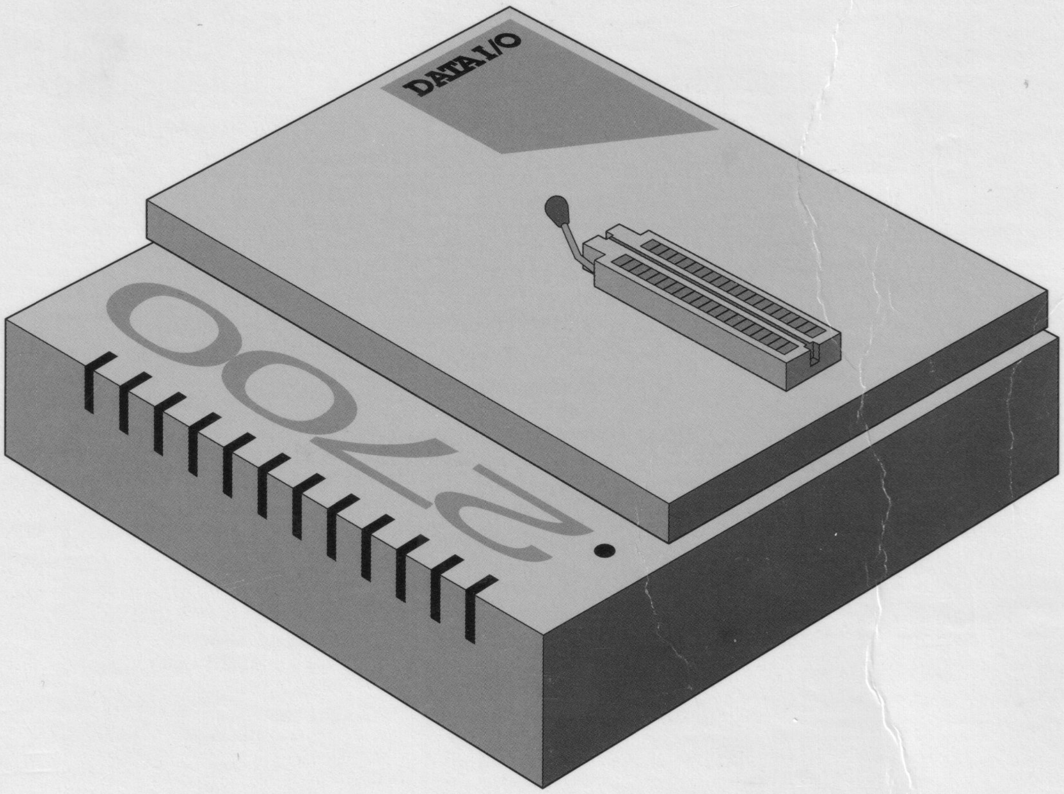
2700

Programming System



User Manual

096-0186

DATA I/O

2700
User Manual

November 1996 096-0186-005

Data I/O Corporation
10525 Willows Road N.E., P.O. Box 97046
Redmond, Washington 98073-9746 USA
(206) 881-6444

Acknowledgments:

Data I/O is a registered trademark and MatchBook is a trademark of Data I/O Corporation.

Data I/O Corporation acknowledges the trademarks of other organizations for their respective products or services mentioned in this document.

Table of Contents

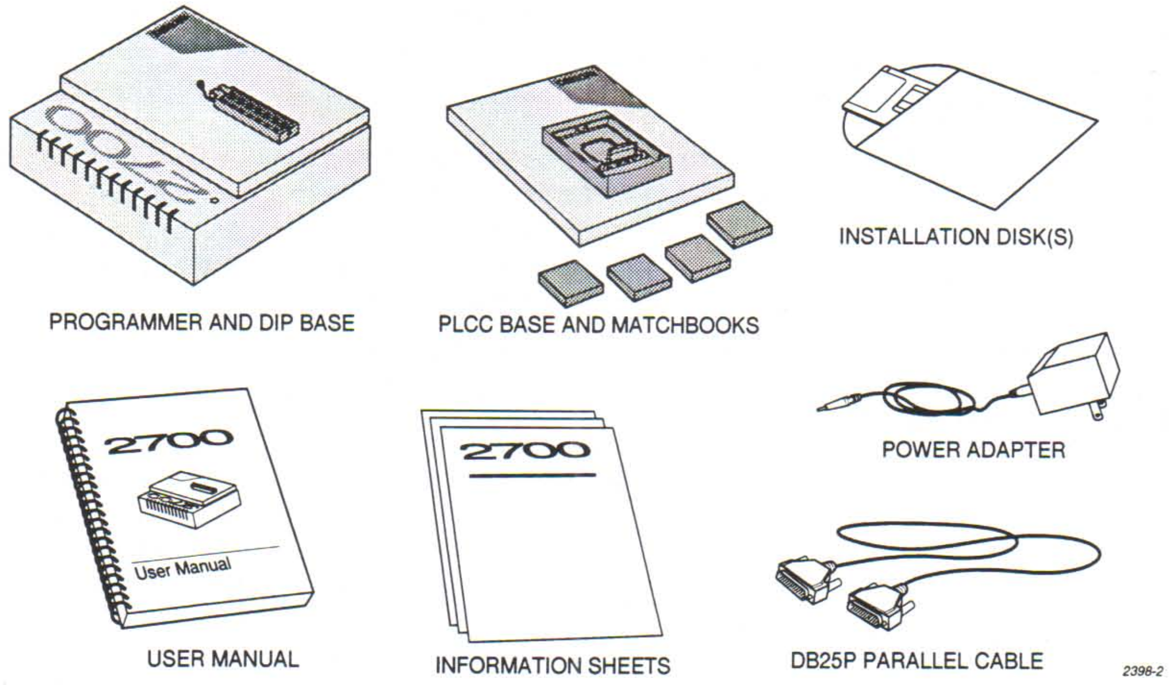
1. Before You Begin

Before you start setting up or using your 2700 programmer, read the information in this chapter. It will help you become familiar with the product and this User Manual.

Package Contents

The contents of the 2700 shipping package are shown in the following figure.

CAUTION: ***To prevent damage to the programmer, use only the power adapter shipped with the 2700.***



Registration Card

So that we can notify you of new products, options, and additional offers, please take the time to fill out the End User Registration Card included at the back of this manual.

Product Definition

The 2700 programmer is a precision tool designed to program and verify programmable device technologies and packages (such as DIPs, PLCCs, and JLCCs).

The 2700 uses a Windows interface to provide an accessible programming environment, the PC's parallel port to communicate with the programmer, and device bases to ensure that you get the specific device support you need for your programming application.

Which Devices Are Supported?

The standard bases included with the 2700 support DIP devices up to 48 pins and PLCC devices up to 44 pins. If you want to program other device packages, you may need to purchase a PPI base and the appropriate PPI or an adapter. The Device List indicates which base is needed for each device. For more information, call Data I/O Customer Support.

Online Device List

The 2700's Device List will be copied to your hard drive during installation. Refer to the Device List for a complete list of current device support and the base that is needed for each device. You can print the **devices.txt** file using a word processing program, or you can load the file into a spreadsheet and print it.

Update Information

Updating the 2700 software ensures that you have the most current device algorithms. You need to update when you want to:

- Use a new version of 2700 system software
- Add support for a specific device
- Use the latest support for a particular device

Updates are made available several times each year. To ensure that you are notified when updates are available, complete and return your Registration Card.

Documentation

The components of the 2700 documentation are described in the following sections. The online documentation is the most complete source of information, although the printed documentation covers most basic operation information.

User Manual

The User Manual is intended to supplement the online Help. For information on accessing and using online Help, see page 4.

An overall view of the contents of each chapter can be found in the Table of Contents. Acronyms and other terminology are defined in the Glossary.

Written Conventions

Throughout this document you will find instructions such as:

"Choose **Open...** from the File menu (ALT, F, O)."

The information before the parentheses tells how to perform the task using the mouse; the information within the parentheses tells how to perform the task using the keyboard only.

To choose **Open...** from the file menu in this example, using the keyboard you would press the ALT key, then the F key, then the O key.

Safety Terminology

CAUTION statements identify conditions or practices that could result in damage to the equipment or other property.

Additional Safety Information can be found in the "Safety Summary," Appendix D.

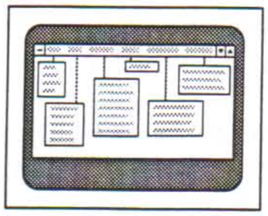
Information Sheets

Because they contain information that may be updated frequently, the information sheets are separate from the User Manual.

The following information sheets are included with the 2700:

- **International Customer Support Contacts**—Phone numbers for contacting the customer support representative for your country.
- **Bulletin Board Service**—Describes the information available from the Bulletin Board Service (BBS) and explains how to access it. (See also the section on the BBS on the following page.)

Online Documentation



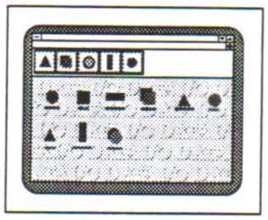
The online documentation contains more detailed information than that contained in this User Manual. For example, the online Help goes into detail on running blank checks and sumchecks on both logic and memory devices. Online documentation for the 2700 can be categorized as follows:

- **Readme.txt**—This file is copied to the 2700 directory during installation. It includes instructions on installing the 2700 software and moving files to another directory, and describes port driver preferences.
- **Online Help**—The online Help can be accessed, once the 2700 software is loaded, by selecting Help from the menu bar or clicking the Help button.

?	**Contents**	Displays an alphabetical list of the Help topics available for the 2700. Select from the list to access Help on the selected topic.
	Messages	Displays a list of error messages. Select from the list to access Help on a specific message.
	Using Help	Contains general directions for using Windows Help.

For more information, see page 14.

Bulletin Board Service



The Data I/O Bulletin Board Service (BBS) may be accessed either via the Internet or by dial-up modem to Data I/O. The BBS enables you to:

- Obtain a wide range of information on Data I/O products, including current product descriptions, new revision information, technical support information, application notes, and other miscellaneous information.
- Access device support information.
- Request support for a particular device.
- Leave messages for the BBS system operator, Customer Support personnel, or other customers.
- Download many DOS and Windows utilities.

Online Help files and the BBS information sheets provide more information about the BBS and its capabilities.

The BBS can be accessed from the following countries:

Germany	+49-(0)89-8585833
Japan	+81-33779-2233
United States	+1-206-882-3211

For other BBS numbers available in your area, please call your local Customer Support number.

World Wide Web

The Data I/O Home Page on the World Wide Web (WWW) includes links to online information about technical products, general information about Data I/O, a list of sales offices, and technical user information such as application notes and device lists.

To access the World Wide Web, you will need an Internet account with Web access, and a Web browser such as Netscape or Mosaic.

The address of the Data I/O home page is:

http://www.data-io.com

2. Setting Up

Before you can use the 2700 programmer, the host computer (PC) and the programmer need to be set up to communicate with each other. This requires that all cables are properly connected and that the 2700 software is installed on the PC, as described in this chapter.

System Requirements

You will need the following items to use the 2700 with your PC:

- IBM-compatible PC, 386 minimum, running Windows 3.1 or higher

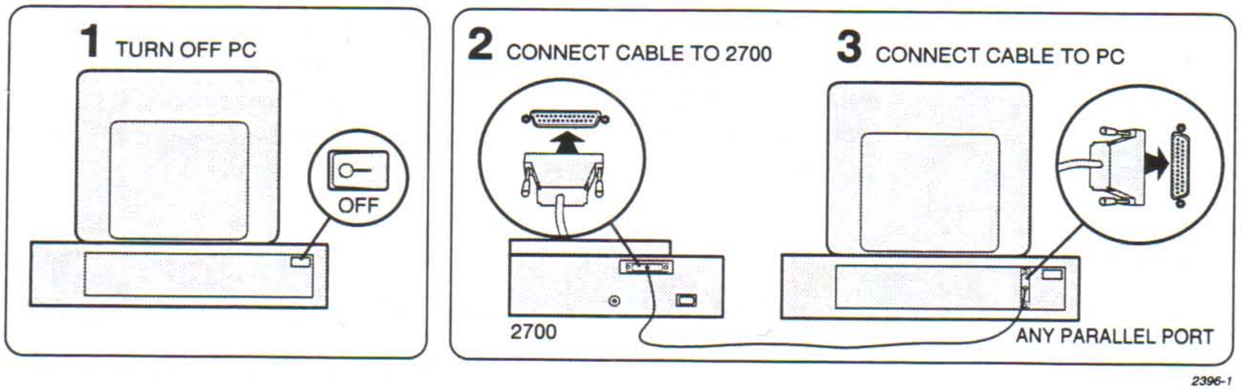
---

Note: *The more powerful the PC, the faster the programmer will be able to perform.*

- One free parallel port with nothing else attached
- Minimum 3 MB extended memory; 4 to 8 MB recommended
- 5 MB minimum (10 MB recommended) of free hard disk space for the 2700 software
- 3.5-inch high-density disk drive
- Microsoft-compatible mouse (optional, but recommended)
- VGA color monitor (minimum)

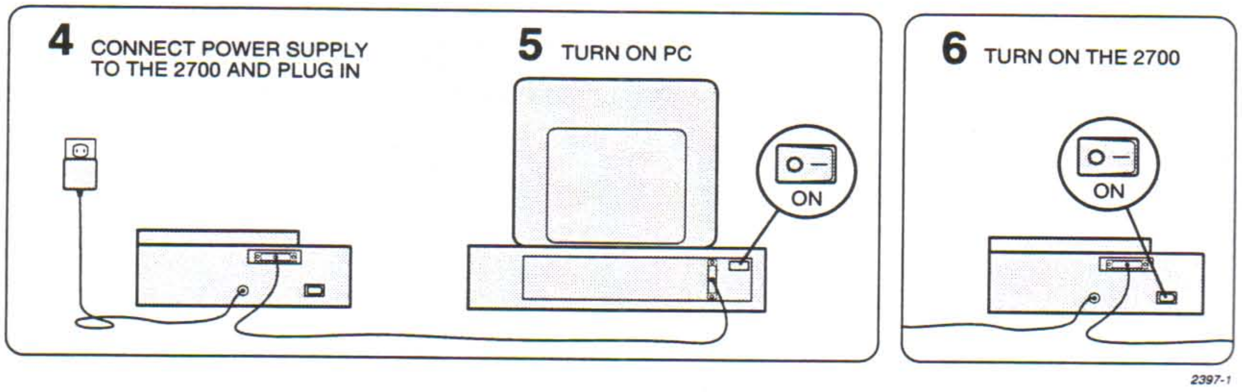
Connect Hardware

Set up your PC and the 2700 as shown in the following figures.



CAUTION: ***Make sure that the parallel port is dedicated to 2700 operation only. Tighten the parallel cable connector screws to ensure a good chassis-to-chassis ground.***

Note: *Before you turn on the 2700, a programming base must be installed. To install a base, see page 15.*



Install Software

Before you install the 2700 software, do the following:

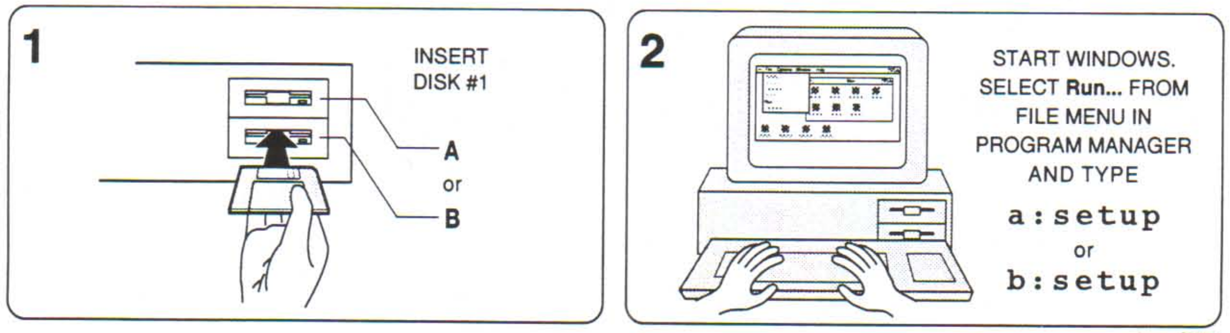
- Disable any anti-virus software
- Connect the 2700 to the PC
- Close any open Windows applications

CAUTION: ***Neglecting to exit programs running in Windows while the 2700 is being installed may damage the programs.***

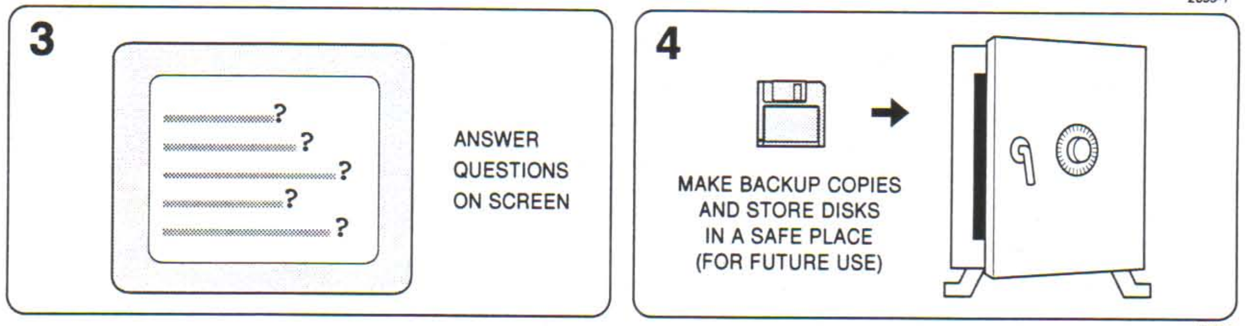
A read-me file, **readme.txt**, is copied to the 2700 directory during installation. Double click the "Read Me" icon to view it.

If you receive error messages during the installation process, you might want to refer to the information in the read-me file. You can also find information on error messages in Chapter 6.

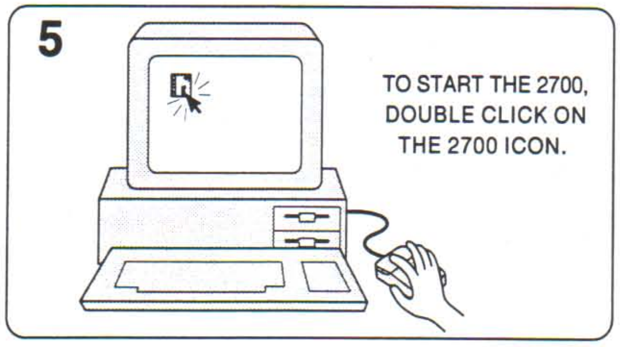
Install the software as shown in the following figures.


1
INSERT DISK #1
A
or
B
2
START WINDOWS. SELECT Run... FROM FILE MENU IN PROGRAM MANAGER AND TYPE
a:setup
or
b:setup

2399-1


3
ANSWER QUESTIONS ON SCREEN
4
MAKE BACKUP COPIES AND STORE DISKS IN A SAFE PLACE (FOR FUTURE USE)

1703-3



2400-1

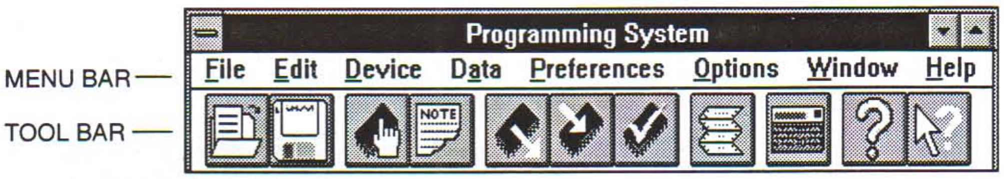
After you start the software, the 2700 performs a self-text that lasts a few minutes. Do not attempt to change bases or select a device until the test is finished.

3. Getting Started

The information in this chapter is organized as follows:

Learning About Menus, Commands, and the System Log

All commands are available from the menu bar and its pulldown menus. Buttons in the tool bar provide shortcuts to some of the commands.



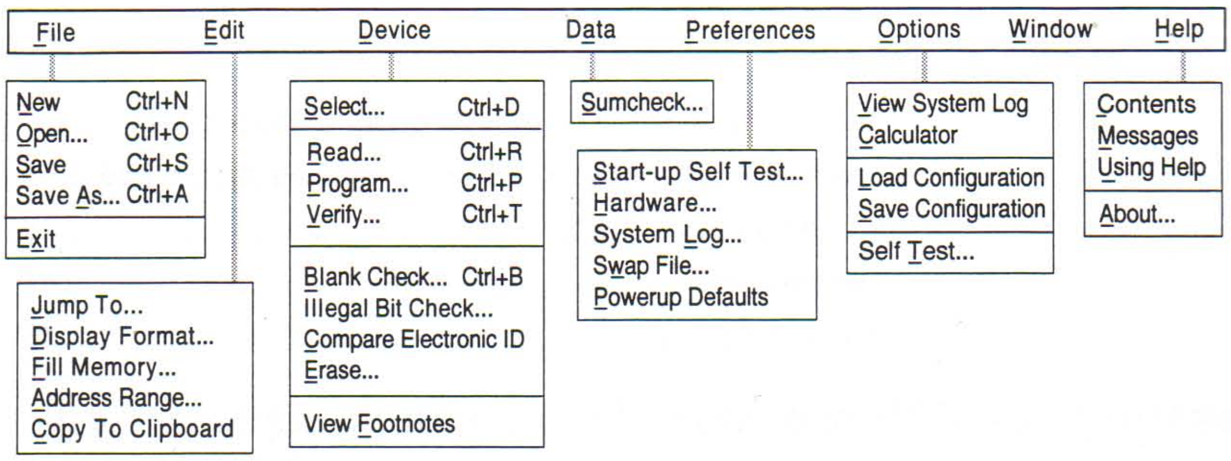
To select menus and commands, perform one of the following:

- Click your selection by using the mouse to access the pulldown menus.
- Press **ALT** and then the underlined letter on the menu bar (such as **ALT** and **F** to see the **File** menu). Then press the underlined letter of the menu selection (such as **O** to open a file) you wish to choose.
- Press **CTRL** + ***Letter*** for those items with shortcut keys. For example, **CTRL + D** brings up the **Select a Device** dialog box. The list of commands starting on page 12 includes shortcut keys. Commands followed by ellipses (...) have submenus.

You can also access a command by clicking its tool bar button, if one is available.

The menus, commands, and tool bar buttons are described briefly in the following sections. For detailed information on these items and the submenus, refer to Chapter 5, "Commands" or the online Help (see page 14 for details).

Menu Bar, Pull-down Menus, and Commands



File	**New (Ctrl+N)**	Creates a new file.
	Open... (Ctrl+O)	Reads an existing data file into memory. A display box shows the files in the current directory.
	Save (Ctrl+S)	Saves to disk the data file loaded in memory, overwriting the current version.
	Save As... (Ctrl+A)	Specifies a new name for the active file (or part of the file) and saves it as a separate file. The file also can be saved in a new format.
	Exit	Ends the current 2700 session.
Edit	**Jump To...**	Specifies an address to jump to in the active data file.
	Display Format...	Specifies the format in which the data will be displayed.
	Fill Memory...	Fills a specified address range in user memory with a data pattern.
	Address Range...	Determines the range of memory that can be edited.
	Copy To Clipboard	Copies the selected data to the Clipboard, a temporary storage area in memory.

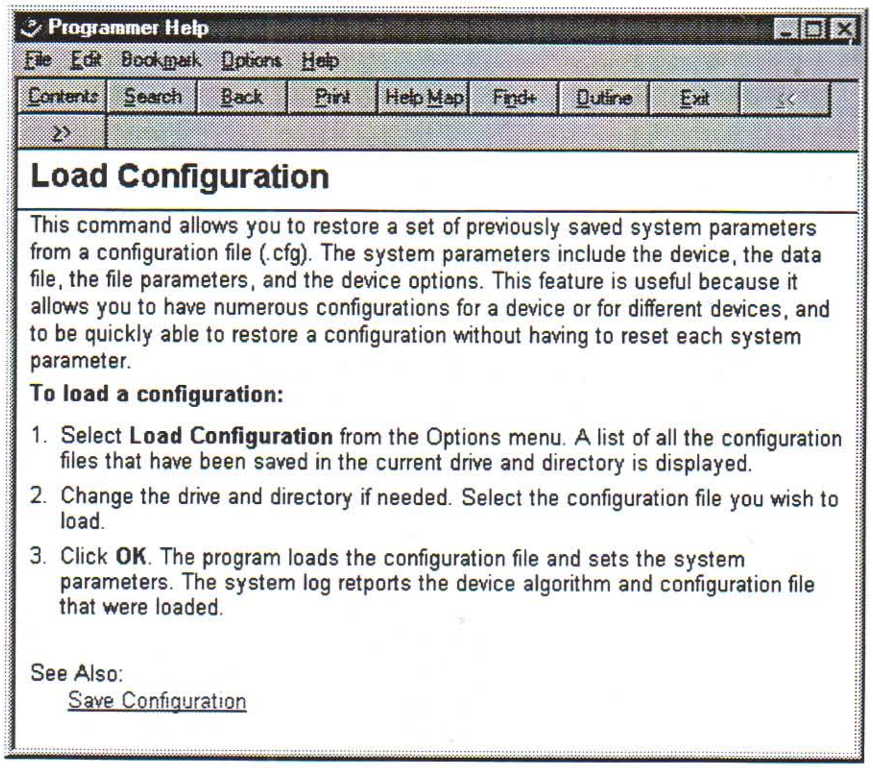
Device	**Select... (CTRL+D)**	Specifies the manufacturer and type of device you want to program.
	Read... (CTRL+R)	Reads data in the socketed device into memory.
	Program... (CTRL+P)	Transfers data in memory into the device in the programmer socket.
	Verify... (CTRL+T)	Compares the data in the socketed device to the data in memory to ensure that they match.
	Blank Check... (CTRL+B)	Performs a blank check operation on the selected device by searching the socketed device for programmed locations.
	Illegal Bit Check...	Performs an illegal bit check on the selected device by comparing device data against memory to determine if the device has programmed locations of incorrect polarity.
	Compare Electronic ID	Performs an Electronic ID Check on the selected device by verifying that the Electronic ID in the device is compatible with the selected device.
	Erase...	Erases the selected electrically-erasable device (if the selected device supports this feature).
	View Footnotes	Displays the footnote, if there is one, of the currently selected device. The footnote contains device-specific information.
Data	**Sumcheck**	Computes a sumcheck for data in the open file.
Preferences	**Start-up Self-test...**	Activates an automatic 2700 self-test on powerup, and enables you to set various test routines.
	Hardware...	Sets the parallel port to which the 2700 is connected, and selects which port driver to use when communicating with the 2700.
	System Log...	Sets the startup preferences for the System Log and other options.
	Swap File...	Determines the size of user memory and the memory buffer you want the 2700 to use when swapping data during an operation.
	Powerup Defaults	Allows you to start up the program with preset parameters.

Options	**View System Log**	Displays the System Log, which contains system and device information.
	Calculator	Starts the calculator.
	Load Configuration	Allows you to restore a set of previously saved system parameters from a configuration file.
	Save Configuration	Allows you to save a set of system parameters to a configuration file for future use.
	Self Test...	Enables you to specify the elements to be tested, run the tests, and view the results.
Window	**New Window**	Opens another window that has the same display settings and contents as the previously active window.
	Cascade	Resizes and layers the open windows, leaving the title bars visible.
	Tile	Resizes and arranges the open windows side by side.
	Arrange Icons	Evenly arranges the icons in a group window.
Help	**Contents**	Displays an alphabetical list of the Help topics available for the 2700. Select from the list to access Help on the selected topic.
	Messages	Displays a list of error messages. Select from the list to access Help on a specific message.
	Using Help	Contains general directions for using Windows Help.
	About...	Displays the current software version number and copyright information.

Note: *For context-sensitive help on a selected topic, press F1 or click on the question mark button (pictured at right) on the tool bar.*

You can also click on the Help button from a dialog box for help on that dialog box.

An example of a Help topic is shown below.


Programmer Help
File Edit Bookmark Options Help
Contents Search Back Print Help Map Find+ Outline Exit <<
>>

Load Configuration

This command allows you to restore a set of previously saved system parameters from a configuration file (.cfg). The system parameters include the device, the data file, the file parameters, and the device options. This feature is useful because it allows you to have numerous configurations for a device or for different devices, and to be quickly able to restore a configuration without having to reset each system parameter.

To load a configuration:

1. Select **Load Configuration** from the Options menu. A list of all the configuration files that have been saved in the current drive and directory is displayed.
2. Change the drive and directory if needed. Select the configuration file you wish to load.
3. Click **OK**. The program loads the configuration file and sets the system parameters. The system log retports the device algorithm and configuration file that were loaded.

See Also:
Save Configuration

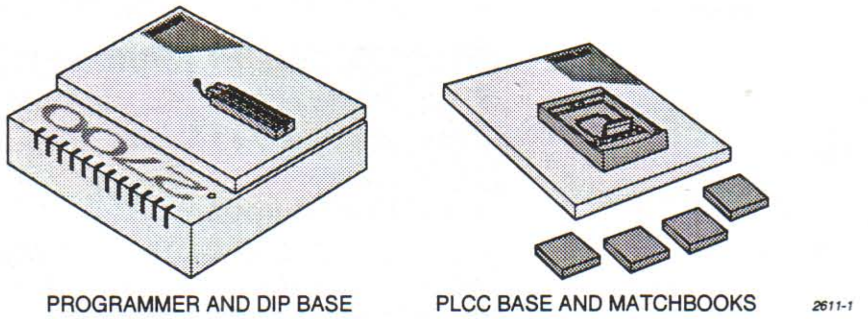
System Log

The System Log, which displays information such as checksums, error messages, and operation status, appears at the bottom of the screen after powerup. You can toggle it off by clicking the button in the tool bar. You can also resize the Log or move it.

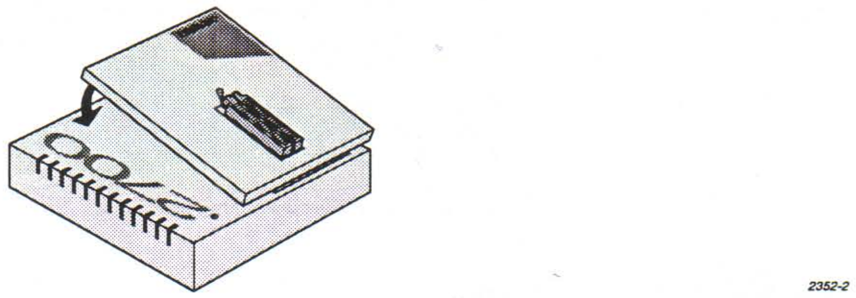
Installing a Base

Before you insert a device into a socket or download (open) a file, you must install a base that supports the device. Depending on which base you use, you also may need to install an adapter on the base. To determine the base and adapter configuration required for the device, refer to the Device List loaded with the 2700 software (see page 2).

The 2700 is packaged with two bases: the DIP base and the PLCC base. The following figure shows the DIP base mounted on the programmer and the PLCC base with the MatchBooks.



To change from one base to another, remove the installed base by carefully pulling up the end toward the back panel of the 2700, and lift the lip out of the slot on the front. To install a new base, insert the lip into the slot and lower the back of the base, as shown in the figure below.



Selecting Devices

You must select a device before you can program, verify, or read data from a device. You can select a device either before or after you insert a device in the socket. To select a device, click the Select Device button in the tool bar and make the appropriate selections. For more information on selecting devices, see the tutorials in Chapter 4.

To view additional information about a device, click the Footnote button in the tool bar.

Inserting Devices

After the appropriate base is installed, you can insert a device. Directions on properly inserting specific devices are outlined in the following sections.

CAUTION: Incorrect insertion can damage or destroy a device, so follow the device insertion procedures carefully.

The figure on page 21 shows several profiles of devices, including top views and side views.

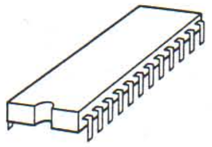
Locating Pin 1

To properly insert a device, you must use pin 1 on the device as a reference point. Identifying pin 1 will simplify the proper insertion of the device.

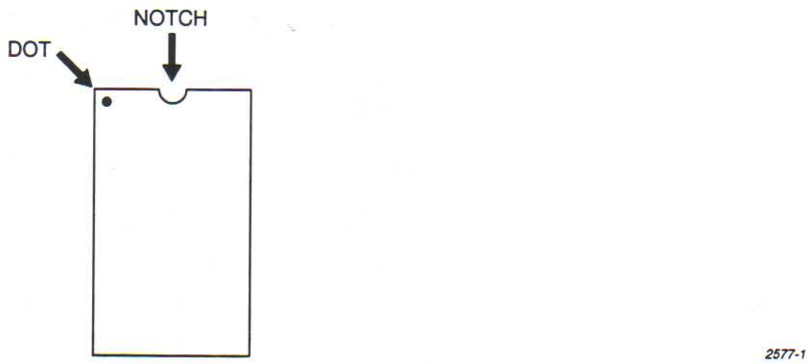
Depending on the device you are using, pin 1 could be indicated in several different ways. The following sections describe how to locate pin 1 on some of the most common devices and how to insert the device in the socket.

CAUTION: ***Devices are static sensitive. Operate your programmer at an antistatic workstation. To avoid electric shock and damage to the devices, use an antistatic wrist strap containing a 1 MΩ (minimum) to 10 MΩ (maximum) isolating resistor.***

Inserting a DIP Device in a DIP Base

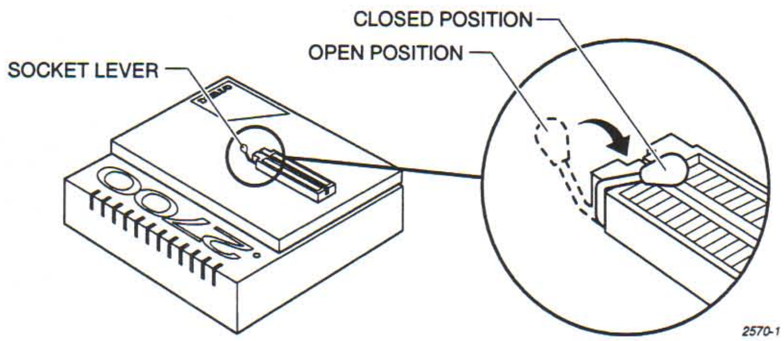


Pin 1 on a DIP package is generally indicated by a notch on one end of the device. Pin 1 also may be indicated by a stamped or recessed dot on one corner of the device.

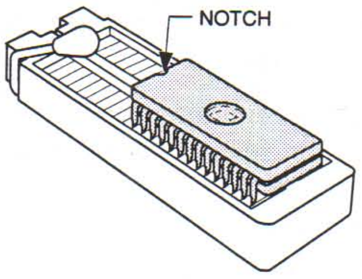


1. Lift the socket lever to the open position.

   The enlarged diagram below shows the DIP base socket lever in both open and closed (locked) positions.



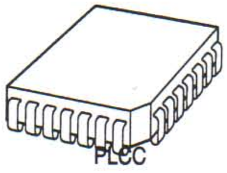
2. With the notched end facing the top of the socket, place the device in the socket so that the bottom of the device is aligned with the bottom of the socket (bottom justified).

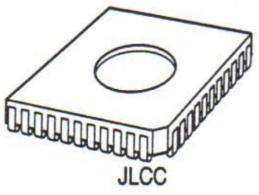


2628-2

3. Lower the socket lever to lock the device into the socket.
4. If you have not done so already, select the inserted device (see page 24).

Installing a MatchBook on a PLCC Base and Inserting a PLCC or JLCC Device

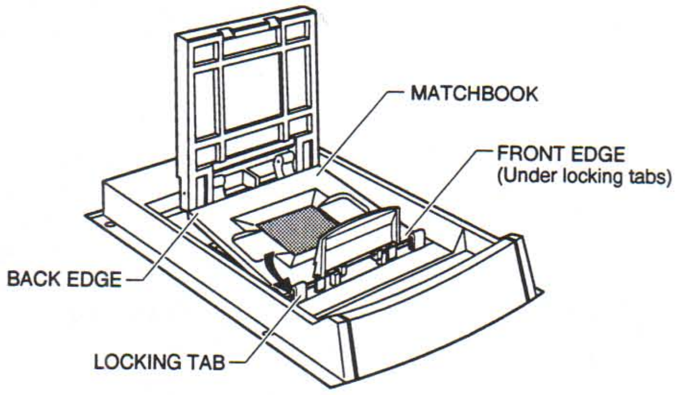




The MatchBook holds a PLCC or JLCC device on the PLCC base. When the device is locked in place, the conductive pad on the base forms a path between the 2700 and the device.

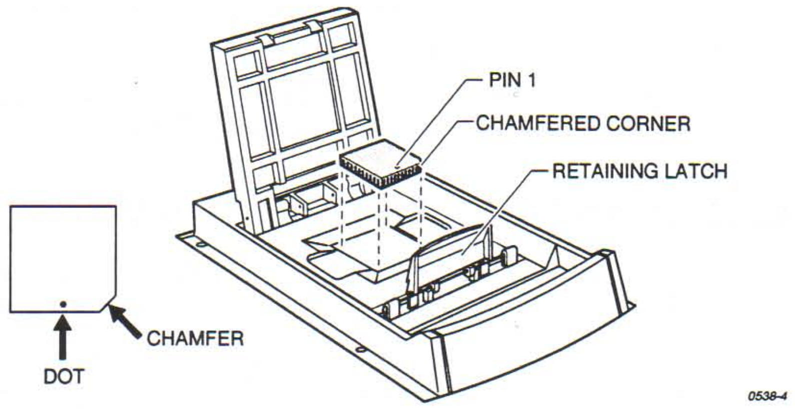
To install a MatchBook and a PLCC or JLCC device, perform the following steps.

1. Install the PLCC Base on the 2700.
2. Select the appropriate MatchBook for the device you wish to program.
3. Open the MatchBook 90 degrees, set its front edge under the two tabs at the front of the base opening, and lower its back edge into place. Each MatchBook has a small molded dot that represents pin 1 and an angled corner that can be used to align chamfered corners of devices.

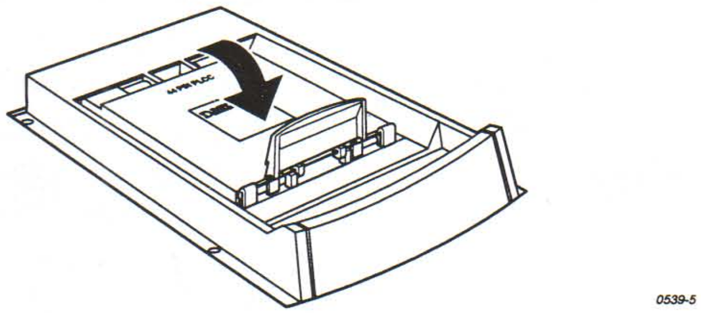


0537-4

4. Orient the device you want to use so that pin 1 is on the side closest to the retaining latch. Pin 1 on a PLCC or JLCC package is generally indicated by the chamfered corner of the device. Pin 1 also may be indicated by a stamped or recessed dot on one side of the device.

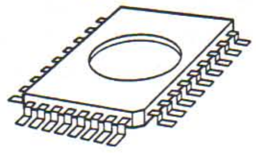


5. Insert the device into the open MatchBook.
6. Close the MatchBook and press the retaining latch forward with your thumb until the latch snaps into place.



7. If you have not already done so, select the inserted device (see page 24).

Inserting a QFP Device into a PPI Adapter

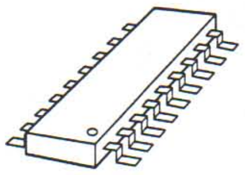


To accommodate QFP devices, the 2700 programmer must be fitted with a PPI base and the appropriate adapter.

Pin 1 on the QFP device is usually at the front left of the device. Orient the QFP device according to the icon or indicator on the adapter.

For more detailed instructions, refer to the User Manual included with the PPI adapter.

Inserting an SOIC Device into a PPI Adapter

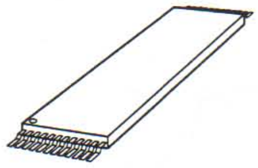


To accommodate SOIC devices, the 2700 programmer must be fitted with a PPI base and the appropriate adapter.

When inserting an SOIC device, make sure pin 1 (represented by the notch or the circle indented into the package at one end of the device) is to the left.

For more detailed instructions, refer to the User Manual included with the PPI adapter.

Inserting a TSOP Device into a PPI Adapter



To accommodate TSOP devices, the 2700 programmer must be fitted with a PPI base and the appropriate PPI adapter.

When inserting a TSOP device, make sure pin 1 (represented by the circle or triangle indented into the package at one end of the device) is in the upper left corner.

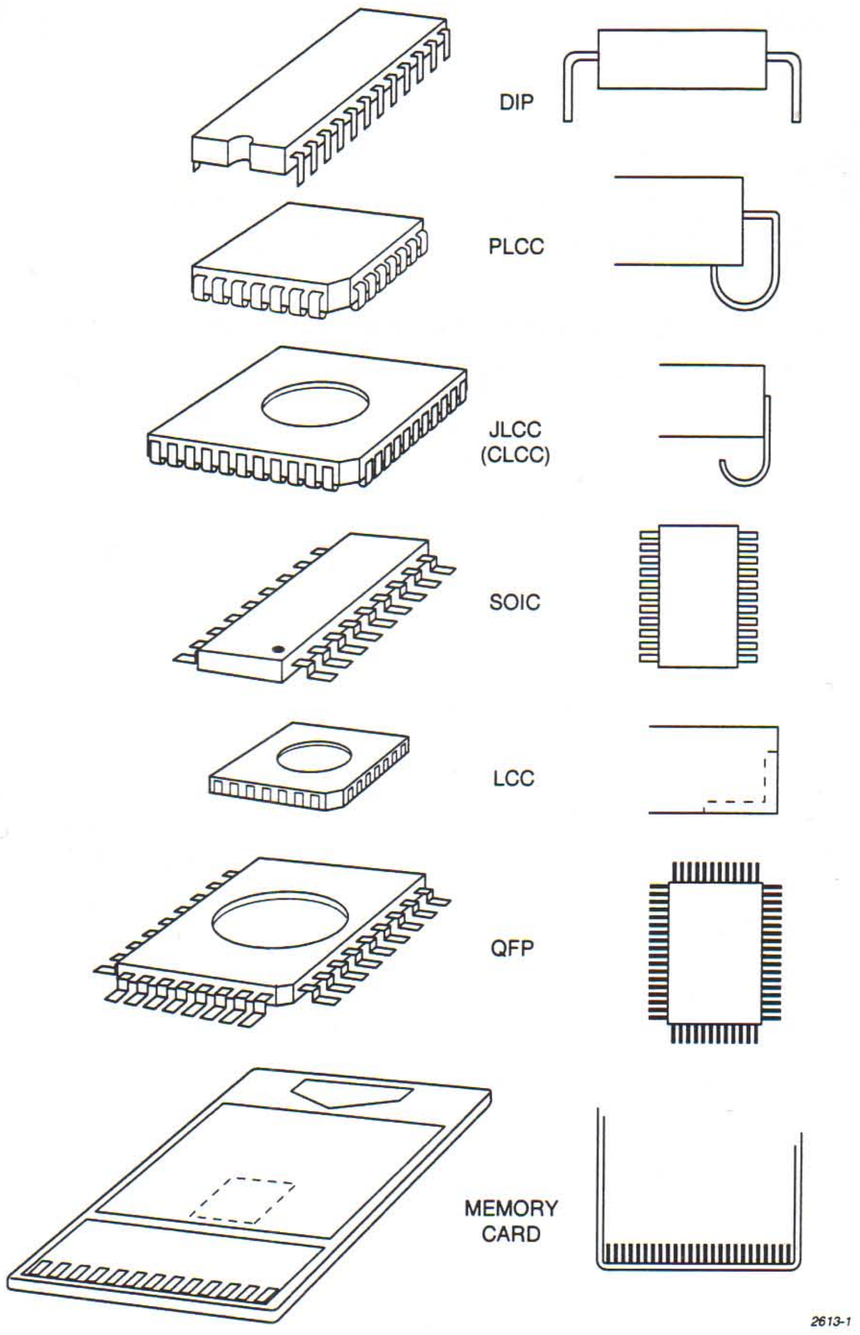
For more detailed instructions, refer to the User Manual included with the adapter.

Inserting Other Devices

In order to accommodate other devices (such as SDIPs and Memory Cards), the 2700 programmer must be fitted with a base and adapter appropriate for the device.

When inserting a device into an adapter socket, consult the illustrations on the adapter for instructions on how to orient the device. Most devices are keyed so that they can fit into the socket only one way. For more information, refer to the User Manual and any additional documentation included with the specific adapter.

Devices Used with the 2700



4. Tutorials

The tutorials in this chapter are arranged in order of complexity. If you are not familiar with the procedures involved in selecting and programming a device, you may wish to begin with the Quick Start tutorial.

I. Quick Start: Programming Memory Devices

This tutorial describes how to select, read, and program a memory device (an Intel 27C256 DIP 28-pin EPROM) using the default programming settings and using a master device that is the same type as the target device. For information on other operations, refer to the online Help, Chapter 5, "Commands," and the other tutorials in this chapter.

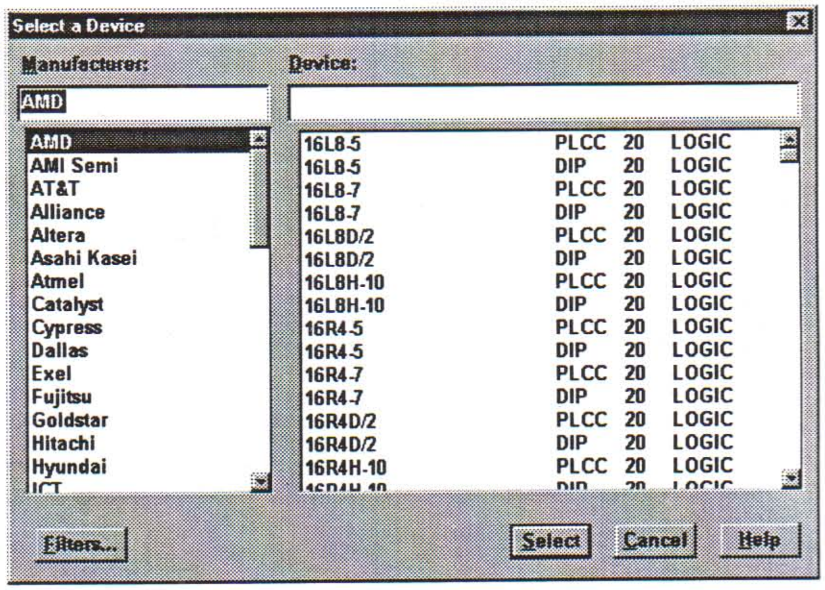
1. Connect Hardware and Install Software

Connect the hardware, install the software, and start the 2700 as described in Chapter 2.

Install the DIP base if it is not already installed (see page 15).

2. Select Device

1. From the Device menu, choose the Select... command or click on the Select Device button on the tool bar. The following dialog box appears.



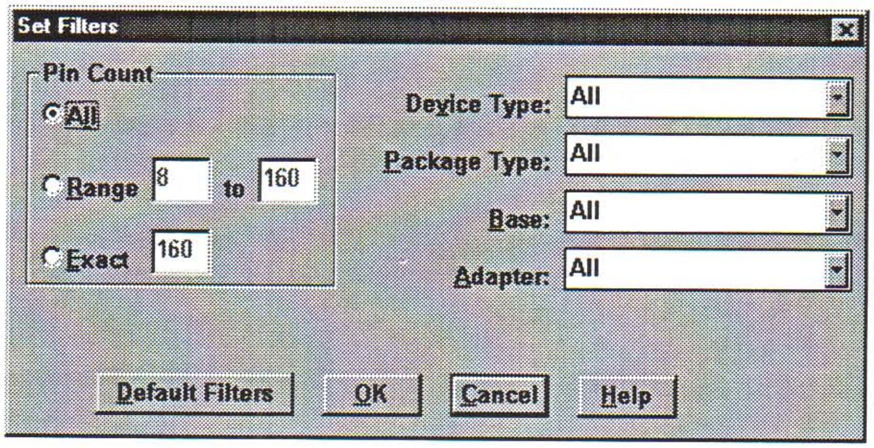
Manufacturer:
AMD
AMI Semi
AT&T
Alliance
Altera
Asahi Kasei
Atmel
Catalyst
Cypress
Dallas
Exel
Fujitsu
Goldstar
Hitachi
Hyundai
ICT

Device:			
16L8-5	PLCC	20	LOGIC
16L8-5	DIP	20	LOGIC
16L8-7	PLCC	20	LOGIC
16L8-7	DIP	20	LOGIC
16L8D/2	PLCC	20	LOGIC
16L8D/2	DIP	20	LOGIC
16L8H-10	PLCC	20	LOGIC
16L8H-10	DIP	20	LOGIC
16R4-5	PLCC	20	LOGIC
16R4-5	DIP	20	LOGIC
16R4-7	PLCC	20	LOGIC
16R4-7	DIP	20	LOGIC
16R4D/2	PLCC	20	LOGIC
16R4D/2	DIP	20	LOGIC
16R4H-10	PLCC	20	LOGIC

2. In the manufacturer selection box, click the Intel selection, or type **Intel** in the entry field above the manufacturer list.
3. In the device selection box, scroll (if necessary) and select the 27C256 DIP 28-pin EPROM.
4. Click the **Select** button. The main screen is displayed with the selected device displayed at the top.

TIP: Using the Selection Filter

To speed up device selection, you can limit the number of devices that appear in the list by selecting **Filters...** from the **Select a Device** dialog box. The following dialog box appears.



Range: 8 to 160
Exact: 160
Device Type: All
Package Type: All
Base: All
Adapter: All

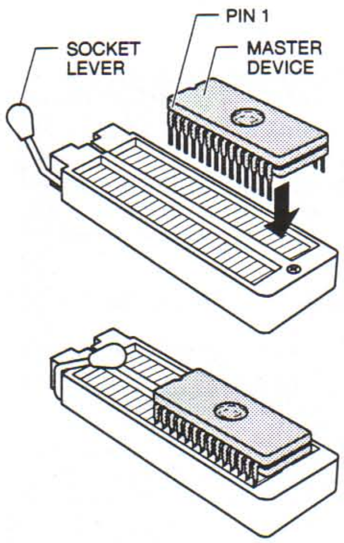
To set the filters, specify pin count, device type, package type, base, and adapter for the devices from which you want to choose.

For the device used in this tutorial, you would fill in the spaces as follows:

Pin Count	Exact: 28
Device Type	EPROM
Package Type	DIP
Base	DIP 48-1
Adapter	All (Default)

Setting the filters in this manner narrows the list that appears in the Select a Device dialog box to those devices that are 28-pin EPROMs in a DIP package.

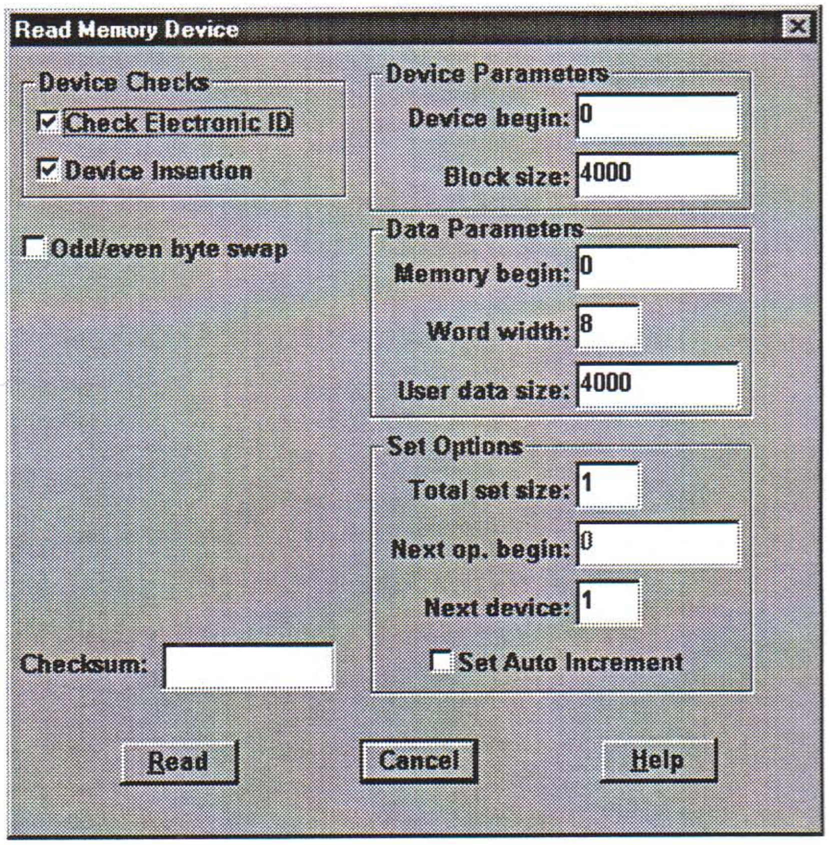
3. Read Data



In this procedure, the 2700 will read data from a master device into memory.

1. Insert the master device in the socket. Make sure that pin 1 is toward the lever end of the socket and that the device is bottom justified, as shown in the illustration.
2. Lock the socket lever (see page 17).
3. Select **Read**... from the Device menu or click the Read button on the tool bar.

   The following dialog box appears.



Read Memory Device

Device Checks
- [✓] Check Electronic ID
- [✓] Device Insertion
- [ ] Odd/even byte swap

Checksum:

Device Parameters
- Device begin: 0
- Block size: 4000

Data Parameters
- Memory begin: 0
- Word width: 8
- User data size: 4000

Set Options
- Total set size: 1
- Next op. begin: 0
- Next device: 1
- [ ] Set Auto Increment

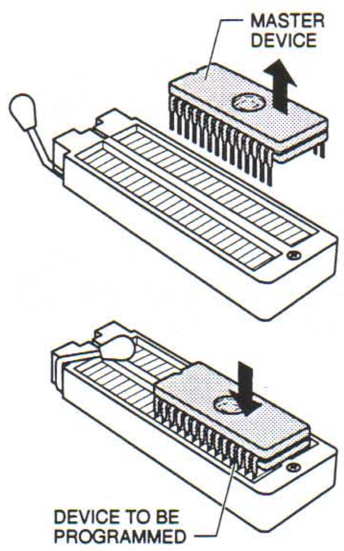
   You will use the default values that appear in the Read Memory device fields. For information on the options in this dialog box, press the F1 key.

4. Click the **Read** button.

   As the device is being read, a status box indicates that the operation is being performed and shows the percentage of the operation that is complete.

When the operation is complete, the status box disappears and the checksum is displayed in the Read Memory Device screen and the System Log.

4. Program Device



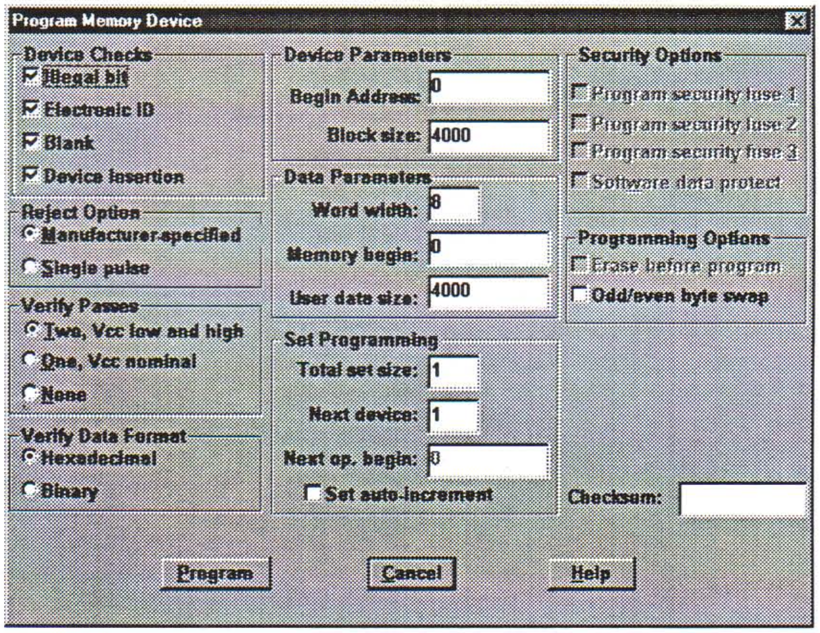
In this procedure, the 2700 will program a blank device with the data in memory.

1. Remove the master device, and insert a blank Intel 27C256 device in the socket.

Note: *If the device to be programmed is not blank, an error will occur. The device must be erased before it can be programmed (refer to the page 53 for instructions on erasing a device).*

2. Select **Program...** from the Device menu or click the Program button on the tool bar.

   The following dialog box appears.


Program Memory Device

Device Checks
- ☑ Illegal bit
- ☑ Electronic ID
- ☑ Blank
- ☑ Device insertion

Reject Option
- ● Manufacturer-specified
- ○ Single pulse

Verify Passes
- ● Two, Vcc low and high
- ○ One, Vcc nominal
- ○ None

Verify Data Format
- ● Hexadecimal
- ○ Binary

Device Parameters
- Begin Address: 0
- Block size: 4000

Data Parameters
- Word width: 8
- Memory begin: 0
- User data size: 4000

Set Programming
- Total set size: 1
- Next device: 1
- Next op. begin: 0
- ☐ Set auto-increment

Security Options
- ☐ Program security fuse 1
- ☐ Program security fuse 2
- ☐ Program security fuse 3
- ☐ Software data protect

Programming Options
- ☐ Erase before program
- ☐ Odd/even byte swap

Checksum:

   For this example, you will use the default values that appear in the Program Memory Device fields. For information on the options in this dialog box, press the F1 key.

3. Click the **Program** button.

   As the device is being programmed and verified, a status box indicates what operation is being performed and what percentage of that operation is complete.

When the operation is complete, the status box disappears and the checksum is displayed in the System Log.

You have completed this Quick Start tutorial. You are now ready to use the 2700 to read and program memory devices using the default settings.

II. Samples of Process Flow in Programming

Use the following process flows for quick reference. For a detailed explanation on selecting, opening, and programming a device, refer to the Quick Start tutorial on page 24.

Program from Master Device (same as target device)

To program a device with information from a master device that is the **same** as the device you wish to program, follow these steps (also refer to the Quick Start tutorial on page 24):

1. Select the master device.
2. Read the device.
3. Program the device.

Program from Master Device (different from target device)

To program a device with information from a master device that is **different** from the device you wish to program, follow these steps:

1. Select the master device.
2. Read the device.

   Reading the master creates a file from which the 2700 programs the designated device.
3. Make note of the checksum displayed in the System Log.
4. Select the blank device.
5. Program the device.
6. Make sure the checksum displayed in the System Log matches the checksum of the master device.

Program from Data File

To program a device with information from a data file, follow these steps:

1. Select a device.
2. Open the data file (see page 29 for detailed instructions).
3. Edit the data file, if you wish.
4. Program the device.

III. Opening Files

This tutorial focuses on more complex tasks. You will be given step-by-step directions on:

- Opening a data file using default parameters.
- Opening a data file using specific parameters.

A. Opening a Data File using Default Parameters

You must open a data file before you can work with the data in the file and before you can program the data into a device. Follow these steps to open a data file using the default parameters of the 2700:

1. Select the device you are going to work with. You must select the device before you open the data file.
2. Open the data file using one of the following methods:
   - Click the **File Open** button in the tool bar.
   - Choose **Open...** from the File menu (ALT, **F**, **O**).
3. From the **File Open** dialog box, select the file you want to open by specifying the following items:

Item	Description
File Name	The name of the file to be opened.
Directories	The directory path of the file.
Drives	The drive on which the file is stored.
List Files of Type	Allows you to specify the type of data files to display in the File Name box. Supported data file formats include: *.jed (JEDEC files) *.bin (binary files) *.hex (hex files) *.pof (POF files) *.lof (LOF files)

4. If your PC is connected to a supported network, you can click the **Network** button and select a file from a network location.
5. Click **OK** to open the data file.

The file is stored in memory, can be edited, and can be used to program a selected device.

TIPS:

- Use the **List Files of Type** box to limit the files listed in the File Name box. Or, type an asterisk plus the file extension (such as *.bin, *.hex, etc.) in the File Name box. Note that the asterisk (*) is a wildcard character that stands for any filename.
- If the file you are looking for is not shown, make sure that the correct drive and directory are selected.

B. Opening a File with Specific Parameters

You can use the File Open Parameters dialog box to set several parameters before you open the data file to control how the file is read from disk into memory.

To select parameters, follow these steps:

1. Perform steps 1 through 4 in the "Opening a Data File Using Default Parameters" section.
2. In the File Open Parameters box, select **Parameters.**
3. Set the File Open parameters.

   You can set the following in the File Open Parameters dialog box:

   - Data Format
   - Fill Options
   - Read Options
   - Translate DIP to JLCC Vectors
4. Click **OK** to accept the parameters.

These options are described in the following sections. Once you have set the File Open Parameters, click **OK** to return to the File Open dialog box.

Data Format

Data formats (also known as translation formats) are patterns for encoding data in a data file. A data file contains the information to be programmed into a device (in a specific format). Before you can open a data file, you must indicate which format was used to store the file, or use the default, which will automatically detect the format of your file.

The following data formats are supported:

Format	Code
Automatic Format Detect	*
Formatted Binary	10
POF	14
Absolute Binary	16
LOF	17
ASCII-Hex SP STX	50
ASCII-Hex "%" STX	51
ASCII-Hex "'" STX	52
ASCII-Hex "," STX	53
ASCII-Hex SP SOH	55
ASCII-Hex "%" SOH	56
ASCII-Hex SMS	57
ASCII-Hex "," SOH	58
Motorola Exorcisor	82
Intel Intellec 8/MDS	83
Motorola Exormax	87
Intel MCS-86 Hex Object	88
JEDEC (FULL)	91
JEDEC (KERNEL)	92
Motorola 32-bit	95
Intel Hex-32	99

* *When Automatic Format Detect is selected, the programmer software detects which format the data file is in.*

If you know the format of the data file you want to open, select that format from the **Data Format** box.

If you do not know the format of the data file you want to open, select Automatic Detect in the Data Format box. When this option is selected, the programmer will examine the data file to determine the format.

Note: *Automatic Detect cannot detect Absolute Binary format or Formatted Binary format. If you are using a data file stored in one of these formats, you must select the data format from the Data Format list box.*

For more information about specific data formats, see Appendix B.

Fill Options

Fill Options determine how your programmer handles unused locations in the data file when the data file is read into memory. For instance, you can have the programmer fill the unused locations in memory with a data pattern.

*Note: If you enable the Fill Option, the data file on the disk is **not** modified. The fill affects only how the data are stored in memory.*

Select one of the following Fill Options:

Fill Option	Explanation
None	Disables the filling of any unused locations when a data file is read. The data stored in the unused locations maintain their current state.
Default Value	Fills unused locations in a data file with a value specified by the programming algorithm for the selected device. The default value is displayed to the right of the Default radio button. *Note: The Default value is read-only and cannot be changed.*
Specific Value	Fills unused locations in a data file with the value specified in the box to the right of the Specific Value radio button. If you select this option, you must specify a value. Make sure you specify a two-digit hexadecimal number, such as 04, EF, or C3.

Examples

- You have a data file for a memory device, and the data file contains 100h bytes of data starting at address A00h. The memory locations from 000h to 9FFh are "empty." You can use the Fill Option to fill the "empty" locations with a data pattern.
- You have a data file for a memory device, and the data file contains 100h bytes of data starting at address 000h. The memory device you are going to program can hold 1000h bytes of data. The locations from 100h to FFFh are "empty." You can use the Fill Option to fill that "unused" range with a data pattern.

Read Options

The Read Options parameter enables you to specify the data size (how much data to read), the beginning address, and the address offset that will be used when the data file is read.

The following parameters are available in the Read Options group:

Parameter	Description
Begin Address	Specifies where in user memory to start storing data when opening a data file or reading data from a device. When saving a data file or programming a device, the memory begin address specifies at which address in user memory to start saving or programming data.
User Data Size	Specifies how much data to read from the data file.
Address Offset	The address offset (also called I/O Offset) specifies how much of an offset is used during a data transfer.

Address Offset Examples

The address offset is added to all addresses sent out from the programmer (data uploaded), and is subtracted from all addresses received by the programmer (data downloaded). For example, if you set the address offset to 001000, the data downloaded to the programmer would be placed at the address specified by the data record minus 001000 in user RAM.

If you set address offset to 000000, all data is stored at the address specified by the data (since 000000 or zero is subtracted from the address of the incoming data record).

If you set address offset to the default (FFFFFF), the programmer sets the address offset to the first incoming address of data received from download operations (such as opening a data file). For example, if address offset is set to the default of FFFFFF and the first data record is downloaded with a specified address of 001000, the address offset is set to 001000.

TIPS: Address Offset and Begin Address

- To read only a portion of a data file, set the address offset to where you want to start reading and set the data size to the amount of memory you want to read. For instance, loading a data file with data size of 001000 and address offset of 2000 would load 1000h bytes of data starting at address 002000 in the file (002000 to 002FFF).
- To store the data file in a different location in memory than where it was originally saved, set the begin address to the memory address where you want to store the data and set address offset to the default (FFFFFF).
- To read the data into memory at the same location it was when saved, set the begin address to the default value (FFFFFF) and address offset to 000000.

TIPS: Odd/Even Byte Swap

- If the byte order of the file you opened appears incorrect, you can specify that the programmer perform an odd/even byte swap. Refer to the information on byte swapping in Chapter 8.

Translate DIP to PLCC/JLCC Vectors

Use this feature if you have created test vectors for a DIP device but want to program the PLCC/JLCC version of the same device.

When this feature is selected, the programmer alters the test vectors during I/O translation (when the data file is loaded), allowing for the different pinouts of the two package types (DIP and PLCC/JLCC). During the reading of the data file, vectors are converted from DIP to PLCC/JLCC; during saving, you can have vectors converted from PLCC/JLCC to DIP.

5. Commands

This chapter describes the commands you can access from the 2700's menus.

File Menu

New

Creates a new data file and opens a new window for the file. Use the Save As command to save the file under a new name.

The newly created data file is as large as your user data size.

Open

Use the Open command to open an existing data file that is stored on disk. You must open a data file before you can work with the data in the file or before you can program the data into a device.

To open a data file

1. Select the device you are going to work with. You must select the device before you open the data file.
2. Open the data file by one of the following methods:
   - Click the **File Open** button on the toolbar.
   - Choose **Open...** from the File menu (ALT, **F**, **O**).

   You can set several parameters before you open the data file. The parameters affect how the data is read into memory.
3. To select parameters click the **Parameters** button. After you have set the File Open parameters, click **OK** to accept the parameters.
4. From the File Open dialog box, select the file to open by specifying the following items:

Item	Explanation
Filename	The name of the file to be opened.
Directories	The directory path of the file.
Drives	The drive on which the file is stored.

Item	Explanation
List Files of Type	The type of data files to display in the Filename box. Supported data file types include the following: *.jed (JEDEC files) *.bin (binary files) *.hex (hex files) *.pof (POF files) *.lof (LOF files)

5. If your PC is connected to a supported network, you can click the **Network** button and select a file from a network location.
6. Click **OK** to open the data file.

   The file is stored in memory, can be edited, and can be used to program a selected device.

TIP: Limiting the Listed Files

Use the List Files of Type box to limit the files listed in the Filename box. Or type an asterisk and the file extension (.bin, .hex, etc.). The asterisk (*) is a character than stands for any filename.

If the file you are looking for is not shown, make sure that the drive and directory are correct.

Parameters

Use the File Open dialog box to control how a data file is read from disk.

You can change the load options for a data file, such as loading the data into a different range of memory than it was originally saved to, or you can load only a part of the data file.

You can set the following options in the File Open Parameters dialog box:

- Data Format
- Fill Options
- Read Options
- DIP to PLCC Vector Translation

One you have set the File Open parameters, click **OK** to return to the File Open dialog box.

Data Format

Data files are stored in a number of different data formats. Before a data file is read from disk, you must indicate what format was used to store the file you are about to open.

Formats Available

If you know the format of the data file you want to open, select that format from the **Data Format** box.

Auto Format Detect

If you do not know the format of the data file you want to open, select Automatic Format Detect in the Data Format box. When this parameter is selected, the data file will be examined and the data format will be determined automatically.

Note: Auto Detect cannot detect Absolute binary format or Formatted Binary format. If you are using a data file stored in one of these formats, you must select the data format from the Data Format list box.

Data Formats

Data formats (also known as Translation Formats) are patterns for encoding data in a data file. A data file contains the information to be programmed into a device. The following data formats are supported:

Format	Code
Automatic Format Detect	*
Formatted Binary	10
POF	14
Absolute Binary	16
LOF	17
ASCII-Hex SP STX	50
ASCII-Hex "%" STX	51
ASCII-Hex "" STX	52
ASCII-Hex "," STX	53
ASCII-Hex SP SOH	55
ASCII-Hex "%" SOH	56
ASCII-Hex SMS	57
ASCII-Hex "," SOH	58
Motorola Exorcisor	82
Intel Intellec 8/MDS	83
Motorola Exormax	87
Intel MCS-86 Hex Object	88
JEDEC (Full)	91
JEDEC (Kernel)	92
Motorola 32-bit	95

* *When Automatic Format Detect is selected, the programmer software detects the format of the data file.*

Fill Options

File Options determine how your programmer handles unused locations in the data file when the data file is read into memory. For instance, you can have the programmer fill the unused locations in memory with a data pattern.

*Note: If you enable the Fill Options, the data file on disk is **not** modified. The fill only affects how the data is stored in memory.*

Select one of the following Fill Options:

Fill Options	Explanation
None	Disables the filling of any unused locations when a data file is read. The data stored in the unused locations maintains its current state.
Default Value	Fills unused locations in a data file with a value specified by the programming algorithm for the selected device. The default value is displayed to the right of the Default radio button. *Note: The Default value is read-only and cannot be changed.*
Specific Value	Fills unused locations in a data file with the value specified in the box to the right of the Specific Value radio button.

Examples

- You have a data file for a memory device, and the data file contains 100h bytes of data starting at address A00h. The memory locations from 000h to 9FFh are "empty." You can use the Fill Option to fill the "empty" locations with a data pattern.
- You have a data file for a memory device, and the data file contains 100h bytes of data starting at address 000h. The memory device you are going to program can hold 1000h bytes of data. The locations from 100h to FFFh are "empty." You can use the Fill Option to fill that "unused" range with a data pattern.

Read Options

You can specify the data size (how much data to read), the beginning address, and the address offset that will be used when the data file is read.

The following parameters are available in the Read Options group:

Parameter	Description
Data Size	Specifies how much data to read from the data file.
Begin Address	Specifies the first address in which to store the data read from disk.
Address Offset	Specifies how much of an offset is used during a data transfer.

TIPS:

- To read only a portion of a data file, set the address offset to where you want to start reading, and set the data size to the amount of memory you want to read. For instance, loading a data file with data size of 001000 and address offset of 2000 would load the data specified in the file from address 002000 (because the offset of 2000 is subtracted from all data records read from the file) to 002FFF (because the data size of 1000 is loaded).
- To store the data file in a different locations in memory than the file was originally saved in, set the begin address to the address where you want to store the data in memory, and set address offset to 00000000.
- To read the data into memory at the same location that it was when saved, set the begin address to the default value (000000) and the address offset to the default value (FFFFFF).

Translate DIP to PLCC Vectors

Translates test vectors for a device from its DIP package to its PLCC/JLCC package.

If this feature is selected, the programmer alters the test vectors during I/O translation (when the data file is loaded), allowing for the different pinouts of the two package types (DIP and PLCC). During this reading of the data file, vectors are converted from DIP to PLCC; during saving, you can have vectors converted form PLCC to DIP.

Use this feature if you have created test vectors for a DIP device but actually want to program the PLCC version of the same device.

Save

Saves the data file loaded in memory to disk, overwriting the older version. Use the **Save As** command to save a file with a different name or save options.

Save As

Select the Save As command to save the current data file under a new name. You can also save a current data file in a new data format, and you can save a portion of a currently-opened data file to a new file.

To save a data file

1. Do one of the following:
    - If the file is new (title bar shows "untitled"), click the **Save File** button in the toolbar. The Save As dialog box appears.
    - Choose **Save As...** from the File menu (**Alt**, **F**, **A**).
2. You can select several parameters before you save the data file. The parameters affect how the file will be written to disk. To set parameters, click the **Parameters** button.

    After you have set the parameters, choose **OK** to return to the Save As dialog box.
3. Select where you want to save the file by specifying the following items:

Item	Explanation
Filename	The filename to use when saving the file.
Directories	The directory path to the file.
Drives	The drive on which the file is stored.
List Files of Type	Allows you to specify the type of data files to display in the File Name box. Supported data file types include the following: *.jed (JEDEC files) *.bin (binary files) *.hex (hex files) *.pof (POF files) *.LOF (LOF files)

4. If your PC is connected to a supported network, you can click the **Network** button and save the file at a network location.
5. Click **OK** to save the file. If you do not want to save the file, choose **Cancel**.

TIP:

To copy over (overwrite) a file, select the drive, directory, and file type of the file (you can use wildcards to limit the choices). When the file list appears, click the name of the file (the file name will appear in the File Name field). Choose **Save**.

When you attempt to overwrite a file, you are prompted to confirm the file replacement. Choose **OK** to overwrite the file or **Cancel** to exit the dialog box without saving the file.

Parameters

Before a data file is written to disk, you can set various parameters that affect how the data file will be written to disk.

You can set the following options in the File Save Parameters dialog box:

- Data Format
- Save Options
  - Begin Address
  - User Data Size
  - Address Offset
- Translate PLCC to DIP

Note: Parameters that don't apply to the selected device are not available.

Once you have set the File Save parameters, click the **OK** button to return to the File Save As dialog box.

Data Format

Data files are stored in a number of different data formats. Before a data file is saved to disk, you can indicate what format to store the file in (if you do not specify, the file is saved in the same format it was when opened).

Formats Available

If you know the format of the data file you want to save, select that format from the **Data Format** box.

Translate PLCC/JLCC to DIP Vectors

Translates test vectors for a device from its DIP package to its PLCC/JLCC package.

If this feature is selected, the programmer alters the test vectors during I/O translation (when the data file is saved), allowing for the different pinouts of the two package types (DIP and PLCC/JLCC). During the saving of a data file, vectors are converted from PLCC/JLCC to DIP.

Edit Menu

Jump To...

Use the **Jump To...** command from the Edit menu to move to a specific memory location in memory.

Display Format...

You can choose to display the data in different ways by changing the options in the Display Format dialog box.

Word Size	Selects the word size that the data is displayed as.
Byte Order	Selects which byte in each word is the Most Significant Byte (MSB).

Fill Memory...

Fill allows you to fill locations in memory with a data pattern (overwriting any data that might already exist at the memory addresses).

To fill an area of memory with a data pattern

1. Select **Fill Memory** from the Edit menu.
2. Select the starting address for the fill in the **Starting** field.
3. Select the size of the block of memory to be filled in the **Block Size** field.
4. Select the data pattern to fill the specified memory block in the **Fill With** field. Make sure you specify a two-digit hexadecimal number, such as 04, EF, or C3.
5. Click **Fill** or press ENTER.

Address Range...

This command determines the range of memory that can be edited.

User Data (default)	Only the block of memory that starts at the Memory Begin Address and ends at memory begin address + User Data Size can be edited.
Full	All User Memory can be edited.
User Defined	You can specify the range in memory that can be edited.

Copy To Clipboard

If you are editing user memory, this command copies the data in the current page (256 bytes) to the Windows Clipboard.

If you are viewing the system Log file, this command copies the contents of the log file to the Windows Clipboard.

Device Menu

Select...

Use the Select command from the Device menu to select the device you want to program.

You must select a device before you can work with a data file or work with a device.

To select a device

1. Click the **Select Device** button in the toolbar or choose **Select...** from the Device menu (**ALT**, **D**, **S**).
2. If you want, you can set filters to display only the types of devices that are similar to the device you will be working with.

   To set device filters, click the Filters button in the Select a Device dialog box.
3. Select the manufacturer of the device you intend to select. To select a manufacturer, choose one of the manufacturers listed in the Manufacturer box.

   You may have to scroll through the manufacturers listed in the Manufacturer box until you find the manufacturer you are looking for.
4. Select the specific device you will be working with. To select a specific device, choose one of the devices listed in the Device box.

   You may have to scroll through the devices listed in the Device box until you find the device you are looking for.

Read...

The Read Device command reads data stored in the socketed device into memory. You must read data from a device before you edit or duplicate the data.

Note: The Read Device command will not work on a device whose security fuse has been programmed.

Read Logic Device

The Read Logic Device command reads data stored in the socketed device into memory.

You must read data from a device before you edit or duplicate the data.

Note: The Read Logic Device command will not work on a device whose security fuse has been programmed.

To read data from a logic device

1. Select the device you want to read.
2. Socket the device you want to read.
3. Select **Read...** from the Device menu (**A**LT, **D**, **R**). The Read Logic Device dialog box opens.
4. Check the settings of the parameters.
5. Click **Read** to read the data in the currently socketed device.

When the data in the socketed device has been read, the results of the read operation are noted in the System Log. If the data was not successfully read from the device, the operation is halted and a message is displayed.

After a successful Read Device operation, the data is in memory and can be edited or duplicated.

To read data from another logic device

- To read data from another logic device, remove the device currently in the socket, insert another device of the same type, and click **Read**. If you insert a device that is different from the device that is currently selected, you must select the device before you read it.

When you are finished, click **Close** to close the Read Logic Device dialog box.

Read Memory Device

Use the Read Memory Device command to read data stored in the socketed device into memory.

You must read data from a device before you edit or duplicate the data.

Note: The Read Memory Device command will not work on a device whose security fuse has been programmed.

To read data from a memory device

1. Select the device you want to read.
2. Socket the device you want to read.
3. Select **Read...** from the Device menu (**A**LT, **D**, **R**). The Read Memory Device dialog box opens.
4. Check the settings of the parameters.
5. Click **Read** to read the data in the currently socketed device.

When the data in the socketed device has been read, the results of the read operation are noted in the System Log. If the data was not successfully read from the device, the operation is halted and a message is displayed.

After a successful Read Device operation, the data is in memory and can be edited or duplicated.

To read data from another memory device

- To read data from another memory device, remove the device currently in the socket, insert another device of the same type, and click **Read**. If you insert a device that is different from the device that is currently selected, you must select the device before you read it.

When you are finished, click **Close** to close the Read Memory Device dialog box.

Program...

Use the Program Device command to program the current data file (data in user memory) into the socketed device.

When you select this command, you can choose how to program a device by changing the programming parameters. For instance, to swap the high and low bytes before programming the data into a device, change the Odd/even byte swap parameter.

Program Logic Device

The Program Logic Device command programs the current data file into the socketed device.

To program a logic device

1. Select the device you want to program.
2. Socket the device you want to program.
3. Select **Program...** from the Device menu. The Program Logic Device dialog box appears.

Note: The Program command is not available if you have not opened a data file.

4. Check the settings of the parameters.
5. Click **Program** to program the current data file into the socketed device.

When the device has been programmed, the result of the programming operation is noted in the System Log. If any errors were detected during programming, or during the optional post-programming verify, the operation is halted, a message is displayed, and the errors are noted in the System Log.

To program another logic device

- To program another logic device, remove the device currently in the socket, insert another device of the same type, and click **Program**. If you insert a device that is different from the device that is currently selected, you must select the device before you program it.

When you are finished, click **Close** to close the Program Logic Device dialog box.

Program Memory Device

The Program Memory Device command programs the current data file into the socketed device.

To program a memory device

1. Select the device you want to program.
2. Socket the device you want to program.
3. Select **Program...** from the Device menu. The Program Memory Device dialog box appears.

Note: The Program command is not available if you have not opened a data file.

4. Check the settings of the parameters.
5. Click **Program** to program the current data file into the socketed device.

When the device has been programmed, the result of the programming operation is noted in the System Log. If any errors were detected during programming, or during the optional post-programming verify, the operation is halted, a message is displayed, and the errors are noted in the System Log.

To program another memory device

- To program another memory device, remove the device currently in the socket, insert another device of the same type, and click **Program**. If you insert a device that is different from the device that is currently selected, you must select the device before you program it.

When you are finished, click **Close** to close the Program Memory Device dialog box.

Verify...

Choose the Verify command from the Device menu to display the Verify Device dialog box, which allows you to verify the current data file against the data in the socketed device.

Verify Logic Device

The Verify Logic Device command verifies the current data file against the data in the socketed device.

To verify a logic device

1. Select the device you want to verify.
2. Socket the device you want to verify.
3. Select **Verify...** from the Device menu. The Verify Device dialog box appears.
4. Check the settings of the parameters.
5. Click **Verify** to verify the currently socketed device.

When the device has been verified, the result of the verify is noted in the System Log. If any differences were found between the data in the current data file and the data in the device, the verify operation is halted immediately. A message is displayed indicating the location of the verify error and the data in the device and in the data file. This information is also noted in the System Log.

To verify another logic device

- To verify another logic device, remove the device currently in the socket, insert another device of the same type, and click **Verify**. If you insert a device that is different from the device that is currently selected, you must select the device before you verify it.

When you are finished, click **Close** to close the Verify Logic Device dialog box.

Verify Memory Device

The Verify Memory Device command verifies the current data file against the data in the socketed device.

To verify a memory device

1. Select the device you want to verify.
2. Socket the device you want to verify.
3. Select **Verify...** from the Device menu. The Verify Device dialog box appears.

4. Check the settings of the parameters.
5. Click **Verify** to verify the currently socketed device.

When the device has been verified, the result of the verify is noted in the System Log. If any differences were found between the data in the current data file and the data in the device, the verify operation is halted immediately. A message is displayed indicating the location of the verify error and the data in the device and in the data file. This information is also noted in the System Log.

To verify another memory device

- To verify another memory device, remove the device currently in the socket, insert another device of the same type, and click **Verify**. If you insert a device that is different from the device that is currently selected, you must select the device before you verify it.

When you are finished, click **Close** to close the Verify Memory Device dialog box.

Blank Check...

The Blank Check command tests the socketed device for programmed bits. If programmed bits are found in the socketed device, the test is halted and a message is displayed. Before a device can be programmed, it must be blank.

To blank check a device

1. Select the device you want to blank check.
2. Socket the device you want to blank check.
3. Select **Blank check...** from the Device menu (**ALT**, **D**, **B**). The Blank Check Device dialog box appears.

Note: *Not all devices support Blank Check. If the test is not supported, the Blank Check command is not available (it appears grayed out on the menu).*

4. Make sure the settings for the following check box items are correct:
   - Electronic ID
   - Insertion Check
5. Click **Check**.

When the device has been checked, the result of the blank check is noted in the System Log. If any non-blank locations were found in the socketed device, the operation is halted and a message is displayed.

To check another device

- To check another device, remove the device currently in the socket, insert another device of the same type, and click **Check**. If you insert a device that is different from the currently selected device, you must select the device before you check it.

When you are finished, click **Cancel** to close the Blank Check Device dialog box.

Illegal Bit Check...

The Illegal Bit Check command tests the socketed device for already-programmed locations that would have to be reversed to match the current data file.

The Illegal Bit Check test compares the data in the socketed device against the data in the data file. An illegal-bit error occurs when data in the selected data file indicates that a specific bit should be in an unprogrammed state, while the corresponding bit in the socketed device is in a programmed state.

Illegal Bit Check (Memory Device)

The Illegal Bit Check command tests the socketed device for already-programmed locations that would need to be reversed to match the current data file.

To check a memory device for illegal bits

1. Select the device you want to check for illegal bits.
2. Socket the device you want to check for illegal bits.
3. Select **Illegal bit check...** from the Device menu (**ALT**, **D**, **I**). The Illegal Bit Check memory Device dialog box appears.
4. Check the settings of the parameters.
5. Click **Check** to check the currently socketed device for illegal bits.

When the device has been checked, the results of the illegal bit check are noted in the System Log. If illegal bits were found, a message is displayed.

To check another device

- To check another device, remove the device currently in the socket, insert another device of the same type, and click **Check**. If you insert a device that is different from the currently selected device, you must select the device before you check it.

When you are finished, click **Cancel** to close the Blank Check Device dialog box.

Illegal Bit Check (Logic Device)

The Illegal Bit Check command tests the socketed device for already-programmed locations that would need to be reversed to match the current data file.

To check a logic device for illegal bits

1. Select the device you want to check for illegal bits.
2. Socket the device you want to check for illegal bits.
3. Select **Illegal bit check...** from the Device menu (**ALT**, **D**, **I**). The Illegal Bit Check Logic Device dialog box opens.
4. Check the settings of the parameters.
5. Click **Check** to check the currently socketed device for illegal bits.

When the device has been checked, the results are noted in the System Log. If illegal bits were found, a message is displayed.

To check another device

- To check another device, remove the device currently in the socket, insert another device of the same type, and click **Check**. If you insert a device that is different from the currently-selected device, you must select the device before you check it.

When you are finished, click **Cancel** to close the Illegal Bit Check dialog box.

Compare Electronic ID (Memory Device)

Compares the electronic ID of the device in the socket with the ID of the selected device.

If the electronic ID of the socketed device does not match the ID for the selected device, an error message is generated. You can select a new device to match the socketed device, or remove the socketed device and insert a device that is the same type as the selected device.

If the IDs match, no message is generated and you can proceed with device operations.

Erase...

The Erase Device command erases an electronically erasable device, such as an EEPROM, and EEPAL, or an EEMICRO.

Note: *Once you erase a device, the information contained in the device is permanently erased and cannot be recovered.*

To erase a device

1. Select the device you want to erase.
2. Socket the device you want to erase.
3. Select **Erase...** from the Device menu. The Erase Device dialog box appears.

Note: *The Erase Device command is not available if the device you have selected is not electronically erasable.*

4. Check the settings of the parameters.
5. Click **Erase** to erase the currently socketed device.

When you are finished, click **Cancel** to close the Erase Device dialog box.

View Footnotes

This command displays information about the currently-selected device.

Note: *Not all devices have footnotes. If the currently-selected device does not have any footnotes, the Footnotes button in the toolbar and the Footnotes option in the Device menu are disabled (they appear grayed out).*

Footnotes tell you about operations that are specific to a particular device, such as if the security fuse option is available, and if an adapter or PPI is needed to program the device.

To view the footnotes for the current device

- Click the **Footnotes** button in the toolbar.
- Select **View Footnotes** from the Device menu (**ALT, D, S**).

To close the footnotes window

- If the footnotes window is open and is minimized (in an icon form), do one of the following:
  - Click the **Footnotes** button in the toolbar.
  - Click the Footnotes icon, then choose **Close** from the menu in the popup window.
  - Select **View Footnotes** from the Device menu (**ALT**, **D**, **S**), then reselect **View Footnotes** from the Device menu.
- If the footnotes window is open and is displayed as a window, do one of the following:
  - Click the **Footnotes** button in the toolbar.
  - Select **View Footnotes** from the Device menu (**ALT**, **D**, **S**).

Data Menu

Sumcheck...

Select the **Sumcheck** command from the Data menu to display the Sumcheck Device dialog box, which allows you to compute a checksum for the data in memory.

Sumcheck Logic Device

The Sumcheck Logic Device command computes a sumcheck for the data in memory.

To sumcheck a logic device

1. Select the device you want to sumcheck.
2. Select **Sumcheck...** from the Data menu. The Sumcheck Logic Device dialog box appears.

Note: The Sumcheck command is available only if you have selected a device.

3. Choose **Sumcheck** to sumcheck the currently socketed device.

When the sumcheck has been computed, it is displayed in the Sumcheck Logic Device dialog box and noted in the System Log.

To sumcheck another logic device

- To sumcheck another device, select another device and choose **Sumcheck**.

When you are finished, click **Cancel** to close the Sumcheck Logic Device dialog box.

More About Sumchecking Logic Data

A different algorithm is used to compute the sumcheck for logic and memory devices.

About the sumcheck algorithm

For the logic device, the algorithm takes all the data in a specified range and steps through the following algorithm:

1. Each group of 8 fuses is encoded into 8 bits, with 1s representing programmed fuses and 0s representing intact fuses.
2. The high and low bits in the 8-fuse chunk are swapped.
3. The 8 bits are encoded into a 2-digit hexadecimal number.
4. The resulting 2-digit hexadecimal number is added to the 4-digit sumcheck. If any carry is present, it is dropped.

Example:

Step 1: Consider the following JEDEC file.

Note: In this case, the fuse state is already encoded into 1s and 0s.

```
QP20* QF128*
L0000
1011111111111111
1110111111111111
1111101111111111
1111111011111111
1111111110111111
1111111111101111
1111111111111011
1111111111111110*
C0E9C*
```

Step 2: Swap high and low bits.

Note: Each group of 8 bits is swapped. For example, `11001010` *becomes* `01010011` *when swapped.*

When high and low bits are swapped, the data file becomes:

Before	After
1011 1111 1111 1111	1111 1101 1111 1111
1110 1111 1111 1111	1111 0111 1111 1111
1111 1011 1111 1111	1101 1111 1111 1111
1111 1110 1111 1111	0111 1111 1111 1111
1111 1111 1011 1111	1111 1111 1111 1101
1111 1111 1110 1111	1111 1111 1111 0111
1111 1111 1111 1011	1111 1111 1101 1111
1111 1111 1111 1110*	1111 1111 0111 1111*

Step 3: Encode each group of 8-bits into a 2-digit hex number. The data file now looks like this:

Before	After
1011 1111 1111 1111	F D F F
1110 1111 1111 1111	F 7 F F
1111 1011 1111 1111	D F F F
1111 1110 1111 1111	7 F F F
1111 1111 1011 1111	F F F D
1111 1111 1110 1111	F F F 7
1111 1111 1111 1011	F F D F
1111 1111 1111 1110*	F F 7 F*

Step 4: Add each 2-digit hex number to the 4-digit sumcheck. If any carry is present, it is dropped. The data file now looks like this:

Before	After
	Initial sumcheck = 0000
F D F F	Sumcheck = 0000 + FD + FF = 01FC
F 7 F F	Sumcheck = 01FC + F7 + FF = 03F2
D F F F	Sumcheck = 05D0 + 7F + FF = 074E
7 F F F	Sumcheck = 05D0 + 7F + FF = 074E
F F F D	Sumcheck = 094A + FF + FD = 094A
F F F 7	Sumcheck = 0B40 + FF + F7 = B40
F F D F	Sumcheck = 0D1E + FF + DF = D1E
F F 7 F*	Sumcheck = 0E9C + FF + 7F = 0E9C

The sumcheck of the data in the sample JEDEC file is 0E9C.

Sumcheck Memory Device

The Sumcheck Memory Device command computes a sumcheck for the data in memory.

To sumcheck a memory device

1. Select the device you want to sumcheck.
2. Select **Sumcheck...** from the Data menu. The Sumcheck Memory Device dialog box appears.

Note: The Sumcheck command is available only if you have selected a device.

3. Choose **Sumcheck** to sumcheck the currently socketed device.

When the sumcheck has been computed, it is displayed in the Sumcheck Logic Device dialog box and noted in the System Log.

To sumcheck another logic device

- To sumcheck another device, select another device and choose **Sumcheck.**

When you are finished, click **Cancel** to close the Sumcheck Logic Device dialog box.

More About Sumchecking Memory Data

A different algorithm is used to compute the sumcheck for logic and memory devices.

About the sumcheck algorithm

For a memory device, the algorithm takes all the data in the specified range and adds it together in successive 8-bit byte chunks (regardless of word size of the selected device) to produce an 8-place hexadecimal result.

The carry on the 8-place hexadecimal result is dropped.

The 8-bit byte chunks and the 8-place result are handled in high-low fashion; the high order bit is the left-most bit and the low-order bit is the right-most bit.

Example

The following data range

```
12 63 04 EF01 23 0A BC
```

yields a sumcheck of

```
00000252
```

Preferences Menu

Start-up Self Test...

Select the **Start-up Self Test...** command from the Preferences menu to modify how the programmer conducts its startup self test.

Each time communication is established between your PC and the programmer, several tests are run on the programmer to ensure that the programmer is functioning properly.

Tests

Tests marked with an "X" are enabled to run at startup. You can toggle (enable/disable) a test by clicking the test's checkbox.

Enable All

Choose this command button to enable all the tests in the Tests area.

Clear All

Choose this command button to disable all the tests in the Tests area.

Note: The test will be run after the time limit specified by Retest After.

Retest After ___ Hours

Enter a number in this text box to specify the time interval between full self test runs. After the specified time has passed since the last full self test, a full self test is run regardless of which tests are enabled and disabled.

When a full self test is run, the date and time are recorded. After the amount of time specified in the Retest After box has passed, a full self test is repeated.

OK

Click this button to close the Start Up Self Test Preferences dialog box and **save** the settings to disk.

Cancel

Click this button to close the Start Up Self Test Preferences dialog box and **abandon** all changes made since the dialog box was opened.

Force Retesting

Use this command to specify how often to force a complete system test at power up.

Your programmer performs a performance verification as part of a complete self test, which is performed when it powers up. However, you can use the Self Test Preferences dialog box to disable some or all of the startup self tests.

Although disabling some or all of the startup self tests may allow the system to start up more quickly, over time it can cause the programmer to drift out of performance verification. This may result in poor programming yields and in marginally programmed devices.

To ensure that the performance of your programmer remains fully verified, the results of the system self test are aged. This means that, regardless of the settings of the Startup Preferences dialog box, a complete self test—and system verification—is run after a specified amount of time. The amount of time is specified in the Retest After dialog box.

Hardware...

Select this command from the Preferences menu to choose which parallel port the programmer is connected to and which parallel port driver to use for PC-to-programmer communications.

From the Hardware Preferences dialog box you can change the following settings:

- Parallel port
- Port driver

After making changes

After you have made changes in the Hardware Preferences dialog box, click the **OK** button to accept the changes, or click the **Cancel** button to exit the dialog box without making changes.

Note: *A message is displayed if the programmer software could not detect the programmer, or if the software could not establish reliable communication with the programmer.*

Parallel Port

You must specify which port (usually LPT1) on your PC is connected to the programmer.

The parallel port works with the parallel port driver to affect how the software and programmer communicate.

Selecting a parallel port

1. Select one of the following options:

Options	Result
Auto	Your programmer cycles through the parallel ports on your PC, trying to establish communication with the programmer on each port. This option should work in most cases.
LPT1:	Communicate with the programmer through parallel port 1 (LPT1).
LPT2:	Communicate with the programmer through parallel port 2 (LPT2).
LPT3:	Communicate with the programmer through parallel port 3 (LPT3).

Options	Result
Special Address	Communicate with the programmer through the specified I/O base address. This option should be used as a last resort or if you have a non-standard PC configuration. For more information on which I/O base address to use, consult the documentation that came with your PC.

After selecting a parallel port

2. If necessary, change the setting of the port driver.
3. Click the **OK** button to accept the changes.

Note: *A message is displayed if the programmer software could not detect the programmer, or if the software could not establish reliable communication with the programmer.*

Port Driver

In addition to selecting a parallel port, you must also specify which port driver to use for PC-to-programmer communication.

The port driver determines how fast data is passed between the programmer and the PC.

Which port driver to select?

Choosing a port driver depends on a number of factors, including:

- The configuration of your PC
- The length of the parallel cable connecting the PC and the programmer
- The speed of your PC

In some cases, trial and error is the only way to determine which port driver is best for your configuration.

The default port driver attempts to contact the programmer with the slowest available port driver and, once communication is established, switches to the fastest port driver for all other operations. This driver should work for most PC configurations.

Selecting a port driver

- To select a port driver, choose one of the drivers listed in the Port Driver list box.

Testing the new port driver

- Click the **OK** button to accept changes.

Note: *A message is displayed if the programmer software could not detect the programmer, or if the software could not establish reliable communication with the programmer.*

Troubleshooting

If you are having trouble establishing and maintaining communication between the PC and the programmer, try choosing a slower port driver. For example, if you are using a port driver with a rating of 1.0 and are having trouble communicating with the programmer, try a port driver with a rating of 0.75 or 0.50.

System Log...

Choose this command from the Preferences menu to specify where the log file for your programmer is written and what happens to older version of the log file.

From the System Log Preferences dialog box you can change the following settings:

- Logfile Name
- Log History
- Options

After making changes

After you have made changes to the System Log Preferences, choose **OK** to close the System Log Preferences dialog box. The changes you made are saved and will be in effect until you choose to change them again.

Setting the System Log Filename

Select the Logfile text box in the System Log Preferences dialog box to specify where the system Log file is stored on disk.

Syntax

The filename you enter in the Logfile box must be in the following form:[*]

```
[x:][\some\path\]filename.ext
```

where

`[x:]`	specifies the drive,
`[\path\]`	specifies the path, and
`filename.ext`	specifies the filename

Entering the log file filename

You can specify the filename for the System Log in one of four different ways:

Filename	Description
`<blank>`	Not specifying a filename will cause the log to be written to the default drive, path, and filename. The system default is **system.log**, saved in the same directory where your programmer software is installed.
`filename.ext`	The log will be written to the file you specify in the current directory on the current drive.
`\path\filename.ext`	The log will be written to the file you specify in the directory you specify on the current drive.
`x:filename.ext`	The log will be written to the file you specify in the current directory on the drive you specify.
`x:\path\filename.ext`	The log will be written to the file you specify in the directory you specify on the drive you specify.

* *The items shown in square brackets "[ ]" are optional. The only required element is the filename.*

About the System Log

The programmer stores all system events (such as status and error messages) in a System Log file.

The log file is a standard text file (default file name is **system.log**) that can be saved to disk and viewed using a text editor (such as Notepad).

If a problem occurs, consult your System Log to see if the information it contains can help solve the problem.

System Log History

Select an option in the Log History area to tell the programmer what to do with previous System Log files saved to disk.

Note: Log History may be ignored unless the Write Log to File *checkbox is enabled (checked).*

Select one of the following options for log history:

Option	Action
Clear log at startup	The existing log file, if any, is erased each time the programmer software is started.
Append to previous (default)	The log file information is appended to an existing log file.

System Log Options

Two options are available that affect the way the System Log file is handled.

You can choose to enable or disable the following option:

Option	Action
Write log to file	If enabled (checked), the System Log is written to a disk file. The filename is specified in the Logfile text box. If this option is not enabled, the System Log is kept in memory while your programmer is running. However, the System Log is lost when you exit your programmer software.

Note: The System Log is still kept in memory regardless of whether the Display Log at Startup *and* Write Log to File *options are selected.*

Swap File...

Select this command from the Preferences menu to specify the size and storage location of the swap file for your programmer.

From the Swap File Preferences dialog box you can change the following settings:

- Filename
- Memory Buffer Size
- User Memory Size

Note: *The programmer software swap file affects only your programmer and has no effect on the swap file used by Windows.*

Restoring Default Settings

You can restore the swap file preferences to their system defaults by choosing the **Set To Default** button.

If you choose the Set To Default button, the Filename, Memory Buffer Size, and User Memory Size will be returned to their system defaults, which are:

Parameter	Default Setting
Filename	dioswap.swp
Memory Buffer Size	2048 KB
User Memory Size	2048 KB

TIP:

You can effectively disable swapping by setting the Memory Buffer Size greater than the User Memory Size. This is recommended if your PC has 8 MB or more of RAM.

After Making Changes

After you have made changes in the Swap File Preferences dialog box, choose **OK** to accept the changes. If the parameters you entered are valid, they will be saved to disk and take effect immediately. If the parameters you entered are not valid, a message will be displayed prompting you to either correct the parameters or choose Cancel.

If you choose Cancel, your changes are not saved and the previous Swap File Preferences settings are used.

About the Swap File

The programmer software can use virtual memory to increase the amount of memory available for your programmer and data files. Virtual memory is a combination of the RAM in your PC and disk space on your PC's hard drive.

Virtual memory works by swapping data to and from your PC's RAM to a **swap file**, which is usually created on your PC's hard disk. The advantage of using virtual memory is that more memory is available for you to use. The disadvantage is that because memory is being swapped to a file (instead of RAM), the system might run more slowly.

Note: The swap file affects only the memory for your programmer. The programmer software swap file has no effect whatsoever on the swap file used by Windows.

For example, if you have 4MB of free RAM and 75MB of free disk space, you have a total of 79MB of virtual memory that could be available.

Of course, you would not want to devote all of your free disk space to virtual memory. You can tell your programmer how much virtual memory to use. The total amount of memory you want your programmer to use is the size of the swap file.

Normally, your programmer will keep as much data as allowed in RAM, which makes operations faster. However, as you work with larger data files, your programmer might need to temporarily swap some data to disk, allowing it to work with the entire data file. When data is temporarily swapped to disk, the file the data is temporarily stored in is called the swap file.

Swap File Filename

Use the Filename field in the Swap File Preferences dialog box to specify where the Swap File is stored on disk.

The filename you enter in the Filename box must be in the following form:*

```
[x:][\some\path\]filename.ext
```

where

`[x:]`	specifies the drive,
`[\path]`	specifies the path, and

* *The items shown in square brackets "[ ]" are optional. The only required element is the filename.*

`filename.ext`	specifies the filename.

Entering the Log File Filename

You can specify the filename for the swap file in one of five ways:

Filename	Description
`<blank>`	Not specifying a filename will cause the swap file to be written to the default drive, path, and filename.
`filename.ext`	The swap file will be written to the file you specify in the current directory on the current drive.
`\path\filename.ext`	The swap file will be written to the file you specify in the directory you specify on the current drive.
`x:filename.ext`	The swap file will be written to the file you specify in the current directory on the drive you specify.
`x:\path\filename.ext`	The swap file will be written to the file you specify in the directory you specify on the drive you specify.

TIP:

You can speed up the swap file by specifying a RAMdisk as the location for the swap file. For example, if drive D is a RAMdisk, you could specify `D:\swap.swp` in the Filename field.

Memory Buffer Size

The Memory Buffer Size is the amount of RAM in the PC to be reserved exclusively for use by your programmer.

The amount of memory you should reserve for you programmer via the Memory Buffer Size parameter is dependent on the amount of RAM you have in your PC. If you have a relatively small amount of RAM in your PC, such as 4 or 6MB, you might want to stay with a conservative value. In this case, the system default of 2048 KB should be fine.

If you have more memory in your PC, you might want to increase the Memory Buffer Size.

User Memory Size

User Memory Size is the largest amount of data that your programmer will be able to handle. The default User Memory Size is 2048 KB, which is 200,000h bytes.

The size of the swap file is User Memory Size + 512 KB.

Note: *The default value for User Memory might vary depending on your programmer system. The default value is listed in the* winchip.ini *file.*

Powerup Defaults

This command allows you to start up the program with preset parameters.

The Powerup Defaults command, from the Preferences menu, opens a dialog box that allows you to enable and save Powerup Defaults. There are three selections in the Powerup Defaults dialog box: Factory Defaults, Automatic, and User Defaults. The selection is saved the in the program's initialization file (winchip.ini) and is carried over from session to session.

Factory Defaults

When this option is selected, no powerup defaults are in effect. The factory defaults are loaded at startup. This is the default option.

Factory Defaults

The factory defaults include the following system settings:

Parameter	Factory Default Setting
Blank Check	Yes
Compare Electronic ID	Yes
Data Word Width	8
Device Begin Address	0
Device Block Size	0
Device Name	Blank
Enable Security Fuse 1	No
Enable Security Fuse 2	No
Enable Security Fuse 3	No
Enable Software Data Protect	No
Enable Erase Before Program	No
File Address Offset	FFFFFFFF

Parameter	Factory Default Setting
File Data Format	Auto Detect
Filename	Blank
Fill Memory option	None
Fill Memory value	0
Illegal Bit Check	Yes
Insertion Check	Yes
JEDEC translate DIP/LCC option	Yes
Logic verify options (both, fuse, functional)	Both
Manufacturer	Blank
Memory Begin Address	0
Odd/Even Byte Swap	No
Reject option (manufacturer or single)	Manufacturer
Security Fuse Data (0 or 1)	0
Serial set auto increment option	No
User Data Size	0
Verify Data Format	Hex
Verify Passes	2

Automatic

When this option is selected, the program saves the system settings which are in effect when you exit the program. These settings are saved to a configuration file (autosave.cfg) in the program directory and are then restored the next time the program is run. The Powerup Defaults menu item will be checked when this option is enabled.

Note: *A device must be selected or the configuration will not be saved on exit. A warning will be displayed when you click OK to exit the Powerup Defaults dialog box.*

User Defaults

When selected, this feature causes the program to restore the user-defined system settings during startup. Save the user-defined system settings by clicking the Save Current Settings button. If no powerup configuration file (powerup.cfg) exists and this option is selected, a warning will be displayed when you click OK to exit the Powerup Defaults dialog box. The Powerup Defaults menu item will be checked when this option is enabled.

Save the current settings

When this button is clicked, the program saves the current system settings in a configuration file (powerup.cfg), which can be restored the next time the program is run with User Defaults selected in the Powerup Defaults dialog box.

OK

Click this to close the Powerup Defaults dialog box and accept the options you have set.

Cancel

Click Cancel to close the Powerup Defaults dialog box and abandon any changes you have made.

If Powerup Defaults are enabled (Automatic or User Defaults is selected) but no powerup configuration file (autosave.cfg or powerup.cfg) is found, the factory defaults are loaded at startup.

Options Menu

View System Log

Select this command to display the System Log, which keeps a record of all events that happen while you are using the programmer.

To display the System Log

- To display the System Log, click the **System Log** button on the toolbar or select **View System Log** from the Options menu (ALT, O, S).

To close the System Log

Note: Even though the System Log is closed, events are still being written to the System Log. Choose the **System Log Preferences** *command to set preferences (such as saving the System Log to disk) for the System Log.*

- If the System Log is open and is minimized (in an icon form), perform one of the following actions:
  - Double-click the **System Log** button in the toolbar.
  - Click the System Log icon, then choose **Close** from the System menu in the System Log window.
  - Select **View System Log** from the Options menu (**ALT, O, S**). Then reselect **View System Log** from the Options menu.
- If the System Log is open and is displayed as a window, do one of the following:
  - Click the **System Log** button in the toolbar.
  - Select **View System Log** from the Options menu (**ALT, O, S**).
- You can also use the **System Log Preferences** command to change the behavior of the System Log.

Printing the System Log

Choose this command to print the System Log.

To print the System Log

1. Click the **System Log** button in the toolbar or choose **View System Log** from the Options menu (**ALT, O, S**) to display the System Log.
2. Choose **Copy to Clipboard** from the Edit menu or press **CTRL + C** to copy the System Log to the Windows Clipboard.
3. Start a text editor such as Windows Notepad.
4. Paste the contents of the Clipboard into the text editor.

Note: Windows Notepad has a limit of approximately 50,000 characters, which is approximately 625 lines of System Log information (depending on line length).

5. From the text editor, select the Print command.

About the System Log

The System Log keeps a record of all events that happen while you are using the programmer. Events that are noted in the System Log include the following:

- Status messages
- Error messages
- Device and manufacturer selected
- Results of device operations, including reading, programming, and verifying
- Sumchecks of data operations, including reading from device, opening a data file, and calculating the sumcheck of a range of data.

Calculator

The calculator is the standard Windows calculator.

If the calculator comes up in Standard mode, choose **View Scientific** (ALT, **V**, **S**) to switch to Scientific mode. In Scientific mode you can operate in hexadecimal, octal, binary, and decimal. You can also perform common bitwise operations, such as Xor, And, and Or.

For more information, consult the online help supplied with the Windows Calculator. To view the online help for the Windows Calculator and press **F1**.

Load Configuration

This command allows you to restore a set of previously saved system parameters from a configuration file (.cfg). The system parameters include the device, the data file, the file parameters, and the device options. This feature is useful because it allows you to have numerous configurations for a device, or for different devices, and to be quickly able to restore a configuration without having to reset each system parameter.

To load a configuration:

1. Select **Load Configuration** from the Options menu. A list of all the configuration files that have been saved in the current drive and directory is displayed.
2. Change the drive and directory if needed. Select the configuration file you wish to load.
3. Click **OK**. The program loads the configuration file and sets the system parameters. The System Log reports the device algorithm and configuration file that were loaded.

Save Configuration

Use the Save Configuration command from the Options menu to save a set of system parameters to a configuration file (.cfg) for future use. The system parameters include the currently-selected device, the open data file, the file parameters, and the device options. A device must be selected in order to save a configuration.

To save a configuration

1. Before using the Save Configuration command, you must select a device. You can then create or open a data file and/ or go through the various File and Device menus, setting the system parameters to fit your particular applications.
2. When you have the program configured the way you want it, select Save Configuration from the Options menu.
3. The Save Configuration dialog box will be displayed. Choose the drive and directory in which you want the configuration file to be saved and enter a name for the configuration file. By default, the extension ".cfg" will be appended to the filename.
4. Click OK. When the process is complete, the program will display a message stating that your configuration has been saved and showing the path and filename of the saved configuration.

System parameters

The following system parameters are saved in the configuration file:

General

Manufacturer, Device, Hardware configuration
Data File
File Parameters
Data Format
Memory Fill Option
Fill Specific Value
Memory Begin Address
User Data Size
Address Offset
DIP/LCC Translate

Device Options

ID Check
Insertion Check
Byte Swap
Set Auto Increment
Illegal Bit Check
Blank Check
Program Security Fuse 1
Program Security Fuse 2
Program Security Fuse 3
Erase Device
Device Begin Address
Device Block Size
Data Word Width
Load User Data Size
Load Set Size
Total Set Size
Security Fuse Data
Reject Option
Verify Passes
Verify Format
Verify Option

Self Test

Use the programmer Self Test command to run a system self test. Before running the self test, you can choose how long to run the self test and which tests to run.

Tests

This option allows you to choose which tests to run and which tests to skip. The results of each test are displayed to the right of each test.

The following self tests are available:

- Contact Programmer
- Initialize Interface
- RAM Check
- Program Control
- DAC Calibration
- Pin Logic Control
- Pin Drivers
- Base/Adapter Relays

Contact Programmer

Tests the low-level communication between the PC and the programmer.

Test Results:

PASS	Communication between the programmer and the PC has been established.
FAIL	No communication has been established between the programmer and the PC, or the communication was not sufficient to ensure consistent communication between the programmer and the PC. A message is displayed informing you of the test failure or, if you are running the self test at startup, the event is noted in the System Log.

Initialize Interface

Tests the mid-level communication between the PC and the programmer.

Test Results:

PASS	Communication sufficient for operation has been established between the programmer and the PC.
FAIL	The programmer and the PC were not able to establish the level of communication that is needed for operation of the programmer. A message is displayed informing you of the test failure or, if you are running the self test at startup, the event is noted in the System Log.

RAM Check

Test the RAM in the programmer.

The test writes, reads, and verifies four different patterns into the RAM in the programmer. The patterns are 00, FF, 5A, and A5. Each pattern is written to the RAM in the programmer then read back from the programmer, and the two values (the written value and the read value) are compared.

The test continues until all RAM has been tested, or until a location in RAM fails the test.

Note: This test checks only the RAM in the programmer; RAM in the PC is not tested.

Test Results:

PASS	The RAM in the programmer is functioning normally.
FAIL	At least one memory location in the RAM in the programmer did not pass the RAM test. A message is displayed informing you of the test failure or, if you are running the self test at startup, the event is noted in the System Log.

Program Control

Tests high-level communications between the PC and the programmer. This test also checks basic I/O functions and verifies that the programmer is fully functional before any further tests are performed.

This test differs slightly from the Contact Programmer test and the Initialize Interface tests.

Test Results:

PASS	The PC is able to control the programmer.
FAIL	Consult the displayed message or the System Log (if no message is displayed) for the reasons for test failure. Your programmer might not be properly installed, or there might be a hardware failure.

DAC Calibration

Tests and calibrates the DACs (Digital to Analog Converters) in the programmer.

Test Results:

PASS	DACs are properly calibrated.
FAIL	Consult the displayed message or the System Log (if no message is displayed) for the reasons for test failure. Your programmer might not be properly installed, or there might be a hardware failure.

Pin Logic Control

Tests the pin logic ICs inside the programmer.

Test Results:

PASS	Pin logic is functioning properly.
FAIL	Consult the displayed message or the System Log (if no message is displayed) for the reasons for test failure. Your programmer might not be properly installed, or there might be a hardware failure.

Pin Drivers

Performs digital and analog test on the pin drivers inside the programmer.

Test Results:

PASS	Pin drivers are functioning properly.
FAIL	Consult the displayed message or the System Log (if no message is displayed) for the reasons for test failure. Your programmer might not be properly installed, or there might be a hardware failure.

Base/Adapter Relays

Tests the base in the programmer and the relays in the programmer.

Test Results:

PASS	Base and adapter are functioning properly.
FAIL	Consult the displayed message or the System Log (if no message is displayed) for the reasons for test failure. Your programmer might not be properly installed, or there might be a hardware failure.

Enable All

Enables all the tests shown in the Tests group.

Your programmer records the date and time when a full self test is run. After the amount of time specified in the Retest After field has passed, your programmer runs a full self test the next time it is powered up.

Clear All

Disables all the tests shown in the Tests group.

Note: The tests will be run after the time limit specified by the Retest After option.

Same as Startup

Sets up the tests the same as in the Startup Self Test dialog box.

Test Duration

Runs the tests selected in the Tests group once or continuously.

Stop

Stops a running self test.

Test

Starts running the tests selected in the Tests group.

Close

Closes the Start-up Self Test Preferences dialog box and **saves** all changes made since the dialog box was opened.

Cancel

Closes the Start-up Self Test Preferences dialog box and **abandons** all changes made since the dialog box was opened or since the last time you ran a self test.

Aging Self Test Results

The programmer verifies its performance every time a device operation (such as programming) is run. To ensure that your programmer's performance remains fully verified, we recommend that you run a full self test at least once every three months.

To age the self test results, type an amount of time in the **Retest After** field. When this amount of time has passed since the last full self test was run, a full self test is run (regardless of how the Powerup Self Test is set).

Why Run a Full Self Test?

There is a trade-off between reliability and convenience. Some tests (such as the RAM test and the DAC Calibration) take a few seconds and lengthen the time required to get the programmer ready for use. To reduce the time required to get the system up and running, many users configure their systems to not perform any powerup tests. While this is an effective way of minimizing boot time, it also creates a subtle, risky side effect. Over time, the programmer can drift out of conformance to standards, causing programmer device yield to drop.

To help achieve a balance between reliability and convenience, you can "age" the self test results. With aging, the programmer records the date and time when a full self test is run. Then, after a preset length of time has passed, the programmer runs a full self test the next time the programmer is started.

Window Menu

New Window

To open another window, choose **New Window** from the Window menu. The new window has the same display settings and contents as the one that was previously active.

Arrange Windows and Icons

The Tile command resizes and arranges the open group of windows side by side. The Cascade command resizes and layers open group windows so that each title bar is visible. To arrange group windows on your desktop, from the Window menu, choose **Tile** or **Cascade**.

Use the Arrange Icons command to evenly arrange the icons in a group window. To arrange program-item icons, choose **Arrange Icons** from the Window menu.

List of Open Windows

Displays a list of the files that are currently loaded. The file with the check (✓) by it is the active window.

Help Menu

Contents	This command displays the Contents topic for the online Help.
Messages	This command displays the online Help for system messages.
Using Help	This command opens the Windows Help system.
About	This command displays version and registration information for the programmer software.

6. Troubleshooting

Introduction

This chapter includes the following sections:

0002 Device Overcurrent Fault

Probable Cause	Solution
Improper device selected	Make sure the device selection matches the manufacturer and part number of your device as precisely as possible. If it doesn't, select the proper characteristics and perform the operation again. *Note: Choosing an incorrect manufacturer and/or part number (via Select from Device menu) for the device you are using causes the programmer to expect an electronic ID different from your device's ID.*
Faulty device(s)	Load the suspect device (via Read from the Device menu). If an overcurrent error occurs, try reading new devices. If other devices are read with no errors, a single device might be faulty. If no errors occur during the read operation, try to program another device (via Program from the Device menu). If other devices program without error, a single device might be faulty. If other devices also produce overcurrent errors during read or program operations, check the date code. If failures occur only on devices with a particular date code while other parts are programmed or read successfully, the problem is probably device related. *Note: If the problem appears to be device related, you may wish to notify the device manufacturer.*
Device-Specific Error	If devices of the same type produce overcurrent errors during read and program operations, disable the insertion check parameter (found in the Read, Program, and Verify Device dialog boxes). If this error occurs during a program but not a read operation, a device test might be causing the problem. Report your findings to Data I/O customer support; there may be a bug in the software. Disable all device checks listed on the Program Device screen (such as Device check and Illegal bit check) by changing the appropriate check boxes from checked to unchecked.
Improper algorithm applied	If an algorithm error occurs on devices with old and recent date codes, the devices might require a modified programming algorithm. *Note: Report your findings to Data I/O Customer Support.*
Programmer hardware	An overcurrent error occurring with different devices may indicate a hardware problem. To determine whether a hardware malfunction exists, perform a self test (via **Self-test** from the Options menu) with no devices in the socket. If a malfunction occurs, refer to your warranty for service procedures.

Additional Information

Overcurrent errors are reported by the hardware overcurrent detection circuitry on the programmer. The trip level for the overcurrent error is set by the programming algorithm. From the error alone, it is not possible to determine which operation the programmer was performing (device tests, program, verify/ read) when the overcurrent condition was detected. To determine the nature of the problem, you need to isolate the operation being performed.

5025 I/O Translation: Partial or No Data Translated

Probable Cause	Solution
I/O Address Offset = FFFFFFFF and first file address is not the lowest	Absolute translation: To transfer data from file to corresponding RAM locations, enter an I/O Address Offset value of 00000000. Offset translation: If file data must begin at the programmer RAM location 0000h, find the lowest file address and use that value as the I/O Address Offset.
Improper use of Begin Address and/or User Data Size	Your file must be transferred to proper locations in programmer RAM. Edit programmer RAM to determine whether data has been transferred properly. If it hasn't, make sure your Begin Address and User Data Size parameters are appropriate.
File addressing places data outside RAM address range	View your file with an ASCII editor and look for addresses that exceed your programmer's RAM address range.

Additional Information

This warning message indicates that a portion of your file's data has not been transferred into programmer RAM. Transfer problems of this nature occur more often with files that have been generated with addresses in non-sequential order, where the lowest address is embedded somewhere in the middle of the file.

In general, the following formula represents where in programmer RAM data will be transferred.

```
Physical RAM Address = (File Address - Offset Address)
+ Begin Address
```

The default Address Offset, FFFFFFFF, does not represent a numerical hex value. It is simply a flag to indicate that the first address in your file will be used as the I/O Address Offset. By default, the programmer interprets the first file address as the Address Offset and subtracts that value from all of the remaining addresses in the file. Consequently, the data contained in address locations lower than the first address will be lost.

6017 Device Insertion Error

Probable Cause	Solution
Device inserted improperly	Ensure that the device is properly justified in the socket or properly oriented in the MatchBook.
Faulty device(s)	Check the device for bent or damaged pins/leads. Repeat the operation with similar devices from the same manufacturer and from other manufacturers. If the operation proves successful with similar devices, then the suspect part is probably defective.
Socket/pad is dirty or worn	Examine the socket or pad for debris and wear. Clean or replace the socket or pad as necessary. Refer to Chapter 6 for cleaning and replacement instructions.
Insertion check problem	If following the steps described in the previous section causes the device to fail programming, a subtle insertion check problem might exist.

Additional Information

Device insertion errors are caused by a failure of the device insertion check. During the device insertion check, which is activated prior to device programming, the programmer applies low-level current to each pin on the device to determine whether it is making good contact with the programming fixture.

After disabling the device insertion test, we suggest reading a device rather than programming one. A read operation is less apt to harm the device because no programming voltages are applied.

6021 Electronic ID Verify Error With Memory Device

Probable Cause	Solution
Improper device selected	Make sure the device selection matches the manufacturer and part number of your device as precisely as possible. If it doesn't, select the proper characteristics and perform the operation again. *Note: Choosing the wrong manufacturer and/or part number (via Select from the Device menu) causes the programmer to expect an electronic ID that differs from the ID in your device.*
Electronic ID has changed	If the device is labeled with a recent date, the manufacturer may have assigned a new ID to the device that is not recognized by your programmer. To minimize ID errors, use the latest version of your programmer software. Workaround: If your programmer is at the current version, disable the Compare Electronic ID parameter in the Verify Device or Program Device dialog boxes. *Note: You might wish to contact the device manufacturer to find out if the device ID has been changed. If it has, please notify Data I/O Customer Support.*
Faulty device(s)	If disabling the Compare Electronic ID parameter causes an operation (such as load, program, or verify) to fail, try the operation on other devices with the same date code. If the operation is successful on these devices, the original device might be defective. *Note: If a high percentage of parts fails the operation, you might wish to contact the device manufacturer and report your findings.*
Device-Specific Error	If disabling the Compare Electronic ID parameter causes the operation to fail on a high percentage of parts across several date codes, there might be a software bug in the programming algorithm associated with the device. *Note: If you suspect a software defect, report your findings to Data I/O Customer Support.*

Additional Information

Most memory devices are uniquely identified by their silicon signatures (electronic IDs). At times, device manufacturers change the electronic IDs of devices that undergo changes in the manufacturing process. Typically, the electronic ID is altered to promote automatic device selection and usually does not reflect a change in the device's programming specifications. Consequently, disabling the electronic ID check is a viable workaround for most memory devices.

When electronic ID verify errors occur, the electronic ID of the device is displayed. You can also determine the electronic ID of your part by selecting the Compare Electronic ID device check option.

The Compare Electronic ID option is located in the Read Device, Program Device, and Verify Device dialog boxes.

6023 Illegal Bit Error

Probable Cause	Solution
Windowed device not blank	Perform a blank check on the device (via Blank Check from the Device menu) to determine whether it is truly blank. If blank check reports "non-blank device," place the device under a UV lamp and allow it sufficient time to fully erase. After erasure, reprogram the device.
EE device not blank	Perform an Erase on the electrically erasable (EE) device (via Erase from the Device menu). After erasing, reprogram the device. Or reprogram the device with the electronic erase option enabled.
OTP device not blank	For One Time Programmable (OTP) devices, perform a blank check on the device to determine whether the device contains data. If so, your OTP device had been programmed previously and most likely cannot be over-programmed with different data or fuse pattern. Program another device.
Faulty device(s)	Program at least one more device labeled with the same date code. If it programs successfully, your original device was probably faulty. If other devices with the same date code also fail, attempt to program devices labeled with different date codes. If these devices program successfully, the devices from the original date code are probably faulty. *Note: If you find that many devices with the same date code are faulty, you may want to contact the device manufacturer to report your findings.*
Improper algorithm applied	If an illegal bit error occurs only on devices with recent date codes, the devices might require a modified programming algorithm. *Note: You may wish to contact the device manufacturer to determine whether the programming specifications for the device have changed.* If an illegal bit error occurs on devices with old and recent date codes, this might indicate an algorithm-related problem in your programmer software. *Note: Report your findings to Data I/O Customer Support.*

Additional Information

An "illegal bit error" indicates that at least one location in the device contains data (programmed state) while its corresponding location in RAM has no data (unprogrammed state). For example, the unprogrammed state of a PROM is 0, while its programmed state is 1. If a particular 8-bit PROM's memory location contains 09 hex (00001001 binary) and the corresponding memory location in RAM contains F0 hex (11110000 binary), then an "illegal bit error" will occur because the programmer is not able to unprogram the first and fourth least significant bits.

6025 Device Programming Error

Probable Cause	Solution
Improper device selected	Make sure the device selection matches the manufacturer and part number of your device as precisely as possible. If it doesn't, select the proper characteristics and perform the operation again. *Note: Choosing the wrong manufacturer and/or part number (via Select from the Device menu) causes the programmer to expect an electronic ID that differs from the ID in your device.*
Faulty device(s)	Program at least one more device labeled with the same date code. If the operation is successful, the original device is probably faulty. If other devices with the same date code fail, try to program devices labeled with different date codes. If devices with different date codes are programmed successfully, the devices from the original date code are probably faulty. *Note: If you find that many devices with the same date code do not program successfully, you might wish to contact the device manufacturer and report your findings.*
Improper algorithm applied	If device programming errors occur only on devices with recent date codes, the devices may require a modified programming algorithm. *Note: You might wish to contact the device manufacturer to determine whether the programming specifications for the device have changed. If they have, please notify Data I/O Customer Support.* If device programming errors occur on devices spanning old and recent date codes, this might indicate an algorithm-related problem in your programmer software. *Note: Call Data I/O Customer Support and report your findings.*

Additional Information

A device programming error is reported when a repeated attempt to program a particular cell or fuse has failed. This error may be caused by a faulty device or an improper programming algorithm.

Error Messages

0002 Hardware Error: Overcurrent

Cause

The hardware detected an overcurrent condition.

Solution

Remove the device from the socket and try the operation again. If the error message goes away, try a different device. Otherwise, the hardware is not functioning correctly or the algorithm for the device is defective. Refer to "Device Overcurrent Fault" on page 82 for more detailed information.

2019 Cannot find a Programmer Pod at port *port_number*

Cause

The programmer software failed to connect with the programmer pod via the specified port.

Solution

Make sure the programmer and PC are connected properly and that the port specified in the **Preferences: Hardware** dialog box is correct. If you are unsure about the connection, you can have the programmer software select a port for you by selecting **Automatic** in the **Preferences: Hardware** dialog box. (If you change the port settings, you might need to change the port driver to match the new port settings.)

2020 Cannot find a Programmer Pod at port address *address*

Cause

The programmer software did not find a port at an address specified for a port for one of the following reasons:

- The port (such as LPT1) does not exist, or the memory associated with the port has been overwritten by another program or device.
- The software is looking for a port (such as LPT1 or LPT2) at a standard address in memory, but your PC has a non-standard configuration.
- A specific address for the port was specified in the **Preferences: Hardware** dialog box, but no port was found at the specified address.

Solution

You might want to explicitly specify the parallel port to which the programmer is connected, or, if you are using a non-standard PC configuration, you can select the I/O base address to use in the **Preferences: Hardware** dialog box.

5001 Disk file: Could not open file *file_name*

Cause	You have tried to open a file whose contents cannot be read (the data is in an unreadable format).
Solution	Make sure the correct filename and path are specified.

5003 Memory: Out of Memory

Cause	An operation cannot be completed because it requires more memory than is currently available.
Solution	Free up more extended memory (close unnecessary applications or reduce the size of any disk-caching programs or any installed RAM drives). You might need to add more extended memory to your PC.

5020 I/O translation: Invalid format number selected

Cause	This message appears if you are loading a data file using a batch file or the command line, and the number specified for the data translation format is not supported or is incorrect.
Solution	Make sure that the number is typed correctly and try again. Refer to the help on data formats for supported format codes

5021 I/O translation: Format error in file

Cause	This message appears when the user data file being translated does not conform to the rules of the specific format being used. The file might be in a different format than specified or data in the file might be corrupted.
Solution	Make sure that you specified the correct format for the file and try opening the file again. If the error persists, try different formats and check to see if the contents of the file have been corrupted.

5022 I/O translation: Checksum error in file

Cause	The sum of the data translated from one record of the user data file does not match the checksum number at the end of the record. For more information about the checksum calculation, see page 54.

5023 I/O translation: Premature end of file detected

Cause	The programmer software encountered an EOF (end of file) string before the physical end of the file was reached.
Solution	Make sure that the selected data format is correct for the file being opened.

5024 I/O translation: Escape character detected in file

Cause	The programmer software encountered an ESC (escape) character while reading the file.
Solution	Make sure that the selected data format is correct for the file being opened.

5025 I/O translation: Partial or no data translated

Cause	Some or all of the data from a file were skipped because of the addresses of the data in the file. Refer to page 84 for more detailed information.

5026 I/O translation: No data translated from file

Cause	The file specified to be loaded does not contain any data to be translated.
Solution	Make sure that the selected data format is correct for the file being opened.

5027 I/O translation: Format error in JEDEC file

Cause	The user data file being translated does not conform to the rules of the specific format being used.
Solution	Make sure that the selected data format is correct for the file being opened.

5028 I/O translation: Fuse checksum error in file

Cause

The sum of the fuses of the fuse array (translated from the user JEDEC file) does not match the Fuse Checksum at the end of the JEDEC file. For more information about the checksum calculation, see page 54.

5029 I/O translation: Transmit checksum error in file

Cause

The sum of the characters in the translated user JEDEC file does not match the Xmit Checksum at the end of the JEDEC file. For more information about the checksum calculation, see page 54.

5030 User data: Incompatible (fuse) in user data for device selected

Cause

A device operation was attempted with an inappropriate fuse map in user memory.

Solution

Select an appropriate device for the fuse map (JEDEC file) you intend to program.

5031 User data: Incompatible user data for devices selected

Cause

A device operation was attempted without the appropriate vectors in user memory. This message may also appear if a JEDEC file is being used that contains vectors for a PLCC device to be programmed in a DIP socket adapter.

Solution

Select an appropriate device for the JEDEC file you intend to program with, or, if using a PLCC in a DIP socket, try regenerating this file for a DIP device.

5033 Disk file: Could not write to file

Cause

This message appears when the programmer software unsuccessfully attempts to save or write to a file.

Solution

If saving a file, make sure the filename and path are correct. Also make sure the file is not write-protected or locked by another user.

5037 Memory: End of user memory exceeded

Cause

A RAM variable has exceeded the amount of user RAM allocated. For instance, a Fill Memory operation might have attempted to fill an address outside of user RAM.

Solution

Check the amount of user RAM (user RAM is set in the Swap File dialog from the Preferences menu) and check the parameters for the last operation the programmer attempted. Either set all parameters so that user RAM is not exceeded, or increase your user RAM.

6001 Device is not blank

Cause

A programming operation has been attempted on a non-blank device and the programmer is not set to erase a device before programming (a programming operation cannot be attempted on a non-blank device).

Solution

Make sure the master device was not left in the socket. If you still wish to program the device, either erase the device, or remove the device and insert a blank device before programming.

6017 Device Insertion Error

Cause

This message can occur if any one of the following is true:

- The device socket is not in the locked position.
- The device pins are not making good contact.
- The device is inserted backwards or is not bottom-justified in the socket.

Solution

Check the device and try the operation again, or try another device. See "Device Insertion Error" on page 85 for detailed information.

6020 Security Fuse Violation

Cause	This message appears if a device operation (such as read, program, or verify) was attempted on a device that has its security fuse programmed.
Solution	Use a new or master device that does not have its security fuse programmed.

6021 Compare electronic ID operation failed

Cause	The electronic ID read from a device does not match the manufacturer's specified ID. See "Electronic ID Verify Error" on page 86 for detailed information.
Solution	Verify that the correct device is selected from the menu, or try another device.

6022 Bulk erase operation failed

Cause	The programmer is unable to erase an EEPROM device.
Solution	The device may be defective. Try the operation again, or try another device.

6023 Illegal bit check failed

Cause	Appears when a device has a bit programmed to the incorrect state. See "Illegal Bit Error" on page 88 for detailed information.
Solution	Try erasing the device and attempt to program the part again. If this message continues to appear, the device may be defective. Try another device.

6025 Device programming error

Cause	The programmer detected a defective memory cell in a device during the programming operation. See "Device Programming Error" on page 90 for detailed information.
Solution	Try another device.

6027 Device verify error

Cause	The data in User RAM does not match the Device data.
Solution	If this message appears after a programming operation, try to reprogram the device or try another device.

6028 Program security operation failed

Cause	A device-programming operation failed while trying to program the security fuse (the security fuse is programmed when the security fuse option is enabled and the security data is equal to 1).
Solution	If this message appears after a programming operation, try to reprogram the device or try another device.

6030 Logic data verify operation failed

Cause	Appears when a device verify operation is performed and the data in User RAm does not match the Device data.
Solution	If this message appears after a programming operation, try to reprogram the device or try another device.

6033 Test vector failed

Cause	A functional test on a logic device has failed, or a test was attempted on a device with more than 44 pins (devices with more than 44 pins are not supported).
Solution	Make sure the device is properly seated in the socket. Try the test again. If the message persists, erase and reprogram the device or try another device.

6054 Partial operation not allowed on this device

Cause	This message appears when custom block limits are used for a device that only supports the default block limits.
Solution	Use the default block limits or set the block size to be equal to or greater than the device size.

6058 Partial data operation not allowed on this device

Cause

This message appears when custom block limits are used for a device that only supports the default block limits.

Solution

Use the default block limits or set the block size to be equal to or greater than the device size.

6085 User Data Size is invalid for the specified Data Word Width

Cause

The specified User Data Size is not allowed because it is not a multiple of the Data Word Width.

Solution

Specify a User Data Size that is a multiple of the number of bytes used to represent one data word. For example, if the Data Word Width is set to 16 bits (2 bytes), the User Data Size must be a multiple of 2 (2 bytes are used to contain the 16 bits).

9001 Unable to create swap file *swapfile_name*

Cause

This message appears when the programmer software unsuccessfully attempts to create the specified swap file.

Solution

Check to make sure the path and filename for the swap file are correct. Also make sure that a file by the same name does not exist in the destination directory.

9002 Not enough space for swap file *swapfile_name* size number

Cause

This message appears when the programmer software fails to write to a swap file because there is not enough disk space available for the file.

Solution

Either select a smaller size for the swap file or clear more disk space.

9003 Write to swap file *swapfile_name* failed

Cause

This message appears when the programmer software fails to write to a swap file.

Solution

Check to make sure the path and filename for the swap file are correct. Also, check to make sure the swap file is not write-protected.

9004 Read from swap file *swapfile_name* failed

Cause

This message appears when the programmer software fails to read from a swap file.

Solution

Check to make sure the path and filename for the swap file are correct. If path and filename are correct, the problem might be caused by a corrupted swap file. You might need to exit the program, delete the old swap file, and create a new swap file.

9005 Error writing to swap file

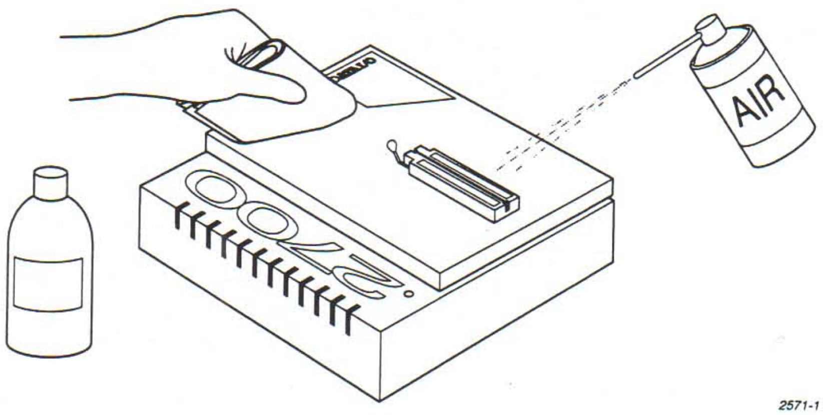
Refer to message 9003.

7. Maintenance

Cleaning the 2700

Clean the 2700 exterior with a soft cloth dampened with anhydrous ethyl alcohol (see Caution statement below). To remove particles from the socket area, use low-pressure, clean, dry air.

CAUTION: ***Make sure the solvent has been approved for use by your company.***



2571-1

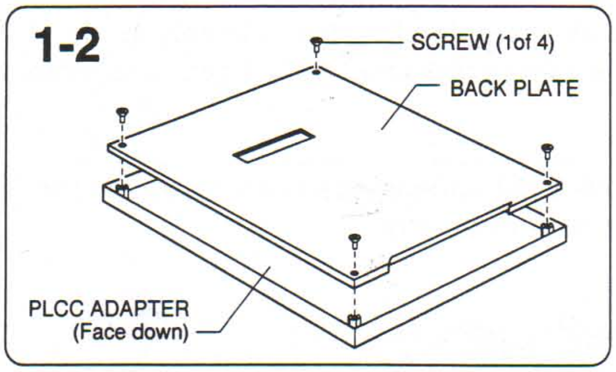
Changing the Conductive Pad on the PLCC Base

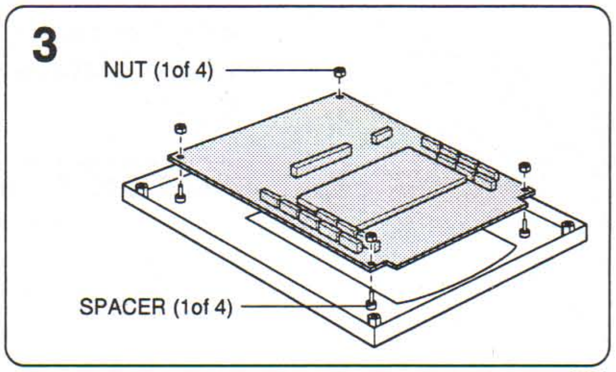
The manufacturer recommends that the conductive pad be replaced after approximately 1000 device insertions. Perform the following steps to replace the pad.

1. Remove the PLCC base from the 2700, move the retaining latch to a horizontal position, and place the base face down on a flat, antistatic surface.

CAUTION: ***To prevent damage from electrostatic discharge, wear an antistatic wrist strap with a 1 MΩ (min.) to 10 MΩ (max.) isolating resistor.***

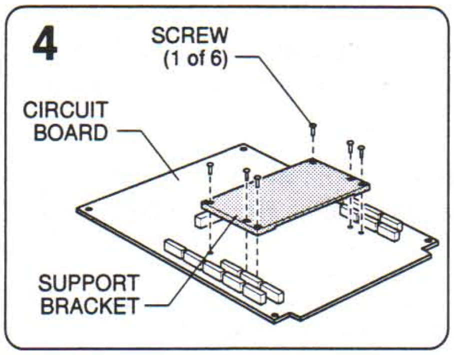
2. With a small Phillips screwdriver, remove the four screws holding the back plate to the top of the base and remove it.
3. With a 1/4-inch wrench, remove the four nuts from the screws holding the circuit board to the top of the base and remove the board. Set aside the four plastic spacers.

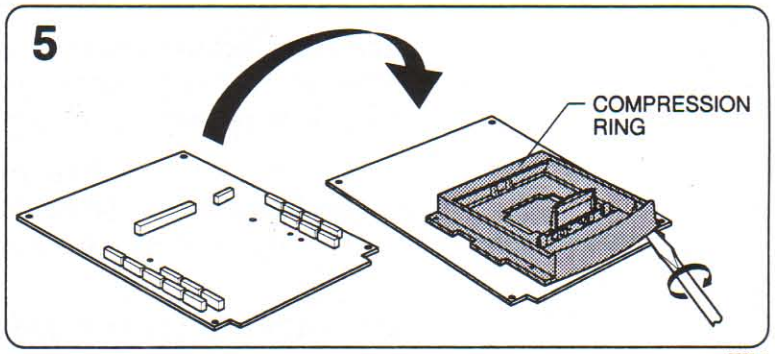




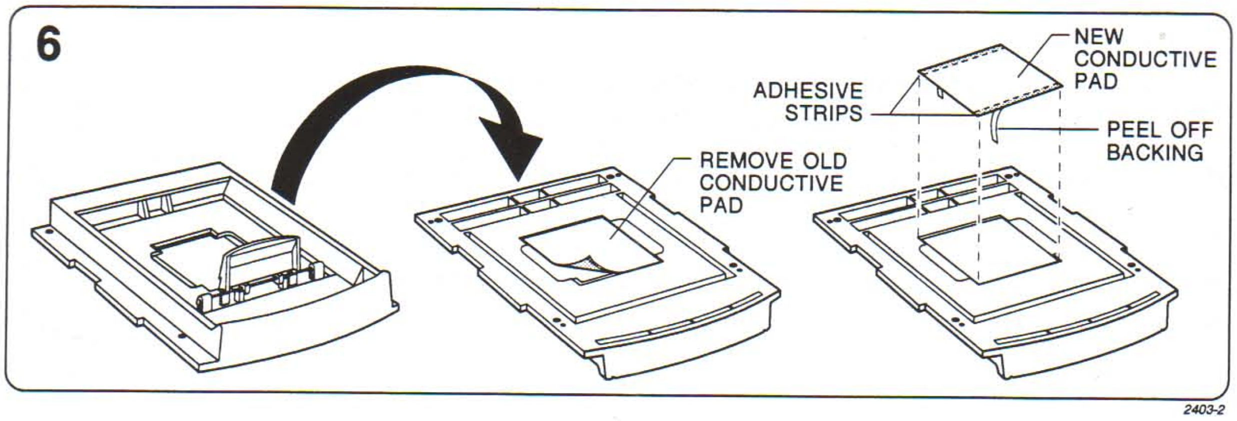
4. With a small Phillips screwdriver, loosen the six screws holding the support bracket to the circuit board and remove the bracket.
5. Turn over the circuit board. Carefully insert a screwdriver around the edges of the compression ring and pry it off.

CAUTION: Do not scratch the exposed surface of the board under and around the conductive pad.





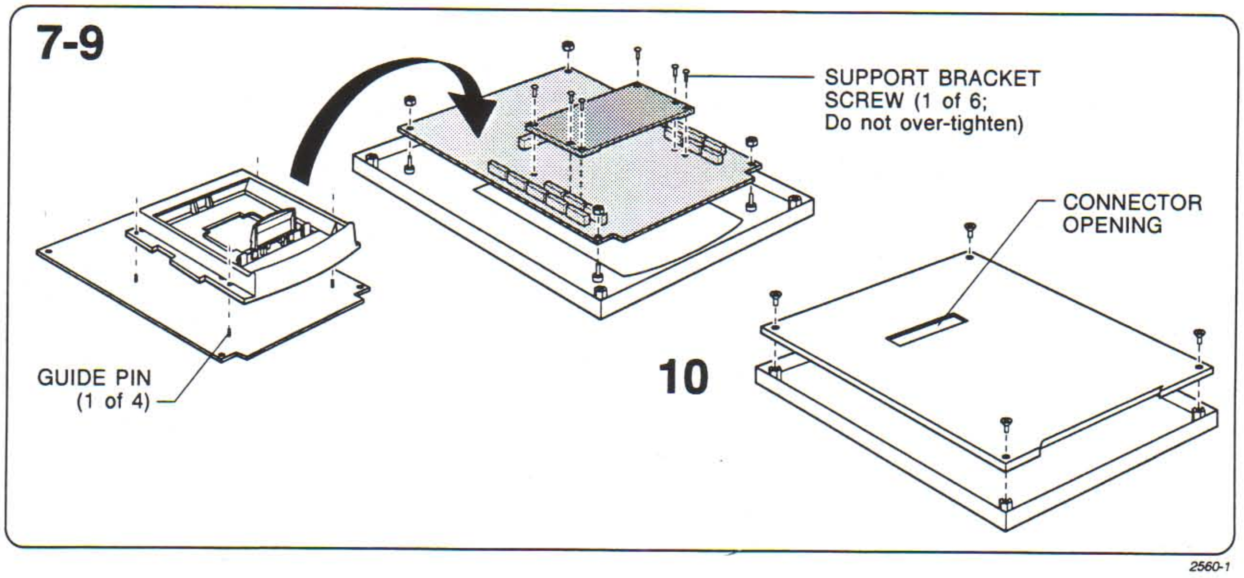
6. Turn over the compression ring, remove the old conductive pad, and attach a new one.



7. Line up the base over the guides on the circuit board and press the base onto the board.
8. Turn over the board and reinstall the six support bracket screws.

CAUTION: To prevent damage to the base, do not overtighten the screws.

9. Place the board over the four screws in the base top cover, and reinstall the nuts.
10. Reinstall the back plate so that the connector is visible through the opening.



The replacement procedure is complete.

8. Programming Considerations

This section includes the following information on some advanced uses of the 2700 programmer:

Odd/Even Split: Using 8-bit devices with 16-bit data

General Information

This section discusses programming applications that require splitting data across multiple devices that are to be placed "side by side" in a system and share the same addresses. For such programming applications, the programmer looks for a Total Set Size that reflects the number of virtual (and not physical) devices in your system.

For example: When programming 8-bit devices in a 16-bit format, the number of physical devices is two (2), while the number of virtual devices is one (1). The programmer will properly split the data between the two devices if the Total Set Size is given a value of "1."

Procedure

To program two 8-bit devices in a 16-bit format, perform the following steps:

1. Select manufacturer and part number of the 8-bit device to be programmed (via Device: Select from the Main menu).
2. Open the 16-bit wide file (via File: Open from the Main menu).
3. Set the parameters in the Program Memory Device screen (via Device: Program Device from main menu) as follows:

   Data Word Width = 16

   Total Set Size = 1

   Set Auto-Increment - [X]
4. Program the devices.

When the **Set Auto-increment** parameter is enabled, the programmer alternately programs devices with even-addressed data and odd-addressed data, and consequently causes the **Next Device** parameter to alternate between "1" (first device) and "2" (second device). Furthermore, the read-only **Next Operation begins at** parameter alternates between "0000" (representing devices programmed with even-addressed data) and "0001" (representing devices programmed with odd-addressed data).

Set Programming

Set programming occurs when data is programmed into devices and each device contains only a part of the total data. For example, if the data to be programmed consist of 4000h bytes but the selected device holds only 1000h bytes, four devices are needed to program all the data.

Serial set programming is a subset of set programming in which each device is programmed separately with the appropriate range of data (as opposed to parallel set programming, which is when all devices in a set are programmed simultaneously).

Set programming can be difficult to understand. Some set programming concepts are discussed in the following sections. It is not necessary to be an expert to use set programming, since the programmer does most of the calculations automatically. However, an understanding of the information in this section might prove to be valuable if a problem occurs.

Set Programming with Word Width Equal to Device Width

Each device in a set is programmed with data using sequential addresses. For example, if 4000h bytes of data were to be programmed into four 1000h byte sized devices, the first device would contain the first 1000h bytes of data, the second device would contain the second 1000h bytes of data, and so on.

Set Programming with Word Width Smaller than Device Width

The programmer automatically sets the word width to equal the device width.

Set Programming with Word Width Larger than Device Width

In this discussion, each device to be programmed has the following characteristics:

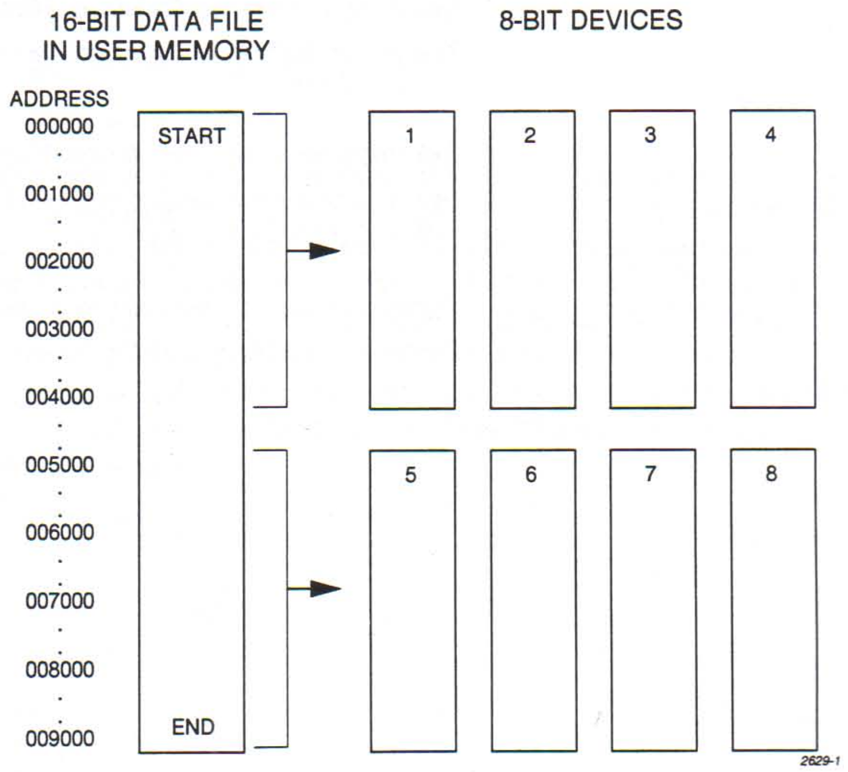
Type:	Generic 8-bit memory device (blank)
Size:	1000h device block size

The programmer is set as follows:

Device Begin:	**0000h device begin address** (where the first byte of data is programmed in each device)
Mem. Begin:	**1000h Memory Begin Address**
I/O Offset:	**FFFFFFh** I/O address offset (I/O offset was set to the default FFFFFF when the data file was read into memory, meaning that the first byte of data was stored where specified by the memory begin address)
Data Width	**32-bit** data word width
Data Size	**8000h** User Data Size
Next Device	**1** in the set for next device
Auto Inc.	**Enabled** (auto increment parameter)

Eight devices are needed to program the data in memory because the user data size is 8000h bytes and each device holds 1000h bytes. The data that will be programmed start at the memory begin address (1000h) and end at memory begin address + user data size (8000h), or addresses 1000 to 8FFF.

The following diagram illustrates the data in user memory and the corresponding devices into which that data was programmed.



Devices 1 through 4 contain the first 4000h bytes of the data file (address 1000 to 4FFF in user memory because of the Memory Begin Address of 1000). Note that each device contains 1000h bytes of data.

- **Device 1** contains the LSBs so addresses 1000, 1004, 1008, 100C, 1010,...4FFC are programmed into the device at addresses 0000, 0001, 0002, 0003, 0004,...0FFF.
- **Device 2** contains addresses 1001, 1005, 1009, 100D, 1011,...4FFD, which are programmed into the device at addresses 0000, 0001, 0002, 0003, 0004,...0FFF.
- **Device 3** contains addresses 1002, 1006, 100A, 100E, 1012,...4FFE, which are programmed into the device at addresses 0000, 0001, 0002, 0003, 0004,...0FFF.
- **Device 4** contains the MSBs or addresses 1003, 1007, 100B, 100F, 1013,...4FFF, which are programmed into the device at addresses 0000, 0001, 0002, 0003, 0004,...0FFF.

Devices 5 through 8 contain the second 4000h bytes of the data file (address 5000 to 8FFF in user memory). Note that each device contains 1000h bytes of data.

- **Device 5** contains the LSBs so addresses 5000, 5004, 5008, 500C, 5010,...8FFC are programmed into the device at addresses 0000, 0001, 0002, 0003, 0004,...0FFF.
- **Device 6** contains addresses 5001, 5005, 5009, 500D, 5011,...8FFD, which are programmed into the device at addresses 0000, 0001, 0002, 0003, 0004,...0FFF.
- **Device 7** contains addresses 5002, 5006, 500A, 500E, 5012,...8FFE, which are programmed into the device at addresses 0000, 0001, 0002, 0003, 0004,...0FFF.
- **Device 8** contains the MSBs or addresses 5003, 5007, 500B, 500F, 5013,...8FFF, which are programmed into the device at addresses 0000, 0001, 0002, 0003, 0004,...0FFF.

Set Size

Used during serial set programming, this value specifies how many virtual devices are in a set.

The set size (total set size) is affected by user data size and the device block size. If either of these parameters is changed, the set size is automatically adjusted according to the following formula.

$$\text{set size} = \frac{\text{user data size X physical device width}}{\text{device block size X data word width}}$$

(rounded up to the nearest integer)

For example, if you selected 8-bit (physical device width) devices with a block size of 4000h, 16-bit data word width, and changed the user data size to 10000h, the set size would be adjusted to 2 [(10000h x 8)/(4000h x 16)].

Set Auto-Increment

When enabled, this option directs the programmer (in serial set programming) to the starting memory address of the next block in the set on which a device operation (such as program and verify) is to be performed.

For example, if you have four 1Kx8 devices to program from a 4Kx8 block of data, using auto-increment directs the programmer to point to the first address of the next 1K block after each device has been programmed. For single device operations, this feature should be disabled and the next device parameter should be set to 1.

Odd/Even Byte Swapping

Disabled (Intel Convention)

When the Odd/even Byte Swap option is disabled (the default condition), the 2700 uses and displays data according to the Intel convention in which the most significant byte (MSB) is the **second** (odd) byte in each 16-bit data word.

Enabled (Motorola convention)

When the Odd/even Byte Swap option is enabled, the 2700 uses and displays data according to the Motorola convention in which the MSB is the **first** (even) byte in each data word.

Examples

The following examples show how user data is allocated when the Odd/even Byte Swap option is enabled and disabled during programming of 16-bit devices.

Sample data file (Motorola EXORmax Format, Code 87):

```
S00B00004441544120492F4FF3

S11300000123456789ABCDEF00112233445566775O

S9030000FC
```

Data File Opened With Format 87 And Displayed In Editor (8-bit Addressing Mode):

```
 CURSOR AT LOCATION: 00000000      8 BIT ADDRESSING

Address                 Hexadecimal                                            ASCII
         -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F  0 1 2 3 4 5 6 7 8 9 A B C D E F
00000000 01 23 45 67 89 AB CD EF 00 11 22 33 44 55 66 77      .#Eg . . "3DUf w

00000010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

2630-1

Programming one 16-bit device, Data word width =16

Odd/even byte swap	Device Address	Device MSB	LSB
Disabled	0	23	01
	1	67	45
	2	AB	89
	3	EF	CD
Enabled	0	01	23
	1	45	67
	2	89	AB
	3	CD	EF

Programming two 16-bit devices, Data word width = 32

Odd/even byte swap	Device Address	Device #1 MSB	LSB	Device #2 MSB	LSB
Disabled	0	23	01	67	45
	1	AB	89	EF	CD
	2	11	00	33	22
	3	55	44	77	66
Enabled	0	01	23	45	67
	1	89	AB	CD	EF
	2	00	11	22	33
	3	44	55	66	77

Appendix A:
Technical Assistance, Warranty, and Repair

This appendix includes details about contacting Data I/O for technical assistance, repair, and warranty services, and the Keep Current™ subscription service.

Customer Support Contacts

You can contact Data I/O for technical assistance by calling or sending a fax. To help us give you quick and accurate assistance, please be at your 2700 and computer when you call, and have the following information ready:

- Version number found in the Help/About information box
- Product serial number found on the bottom of the 2700
- Detailed description of the problem you are experiencing
- Error messages (if any)
- Device manufacturer and part number (if device related)
- PC information (such as model, manufacturer, speed) and operating system version (such as Windows 3.11)

United States

For technical assistance, contact:

Data I/O Customer Resource Center
Telephone: 800-247-5700 (*Press 2 on your touch-tone telephone to bypass the recorded message and speak to the first available Support Engineer*)
Fax: 206-869-2821

For repair or warranty service, contact:

Data I/O Central Dispatch
Telephone: 800-735-6070
Fax: 206-881-0561

For Keep Current subscription service, contact:

Data I/O Sales
Telephone: 800-332-8246

Canada

For technical assistance, repair, warranty service, or Keep Current subscription service, contact:

Data I/O Canada
6725 Airport Road, Suite 102
Mississauga, Ontario, L4V 1V2
Telephone: 905-678-0761
Fax: 905-678-7306

United Kingdom

For technical assistance, repair, warranty service, or Keep Current subscription service, contact:

Data I/O Limited
4 The Business Centre
Molly Millars Lane
Wokingham
Berkshire RG41 2QZ
Telephone: (44) (0) 1189 361200
Fax: (44) (0) 1189 361211

Japan

For technical assistance, repair, warranty service, or Keep Current subscription service, contact:

Data I/O Japan
Osaki CN Building 2F
5-10-10 Osaki
Shinagawa-Ku
Tokyo 141
Telephone: 3-3779-2152
Fax: 3-3779-2203

Germany

For technical assistance, repair, warranty service, or Keep Current subscription service, contact:

Data I/O GmbH
Lochhamer Schlag 5a
82166 Gräfelfing
Telephone: 089-858580
Fax: 089-8585810

Other Countries

For technical assistance, repair, warranty service, or Keep Current subscription service, contact the distributor from whom you purchased your 2700. Refer to the Customer Support information sheet to determine whom to contact.

If you have troubleshooting questions or other operation difficulties, your distributor can answer your questions and direct you to the appropriate Service Center, if necessary.

Keep Current Subscription Service

Data I/O offers a 1-year renewable subscription to keep your product and documentation up to date with the latest features and device support. This subscription, called the Keep Current subscription service, also incorporates changes to existing device support recommended by the manufacturer to maintain optimum yields, throughput, and long-term reliability.

For more information, or to order Keep Current subscription service, contact Data I/O Customer Support.

Update Information

Updating assures that you have the most current device algorithms. You need to update when you want to:

- Use a new version of 2700 system software
- Add support for a specific device
- Use the latest version of a particular device algorithm

Updates are made available several times each year. To ensure that you are notified when updates are available, complete and return your Registration Card.

Warranty Information

Data I/O Corporation warrants this product against defects in materials and workmanship at the time of delivery and thereafter for a period of one (1) year.

The foregoing warranty and the manufacturers' warranties, if any, are in lieu of all other warranties, expressed, implied or arising under law, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.

Data I/O maintains customer service offices throughout the world, each staffed with factory-trained technicians to provide prompt, quality service. For warranty service contact your distributor listed in the Customer Support Information Sheet.

Repair Service

After the warranty period expires, repair or replacement services are available through your distributor or at authorized Data I/O Service Centers.

Appendix B: Translation Formats

Translation formats are different ways of encoding the data in a data file. A data file contains the information to be programmed into a device. The data file could contain the fuse pattern and test vectors for a logic device or the data for a memory device.

Generally, the data, such as the fuse pattern for a logic device, are created on a development platform and are then stored in a particular data translation format. When you want to transfer the data file to the programmer, you will need to set up the programmer to handle the correct translation format. During download, the programmer translates the formatted data and stores them in user memory as a binary image file.

Below you will find a list, in ascending numerical order, of all the translation formats supported by the programmer. Following the list is a description and, in most cases, an example of each translation format, presented in order by format number.

Format	**Code**
Formatted Binary	10
POF	14
Absolute Binary	16
LOF	17
ASCII-Hex Space	50 (55*)
ASCII-Hex Percent	51 (56*)
ASCII-Hex Apostrophe	52
ASCII-Hex Comma	53 (58*)
ASCII-Hex SMS	57
Motorola EXORcisor	82
Intel Intellec 8/MDS	83
Motorola EXORmax	87
Intel MCS-86 Hex Object	88
JEDEC format (full)	91
JEDEC format (Kernal)	92
Motorola 32 bit (S3 record)	95
Intel Hex-32	99

* *This alternate code is used to transfer data using the SOH start code instead of the usual STX.*

General Notes

Some information about data translation is listed below:

Compatibility

When translating data, you may use any remote source that produces formats compatible with the descriptions listed in this section.

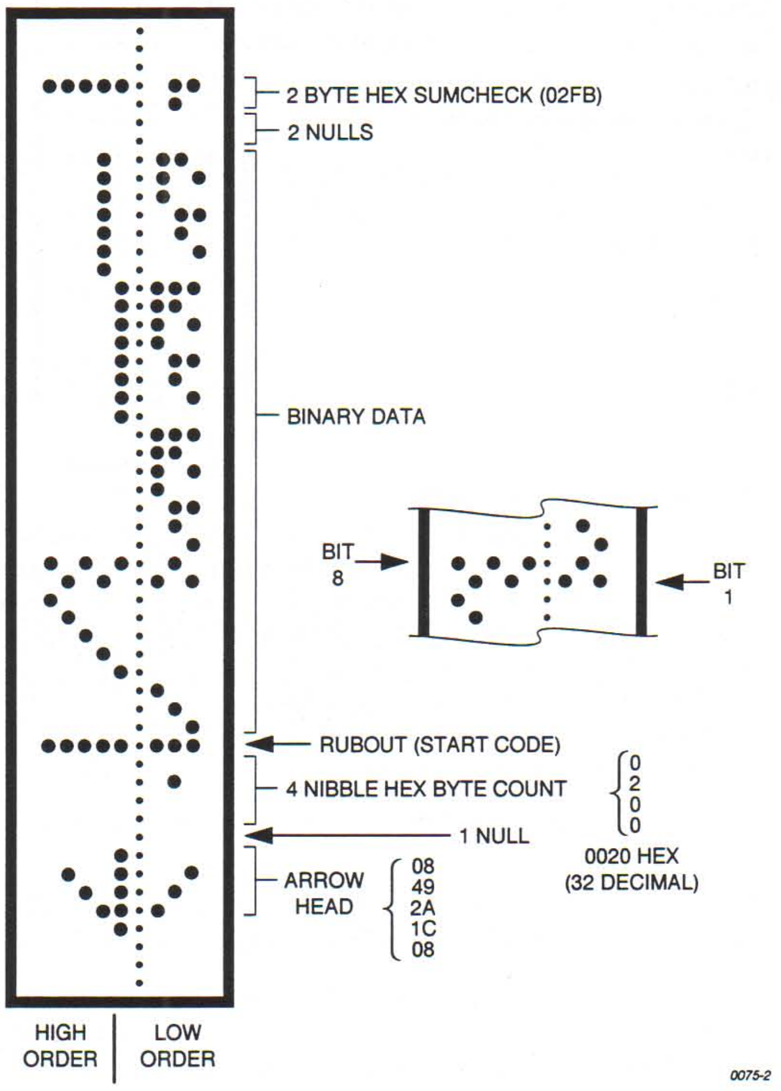
Formats with Limited Address Fields

Some formats are not defined for use with address fields greater than 64K. Thus, if you transfer a block greater than 64K, the address fields that would be greater than 64K may wrap around and overwrite data transferred in previous data records. Formats 70 through 86, and 90 may exhibit this characteristic.

Formatted Binary Format, Code 10

Data transfer in the Formatted Binary format consists of a stream of 8-bit data bytes preceded by a byte count and followed by a sumcheck, as shown in Figure B-1. The Formatted Binary format does not have addresses.

Figure B-1
An Example of Formatted Binary Format

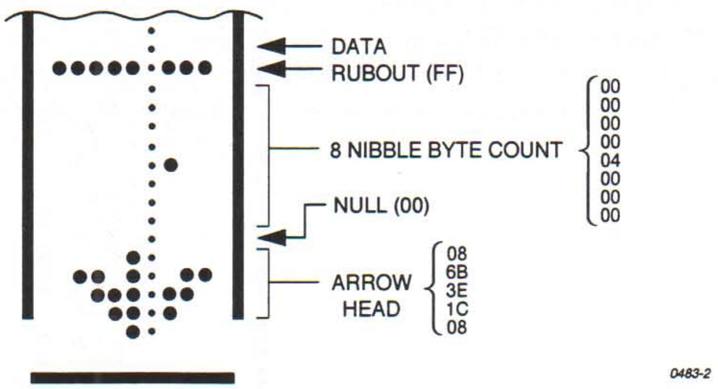


The programmer stores incoming binary data upon receipt of the start character. Data are stored in RAM starting at the first RAM address specified by the Memory Begin Address parameter and ending at the last incoming data byte.

A paper tape generated by a programmer contains a 5-byte, arrow-shaped header followed by a null and a 4-nibble byte count. The start code, an 8-bit rubout, follows the byte count. The end of data is signaled by two nulls and a 2-byte sumcheck of the data field. Refer to Figure B-2.

If the data output has a byte count GREATER than or equal to 64K, an alternate arrow-shaped header is used. This alternate header (shown below) is followed by an 8-nibble byte count, sandwiched between a null and a rubout. The byte count shown here is 40000H (256K decimal). If the byte count is LESS than 64K, the regular arrowhead is used instead. Data that are input using Formatted Binary format will accept either version of this format.

Figure B-2
An Example of Formatted Binary Format



In addition, a third variation of this binary format is accepted on download. This variation does not have an arrowhead and is accepted only on input. The rubout begins the format and is immediately followed by the data. There is no byte count or sumcheck.

POF (Programmer Object File) Format, Code 14

The POF (Programmer Object File) format provides a highly compact data format to enable translation of high bit count logic devices efficiently. This format currently applies to MAX™ devices, such as the Altera 5032.

The information contained in the file is grouped into "packets." Each packet contains a "tag," identifying what sort of data the package contains plus the data itself. This system of packeting information allows for future definitions as required.

The POF is composed of a header and a list of packets. The packets have variable lengths and structures, but the first six bytes of every packet always adhere to the following structure.

```
struct PACKET_HEAD
{
short tag;          /*tag number - type of packet */
long length;        /*number of bytes in rest of packet
*/
}
```

A POF is read by the program examining each packet and if the tag value is recognized, then the packet is used. If a tag value is not recognized, the packet is ignored.

Any packet except the terminator packet may appear multiple times within a POF. Packets do not need to occur in numerical tag sequence. The POF reader software is responsible for the interpretation and action taken as a result of any redundant data in the file, including the detection of error conditions.

The POF format currently uses the following packet types.

Note: In the following packet type descriptions, one of the terms — Used, Skipped, or Read — will appear after the tag and name.

***Used**: The information in this packet is used by the programmer.*
***Skipped**: This information is not used by the programmer.*
***Read**: This information is read by the programmer but has no direct application.*

Creator_ID

tag=1 Used

This packet contains a version ID string from the program which created the POF.

Device_Name

tag=2 Used

This packet contains the ASCII name of the target device to be programmed, for example, PM9129.

Comment_Text	tag=3	Read	This packet contains a text string which may consist of comments related to the POF. This text may be displayed to the operator when the file is read. The string may include multiple lines of text, separated by appropriate new line characters.
Tag_Reserved	tag=4	Skipped	
Security_Bit	tag=5	Used	This packet declares whether security mode should be enabled on the target device.
Logical_Address_and _Data_16	tag=6	Read	This packet defines a group of logical addresses in the target device and associates logical data with these addresses. The addresses comprise a linear region in the logical address space, bounded on the low end by the starting address and extending upward by the address count specified in the packet.
Electrical_Address _and_Data	tag=7	Used	This packet defines a group of electrical addresses in the target device and associates data values with those addresses. The data field is ordered in column-row order, beginning with the data for the least column-row address, continuing with increasing row addresses until the first column is filled, then incrementing the column address, etc.
Terminator	tag=8	Used	This packet signals the end of the packet list in the POF. This packet must be the Nth packet, where N is the packet count declared in the POF header. The CRC field is a 16-bit Cyclic Redundancy Check computed on all bytes in the file up to, but not including, the CRC value itself. If this CRC value is zero, the CRC check should be ignored.
Symbol table	tag=9	Skipped	
Test Vectors	tag=10	Used	This packet allows the POF to contain test vectors for post programming testing purposes. Each vector is a character string and uses the 20 character codes for vector bits defined in JEDEC standard 3A, section 7.0.
Electrical_Address_and _Constant_data	tag=12	Skipped	

Number of programmable elements

tag=14 Read

This packet defines the number of programmable elements in the target device.

Logical_Address_and_Data_32

tag=17 Read

This packet defines a group of logical addresses in the target device and associates logical data with these addresses. The addresses comprise a linear region in the logical address space, bounded on the low end by the starting address and extending upward by the address count specified in the packet.
The starting address and address count are each specified by 4-byte fields (32 bits).

Absolute Binary Format, Code 16

Absolute Binary format is a literal representation of the data to be transferred and no translation of the data takes place during the transfer. There are no overhead characters added to the data (i.e. no address record, start code, end code, nulls, or checksum). Every byte transferred represents the user's data. This format can be used to download unformatted data such as an ".exe" file to the programmer.

Since this format does not have an end of file character, download transfers will terminate after no more data are received and an I/O timeout occurs. This is true for all data formats which don't have an end of file indicator. For this reason do not use a value of 0 for the I/O timeout parameter on the communication parameters screen, since this will disable the timeout from occurring. A value between 1 and 99 (inclusive) should be used for the I/O timeout parameter when using formats which require the timeout to occur.

LOF Format, Code 17

The Link Object Format (LOF) is an extension of the standard JEDEC data translation format and is used to transfer fuse and test vector data between the programmer and a host computer. LOF is designed to support the Quicklogic QL8x12A family of FPGAs. An LOF data file is stored as an imploded ZIP file, which yields data compression approaching 95%.

Note: The specification for the ZIP data compression algorithm allows for multiple data files to be compressed into one ZIP file. In addition, the ZIP data compression algorithm allows for multiple types of data compression.

The programmer's implementation of UNZIP supports only imploded data files and will extract only the first file in a ZIP file. All remaining files in the ZIP file will be ignored, as will all files not stored in the imploded format.

The LOF format contains both a subset and a superset of the JEDEC format described in this chapter. This section describes only the fields that are extensions of the JEDEC standard or that are unique to the LOF format. See the section explaining the JEDEC format for information on the standard JEDEC fields. See page 134 for information on obtaining a copy of the JEDEC Standard 3A.

LOF Field Syntax

The LOF character set consists of all the characters that are permitted with the JEDEC format: all printable ASCII characters and four control characters. The four allowable control characters are STX, ETX, CR (Return), and LF (line feed). Other control characters, such as Esc or Break, should not be used.

Note: This is Data I/O Corporation's implementation of Quicklogic's Link Object Format. Contact Quicklogic for a more in-depth explanation of the format and its syntax.

LOF Fields

The following fields are included in Data I/O's implementation of the LOF format:

<STX>	*	Start of Data (ASCII Ctrl-B, 0x02 hex)
C	*	Fuse Checksum
K		Fuse data, followed by control words and pulse link cycles

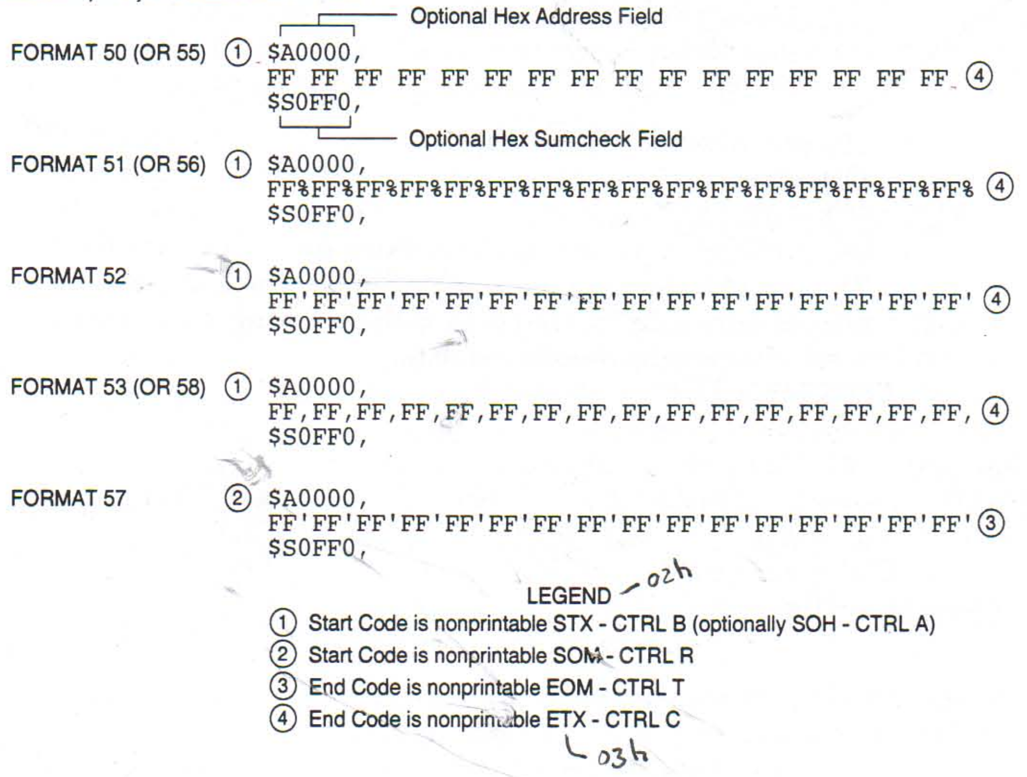
N	*	Notes Field
QB		Number of bits per word
QC		Number of control words at the end of each K field
QF		Number of Fuses in Device (# of K fields)
QM		Number of macro cells in the data file
QP	*	Number of Device Package Pins
QS		Number of Hex-ASCII words in each K field and each control word
QV	*	Maximum Number of Test Vectors
R		Signature Analysis (reserved for future use)
S		SpDE Checksum
T		Signature Analysis (reserved for future use)
V	*	Test Vectors (reserved for future use)
X	*	Default Test Conditions (reserved for future use)
<ETX>	*	End of Data (ASCII Ctrl-C, 0x03 hex)

* *These fields are already defined as part of the JEDEC standard and will not be defined in this section.*

ASCII Hex Formats, Codes 50-58

Each of these formats has a start and end code, and similar address and checksum specifications. Figure B-3 illustrates 4 data bytes coded in each of the 9 ASCII Hexadecimal formats. Data in these formats are organized into sequential bytes separated by the execute character (space, percent, apostrophe, or comma). Characters immediately preceding the execute character are interpreted as data. ASCII Hex formats can express 8-bit data by 2 hexadecimal characters. Line feeds, carriage returns, and other characters may be included in the data stream as long as a data byte directly precedes each execute character.

Figure B-3
An Example of ASCII Hex Formats



```
                           Optional Hex Address Field
FORMAT 50 (OR 55)  ①  $A0000,
                      FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ④
                      $S0FF0,
                           Optional Hex Sumcheck Field
FORMAT 51 (OR 56)  ①  $A0000,
                      FF%FF%FF%FF%FF%FF%FF%FF%FF%FF%FF%FF%FF%FF%FF%FF% ④
                      $S0FF0,

FORMAT 52          ①  $A0000,
                      FF'FF'FF'FF'FF'FF'FF'FF'FF'FF'FF'FF'FF'FF'FF'FF' ④
                      $S0FF0,

FORMAT 53 (OR 58)  ①  $A0000,
                      FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF, ④
                      $S0FF0,

FORMAT 57          ②  $A0000,
                      FF'FF'FF'FF'FF'FF'FF'FF'FF'FF'FF'FF'FF'FF'FF'FF' ③
                      $S0FF0,
```

LEGEND – 02h

① Start Code is nonprintable STX - CTRL B (optionally SOH - CTRL A)
② Start Code is nonprintable SOM - CTRL R
③ End Code is nonprintable EOM - CTRL T
④ End Code is nonprintable ETX - CTRL C

└ 03h

2614-1

Although each data byte has an address, most are implied. Data bytes are addressed sequentially unless an explicit address is included in the data stream. This address is preceded by a $ and an A, must contain 2 to 8 hex characters, and must be followed by a comma, except for the ASCII-Hex (Comma) format, which uses a period. The programmer skips to the new address to store the next data byte; succeeding bytes are again stored sequentially.

Each format has an end code, which terminates input operations. However, if a new start code follows within 16 characters of an end code, input will continue uninterrupted. If no characters come within 2 seconds, input operation is terminated.

After receiving the final end code following an input operation, the programmer calculates a sumcheck of all incoming data. Optionally, a sumcheck can also be entered in the input data stream. The programmer compares this sumcheck with its own calculated sumcheck. If they match, the programmer will display the sumcheck; if not, a sumcheck error will be displayed.

Note: The sumcheck field consists of 2-4 hex characters sandwiched between the $ and comma characters. The sumcheck immediately follows an end code. The sumcheck is optional in the input mode but is always included in the output mode.

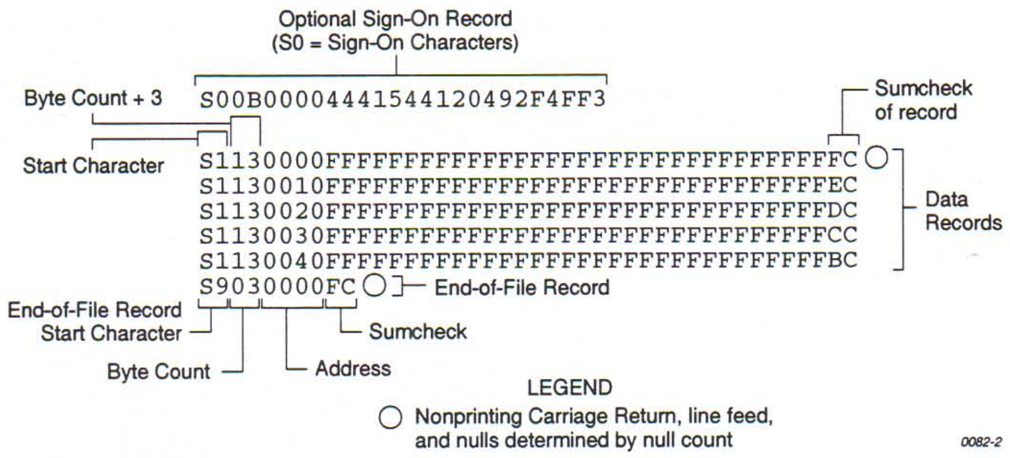
The programmer divides the output data into 8-line blocks. Data transmission is begun with the start code, a nonprintable STX character, or optionally, SOH.* Data blocks follow, each one prefaced by an address for the first data byte in the block. The end of transmission is signaled by the end code, a nonprintable ETX character. Directly following the end code is a sumcheck of the transferred data.

* *ASCII-Hex SMS uses SOM (CTRL-R) as a start code and EOM (CTRL-T) as an end code.*

Motorola EXORciser Format, Code 82

Motorola EXORciser data files may begin with an optional sign-on record, which is initiated by the start characters S0. Valid data records start with an 8-character prefix and end with a 2-character suffix. Figure B-4 shows a series of valid Motorola data records.

Figure B-4
An Example of Motorola EXORciser Format



```
S00B00004441544120492F4FF3
S1130000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC ○
S1130010FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEC
S1130020FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDC
S1130030FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFCC
S1130040FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBC
S9030000FC ○
```

Each data record begins with the start characters S1. The third and fourth characters represent the byte count, which expresses the number of data, address, and checksum bytes in the record. The address of the first data byte in the record is expressed by the last 4 characters of the prefix. Data bytes follow, each represented by 2 hexadecimal characters. The number of data bytes occurring must be three less than the byte count. The suffix is a 2-character checksum, which equals the one's complement of the binary summation of the byte count, address, and data bytes.

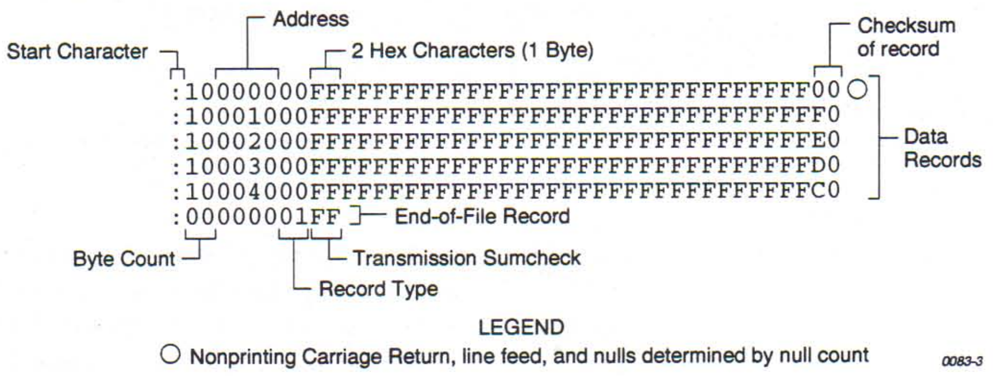
The end-of-file record consists of the start characters S9, the byte count, the address (in hex), and a checksum. The maximum record length is 250 data bytes.

Intel Intellec 8/MDS Format, Code 83

Intel data records begin with a 9-character prefix and end with a 2-character suffix. The byte count must equal the number of data bytes in the record.

Figure B-5 simulates a series of valid data records. Each record begins with a colon, which is followed by a 2-character byte count. The 4 digits following the byte count give the address of the first data byte. Each data byte is represented by 2 hexadecimal digits; the number of data bytes in each record must equal the byte count. Following the data bytes of each record is the checksum, the two's complement (in binary) of the preceding bytes (including the byte count, address, record type, and data bytes), expressed in hex.

Figure B-5
An Example of Intel Intellec 8/MDS Format



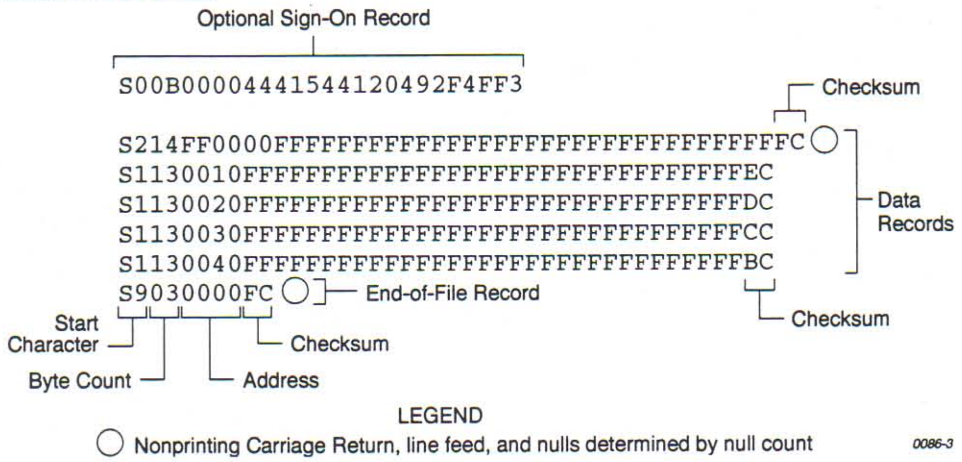
```
:10000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00
:10001000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0
:10002000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE0
:10003000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD0
:10004000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC0
:00000001FF
```

The end-of-file record consists of the colon start character, the byte count (equal to 00), the address, the record type (equal to 01), and the checksum of the record.

Motorola EXORmax Format, Code 87

Motorola data files may begin with an optional sign-on record, initiated by the start characters S0. Data records start with an 8- or 10-character prefix and end with a 2-character suffix. Figure B-6 shows a series of Motorola EXORmacs data records.

Figure B-6
An Example of Motorola EXORmacs Format



```
S00B00004441544120492F4FF3

S214FF0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC
S1130010FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEC
S1130020FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDC
S1130030FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFCC
S1130040FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBC
S9030000FC
```

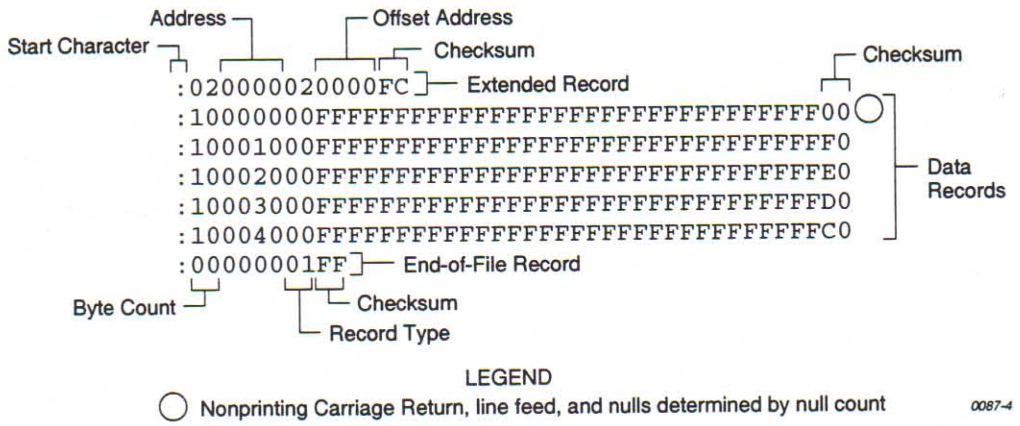
Each data record begins with the start characters S1 or S2: S1 if the following address field has 4 characters, S2 if it has 6 characters. The third and fourth characters represent the byte count, which expresses the number of data, address, and checksum bytes in the record. The address of the first data byte in the record is expressed by the last 4 characters of the prefix (6 characters for addresses above hexadecimal FFFF). Data bytes follow, each represented by 2 hexadecimal characters. The number of data bytes occurring must be 3 or 4 less than the byte count. The suffix is a 2-character checksum, the one's complement (in binary) of the preceding bytes in the record, including the byte count, address, and data bytes.

The end-of-file record begins with an S9 start character. Following the start characters are the byte count, the address, and a checksum. The maximum record length is 250 data bytes.

Intel MCS-86 Hexadecimal Object, Code 88

The Intel 16-bit Hexadecimal Object file record format has a 9-character (4-field) prefix that defines the start of record, byte count, load address, and record type and a 2-character checksum suffix. Figure B-7 shows a sample record of this format.

Figure B-7
An Example of Intel MCS-86 Hex Object



```
:020000020000FC
:10000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00
:10001000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0
:10002000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE0
:10003000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD0
:10004000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC0
:00000001FF
```

The four record types are described below.

00-Data Record

This begins with the colon start character, which is followed by the byte count (in hex notation), the address of the first data byte, and the record type (equal to 00). Following these are the data bytes. The checksum follows the data bytes and is the two's complement (in binary) of the preceding bytes in the record, including the byte count, address, record type, and data bytes.

01-End Record

This end-of-file record also begins with the colon start character. This is followed by the byte count (equal to 00), the address (equal to 0000), the record type (equal to 01), and the checksum, FF.

02-Extended Segment Address Record

This is added to the offset to determine the absolute destination address. The address field for this record must contain ASCII zeros (Hex 30s). This record type defines bits 4 to 19 of the segment base address. It can appear randomly anywhere within the object file and affects the absolute memory address of subsequent data records in the file. The following example illustrates how the extended segment address is used to determine a byte address.

Problem:

Find the address for the first data byte for the following file.

```
:02 0000 02 1230 BA
:10 0045 00 55AA FF.....BC
```

Solution:

Step 1. Find the record address for the byte. The first data byte is 55. Its record address is 0045 from above.

Step 2. Find the offset address. The offset address is 1230 from above.

Step 3. Shift the offset address one place left, then add it to the record address, like this:

```
    1230              Offset address (upper 16 bits)
+    0045             Record address (lower 16 bits)
    12345             20-bit address
```

The address for the first data byte is 12345.

Note: Always specify the address offset when using this format, even when the offset is zero.

During output translation, the firmware will force the record size to 16 (decimal) if the record size is specified greater than 16. There is no such limitation for record sizes specified less than 16.

03-Start Record

This record type is not sent during output by Data I/O translator firmware.

JEDEC Format, Codes 91 and 92

Introduction

The JEDEC (Joint Electron Device Engineering Council) format is used to transfer fuse and test vector data between the programmer and a host computer. Code 91 is full format and includes all the data fields (such as note and test fields) described on the following pages. Code 92 is the Kernel, or shorter, format. The JEDEC Kernel format includes only the minimum information needed for the programming; it does not, for example, include information fields or test vector fields. Prior to transferring a JEDEC file, the appropriate Logic device must be selected.

JEDEC's legal character set consists of all the printable ASCII characters and four control characters. The four allowable control characters are STX, ETX, CR (RETURN), and LF (line feed). Other control characters, such as ESC or BREAK, should not be used.

Note: This is Data I/O Corporation's implementation of JEDEC Standard 3A. For a copy of the strict standard, write to:

Electronic Industries Association
Engineering Department
2001 Eye Street NW
Washington, D.C. 20006

BNF Rules and Standard Definitions

The Backus-Naur Form (BNF) is used in the description here to define the syntax of the JEDEC format. BNF is a shorthand notation that follows these rules:

:: = denotes "is defined as."

Characters enclosed by single quotes are literals (required).

Angle brackets enclose identifiers.

Square brackets enclose optional items.

Braces {} enclose a repeated item. The item may appear zero or more times.

Vertical bars indicate a choice between items.

Repeat counts are given by a :n suffix. For example, a 6-digit number would be defined as:

<number< :: = <digit>:6

For example, in words the definition of a person's name reads:

The full name consists of an optional title followed by a first name, a middle name, and a last name. The person may not have a middle name, or may have several middle names. The titles consist of: Mr., Mrs., Ms., Miss, and Dr.

The BNF definition for a person's name is:

<full name> :: = [<title>] <f. name> {<m.name>} <l. name>

<title> :: = 'Mr.' | 'Mrs.' | 'Ms.' | 'Miss' | 'Dr.'

The following standard definitions are used throughout the rest of this document:

<digit> :: = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

<hex-digit> :: = <digit> | 'A' | 'B' | 'C' | 'D' | 'E' | 'F'

<binary-digit> :: = '0' | '1'

<number> :: = <digit> {<digit>}

<del> :: = <space> | <carriage return>

<delimiter> :: = <del> {<del>}

<printable character> :: = <ASCII 20 hex ... 7E hex>

<control character> :: = <ASCII 00 hex ... 1F hex> | <ASCII 7F hex>

<STX> :: = <ASCII 02 hex>

<ETX> :: = <ASCII 03 hex>

<carriage return> :: = <ASCII 0D hex>

<line feed> :: = <ASCII 0A hex>

<space> :: = <ASCII 20 hex> | "

<valid character> :: = <printable character> | <carriage return> | <line feed>

<field character> :: = <ASCII 20 hex ... 29 hex> | <ASCII 2B hex ... 7E hex> | <carriage return> | <line feed>

The Design Specification Field

<design spec> ::= {<field character>}'*'

The first field sent in a JEDEC transmission is the design specification. Both the full and kernel JEDEC formats accept the design specification field. This field is mandatory and does not have an identifier (such as an asterisk) signaling its beginning. The design specification field consists of general device information. It could, for example, consist of the following information: your name, your company's name, the date, the device name and manufacturer, design revision level, etc. This field is terminated by an asterisk character. Examine the sample transmission shown on the next page of this description—the first three lines of the file comprise the design specification field. The programmer ignores the contents of this field for downloads and places "Data I/O" in this field for upload operations.

Note: *You do not need to send any information in this field if you do not wish to; a blank field, consisting of the terminating asterisk, is a valid design specification field.*

The Transmission Checksum Field

```
<xmit checksum> ::= <hex digit>:4
```

The transmission checksum is the last value sent in a JEDEC transmission. The full JEDEC format requires the transmission checksum. The checksum is a 16-bit value, sent as a 4-digit hex number, and is the sum of all the ASCII characters transmitted between (and including) the STX and ETX. The parity bit is excluded in the calculation of the transmission checksum.

Some computer systems do not allow you to control what characters are sent, especially at the end of a line. You should set up the equipment so that it will accept a dummy value of 0000 as a valid checksum. This zero checksum is a way of disabling the transmission checksum while still keeping within the JEDEC format rules.

JEDEC Full Format, Code 91

The full JEDEC format consists of a start-of-text character (STX), various fields, an end-of-text character (ETX), and a transmission checksum. A sample JEDEC transmission sent in the full format is shown in Figure B-8. Each of the fields is described on the following pages.

Figure B-8
An Example of JEDEC Full Format

```
ABEL(tm) Version 2.00b   JEDEC file for:P20R8
  Large Memory Version
Created on: 09-Mar-87 04:45 PM
8-bit barrel shifter
EngineerI        Data I/O Corp  Redmond WA   10 Jan 1986*
QP24* QF2560*
L0000
1101111111111111111111111111111101110111010
1101111111111111111111111101111111011100 1
1101111111111111111101111111111110110110
1101111111111111101111111111111110110101
1101111111111101111111111111111101111010
1101111110111111111111111111111101111001
1001101111111111111111111111111101110110
1001111111111111111111111111111101110101
1001111111111111111111111111110110111010 1
1101111111111111111111111111110111011101 0
1101111111111111111111111011111110111001
11011111111111111111101[illegible]0110110
110111111111111111011[illegible]01
  11111111111011[illegible]111111110
  [illegible]1111111111110111
100[illegible]11111111111111110111011
1001111111111111111111111111111101110101
100111111111111111111111111110110111010 1*
V0001 C1000000000N00HLLLLLLL1N*
V0002 C1000000000N01LHLLLLLL1N*
V0003 C1000000001N00LLHLLLLL1N*
V0004 C1000000001N01LLLHLLLL1N*
V0005 C1000000010N00LLLLHLLL1N*
V0006 C1000000010N01LLLLLHLL1N*
V0007 C1000000011N00LLLLLLHL1N*
V0008 C1000000011N01LLLLLLLH1N*
V0009 C0111111100N00LHHHHHHH1N*
V0010 C0111111100N01HLHHHHHH1N*
V0011 C0111111101N00HHLHHHHH1N*
V0012 C0111111101N01HHHLHHHH1N*
V0013 C0111111110N00HHHHLHHH1N*
V0014 C0111111110N01HHHHHLHH1N*
V0015 C0111111111N00HHHHHHLH1N*
V0016 C0111111111N01HHHHHHHL1N*
V0017 C0000000100N01HLLLLLLL1N*
V0018 C1111111000N01LHHHHHHH1N*
V0019 C0000000000N00HHHHHHHH0N*
V0020 C0000000000N10ZZZZZZZZ1N*
C1B20*
B8C0
```

Header (comment area - everything preceeding first * is ignored)

Number of Pins (24) and Number of Fuses (2560)

Fuse Address (0000)

Fuse States:
0 = intact
1 = blown

Vector Number

Test Vectors

Fuse Map Checksum

Transmission Checksum

0090-

JEDEC Field Syntax

```
<field> ::= [<delimiter>]<field identifier>{<field
character>}'*'

<field identifier>::= 'A' | 'C' | 'D' | 'F' | 'G' |
'K' | 'L' | 'N' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'V'
| 'X'

<reserved identifier>::= 'B' | 'E' | 'H' | 'I' | 'J'
| 'M' | 'O' | 'U' | 'W' | 'Y' | 'Z'
```

Following the design specification field in a JEDEC transmission can be any number of information fields. Each of the JEDEC fields begins with a character that identifies what type of field it is. Fields are terminated with an asterisk character. Multiple character identifiers can be used to create sub-fields (i.e., A1, A$, or AB3). Although they are not required, you may use carriage returns (CR) and line feeds (LF) to improve readability of the data.

Field Identifiers

Field identifiers which are currently used in JEDEC transmissions are shown above on the "field identifiers" line. The "reserved identifier" line indicates characters not currently used (reserved for future use as field identifiers). JEDEC field identifiers are defined as follows:

A	Access time	N	Note field
B	*	O	*
C	Checksum field	P	Pin sequence
D	Device type	Q	Value field
E	*	R	Resulting vector field
F	Default fuse state field	S	Starting vector
G	Security fuse field	T	Test cycles
H	*	U	*
I	*	V	Test vector field
J	*	W	*
K	Fuse list field (hex format)	X	Default test condition
L	Fuse list field	Y	*
M	*	Z	*

* *Reserved for future use.*

Device Field (D)

Device selection by this field is not supported by the programmer. It has been replaced by the QF and QP fields and manual selection of devices.

Fuse Information Fields (L, K, F, C)

```
<fuse information> :: = [<default state>] <fuse list>
{<fuse list>} [<fuse checksum>]

<fuse list> : = 'L' <number> <delimiter> {<binary-
digit> [<delimiter>]} ' * '

<fuse list> :: = 'K' <number> <delimiter> {<hex-
digit> [<delimiter>]} '*'

<default state> :: = 'F' <binary-digit> '* '

<fuse checksum> :: = 'C' <hex-digit>:4 '* '
```

Each fuse of a device is assigned a decimal number and has two possible states: zero, specifying a low-resistance link, or one, specifying a high resistance link. The state of each fuse in the device is given by three fields: the fuse list (L field or K field), the default state (F field), and the fuse checksum (C field).

Fuse states are explicitly defined by either the L field or the K field. The character L begins the L field and is followed by the decimal number of the first fuse for which this field defines a state. The first fuse number is followed by a list of binary values indicating the fuse states.

The information in the K field is the same as that of the L field except that the information is represented by hex characters instead of binary values. This allows more compact representation of the fusemap data. The character K begins the K field and is followed by the decimal number of the first fuse. The fuse data follow the fuse number and are represented by hex characters. Each bit of each hex character represents the state of one fuse, so each hex character represents four fuses. The most significant bit of the first hex character following the fuse number corresponds to the state of that fuse number. The next most significant bit corresponds to the state of the next fuse number, etc. The least significant bit of the first hex character corresponds to the state of the fuse at the location specified by the fuse number plus three.

The K field supports download operations only. The K field is not part of the JEDEC standard, but is supported by Data I/O for fast data transfer. The L and K fields can be any length desired, and any number of L or K fields can be specified. If the state of a fuse is specified more than once, the last state specified replaces all previous ones for that fuse. The F field defines the states of fuses that are not explicitly defined in the L or K fields. If no F field is specified, all fuse states must be defined by L or K fields.

The C field, the fuse information checksum field, is used to detect transmitting and receiving errors. The field contains a 16-bit sum (modulus 65535) computed by adding 8-bit words containing the fuse states for the entire device. The 8-bit words are formed as shown in the following figure. Unused bits in the final 8-bit word are set to zero before the checksum is calculated.

Word 00 Fuse No.	msb 7	6	5	4	3	2	1	lsb 0
Word 01 Fuse No.	msb 15	14	13	12	11	10	9	lsb 8
Word 62 Fuse No.	msb 503	-	-	-	499	498	497	lsb 496

Following is an example of full specification of the L, C, and F fields:

```
F0*L0 01010101* L0008 01010111* L1000 0101*C019E*
```

Following is an alternate way of defining the same fuse states using the K field:

```
F0*K0 55* K0008 57* K1000 5* C019E*
```

Another example, where F and C are not specified:

```
L0200     01101010101010101011

0101110101101000100100100l0*
```

The Security Fuse Field (G)

```
<security fuse>::='G'<binary-digit>'*'
```

The JEDEC G field is used to enable the security fuse of some logic devices. To enable the fuse, send a 1 in the G field:

```
G1*
```

The Note Field (N)

```
<note>::='N'<field characters>'*'
```

The note field is used in JEDEC transmission to insert notes or comments. The programmer will ignore this field; it will not be interpreted as data. An example of a note field would be:

```
N Test Preload*
```

The Value Fields (QF, QP, and QV)

JEDEC value fields define values or limits for the data file, such as number of fuses. The QF subfield defines the number of fuses in the device. All of the value fields must occur before any device programming or testing fields appear in the data file. Files with ONLY testing fields do not require the QF field, and fields with ONLY programming data do not require the QP and QV fields.

The QF subfield tells the programmer how much memory to reserve for fuse data, the number of fuses to set to the default condition, and the number of fuses to include in the fuse checksum. The QP subfield defines the number of pins or test conditions in the test vector, and the QV subfield defines the maximum number of test vectors.

The P Field

The P field remaps the device pinout and is used with the V (test vector) field. An asterisk terminates the field. The syntax of the field is as follows:

```
<pin list>::='P'<pin number>:N'*'

<pin number>::=<delimiter><number>
```

The following example shows a P field, V field, and the resulting application:

```
P 1 2 3 4 5 6 14 15 16 17 7 8 9 10 11 12 13 18 19 20
*

V0001 111000HLHHNNNNNNNNNN*

V0002 100000HHHLNNNNNNNNNN*
```

The result of applying the above P and V fields is that vector 1 will apply 111000 to pins 1 through 6, and HLHH to pins 14 through 17. Pins 7 through 13 and 18 through 20 will not be tested.

JEDEC U and E Fields

As of Version 5.2, the programmer supports the optional JEDEC U (user data) and E (electrical data) fields. The U and E fields are described below.

Note: Implementation of the JEDEC U and E fields is not part of the JEDEC-3C (JESD3-C) standard.

User Data (U Field)

The **U** field allows user data fuses that do not affect the logical or electrical functionality of the device to be specified in JEDEC files. For instance, the U field can be used to specify the User Data Signature fuse available in some types of PLD devices because this fuse contains information only (it has no logical or electrical functionality).

Note: To have the JEDEC U field processed correctly, you must select the device before downloading the JEDEC file.

The following guidelines apply to the U field:

- The U field must be included for devices with U fuses.
- Each U-field cell must be explicitly provided if the U field is present.
- The F (default fuse state) field does not affect U fuses.
- There can only be one U field in a JEDEC file.
- The U field fuses must be listed in the order they appear in the device.
- The U field must be listed after the L field and E field (if used), and before the V (test vector) field (if used).
- The U field is specified using binary numbers, since the full number of U-field cells is otherwise unknown.
- The number of cells specified in the U field is not included in the QF (number of fuses) field.
- The U-field cells are not included in the C (fuse checksum) field.
- The U field reads left to right to be consistent with the L (fuse list) and E fields.

The syntax for the U field is as follows:

```
<User Data Fuse List>::'U'<binary-digit(s)>'*'
```

The character U begins the U field and is followed by one binary digit for each U fuse. Each binary digit indicates one of two possible states (zero, specifying a low-resistance link, or one, specifying a high-resistance link) for each fuse.

For example,

```
QF24*
L0000
101011000000000000000000*
E10100111*
C011A*
U10110110*
```

Electrical Data (E field)

The E field allows special feature fuses that do not affect the logic function of the device to be specified in JEDEC files.

The following guidelines apply to the E field:

- The E-field cell must be explicitly provided if the E field is present.
- The F (default fuse state) field does not affect E fuses.
- There can only be one E field in a JEDEC file.
- The E field fuses must be listed in the order they appear in the device.
- The E field must be listed before the C (checksum) field. If the U field is used, the E field must come before the U (user data) field.
- The E field is specified using binary numbers, since the full number of E-field cells is otherwise unknown.
- The number of cells specified in the E field is not included in the QF (number of fuses) field.
- The E-field cells are included in the C (fuse checksum) field.
- The E field reads left to right for the purpose of checksum calculation.

The syntax for the E field is as follows:

```
<Electrical Data Fuse List>::'E'<binary digit(s)>'*'
```

The character E begins the E field and is followed by one binary digit for each E fuse. Each binary digit indicates one of two possible states (zero, specifying a low-resistance link, or one, specifying a high-resistance link) for each fuse. For example,

```
QF24*
L0000
101011000000000000000000*
E10100111*
C011A*
U10110110*
```

Test Field (V field)

```
<function test> :: = [<pin list>] <test vector> {<test
vector>}

<pin number> :: = <delimiter> <number>

N :: = number of pins on device

<test vector> :: = 'V' <number> <delimiter> < test
condition> :N '* '

<test condition> :: = <digit> 'B' | 'C' | 'D' | 'F' |
   'H' | 'K' | 'L' | 'N' | 'P' | 'U' | 'X' | 'Z'

<reserved condition> :: = 'A' | 'E' | 'G' | 'I' | 'J' |
   'M' | 'O' | 'Q' | 'R' | 'S' | 'T' | 'V' | 'W' | 'Y'
   | 'Z'
```

Functional test information is specified by test vectors containing test conditions for each device pin. Each test vector contains *n* test conditions, where n is the number of pins on the device. The following table lists the conditions that can be specified for device pins.

When using structured test vectors to check your logic design, do NOT use 101 or 010 transitions as tests for clock pins: use C, K, U, or D instead.

Test Conditions

0	Drive input low
1	Drive input high
2-9	Drive input to supervoltage #2-9
B	Buried register preload (not supported)
C	Drive input low, high, low
D	Drive input low, fast slew
F	Float input or output
H	Test output high
K	Drive input high, low, high
L	Verifies that the specified output pin is low
N	Power pins and outputs not tested
P	Preload registers
U	Drive input high, fast slew
X	Output not tested, input default level
Z	Test input or output for high impedance

Note: C, K, U, and D are clocking functions that allow for setup time.

The C, K, U, and D driving signals are presented after the other inputs are stable. The L, H, and Z tests are performed after all inputs have stabilized, including C, K, U, and D.

Test vectors are numbered by following the V character with a number. The vectors are applied in numerical order. If the same numbered vector is specified more than one time, the data in the last vector replace any data contained in previous vectors with that number.

The following example uses the V field to specify functional test information for a device:

```
V0001C01010101NHLLLHHLHLN*

V0002C01011111NHLLHLLLHLN*

V0003C10010111NZZZZZZZZZN*

V0004C01010100NFLHHLFFLLN*
```

JEDEC Kernel Mode, Code 92

```
<kernel>::=<STX><design spec><min. fuse
information><ETX><xmit checksum>
```

```
<design spec>::={<field character>}'*'
```

```
<min. fuse information>::=<fuse list>{<fuse list>}
```

You may use the JEDEC kernel format if you wish to send only the minimum data necessary to program the logic device; for example, if you do not want to send any test vectors. If you specify format code 92, the programmer will ignore everything except the design specification field and the fuse information field. The following fields will be ignored if format 92 is specified: C, F, G, Q, V, and X. Also, the security fuse will be set to zero and the transmission checksum will be ignored.

Figure B-9 shows an example of a kernel JEDEC transmission.

Figure B-9
An Example of JEDEC Kernel Mode Format

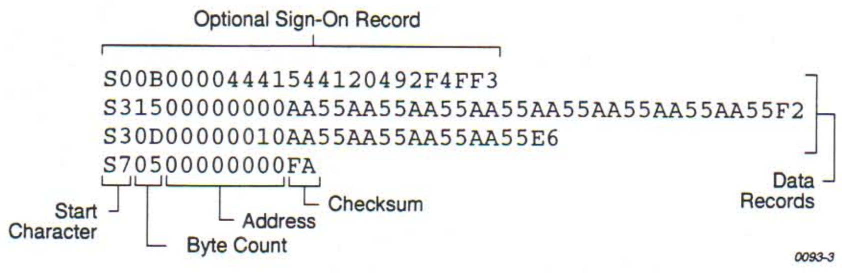
```
<STX>
Acme Logic Design  Jane Engineer   Feb. 29 1983
Widget Decode  756-AB-3456 Rev C Device Mullard 12AX7*

L0000 1111111011 1111111111  1111000000 0000000000
      0000000000 0000000000  0000000000 0000000000
      0000000000 0000000101  1111111111 1111111111
      0000000000 0000000000  0000111101 1111111111
      1111111111 1111110111  1111111111 1111111111*

L0200 1110101111 1111110000  0000000000 0000000000
      1111111111 1111011011  1111111111 1111111110
      0111111111 1111111111  1111111110 1111111111
      1111111111 1111101111  1111111111 1111101111
      0000000000 0000000000  0000*

<EXT>0000
```

0091-2

Motorola 32-Bit Format, Code 95

The Motorola 32-bit format closely resembles the Motorola EXORmacs format, the main difference being the addition of the S3 and S7 start characters. The S3 character is used to begin a record containing a 4-byte address. The S7 character is a termination record for a block of S3 records. The address field for an S7 record may optionally contain the 4-byte instruction address that identifies where control is to be passed and is ignored by the programmer. Figure B-10 shows a sample of the Motorola 32-bit format.

Figure B-10
An Example of Motorola S3 Format



```
S00B00004441544120492F4FF3
S31500000000AA55AA55AA55AA55AA55AA55AA55AA55F2
S30D00000010AA55AA55AA55AA55E6
S70500000000FA
```

Motorola data files may begin with an optional sign-on record, initiated by the start characters S0 or S5. Data records start with an 8- or 10-character prefix and end with a 2-character suffix.

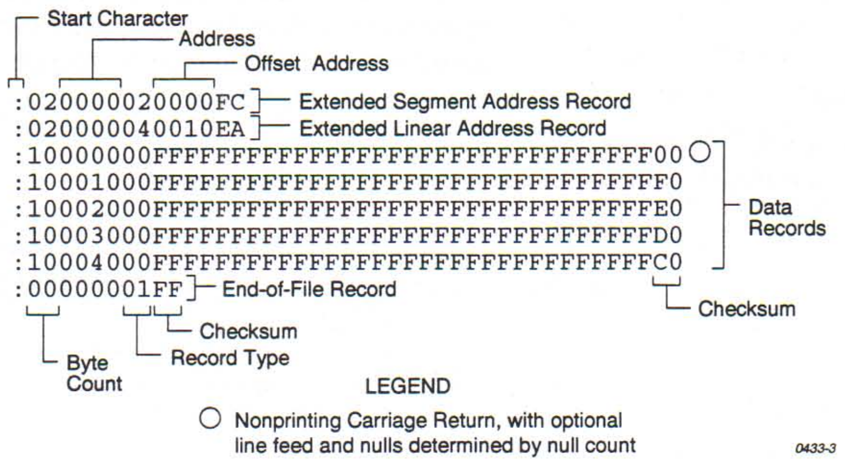
Each data record begins with the start characters S1, S2, or S3: S1 if the following address field has 4 characters, S2 if it has 6 characters, S3 if it has 8 characters. The third and fourth characters represent the byte count, which expresses the number of data, address, and checksum bytes in the record. The address of the first data byte in the record is expressed by the last 4 characters of the prefix (6 characters for addresses above hexadecimal FFFF, and 8 characters for addresses above hexadecimal FFFFFF). Data bytes follow, each represented by 2 hexadecimal characters. The number of data bytes occurring must be 3, 4, or 5 less than the byte count. The suffix is a 2-character checksum, the one's complement (in binary) of the preceding bytes in the record, including the byte count, address, and data bytes.

The end-of-file record begins with an S8 or S9 start character. Following the start characters are the byte count, the address, and a checksum. The maximum record length is 250 data bytes.

Intel Hex-32, Code 99

The Intel 32-bit Hexadecimal Object file record format has a 9-character (4-field) prefix that defines the start of record, byte count, load address, and record type, and a 2-character checksum suffix. Figure B-11 illustrates the sample records of this format.

Figure B-11
An Example of the Intel Hex-32 Format



```
:020000020000FC
:020000040010EA
:10000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00
:10001000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0
:10002000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE0
:10003000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD0
:10004000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC0
:00000001FF
```

Start Character
Address
Offset Address
Extended Segment Address Record
Extended Linear Address Record
Data Records
End-of-File Record
Checksum
Checksum
Record Type
Byte Count
LEGEND
○ Nonprinting Carriage Return, with optional line feed and nulls determined by null count
0433-3

The six record types are described below.

00-Data Record

This record begins with the colon start character, which is followed by the byte count (in hex notation), the address of the first data byte, and the record type (equal to 00). Following these are the data bytes. The checksum follows the data bytes and is the two's complement (in binary) of the preceding bytes in the record, including the byte count, address, record type, and data bytes.

01-End Record

This end-of-file record also begins with the colon start character and is followed by the byte count (equal to 00), the address (equal to 0000), the record type (equal to 01), and the checksum, FF.

02-Extended Segment Address Record

This is added to the offset to determine the absolute destination address. The address field for this record must contain ASCII zeros (Hex 30s). This record type defines bits 4 to 19 of the segment base address. It can appear randomly anywhere within the object file and affects the absolute memory address of subsequent data records in the file. The following example illustrates how the extended segment address is used to determine a byte address.

Problem:

Find the address for the first data byte for the following file.

```
:02 0000 04 0010 EA
:02 0000 02 1230 BA
:10 0045 00 55AA FF ..... BC
```

Solution:

Step 1. Find the extended linear address offset for the data record (0010 in the example).

Step 2. Find the extended segment address offset for the data record (1230 in the example).

Step 3. Find the address offset for the data from the data record (0045 in the example).

Step 4. Calculate the absolute address for the first byte of the data record as follows:

`00100000`	Linear address offset, shifted left 16 bits
`+   12300`	Segment address offset, shifted left 4 bits
`+    0045`	Address offset from data record
`00112345`	32-bit address for first data byte

The address for the first data byte is 112345.

Note: Always specify the address offset when using this format, even when the offset is zero.

During output translation, the firmware will force the record size to 16 (decimal) if the record size is specified greater than 16. There is no such limitation for record sizes specified less than 16.

03-Start Segment Address Record

This record, which specifies bits 4-19 of the execution start address for the object file, is not used by the programmer.

04-Extended Linear Address Record

This record specifies bits 16-31 of the destination address for the data records that follow. It is added to the offset to determine the absolute destination address and can appear randomly anywhere within the object file. The address field for this record must contain ASCII zeros (Hex 30s).

05-Start Linear Address Record

This record, which specifies bits 16-31 of the execution start address for the object file, is not used by the programmer.

Highest I/O Addresses

The following table shows the highest I/O addresses accepted for each Data Translation Format.

Format Number	Format Name	Highest Address (hex bytes)
16	Absolute Binary	N/A
17	LOF	N/A
50-52	ASCII-Hex (Space, Percent, and Apostrophe)	FFFF
55-58	ASCII-Hex (Space, Percent, SMS, and Comma)	FFFF
82	Motorola EXORciser	FFFF
83	Intel Intellec 8/MDS	FFFF
87	Motorola EXORmax	FFFFFF
88	Intel MCS-86 Hex Object	FFFFF
91, 92	JEDEC (Full and Kernel)	N/A
95	Motorola 32 bit (S3 record)	FFFFFFFF
99	Intel Hex-32	FFFFFFFF

Appendix C: Specifications

Power Requirements

Input Voltage*

105–129 Vac	60 Hz
198–264 Vac	50 Hz
90–110 Vac	50–60 Hz

2700 Voltage

24V (ac or dc) ± 10%

2700 Current

ac = 1.67A maximum
dc = 1.25A maximum

Physical and Environmental

Dimensions

228 x 221 x 76 mm
9.00 x 8.75 x 3.00 inches

Weight

5 kg (11 lb)

Temperature

Operating:	+0° to +40°C (+32° to +104°F)
Storage:	-40° to +55°C (-40° to +130°F)

Relative Humidity

Operating:	20% to 80% RH noncondensing
Storage:	10% to 90% RH noncondensing

Altitude

Operating:	To 5,000 meters (16,404 ft)
Storage:	To 15,000 meters (49,212 ft)

* *The 2700 is shipped with the appropriate power supply for your location. Verify that its input requirements are met before you use it.*

Safety

The 2700 complies with the following safety standards:

Underwriters Laboratories — UL 1950

Canadian Standards Association —
CSA C22.2 No. 950

Technischer Überwachungsverein — TÜV
GS-Mark Certification EN60950

Electrostatic Discharge (ESD)

IEC 801-2 (± 8 kV)

Certificate of RFI/EMI Conformance

Data I/O certifies that the 2700 complies with the Radio Frequency Interference (RFI) and Electromagnetic Interference (EMI) requirements of EN55022 Class A and EN50082-1 as called out in 89/336/EEC, the EMC Directive for the European Community.

CE - *EC conformity mark*

Performance Verification

The 2700 verifies internal voltages every time it is powered up and every time a complete self-test is run. The voltage verification is performed by software and is compared to a laser-trimmed voltage reference. To ensure that your 2700 continues to meet product performance specifications, Data I/O recommends that you restart the 2700 software and run a complete self-test cycle once every three months.

Appendix D: Safety Summary

This summary contains general safety information for operating personnel. In addition, specific **WARNINGS** and **CAUTIONS** appear throughout this manual where they apply and are not included in this summary.

Antistatic Wrist Strap

To avoid electric shock, the antistatic wrist strap must contain a 1 MΩ (minimum) to 10 MΩ (maximum) isolating resistor.

Definitions

WARNING statements identify conditions or practices that could result in personal injury or loss of life. **CAUTION** statements identify conditions or practices that could result in damage to equipment or other property.

Grounding the Product

The product is grounded through the grounding conductor of the power cord. To avoid electric shock, plug the power cord into a properly wired and grounded receptacle only. Grounding this equipment is essential for its safe operation.

Power Cord

Use only the power cord specified for your equipment.

Power Source

To avoid damage, operate the equipment only within specified line (ac) voltage.

Servicing

To reduce the risk of electric shock, perform only the servicing described in this manual.

Symbols

⚠	This symbol on equipment indicates that the user should consult the manual for further detail.
V∿	This symbol stands for V ac, for example, 120 V∿ = 120V ac.
⚠	This symbol denotes a fuse rating for a user-replaceable fuse.
⊜	This symbol denotes the protective ground connection.
⏚	This symbol denotes a ground connection for a signal or for an antistatic wrist strap with impedance of 1 MΩ (minimum) to 10 MΩ (maximum).

Zusammenfassende Sicherheitsinformationen

Diese Zusammenfassung enthält allgemeine Sicherheitsinformationen für das Bedienerpersonal. Zusätzlich erscheinen, wenn zutreffend, ausdrückliche Hinweise (ACHTUNG!, VORSICHT!) im Verlauf des Textes. Diese Hinweise werden in dieser Zusammenfassung nicht wiederholt.

Antistatik-Armband

Zum Schutz gegen Stromschläge muß das Antistatik-Armband einen Isolierwiderstand von minimal 1MΩ und maximal 10MΩ enthalten.

Definitionen

Mit **ACHTUNG!** ("WARNING") überschriebene Hinweise dienen zur Identifizierung und Warnung vor Zuständen oder Vorgängen, die Verletzungen oder Tod herbeiführen können. **VORSICHT!** ("CAUTION") dient zum Hinweis auf Zustände und Schritte, die zu Geräte-oder andersartigen Sachschäden führen können.

Erdung des Gerätes

Das Gerät wird durch den dritten Leiter des Netzkabel geerded. Stecken Sie die Netzschnur zur Vermeidung von Stromschlägen nur in eine geerdete Steckdose. Richtige Erdung ist für den problemfreien Betrieb dieses Gerätes unerläßlich.

Netzkabel

Verwenden Sie nur die für dieses Gerät vorgesehene Netzkabel.

Stromquelle

Vermeiden Sie Beschädigungen des Gerätes durch den Betrieb an der vorgeschriebenen Wechselspannung.

Wartung/Reparatur

Führen Sie zum Vermeiden von Stromschlägen nur die in diesem Handbuch erwähnten Wartungsarbeiten durch.

Symbole

⚠	Dieses Symbol bedeutet, daß das Handbuch weitere dem Bediener hilfreiche Hinweise enthält.
V∿	Dieses Symbol bedeutet V ac (Volt Wechselstrom); z.B. 120V∿ = 120V ac.
⚠	Dieses Symbol bezeichnet Sicherungsdaten für vom Bediener auszuwechseinde Sicherungen.
⊜	Dieses Symbol bezeichnet eine Schutzerde-Verbindung.
⏚	Dieses Symbol bezeichnet eine Masseverbindung für ein Signal oder ein Antistatik-Armband mit einer Impedanz von 1 MΩ (min) bis 10 MΩ (max).

Résumé des consignes de sécurité

Ce résumé comprend les informations relatives à la sécurité pour les opérateurs. De plus, tout au long de ce manuel, on retrouve aux endroits appropriés, des **MISES EN GARDE** et des **AVERTISSEMENTS** spécifiques qui ne sont pas inclus dans ce résumé.

Bracelet antistatique

Afin d'éviter tout choc électrique, le bracelet antistatique doit renfermer un résistor de 1MΩ (minimum) à 10MΩ (maximum).

Définitions

Les indications de **MISE EN GARDE** ("WARNING") signalent les conditions ou pratiques qui pourraient causer des blessures corporelles ou la mort. Les indications d'**AVERTISSEMENTS ("CAUTION")** signalent les conditions ou pratiques qui pourraient endommager l'équipement ou entraîner d'autres dommages matériels.

Mise à la terre du produit

Le produit est mis à la terre par l'entremise de la borne de mise à la terre du cordon d'alimentation. Pour éviter tout choc électrique, il faut brancher le cordon d'alimentation uniquement dans un réceptacle mis à la terre correctement et dont les fils ont été rattachés correctement. Il est essentiel de mettre cet appareil à la terre pour qu'il puisse fonctionner sans danger.

Cordon d'alimentation

N'utiliser que le cordon d'alimentation spécifié pour votre appareil.

Source d'alimentation

Pour éviter d'endommager l'appareil, il faut respecter la tension (ca) spécifiée.

Service
Afin de réduire les risques de choc électrique, il faut s'en tenir aux opérations d'entretien et de réparation spécifiées dans ce manuel.

Symboles

⚠ Ce symbole indique que l'utilisateur doit consulter le manuel pour obtenir de plus amples détails.

V∿ Ce symbole représente le voltage en courant alternatif V ca, par exemple, 120 V∿ = 120 V ca.

⚠ Ce symbole indique la valeur nominale d'un fusible remplaçable par l'utilisateur.

⊜ Ce symbole indique la connexion d'isolation à la masse.

⏚ Ce symbole indique une connexion de masse pour un signal ou un bracelet antistatique avec une impédance de 1 MΩ (minimum) à 10 MΩ (maximum).

Riepilogo di sicurezza

Questo riepilogo contiene informazioni di sicurezza per il personale addetto alle operazioni. Inoltre, specifiche note di **ATTENZIONE** e di **AVVISO** relative al contesto fanno parte di questo manuale e non sono state ripetute in questo riepilogo.

Cinghia antistatica da polso
Per evitare le scosse elettriche, la cinghia antistatica da polso deve contenere un resistore di isolamento da 1MΩ (minimo) a 10MΩ (massimo).

Definizioni
Le note di **ATTENZIONE ("WARNING")** identificano condizioni o procedure che potrebbero causare infortuni personali o decessi. Le note di **AVVISO** ("CAUTION") identificano condizioni o procedure che potrebbero causare danni all'equipaggiamento o ad altra proprietà.

Messa a terra del prodotto
Il prodotto viene messo a terra tramite il conduttore della messa a terra del cavo elettrico. Per evitare scosse elettriche, innestare il cavo elettrico in una presa correttamente cablata e messa a terra. La messa a terra di questo equipaggiamento è essenziale per un funzionamento sicuro.

Cavo elettrico
Usare solo il cavo elettrico specificato per l'equipaggiamento.

Fonte di alimentazione
Per evitare danni, operare l'equipaggiamento solo entro la tensione (ca) di linea specificata.

Manutenzione
Per ridurre il rischio di scossa elettrica, svolgere solo la manutenzione descritta in questo manuale.

Simboli

⚠ Questo simbolo indica che l'utente deve consultare il manuale per ulteriori dettagli.

V∿ Questo simbolo indica V ca, ad esempio, 120V∿ = 120 V ca.

⚠ Questo simbolo indica la capacità nominale di un fusibile che può venire sostituito dall'utente.

⊜ Questo simbolo contrassegna la messa a terra di protezione.

⏚ Questo simbolo contrassegna una messa a terra per un segnale o per un cinturino da polso antistatico con impedenza compresa tra 1 MΩ (minimo) e 10 MΩ (massimo).

Resumen de seguridad

En este resumen se proporciona información general sobre seguridad para el personal operario. Además, aparecen notas de **ADVERTENCIA** y **CUIDADO** por todo el manual, donde son apropiadas y no se incluyen en este resumen.

Muñequera antiestática
Para evitar descargas eléctricas, la muñequera antiestática debe contener un resistor aislante de 1 MΩ (como mínimo) a 10 MΩ (como máximo).

Definiciones
Las notas de **ADVERTENCIA** ("WARNING") identifican condiciones o prácticas que pudieran dar como resultado lesiones personales o pérdida de la vida. Las notas de **PRECAUCION** ("CAUTION") identifican condiciones o prácticas que pudieran dar como resultado daños en equipos u otras propiedades.

Conexión a tierra del producto
El producto se conecta a tierra por medio del conductor de masa del cable de alimentación. Para evitar descargas eléctricas, enchufe el cable de alimentación en un receptáculo alambrado y conectado a tierra de modo correcto.

Cable de alimentación
Use sólo el cable de alimentación especificado para el equipo de que se trate.

Fuente de alimentación
Para evitar daños, haga funcionar el equipo sólo dentro de los voltajes de línea especificados (de ca).

Servicios
Para reducir los riesgos de que se produzcan descargas eléctricas, lleve a cabo sólo los servicios descritos en este manual.

Símbolos

⚠ Este símbolo indica que el usuario debería consultar el manual para obtener más detalles.

V∿ Este símbolo representa V ca. Por ejemplo,120 V∿ = 120V ca.

⚠ Este símbolo denota un valor nominal para un fusible reemplazable por el usuario.

⊜ Este símbolo indica la conexión a tierra de protección.

⏚ Este símbolo equivale a una conexión a tierra para una señal o una banda pulsera de antiestática con una impedancia de 1 MΩ (mínima) a 10 MΩ (máxima).

Glossary

Adapter	An adapter, such as the MatchBook and PPI adapters, is installed onto a programmer base. Adapters contain a device socket(s) and allow for support of virtually every device on the market. For information on how to install an adapter, refer to the documentation included with the specific adapter.
Address	A coded instruction designating the location of data or program segments in storage.
Address Offset	See *I/O Address Offset*.
Algorithm	A sequence of voltage or current waveforms used to program a specific device, usually contained on a floppy disk.
Approval	Indication that a device manufacturer has tested an algorithm to support a specific device on a programmer. The level of an approval varies by device manufacturer, but an approval usually indicates both yield and waveform analysis.
Base	The interface between the programmer and the device. The base routes the signals between the device and the Universal Pin Drivers inside the programmer. The base is installed on top of the programmer. For information on how to install a base, refer to Chapter 3 of this User Manual. For some bases, such as most DIP bases, the devices are inserted into socket(s) on the base. For other bases, such as the PPI base, an adapter (which contains the device sockets) is installed on the base.
Baud Rate	A measure of data flow. The number of signal elements per second based on the duration of the shortest element. When each element carries one bit, the Baud rate is numerically equal to bits per second.
BGA	An acronym for *ball grid array*, a type of device package. Usually a square device with one side populated with small solder balls as leads.
Blank Check	A device check that checks a device for programmed bits. If no programmed bits are found, the device is considered blank.

Block Size	The hexadecimal number of bytes to be transferred in a data transfer. The beginning of the block is defined by a begin address, and the end of the block is the sum of the block size and the begin address minus one.
Byte Swap	See *Odd/Even Byte Swap*.
Cards	Also known as PC Cards, cards are 68-pin, integrated circuit cards. PC Cards usually follow the PCMCIA/JEIDA (PC Memory Card International Association) standards, which govern 68-pin memory or I/O-type cards.
Checksum	The sum of all the bytes, expressed in hexadecimal format.
Communications Parameters	The various settings that determine the I/O characteristics of a piece of equipment. The parameters include baud rate, stop bits, data bits, and handshaking.
Compare Electronic ID	A command that compares the electronic signature of the socketed device against the electronic signature specified in the currently selected algorithm.
Context-sensitive	Information that changes depending on the screen position. With the programmer, every time the cursor is moved to a different field, the information on the online help screen changes to reflect the movement.
Continuity Check	A device check that tests for open device pins before performing a device operation.
Data Word Width	The word width of the data to be used during a device operation. For 8-bit (or above) devices, the maximum word width is 64, and the minimum word width is equal to the device width. For 4-bit devices, the word width can be 4, 8, 16, or 32. This value should match the word width of the data bus in the target system for the device being programmed.
Destination	The place where something is sent. The "something" being sent is almost always data. The destination can be RAM, a disk file, or one of the programmer's parallel ports.
Device Begin Address	The first hexadecimal address of device data to use for a device operation. If programming, it represents the first address to program. If verifying, it represents the first address to verify.
Device Block Size	The size of device data to be used in device operations.

Device Libraries	The method of device support for some programmers. The device algorithms are organized according to device technology and pin count.
Device Operation	Usually refers to loading, programming, or verifying, but it can also refer to other available commands, such as device checks and electronic erasing.
Device Word Width	The number of bits in the data word of the device.
DIP	An acronym for *dual in-line package*, the standard rectangular device package with pins extending down from the two longer sides.
Download Data	A file operation that moves a data file from a host computer to the programmer's RAM (or disk drive if the programmer has one).
Download Echoing	Displays the data being downloaded.
E-MICRO	An acronym for *erasable programmable microcontroller*, a type of device technology.
EEPROM	An acronym for *electrically erasable programmable read-only memory*. The device can be either completely or partially erased, electrically in circuit or on the programmer.
Electronic ID	The combination of bytes that identifies the device number and manufacturer of a programmable device.
Enhanced Security Fuse Capability	Found on EMICROs, the Enhanced Security Fuse Capability allows security fuse data to be stored in a data file. For more information, or to see if a device supports this capability, see the device manufacturer's data book.
EPROM	An acronym for *erasable programmable read-only memory*. (Usually refers to UV erasable memories.)
ESD	An acronym for *electrostatic discharge*.
False Positive	In programming, a misprogrammed fuse that retains minimal operational characteristics so that it passes the fuse test. These may be inadequately programmed or over-programmed so that they will fail later in circuit.

Filename	The name of the disk file to use during file operations. The filename must follow standard DOS conventions: up to eight alphanumeric characters, followed by an optional three-character file extension, with the two fields separated by a period. Examples of a valid filename are 27256.dat and filename.c.
Fuse Verification	A type of post-programming device check that checks the fuse pattern programmed into a logic device with the pattern in user memory.
Fusemap	The fuse-level description portion of a programmable integrated circuit. Fusemaps are typically files in JEDEC Standard #3A and are downloaded to PLD programmers for device implementation.
I/O Address Offset	A value subtracted from addresses during input translation and then added to addresses during output translation.
I/O Translation Format	See *Translation Formats*.
Illegal Bit	An illegal bit occurs when a device contains a programmed location and the data file specifies that the location should be unprogrammed.
Illegal Bit Check	A test that determines whether or not a socketed device contains any illegal bits.
JEDEC	An acronyms for *Joint Electron Design Engineering Committee*: a committee of programmer and semiconductor manufacturers that provides common standards for programmable issues. Examples of JEDEC standards include acceptable test characters for PLDs and standard data transfer/programming formats for PLDs. JEDEC Standard #3 is the industry standard for PLD formats.
JEDEC I/O Translate DIP/JLCC Vectors	A feature on the programmer that translates test vectors for a device from its DIP package to its PLCC/JLCC package, allowing for the different pinouts of the two package types.
JEDEC Standard #3A	The standard PLD data translation format, as defined by JEDEC for PLD design software to communicate with PLD programmers. It defines the states of all fuses in the device (the fusemap) and may include test vectors for device testing.

JLCC	An acronym for *J-style leadless chip (or ceramic) carrier*, a type of device package used for surface-mount applications. The device has J-shaped leads that are "open" at one end (leads are usually present on all four sides).
LCA	An acronym for *logic cell array*: a reconfigurable programmable gate array.
Logic Verification	After a device has been programmed, the user may select test vector verification, fuse verification, or both.
LSB	An abbreviation for *least significant byte.*
Master Device	A device that contains data to be programmed into another device. Data is loaded from a master device and programmed into a blank device.
MatchBook	A durable plastic carrier that simplifies the handling of surface-mount devices.
Memory Begin Address	The first address, in hex, of the first byte of data to be used in device operations.
MSB	An abbreviation for *most significant byte.*
Next Device	Used during serial set programming, this value specifies the next device in the set. For example, if 8-bit devices are being used and a word width of 16 bits has been specified, two devices will be required to store each 16-bit word. Depending on the value entered, the data programmed into the next device will come from either even addresses or odd addresses.
Odd/even Byte Swap	Used during device operations for 16-bit devices, this option swaps the most significant bytes (MSB) and the least significant bytes (LSB) of 16-bit words. The programmer stores RAM data and disk file data with the convention that the LSB of a 16-bit word resides in the even byte of memory.
Overblow	A condition in which fuses are blown that should not have been.
Overblown Fuse	A fuse that has been over-programmed to the extent that the surrounding area may have been damaged or fuse material "splatter" was created. Splatter (or rattlers) can cause intermittent shorting.

PAL	An acronym for *programmable array logic*. PALs are devices with programmable AND and fixed OR arrays. This is a slightly different architecture from a PROM or an FPLA. Other examples of PAL-type architectures from other manufacturers include PEEL and GAL.
Part Number	The number on the device. For example, if the device being used is an Intel 27C256, the part number would be 27C256.
PGA	An acronym for *pin grid array*, a type of device package. Usually a square device with one side populated with small pins as leads.
PIC	An acronym for *programmable integrated circuit*. PICs are programmable devices (memory or logic) that are delivered to customers in a generic state and programmed by the customer to perform a specific function. Examples include PLDs, PROMs, ASPLs, and FPGAs.
Pin driver	The electric circuit reading or applying voltage and current pulses to the individual pin of a device, for programming or testing. See also ***Universal Pin Driver***.
PLCC	An acronym for *plastic leaded chip carrier*, a plastic device package with J-shaped leads that extend from four sides downward. It is used for surface-mount applications.
PLD	An acronym for *programmable logic device*, a type of programmable integrated circuit. Architectures range from very simple to very complex. Most PLDs contain two levels of logic, an AND array followed by an OR array.
Program	The controlled application of electrical pulses to program specific fuses or cells.
Program Device	A device operation that copies device data into a socketed device. The programming is done according to the programming algorithm selected in the select device stage. The programming operation can also include a verify operation.
Program Security Fuse	A programming parameter that enables/disables the programming of the device's security fuse.
Program Signature	Available on only a few devices, the program signature is a user-definable field that allows the user to program data into the program signature array. For example, the program signature could contain the revision level or modification date of the data in the remainder of the device.

Programmable Integrated Circuit	One of the four basic categories of ASICs: the other three being gate arrays, standard cells, and full custom devices. PICs are ICs that are user configurable. PLDs and PGAs are examples of programmable integrated circuits.
PROM	An acronym for *programmable read-only memory*. A device with fixed AND and programmable OR arrays. This is a slightly different architecture from an FPLA or a PAL.
QFP	An acronym for *quad flat pack,* a type of device package. A QFP is a square or rectangular device with leads on all four edges (leads can be either straight or gull-wing); it is used for surface-mount applications.
QUIP	An acronym for *quad inline package*, a type of device package. Similar to DIP, but with staggered long & short leads (leads on long sides).
Read Device	A device operation that copies data from a master device into User memory.
Reboot	The process of re-initializing the programmer or PC. After rebooting, the programmer or PC is in the same state it would be in if it had just been turned on.
Registered Devices	Devices that contain registers, rather than being combinatorial only. Registered devices are typically used for sequencers and state machine designs. Typical examples are 16R8, 82S159, and 22V10.
Reject Option	A post-programming device check that pulses the programmed device with voltage to see if the device has programmed per specification. The number of times a device is pulsed varies by manufacturer and by the reject option selected.
SDIP	An acronym for *shrink dual inline package*, a device similar to a DIP but with more leads at tighter pitch. A separate adapter is required to program SDIPs.
Security Fuse	A location in a programmable device that, when programmed, secures the device from readback: the data in the device is unreadable.
Security Fuse Data	The actual data to program into the device's security fuse.

Select a Device	A procedure that tells the programmer which device will be used. The device can be selected by specifying the manufacturer and the device part number.
Self test	A built-in self-diagnosis command that allows the user to test various circuits and subsystems in the programmer, verifying proper operation or isolating possible problem areas.
Serial Set	A method of set programming in which the devices of the set are programmed one at a time instead of all at once.
Set Programming	A type of programming in which a large data file is partitioned and programmed into multiple memory devices.
SIMM	An acronym for *single inline memory module*, a type of device package. A rectangular device with leads on one long edge.
SOIC	An acronym for *small outline integrated circuit*, a type of surface-mount device package. The device is rectangular with gull-wing leads on the long sides. SOIC is synonymous with **SOP** (small outline package).
SSOP	An acronym for *shrink small outline package*, a type of device package. An SOP with more leads at higher and finer pitch; also called TSOP II.
Structured Test Vectors	A string of test conditions applied to a PLD in a programmer/ tester to stimulate inputs and test outputs to ensure functionality. An example of a test vector is. 20 characters for a 20-pin PLD, with 10 input signals and 10 expected outputs.
Structured Test Vectors (design)	Structured vectors created by the design engineer to confirm that the design is operating as intended: for instance, that a 10-bit counter is counting to 10. Design vectors are used in both preprogramming simulation and in manufacturing.
Structured Test Vectors (device)	Structured vectors created by the design engineer, test engineer, or an automatic test vector generation program, which confirm that the device is operating properly after programming. For instance, structured vectors can ensure that nothing can happen in the device to prevent the 10-bit counter from operating correctly. An exhaustive set of device vectors will assure that no undetectable faults may occur.

Sumcheck	A 4- or 8-digit hexadecimal number that, when compared to the original data, allows the user to verify that a copy of the data matches the original data. Memory devices have 8-digit sumchecks and logic devices have 4-digit sumchecks. For devices in a set, both the individual sumcheck of the device and the sumcheck of the entire set can be calculated.
Test Vector	A sequence of inputs and outputs used to verify the function of the design before programming, and of the device after programming. Inputs specified by a test vector are applied to the design or device, and the outputs obtained are checked against the outputs in the test vector.
Test Vector Stretching	Conversion of DIP test vectors to equivalent PLCC test vectors by adding "don't care" vector characters into the string to correspond with the PLCC's dead pins.
Total set size	Used during serial set programming, this value specifies how many devices are in a set.
Translate DIP/JLCC Vectors	See *JEDEC I/O **Translate DIP/JLCC Vectors.***
Translation Formats	A form of transmission protocol, these formats are used when transferring data between the programmer and a host computer. The different formats represent different ways of encoding the device data in a data file. The data file could contain the fuse pattern for a logic device or the data for a memory device.
TSOP	An acronym for *thin small outline package,* a rectangular device package with gull-wing leads on the short sides. Also called TSOP I.
Underblow	A condition in which fuses that should have been blown or programmed were not.
Underblown Fuse	A fuse that did not disconnect as specified by the manufacturer. These fuses may test properly, but tend to be more prone to grow back when in circuit, rendering the PLD useless.
User Data Size	The hexadecimal number of bytes of a data block to use for a device operation. Normally, this value is equal to the device size. During serial set operations, this value works with Total Set Size to determine the total number of bytes to program into a set of devices.

User memory	The workspace used during device operations. It can be either internal RAM or a disk file (on programmers that support this option). Normally, RAM is used for small, quick device operations, such as programming a single device, while disk is used for larger device operations, such as serial set programming. (User memory can also refer to ***User RAM*** in some situations.)
User RAM	The RAM dedicated to the programmer. User RAM can be used as a source/destination for an operation. Several operations use user RAM as a temporary storage buffer, overwriting any data that may have been there previously.
Verify Device	A device operation that compares data in a programmed device with data in RAM or in a data file. With logic devices, verifying can also include functional testing. Verify is an automatic part of the program operation, but additional verify operations can provide useful information about any errors.
Verify Pass	A verify pass is a trip through a device at a specified Vcc to see if the device programmed properly. The pass is usually done once at 5V. The pass can also be done twice, with the first pass at 4.5V and the second pass at 5.5V.
Waveforms	Images of the programming pulses that program a device. Usually created by programmer manufacturers and submitted to device manufacturers as part of the approval process and to record the correct programming spec for a specific device.
Word size	The word size is the word width of the data to be read, programmed, or verified. For 8-bit (or larger) devices, the minimum word width is equal to the device word width and the maximum is 64. For 4-bit devices, word width choices are 4, 8, 16, and 32. Word size is the same as data word width.
Yield	The percentage of successfully programmed devices.
ZIF Socket	An acronym for *zero insertion force socket*, a socket in which the device can be dropped in and engaged by means of a lever.

Index

End User Registration

Please complete and return this card so we can keep you informed of product updates and upgrades. Also, use this card to notify us if your address changes or if you are the new end user of this product.

Customer Information

☐ Check if this is a new address or end user.

Name ______________________________

Department ____________________ Mail Stop ____________

Company ____________________ Phone ____________

Address ______________________________

____

City ____________ State ____________ Zip ____________

Country ______________________________

Product Information

Product Name ______________________________

Version Number ____________ Serial Number ____________

Comments on Documentation

Product Name ______________________________

Manual Part Number *(on Title Page)* ______________________________

Why do you use the manual? ___ For setup ___ For reference ___ For problem-solving

Can you find the information you need quickly? ______________________________

Is the information accurate and complete? ______________________________

What sections of the manual do you find most useful? ______________________________

____

What information would you like to see added? ______________________________

____

How would you rate the overall effectiveness of the manual? (Low) 1 2 3 4 5 (High)

Additional comments ______________________________

____

____

Name ______________________________

Department ____________________ Mail Stop ____________

Company ____________________ Phone ____________

Address ______________________________

City ____________ State ____________ Zip ____________

Country ______________________________