iUP-200/201
UNIVERSAL PROGRAMMER
USER'S GUIDE

Order Number: 162613-001

Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051

CAUTION

This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instruction manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A Computing Device pursuant to Subpart J of Part 15 of FCC rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user, at his own expense, will be required to take whatever measures may be required to correct the interference.

Additional copies of this manual or other Intel literature may be obtained from:

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

The information in this document is subject to change without notice.

Intel Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Intel Corporation assumes no responsibility for any errors that may appear in this document. Intel Corporation makes no commitment to update nor to keep current the information contained in this document.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied.

The following are trademarks of Intel Corporation and its affiliates and may be used only to identify Intel products:

BXP	Insite	iSBC	Multibus
CREDIT	Intel	iSBX	Multimodule
i	Int$_e$l	Library Manager	Plug-A-Bubble
ICE	Intelevision	MCS	PROMPT
iCS	Intellec	Megachassis	RMX/80
im	iOSP	Micromainframe	System 2000
iMMX	iRMX	Micromap	UPI

A552/182/ 2K DD

REV.	REVISION HISTORY	DATE
-001	Original Issue	2/82

PREFACE

About this Manual

This manual describes how to use the Intel iUP-200/201 Universal Programmer to store programs and data into programmable read-only memory (PROM) devices. It is intended for use by engineers and designers who are developing firmware for ROM, PROM, EPROM or E^2PROM-based systems.

The following paragraphs describe the general contents of each chapter. An Index is provided at the end of the book.

Chapter 1 provides a general description of the Universal Programmer and a tutorial overview of PROM programming and firmware development.

Chapter 2 describes how to set up and initialize the Universal Programmer system.

Chapter 3 covers the on-line operation of the Universal Programmer and describes the Intel PROM Programming Software (iPPS) commands.

Chapter 4 covers the off-line operation of the Universal Programmer and describes the front panel operation of the iUP-201.

Chapter 5 provides a number of programming examples using the Universal Programmer. They illustrate the use of the Universal Programmer in the typical firmware development applications.

Appendix A describes error conditions and error messages.

Appendix B describes how to control the Universal Programmer in its on-line mode when it is connected as an RS-232 peripheral to a computer other than an Intellec Microcomputer Development System.

Appendix C contains information about the various disk file formats that the iPPS software reads and writes.

Appendix D contains reference tables for numeric conversion and ASCII codes.

Conventions Used in this Book

For simplicity throughout this book, Universal Programmer is used to refer generally to the iUP-200/201 product. Likewise, the iPPS software is used to refer to the Intel PROM Programming Software, which is used to operate the iUP-200.

The following symbol conventions are used in this manual to document Notes, Cautions, and Warnings:

A section of text introduced by the symbol

NOTE

gives emphasis to comments with special significance for the user.

A section of text introduced by the symbol

CAUTION

gives instructions necessary to avoid possible damage to equipment or loss of stored information.

A section of text introduced by the symbol

WARNING

gives instructions necessary for safety reasons.

Other Pertinent Intel Literature

While this Guide is a self-contained document describing the use of the Universal Programmer, several other Intel documents are related to firmware development and to the development of microprocessor-based systems:

- iUP-xx Personality Module User's Guide

  Order Number: several

  Each Intel Personality Module has its own User's Guide describing its installation, applicable devices, applicable hosts, and unique characteristics. Consult the current Intel Component Data Catalog to determine which Personality Module to use for specific PROM devices. The appropriate Personality Module User's Guide is supplied with each Personality Module. Additional copies can be obtained separately from the Intel Literature Department. See the back of the title page in this book for the address.

- iUP-200/201 Universal Programmer Pocket Reference

  Order Number: 162614

  This publication is a short pocket reference describing the Universal Programmer operation and command syntax.

- ISIS-II User's Guide

  Order Number: 9800306

  This manual is a comprehensive User's Guide to the Intel Operating System under which the iUPS-200 Universal Programming Software runs.

- Intel Systems Data Catalog (current edition)

  Order Number: varies

  This catalog provides complete specifications on most Intel standard memory systems, single board computers, Intellec microcomputer development systems, software and peripherals.

- Intel Component Data Catalog (current edition)

  Order Number: varies

  This catalog is a comprehensive catalog of Intel components with their technical specifications.

SERVICE INFORMATION

United States customers can obtain service and repair assistance from Intel Corporation by contacting the Intel Product Service Hotline in Phoenix, Arizona. Customers outside the United States should contact their sales source (Intel Sales Office or Authorized Distributer) for service information and repair assistance.

Before calling the Product Service Hotline, have the following information available:

1. Date the product was received.

2. Complete part number of the product (including dash number). On boards, this number is usually silk-screened onto the board. On other products, it is usually stamped on a label.

3. Serial number of product. On boards, this number is usually stamped on the board. On other products, the serial number is usually stamped on a label.

4. Shipping and billing address.

5. If the Intel Product warranty has expired, a purchase order number for billing purposes must be provided.

6. Make sure to advise the Hotline personnel if an extended warranty agreement is in effect.

Use the following numbers for contacting the Intel Product Service Hotline:

Telephone:
All U.S. locations (except Alaska, Arizona, and Hawaii):
(800) 528-0595
All other locations:
(602) 869-4600

TWX Number:
910 - 951 - 1330

Always contact the Product Service Hotline before returning a product to Intel for repair. The Hotline personnel will provide a repair authorization number, shipping instructions, and other important information which enable fast, efficient service. If the product is being returned because of damage sustained during shipment or if the product is out of warranty, a purchase order is required before Intel can initiate the repair.

CONTENTS

CONTENTS (continued)

CONTENTS (continued)

ILLUSTRATIONS

TABLES

CHAPTER 1
GENERAL INFORMATION

This chapter contains a general description of the iUP-200/201 Universal Programmer, an overview of firmware development, and a description of system configurations.

Overview of the Universal Programmer

The iUP-200 Intel Universal Programmer programs and verifies Intel Programmable Read Only Memory (PROM) components and other Intel integrated circuits that contain PROM. It can be used to program: non-erasable bi-polar PROMs, erasable PROMs (EPROMs) and electrically erasable PROMs (E^2PROMs). It can also be used to read ROM (read only memory), in certain ROM-based components.

There are two models of the Intel Universal Programmer: the iUP-200, which operates on line as a peripheral of an Intel Intellec Microprocessor Development System; and the iUP-201, which can be operated either on or off line.

The Universal Programmer accommodates the various PROM devices through the use of Personality Modules. Personality Modules are small units that are plugged into the front of the Universal Programmer. They personalize the Universal Programmer for a specific PROM or family of PROM devices.

In the on-line mode, the Intel PROM Programming Software (iPPS) controls the Universal Programmer.

The iPPS software is a utility that runs under the ISIS-II Operating System. Its controls the operation of the Universal Programmer through a serial I/O port on the development system. The iPPS software provides a comprehensive set of commands that allow the following operations to be performed:

- Reading and writing data to and from disk files
- Manipulating the data in the Intellec memory Buffer
- Mapping the data for a particular PROM word size
- Interleaving data for different addressing schemes
- Programming a particular PROM device
- Reading back the contents of that device
- Verifying the contents of the PROM device

The iUP-201 can be used as a stand-alone programming system in the off-line mode. Three basic kinds of off-line operation can be performed:

- Duplicating and verifying a PROM
- Downloading data from any host system that has an RS-232 interface and programming a PROM with the data. (This feature requires that the data be in Intel 8080 hexadecimal format.)
- Displaying and editing data in local read only memory (RAM).

These off-line operations are performed independent of any host computer system.

The user can easily expand an iUP-200 to full iUP-201 capability with the installation of the off-line option. This option can be installed without service assistance. The off-line option consists of:

- A keyboard with a numeric (Hex) keypad and various function keys.
- An alphanumeric display.
- A circuit board that contains control circuitry and 16K bytes of local RAM. Throughout this manual, the local RAM is referred to as URAM (for User RAM).

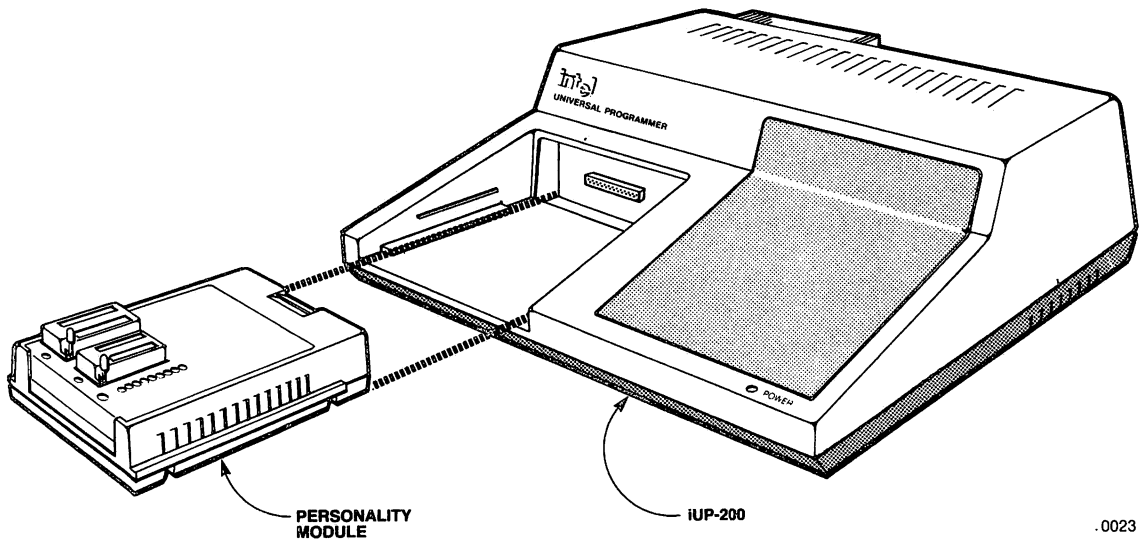


Figure 1-1. iUP-200 with Personality Module

General Description

The Universal Programmer is housed in a low profile enclosure with an angled front panel. The ON/OFF switch is located on the back panel and a POWER on indicator light is located on the front panel.

Air convection through vents at the rear, on the top and bottom of the enclosure, cool the unit. The left side of the front panel is a recepticle area for a Personality Module. Figure 1-1 shows the basic iUP-200 with an un-installed Personality Module beside it.

The iUP-201 is identical to the iUP-200 except that it has the off-line option installed. Figure 1-2 shows an iUP-201 with a Personality Module installed.

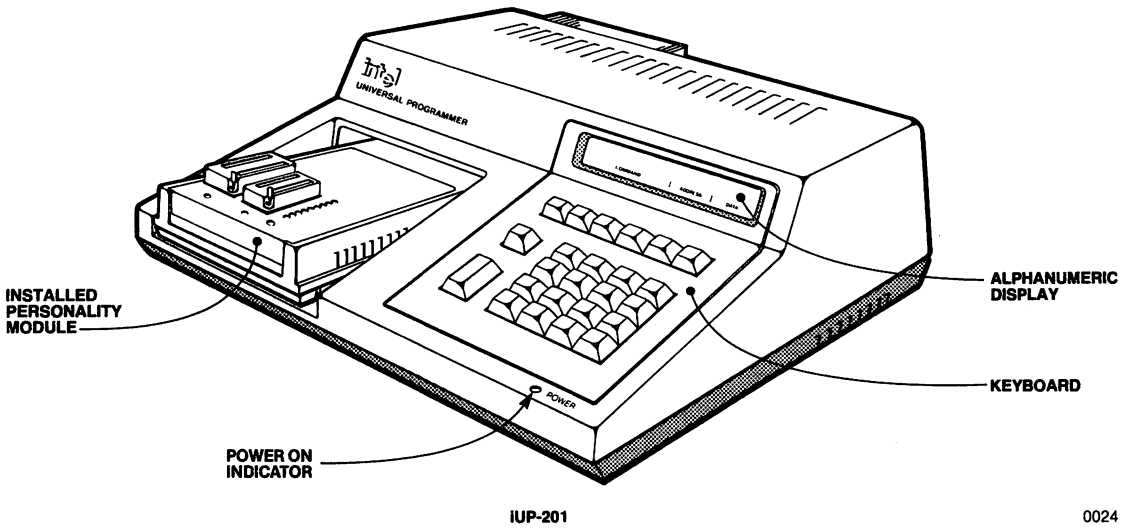


Figure 1-2. iUP-201 with an Installed Personality Module

The hand-sized Personality Modules plug into the Universal Programmer. Each has one or more device sockets. LEDs indicate the PROM type selected and the socket to be used. Consult the current Intel Component Data Catalog or Intel Systems Data Catalog for further information on the Personality Module required for specific PROMs. Refer to the back of the title page of this manual for literature ordering information.

The off-line option of the iUP-201 occupies the right half of the front panel and consists of an alphanumeric display section and a keyboard

section. The alphanumeric display has labeled fields for Command,Address, and Data. The keyboard section contains a numeric (hex) keypad and various function keys. The off-line option also adds 16K of user accessible RAM (URAM). Sockets are provided on the circuit board, which allow the user to easily expand this local RAM to 32K; no service assistance is needed. Expansion of the iUP-201 URAM is required in order to use certain Personality Modules. See the current Intel System Data Catalog for information about what Personality Modules require the expansion to 32K.

The back panel of the Universal Programmer (shown in Figure 1-3) is the same for both the iUP-200 and the iUP-201. When facing the back panel, the main power switch is a rocker switch located in the upper left corner. To its right are two line voltage selection switches. The main power fuse is located below the power switch. The AC power connector is in the lower left corner. An RS-232 connector is provided at the bottom center.

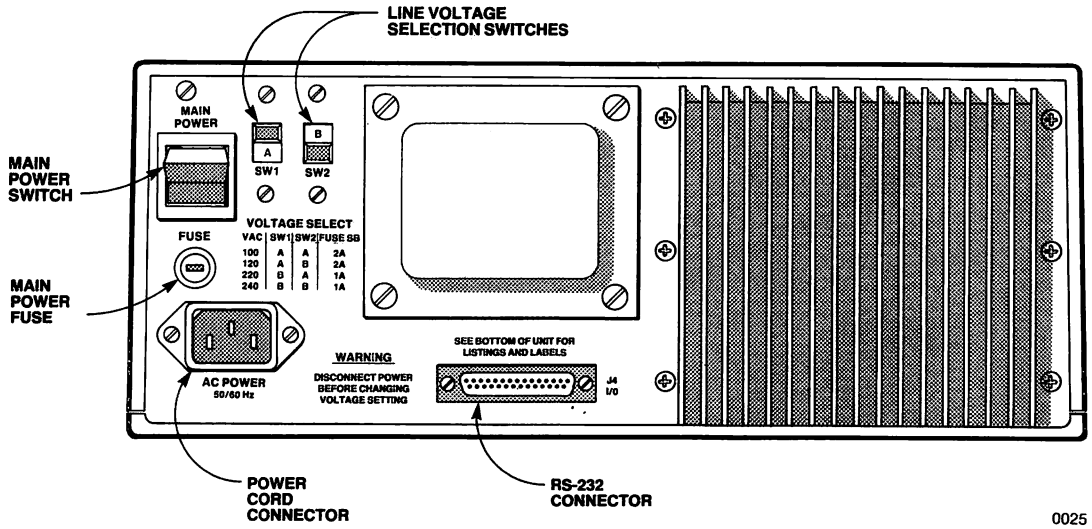


Figure 1-3. Universal Programmer Back Panel

The various safety listing approval labels and the serial number identification plate are located on the bottom of the Universal Programmer.

Physical Specifications

Dimensions:	iUP-200/201:	Depth Width Height 15" 15" 6"
	Personality Module:	Depth Width Height 7" 5.5" 1.6"
Weight:	iUP-200:	15 pounds maximum
	iUP-201:	15 pounds maximum
	Personality Module:	1 pound maximum
Operating Temperature Range:		10 to 40 degrees C.
Operating Humidity Range:		20% to 80% RH (non-condensing)

Electrical Specifications

Operating Voltage:	Selectable 100, 120, 220, or 240 VAC (each within -10% to +10%) at 47-63 Hz single phase grounded.
Power Consumption:	iUP-200, Maximum 120 watts iUP-201, Maximum 120 watts
Fuse Protection:	100 VAC, use 2 ampere slow blow 120 VAC, use 2 ampere slow blow 220 VAC, use 1 ampere slow blow 240 VAC, use 1 ampere slow blow

The Universal Programmer is an intelligent peripheral for an Intellec development system. A dedicated Intel 8085 microprocessor with its own RAM and firmware allows it to perform many operations, independent of the host. The Universal Programmer is connected to a host computer via a two way serial interface.

Refer to the section on "Compatible Hosts" in Chapter 2 for further information on interfacing the Universal Programmer.

Firmware Development

Microprocessors and microcomputers are becoming a central element in the design of electronic products. The typical development process for microprocessor and microcomputer based systems involves an integration of hardware and software. Once the software has been perfected, it is often installed permanently in a Read Only Memory (ROM) device. Software that has been installed in ROM is referred to as firmware. Firmware is used in systems design because of its relative low cost, high speed, and data non-volatility. Non-volatility means that firmware is retained even when power to the system is turned off.

Intel manufactures a wide variety of ROM memory components, which can be divided into two general types: masked ROMs and electrically programmable ROMs (PROMs).

Firmware is fabricated into a masked ROM during manufacturing. This is often the cost effective method for including firmware in a mass produced product.

Programs and data is electrically programmed into a PROM device after it has been manufactured. To electrically program a PROM, data is presented to a particular address. A specified voltage is then applied to programming pins to set or "burn in" the code in the selected cell.

There are three kinds of PROMs: non-erasable bipolar PROMs, Erasable PROMs (EPROMs), and Electrically Erasable PROMs (E^2PROMs). Bipolar PROMs can be electrically programmed only once; thereafter, they retain their data permanently. The programs or data programmed into an EPROM can be erasured by exposing the device to ultraviolet light. An E^2PROMs can be electrically erased in a manner similar to that used to program it.

EPROMs and E^2PROMs allow flexibility during the firmware development cycle because they can be repeatedly erased and reprogrammed. Some microcomputer components, such as the Intel 8751, have built-in EPROM in addition to other logic.

For design convenience, some of these components have pin-compatible counterparts containing masked ROM instead of EPROM. This allows the re-programmable EPROM version to be used during prototype development and the masked ROM version to be used for the final mass production.

PROM Programming Overview

All PROMs have a characteristic physical word length. This word length is the number of parallel bits that are accessed when a given address is specified. Almost all of Intel's recent PROM components have an 8-bit word; however, many earlier PROMs had 4 bit words.

The difference between a PROM's physical word length and the logical word length being used in a system must be considered when programming the PROM. For example, two 8-bit PROM devices must be connected in parallel to provide the 16-bit wide memory word that the 8086 microprocessor requires. To put an 8086 machine language program (consisting of a series of 16-bit words) into 8-bit PROM devices, the machine code must thus be formatted so that the 16-bit words are mapped correctly into each pair of 8-bit devices.

For example, the first PROM might be programmed with the upper 8 bits of each machine code word, while the second PROM is programmed with the lower 8 bits of each machine code word.

The Intel PROM Programming Software (iPPS) provides the software tools necessary to perform this type of interleave mapping of data and machine code into Intel PROMs, automatically.

System Configurations

To program PROMs with the iUP-200/201 in the on-line mode, the iPPS software must be running on the Intellec Development System. In addition, the Universal Programmer must be powered on and must be connected to the Intellec Development System through an RS-232 configured cable. On-line system initialization is described in Chapter 2. If the iUP-201 model is being used, the on-line mode must be selected with the appropriate function switch (see Chapter 4).

Figure 1-4 illustrates the system data flow in the on-line mode. The four major system devices (PROM, Buffer, File, and URAM) are shown with arrows indicating the directions of data flow between these devices.

The Host development system contains the Buffer and File devices while the Universal Programmer contains the PROM and URAM devices. The URAM device is only available in the iUP-201 model of the Universal Programmer. The PROM device is shown as part of a Personality Module.

The PROM device is the PROM component that is plugged into a socket on the Personality Module.

The Buffer device provides a temporary area where the iPPS software stores and manipulates data.

The File device is an ISIS-II file that contains programs or data.

The URAM device is user RAM that is physically located in the iUP-201. This RAM is generally used for off-line data manipulation, but is accessable for certain on-line operations.

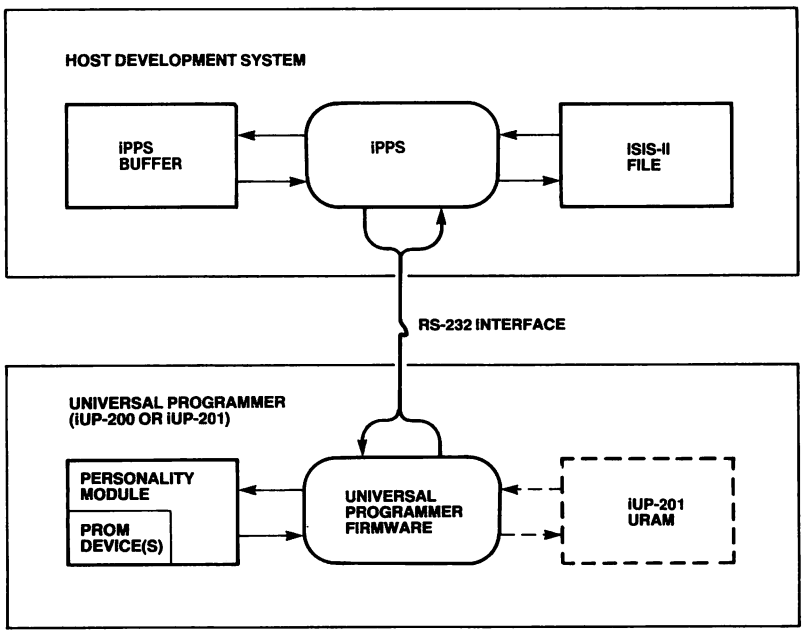


Figure 1-4. On-line System Data Flow

The iPPS software and the iUP-200/201 firmware, which run on the Host Development System and the Universal Programmer, respectively, handle all data transfers between the four logic devices. The two programs communicate by means of a serial interface command protocol. See Appendix B for more information on this protocol.

Figure 1-5 illustrates the system data flow for the Universal Programmer in the off-line configuration. In this configuration, only the PROM and URAM devices are available.

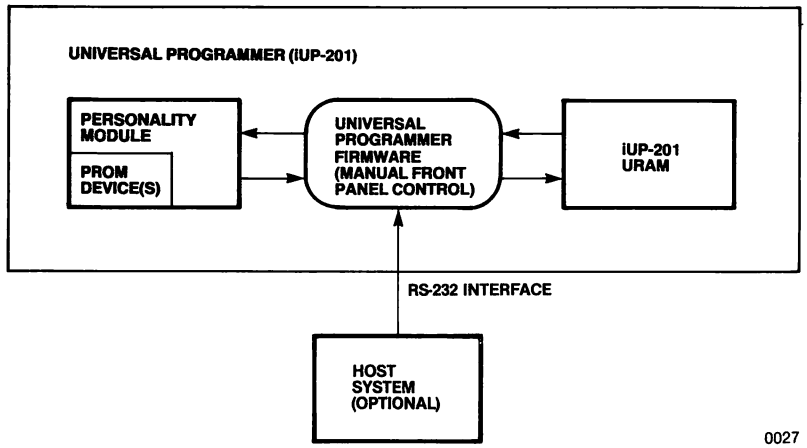


Figure 1-5. Off-line System Data Flow

Through front panel controls (on the iUP-201), the operator directs the Universal Programmer to modify the contents of the local URAM, to program URAM data into the PROM device, or to load the URAM with the data in the PROM device. The user also has the option of down loading data in Intel 8080 hexadecimal format into the URAM from any host system that has an RS-232 interface. In this off-line configuration, the Universal Programmer firmware does not recognize any iPPS commands via the serial interface.

Summary of On-Line Commands

In the On-line configuration shown in Figure 1-4, iPPS commands control the modification and transfer of data between the four iPPS storage devices. The commands that the iPPS software recognizes can be divided into five groups: the program control group, the utility group, the Buffer group, the formatting group, and the copy group. Complete descriptions of each command (in alphabetical order) are included in Chapter 3.

Program Control Group

The program control group of iPPS commands controls iPPS program execution in various ways.

EXIT	Exits the iPPS software and returns control to ISIS-II
<ESC>	Terminates current command
REPEAT	Repeats full execution of previous command
ALTER	Allows edit and re-execution of previous command

Utility Group

The utility group of iPPS commands displays data, help, and status information, and sets default values.

DISPLAY	Displays PROM, Buffer, or File data on Console
PRINT	Prints PROM, Buffer, or File data on Printer
HELP	Selectively displays help information
MAP	Displays Buffer structure and status
BLANKCHECK	Checks for unprogrammed PROM
OVERLAY	Checks if non-blank PROM device can be programmed
TYPE	Selects PROM type
INITIALIZE	Initializes default number base and default file type
WORKFILES	Specifies drive device for temporary work files

Buffer Group

The Buffer group edits, modifies, and verifies data in the Buffer:

SUBSTITUTE	Examines and modifies Buffer data
LOADDATA	Loads section of Buffer with a constant
VERIFY	Verifies data in PROM with Buffer data

Formatting Group

The data formatting group formats and generally permits rearrangement of data from the PROM, Buffer, or File devices.

FORMAT	Interactively formats Buffer, PROM, or File data and places the result in a workfile

Copy Group

The copy group consists of variations of the general purpose COPY command. The following commands allow copying data between the Buffer, PROM, or File devices:

COPY (File to PROM)	Programs PROM with data from a disk file
COPY (PROM to File)	Stores PROM data in a disk file
COPY (Buffer to PROM)	Programs PROM device with Buffer data
COPY (PROM to Buffer)	Loads Buffer with data in PROM
COPY (Buffer to File)	Stores Buffer data in a disk file
COPY (File to Buffer)	Loads Buffer with data in a disk file

The following COPY commands apply only if the the iPPS software accesses the iUP-201 Universal Programmer. See Chapter 4 for a complete description of off-line functions. The following iPPS commands transfer data to and from the iUP-201 local RAM (URAM) during on-line operation:

COPY (File to URAM)	Loads disk file data into local URAM
COPY (URAM to File)	Stores local URAM data in a disk file
COPY (Buffer to URAM)	Loads Buffer data into local URAM
COPY (URAM to Buffer)	Loads local URAM data into Buffer

Summary of Off-Line Commands

In the Off-line configuration shown in Figure 1-5, the iUP-201 function keys are used to manually control the manipulation and transfer of data between the URAM and PROM devices. These keys perform the following functions:

ON-LINE	Selects on-line or off-line operation
ROM TO RAM	Loads PROM data into URAM
VER	Verifies PROM data against URAM data

PROG	Programs PROM with URAM data
DEVICE SELECT	Selects Type of PROM device
CLEAR	Clears entry or terminates current off-line function
ENTER	Enters edited fields
SHIFT ADDR-0	Selects data display function
SHIFT DATA-1	Selects data editiing function
SHIFT FILL-2	Fills a selected address range in URAM with a constant
SHIFT LOAD-3	Loads URAM with data in Intel 8080 hexadecimal file fromat, downloaded from a host system via a RS-232 interface.

Complete descriptions of the iUP-201 function key are included in Chapter 4.

CHAPTER 2
PREPARATION FOR USE

This chapter describes the Universal Programmer's hardware specifications and installation procedures, and describes how to prepare and initialize the system.

Installation Considerations

Compatible On-Line Hosts

The Universal Programmer can be connected as a peripheral to the following Intel development systems (See the latest Intel Systems Data Catalog for information on other compatible Intel hosts):

- Intellec 800
- Intellec Series II
- Intellec Series III

These hosts must have a single or double density disk drive or a hard disk. A separate CRT terminal is also required for the user interface if the Intellec-800 is used. The Intellec-800 also requires a serial converter (see "RS-232 Cable Installation" in this chapter). The Universal Programmer automatically adjusts itself during initialization to the host computer's serial baud rate (110 to 9600 baud).

All of the hosts use the Intel PROM Programming Software (iPPS) which runs as a utility under the ISIS-II Operating System. See the appropriate development system documentation for the details about how to boot up the ISIS-II Operating System.

A minimum of 64K of memory is required for the iPPS software. The ISIS-II Operating System (version V3.4 or later) must be resident in memory when the iPPS software is executing because ISIS-II Input/Output routines are used.

CAUTION

A minimum of 3 inches of clearance is required for the Universal Programmer to allow for proper cooling. Also, for proper cooling, the Universal Programmer's air vents must be clear of any obstructions.

The iPPS software runs only on Intel development systems. Since the Universal Programmer uses the industry standard serial RS-232

interface, it can also communicate with non-Intellec development systems or computers. This communication can be handled in two ways:

- User written RS-232 control drivers.
- iUP-201 download feature.

RS-232 control drivers written to operate the Universal Programmer in its on-line mode must follow a strictly defined command protocol. For further information on this command protocol, see Appendix B.

The iUP-201's serial download feature allows data in Intel 8080 hexadecimal format to be loaded into the iUP-201 URAM from any host system that has an RS-232 interface. In this mode, no special software drivers need to be written. See the discussion of SHIFT-LOAD 3 in Chapter 4 for detailed instructions for using this feature.

Service and Repair Assistance

United States customers can obtain service and repair assistance from Intel Corporation by contacting the Intel Product Service Hotline in Phoenix, Arizona. Customers outside the United States should contact their sales source (Intel Sales Office or Authorized Distributer) for service information and repair assistance.

Before calling the Product Service Hotline, have the following information available:

1. Date the product was received.

2. Complete part number of the product (including dash number). On boards, this number is usually silk-screened onto the board. On other products, it is usually stamped on a label.

3. Serial number of product. On boards, this number is usually stamped on the board. On other products, the serial number is usually stamped on a label.

4. Shipping and billing address.

5. If the Intel Product warranty has expired, a purchase order number for billing purposes must be provided.

6. Make sure to advise the Hotline personnel if an extended warranty agreement is in effect.

Use the following numbers for contacting the Intel Product Service Hotline:

Telephone:
All U.S. locations (except Alaska, Arizona, and Hawaii):
(800) 528-0595
All other locations:
(602) 869-4600

TWX Number:
910 - 951 - 1330

Always contact the Product Service Hotline before returning a product to Intel for repair. The Hotline personnel will provide a repair authorization number, shipping instructions, and other important information which enable fast, efficient service. If the product is being returned because of damage sustained during shipment or if the product is out of warranty, a purchase order is required before Intel can initiate the repair.

EPROM Erasure

EPROMs are erased by exposing the integrated circuit to ultraviolet light through a window provided on the chip package. Erasure occurs when the exposure light has a wavelength shorter than approximately 4000 Angstroms.

Sunlight and certain types of flourescent lamps have wavelengths in the 3000-4000 Angstrom range. Constant exposure to room level flourescent light could erase the typical EPROM in approximately 3 years, while it would take approximately 1 week to erase when exposed to direct sunlight.

If the EPROM device is exposed to these lighting conditions for extended periods of time, the device window should be covered with opaque labels (available from Intel) to prevent unintentional erasure.

The optimum light for erasing EPROMs has a wavelength of 2537 Angstroms. The integrated dose (UV intensity X exposure time) for erasure should be a minimum of 15 W-sec/cm^2. The erasure time is approximately 15 to 20 minutes using an ultraviolet lamp with 12,000 uW/cm^2 power rating. The EPROM should be within 1 inch of the lamp tubes during erasure.

The EPROM should not be powered up during erasure. If the EPROM device is powered up during erasure, the internal current paths will effectively cancel the energy that the UV light provides, and the device will not be erased.

Consult the "PROM and ROM Programming Instructions" section in the Intel Component Data Catalog for further information on erasing EPROM components. Individual device specifications also contain erasure information.

Installation Procedures

This section contains detailed descriptions of installation procedures for the Universal Programmer.

CAUTION

To prevent possible damage to the Universal Programmer, make sure that the line voltage select switches are set properly and the power fuse is the right value for the line voltage that is to be used, before plugging the unit in and turning it on. See the sections of this chapter titled "Setting Line Voltage" and "Checking and Replacing Power Fuse".

Setting Line Voltage

WARNING

Changing the line voltage setting should be performed by a qualified technical person.

The Universal Programmer accepts one of four different AC line voltages. Prior to plugging the unit in and turning it on, it should be set to the appropriate available line voltage as follows:

1. Assure that the Universal Programmer is unplugged from its power source.

2. The four line voltage settings are summarized in a table printed on the back panel of the Universal Programmer. (Figure 2-1 shows the switches set for 120 VAC.)

   Set SW1 and SW2 as follows, according to the line voltage to be used to power the unit (Be sure that the letters are visible on the switches):

   - o 100 VAC: set both SW1 and SW2 to position A.
   - o 120 VAC: set SW1 to position A and SW2 to position B.
   - o 220 VAC: set SW1 to position B and SW2 to position A.
   - o 240 VAC: set both SW1 and SW2 to position B.

3. The Universal Programmer is now ready to be plugged into a power source of the selected voltage and turned on, provided that the correct power fuse has been installed.

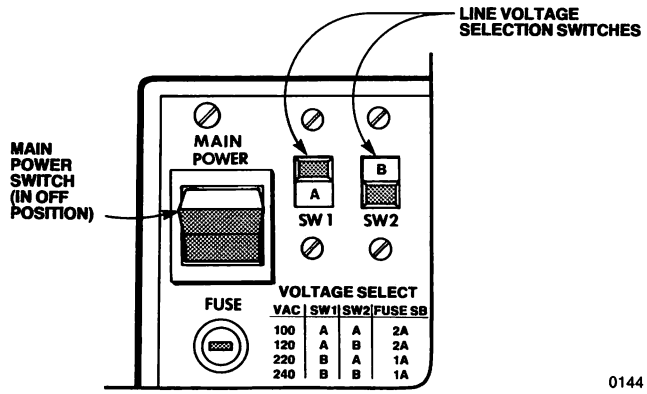


Figure 2-1. Setting Line Voltage

Checking and Replacing Power Fuse

WARNING

Checking or replacing the power fuse should be performed by a qualified technical person.

The fuse is located directly beneath the Main Power switch on the back panel. (Figure 2-2 shows the fuse being removed.)

To replace the fuse:

1. Unplug the main power plug from its power source.
2. Using a flat blade screwdriver or a small coin, turn the fuse holder counterclockwise as shown in Figure 2-2.
3. Once loosened, slide the fuseholder and fuse out as shown.
4. Check the fuse for continuity and that it is the proper size and amperage rating.
5. If the fuse is damaged or the wrong size or rating, replace it with one that is the correct amperage rating and size for the line

voltage at which the unit is to be operated. These amperage ratings are specified as follows:

Fuse Protection:	100 VAC:	use 2 ampere slow blow
	120 VAC:	use 2 ampere slow blow
	220 VAC:	use 1 ampere slow blow
	240 VAC:	use 1 ampere slow blow

5. Slide the fuse and fuseholder back into the fuse receptical and tighten by turning the fuseholder clockwise.

6. The main power plug can now be plugged back into the power source. The Universal Programmer can be turned on, provided that the line voltage selector is set properly.

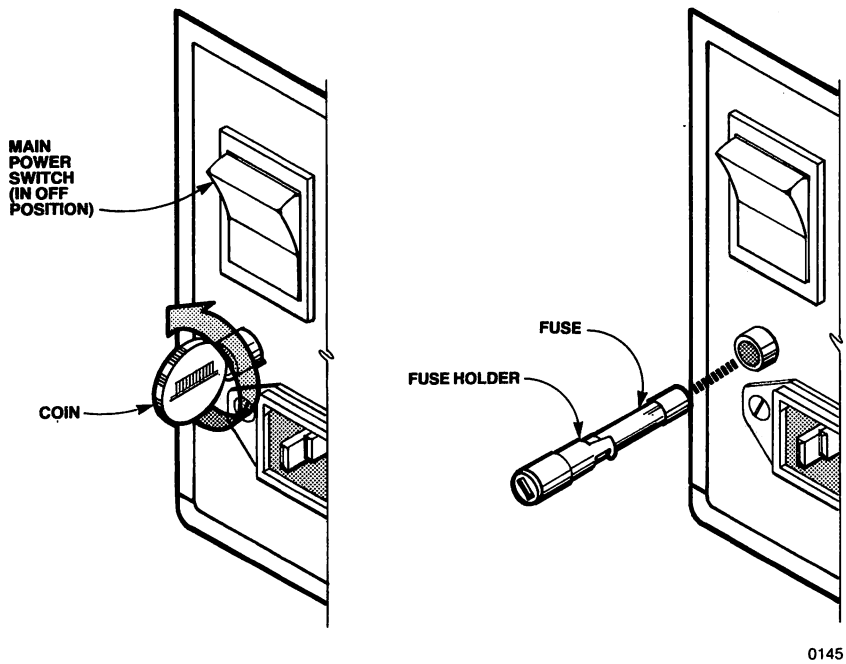


Figure 2-2. Replacing Power Fuse

RS-232 Cable Installation

The RS-232 cable physically connects the host development system and the Universal Programmer. To connect this cable:

1. Plug either end of the RS-232 cable into the RS-232 connector on the back panel of the Universal Programmer (see Figure 1-3). The Main Power can be either on or off.

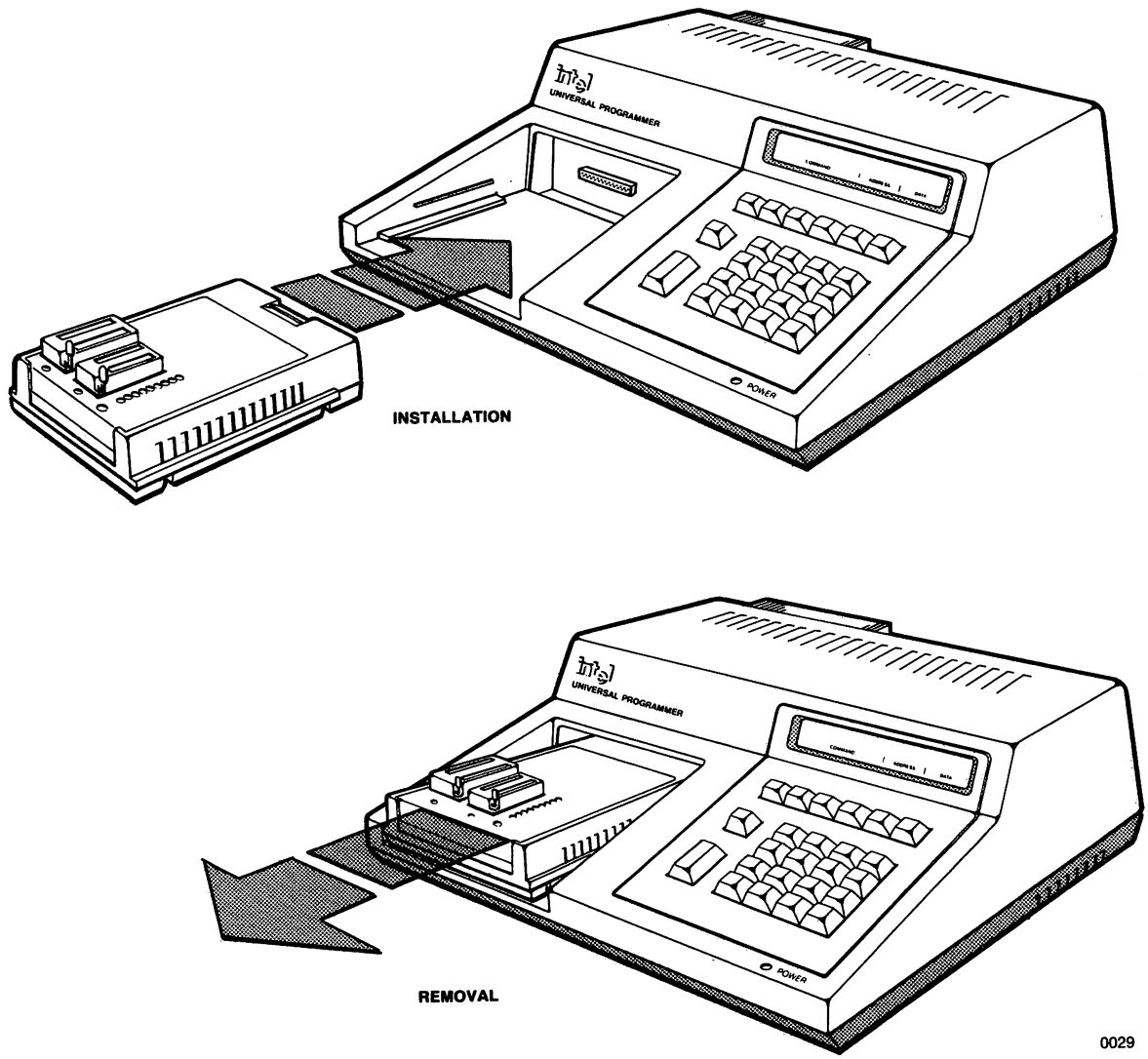


Figure 2-3. Personality Module Installation

2. Plug the other end of the RS-232 cable into the appropriate serial connector on the back panel of the host development system. These serial connectors are listed below for Intellec systems.

   - Intellec-800. To use the Intellec-800 as a host system, a serial converter is required. The RS-232 cable from the Universal Programmer connects to the converter and another cable may be required between the converter and the TTY connecter on the back panel of the Intellec-800. The converter is used for bi-directional conversion between the Intellec-800's 20 ma teletype connection and the Universal Programmer's RS-232C connection. Such converters are available from many manufacturers.

   - Intellec Series II and Series III. To use either of these development systems as a host, use the connector labeled "SERIAL CH1/TTY" (J2) on the development system back panel.

Personality Module Installation

The Personality Module plugs into the front-panel connector of the Universal Programmer (see Figure 2-3). No further connection is required. During insertion and removal of the Personality Module, the main power switch can be either on or off.

System Initialization

This section describes system behavior during power on and initialization. Before powering on the Universal Programmer it is assumed that the required equipment is in place and properly interconnected. See previous sections of this chapter for details on compatible hosts and installation procedures.

Powering ON

This section describes the behavior of the Universal Programmer immediately after it is powered ON. This section assumes a properly connected system. See earlier sections of this chapter for installation procedures.

CAUTION

To prevent damage to the Universal Programmer, make sure the line voltage select switches are set properly and the power fuse is the right value for the line voltage that is to be used, before plugging the unit in and turning it on. See the sections of this chapter titled "Setting Line Voltage" and "Checking and Replacing Power Fuse".

> **Also, to prevent damage to a PROM device or accidental programming of it at one or more random memory locations, remove any PROM from the Personality Module before powering on the Universal Programmer.**
>
> **A device should not be in the PROM socket(s) when the power is turned on or off.**

Once assured that the line voltage selector and fuse are properly set, plug the Universal Programmer into a power source and set the MAIN POWER switch (rear panel) to its on position (see Figure 2-1).

Self-test diagnostics are run when the Universal Programmer is initially powered ON. If any of these tests fail, error messages are displayed.

If the Universal Programmer is a model iUP-200, these messages are sent to the iPPS software for console display. If the iPPS software is not running when the iUP-200 is turned on, however, none of these messages are displayed on the console. Thus, it is recommended that the iPPS software be running before the Universal Programmer is powered on.

If the Universal Programmer is the model iUP-201, these messages are displayed only on the unit's alphanumeric display. This is because the iUP-201 initializes in the off-line mode of operation. While diagnostics are being performed, the following message is displayed:

DIAGNOSTICS IN PROGRESS

The following self-test diagnostics are performed:

1. A power supply test is performed to verify that the Universal Programmer's internal voltages are within tolerance. If this test fails then the following error message is displayed:

   POWER SUPPLY FAILURE

2. The following motherboard tests are performed (for both the iUP-200 and iUP-201):

   - CPU instruction test of the 8085 processor
   - Checksum test on the internal firmware ROM
   - Read/write memory test on the 8085's RAM
   - Internal timer test

   If any of these tests fail, the following error message is displayed:

   MOTHERBOARD FAILURE

3. An interface test is performed (iUP-201 only) with the iUP-201's KEY/RAM board. If this test fails, the following error message is displayed:

   KEY/RAM BOARD FAILURE

If all the self-diagnostic tests are passed, the iUP-201 displays this message in the Command, Address, and Data fields of the alphanumeric display:

```
IUP READY      0000      55
```

The iUP-201 can then be operated in the off-line mode or switched to on-line mode for control by the iPPS software on the Intellec Development System.

On-Line Initialization

To operate in the on-line mode the Universal Programmer (iUP-200 or iUP-201) must be connected to an Intellec Development System that is running under ISIS-II (versions V3.4 and later). An iUP-201 must be switched to the ON LINE mode to be operated by the Universal Programming Software (iPPS).

The following files are necessary for normal iPPS software operation:

- IPPS — The main command file that is loaded and executed under the ISIS-II operating system.
- IPPS.ERR — The error message file containing strings for the various iPPS error messages.
- IPPS.HLP — The help message file containing text explanations of the various iPPS commands.
- IPPS.OV0 — An overlay file containing machine code that is loaded during normal iPPS software operation.
- IPPS.OV1 — An overlay containing machine code that is loaded only during operation of the FORMAT command.

Use the following systax to invoke the iPPS software as a command under ISIS-II:

```
:F<n>:IPPS<CR>
```

:F<n>: specifies the drive on which the iPPS files are located. For example, if the iPPS files are located on disk device 1 then the iPPS software would be invoked by entering

```
:F1:IPPS
```

followed by a carriage return. After successful invocation, the iPPS software will seek to use its other files (IPPS.ERR, IPPS.HLP, IPPS.OV0, and IPPS.OV1) on drive 1.

When the iPPS command is entered, ISIS-II loads and executes the iPPS software. After the software is properly initialized, it displays the following sign-on message:

```
INTEL UNIVERSAL PROGRAMMING SOFTWARE Vv.r
PPS>
```

Vv.r is the current version (v) and revision (r) number of the software. PPS> is the main iPPS prompt; it indicates that the iPPS software is ready to receive commands. Use the EXIT command to return to ISIS-II. See Chapter 3 for full details of on-line operation.

Off-Line Initialization

After checking the Line Voltage Selector switches and power fuse, plug the model iUP-201 into a power source and power it on. When first powered on the iUP-201 always initializes in the off-line mode.

Off-line operations do not require that the Universal Programmer be connected to a host development system. However, off-line functions are often performed in conjunction with on-line operations. For example, data may be copied to the URAM during on-line operation and then copied to a PROM off-line. Data can also be copied from a host system using the off-line SHIFT-LOAD 3 function (see the discussion of SHIFT-LOAD 3 in Chapter 4).

PROM Device Installation

Insert the PROM to be programmed or read as follows:

1. Assure that the appropriate Personality Module is installed for the device that is to be programmed (See "Personality Module Installation" in this chapter).

CAUTION

> Do not switch the Universal Programmer's power on or off when a PROM device is installed in a socket of the Personality Module. Damage to the PROM device could result.

2. If the Universal Programmer is not already powered on, switch on the power switch and wait for the initialization procedure to be completed. (See the section in this chapter titled "System Initialization".)

   Do not power on the system until you have read the Powering On section of this Chapter.

3. Set the Universal Programmer for the PROM type to be programmed or read, using either the TYPE command (during on-line operation) or the DEVICE SELECT key (during off-line operation). LEDs next to the PROM designations on the Personality Module indicate the PROM type selected. If the Personality Module has more than one socket, a socket indicator light indicates the appropriate socket for the selected PROM type.

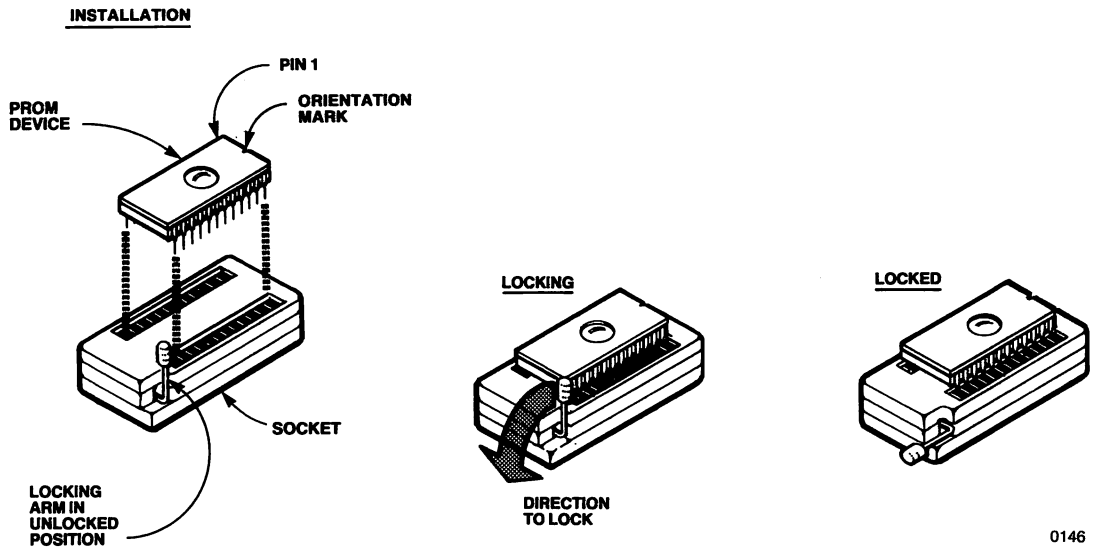


Figure 2-4. PROM Device Installation

4. Raise the socket locking arm on the selected socket (see Figure 2-4).

CAUTION

If the Personality Module has more than one socket, install the PROM to be programmed or read only in the selected socket (as determined by the indicator light). Do not install a PROM or leave one installed in an unselected socket. Damage to the PROM device can result.

5. Insert the device to be programmed into the socket with pin 1 of the device going into the socket pin hole at the upper left corner of the socket.

CAUTION

The orientation mark on one end of the device must be toward the top of the socket. If a device is not oriented properly, it cannot be programmed and it may be damaged. If the device is not oriented properly, the following error message may be displayed when the Universal Programmer attempts to access the device:

CHECK PROM INSTALLATION

6. To secure the device in the socket, move the locking arm forward and down until it is parallel with the top of the Personality Module.

The PROM device is now ready for reading, programming, or verifying.

CHAPTER 3
ON-LINE OPERATION

This chapter covers the on-line operation of the Universal Programmer and describes in detail the operation of the Intel PROM Programming Software (iPPS). This software runs as a command under the ISIS-II operating system. Most iPPS commands assume that the iUP-200 or iUP-201 hardware is powered up and properly connected to the host Intellec development system, although certain file and buffer manipulation commands can be used without a connection to the iUP-200/201. Chapter 2 provides information about configuring a development system for on-line PROM programming.

iPPS Software Initialization

Command Line Invocation

The Intel PROM Programming Software (iPPS) is invoked as a command under ISIS-II (versions V3.4 and later) with the following syntax:

```
[:F<n>:]IPPS<CR>
```

The conventions used to describe commands are covered in the section "Notation for Syntax Description" in this chapter. After the iPPS command is keyed in, ISIS-II loads and executes the iPPS software. If the software is properly initialized, the following sign-on message is displayed:

```
INTEL PROM PROGRAMMING SOFTWARE Vv.r
PPS>
```

Vv.r is the current version (v) and revision (r) of the software. PPS> is the main iPPS prompt; it indicates that the iPPS software is ready to receive commands. The EXIT command returns control to ISIS-II.

Invocation Via a SUBMIT File

The iPPS software can run under the control of a SUBMIT file. SUBMIT is an ISIS-II command that allows a disk text file to be used as input for further ISIS-II commands or as command inputs to utilities running under ISIS-II. Thus, a SUBMIT file can contain the ISIS-II command line to invoke the iPPS software and then a sequence of commands for the iPPS software itself. The SUBSTITUTE and ALTER commands can not be used as part of a SUBMIT file. The iPPS software recognizes comment lines in a SUBMIT control file as any characters in the line following a semicolon (;). For more information on the SUBMIT command, consult the ISIS-II User's Guide. In order to invoke and control the iPPS software via a SUBMIT file, the SUBMIT command itself must be available on the system disk.

iPPS Software General Operation

Major Functions

The purpose of the iPPS software is to control the Universal Programmer from the Intellec Development System console. The iPPS software recognizes a small set of commands for doing this. The commands perform functions on four major storage devices: PROM, Buffer, File, or URAM. These devices are described in detail in the following section.

The major functions of the iPPS software are:

- To read and write data on any of the four major storage devices (PROM, Buffer, File, or URAM)
- To manipulate data in the Buffer
- To display or print the data stored in any of the major storage devices on the console or line printer
- To check for an unprogrammed PROM and to allow the data in the Buffer to be overlayed with the bits in the PROM device. This operation determines whether a non-blank PROM can be programmed.
- To format data for different programmable devices (ie., select different PROM word sizes and use interleaving techniques)
- To accomodate the following Intel absolute file formats during any operations that require files: 8080 Hex ASCII, 8080 Absolute Object, 8086 Hex ASCII, 8086 Absolute Object, and 286 Absolute Object. See Appendix C for further information on these file formats.

iPPS Storage Devices

The iPPS software transfers data between any two of the four storage devices: PROM, Buffer, File, or URAM. The data flow relationship among these devices is illustrated in figure 1-4. These devices are defined in the following sections.

PROM Device

The PROM device is plugged into a socket on the Personality Module of the Universal Programmer.

During iPPS operations, the size of the PROM device varies based on the most recent TYPE command. The default address boundaries for the PROM device are 0 to the PROM size - 1. The iPPS software does not recognize the PROM device until a TYPE command is issued. The TYPE command automatically sets the appropriate buffer size according to the type of PROM device specified. See the TYPE command description for further details.

Buffer Device

The Buffer device is a section of Intellec Development System memory that the iPPS software allocates and uses as a working area for temporary storage and for rearranging of data. Its boundaries can exist anywhere in a virtual address range from 0 to 16777215 (0 to $2^{24}-1$).

When the iPPS software is initialized, the Buffer start address is set to 0 and the Buffer end address is set to 8K-1. This implies an initial Buffer size of 8K bytes (the default Buffer size when no PROM type is specified). During subsequent iPPS operations, the size and boundaries can vary. Specific iPPS commands determine these variations. The most recent command that changed the lower boundary of the Buffer determines the Buffer start address. The following expressing determines the Buffer end address:

(Buffer Start address) + (Buffer size - 1)

The TYPE command affects both the size and location of the Buffer in important ways. For example, the TYPE command always resets the Buffer start address to 0. The most recent TYPE command determines the size of the Buffer. See the TYPE command description for further details.

The size of the Buffer is determined as follows:

- It is equal to the size of the PROM if the PROM type has been specified via the TYPE command.
- It is initialized to 8K when the iPPS software is invoked.

The need for a virtual Buffer arises when the PROM size exceeds 8K. If the PROM size exceeds the 8K memory Buffer space available on the development system, the iPPS software creates a virtual Buffer area using temporary file space on disk. The iPPS software allows the user to specify the storage device on which the virtual Buffer is created; otherwise the virtual Buffer is transparent to the user.

Two temporary work files are used to create this virtual Buffer. Their file names are IPPS.BUF and IPPS.TMP. These temporary files are created on the disk device that the most recent WORKFILES command specified. If no previous WORKFILES command was issued and a command is attempted that requires virtual buffering, the iPPS software then displays the prompt,

WORKFILES (:FX:)? ; X =

This prompt requests a number indicating which disk device to use for the temporary work files. During subsequent virtual Buffer operations, the iPPS software automatically swaps data in and out of development system memory to work files on the specified device. The default work files device then remains the same until the next WORKFILES command. See the WORKFILES command for more details. The temporary files are deleted either when the EXIT command is executed or when a new PROM type is selected that is less than 8K in size.

File Device

The files device is a normal ISIS-II file on disk. It is specified within iPPS commands, using the normal ISIS-II file specification format:

:<device>:<filename>.<extension>

The data is stored in the disk file is in one of the following Intel absolute formats: 8080 Hex, 8080 Object, 8086 Hex, 8086 Object, and 286 Object. The iPPS software can read any of these formats as input, however, all iPPS commands that output data to a File destination produce files of the 286 format. See Appendix C for more information on the file formats. Basically, these files contain representations of blocks of memory data. Included with the data are addresses for the locations of the data. The data blocks are not necessarily in consecutive address order. The method used to create the file determines the order of the data.

The iPPS file device has address boundaries that exist in the virtual range from 0 to 16777215 (0 to $2^{24}-1$). These boundaries are determined as follows:

- The file's lowest address is the lowest address encountered while reading the file.
- The file's highest address is the highest address encountered while reading the file.

If the iPPS software creates the file (i.e., it was a destination device in an iPPS command), the specific command issued determines these boundaries.

When a particular address range is specified to be read from a file, all sections in the address range that are not present in the file are written in a PROM or URAM destination device with the blank state of the currently selected PROM type (all ones if no PROM type has been selected). If the destination device is the Buffer, those non-existent sections in the file are not affected in the Buffer.

During the operation of commands that use the File device as a source, the iPPS software only reads the actual data from the file and ignores any other information in the file. For example, the file can contain special information used later for debugging. Since the iPPS software ignors this information, it will not appear in any new files generated. If the data is written back to the original file, the original file is in effect deleted.

URAM Device

The URAM is available only with iUP-201 models of the Universal Programmer. It consists of a RAM buffer that resides locally in the iUP-201, but which iPPS can write to or read from. This RAM is generally used for off-line data manipulation, but is accessable for certain on-line operations.

During on-line operations, iPPS accesses the URAM as a device. (If iPPS commands specify this device when the iUP-200 model is connected, the iPPS issues appropriate error messages.) iPPS operations can access all of the URAM or portions of it.

The default URAM address boundaries are 0 to URAM size -1. The URAM size can be 16K or 32K bytes depending on the amount of local user memory available in the iUP-201 model. The iUP-201 hardware is designed to allow URAM expansion from 16K to 32K.

Command Entry

iPPS commands consist of a command keyword to identify the command followed by other keywords and their associated parameters to make up the arguments of the command. Arguments are separated from each other by at least one space (extra spaces are ignored).

A command line must be terminated by a carriage return, <CR>. No action is taken on the command until the carriage return is recognized unless ESC is pressed, in which case the command is aborted. Once entered, a command line is then verified for the correct syntax and executed if correct. If a syntax error is detected, the following error message is displayed:

--SYNTAX ERROR--<specific error>.

If a required keyword is omitted, iPPS prompts for the keyword and its associated parameters. If the keyword is entered but its parameters are omitted, either a default value is assumed if applicable, or an error message is displayed if no default is applicable. In certain commands, default keywords are also assumed. All commands can be entered in either lower or upper case ASCII.

Input lines do not directly correspond to an 80 character screen line. If the echoed input line exceeds the screen display line, it simply wraps around to the next screen line. Command inputs which are too long to fit in one input line can be continued in successive lines. To continue a line, an ampersand (&) is entered as the last non-blank character preceeding the carriage return. Command inputs are accumulated character-by-character in a line input buffer that can hold a maximum of 255 characters (including ampersands, spaces, and carriage returns). The maximum non-continued line size is 127 characters (including the final carriage return).

iPPS keywords can be entered in their entirety or as any unique abbreviation (only the first character is required). For example, command keywords of C, CO, COP, and COPY are all interpreted as the COPY command.

iPPS accepts numeric entries in any one of four number bases: Binary (Y), Octal (O or Q), Decimal (T), or Hexadecimal (H). Numbers can be entered in any of these bases by appending the appropriate single letter identifier to specify the base; for example, 11111111Y, 377Q, 255T, or FFH. An explicit number base identifier overrides the default number base which is initially hexadecimal.

To illustrate these command entry features, the examples below show different ways in which the same COPY command can be entered to read a 2K PROM into the Buffer. Boxes around items indicate user entries.

```
COPY PROM(0,7FF) TO BUFFER(0)<CR>

C P(0,11111111111Y) TO B<CR>

C<CR>
FROM? PR(0,7FF)<CR>
TO? BU<CR>

copy pr(0,7FF)<CR>
TO? b<CR>

C P T B<CR>
```

Command Entry Editing

A new command can be entered when iPPS displays the prompt, PPS>. iPPS commands can be edited using ISIS-II command line editing features as described below. Control characters are entered by holding down the control key (CNTL) while the character is typed.

<RUBOUT> Deletes the previous character that was entered.

<CNTL-X> Deletes the entire contents of the line-editing buffer, including the <CNTL-X> itself. It is echoed as a '#' character followed by a carriage return/line feed. The line-editing buffer is cleared and command input is restarted on the next screen line.

<CR> (Carriage Return) terminates the command input and submits the entire contents of the line-editing buffer to iPPS as a command.

In addition, iPPS provides the ALTER command which permits the editing and re-execution of the most recently entered iPPS command.

The Form of iPPS Commands

Notation for Syntax Description

Square brackets, braces, angle brackets and underscores are used in this manual solely for the purpose of defining the syntax of iPPS commands; they are not recognized as valid entries by iPPS. For instance, brackets are used in this manual to indicate items that are optional parts of a valid command input. The brackets themselves are not keyed in. All Punctuation other than square brackets, braces and angle brackets must be entered as shown. The syntax descriptions use the following notational conventions:

UPPERCASE Denotes specific words that can be entered.

[ ] (Square brackets) indicate that all the parameters and punctuation enclosed are optional.

{ } (Braces) enclose a list of parameters where one and only one of the parameters must be selected. The different choices are listed vertically on separate lines.

< > (Angle brackets) denote an entry for which the user must supply a specific value to replace the generic term enclosed within the brackets. Generic terms that are used throughout the syntax descriptions for iPPS commands are described in the following section, "Special iPPS Syntax Terms."

UNDERSCORE Indicates a valid single character abbreviation of the keyword being entered.

Special iPPS Syntax Terms

In addition to the notation conventions, the syntax descriptions in this manual contain generic terms that are common to most iPPS commands:

<startaddr> Starting address of a section of source data. It can have values in the range 0 to 16777215 (i.e., 0 to 2^{24} - 1).

<endaddr> Ending address of a section of source data. It can have values in the range 0 to 16777215 (i.e., 0 to 2^{24} - 1). This value can be optionally entered as a length value in the form L<length>. See the description of <length> below.

<destaddr> Destination start address for a section of data that is to be written. It can have values in the range 0 to 16777215 (i.e., 0 to 2^{24} - 1). This value can be optionally entered as an offset value in the form O<offset>. See the description of <offset> below.

<length> The length (in bytes) of a section of data. It can have values in the range 1 to 16777216 (i.e., 1 to 2^{24}).

<offset> An offset value used to specify the destination start address for a section of data to be written. The offset can be positive or negative and is relative to the <startaddr> specified for the source device. The address is calculated by adding the <offset> to the <startaddr>; the calculated address must be in the range 0 to 16777215 (i.e., 0 to 2^{24} - 1).

<file> The standard ISIS-II file specification of the form :<device>:<filename>.<extension>. See "Device/File Name Format" in the ISIS-II User's Guide for more information. iPPS accepts any <filename>. However, if neither the device nor extension is specified, the following <filenames> are invalid:

B	P	U
BU	PR	UR
BUF	PRO	URA
BUFF	PROM	URAM
BUFFE		
BUFFER		

Though it is not recommended, a file with any of these names can be created by always specifying it with its device (eg., :F0:BUFFER). iPPS recognizes :F0:BUFFER as a filename that is distinct from BUFFER.

In addition, iPPS uses IPPS.BUF and IPPS.TMP as temporary files. It is also recommended that these file names not be used.

General Command Syntax

Most iPPS commands conform to the following general syntax:

<cmd> <source> [(<startaddr>[,<endaddr>])]
<prep> <dest> [(<destaddr>)] [<switches>]

The generic terms used above have the following meaning:

<cmd> Specifies the iPPS command keyword for the command to be executed. iPPS keywords can be entered in their entirety or as any unique abbreviation (only the first character is required).

<source> Specifies the source device as any one of the four devices that iPPS recognizes: PROM, Buffer, File, or URAM.

<prep> Represents an appropriate preposition (WITH or TO) which serves to syntactically separate the source and destination portions of the command input. Only the first character of these prepositions need be used. At least one leading and one trailing space character must be entered to delimit these prepositions.

<dest> Specifies the destination as any one of the four recognized devices; however, the <dest> cannot be the same as the <source>. If this is attempted an error message occurs.

<switch> Specifies any of the four valid command switches discussed in the next section of this chapter. Not all command switches are valid for all commands. The valid command switches for a given command are indicated in the description for that command.

The terms <startaddr>, <endaddr>, and <destaddr> are described generally in the section of this chapter titled "Special iPPS Syntax Terms". Their defaults are described in the section titled "Command Defaults", which contains a convenient Table of iPPS Command Defaults. More detailed information about these parameters can also be found in the specific command descriptions.

The INITIALIZE, SUBSTITUTE, TYPE, and WORKFILES commands do not fully obey the above syntax form. See their individual command descriptions in this chapter for further details.

Command Switches

Command switches can be specified in a command entry. These software switches are active only for the duration of the command in which they are specified. The command switches have defaults associated with them that remain constant across all iPPS commands. The INITIALIZE command allows the Base switch and File switch defaults to be changed. The defaults for the other switches cannot be changed.

Command switches are specified as the last part of a command entry and are preceeded by at least one space. Not all command switches are applicable to all iPPS commands; but, when applicable, command switches are always optional. If more than one command switch is specified, spaces must be used to separate them. The order in which command switches are entered is not significant. The switches with their respective command line entries are:

Base switch Determines the number base to be used for the duration of the command. The possible single character entries are:

where:
Y - Binary
O - Octal
Q - Octal
T - Decimal
H - Hexadecimal

Only one of these can be specified at a time. The default for this switch is the current default number base that the most recent INITIALIZE command set. If no INITIALIZE command has been issued, Hexadecimal is the default.

File switch

This switch specifies the format of the file in which the data is stored. The possible entries are:

$$\left\{\begin{matrix}80\\86\\286\end{matrix}\right\}$$

It can be any of the following Intel formats:

80 - 8080 Absolute Object or 8080 Hexadecimal ASCII

86 - 8086 Absolute Object or 8086 Hexadecimal ASCII

286 - Special iUP-200 output file format.

These file formats are described in more detail in Appendix C. The default for this switch is the currently selected default file format. See the INITIALIZE command description for more details. If no INITIALIZE command has been issued, 286 is the default.

Data switch

Specifies that data is to be represented in memory in its false (complemented) form. The only allowable entry is:

F

F specifies false; when F is not specified, iPPS defaults to true (uncomplemented).

Patch switch

Specifies that patched object files can be read. The only allowable entry is:

P

Certain object program files can contain sections of object code with overlapping addresses. iPPS finds any overlapped addresses within the address range it was directed to read, it generates the following error message:

-GENERAL ERROR--DATA OVERLAP IN THE FILE.

The Patch switch allows iPPS to read files that have overlapping addresses. In this case, the sections of data with overlapping addresses are written over one another. In general this switch should be used for all object program files that have been patched.

If the Patch switch is specified, slower response may be experienced during file read commands. This is due to iPPS's scanning of the entire file to find any possible patch blocks.

When P is not specified, iPPS defaults to the mode where the reading of an object file with overlapping addresses results in an error.

Command Defaults

iPPS assumes default values for various command parameter entries. The command switches have defaults in addition to other command parameter inputs. The input defaults are:

Base switch — The default for the Base switch is initially hexadecimal (H), but it can be changed by the INITIALIZE command. The current default base can be overridden for a numeric entry by appending a single letter identifier to specify the base of that number only. Thus, 11111111Y, 377Q, 255T, or FFH all represent the same numeric entry. In the absence of an appended letter, a numeric entry is interpreted according to the default number base.

File switch — The default for the File switch is the currently selected default file format. See the INITIALIZE command description in this chapter for more details.

Data switch — The default for the Data switch is always true. Entering F for the Data switch changes the switch to false for the duration of the command.

Patch switch — The default for the Patch switch is always Off. Entering P for the Patch switch change the switch to On for the duration of the specific file read command.

<file> — The only default associated with <file> is a default storage drive. If no drive device is specified, :F0: is assumed.

Command defaults associated with the source and destination devices are shown in Table 3-1. Some commands require only a source device (e.g., the DISPLAY command). Table 3-1 also shows defaults for commands that have a single device specification.

Table 3-1. iPPS Command Defaults

DEVICES		DEFAULTS		
Source	Destination	<startaddr>	<endaddr>	<destaddr>
BUFFER	<file>	Buffer start address(3)	Buffer end address(4)	0
<file>	BUFFER	File's lowest address(1)	File's highest address(2) limited by buffer size(5)	0
BUFFER	PROM	Buffer start address(3)	Buffer end address(4) Limited by PROM size (5)	0
PROM	BUFFER	0	PROM size - 1	0
<file>	PROM	File's lowest address(1)	File's highest address(2)	0
PROM	<file>	0	PROM size - 1	0
<file>	none	File's lowest address(1)	File's highest address(2)	none
Buffer	none	Buffer start address(3)	Buffer end address(4)	none
PROM	none	0	PROM size - 1	none
BUFFER	URAM	Buffer start address(3)	Buffer end address(4) limited by URAM size(8)	0
URAM	BUFFER	0	URAM size - 1 limited by buffer size(6)	0
<file>	URAM	File's lowest address(1)	File's highest address(2) limited by URAM size(7)	0
URAM	<file>	0	URAM size - 1	0

NOTES:

(1) File's lowest address is the lowest address encountered while reading the file.

(2) File's highest address is the highest address encountered while reading the file.

(3) The most recent command that changed the lower boundary of the Buffer determines the Buffer start address. The TYPE command always resets the Buffer start address to 0.

(4) The following expression determines the Buffer end address: (Buffer start address) + (Buffer size - 1).

(5) The most recent TYPE command fixes the Buffer size to the selected PROM size. iPPS initializes with a default Buffer size of 8K. If a <file> to Buffer operation is limited by the Buffer size, the default <endaddr> is : <startaddr> + (Buffer size - 1).

(6) See note (5) above for meaning of Buffer size. If a URAM to Buffer operation is limited by the Buffer size, the default <endaddr> is: (Buffer size - 1).

(7) URAM size is normally 16K (expandable to 32K). If a <file> to URAM operation is limited by the URAM size, the default <endaddr> is: <startaddr> + (URAM size - 1).

(8) See note (7) above for meaning of URAM size. If a Buffer to URAM operation is limited by the URAM size, the default <endaddr> is: <startaddr> + (URAM size - 1).

iPPS Command Descriptions

The remainder of this chapter is dedicated to descriptions of each iPPS command. For convenience, the commands are in alphabetical order for convenience. Each command description includes a syntax examples section to illustrate the syntax with actual commands.

Unless overridden or otherwise noted, the default number base is hexadecimal and the default file type is 286 in the syntax examples of this chapter.

Also, the syntax examples use a graphic shading convention to differentiate operator keyins from iPPS console output.

ALTER

Edit and Re-execute
Previous Command

SYNTAX

ALTER

OPERATION

The ALTER command is used to edit the previously entered command and then optionally execute the edited command. When the ALTER command is entered, the most recently entered iPPS command line is displayed on the console with the cursor blinking under the left-most character of the text line. A new line may then simply be typed over the old or the following special keys may be used to edit the line (a carriage return need not be entered after these single character ALTER commands):

> moves the cursor to the right one character position.

< moves the cursor to the left one character position.

<RUBOUT> moves the cursor to the left one character position.

<CTRL-I> (insert) opens one empty character position in the line immediately before the character the cursor was under when the <CTRL-I> was entered. The cursor is placed under the new empty position and any remaining characters in the line are shifted right one character position. A new character can then be inserted and will occupy the opened character position. Any subsequent characters typed will overwrite the existing characters in the line. Thus, <CTRL-I> must be entered before each consecutive character that is inserted if the remainder of the line is to be retained.

<CTRL-D> (delete) causes the deletion of the character currently under the cursor. Any remaining characters in the line are shifted left by one character position to fill the deleted character position.

<CR>
or
<ESC> indicates to the ALTER command to exit the command line edit mode. After a <CR> or <ESC> is typed (regardless of cursor position), the edited command line is redisplayed followed by a Y/N prompt. Typing N causes return to iPPS without attempting the newly edited command line. Typing Y causes the newly edited command line to be executed by iPPS.

EXAMPLES

The following COPY command was entered with a mistyped keyword:

```
COPPY PROM TO BUFFER
--SYNTAX ERROR--ILLEGAL KEYWORD.
PPS>
```

The following illustrates the use of the ALTER command to repair the previously mistyped command and reissue it to iPPS:

```
ALTER
COPPY PROM TO BUFFER
```

The cursor is moved right two characters to the first P character using the > command:

```
COPPY PROM TO BUFFER
```

<CTRL-D> is then typed to delete the extra P character and then <CR> is entered to indicate editing is complete:

```
COPY PROM TO BUFFER  --- Y/N? Y
```

The corrected COPY command is then executed by iPPS immediately after the Y is entered.

BLANKCHECK

Check for
Unprogrammed PROM

SYNTAX

BLANKCHECK [PROM (<startaddr>[,<endaddr>])]

OPERATION

The BLANKCHECK command tests the PROM device in the iUP-200 Personality Module socket to determine if it is blank (i.e., in the unprogrammed state for that particular device--either all logic ones or all logic zeroes). The BLANKCHECK command can not be executed unless a prior TYPE command was executed to select a PROM device type.

<startaddr> denotes the first location in the PROM to test. Together with the <endaddr>, this parameter specifies the range of addresses to be tested. Its default is address 0.

<endaddr> denotes the last location in the PROM to test. This parameter can be optionally specified as a length, L<length>. The <endaddr> is then <startaddr> + <length>-1. If <endaddr> is not specified, <endaddr> defaults to the size of the PROM - 1.

If the command keyword is entered with no parameters, the test is performed on the full PROM. A part of the PROM can be checked by specifying the additional parameters as shown. If the PROM device is in the unprogrammed state, iPPS responds with the message:

PROM SEGMENT BLANK

Otherwise, the message

-COMMAND ERROR--BLANKCHECK TEST FAILED.

is displayed.

EXAMPLES

In the following example, a PROM is checked and found to be blank.

```
PPS>BLANKCHECK PROM
PROM SEGMENT BLANK
```

In the next example, the first 256 addresses of the PROM are checked and at least one address is found to have been previously programmed. Valid abbreviations are used for the keywords.

```
PPS>BLANK PR(0,L100)
-COMMAND ERROR--BLANKCHECK TEST FAILED.
```

The command

```
PPS>B
PROM SEGMENT BLANK
PPS>
```

uses the shortest valid abbreviation for the command. In this case the PROM device tested was completely blank.

COPY

Transfer Data

SYNTAX

COPY <source> [(<startaddr>[,<endaddr>])]
TO <dest> [(<destaddr>)] [<switches>]

OPERATION

The COPY command is a general purpose command that programs PROM devices, reads files from disk, and performs other data transfers. Different versions of the COPY command used for specific devices are described in detail on the following pages.

<source> specifies the source device as any one of the four devices recognized by iPPS: PROM, Buffer, File, or URAM.

<dest> specifies the destination as any one of the four recognized devices; however, the <dest> cannot be the same as the <source>. If this is attempted an error message occurs.

iPPS does not accept COPY operations between URAM and PROM.

<switches> specifies any (or all) of the three valid command switches discussed previously in the section COMMAND SWITCHES. Some of the command swithces are not valid with some versions of the COPY command. Valid command switches for a given version of COPY are indicated in the description for that version.

The terms <startaddr>, <endaddr>, and <destaddr> are described generally in the section of this chapter titled "Special iPPS Syntax Terms". Their defaults are described in the section titled "Command Defaults". That section contains a convenient Table of iPPS Command Defaults. More detail about these parameters can also be found in the specific COPY command descriptions.

After the completion of every copy command, a checksum is displayed. The checksum is the 2's complement of a 16 bit sum of all the bytes copied to the destination device. Any carries out of the 16 bit cummulative sum are ignored during the summation. The final checksum is produced by calculating the 2's complement of the final cummulative sum.

COPY

File to PROM

SYNTAX

```
COPY <file>  [(<startaddr>[,<endaddr>])]

     TO PROM  [(<destaddr>)]  [F]  [ { 80 }  ]  [P]
                                   { 86 }
                                   { 286}
```

OPERATION

The COPY File to PROM command allows the user to program a PROM directly with data from a disk file. By programming directly from a file, the intermediate step of copying the data from the file to the Buffer is eliminated. If the data in the file must be modified before programming the PROM, use the COPY File to Buffer command.

The TYPE command must be executed prior to invoking this command.

This command allows multiple PROM devices to be programmed by prompting for the next device after the programming of each is completed.

COPY File to PROM automatically performs the BLANKCHECK operation to verify that the PROM device is blank. The VERIFY operation is automatically performed after each byte or word of the PROM is programmed and again after all the locations have been programmed. Note that the OVERLAY operation is not performed by this command.

<file> specifies the normal ISIS-II file name for the source file.

<startaddr> specifies the start address of the section of data in the file to be copied into the PROM device. If <startaddr> is not entered then it defaults to the lowest address encountered while reading in the file. Its omission inherently implies that the <endaddr> default is also assumed. If <startaddr> is lower than the lowest address in the file or higher than the highest address in the file, a warning message is displayed.

<endaddr> specifies the end address for the section of File data to be copied. This parameter can be optionally specified as a length, L<length>. The <endaddr> is then <startaddr> + <length>-1. If <endaddr> is not specified then it defaults to the highest address encountered while reading in the file.

<destaddr> specifies the desired start address for the destination data in the PROM device. This parameter can be optionally specified as an offset, O<offset>, relative to <startaddr>. The <destaddr> is then <startaddr> + <offset>. The relative <offset> can be either positive or negative as indicated by a + sign or a - sign. If the relative offset results in a destination starting address less than zero or greater than OFFFFFFH (16777215T), an error message is displayed. If <destaddr> exceeds (PROM size - 1), an error message is displayed. If <destaddr> is not specified it defaults to 0.

[F] is the optional data switch which complements the data before it is programmed. If no data switch is specified, the data is programmed into the PROM device in its uncomplemented form.

{ 80 }
{ 86 }
{286 } is the optional file format switch selection. If this switch is not specified then the currently selected default file format is used.

[P] is the optional patch switch which allows the reading of patched object files (i.e., files with overlapped sections of data). If this switch is not specified and a patched file is read, the following error message will be displayed and the COPY command will be aborted.

-GENERAL ERROR--DATA OVERLAP IN THE FILE.

When a particular address range is specified to be read from a file, all sections in that address range that are not present in the file are written in the PROM with the blank state of the currently selected PROM type (all ones if no PROM type has been selected).

The starting and ending addresses can be specified if only a part of the PROM is to be programmed or if the file size is greater than the PROM size and only part of the file is to be copied.

If the address range specified is greater than the size of the PROM or if the part of the file to be copied is larger than the size of the PROM device, the user is prompted to load subsequent PROM devices into the programming socket. They are programmed sequentially until one of the following conditions is met:

- an end-of-file occurs,
- the address range specified (or defaulted) on the command line has been programmed, or,
- the user depresses the <ESC> key to abort the operation.

EXAMPLES

In the following example, a PROM is programmed with data from file CODE4.IUP on drive :F1:. The data is programmed into the PROM device starting at address 400H.

```
PPS>COPY :F1:CODE4.IUP TO PROM(400)
 CHECK SUM = 6742
```

In the next example, the data in the file exceeds the size of one PROM device. Thus more than one PROM device can be programmed from the data in the source file. In such a case, the command prompts for each new PROM to be programmed.

```
PPS>C  :F1:OBJPGM TO PROM  80
----CAUTION---- PROGRAMMING THE FULL LENGTH REQUIRES MORE THAN ONE PROM.
 CHECK SUM = 619F
FIRST INSTALL THE NEW/NEXT PROM AND THEN CONTINUE.
 CONTINUE--Y/N?Y
 CHECK SUM = B43C
FIRST INSTALL THE NEW/NEXT PROM AND THEN CONTINUE.
 CONTINUE--Y/N?Y
 CHECK SUM = CEA5
PPS>
```

COPY

PROM to File

SYNTAX

COPY PROM [(<startaddr>[,<endaddr>])]
TO <file> [(<destaddr>)] [F]

OPERATION

The COPY PROM to File command transfers data directly from the PROM device to the file specified on the command line. If a file with the same name already exists on the specified disk, the user is prompted before the existing file is overwritten. The TYPE command must be executed prior to invoking this command.

<startaddr> specifies the start address of the section of PROM data to be copied to a file. If <startaddr> is not entered then it defaults to 0. Its omission inherently implies that the <endaddr> default is also assumed.

<endaddr> specifies the end address for the section of PROM data to be copied. This parameter can be optionally specified as a length, L<length>. The <endaddr> is then <startaddr> + <length>-1. If <endaddr> is not specified then it defaults to PROM size -1. The PROM size is determined by the most recent TYPE command.

<file> specifies the normal ISIS-II file name for the file to be written.

<destaddr> specifies the start address of the destination data in the iPPS File device. This parameter can be optionally specified as an offset, O<offset>, relative to <startaddr>. The <destaddr> is then <startaddr> + <offset>. The relative <offset> can be either positive or negative as indicated by a + sign or a - sign. If the relative offset results in a destination starting address less than zero or greater than OFFFFFFH (16777215T), an error message is displayed. If <destaddr> is not specified, it defaults to 0.

[F] is the only valid Command Switch. This is the optional data switch which complements the data before it is written. If no data switch is specified, the data is written into the file in its uncomplemented form.

All files written with this command are in the 286 Absolute file format described in Appendix C.

EXAMPLES

In the following example, the first 1K of PROM data (from 0 for a length of 400H bytes) is copied into the file CODE4.IUP on drive :F1:.

```
PPS>COPY PROM(0,L400) TO :F1:CODE4.IUP
 CHECK SUM = 6472
PPS>
```

In the next example, the complete complemented contents of the PROM device are copied to file :F1:CMPL4.IUP. Note that lower case is used for the keyin and makes no difference.

```
PPS>c p t :f1:cmpl4.iup  f
 CHECK SUM = FE55
PPS>
```

If the PROM keyword or the file is not specified, iPPS prompts for the missing items. In this next example, an error was made in typing in the source PROM address range. An extra F was typed in the address specification. This caused an address range that exceeded that for the current specified PROM device type.

The example shows how the ALTER command can be used to conveniently correct and restart a command. See the ALTER command description in this chapter for more information. In the following example, the ALTER command shows the previous erroneous command and allows editing of it. The cursor is advanced to one of the Fs and a <CTRL-D> is pressed to delete one. A <CR> is then pressed to indicate to ALTER that editing of the line is complete. ALTER then shows the edited command and prompts for execution of it with -Y/N?. As shown, pressing Y then causes iPPS to execute the edited command as though it had been normally keyed in.

```
PPS>C P(400,7FFF)
TO? :F1:CODE5.IUP
-GENERAL ERROR--FINAL ADDRESS OUT OF RANGE.
PPS>ALTER
C P(400,7FFF) TO :F1:CODE5.IUP
C P(400,7FF) TO :F1:CODE5.IUP
-Y/N? Y
 CHECK SUM = A539
PPS>
```

COPY

Buffer to PROM

SYNTAX

```
COPY BUFFER [(<startaddr>[,<endaddr>])]
     TO PROM [(<destaddr>)] [F]
```

OPERATION

The COPY Buffer to PROM command programs the PROM device with the specified data in the Buffer. The TYPE command must be executed prior to attempting this command. See the TYPE command for further details.

This command automatically performs the BLANKCHECK operation to verify that the PROM device is blank. If the device is not in its erased state, the OVERLAY operation is also performed to determine if the device can still be programmed. The VERIFY operation is automatically performed after each byte or word of the PROM is programmed and again after all the locations have been programmed.

<startaddr> specifies the start address of the section of data in the Buffer to be copied into the PROM device. If <startaddr> is not entered then it defaults to the Buffer start address. Its omission inherently implies that the <endaddr> default is also assumed.

<endaddr> specifies the end address for the section of Buffer data to be copied. This parameter can be optionally specified as a length, L<length>. The <endaddr> is then <startaddr> + <length>-1. If <endaddr> is not specified then it defaults to an address determined by the expression, (Buffer start address) + (Buffer size -1). If the address range specified is greater than the size of the PROM then an error message is displayed.

<destaddr> specifies the desired start address for the destination data in the iPPS Buffer device. This parameter can be optionally specified as an offset, O<offset>, relative to <startaddr>. The <destaddr> is then <startaddr> + <offset>. The relative <offset> can be either positive or negative as indicated by a + sign or a - sign. If the relative offset results in a destination starting address less than zero or greater than OFFFFFFH (16777215T), an error message is displayed. If <destaddr> is not specified it defaults to 0. If <destaddr> exceeds (PROM size - 1), an error message occurs and the COPY command is aborted.

[F] is the optional data switch which complements the data before it is programmed into the PROM device. If no data switch is specified, the data is programmed in its uncomplemented form.

EXAMPLES

In the following example, the PROM device is programmed with the contents of the buffer.

```
PPS>COPY BUFFER TO PROM
 CHECK SUM = 09AB
```

The next example copies the contents of a specified range of buffer addresses to the PROM. The range is given in octal using number base tail specifications. Buffer data starting at 2000Q (400H) and ending at 2777Q (5FFH) are programmed into the PROM device starting at its address 0.

```
PPS>COP BUFF (2000Q,2777Q) TO PROM
 CHECK SUM = 512A
```

If the Buffer or PROM is not specified, iPPS prompts for the missing keywords. Also note that this next example attempts to program a non-blank PROM device. In such a case, iPPS prompts for an OVERLAY test. See the description of the OVERLAY command in this chapter for more information.

```
PPS>C B
TO? P
PROM NOT BLANK--OVERLAY TEST--Y/N? Y
OVERLAY ERROR--DISPLAY Y/N? Y
  PROM ADDRESS             PROM DATA  EXPECTED DATA
000400                       FB         04
000401                       FE         C3
-COMMAND ERROR--OVERLAY TEST FAILED.
PPS>
```

COPY

PROM to Buffer

SYNTAX

COPY PROM [(<startaddr>[,<endaddr>])]
TO BUFFER [(<destaddr>)] [F]

OPERATION

The COPY PROM to Buffer command reads data from the PROM device into the Buffer. Data stored in the Buffer prior to execution of this command is overwritten. The TYPE command must be executed prior to invoking this command.

<startaddr> specifies the start address of the section of data in the PROM to be copied into the Buffer. If <startaddr> is not entered then it defaults to 0. Its omission inherently implies that the <endaddr> default is also assumed.

<endaddr> specifies the end address for the section of PROM data to be copied. This parameter can be optionally specified as a length, L<length>. The <endaddr> is then <startaddr> + <length>-1. If <endaddr> is not specified then it will default to PROM size - 1. PROM size is determined by the most recent TYPE command.

<destaddr> specifies the desired start address for the destination data in the iPPS Buffer device. This parameter can be optionally specified as an offset, O<offset>, relative to <startaddr>. The <destaddr> is then <startaddr> + <offset>. The relative <offset> can be either positive or negative as indicated by a + sign or a - sign. If the relative offset results in a destination starting address less than zero or greater than 0FFFFFFH (16777215T), an error message is displayed. If <destaddr> is not specified it defaults to 0.

[F] is the optional data switch which complements the data before it is written. If no data switch is specified, the data is written into the Buffer in its uncomplemented form.

EXAMPLES

In this example, the contents of the PROM are copied to the Buffer.

```
PPS>COPY PROM TO BUFFER
 CHECK SUM = 572F
```

In the following example, 256 bytes of the data in the PROM starting at address 100H are complemented as they are copied to the Buffer.

```
PPS>C P(100,1FF) T BUF F
 CHECK SUM = 6AF2
```

If the source or destination devices are not specified, iPPS prompts for the missing keywords. This next example copies 256 bytes of PROM data into the Buffer starting at Buffer address 300H. The L100 specifies that the source data is a section of length 100H bytes (or 256 in decimal).

```
PPS>C
FROM? P(200,L100)
TO? B(300)
 CHECK SUM = B840
```

COPY

Buffer to File

SYNTAX

COPY BUFFER [(<startaddr>[,<endaddr>])]

TO <file> [(<destaddr>)] [F]

OPERATION

The COPY Buffer to File command saves the memory image of the iPPS Buffer in a diskette or a hard disk file. If a file with the same name already exists on the specified disk, the user is prompted before the existing file is overwritten.

<startaddr> specifies the start address of the section of Buffer data to be copied to a file. If <startaddr> is not entered it defaults to the present Buffer start address. Its omission inherently implies that the <endaddr> default is also assumed.

<endaddr> specifies the end address for the section of Buffer data to be copied. This parameter can be optionally specified as a length, L<length>. The <endaddr> is then <startaddr> + <length>-1. If <endaddr> is not specified, it defaults to the present Buffer end address. This address is determined by the following expression: (Buffer start address) + (Buffer size - 1).

<file> specifies the normal ISIS-II file name for the file to be written.

<destaddr> specifies the start address of the destination data in the iPPS File device. This parameter can be optionally specified as an offset, O<offset>, relative to <startaddr>. The <destaddr> is then <startaddr> + <offset>. The relative <offset> can be either positive or negative as indicated by a + sign or a - sign. If the relative offset results in a starting address less than zero or greater than OFFFFFFH (16777215T), an error message is displayed. If <destaddr> is not specified, it defaults to 0.

[F] is the only valid Command Switch. This is the optional data switch which complements the data before it is written. If no data switch is specified, the data is written into the file in its uncomplemented form.

All files written with this command are in the 286 Absolute file format described in Appendix C.

EXAMPLES

The following command saves the contents of the buffer in file CODE1.IUP on drive 1.

```
PPS>COPY BUFFER TO :F1:CODE1.IUP
 CHECK SUM = 09AB
```

In the next example, only a part of the buffer is copied to a file. Abbreviations are used for the keywords.

```
PPS>C B(0,3FF) T :F1:CODE2.IUP
 CHECK SUM = 6742
```

The next command saves the full complemented contents of the buffer in file CODE3.IUP on drive :F1:.

```
PPS>C B T :F1:CODE3.IUP  F
 CHECK SUM = FE55
```

In the following command, no destination is specified, so iPPS prompts for the filename.

```
PPS>C B(0,3FF)
TO? :F1:CODE4.IUP
 CHECK SUM = 6472
```

If no source is specified, iPPS prompts for that too. For the following example, the data in the Buffer starting at address 400H and ending at the Buffer end address (eg. 7FFH for a 2K Buffer) is copied to file :F1:CODE5.IUP.

```
PPS>C
FROM? B(400)
TO? :F1:CODE5.IUP
 CHECK SUM = A539
```

COPY

File to Buffer

SYNTAX

COPY <file> [(<startaddr>[,<endaddr>])]

TO BUFFER [(<destaddr>)] [F] [80 | 86 | 286] [P]

OPERATION

The COPY File to Buffer command transfers a data file into the Buffer device from a disk file. This command is used when the data in the file must be modified in some way before it can be used in further iPPS commands. In such a case, the Buffer acts as a temporary holding area for the data during modifications.

<file> specifies the normal ISIS-II file name for the source file.

<startaddr> specifies the start address of the section of data in the file to be copied into the Buffer. If <startaddr> is not entered then it defaults to the lowest address encountered while reading in the file. Its omission inherently implies that the <endaddr> default is also assumed.

<endaddr> specifies the end address for the section of File data to be copied. This parameter can be optionally specified as a length, L<length>. The <endaddr> is then <startaddr> + <length>-1. If an <endaddr> is specified such that the address range (<endaddr> - <startaddr> + 1) exceeds the current Buffer size, an error message is displayed. If <endaddr> is not specified, then it defaults to the smaller of the following: the file highest address or (<startaddr> + Buffersize - 1). This assures the address range never exceeds the Buffer size. Note that to determine the defaults for <startaddr> or <endaddr>, iPPS must perform an extra pass of the file.

<destaddr> specifies the desired start address for the destination data in the iPPS Buffer device. This parameter can be optionally specified as an offset, O<offset>, relative to <startaddr>. The <destaddr> is then <startaddr> + <offset>. The relative <offset> can be either positive or negative as indicated by a + sign or a - sign. If the relative offset results in a destination starting address less than zero or greater than 0FFFFFFH (16777215T), an error message is displayed. If <destaddr> is not specified it defaults to 0.

[F] is the optional data switch which complements the data before it is written. If no data switch is specified, the data is written in its uncomplemented form.

{80 / 86 / 286} is the optional file format switch selection. If this switch is not specified than the currently selected default file format is used.

[P] is the optional patch switch which allows the reading of patched object files (i.e., files with overlapped sections of data). If this switch is not specified and a patched file is read, the following error message will be displayed and the COPY command will be aborted.

-GENERAL ERROR--DATA OVERLAP IN THE FILE.

When a particular address range is specified to be read from a file, all sections in that address range that are not present in the file are not affected in the Buffer (ie. the data in these sections of the Buffer remain unchanged).

EXAMPLES

In the following example data is copied from a file into the Buffer starting at Buffer address 400H.

```
PPS>COPY :F1:CODE4.IUP TO BUFFER(400)
 CHECK SUM = 6472
```

If the source file or the keyword BUFFER is not specified, iPPS prompts for those devices. In the next example, the address L100 specifies the range as a length (in Hex) from the file data start address of 200H. The data is copied to the buffer starting at its address 300H. The copy is directed to read the data from a file formatted in the 8086 format.

```
PPS>C
FROM? :F1:CODE5 (200,L100)
TO? B(300) 86
 CHECK SUM = B8B8
```

COPY

File to URAM

SYNTAX

```
COPY <file>  [(<startaddr>[,<endaddr>])]

     TO URAM  [(<destaddr>)]  [F][ 80 | 86 | 286 ]  [P]
```

OPERATION

The COPY File to URAM command transfers data from a file directly to the URAM for off-line operations with the data. The COPY File to URAM command can only be used with the iUP 201 model of the Universal Programmer.

<file> specifies the normal ISIS-II file name for the source file.

<startaddr> specifies the start address of the section of data in the file to be copied into the URAM device. If <startaddr> is not entered then it defaults to the lowest address encountered while reading in the file. Its omission inherently implies that the <endaddr> default is also assumed.

<endaddr> specifies the end address for the section of File data to be copied. This parameter can be optionally specified as a length, L<length>. The <endaddr> is then <startaddr> + <length>-1. If <endaddr> is not specified then it defaults to the smaller of the File's highest address or (<destaddr> + URAM size - 1). If the address range (<endaddr> - <startaddr> + 1) is larger than the URAM size an error message occurs.

<destaddr> specifies the desired start address for the destination data in the URAM device. This parameter can be optionally specified as an offset, O<offset>, relative to <startaddr>. The <destaddr> is then <startaddr> + <offset>. The relative <offset> can be either positive or negative as indicated by a + sign or a - sign. If the relative offset results in a destination starting address less than zero or greater than OFFFFFFH (16777215T), an error message is displayed. If <destaddr> exceeds (URAM size -1), an error message occurs. If <destaddr> is not specified, it defaults to 0.

[F] is the optional data switch which complements the data before it is programmed. If no data switch is specified, the data is written into the PROM device in its uncomplemented form.

{ 80 / 86 / 286 } is the optional file format switch selection. If this switch is not specified then the currently selected default file format is used.

[P] is the optional patch switch which allows the reading of patched object files (i.e., files with overlapped sections of data). If this switch is not specified and a patched file is read, the following error message will be displayed and the COPY command will be aborted.

-GENERAL ERROR--DATA OVERLAP IN THE FILE.

When a particular address range is specified to be read from a file, all sections in that address range that are not present in the file are written in the URAM destination device with the blank state of the currently selected PROM type (all ones if no PROM type has been selected).

EXAMPLES

In the following example, data from file CODE4.IUP on drive :F1: is copied (starting at address 0 in the file data) to the URAM (starting at URAM address 400H).

```
PPS>COPY :F1:CODE4.IUP TO URAM(400)
 CHECK SUM = 6472
```

In the next example, previously complemented data in file CMPL4.IUP on drive :F1: is copied to the URAM (starting at URAM address 0). The data is complemented back to its original uncomplemented form. Abbreviations are used for the keywords.

```
PPS>C :F1:CMPL4.IUP T U  F
 CHECK SUM = 6472
```

If the source or destination are not specified, iPPS prompts for them. In this next example, the data in an object program file (OBJPGM on drive :F1:) is copied into the URAM (starting at URAM address 0). In the first attempt, the default file type (286) is assumed and an error occurs because the OBJPGM file format is not 286

format. In this case, it is 8080 absolute object format. The command is correctly executed using the 80 file switch.

```
PPS>C
FROM? :F1:OBJPGM
TO? U
-GENERAL ERROR--ILLEGAL FILE TYPE SPECIFIED.
PPS>C
FROM? :F1:OBJPGM
TO?U  80
 CHECK SUM = 2C38
```

COPY

URAM to File

SYNTAX

COPY URAM [(<startaddr>[,<endaddr>])]
TO <file> [(<destaddr>)] [F]

OPERATION

The COPY URAM to File command transfers data directly from the URAM device to the file specified on the command line. The COPY URAM to File command can only be used with the iUP-201 model of the Universal Programmer. If a file with the same name already exists on the specified disk, the user is prompted before the existing file is overwritten.

<startaddr> specifies the start address of the section of URAM data to be copied to a file. If <startaddr> is not entered then it defaults to 0. Its omission inherently implies that the <endaddr> default is also assumed.

<endaddr> specifies the end address for the section of URAM data to be copied. This parameter can be optionally specified as a length, L<length>. The <endaddr> is then <startaddr> + <length>-1. If <endaddr> is not specified then it defaults to URAM size -1.

<file> specifies the normal ISIS-II file name for the file to be written.

<destaddr> specifies the start address of the destination data in the iPPS File device. This parameter can be optionally specified as an offset, O<offset>, relative to <startaddr>. The <destaddr> is then <startaddr> + <offset>. The relative <offset> can be either positive or negative as indicated by a + sign or a - sign. If the relative offset results in a destination starting address less than zero or greater than OFFFFFFH (16777215T), an error message is displayed. If <destaddr> is not specified, it defaults to 0.

[F] is the only valid Command Switch. This is the optional data switch which complements the data before it is written. If no data switch is specified, the data is written into the file in its uncomplemented form.

All files written with this command are in the 286 Absolute file format described in Appendix C.

EXAMPLES

In the following example, the entire contents of the URAM is copied to file URAM1.IUP on drive :F1:. The file is written in the 286 format.

```
PPS>COPY URAM TO :F1:URAM1.IUP
 CHECK SUM = 2B93
```

In the next example, the first 1024 (1K) bytes of the URAM data is copied to file URAM2.IUP on drive :F1: (starting at destination file address 400H).

```
PPS>C U(0,L400) TO :F1:URAM2.IUP(400)
 CHECK SUM = 6472
```

If the source or destination are not specified, iPPS prompts for them. In this next example, data in the URAM starting at 400H and ending at 7FFH is copied to file URAM3.IUP on drive :F1: (starting at file address 800H). The destination starting address in the file is specified as an offset (0400) from the source start address of 400H in the URAM. Note that this command establishes a new file start address of 800H and a new file final address of BFFH.

```
PPS>C
FROM?U (400,7FF)
TO?:F1:URAM3.IUP  (0400)
 CHECK SUM = A539
```

COPY

Buffer to URAM

SYNTAX

COPY BUFFER [(<startaddr>[,<endaddr>])]

TO URAM [(<destaddr>)] [F]

OPERATION

This command transfers data from the iPPS Buffer to the RAM contained in the iUP-201 for off-line operation. The COPY Buffer to URAM command can only be used with the iUP-201 model of the Universal Programmer. If the Buffer address range specified is greater than the size of the URAM, an error message is displayed.

<startaddr> specifies the start address of the section of data in the Buffer to be copied into the URAM device. If <startaddr> is not entered then it defaults to the Buffer start address. Its omission inherently implies that the <endaddr> default is also assumed.

<endaddr> specifies the end address for the section of Buffer data to be copied. This parameter can be optionally specified as a length, L<length>. The <endaddr> is then <startaddr> + <length>-1. If <endaddr> is not specified then it defaults to an address determined by the expression, (Buffer start address) + (Buffer size -1). If <endaddr> (either specified or defaulted) is higher than (URAM size -1), then <endaddr> is instead forced to the value, (Buffer start address + URAM size -1).

<destaddr> specifies the desired start address for the destination data in the URAM device. This parameter can be optionally specified as an offset, O<offset>, relative to <startaddr>. The <destaddr> is then <startaddr> + <offset>. The relative <offset> can be either positive or negative as indicated by a + sign or a - sign. If the relative offset results in a destination starting address less than zero or greater than OFFFFFFH (16777215T), an error message is displayed. If <destaddr> exceeds (URAM size - 1), an error message occurs. If <destaddr> is not specified, it defaults to 0.

[F] is the optional data switch which complements the data before it is programmed into the URAM device. If no data switch is specified, the data is programmed in its uncomplemented form.

EXAMPLES

In the following example, the contents of the buffer are copied to the local URAM.

```
PPS>COPY BUFFER TO URAM
 CHECK SUM = 09AB
```

In the next example, 1K bytes of Buffer data (starting at 400H) are complemented and copied to the URAM (starting at URAM address 0). Abbreviations are used for the keywords.

```
PPS>C B(400,L400) T U  F
 CHECK SUM = 5EC7
```

If the source or destination are not specified, iPPS prompts for the missing keywords. In this next example, 1024 (1K) bytes of Buffer data (starting at Buffer address 100H) are copied to the URAM (starting at URAM address 500H). The URAM destination start address is specified as an offset from the start of the 100H source Buffer address.

```
PPS>C
FROM?B(100,4FF)
TO?U(0400)
 CHECK SUM = 6472
```

COPY

URAM to Buffer

SYNTAX

COPY URAM [(<startaddr>[,<endaddr>])]
TO BUFFER [(<destaddr>)] [F]

OPERATION

This command reads data from the iUP-201 RAM into the Buffer device. The COPY URAM to Buffer command can only be used with the iUP-201 model of the Universal Programmer. Data stored in the iPPS buffer prior to execution of this command is overwritten.

<startaddr> specifies the start address of the section of data in the URAM to be copied into the Buffer. If <startaddr> is not entered then it defaults to 0. Its omission inherently implies that the <endaddr> default is also assumed.

<endaddr> specifies the end address for the section of URAM data to be copied. This parameter can be optionally specified as a length, L<length>. The <endaddr> is then <startaddr> + <length>-1. If <endaddr> is not specified then it defaults to the smaller of (URAM size - 1) or (<startaddr> + Buffer size - 1). If the address range (<endaddr> - <startaddr> +1) is larger than the Buffer size or larger than the URAM size, then an error message occurs.

<destaddr> specifies the desired start address for the destination data in the iPPS Buffer device. This parameter can be optionally specified as an offset, O<offset>, relative to <startaddr>. The <destaddr> is then <startaddr> + <offset>. The relative <offset> can be either positive or negative as indicated by a + sign or a - sign. If the relative offset results in a destination starting address less than zero or greater than OFFFFFFH (16777215T), an error message is displayed. If <destaddr> is not specified, it defaults to 0.

[F] is the optional data switch which complements the data before it is written. If no data switch is specified, the data is written into the Buffer in its uncomplemented form.

EXAMPLES

In the following example, 1024 (1K) bytes of URAM data (starting at URAM address 400H) are copied to the Buffer (starting at Buffer address 0). The section of source data is specified as a length of 1K (400H) from the source start address.

```
PPS>COPY URAM(400,L400) TO BUFFER
 CHECK SUM = 6472
```

In the next example, the data in the URAM from address 100H to 4FFH is copied to the Buffer (starting at Buffer address 400H). The destination start address in the Buffer is specified as a 300H offset from the URAM start address of 100H. Abbreviations are used for some of the keywords. Also note that lower case is used for some of the keyin. This makes no difference to iPPS.

```
PPS>COP  URAM(100,4ff) to buf(o300)
 CHECK SUM = 5EC7
```

If the source or destination are not specified, iPPS prompts for the missing keywords. In this next example, 1024 (1K) bytes of URAM data (starting at URAM address 400H) are copied to the Buffer (starting at Buffer address 1000H). Note that this command establishes a new Buffer start address of 1000H.

```
PPS>C
FROM?U(400,L400)
TO?B(1000)
 CHECK SUM = 6472
```

DISPLAY

Displays Data on Console

SYNTAX

DISPLAY { PROM / BUFFER } [(<startaddr>[,<endaddr>])] [F] [{ Y / O / Q / T / H }]

OR

DISPLAY <file> [(<startaddr>[,<endaddr>])] [F] [{ 80 / 86 / 286 }] [{ Y / O / Q / T / H }] [P]

OPERATION

The DISPLAY command outputs data from the PROM, Buffer, or File devices in a formatted display on the console. As shown above, the DISPLAY <file> command is different in form compared to DISPLAY PROM and DISPLAY BUFFER.

<file> specifies the normal ISIS-II file name for a source file.

<startaddr> specifies the start address of the section of data in the source device to be displayed on the console. Together with the <endaddr>, this parameter specifies the range of addresses to be displayed. The default for <startaddr> depends on the source of the data. If the source is PROM, <startaddr> defaults to 0. If the source is BUFFER, its default is the present Buffer start address. If the source is a file, its default is the lowest address encountered while reading in the file. The omission of <startaddr> inherently implies that the <endaddr> default is also assumed.

<endaddr> specifies the end address for the section of source data to be displayed. This parameter can be optionally specified as a length, L<length>. The <endaddr> is then <startaddr> + <length>-1. The default for <endaddr> depends on the source of the data. If the source is PROM, <endaddr> defaults to (PROM size -1). PROM size is determined by the most recent TYPE command. If the source is BUFFER, <endaddr> defaults to an address determined by the expression, (Buffer start address) + (Buffer size -1). If the source is a file, <endaddr> defaults to the highest address encountered while reading in the file.

[F] is the optional data switch which complements the data before it is displayed. If no data switch is specified, the data is displayed in its uncomplemented (true) form.

{Y, O, Q, T, H} is the optional base switch. Only one of these can be specified at once. It specifies the number base to be used for the duration of the command. Binary is specified by the letter Y; octal by O or Q; decimal by T; and Hexadecimal by H. The default for this switch is the current default number base determined by the most recent INITIALIZE command.

{80, 86, 286} is the optional file format switch selection. If this switch is not specified then the currently selected default file format is assumed when reading the file for display.

[P] is the optional patch switch which allows the reading of patched object files (i.e., files with overlapped sections of data). If this switch is not specified and a patched file is read, the following error message will be displayed and the DISPLAY command will be aborted.

-GENERAL ERROR--DATA OVERLAP IN THE FILE.

When a particular address range is specified to be read from a file, all sections in the address range that are not present in the file are displayed with the blank state of the currently selected PROM type (all 1s if no PROM type has been selected).

The data is displayed one page at a time. At the end of a page, a pause between succeeding pages is indicated by the message:

ENTER <CR> TO CONTINUE

To continue the display of data, simply type the RETURN key. Any other keys that are typed will be echoed on the screen but will not cause display continuation. Press the ESC key at any time to abort the DISPLAY command and return the iPPS prompt.

The format of the data displayed varies with the number base currently in effect (or specified for the duration of this command). To the right of the actual data is the ASCII character equivalent for each displayed byte value. If there is no meaningful ASCII character equivalent, a . character is displayed as a place holder. See the following Examples section.

EXAMPLES

The following example displays the contents of the PROM device one screenful at a time in hexadecimal (the default number base). This display is from a monitor PROM containing 8085 code. Note that the data displayed has some embedded ASCII character strings. The ASCII equivalent display is convenient for revealing this kind of content.

```
PPS>DISPLAY PROM
000000: C3 40 00 20 20 44 20 2D 20 44 49 53 4B 00 20 20      .@.  D - DISK.
000010: 47 20 2D 20 47 45 4E 45 52 41 4C 00 20 20 4B 20      G - GENERAL.  K
000020: 2D 20 4B 45 59 42 4F 41 52 44 2F 43 52 54 00 FF      - KEYBOARD/CRT..
000030: FF FF FF FF FF FF FF FF C3 36 1C FF FF FF FF FF      .........6......
000040: F3 DB 80 E6 20 CA 03 08 3E 00 D3 D1 DB 80 E6 01      .... ...>.......
000050: C2 66 00 3E 4F D3 D0 3E 58 D3 D0 3E 89 D3 D0 3E      .f.>O..>X..>...>
000060: 99 D3 D0 C3 76 00 3E 4F D3 D0 3E 98 D3 D0 3E 8A      ....v.>O..>...>.
000070: D3 D0 3E 9C D3 D0 21 00 00 11 00 08 AF 47 7B B2      ..>...!......G{.
000080: CA 8A 00 78 86 23 1B C3 7D 00 78 FE 55 C2 8D 00      ...x.#..}.x.U...
000090: 3E 34 D3 E3 3E 1F D3 E0 3E 00 D3 E0 01 30 00 DB      >4..>...>....0..
0000A0: 80 E6 01 C2 A9 00 01 2C 00 3E 72 D3 E3 79 D3 E1      .......,.>r..y..
0000B0: 78 D3 E1 3E B2 D3 E3 3E 00 D3 E2 3E 16 D3 E2 D3      x..>...>...>....
0000C0: 10 3E 22 D3 60 D3 50 DB 80 E6 04 CA C7 00 DB 80      .>".`.P.........
0000D0: E6 04 C2 CE 00 AF D3 F0 D3 F0 D3 F1 D3 F1 3E A1      ..............>.
0000E0: D3 F8 3E 23 D3 60 3E C8 D3 E2 3E 00 D3 E2 D3 50      ..>#.`>...>....P
0000F0: 21 EF 00 2B 7C B5 C2 F3 00 DB 80 E6 04 C2 FD 00      !..+|...........
000100: 3E 00 D3 E2 3E 16 D3 E2 D3 50 DB 80 E6 04 CA 0A      >...>....P......
000110: 01 DB 80 E6 04 C2 11 01 3E 22 D3 60 D3 50 DB 80      ........>".`.P..
000120: E6 04 CA 1E 01 DB 80 E6 04 C2 25 01 21 00 40 11      ..........%.!.@.
ENTER <CR> TO CONTINUE $
ABORTED
PPS>
```

Press the ESC key at any time to end the display. The $ is the echo of the ESC key.

The following example displays in octal the contents of the specified range of addresses in the PROM device. The range is specified in hexadecimal.

```
PPS>DISPLAY  PROM(0,L18H) Q
00000000: 303 100 000 040 040 104 040 055                    .@.  D -
00000010: 040 104 111 123 113 000 040 040                     DISK.
00000020: 107 040 055 040 107 105 116 105                    G - GENE
```

In the following example, the contents of the specified range of buffer addresses are displayed in binary. The F range specification is interpreted as hexadecimal since it has no explicit number base tail and the current default number base is assumed to be hexadecimal.

```
PPS>DISP BUFF(0,F) Y
000000000000000000000000: 11000011 01000000 00000000 00100000  .@.
000000000000000000000100: 00100000 01000100 00100000 00101101   D -
000000000000000000001000: 00100000 01000100 01001001 01010011   DIS
000000000000000000001100: 01001011 00000000 00100000 00100000  K.
```

In the next example, the contents of a range of buffer addresses are displayed in complemented form in binary. The range is specified in binary as a length from the start address.

```
PPS>D B(0,L110Y) F Y
000000000000000000000000: 00111100 10111111 11111111 11011111  <...
000000000000000000000100: 11011111 10111011                    ..
```

This next example displays in decimal the contents of a range of buffer addresses. The range is specified in hexadecimal.

```
PPS>D B(0,1AH) T
00000000: 195 064 000 032 032 068 032 045 032 068              .@.  D - D
00000010: 073 083 075 000 032 032 071 032 045 032              ISK.  G -
00000020: 071 069 078 069 082 065 076                         GENERAL
PPS>
```

Returns to Main
iPPS Command Input

SYNTAX

<ESC>

OPERATION

The <ESC> command is entered by depressing ESC key. If the <ESC> command is entered during any request by iPPS for keyboard input, control will be returned to the main command input for iPPS (indicated by the PPS> prompt). Thus, an iPPS command may be terminated at any keyin prompt by entering the ESC key. In such a case, no portion of the command is executed. During certain non-interruptable iPPS operations, the ESC key will be recognized only after that non-interruptable operation is completed. In this case, the following message is displayed:

ABORTED

EXAMPLES

In the following example, the DISPLAY command has been run. After displaying, one screen of data, the ESC key is pressed to return to the iPPS prompt. The ESC key is echoed as a dollar sign.

```
ENTER <CR> TO CONTINUE $
ABORTED
PPS>
```

A new iPPS command can now be entered.

In the next example, a command was entered incorrectly and the ESC key was pressed to return to the iPPS prompt and re-enter the command.

```
PPS>D X$
PPS>
```

EXIT

Exits iPPS and Returns
Control to ISIS-II

SYNTAX

EXIT

OPERATION

The EXIT command is used to return control to the ISIS-II operating system when the PROM programming session is ended.

Any temporary files created by iPPS are deleted, and any files opened by iPPS are closed. Return to the ISIS-II operating system is indicated on the console by the ISIS-II prompt, >.

EXAMPLES

The following example returns to the ISIS-II operating system as indicated by the > prompt.

```
PPS>E
>
```

FORMAT

Interactively Formats Data

SYNTAX

FORMAT {PROM / BUFFER} [(<startaddr>[,<endaddr>])] [F]

OR

FORMAT <file> [(<startaddr>[,<endaddr>])] [F] [{80 / 86 / 286}] [P]

OPERATION

The FORMAT command allows the complex manipulation of data in the PROM, Buffer, or File devices. The command can be utilized for operations such as interleaving, nibble swapping, and bit reversal. It also allows the creation of multiple output files from a single input file. The FORMAT command is flexible and can be used to perform many other types of data manipulation.

Due to its complexity, the FORMAT command operates in an interactive fashion. Once the command is entered, the user is prompted with a series of questions, asking for different needed parameters. The command may be terminated at the time of any of these prompts by entering the <ESC> key.

<file> specifies the normal ISIS-II file name for a source file.

<startaddr> specifies the start address of the section of data in the source device to be manipulated. Together with the <endaddr>, this parameter specifies the range of addresses to be formatted. The default for <startaddr> depends on the source of the data. If the source is PROM, <startaddr> defaults to 0. If the source is BUFFER, its default is the present Buffer start address. If the source is a file, its default is the lowest address encountered while reading in the file. The omission of <startaddr> inherently implies that the <endaddr> default is also assumed.

<endaddr> specifies the end address for the section of source data to be formatted. This parameter can be optionally specified as a length, L<length>. The <endaddr> is then <startaddr> + <length>-1. The default for <endaddr> depends on the source of the data. If the source is PROM, <endaddr> defaults to (PROM size -1). PROM size is determined by the most recent TYPE command. If the source is BUFFER, <endaddr> defaults to an address determined by the expression, (Buffer start address) + (Buffer size -1). If the source is a file, <endaddr> defaults to the highest address encountered while reading in the file.

[F] is the optional data switch which complements the data before it is written to the output file. If no data switch is specified, the data is output in its uncomplemented (true) form.

{80 / 86 / 286} is the optional file format switch selection. If this switch is not specified then the currently selected default file format is assumed when reading in the source file.

[P] is the optional patch switch which allows the reading of patched object files (i.e., files with overlapped sections of data). If this switch is not specified and a patched file is read, the following error message will be displayed and the FORMAT command will be aborted.

-GENERAL ERROR--DATA OVERLAP IN THE FILE.

When a particular address range is specified to be read from a file, all sections in the address range that are not present in the source file are written in the FORMAT's output destination file with the blank state of the currently selected PROM type (all 1s if no PROM type has been selected).

EXAMPLES

The following example starts an interactive session in which PROM data is formatted into an output file.

```
FORMAT PROM
```

The next example starts an interactive session in which the first 256 bytes of Buffer data are formatted.

```
F BUFF(0,FF)
```

The next example starts an interactive session where data in source file :F1:PROGA (in 8086 format) is formatted.

```
F :F2:PROGA 86
```

INTERACTIVE FORMAT OPERATION

The FORMAT command acts on an array of input data to produce a formatted output array of data in a file. The data manipulation software breaks the full array of input data into small blocks. The size of these blocks depends on the type of manipulation desired. The data of each of these small blocks is manipulated to produce an output block. When all the output blocks are appended, the result is a formatted array of output data. Figure 3-1 shows logically how an output array is produced from an input array in one interactive output cycle of the FORMAT command. More than one output formatted data array can be created from a single input array by repeating the output cycle.

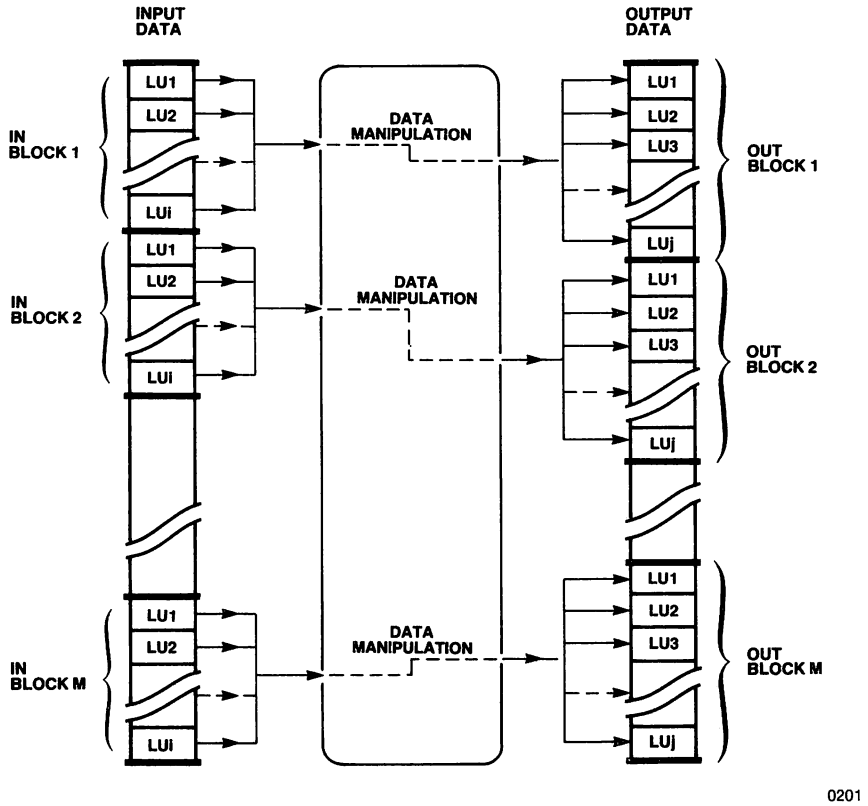


Figure 3-1. FORMAT Command Data Manipulation

Once the interactive mode of the FORMAT command is entered, it prompts for the Logical Unit, Input Block Size, and Output Block Size. After these parameters are entered, the software prompts with a pictorial representation of the Input Buffer Structure, and an asterisk (*). The * prompts for the output specifications. Based on the output specifications entered, the software generates an output array stored in the specified output file. Once the output file has been created, the message

OUTPUT STORED

is displayed on the console. This is followed by the start of another output cycle indicated by the * prompt. Each output cycle allows the creation of variously formatted output files (using the same Logical Unit and Input Block Size). To exit from the FORMAT command, enter <CR> as the first non-blank character in answer to a prompt or simply press <ESC> during any keyin to abort.

The Input Block Size and Output Block Size should be perfect multiples of the Logical Unit. If these sizes are not perfect multiples of the Logical Unit, iPPS truncates the offending block size to a perfect multiple of the Logical Unit.

NOTE

Within the interactive FORMAT command, numeric entries are interpreted with a default number base of decimal. However, a numeric entry may be specified in another base by appending the appropriate number base tail (i.e., binary: Y, octal: O or Q, hexidecimal: H).

In the event of a non-fatal error, an error message is displayed and the FORMAT software prompts for the same parameter again.

To understand the FORMAT command's interactive data manipulation, the following terms must be understood.

Logical Unit

The Logical Unit is the basic unit in which data is to be manipulated. Valid Logical Units are bit, nibble, byte, or N-Bytes (N=integer). In the case of N-Byte, the largest meaningful value is 32,767.

Input Block Size

Input Block Size specifies the size of the smallest block into which the full input data array is to be split. This size is always specified in bytes. Figure 3-1 shows the input array split into M blocks. The Input Block Size should be greater than or equal to the Logical Unit. The total number of Logical Units in an input block size should not exceed 24. The input block size can not exceed 65,535 bytes.

Output Block Size

Output Block Size specifies the size of the output block generated after manipulating the data of the input block. Output Block Size is always specified in bytes. The total number of Logical Units in an output block should not exceed 24. The output block size can not exceed 65,535 bytes.

Input Block Structure

The Input Block Structure shows the way in which the input block is split into i Logical Units. The Logical Unit corresponding to the lowest address or the least significant bit of the input block is assigned the number 0. The subsequent Logical Units of the block are numbered in sequence. The FORMAT command displays the input block structure as follows:

```
INPUT BLOCK STRUCTURE:
NUMBER OF INPUT LOGICAL UNITS = 00<i + 1>

LSB
----------------  ------
| 00 | 01 | 02 |    | 0i|
----------------  ------
```

Output Specification

The Output Specification is used to specify the sequence in which the Logical Units of each input block are to be positioned to make up each output block. Output Logical Units can also be filled with all zeros or all ones by using the symbols F or T, respectively. Any unspecified logical units at the end of each output block are filled with the blank state of the current PROM type. If no PROM type has been specified using the TYPE command, the unspecified locations are filled with ones.

The Output Specification also specifies the ISIS-II file where the output data is to be stored. The syntax of the Output Specification is as follows:

$$\left\{\begin{matrix}\text{<ILU>}\\ \text{F}\\ \text{T}\end{matrix}\right\}, \left\{\begin{matrix}\text{<ILU>}\\ \text{F}\\ \text{T}\end{matrix}\right\}, \left\{\begin{matrix}\text{<ILU>}\\ \text{F}\\ \text{T}\end{matrix}\right\}, \ldots, \left\{\begin{matrix}\text{<ILU>}\\ \text{F}\\ \text{T}\end{matrix}\right\} \quad \text{TO <file>}$$

Where <ILU> is any of the Input Logical Units shown in the Input Block Structure, and the total number of logical units or constants specified must not exceed the number of logical units specified for the output block. The Input Logical Units are referred to by the number assigned to them in the Input Block Structure. If the Input Logical Units are entered in a contiguous sequence of the Input Block Structure, the Output Specification can also be:

<ILU1>/<ILUn> TO <file>

This notation can also be used to denote a reversal of the order of the logical units: <ILUn>/<ILU1>

The following Output Specification example assumes an Input Block Structure with four Logical Units:

```
INPUT BLOCK STRUCTURE:
NUMBER OF INPUT LOGICAL UNITS = 004

LSB
--------------------
| 00 | 01 | 02 | 03 |
--------------------

NUMBER OF OUTPUT LOGICAL UNITS = 008
OUTPUT SPECIFICATION (CR TO EXIT):
*0,F,1,F,2,F,3,F  TO  :F1:ALTZERO.BYT
```

This Output Specification would thus expand each four byte input block into an eight byte output block with every alternate byte set to zero. The resultant output data would be stored in file ALTZERO.BYT on disk drive 1.

INTERACTIVE FORMAT EXAMPLES

The following examples illustrate some of the interactive data manipulation operations that the FORMAT command can perform.

Example: Nibble Swapping

This example shows how the FORMAT command can be used to swap all nibbles (half-bytes) of the first 1000H bytes of the file NIBBLE.OLD. The data is stored in the new file NIBBLE.SWP. Each low-order nibble of file NIBBLE.OLD becomes the high-order nibble of the corresponding byte in NIBBLE.SWP. Each high-order nibble of file NIBBLE.OLD becomes the low-order nibble of the corresponding byte in NIBBLE.SWP.

```
PPS>FORMAT NIBBLE.OLD (0,FFFH)
LOGICAL UNIT (BIT=1,NIBBLE=2,BYTE=3,N-BYTE=4)
LU = 2<CR>
INPUT BLOCK SIZE (N BYTES)
N = 1<CR>
OUTPUT BLOCK SIZE (N BYTES)
N = 1<CR>
INPUT BLOCK STRUCTURE:
NUMBER OF INPUT LOGICAL UNITS = 002

LSB
-----------
| 00 | 01 |
-----------
```

```
NUMBER OF OUTPUT LOGICAL UNITS = 002
OUTPUT SPECIFICATION (CR TO EXIT):
*1,0
TO? NIBBLE.SWP
OUTPUT STORED
*<CR>
PPS>
```

Note that "1,0" is the output specification that selects the nibble swap. Also, note that since no file was specified in the output specification, iPPS prompted for it. The command "COPY NIBBLE.SWP TO PROM" could now be used, for example, to program a PROM with the new data.

Example: Two-Way Interleaving

This example shows how the FORMAT command can be used to perform a two-way interleaving of data. The two output arrays are stored in output files LOWER.BYT and UPPER.BYT. This command might be used, for example, to format a series of double-byte addresses (in the Intel standard low/high form from file DOUBLE.BYT) so that alternate bytes could be programmed as consecutive bytes in two separate 8-bit wide PROMs. One PROM would be programmed with the bytes from LOWER.BYT and another PROM from UPPER.BYT. The PROMs might then serve in parallel to provide 16-bit wide address data words to a 16-bit microprocessor.

```
PPS>FORMAT DOUBLE.BYT (0,FFFH)
LOGICAL UNIT (BIT=1,NIBBLE=2,BYTE=3,N-BYTE=4)
LU = 3
INPUT BLOCK SIZE (N BYTES)
N = 2
OUTPUT BLOCK SIZE (N BYTES)
N = 1
INPUT BLOCK STRUCTURE:
NUMBER OF INPUT LOGICAL UNITS = 002

LSB
-----------
| 00 | 01 |
-----------

NUMBER OF OUTPUT LOGICAL UNITS = 001
OUTPUT SPECIFICATION (CR TO EXIT):
*0 TO :F1:LOWER.BYT
OUTPUT STORED
*1 TO :F1:UPPER.BYT
OUTPUT STORED
*<CR>
PPS>
```

Example: Bit Reversal

This example shows how the FORMAT command can be used to perform bit reversal on each byte in the specified 1000H byte array of the source

device (the PROM device in this example). The output data is stored in file REVERS.BIT.

```
PPS>F PROM(0,FFFH)
LOGICAL UNIT (BIT=1,NIBBLE=2,BYTE=3,N-BYTE=4)
LU = 1
INPUT BLOCK SIZE (N BYTES)
N = 1
OUTPUT BLOCK SIZE (N BYTES)
N = 1
INPUT BLOCK STRUCTURE:
NUMBER OF INPUT LOGICAL UNITS = 008

LSB
-----------------------------------------
| 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 |
-----------------------------------------

NUMBER OF OUTPUT LOGICAL UNITS = 008
OUTPUT SPECIFICATION (CR TO EXIT):
*7/0 TO REVERS.BIT
OUTPUT STORED
*<CR>
PPS>
```

The "7/0" specification causes each byte in the PROM to be stored in the REVERS.BIT file in reverse order, ie. the most significant bit in the PROM byte becomes the least significant bit in the file byte. The slash "/" indicates that the Input Logic Units are to be arranged in the output block in a contiguous sequence: 7, 6, 5, 4, 3, 2, 1, 0.

Example: Block Move

This example shows how the FORMAT command can be used to move blocks of data. In this example, the block of bytes located from Buffer location 0 to 0FFFH are moved to locations starting from 1000H to 1FFFH. Also the block of bytes located from locaton 1000H to 1FFFH are moved to locations from 0 to 0FFFH. The data is not actually moved in the Buffer. Instead, it is stored in the file BLKRVR.MOV.

```
PPS>FORMAT BUFFER(0,FFFH)
LOGICAL UNIT (BIT=1,NIBBLE=2,BYTE=3,N-BYTE=4)
LU = 4
N = 1000H
INPUT BLOCK SIZE (N BYTES)
N = 2000H
OUTPUT BLOCK SIZE (N BYTES)
N = 2000H
INPUT BLOCK STRUCTURE:
NUMBER OF INPUT LOGICAL UNITS = 002

LSB
-----------
| 00 | 01 |
-----------
```

```
NUMBER OF OUTPUT LOGICAL UNITS = 002
OUTPUT SPECIFICATION (CR TO EXIT):
*1,0 TO :F1:BLKRVR.MOV
OUTPUT STORED
*<CR>
PPS>
```

The command "COPY :F1:BLKRVR.MOV TO BUFFER" could next be used to load the two interchanged blocks of data back into the Buffer. The old blocks of data in the Buffer (from address 0 to 1FFFH) would thus be overwritten by the new interchanged blocks.

HELP

Selectively Displays
Help Information

SYNTAX

HELP [<command keyword>]

OPERATION

The HELP command displays reference information describing the iPPS commands.

If HELP is entered without the optional <command keyword>, a summary of iPPS commands is displayed on the console. If the optional <command keyword> is entered, detailed information on the syntax and operation of that command is displayed.

The help information is displayed one screen at a time. At the end of each screen, the display pauses with the following prompt:

ENTER <CR> TO CONTINUE -

To continue displaying the help file, simply type the RETURN key. To abort the present HELP command and return to the iPPS command interpreter, type the <ESC> key.

The file IPPS.HLP must be available on the same device as the iPPS command file or an error occurs when the HELP command is attempted.

EXAMPLES

In the following example, information about the HELP command itself is displayed.

```
PPS>HELP HELP

HELP

    {H}ELP [<command keyword>]

The HELP command displays reference information describing the iPPS
commands.  If HELP is entered without the optional <command keyword>,
a summary of the iPPS commands is displayed.  If the optional
<command keyword> is entered, detailed information on the syntax
and operation of that command is displayed.

PPS>
```

In the following example, information about the DISPLAY command is displayed. In the syntax diagrams, a character enclosed in braces is the single character required to uniquely identify a keyword to iPPS. The vertical bars enclose a set of values of which one must be chosen.

PPS>H D

DISPLAY

```
                                                                  ¦Y¦
                ¦{P}ROM  ¦                                        ¦O¦
    {D}ISPLAY   ¦        ¦  [(<startaddr>[,<endaddr>])]  [F]  [¦Q¦]
                ¦{B}UFFER¦                                       ¦T¦
                                                                 ¦H¦
                         OR
                                                                      ¦Y¦
                                                          ¦ 80¦       ¦O¦
    {D}ISPLAY  <file>  [(<startaddr>[,<endaddr>])]  [F]  [¦ 86¦] [¦Q¦] [P]
                                                          ¦286¦       ¦T¦
                                                                      ¦H¦
```

The DISPLAY command outputs data from the PROM, Buffer, or File devices in a formatted display on the console.

Enter <CR> to continue - <CR>

The data is displayed one page at a time. At the end of a page, a pause between succeeding pages is indicated by the message:

ENTER <CR> TO CONTINUE

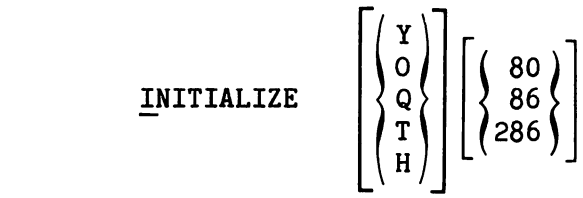
To continue the display of data, simply type the RETURN key. The format of the data displayed varies with the number base currently in effect (or specified for the duration of this command).

F	Switch which complements data before it is written.
80,86,286	Switch which selects the absolute file format.
H,T,O,Q,B	Switch which selects the display number base.
P	Switch which allows reading of a patched file.

INITIALIZE

Sets Default Number Base
and Default File Format

SYNTAX



OPERATION

The INITIALIZE command allows initialization (or subsequent changing) of the default number base and the default file format to be used by iPPS.

The default number base determines the base in which data and address values are displayed on output or interpreted on entry, unless it is overridden by an explicit base mode value. An explicit base mode only applies to the value to which it is appended. The base specified on the command line is used as the default for all values displayed or entered until a new base is selected. Upon initialization, the number base default is hexadecimal.

The default file format determines the default file format that iPPS uses in commands that read files. See Appendix C for more details on the file formats that iPPS can read. iPPS always writes files in the 286 format.

{ Y | O | Q | T | H }

specifies the default base to be used. Only one of these can be specified at once. It specifies the default number base to be used for iPPS. Binary is specified by the letter Y; octal by O or Q; decimal by T; and Hexadecimal by H.

{ 80 | 86 | 286 }

specifies the default file format to be used by iPPS when executing commands that read data from files.

EXAMPLES

In the following example, the default number base is changed to octal.

```
PPS>INITIALIZE Q
```

The next example changes the default number base to decimal and the default file format to 8080 format.

```
PPS>I T 80
```

In the next example, only the default file format is changed (to 8086 format).

```
PPS>I 86
```

If the file type or base are not specified, iPPS prompts for them:

```
PPS>I
<FILE TYPE and/or BASE> = 80 Y
PPS>
```

The file type is changed to 8080 and the base to binary.

LOADDATA

Fills Section of
Buffer with Constant

SYNTAX

```
LOADDATA BUFFER [(<startaddr>[,<endaddr>])]
         WITH [<byteval>] [F]
```

OPERATION

The LOADDATA command fills all or part of the iPPS Buffer with a specified constant.

<startaddr> specifies the start address of the section of data in the Buffer to be filled with a constant. If <startaddr> is not entered then it defaults to the Buffer start address. Its omission inherently implies that the <endaddr> default is also assumed.

<endaddr> specifies the end address for the section of Buffer to be filled with a constant. The <endaddr> parameter can be optionally specified as a length, L<length>. The <endaddr> is then <startaddr> + <length> - 1. If <endaddr> is not specified then it defaults to an address determined by the expression, (Buffer start address) + (Buffer size - 1). If the address range specified is greater than the size of the current PROM type then an error message is displayed.

<byteval> specifies a constant to be loaded into each byte of the selected address range.

[F] is the optional data switch which complements <byteval> before it is loaded into the Buffer. If no data switch is specified, <byteval> is loaded into the Buffer section in its uncomplemented form.

EXAMPLES

In the following example, the buffer is loaded with the constant FFH.

```
PPS>LOADDATA BUFFER WITH FF
```

The next example loads the first 256 bytes of Buffer addresses with the constant value of 1.

```
PPS>L BUFF(0,FF) W 1
```

In the next example, the Buffer is loaded with FE because the constant value is complemented by the F switch.

PPS>L B W 1 F

MAP

Displays iPPS Status with
File and Buffer Structure

SYNTAX

MAP [<file>] [{ 80 / 86 / 286 }] [P]

OPERATION

The MAP command displays the status of various iPPS working parameters and displays the buffer boundaries and file structure.

The currently selected PROM type, the current default number base, the current default file format, and the current workfiles drive are also displayed.

When data is read into the Buffer from a file, a memory map of the File structure is maintained. It records starting and ending addresses of distinct data blocks in the file. This is useful to see what sections of the file contain or don't contain data. The MAP command can be used to display this file structure.

<file> specifies the normal ISIS-II file name for a source file. If <file> is not specified, the file structure of the most recently read file (i.e., read by any previous iPPS command) is displayed.

{ 80 / 86 / 286 } is the optional file format switch selection. If this switch is not specified then the currently selected default file format is assumed when reading in the source file.

[P] is the optional patch switch which allows the reading of patched object files (i.e., files with overlapped sections of data). If this switch is not specified and a patched file is read, the following error message will be displayed and the MAP command will be aborted.

-GENERAL ERROR--DATA OVERLAP IN THE FILE.

EXAMPLES

The following MAP command example assumes the file :F1:CODE2.IUP had been created with the following copy command:

COPY PROM(0,3FF) TO :F1:CODE2.IUP(400)

The following MAP command is then used to display the structure of the file in addition to other iPPS status information. S.A. and F.A. stand for the start address and final address of a data block, respectively.

```
PPS>MAP :F1:CODE2.IUP

FILE STRUCTURE:

 S.A. = 000400      F.A. = 0007FF

STATUS:

BUFFER S.A. = 000000   BUFFER F.A. = 0007FF
DEFAULT BASE = HEX
DEFAULT FILE TYPE = 286
WORKFILE :F1:
PROM TYPE = 2716
PPS>
```

OVERLAY

Checks Buffer Data Against
Pre-programmed Bits in PROM

SYNTAX

OVERLAY BUFFER [(<startaddr>[,<endaddr>])]

TO PROM [(<destaddr>)] [F]

OPERATION

The OVERLAY command tests to determine if a PROM can be programmed even though it is not completely blank. That is, it checks to see if a particular set of Buffer data can be overlayed on the existing data of the PROM without conflict.

<startaddr> specifies the start address of the section of data in the Buffer to be overlayed on the data in the PROM. If <startaddr> is not entered then it defaults to the Buffer start address. Its omission inherently implies that the <endaddr> default is also assumed.

<endaddr> specifies the end address for the section of Buffer to be overlayed on the data in the PROM. The <endaddr> parameter can be optionally specified as a length, L<length>. The <endaddr> is then <startaddr> + <length>-1. If <endaddr> is not specified then it defaults to an address determined by the expression, (Buffer start address) + (Buffer size -1). If the address range specified is greater than the size of the current PROM type then an error message is displayed.

<destaddr> specifies the desired start address for the comparison data in the PROM device. This parameter can be optionally specified as an offset, O<offset>, relative to <startaddr>. The <destaddr> is then <startaddr> + <offset>. The relative <offset> can be either positive or negative as indicated by a + sign or a - sign. If the relative offset results in a destination starting address less than zero or greater than OFFFFFFH (16777215T), an error message is displayed. If <destaddr> exceeds (PROM size - 1), an error message occurs. If <destaddr> is not specified it defaults to 0.

[F] is the optional data switch which complements the Buffer data before comparison with the PROM data. If no data switch is specified, data is compared in its uncomplemented form.

In effect, the OVERLAY command checks the PROM for stuck bits. If stuck bits are not found, it displays the message

OVERLAY TEST PASSED

and returns to the main iPPS prompt. Otherwise, it displays the message

OVERLAY ERROR--DISPLAY Y/N?

To see the mismatched data, press Y; otherwise, press N.

EXAMPLES

In the following example, the OVERLAY command detects that the PROM cannot be programmed with the data in the buffer. The mismatched data is displayed because the response Y is given to the prompt.

```
PPS>OVERLAY BUFFER TO PROM
OVERLAY ERROR--DISPLAY Y/N? Y
  PROM ADDRESS              PROM DATA  EXPECTED DATA
00023C                       FE         01
000312                       EF         81
-COMMAND ERROR--OVERLAY TEST FAILED.
PPS>
```

In the next example, the overlay test is applied to a specified range of buffer addresses and successfully passes for the specified range.

```
PPS>OV BUFF(0,23B)
TO? P
OVERLAY TEST PASSED.
PPS>
```

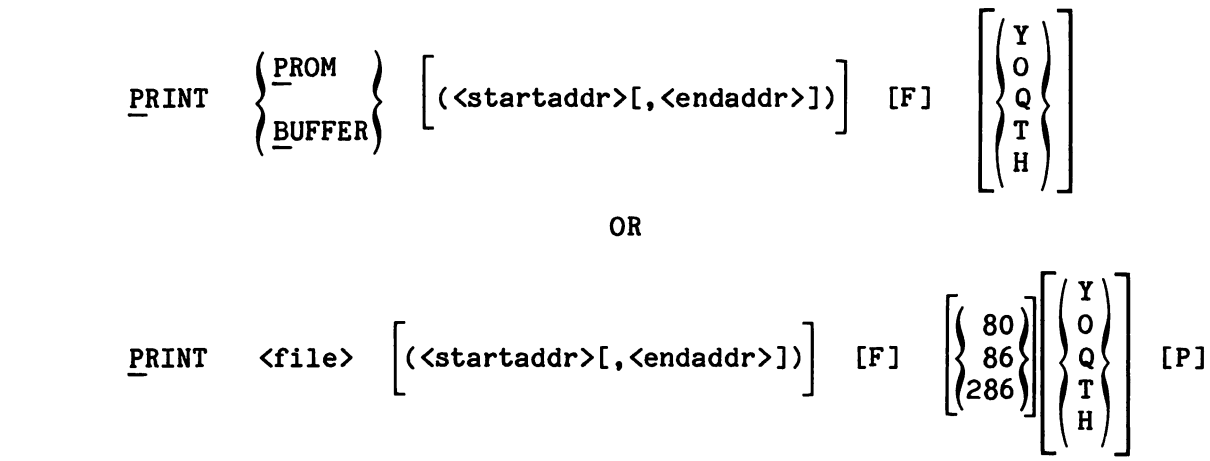
In the next simple example, the entire Buffer and PROM device are tested. The abbreviated OVERLAY keyword is all that is required to do this.

```
PPS>O
OVERLAY TEST PASSED.
PPS>
```

PRINT

Prints Data on
Development System Printer

SYNTAX



OPERATION

The PRINT command outputs data from the PROM, Buffer, or File devices in a formatted print-out on the development system printer. As shown above, the PRINT <file> command has the additional switch options to specify the file format being read and whether or not the source file is patched.

<file> specifies the normal ISIS-II file name for a source file.

<startaddr> specifies the start address of the section of data in the source device to be printed on the line printer. Together with the <endaddr>, this parameter specifies the range of addresses to be printed. The default for <startaddr> depends on the source of the data. If the source is PROM, <startaddr> defaults to 0. If the source is BUFFER, its default is the present Buffer start address. If the source is a file, its default is the lowest address encountered while reading in the file. The omission of <startaddr> inherently implies that the <endaddr> default is also assumed.

<endaddr> specifies the end address for the section of source data to be printed. This parameter can be optionally specified as a length, L<length>. The <endaddr> is then <startaddr> + <length>-1. The default for <endaddr> depends on the source of the data. If the source is PROM, <endaddr> defaults to (PROM size -1). PROM size is determined by the most recent TYPE command. If the source is BUFFER, <endaddr> defaults to an address determined by the expression, (Buffer start address) + (Buffer size -1). If the source is a file, <endaddr> defaults to the highest address encountered while reading in the file.

[F] is the optional data switch which complements the data before it is printed. If no data switch is specified, the data is printed in its uncomplemented (true) form.

{Y, O, Q, T, H} is the optional base switch. Only one of these can be specified at once. It specifies the number base to be used for the printed data during the execution of the command. Binary is specified by the letter Y; octal by O or Q; decimal by T; and Hexadecimal by H. The default for this switch is the current default number base determined by the most recent INITIALIZE command.

{80, 86, 286} is the optional file format switch selection. If this switch is not specified then the currently selected default file format is assumed when reading the file for printing.

[P] is the optional patch switch which allows the reading of patched object files (i.e., files with overlapped sections of data). If this switch is not specified and a patched file is read, the following error message will be displayed and the PRINT command will be aborted.

-GENERAL ERROR--DATA OVERLAP IN THE FILE.

When a particular address range is specified to be read from a file, all sections in the address range that are <u>not</u> present in the file are printed using the blank state of the currently selected PROM type (all 1s if no PROM type has been selected).

The format of the data printed varies with the number base currently in effect (or specified for the duration of this command). To the right of the actual data is the ASCII character equivalent for each printed byte value. If there is no meaningful ASCII character equivalent, a . character is printed as a place holder.

EXAMPLES

The syntax and output format of the PRINT command is the same as that of the DISPLAY command. See the DISPLAY command description for examples of the format of the printout for all the following PRINT command examples.

The following example prints the contents of the PROM device in hexadecimal (the default number base).

```
PPS>PRINT PROM
```

The next example prints in octal the contents of the specified range of addresses in the PROM device. The range is specified in hexadecimal.

```
PPS>PRINT  PROM(0,L18H) Q
```

In the next example, the contents of the specified range of buffer addresses are printed in binary. The F range specification is interpreted as hexadecimal since it has no explicit number base tail and the current default number base is assumed to be hexadecimal.

```
PPS>PRI BUFF(0,F) Y
```

In the next example, the contents of a range of buffer addresses are printed in complemented form in binary. The range is specified in binary as a length from the start address.

```
PPS>P B(0,L110Y) F Y
```

This next example prints in decimal the contents of a range of buffer addresses. The range is specified in hexadecimal.

```
PPS>P B(0,1AH) T
```

REPEAT

Repeats Full Execution
of Previous Command

SYNTAX

REPEAT

OPERATION

The REPEAT command executes the most recently entered command again without requiring the entire command and its parameters to be re-entered. All parameters retain the value they had in the most recently entered command. The most recently entered command is displayed before it is executed.

EXAMPLES

In the following example, the REPEAT command is used to program several PROMs in succession without retyping the original copy command. Each time the REPEAT command is used it re-displays the command it is repeating. After each PROM has been programmed, the PROM just programmed is replaced by a new blank PROM to be programmed.

```
PPS>COPY BUFFER TO PROM
 CHECK SUM = 09AB
PPS>REPEAT
COPY BUFFER TO PROM
 CHECK SUM = 09AB
PPS>REP
COPY BUFFER TO PROM
 CHECK SUM = 09AB
PPS>r
COPY BUFFER TO PROM
 CHECK SUM = 09AB
PPS>
```

SUBSTITUTE

Examines and Modifies
Buffer Data

SYNTAX

SUBSTITUTE <addr> [F] [{ Y / O / Q / T / H }]

OPERATION

The SUBSTITUTE command allows the user to interactively examine and modify specific locations in the development system Buffer.

<addr> specifies the starting address at which data is to be displayed and/or substituted.

[F] is the optional data switch which complements the data before it is displayed. If no data switch is specified, the data is displayed in its uncomplemented (true) form.

{ Y / O / Q / T / H } is the optional base switch. Only one of these can be specified at once. It specifies the number base to be used for the printed data during the execution of the command. Binary is specified by the letter Y; octal by O or Q; decimal by T; and Hexadecimal by H. The default for this switch is the current default number base determined by the most recent INITIALIZE command.

When the SUBSTITUTE command is invoked, it displays the contents of the buffer starting at the address specified in the command. For example,

```
S 0
000000: 2A A0 2B 36 80 2A A6 2B 36 01 2A AC 2B 36 00 2A
```

The underline represents the position of the cursor.

The user can then move the cursor and change the contents of the data pointed to by the cursor with the following keys:

<SPACE BAR> moves the cursor one character to the right skipping spaces between bytes. If the display is in binary, 8 spaces would be required to move to the next byte displayed. No locations are modified. The display is automatically scrolled to the next line when the cursor is moved beyond the end of the line.

< moves the cursor one character at a time to the left. The cursor cannot be moved back to a previous line.

> moves the cursor one character at a time to the right. The display automatically scrolls to the next line when the cursor is moved beyond the end of the line.

<CR>
or
<ESC> terminates the SUBSTITUTE command and returns to the main iPPS prompt. Any edited Buffer changes become permanent after pressing <CR> or <ESC>.

Typing a valid numeric value (determined by the current number base) changes the contents of the byte at the current cursor position. An invalid key entry results in a warning beep at the console terminal.

EXAMPLES

In the following example, Buffer data is interactively modified starting at address 400H. Specifically, the byte at address 402H is changed from 49H to 4CH. (Note that since the address was not specified, iPPS prompted for it.)

```
PPS>S<CR>
ADDRESS? 400<CR>
000400: 04 C3 49 05 3E 00 D3 E2 3E 16 D3 E2 CD B7 05 3E
```

Next, using the > key, the cursor is advanced to character "9" of the byte "49" (at address 402H).

```
000400: 04 C3 49 05 3E 00 D3 E2 3E 16 D3 E2 CD B7 05 3E
```

The new value of C is simply typed over the old value of 9 and the changed is written after pressing the <CR> key. Control is returned to the main iPPS command input.

```
000400: 04 C3 4C 05 3E 00 D3 E2 3E 16 D3 E2 CD B7 05 3E
PPS><CR>
```

TYPE

Selects PROM Type

SYNTAX

TYPE [<PROM device number>]

OPERATION

The TYPE command is required prior to executing any command which interfaces with the memory device in the PROM programmer. It specifies the type of device which is to be programmed.

If the command is entered without the argument, the user is prompted to enter one of the allowable PROM types for the Personality Module currently installed. If the PROM type entered is not supported by the currently installed module or if no module is currently installed, an error message is displayed.

If a PROM type is selected that requires more Buffer space than 8K, a virtual buffer is automatically built and maintained on the currently selected workfiles drive. If no prior workfiles drive has been established (see the WORKFILES command), a workfiles drive is requested by the following prompt:

WORKFILES (:FX:)? ; X =

At this point, a value for X may be entered. If no value for X is entered, and a <CR> or <ESC> is entered, the workfiles drive is made the drive from which the iPPS software was originally obtained. The MAP command can be used to determine what drive is currently the workfiles drive. The WORKFILES command can be used to change the current workfiles drive.

EXAMPLES

In the following example, the TYPE command is entered without specifying a PROM type on the command line. iPPS displays the valid types for the specific Personality Module installed in the Universal

Programmer and prompts for one of these types. In this example, the PROM type selected is 2758.

```
PPS>TYPE
 PROM TYPES:
2716
2732
2732A
2758
2758S
2764
27128
2815
2816
ENTER ONE PROM TYPE - 2758
```

In the next example, the TYPE is specified on the command line. The single character abbreviation for the TYPE keyword is used. The type selected (27128) requires a virtual buffer and the command prompts for the workfiles drive.

```
PPS>T 27128
WORKFILES (:F1:)? ; X = 1
PPS>
```

VERIFY

Verifies PROM Data
with Buffer Data

SYNTAX

```
VERIFY [ BUFFER [(<startaddr>[,<endaddr>])]
       [ TO PROM [(<destaddr>)] [F] ]
```

OPERATION

The VERIFY command compares the data in the PROM device with the data in the Buffer. Any discrepancies between the two can then be displayed. The Buffer contents are displayed first followed by the PROM contents.

If VERIFY is entered without any arguments, the entire PROM is compared starting at the Buffer start address.

The TYPE command must be executed prior to this command; otherwise, an error message is displayed. An error message is also displayed if the address range exceeds the size of the PROM device.

<startaddr> specifies the start address of the section of data in the Buffer to be verified with the data in the PROM. If <startaddr> is not entered then it defaults to the Buffer start address. Its omission inherently implies that the <endaddr> default is also assumed.

<endaddr> specifies the end address for the section of Buffer to be verified. The <endaddr> parameter can be optionally specified as a length, L<length>. The <endaddr> is then <startaddr> + <length>-1. If <endaddr> is not specified then it defaults to an address determined by the expression, (Buffer start address) + (Buffer size -1). If the address range specified is greater than the size of the current PROM type then an error message is displayed.

<destaddr> specifies the desired start address for the comparison data in the PROM device. This parameter can be optionally specified as an offset, O<offset>, relative to <startaddr>. The <destaddr> is then <startaddr> + <offset>. The relative <offset> can be either positive or negative as indicated by a + sign or a - sign. If the relative offset results in a destination starting address less than zero or greater than OFFFFFFH (16777215T), an error message is displayed. If <destaddr> exceeds (PROM size - 1), an error message occurs. If <destaddr> is not specified, it defaults to 0.

[F] is the optional data switch which complements the Buffer data before comparison with the PROM data. If no data switch is specified, data is compared in its uncomplemented form.

EXAMPLES

In the following example, the VERIFY command detects a discrepancy between data in the Buffer and data in the PROM. The mismatched data is diplayed because the response Y is given to the prompt. If the N response had been given, iPPS would return to its main command input prompt.

```
PPS>VERIFY
VERIFY ERROR--DISPLAY Y/N? Y
  PROM ADDRESS              PROM DATA  EXPECTED DATA
000400                        04         84
000402                        49         C9
000403                        05         85
-COMMAND ERROR--VERIFY TEST FAILED
```

In the next example, the verification test is applied to a specified range of the Buffer and successfully passes.

```
PPS>V B(0,3FF)
TO? P
VERIFY TEST PASSED.
PPS>
```

WORKFILES

Specifies Drive for
Temporary Workfiles

SYNTAX

WORKFILES :F<x>:

OPERATION

The WORKFILES command allows the user to specify the storage device for temporary files IPPS.BUF and IPPS.TMP that iPPS creates. These temporary workfiles are used in programming PROMs larger than 8K bytes.

All temporary files are deleted upon execution of the EXIT command. If the user exits iPPS without using the EXIT command, these files may remain on the disk.

If the WORKFILES command has not been given before a temporary file is needed (by certain iPPS commands), the software prompts for the workfiles drive number:

WORKFILES (:FX:)? ; X =

At this point, a value for X may be entered. If no value for X is entered, and a <CR> or <ESC> is entered instead, the workfiles drive is made the drive from which the iPPS software was originally obtained. The MAP command can be used to determine what drive is currently the workfiles drive. The WORKFILES command is used to change the current workfiles drive.

EXAMPLES

The following example command selects drive 1 for temporary iPPS workfiles.

```
PPS>WORKFILES :F1:
PPS>
```

In this example, no drive is specified after the command keyword, the command will prompt for the workfiles drive number. The single character abbreviation for WORKFILES is used. Drive 0 is selected for all temporary iPPS workfiles.

```
PPS>W
WORKFILES (:FX:)? ; X = 0
PPS>
```

CHAPTER 4
OFF-LINE OPERATION

This chapter describes the off-line operation of the Universal Programmer. Off-line operation applies only to the iUP-201 model of the Universal Programmer. The chapter first presents an overview of off-line operation and then describes in detail the iUP-201 function keys.

General Off-Line Functioning

The iUP-201 is equipped with a keyboard and an alphanumeric display for off-line operation. Internally, the iUP-201 contains 16K of RAM, expandable to 32K. This RAM buffer constitutes the URAM logical device that the iPPS software communicates with during on-line operation. Thus, during on-line operation, the RAM can be loaded with data that can then be manipulated and programmed into a PROM device off-line.

In addition, the iUP-201 local firmware provides an off-line load function that allows the Universal Programmer to download data in Intel 8080 hexadecimal format into the RAM buffer from any host system via a RS-232 interface. This function does not use the iPPS software.

During off-line operation only two storage devices are accessible: PROM and RAM. Off-line operations with these devices often consist of copying the data from a master PROM into the off-line RAM device, then programming the data back into a blank PROM. For example, the following off-line steps illustrate how copies can be made of a master PROM:

1. Switch the iUP-201 to off-line mode.
2. Insert the PROM to be copied.
3. Select the appropriate PROM type if required.
4. Press the ROM TO RAM switch to copy the PROM data into the RAM.
5. Remove the PROM that was copied.
6. Insert a blank PROM of the proper type.
7. Press the PROG switch to program the RAM data into the new PROM.

Additional copies of the original PROM can be made by repeating steps 6 and 7.

The iUP-201 also allows data in the RAM to be modified on a byte by byte basis or a selected address range to be filled with a constant.

The off-line operation of the iUP-201 requires all data transfers between the PROM device and the RAM to include the whole range of device addresses. Partial programming of a PROM can be accomplished in the off-line mode by filling the addresses in RAM that do not contain valid data with a constant equal to the blank state of the PROM.

Internal Memory

The 16K (expandable to 32K) of Universal Programmer memory is referred to as RAM in this Chapter (It is referred to as URAM when regarded as a logical device during on-line operations--see Chapter 3). This memory is used as a buffer to hold data that is to be manipulated off-line and programmed into the PROM.

The selected PROM type determines the address range of the RAM. Addresses are entered at the Universal Programmer hexadecimal keypad whenever access to RAM is required during off-line operations.

The expansion of the RAM to 32K is only necessary when the size of a specific PROM device requires more than 16K. The user can easily expand the iUP-201 RAM to 32K, with no service assistance, using an expansion kit available from Intel. See the Intel System Data Catalog for more information about which Personality Modules require the RAM expansion for the iUP-201. The address of the Intel Literature Department is on the back of this manual's title page.

Keyboard/Display Overview

The keyboard illustrated in Figure 4-1 consists a set of function keys plus a 16-key numeric keypad for entering addresses and data in hexadecimal.

The alphanumeric display, a 24-character LED-type display, is divided into three fields: the command field, the address field, and the data field. See figure 4-1.

The function keys select the operation desired. Some operations require additional parameters such as addresses and data to be specified. These parameters are entered at the 16-key hexadecimal keypad.

Function Key Descriptions

The remainder of this chapter is dedicated to complete descriptions of each off-line function key.

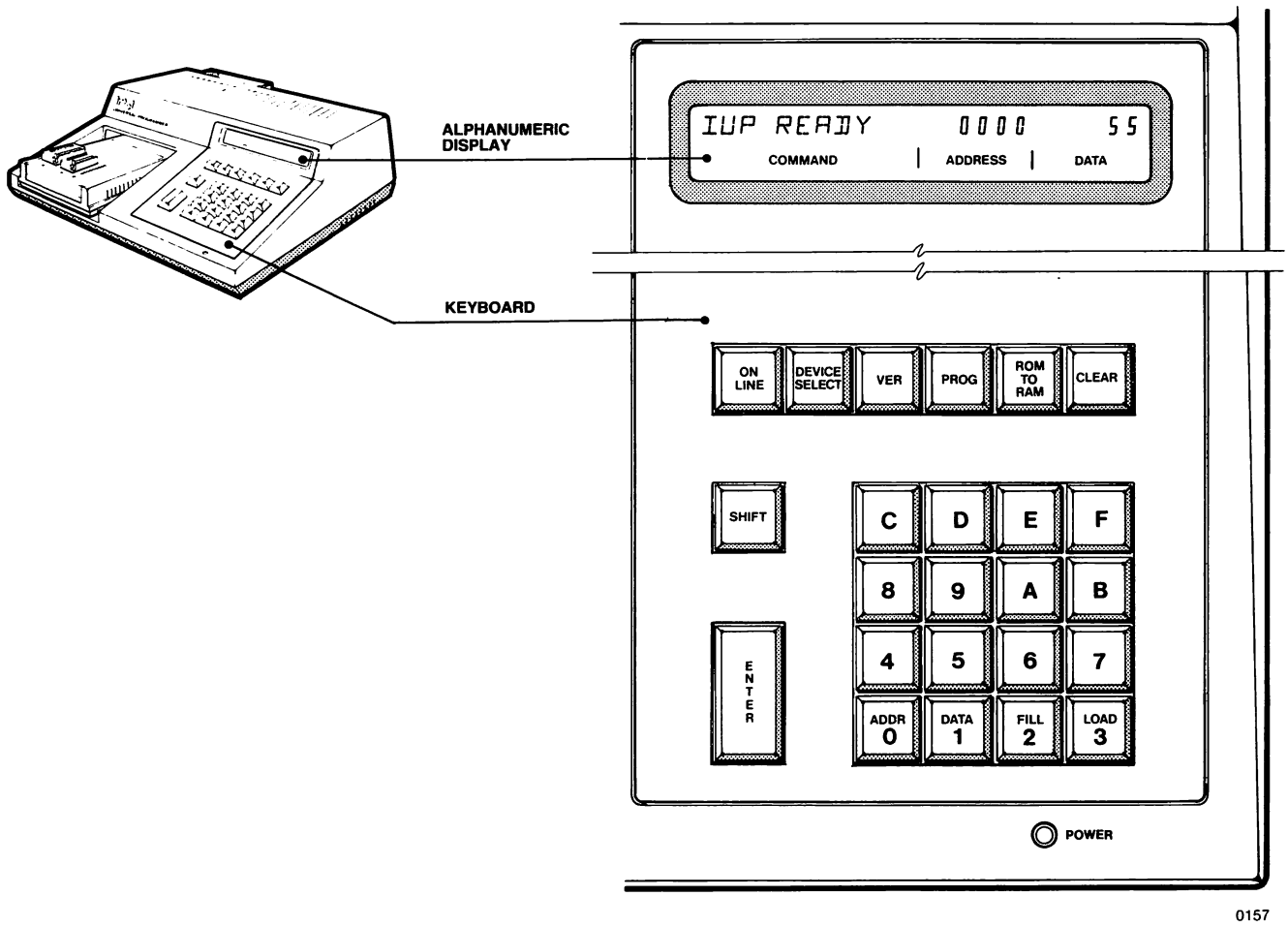


Figure 4-1. iUP-201 Keyboard and Display

Selects on-line or
off-line operation

FUNCTION

The ON LINE key selects either the on-line or the off-line mode of operation for the Universal Programmer (each time the ON LINE key is pressed, the Universal Programmer switches from one mode to the other). After pressing the ON LINE key to enter the on-line mode, the message

ON LINE

is displayed in the command field of the iUP-201 alphanumeric display. When in the on-line mode of operation, all iUP-201 keys are disabled (with the exception of the ON LINE key itself).

Pressing the ON LINE key again switches the Universal Programmer to the off-line mode and causes the following message to be displayed:

IUP READY 0000 55

NOTE

The remaining keys described in this chapter are only operative when the Universal Programmer is in the off-line mode.

Loads PROM Data
into RAM

FUNCTION

The ROM TO RAM key loads the RAM with the data from the PROM device that has been installed in the Personality Module IC socket. While loading is in progress, the message

LOADING

is displayed in the Command field of the alphanumeric display. When loading is complete, the following message is displayed:

END LOAD CKSUM = dddd

Where dddd is the hexadecimal value of the check sum obtained. The checksum is the 2's complement of the 16-bit sum of all the bytes loaded into the RAM.

If the PROM device is blank, the following message is displayed after loading is complete:

PROM BLANK CKSUM = dddd

The ROM TO RAM key thus provides a convenient off-line way to determine if a PROM device is blank.

Verifies PROM against
RAM Data

FUNCTION

The VER key compares the PROM device data against the RAM data. While verification is in progress, the message

VERIFY

is displayed in the Command field of the alphanumeric display. If all locations compare, the "VERIFY" message in the Command field is blanked out, and the following message is displayed at the end of the verification:

END VERIFY CKSUM = dddd

where dddd is the hexadecimal value of the check sum obtained. The checksum is the 2's complement of the 16-bit sum of all the bytes verified.

If a mismatch occurs, the message

VERIFY ERR @ aaaa dd

is displayed.

The Address field contains the address (aaaa) at which the mismatch occured, and the Data field (dd) contains the Exclusive-OR of the expected and actual data.

After such a mismatch, pressing the VER key again continues the verification starting at the next address after the mismatch address. A subsequent mismatch causes the same behavior as just described. The CLEAR key will abort the verify function if a mismatch occurred.

PROG

Programs PROM
with RAM Data

FUNCTION

The PROG key programs the PROM device after first ensuring that the PROM is blank. If the PROM device passes the blank PROM check, programming begins. If the PROM fails the blank PROM check, the message

BLANKCK ERR aaaa dd

is displayed. The address at which the error was detected is displayed in the Address (aaaa) field. The hex value of the Exclusive OR of the expected blank state and the actual value found at that address of the PROM device is displayed in the Data (dd) field.

If the PROM is already programmed, the PROG key can be pressed again to perform a stuck bit check. The stuck bit check compares all PROM device bits that are preprogrammed with corresponding bits in the RAM. If all the preprogrammed (stuck) bits in the PROM are the same as their corresponding bits in the RAM, the Universal Programmer begins programming the PROM. Otherwise, the message

DEVICE WILL NOT PROGRAM

is displayed. If this occurs, press the CLEAR key and try another PROM device.

Whenever a blank PROM check is in progress, the following message is displayed:

BLANKCK

Whenever the stuck bit check is in progress, the following message is displayed:

STUCKBIT CHK

The BLANKCK and STUCKBIT checks usually occur too fast to be noticed.

Whenever programming is in progress, the message

PROGRAMMING

blinks on the alphanumeric display.

If an error occurs during programming and a location on the PROM device cannot be programmed to match the RAM data, the message

```
PROGRAM ERR   aaaa    dd
```

is displayed. The address at which the error was detected is displayed in the Address field (aaaa). The result of an Exclusive OR of the expected and the actual (RAM) data is displayed in the Data field. If this occurs, the PROM is probably faulty. Press the CLEAR key and try another PROM.

A verification is automatically performed after the PROM device has been programmed. See the VER function key description in this chapter for more details. If all locations are programmed without error the message

```
END VERIFY  CKSUM = dddd
```

is displayed, where dddd is the hexadecimal value of the checksum obtained. The checksum is the 2's complement of the 16-bit sum of all the bytes programmed into the PROM device.

Selects Type
of PROM Device

FUNCTION

The DEVICE SELECT key selects the PROM type when a Personality Module capable of programming more than one type of PROM is used.

Each time the DEVICE SELECT key is pressed, the next valid PROM type on the installed Personality Module is indicated. The selected type is indicated by a lighted green indicator next to the PROM type printed on the Personality Module front panel. When the key is pressed on the last PROM type, the first PROM type is again indicated. Press the key until the PROM type desired is indicated.

The DEVICE SELECT key is inoperative if the installed Personality Module can only program one type of PROM. Also, DEVICE SELECT is inactive during programming, verifying, or ROM to RAM loading.

Terminates Current off-line
Function or Clears Entry

FUNCTION

The CLEAR key terminates any ROM TO RAM, VER, or PROG function in progress and restores the display to the following:

IUP READY 0000 dd

Where dd is the data value in RAM at address 0000.

CLEAR also restores the display after some error conditions.

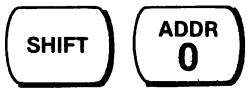
When entering addresses or data in the display, modify, fill or load modes, pressing the CLEAR key causes the field to be returned to the value it was at prior to the entry of the new address or data. If no entry has been made, pressing CLEAR causes the function to be exited and the IUP READY prompt displayed.

Enters Addresses,
Data and Baud Rates

FUNCTION

The ENTER key facilitates the entry of addresses, data and baud rates in the display, modify, fill and load modes. Following the entry of an address or data from the hexadecimal keyboard, the user presses the ENTER key to indicate to the firmware that the entry should be accepted.

See the following descriptions of the SHIFT-ADDR 0, SHIFT-DATA 1, SHIFT-FILL 2 and SHIFT-LOAD 3 functions for further information on the use of the ENTER key.



Selects Display Data function

FUNCTION

The SHIFT key used with the ADDR 0 key selects the display data function. When the SHIFT and ADDR 0 keys are pressed together, the following appears on the alphanumeric display:

EDIT ADDRESS 0000 dd

The least significant digit of the address field blinks to indicate the current field of entry. The most significant digit is entered first. Upon entering another digit, the previous digits entered are shifted left. Once the address field is filled, the least significant digit stops blinking. The user then presses the ENTER key to display the data at the selected address in RAM. If the CLEAR key is pressed instead of the ENTER key, the address field is returned to the original address.

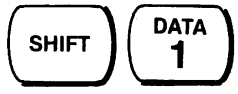
Any valid address entered on the hex keypad followed by the ENTER key causes that address to become the current address value and the data at that location in RAM is displayed. If the ENTER key is pressed again without entering an address, the next sequential location is displayed. If the ENTER key is held down for more than one and a half seconds, sequential addresses are displayed at half second intervals until the key is released.

If the address entered is greater than the current valid size for the selected PROM type, the message

ADDRESS OUT OF RANGE

is flashed on the display for 1 second. Then, all fields (command, address, and data) are displayed as they were prior to entering the invalid address. Address field editing is thus restarted at the address that was in the Address field prior to the erroneous entry.

Note that if no Personality Module is installed, the entire range of the RAM (16K or 32K) may be displayed. Otherwise, the currently selected PROM type selects the valid range.



Selects Modify
Data Function

FUNCTION

The SHIFT key used with the DATA 1 key selects the modify data function. When the SHIFT and DATA 1 keys are pressed together, the following appears on the alphanumeric display:

EDIT DATA aaaa dd

The data field displays the hexadecimal value for the byte of data located at the RAM address currently in the Address field. The least significant digit of the data field blinks to indicate the current field of entry. The most significant digit is entered first. Upon entering another digit, the previous digit entered is shifted left. Once the data field is filled, the least significant digit stops blinking. The user then presses the ENTER key to modify the data at the selected address in RAM. If the CLEAR key is pressed instead of the ENTER key, the data field is returned to the original value.

Once the ENTER key is pressed the address is incremented automatically and the data from the next sequential location is displayed in the Data field for editing.

To exit the modify data mode, press the CLEAR key. The IUP READY prompt is then returned. To modify data at another address or range of addresses, press the SHIFT and ADDR 0 keys to select the address, then press the SHIFT and DATA 1 keys to enter the data modify mode.

Interactively Fills
RAM with Constant

FUNCTION

The SHIFT key used with the FILL 2 key selects the fill function. The fill function allows a contiguous section of RAM locations to be loaded with a constant.

When the FILL function is selected, the user is prompted for needed parameters in an interactive fashion. During entry of these prompted values, the least significant digit of the input field blinks to indicate the current digit of entry. The most significant digit is entered first; then upon entering another digit, the previous digits entered are shifted left. Once the prompted field is filled, the least significant digit stops blinking. The entry must then be entered by1pressing the ENTER key or a function key must be pressed.

Upon pressing the SHIFT and the FILL 2 keys together, the following message is displayed:

FILL FROM 0000

The address field displays 0000 with the least significant digit blinking. A starting address can then be edited and entered using the hex keypad. The data field is blank.

After editing and entering the start address, the following message is displayed:

FILL TO aaaa

The address field contains the previously entered start address as a default ending address. An ending address can then be edited and entered using the hex keypad. The data field is blank. If the ending address that is entered is less than the starting address or greater than the highest PROM address, the message

ADDRESS OUT OF RANGE

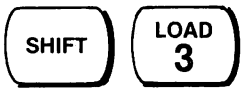
flashes on the display for 1 second, and the FILL TO address is requested again.

When an acceptable address is entered, the following message is displayed:

FILL WITH dd

The address field is blank. The dd is the blank state of the current PROM device selected. A data constant can then be entered using the hex keypad. After pressing ENTER, the selected address range in RAM is filled with the selected constant value.

When the operation is complete, the Universal Programmer prompts for another starting address. Press CLEAR to exit the function and return to the IUP READY prompt.



Downloads Data from
Host System Off-Line

FUNCTION

The SHIFT key used with the LOAD 3 key selects the load data off-line (HEX LOAD) function. This function allows data to be downloaded into the RAM buffer in the off-line mode from a host system via a RS-232 interface. The data loaded must be in the Intel 8080 hexadecimal file format (see Appendix C). The user can select any baud rate from 110 to 9600: 110, 150, 300, 600, 1200, 2400, 4800 and 9600.

When the SHIFT and LOAD 3 keys are pressed together, the following appears on the alphanumeric display:

BAUD RATE = 2400

The least significant digit blinks. If 2400 baud is acceptable, press ENTER. To change the baud rate, enter the desired baud rate from the hexadecimal keypad. Press ENTER when the baud rate has been entered. If an unacceptable baud rate is entered, the following message is flashed on the display for approximately two seconds:

ILLEGAL BAUD RATE

The BAUD RATE request is then displayed again. Re-enter the baud rate.

When the baud rate has been accepted, the following request is displayed:

START ADDRESS 0000

The least significant digit blinks. Enter the address in the file that is to be loaded into location 0000H of the RAM buffer and press ENTER. The data from this file address will be mapped to location 0000H in the RAM buffer. The iUP-201 firmware recognizes addresses from 0000H to FFFFH (64K range). If the START ADDRESS is greater than B000H for a 16K RAM buffer or 8000H for a 32K RAM buffer, the firmware automatically wraps around from memory location FFFFH to 0000H in mapping the RAM memory (see example in Figure 4-2).

When the start address has been entered, the message

HEX LOAD MODE

is displayed, indicating that the iUP-201 is ready to receive data from the RS-232 communications link.

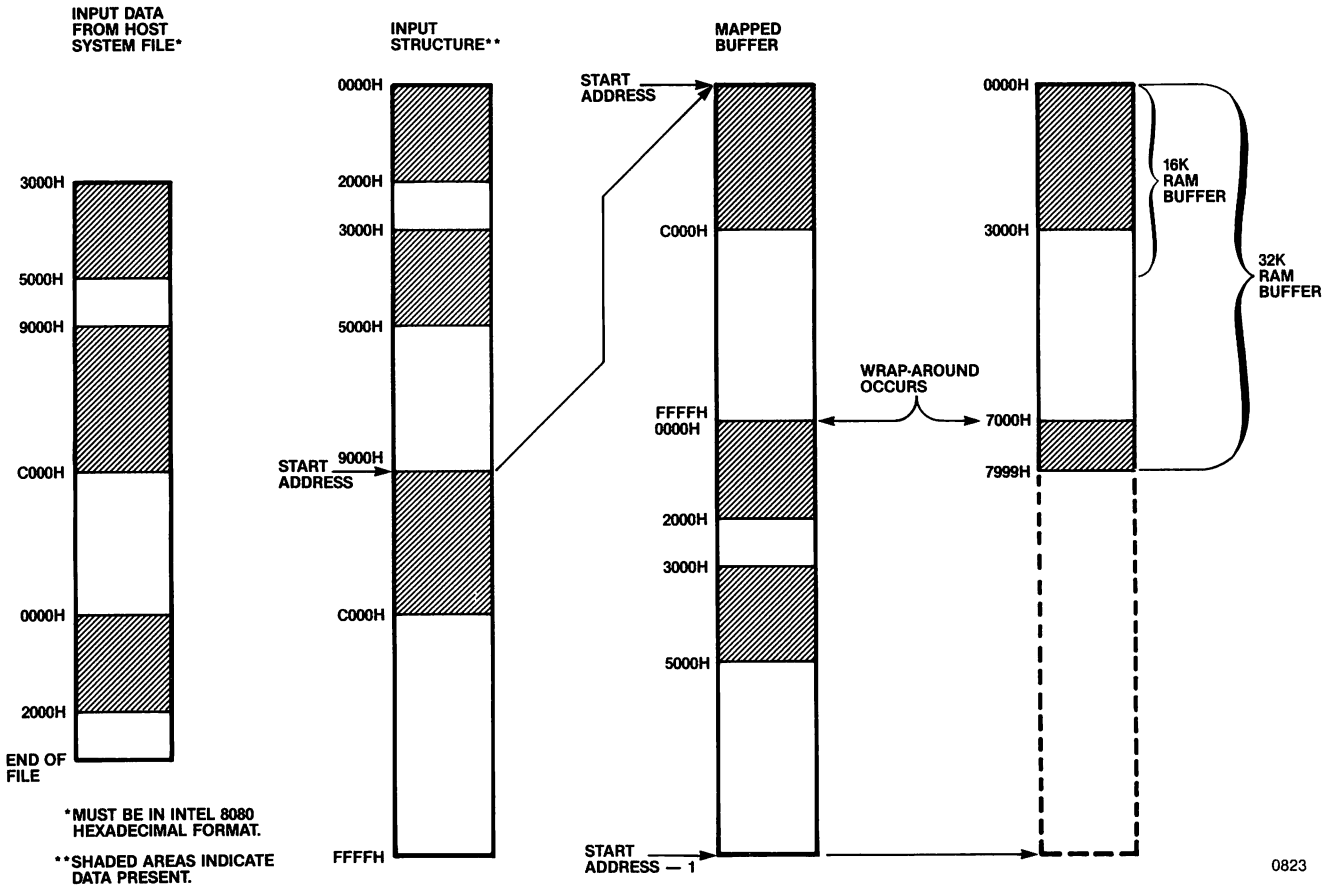


Figure 4-2. Shift-Load 3 Function Data Manipulation

As the data is being read, a check sum is calculated at the end of each record. If an error in the check sum occurs, the following error message is displayed:

CHECK SUM ERROR

The load operation is then terminated.

If the data being read does not match the Intel 8080 hexadecimal file format, the error message

ILLEGAL FILE TYPE

is displayed and the load operation is terminated.

When the end of file is reached, the message

END OF FILE

is displayed. This indicates that the operation is completed. Press the CLEAR key to return to the IUP READY prompt.

CHAPTER 5
PROM PROGRAMMING EXAMPLES

Introduction

The iUP-200/201 offers a wide variety of PROM programming, data display and data editing flexibility. Not only can you quickly duplicate PROMs or program a file from a development system into a PROM, but you can also display and modify data before it is programmed into a PROM. In addition, the FORMAT command offers you a number of more sophisticated data manipulation capabilities like interleaving data in order to load 16-bit words into a pair of 8-bit PROMs, bit masking, byte swapping, nibble swapping and bit reversal.

The following examples are provided to give you hands on experience in using these features. They are divided into two groups: on-line examples, using the iPPS commands; and off-line examples, using the iUP-201 keyboard and alphanumeric display.

Before you go through these examples, it is recommended that you read Chapters 3 and 4 of this manual. You will then be familiar with the iPPS commands and have a working knowledge of the capabilities of the iUP-200/201 and the iPPS software.

If you are using the iUP-201 strictly in its off-line mode, read only Chapter 4 before going to the off-line examples in this chapter.

Table of Contents of Examples

The following table of contents provides a quick reference to the examples in this chapter. It lists the programming or editing functions being illustrated and the commands or iUP-201 function keys being demonstrated in each example.

On-Line Examples

In the following on-line examples, it is assumed that the iUP-200/201 has been powered on as described in Chapter 2, and that it is connected to an Intel Development System via the RS-232 interface. It is also assumed that the Development System has been powered on and is under control of the ISIS-II software.

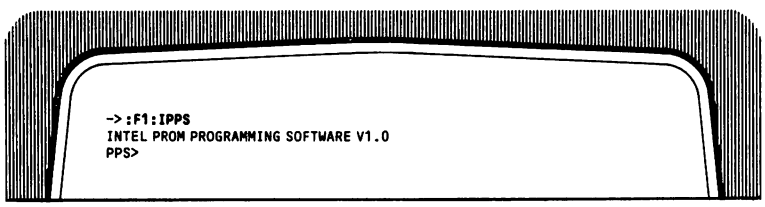
The screen examples are facsimilies of the development system screen. The bold faced characters indicate user entries. The key-in sequence to the left of each screen gives the actual entries that you must key in to obtain the screen display. The words in italics are key cap titles on the development system keyboard.

On-Line iPPS Software Initialization

Before any on-line functions can be performed, the iPPS software must be initialized. After checking that the ISIS-II prompt, ">", is present on the development system CRT screen, enter the following command to invoke the iPPS software:

Key-in Sequence

:F1:IPPS *RETURN*



0825

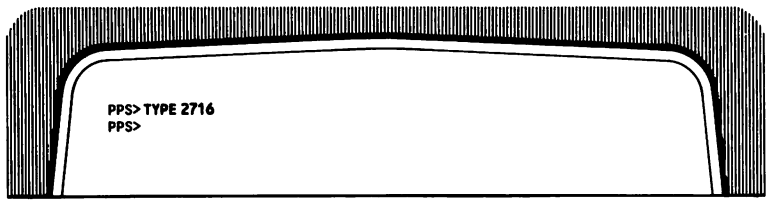
Comments

The iPPS files can be located on any drive. In the above example, they are located on drive F1. The iPPS prompt PPS> indicates that the iPPS files have been loaded and the development system will now execute iPPS commands. The version number of the iPPS software may be different depending on how late a version you have. To return to ISIS-II, type EXIT and a carriage return, anytime following the iPPS prompt.

Before you begin duplicating PROMs on-line or copying files into PROMs, you should establish the PROM type. You may also wish at this time to specify the drive that workfiles are to be put on, and reset the number base and file type default parameters.

Key-in Sequence

TYPE 2716 *RETURN*



0826

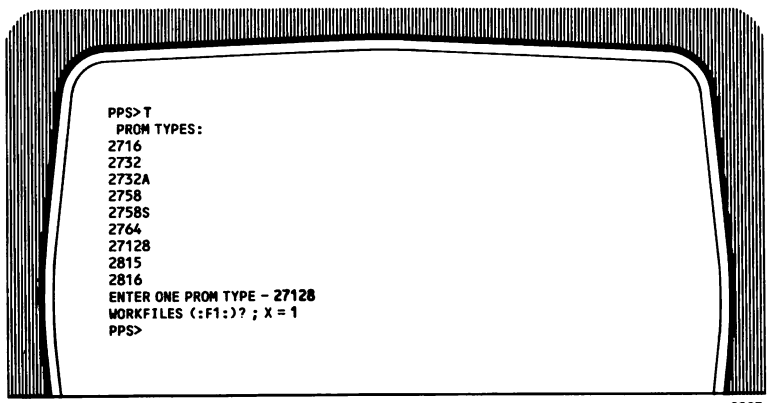
Comments

In this example, the PROM type is specified and the iPPS prompt is returned. The LED indicators on the Personality Module indicate the PROM type selected and the socket to be used. Once the PROM type is selected, the iPPS software automatically sets the boundaries of the iPPS Buffer to be equal to the size of the PROM.

The TYPE command will also give you a list of the PROM types that can be programmed with the Personality Module that has been installed.

Key-in Sequence

T *RETURN*
27128 *RETURN*
1 *RETURN*



0827

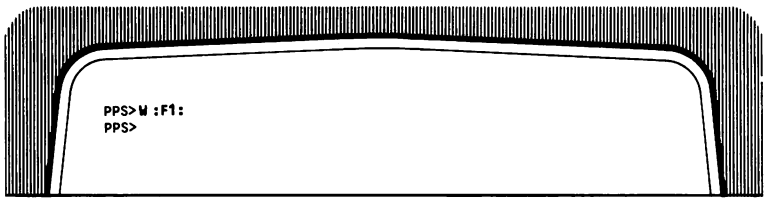
Comments

In this example, the PROM type was not specified in the TYPE command. (Note that the single character abbreviation for the TYPE keyword is used.) All the PROM types that the installed Personality Module supports are then displayed, and the PROM type is requested. In this case, the PROM type selected (27128) requires a virtual buffer. The command thus prompts for a drive to be specified for the temporary workfile. If the workfile drive has not already been specified (see next example) you are prompted for it whenever you specifiy a PROM type with greater than 8K storage capacity.

The WORKFILES command allows you to specify the workfiles drive at any time. This function may be used to change the drive specified earlier or as part of a submit file to avoid the prompt for a workfiles drive specification.

Key-in Sequence

W :F1: *RETURN*



0828

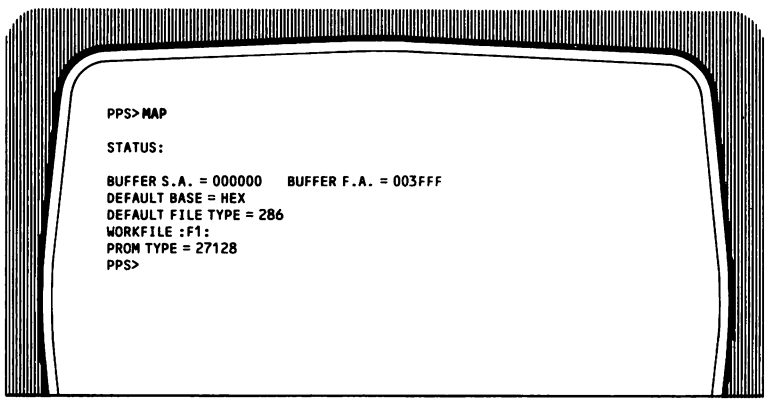
Comments

Drive 1 is selected for all workfiles. Note that the keyword abbreviation was used instead of entering the whole command keyword.

The data type and file type default values may be reset as part of the initialization procedure. Using the MAP command, you can see that the data type and file type default values are initially set to hexadecimal and 286 (Special iPPS-200 output file format), respectively.

Key-in Sequence

MAP *RETURN*



0829

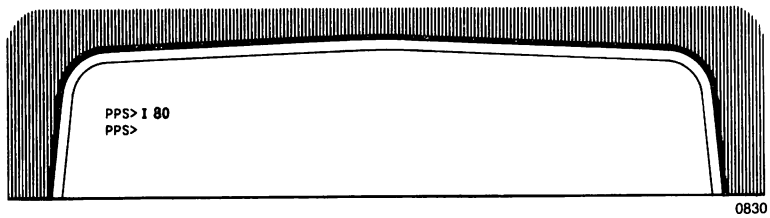
Comments

Besides showing the default values for the number base and file type, the MAP command also shows the boundaries of the iPPS Buffer, the drive specified for workfiles, and the PROM type selected.

The INITIALIZE command allows you to specify the default number base or the default file type or both.

Key-in Sequence

I 80 *RETURN*



0830

Comments

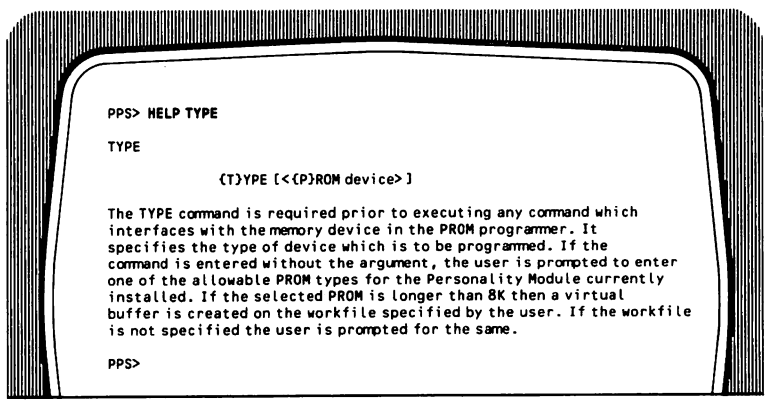
The INITIALIZE command is used in this example to set the default file type to Intel 8080 hexadecimal; the default number base remains at the initial setting of hexadecimal. The file type parameter sets up the iPPS software to read files written in the specified format. If the wrong file type is specified, an error message will be displayed when you attempt to read or copy a file.

Also, many of the iPPS commands allow you to change the number base or file type for the duration of the command. At the end of the command, the base or file type reverts back to the default values. This feature is demonstrated in several of the examples in this chapter.

The iPPS software offers a convenient help feature that allows you to display a description of any of the iPPS commands on the development system screen.

Key-in Sequence

HELP TYPE *RETURN*



0831

Comments

In this example, a description of the TYPE command is requested. The HELP command is particularly useful in determining the correct syntax for an entry.

Duplicating a PROM

One of the most often used applications of the iUP-200/201 is to copy data from a PROM into a buffer or file, then copy it into another PROM. This operation can be performed on-line, using the iPPS buffer or a file in the development system for intermediate storage and the iPPS COPY commands.

If you are performing these examples as you read them, reset the iPPS sofware for the type of PROM you are going to use for the following copy operations. A 2716 EPROM that contains sample code is used in these examples.

The following example illustrates a direct PROM to Buffer to PROM duplication.

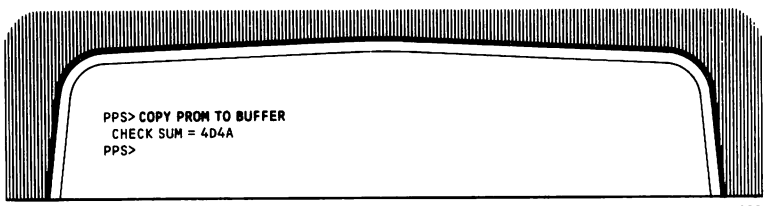
Place the master PROM (the PROM to be copied) in the PROM socket of the Personality Module (See "PROM Device Installation" in Chapter 2.) Then copy the contents of the PROM to the iPPS buffer:

CAUTION

Be sure that the type of PROM that you are installing is the same as the type selected with the TYPE command. A PROM can be damaged when you attempt to program it or read it, if you have specified its type incorrectly.

Key-in Sequence

COPY PROM TO BUFFER
RETURN



0832

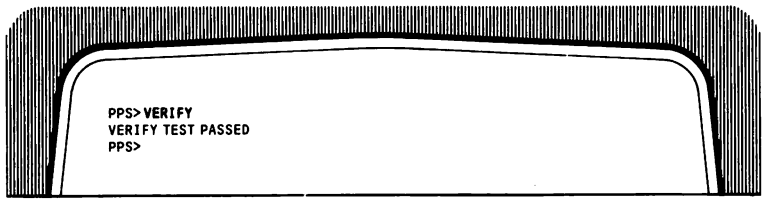
Comments

This command copies every memory location in the PROM to the buffer beginning at destination address 00H in the buffer. The check sum is the two's complement of the 16-bit sum of all the bytes read.

To check that the copy was correct, you can then perform the VERIFY command.

Key-in Sequence

VERIFY *RETURN*



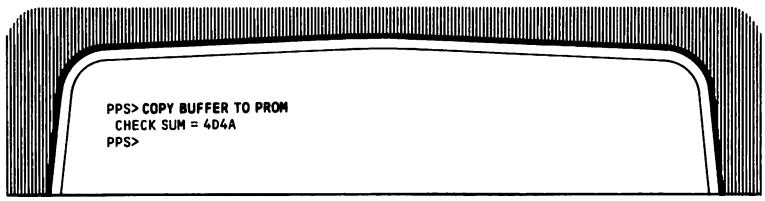
0833

Comments

Now that you are assured that the data in the buffer matches the data in the PROM, you are ready to copy the buffer to a blank PROM. Remove the master PROM from the PROM socket and insert the blank PROM. Then copy the contents of the iPPS buffer to the blank PROM.

Key-in Sequence

COPY BUFFER TO PROM *RETURN*



0834

Comments

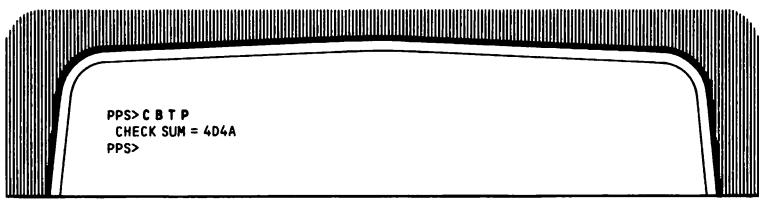
The display of the check sum and the return of the iPPS prompt indicates that the PROM has been successfully programmed.

If you wish to program another blank PROM with the same data, either type the same command, or use the REPEAT command.

Install the blank PROM in the personality module.

Key-in Sequence

C B T P *RETURN*



0835

Comments

In this case, the same command is entered, using the keyword abbreviations. Install the next blank PROM.

Key-in Sequence

REPEAT *RETURN*

```
PPS> REPEAT
C B T P
 CHECK SUM = 4D4A
PPS>
```

0836

Comments

Typing REPEAT, causes the previous command to be echoed and executed.

As part of the copy operation, the iPPS software performs the blankcheck operation to insure that the PROM is indeed blank. If it is not blank, it then performs an overlay test to see if the data in the buffer matches the data already programmed on the PROM, thus allowing the copy to be completed successfully.

Here is an example of an attempt to program a PROM that was not blank.

Key-in Sequence

C B T P *RETURN*
Y *RETURN*
Y *RETURN*

```
PPS>C B T P
PROM NOT BLANK--OVERLAY TEST--Y/N? Y
OVERLAY ERROR--DISPLAY Y/N? Y
  PROM ADDRESS                PROM DATA  EXPECTED DATA
000400                         FB         04
000401                         FE         C3
-COMMAND ERROR--OVERLAY TEST FAILED.
PPS>
```

0837

Comments

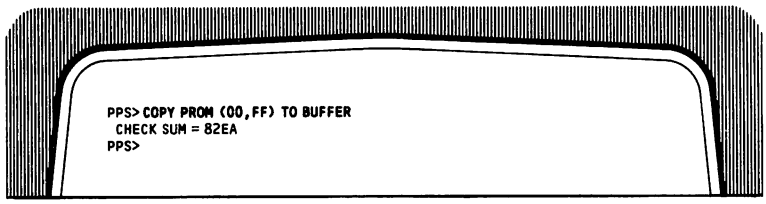
Note that the keyword abbreviations are used.

Now assume that you wish to copy only part of the master PROM and then program the blank PROM with this information, leaving the rest of the PROM in its blank state.

The COPY command permits you to copy a selected range of data from the master PROM into the iPPS buffer, then copy the same range from the buffer back to the blank PROM.

Key-in Sequence

COPY PROM (00,FF) TO BUFFER
RETURN



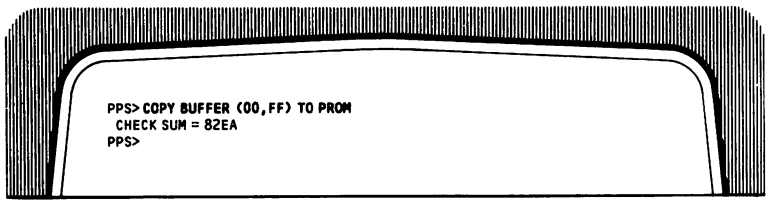
0838

Comments

The program code or data from the address range 00H to FFH in the PROM is copied into the iPPS Buffer.
Now install the blank PROM in the personality module, and copy the contents of the Buffer to the new PROM.

Key-in Sequence

COPY BUFFER (00,FF) TO PROM
RETURN



0839

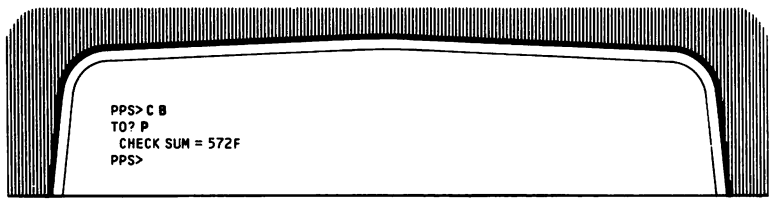
Comments

The data in the selected address range in the Buffer is programmed into the PROM. The other addresses in the PROM remain in their blank state.

The iPPS software prompts you if you do not enter sufficient information in a command or if you enter a syntax error, such as an incorrect command keyword.

Key-in Sequence

C B *RETURN*
P *RETURN*



0840

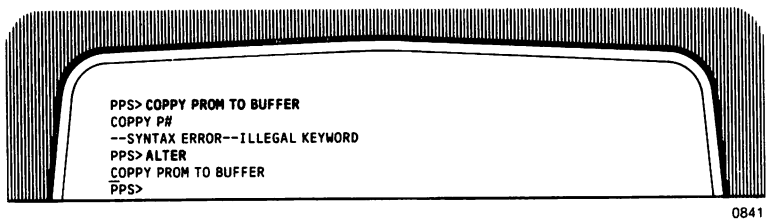
Comments

In this case, the iPPS software prompts you for the second half of the command argument.

In the following example, an incorrect keyword has been entered.

Key-in Sequence

COPPY PROM TO BUFFER
RETURN
ALTER *RETURN*



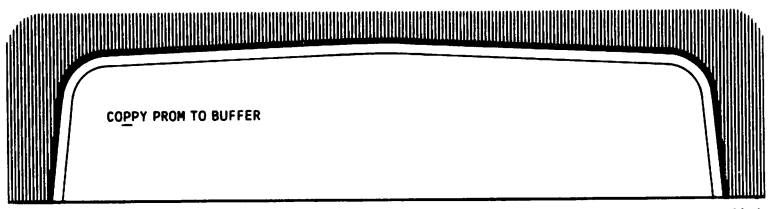
0841

Comments

The part of the command where the syntax error occurred and the error message are displayed. You then use the ALTER command to correct the error. The underscore on the screen display indicates the cursor position.

Key-in Sequence

>
>



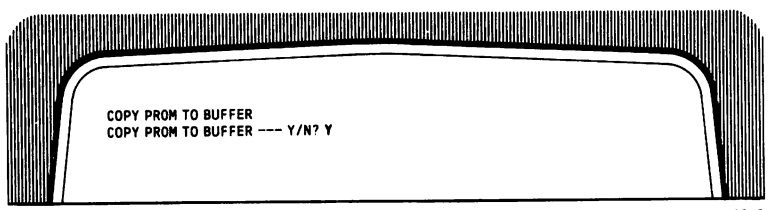
0842

Comments

The cursor is moved under the first P character using the > key.

Key-in Sequence

CNTL-D RETURN
Y *RETURN*



0843

Comments

CNTL-D is typed to delete the extra P character, followed by a RETURN to indicate that editing is complete. The iPPS software executes the corrected COPY command following the entry of Y. The CNTL-I key is used to insert characters in a command.

Examining the Contents of a Masked ROM

The DISPLAY command allows you to examine the contents of a PROM or masked ROM.

Key-in Sequence

DISPLAY PROM *RETURN*
ESC

```
PPS>DISPLAY PROM
000000: C3 40 00 20 20 44 20 2D 20 44 49 53 4B 00 20 20   .@.  D - DISK.
000010: 47 20 2D 20 47 45 4E 45 52 41 4C 00 20 20 4B 20   G - GENERAL.  K
000020: 2D 20 4B 45 59 42 4F 41 52 44 2F 43 52 54 00 FF   - KEYBOARD/CRT..
000030: FF FF FF FF FF FF FF FF C3 36 1C FF FF FF FF FF   .........6......
000040: F3 DB 80 E6 20 CA 03 08 3E 00 D3 01 DB 80 E6 01   .... ...>.......
000050: C2 66 00 3E 4F D3 D0 3E 58 D3 D0 3E 89 D3 D0 3E   .f.>O..>X..>...>
000060: 99 D3 D0 C3 76 00 3E 4F D3 D0 3E 98 D3 D0 3E 8A   ....v.>O..>...>.
000070: D3 D0 3E 9C D3 D0 21 00 00 11 00 08 AF 47 7B B2   ..>...!......G{.
000080: CA 8A 00 78 86 23 1B C3 7D 00 78 FE 55 C2 8D 00   ...x.#..}.x.U...
000090: 3E 34 D3 E3 3E 1F D3 E0 3E 00 D3 E0 01 30 00 DB   >4..>...>....0..
0000A0: 80 E6 01 C2 A9 00 01 2C 00 3E 72 D3 E3 79 D3 E1   .......,.>r..y..
0000B0: 78 D3 E1 3E B2 D3 E3 3E 00 D3 E2 3E 16 D3 E2 D3   x..>...>...>....
0000C0: 10 3E 22 D3 60 D3 50 DB 80 E6 04 CA C7 00 DB 80   .>".;.P.........
0000D0: E6 04 C2 CE 00 AF D3 F0 D3 F0 D3 F0 D3 F1 3E A1   ..............>.
0000E0: D3 F8 3E 23 D3 60 3E C8 D3 E2 3E 00 D3 E2 D3 50   ..>#.;>...>....P
0000F0: 21 EF 00 2B 7C B5 C2 F3 00 DB 80 E6 04 C2 FD 00   !..+|.............
000100: 3E 00 D3 E2 3E 16 D3 E2 D3 50 DB 80 E6 04 CA 0A   >...>....P......
000110: 01 DB 80 E6 04 C2 11 01 3E 22 D3 60 D3 50 DB 80   ........>".;.P..
000120: E6 04 CA 1E 01 DB 80 E6 04 C2 25 01 21 00 40 11   ..........%.!.@.
ENTER <CR> TO CONTINUE $
ABORTED
PPS>
```

0844

Comments

This example shows the data in the PROM in hexadecimal format, which is the default number base that the iPPS software was set for in this example. Press the ESC key at any time to end the display. The "$" sign is the echo of the ESC key. You can also display the data in other number bases.

Key-in Sequence

DISP P(0,L18H) Q *RETURN*

```
PPS>DISP P(0,L18H) Q
00000000: 303 100 000 040 040 104 040 055          .@.  D -
00000010: 040 104 111 123 113 000 040 040           DISK.
00000020: 107 040 055 040 107 105 116 105          G - GENE
PPS>
```

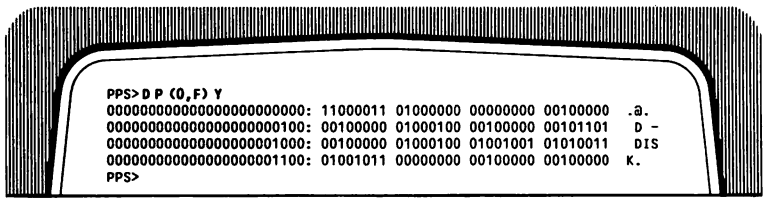
0845

Comments

In the above example, the data at addresses 00Q to 27Q is displayed. Note that although the "Q" specifies that the data be displayed in octal, the address range is specified in hexadecimal.

Key-in Sequence

D P (0,F) Y *RETURN*



0846

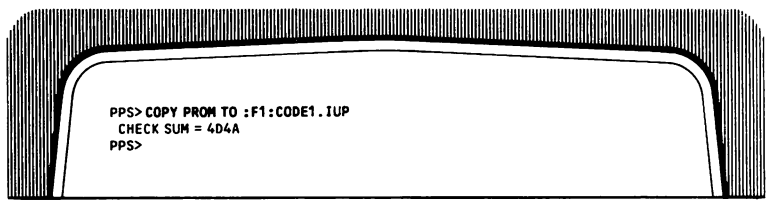
Comments

Here, 16 bytes of ROM data beginning with address 00Y are displayed in binary. Note the ASCII code displayed on the far right column.

To save the information in the PROM, you can copy the contents of the PROM into a file.

Key-in Sequence

COPY PROM TO :F1:CODE1.IUP
RETURN



0847

Comments

The contents of the PROM are saved in an ISIS-II file titled CODE1.IUP, which is located on Drive 1.

Copying a File to a PROM

Another often used application of the iUP-200/201 is to copy programs and data stored in an ISIS-II file into a PROM. This is done using the COPY FILE TO PROM command

Here is an example of a direct copy of a file to a PROM. Assume in the following examples that the blank PROM has been installed in the personality module and the iPPS software has been initialized for the type of PROM to be programmed.

Key-in Sequence

COPY :F1:USER.FIL TO PROM
RETURN

```
PPS>COPY :F1:USER.FIL TO PROM
 CHECK SUM = 4D4A
PPS>
```

0848

Comments

The entire file is copied to the blank PROM beginning at location 0000H.

All the features of the COPY command can be used when copying a file to a PROM. For example, selected sections of code in the file can be copied into a blank PROM. Before you perform this operation, you might wish more information about the structure of the file. The MAP command gives you the boundaries of the file.

Key-in Sequence

MAP :F1:USER.FIL
RETURN

```
PPS>MAP :F1:USER.FIL

FILE STRUCTURE:

 S.A. = 000000      F.A. = 0001FF
 S.A. = 000300      F.A. = 0004AB
 S.A. = 000600      F.A. = 0007FF

STATUS:

BUFFER S.A. = 000000     BUFFER F.A. = 0007FF
DEFAULT BASE = HEX
DEFAULT FILE TYPE = 80
WORKFILE :F1:
PROM TYPE = 2716
PPS>
```

0849

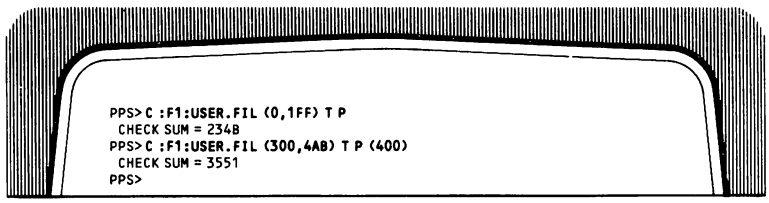
Comments

In this example, the MAP command shows the starting and ending addresses of the various sections of a discontinuous file. The MAP command also gives the buffer boundaries, the status of the iPPS default parameters and the current PROM type selected.

You can now program a PROM with the first two sections of the file.

Key-in Sequence

C :F1:USER.FIL (0,1FF) T P
RETURN
C :F1:USER.FIL (300,4AB) T P (400)
RETURN



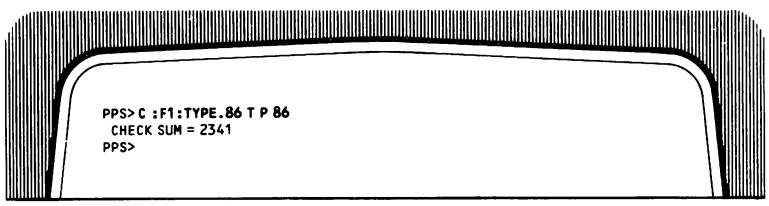
0850

Comments

In the first copy command, code from address locations 00H to 1FFH is loaded into the PROM beginning at location 00H. In the next command, the code from address locations 300H to 4ABH is loaded into the PROM beginning at address 400H. Note that the second section of data has been relocated. Abbreviations are used for the keywords and prepositions.

Key-in Sequence

C :F1:TYPE.86 T P 86
RETURN



0851

Comments

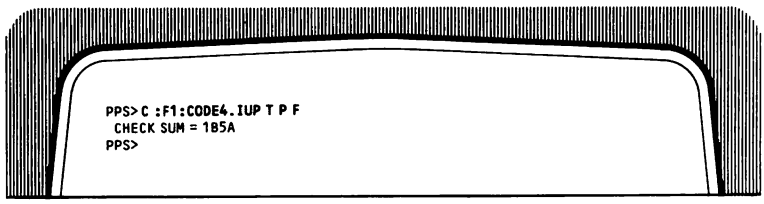
In this example, a file written in Intel 8086 Hexadecimal code is programmed into a PROM. The file type switch "86" was entered as part of the command, because the default file type in this case was "80". This command changes the file type only for the duration of the copy operation. To change the default file type, you must use the INITIALIZE command.

Modifying a File

The iPPS software provides a number of features to allow you to make minor modifications to the data in a file or in the data contained in the iPPS Buffer before programming it into a PROM. One modification, the complementing of data, can be specified as part of the COPY FILE TO PROM command.

Key-in Sequence

C :F1:CODE4.IUP T P F
RETURN



0852

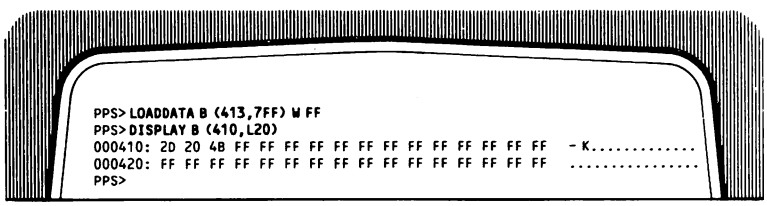
Comments

The F in this command sets the data switch to complement the data as it is being transferred from the file into the PROM. As with the file type specification in the previous example, the F must be specified, because the default value data switch in the iPPS software is uncomplemented.

The availability of the iPPS Buffer allows you to perform other modifications on the data in a file. Assume that the data from a file has been loaded into the iPPS Buffer. Also, using the MAP command, you have determined that the boundaries of the file are from 00H to 412H, and that the boundaries of the Buffer (or PROM) are 00H to 7FFH. The following example then illustrates the use of the LOADDATA command to load a constant into all the locations in the Buffer that do not contain valid data.

Key-in Sequence

LOADDATA B (413,7FF) W FF
RETURN
DISPLAY B (410,L20)
RETURN
RETURN



0853

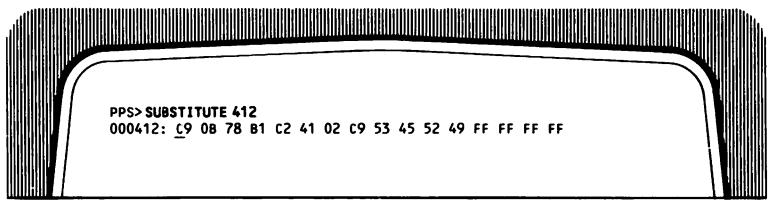
Comments

In this case, the Buffer from address range 413H to 7FFH is loaded with the blank state of the PROM. The use of the display command to display addresses 410H through 42FH shows that addresses 413H and above were indeed filled with the constant FF. Note again that abbreviations were used in some cases for keywords and prepositions.

The SUBSTITUTE command allows you to modify specific bytes in the iPPS Buffer. Again, in the following example it is assumed that the file has been loaded into the iPPS Buffer.

Key-in Sequence

SUBSTITUTE 412 *RETURN*



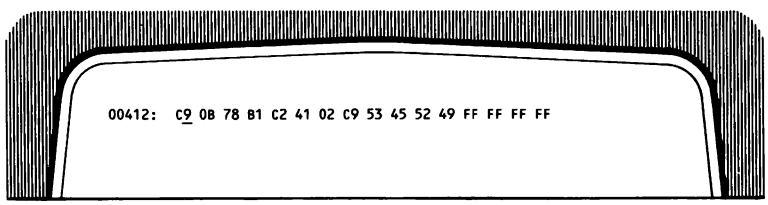
0854

Comments

When the SUBSTITUTE command is invoked, the data at the selected address is displayed along with the data from up to the next 16 addresses. The number of bytes of data displayed depends on the number base specified: 16 for hexadecimal, eight for octal, four for binary and 10 for decimal. In this case, the data from addresses 412H to 421H is displayed. The cursor is then positioned under the first character in the display (indicated in the screen display above with a underscore). By pressing the Space Bar, the <RUBOUT> key, the "<" key or the ">" keys, you can then move the cursor under the character you wish to change.

Key-in Sequence

SPACE



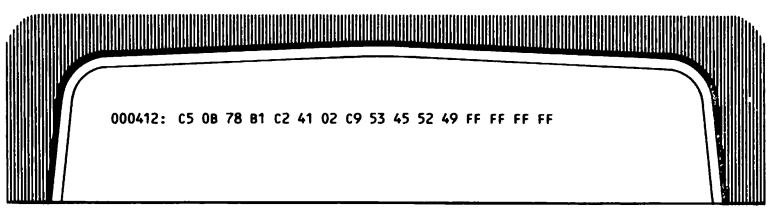
0855

Comments

Pressing the Space Bar once moves the cursor to the least significant hexadecimal character of the data at address 412H.

Key-in Sequence

5 RETURN



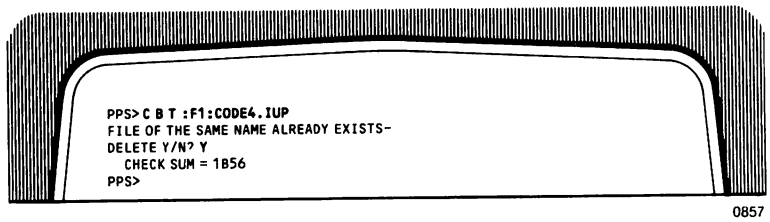
0856

Comments

The byte is then changed from C9H to C5H. The return following the entry causes iPPS to exit the SUBSTITUTE command. Note that the changed data is only temporarily stored in the iPPS Buffer. To save it, you must copy the buffer to a PROM or back to the original file.

Key-in Sequence

C B T :F1:CODE4.IUP
RETURN
Y *RETURN*



0857

Comments

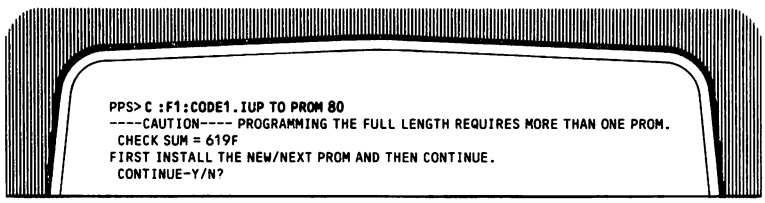
In this example, the modified contents of the Buffer were copied back to the original file. The iPPS software prompts you that a file already exists by that name and allows you to decide whether to delete it or assign a new file name for the data in the buffer.

Copying a File into Two or More PROMs

Files of program code or data are often too large to be stored on a single PROM and must be stored on two or more PROMs. The COPY FILE TO PROM command performs this function automatically, prompting you when to insert the next blank PROM. In this example, assume that the first blank PROM to be programmed has been inserted into the programming socket on the Personality Module.

Key-in Sequence

C :F1:CODE1.IUP TO PROM 80
RETURN



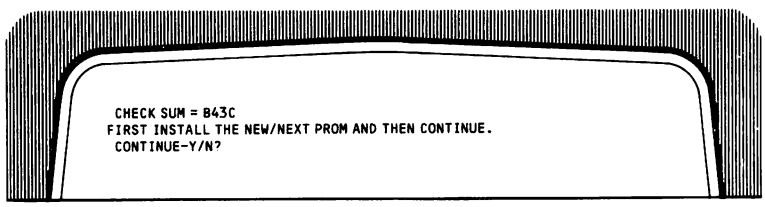
0858

Comments

When the command is initially invoked, it programs the first PROM with as much of the file as will fit on it, then displays a message requesting that the next PROM be installed. Do not remove the first PROM until you see the check sum. Note that the file type is changed to Intel 8080 hexadecimal for the duration of the command.

Key-in Sequence

Y *RETURN*



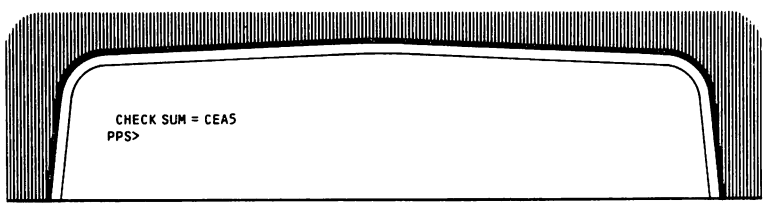
0859

Comments

After you insert the second PROM, press Y and a return. The next section of the file is read into the PROM. When the PROM has been programmed, the check sum is displayed along with a message indicating that still another PROM is required. Remove the second PROM and install the third PROM.

Key-in Sequence

Y *RETURN*



0860

Comments

When the entire file has been read into the PROMs, the check sum is displayed, followed by the iPPS prompt.

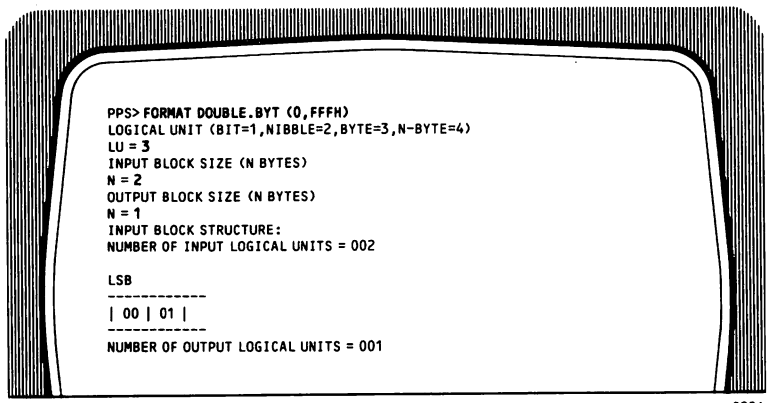
Interleaving a File Between Two PROMs

It is often desirable to have code or data arranged into 16-bit words and stored on a pair of PROMS. This would be the case, for example, when working with an 8086 microprocessor that reads to and writes from memory on a 16-bit data bus. The data must thus be interleaved between the two PROMS, the odd (or low) bytes being stored in one PROM and the even (or high) bytes being stored in the other PROM. The iPPS FORMAT command allows this interleaving function to be handled automatically.

In the following example, a file written in Intel 8086 hexadecimal format will be interleaved into two PROM devices. (It is suggested that you read the description of the FORMAT command in chapter 3 before going through this example to familiarize yourself with the terminology of the command.)

Key-in Sequence

FORMAT DOUBLE.BYT (0,FFFH)
RETURN
3 *RETURN*
2 *RETURN*
1 *RETURN*



0861

Comments

In this example, a file called DOUBLE.BYT is going to be split into two files, with alternate bytes being loaded into alternate files. After establishing the FORMAT command and the file name with the first entry, the iPPS software prompts for the size of the logical unit that is going to be manipulated. Byte is selected as the logical unit. You are then prompted to set up the input block size (in this case two bytes) and the output block size (one byte). A diagram of the input block structure is then displayed with the logical units labeled. The least significant bit in the input block is shown on the left. The number of logical units in the output block is also displayed.

Key-in Sequence

***0 TO :F1:LOWER.BYT** *RETURN*
***1 TO :F1:UPPER.BYT** *RETURN*
* *RETURN*

```
*0 TO :F1:LOWER.BYT
OUTPUT STORED
*1 TO :F1:UPPER.BYT
OUTPUT STORED
*
PPS>
```

0862

Comments

Once the size of the logical unit, the input block size and the output block sizes have been established, you are prompted for the output specification (how you want the data in the file to be manipulated in terms of logical units). In this example, it is specified that the least significant byte in each input block be stored in a file titled LOWER.BYT. The iPPS software then sorts through the DOUBLE.BYT file and copies every even byte into the LOWER.BYT file. Next it is specified that the most significant byte be stored in a file titled UPPER.BYT. The iPPS software then sorts through the DOUBLE.BYT file and copies every odd byte to the UPPER.BYT file. OUTPUT STORED is displayed after each output specification is implemented. You then have the option of entering another output specification. Pressing RETURN exits the FORMAT command and returns the iPPS prompt.

Key-in Sequence

D :F1:DOUBLE.BYT(0,L20)
RETURN
D :F1:LOWER.BYT(0,L10)
RETURN
D :F1:UPPER.BYT(0,L10)
RETURN

```
PPS>D :F1:DOUBLE.BYT(0,L20)
000000: C3 40 00 20 20 44 20 2D 20 44 49 53 4B 00 20 20  .@.  D - DISK.
000010: 47 20 2D 20 47 45 4E 45 52 41 4C 00 20 20 4B 20  G - GENERAL. K
PPS>D :F1:LOWER.BYT(0,L10)
000000: C3 00 20 20 20 49 4B 20 47 2D 47 4E 52 4C 20 4B  ..   IK G-GNRL K
PPS>D :F1:UPPER.BYT(0,L10)
000000: 40 20 44 2D 44 53 00 20 20 20 45 45 41 00 20 20  @ D-DS.   EEA.
```

0863

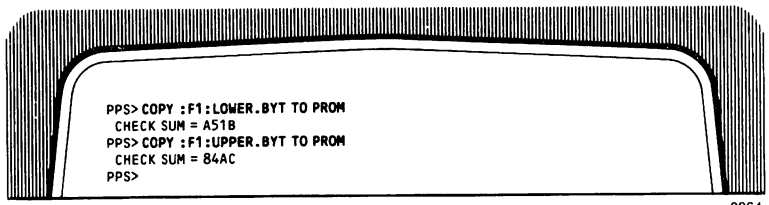
Comments

By displaying the first few lines of the DOUBLE.BYT, LOWER.BYT and UPPER.BYT files, you can see that the bytes from even address in the DOUBLE.BYT file have been stored in the LOWER.BYT file and the odd address bytes have been stored in the UPPER.BYT file.

The two files created with this FORMAT operation can be used to program two PROMs which can then be installed in parallel to provide 16-bit wide address data words to a 16-bit microprocessor.

Key-in Sequence

COPY :F1:LOWER.BYT TO PROM
RETURN
COPY :F1:UPPER.BYT TO PROM
RETURN



0864

Comments

A blank PROM must be installed in the iUP-200/201 before entering each COPY command.

Masking a Parity Bit

The preceding example illustrates one of the many possible applications of the FORMAT command. This example shows how the FORMAT command can be used to mask a parity bit in a file. In addition, all the lower case letters in the file will be changed to upper case.

Key-in Sequence

FORMAT :F1:PARITY.EVN
(0,FFFH) *RETURN*
1 *RETURN*
1 *RETURN*
1 *RETURN*

```
PPS> FORMAT :F1:PARITY.EVN (0,FFFH)
LOGICAL UNIT (BIT=1,NIBBLE=2,BYTE=3,N-BYTE=4)
LU = 1
INPUT BLOCK SIZE (N BYTES)
N = 1
OUTPUT BLOCK SIZE (N BYTES)
N = 1
INPUT BLOCK STRUCTURE:
NUMBER OF INPUT LOGICAL UNITS = 008

LSB
-----------------------------------------
| 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 |
-----------------------------------------
NUMBER OF OUTPUT LOGICAL UNITS = 008
```

0865

Comments

A file titled PARITY.EVN is used as a source of input data. The bit is established as the input logical unit, and the input block size and the output block size are both set at 1 byte.

Key-in Sequence

***0,1,2,3,4,F,6,F TO**
:F1:PARITY.MSK *RETURN*
***** *RETURN*

```
*0,1,2,3,4,F,6,F TO :F1:PARITY.MSK
OUTPUT STORED
*
PPS>
```

0866

Comments

Bit 5 is the upper case bit in each byte and bit 7 is the parity bit. In the output specification, the constant F is loaded in bits 5 and 7 in each output block (F causes a 0 to be loaded in the specified bit location). The other bits are loaded with the same data that is in the input block. Each character is thus switched from lower case to upper case and the parity bit is masked.

Key-in Sequence

D :F1:PARITY.EVN (0,F) Y
RETURN
D :F1:PARITY.MSK (0,F) Y
RETURN

```
PPS>D :F1:PARITY.EVN (0,F) Y
0000000000000000000000000: 11110000 11100001 01110010 01101001 ..ri
0000000000000000000000100: 01110100 11111001 11111111 11101101 t...
0000000000000000000001000: 11100001 11110011 11101011 11111111 ....
0000000000000000000001100: 01101111 11110101 01110100 11111111 o.t.
PPS>D :F1:PARITY.MSK (0,F) Y
0000000000000000000000000: 01010000 01000001 01010010 01001001 PARI
0000000000000000000000100: 01010100 01011001 01011111 01001101 TY_M
0000000000000000000001000: 01000001 01010011 01001011 01011111 ASK_
0000000000000000000001100: 01001111 01010101 01010100 01011111 OUT_
PPS>
```

0867

Comments

By displaying the first 16 bytes of both the PARITY.EVN and PARITY.MSK files you can see that bits 5 and 7 in each byte of the PARITY.MSK file has been changed to 0. The ASCII display to the right of the screen shows that the characters have been switched to upper case.

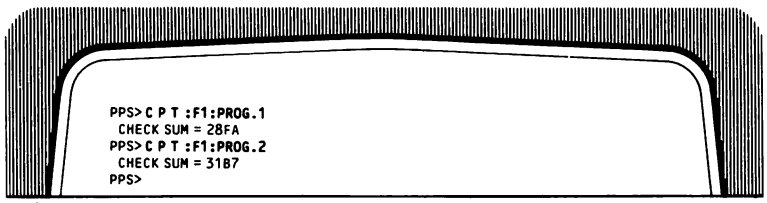
Programming a Single PROM with Code from a Number of Smaller PROMS

The iPPS also allows you to copy the program code or data from a number of smaller PROMs and copy it to a larger PROM. For example, perhaps you have a program stored on two 2716 PROMs, which have 2K of storage capacity each, and you wish to put the entire program on a 2732 PROM, which has 4K of storage space.

You first need to copy the code for the two PROMs into two separate files.

Key-in Sequence

C P T :F1:PROG.1 *RETURN*
C P T :F1:PROG. 2 *RETURN*



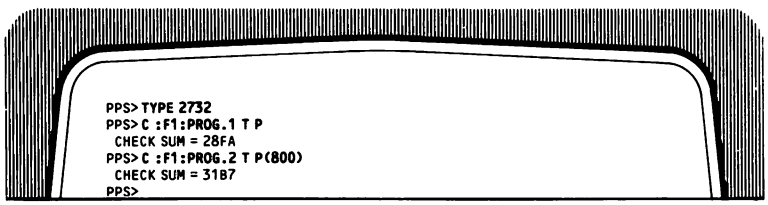
0868

Comments

The contents of the two PROMs are stored in files titled PROG. 1 and PROG.2. (In between the two copy commands the PROMs are exchanged in the PROM socket on the Personality Module.)

Key-in Sequence

TYPE 2732 *RETURN*
C :F1:PROG.1 T P *RETURN*
C :F1:PROG.2 T P(800) *RETURN*



0869

Comments

Having created the two files, you change the PROM type to 2732 and insert the new blank PROM. You then copy the two files into the PROM. Note that the second file must be copied into the upper 2K of the new PROM.

Off-Line Examples

The iUP-201 keyboard and display module allows you to perform many of the PROM to PROM programming functions described for on-line operations. These functions include:

1. Copying a PROM to a PROM
2. Copying an Intel 8080 hexadecimal file from a host system to a PROM.
3. Modifying the data in iUP-201 RAM before programming a PROM.

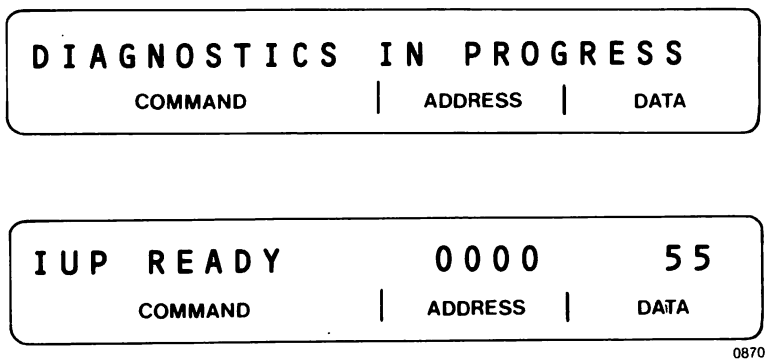
The following examples illustrate these functions. In these examples, the key-in sequence for a particular operation is given on the left of the page and the resulting display is given in the facsimile alphanumeric display on the right. Any numbers given in a key-in sequence are entered on the hexadecimal keypad. After the number has been entered, the <ENTER> key must be pressed to complete the entry.

Off-Line iUP-201 Initialization

Before any on-line functions can be performed, the iUP-201 firmware must be initialized.

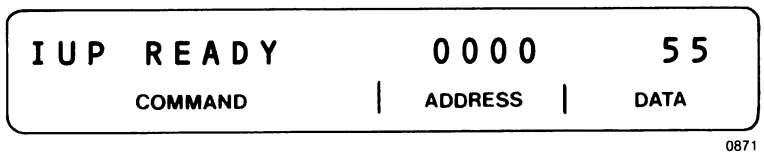
Key-in Sequence

Power Switch turned on



When the rear panel POWER switch is initially switched to its on position, a diagnostics routine is run. When the diagnostics have been completed, the "IUP READY" prompt is displayed. This is the off-line prompt. Pressing the <ON LINE> key cause the "ON LINE" prompt to be displayed. The iUP-201 must be in the off-line mode, however, to perform the following examples.

Key-in Sequence



Before you begin off-line programming, you also need to set the iUP-201 for the type of PROM to be duplicated or programmed. Each time the <DEVICE SELECT> key is pressed the Personality Module is set to the next device type. The LED indicators on the Personality Module indicate the current PROM type selected and the socket in which it should be installed.

A 2716 EPROM that contains sample code is used in the following examples. The data and check sums shown were obtained from this code.

Duplicating a PROM

The first step in copying a PROM to a PROM is to read the contents of the master PROM into the iUP-201's RAM memory.

NOTE

The off-line RAM and the URAM refered to in the on-line iPPS program are the same.

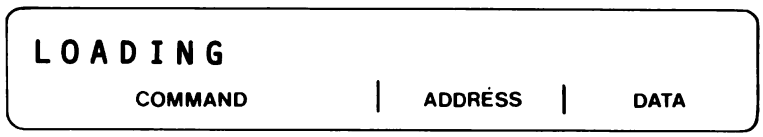
To do this, the master PROM is placed in the PROM socket of the personality module (see "PROM Device Installation" in Chapter 2.).

CAUTION

Be sure that the type of PROM that you are installing is the same as the type selected with the DEVICE SELECT key. A PROM can be damaged when you attempt to program it or read it, if you have not specified its type correctly.

To copy the contents of the master PROM into the iUP-210 RAM, the ROM TO RAM key is pressed.

Key-in Sequence



0872

The check sum is the two's complement of the 16-bit sum of each byte read.

The verify function allows the contents of the RAM to be checked against the contents of the master PROM.

Key-in Sequence

VER

```
VERIFY
COMMAND | ADDRESS | DATA

END VERIFY  CKSUM = 09AB
COMMAND | ADDRESS | DATA
```

0873

With the RAM loaded and verified, the new PROM can now be programmed. The master PROM is removed from the personality module and replaced with the blank PROM. The contents of the RAM are then copied to the blank PROM.

Key-in Sequence

PROG

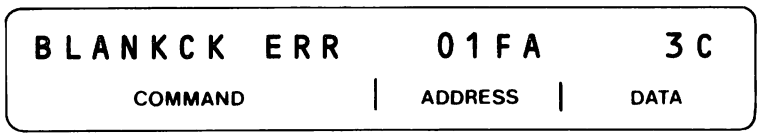
```
BLANKCK
COMMAND | ADDRESS | DATA

PROGRAMMING
COMMAND | ADDRESS | DATA

END VERIFY  CKSUM = 09AB
COMMAND | ADDRESS | DATA
```

0874

As part of the programming routine, the iUP-201 performs a blankcheck of the new PROM to insure that it can be programmed. When the programming is complete, it performs the verify function to check that the programming was done correctly.

Note that the blank check routine generally happens so fast that the message "BLANKCK" is only flashed momentarily. Also, the message "PROGRAMMING" blinks to indicate that programming is in progress. The programming of a PROM can last several minutes.

If the iUP-201 discovers that the PROM to be programmed is not blank, it displays a blank check error message:



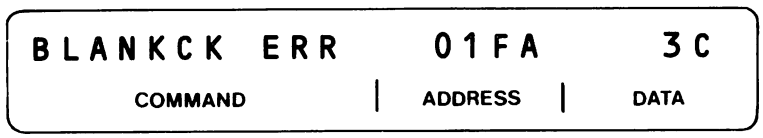
0875

The address of the byte that was not blank is displayed.

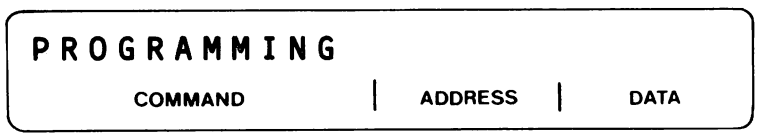
The fact that the PROM is not blank, does not necessarily mean that it can not be programmed. Pressing the PROG key again causes the iUP-201 to perform a stuck bit routine to check if the non-blank bits correspond with bits in the RAM. If the routine indicates that the PROM can be programmed successfully, the programming routine is initiated automatically. Here is an example of non-blank PROM being successfully programmed.

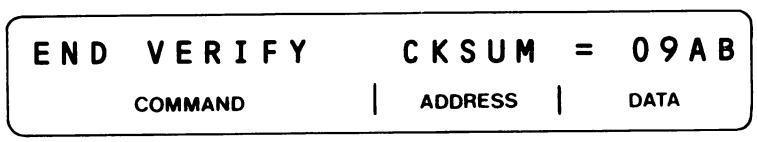
Key-in Sequence

PROG



PROG





0876

The stuck bit routine usually occurs so fast for the "STUCKBIT CHK' message is only flashed momentarily.

In the following example, the stuck bit routine determines that the PROM cannot be programmed.

Key-in Sequence

PROG

```
BLANKCK ERR    003A    5F
COMMAND | ADDRESS | DATA
```

PROG

```
STUCKBIT CHK
COMMAND | ADDRESS | DATA
```

```
DEVICE WILL NOT PROGRAM
COMMAND | ADDRESS | DATA
```

0877

If a error occurs during programming or if the PROM is defective and one or more of the bits cannot be programmed to match the RAM data, the following message is displayed:

Key-in Sequence

PROG

```
BLANKCK
COMMAND | ADDRESS | DATA
```

```
PROGRAMMING
COMMAND | ADDRESS | DATA
```

```
PROGRAM ERR    03A1    5B
COMMAND | ADDRESS | DATA
```

0878

The address at which the error was detected is displayed in the Address Field. A faulty PROM is most often the cause of a program error. Pressing the CLEAR key then allows another blank PROM to be programmed with the same information.

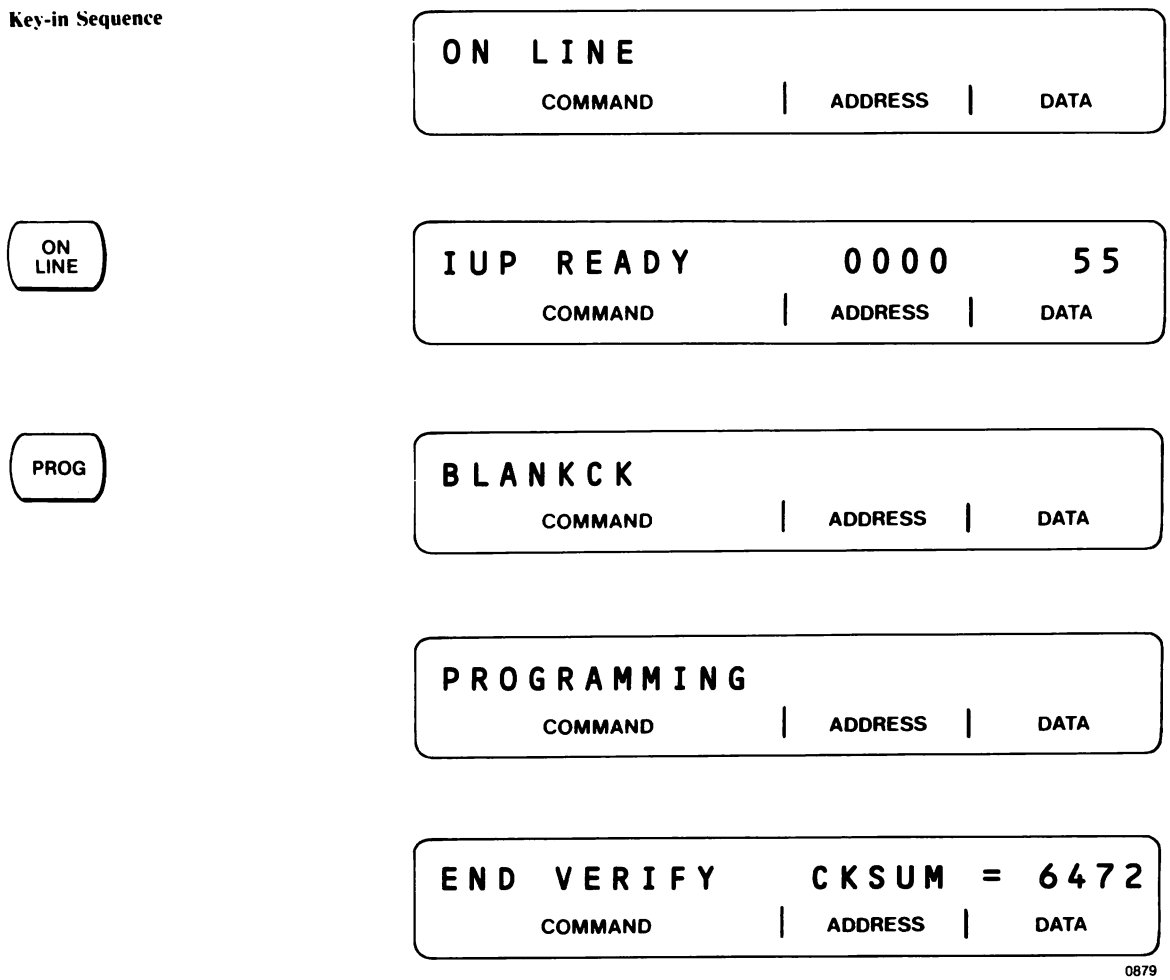
Copying a Development System File to a PROM

The iUP-201 can copy a file from a development system into its RAM buffer either on-line with the iPPS software or off-line. Once the file is in the RAM, it can then be copied into a PROM off-line.

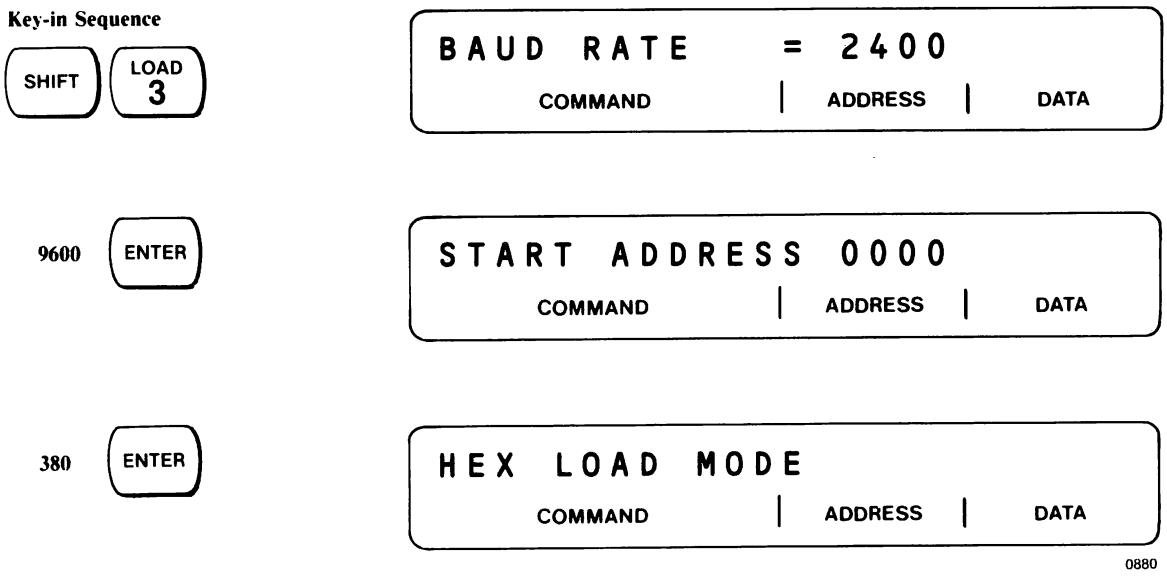
In the on-line mode, the COPY FILE TO URAM command is used. (Note that in the on-line mode, the term "URAM" is used to refer to the RAM buffer in the iUP-201.) In the following example, the data from file CODE4.IUP on drive :F1: is copied (starting at address 380H in the file) to the RAM (starting at URAM address 00H).

```
PPS>COPY :F1:CODE4.IUP(380,7FF) TO URAM
 CHECK SUM = 6472
PPS>
```

A PROM can then be programmed from the RAM, off line as in the following example.



This same function can be carried out off-line (without invoking iPPS software) provided that the file is in Intel 8080 hexidecimal format. You initiate the writing of the file into the iUP-201 RAM buffer off line as shown in this example.



When the SHIFT-LOAD 3 function is initiated, you are prompted for a baud rate. Any standard baud rate from 110 to 9600 may be selected. In this example 9600 baud is entered. You are then prompted for a starting address in the file to be downloaded. This address will be loaded into location 00H in the RAM buffer. In this case a starting address of 380H is selected. Once the starting address is entered, the message "HEX LOAD MODE" is displayed. You can now initiate the download from the host system.

When using a Intel development system, ISIS commands are then entered to instruct the development system to download the file to the teletype port:

```
>SPEED 9600
>COPY :F1:CODE4.IUP TO :TO:
COPIED :F1:CODE4.IUP TO :TO:
>
```

When the file has been read, the message "END OF FILE" is displayed, on the iUP-201 alphanumeric display. Press the <CLEAR> key to return to the "IUP READY" prompt. If the file is not in Intel 8080 hexadecimal format, the message "ILLEGAL FILE TYPE" is displayed

In this example, the file is loaded into the RAM buffer, with address 380H of the file stored at address 00H of the RAM buffer (see figure 5-1).

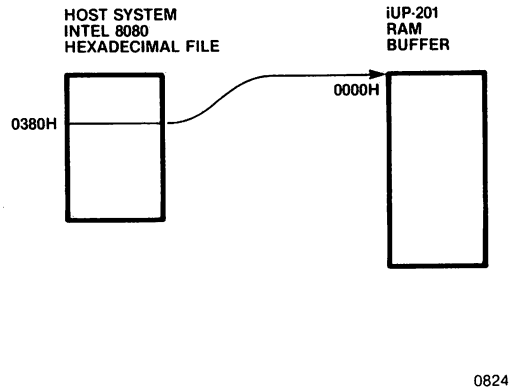


Figure 5-1. Example of SHIFT-LOAD 3 Function.

Once the file has been loaded into RAM, it can be copied to a PROM.

Key-in Sequence

IUP READY	0000	55
COMMAND	ADDRESS	DATA

PROG

BLANKCK		
COMMAND	ADDRESS	DATA

PROGRAMMING		
COMMAND	ADDRESS	DATA

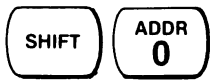
END VERIFY	CKSUM = 6472	
COMMAND	ADDRESS	DATA

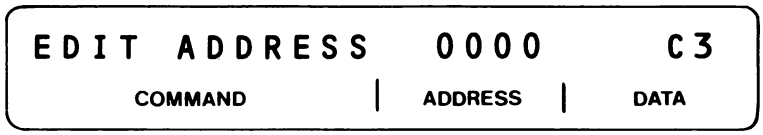
0881

Modifying Data in the iUP-201 RAM

The iUP-201 provides the facilities to both display data in its RAM buffer and modify selected bytes of data. The SHIFT-ADDR 0 function is used to display data.

Key-in Sequence

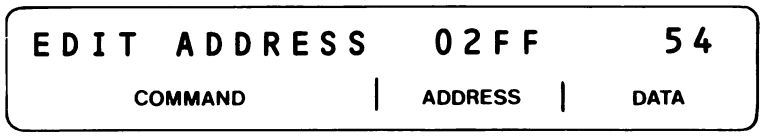




0882

The least significant digit of the address field blinks to indicate that addresses are to be entered. To look at the contents of a specific byte, its address is then entered:

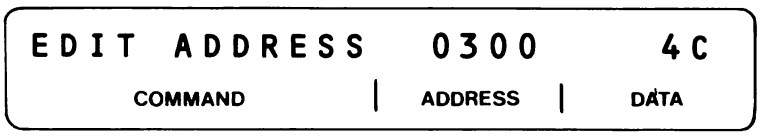
Key-in Sequence

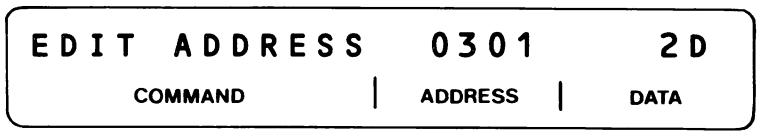


0883

Pressing the ENTER key alone automatically increments the address by 1.

Key-in Sequence





0884

If an address greater than the highest allowable PROM address is entered, the following message is displayed.

Key-in Sequence

5FFF ENTER

```
ADDRESS OUT OF RANGE
COMMAND | ADDRESS | DATA
```

```
EDIT ADDRESS  0301  2D
COMMAND | ADDRESS | DATA
```

0885

Approximately one second after the "ADDRESS OUT OF RANGE" message is displayed, the previous address and data are again displayed.

To modify the data in a byte, you first select the address of the byte that you wish to modify.

Key-in Sequence

3A8 ENTER

```
EDIT ADDRESS  03A8  D3
COMMAND | ADDRESS | DATA
```

0886

You then enter the data edit mode (SHIFT-DATA 1).

Key-in Sequence

SHIFT DATA 1

```
EDIT DATA  03A8  D3
COMMAND | ADDRESS | DATA
```

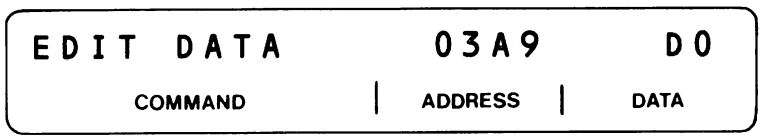
0887

The least significant field of the data field blinks, indicating that you are in the data modify mode.

Enter the new data to be stored at the selected address.

Key-in Sequence

2F ENTER

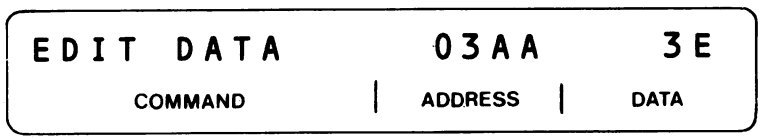


0888

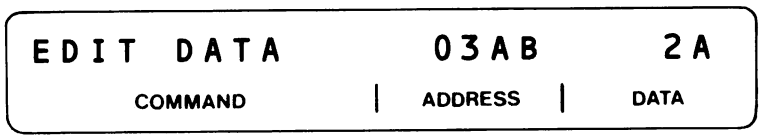
Once the data is modified, the address is automatically incremented and the least significant field on the data field blinks indicating that the next byte can be modified. You can continue to modify bytes in sequential addresses in this manner

Key-in Sequence

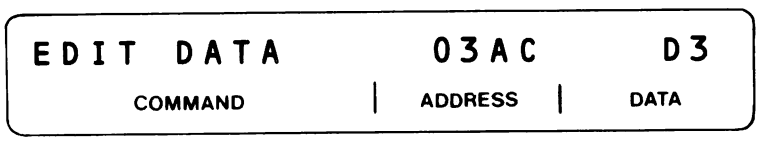
34 ENTER



6F ENTER

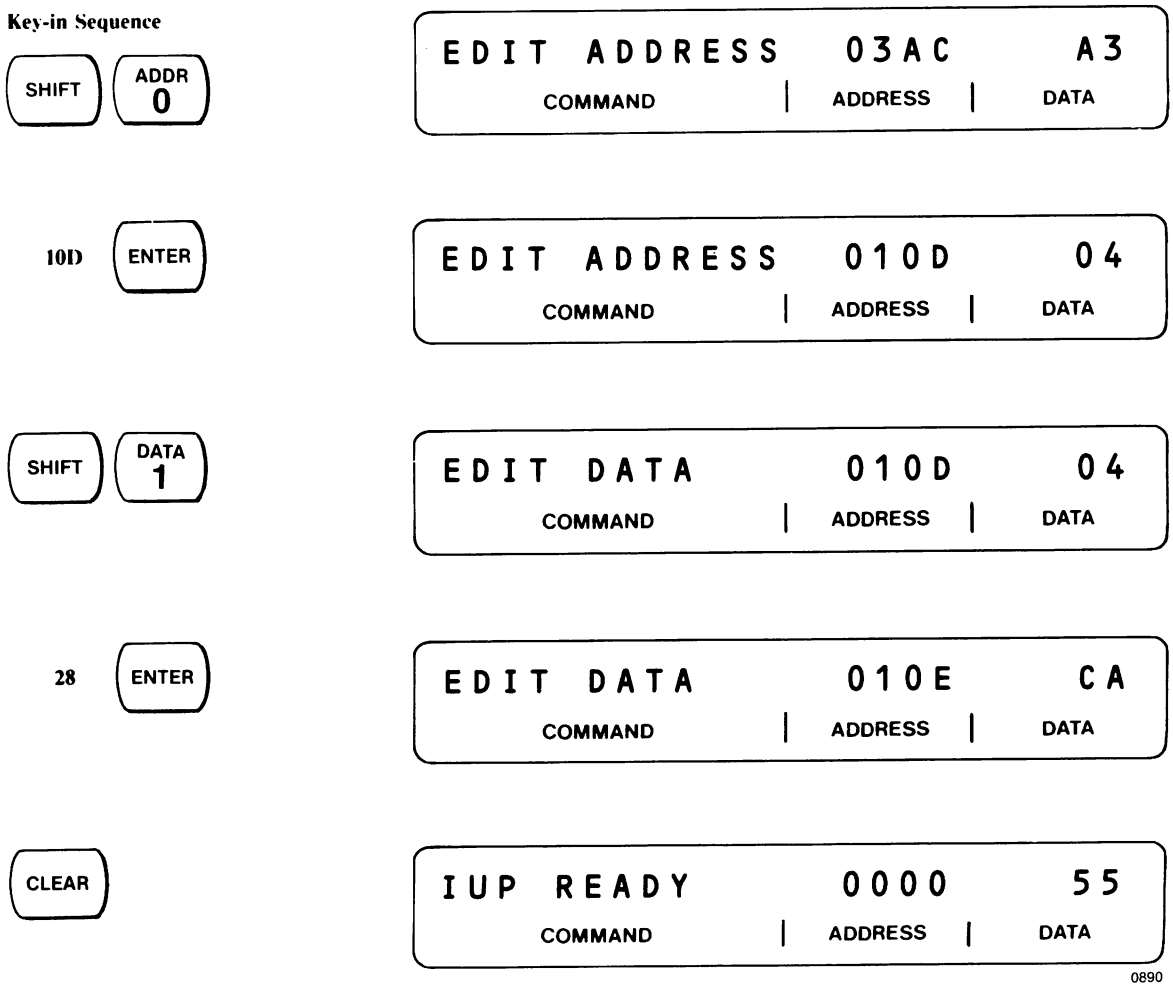


77 ENTER



0889

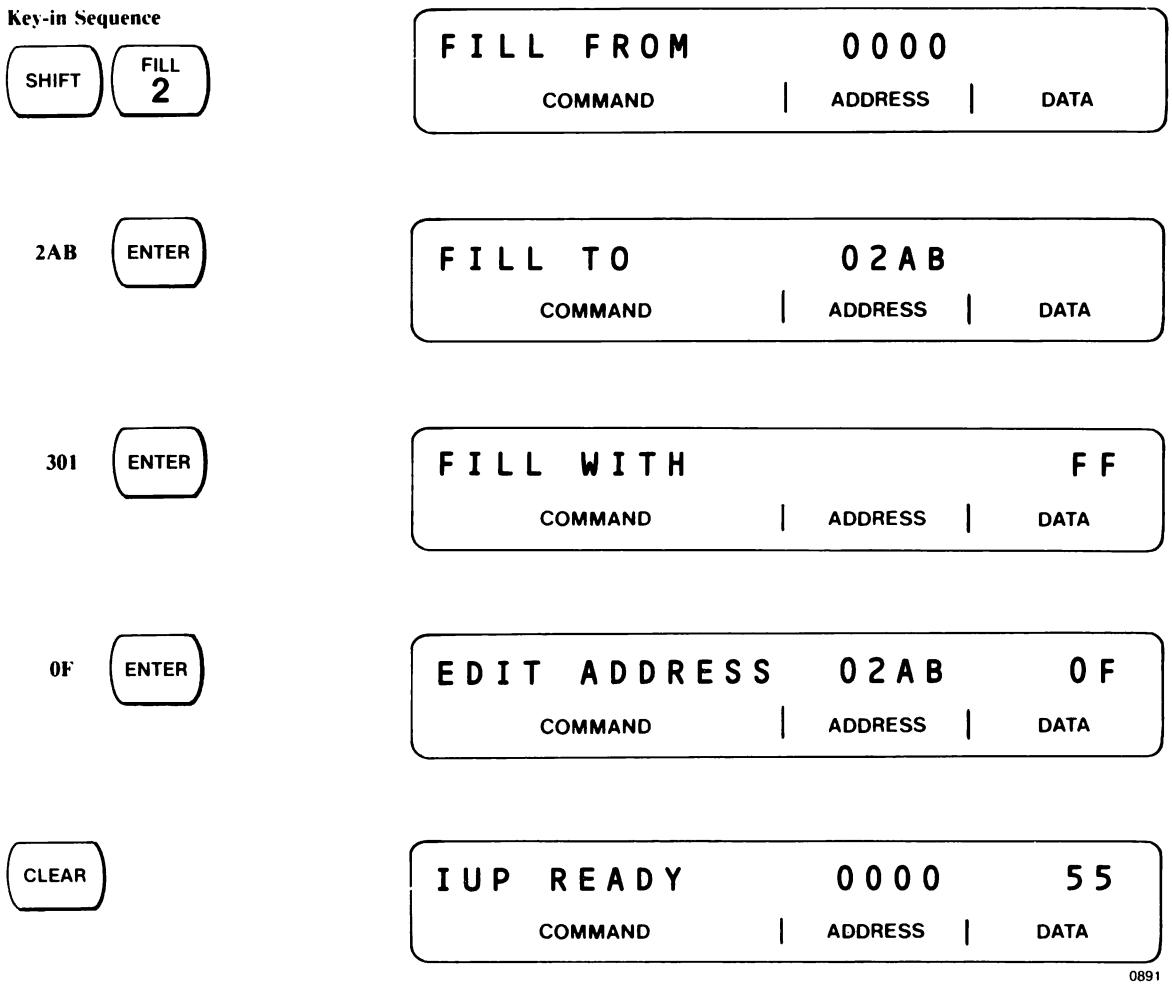
To modify a byte that is not insequence, the display data mode (SHIFT-ADDR 0) must again be entered to establish the address to be modified, then the edit data mode is entered again.



Pressing the CLEAR key returns program control to the IUP READY state

The SHIFT-FILL 2 function provides an alternate method of modifying the data in the iUP-201 RAM buffer. This function allows all the bytes in a selected address range to be set for a specific constant.

In the following example, the bytes from addresses 2ABH to 301H are filled with the constant 0FH.



Once the addresses have been filled with the constant, the iUP-201 goes into the display address mode and the data in the first address to be filled is displayed. Press the <CLEAR> key to return to the "IUP READY" prompt.

Note that the maximum address that can be entered from a 2716 EPROM is 7FFH. If an address greater than that is entered, the messasge "ADDRESS OUT OF RANGE" is displayed.

APPENDIX A
ERROR MESSAGES AND CONDITIONS

Self-Diagnostic Errors

Self-test diagnostics are run when the Universal Programmer is initially powered ON. These errors indicate hardware failures in the Universal Programmer. See the Service and Repair Assistance section of Chapter 2 for information about obtaining service for these hardware failures. For any of the tests that fail, the corresponding error messages are displayed on the iUP-201 alphanumeric display:

POWER SUPPLY FAILURE
: The internal voltages in the iUP-200/201 are not within correct tolerances.

MOTHERBOARD FAILURE
: One of the components of the iUP-200/201 motherboard did not pass the diagnostics. The bad component could be the 8085 CPU, the ROM firmware, the RAM, or the timer.

KEY/RAM BOARD FAILURE
: The circuit board containing the iUP-201 keyboard and URAM did not pass the diagnostics.

If the Universal Programmer is the model iUP-201, then these messages are displayed on the off-line alphanumeric display only. This is because the iUP-201 initializes in the off-line mode of operation. If the Universal Programmer is the model iUP-200, then these messages will appear on the host-development console.

If all the self-diagnostic tests are passed, the iUP-201 displays

```
IUP READY      0000      55
```

respectively in the Command, Address, and Data fields of the alphanumeric display.

Off-Line Errors

With the model iUP-201, several messages can appear on the LED display to indicate an error during off-line functions:

MODULE CHECKSUM ERROR
: The load function (ROM TO RAM key), verify function (VER key), and program PROM function (PROG key) all compute a checksum on the data stored in the Personality Module's firmware ROM. If an error is detected, this message is displayed.

MODULE NOT INSTALLED

An attempt was made to access a PROM device when no Personality Module was installed.

CHECK PROM INSTALLATION

There is no PROM device installed, or the PROM is installed improperly. See Chapter 2 for instructions on installing the PROM.

VERIFY ERR @ aaaa dd

The verify function (VER key) displays this error message if it detects a mismatch of data in the RAM and PROM. See the VER function key description in Chapter 4.

BLANKCK ERR aaaa dd

The program function (PROG key) displays this error message if the PROM fails the blank PROM test before programming begins. See the PROG function key description in Chapter 4.

DEVICE WILL NOT PROGRAM

The program function (PROG key) displays this error message if the preprogrammed bits in the PROM do not match the corresponding RAM bits, and the PROM cannot be programmed. See the PROG function key description in Chapter 4.

PROGRAM ERR aaaa dd

The program function (PROG key) displays this error message if an error occurs during programming of the PROM, and the PROM cannot be programmed to match the data in the RAM. See the PROG function key description in Chapter 4.

ADDRESS OUT OF RANGE

The address keyed into the address field is greater than the last valid address for the currently selected PROM device type. This error is also displayed during the FILL function when the ending address is less than the starting address or is out of range.

ILLEGAL BAUD RATE

The baud rate keyed in is not a standard baud rate within the range of 110 to 9600. Enter correct baud rate.

CHECK SUM ERROR

The check sum calculated for a record being read from a file does not match the check sum in the file.

ILLEGAL FILE TYPE

The data being downloaded from a file in the off-line mode does not match the required Intel 8080 hexadecimal file requirements.

On-Line Errors

The general form of an IUPS error message is as follows:

<error-group>--<specific-error>

The following <error-groups> are used in error messages: Syntax Error, General Error, IUP Error, Command Error, Disk Error, and Format Error. IUPS error messages are described below.

Syntax Errors

This type of error message indicates a syntax error in the entered command.

--SYNTAX ERROR--

--SYNTAX ERROR--ILLEGAL KEYWORD.

--SYNTAX ERROR--ILLEGAL PARAMETER.

--SYNTAX ERROR--LENGTH OUT OF RANGE.

--SYNTAX ERROR--MISSING ')'.

--SYNTAX ERROR--NON DIGIT CHARACTER.

--SYNTAX ERROR--ILLEGAL DELIMITER.

--SYNTAX ERROR--NEAR DESTINATION START ADDRESS.

--SYNTAX ERROR--KEYWORD 'TO' MISSING.

--SYNTAX ERROR--KEYWORD 'WITH' MISSING.

--SYNTAX ERROR--ILLEGAL SWITCH.

--SYNTAX ERROR--DIGIT SPECIFICATION ILLEGAL.

--SYNTAX ERROR--ILLEGAL CHARACTER.

--SYNTAX ERROR--SOURCE AND DESTINATION SAME.

--SYNTAX ERROR--ILLEGAL BASE.

--SYNTAX ERROR--ILLEGAL WORKFILE.

--SYNTAX ERROR--COMMAND FORM NOT ALLOWED.

--SYNTAX ERROR--ILLEGAL FILE TYPE.

--SYNTAX ERROR--ILLEGAL USE OF PATCH SWITCH.

GENERAL ERRORS

-GENERAL ERROR--START ADDRESS OUT OF RANGE.

-GENERAL ERROR--IUP-201 NOT INSTALLED.

-GENERAL ERROR--TYPE COMMAND NOT SPECIFIED.

-GENERAL ERROR--DESTINATION ADDRESS OUT OF RANGE.

-GENERAL ERROR--FINAL ADDRESS OUT OF RANGE.

-GENERAL ERROR--DATA OVERLAP IN THE FILE.

-GENERAL ERROR--ILLEGAL FILE TYPE SPECIFIED.

-GENERAL ERROR--FULL RANGE OF DATA NOT FOUND IN THE FILE.

IUP Errors

These error conditions are caused by the Universal Programmer or are due to communication failure.

--IUP ERROR----PROGRAMMING ERROR:
The location, expected data, and PROM data is also displayed for this error condition.

--IUP ERROR----CHECK-SUM ERROR.

--IUP ERROR----MODULE NOT INSTALLED.

--IUP ERROR----DEVICE INSTALLED IMPROPERLY.

--IUP ERROR----MODULE UPLOAD ERROR.

--IUP ERROR----HARDWARE FAILURE.

--IUP ERROR----COMMUNICATION FAILURE (RS-232 INTERFACE).

--IUP ERROR----DIFFERENT PROM TYPE SELECTED.

--IUP ERROR----COMMAND ABORTED.

--IUP ERROR----INTERFACE FORMAT ERROR.

--IUP ERROR----MOTHER BOARD FAILURE.

--IUP ERROR----KEY/EXPANSION RAM FAILURE.

--IUP ERROR----TYPE COMMAND SYNTAX.

Command Errors

These errors are related to the individual commands.

-COMMAND ERROR--OVERLAY TEST FAILED.

-COMMAND ERROR--VERIFY TEST FAILED.

-COMMAND ERROR--BLANKCHECK TEST FAILED.

Disk Errors

Disk errors refer to ISIS-II system errors.

Non-fatal errors are displayed in the following format:

--DISK ERROR---<ISIS-II Error Message>

Fatal errors return program control to ISIS-II. See Appendix C of the ISIS-II User's Guide.

Format Errors

These errors are related to the interactive FORMAT command.

--FORMAT ERROR--ILLEGAL DIGIT SPECIFICATION.

--FORMAT ERROR--NUMBER OF LOGICAL UNITS GREATER THAN 24.

--FORMAT ERROR--OUTPUT LOGICAL UNIT DOES NOT MATCH INPUT LOGICAL UNIT.

--FORMAT ERROR--OUTPUT LOGICAL UNIT EXCEEDS THE SPECIFIED RANGE.

--FORMAT ERROR--IN OUTPUT SPECIFICATION SYNTAX.

--FORMAT ERROR--KEYWORD 'TO' NOT SPECIFIED.

--FORMAT ERROR--ILLEGAL FILE SPECIFICATION.

--FORMAT ERROR--ILLEGAL LOGICAL UNIT.

--FORMAT ERROR--BLOCKSIZE SMALLER THAN LOGICAL UNIT.

--FORAMT ERROR--DATA RANGE IN SOURCE DEVICE TOO SMALL.

Cautions

These messages caution or warn about abnormal but non-fatal conditions.

----CAUTION----FILE MAY BE TOO LARGE FOR THE IUPS FILE HANDLER.

----CAUTION----PROGRAMMING THE FULL LENGTH REQUIRES MORE THAN ONE PROM.

----CAUTION----THE FILE STRUCTURE MAY BE INCOMPLETE.

APPENDIX B
HOST SERIAL COMMAND PROTOCOL

This appendix describes the command protocols for connecting the Universal Programmer to a host system other than an Intel Development System.

Universal Programmer to Host System Interface

The Universal Programmer communicates to the host system using an serial interface. The processor in the Universal Programmer automatically sets the receive and transmit baud rates to those of the host system. The automatic baud rate adjustment selects one of the following rates between 110 and 9600 baud: 110, 150, 300, 600, 1200, 2400, 4800, and 9600.

Communication Protocol

The Universal Programmer unit conforms to the E.I.A. Standard RS-232-C for data communicaton equipment. The RS-232 interface signals implemented in the hardware are shown in the following chart.

PIN	SIGNAL (From Host)	DESCRIPTION
1	CGND	Chassis Ground
2	XTD	Transmitter data
3	RCD	Receiver data
7	SGND	Signal ground

Transmit Formats

The types of information transferred via the serial interface are: control commands and data.

The control commands are sent from the host development system to the Universal Programmer to control PROM programming, resetting, etc. These commands must be generated by host software such as the IUPS software that runs on an Intellec Development System. A control command summary is shown in the following chart.

Data is also transmitted with some of the commands as decribed after the command summary.

Most of the commands are followed by a command status code response that is sent back to the host development system by the Universal Programmer.

ASCII CHARACTER	DESCRIPTION
A	ACKNOWLEDGE. The A command tells the Universal Programmer that the software running on the host system is ready to accept data. Used for handshaking.
B	UNPROGRAMMED PROM TEST. The B command causes the Universal Programmer to perform a test that determines if the PROM is unprogrammed (blank) and reflects the outcome of the test in the command status code returned to the host.
C	CLEAR. The C command resets any operation in progress and clears all Universal Programmer flags. This command resets the Personality Module and the Device Select Pointer. This command must be the first command issued to the Universal Programmer--it is used to calculate the baud rate.
D	DOWNLOAD. The D command sends data from the host system to the local RAM in the Universal Programmer. The user RAM begins at absolute location 8000H.
E	EXECUTE. The E command causes the Universal Programmer processor to transfer control to the Personality Module specific code at the processor's local RAM address 7020H.
F	FINISH. The F command terminates Read (R) or Program (P) commands. It clears the Personality Module control bits and turns off the Personality Module power supplies. This command can be used to re-establish the baud rate.
P	PROGRAM. The P command programs a specified PROM location with data supplied by the host.
R	READ PROM. The R command requests the Universal Programmer to transmit the content of specified PROM locations.
S	SELECT PROM TYPE. The S command selects the next PROM type that is valid for a particular Personality Module. Some Personality Modules are capable of programming more than one type of PROM. Each time this command is issued by the host development system, the Universal Programmer steps once through a list of valid PROMs for the Personality Module installed.
U	UPLOAD. The U command requests the Universal Programmer to transmit data from the local RAM to the host system buffer. The user RAM begins at absolute location 8000H.

V VERIFY. The V command requests the Universal Programmer to transmit the currently selected PROM type and PROM parameters to the host system.

Two different command formats are used to transmit information from the host to the Universal Programmer: commands with parameters and commands without parameters.

The commands with parameters are:

the Program (P) command
the Read (R) command
the Download (D) command
the Upload command (U)

The P and D commands require up to 67 additional bytes--three bytes of address and up to 64 bytes of data. The R and U commands require only four additional bytes--three bytes of address and a byte that indicates the number of data bytes (which can be up to 64) to be read (0 bytes is illegal).

Both command formats (with or without parameters) require a 20H start byte followed by a byte count prefix of the number of bytes to follow. The start byte and byte count prefix precedes the control command character (the ASCII character for the command as shown in the preceding chart).

Both command formats (with or without parameters) are followed by a two byte check sum suffix. The check sum is the two's complement of the binary sum of the previous bytes in the record being transmitted. This check sum does not include the start byte or the check sum itself.

Figure B-1 shows both host transmit formats.

Acknowledge Formats

The Universal Programmer acknowledges a host transmission (except for the host Acknowledge command and the host Execute command) by sending a block of information back to the host. The host software must be able to interpret this block of information correctly. For example, if a message is sent back (status code 21H), the host software must receive the message data and display the message on the CRT screen.

The information contained in the acknowledge block varies depending on the command requested by the host and whether or not any check sum errors were detected.

Every acknowledge block sent to the host contains a byte count prefix, a status byte, an echo of the command character, and a check sum suffix.

The echo of the command character is a repeat of the ASCII character for the command being acknowledged.

The check sum is the two's complement of the binary sum of the previous bytes in the record.

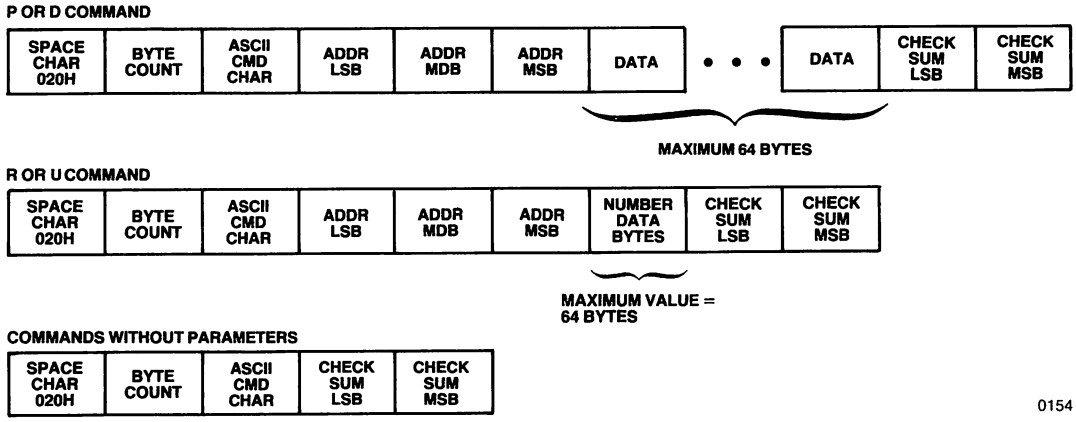


Figure B-1. Host Transmit Formats

Only the first byte of the block is initially sent to the host; the remainder is transmitted after the host responds with an Acknowledge command. If the host responds with any command besides Acknowledge, the entire block awaiting transmission in the Universal Programmer is lost, and the Universal Programmer responds to the new command issued by the host.

The status word is encoded as shown in the following chart. The codes are in hexadecimal.

CODE	DESCRIPTION
00	OPERATION COMPLETED, NO ERRORS. The 00H status byte indicates that the transmission was completed correctly for the iUP 200 Universal Programmer.
80	OPERATION COMPLETED, NO ERRORS. The 80H status byte indicates that the transmission was completed correctly for the iUP 201 Universal Programmer with the 16K URAM installed.
C0	OPERATION COMPLETED, NO ERRORS. The C0H status byte indicates that the transmission was completed correctly for the iUP 201 Universal Programmer with the 32K URAM installed.

The following entries are logically ORed with the appropriate value above (00H, 80H, or C0H) to produce the status byte:

01 PROGRAMMING ERROR. The 01H status byte occurs in response to the host Program (P) command if a location cannot be programmed with the specified data.

02 CHECK SUM ERROR. The 02H status byte indicates that the command record received by the Universal Programmer contained some error.

03 MODULE NOT INSTALLED. The 03H status byte indicates that an operation was attempted without a Personality Module installed.

04 DEVICE INSTALLED IMPROPERLY. The 04H status byte indicates that the device to be programmed or read is either installed improperly or is not present at all. NOTE: Not all Personality Modules are able to detect if a device is improperly installed.

05 UNPROGRAMMED ERROR. The 05H status byte is sent by the Universal Programmer in response to the UNPROGRAMMED PROM TEST command (B) if the PROM in question has been previously programmed.

06 MODULE UPLOAD ERROR. The 06H status byte indicates that the Personality Module specific code could not be loaded into address 7020H of the Universal Programmer memory, correctly.

07 ERROR. The 07H status byte indicates that a system error has occurred and the command should be aborted.

08 FORMAT ERROR. The 08H status byte indicates that the last transmission to the iUP did not meet the host to iUP command protocol.

09 to 0F UNDEFINED. These status bytes are not defined.

10 to 1F SELF-TEST ERROR. One of these status bytes can be sent in response to the CLEAR command (C) if a Universal Programmer hardware failure is detected. The assignment for these error codes follows:

10H	MOTHERBOARD FAILURE
11H	KEY/EXPANSION RAM FAILURE
12H	POWER SUPPLY FAILURE
13H to 1FH	UNDEFINED

20 REPEAT OPERATION. The 20H status byte requests that the host software transmit the full data buffer again. This feature can be used to program certain PROMs.

21 MESSAGE FOLLOWS. The 21H status byte indicates that a message will follow, which can be displayed on the host system CRT by the host software. Following this message another status message will be send in response to the command.

22 RESET DEAD-MAN TIMER. The 22H status byte indicates that the host system timer should be reset to prevent the host system from timing out.

23 to FF NOT USED.

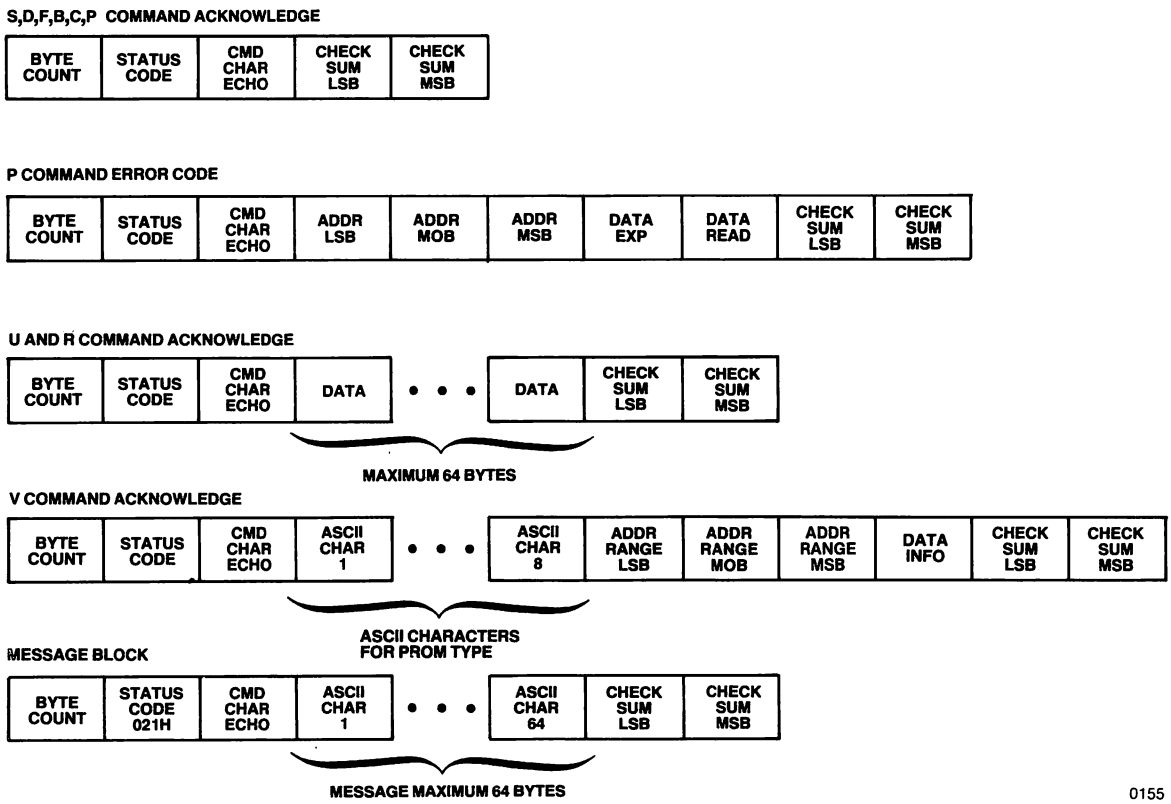


Figure B-2. Universal Programmer Acknowledge Formats

The information sent back to the host by the Universal Programmer depends on the command. The host commands:

UNPROGRAMMED PROM TEST (B)
CLEAR (C)
PROGRAM (P)
FINISH (F)
SELECT PROM TYPE (S)
DOWNLOAD (D)

require only the status byte and check sum.

The commands:

READ (R)
UPLOAD (U)
VERIFY PROM TYPE (V)
PROGRAM (P) with error condition

require additional information. All Universal Programmer acknowledge formats are shown in figure B-2.

The ADDRESS RANGE bytes shown in figure B-2 indicate with a 24-bit number the total available address space for a selected PROM type.

The DATA INFO byte in the V command acknowledge is defined as follows:

BIT 0 - BIT 5: These bits indicate the word length of the selected PROM type (e.g., 01000 = 8 bits length).

BIT 6: This bit indicates whether the selected PROM type is programmed high or low. A zero in this bit position indicates that the PROM type can be programmed only with lows (zeros). A one in this bit position indicates that only highs (ones) can be programmed into the PROM type.

BIT 7: NOT USED.

Data Link Format

All records passed between the host and the Universal Programmer are sent a byte at a time. Each byte received by the Universal Programmer must be prefixed by a start bit and suffixed with at least one stop bit. All bytes transmitted by the Universal Programmer are prefixed with one start bit and suffixed with one stop bit.

Figure B-3 shows the typical format for passing bytes over the serial data link.

On power-up or reset or when the serial data link baud rate is changed, the host system must transmit a Clear (C) command. The C command is used by the Universal Programmer processor to automatically set the baud rate. If the baud rate is changed after it is initially identified, three sequential check sum errors will occur before the new rate is identified. This means that four clear commands must be sent before the new rate is identified.

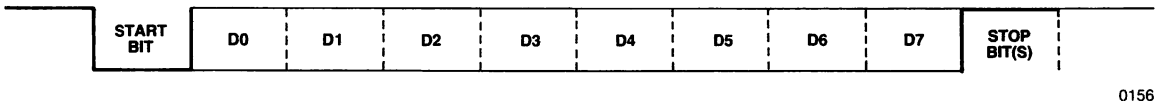


Figure B-3. Serial Byte Format

APPENDIX C
FILE FORMATS

The iUPS-200 software reads the following Intel file formats (it always writes files in the 286 format):

- 8080 Hex ASCII and 8080 Absolute Object. For a description of these formats, see the MCS 80/85 Absolute Object File Formats, An Intel Technical Specification, Order no. 9800183.
- 8086 Hex ASCII and 8086 Absolute Object. For a description of these formats, see the MCS-86 Absolute Object File Formats, An Intel Technical Specification, Order no. 9800821.
- 286 Absolute Object. This format is written by the IUPS software and is described below.

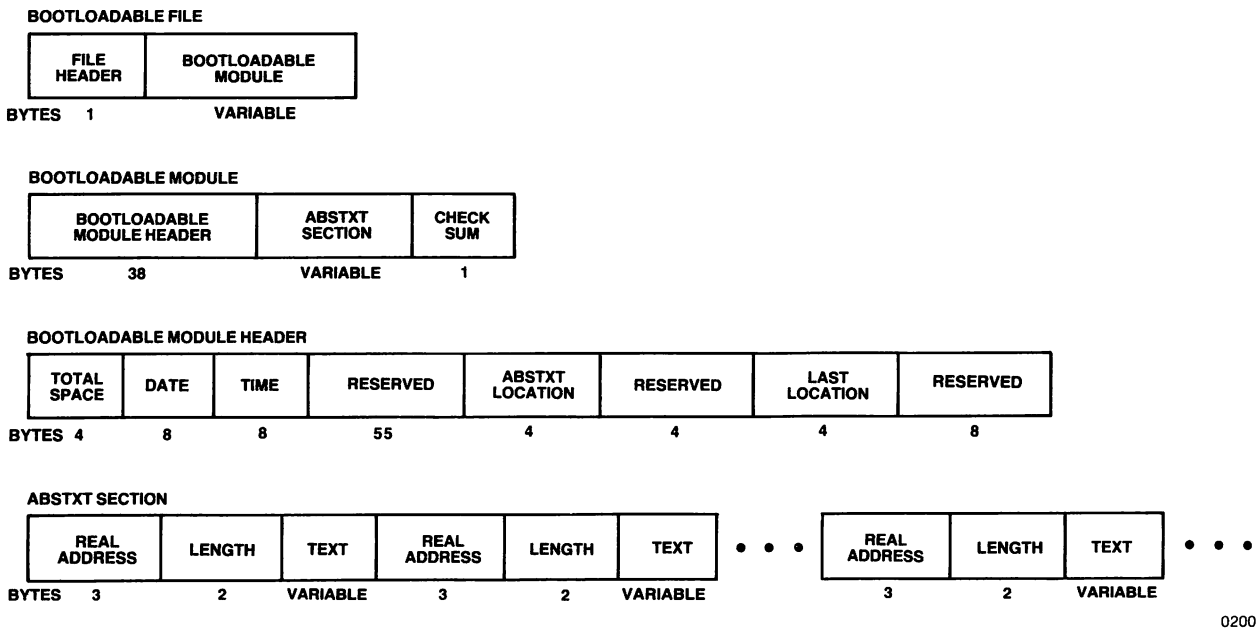


Figure C-1. 286 Absolute Bootloadable File Format

The 286 file format provides a consistent file format for saving data on diskette via the IUPS software, and to allow addresses to range up to 16Mbytes (FFFFFFH). The existing 8086 format only allows addresses to range up to 1Mbytes (FFFFFH). This flexibility permits IUPS to handle future processors which may have a larger address space than the 8086 processor.

A 286 bootloadable file is formatted as:

- A one-byte FILE HEADER followed by
- A variable number of BOOTLOADABLE MODULES

as shown in figure C-1.

Each BOOTLOADABEL MODULE in the file consists of:

- A 38-byte BOOTLOADABLE MODULE HEADER as described below.
- A variable length ABSTXT SECTION which specifies the initial main memory content. ABSTXT is a mnemonic for absolute text; text is the code or data to be loaded.
- A one-byte CHECK SUM which is calculated as the complement of the mod 256 sum of all the previous bytes in the module.

as shown in figure C-1.

Each BOOTLOADABLE MODULE HEADER consists of:

- A four-byte value TOTAL SPACE which indicates the minimum number of bytes in main memory needed to load the module.
- An eight-byte date area (not used)
- An eight-byte time area (not used)
- A 55 byte reserved area that is not used by IUPS.
- A four-byte ABSTXT LOCATION which is the offset in bytes of the start of the ABSTXT section from the start of the file.
- A four-byte reserved area that is not used by IUPS.
- A four-byte LAST LOCATION which is the offset in bytes of the last byte in the file from the start of the file.
- An eight-byte RESERVED field which is not currently used.

as shown in figure C-1.

Each ABSTXT SECTION consists of a repeating sequence of the following three fields:

- A three-byte REAL ADDRESS which is the starting address of the following text field in the memory device, i.e., the memory address at which to load the text.
- A two-byte LENGTH which specifies the length of the following text in bytes.
- A variable length TEXT field which is the code or data to be loaded into memory at the REAL ADDRESS specified.

as shown in figure C-1.

APPENDIX D
REFERENCE TABLES

Hexadecimal to Decimal Conversion

The following table is for hexadecimal to decimal and decimal to hexadecimal conversion. To find the decimal equivalent of a hexadecimal number, locate the hexadecimal number in the correct position and note the decimal equivalent. Add the decimal numbers.

To find the hexadecimal equivalent of a decimal number, locate the next lower decimal number in the table and note the hexadecimal number and its position. Subtract the decimal number shown in the table from the starting number. Find the difference in the table. Continue this process until there is no difference.

Table D-1. Hexadecimal to Decimal Conversion

MOST SIGNIFICANT BYTE				LEAST SIGNIFICANT BYTE			
DIGIT 4		DIGIT 3		DIGIT 2		DIGIT 1	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0
1	4 096	1	256	1	16	1	1
2	8 192	2	512	2	32	2	2
3	12 288	3	768	3	48	3	3
4	16 384	4	1 024	4	64	4	4
5	20 480	5	1 280	5	80	5	5
6	24 576	6	1 536	6	96	6	6
7	28 672	7	1 792	7	112	7	7
8	32 768	8	2 048	8	128	8	8
9	36 864	9	2 304	9	144	9	9
A	40 960	A	2 560	A	160	A	10
B	45 056	B	2 816	B	176	B	11
C	49 152	C	3 072	C	192	C	12
D	53 248	D	3 328	D	208	D	13
E	57 344	E	3 548	E	224	E	14
F	61 440	F	3 840	F	240	F	15
0123		4567		0123		4567	
BYTE				BYTE			

Table D-2. Base Conversions

DEC	BIN	HEX	OCT	DEC	BIN	HEX	OCT
0	0000 0000	00	000	51	0011 0011	33	063
1	0000 0001	01	001	52	0011 0100	34	064
2	0000 0010	02	002	53	0011 0101	35	065
3	0000 0011	03	003	54	0011 0110	36	066
4	0000 0100	04	004	55	0011 0111	37	067
5	0000 0101	05	005	56	0011 1000	38	070
6	0000 0110	06	006	57	0011 1001	39	071
7	0000 0111	07	007	58	0011 1010	3A	072
8	0000 1000	08	010	59	0011 1011	3B	073
9	0000 1001	09	011	60	0011 1100	3C	074
10	0000 1010	0A	012	61	0011 1101	3D	075
11	0000 1011	0B	013	62	0011 1110	3E	076
12	0000 1100	0C	014	63	0011 1111	3F	077
13	0000 1101	0D	015	64	0100 0000	40	100
14	0000 1110	0E	016	65	0100 0001	41	101
15	0000 1111	0F	017	66	0100 0010	42	102
16	0001 0000	10	020	67	0100 0011	43	103
17	0001 0001	11	021	68	0100 0100	44	104
18	0001 0010	12	022	69	0100 0101	45	105
19	0001 0011	13	023	70	0100 0110	46	106
20	0001 0100	14	024	71	0100 0111	47	107
21	0001 0101	15	025	72	0100 1000	48	110
22	0001 0110	16	026	73	0100 1001	49	111
23	0001 0111	17	027	74	0100 1010	4A	112
24	0001 1000	18	030	75	0100 1011	4B	113
25	0001 1001	19	031	76	0100 1100	4C	114
26	0001 1010	1A	032	77	0100 1101	4D	115
27	0001 1011	1B	033	78	0100 1110	4E	116
28	0001 1100	1C	034	79	0100 1111	4F	117
29	0001 1101	1D	035	80	0101 0000	50	120
30	0001 1110	1E	036	81	0101 0001	51	121
31	0001 1111	1F	037	82	0101 0010	52	122
32	0010 0000	20	040	83	0101 0011	53	123
33	0010 0001	21	041	84	0101 0100	54	124
34	0010 0010	22	042	85	0101 0101	55	125
35	0010 0011	23	043	86	0101 0110	56	126
36	0010 0100	24	044	87	0101 0111	57	127
37	0010 0101	25	045	88	0101 1000	58	130
38	0010 0110	26	046	89	0101 1001	59	131
39	0010 0111	27	047	90	0101 1010	5A	132
40	0010 1000	28	050	91	0101 1011	5B	133
41	0010 1001	29	051	92	0101 1100	5C	134
42	0010 1010	2A	052	93	0101 1101	5D	135
43	0010 1011	2B	053	94	0101 1110	5E	136
44	0010 1100	2C	054	95	0101 1111	5F	137
45	0010 1101	2D	055	96	0110 0000	60	140
46	0010 1110	2E	056	97	0110 0001	61	141
47	0010 1111	2F	057	98	0110 0010	62	142
48	0011 0000	30	060	99	0110 0011	63	143
49	0011 0001	31	061	100	0110 0100	64	144
50	0011 0010	32	062	101	0110 0101	65	145

Table D-2. Base Conversions (CON'T.)

DEC	BIN	HEX	OCT	DEC	BIN	HEX	OCT
102	0110 0110	66	146	153	1001 1001	99	231
103	0110 0111	67	147	154	1001 1010	9A	232
104	0110 1000	68	150	155	1001 1011	9B	233
105	0110 1001	69	151	156	1001 1100	9C	234
106	0110 1010	6A	152	157	1001 1101	9D	235
107	0110 1011	6B	153	158	1001 1110	9E	236
108	0110 1100	6C	154	159	1001 1111	9F	237
109	0110 1101	6D	155	160	1010 0000	A0	240
110	0110 1110	6E	156	161	1010 0001	A1	241
111	0110 1111	6F	157	162	1010 0010	A2	242
112	0111 0000	70	160	163	1010 0011	A3	243
113	0111 0001	71	161	164	1010 0100	A4	244
114	0111 0010	72	162	165	1010 0101	A5	245
115	0111 0011	73	163	166	1010 0110	A6	246
116	0111 0100	74	164	167	1010 0111	A7	247
117	0111 0101	75	165	168	1010 1000	A8	250
118	0111 0110	76	166	169	1010 1001	A9	251
119	0111 0111	77	167	170	1010 1010	AA	252
120	0111 1000	78	170	171	1010 1011	AB	253
121	0111 1001	79	171	172	1010 1100	AC	254
122	0111 1010	7A	172	173	1010 1101	AD	255
123	0111 1011	7B	173	174	1010 1110	AE	256
124	0111 1100	7C	174	175	1010 1111	AF	257
125	0111 1101	7D	175	176	1011 0000	B0	260
126	0111 1110	7E	176	177	1011 0001	B1	261
127	0111 1111	7F	177	178	1011 0010	B2	262
128	1000 0000	80	200	179	1011 0011	B3	263
129	1000 0001	81	201	180	1011 0100	B4	264
130	1000 0010	82	202	181	1011 0101	B5	265
131	1000 0011	83	203	182	1011 0110	B6	266
132	1000 0100	84	204	183	1011 0111	B7	267
133	1000 0101	85	205	184	1011 1000	B8	270
134	1000 0110	86	206	185	1011 1001	B9	271
135	1000 0111	87	207	186	1011 1010	BA	272
136	1000 1000	88	210	187	1011 1011	BB	273
137	1000 1001	89	211	188	1011 1100	BC	274
138	1000 1010	8A	212	189	1011 1101	BD	275
139	1000 1011	8B	213	190	1011 1110	BE	276
140	1000 1100	8C	214	191	1011 1111	BF	277
141	1000 1101	8D	215	192	1100 0000	C0	300
142	1000 1110	8E	216	193	1100 0001	C1	301
143	1000 1111	8F	217	194	1100 0010	C2	302
144	1001 0000	90	220	195	1100 0011	C3	303
145	1001 0001	91	221	196	1100 0100	C4	304
146	1001 0010	92	222	197	1100 0101	C5	305
147	1001 0011	93	223	198	1100 0110	C6	306
148	1001 0100	94	224	199	1100 0111	C7	307
149	1001 0101	95	225	200	1100 1000	C8	310
150	1001 0110	96	226	201	1100 1001	C9	311
151	1001 0111	97	227	202	1100 1010	CA	312
152	1001 1000	98	230	203	1100 1011	CB	313

Table D-2. Base Conversions (CON'T.)

DEC	BIN	HEX	OCT	DEC	BIN	HEX	OCT
204	1100 1100	CC	314	230	1110 0110	E6	346
205	1100 1101	CD	315	231	1110 0111	E7	347
206	1100 1110	CE	316	232	1110 1000	E8	350
207	1100 1111	CF	317	233	1110 1001	E9	351
208	1101 0000	D0	320	234	1110 1010	EA	352
209	1101 0001	D1	321	235	1110 1011	EB	353
210	1101 0010	D2	322	236	1110 1100	EC	354
211	1101 0011	D3	323	237	1110 1101	ED	355
212	1101 0100	D4	324	238	1110 1110	EE	356
213	1101 0101	D5	325	239	1110 1111	EF	357
214	1101 0110	D6	326	240	1111 0000	F0	360
215	1101 0111	D7	327	241	1111 0001	F1	361
216	1101 1000	D8	330	242	1111 0010	F2	362
217	1101 1001	D9	331	243	1111 0011	F3	363
218	1101 1010	DA	332	244	1111 0100	F4	364
219	1101 1011	DB	333	245	1111 0101	F5	365
220	1101 1100	DC	334	246	1111 0110	F6	366
221	1101 1101	DD	335	247	1111 0111	F7	367
222	1101 1110	DE	336	248	1111 1000	F8	370
223	1101 1111	DF	337	249	1111 1001	F9	371
224	1110 0000	E0	340	250	1111 1010	FA	372
225	1110 0001	E1	341	251	1111 1011	FB	373
226	1110 0010	E2	342	252	1111 1100	FC	374
227	1110 0011	E3	343	253	1111 1101	FD	375
228	1110 0100	E4	344	254	1111 1110	FE	376
229	1110 0101	E5	345	255	1111 1111	FF	377

Table D-3. Powers of Two

2^n	n
256	8
512	9
1 024	10
2 048	11
4 096	12
8 192	13
16 384	14
32 768	15
65 536	16
131 072	17
262 144	18
524 288	19
1 048 576	20
2 097 152	21
4 194 304	22
8 388 608	23
16 777 216	24

Table D-4. Conversion Between Between Powers of Two and Sixteen

$2^m = 16^n$
$2^0 = 16^0$
$2^4 = 16^1$
$2^8 = 16^2$
$2^{12} = 16^3$
$2^{16} = 16^4$
$2^{20} = 16^5$
$2^{24} = 16^6$
$2^{28} = 16^7$
$2^{32} = 16^8$
$2^{36} = 16^9$
$2^{40} = 16^{10}$
$2^{44} = 16^{11}$
$2^{48} = 16^{12}$
$2^{52} = 16^{13}$
$2^{56} = 16^{14}$
$2^{60} = 16^{15}$
$2^{64} = 16^{16}$

Table D-5. Powers of Sixteen

16^n	n
1	0
16	1
256	2
4 096	3
65 536	4
1 048 576	5
16 777 216	6
268 435 456	7
4 294 967 296	8
68 719 476 736	9
1 099 511 627 776	10
17 592 186 044 416	11
281 474 976 710 656	12
4 503 599 627 370 496	13
72 057 594 037 927 936	14
1 152 921 504 606 846 976	15

Table D-6. ASCII Code List

Decimal	Octal	Hexadecimal	Character	Decimal	Octal	Hexadecimal	Character
0	000	00	NUL	32	040	20	SP
1	001	01	SOH	33	041	21	!
2	002	02	STX	34	042	22	"
3	003	03	ETX	35	043	23	#
4	004	04	EOT	36	044	24	$
5	005	05	ENQ	37	045	25	%
6	006	06	ACK	38	046	26	&
7	007	07	BEL	39	047	27	'
8	010	08	BS	40	050	28	(
9	011	09	HT	41	050	29	)
10	012	0A	LF	42	052	2A	*
11	013	0B	VT	43	053	2B	+
12	014	0C	FF	44	054	2C	,
13	015	0D	CR	45	055	2D	-
14	016	0E	SO	46	056	2E	.
15	017	0F	SI	47	057	2F	/
16	020	10	DLE	48	060	30	0
17	021	11	DC1	49	061	31	1
18	022	12	DC2	50	062	32	2
19	023	13	DC3	51	063	33	3
20	024	14	DC4	52	064	34	4
21	025	15	NAK	53	065	35	5
22	026	16	SYN	54	066	36	6
23	027	17	ETB	55	067	37	7
24	030	18	CAN	56	070	38	8
25	031	19	EM	57	071	39	9
26	032	1A	SUB	58	072	3A	:
27	033	1B	ESC	59	073	3B	;
28	034	1C	FS	60	074	3C	<
29	035	1D	GS	61	075	3D	=
30	036	1E	RS	62	076	3E	>
31	037	1F	US	63	077	3F	?

Table D-6. ASCII Code List (CON'T)

Decimal	Octal	Hexadecimal	Character	Decimal	Octal	Hexadecimal	Character
64	100	40	@	96	140	60	'
65	101	41	A	97	141	61	a
66	102	42	B	98	142	62	b
67	103	43	C	99	143	63	c
68	104	44	D	100	144	64	d
69	105	45	E	101	145	65	e
70	106	46	F	102	146	66	f
71	107	47	G	103	147	67	g
72	100	48	H	104	150	68	h
73	101	49	I	105	151	69	i
74	102	4A	J	106	152	6A	j
75	103	4B	K	107	153	6B	k
76	104	4C	L	108	154	6C	l
77	105	4D	M	109	155	6D	m
78	106	4E	N	110	156	6E	n
79	107	4F	O	111	157	6F	o
80	100	50	P	112	160	70	p
81	101	51	Q	113	161	71	q
82	102	52	R	114	162	72	r
83	103	53	S	115	163	73	s
84	104	54	T	116	164	74	t
85	105	55	U	117	165	75	u
86	106	56	V	118	166	76	v
87	107	57	W	119	167	77	w
88	100	58	X	120	170	78	x
89	101	59	Y	121	171	79	y
90	102	5A	Z	122	172	7A	z
91	103	5B	[	123	173	7B	{
92	104	5C	\	124	174	7C	|
93	105	5D	]	125	175	7D	}
94	106	5E	^	126	176	7E	~
95	107	5F	_	127	177	7F	DEL

Table D-7. ASCII Control Code Definition

Abbreviation	Meaning	Decimal Code
NUL	NULL Character	0
SOH	Start of Heading	1
STX	Start of Text	2
ETX	End of Text	3
EOT	End of Transmission	4
ENQ	Enquiry	5
ACK	Acknowledge	6
BEL	Bell	7
BS	Backspace	8
HT	Horizontal Tabulation	9
LF	Line Feed	10
VT	Vertical Tabulation	11
FF	Form Feed	12
CR	Carriage Return	13
SO	Shift Out	14
SI	Shift In	15
DLE	Data Link Escape	16
DC1	Device Control 1	17
DC2	Device Control 2	18
DC3	Device Control 3	19
DC4	Device Control 4	20
NAK	Negative Acknowledge	21
SYN	Synchronous Idle	22
ETB	End of Transmission Block	23
CAN	Cancel	24
EM	End of Medium	25
SUB	Substitute	26
ESC	Escape	27
FS	File Separator	28
GS	Group Separator	29
RS	Record Separator	30
US	Unit Separaotr	31
SP	Space	32
DEL	Delete	127

Table D-8. ASCII Code in Binary

LSB \ MSB	0	1	2	3	4	5	6	7
	000	001	010	011	100	101	110	111
0 0000	NUL	DLE	SP	0	@	P		p
1 0001	SOH	DC1	!	1	A	Q	a	q
2 0010	STX	DC2	"	2	B	R	b	r
3 0011	ETX	DC3	#	3	C	S	c	s
4 0100	EOT	DC4	$	4	D	T	d	t
5 0101	ENQ	NAK	%	5	E	U	e	u
6 0110	ACK	SYN	&	6	F	V	f	v
7 0111	BEL	ETB	`	7	G	W	g	w
8 1000	BS	CAN	(	8	H	X	h	x
9 1001	HT	EM	)	9	I	Y	i	y
A 1010	LF	SUB	*	:	J	Z	j	z
B 1011	VT	ESC	+	;	K	[	k	{
C 1100	FF	FS	,	<	L	\	l	¦
D 1101	CR	GS	-	=	M	]	m	}
E 1110	SO	RS	.	>	N	^	n	~
F 1111	SI	VS	/	?	O	_	o	DEL

INDEX