intel®

iPPS PROM PROGRAMMING SOFTWARE USER'S GUIDE

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

Order Number: 164861-001

iPPS PROM PROGRAMMING SOFTWARE USER'S GUIDE

Order Number: 164861-001

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

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.

Intel Corporation makes no warranty for the use of its products and assumes no responsibility for any errors which may appear in this document nor does it make a commitment to update the information contained herein.

Intel retains the right to make changes to these specifications at any time, without notice.

Contact your local sales office to obtain the latest specifications before placing your order.

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

BITBUS	iLBX	iPDS	Plug-A-Bubble
COMMputer	im	iRMX	PROMPT
CREDIT	iMMX	iSBC	Promware
Data Pipeline	Insite	iSBX	QUEX
GENIUS	Intel	iSDM	QUEST
i	intel	iSXM	Ripplemode
i2ICE	intelBOS	Library Manager	RMX/80
ICE	Intelevision	MCS	RUPI
iCS	inteligent Identifier	Megachassis	Seamless
iDBP	inteligent Programming	MICROMAINFRAME	SOLO
iDIS	Intellec	MULTIBUS	SYSTEM 2000
	Intellink	MULTICHANNEL	UPI
	iOSP	MULTIMODULE	

MDS is an ordering code only and is not used as a product name or trademark. MDS® is a registered trademark of Mohawk Data Sciences Corporation.

*MULTIBUS is a patented Intel bus.

REV.	REVISION HISTORY	DATE
-001	Original Issue.	3/84

PREFACE

About This Manual

This manual describes how to use the Intel PROM programming software (iPPS) to store programs and data into programmable read-only memory (PROM) devices. It is for engineers and designers who are developing firmware for ROM-, PROM-, EPROM-, or E^2PROM-based systems.

The following describes the general contents of each chapter.

Chapter 1	Contains a general description of the the iPPS software.
Chapter 2	Describes each iPPS command.
Chapter 3	Provides a number of programming examples that illustrate using the iPPS software in typical firmware development applications.
Appendix A	Describes iPPS error conditions and error messages.
Appendix B	Contains information about the various disk file formats that the iPPS software reads and writes.
Appendix C	Contains reference tables for numeric conversion and ASCII codes.

Conventions Used In This Book

Throughout this book, the iUP-200A/201A universal programmer is called the universal programmer, the Intel personal development system is called the iPDS™ system, and the Intel PROM programming software is called the iPPS software.

A section of text introduced by the symbol

NOTE

emphasizes comments with special significance.

A section of text introduced by the symbol

CAUTION

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

CONTENTS

FIGURES

TABLES

CHAPTER 1
OPERATING INFORMATION

This chapter describes the operation of the Intel PROM programming software (iPPS), which runs under the ISIS operating system, and the form of the iPPS commands. Read this chapter before using the iPPS commands described in Chapter 2.

Compatible Hosts

The iPPS software runs on Intel hosts (Intellec® 800 and Intellec Series II, III, and IV) and the Intel personal development system (iPDS). Refer to the *iUP-200A /201A Universal Programmer User's Guide* for information on connecting Intel hosts to the universal programmer; refer to the *iPDS™ Personal Development System User's Guide* for information on using the iPDS system.

iPPS File Configuration

The following files are necessary for normal iPPS operation:

IPPS	The main object file that is loaded under the ISIS operating system.
IPPS.ERR	The error message file that contains the various iPPS error messages.
IPPS.HLP	The help message file that explains the various iPPS commands.
IPPS.OV0	An overlay file that contains executable code which is loaded during normal iPPS operation.
IPPS.OV1	An overlay file that contains executable code which is loaded only during operation of the FORMAT command.
IPPS.OV2	An overlay file that contains executable code which is loaded only during operation of the SUBSTITUTE command (iPDS system only).

iPPS Software Initialization

The following two sections discuss the two methods of invoking the iPPS software: command lines and submit files. The conventions used to describe commands are covered in the Notation for Syntax Description section in this chapter.

Command Line Invocation

Invoke the iPPS software under the ISIS operating system (Intellec 800, Series II and III, version V3.4 and later; the iPDS system, ISIS.PDS version 1.0 and later; Series IV, version V1.0 and later) with the following syntax:

[:F*n*:]IPPS [CHANNEL(*n*)]

FIGURES

TABLES

CHAPTER 1
OPERATING INFORMATION

This chapter describes the operation of the Intel PROM programming software (iPPS), which runs under the ISIS operating system, and the form of the iPPS commands. Read this chapter before using the iPPS commands described in Chapter 2.

Compatible Hosts

The iPPS software runs on Intel hosts (Intellec® 800 and Intellec Series II, III, and IV) and the Intel personal development system (iPDS). Refer to the *iUP-200A /201A Universal Programmer User's Guide* for information on connecting Intel hosts to the universal programmer; refer to the *iPDS™ Personal Development System User's Guide* for information on using the iPDS system.

iPPS File Configuration

The following files are necessary for normal iPPS operation:

IPPS	The main object file that is loaded under the ISIS operating system.
IPPS.ERR	The error message file that contains the various iPPS error messages.
IPPS.HLP	The help message file that explains the various iPPS commands.
IPPS.OV0	An overlay file that contains executable code which is loaded during normal iPPS operation.
IPPS.OV1	An overlay file that contains executable code which is loaded only during operation of the FORMAT command.
IPPS.OV2	An overlay file that contains executable code which is loaded only during operation of the SUBSTITUTE command (iPDS system only).

iPPS Software Initialization

The following two sections discuss the two methods of invoking the iPPS software: command lines and submit files. The conventions used to describe commands are covered in the Notation for Syntax Description section in this chapter.

Command Line Invocation

Invoke the iPPS software under the ISIS operating system (Intellec 800, Series II and III, version V3.4 and later; the iPDS system, ISIS.PDS version 1.0 and later; Series IV, version V1.0 and later) with the following syntax:

[:F*n*:]IPPS [CHANNEL(*n*)]

The :F*n*: specifies the drive on which the iPPS files are located, and CHANNEL(*n*) specifies the serial channel (*n* is 1 or 2). For example, if the iPPS files are located on disk drive 1 and you have configured serial channel 1, then the iPPS software is invoked by entering the following command:

```
:F1:IPPS CHANNEL(1)
```

If you do not specify a channel or specify a channel incorrectly, the iPPS software assigns the default channel. The default channel is channel 1 unless channel 1 is being used for the system console, in which case channel 2 is the default channel. You get the following message when the iPPS software overrides the channel specified in the invocation and assigns the default channel:

```
---- CAUTION ---- DEFAULTING TO SERIAL CHANNEL n
```

NOTE

The channel option is not available on the iPDS system. The *iUP-200A/201A Universal Programmer User's Guide* contains more information on the serial channels.

After successful invocation, the iPPS software looks for its files (IPPS.ERR, IPPS.HLP, IPPS.OV0, and IPPS.OV1/IPPS.OV2) on drive 1.

When you enter the IPPS command, the ISIS operating system loads and executes the iPPS software. After the software is properly initialized, it displays the following sign-on message:

```
INTEL PROM PROGRAMMING SOFTWARE Vv.r
COPYRIGHT 1980,1981,1982,1983,1984 INTEL CORP.
PPS>
```

The V*v.r* is the current version (*v*) and revision (*r*) numbers 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 the ISIS operating system.

Invocation Using a Submit File

The iPPS software can run under the control of a submit file. SUBMIT is an ISIS command that allows a disk text file to be used as input for further ISIS commands or as command inputs to utilities running under the ISIS operating system. Thus, a submit file can contain the ISIS command line to invoke the iPPS software and then a sequence of commands for the iPPS software itself. Note that you cannot use the SUBSTITUTE and ALTER commands in 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 an *ISIS-XX User's Guide.* In order to invoke and control the iPPS software using a submit file, the SUBMIT command itself must be available on the system disk. Note that you must not name your submit file IPPS.CSD if you are on a network because the SUBMIT command will then create a file named IPPS.TMP which is reserved for the iPPS temporary workfiles.

iPPS General Operation

The following sections discuss the general operation of the iPPS software.

Major Functions

The following are the major functions of the iPPS software:

- To read and write data on any of the four major storage devices (PROM, buffer, file, and URAM).
- To manipulate data in the buffer.
- To display or print the data stored in any of the major storage devices on the terminal, local line printer, or network spooled printer.
- To check for an unprogrammed PROM and to allow the data in the buffer to be overlaid 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 (i.e., select different PROM word sizes and use interleaving techniques).
- To lock selected PROM devices from unauthorized access.
- To accommodate 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. Refer to Appendix B 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, and URAM. These devices are defined in the following four sections.

PROM Device

The PROM device is plugged into a socket on the personality module installed in the universal programmer/iPDS system. The iPPS software does not recognize the PROM device until you enter a TYPE command. The TYPE command automatically sets the appropriate buffer size according to the size of the PROM device specified. Refer to the TYPE command description in Chapter 2 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 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 expression determines the buffer end address:

(*buffer start address*) + (*buffer size* − 1)

The TYPE command affects both the size and location of the buffer. For example, the TYPE command always resets the buffer start address to 0. The most recent TYPE command determines the size of the buffer. Refer to the TYPE command description in Chapter 2 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 using the TYPE command.
- It is initialized to 8K when the iPPS software is invoked.

You need a virtual buffer 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 lets you specify the storage device on which the virtual buffer is created; in all other respects, the virtual buffer is transparent.

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

```
WORKFILES (:FX:)? ; X =
```

This prompt requests a number indicating which disk drive 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 drive. The default work files drive then remains the same until the next WORKFILES command. Refer to the WORKFILES command in Chapter 2 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.

File Device

The file device is an ISIS file on disk. It is specified within iPPS commands, using the following ISIS file specification format:

[:device:]filename[.extension]

The data stored in the disk file is in one of the following Intel absolute formats: 8080 hex, 8080 object, 8086 hex, 8086 object, or 286 object. The iPPS software can read any of these formats as input but writes data to a file in 8080 object, 8086 object, or 286 object format only. (Refer to Appendix B for more information on 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., the file is 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 was selected). If the destination device is the buffer, the nonexistent sections in the file do not overwrite the corresponding sections 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 ignores 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 deleted.

URAM Device

The user RAM (URAM) is available only with iUP-201A models of the universal programmer (not available with the iPDS system). It consists of a RAM buffer that resides locally in the iUP-201A universal programmer which the iPPS software can write to or read from. This RAM is generally used for off-line data manipulation, but it is accessible for certain on-line operations. During on-line operations, iPPS software accesses the URAM as a device.

iPPS operations can access all of the URAM or portions of it. The default URAM address boundaries are from 0 to *URAM size* − 1. The URAM size is 32K bytes. The data in the URAM is useful to the iPPS software only if the PROM to be accessed off-line is 32K bytes or smaller.

Command Entry

iPPS commands consist of a command keyword that identifies the command, followed by other keywords and associated parameters which are the arguments of the command. Arguments are separated from each other by at least one space (extra spaces are ignored).

You must end a command line with a carriage return, <CR>. The command is not executed until you press the carriage return. If you press ESC the command is aborted. You can enter a new command when iPPS software displays the prompt PPS>.

Once entered, a command line is verified for correct syntax and executed. If a syntax error is detected, the following error message is displayed:

--SYNTAX ERROR- -*specific error.*

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

Input lines do not directly correspond to an 80-character screen line. If the echoed input line exceeds the screen display line, it wraps around to the next screen line. You can continue command inputs which are too long to fit in one input line on successive lines. To continue a line, enter an ampersand (&) as the last non-blank character before 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).

You can enter complete iPPS keywords or 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.

The iPPS software 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 letter identifier to specify the base (e.g., 11111111Y, 377Q, 255T, FFH). An explicit number base identifier overrides the default number base which is initially hexadecimal.

To illustrate these command entry features, the following examples show different ways to enter the same COPY command to read a 2K PROM into the buffer. User input is in bold type to differentiate it from iPPS terminal output. All commands are terminated with a carriage return.

COPY PROM(0,7FF) TO BUFFER(0)

C P(0,11111111111Y) TO B

C
FROM? **PR(0,7FF)**
TO? **BU**

copy pr(0,7FF)
TO? **b**

C P T B

Command Entry Editing

You can edit iPPS commands using the following ISIS command line editing features. Control characters are entered by holding down the control key (CNTL) while the character is entered.

<RUBOUT>	Deletes the previously-entered character.
<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. The line-editing buffer is cleared and command input is restarted on the next screen line.
<CR>	(carriage return) Ends command input and submits the entire contents of the line-editing buffer to the iPPS software as a command.

In addition, the iPPS software provides the ALTER command which lets you edit and re-execute the most recently entered iPPS command.

The Form of iPPS Commands

The following five sections describe the iPPS command format.

Notation For Syntax Description

Brackets, braces, italic face, and underscores are used in this manual to define the syntax of iPPS commands; they are not recognized as valid entries by the iPPS software. 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 entered. All punctuation other than brackets, braces, and underscores must be entered as shown. The syntax descriptions use the following notational conventions:

UPPERCASE	Denotes specific words that must be entered as shown (although you can enter the words in either uppercase or lowercase).
[ ]	(brackets) Indicates that all the parameters and punctuation enclosed are optional.
{ }	(braces) Encloses a list of parameters from which you must select one and only one of the parameters. The different choices are listed vertically on separate lines.
italic	Denotes an entry for which you must supply a specific value to replace the generic term. Generic terms that are used throughout the syntax descriptions for iPPS commands are described in the Special iPPS Syntax Terms section.
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 as follows:

startaddr	Starting address of a section of source data. Its values can range from 0 to 16777215 (i.e., 0 to $2^{24}-1$).
endaddr	Ending address of a section of source data. Its values can range from 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*. Refer to the following description of *length*.
destaddr	Starting destination address for a section of data that is to be written. Its values can range from 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*. Refer to the following description of *offset*.
length	The length (in bytes) of a section of data. Its values can range from 1 to 16777216 (i.e., 1 to 2^{24}).
offset	An offset value used to specify the starting destination 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 range from 0 to 16777215 (i.e., 0 to $2^{24}-1$).

file The standard ISIS file specification of the form [:*device*:]*filename*[.*extension*]. (The *ISIS-XX User's Guide* contains more information.) The iPPS software accepts any *filename*. However, if neither the device nor the extension is specified, the following *filenames* are invalid:

B	P	IPPS.BUF
BU	PR	IPPS.TMP
BUF	PRO	
BUFF	PROM	
BUFFE		
BUFFER		

In addition, you cannot use the following file names on the universal programmer (although they are valid for the iPDS system):

U
UR
URA
URAM

Though it is not recommended, you can create a file with any of these names by always specifying it with its device (e.g., :F0:BUFFER). The iPPS software recognizes :F0:BUFFER as a file name that is distinct from BUFFER.

General Command Syntax

Most iPPS commands conform to the following general syntax:

cmd source [(*startaddr*[,*endaddr*])]

prep dest [(*destaddr*)] [*switch*]

The generic terms have the following meanings:

cmd Specifies the iPPS command keyword for the command to be executed. You can enter complete iPPS keywords or any unique abbreviation (only the first character is required).

source Specifies the source device as any one of the four devices that the iPPS software recognizes: PROM, buffer, file, or URAM.

prep Represents an appropriate preposition (WITH or TO) which syntactically separates the source and destination portions of the command input. You need to use only the first character of these prepositions. You must enter at least one leading and one trailing space to delimit these prepositions.

dest Specifies the destination as any one of the four recognized devices; however, *dest* cannot be the same as *source*. An error message is displayed if you try to do this.

switch Specifies any of the five valid command switches discussed in the Command Switches 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.

Command Switches

You can specify command switches 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 lets you change the base switch and file switch defaults. You cannot change the defaults for the other switches.

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

Base switch

Determines the number base to be used for the duration of the command. Following are the possible single-character entries:

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

You can specify only one base switch per command. The default for this switch is the default number base set by the most recent INITIALIZE command. Hexadecimal is the default if you did not enter a base with the INITIALIZE command.

File switch

Specifies the format of the file in which the data is stored. The possible entries are:

80	- 8080 absolute object or 8080 hexadecimal ASCII
86	- 8086 absolute object or 8086 hexadecimal ASCII
286	- 286 absolute object

These file formats are described in more detail in Appendix B. The default for this switch is the currently selected default file format. Refer to the INITIALIZE command description in Chapter 2 for more details. If you did not enter a file switch with the INITIALIZE command, 286 is the default.

Data switch

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

F

F specifies false; when you do not specify F, the iPPS software defaults to true (uncomplemented).

Patch switch

Specifies that the iPPS software can read patched object files. The following is the only allowable entry:

P

Certain object program files can contain sections of object code with overlapping addresses. When the iPPS software 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 the iPPS software to read files that have overlapping addresses. In this case, the sections of data with overlapping addresses are written over one another. This switch should be used for all object program files that have been patched.

If you specify the patch switch, file read commands may be slower because the iPPS software scans the entire file to find any possible patch blocks.

If you do not specify P, the iPPS software returns an error when it finds overlapping addresses.

Clear switch — Specifies that the buffer is cleared before you copy a file to it. The following is the only allowable entry:

C

When you do not specify C, the iPPS software defaults to not clear.

Command Defaults

The iPPS software assumes default values for various command parameter entries. The command switches have defaults too. Following are the input defaults:

Base switch — The default for the base switch is initially hexadecimal (H), but you can change it with the INITIALIZE command. The current default base can be overridden for a numeric entry by appending a single letter identifier to specify the base for that number. Thus, 11111111Y, 377Q, 255T, and 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 initially 286, but you can change it with the INITIALIZE command.

Data switch — The default for the data switch is 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 OFF. Entering P for the patch switch changes the switch to ON for the duration of the specific file read command.

Clear Switch — The default for the clear switch is OFF (not clear). Entering C for the clear switch clears the buffer before a copy from a file.

file — The only default associated with *file* is a default storage drive. If you do not specify a drive device, :F0: is assumed.

Command defaults associated with the source and destination devices are shown in Table 1-1. Note that some commands require only a source device (e.g., the DISPLAY command).

Summary of Commands

The 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 six groups: program control group, utility group, buffer group, formatting group, copy group, and security group. Chapter 2 describes the commands in detail.

Program Control Group

The program control commands control iPPS program execution in various ways, as follows:

EXIT	Exits the iPPS software and returns control to the ISIS operating system.
<ESC>	Terminates current command.
REPEAT	Re-executes previous command.
ALTER	Allows edit and re-execution of previous command.

Utility Group

The utility commands display data, help, and status information and set default values, as follows:

DISPLAY	Displays PROM, buffer, or file data on the terminal.
PRINT	Prints PROM, buffer, or file data on the local printer.
QUEUE	Prints PROM, buffer, or file data on the network spooled printer (not available on the iPDS system).
HELP	Displays help information.
MAP	Displays buffer structure and status.
BLANKCHECK	Checks for unprogrammed PROMs.
OVERLAY	Checks if the non-blank PROM device can be programmed.
TYPE	Selects PROM type.
INITIALIZE	Initializes default number base and default file type.
WORKFILES	Specifies the drive device for temporary work files.

Table 1-1 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. The iPPS software 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) Refer to note (5) for the 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 32K. If a *file* to URAM operation is limited by the URAM size, the default *endaddr* is *startaddr* + *URAM size* − 1.

(8) Refer to note (7) for the meaning of URAM size. If a buffer to URAM operation is limited by the URAM size, the default *endaddr* is *startaddr* + *URAM size* − 1.

Buffer Group

Following are the buffer commands which edit, modify, and verify data in the buffer:

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

Formatting Group

The data formatting command formats and rearranges data from the PROM, buffer, or file devices.

FORMAT	Interactively formats buffer, PROM, or file data and places the result in a work file.

Copy Group

The copy group contains variations of the general-purpose COPY command. The following commands let you copy 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 iPPS commands transfer data to and from the universal programmer local RAM (URAM) (not available on the iPDS system):

COPY (file to URAM)	Loads disk file data into local URAM.
COPY (URAM to file)	Stores local URAM data into a disk file.
COPY (buffer to URAM)	Loads buffer data into local URAM.
COPY (URAM to buffer)	Loads local URAM data into buffer.

Security Group

The following security group command programs selected devices to prevent unauthorized access:

KEYLOCK	Interactively prompts for any related parameters and locks the device from access.

ALTER

Edits and re-executes previous command

Syntax

ALTER

Where:

>	moves the cursor right one character position.
<	moves the cursor left one character position.
<RUBOUT>	deletes the last-entered character.
<CTRL-I>	(insert) opens one empty character position in the line immediately before the character the cursor was under when you entered <CTRL-I>. The cursor is placed under the new empty position, and any remaining characters in the line are shifted right one character position. You can then insert a new character which will occupy the opened character position. Any subsequent characters entered overwrite the existing characters in the line. Thus, you must enter <CTRL-I> before each consecutive character that is inserted to retain the remainder of the line.
<CTRL-D>	(delete) deletes the character currently under the cursor. Any remaining characters in the line are shifted left one character position to fill the deleted character position.
<CR> or <ESC>	tells the ALTER command to exit the command line edit mode. After you enter a <CR> or <ESC> (regardless of cursor position), the edited command line is re-displayed followed by a Y/N prompt. Entering N returns you to the iPPS software without executing the newly edited command line. Entering Y executes the edited command line.

Operation

The ALTER command lets you edit the previously entered command and then optionally execute the edited command. When you enter the ALTER command, the most recently entered iPPS command line is displayed on the terminal with the cursor blinking under the left-most character of the text line. You can then enter a new line over the old line or use the keys described in the syntax section to edit the line. (Do not enter a carriage return after these single-character ALTER commands.)

Examples

The following COPY command was entered with a misspelled keyword:

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

The following example illustrates using the ALTER command to correct the misspelled command and re-issue it to the iPPS software:

```
ALTER
COPPY PROM TO BUFFER
```

Using the > command, move the cursor right two characters to the first P:

```
COPPY PROM TO BUFFER
```

Enter <CTRL-D> to delete the extra P and then enter <CR> to indicate that editing is complete:

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

The iPPS software executes the corrected COPY command immediately after you enter Y.

BLANKCHECK

Checks for
unprogrammed PROM

Syntax

BLANKCHECK [PROM (*startaddr*[,*endaddr*])]

Where:

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 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.

Operation

The BLANKCHECK command tests the PROM device in the personality module socket to determine if the PROM is blank (i.e., in the unprogrammed state for that particular device, either all logic ones or all logic zeros). The BLANKCHECK command cannot be executed unless a prior TYPE command was executed to select a PROM device type.

If you enter the command keyword without parameters, the test is performed on the full PROM. A part of the PROM can be checked by specifying the parameters. If the PROM device is unprogrammed, the iPPS software responds with the following message:

```
PROM SEGMENT BLANK
```

If the PROM device is programmed, the iPPS software responds with the following message:

```
-COMMAND ERROR- -BLANKCHECK TEST FAILED.
```

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 be programmed. Valid abbreviations are used for the keywords.

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

The following command uses the shortest valid abbreviation for the command:

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

In this case, the PROM device tested was blank.

COPY

Transfers data

Syntax

COPY *source* [(*startaddr*[,*endaddr*])]

TO *dest* [(*destaddr*)] [*switch*]

Where:

source> specifies the source device as any one of the four devices recognized by the iPPS software: PROM, buffer, file, or URAM.

dest specifies the destination as any one of the four recognized devices; however, *dest* cannot be the same as *source*. An error message is displayed if *dest* and *source* are the same.

The iPPS software does not accept COPY operations between URAM and PROM.

switch specifies any (or all) of the three valid command switches (C, F, or P) discussed in Chapter 1. Some of the command switches 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.

Operation

The COPY command is a general 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 in the following sections.

CAUTION

You can use the file switch to reformat a file to a smaller address format (i.e., you can reformat a 286 format to an 80 or 86 format and an 86 format to an 80 format). Before choosing a smaller file switch, verify that no addresses exceed the range of the output file. Any addresses exceeding the range of the output file are truncated. The maximum allowable address for each format is as follows:

80 - 64K
86 - 1M
286 - 16M

The terms *startaddr*, *endaddr*, and *destaddr* are described in Chapter 1, along with their defaults. The specific COPY command descriptions contain more details about these parameters.

A check-sum is displayed at the completion of every copy command. The check-sum 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 cumulative sum are ignored during summation. The final check-sum is produced by calculating the 2's complement of the final cumulative sum.

COPY

File to PROM

Syntax

COPY *file* [(*startaddr*[,*endaddr*])]

TO PROM [(*destaddr*)] [F] [{ 80 / 86 / 286 }] [P]

Where:

file specifies the standard ISIS 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, it defaults to the lowest address encountered while reading in the file. Omitting *startaddr* 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 specified as a length, L*length*. The *endaddr* is then *startaddr* + *length* − 1. If *endaddr* is not specified, it defaults to the highest address encountered while reading in the file.

destaddr specifies the start address for the destination data in the PROM device. This parameter can be 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* 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, 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 is displayed and the COPY command is aborted.

-GENERAL ERROR- -DATA OVERLAP IN THE FILE.

Operation

The COPY file to PROM command lets you program a PROM directly with data from a disk file. Programming directly from a file eliminates the intermediate step of copying the data from the file to the buffer. If you must modify the data in the file before programming the PROM, use the COPY file to buffer command.

You must execute the TYPE command before invoking this command.

The COPY file to PROM command lets you program multiple PROM devices by prompting for the next device after the programming of each device 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 are programmed. Note that the OVERLAY operation is not performed by this command.

When an address range is 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, you are prompted to load subsequent PROM devices into the programming socket. The PROMs 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.
- You press the <ESC> key to abort the operation.

The COPY command may take several minutes to copy to a PROM. If no errors occur, the Intellec development system/iPDS system beeps when it completes the COPY command.

Examples

In the following example, a PROM is programmed with data from file CODE4.IUP on drive 1. 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 following 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*)] [{ 80 / 86 / 286 }] [F]

Where:

startaddr specifies the start address of the section of PROM data to be copied to a file. If *startaddr* is not entered, it defaults to 0. Omitting *startaddr* 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 specified as a length, L*length*. The *endaddr* is then *startaddr* + *length* − 1. If *endaddr* is not specified, it defaults to PROM size − 1. The PROM size is determined by the most recent TYPE command.

file specifies the standard ISIS file name for the file to be written.

destaddr specifies the start address of the destination data in the iPPS file. 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.

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

[F] is the optional data switch which complements the data before it is written. If you do not specify a data switch, the data is written into the file in its uncomplemented form.

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, you are prompted before the existing file is overwritten. You must execute the TYPE command before invoking this command.

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 1.

```
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 you can use lowercase characters.

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

If you do not specify the PROM keyword or the file, the iPPS software prompts for the missing items. In the next example, an error was made in entering the source PROM address range (an extra F was entered in the address specification). This caused the address range to exceed the address range for the specified PROM device type.

The following example shows how you can use the ALTER command to correct and restart a command. (Refer to the ALTER command description in this chapter for more information.) In the following example, the ALTER command shows the previous erroneous command and lets you edit it. Advance the cursor to one of the F's and press <CTRL-D> to delete it. Press <CR> to indicate that editing is complete. ALTER then shows the edited command and prompts for execution with -Y/N?. As shown, press Y to execute the edited command.

```
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]

Where:

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, it defaults to the buffer start address. Omitting *startaddr* 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 specified as a length, L*length*. The *endaddr* is then *startaddr* + *length* − 1. If *endaddr* is not specified, 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, an error message is displayed.

destaddr specifies the 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 0FFFFFFH (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.

Operation

The COPY buffer to PROM command programs the PROM device with the specified data in the buffer. You must execute the TYPE command before using this command.

This command automatically performs the BLANKCHECK operation to verify that the PROM device is blank. If the PROM device is not blank, 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 are programmed.

The COPY command may take several minutes to copy to a PROM. If no errors occur, the Intellec development system/iPDS system beeps when it completes the COPY command.

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 switch 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 you do not specify the buffer or PROM, the iPPS software prompts for the missing keywords. Note that the next example tries to program a non-blank PROM device, and the iPPS software prompts for an OVERLAY test. Refer to 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]

Where:

startaddr specifies the start address of the section of data in the PROM to be copied into the buffer. If *startaddr* is not entered, it defaults to 0. Omitting *startaddr* 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 specified as a length, L*length*. The *endaddr* is then *startaddr* + *length* − 1. If *endaddr* is not specified, it defaults to *PROM size* − 1. PROM size is determined by the most recent TYPE command.

destaddr specifies the 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 you do not specify a data switch, the data is written into the buffer in its uncomplemented form.

Operation

The COPY PROM to buffer command reads data from the PROM device into the buffer. Data stored in the buffer before execution of this command is overwritten. You must execute the TYPE command before invoking this command.

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 you do not specify the source or destination devices, the iPPS software prompts for the missing keywords. The following 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*)] [{ 80 / 86 / 286 }] [F]

Where:

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. Omitting *startaddr* 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 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 expression *buffer start address* + *buffer size* − 1.

file specifies the standard ISIS file name for the file to be written.

destaddr specifies the start address of the destination data in the iPPS file. This parameter can be 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 0FFFFFFH (16777215T), an error message is displayed. If *destaddr* is not specified, it defaults to 0.

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

[F] is the optional data switch which complements the data before it is written. If you do not specify a data switch, the data is written into the file in its uncomplemented form.

Operation

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

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 entire complemented contents of the buffer in file CODE3.IUP on drive 1.

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

No destination is specified in the following command, so the iPPS software prompts for the file name.

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

If you do not specify a source, the iPPS software prompts for that, too. In the following example, the data in the buffer starting at address 400H and ending at the buffer end address 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] [C]

Where:

file specifies the standard ISIS 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 you do not enter *startaddr*, it defaults to the lowest address encountered while reading in the file. Omitting *startaddr* implies that the *endaddr* default is also assumed. Note that the iPPS software must perform an extra pass of the file to determine the defaults for *startaddr* or *endaddr*.

endaddr specifies the end address for the section of file data to be copied. This parameter can be specified as a length, L*length*. The *endaddr* is then *startaddr* + *length* − 1. If *endaddr* − *startaddr* + 1 exceeds the current buffer size, an error message is displayed. If you do not specify *endaddr*, it defaults to the smaller of the following: the file's highest address or *startaddr* + *buffer size* − 1. This ensures that the address range never exceeds the buffer size.

destaddr specifies the start address for the destination data in the iPPS buffer device. This parameter can be 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 you do not specify *destaddr*, it defaults to 0.

[F] is the optional data switch which complements the data before it is written. If you do not specify a data switch, the data is written in its uncomplemented form.

{ 80 | 86 | 286 } is the optional file format switch selection. If you do not specify this switch, 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 you do not specify this switch and a patched file is read, the following error message is displayed and the COPY command is aborted.

-GENERAL ERROR- -DATA OVERLAP IN THE FILE.

[C] is the optional buffer clear switch which clears the buffer before a copy from a file. If you do not specify this switch, the buffer is not cleared.

Operation

The COPY file to buffer command transfers a data file into the buffer device from a disk file. Use this command when the data in the file must be modified 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.

When you specify a particular address range to be read from a file, all sections in that address range that do not contain data are not placed in the buffer (i.e., the data in these sections of the buffer are not overwritten). This means that you can combine two or more files in the buffer.

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 you do not specify the source file or the keyword BUFFER, the iPPS software 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 command is directed to read the data from a file formatted in 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]

Where:

file specifies the standard ISIS 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 you do not specify *startaddr*, it defaults to the lowest address encountered while reading in the file. Omitting *startaddr* 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 specified as a length, L*length*. The *endaddr* is then *startaddr* + *length* − 1. If you do not specify *endaddr*, 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 (32K bytes), an error message occurs.

destaddr specifies the start address for the destination data in the URAM device. This parameter can be 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* exceeds URAM size − 1, an error message occurs. If you do not specify *destaddr*, it defaults to 0.

[F] is the optional data switch which complements the data before it is programmed. If you do not specify a data switch, the data is written into the PROM device in its uncomplemented form.

{ 80 / 86 / 286 } is the optional file format switch selection. If you do not specify this switch, 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 is displayed and the COPY command is aborted:

-GENERAL ERROR- -DATA OVERLAP IN THE FILE.

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-201A model of the universal programmer (and not with the iPDS system).

NOTE

Data stored in the URAM by the iPPS software can be used by the universal programmer only if the PROM to be programmed off-line is 32K bytes or smaller. For PROMs larger than 32K bytes, the universal programmer uses the URAM only for saving off-line editing changes.

When you specify a particular address range 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 was selected).

Examples

In the following example, data from file CODE4.IUP on drive 1 is copied (starting from file address 0) 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 1 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 you do not specify the source or destination, the iPPS software will prompt for them. In the next example, the data in an object program file (OBJPGM on drive 1) 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 executed correctly 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*)] [{ 80 / 86 / 286 }] [F]

Where:

startaddr specifies the start address of the section of URAM data to be copied to a file. If you do not specify *startaddr*, it defaults to 0. Omitting *startaddr* 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 specified as a length, L*length*. The *endaddr* is then *startaddr* + *length* − 1. If you do not specify *endaddr*, it defaults to *URAM size* − 1 (32K bytes − 1).

file specifies the standard ISIS file name for the file to be written.

destaddr specifies the start address of the destination data in the iPPS file. This parameter can be 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 you do not specify *destaddr*, it defaults to 0.

{ 80 / 86 / 286 } is the optional file format switch selection. If you do not specify this switch, the currently selected default file format is used.

[F] is the optional data switch which complements the data before it is written. If you do not specify a data switch, the data is written into the file in its uncomplemented form.

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-201A model of the universal programmer (and not with the iPDS system). If a file with the same name already exists on the specified disk, you are prompted before the existing file is overwritten.

NOTE

The iPPS software can use data read into the URAM during off-line operation only if the PROM device read is 32K bytes or smaller. When reading PROMs larger than 32K bytes, the universal programmer uses the URAM only for saving off-line editing changes.

Examples

In the following example, the entire contents of the URAM is copied to file URAM1.IUP on drive 1. The file is written in the 286 format (the default format).

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

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

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

If you do not specify the source or destination, the iPPS software will prompt for them. In the next example, data in the URAM starting at 400H and ending at 7FFH is copied to file URAM3.IUP on drive 1 (starting at file address 800H). The destination starting address in the file is specified as an offset (O400) 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   (O400)
 CHECK SUM = A539
```

COPY

Buffer to URAM

Syntax

COPY BUFFER [(*startaddr*[,*endaddr*])]

TO URAM [(*destaddr*)] [F]

Where:

startaddr specifies the start address of the section of data in the buffer to be copied into the URAM device. If you do not specify *startaddr*, it defaults to the buffer start address. Omitting *startaddr* 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 specified as a length, L*length*. The *endaddr* is then *startaddr* + *length* − 1. If you do not specify *endaddr*, 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 forced to the value *buffer start address* + *URAM size* − 1.

destaddr specifies the start address for the destination data in the URAM device. This parameter can be 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* exceeds *URAM size* − 1, an error message occurs. If you do not specify *destaddr*, it defaults to 0.

[F] is the optional data switch which complements the data before it is programmed into the URAM device. If you do not specify a data switch, the data is programmed in its uncomplemented form.

Operation

The COPY buffer to URAM command transfers data from the iPPS buffer to the RAM contained in the iUP-201A universal programmer for off-line operation. The COPY buffer to URAM command can only be used with the iUP-201A model of the universal programmer (and not with the iPDS system). If you specify a buffer address range greater than the size of the URAM, an error message is displayed. If you do not specify a buffer address range and the buffer is larger than the URAM, the additional data is truncated. Note that this condition does not return an error message.

NOTE

Data stored in the URAM by the iPPS software can be used by the universal programmer only if the PROM to be programmed off-line is 32K bytes or smaller. For PROMs larger than 32K bytes, the universal programmer uses the URAM only for saving off-line editing changes.

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. Note that lowercase characters are used.

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

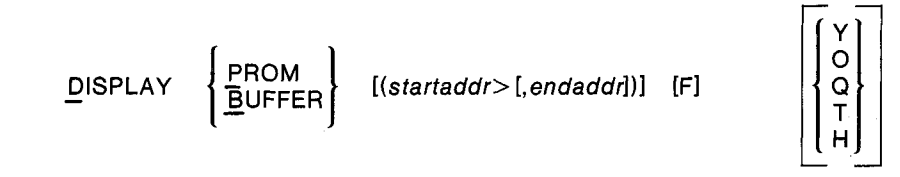
If you do not specify the source or destination, the iPPS software prompts for the missing keywords. In the next example, 1024 (1K) bytes of URAM data (starting at URAM address 400H) are copied to the buffer (starting at buffer address 1000H).

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

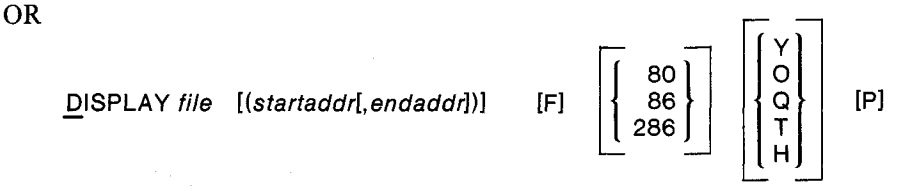
DISPLAY

Displays data on the console

Syntax



OR



Where:

file specifies the standard ISIS 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 terminal. Together with *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 the file. Omitting *startaddr* 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 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 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 form.

{Y, O, Q, T, H} is the optional base switch, which 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. You can specify only one optional base switch at a time. 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 you do not specify this switch, 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 you do not specify this switch and a patched file is read, the following error message is displayed and the DISPLAY command is aborted.

```
-GENERAL ERROR- -DATA OVERLAP IN THE FILE.
```

Operation

The DISPLAY command writes data from the PROM, buffer, or file device in a formatted display on the terminal. As shown in the syntax section, the DISPLAY *file* command is formatted differently from DISPLAY PROM and DISPLAY BUFFER.

When you specify an address range 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 ones 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 following message:

```
ENTER <CR> TO CONTINUE
```

To continue the display of data, press RETURN. Any other keys you press are echoed on the screen but do not continue the display. 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 the command). To the right of the actual data is the ASCII character equivalent for each displayed byte value. If there is no ASCII character equivalent, a period (.) is displayed as a place holder. Press the ESC key at any time to end the display. The ESC key is echoed as a dollar sign.

Examples

The following example displays the contents of the PROM device in hexadecimal (the default number base) one screenful at a time. This display is from a monitor PROM containing 8085 code. Note that the data displayed has embedded ASCII character strings. The ASCII equivalent display is convenient for revealing embedded ASCII character strings. Press the ESC key at any time to end the display. The ESC key is echoed as a dollar sign.

```
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 6F 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>
```

NOTE

The message "ENTER <CR> TO CONTINUE" is not displayed if the display occurs while the iPPS software is controlled from a submit file.

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 hexadecimal.

```
PPS>DISP BUFF(0,F) Y
00000000000000000000000: 11000011 01000000 00000000 00100000  .@.
00000000000000000000100: 00100000 01000100 00100000 00101101   D -
00000000000000000001000: 00100000 01000100 01001001 01010011   DIS
00000000000000000001100: 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                     ..
```

The next example displays the contents of a range of buffer addresses in decimal. The ending address 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>
```

<ESC>

Terminates current command

Syntax

<ESC>

Operation

Enter the <ESC> command by pressing the ESC key. If the <ESC> command is entered during any request by the iPPS software for keyboard input, control returns to the main command input for the iPPS software (indicated by the PPS> prompt). Thus, an iPPS command can be terminated at any key-in prompt by pressing the ESC key. In such a case, no portion of the command is executed. During certain non-interruptable iPPS operations, the ESC key is recognized only after that non-interruptable operation is completed, and 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>
```

You can now enter a new iPPS command.

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

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

EXIT

Exits the iPPS software and returns control to the ISIS operating system

Syntax

EXIT

Operation

The EXIT command returns control to the ISIS operating system when the PROM programming session is ended.

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

Examples

The following example returns to the ISIS operating system.

```
PPS>E
```

FORMAT

Interactively formats data

Syntax

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

OR

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

Where:

file specifies the standard ISIS 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 the file. Omitting *startaddr* 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 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 the file.

[F] is the optional data switch which complements the data before it is written to the output file. If you do not specify a data switch, the data is written in its uncomplemented form.

{80 / 86 / 286} is the optional file format switch selection. If you do not specify this switch, the currently selected default file format is assumed when reading and writing files.

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

-GENERAL ERROR- -DATA OVERLAP IN THE FILE.

Operation

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

Due to its complexity, the FORMAT command is interactive. Once the command is entered, you are prompted for parameters. You can terminate the command at any of these prompts by pressing the <ESC> key.

When you specify an address range 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 ones if no PROM type was 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 :F2: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 in 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 2-1 shows logically how an output array is produced from an input array in one interactive output cycle using the FORMAT command. You can create more than one output formatted data array from a single input array by repeating the output cycle.

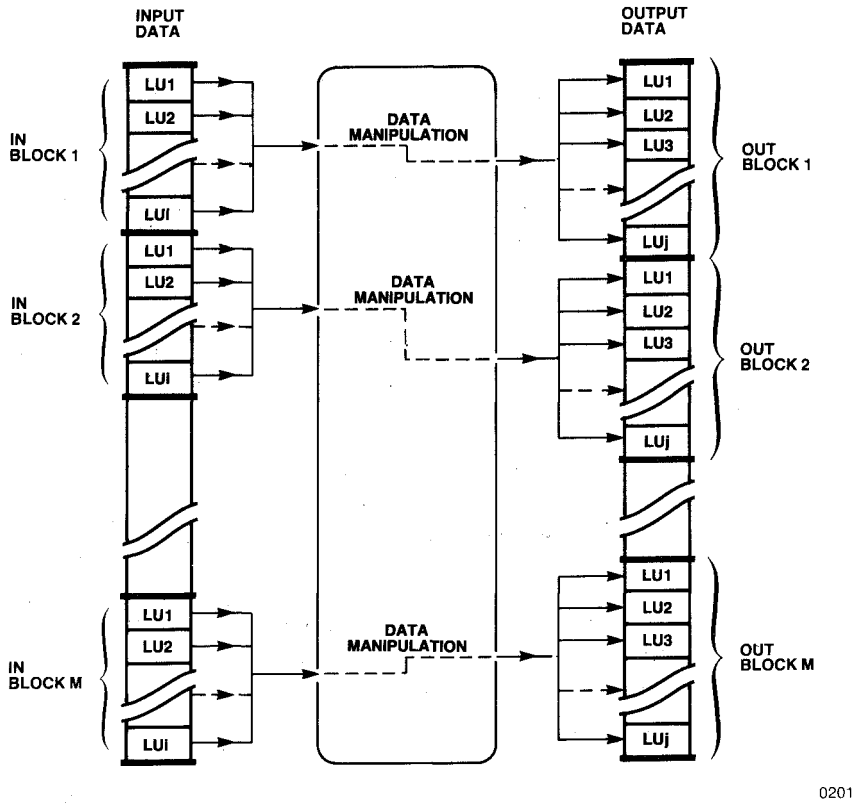


Figure 2-1 FORMAT Command Data Manipulation

Once you enter the interactive mode of the FORMAT command, it prompts you for the logical unit, input block size, and output block size. After you enter these parameters (which are defined in the following sections), 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 you entered, the software generates an output array stored in the specified output file. Once the output file is created, the Intellec development system/iPDS system beeps and the following message is displayed:

```
OUTPUT STORED
```

This is followed by the start of another output cycle indicated by the * prompt. Each output cycle lets you create formatted output files (using the same logical unit and input block size). To exit from the FORMAT command, press <CR> as the first non-blank character in answer to a prompt or press <ESC> during any key-in 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, the iPPS software 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, you can specify a numeric entry in another base by appending the appropriate number base tail (i.e., binary, Y; octal, O or Q; hexadecimal, H).

If you make a non-fatal error, an error message is displayed and the FORMAT software prompts for the same parameter.

Logical Unit

The logical unit is the basic unit in which data is 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 2-1 shows the input array split into M blocks. The input block size must be greater than or equal to the logical unit. The total number of logical units in an input block size must not exceed 24. The input block size cannot 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 cannot exceed 24. The output block size cannot exceed 65,535 bytes.

Input Block Structure

The input block structure shows how the input block is split into 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:

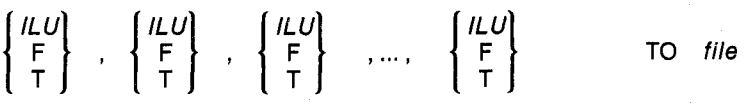
```
INPUT BLOCK STRUCTURE:
NUMBER OF INPUT LOGICAL UNITS = 00<i + 1>

    LSB
    -----------------------------------
    ! 00  ! 01  ! 02  ! ...  ! 0i  !
    -----------------------------------
```

Output Specification

The output specification specifies 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 you did not specify a PROM type using the TYPE command, the unspecified locations are filled with ones.

The output specification also specifies the ISIS file where the output data is to be stored. Following is the syntax of the output specification:



ILU is any of the input logical units shown in the input block structure. 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 as follows:

ILU1/ILUn TO *file*

You can also use the following notation to denote a reversal in 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 expands each 4-byte input block into an 8-byte output block with every alternate byte set to zero. The resulting output data is stored in file ALTZERO.BYT on drive 1.

Interactive Format Examples

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

Example: Nibble Swapping

This example shows how the FORMAT command swaps 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.

The command COPY :F1:BLKRVR.MOV TO BUFFER can 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) are 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 you enter HELP without the optional *command keyword*, a summary of iPPS commands is displayed on the terminal. If you enter the optional *command keyword*, 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, press the RETURN key. To abort the present HELP command and return to the iPPS software, press the <ESC> key.

The file IPPS.HLP must be available on the same drive as the iPPS command file or an error occurs when you enter the HELP command.

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 you enter HELP without the optional *command keyword*, a summary of the iPPS commands is displayed. If you enter the optional *command keyword*, detailed information on the syntax and operation of that command is displayed.

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 the iPPS software. The vertical bars enclose a set of values of which you must choose one.

PPS>**H D**

DISPLAY

<D>ISPLAY | <P>ROM / <B>UFFER | [(<startaddr>[,<endaddr>])] [F] [Y / O / Q / T / H]

OR

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

The DISPLAY command writes data from the PROM, buffer, or file devices in a formatted display to the terminal.

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 following message:

ENTER <CR> TO CONTINUE

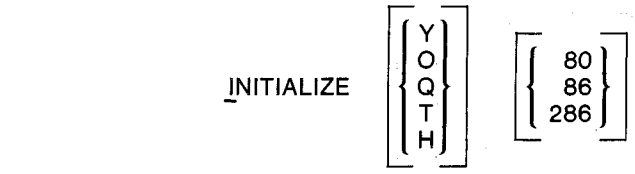
To continue the display of data, press the RETURN key. The format of the data displayed varies with the number base currently in effect (or specified for the duration of the command).

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



Where:

Y O Q T H	specifies the default base to be used. You can specify only one number base at a time. 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 the iPPS software when executing commands.

Operation

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

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 you select a new base. The number base default is hexadecimal.

The default file format determines the default file format that the iPPS software uses in commands that read and write files. Refer to Appendix B for more details on the file formats. The iPPS software can read both hexadecimal and binary object files, but it writes only binary object 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 you do not specify the file type or base, the iPPS software prompts for them as follows:

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

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

KEYLOCK

Locks the EPROM device

Syntax

KEYLOCK

Operation

On EPROM devices for which the locking function is supported, the KEYLOCK command locks the EPROM memory of a device from unauthorized access. If the EPROM device contains a security bit it cannot be unlocked without erasing the device. Authenticated EPROM devices recognize an identity code. With the identity code the user system can unlock the EPROM under software control. The iPPS software cannot unlock an EPROM device once it is locked.

Due to its complexity, the KEYLOCK command is interactive. Once the command is entered, you are prompted for needed parameters. You can terminate the command at any of these prompts by pressing the <ESC> key.

You will receive the following error message if you try to lock an EPROM on which the locking function is not supported:

```
---IUP ERROR----COMMAND NOT SUPPORTED BY DEVICE
```

Examples

The following examples show how the KEYLOCK command works. The wording of the prompts depends on the personality module and the EPROM to be programmed.

EPROM devices with a security bit (such as the 8751H microcontroller) can be locked from external access but cannot be unlocked without erasing the device. The KEYLOCK command asks you to confirm your intention to lock the PROM device. If you enter Y, the security bit is programmed, locking the device. If you enter N, the command terminates and the EPROM remains unlocked.

The following example shows successful locking of an 8751H microcontroller.

```
PPS>KEYLOCK
EXECUTE--Y/N?  Y
PPS>
```

The following example shows the message you receive if you enter N (or any character but Y).

```
PPS>KEYLOCK
EXECUTE--Y/N?  N
COMMAND ABORTED
PPS>
```

The next example shows trying to lock a blank or locked 8751H microcontroller.

```
PPS>KEYLOCK
EXECUTE- -Y/N?  Y
- - -IUP ERROR- - - -ALREADY LOCKED
PPS>
```

Authenticated EPROMs can be locked, but they can also be unlocked by the user system software using an identity code. You program the identity code into the EPROM before locking it.

The KEYLOCK command displays the value of each parameter currently programmed into the EPROM device. Because parameter values are saved into a buffer until the EPROM device is locked, you can change the value of a parameter by entering a new value and pressing carriage return or retain the value of a parameter by pressing carriage return. The first parameter is the delay count which lets you specify the speed at which the authentication handshake takes place. The second and third parameters are the key numbers that identify the PROM to the key manager. The next eight parameters are the identity code. The command mask is the last parameter; it must be programmed before you can lock the EPROM.

After you have entered all parameters for the particular EPROM device, the KEYLOCK command asks if you want to program the parameters into the EPROM. Programming the last parameter (the command mask) and entering Y to the EXECUTE- -Y/N? prompt locks the EPROM. (If you enter Y but have not programmed the command mask with a valid parameter, the other parameters entered will be programmed but the EPROM will not lock.) If you enter N, the iPPS software prompts for all parameters again which lets you verify the parameters you entered.

Legal parameter values depend on the EPROM you are programming. The user's guide for the personality module you are using contains legal parameter values.

As with other iPPS commands, you do not need to program all parameters in one command session. You can use many KEYLOCK commands to program an EPROM. Note that you cannot change a bit in a parameter from a programmed to an unprogrammed state (e.g., 35H to FFH).

The following example shows verification and locking of an authenticated EPROM. Note that values are displayed in the current default number base specified by the most recent INITIALIZE command.

```
PPS>INIT H                  ;set default number base to hexadecimal
PPS>KEYLOCK
DELAY CNT   (FF) = 3F       ;Delay count changed to 3F hex
KEY NUM-1   (FF) = 10       ;Key number 1 changed to 10 hex
KEY NUM-2   (FF) = 1D       ;Key number 2 changed to 1D hex
KEY 1   (FF) = <CR>         ;Key 1 unchanged
KEY 2   (01) = <CR>         ;Key 2 unchanged
KEY 3   (FF) = 11           ;Key 3 changed to 11 hex
KEY 4   (FF) = FF           ;Key 4 unchanged
KEY 5   (FF) = 16T          ;Key 5 changed to 16 decimal
KEY 6   (FF) = 101Q         ;Key 6 changed to 101 octal
```

```
KEY 7   (FF) = <CR>          ;Key 7 unchanged
KEY 8   (FF) = 77            ;Key 8 changed to 77 hex
CMD MASK   (FF) = E2         ;Command mask changed to E2 hex
EXECUTE- -Y/N?  N            ;Enter no to verify entries
DELAY CNT   (3F) = <CR>      ;Delay count unchanged
KEY NUM-1   (10) = <CR>      ;Key number 1 unchanged
KEY NUM-2   (1D) = <CR>      ;Key number 2 unchanged
KEY 1   (FF) = 23            ;Key 1 changed to 23 hex
KEY 2   (01) = <CR>          ;Key 2 unchanged
KEY 3   (11) = 00100010Y     ;Key 3 changed to 22 hex
KEY 4   (FF) = <CR>          ;Key 4 unchanged
KEY 5   (10) = 33            ;Key 5 changed to 33 hex
KEY 6   (41) = <CR>          ;Key 6 unchanged
KEY 7   (FF) = <CR>          ;Key 7 unchanged
KEY 8   (77) = <CR>          ;Key 8 unchanged
CMD MASK   (E2) = <CR>       ;Command mask unchanged
EXECUTE- -Y/N?  N            ;Enter no to verify entries
DELAY CNT   (3F) = <CR>      ;Delay count unchanged
KEY NUM-1   (10) = <CR>      ;Key number 1 unchanged
KEY NUM-2   (1D) = <CR>      ;Key number 2 unchanged
KEY 1   (23) = <CR>          ;Key 1 unchanged
KEY 2   (01) = <CR>          ;Key 2 unchanged
KEY 3   (22) = <CR>          ;Key 3 unchanged
KEY 4   (FF) = <CR>          ;Key 4 unchanged
KEY 5   (33) = 35            ;Key 5 changed to 35 hex
KEY 6   (41) = <CR>          ;Key 6 unchanged
KEY 7   (FF) = <CR>          ;Key 7 unchanged
KEY 8   (77) = <CR>          ;Key 8 unchanged
CMD MASK   (E2) = <CR>       ;Command mask unchanged
EXECUTE- -Y/N?  Y            ;Enter yes to lock EPROM
PPS>
```

The next example illustrates user-generated errors while trying to program an authenticated EPROM.

```
PPS>KEYLOCK
DELAY CNT   (FF) = 200T      ;Delay count changed to 200 decimal
- - -IUP ERROR- - - -ILLEGAL PARAMETER VALUE
DELAY CNT   (FF) = 3F        ;Delay count changed to 3F hex, a legal size
KEY NUM-1   (FF) = F0        ;Key number 1 changed to F0 hex
KEY NUM-2   (FF) = 2C        ;Key number 2 changed to 2C hex
KEY 1   (12) = <CR>          ;Key 1 unchanged
KEY 2   (F0) = 00            ;Key 2 changed to 00
KEY 3   (FF) = 256T          ;Key 3 changed to 256 decimal
-KEYLOCK ERROR- -VALUE EXCEEDS 8 BITS
KEY 3   (FF) = 250T          ;Key 3 changed to 250 decimal
KEY 4   (F0) = FFG           ;Key 4 changed to FFG - G is an illegal base
- -SYNTAX ERROR- -ILLEGAL DIGIT SPECIFICATION
KEY 4   (F0) = FF            ;Key 4 changed to FF hex
KEY 5   (FF) = 22            ;Key 5 changed to 22 hex
KEY 6   (FF) = 101Q          ;Key 6 changed to 101 octal
KEY 7   (FF) = <CR>          ;Key 7 unchanged
KEY 8   (FF) = <CR>          ;Key 8 unchanged
CMD MASK   (FF) = E2         ;Command mask changed to E2 hex
EXECUTE- -Y/N?  Y            ;Enter yes to lock the EPROM
- - -IUP ERROR- - - -LOCK FAILED AT KEY 4
                             ;Key 4 did not program because it was
                             ;changed from a programmed to an
                             ;unprogrammed state
PPS>
```

LOADDATA

Fills section of
buffer with constant

Syntax

LOADDATA BUFFER [(*startaddr*[,*endaddr*])]

WITH *byteval* [F]

Where:

startaddr specifies the start address of the section of data in the buffer to be filled with a constant. If you do not specify *startaddr*, it defaults to the buffer start address. Omitting *startaddr* 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 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 you specify an address range greater than the size of the current PROM type, 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 you do not specify a data switch, *byteval* is loaded into the buffer section in its uncomplemented form.

Operation

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

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 buffer and file structure and iPPS status

Syntax

MAP [*file*] [{ 80 / 86 / 286 }] [P]

Where:

file specifies the standard ISIS file name for a source file. If you do not specify *file*, the file structure of the most recently read file (i.e., read by any previous iPPS command) is displayed. Because the MAP command works only with absolute addresses, it returns an error if you specify a file containing relocatable code.

{ 80 / 86 / 286 } is the optional file format switch selection. If you do not specify this switch, 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 you do not specify this switch and a patched file is read, the following error message is displayed and the MAP command is aborted.

```
-GENERAL ERROR- -DATA OVERLAP IN THE FILE.
```

Operation

A memory map of the file structure is maintained when data is read into the buffer from a file. The memory map records starting and ending addresses of distinct data blocks in the file. This is useful to see what sections of the file contain or do not contain data. The MAP command displays this file structure. The MAP command also displays the status of various iPPS working parameters, the buffer boundaries, the currently selected PROM type, the current default number base, the current default file format, and the current work files drive.

Examples

The following MAP command example assumes that the :F1:CODE2.IUP file was 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. are 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]]

Where:

startaddr specifies the start address of the section of data in the buffer to be overlaid on the data in the PROM. If you do not specify *startaddr*, it defaults to the buffer start address. Omitting *startaddr* implies that the *endaddr* default is also assumed.

endaddr specifies the end address for the section of buffer to be overlaid on the data in the PROM. The *endaddr* parameter can be specified as a length, L*length*. The *endaddr* is then *startaddr* + *length* − 1. If you do not specify *endaddr*, it defaults to an address determined by the expression *buffer start address* + *buffer size* − 1. An error message is displayed if the address range specified is greater than the size of the current PROM type.

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 0FFFFFFH (16777215T), an error message is displayed. If *destaddr* exceeds *PROM size* − 1, an error message occurs. If you do not specify *destaddr*, it defaults to 0.

[F] is the optional data switch which complements the buffer data before comparison with the PROM data. If you do not specify a data switch, data is compared in its uncomplemented form.

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 overlaid on the existing data of the PROM without conflict.

In effect, the OVERLAY command checks the PROM for stuck (pre-programmed) bits. If stuck bits are not found, it displays the following message:

```
OVERLAY TEST PASSED
PPS>
```

If it finds stuck bits, it displays the following message:

```
OVERLAY ERROR- -DISPLAY Y/N?
```

To see the mismatched data, enter Y; otherwise, enter 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 command tests and passes a specified range of buffer addresses.

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

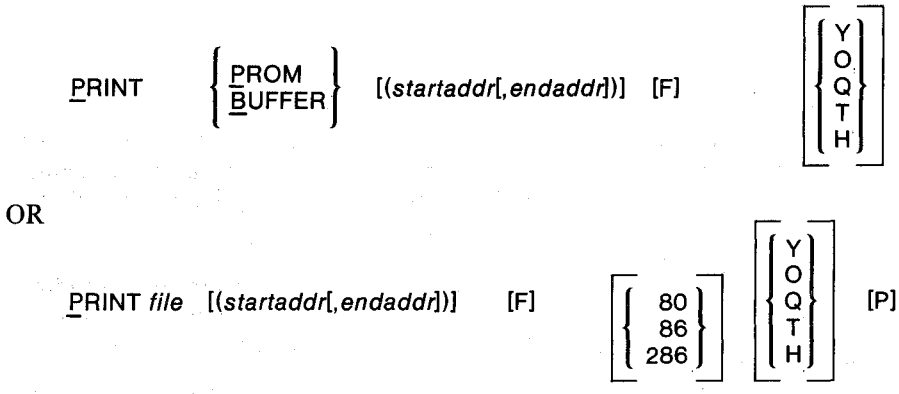
Using just the abbreviated OVERLAY keyword, the entire buffer and PROM device are tested in the next example.

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

PRINT

Prints data on development system printer

Syntax



Where:

file specifies the standard ISIS 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 the file. Omitting *startaddr* 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 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 form.

{Y, O, Q, T, H} is the optional base switch which specifies the number base to be used for the printed data during execution of the command. Binary is specified by the letter Y; octal by O or Q; decimal by T; and hexadecimal by H. You can specify only one base switch at a time. 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 you do not specify this switch, 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 you do not specify this switch and a patched file is read, the following error message is displayed and the PRINT command is aborted.

```
-GENERAL ERROR- -DATA OVERLAP IN THE FILE.
```

Operation

The PRINT command writes data from the PROM, buffer, or file device in a formatted printout on the development system printer. As shown in the syntax section, 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.

When you specify an address range to be read from a file, all sections in the address range that are *not* present in the file are printed using the blank state of the currently selected PROM type (all ones if no PROM type was selected).

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

Examples

The syntax and output format of the PRINT command is the same as for the DISPLAY command. Refer to the DISPLAY command description for example formats 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 is printed in binary. The F range specification is interpreted as hexadecimal because it has no explicit number base switch and the current default number base is 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
```

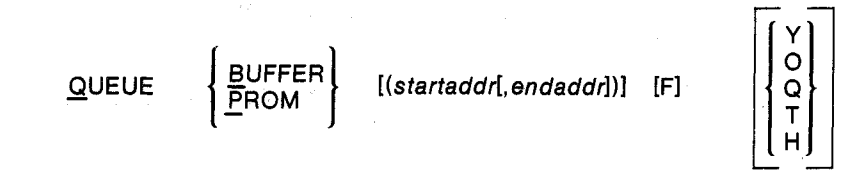
The next example prints the contents of a range of buffer addresses in decimal. The range is specified in hexadecimal.

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

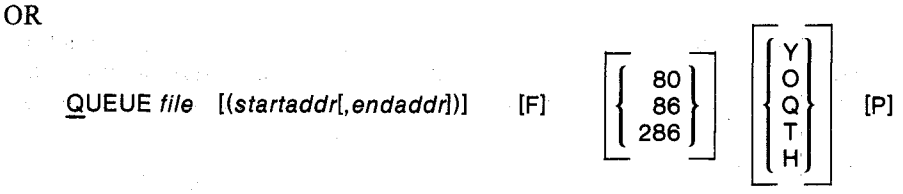
QUEUE

Prints data on network spooled printer

Syntax



OR



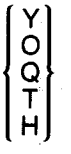
Where:

file specifies the standard ISIS 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 the file. Omitting *startaddr* 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 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 form.



is the optional base switch which specifies the number base to be used for the printed data during execution of the command. Binary is specified by the letter Y; octal by O or Q; decimal by T; and hexadecimal by H. You can specify only one base switch at a time. 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 you do not specify this switch, 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 you do not specify this switch and a patched file is read, the following error message is displayed and the QUEUE command is aborted.

```
-GENERAL ERROR- -DATA OVERLAP IN THE FILE.
```

Operation

The QUEUE command writes data from the PROM, buffer, or file device in a formatted printout on the network printer. As shown in the syntax section, the QUEUE *file* command has the additional switch options to specify the file format being read and whether or not the source file is patched.

NOTE

You cannot use the QUEUE command on the iPDS system.

When you specify an address range to be read from a file, all sections in the address range that are not present in the file are printed using the blank state of the currently selected PROM type (all ones if no PROM type was selected).

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

The QUEUE command sends data to the file :SP:IPPS.*xx*, where *xx* is a decimal number from 00 to 99. Each time you issue the QUEUE command, *xx* increments to the next available *xx*; after *xx* reaches 99, it starts over at 00.

If your version of the ISIS operating system does not support :SP:, you will get an error if you try to use the QUEUE command.

Examples

The syntax and output format of the QUEUE command is the same as for the DISPLAY command. Refer to the DISPLAY command section for example formats of the printout for all the following QUEUE command examples.

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

```
PPS>QUEUE PROM
OUTPUT TO :SP:IPPS.00
```

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>QUEUE PROM(0,L18H) Q
OUTPUT TO :SP:IPPS.01
```

In the following example the contents of the specified range of buffer addresses is printed in complemented form in binary. The range specification is interpreted as hexadecimal because it has no explicit number base switch and the current default number base is hexadecimal.

```
PPS>QUE PROM(0,1FF) Y F
OUTPUT TO :SP:IPPS.02
```

The next example prints the contents of a file in decimal.

```
PPS>Q file.tmp T
OUTPUT TO :SP:IPPS.03
```

REPEAT

Repeats previous command

Syntax

REPEAT

Operation

The REPEAT command re-executes the most recently entered command without requiring you to re-enter the entire command and its parameters. Except for parameters entered interactively (the FORMAT and KEYLOCK commands), all parameters retain the values 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 programs several PROMs in succession without re-entering the original copy command. Each time you use the REPEAT command, it redisplays the command it is repeating. After a PROM is programmed, it is replaced by a new blank PROM to be programmed before entering the REPEAT command.

```
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}

Where:

addr specifies the starting address at which data is to be displayed or modified.

[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 form.

{Y | O | Q | T | H} is the optional base switch, which specifies the number base to be used for the displayed data during command execution. Binary is specified by the letter Y; octal by O or Q; decimal by T; and hexadecimal by H. You can specify only one base switch at a time. The default for this switch is the current default number base determined by the most recent INITIALIZE command.

Operation

The SUBSTITUTE command lets you interactively examine and modify specific locations in the development system buffer.

When you invoke the SUBSTITUTE command, it displays the contents of the buffer starting at the address specified in the command, as in the following:

```
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.

You 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, eight spaces are required to move to the next byte displayed. No locations are modified. The display is automatically scrolled to the next line when the cursor moves beyond the end of the line.

< moves the cursor one character at a time to the left. The cursor cannot move 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 moves 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>.

Entering a valid numeric value (determined by the current number base) changes the contents of the byte at the current cursor position. Pressing an invalid key causes a warning beep at the terminal.

Examples

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

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

In the next example, using the > key, the cursor is advanced to character 9 (at address 402H).

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

In the following example, the new value of C is entered over the old value of 9; the change is written after pressing the <CR> key. Control returns to the iPPS software.

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

TYPE

Selects PROM type

Syntax

TYPE [*PROM device number*]

Where:

PROM device number	is an allowable PROM type for the personality module installed in the universal programmer/iPDS system.

Operation

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

If you enter the command without the argument, you are prompted to enter one of the allowable PROM types for the personality module currently installed. An error message is displayed if the PROM type entered is not supported by the currently installed personality module or if no personality module is installed.

If you select a PROM type that requires more buffer space than 8K, a virtual buffer is automatically built and maintained on the currently selected work files drive. If no prior work files drive was established (refer to the WORKFILES section), a work files drive is requested by the following prompt:

```
WORKFILES (:FX:)? ; X =
```

At this point, you may enter a value for X. If you do not enter a value for X but do enter a <CR> or <ESC>, the work files drive is the drive from which the iPPS software was originally obtained. Use the MAP command to determine what drive is currently the work files drive. Use the WORKFILES command to change the current work files drive.

Examples

In the following example, the TYPE command is entered without specifying a PROM type on the command line. The iPPS software displays the valid types for the specific personality module installed 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 following example, the PROM 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 work files 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]]

Where:

startaddr specifies the start address of the section of data in the buffer to be verified with the data in the PROM. If you do not enter *startaddr*, it defaults to the buffer start address. Omitting *startaddr* 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 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, an error message is displayed.

destaddr specifies the desired start address for the comparison data in the PROM device. This parameter can be 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* exceeds *PROM size* − 1, an error message occurs. If you do not specify *destaddr*, it defaults to 0.

[F] is the optional data switch which complements the buffer data before comparison with the PROM data. If you do not specify a data switch, data is compared in its uncomplemented form.

Operation

The VERIFY command compares the data in the PROM device with the data in the buffer. You can then display any discrepancies between the two. The buffer contents are displayed first followed by the PROM contents.

If you enter VERIFY without any arguments, the entire PROM is compared starting at the buffer start address.

You must execute the TYPE command before executing the VERIFY command; otherwise, an error message is displayed. An error message is also displayed if the address range exceeds the size of the PROM device.

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 displayed because the response Y is given to the prompt. If the N response had been given, the iPPS prompt, PPS>, would return.

```
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
PPS>
```

In the next example, the verification test tests and passes a specified range in the buffer.

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

WORKFILES

Specifies drive for temporary work files

Syntax

WORKFILES :F*x*:

Operation

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

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

If the WORKFILES command was not entered before a temporary file is needed (by certain iPPS commands), the software prompts for the work files drive number with the following display:

```
WORKFILES (:FX:)? ; X =
```

At this point, you may enter a value for X. If you enter a <CR> or <ESC> instead of a value for X, the work files drive is the drive from which the iPPS software was originally obtained. Use the MAP command to determine what drive is currently the work files drive. Use the WORKFILES command to change the current work files drive.

Examples

The following example selects drive 1 for temporary iPPS work files.

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

In the next example, no drive is specified after the command keyword, so the command prompts for the work files drive number. The single-character abbreviation for WORKFILES is used. Drive 0 is selected for all temporary iPPS work files.

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

CHAPTER 3
PROM PROGRAMMING EXAMPLES

Introduction

The iPPS software offers a wide variety of PROM programming, data display, and data editing capabilities. 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 such as 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 will give you hands-on experience in using these features.

Before you go through these examples, read Chapters 1 and 2 of this manual to become familiar with the iPPS commands.

Table of Contents of Examples

The following table of contents is a quick reference to the examples in this chapter. It lists the programming or editing functions illustrated and the commands demonstrated in each example.

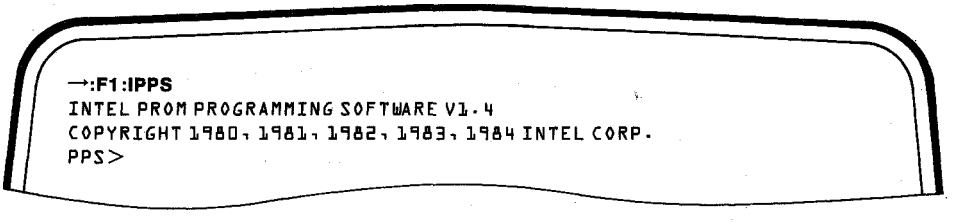
Examples

The following examples assume that the Intellec development system/iPDS system has been powered on and is under control of the ISIS software.

The screen examples are the development system screen. The bold-face characters indicate user entries. The key-in sequence below each screen gives the actual entries that you must key in to obtain the screen display.

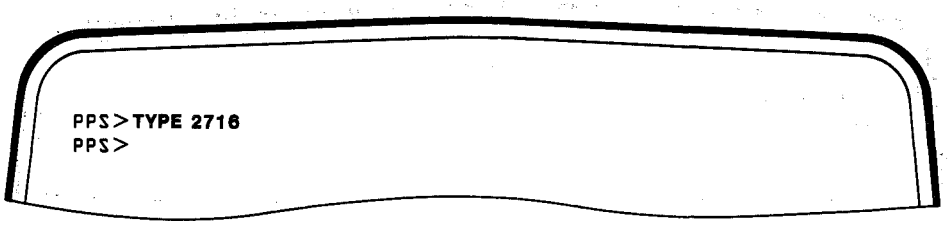
On-line iPPS Software Initialization

You must initialize the iPPS software before performing any on-line functions. After checking that the ISIS prompt is on the CRT screen, enter the following command to invoke the iPPS software:



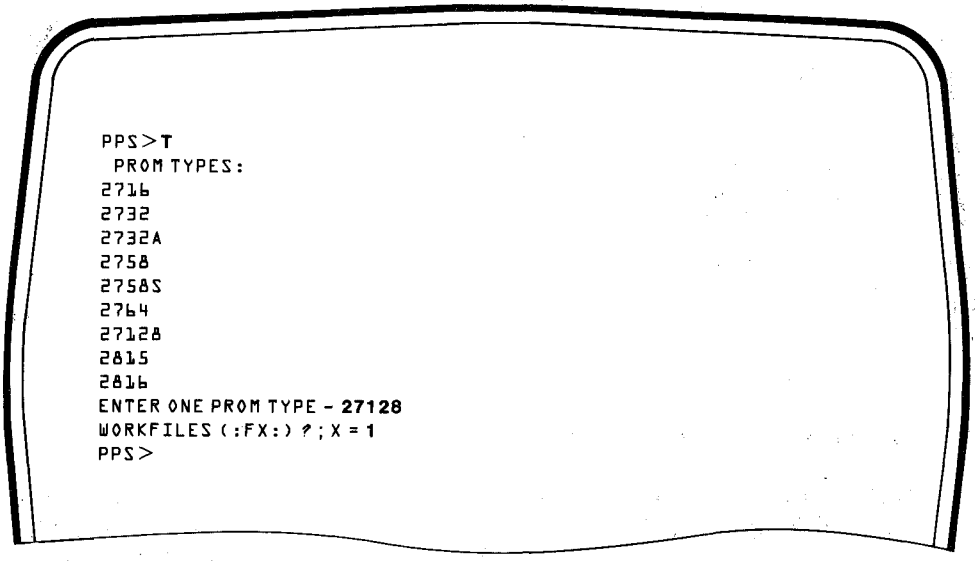
Key-in Sequence	Comments
:F1:IPPS RETURN	You can locate the iPPS files on any drive. In this example, they are located on drive F1. The iPPS prompt PPS> indicates that the iPPS files have been loaded and the development system will now execute iPPS commands. The version number of the iPPS software may be different, depending on your version. To return to ISIS, enter EXIT and a carriage return anytime following the iPPS prompt.

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



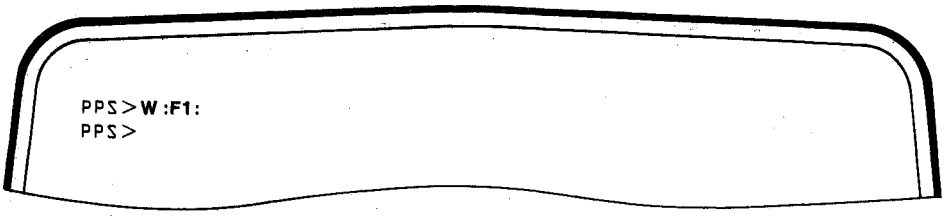
Key-in Sequence	Comments
TYPE 2716 RETURN	In this example, the PROM type is specified and the iPPS prompt is returned. The LED indicators on the personality module indicate the PROM type selected and the sockets to be used. Once the PROM type is selected, the iPPS software automatically sets the boundaries of the iPPS buffer to be equal to the size of the PROM.

Entering the TYPE command without parameters gives you a list of the PROM types that can be programmed with the personality module that has been installed.



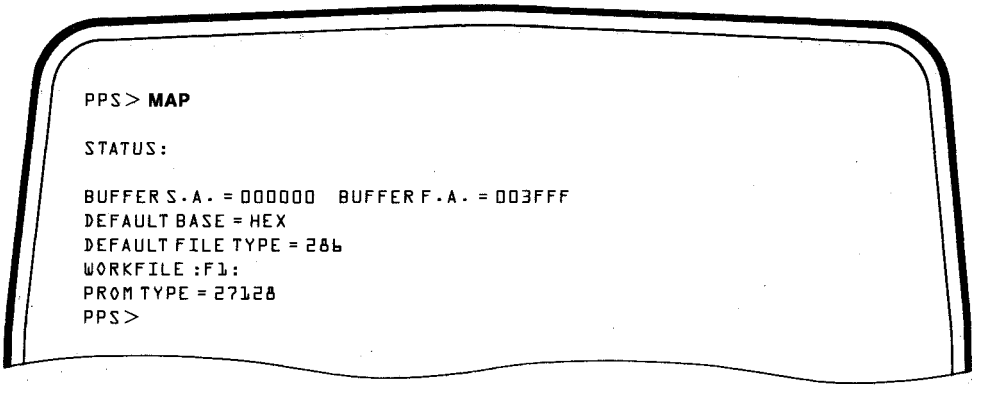
Key-in Sequence	Comments
T RETURN **27128** RETURN **1** RETURN	In this example, the PROM type was not specified in the TYPE command. (Note that the single-character abbreviation for the TYPE keyword is used.) All the PROM types that the installed personality module supports are then displayed, and the PROM type is requested. In this case, the PROM type selected (27128) requires a virtual buffer; so the command prompts for a drive to be specified for the temporary workfile. If the workfile drive has not already been specified (refer to the next example) you are prompted for it whenever you specify a PROM type with greater than 8K storage capacity.

The WORKFILES command lets you specify the work files drive at any time. You can use this function to change the drive specified earlier or as part of a submit file to avoid the prompt for a work files drive specification.



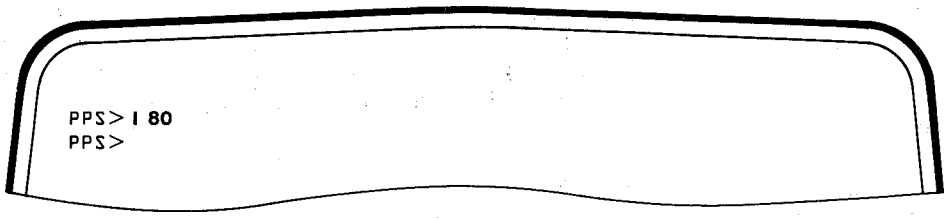
Key-in Sequence	Comments
W :F1: RETURN	Drive 1 is selected for all workfiles. Note that the keyword abbreviation was used instead of entering the whole command keyword.

You can reset the data type and file type default values 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 (output file format), respectively.



Key-in Sequence	Comments
MAP RETURN	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.

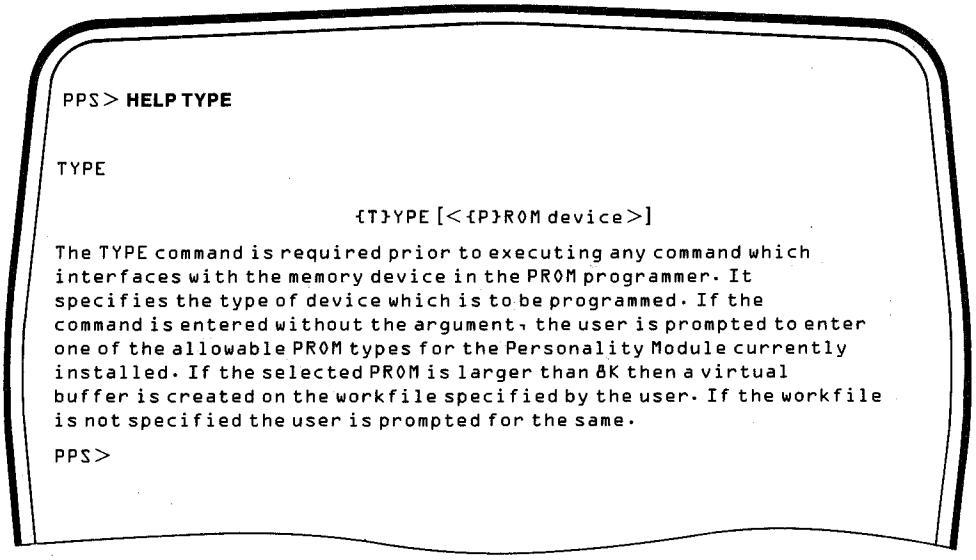
The INITIALIZE command lets you specify the default number base, the default file type, or both.



Key-in Sequence	Comments
I 80 RETURN	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 and write files of a specified format. If you specify the wrong file type, an error message is displayed when you try to read or copy a file.

Many of the iPPS commands let you 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 value. This feature is demonstrated in several of the examples in this chapter.

The iPPS software offers a convenient help feature that lets you display a description of any of the iPPS commands on the CRT screen.



Key-in Sequence	Comments
HELP TYPE RETURN	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 universal programmer/iPDS system is copying data from a PROM into a buffer or file, then copying it into another PROM. You can perform this operation on-line, using the iPPS buffer (or a file in the development system for intermediate storage) and the iPPS COPY commands.

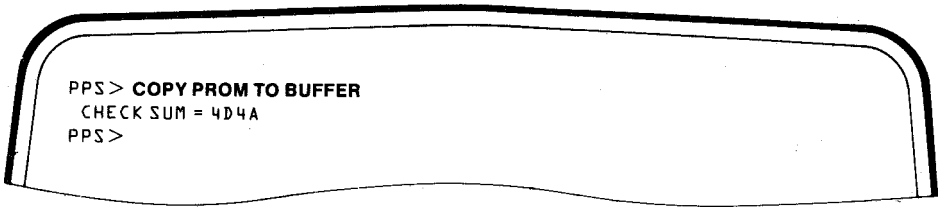
If you are performing these examples, reset the iPPS software (using the TYPE command) for the type of PROM you are going to use for the following copy operations. A 2716 EPROM that contains sample code is used in these examples.

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. (The user's guide for the personality module you are using describes PROM device installation.) Then copy the contents of the PROM to the iPPS buffer.

CAUTION

Verify that the type of PROM that you are installing is the same as the type selected with the TYPE command. If you specify its type incorrectly, you can damage a PROM when you try to program it or read it.



Key-in Sequence

COPY PROM TO BUFFER

RETURN

Comments

This command copies every memory location in the PROM to the buffer beginning at destination address 00H in the buffer. The checksum is the 2's complement of the 16-bit sum of all the bytes read.

Use the VERIFY command to check that the copy was correct.

```
PPS> VERIFY
VERIFY TEST PASSED
PPS>
```

Key-in Sequence	Comments
VERIFY RETURN	The data in the buffer matches the data in the PROM.

Now that you have verified 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.

```
PPS>COPY BUFFER TO PROM
 CHECK SUM = 4D4A
PPS>
```

Key-in Sequence	Comments
COPY BUFFER TO PROM RETURN	The display of the check-sum and the return of the iPPS prompt indicates that the PROM was successfully programmed.

If you wish to program another blank PROM with the same data, remove the programmed PROM and install a blank PROM in the personality module. Enter either the same command or use the REPEAT command.

```
PPS>CBTP
 CHECK SUM = 4D4A
PPS>
```

Key-in Sequence	Comments
C B T P RETURN	In this case, the same command is entered, using the keyword abbreviations. Install the next blank PROM.

```
PPS> REPEAT
C B T P
 CHECK SUM = 4D4A
PPS>
```

Key-in Sequence	Comments
REPEAT RETURN	Entering REPEAT causes the previous command to be echoed and executed.

As part of the copy operation, the iPPS performs a blankcheck to ensure that the PROM is blank. If it is not blank, the iPPS 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 successfully completed.

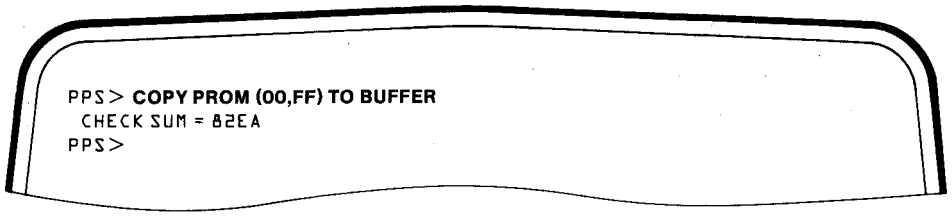
The following example attempts to program a PROM that was not blank.

```
PPS> CBTP
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>
```

Key-in Sequence	Comments
C B T P RETURN	Note that the keyword abbreviations are used.
Y RETURN	
Y RETURN	

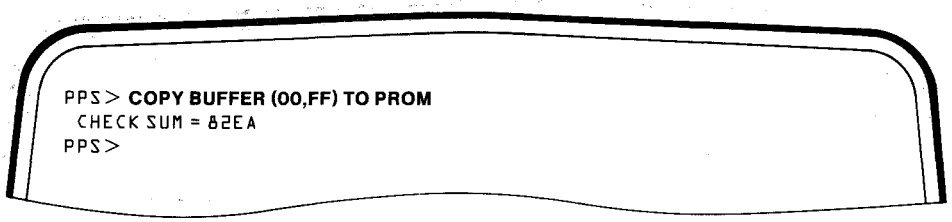
Now assume that you wish to copy only part of the master PROM and then program the blank PROM with this information, leaving the rest of the PROM in its blank state.

The COPY command lets you 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.



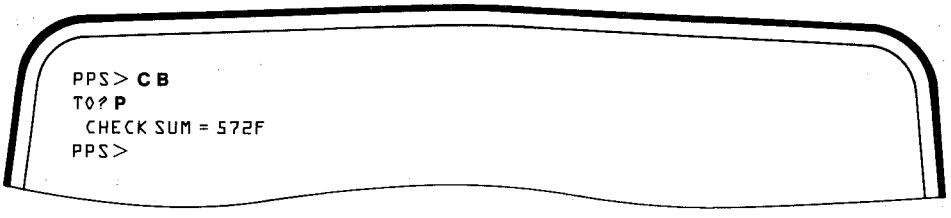
Key-in Sequence	Comments
COPY PROM (00,FF) TO BUFFER RETURN	The program code or data from the address range 00H to FFH in the PROM is copied into the iPPS buffer.

Now install the blank PROM in the personality module and copy the contents of the buffer to the new PROM.



Key-in Sequence	Comments
COPY BUFFER (00,FF) TO PROM RETURN	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 make a syntax error, such as an incorrect command keyword.



Key-in Sequence	Comments
C B RETURN **P** RETURN	In this case, the iPPS software prompts you for the second half of the command argument.

In the following example, an incorrect keyword was entered.

```
PPS> COPPY PROM TO BUFFER
COPY P#
- -SYNTAX ERROR- -ILLEGAL KEYWORD
PPS>ALTER
COPPY PROM TO BUFFER
PPS>
```

Key-in Sequence	Comments
COPPY PROM TO BUFFER [RETURN]	The part of the command containing the syntax error and the error message is displayed. You can use the ALTER command to correct the error. The underscore on the screen display indicates the cursor position.
ALTER [RETURN]	

```
COPPY PROM TO BUFFER
```

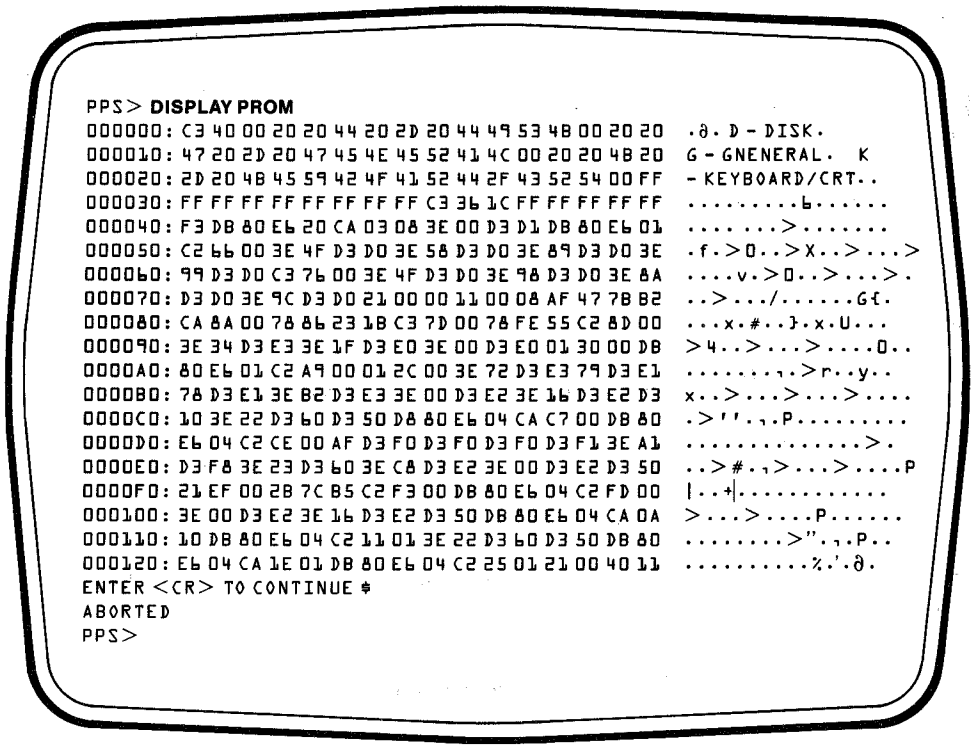
Key-in Sequence	Comments
[>]	The cursor is moved under the first P character using the > key.
[>]	

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

Key-in Sequence	Comments
[CNTL] [D]	CNTL-D is entered 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 inserts characters in a command.)
[RETURN]	
Y	

Examining the Contents of a Masked ROM

The DISPLAY command lets you examine the contents of a PROM or masked ROM.



```
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 - GNENERAL.  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 7B 86 23 1B C3 7D 00 7B 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: 7B 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: 10 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>
```

Key-in Sequence

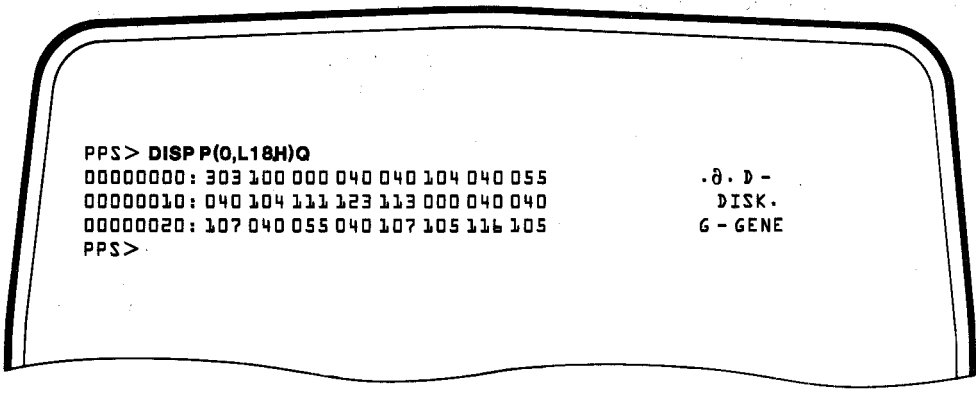
DISPLAY PROM

RETURN

ESC

Comments

This example shows the data in the PROM in hexadecimal format, which is the default base 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. Note the ASCII code displayed in the far right column.



```
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>
```

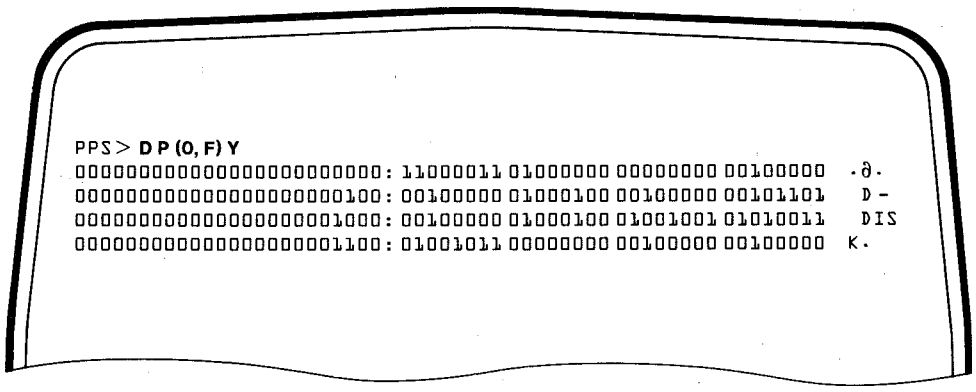
Key-in Sequence

DISP P(0,L18H) Q

RETURN

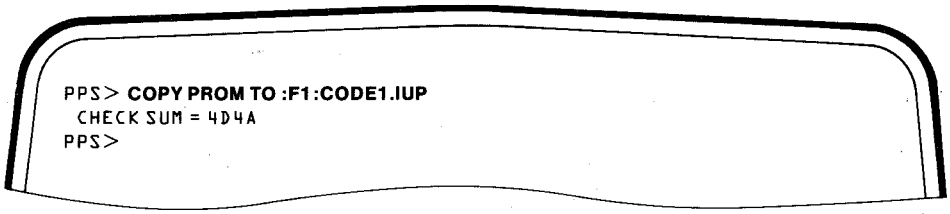
Comments

In this 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	Comments
D P (0,F) Y RETURN	Here, 16 bytes of ROM data beginning with address 00Y are displayed in binary.

To save the information in the PROM, you can copy the contents of the PROM into a file.

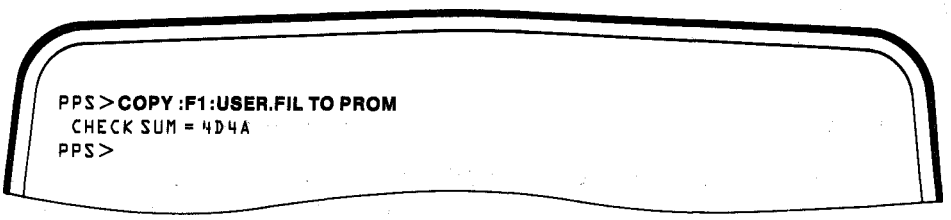


Key-in Sequence	Comments
COPY PROM TO :F1:CODE1.IUP RETURN	The contents of the PROM are saved in an ISIS file titled CODE1.IUP, which is located on drive 1.

Copying a File to a PROM

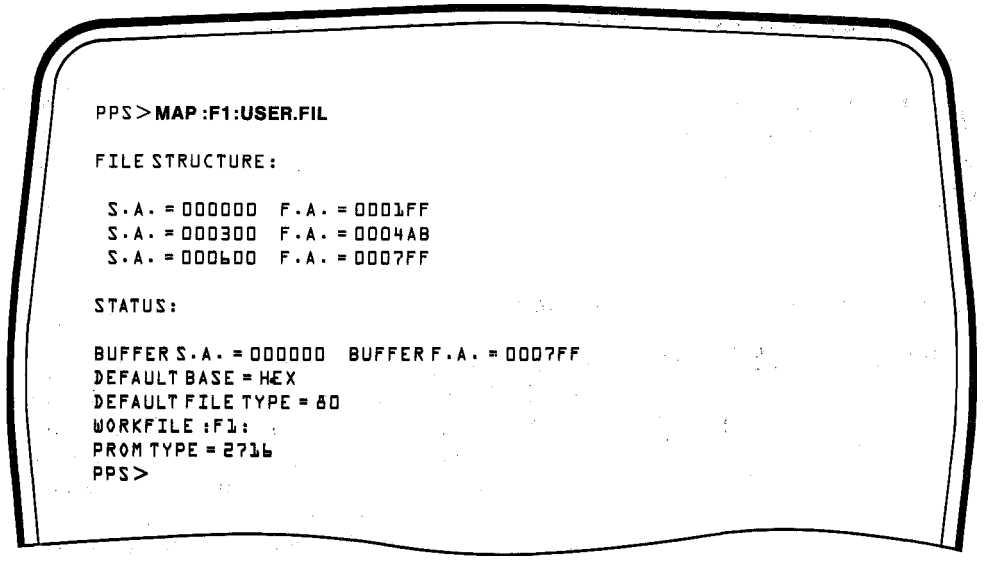
Another often used application of the universal programmer/iPDS system is copying programs and data stored in an ISIS file into a PROM. This is done using the COPY FILE TO PROM command.

Here is an example of a direct copy of a file to a PROM. Assume in the following examples that the blank PROM is installed in the personality module and the iPPS software is initialized for the type of PROM to be programmed.



Key-in Sequence	Comments
COPY :F1:USER.FIL TO PROM RETURN	The entire file is copied to the blank PROM beginning at location 0000H.

You can use all the features of the COPY command 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 need more information about the structure of the file. The MAP command gives you the boundaries of the file.



Key-in Sequence	Comments
MAP :F1:USER.FIL RETURN	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) as in the following example.

```
PPS>C :F1:USER,FIL (0,1FF) T P
 CHECK SUM = 234B
PPS>C :F1:USER.FIL (300,4AB) T P (400)
 CHECK SUM = 3551
PPS>
```

Key-in Sequence

C :F1:USER.FIL (0,1FF) T P

RETURN

C :F1:USER.FIL (300,4AB) T P (400)

RETURN

Comments

In this 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.

```
PPS>C :F1:TYPE.86 T P 86
 CHECK SUM = 2341
PPS>
```

Key-in Sequence

C :F1:TYPE.86 T P 86

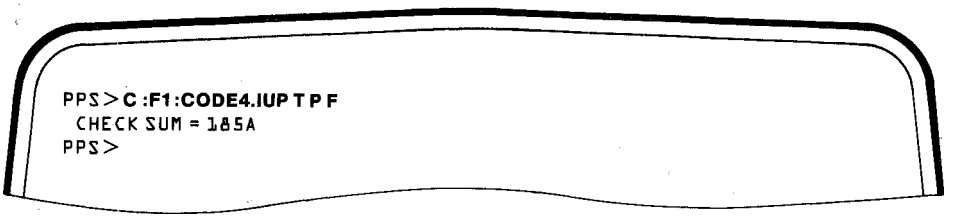
RETURN

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.

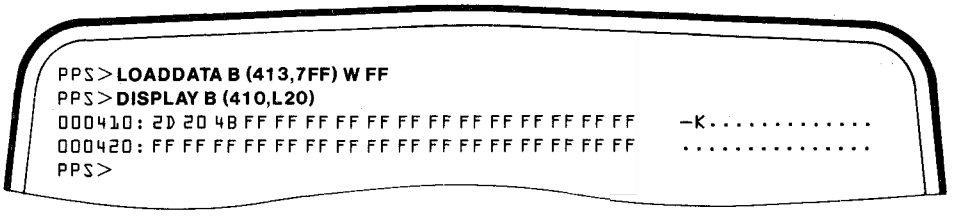
Modifying a File

The iPPS software lets you 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, complementing data, can be specified as part of the COPY FILE TO PROM command.



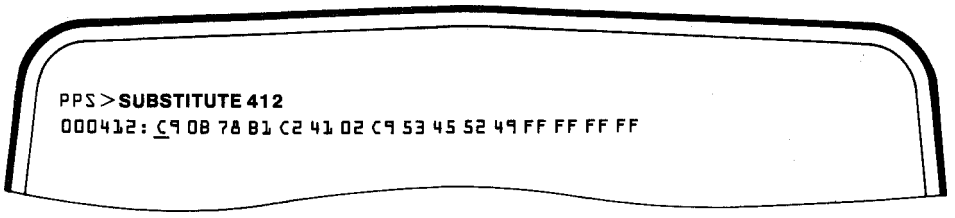
Key-in Sequence	Comments
C :F1:CODE4.IUP T P F RETURN	The F in the command sets the data switch to complement the data as it is being transferred from the file into the PROM. As with the file type specification in the previous example, you must specify the F because the default value data switch in the iPPS software is uncomplemented.

You can make other modifications on the data in a file by using the iPPS buffer. Assume that the data from a file has been loaded into the iPPS buffer. 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 illustrates using the LOADDATA command to load a constant into all the locations in the buffer that do not contain valid data.

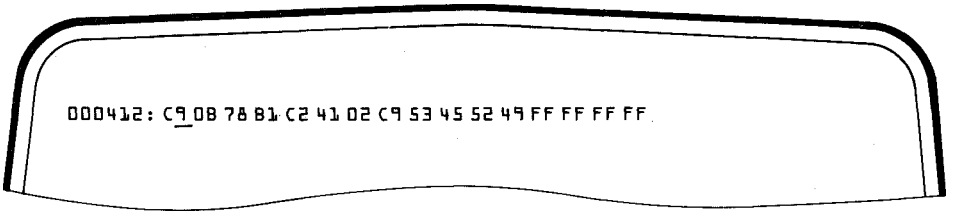


Key-in Sequence	Comments
LOADDATA B (413,7FF) W FF RETURN **DISPLAY B (410,L20)** RETURN	In this case, the buffer from address range 413H to 7FFH is loaded with the blank state of the PROM. Using the display command to display addresses 410H through 42FH shows that addresses 413H and above were indeed filled with the constant FF. Note again that abbreviations were used in some cases for keywords and prepositions.

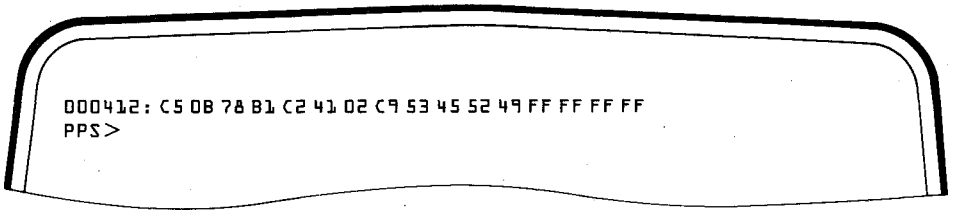
The SUBSTITUTE command lets you modify specific bytes in the iPPS buffer. In the following example it is assumed that the file has been loaded into the iPPS buffer.



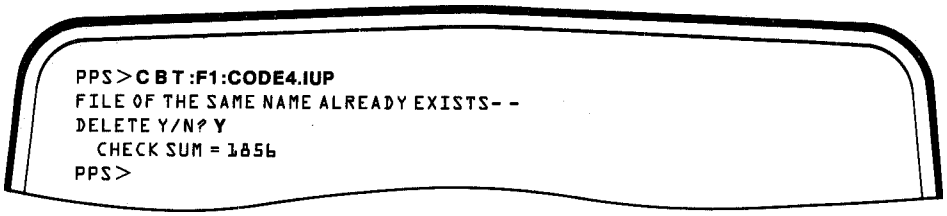
Key-in Sequence	Comments
SUBSTITUTE 412 RETURN	When the SUBSTITUTE command is invoked, the data at the selected address is displayed along with the data from up to the next 16 addresses. The number of bytes of data displayed depends on the number base specified: 16 for hexadecimal, 8 for octal, 4 for 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 with an underscore). By pressing the space bar, the <RUB-OUT> key, the < key, or the > key, you can move the cursor under the character you wish to change.



Key-in Sequence	Comments
SPACE	Pressing the space bar once moves the cursor to the least significant hexadecimal character of the data at address 412H.



Key-in Sequence	Comments
5 RETURN	The byte is changed from C9H to C5H. The return following the entry causes the iPPS software 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

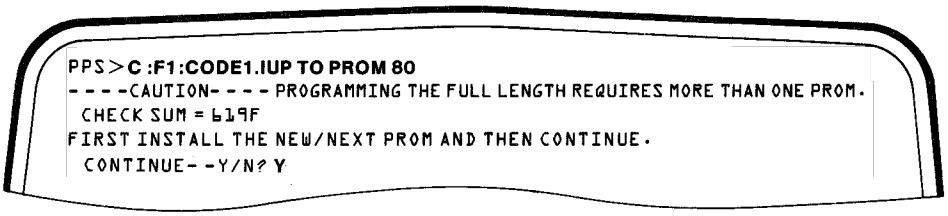
Y RETURN

Comments

In this example, the modified contents of the buffer are 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 store 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

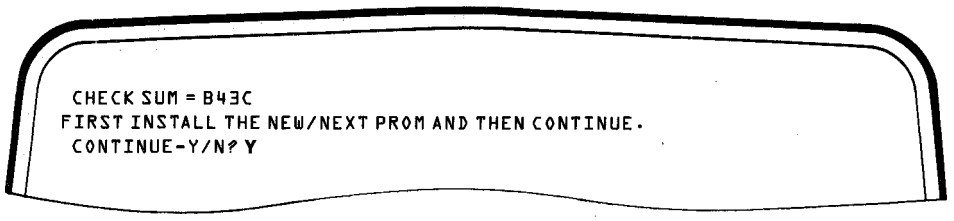
C :F1:CODE1.IUP TO PROM 80

RETURN

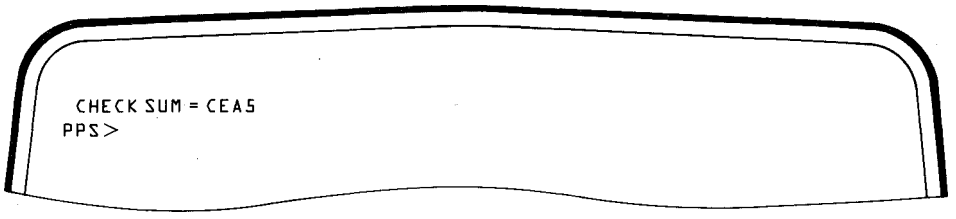
Y RETURN

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	Comments
Y RETURN	After you insert the second PROM, press Y and 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	Comments
	When the rest of the file has been read into the PROM, 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 is 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 is interleaved between the two PROMs, the even (or low) bytes stored in one PROM and the odd (or high) bytes stored in the other PROM. The FORMAT command handles this interleaving automatically.

In the following example, a file written in Intel 8086 hexadecimal format is interleaved into two PROM devices. (Read the description of the FORMAT command in Chapter 2 to familiarize yourself with the terminology of the command before doing this example.)

```
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):
*
```

Key-in Sequence

FORMAT DOUBLE.BYT (0,FFFH) RETURN

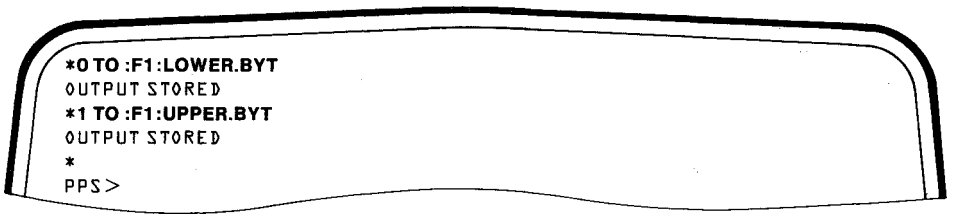
3 RETURN

2 RETURN

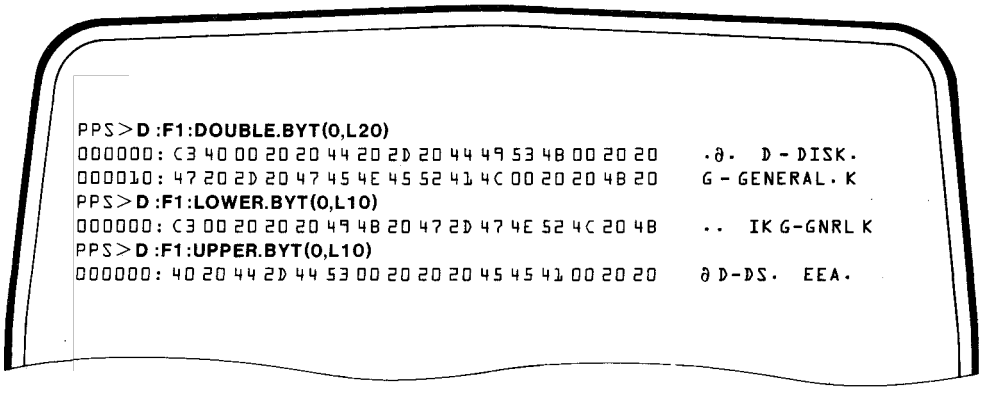
1 RETURN

Comments

In this example, a file called DOUBLE.BYT is 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 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. You are then prompted with an asterisk (*) to enter the output specification.



Key-in Sequence	Comments
0 TO :F1:LOWER.BYT RETURN **1 TO :F1:UPPER.BYT** RETURN RETURN	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). This example 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 specifies 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	Comments
D :F1:DOUBLE.BYT(0,L20) RETURN **D :F1:LOWER.BYT(0,L10)** RETURN **D :F1:UPPER.BYT(0,L10)** RETURN	By displaying the first few lines of the DOUBLE.BYT, LOWER.BYT, and UPPER.BYT files, you can see that the even address bytes in the DOUBLE.BYT file are stored in the LOWER.BYT file and the odd address bytes are stored in the UPPER.BYT file.

You can use the two files created with this FORMAT operation to program two PROMs, which you can then install in parallel to provide 16-bit wide address data words to a 16-bit microprocessor.

```
PPS>COPY :F1:LOWER.BYT TO PROM
 CHECK SUM = A51B
PPS>COPY :F1:UPPER.BYT TO PROM
 CHECK SUM = 84AC
PPS>
```

Key-in Sequence	Comments
COPY :F1:LOWER.BYT TO PROM RETURN **COPY :F1:UPPER.BYT TO PROM** RETURN	You must install a blank PROM in the personality module before entering each COPY command.

Masking a Parity Bit

The following example shows how the FORMAT command masks a parity bit in a file. In addition, all the lowercase letters in the file are changed to uppercase.

```
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
OUTPUT SPECIFICATIONS (<CR> TO EXIT:
*
```

Key-in Sequence	Comments
FORMAT :F1:PARITY.EVN (0,FFFH) RETURN **1** RETURN **1** RETURN **1** RETURN	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 one byte.

```
*0,1,2,3,4,F,6,F TO :F1:PARITY.MSK
OUTPUT STORED
*
PPS>
```

Key-in Sequence

0,1,2,3,4,F,6,F TO :F1:PARITY.MSK

RETURN

RETURN

Comments

In this example, bit 5 is the uppercase 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 lowercase to uppercase, and the parity bit is masked.

```
PPS>D :F1:PARITY.EVN (0,F) Y
00000000000000000000000000: 11110000 11100001 01110010 01101001  ..ri
00000000000000000000000100: 01110100 11111001 11111111 11101101  t...
00000000000000000000001000: 11100001 11110011 11101011 11111111  ....
00000000000000000000001100: 01101111 11110101 01110100 11111111  o.t.
PPS>D :F1:PARITY.MSK (0,F) Y
00000000000000000000000000: 01010000 01000001 01010010 01001001  PARI
00000000000000000000000100: 01010100 01011001 01011111 01001101  TY_M
00000000000000000000001000: 01000001 01010011 01001011 01011111  ASK_
00000000000000000000001100: 01001111 01010101 01010100 01011111  OUT_
PPS>
```

Key-in Sequence

D :F1:PARITY.EVN (0,F) Y

RETURN

D :F1:PARITY.MSK (0,F) Y

RETURN

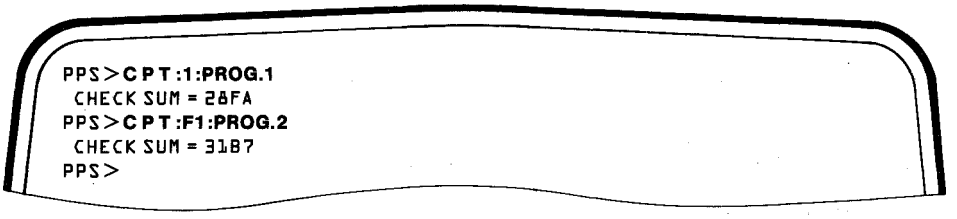
Comments

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 were changed to 0. The ASCII display to the right of the screen shows that the characters were switched to uppercase.

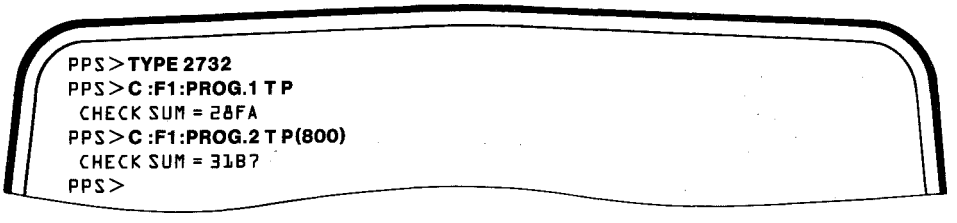
Programming a Single PROM With Code From a Number of Smaller PROMS

The iPPS software lets you copy the program code or data from a number of smaller PROMs 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.

First, you must copy the code for the two PROMs into two separate files.



Key-in Sequence	Comments
C P T :F1:PROG.1 [RETURN]	The contents of the two PROMs are stored in files titled PROG.1 and PROG.2. Between the two copy commands, the PROMs are exchanged in the PROM socket on the personality module.
C P T :F1:PROG.2 [RETURN]	



Key-in Sequence	Comments
TYPE 2732 [RETURN]	Having created the two files, change the PROM type to 2732 and insert the new blank PROM. Copy the two files into the PROM. Note that the second file must be copied into the upper 2K of the new PROM.
C :F1:PROG.1 T P [RETURN]	
C :F1:PROG.2 T P(800) [RETURN]	

Combining Two Files in the Buffer

The iPPS software lets you combine two or more files in the buffer, where you can edit or copy them to the URAM or a PROM. The following examples assume that the MODULE.1 file contains data at locations 10H through 1DH and that the MODULE.2 file contains data at locations 35H through 42H.

```
PPS>COPY MODULE.1(0) TO BUFFER C
   CHECK SUM = ECA0
PPS>C MODULE.2(0) T B
   CHECK SUM = D246
 PPS>
```

Key-in Sequence	Comments
COPY MODULE.1(0) TO BUFFER C  **C MODULE.2(0) T B** 	Specify a starting address of 0 with the COPY command so that the files will be placed in the buffer without being relocated. Data in the MODULE.1 file is placed in the buffer starting at location 10H. The C switch sets all other locations to the blank state. Data in the MODULE.2 file is placed in the buffer starting at location 35H. If the files had overlapped in the buffer, the overlapped locations in MODULE.1 would be overwritten. (Use the MAP command to determine address ranges in a file.)

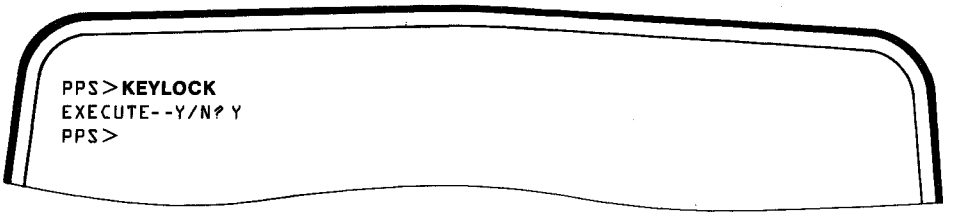
```
PPS>DISPLAY MODULE.1
000010: 53 41 4D 50 4C 45 20 44 41 54 41 20 23 31        SAMPLE DATA # 1
PPS>D MODULE.2
000035: 44 41 54 41 20 53 41 4D 50 4C 45 20 23 32        DATA SAMPLE # 2
PPS>D BUFFER (0,5FH)
000000: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
000010: 53 41 4D 50 4C 45 20 44 41 54 41 20 23 31 FF FF  SAMPLE DATA #1..
000020: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
000030: FF FF FF FF FF 44 41 54 41 20 53 41 4D 50 4C 45  .....DATA SAMPLE
000040: 20 23 32 FF FF FF FF FF FF FF FF FF FF FF FF FF  #2 ..............
000050: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
PPS>
```

Key-in Sequence	Comments
DISPLAY MODULE.1  **DISPLAY MODULE.2**  **D BUFFER(0,5FH)** 	Locations in the buffer that do not contain data from the files were set to the blank state.

Locking an EPROM

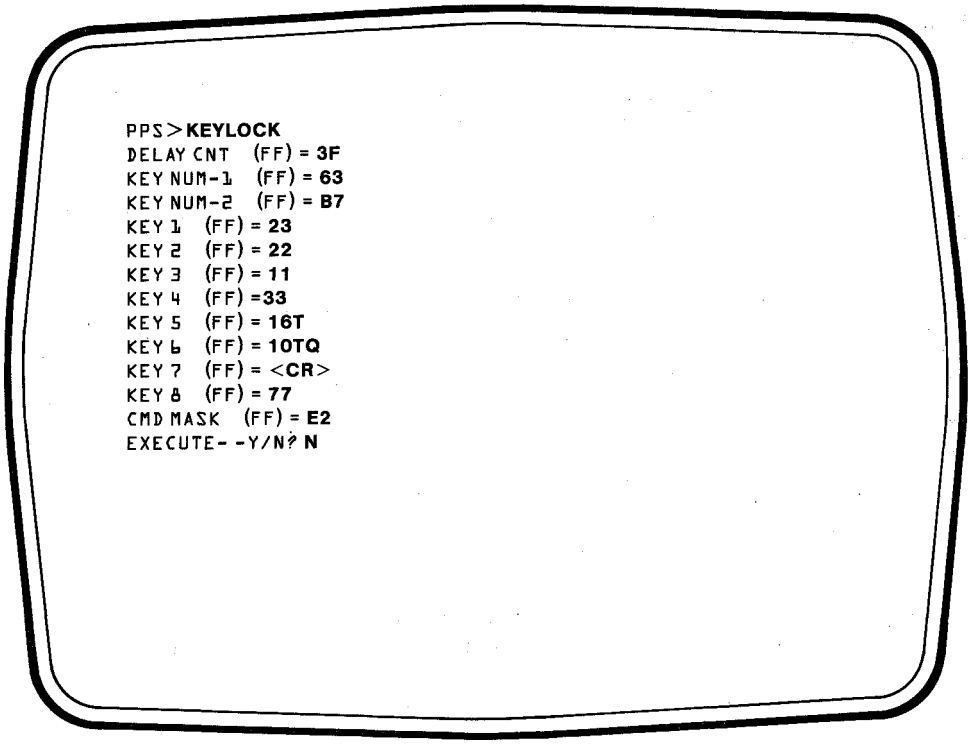
After programming an EPROM, you can protect it from unauthorized access by locking it with the KEYLOCK command (the KEYLOCK is not supported on all EPROMs).

The following example locks an 8751H microcontroller, which then cannot be unlocked without erasing it.



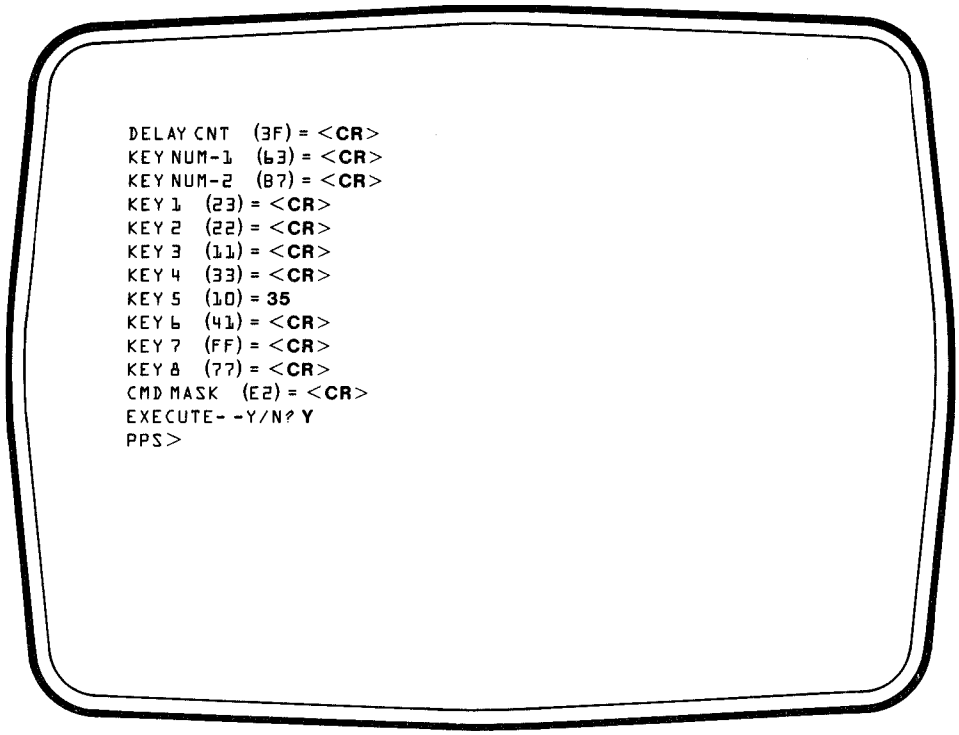
Key-in Sequence	Comments
KEYLOCK RETURN	Entering Y locks the EPROM. If you enter N, the command terminates and EPROM remains unlocked.
Y RETURN	

The following example programs an identity code into an authenticated EPROM. The identity code lets you unlock the EPROM without erasing it. The KEYLOCK command is interactive when used with authenticated EPROMs; the iPPS software prompts you for the parameters it needs. The values in parentheses are the values currently programmed into the EPROM.



Key-in Sequence	Comments
KEYLOCK RETURN	Enter values for the keys you want to change; use the carriage return to skip keys you do not want to change. Entering N to the EXECUTE prompt lets you verify the information you entered.
3F RETURN	
63 RETURN	
B7 RETURN	
23 RETURN	
22 RETURN	
11 RETURN	
33 RETURN	
16T RETURN	
101Q RETURN	
RETURN	
77 RETURN	
E2 RETURN	
N RETURN	

The parameters are displayed again because you entered N to the EXECUTE prompt.



Key-in Sequence	Comments
RETURN	All parameter values entered are saved in a buffer; they are programmed into the EPROM only if you enter Y for the EXECUTE prompt.
RETURN	
RETURN	
RETURN	
RETURN	
RETURN	
RETURN	
35 RETURN	
RETURN	
RETURN	
RETURN	
RETURN	
Y RETURN	

APPENDIX A
ERROR MESSAGES AND CONDITIONS

The general form of an iPPS error message is as follows:

error-group- -specific-error

where *error-group* is one of the following: SYNTAX ERROR, GENERAL ERROR, IUP ERROR, COMMAND ERROR, DISK ERROR, KEYLOCK ERROR, FORMAT ERROR, or CAUTION. Each type of *error-group* is described in the following sections.

Syntax Errors

Syntax errors indicate a problem with the syntax of the 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.

- -IUP ERROR- - - -LOCK FAILED AT *parameter.*

- -IUP ERROR- - - -ILLEGAL PARAMETER VALUE.

Command Errors

Command errors are related to the individual commands.

-COMMAND ERROR- -OVERLAY TEST FAILED.

-COMMAND ERROR- -VERIFY TEST FAILED.

-COMMAND ERROR- -BLANKCHECK TEST FAILED.

-COMMAND ERROR- -NO HELP FILE OPENED.

Disk Errors

Disk errors refer to ISIS operating system errors.

Non-fatal errors are displayed in the following format:

- -DISK ERROR- - -*ISIS-XX Error Message*

Fatal errors return program control to ISIS. Refer to the *ISIS-XX User's Guide* for ISIS error messages.

Format Errors

Format 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.

- -FORMAT ERROR- -DATA RANGE IN SOURCE DEVICE TOO SMALL.

Keylock Errors

-KEYLOCK ERROR- -VALUE EXCEEDS 8 BITS

Cautions

Cautions warn about abnormal but non-fatal conditions.

- - - -CAUTION- - - -FILE MAY BE TOO LARGE FOR THE
IPPS FILE HANDLER.

- - - -CAUTION- - - -PROGRAMMING THE FULL LENGTH REQUIRES
MORE THAN ONE PROM.

- - - -CAUTION- - - -THE FILE STRUCTURE MAY BE INCOMPLETE.

- - - -CAUTION- - - -DEFAULTING TO SERIAL CHANNEL *n*

- - - -CAUTION- - - -BLOCKSIZE HAS BEEN TRUNCATED
TO MATCH THE LOGICAL UNIT.

APPENDIX B
FILE FORMATS

The iPPS software reads and writes the following Intel file formats:

- 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 number 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 number 9800821.
- 286 Absolute Object. This format is described in Figure B-1.

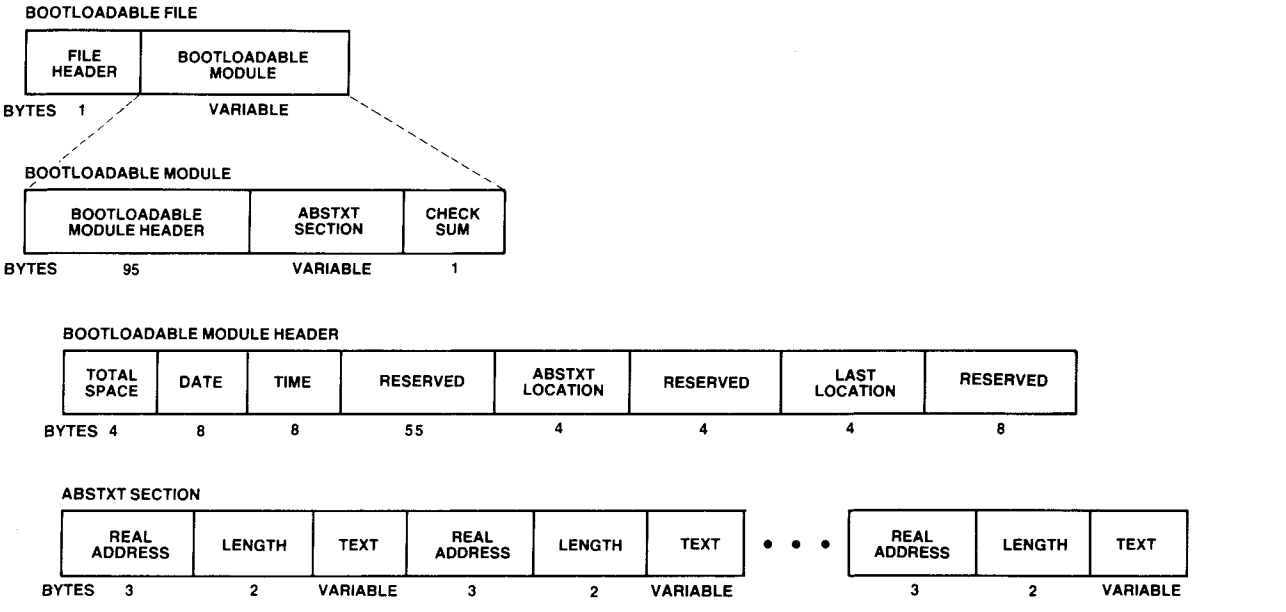


Figure B-1 286 Absolute Bootloadable File Format

The 286 file format allows addresses to range up to 16 Mbytes (FFFFFFH). The existing 8086 format allows addresses to range up to only 1 Mbyte (FFFFFH). This flexibility permits the iPPS software to handle future processors which may have a larger address space than the 8086 processor.

As shown in Figure B-1, a 286 bootloadable file is formatted as follows:

- A 1-byte FILE HEADER
- A variable number of BOOTLOADABLE MODULES

As shown in Figure B-1, each BOOTLOADABLE MODULE in the file consists of the following:

- A 95-byte BOOTLOADABLE MODULE HEADER as described next.
- 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 1-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 B-1, each BOOTLOADABLE MODULE HEADER consists of the following:

- A 4-byte value TOTAL SPACE which indicates the minimum number of bytes in main memory needed to load the module.
- An 8-byte data area (not used).
- An 8-byte time area (not used).
- A 55-byte reserved area that is not used by the iPPS software.
- A 4-byte ABSTXT LOCATION which is the offset in bytes of the start of the ABSTXT section from the start of the file.
- A 4-byte reserved area that is not used by the iPPS software.
- A 4-byte LAST LOCATION which is the offset in bytes of the last byte in the file from the start of the file.
- An 8-byte RESERVED field which is not currently used.

As shown in Figure B-1, each ABSTXT SECTION consists of a repeating sequence of the following three fields:

- A 3-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 2-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.

APPENDIX C
REFERENCE TABLES

Table C-1 shows 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, use the following steps:

1. Find the largest decimal number that does not exceed your starting number. Record its hexadecimal equivalent, including place holders (zeros).

2. Subtract the decimal number you found from the starting decimal number. The difference is your new starting number. If the new starting number is greater than zero, repeat steps one and two.

3. Add the hexadecimal equivalents.

Table C-2 contains base conversions for decimal, binary, hexadecimal, and octal.

Table C-3 contains powers of two.

Table C-4 shows conversions between powers of 2 and 16.

Table C-5 contains powers of 16.

Table C-6 lists ASCII code.

Table C-7 defines ASCII control code.

Table C-8 translates ASCII code to binary.

Table C-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
7654		3210		7654		3210	
Byte				Byte			

Table C-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 C-6 ASCII Code List (continued)

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	110	48	H	104	150	68	h
73	111	49	I	105	151	69	i
74	112	4A	J	106	152	6A	j
75	113	4B	K	107	153	6B	k
76	114	4C	L	108	154	6C	l
77	115	4D	M	109	155	6D	m
78	116	4E	N	110	156	6E	n
79	117	4F	O	111	157	6F	o
80	120	50	P	112	160	70	p
81	121	51	Q	113	161	71	q
82	122	52	R	114	162	72	r
83	123	53	S	115	163	73	s
84	124	54	T	116	164	74	t
85	125	55	U	117	165	75	u
86	126	56	V	118	166	76	v
87	127	57	W	119	167	77	w
88	130	58	X	120	170	78	x
89	131	59	Y	121	171	79	y
90	132	5A	Z	122	172	7A	z
91	133	5B	[	123	173	7B	{
92	134	5C	\	124	174	7C	|
93	135	5D	]	125	175	7D	}
94	136	5E	^	126	176	7E	~
95	137	5F	_	127	177	7F	DEL

Table C-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	Inquiry	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 Separator	31
SP	Space	32
DEL	Delete	127

Table C-8 ASCII Code in Binary

MSB / LSB	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
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

REQUEST FOR READER'S COMMENTS

Intel's Technical Publications Departments attempt to provide publications that meet the needs of all Intel product users. This form lets you participate directly in the publication process. Your comments will help us correct and improve our publications. Please take a few minutes to respond.

Please restrict your comments to the usability, accuracy, readability, organization, and completeness of this publication. If you have any comments on the product that this publication describes, please contact your Intel representative. If you wish to order publications, contact the Intel Literature Department (see page ii of this manual).

1. Please describe any errors you found in this publication (include page number).

2. Does the publication cover the information you expected or required? Please make suggestions for improvement.

3. Is this the right type of publication for your needs? Is it at the right level? What other types of publications are needed?

4. Did you have any difficulty understanding descriptions or wording? Where?

5. Please rate this publication on a scale of 1 to 5 (5 being the best rating). _____

NAME _____ DATE _____

TITLE _____

COMPANY NAME/DEPARTMENT _____

ADDRESS _____

CITY _____ STATE _____ ZIP CODE _____
(COUNTRY)

Please check here if you require a written reply. ☐

WE'D LIKE YOUR COMMENTS ...

This document is one of a series describing Intel products. Your comments on the back of this form will help us produce better manuals. Each reply will be carefully reviewed by the responsible person. All comments and suggestions become the property of Intel Corporation.

NO POSTAGE
NECESSARY
IF MAILED
IN U.S.A.

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 79 BEAVERTON, OR

POSTAGE WILL BE PAID BY ADDRESSEE

Intel Corporation
5200 N.E. Elam Young Parkway.
Hillsboro, Oregon 97123

DSHO Technical Publications

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, California 95051 (408) **987-8080**

Printed in U.S.A.

DS-103/3K/0484/WCP
INSTRUMENTATION