R1                      ;Init R1 for MTPS operand
        MTPS    R1                      ;When all locations have been initialized,
                                        ; start clock module operating by allowing
                                        ;  interrupts.
KEYCHK: TSTB    @#177560                ;Wait for console keyboard ready
        BPL     KEYCHK                  ;If PL, no character has been typed
        TSTB    @#177562                ;Else read the character
        JSR     R5,PRINT                ;And begin the print the time message
         .WORD  MESS1                   ; by calling the PRINT subroutine
        MOV     HOURS,R0                ;Place number of hours elapsed in R0
        JSR     PC,DECOUT               ;And convert it to decimal ASCII, typing it
                                        ; out on the console
        JSR     R5,PRINT                ;Print out " hours, "
         .WORD  MESS2
        MOV     MINS,R0                 ;Get number of elapsed minutes
        JSR     PC,DECOUT               ;Convert and print out minutes
        JSR     R5,PRINT                ;Print out " minutes, and "
         .WORD  MESS3
```

```
        MOV     SECS,R0         ;Get number of elapsed seconds
        JSR     PC,DECOUT       ;Convert and print out seconds
        JSR     R5,PRINT        ;Print out " seconds," and carriage return +
         ,WORD  MESS4           ; line feed characters to return carriage
        BR      KEYCHK          ;Go wait for another character to be typed.

PRINT:  MOV     (R5)+,R4        ;R4 = address of message to be printed
PCHAR:  MOVB    (R4)+,R0        ;R0 = next character in message to print

        BEQ     PRET            ;If EQ, entire message has been printed
        JSR     PC,TYPECH       ;Else type out the character in R0
        BR      PCHAR           ;And go get next message character

PRET:   RTS     R5              ;Return to calling routine

DECOUT: SWAB    R0              ;Convert binary 00-99 to two decimal ASCII
DLOOP:  ADD     #<-10,*256,>+1,R0 ; digits, by doing a byte division operation
        BPL     DLOOP           ;   maintaining quotient in low byte and
        ADD     #<10,*256,>-1+"00,R0 ; remainder in high byte of R0
        JSR     PC,TYPECH       ;Type out most significant digit
        SWAB    R0              ;Move least significant digit down
                                ; and type it out also ...

TYPECH: TSTB    @#177564        ;Is printer ready to accept a character?
        BPL     TYPECH          ;If PL no - still printing last one
        MOVB    R0,@#177566     ;Else print this character
        RTS     PC              ;And return to calling routine

        ,PSECT  RAM,D           ;Origin into RAM to define stack
        ,BLKW   64.             ;Reserve space for hardware stack at beginning
                                ; of RAM
SSTACK::                        ;And place reference label at end (as stack
                                ; grows towards lower addresses)

        ,PSECT  ROM             ;Origin back to ROM to define message text
MESS1:  ,ASCIZ  /Elapsed time since power-up is /

MESS2:  ,ASCIZ  / hours, /

MESS3:  ,ASCIZ  / minutes, and /

MESS4:  ,ASCIZ  / seconds,/<015><012>

        ,END
```

3.3.1.2 Typical PROM/RAM Structure for Assembly Language Subroutine Used in Application Developed With Both FORTRAN IV and MACRO-11 - This section presents an example of how PSECTs are used when generating a subroutine in MACRO-11 assembly language. The difference between this example and that of the preceding paragraph results from the fact that the below application is developed using a combination of FORTRAN IV and MACRO-11. The .ASECT program section defines the absolute address section just as in the preceding example. However here, PSECTs USER$I

and USER$D are used to define PROM based data and RAM based data respectively. When coding subroutines in MACRO-11 assembly language for use in FORTRAN PROM based applications the USER$I and USER$D .PSECTs must be used. The example given here is a subroutine taken from the second sample application in Appendix A.

```
     .TITLE  Subroutines for SCALER.FOR

     .SBTTL  DRV11 Interrupt Servicing

; The routine DRVINT services interrupts from all 8 DRV11 interfaces
; installed in the system. The line number of the interrupting DRV11
; is encoded in the condition code bits of the new processor status
; word fetched as the result of the interrupt.

; Initialization of the DRV11 interrupt vectors:

        .ASECT
.=300
        .WORD   DRVINT, 340+1,  ;Vector for line #1
        .WORD   DRVINT, 340+2,  ;Vector for line #2
        .WORD   DRVINT, 340+3,  ;Vector for line #3
        .WORD   DRVINT, 340+4,  ;Vector for line #4
        .WORD   DRVINT, 340+5,  ;Vector for line #5
        .WORD   DRVINT, 340+6,  ;Vector for line #6
        .WORD   DRVINT, 340+7,  ;Vector for line #7
        .WORD   DRVINT, 340+8,  ;Vector for line #8

; Now, the interrupt routine itself (in the PROM Psect):

        .GLOBL  WEIGH           ;External FORTRAN routine to service intr

        .PSECT  USER$I
DRVINT: JSR     R0,@PC          ;Save a register on stack with destroying
        MFPS    R0              ; condition codes, and retrieve them
        BIC     #177760,R0      ;Clear all but line #
        MOV     R0,DRVLIN       ;And store for reference by WEIGH routine
        MOV     R1,-(SP)        ;Save remaining registers
        MOV     R2,-(SP)        ; on stack, as FORTRAN
        MOV     R3,-(SP)        ;  routine may use them
        MOV     R4,-(SP)
        MOV     R5,-(SP)
        MOV     #ARGLST,R5      ;Load FORTRAN argument list pointer
        JSR     PC,WEIGH        ;And call routine to process interrupt data
        MOV     (SP)+,R5        ;Restore registers
        MOV     (SP)+,R4        ; after return from
        MOV     (SP)+,R3        ;  FORTRAN routine
        MOV     (SP)+,R2
        MOV     (SP)+,R1
        MOV     (SP)+,R0
        RTI                     ;And return from the interrupt

ARGLST: .WORD   1, DRVLIN       ;Argument list for call to WEIGH

        .PSECT  USER$D,D        ;Origin to RAM section
```

```
DRVLIN:  .BLKW                  ;And define variable to get line #
```

3.4 LINKING PROM BASED APPLICATION PROGRAMS

After developing the application program, you must invoke the LINKage editor to properly sequence PROM destined and RAM destined data. The protocol for invoking the LINK command differs somewhat depending on the following

1. The application is coded in assembly language only
2. The application is coded in a combination of FORTRAN IV and MACRO-11 assembly language

When LINKing any PROM based application the LINKage editor (LINK) must:

1. Load .PSECTs into memory image in desired order (absolute section, then ROM section, then RAM)
2. Set base of ROM section above vectors (not at 1000 as normally done in RT-11)
3. Round up base of RAM section to align it with the next 4K word boundary
4. Include some stack space in RAM

The LINK utility loads .PSECTs into memory image in the order in which they are mentioned in the input object module stream.

3.4.1 LINKing an Assembly Language Application Program

Creating a definition module which mentions the .PSECTs in the desired order solves any ordering problem. This definition module is given in Appendix F.

Once you have assembled ROMDEF module has been assembled to produce ROMDEF.OBJ, the linking process for applications which have been coded with the listed .PSECTs is:

```
.LINK/BOTTOM:400 /BOUNDARY:020000 /EXE:APPLIC
Files? ROMDEF,user-file-1,,,,,user-file-n
Boundary section? $STACK
```

NOTE

If ROMDEF is not used, the first RAM based .PSECT name should replace "$STACK" as the reply to the "Boundary

section?" query.

To extend the size of the stack space to mmmmmm (octal), use:

```
,LINK/BOTTOM:400 /BOUNDARY:020000 /EXE:APPLIC /EXTEND:mmmmmm
Files? ROMDEF,user-file-1,...,user-file-n
Extend section? $STACK
Boundary section? $STACK
```

3.4.2 Linking an Application Developed in Both FORTRAN IV and Assembly Language

To LINK a PROM FORTRAN application, use the following command:

```
,LINK/BOTTOM:400 /BOUNDARY:020000 /INCLUDE /MAP
Files? user-file-1,...,user-file-n
Boundary section? OTS$S
Library search? $SIMRT
Library search?
```

Default stack size is 64. words. To extend the stack to mmmmmm (octal), use the following command:

```
,LINK/BOTTOM:400 /BOUNDARY:020000 /INCLUDE /MAP /EXTEND:mmmmmm
Files? user-file-1,...,user-file-n
Extend section? $STACK
Boundary section? OTS$S
Library search? $SIMRT
Library search?
```

NOTE

Using values for the /BOTTOM (8) switch that are less than 400 requires use of the /X switch to the LINKage editor. See RT-11 System User's Guide.

The number of PROM boards required (and RAM size) can be determined by examining the map produced by LINK. Refer to Section 3.5.

3.5 OBTAINING LINK MAPS

Printing or displaying a LINK map is a necessary step prior to blasting the PROMs as described in Chapter 5. Through appropriate use of program sections (.PSECTs) and execution of the LINK utility, all PROM destined data is now separated from RAM destined data. The LINK map is now required so that when you are blasting the PROMs you can determine the high order address of the PROM destined data; (see the description of the PROGRAM command in Chapter 5).

For any given application, you must generate LINK maps in two steps for the following reasons;

1. The first map is produced by invoking the LINK utility without assigning RAM destined data on the next 4K boundary (the BOUNDARY switch of the LINK instruction is not used). In this case, the LINK utility places the RAM data immediately after the PROM data in the memory image file. The map produced by LINK reflects this sequence by defining the RAM data immediately above the PROM data.

2. The second LINK map is produced by invoking the LINK utility with the BOUNDARY switch used to assign RAM data at the start of the next 4K boundary.

The illustration below shows pre-boundary and post boundary memory image maps.

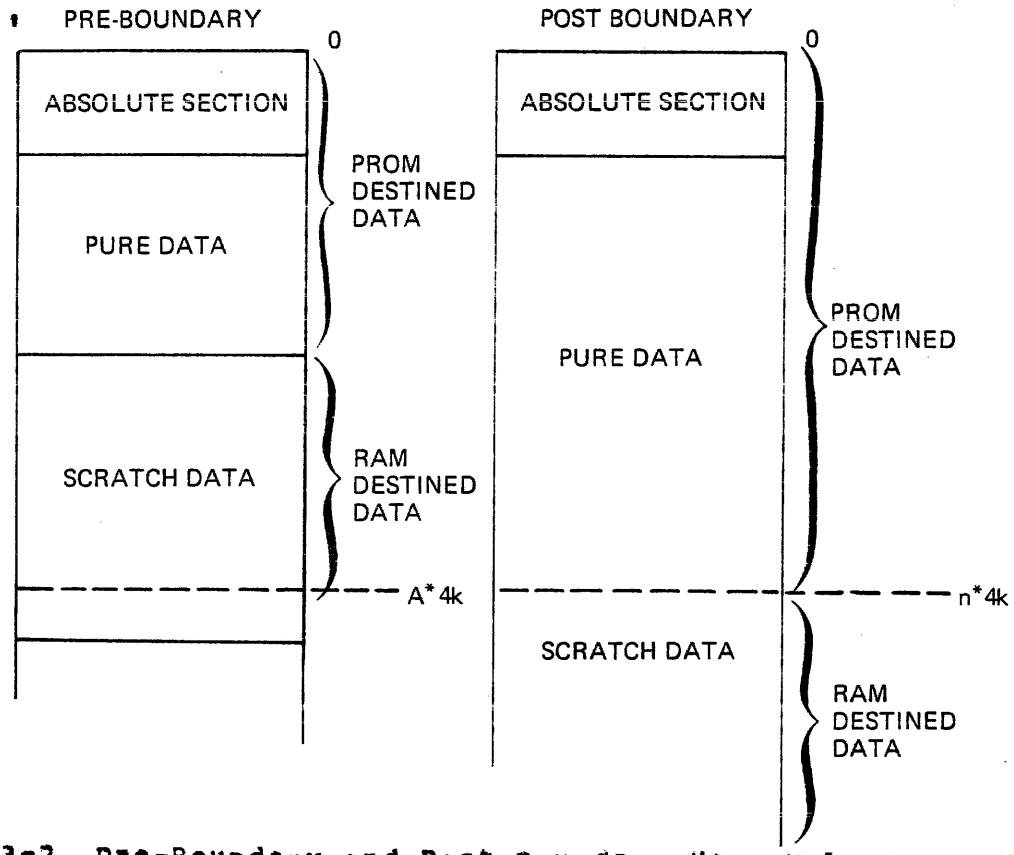


Figure 3-2 Pre-Boundary and Post Boundary Memory Language Maps

3.5.1 Assembly Language LINK Map

The examples below show two LINK maps for the sample application program described in Section A.2 of Appendix A. Note that the first map is produced when no BOUNDARY switch is given as part of the LINK instruction parameters. This map shows RAM destined data beginning at location 000776(8). Thus it can be concluded that the highest order

address for PROM destined data is 000775(8). This is the value that must be fed to the PROGRAM command (Chapter 5) when blasting the PROMs.

The second LINK map is produced when the BOUNDARY switch is used to assign RAM destined data at next 4K boundary. RAM data now starts at location 020000(8). The second LINK map is useful because it indicates the beginning and ending address of the STACK in RAM as well as defining RAM memory requirements for other program elements.

PRE-BOUNDARY LINK MAP

```
LINK/BOTTOM:400/EXECUTE:MAIN/MAP:TT: MAIN,CLOCK
RT-11 LINK  V05.02      Load Map
MAIN  .SAV      Title:  MAIN    Ident:          /B:000400

Section  Addr   Size    Global  Value   Global  Value   Global  Value

. ABS.  000000  000400   (RW,I,GBL,ABS,OVR)
ROM     00400   00376    (RW,I,LCL,REL,CON)
                INIT    000752
RAM     000776  000210   (RW,D,LCL,REL,CON)
                $STACK  001176  TICKS   001176  SECS    001200
                MINS    001202  HOURS   001204

Transfer address = 000001, High limit = 001206 =   323. words
```

POST-BOUNDARY LINK MAP

```
 LINK/BOTTOM:400/EXECUTE:MAIN/MAP:TT:/BOUNDARY:20000 MAIN,CLOCK
Boundary section? RAM
RT-11 LINK  V05.02      Load Map
MAIN  .SAV      Title   MAIN    Ident:          /B:000400

Section  Addr   Size    Global  Value   Global  Value   Global  Value

. ABS   000000  000400   (RW,I,GBL,ABS,OVR)
ROM     000400  017400   (RW,I,LCL,REL,CON)
                INIT    000752
RAM     020000  000210   (RW,D,LCL,REL,CON)
                $STACK  020200  TICKS   020200  SECS    020202
                MINS    020204  HOURS   020206

Transfer address = 000001, High limit = 020210 =  4164. words
```

3.5.2 FORTRAN IV and MACRO-11 LINK Maps

This paragraph presents two LINK maps for the second application described in Appendix A (section A.3). For applications coded in a combination of FORTRAN IV and MACRO-11, program section OTSsD is the high order PSECT for PROM destined data (see Figure 3-1). Similarly

OTSSS is the low order program section for RAM destined data. In the first example given below, the LINK BOUNDARY switch is not invoked. Consequently program section OTS$S follows immediately after OTSSD in the memory image file. The high order PROM address given to the PROM blasting PROGRAM command is 025051(8) in this case.

When the second LINK map is produced, the BOUNDARY swtich is used to set the OTSSS program section on the next 4K boundary. In this case the boundary is set at 040000(8) because 2 MRV11-AA/MRV11-BA modules are required to accommodate the PROM based program section.

PRE-BOUNDARY LINK MAP

```
RT-11 LINK  V05.04A      Load Map          Fri 01-Sep-78 00:38:04
SCALER,SAV       Title:  SCALER  Ident:  FORY02  /B:000400

. ABS     000000 000400   (RW,I,GBL,ABS,OVR)
                          $USRSW  000000  $RF2A1  000000  .VIR    000000
                          $NLCHN  000006  $HRDWR  000006  $SYSV$  000010
                          $WASIZ  000131  $LRECL  000210
OTS$I     000400 020330   (RW,I,LCL,REL,CON)
                          $$OTSI  000400  $SIMRT  000566  $TKS    001416
                          $TKB    001420  $TPS    001422  $TPB    001424
                          $CVTFB  002462  $CVTFI  002462  $CVTCB  002476
                          $CVTCI  002476  $CVTDB  002476  $CVTDI  002476
                          CIC$    002510  CID$    002510  CLC$    002510
                          CLD$    002510  $DI     002510  CIF$    002520
                          CLF$    002520  $RI     002520  CIL$    002626
                          CLI$    002632  $OTI    002662  $$OTI   002664
                          $$SET   004450  RCI$    005062  GCO$    006076
                          FCO$    006104  ECO$    006110  DCO$    006116
                          $CVTIF  007040  $CVTIC  007054  $CVTID  007054
                          CCI$    007066  CDI$    007066  $IC     007066
                          $ID     007066  CFI$    007102  $IR     007102
                          TVL$    007166  $TVL    007166  TVF$    007174
                          $TVF    007174  TVD$    007202  $TVD    007202
                          TVQ$    007210  $TVQ    007210  TVP$    007216
                          $TVP    007216  TVI$    007224  $TVI    007224
                          OCI$    007360  ICI$    007366  $ECI    007402
                          OCO$    007562  ICO$    007570  IFW$    007766
                          $IFW    007772  IFW$$   010034  IFR$    010104
                          $IFR    010110  IFRS$   010146  $CHKER  010170
                          $IOEXI  010214   EOL    010242  EOL$    010244
                          $OTIS   010360  $$OTIS  010362  SAVRG$  010502
                          THRD$   010660  $PUTRE  010662  $STPS   011170
                          STP$    011176  $STP    011176  FOO$    011202
                          $EXIT   011222  $ERRTB  011346  $ERRS   011453
                          $WAIT   015146  $FCHNL  015210  $INITI  015306
                          $CLOSE  015420  $GETRE  016064  $TTYIN  016140
                          $PUTBL  016274  $GETBL  016504  $EOFIL  016670
                          $EOF2   016704  $FIO    017444  $$FIO   017450
                          $VRINT  020600  $DUMPL  020602
```

```
OTS$P     020730 000050   (RW,D,GBL,REL,OVR)
SYS$I     021000 000020   (RW,I,LCL,REL,CON)
                          IPEEK   021000  IPOKE   021010
USER$I    021020 000214   (RW,I,LCL,REL,CON)
                          IBCDI   021100  IBCDO   021134  IADC    021200
$CODE     021234 001662   (RW,I,LCL,REL,CON)
                          $OTSC   021234  WEIGH   022376
OTS$O     023116 001010   (RW,I,LCL,REL,CON)
                          $$OTSO  023116  $OPEN   023116
SYS$O     024126 000000   (RW,I,LCL,REL,CON)
$DATAP    024126 000716   (RW,D,LCL,REL,CON)
OTS$D     025044 000006   (RW,D,LCL,REL,CON)
OTS$S     025052 000304   (RW,D,LCL,REL,CON)
                          $AOTS   025354
SYS$S     025356 000004   (RW,D,LCL,REL,CON)
                          $SYSLB  025356  $LOCK   025360  $CRASH  025361
$DATA     025362 000072   (RW,D,LCL,REL,CON)
USER$D    025454 000002   (RW,D,LCL,REL,CON)
.$$$$.    025456 000160   (RW,D,GBL,REL,OVR)
$STACK    025636 000200   (RW,D,LCL,REL,CON)
$STKST    026036 000000   (RW,D,LCL,REL,CON)
                          $$$STK  026036

Transfer address = 021234, High limit = 026036 =  5647. words


POST-BOUNDARY LINK MAP

RT-11 LINK  V05.04A       Load Map        Fri 01-Sep-78 00:39:46
SCALER.SAV        Title:  SCALER  Ident:  FORY02  /B:000400

Section   Addr    Size    Global  Value   Global  Value   Global  Value

. ABS.    000000 000400   (RW,I,GBL,ABS,OVR)
                          $USRSW  000000  $RF2A1  000000  .VIR    000000
                          $NLCHN  000006  $HRDWR  000006  $SYSV$  000010
                          $WASIZ  000131  $LRECL  000210
OTS$I     000400 020330   (RW,I,LCL,REL,CON)
                          $$OTSI  000400  $SIMRT  000566  $TKS    001416
                          $TKB    001420  $TPS    001422  $TPB    001424
                          $CVTFB  002462  $CVTFI  002462  $CVTCB  002476
                          $CVTCI  002476  $CVTDB  002476  $CVTDI  002476
                          CIC$    002510  CID$    002510  CLC$    002510
                          CLD$    002510  $DI     002510  CIF$    002520
                          CLF$    002520  $RI     002520  CIL$    002626
                          CLI$    002632  $OTI    002662  $$OTI   002664
                          $$SET   004450  RCI$    005062  GCO$    006076
                          FCO$    006104  ECO$    006110  DCO$    006116
                          $CVTIF  007040  $CVTIC  007054  $CVTID  007054
                          CCI$    007066  CDI$    007066  $IC     007066
                          $ID     007066  CFI$    007102  $IR     007102
                          TVL$    007166  $TVL    007166  TVF$    007174
                          $TVF    007174  TVD$    007202  $TVD    007202
                          TVQ$    007210  $TVQ    007210  TVP$    007216
                          $TVP    007216  TVI$    007224  $TVI    007224
```

```
                                OCI$     007360  ICI$     007366  $ECI     007402
                                OCO$     007562  ICO$     007570  IFW$     007766
                                $IFW     007772  IFW$$    010034  IFR$     010104
                                $IFR     010110  IFR$$    010146  $CHKER   010170
                                $IOEXI   010214   EOL     010242  EOL$     010244
                                $OTIS    010360  $$OTIS   010362  SAVRG$   010502
                                THRD$    010660  $PUTRE   010662  $$TPS    011170
                                STP$     011176  $STP     011176  FOO$     011202
                                $EXIT    011222  $ERRTB   011346  $ERR$    011453
                                $WAIT    015146  $FCHNL   015210  $INITI   015306
                                $CLOSE   015420  $GETRE   016064  $TTYIN   016140
                                $PUTBL   016274  $GETBL   016504  $EOFIL   016670
                                $EOF2    016704  $FIO     017444  $$FIO    017450
                                $VRINT   020600  $DUMPL   020602
OTS$P     020730 000050          (RW,D,GBL,REL,OVR)
SYS$I     021000 000020          (RW,I,LCL,REL,CON)
                                IPEEK    021000  IPOKE   021010
USER$I    021020 000214          (RW,I,LCL,REL,CON)
                                IBCDI    021100  IBCDO   021134  IADC     021200
$CODE     021234 001662          (RW,I,LCL,REL,CON)
                                $$OTSC   021234  WEIGH   022376
OTS$O     023116 001010          (RW,I,LCL,REL,CON)
                                $$OTSO   023116  $OPEN   023116
SYS$O     024126 000000          (RW,I,LCL,REL,CON)
$DATAP    024126 000716          (RW,D,LCL,REL,CON)
OTS$D     025044 012734          (RW,D,LCL,REL,CON)
OTS$S     040000 000304          (RW,D,LCL,REL,CON)
                                $AOTS    040302
SYS$S     040304 000004          (RW,D,LCL,REL,CON)
                                $SYSLB   040304  $LOCK   040306  $CRASH   040307
$DATA     040310 000072          (RW,D,LCL,REL,CON)
USER$D    040402 000002          (RW,D,LCL,REL,CON)
.$$$$.    040404 000160          (RW,D,GBL,REL,OVR)
$STACK    040564 000200          (RW,D,LCL,REL,CON)
$STKST    040764 000000          (RW,D,LCL,REL,CON)
                                $$$STK   040764

Transfer address = 021234, High limit = 040764 =  8442. words
```

3.6 SAMPLE PROM BASED APPLICATIONS

This section briefly describes the design objective of the two sample application programs given in Appendix A. The first sample application program is coded in MACRO-11 assembly language only. The second sample application is coded in a combination of FORTRAN IV and MACRO-11 assembly language.

3.6.1 Elapsed Time Since Power Up Application Program

The elapsed time since power up application program consists of two modules, both coded in assembly language. These modules and their purposes are:

1. MAIN.MAC. The module prints out the elapsed time since power up calculated by the second module (CLOCK.MAC). Print out is in hours, minutes and seconds and occurs only when a terminal key is pressed. The handler that detects typing of any key and printing the message is contained in this module.

2. CLOCK.MAC. This module provides a clock interrupt handler and the code to maintain the current time of day (elapsed time since bootstrap on power up) for the main module (MAIN.MAC)

3.6.2 Price Computation for Weighed Merchandise Application

Figure 3-3 shows a sample PROM based application involving price computation for weighed merchandise. The hardware shown at the right of this illustration accommodates one of 8 weigh-in/price computation stations. Though not representing an actual PROM based application, the devices shown here convey and receive the types of data appropriate to PROM based programs. Specifically, the hardware devices consist of,

1. An electronic scale used to weigh various kinds of merchandise. The output of the scale is a dc voltage (consistent with the weight of the item being weighed) that's fed to one channel of an analog-to-digital converter module (ADV11). Because not all electronic scales are the same from station-to-station, there may be some differences between stations as to the dc voltages generated for a given weight. That is, one scale may output 2.5 Vdc for 5 kilograms of merchandise, while another scale may generate 2.6 Vdc for the same weight. Such disparities in output voltages can be compensated for by a scale calibration feature built into the application program

2. Set of four knurled knob read-in switches that are set to define the price per kilogram of the merchandise being weighed. Each of the four switches produces a BCD (Binary Coded Decimal) encoded output over four lines. Hence 16 lines are used to convey pricing information to a DRV11 module

3. Four Nixie lights to display the value of the goods after the weight-times-price product has been computed by the PROM based program. Each Nixie light receives a BCD encoded value over 4 output lines from the DRV11 module. Hence a total of 16 lines feed the "VALUE OF GOODS" display

4. Pushbutton switch that's activated by the pricing station operator after he has set the price/kilogram switches. The pushbutton switch generates the interrupt (via the DRV11 module) that initiates the price computation program

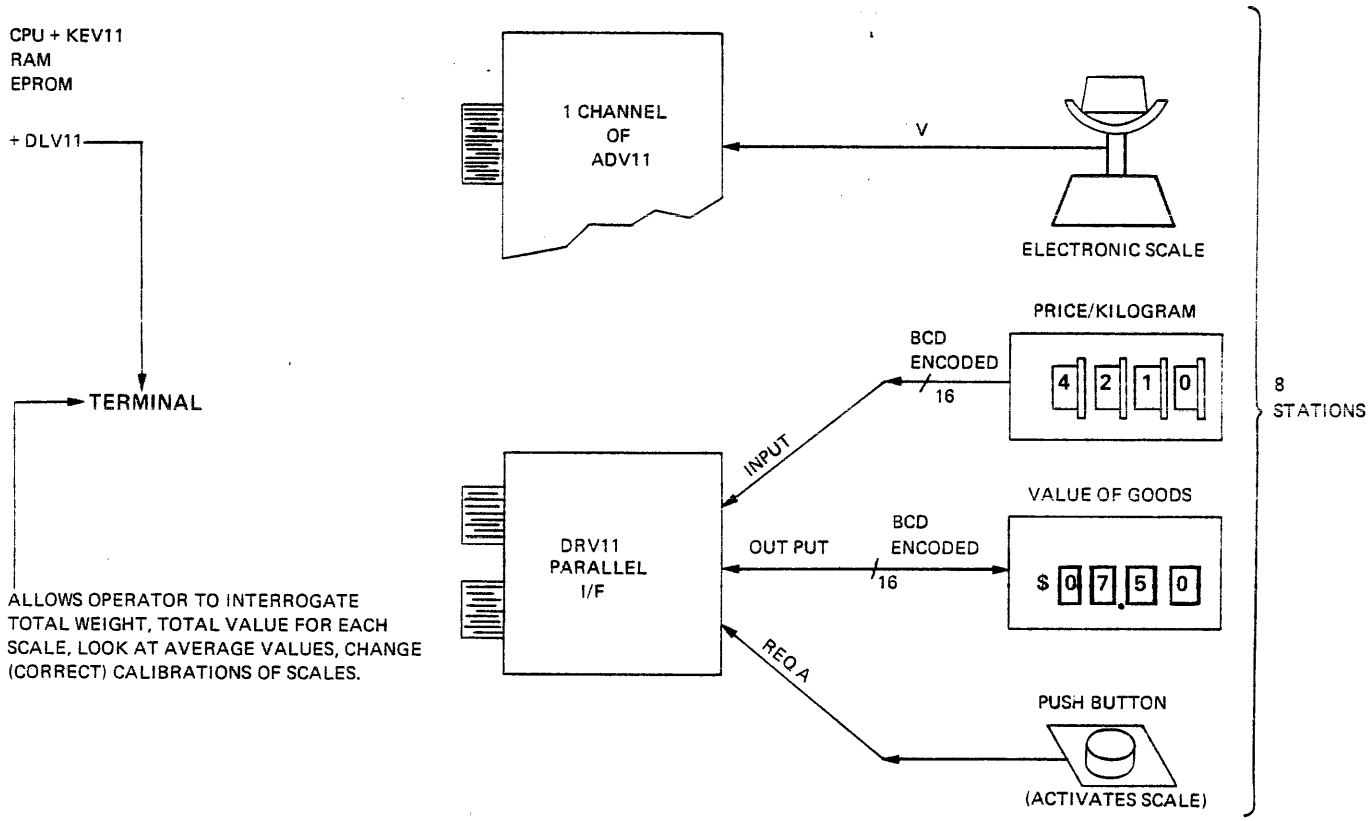


Figure 3-3 Hardware Devices for Price Computation Application

The role played by each station operator in this sample application involves placing the merchandise on the scale, setting the PRICE/KILOGRAM switches in accordance with the type of merchandise on the scale, and then activating the pushbutton.

Other features of this PROM based application permit the following types of data access from the system terminal.

1. Total weight processed by merchandise category
2. Total value processed by each scale station
3. Average value
4. Change (correct) calibration of each scale. Such changes are necessary when the dc voltage output from a scale changes over the course of time (e.g., for a fixed weight of 5 kilograms)

3.7 PROGRAMMING HINTS

This section presents software design hints for various aspects of PROM based application developement.

3.7.1 Setting Initial Conditions Prior to Enabling CPU Interrupts

Before enabling CPU interrupts, make sure that all initial conditions are set. The application startup code must be responsible for storing initial values into any required RAM variables and for conditioning of input/output interfaces before CPU priority is dropped to PR0, enabling interrupts. This is especially true of variables used by line time clock handling routines, as the clock may interrupt as soon as the CPU's priority is lowered.

3.7.2 Using LINK/X Switch When Placing Bottom of Relocatable PROM Code Below Ad

The RT-11 LINK utility stores .SAV file bitmap information into addresses 360(8) to 377(8) of memory image files. This information is used by the RT-11 system when loading background executable programs.

If you wish to use a BOTTOM address which is below 400(8), you must specify the /X switch to the RT-11 LINK utility. Refer to the LINK section of the RT-11 System User's Guide for more details on this switch.

3.7.3 Minimizing PROM Storage Requirements When Developing FORTRAN Applications

Some simple rules can aid the user in minimizing the amount of PROM storage required when developing an application using FORTRAN IV. These rules are:

1. Use the $SHORT error message module when linking the FORTRAN program. The default module will build the complete ASCII text of all run-time error messages into the PROM segment; using the short form of the messages will recover over 800 words of PROM space.

   Refer to the RT-11/RSTS/E FORTRAN IV User's Guide for information on selection of the $SHORT error module at link time.

2. If your application does not require numeric input/output conversions (FORTRAN "Formatted" I/O), but does need to print simple messages to the operator, use the SYSLIB terminal I/O calls rather than the FORTRAN TYPE and ACCEPT statements. For example,

```
CALL PRINT ("Process monitor is now running")
```

   is preferable to:

```
        TYPE 100
100     FORMAT (' Process monitor is now running')
```

   or:

```
        Type *, 'Process monitor is now running'
```

   If any TYPE or ACCEPT statements are used in the application, however, no economy is gained by replacing individual simple statements with PRINT calls.

CHAPTER 4

INSTALLATION

This chapter presents installation procedures for the PB11 hardware and software. Also discussed are procedures to be used for acceptance testing the hardware/software and methods for reporting hardware/software problems.

4.1 INSTALLING PROM/RT-11 HARDWARE

The hardware supplied for PROM/RT-11 consists of the following:

1. PROM blaster
2. Signal cable for connecting PROM blaster and DLV11 serial line unit
3. Personality cards. The PROM blaster may use a different pair of printed circuit boards to accommodate blasting of different type PROM chips. If the PROM/RT-11 is to accommodate blasting of only one type PROM chip, then a single pair of personality cards is supplied
4. Socket adapter(s). Different type PROMs may also require different types of adapter sockets for insertion in the blaster socket

4.1.1 Unpacking

No special unpacking instructions are necessary for the PROM/RT-11 hardware. Personality cards with appropriate socket adapters are to be found in separate cartons within the shipping container.

4.1.2 Installing the PROM Blaster

When removed from the shipping container, the PROM blaster may be placed on any sturdy table surface. Preferably, the table surface should be adjacent to the development system terminal to facilitate the PROM blasting procedure.

4.1.3 Connecting the PROM Blaster Signal Cable

The signal cable must now be connected between PROM blaster and the serial line unit. The cable connectors are of different design so there can be no mistake as to which end goes where. The DLV11 development module is mounted to the development system backplane along with the other modules. The PROM blaster signal cable socket is at the rear of the PROM blaster.

For this installation, switches/jumpers on the serial line unit must be set to select the following operational characteristics.

1. Eight data bits
2. No parity
3. 1 stop bit
4. No framing error halt (LSI-11 only)
5. 9600 BAUD max.

Refer to the appropriate serial line unit hardware manual for procedures on selecting these characteristics.

4.1.4 Inserting the Personality Cards and Socket Adapter

If more than one set of personality cards are supplied, the user must determine which pair of cards he wishes to use. This can be determined from the data given in Table 4-1.

After removal of the personality cards from their carton, they are to be installed in the PROM blaster. For card installation procedures, refer to the hardware manual.

Table 4-1
PROM/RT-11 Module Options and Related
PROM Blasting Hardware

PROM/EPROM Option Boards	Chip Types	Socket Adapter	Personality Cards

MRV11-AA	82S129	1035-1	1226-2F
	82S131	1035,2	1226-2F
MRV11-BA	2708	1033,4	1174-1F
	8708	1033,4	1174-1F
BDV11	2716	1366	1319-1A
	8716	1366	1319-1A

Once the personality cards are installed, the socket adapter can be inserted, This is designed so that it can only be inserted in one way; that is with the socket designation number oriented toward the top, When the socket adapter is inserted, hardware installation is complete,

4,2 INSTALLING PROM/RT-11 SOFTWARE

The PROM/RT-11 utility program is supplied on one of two distribution media as follows

1, Single density RX01 flexible diskette (floppy disk)

2, RL01 cartridge disk

The supplied disk is to be installed in its proper port, After this, the user may invoke the Keyboard Monitor COPY command to transfer the PROM/RT-11 utility program onto the development system disk, This approach assumes that the PB11 is purchased as an add-on option to an existing system, If the user purchases the PB11 as an option with a new system, he first uses SYSGEN to build a system disk, (Refer to RT-11 System Generation Manual,)

4,2,1 COPYing PROM/RT-11 Software onto System Disk

Appendix C presents a complete description of the procedures used to install PROM/RT-11 software,

4,2,2 Acceptance Testing

Following PROM/RT-11 hardware/software installation, the hardware should be exercised to ensure proper operation, The DIAGNOSE and LIST commands built into the software command repertoire provide a convenient means of checking PROM/RT-11 hardware operation,

Installation personnel can check the hardware using the following

steps

1. Invoke the PROM/RT-11 utility program using the procedures specified in Chapter 5. After invoking the utility, check that the header defining the proper PROM size appears on the terminal.

2. Invoke the DIAGNOSE command as described in Chapter 6. When the test options appear, select "Test #3. This test dialogue informs the user to mount a blank PROM to be patterned when the user types <RETURN>. The pattern entered into the PROM by the diagnose command is one that inserts a binary value of 0 into address 0, a binary value of 1 into address 1, a binary value of 2 into address 2, etc. The pattern continues until the top of the range allowable by the bit width is attained (binary 15 for 4-bit wide chips and binary 255 for 8-bit wide chips). The pattern is then repeated

3. When the message "PROM verified successfully" appears and the utility program returns to the "command" mode, installation personnel may invoke the LIST command to obtain a hard copy of the PROM contents. (See Chapter 5 for LIST command procedures.) If the listing indicates the pattern described in step 2, then the PROM/RT-11 hardware is operating satisfactorily.

4.3 REPORTING HARDWARE/SOFTWARE PROBLEMS

Procedures for reporting problems differ depending on location of user site. The procedures and differences are

1. United States. If any problems are encountered during hardware/software installation procedures, seek assistance via the LSI-11 hot line. Call toll free 800 225-9220, (in Mass. 617 481-7400 ext. 5144) or write Digital Equipment Corporation, One Iron Way, Marlborough, Ma. 01752.

2. Outside United States. If any problems are encountered during hardware software installation, contact your nearest DIGITAL sales representative for assistance.

4.4 TROUBLESHOOTING

Chapter 6 presents troubleshooting procedures for the PROM/RT-11 hardware and software.

CHAPTER 5

PROM/RT-11 OPERATING INSTRUCTIONS

5.1 INTRODUCTION

This chapter describes the PROM/RT-11 operating procedures centering around the use of the PROM/RT-11 utility instruction set. These instructions are invoked (at the development system terminal) to program or modify PROM chips after development or modification of the application program. Procedures presented in the subsequent paragraphs assume certain prerequisite actions on the part of the user. These are:

1. Personality cards appropriate to the PROM chip type have been inserted in the PROM blaster
2. The correct PROM chip adapter has been installed in the PROM blaster socket
3. A binary image file for the generated application program is available to the user
4. The POWER switch on the PROM blaster is set to ON

5.2 INVOKING THE PROM/RT-11 BLASTER PROGRAM

The PROM/RT-11 program is designed to run in either background or foreground mode. For foreground operation, the system must have a minimun of 28K memory. In systems capable of foreground operation, the PROM/RT-11 blaster program is normally run in the foreground since it does not require a high level of access to the CPU.

To invoke the PROM/RT-11 blaster program in the background mode, the user types the following at the terminal:

```
RUN DEV: PROM.SAV<RET>
```

Where DEV: defines the disk drive containing the PROM/RT-11 utility program.

To invoke the PROM/RT-11 blaster program in the foreground mode the user types the following:

FRUN DEV: PROM,REL<RET>

where DEV: defines the disk drive containing the PROM/RT-11 utility program,

FRUN may be abbreviated to FR or FRU,

If the user wishes to invoke the foreground version for operation on some other terminal in a multi-terminal system, he types:

FRUN PROM/T:n <return>

Where "n" specifies the number of a valid terminal in the multi-terminal system

Once invoked, the PROM/RT-11 blaster program responds by typing a header followed by a prompt at the terminal. The header and prompt are as follows:

PROM/RT-11 V1.0 Current PROM Size 1024 by 08

Command:

The header, besides identifying the PROM/RT-11 utility program and version, indicates the size and bit width of the PROM chip(s) to be blasted. This is because the PROM/RT-11 utility queries the PROM blaster as to type of personality cards before typing the header on the terminal. In this case, the personality cards contained in the blaster accommodate PROMs having 1024 address locations that are 8 bits wide. If the personality cards accommodating PROMs 512 by 4 bits size are inserted in the blaster then the header appears as:

The "Command:" prompt indicates that the utility program wants the user to type one of the PROM/RT-11 commands on the terminal. The commands are described in the subsequent paragraphs.

PROM/RT-11 V1.0 current PROM Size 0512 by 04
Command:

5.3 COMMANDS TO RUN PROM/RT-11

Once invoked, the PROM/RT-11 utility program responds to any of ten commands initiated by the user at the terminal. All commands except the DIAGNOSE command are described here. (The DIAGNOSE command is described separately in Chapter 6.)

PROM/RT-11 commands are described in the sequence in which they are most normally used. Hence the PROGRAM command is described first

because once the user has generated and formatted his application program, his first effort is to pattern the PROMs in accordance with the binary image map.

The range of PROM/RT-11 commands in the order of their presentation is

1. PROGRAM
2. MODIFY
3. COPY
4. LIST
5. VERIFY
6. SEQUENTIAL
7. INTERFACE
8. HELP
9. EXIT

All commands are typed in response to the "Command:" prompt printed by the PROM/RT-11 utility program. When executed, most commands initiate a running dialogue consisting of queries (typed by the utility at the terminal) and responses (given by the user at the terminal). To abort an in-process command, the user types CTRL/C. To return to the RT-11 monitor, the EXIT command must be given in response to the "Command:" prompt.

5.3.1 PROGRAM Command

The PROGRAM command is used to enter an application program into a set of PROM chips. The application program is currently contained in the input file created by the user with the RT-11 LINK utility program (Refer to Chapter 3). The LINKage editor map produced by the LINK command must be available to the user in order to execute the PROGRAM command. This map indicates the address assignments for the input file and the PROM/RT-11 utility must be informed of the highest pure code address during execution of the PROGRAM command.

Any user wishing to blast selected chips (rather than an entire program set) should use the MODIFY command described later in this section.

The PROGRAM command generates a number of queries and directives to the user throughout the course of its execution. Such queries are necessary because the PROM/RT-11 utility needs certain information in order to properly execute the PROGRAM command. Queries displayed by the PROGRAM command require the following information from the user:

1. Does the PC module being used to house the PROM chips invert (complement) the data or address values being supplied to it.

2. What is the name of the input file containing the instruction to be blasted?

3. What is highest address of pure code in the input file? The user can determine this from the map produced by the LINKage editor.

Once the final address is entered at the terminal, the actual PROM blasting process begins. During this phase, additional queries and directives are displayed to the user at the terminal. The dialogue and procedures for blasting the PROMs are described in a later paragraph (Blasting the PROMs).

An additional feature of the PROGRAM instruction is that it automatically verifies that the correct data is written into each PROM chip.

5.3.1.1 Invoking the PROGRAM Command - To invoke the PROGRAM command, the user simply types P (or PROGRAM) and <RETURN> in response to the "Command:" prompt. An example of the process is as follows:

```
Command: P[ROGRAM]<RET>
Do you want inverted data (Y or N)?
```

Once the PROGRAM command is invoked, the PROM/RT-11 utility immediately responds by asking whether the user wants inverted data. Considerations regarding this query are discussed below.

5.3.1.2 Working with Inverted Data or Inverted Address InPuts - Certain printed circuit boards may invert the data or address information supplied to them. This is the case for the MRV11-AA module which addresses PROM chips in low (rather that high) active address bits. (This aspect of MRV11-AA addressing is discussed in the Digital Microcomputer Handbook). If a user is working with an MRV11-AA module and notifies the software of the inverted address condition, PROM/RT-11 inverts the address bits so that the PROMs are blasted in a low-to-high order address sequence rather than vice versa.

It is up to the user to know whether or not the PC module he's using complements either data or address information. Otherwise he cannot give proper response to the PROM/RT-11 software queries described below.

When the PROGRAM command is invoked PROM/RT-11 responds with two queries in sequence. These are:

```
Do you want inverted data (Y or N)?
```

Do you want inverted addresses (Y or N)?

After each Y or N response, the user presses the <RET> key. The below example shows a condition where the user requests both data and addresses as being inverted

Do you want inverted data (Y or N)? Y<RET>

Do you want inverted addresses (Y on N)? Y<RET>

Once the user has defined the data and address format, the PROM/RT-11 utility wishes to know the name of the input file. This is discussed next.

5.3.1.3 Specifying the Input File to be Used in the PROM Blasting Process - The next query displayed by the PROM/RT-11 utility is

Name of input file (DEV:FILNAM.TYP)?

To this, the user responds with the name of the .SAV image file containing the application program to be blasted into PROM chips. If the device (DEV:) is not specified, the system default storage device (DK:) is used. Also if no file extension (.TYP) is given, an extension of .SAV is assumed. The below example indicates a case where the user has specified a demonstration program as the input file

Name of input file (DEV:FILNAM.TYP)? DEMPGM<RET>

5.3.1.4 Specifying the Highest Address in the Range - The next request by the PROM/RT-11 utility is for the user to supply the highest address of pure code contained in the input file.

CAUTION

The last address supplied to the PROGRAM command must be the highest byte address to be placed in PROM.

The upper pure code limit is available from the LINKage map produced by the LINK editor (Refer to Chapter 3). Once informed of the pure code upper limit, the PROM/RT-11 utility is able to determine the number of PROM chips to be blasted. For example, if the user is working with 1024 X 8-bit PROMs and specifies a pure code upper limit of 017776(8), the utility calculates that a total of 8 PROM chips (4 low byte and 4 high byte) are to be blasted.

When the last PROM in the entire range has been blasted, the utility knows that it is time to return to the command mode. The PROM/RT-11 request the upper limit of pure code with the following query:

```
Final program address (octal)?
```

The user now refers to his LINK map to determine the highest address of pure code to be entered into PROM. The pair of maps shown below indicate the pre boundary and post-boundary code layout for a sample program.

Pre-Boundary Code Layout

```
LINK/BOTTOM:400/EXECUTE:MAIN/MAP:TT: MAIN,CLOCK
RT-11 LINK  V05.02      Load Map
MAIN  .SAV      Title:  MAIN    Ident:          /B:000400

Section  Addr   Size    Global  Value   Global  Value   Global  Value

. ABS.  000000  000400   (RW,I,GBL,ABS,OVR)
ROM     000400  000376   (RW,I,LCL,REL,CON)
                       INIT    000752
RAM     000776  000210   (RW,D,LCL,REL,CON)
                       $STACK  001176  TICKS   001176  SECS    001200
                       MINS    001202  HOURS   001204  SECS

Transfer address = 000001, Hight limit = 001206 =   323. words
```

Post-Boundary Code Layout

```
 LINK/BOTTOM:400/EXECUTE:MAIN/MAP:TT:/BOUNDARY:20000 MAIN,CLOCK
Boundary section? RAM
RT-11 LINK  V05.2       Load Map
MAIN  .SAV      Title:  MAIN    Ident:          /B:000400

Section  Addr   Size    Global  Value   Global  Value   Global  Value

. ABS.  000000  000400    (RW,I,GBL,ABS,OVR)
ROM     000400  017400    (RW,I,LCL,REL,CON)
                       INIT    000752
RAM     020000  000210   (RW,D,LCL,REL,CON)
                       $STACK  020200  TICKS   020200  SECS    020202
                       MINS    020204  HOURS   020206

Transfer address = 000001, High limit = 020210 =  4164. words
```

5.3.1.5 Blasting the PROMs - When the user presses the <RETURN> key (after specifying the upper limits of pure code), the PROM/RT-11 utility responds by instructing the user to mount the PROM chip for the first address segment. Under conditions where 1024 X 8-bit PROM chips are being blasted, the displayed directive to the user is as follows:

```
Mount PROM for address 000000-003776, bits 07-00,
and type<RETURN>:
```

At this time, the user mounts the first PROM chip in the blaster adapter socket, Three things should be borne in mind when mounting a PROM chip:

1. The PROM chip must be blank

2. The chip must be oriented so that dimpled cutout faces the top of the adapter socket,

3. The chip should be labeled with the segment address,

Identifying chips with a label aids the user if he later wishes to MODIFY or erase a particular chip, When working with 1024X 8-bit PROMs a suitable label for the first PROM might be 1K,LB - indicating that it contains the low byte for the first 1024 locations, Similarly, the second chip might be labeled 1K,HB meaning that it contains the high byte for the first 1024 locations,

CAUTION

In the case of EPROMs, the label should not be placed over the chip window as it may adversely affect chip erasability,

If the user inserts a non-blank chip in the adapter socket and then presses <RETURN>, the utility program responds with the following error message and directive:

```
?PROM-F-FROM is not blank
Mount PROM for address 000000-003776, bits 07-00,
and type <RETURN>:
```

In such a case, the user should check to make sure that the PROMs he intends to pattern are blank and then mount a blank chip,

When the first blank chip is programmed successfully, the following message is generated:

```
PROM for address 000000-003776, bits 07-00, has been programmed
Mount PROM for address 000001-003776, bits 15-08, and type
<RETURN>:
```

This informs the user that the first PROM chip has been programmed successfully, The first chip can now be removed from the blaster adapter socket and inserted in the proper location on the PROM memory board,

The user should now mount the second chip for the blasting process, Again, the PROM chip should be labeled with the segment address, When the <RETURN> key is pressed and the second chip has been programmed,

the below message appears:

```
PROM  for address  000001-003776, bits 15-08, has been programmed
Mount PROM for address 004000-017776, bits 07-00, and type
<RETURN>:
```

The blasting process continues until all PROM chips for the application program have been blasted. After the last chip, the PROM/RT-11 utility returns to the command mode.

Complete PROM blasting sessions for an MRV11AA memory board and an MRV11BA memory board are given in the next two paragraphs.

5.3.1.6 Programming Four Bit Wide PROMs for an MRV-11AA Module - This paragraph presents an example of using the PROGRAM Command for blasting a set of 512 X 4-bit wide PROMs. Specifically, the example shows:

1. Invoking the PROM/RT-11 utility program
2. Execution of the PROGRAM Command
3. The terminal/user dialogue for the entire blasting process

The example is given below:

```
,RU DX0:PROM11
PROM/RT-11   V1.0     Current PROM Size 0256 by 04

Command: P<RET>
Do you want inverted data (Y or N)? N<RET>
Do you want inverted addresses (Y or N)? Y<RET>
Name of input file (DEV:FILNAM.TYP)? DX1:MAIN.SAV<RET>
Final program address (octal)? 775<RET>
Mount PROM for address 000000-000774, bits 03-00, and type <RETURN>:
Mount PROM for address 000000-000774, bits 07-04, and type <RETURN>:
PROM for address 000000-000774, bits 07-04, has been programmed
Mount PROM for address 000000-000774, bits 11-08, and type <RETURN>:
PROM for address 000000-000774, bits 11-08 has been programmed
Mount PROM for address 000000-000774, bits 15-12, and type <RETURN>:
PROM for address 000000-000774, bits 15-12 has been programmed

Command:
```

5.3.1.7 Programming Eight Bit Wide PROMs for an MRV11-BA Module - This paragraph presents an example of using the PROGRAM Command to blast a set of 1024X 8-bit wide PROMs. The example follows:

```
,RU DX0:PROM11
PROM/RT-11   V1.0     Current PROM Size 1024 by 08
```

```
Command: P<RET>
Do you want inverted data (Y or N)? N<RET>
Do you want inverted addresses (Y or N)? N<RET>
Name of input file (DEV:FILNAM.TYP)? DX1:MAIN.SAV<RET>
Final program address (octal)? 775<RET>
Mount PROM for address 000000-000774, bits 07-00, and type <RETURN>:
PROM for address 000000-000774, bits 07-00, has been programmed
Mount PROM for address 000000-000774, bits 15-08, and type <RETURN>:
PROM for address 000000-000774, bits 15-08, has been programmed

Command:
```

5.3.2 MODIFY Command

The MODIFY Command like the PROGRAM Command can be used to create PROM chips. The difference is that the MODIFY Command allows the user to re-pattern selected chips rather than program an entire set. In general, there are four conditions where a user may wish to invoke the MODIFY Command. These are:

1. A section of an application program is altered and the change affects only a certain EPROM chip. Once the affected EPROM is erased, the MODIFY Command can be used to blast a new pattern consistent with the change in the application program.

2. Some code additions are being made to the high end of an application program and the last PROM chip in the set was only partially consumed in the prior PROM blasting session. Unlike the PROGRAM Command, MODIFY allows the user to write into non blank chips. Consequently if half the locations in a chip were left unused during the initial PROGRAM blasting session, MODIFY can be used to pattern the remaining PROM locations.

3. A single location on a pre-blasted PROM chip requires change. Possibly, the pattern originally blasted into the chip location is one that can accept the specific change desired by the user. In the unprogrammed state, certain PROM chips contains all ONEs. For this type chip, the binary pattern introduced during the chip blasting process is achieved by writing ZEROs into the desired bit positions. Hence if a binary six is written into an address of a four bit wide chip, the blasting process effects a change of from 1111(2) to 0110(2) in the addressed location. Although already blasted to contain a binary 6, this location could be modified (re-blasted) to contain a binary 4, a binary 2 or all zeros. Such changes can be effected because each involves a process of changing ONEs to ZEROs. The contents of the PROM location cannot be modified to contain a binary 8 because for this type chip it is not possible to change ZEROs to ONEs. If the user wishes to alter the contents of a single location, the software checks the contents of that

location and informs the user whether such a change is possible.

4. The user wishes to enter a value into some high order area of the overall PROM address range. Unlike the PROGRAM Command which always assumes a zero starting address value for the PROM blasting process, the MODIFY Command queries the user as to the starting address for his desired blasting/re-blasting area. (Note that the PROGRAM Command only asks for the final program address.) If a user wishes to enter a bootstrap program into the high order area of PROM, he can do so with the MODIFY Command because this command allows him to select a high order starting address. The PROGRAM command does not.

There are certain constraints that the user must be aware of when using the MODIFY Command. During the MODIFY Command dialogue, the PROM/RT-11 queries the the user as to starting and ending address of the PROM chip area he wishes to modify.

NOTE

PROM/RT-11 always constructs 32 byte data records for transfer to the programmer (blaster). locations within the 32 byte boundaries in the PROM but outside the starting and final addresses of the file will be filled with the unprogrammed state of the given PROM.

For the above reasons, modification of a PROM must be done in 32 byte blocks, i.e., modification addresses to the MODIFY command must be:

1. Starting address is the first byte in the 32 byte boundary.

2. Final address is the last byte in the 32 byte section. There is one exception to this which occurs for the uppermost 32 byte block in pure code. In this case, the last byte of meaningful code can fall anywhere in the 32 byte range for the uppermost 32 byte block only, the last meaningful byte in the 32 byte range can be given as the final address to the MODIFY command.

Figure 5-1 shows how bytes are taken from an input file and stored in the 32 byte record when blasting an 8-bit wide chip. The 32 bytes taken for transfer to the low byte PROM are taken from byte addresses 0 through 62. Similarly, the bytes for blasting the high byte PROM are taken from addresses 1 through 63. Consequently if the user wishes to modify the uppermost 32 byte segment in 4K of PROM he specifies a starting address of 017700 and an ending address of 017776. Figure 5-2 shows the input file to PROM chip relationship when attempting to modify the contents of a single address in PROM. The MODIFY Command first compares the changed input file location with the contents of the pre-blasted PROM chip. If the pre-blasted

contents are such that the location can accept the change (see preceding description under item 3) then the software informs the user that the change is possible, If the location cannot accept the change then the user is so informed and he must take other steps such as erasing the chip (i,e, if he is working with EPROMs)

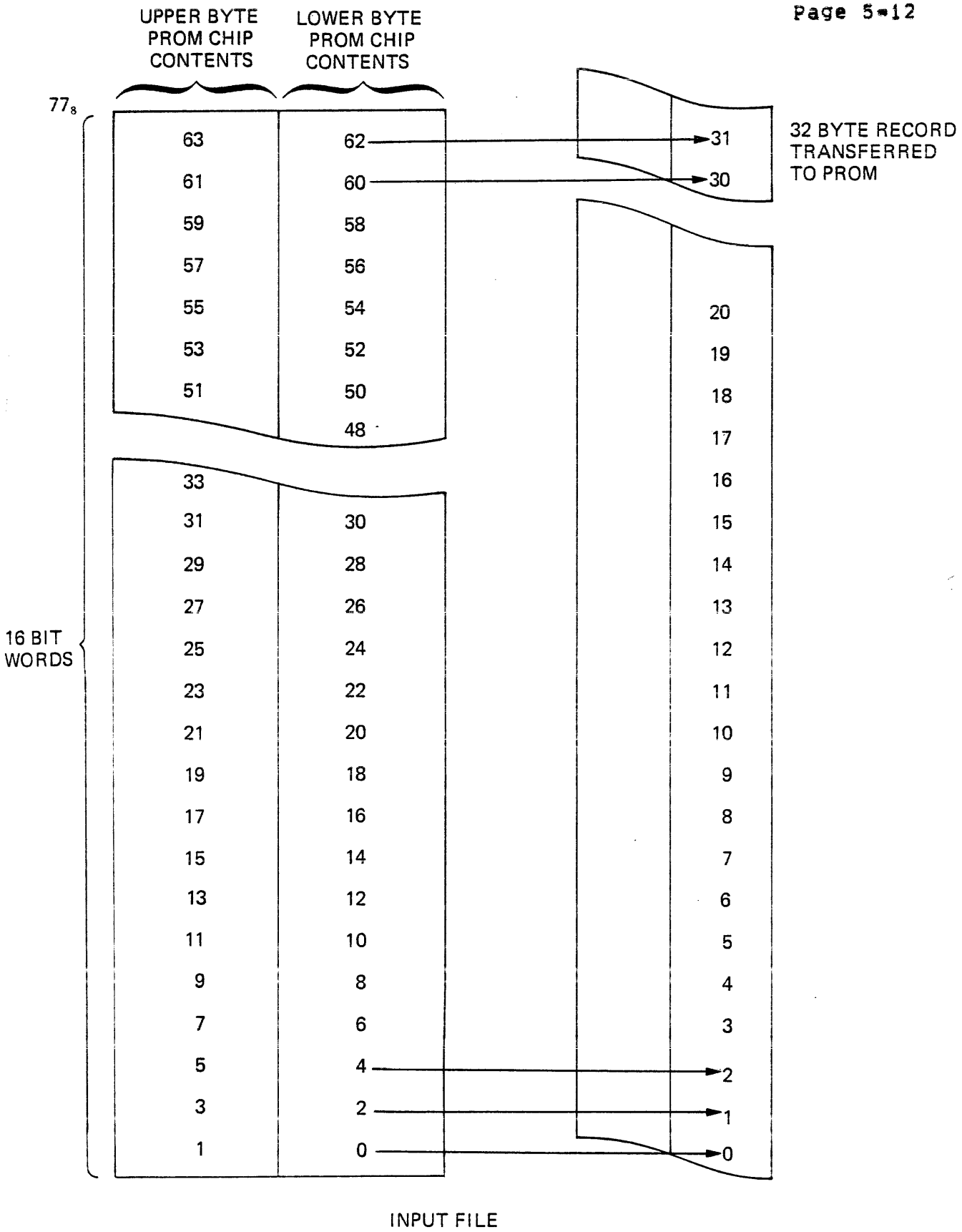


Figure 5-1 Input File to PROM Chip Transfer Pattern for an Eight Bit Wide PROM

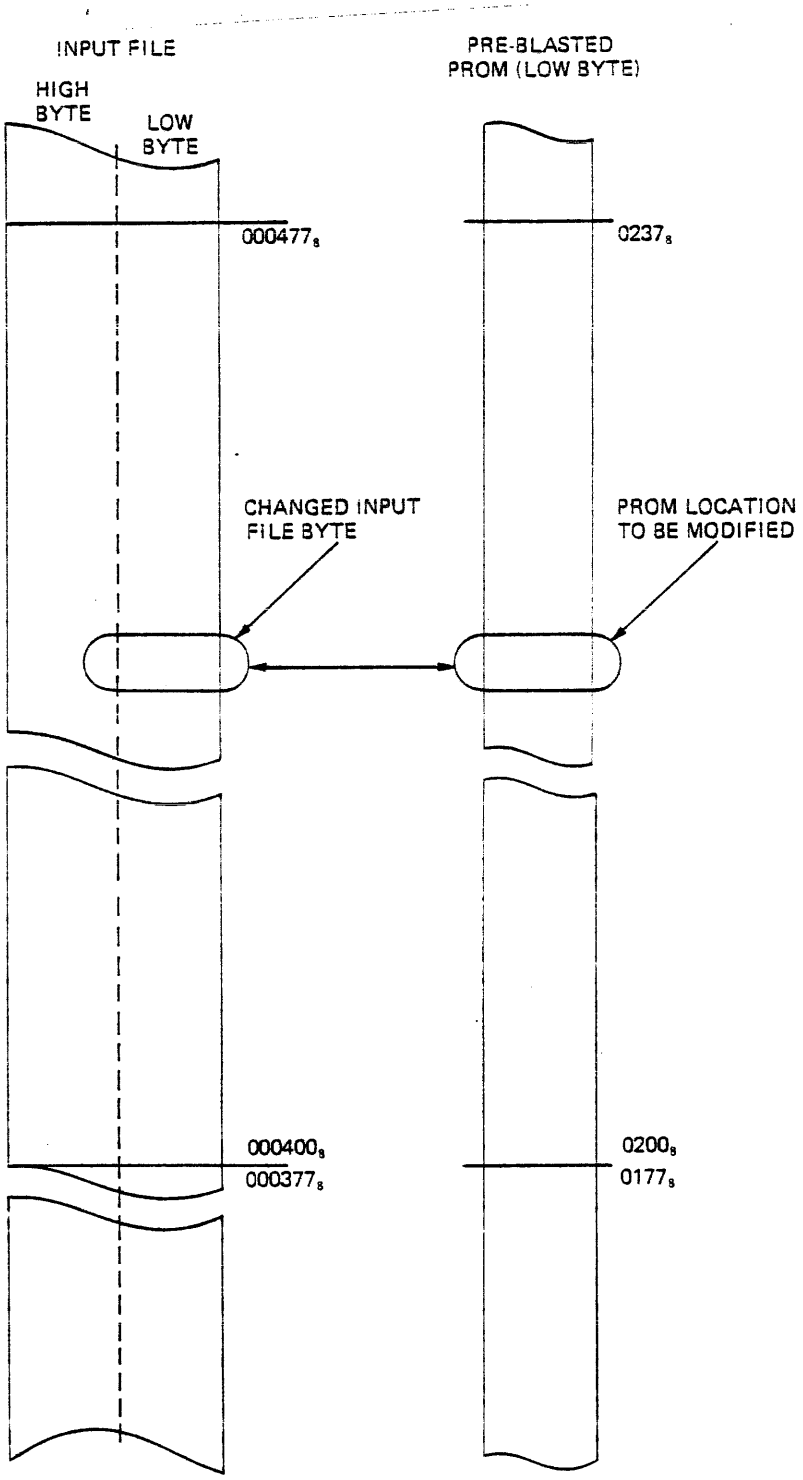


Figure 5-2 Relationship of Input File to Pre-Blasted PROM When Attempting to Modify an 8-Bit Wide PROM Chip

5.3.2.1 Invoking the MODIFY Command - To invoke the MODIFY Command, type the following at the terminal in response to the "Command:" prompt:

```
Command: M[ODIFY]<RET>
```

When the <RETURN> key is pressed, the PROM/RT-11 utility initiates the dialogue for the reblasting process.

5.3.2.2 MODIFY Command Dialogue - The first three queries generated by the MODIFY command have to do with working with inverted data/addresses and naming the input file. Examples of these queries are shown below:

```
Do you want inverted data (Y or N)? N

Do you want inverted addresses (Y or N)? N

Name of input file (DEV:FILNAM.TYP)?
```

For a description of these queries, refer to paragraphs 5.3.1.2 and 5.3.1.3 .

If the user errs in typing the file name or types in a nonexistent file, the following error message occurs

```
?PROM-F-Illegal file specification
```

Immediately after the error message, the utility program repeats the query asking for the file name. The example below shows a case where a demonstration program is named as the input file.

```
Name of input file (DEV:FILNAM.TYP)? DEMPGM<RET>
```

When the <RETURN> key is pressed, the utility program asks for starting and ending addresses of the PROM area to be modified. Examples for a 1024 x 8-bit wide chip are given below:

```
Starting program address (octal)? 017700<RET>

Final program address (octal)? 017776<RET>
```

To the first query, the user responds with the starting address (on a 100(8) boundary) of the PROM area to be modified. On typing <RETURN>, the utility program asks for the final program address. When this is given and the <RETURN> key is pressed, the software asks the user if these chip addresses in PROM are to be modified. This reassures the user that the utility program recognizes the specified address areas. An example of this query and response is:

```
Do you wish to work with addresses 017700-017776, bits 07-00
(Y or N)? Y<RET>
```

The user indicates that this is the desired address area by typing Y (yes) and pressing the <RETURN> key. At this time, the utility program directs the user to insert the PROM in the blaster adapter socket as shown in the below example,

```
Mount PROM for address 017700-017776, bits 07-00,
and type <RETURN>:
```

If the user mounts a non blank chip, that already contains a pre-blasted pattern that can't be changed then the following error message and directive result:

```
?PROM-F-PROM is not blank and cannot be reprogrammed with
this data

Mount PROM for address 017700-017776, bits 07-00, and type
<RETURN>:
```

When a non blank chip having changeable pre-blasted code in the addressed area is mounted, then the following message results

```
?PROM-W-PROM is NOT blank but can be written with this data
Continue (Y or N)?
```

If the user intends to overwrite the addressed area, he then types Y or YES and presses the <RETURN> key.

When the blasting operation is completed for the mounted chip, the utility program generates the following message:

```
PROM for address 017700-017776, bits 00-07, has been programmed
```

The PROM chip in the blaster socket may now be removed. The PROM/RT-11 utility program now automatically sequences to the next chip in the PROM set, (i.e, the chip containing the high byte). When all chips for the specified address area have been reprogrammed, then the utility program automatically exits to the command mode.

5.3.3 COPY Command

The COPY command is designed to let the user make multiple copies of a pre-programmed PROM chip.

5.3.3.1 Invoking the COPY Command - To invoke the COPY Command, type the following at the terminal in response to the "Command:" prompt:

```
Command: C[OPY]<RET>
```

On detecting the COPY Command, the PROM/RT-11 utility program initiates the terminal - to - user dialogue discussed next.

5.3.3.2 COPY Command Dialogue - The first directive produced by the COPY command is to tell the user to mount the master PROM. That is the PROM to be duplicated. An example of this directive is:

Mount master PROM and type <RETURN>:

At this time, the user inserts the PROM chip to be copied in the blaster socket. He then presses the <RETURN> key and a slight pause occurs. After the pause, the user is told to remove the master PROM chip with the below message:

Remove master PROM and mount blank PROM; type <RETURN>:

When this message appears, the master PROM chip is removed from the blaster socket and replaced with a blank PROM chip that will become the first copy. After inserting the blank PROM, the user types <RETURN>. The utility program now needs to know how many copies are desired and generates the following query:

How many copies (decimal)?

The user responds by typing in the number of copies he wishes to make. The COPY command is designed to permit any number of copies in a range from 1 to 99. If more than 99 copies are required, the COPY command should be invoked a second time or as many times as necessary. The user should make sure that he has enough blank PROM chips for the number of copies specified. After entering the number of copies, the user types <RETURN>. This begins the copying process. If the user inadvertantly mounts a non blank PROM chip the following error message and directive result

?PROM-F-PROM is not blank

Remove current PROM and insert blank PROM, type <RETURN>:

In such a case, the user should remove the non blank PROM and replace it with a blank or erased PROM. When the <RETURN> key is pressed, copying will continue with the blank PROM.

When creation of a copy is complete, the PROM/RT-11 generates the following message:

Remove copy number NN and mount blank PROM, type <RETURN>:

Where NN will be replaced by the number of the copy just completed.

The user now removes the newly created PROM chip and replaces it with a blank chip. He may wish to label the duplicate chip with the copy number. With the new blank chip installed, the user continues by pressing the <RETURN> key and another copy is created.

5.3.4 LIST Command

The LIST command is used to obtain a listing of the contents of a particular PROM chip. The listing can be produced on the terminal or a line printer at the user's option. Other options made available by the LIST command are

1. Listing format. The LIST command allows the user to select binary, octal or hexadecimal format
2. Identification. The LIST command allows the user to enter a listing identifier (up to 50 characters) at the top of the listing

Chip addresses appear every 20(8) locations and as chip addresses, they bear no relation to input file addresses

5.3.4.1 Invoking the LIST Command

To invoke the LIST command, type the following at the terminal in response to the "Command:" prompt

```
Command: L[IST]<RET>
```

When the <RETURN> key is pressed it initiates the command dialogue discussed next

5.3.4.2 LIST Command Dialogue

The utility program first asks the user if he wishes to place a text identifier at the head of the listing. It does so with the following query

```
Do you wish to put an identification code on the listing?
(Y or N)
```

Typically, the user assigns an identification code that describes the current application. That is the application in which the PROM chip is used. If the user desires a text description, he types Y or Yes and then presses the <RETURN> key. When the PROM/RT-11 utility recognizes that the user wants an identification for the listing, it types the following message

```
You may input up to 50 characters terminated by RETURN
Identification:
```

At this point, the user types in the desired description following the word "Identification:"

When the <RETURN> key is pressed, the next query in the dialogue is:

```
Do you wish the listing on this terminal (Y or N)? Y
```

If the listing is to be on this terminal, the user responds with Y or YES followed by <RETURN>. If the listing is to be on a line printer

or entered into a disk file, the user types N or NO followed by <RETURN>, A NO reply causes the software to generate the following question

Name of output file (DEV:FILNAM.TYP)?

The user answers with the RT-11 file specification for the desired output file followed by <RETURN>

The software now needs to know the desired format for the listing. It requests this data by printing the following

Which listing format (HEX, OCTAL, or BINARY)?

The user must select one of the three formats for listing the PROM chip contents. The format chosen affects only the format of the PROM chip contents; PROM chip address are always listed in octal. The example below shows selection of the hexadecimal format

Which listing format (HEX, OCTAL or BINARY)? H[EXADECIMAL]<RET>

When the user types <RETURN> after defining the listing format, the software types the following directive

Mount the PROM to be listed and type <RETURN>:

The user now mounts the chip to be listed, When the <RETURN> key is pressed, the listing operation begins immediately. After the last locations in the PROM have been listed, PROM/RT-11 returns to the command mode. The beginning and ending portions of a typical PROM listing appear below

```
PROM - 11 XX-XXX-XX XX:XX:XX Page 01
Identification:  DEMOPGRM


ADDRESS+00   +01   +02   +03   +04   +05   +06   +07
000000  FF   17   45   1F   45   1E   FF   FF
        FF   FF   7F   1F   95   FF   FF   FF
000020  5F   FF   FF   FF   FF   FF   8D   FF
        2F   1F   6F   7F   FF   FF   FF   FF


001740  FF   FF   FF   FF   FF   FF   FF   FF
        FF   FF   FF   FF   FF   FF   FF   FF
001760  FF   FF   FF   FF   FF   FF   FF   FF
        FF   FF   FF   FF   FF   FF   FF   FF


Command:
```

5.3.5 VERIFY Command

The VERIFY command allows the user to check that a set of PROM chips contain the same binary pattern as originally blasted from an input file. It can also be used to compare the contents of the PROM with a master PROM. In the course of time, ambient light or other environmental factors may cause one or more chips in a set to "drop a bit". If it is not known whether this has occurred, the user can invoke the VERIFY Command to check blasted bit patterns against input file contents.

It is not necessary to use VERIFY after the PROGRAM or MODIFY commands since the latter two commands automatically verify chip contents as part of their execution sequence.

5.3.5.1 Invoking the VERIFY Command - To invoke the VERIFY Command, type the following at the terminal in response to the "Command:" prompt

```
Command: V[ERIFY]<RET>
```

Where the <RETURN> key is pressed, the utility program initiates the dialogue described next

5.3.5.2 VERIFY Command Dialogue - When the user types the VERIFY command, the following question appears:

```
Do you want to VERIFY against a master PROM (Y or N)?
```

A yes answer to this question followed by <RETURN> causes PROM/RT-11 to prompt:

```
Mount master PROM and type <RETURN>:
```

The user now mounts the PROM in the blaster socket that serves as the standard or master PROM. After the chip has been positioned in the socket the user types <RETURN>. After a brief pause the PROM/RT-11 software types the following directive:

```
Mount PROM to be VERIFIED and type <RETURN>:
```

The user now removes the master PROM from the blaster socket and inserts the PROM chip to be compared. When the <RETURN> key is pressed, the software compares the contents of the two PROMs. If the PROMs compare successfully then the following message is typed on the terminal.

```
PROM verified successfully
```

If the chip to chip verification failed then the following message will appear.

?PROM-F-PROMfailed to verify

At this time the software will return to the COMMAND mode. If the user answers N or NO to the PROM-to-PROM verification request, the software assumes the user wishes to verify a PROM against an input file and initiates a second dialogue. The next three queries produced by the PROM/RT-11 software have to do with inverted data/addresses and naming of the input file for the verification process. Considerations regarding these queries are described under the PROGRAM Command in paragraphs 5.3.1.2 and 5.3.1.3.

After the user identifies the input file (to be compared against the PROM set) and types <RETURN>, the PROM/RT-11 utility needs to know the address range. The software asks for the starting address as follows:

Starting address (octal)?

The user responds with the 6 octal digits which specify the address of the first location in the application program to be compared with the selected PROM chip(s). When the starting address has been given, the user types <RETURN> causing the software to ask for the ending address as follows

Final address (octal)?

The user now enters the last address in the application program to be compared. After the user presses <RETURN>, the software confirms the address range by asking the following

Do you wish to work with addresses nnnnnn-nnnnnn,
bits nn-nn (Y or N)?

If the address range is the same as that entered by the user he types Y or YES followed by <RETURN>. Assume a user wishes to verify the content of addresses 000376 - 001077 and has entered this value. In such a case, the query generated by the software is:

Do you wish to work with addresses 000376-001077,
bits 07-00 (Y or N)? Y

The user now types <RETURN> and the software responds with the following directive,

Mount PROM for address 000376-001077, bits 07-00,
and type<RETURN>:

The user now mounts the correct PROM chip in the blaster socket. When the chip is properly mounted, the user types <RETURN>. If the data in the selected PROM chip agrees with the application file contents, the following message will be printed:

PROM for address 000376-001077, bits 07-00, agrees with file

At this point, the verified PROM should be removed from the blaster socket. Operation will continue automatically when the next PROM in

the set is inserted and the <RETURN> key is pressed.

If the PROM chip fails to verify then the below message is generated by the software

```
?PROM-F-PROM failed to verify
Do you want a detailed comparison of discrepancies (Y on N)?  Y
```

If the user simply wishes to know whether there was no match between input file and chip, he can type N or NO and remove the chip. The utility program will automatically sequence to the next chip in the set. If the user wishes a detailed bit-for-bit comparison he types Y or YES followed by <RETURN>. To this response, the software prints out input file and PROM contents in the format shown below

```
The following discrepancies were noted:

File Address   File Data      PROM Data

   000376      11001110       11111111
   000400      11101000       11111111
   000402      10001110       11111111
   000404      11100110       11111111
   000406      10010111       11111111
   000410      11111110       11111111
   000412      00010001       11111111

   001066      11110111       11111111
   001070      11101010       11111111
   001072      11110111       11111111
   001074      11100110       11111111
   001076      10000101       11111111
```

After the contents of the last address have been listed, the software continues the dialogue by asking the user if the wants to verify the next chip. An example of this query is

```
Do you wish to work with addresses 000376-001077, bits 15-08
(Y or N)?
```

If the user wishes to verify the high byte, he responds by typing Y followed by <RETURN>. To this, the software responds with the directive:

```
Mount PROM for address 000376-001077, bits 15-08,
and type <RETURN>:
```

In this way, the verification process continues for the entire chip set

5.3.6 SEQUENTIAL Command

The SEQUENTIAL command alters or re-defines the operational mode of three other PROM/RT-11 commands, namely the PROGRAM, MODIFY and VERIFY commands. The change revolves around the manner in which the word slices are taken from the memory image file for insertion into the 32-byte record that is shipped to the PROM blaster. When the operational mode of the PROGRAM command is altered by the SEQUENTIAL command, it allows a user to enter bytes one after another into PROM. This is in contrast to the way say a 1024 X 8-bit chip is normally blasted with the PROGRAM command. Chips of this size usually receive either odd or even numbered bytes depending on whether the low byte or high byte is being blasted. This can be seen from Figure 5-1. Bytes 0,2,4 etc. are entered into one chip as the low byte while bytes 1,3,5 etc, are entered into another chip as the high byte. If a user wishes to blast PROM chips one byte after another (1,2,3, etc.) he may select the SEQUENTIAL Command to achieve this result

5.3.6.1 Invoking the SEQUENTIAL Command - To invoke the SEQUENTIAL Command, the user types the following in response to the "Command:" prompt

```
Command: S[EQUENTIAL]<RET>
```

This causes the PROM/RT-11 to print the following message when <RETURN> is pressed:

```
you are now operating in SEQUENTIAL MODE
This MODE modifies the following commands:

        1. PROGRAM

        2. MODIFY

        3. VERIFY

Command:
```

This message indicates that the user can select any of the three commands (PROGRAM, MODIFY or VERIFY) for operation in the sequential mode. That is, if the user wishes to use the PROGRAM command in the sequential mode, he now types the following in response to the "Command:" prompt:

```
Command: P[ROGRAM]<RET>
```

The PROM/RT-11 software now queries the user to see if wishes to continue operation in the sequential mode. The below example shows a condition where the user has responded Y (YES) to the query. The PROM/RT-11 utility now conducts basically the same dialogue as that described for the PROGRAM command. This is apparent from the sample dialogue below. Note however there is one difference. That is, the bit field indicated for the PROM is now 15-00 instead of 07-00 or

```
15-08

     Do you want want to continue in sequential mode (Y or N)
     ?  Y <RET>

     Do you want inverted data (Y or N)?  N<RET>

     Do you want inverted addresses (Y or N)?  N<RET>

     Name of input file (DEV:FILNAM.TYP)?  DENPGN/N<RET>

     Final program address (octal)?  000377<RET>
     Mount PROM for address 000000-000377, bits 15-00,
     and type <RETURN>:

     PROM for address 000000-000377, bits 15-00, has been programmed
```

When invoking the MODIFY command in the sequential mode, the user must be careful when specifying starting address boundaries. This is because the utility program deals in minimum 32 word blocks when sending data to the PROM blaster (see discussion of MODIFY Command.) Hence when modifying 8-bit wide chips during sequential mode, the user should specify starting addresses in multiples of 40(8).

Figure 5-3 indicates the transfer pattern for an 8-bit wide chip. When modifying during the sequential mode, the PROM transfers slice 0 followed by slice 1 followed by slice 2 and so on. Hence boundaries in multiples of 40(8) (32 bytes/16 words) convey 32 slices of data.

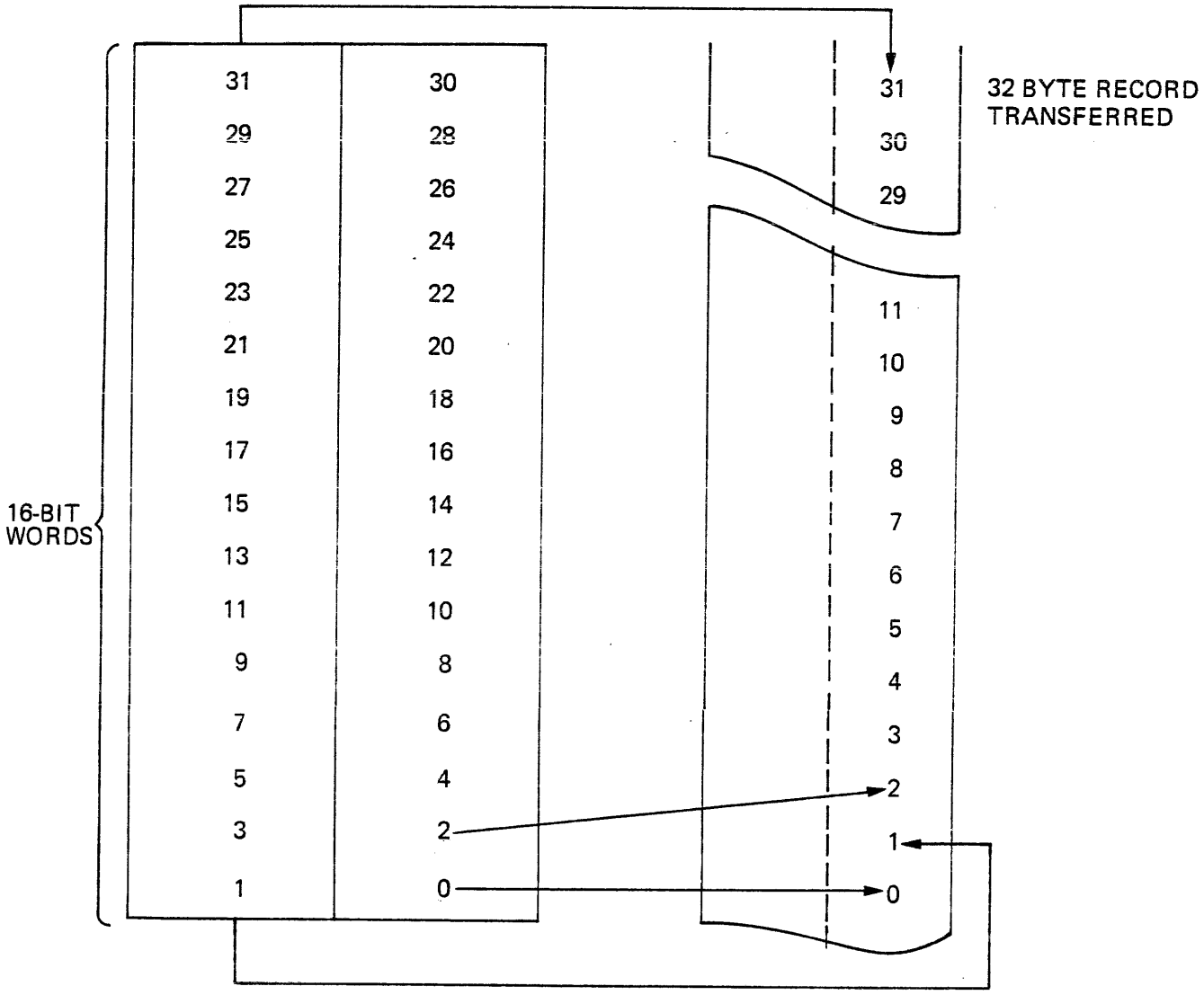


Figure 5-3 Word Slices Transferred to an Eight Bit Wide PROM Chip During Sequential Mode (MODIFY Command) Operation

When modifying four bit wide chips in the sequential mode, boundaries are on a multiple of 20(8) because the slices are taken from 16 bytes (8 words), See Figure 5-4

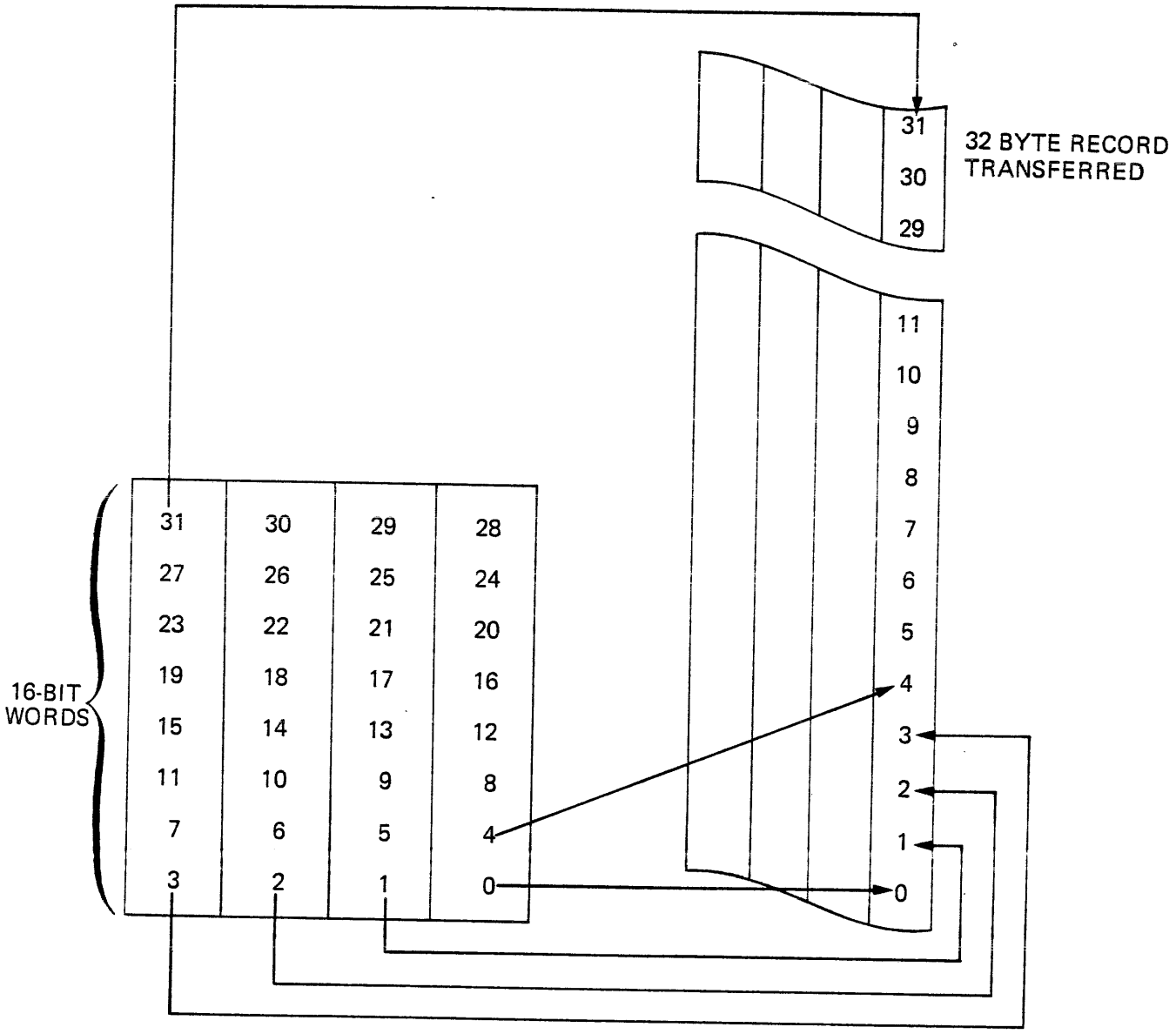


Figure 5-4 Word Slices Transferred to a Four Bit Wide PROM Chip During Sequential Mode (MODIFY Command) Operation

5.3.7 INTERFACE Command

The I[NTERFACE] command is used to change the serial inferface which the software package will use to communicate with the PROM blasting hardware. When the I[NTERFACE] command is invoked, the following response appears on the terminal:

```
Current CSR = 176510, vector = 310
```

Where the appropriate octal numbers replace 176510 and 310. The program then displays the question:

```
Do you wish to change the CSR (Y or N)?
```

If the value displayed for the CSR address is correct, respond with "N" or "NO" followed by RETURN, and the CSR will not be modified. If you wish to change the value, type "Y" or "YES" and RETURN, and the following message will appear:

```
New CSR address?
```

Enter the six-digit octal address of the Receiver CSR of the interface to be used. This address must be in the I/O Page (i.e., must be greater than 160000 and less than 177776). The address should be that of the receiver CSR of the interface to be used, hence the low-order octal digit of the address must be zero.

If you specify a CSR address which does not correspond to any presently-installed interface module, the following message will appear:

```
?PROM-F-Non - existant CSR address
```

and the current CSR default will not be modified. The program will again ask you for the new CSR address.

When the CSR questions have been answered, the program will display:

```
Do you wish to change the VECTOR (Y OR N)?
```

If the vector address originally displayed matches the desired value, type "N" or "NO" followed by RETURN, and PROM/RT-11 will exit from the INTERFACE operation. If the vector is incorrect, respond with "Y" or "YES" and the RETURN key. The following question will be printed:

```
New vector address?
```

Enter the three-digit octal address of the first interrupt vector for the interface to be used. This vector address must fall in the range 300 to 470 octal, inclusive. As this address specifies a pair of two-word interrupt vectors, the low-order octal digit must be zero.

When both CSR and vector have been successfully specified, the program will return to command mode.

5.3.8 HELP Command

If the user cannot remember the name of the command which he needs to use, he can type the H[ELP] command to PROM/RT-11. It will respond with a brief list of the commands available, as follows:

```
Valid commands are:

C - copy an existing PROM
D - diagnose PROM blaster hardware
E - exit from this program
H - type this help message
I - alter interface assignment
L - list PROM contents
M - modify one or more PROMs
P - program a set of PROMs
S - sequential mode
V - verify a PROM against a file
```

The system will then return to command mode, allowing the user to enter the name of the desired command.

5.3.9 EXIT Command

Typing the E[XIT] command causes the PROM/RT-11 software to disable the PROM blaster hardware interface and return control to the RT-11 operating system. This command must be used to terminate the PROM/RT-11 program; that is, the user cannot abort operation by typing CTRL-C.

When operating the PROM/RT-11 package in foreground mode, it is not necessary to type the EXIT command unless the user intends to remove the foreground job from memory.

5.4 PROGRAMMING PROMS IN A PRODUCTION ENVIRONMENT

To create patterned PROM chips in production quantities, the COPY Command can be used; that is after the PROGRAM Command is used to create the master chip or chips. Customarily, personnel well versed in all phases of PROM/RT-11 would use the PROGRAM Command to create master PROMs. After a master chip is mounted in the PROM blaster socket, production personnel need only to be familiar with the COPY Command dialogue to generate copies in blocks of 99

CHAPTER 6

DIAGNOSTICS AND MAINTENANCE

This chapter contains pertinent information about maintenance and troubleshooting techniques.

CAUTION

> ICs and individual components should not be replaced at the user's installation. The only exception is the memory chips that plug into sockets. Equipment should be returned to DIGITAL for repair. Replacement of ICs and components that are soldered into etched circuit boards require special equipment available at DIGITAL repair depots and at the factory.

6.1 DIAGNOSE COMMAND

You can use the DIAGNOSE command to run three tests on the PROM programmer, the data link between the PROM programmer, and the computer system (see Figure 6-1). Each test helps you verify operation of portions of the equipment or isolate an equipment failure. The tests are as follows:

1. Test 1; in conjunction with a successful test 2 is used to check proper operation of the data transfer and programmer data storage capabilities.

2. Test 2; checks the serial-line-unit-interface module and the input/output cable used to connect the computer system to the PROM programmer.

3. Test 3; checks the programmer's ability to transfer information from the programmer's internal storage to a PROM chip.

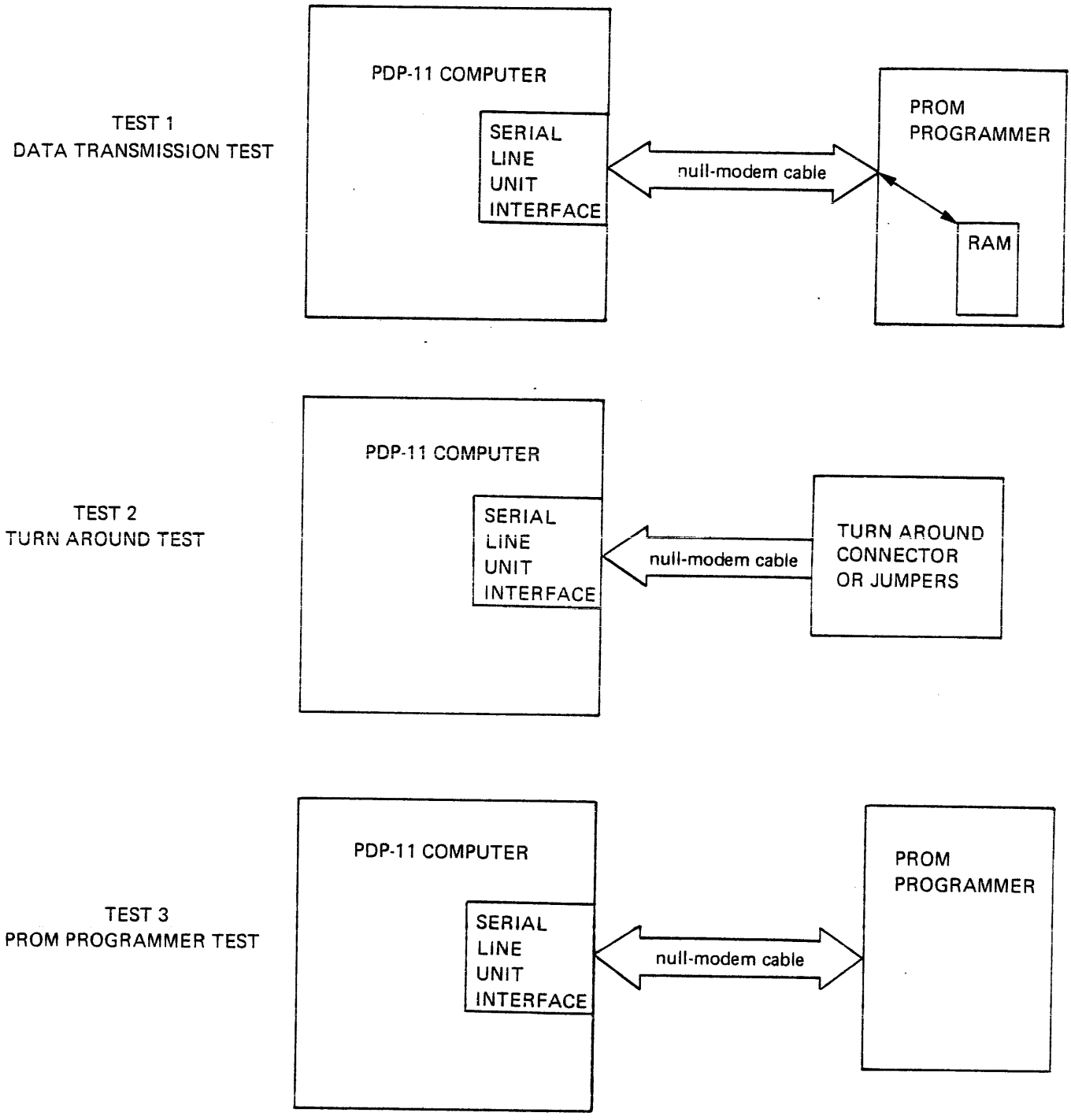


Figure 6-1 PROM/RT-11 Diagnostic Test

6.1.1 Test Preparation

Before attempting to check the PROM/RT-11 hardware you should perform the following test preparation procedures:

1. Ensure the cable between the PROM programmer and the serial-line-unit interface is connected.
2. Install the personality cards and the socket for the PROM or EPROM chip that will be used in Test 3. Test 3 requires a blank PROM or EPROM chip.
3. Install the correct PROM chip adapter in the PROM programmer socket.
4. Set the POWER switch on the PROM programmer to ON.

6.2 INVOKING THE PROM/RT-11 UTILITY

Before you can use the DIAGNOSE command you must invoke the PROM/RT-11 utility.

To invoke the PROM/RT-11 utility as a background program type:

```
.RUN PROM<RET>
```

or

To invoke PROM/RT-11 as a foreground program type:

```
.FRUN PROM<RET>
```

If you want the foreground version to be operated at a terminal different than the invoking terminal, type:

```
.FRUN PROM/T:n<RET>
```

Where:

n is the unit number of a terminal on an RT-11 multi-terminal system.

When PROM/RT-11 is invoked, it identifies itself by printing the following message on the terminal:

```
PROM/RT-11 vxx.yy    current PROM size mmmm by n
```

Where:

vxx.yy is the version of the RT-11 operating system.

mmmm is the number of words per chip for the personality

card installed in the programmer.

n is the number of bits per word for the personality card installed in the programmer.

Example:

```
PROM/RT-11 V3.0    current PROM size 1024 x 8
```

In this example the version of RT-11 being used is version 3.0 and the personality card installed in the programmer is for a 1024 word by 8-bit chip.

6.2.1 Diagnostic Test

After the PROM/RT-11 utility has been invoked, you can enter the DIAGNOSE command to perform diagnostic test as follows:

Type:

```
DIAGNOSE<RET>
```

DIAGNOSE may be abbreviated to D.

After you type the DIAGNOSE command the PROM/RT-11 utility responds with:

```
Which DIAGNOSTIC test would you like to run?

Test #1 ...    Tests the input/output link, i.e., computer to
               programmer RAM memory.

Test #2 ...    Wrap around -- serial-line interface, input line
               jumpered to the output line.

Test #3 ...    Program a PROM with a test pattern.

Enter test number:
```

When this message is printed at the terminal, you should respond by typing 1, 2, or 3 (depending on the type of test you want to do) followed by a carriage return.

Each of the tests is explained in Section 6.2.2 through 6.2.4.

6.2.2 Test Number 1: The Data Transmission Test

The data transmission test checks the serial-line-unit interface, the cable used to connect the PROM programmer to the computer, and parts of the PROM programmer. It also checks to see if you can transfer data between the PDP-11 computer and the PROM programmer (see Figure

6-1).

To run the data transmission test invoke the PROM/RT-11 utility as explained in section 6.2, then type D followed by a carriage return. The following message will be printed at your terminal:

Which DIAGNOSTIC test would you like to run?

Test #1 ... Tests the input/output link, i.e., computer to programmer RAM memory.

Test #2 ... Wrap around -- serial-line interface, input line jumpered to the output line.

Test #3 ... Program a PROM with a test pattern.

Enter test number:

Enter 1 followed by a carriage return and the PROM/RT-11 utility will run 50 passes of a data transmission check. Each pass consists of the following:

1. The RAM in the PROM programmer is filled with a test pattern. The test pattern consists of the least significant bits of the address of each location which are written into that location. The number of bits used depends on the width of the memory chip used. For example a 4-bit PROM receives the four least significant bits and an 8-bit PROM receives the eight least significant bits.

2. The same test data is transfered to the programmer and compared with the data stored in the PROM programmer RAM memory. This operation is repeated ten times.

3. The test data stored in the PROM programmer RAM memory is transferred to the computer and compared with the original test data. This operation is repeated ten times.

At the end of each pass the PROM/RT-11 utility prints the following message:

Diagnostic -- End of PASSxx -- TOTAL ERRORS yy

Where:

xx is the number of the pass that has just been completed.

yy is the number of errors detected since the test started.

If an error is detected, an additional message is printed at the terminal. The messages are explained below:

If an error occurs while data is being transferred to the programmer, the following message is printed at the terminal:

?PROM-F-DIAGNOSTIC failure data XFER to programmer

If an error occurs while data is being transferred from the programmer to the computer, the following message is printed at the terminal,

?PROM-F-DIAGNOSTIC failure data XFER from programmer

Expected	Received	Pass	Sub-pass
wwwwwwww	xxxxxxxx	yyyy	zz

wwwwwwww is the expected 8-bit data pattern,

xxxxxxxx is the received 8-bit data pattern,

yyyy is the pass of the test which detected the error,

zz is the number of times the test had been done before an error occurred, Each test is done ten times,

When 50 passes of the test are completed PROM/RT-11 prompts for a test number as follows:

Which DIAGNOSITC test would you like to run?

Test #1 ,,, Tests the input/output link, i,e,, computer to programmer RAM memory,

Test #2 ,,, Wrap around -- serial-line interface, input line jumpered to the output line,

Test #3 ,,, Program a PROM with a test pattern,

Enter Test number:

To repeat the data-transmission test enter a one and a carriage return, You can also run tests 2 and 3 (see Sections 6,2,3 and 6,2,4) at this time,

6,2,3 Test 2: Wraparound Test

The wraparound test (see Figure 6-1 checks the serial-line-unit interface, If a wraparound connector is used, the cable used to connect the PROM programmer to the computer is also checked, Some PDP-11 serial-line-unit interfaces wrap the data around on the module, A wraparound cable connected to the null-modem cable is not required on these systems but the cable is not checked, On these modules a maintenance bit is set by PROM/RT-11, to enable the data to be wrapped around on the module,

To run the wraparound test, invoke the PROM/RT-11 utility as explained in Section 6,2, then type a D followed by a carriage return, The following message will be printed at your terminal,

Which DIAGNOSTIC test would you like to run?

Test #1 ... Tests the input/output link, i.e., computer to programmer RAM memory.

Test #2 ... Wrap around -- serial-line interface, input line jumpered to the output line.

Test #3 ... Program a PROM with a test pattern.

Enter test number:

Now enter 2 followed by a carriage return to run the turnaround test.

The PROM/RT-11 utility checks the serial-line-interface to see if the maintenance bit is supported by the serial-line-unit interface on your development system. If the maintenance bit is supported on your system, PROM/RT-11 will ask you if you want to use the maintenance bit feature. The message is as follows:

Do you wish to use the maintenance bit in the serial interface?

If you want to use the maintenance bit feature enter a carriage return and the PROM/RT-11 utility starts the test. If you do not want to use the maintenance bit feature enter N followed by a carriage return, and the following message is printed at the terminal.

Install "wrap around" and type <RET>

At this time you must install a wraparound connector on the end of the null-modem cable or jumper the cable connector pins to turn the test data around and return it to the interface.

The pins of the RS232 connector that must be jumpered together using a connector or jumpers to do the wraparound test.

For all serial-line-unit interfaces except the DLV11-J install the jumpers as follows:

1. Connect pin E to Pin F

2. Connect Pin F to Pin J

For the DLV11-J interface install the jumpers as follows:

1. Connect Pin 8 to Pin 3

2. Connect Pin 7 to Pin 4

After the jumper or turnaround connector is installed type a carriage return and the test starts. Fifty passes of the test are made. Each pass transmits and receives all octal numbers from 1 to 377, 256 times.

As each pass is completed, PROM/RT-11 prints the following message at

the terminal:

Diagnostic "Wrap Around"

end of PASS xx TOTAL ERRORS yy

Where:

xx is the number of the pass the PROM/RT-11 has just completed,

yy is the number of errors detected since the test started,

If an error is detected while the test is running the following message is printed at the terminal:

?PROM-F-Diagnostic "wrap around" failure

After PROM/RT-11 completes 50 passes of the turnaround test, the following message is printed at the terminal:

Which DIAGNOSTIC test would you like to run?

Test #1 ... Tests the input/output link, i.e., computer to programmer RAM memory,

Test #2 ... Wrap around -- serial-line interface input line jumpered to the output line,

Test #3 ... Program a PROM with a test pattern,

Enter Test Number:

To repeat the wraparound test, enter a 2 and a carriage return. If you want to do the data transmission test enter a 1 followed by a carriage return (see Section 6.2.2). If you want to do the PROM programmer test enter a 3 followed by a carriage return (see Section 6.2.4).

6.2.4 Test 3: PROM Programmer Test

The PROM programmer test checks the serial-line-unit interface, the cable used to connect the PROM programmer to the computer, and the PROM programmer.

NOTE

A blank PROM or EPROM chip is required for this test.

To run the PROM programmer test, invoke the PROM/RT-11 utility as explained in Section 6.2, then type a D followed by a carriage return. The following message is printed at your terminal.

```
Which DIAGNOSTIC test would you like to run?

Test #1 ...    Tests the input/output link, i.e., computer to
               programmer RAM memory.

Test #2 ...    Wrap around -- serial-line interface, input line
               jumpered to the output line.

Test #3 ...    Program a PROM with a test pattern.

Enter test number:
```

Enter a 3 followed by a carriage return to start the PROM programmer test.

The PROM/RT-11 utility prints the following message at the terminal.

```
Mount blank PROM, and type <RET>
```

Mount a blank PROM or EPROM chip that can be programmed using the personality card installed on the PROM programmer. After the blank chip is installed, enter a carriage-return.

If the PROM or EPROM chip is not blank the following message is printed at the terminal:

```
?PROM-F-PROM is not blank
Mount blank PROM and type<RET>
```

After the blank PROM or EPROM is mounted and a carriage returned is entered at the terminal, the test starts.

The PROM/RT-11 utility writes the least significant bits of the address for each addressable location in that location. The PROM/RT-11 utility automatically executes a verify sequence.

If the PROM programmer does not complete the programming sequence, the following message is printed at the terminal.

```
?PROM-F-PROM Failed to Program
```

If an error is detected when the data in the chip is compared with the data generated by the PROM/RT-11 utility, the following message is printed at the terminal.

```
?PROM-F-PROM Failed to verify
```

If the test is successfully completed the following message is printed at the terminal:

```
PROM verified successfully
```

After the test is completed, the PROM/RT-11 utility prints the following message at the terminal,

Which DIAGNOSTIC test would you like to run?

Test #1 ... Tests the input/output link, i.e., computer to programmer RAM memory,

Test #2 ... Wrap around -- serial-line interface, input line jumpered to the output line,

Test #3 ... Program a PROM with a test pattern,

Enter test number:

If you want to do the test again enter a 3 followed by a carriage return, If you want to do the data transmission test enter a 1 followed by a carriage return (see Section 6.2.2), If you want to do the turnaround test, enter a 2 followed by a carriage return (see Section 6.2.3),

If you have done all the testing you want to do, type a CTRL/C to return to the monitor,

6.3 USING THE DIAGNOSE COMMAND TO ISOLATE EQUIPMENT FAILURES

You should use one of two methods to determine the appropriate diagnostic test(s) to isolate or verify an equipment failure, Each of these methods is based on information you gather from the PROM/RT11 utility while it is operating,

The first method is a general approach based on a lack of information about the failure, If the PROM/RT11 software fails to report an error condition or if the error condition is intermittent, the general approach may yield additional information, This information should help you to isolate the failing sub-system or to compile statistics on frequency and repetition of errors,

The first goal of the general approach is to split the hardware into two pieces; the data transfer link and the programming section, Test 1 accomplishes this goal, If test 1 is successful then the problem lies in the programming section (proceed to Section 6.3.2), If test 1 fails the problem lies in the data transfer link (proceed to Section 6.3.1),

The second method is a direct approach to a specific part of the hardware, It is based on the information from PROM/RT11 in the form of an error message that indicates the failure mode, For example:

```
?PROM-F-Unsuccesstul data transfer to programmer
?PROM-F-Unsuccessful data transfer from programmer
```

These messages indicate a failure to transfer information between the

computer and the programmer RAM, (for example the data transfer link). To isolate the failure further proceed to Section 6.3.1.

?PROM-F-PROM failed to program
?PROM-F-PROM failed to verify

These messages indicate a failure to correctly transfer the contents of the programmer RAM to PROM, the programming section. To isolate the failure further proceed to Section 6.3.2.

6.3.1 Data Transfer Link Failure Isolation

The procedure to isolate a data transfer link failure are as follows:

1. Disconnect the RS232 cable from the back of the programmer and install jumpers between pins 2 and 3 of the cable.
   Run test 2.

2. If test 2 fails remove cabling one piece at a time, and install the jumper between pins 2 and 3 in the end of the remaining cable.
   Run test 2.

3. If test 2 continues to fail after the last cable is removed, install jumpers in the output connector. For all DLV11's (except DLV-11J) install jumpers as follows:

   a. Connent Pin M to Pin E
   b. Connect Pin F to Pin J

   For the DLV-11J install jumpers between the following pins:

   a. Connect Pin 8 to Pin 3
   b. Connect Pin 7 to Pin 4

   Now run test 2.

   NOTE

   You should manufacture a wraparound connector so that the connector will not be jumpered incorrectly during the test. The mating plug for all interfaces except DLV11-J is a Berg H856 connector. The mating plug for the DLV-11J is a DIGITAL option number 3270-A connector with jumpers installed. Or you can purchase an AMP part number 87133-5 connector and manufacture the wraparound connector.

4. If test 2 fails with the jumpers installed on the interface module, replace the serial line interface.

5. If test 2 is successful during steps 2 and 3 then replace the last cable that was removed.

6. If test 2 is successful during step 1, then the programmer may be at fault. Reconnect the RS232 cable to the back of the programmer and run test 1.

7. If test 1 fails then the programmer mainframe is at fault and should be returned to DIGITAL for repair.

8. If test 1 is successful then go to Section 6.3.2.

6.3.2 Programmer Programming Failure Isolation

Proceed as shown in the following steps to isolate a programmer failure:

CAUTION

Test 3 programs a PROM with a test pattern. This test pattern renders the PROM useless for further programming, unless the type of chip selected is an EPROM which you can erase.

1. Run test 3.

2. If test 3 is successful (an error has not been detected) the error may be of an intermittent nature. Test 3 may have to be repeated to display an intermittent error condition.

3. If test 3 fails replace the personality card set and/or socket adapter and retry test 3.

4. If test 3 is successful (after step 3) then return the personality card set and/or socket adapter to DIGITAL for repair.

NOTE

Test 3 cannot be considered an absolute indication if the failure has exhibited an intermittent nature.

5. If test 3 continues to fail, after step 3 has been done then the programmer mainframe is at fault and the programmer should be returned to DIGITAL for repair.

6.4 PROM/RT-11 TROUBLESHOOTING

The PDP-11 and the PROM programmer are constructed of highly reliable IC logic modules. Use of these circuits and a minimum amount of preventive maintenance ensures relatively little down time due to failure. If a malfunction occurs, you should analyze the condition and correct it by replacing the defective module or unit. The modules that are replaced and/or the PROM programmer should be returned to a DIGITAL repair depot for repair.

Switches on modules that are used to replace another module should be set to correspond to the settings on the module that was removed. The hardware manual shipped with the equipment lists the switch settings for all modules.

6.4.1 Operator Errors

Operator errors are the cause of many computer malfunctions. When it has been determined that a module or the programmer is malfunctioning, it is a good idea to check the switch positions on the module before replacing the module. The hardware manual for the module lists the switch positions, basic symptoms, and the corrective action required.

If the PROM programmer does not respond to the PROM/RT-11 utility it is a good idea to check the power switch and make sure the programmer is on.

6.4.2 Troubleshooting Procedures

When a malfunction is detected, gather all information available from other users who have encountered the problem and check the system log book for any previous reference to this problem. Make a note of indications and error messages that you have observed before attempting to locate the module that is malfunctioning. This information is helpful for describing the malfunction in the logbook or to the depot that repairs your module.

Do not attempt to locate the problem using complex software systems. Run the test in Section 6.2 and select the test that exhibits the error condition. The test are carefully written to include test that assist you when isolating the defect to the serial-line-unit module, the connecting cable, or the PROM programmer.

6.4.3 Validation Tests

If a defective module is replaced by a new module or sent to a DIGITAL depot for repair, tag the defective module and note the nature of the failure.

To confirm that a new or repaired module resolved the problem, run the tests that originally exhibited the problem. If modules have been moved during the troubleshooting period return all modules to their original position before running the validation test.

Contact your DIGITAL field service office for the procedure required to repair your module.

6.4.4 Cable Problems

Malfunction of the serial-line-unit interface or the PROM programmer that are not corrected by replacement of the unit or module may be caused by a defective cable. If validation tests are run and the replacement unit has not corrected the problem, it is a good idea to ensure that the cable is connected correctly. If the cable is connected correctly and the problem persists remove and replace the cable.

6.4.5 Log Entry

A log book should be kept for each PROM/RT-11 system. Record all data indicating the symptoms given by the faults you detect, the method of fault detection, the unit at fault, and any comments that would be helpful to maintain the equipment in the future.

The log book should be maintained on a daily basis, recording all operator usage and maintenance results.

APPENDIX A

SAMPLE PROM BASED APPLICATIONS

A.1 INTRODUCTION

This appendix presents two sample applications suitable for operation in a PROM/RAM environment. Descriptions of design goals for both sample applications are given in Chapter 3, Coding, LINK maps etc., are presented here.

A.2 ELAPSED TIME SINCE POWER UP APPLICATION PROGRAM

A.2.1 Program Modules Coded in MACRO-11 Assembly Language

```
                .TITLE  Main Module for Simple PROM Application

; This simple PROM application example is composed of two assembly language
; source files, which will be separately assembled and later combined by
; the LINK utility to form the desired memory image.
;
; The other module of the program contains a clock interrupt handler which
; is responsible for keeping track of the time of day on an interrupt-driven
; basis.
;
; The main program will make use of this information by printing it out in
; a formatted fashion on the terminal whenever any character is typed at the
; keyboard.
;
; Notice that three program sections are used exclusively:
;
;       .ASECT                  to initialize low-memory areas (vectors)
;       .Psect  rom             to define all instructions and data to be
;                               placed in PROM memory
;       .PSECT  RAM,D           to define all RAM (scratchpad) locations

        .GLOBL  INIT, HOURS, MINS, SECS ;External variables from other module

        .ASECT                  ;Go to absolute section to define vectors
.=24                            ;Origin to the power-up vector
        .WORD   PWRUP           ;Power-up PC = routine labelled "PWRUP"
        .WORD   340             ;Power-up PS indicates PRIO 7 (no interrupts)

        .PSECT  ROM             ;Origin to ROM section
PWRUP:  MOV     #$STACK,SP      ;Set up a valid hardware stack pointer
        JSR     PC,INIT         ;And initialize all required RAM locations
                                ; defined in the clock module
        CLR     R1              ;Init R1 for MTPS operand
        MTPS    R1              ;When all locations have been initialized,
                                ; start clock module operating by allowing
                                ;  interrupts.
KEYCHK: TSTB    @#177560        ;Wait for console keyboard ready
        BPL     KEYCHK          ;If PL, no character has been typed
        TSTB    @#177562        ;Else read the character
        JSR     R5,PRINT        ;And begin the print the time message
         .WORD  MESS1           ; by calling the PRINT subroutine
        MOV     HOURS,R0        ;Place number of hours elapsed in R0
        JSR     PC,DECOUT       ;And convert it to decimal ASCII, typing it
                                ; out on the console
        JSR     R5,PRINT        ;Print out " hours, "
         .WORD  MESS2
        MOV     MINS,R0         ;Get number of elapsed minutes
        JSR     PC,DECOUT       ;Convert and print out minutes
        JSR     R5,PRINT        ;Print out " minutes, and "
         .WORD  MESS3
        MOV     SECS,R0         ;Get number of elapsed seconds
        JSR     PC,DECOUT       ;Convert and print out seconds
        JSR     R5,PRINT        ;Print out " seconds," and carriage return -
         .WORD  MESS4           ; line feed characters to return carriage
```

```
        BR      KEYCHK          ;Go wait for another character to be typed.

PRINT:  MOV     (R5)+,R4        ;R4 = address of message to be printed
PCHAR:  MOVB    (R4)+,R0        ;R0 = next character in message to print
        BEQ     PRET            ;If EQ, entire message has been printed
        JSR     PC,TYPECH       ;Else type out the character in R0
        BR      PCHAR           ;And go get next message character

PRET:   RTS     R5              ;Return to calling routine

DECOUT: SWAB    R0              ;Convert binary 00-99 to two decimal ASCII
DLOOP:  ADD     #<-10,*256,>+1,R0 ; digits, by doing a byte division operation
        BPL     DLOOP           ;   maintaining quotient in low byte and
        ADD     #<10,*256,>-1+"00,R0 ; remainder in high byte of R0
        JSR     PC,TYPECH       ;Type out most significant digit
        SWAB    R0              ;Move least significant digit down
                                ; and type it out also ...
TYPECH: TSTB    @#177564        ;Is printer ready to accept a character?
        BPL     TYPECH          ;If PL no - still printing last one
        MOVB    R0,@#177566     ;Else print this character
        RTS     PC              ;And return to calling routine

        .PSECT  RAM,D           ;Origin into RAM to define stack
        .BLKW   64.             ;Reserve space for hardware stack at beginning
                                ; of RAM
SSTACK::                        ;And place reference label at end (as stack
                                ; grows towards lower addresses)

        .PSECT  ROM             ;Origin back to ROM to define message text
MESS1:  .ASCIZ  /Elapsed time since power-up is /


MESS2:  .ASCIZ  / hours, /


MESS3:  .ASCIZ  / minutes, and /


MESS4:  .ASCIZ  / seconds,/<015><012>
```

```
.END
```

```
                .TITLE  Clock Module for Simple ROM Application

;
; This module provides a clock interrupt handler and the code to
; maintain the current time of day (in terms of hours, minutes, and
; seconds since the system was bootstrapped) for the main module
; (MAIN.MAC).
;
; Variables HOURS, MINS, and SECS are global to allow their values
; to be examined by the other module.
;
; The symbol TK$SEC should be equated to the number of clock ticks
; per second, i.e., the line frequency in this location.

TK$SEC=60.                      ;For United States

        .ASECT                  ;Origin to define vectors
.=100                           ;Origin to the line clock vector
        .WORD   CLKINT          ;Line clock interrupt PC = "CLKINT"
        .WORD   340             ;Line clock interrupt PS = PRIO 7

        .PSECT  ROM             ;Now origin to ROM section
CLKINT: DEC     TICKS           ;Count down ticks to next second
        BNE     CLKEXT          ;If NE, not at second boundary yet
        MOV     #TK$SEC,TICKS   ;Else reset ticks/second
        INC     SECS            ;Indicate another second has elapsed
        CMP     SECS,#60.       ;Has a minute gone by?
        BLT     CLKEXT          ;If LT no
        CLR     SECS            ;Zero number of seconds since last minute
        INC     MINS            ;And indicate another minute has lapsed
        CMP     MINS,#60.       ;Has an hour elapsed?
        BLT     CLKEXT          ;If LT no
        CLR     MINS            ;Zero number of minutes since last hour
        INC     HOURS           ;And indicate another hour
CLKEXT: RTI                     ;Dismiss clock interrupt

        .PSECT  RAM,D           ;Origin to RAM section to define variables
TICKS:: .BLKW                   ;Number of ticks to next second
SECS::  .BLKW                   ;Number of seconds since last minute
MINS::  .BLKW                   ;Number of minutes since last hour
HOURS:: .BLKW                   ;Number of elapsed hours

        .PSECT  ROM             ;Origin back to ROM to define initialization
INIT::  MOV     #TK$SEC,TICKS   ;Set up initial ticks/second counter value
        CLR     SECS            ;And clear out elapsed seconds ...
        CLR     MINS            ; ... minutes ...
        CLR     HOURS           ; ... and hours
        RTS     PC              ;Return to main program

        .END
```

```
VIRTUAL MEMORY USED:    300 WORDS    ( 2 PAGES)
DYNAMIC MEMORY AVAILABLE FOR 57 PAGES
DK:CLOCK,LP:CLOCK=DK:CLOCK/E:LC
```

A.2.2 LINK Map Prior to Assigning RAM .PSECTs on Next 4K Boundary

```
LINK/BOTTOM:400/EXECUTE:MAIN/MAP:TT: MAIN,CLOCK
RT-11 LINK  V05.02      Load Map
MAIN  .SAV      Title:  MAIN    Ident:          /B:0004000

Section  Addr    Size    Global  Value   Global  Value   Global  Value
. ABS.   000000  000400   (RW,I,GBL,ABS,OVR)
ROM      000400  000376   (RW,I,LCL,REL,CON)
                         INIT    000752
RAM      000776  000210  (RW,D,LCL,REL,CON)
                         $STACK  001176  TICKS   001176  SECS    001200
                         MINS    001202  HOURS   001204

Transfer address = 000001, High limit = 001206 =   323. words
```

A.2.3 LINK Map After Assigning RAM .PSECTs on Next 4K Boundary

```
 LINK/BOTTOM:400/EXECUTE:MAIN/MAP:TT/BOUNDARY:20000 MAIN,CLOCK
Boundary section? RAM
RT-11 LINK  V05.02      Load Map
MAIN  .SAV      Title:  MAIN    Ident:          /B:000400

Section  Addr    Size    Global  Value   Global  Value   Global  Value

. ABS.   000000  000400   (RW,I,GBL,ABS,OVR)
ROM      000400  017400   (RW,I,LCL,REL,CON)
                         INIT    000752
RAM      020000  000210   (RW,D,LCL,REL,CON)
                         $STACK  020200  TICKS   020200  SECS    020202
                         MINS    020204  HOURS   020206

Transfer address = 000001, High limit = 020210 = 4164. words
```

A,2,4 Invoking the PROM/RT-11 Utility Program

In this sample blasting sequence, 1024 by 8-bit wide chips are used, The PROM/RT-11 utility is invoked with the following command and response.

```
,RUN PROM11

PROM/RT-11  V1,0   Current PROM Size 1024 by 08
```

A,2,5 PROGRAM Command Sequence Used to Enter Application Program into PROM

The following command sequence is used to blast a pair of PROM chips.

```
Command: P
Do you want inverted data (Y or N)? N
Do you want inverted addresses (Y or N)? N
Name of input file (DEV:FILNAM,TYP)? DX1:MAIN,SAV
Final program address (octal)? 775
Mount PROM for address 000000-000774, bits 07-00, and type <RETURN>:
PROM for address 000000-000774, bits 07-00, has been programmed
Mount PROM for address 000000-000774, bits 15-08, and type <RETURN>:
PROM for address 000000-000774, bits 15-08, has been programmed


Command:


Command: E
```

A,2,6 Application Program Execution

The pair of blasted PROM chips in this example are inserted into their proper locations (1KLB and 1KH8) in an MRV11-BA module, The module is then inserted in the LSI-11 backplane and the application program is ready for execution to execute:

1. Power up the terminal
2. Power up the LSI-11
3. After the desired pause, press any key on the terminal

The program responds by printing the following message

Elapsed time since power is: xx hours, xx minutes and xx seconds.

The below examples were taken from the terminal during actual execution.

```
    Elapsed time since power-up is  00  hours,  00  minutes,  and  11
seconds,
    Elapsed time since power-up is  00  hours,  00  minutes,  and  17
seconds,
    Elapsed time since power-up is  00  hours,  01  minutes,  and  51
seconds,
```

A.3 PRICE COMPUTATION FOR WEIGHTED MERCHANDISE APPLICATION PROGRAM

A.3.1 Application Program Coded in FORTRAN IV and MACRO-11 Assembly Language

```
0001   6      PROGRAM SCALER
       C
       C      Sample PROM/RT-11 application program, written in FORTRAN IV
       C      and MACRO-11.
       C
       C      This software system handles 8 weigh-stations, each equipped
       C      with:
       C          (A) an electronic scale, interfaced to the LSI-11
       C                  through one channel of an ADV11 Analog-to-Digital
       C                  converter,
       C          (B) a four-digit thumbwheel switch, used to enter the
       C                  price-per-kilogram of the goods being weighed on
       C                  the scale. The switch generates 16-bit output as
       C                  four BCD digits, and is interfaced through the
       C                  input port of a DRV11 Parallel Line Unit,
       C          (C) a four-digit "seven-segment" display, to provide
       C                  readout of the calculated value of the goods weighed,
       C                  It takes 16-bit BCD input from the output port of
       C                  a DRV11 Parallel Line Unit,
       C          (D) a pushbutton, to signal that the weight should be taken
       C                  and value computed. It is connected to the "REQ A"
       C                  interrupt request input on the DRV11.
       C
       C      While operating the 8 weigh stations on an interrupt-driven basis,
       C      the program provides an operator terminal capability to interrogate
       C      totals for weight and value of goods for each station, and ability
       C      to "recalibrate" each station's scale under software control.
       C
       C-------------------------------------------------------------------
       C
0002           BYTE        ANSWER
0003           COMMON      IUSES(8), TOTWGT(8), TOTVAL(8), CALIBR(8)
       C
       C-------------------------------------------------------------------
       C
       C      Initialization of variables. Note that in PROM applications the
       C      FORTRAN "DATA" statement must not be used!
       C
```

A-10

```
0004          DO 10, I=1,8
0005          IUSES(I) = 0          !Total number of weighings on this scale
0006          TOTWGT(I) = 0         !Total weight recorded on this scale
0007          TOTVAL(I) = 0         !Total value computed for all items weigned
0008          CALIBR(I) = .001!Initial calibration for A/D scale input values
0009    10    CONTINUE
        C
        C     Check for new initial calibration values required.
        C
0010          TYPE 1000
0011    1000  FORMAT(' Do you wish to change scale calibrations? ',$)
0012          ACCEPT 1010,ANSWER
0013    1010  FORMAT(A)
0014          IF (ANSWER.NE.'YES') GOTO 30
0016          DO 20, I=1,8
0017          TYPE 1020,I,CALIBR(I)
0018    1020  FORMAT(' For line ',I1,', calibration is: ',F8.5,', new value? ',$)
0019          ACCEPT 1030,CALIBR(I)
0020    1030  FORMAT(F12.0)
0021    20    CONTINUE
        C
        C     Once calibrations are set, end initialization by setting interrupt
        C     enable for REQ A interrupts on all DRV11 interfaces. Note that
        C     interrupt enables are not set until all initialization is complete!
        C
0022    30    DO 40, I=0,7
0023          CALL IPOKE( "167770 + "10*I, "100 )
0024    40    CONTINUE
        C
        C     Application is now up and running. Enter the idle loop to look
        C     for operator display commands. All other functions are driven by
        C     the DRV11 interrupts.
        C
0025    50    TYPE 1040
0026    1040  FORMAT(' Do you wish to display status (D) or calibrate (C)? ',$)
0027          ACCEPT 1010,ANSWER
0028          IF (ANSWER.EQ.'D') GOTO 70
0030          IF (ANSWER.NE.'C') GOTO 50
        C
        C     Here to reset scale calibration for a particular scale.
        C
0032    60    TYPE 1050
0033    1050  FORMAT(' Enter number of scale station: ',$)
0034          ACCEPT 1060,I
0035    1060  FORMAT(I)
0036          IF (I.LE.0) GOTO 60 !Try again if scale number is out of range
0038          IF (I.GT.8) GOTO 60
0040          TYPE 1020,I,CALIBR(I)       !Display current value
0041          ACCEPT 1030,CALIBR(I)       !And accept new one
0042          GOTO 50                     !Loop to get another command
        C
        C     Display command
        C
0043    70    TYPE 1050             !Get station index
0044          ACCEPT 1060,I
```

```
0045          IF (I,LE,0) GOTO 70  !And if out of range, try again
0047          IF (I,GT,8) GOTO 70
0049          TYPE 1070,I,IUSES(I),TOTWGT(I),TOTVAL(I),TOTVAL(I)/IUSES(I)
0050    1070  FORMAT(' Station #',I1,' has processed ',I4,' transactions.',/,
             *     ' Total weight for all transactions: ',F8,2,' kilograms.',/,
             *     ' Total value for all goods weighed:  ',F8,2,/,
             *     ' Average value of goods per transaction: ',F8,2)
0051          GOTO 50                        !Loop to get another command
0052          END
```

```
FORTRAN IV      Storage Map for Program Unit SCALER

Local Variables, ,PSECT $DATA, Size = 000034 ( 14, words)

Name   Type  Offset     Name   Type  Offset     Name   Type  Offset
ANSWER L*1   000014     I      I*2   000016

COMMON Block /       /, Size = 000160 (  56, words)

Name   Type  Offset     Name   Type  Offset     Name   Type  Offset
IUSES  I*2   000000     TOTWGT R*4   000020     TOTVAL R*4   000060
CALIBR R*4   000120

Local and COMMON Arrays:

Name     Type   Section Offset  ------Size----- Dimensions
CALIBR R*4      ,$$$$, 000120  000040 (   16,) (8)
IUSES  I*2      ,$$$$, 000000  000020 (    8,) (8)
TOTVAL R*4      ,$$$$, 000060  000040 (   16,) (8)
TOTWGT R*4      ,$$$$, 000020  000040 (   16,) (8)

Subroutines, Functions, Statement and Processor-Defind Functions:
Name   Type   Name   Type   Name   Type   Name   Type   Name   Type
IPOKE  I*2
```

```
0001          SUBROUTINE WEIGH( LINE )
      C
      C       This subroutine is activate whenever a weigh station pushbutton
      C       is pressed, It is entered from the assembly language DRV11 interrupt
      C       service routine, with the integer variable LINE set to the line
      C       which caused the interrupt,
      C
      C*************************************************************************
      C
0002          COMMON      IUSES(8), TOTWGT(8), TOTVAL(8), CALIBR(8)
      C
      C*************************************************************************
      C
      C       Calculate address of CSR for DRV11 which interrupted,
      C
0003          IADDRS = "167770 + "10*(LINE-1)
      C
0004          IUSES(LINE) = IUSES(LINE) + 1         !Update count of weighings
0005          WEIGHT = CALIBR(LINE) * IADC(LINE)    !Read scale value for this line
0006          PPKILO = .01 * IBCDI( IPEEK(IADDRS+4) ) !Get price per kilo
0007          VALUE = WEIGHT * PPKILO
0008          IVALUE = IFIX( VALUE*100 )
0009          CALL IPOKE( IADDRS+2, IBCDO( IVALUE ) ) !Write value in display
      C
0010          TOTWGT(LINE) = TOTWGT(LINE) + WEIGHT
0011          TOTVAL(LINE) = TOTVAL(LINE) + VALUE
      C
      C       Interrupt processing is now complete, Return to DRV11 interrupt
      C       service routine,
      C
0012          RETURN
0013          END
```

FORTRAN IV Storage Map for Program Unit WEIGH

Local Variables, ,PSECT $DATA, Size = 000036 (15. words)

Name	Type	Offset	Name	Type	Offset	Name	Type	Offset
IADDRS	I*2	000002	IVALUE	I*2	000020	LINE	I*2 @	000000
PPKILO	R*4	000010	VALUE	R*4	000014	WEIGHT	R*4	000004

COMMON Block / /, Size = 000160 (56. words)

Name	Type	Offset	Name	Type	Offset	Name	Type	Offset
IUSES	I*2	000000	TOTWGT	R*4	000020	TOTVAL	R*4	000060
CALIBR	R*4	000120						

Local and COMMON Arrays:

Name	Type	Section	Offset	------Size------	Dimensions
CALIBR	R*4	,$$$$,	000120	000040 (16,)	(8)
IUSES	I*2	,$$$$,	000000	000020 (8,)	(8)
TOTVAL	R*4	,$$$$,	000060	000040 (16,)	(8)
TOTWGT	R*4	,$$$$,	000020	000040 (16,)	(8)

Subroutines, Functions, Statement and Processor-Defined Functions:

Name	Type	Name	Type	Name	Type	Name	Type	Name	Type
IADC	I*2	IBCDI	I*2	IBCDO	I*2	IFIX	I*2	IPEEK	I*2
IPOKE	I*2								

```
                                                ,ps58,120
,TITLE  Subroutines for SCALER,FOR

                                        ,SBTTL  DRV11 Interrupt Servicing


; The routine DRVINT services interrupts from all 8 DRV11 interfaces
; installed in the system, The line number of the interrupting DRV11
; is encoded in the condition code bits of the new processor status
; word fetched as the result of the interrupt,

; Initialization of the DRV11 interrupt vectors:

        ,ASECT
,=300
        ,WORD   DRVINT, 340+1,  ;Vector for line #1
        ,WORD   DRVINT, 340+2,  ;Vector for line #2
        ,WORD   DRVINT, 340+3,  ;Vector for line #3
        ,WORD   DRVINT, 340+4,  ;Vector for line #4
        ,WORD   DRVINT, 340+5,  ;Vector for line #5
        ,WORD   DRVINT, 340+6,  ;Vector for line #6
        ,WORD   DRVINT, 340+7,  ;Vector for line #7
        ,WORD   DRVINT, 340+8,  ;Vector for line #8

; Now, the interrupt routine itself (in the PROM Psect):

        ,GLOBL  WEIGH           ;External FORTRAN routine to service intr

        ,PSECT  USER$I
DRVINT: JSR     R0,@PC          ;Save a register on stack with destroying
        MFPS    R0              ; condition codes, and retrieve them
        BIC     #177760,R0      ;Clear all but line #
        MOV     R0,DRVLIN       ;And store for reference by WEIGH routine
        MOV     R1,-(SP)        ;Save remaining registers
        MOV     R2,-(SP)        ; on stack, as FORTRAN
        MOV     R3,-(SP)        ;  routine may use them
        MOV     R4,-(SP)
        MOV     R5,-(SP)
        MOV     #ARGLST,R5      ;Load FORTRAN argument list pointer
        JSR     PC,WEIGH        ;And call routine to process interrupt data
        MOV     (SP)+,R5        ;Restore registers
        MOV     (SP)+,R4        ; after return from
        MOV     (SP)+,R3        ;  FORTRAN routine
        MOV     (SP)+,R2
        MOV     (SP)+,R1
        MOV     (SP)+,R0
        RTI                     ;And return from the interrupt

ARGLST: ,WORD   1, DRVLIN       ;Argument list for call to WEIGH

        ,PSECT  USER$D,D        ;Origin to RAM section
DRVLIN: ,BLKW                   ;And define variable to get line #
```

```
                                        ,SBTTL  IBCDI

;
; IBCDI -- Convert 4 BCD digits to binary value 0000 to 9999.
;
; Format of FORTRAN call:
;
;       J = IBCDI( I )
;
; where:
;       I = a 16-bit integer containing 4 4-bit BCD coded digits
;       J = the 16-bit binary value of the conversion

        ,PSECT  USERSI          ;Origin to PROM section
IBCDI:: MOV     #4,R0           ;Set loop count of digits to convert
        CLR     R1              ;Clear value accumulator
        MOV     @2(R5),R3       ;R3 = BCD-coded input value
1$:     CLR     R2              ;Clear high-order accumulator of R2,R3
pair
        ASHC    #4,R2           ;Strip off next four bits from R3
        MUL     #10.,R1         ;Multiply existing value by 10
        ADD     R2,R1           ;And add in newly-extracted digit
        SOB     R0,1$           ;Loop for four digits
        MOV     R1,R0           ;Return result of function in R0
        RTS     PC              ;Return to FORTRAN program

        ,SBTTL  IBCDO

;
; IBCDO -- Converts binary value of integer to 4 BCD-coded digits
;
; Format of FORTRAN call:
;
;       J = IBCDO( I )
;
; where:
;
;       I = the integer binary value to convert (range: 0 to 9999)
;       J = the 16-bit packed with four BCD digits

        ,PSECT  USERSI          ;Origin to PROM section
IBCDO:: MOV     @2(R5),R3       ;R3 = binary value to convert
        MOV     #DIVTAB,R1      ;R1 => table of divisors for
        conversion
        CLR     R0              ;Clear result accumulator
        MOV     #3,R4           ;Initialize loop count for three
        digits
1$:     CLR     R2              ;Clear high-order dividend
        DIV     (R1)+,R2        ;Divide by next power-of-ten
        BIS     R2,R0           ;Insert four quotient bits into result
        ASH     #4,R0           ;And prepare for next four
        SOB     R4,1$           ;Loop for three high-order digits
        BIS     R3,R0           ;Insert remainder of divide-by-ten
        RTS     PC              ;And return with result in R0
```

```
                                        ,SBTTL  IADC

;  IADC -- Read a selected A/D channel and return converted value
;
;  Format of FORTRAN call:
;
;       idata = IADC( ichan )
;
;  where:
;
;       ichan = the integer specification of the channel number for input
;       idata = the 16-bit integer result of the input conversion

ADVCSR  = 170400
ADVDBR  = ADVCSR + 2

        ,PSECT  USERSI          ;Origin to PROM section
IADC::  MOV     @2(R5),R0       ;R0 = input channel number
        BIC     #177760,R0      ;Take modulo 16
        SWAB    R0              ;Move channel to high byte
        INC     R0              ;And set "GO" bit
        MOV     R0,@#ADVCSR     ;Start conversion on channel
1$:     TSTB    @#ADVCSR        ;Is conversion complete?
        BPL     1$              ;If PL no -- wait in loop
        MOV     @#ADVDBR,R0     ;Else get converted value
        RTS     PC              ;And return to FORTRAN program

        ,END
```

A.3.2 LINK Map Prior to Assigning RAM .PSECTs on Next 4K Boundary

```
RT-11 LINK V05.04A      Load Map        Fri 01-Sep-78 00:38:04
SCALER.SAV      Title:  SCALER  Ident:  FORY02  /B:10004000

Section  Addr    Size    Global  Value   Global  Value   Global  Value

. ABS.   000000  000400   (RW,I,GBL,ABS,OVR)
                         $USRSW  000000  $RF2A1  000000  .VIR    000000
                         $NLCHN  000006  $HRDWR  000006  $SYSV$  000010
                         $WASIZ  000131  $LRECL  000210
OTS$I    000400  020330   (RW,I,LCL,REL,CON)
                         $$OTSI  000400  $SIMRT  000566  $TKS    001416
                         $TKB    001420  $TPS    001422  $TPB    001424
                         $CVTFB  002462  $CVTFI  002462  $CVTCB  002476
                         $CVTCI  002476  %CVTDB  002476  $CVTDI  002476
                         CIC$    002510  CID$    002510  CLC$    002510
                         CLD$    002510  $DI     002510  CIF$    002520
                         CLF$    002520  $RI     002520  CIL$    002626
                         CLI$    002632  $OTI    002662  $$OTI   002664
                         $$SET   004450  RCI$    005062  GCO$    006076
                         FCO$    006104  ECO$    006110  DCO$    006116
                         $CVTIF  007040  $CVTIC  007054  $CVTID  007054
                         CCI$    007066  CDI$    007066  $IC     007066
                         $ID     007066  CFI$    007102  $IR     007102
                         TVL$    007166  $TVL    007166  TVF$    007174
                         $TVF    007174  TVD$    007202  $TVD    007202
                         TVQ$    007210  $TVQ    007210  TVP$    007216
                         $TVP    007216  TVI$    007224  $TVI    007224
                         OCI$    007360  ICI$    007366  $ECI    007402
                         OCO$    007562  ICO$    007570  IFW$    007766
                         $IFW    007772  IFW$$   010034  IFR$    010104
                         $IFR    010110  IFR$$   010146  $CHKER  010170
                         $IOEXI  010214  $EOL    010242  EOL$    010244
                         $OTIS   010360  $$OTIS  010362  SAVRG$  010502
                         THRD$   010660  $PUTRE  010662  $STPS   011170
                         STP$    011176  $STP    011176  FOO$    011202
                         $EXIT   011222  $ERRTB  011346  $ERRS   011453
                         $WAIT   015146  $FCHNL  015210  $INITI  015306
                         $CLOSE  015420  $GETRE  016064  $TTYIN  016140
                         $PUTBL  016274  $GETBL  016504  $EOFIL  016670
                         $EOF2   016704  $FIO    017444  $$FIO   017450
                         $VRINT  020600  $DUMPL  020602
OTS$P    020730  000050   (RW,D,GBL,REL,OVR)
SYS$I    021000  000020   (RW,I,LCL,REL,CON)
                         IPEEK   021000  IPOKE   021010
USER$I   021020  000214   (RW,I,LCL,REL,CON)
                         IBCDI   021100  IBCDO   021134  IADC    021200
$CODE    021234  001662   (RW,I,LCL,REL,CON)
                         $$OTSC  021234  WEIGH   022376
OTS$O    023116  001010   (RW,I,LCL,REL,CON)
                         $$OTSO  023116  $OPEN   023116
SYS$O    024126  000000   (RW,I,LCL,REL,CON)
$DATAP   024126  000716   (RW,D,LCL,REL,CON)
```

```
OTS$D   025044  000006   (RW,D,LCL,REL,CON)
OTS$S   025052  000304   (RW,D,LCL,REL,CON)
                         $AOTS   025354
SYS$S   025356  000004   (RW,D,LCL,REL,CON)
                         $SYSLB  025356  $LOCK   025360  $CRASH  025361
$DATA   025362  000072   (RW,D,LCL,REL,CON)
USER$D  025454  000002   (RW,D,LCL,REL,CON)
,$$$$,  025456  000160   (RW,D,GBL,REL,OVR)
$STACK  025636  000200   (RW,D,LCL,REL,CON)
$STK$T  026036  000000   (RW,D,LCL,REL,CON)
                         $$$STK  026036
```

A.3.3 LINK Map After Assigning BAM .PSECIs on Next 4K Boundary

```
Transfer address = 021234, High limit = 026036 = 5647. words
RT-11 LINK  V05.04A     Load Map        Fri 01-Sep-78 00:39:46
SCALER.SAV      Title:  SCALER  Ident:  FURY02  /B:0004000

Section Addr    Size    Global  Value   Global  Value   Global  Value

. ABS.  000000  000400    (RW,I,GBL,ABS,OVR)
                        $USRSW  000000  $RF2A1  000000  .VIR    000000
                        $NLCHN  000006  $HRDWR  000006  $SYSV$  000010
                        $WASIZ  000131  $LRECL  000210
OTS$I   000400  020330    (RW,I,LCL,REL,CON)
                        $$OTSI  000400  $SIMRT  000566  $TKS    001416
                        $TKB    001420  $TPS    001422  $TPB    001424
                        $CVTFB  002462  $CVTFI  002462  $CVTCB  002476
                        $CVTCI  002476  $CVTDB  002476  $CVTDI  002476
                        CIC$    002510  CID$    002510  CLC$    002510
                        CLD$    002510  $DI     002510  CIF$    002520
                        CLF$    002520  $RI     002520  CIL$    002626
                        CLI$    002632  $OTI    002662  $$OTI   002664
                        $$SET   004450  RCI$    005062  GCO$    006076
                        FCO$    006104  ECO$    006110  DCO$    006116
                        $CVTIF  007040  $CVTIC  007054  $CVTID  007054
                        CCI$    007066  CDI$    007066  $IC     007066
                        $ID     007066  CFI$    007102  $IR     007102
                        TVL$    007166  $TVL    007166  TVF$    007174
                        $TVF    007174  TVD$    007202  $TVD    007202
                        TVQ$    007210  $TVQ    007210  TVP$    007216
                        $TVP    007216  TVI$    007224  $TVI    007224
                        OCI$    007360  ICI$    007366  $ECI    007402
                        OCO$    007562  ICO$    007570  IFW$    007766
                        $IFW    007772  IFW$$   010034  IFR$    010104
                        $IFR    010110  IFR$$   010146  $CHKER  010170
                        $IOEXI  010214  $EOL    010242  EOL$    010244
                        $OTIS   010360  $$OTIS  010362  SAVRG$  010502
                        THRD$   010660  $PUTRE  010662  $STPS   011170
                        STP$    011176  $STP    011176  FOD$    011202
                        $EXIT   011222  $ERRTB  011346  $ERRS   011453
                        $WAIT   015146  $FCHNL  015210  $INITI  015306
                        $CLOSE  015420  $GETRE  016064  $TTYIN  016140
                        $PUTBL  016274  $GETBL  016504  $EOFIL  016670
                        $EOF2   016704  $FIO    017444  $$FIO   017450
                        $VRINT  020600  $DUMPL  020602
OTS$P   020730  000050    (RW,D,GBL,REL,OVR)
SYS$I   021000  000020    (RW,I,LCL,REL,CON)
                        IPEEK   021000  IPOKE   021010
USER$I  021020  000214    (RW,I,LCL,REL,CON)
                        IBCDI   021100  IBCDO   021134  IADC    021200
$CODE   021234  001662    (RW,I,LCL,REL,CON)
                        $$OTSC  021234  WEIGH   022376
OTS$O   023116  001010    (RW,I,LCL,REL,CON)
                        $$OTSO  023116  $OPEN   023116
SYS$O   024126  000000    (RW,I,LCL,REL,CON)
```

```
$DATAP   024126   000716    (RW,D,LCL,REL,CON)
OTSSD    025044   012734    (RW,D,LCL,REL,CON)
OTSSS    040000   000304    (RW,D,LCL,REL,CON)
                            $AOTS   040302
SYSSS    040304   000004    (RW,D,LCL,REL,CON)
                            $SYSLB  040304  $LOCK   040306  $CRASH  040307
$DATA    040310   000072    (RW,D,LCL,REL,CON)
USERSD   040402   000002    (RW,D,LCL,REL,CON)
,$$$$,   040404   000160    (RW,D,GBL,REL,OVR)
$STACK   040564   000200    (RW,D,LCL,REL,CON)
$STKST   040764   000000    (RW,D,LCL,REL,CON)
                            $S$STK  040764
```

APPENDIX B

PROM/RT-11 ERROR MESSAGE SUMMARY

All error messages issued by the PROM/RT-11 utility follow the standard RT-11 error message format:

?PROM-severity-message

where:

Severity	is F, if the condition is fatal to the operation, or W, if the message is merely a warning of a non-fatal error.
Message	is the text of the error message.

The PROM/RT-11 error messages are listed below, in alphabetical order.

?PROM-F-DIAGNOSTIC failure data XFER from programmer

Reason:	During a transfer of the diagnostic data pattern from the programmer to the computer, an error was detected when the data being transferred was compared with the original data.
Recovery Procedure:	None, PROM,RT-11 continues to cycle through the current test until it is completed.

?PROM-F-DIAGNOSTIC failure data XFER to programmer

Reason:	During a transfer of a diagnostic data pattern to the programmer an error was detected when the data was compared with the data stored in the programmer RAM memory.
Recovery Procedure:	None, PROM/RT-11 continues to cycle through the current test until it is completed.

?PROM-F-DIAGNOSTIC "wrap around" failure

Reason: During a transfer of test data, the serial-line-interface did not receive the same pattern that was transmitted,

Recovery Procedure: None, PROM/RT-11 continues to cycle through the current test until it is completed,

?PROM-F-Error reading input file

Reason: An error occurred while trying to read the input file, This could be the result of a device malfunction, or an attempt by PROM/RT-11 to read past the end of file,

Recovery Procedure: Ensure the addresses given to PROM/RT-11 are within the file and retry the command that caused the error,

?PROM-F-Fatal error in data transfer to blaster

Reason: An error was detected when PROM/RT-11 retried a data transfer to the programmer,

Recovery Procedure: Verify the programmer is operating properly and retry the command, If this error persists, run the DIAGNOSE command tests to determine what hardware component is causing the problem (see Chapter 6),

?PROM-F-File not found

Reason: PROM/RT-11 attempted to open a nonexistent file for input,

Recovery Procedure: Check for a typing error in the command line, Verify that the file name is as you entered it in the command line, Then retry the command,

?PROM-W-File type not .SAV

Reason: The input file specified did not have a .SAV extension. This message is meant to warn you that the PDP-11 computer cannot execute this file. However, you can use data files to program PROM or EPROM chips.

Recovery Procedure: Verify that the file specified in the command is the file you want to use to program the PROM.

?PROM-F-Illegal device

Reason: The device specification used in the command cannot be used to perform the function indicated in the command.

Recovery Procedure: Ensure you entered the command correctly. Use the RT-11 SHOW command to determine if the device name has been reassigned.

?PROM-F-Illegal file specification

Reason: The file specification does not conform to the standard RT-11 file specification format.

Recovery Procedure: Ensure you typed the command line correctly. Retype the command.

?PROM-F-Illegal response, type H for help

Reason: During the request for a COMMAND an illegal response was detected by PROM/RT-11.

Recovery Procedure: Type H followed by a carriage return for help or refer to Chapters 5 and 6 for assistance.

?PROM-F-Input device handler not loaded

Reason: While running in the foreground mode, PROM/RT-11 attempted to open a file on a device whose RT-11 handler was not resident.

Recovery Procedure: Load device handler and retry.

?PROM-F-Internal system error

Error occurred at PC=xxxxxx, PS=yyyyyy, SP=zzzzzz

xxxxxx	xxxxxx	xxxxxx	xxxxxx	xxxxxx	xxxxxx
R0	R1	R2	R3	R4	R5

yyyyyy	yyyyyy	yyyyyy	yyyyyy	yyyyyy	yyyyyy
@SP	2(SP)	4(SP)	6(SP)	10(SP)	12(SP)

Reason: Indicates an unexpected trap to 4 or 10, Or it may also indicate that PROM/RT-11, through a system error, has become confused and cannot recover,

Recovery Procedure: Fill out the problem sheet in Appendix G, Return the problem sheet and the crash dump listing to DIGITAL for analysis and disposition,

?PROM-F-Invalid CSR address

Reason: The address specified does not conform to the requirements for a CSR address,

Recovery Procedure: Ensure you entered the command line correctly, Refer to RT-11 System Generation Manual for a description of a valid CSR address,

?PROM-F-Invalid vector address

Reason: The vector specified does not conform to the requirements for a vector,

Recovery Procedure: Ensure you entered the command line correctly, Refer to the RT-11 System Generation Manual for a description of a valid vector address,

?PROM-F-Maximum number of copies is 99 (decimal)

Reason: The number of copies you entered in response to the COPY command was too large,

Recovery Procedure: Enter a decimal number between 1 and 99 in response to the COPY command,

?PROM-F-Missing file specification

Reason: The correct name of the file used to program

PROM must be completely specified. There is no default.

Recovery Procedure: Enter the correct file name for the file that will be used to program PROM.

?PROM-F-No handler for input device found on SY:

Reason: An attempt was made to use a device handler that was not installed on the system device.

Recovery Procedure: Ensure you entered the command line correctly. Check the system device for the device handler file and retry the command.

?PROM-F-Non-existant CSR address

Reason: CSR address specified does not exist on the PDP-11 bus.

Recovery Procedure: Check serial-line-unit interface for proper configuration and retry the command.

?PROM-F-Number must be decimal

Reason: You entered non-numeric characters.

Recovery Procedure: Ensure you entered the command line or response correctly.

?PROM-F-Number must be octal

Reason: You entered non-numeric characters. Or you entered the numeric characters 8 or 9.

Recovery Procedure: Ensure you entered the command line or response correctly.

?PROM-F-Old vectors out of range

Reason: The vector you entered was not in the range 0 to 476.

Recovery Procedure: Ensure you entered the command line or response correctly.

?PROM-F-Output device handler not loaded

Reason: While running in the foreground mode, you

attempted to open a file and the RT-11 device handler was not resident.

Recovery Procedure: Load device handler and retry the command.

?PROM-F-Output write error

Reason: A hardware error was reported during a write operation.

Recovery Procedure: Ensure there is enough room on the output device and retry the command.

?PROM-F-Programmer is not responding

Reason: Programmer doesn't respond to a valid command.

Recovery Procedure: Ensure power is applied to the PROM programmer. Cycle the POWER switch on the PROM programmer off and then on. If the problem persists use the DIAGNOSE command (see Chapter 6) to check the interface and programmer.

?PROM-F-PROM failed to load into blaster RAM

Reason: The PROM programmer is probably malfunctioning.

Recovery Procedure: Use the DIAGNOSE command (see Chapter 6) to verify proper operation of the PROM programmer and retry the command.

?PROM-F-PROM failed to program

Reason: The PROM chip that you are trying to program is defective or the PROM programmer is defective.

Recovery Procedure: Install another blank PROM and retry the command.

?PROM-W-PROM failed to program; attempting retry

Reason: An error was detected when the PROM programmer attempted to program the PROM chip.

Recovery Procedure: PROM/RT-11 automatically attempts to program

the PROM again.

?PROM-F-PROM failed to verify

Reason: The PROM chip does not contain the same data as the specified file.

Recovery Procedure: None

?PROM-F-PROM is not blank

Reason: The PROM mounted in the programmer is not blank.

Recovery Procedure: Replace the PROM chip with a blank PROM and retry the programming operation.

?PROM-F-PROM is not blank and cannot be reprogrammed with this data

Reason: An attempt was made to modify a PROM with data that is not compatible with the program already in the PROM.

Recovery Procedure: Erase the EPROM or replace PROM and retry the programming operation.

?PROM-W-PROM is NOT blank but can be written with this data

Reason: An attempt was made to modify a PROM with data that is compatible with the program in the PROM chip.

Recovery Procedure: Ensure this PROM is the one that should be modified.

?PROM-W-Start address not on PROM boundary, xxxxxx assumed

Recovery Procedure: None, PROM/RT-11 Will program with the address specified in the error message as the base address.

?PROM-F-System bus timeout error

Reason: PROM/RT-11 attempted to access a non-existent bus address.

Recovery Procedure: Fill out the problem sheet in Appendix G. Return the problem sheet and the crash dump listing to DIGITAL for analysis and

disposition.

?PROM-F-System illegal instruction trap

Reason: An attempt was made to use an illegal instruction.

Recovery Procedure: Fill out the problem sheet in Appendix G. Return the problem sheet and crash dump listing to DIGITAL for analysis and disposition.

?PROM-F-Unsuccessful data transfer from programmer

Reason: While transferring data from the programmer to the computer an error was detected.

Recovery Procedure: Use the DIAGNOSE command (see Chapter 6) to verify the proper operation of the programmer and retry the command if the programmer is operating properly.

?PROM-F-Vectors already in use

Reason: vectors specified have already been protected in the system protect map.

Recovery Procedure: Choose a different vector, reconfigure the serial interface (see the INTERFACE command in Chapter 5) and retry the command.

APPENDIX C

SOFTWARE INSTALLATION PROCEDURE

The following software installation procedure should be used to transfer the PROM/RT-11 utility from the distribution media to the system disk.

The PROM/RT-11 utility is distributed on the RL01 cartridge disk or RX01 floppy diskette.

PROM/RT-11 operates in the foreground or background of the RT-11 operating system. The distribution media contains two files; one for foreground operation named PROM.REL and one for background operation called PROM.SAV.

For systems where PROM/RT-11 operates in the foreground, the commands required to transfer PROM/RT-11 to the system disk are:

```
COP[Y] DLn:PROM,REL SY:  <RETURN> (for RL01)
COP[Y] DXn:PROM,REL SY:  <RETURN> (for RX01)
```

where

n is the unit number of the device the distribution media is mounted on.

For systems where PROM/RT-11 operates in the background, the commands required to transfer PROM/RT-11 to the system disk are:

```
COP[Y] DLn:PROM,SAV SY:  <RETURN> (for RL01)
COP[Y] DXn:PROM,SAV SY:  <RETURN> (for RX01)
```

where

n is the unit number of the device the distribution media is mounted on.

APPENDIX D

IC DESCRIPTIONS

This appendix contains a description of the integrated circuits that can be used on LSI-11 memory option modules.

D.1 2708: 1K X 8 BIT UV ERASABLE PROM

The 2708 PROM chip (see Figure B-1) is an ultra-violet (UV) light erasable, field programmable, Erasable Programmable Read-Only Memory (EPROM) that can be used for non-volatile storage. Field programmable means that the chip can be programmed with the customer's program or data by the PROM/RT-11. Each chip provides up to 1024(1K) 8-bit bytes of storage for data or programs. Two chips are required to store 1K words of PDP-11 data or programs.

The 2708 EPROM is packaged in a standard 24-pin package and includes a transparent quartz cover over the integrated circuit chip.

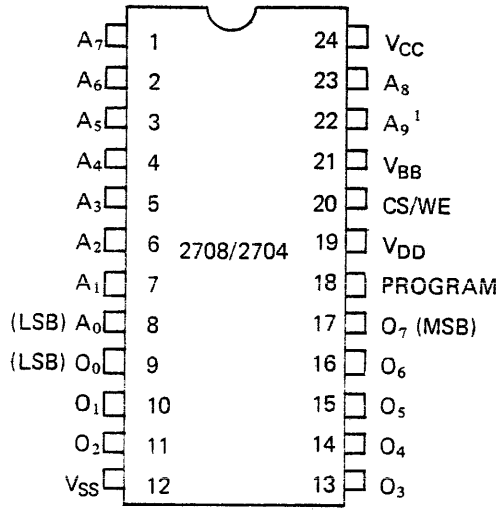
The chip can be erased by exposing the chip to 2537 A of UV light for approximately 10 to 30 minutes.

Operating power required is +5V and -5V. Maximum access time is 350ns (nanoseconds). Address and control inputs are TTL logic compatible. Output pins are three-state TTL logic signals. The third state is a high-impedance condition that effectively disconnects the chip's data output from the output pins. This allows the use of a data bus shared by two or more similar devices.

The unprogrammed (or erased) PROM contents are all 1's (high state). Loading data into the PROM introduces logic 0's (low state).

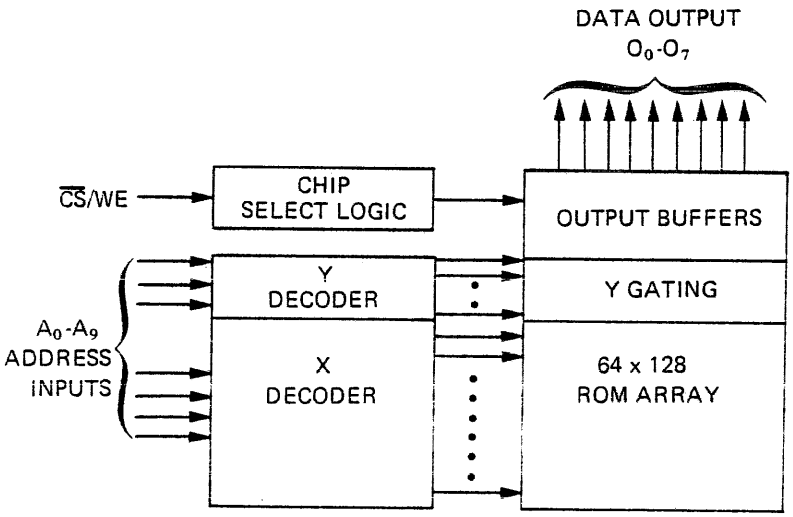
The 2708 is used on the MRV11-BA memory-option module. The module has space for eight chips and up to 4K of memory.

PIN CONFIGURATION



NOTE 1: PIN 22 MUST BE CONNECTED TO V_{SS} FOR THE 2704.

BLOCK DIAGRAM



PIN NAMES

A_0-A_8	ADDRESS INPUTS
O_1-O_8	DATA OUTPUTS-INPUTS
$\overline{CS}$/WE	CHIP SELECT WRITE ENABLE INPUT

PIN CONNECTION DURING READ OR PROGRAM

	PIN NUMBER							
MODE	DATA I/O 9-11, 13-17	ADDRESS INPUTS 1-8, 22,23	V_{SS} 12	PROGRAM 18	V_{DD} 19	$\overline{CS}$/WE 20	V_{BB} 21	V_{CC} 24
READ	D_{OUT}	A_{IN}	GND	GND	+12	V_{IL}	-5	+5
DESELECT	HIGH IMPEDANCE	DON'T CARE	GND	GND	+12	V_{IH}	-5	+5
PROGRAM	D_{IN}	A_{IN}	GND	PULSED +6V	+12	V_{IHW}	-5	+5

Figure D-1 2708 Memory Chip Schematic and Data

D.2 2716: 2K X 8 UV ERASABLE PROM

The 2716 PROM chip (see Figure D-2) is an ultra-violet (UV) light erasable, field programmable, Erasable-Programmable-Read-Only Memory (EPROM) that can be used for non-volatile storage. Field programmable means that the chip can be programmed with your program or data by the PROM/RT-11. Each chip provides up to 2048(2K) 8-bit-bytes of storage for data or programs. Two chips are required to store 2K words of PDP-11 data or programs.

The 2716 EPROM is packaged in a standard 24-pin package and includes a transparent quartz cover over the integrated chip.

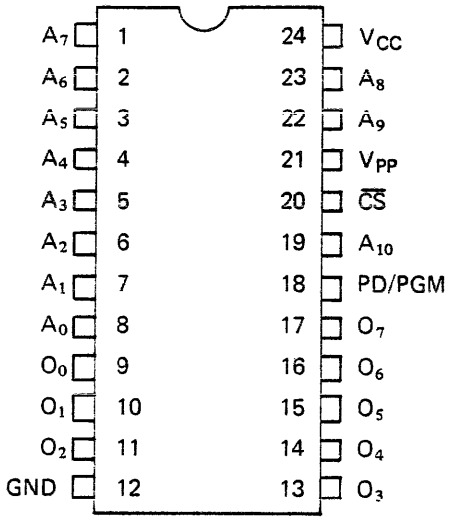
The chip can be erased by exposing the chip to 2537 A of UV light for approximately 10 to 30 minutes.

Operating power required is +5V. Maximum access time is 450ns. Address and control inputs are TTL logic compatible.

The unprogrammed (or erased) PROM contents are all 1's (high state). Loading data into the PROM introduces logic 0's (low state).

The 2716 is used on the BDV11-A memory option module. The module has space for 16 chips and up to 16K of memory.

PIN CONFIGURATION



MODE SELECTION

MODE \ PINS	PD/PGM (18)	$\overline{CS}$ (20)	V_{PP} (21)	V_{CC} (24)	OUTPUTS (9-11,13-17)
Read	V_{IL}	V_{IL}	+5	+5	D_{OUT}
Deselect	Don't Care	V_{IH}	+5	+5	High Z
Power Down	V_{IH}	Don't Care	+5	+5	High Z
Program	Pulsed V_{IL} to V_{IH}	V_{IH}	+25	+5	D_{IN}
Program Verify	V_{IL}	V_{IL}	+25	+5	D_{OUT}
Program Inhibit	V_{IL}	V_{IH}	+25	+5	High Z

PIN NAMES

A_0-A_{10}	ADDRESS
PD/PGM	POWER DOWN/PROGRAM
$\overline{CS}$	CHIP SELECT
O_0-O_7	OUTPUTS

BLOCK DIAGRAM

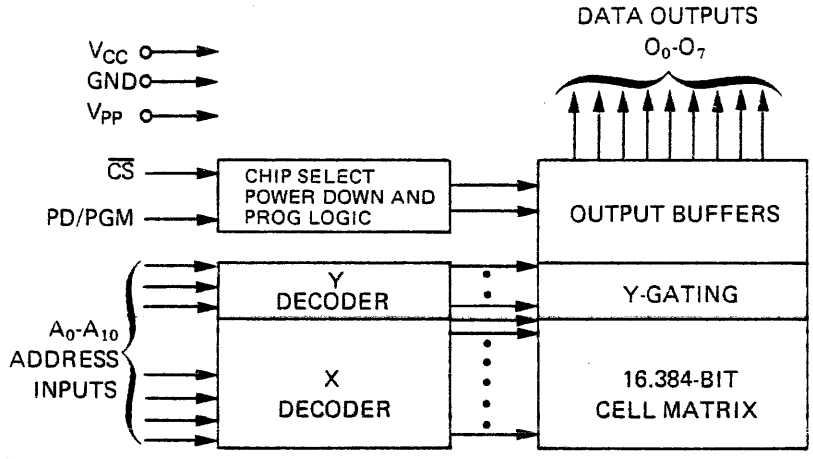


Figure D-2 2716 Memory Chip Schematic and Data

D.3 2732 4K X 8 UV ERASABLE PROM

The 2732 PROM chip (see Figure D-3 is an ultra-violet (UV) light erasable, field programmable, Erasable-Programmable-Read-Only Memory (EPROM) that can be used for non-volatile storage. Field programmable means that the chip can be programmed with your program or data by the PROM/RT-11. Each chip provides up to 4096(4K) 8-bit-bytes of storage for data or programs. Two chips are required to store 4K words of PDL-11 data or programs.

The 2732 EPROM is packaged in a standard 24-pin package and includes a transparent-quartz cover over the integrated chip.

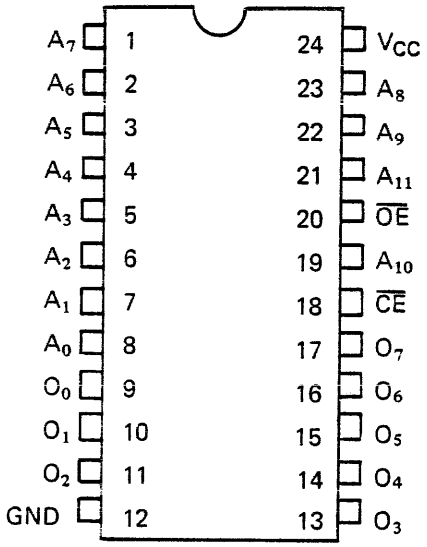
The chip can be erased by exposing the chip to 2537 A of UV light for approximately 10 to 30 minutes.

Operating power required is +5V. Maximum access time is 450ns. Address and control inputs are TTL logic compatible.

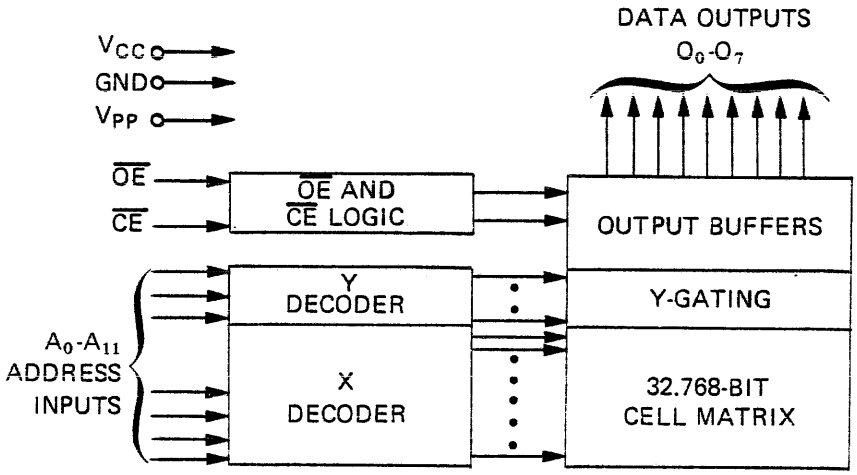
The unprogrammed (or erased) PROM contents are all 1's (high state). Loading data into the PROM introduces logic 0's (low state).

Call your local DIGITAL Sales Representative for information about modules that use the 2732 chip.

PIN CONFIGURATION



BLOCK DIAGRAM



PIN NAMES

A_0-A_{10}	ADDRESS
CE	CHIP ENABLE
OE	OUTPUT ENABLE
O_0-O_7	OUTPUTS

Figure D-3 2732 Memory Chip Schematic and Data

D.4 82S129 256 X 4 BIT PROM

The 82S129 PROM chip (see Figure D-4) is a field programmable, Programmable Read Only Memory (PROM) that can be used for non-volatile storage of your programs or data. Field programmable means that you can program the chip with your program or data using the PROM/RT-11. Each chip stores 256 4-bit bytes. Four chips are required to store 256 words of PDP-11 programs or data.

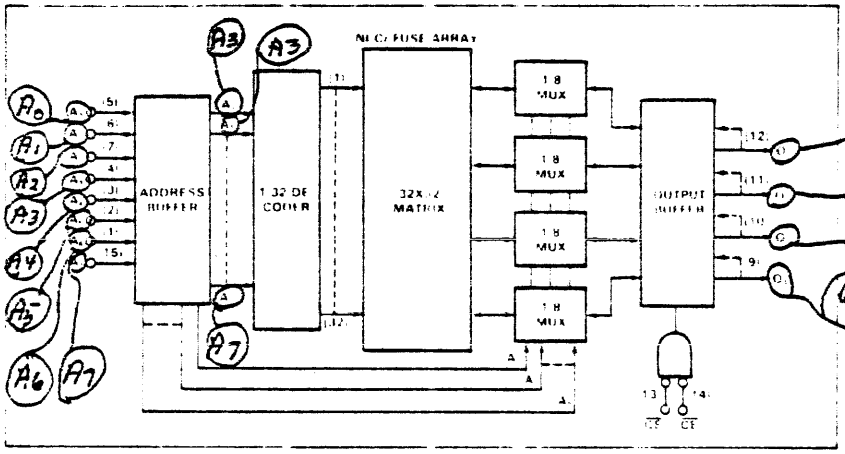
The 82S129 EPROM is packaged in a standard 16-pin package.

Operating power is +5.5Vdc. Maximum access time is 70ns. Address and control inputs are TTL logic compatible. Output pins are three-state TTL logic signals. The third state is a high-impedance condition that effectively disconnects the chips data output from the output pins. This allows the use of a data bus shared by two or more similar devices.

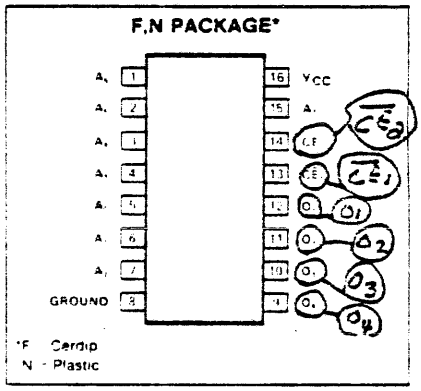
Unprogrammed outputs are logical low. The PROM/RT-11 programs the output to a logical high where the bit is a logical one.

The 82S129 is used on the MRV11-A LSI-11 memory option module. The module has space for 32 chips and up to 2K of memory.

BLOCK DIAGRAM



PIN CONFIGURATION



ABSOLUTE MAXIMUM RATINGS

PARAMETER		RATING	UNIT
V_{CC}	Supply voltage	+7	Vdc
V_{IN}	Input voltage	+5.5	Vdc
	Output voltage		Vdc
V_{OH}	High (82S126)	+5.5	
V_O	Off-state (82S129)	+5.5	
	Temperature range		°C
T_A	Operating		
	N82S126/129	0 to +75	
	S82S126/129	-55 to +125	
T_{STG}	Storage	-65 to +150	

Figure D-4 82S129 Memory Chip Schematic and Data

D.5 82S131: 512 X 4 BIT PROM

The 82S131 PROM chips are field programmable, Programmable Read Only Memory (PROM) that can be used for non-volatile storage, Field programmable means that you can program the chip with your program or data using the PROM/RT-11, Each chip provides up to 512 words of PDP-11 programs or data,

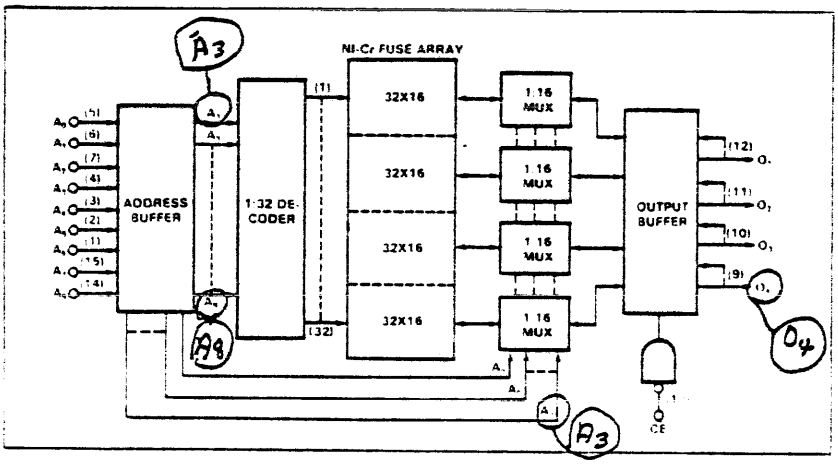
The 82S131 is packaged in a standard 16-pin package,

Operating power required is +5Vdc, Maximum access time is 50ns, Address and control inputs are TTL logic compatible, Output pins are three-state TTL logic signals, The third state is a high impedance condition that effectively disconnects the chip's data output, This allows the use of a data bus shared by two or more similar devices,

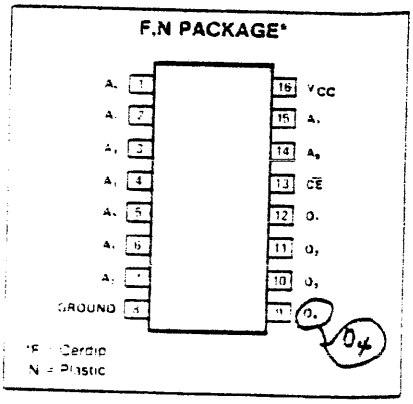
Unprogrammed outputs are at a logical low state, The PROM/RT-11 programs outputs to a logical high state,

The 82S131 is used on the MRV11-A memory option module, The module has space for 32 chips or up to 4K of memory,

BLOCK DIAGRAM



PIN CONFIGURATION



ABSOLUTE MAXIMUM RATINGS

	PARAMETER	RATING	UNIT
V_{CC}	Supply voltage	-7	Vdc
V_{IN}	Input voltage	-5.5	Vdc
	Output voltage		Vdc
V_{OH}	High (82S130)	-5.5	
V_O	Off-state (82S131)	-5.5	
	Temperature range		°C
T_A	Operating		
	N82S130/131	0 to +75	
	S82S130/131	-55 to +125	
T_{STG}	Storage	-65 to +150	

Figure D-5 82S131 Memory Chip Schematic and Data

D.6 82S181 1K X 8 BIT PROM

The 82S181 PROM chips (see Figure D-6) are field programmable, Programmable Read Only Memory (PROM) that can be used for non-volatile storage. Field programmable means that you can program the chip with your program or data using the PROM/RT-11 system. Each chip provides up to 1024 (1K) 8-bit bytes of storage for programs or data. Two 82S181 chips are required to store 1K words of PDP-11 data or programs.

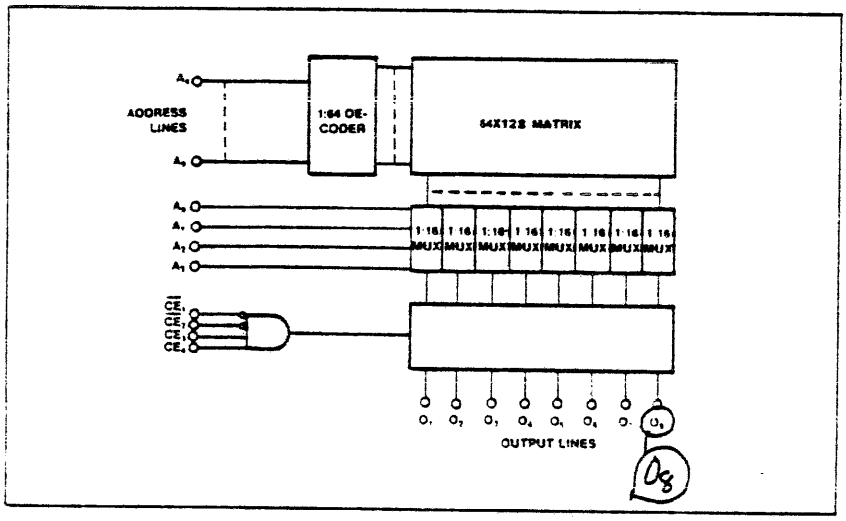
The 82S181 is packaged in a standard 24-pin package (see Figure D-6).

Operating power is +5.5Vdc. Maximum access time is 70ns. Address and control inputs are TTL logic compatible. Output pins are three-state TTL logic signals. The third state is a high impedance condition that effectively disconnects the chips data output from the output pins. This allows the use of a data bus shared by two or more similar devices.

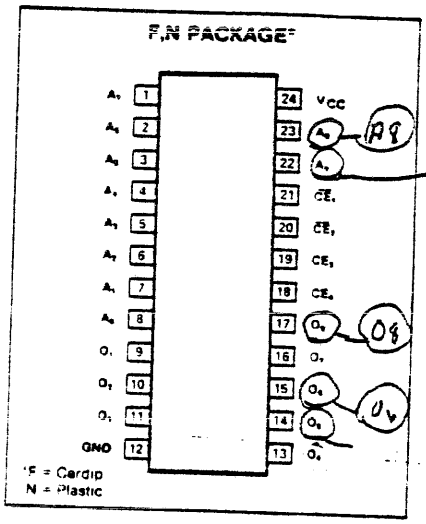
The unprogrammed (or erased) are at a logical low (all zeros). Loading data into the PROM introduces logic 1's (high state).

Call your local DIGITAL Sales Representative for information about LSI-11 memory modules that use the 82S181 chip.

BLOCK DIAGRAM



PIN CONFIGURATION



ABSOLUTE MAXIMUM RATINGS

PARAMETER		RATING	UNIT
V_{CC}	Supply voltage	+7	Vdc
V_{IN}	Input voltage	+5.5	Vdc
	Output voltage		Vdc
V_{OH}	High (82S180)	+5.5	
V_O	Off-state (82S181)	+5.5	
	Temperature range		°C
T_A	Operating		
	N82S180/181	0 to +75	
	S82S181	-55 to +125	
T_{STG}	Storage	-65 to +150	

Figure D-6 82S181 PROM Chip Schematic and Data

D.7 82S191: 2K X 8-BIT PROM

The 82S191 PROM chips (see Figure D-7) are field programmable, Programmable Read Only Memory (PROM) that can be used for non-volatile storage of programs or data. Field programmable means that you can program the chip with your program or data using the PROM/RT-11. Each chip stores 2048 8-bit bytes of programs or data. Two chips are required to store 2K words of PDP-11 data or programs.

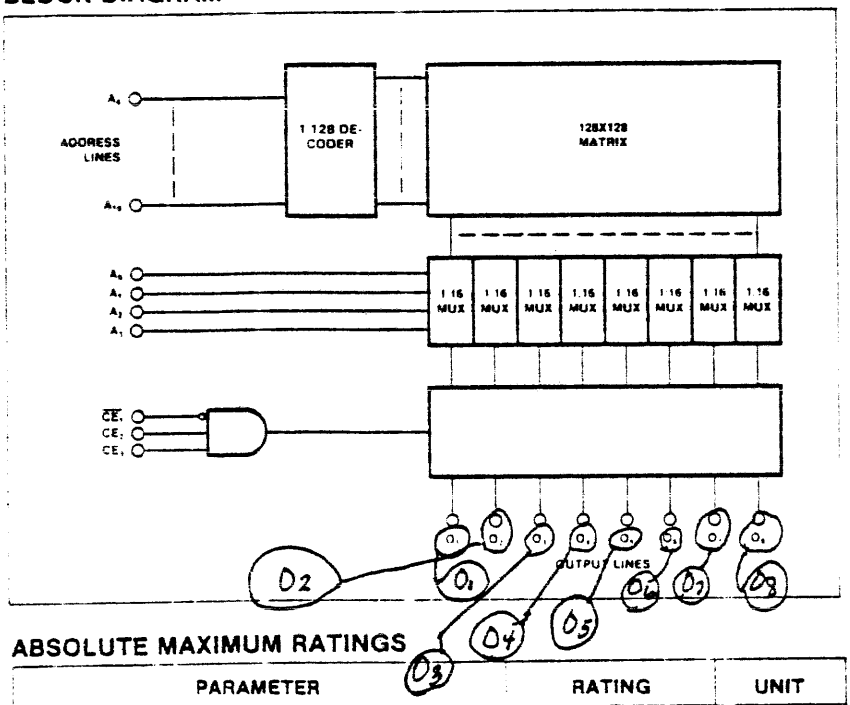
The 82S191 is packaged in a standard 24-pin package.

Operating power is 5.5Vdc. Maximum access time is 80ns. Address and control inputs are TTL compatible. The output pins are three-state TTL logic signals. The third state is a high impedance condition that effectively disconnects the chips data output from the output pins. This allows the use of a data bus shared by two or more similar devices.

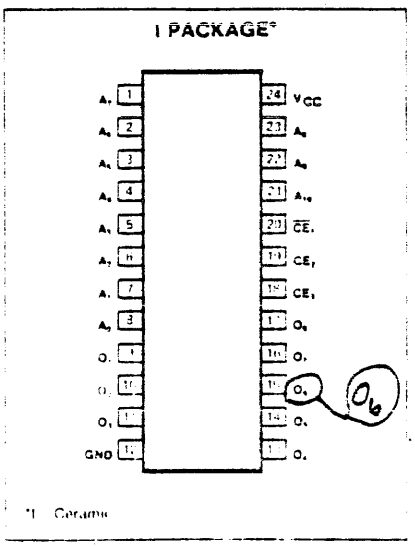
Unprogrammed outputs are logical low (all 0's). The PROM/RT-11 programs the output to a logical high (1) where the bit is a logical one.

Call your local DIGITAL Sales Representative for information about LSI-11 memory modules that use the 82S191 chip.

BLOCK DIAGRAM



PIN CONFIGURATION



ABSOLUTE MAXIMUM RATINGS

PARAMETER		RATING	UNIT
V_{CC}	Supply voltage	+7	Vdc
V_{IN}	Input voltage	+5.5	Vdc
	Output voltage		Vdc
V_{OH}	High (82S140)	+5.5	
V_O	Off-state (82S141)	+5.5	
	Temperature range		°C
T_A	Operating		
	N82S190/191	0 to +75	
	S82S190/191	-55 to +125	
T_{STG}	Storage	-65 to +150	

Figure D-7 82S191 PROM Chip Schematic and Data

APPENDIX E

OPERATOR INSTRUCTIONS FOR PRODUCTION OPERATIONS

The copy command is used to program PROM chips in production quantities. The PROGRAM command is used to create a master chip or chips and the master chip or chips is used to program other chips with the same program or data.

Someone familiar with the PROM/RT-11 system should program the master chips to be used in the production operation. After the master chips are programmed, the operator need only invoke the COPY command and follow the instructions that PROM/RT-11 prints at the terminal. The operator only has to know the number of copies to make, be able to answer the questions that PROM/RT-11 asks, and mount the blank chips after each chip is programmed.

E.1 INVOKING PROM/RT-11

Before you can use the COPY command you must invoke the PROM/RT-11 utility. If it is already invoked go to Section E.2.

To invoke PROM/RT-11 type:

```
.FRUN PROM <RETURN>
```

The command must be typed exactly as shown and it must be followed by a carriage return as indicated by the symbol <RETURN>. The . is printed at the terminal to indicate that the system is ready to accept a command.

When PROM/RT-11 is invoked, it identifies itself by printing the following message on the terminal:

```
PROM/T-11 vxx.yy PROM size mmmm by n
```

Where:

vxx.yy — is the version of the RT-11 operating system.

mmmm — is the number of words per chip for the personality card installed in the programmer.

n is the number of bits per word for the personality card installed in the programmer.

Example:

```
PROM/RT-11 V3.0 PROM size 1024 x 8
```

In the example the version of RT-11 being used is version 3.0 and the personality card installed in the programmer is for a 1024 word by 8 bit chip.

E.2 INVOKING THE COPY COMMAND

To invoke the COPY command, type the following when the prompt:

```
Command:
```

is printed at the terminal.

```
Command: C[OPY]<RET>
```

Type a C followed by a carriage return. After the COPY command is typed the PROM/RT-11 utility program prints the following message at the terminal.

```
Mount master PROM and type <RETURN>:
```

At this time, insert the master PROM chip in the blaster socket. After the master chip is inserted in the socket, press the <RETURN> key and a slight pause occurs. After the pause, PROM/RT-11 will tell you to remove the master PROM chip. The following message is printed:

```
Remove master PROM and mount blank PROM; type <RETURN>:
```

When this message appears, remove the master PROM chip from the blaster socket. Replace it with a blank PROM chip that will become the first copy. After inserting the blank PROM, type <RETURN>. The utility program will ask how many copies to generate by printing the following message:

```
How many copies (decimal)?
```

You should respond by typing in the number of copies that you want to make followed by a carriage return.

The COPY command can program up to 99 chips for each invocation of the COPY command. If more than 99 copies are required, the COPY command should be invoked a second time or as many times as necessary. You should make sure that you have enough blank PROM chips for the number of copies specified. After entering the number of copies, you must type a carriage return. This begins the copying process. If the user inadvertantly mounts a non-blank PROM chip the following error message and instructions are printed at the terminal:

?PROM-F-PROM is not blank

Remove current PROM and insert blank PROM, type <RETURN>:

Remove the non-blank PROM and replace it with a blank or erased PROM. When the <RETURN> key is pressed, PROM/RT-11 programs the blank PROM.

When programming of the PROM chip is completed, PROM/RT-11 generates the following message:

Remove copy number NN and mount blank PROM, type <RETURN>:

Where:

NN is the number of the copy just completed.

Remove the programmed PROM chip and replace it with a blank chip. You may wish to label the duplicate chip with the copy number. Install another blank chip and press the <RETURN> key to continue. A copy is created each time you repeat this dialogue.

APPENDIX F

ASSEMBLY LANGUAGE DEFINITION MODULE FOR PROM APPLICATIONS

The program module given below is provided as an aid to users developing PROM applications in assembly language, It provides 3 essential services as follows:

1. Forces a .PSECT (program section) ordering sequence; that is absolute section (PROM) followed by pure data section (PROM) followed by scratch data section (RAM)

2. Allocates the stack

3. Handles startup of application from vectors at locations 24(8) and 26(8) including setup of stack pointer

ROMDEF.MAC MODULE

```
        .ASECT
.=24                            ;Origin to Power-Up Vector
        .WORD   PWRUP           ;PC of startup routine
        .WORD   340             ;Start with interrupts disabled

        .PSECT  ROM             ;Next to load is PROM section
PWRUP:  MOV     #$$$STK,SP      ;Initialize stack pointer
        CLR     -(SP)           ;Create interrupt-enabled PS
        MOV     @#40,-(SP)      ;And initial program counter
        RTI                     ;Start application execution

        .PSECT  $STACK,D        ;Stack section is first in RAM
        .BLKW   64.             ;Reserve 64 words by default
        .PSECT  $STKST,D        ;Set up section to load after stack
$$$STK::                        ;Defining initial stack pointer value

        .PSECT  RAM,D           ;Last section to load is RAM

        .END                    ;End of ROMDEF.MAC
```

NOTE

This module uses RT-11's location 40 value, and hence will not work in RSX or

IAS environments.

Once the ROMDEF module has been assembled to produce ROMDEF.OBJ, the linking process for applications which have been coded with the listed .PSECTs is:

```
.LINK/BOTTOM:250 /BOUNDARY:020000 /LDA /EXE:APPLIC
Files? ROMDEF,user-file-1,...,user-file-n
Boundary section? $STACK
```

To extend the size of the stack space to mmmmmm (octal), use:

```
.LINK/BOTTOM:250 /BOUNDARY:020000 /LDA /EXE:APPLIC /EXTEND:mmmmmm
files? ROMDEF,user-file-1,...,user-file-n
Extend section? $STACK
Boundary section? $STACK
```

APPENDIX G

PROM/RT-11 PROBLEM REPORT FORM

User's Name:

Location:
Local DIGITAL Sales Office:
Telephone Number (include area code):
PDP-11 Computer Used:
Mass Storage Device (RL01 or RX01):
Operating System and Version:

PROM/RT-11 Version:

Description of Problem:

NOTE

Include the crash dump listing with your problem report.