AutoSite™

Automated Production Programmer

User Manual

DATA I/O

981-0304-006

April 1998 981-0304-006

Data I/O Corporation
10525 Willows Road N.E., P.O. Box 97046
Redmond, Washington 98073-9746 USA
(425) 881-6444
http://www.data-io.com

Acknowledgments:

Table of Contents

Operation

Computer Remote Control

Translation Formats

Glossary

Figures

Safety Summary

General safety information for operating personnel is contained in this summary. 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.

Fuse Replacement — For continued protection against the possibility of fire, replace the fuse only with a fuse of the specified voltage, current, and type ratings.

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.

Hearing Protection — Noise levels generated by the ProMaster handler while it is operating can exceed 70 dB. It is recommended that hearing protection be worn at all times by personnel working near the handler while it is operating.

Power Cord — Use only the power cord specified for your equipment.

Power Source — To avoid damage, operate the equipment only within the specified line (ac) voltage range.

Servicing — To reduce the risk of electric shock, perform only the servicing described in this manual.

Symbol	
	This symbol indicates that the user should consult the manual for further detail.
	This symbol stands for Volts ac, for example: 120 V ∿ = 120 Vac.
	This symbol denotes a fuse rating for a user-replaceable fuse.
	This symbol denotes earth ground. An antistatic wrist strap with impedance of 1 MΩ (minimum) to 10 MΩ (maximum) can be attached to terminals designated for that function and marked with this symbol.
	This symbol denotes compliance of the programmer with the requirements called out by the EC (European Community) for this equipment.
	This symbol denotes dangerous, high voltage is present and precaustions should be taken to prevent injury from electrical shock.
	This symbol denotes that movement of system components can cause physical injury from pinching or crushing.

Preface

The Preface describes how to contact Data I/O for technical assistance, for repair and warranty services, and Keep Current™ subscription service. It also describes how to reach Data I/O's Home Page on the World Wide Web.

Data I/O Customer Support

United States

For technical assistance, repair, or warranty service, contact:

Technical Operations
Telephone: 1-800-247-5700 (*Press 2 on your touch-tone telephone to bypass the recorded message and speak to the first available Support Engineer*)
Fax: 425-867-6972
E-mail: pmhelp@data-io.com

For Keep Current subscription service or repair service contract, contact:

Corporate Sales
Telephone: 1-800-332-8246
Fax: 425-869-7423
E-mail: telsales@data-io.com

Canada

For technical assistance, contact:

Technical Operations
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: 425-867-6972
E-mail: pmhelp@data-io.com

For repair or 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

Japan

For technical assistance, repair or 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: 3-3779-2151
Fax: 3-3779-2203

Germany

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

Data I/O GmbH
Lochhamer Schlag 5
82166 Gräfelfing
Telephone: 89-858-580
Fax: 89-858-5810

Other Countries

For technical assistance, repair, or warranty service, or Keep Current subscription service contact your local Data I/O representative.

Contacting Data I/O

You can contact Data I/O for technical assistance by calling, sending a fax or electronic mail (e-mail), or using the Bulletin Board Service (BBS).

To help us give you quick and accurate assistance, please provide the following information:

- ProMaster serial number
- Product version number
- Detailed description of the problem you are experiencing
- Error messages (if any)
- Device manufacturer and part number (if device-related)

Telephone

Call the appropriate Data I/O Customer Support number listed at the front of the Preface. When you call, please be at your programmer or computer, have the product manual nearby, and be ready to provide the information listed above.

Fax

Fax the information listed above with your name, phone number, and address to the appropriate Data I/O Customer Support fax number listed at the front of the Preface.

E-mail

To reach Data I/O via e-mail, send a message including your name, telephone number, e-mail address, and the information listed above to the following address:

`pmhelp@data-io.com`

Bulletin Board Service

The Data I/O Bulletin Board System (BBS) enables you to:

- Obtain a wide range of information on Data I/O products, including current product descriptions, new revision information, Customer Support information, and application notes.
- 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.

Multiple lines are available, all supporting U.S. Robotics V.34+ modems. Online help files provide more information about the BBS and its capabilities. BBS numbers are as follows:

Japan	81-3-3779-2233
United States	425-882-3211

World Wide Web (www.data-io.com)

The Data I/O Home Page on the World Wide Web 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 Web, you need an Internet account with Web access, and a Web browser.

The address of the Data I/O Home Page is **http://www.data-io.com**.

Warranty Information

Data I/O Corporation warrants this product to be free from defects in material and/or workmanship for a period of twelve months from the original date of shipment to the buyer.

The warranty does not include normal wear or replacement components, programming sockets, drive belts, rollers, and socket contacts that contact devices being processed.

This warranty shall apply only if the product fails to function properly under the normal intended use. Should this product fail to be in good working order anytime during the twelve-month warranty period, Data I/O Corporation shall, at its sole option, repair or replace this product at no additional charge, except as set forth below.

The foregoing is the sole responsibility of Data I/O Corporation under this warranty, and any liability for incidental or consequential damages is expressly disclaimed.

Repair parts and replacement products shall be on an exchange basis and shall be either new or reconditioned. All replaced parts and products shall become the property of Data I/O Corporation. Parts will be delivered to buyer for repair by buyer or, at buyer's option, the product may be returned to Data I/O Corporation for repair.

Warranty coverage will not be granted if, in the sole opinion of Data I/O Corporation, the defect or malfunction was caused by accident, abuse or misuse, neglect, improper packing, or improper or unauthorized modifications or service. Problems resulting from use of non-Data I/O labels, ribbons, and other components will invalidate this warranty and result in service charges.

The buyer is responsible for returning the product, properly packaged in its original container or equivalent, to a Data I/O service office. Any insurance or shipping costs incurred in presenting or sending the product for service is the sole responsibility of the buyer.

This warranty is in lieu of any other warranty, expressed or implied, including but not limited to, any implied warranty of merchantability or fitness for a particular purpose, and any other obligations or liability on the part of Data I/O Corporation.

Repair Service

After the warranty period expires, repair services are available at Data I/O Service Centers on a time-and-materials basis, and through a fixed price annual agreement that covers all parts and labor needed to correct normal malfunctions. The annual agreement includes semiannual performance certification.

For more information, call the Data I/O Customer Resource Center at the numbers listed at the front of the Preface. To order a Repair Service Contract, call Data I/O Corporate Sales at **1-800-332-8246**.

End User Registration and Address Change

If the end user for this product or your address has changed since the Registration Card was mailed, please notify Data I/O Customer Support at the numbers listed at the front of the Preface. This ensures that you receive information about product enhancements. Be sure to include the product serial number, if available.

1 Introduction

What Is AutoSite?

The AutoSite Automated Production Programmer is designed specifically for use with device handlers in a production environment. AutoSite employs “pin driver at the pin” technology, which allows the pin drivers to be as close to the device as possible.

AutoSite supports virtually every programmable memory, logic, and microcontroller device in DIP, PLCC, and most SOIC packages. Device support is available through use of different programming modules, each of which is designed for a specific package type and device pin count. With programming modules, you can customize device support to suit your specific programming needs and budget.

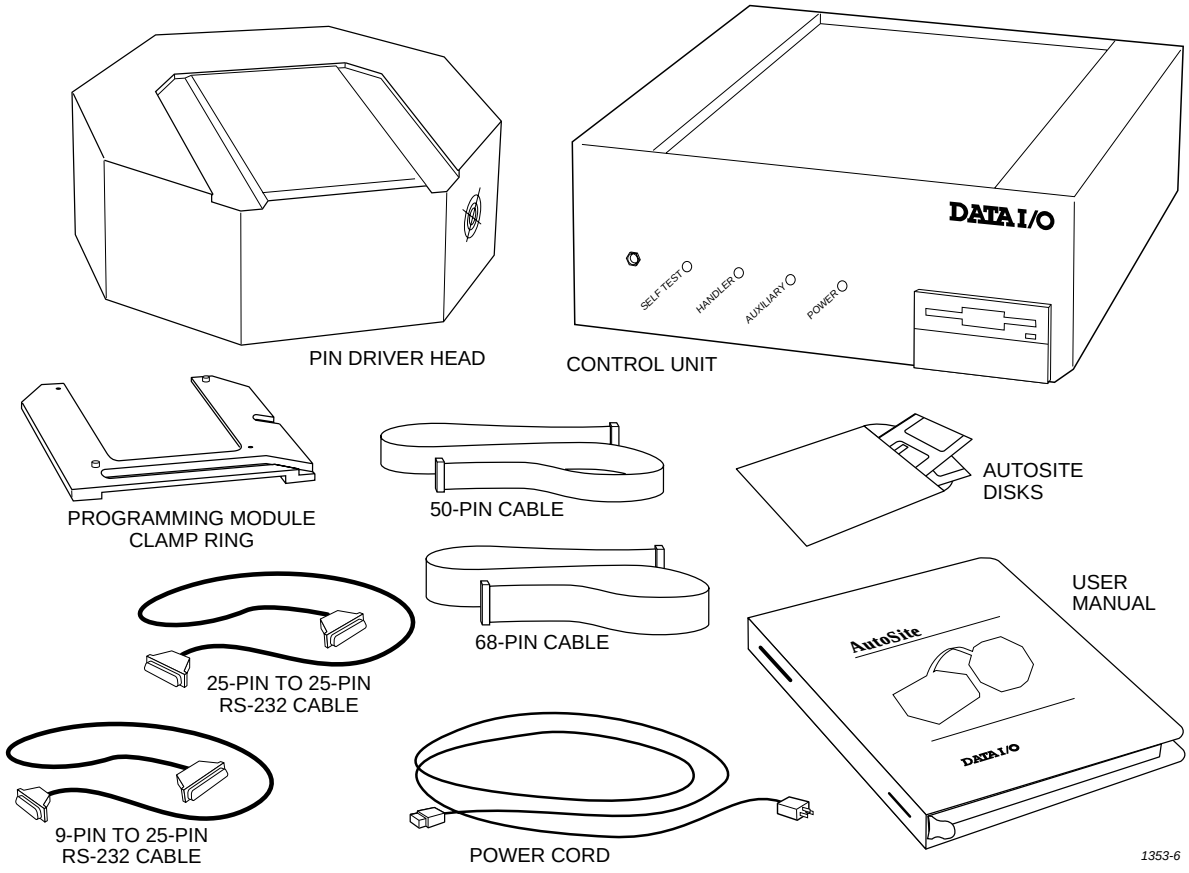
AutoSite is composed of two main pieces: the control unit and the pin driver head. The control unit contains the disk drive, the serial ports, the power supply, and the circuitry that generates control signals that are sent to the pin driver head. The pin driver head receives the control signals from the control unit and generates the voltages necessary to program a device.

AutoSite comes in two configurations: support for up to 88 pins, with 8MB of RAM; and support for up to 44 pins, with 8MB of RAM. The 44-pin configuration can easily be upgraded to the 88-pin configuration.

Package Contents

Figure 1-1 shows the contents of the AutoSite system.

Figure 1-1
AutoSite Package Contents



Note: If you purchased a ProMaster system, the control unit will already be installed in the handler.

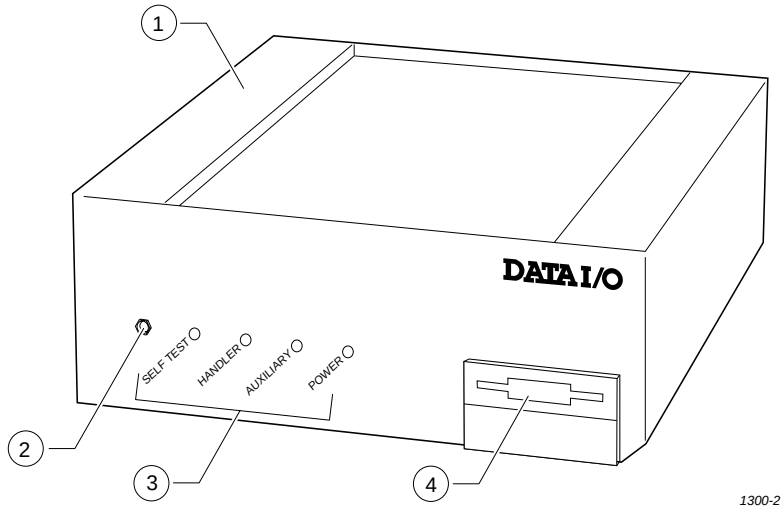
If you purchased a non-ProMaster system, you should have received a handler interface kit from your handler manufacturer. See the documentation supplied with the interface kit for more information and for a list of the kit's contents.

AutoSite External Features

The Control Unit

The front panel features of the control unit are shown in Figure 1-2.

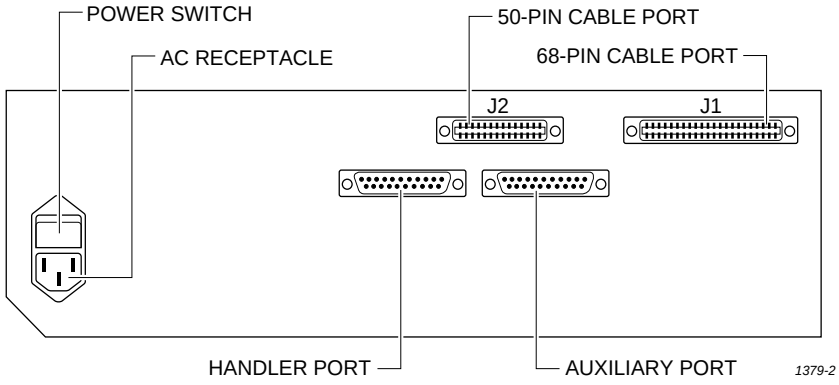
Figure 1-2
Front Panel Features



1. **Control Unit**—Houses the circuitry that controls the pin driver head.
2. **Ground Connection**—Connect an antistatic wrist strap here.
3. **AutoSite Status Indicators**—These indicators provide information about AutoSite's operational status:
   - Self-Test Indicator—This lamp is lit when AutoSite is performing a self-test.
   - Handler Indicator—This lamp is lit when AutoSite is communicating with the equipment connected to AutoSite's Handler port.
   - Auxiliary Indicator—This lamp is lit when AutoSite is communicating with the equipment connected to AutoSite's Auxiliary port.
   - Power Indicator—This lamp is lit when the power is on.
4. **Disk Drive**—Insert the Boot disk and Algorithm / System disk here.

The back panel features of the control unit are shown in Figure 1-3.

Figure 1-3
Back Panel Features

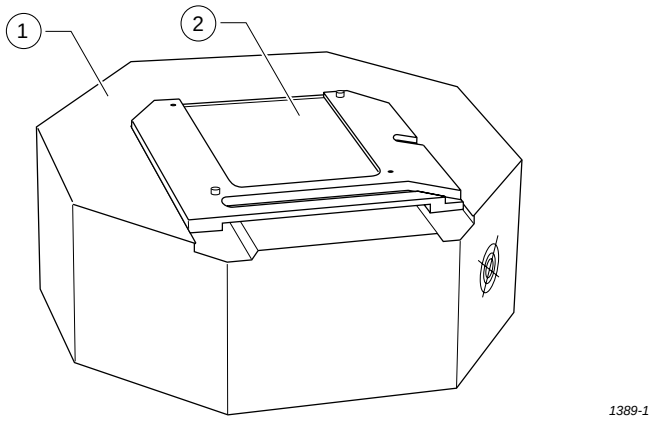


- ac Receptacle—Connects AutoSite to ac power.
- Power Switch—Applies ac power to AutoSite.
- Handler Port—Connects AutoSite to a PC.
- Auxiliary Port—Used for system diagnostics and field service.
- 50-pin Cable Port—A 50-pin cable attaches here, connecting the AutoSite control unit to the pin driver head.
- 68-pin Cable Port—A 68-pin cable attaches here, connecting the AutoSite control unit to the pin driver head.

The Pin Driver Head

The features of the pin driver head are shown in Figure 1-4.

Figure 1-4
Pin Driver Head Features



1. **Pin Driver Head**—Contains the universal pin drivers that supply power and ground to either 44 or 88 pins, depending on your system configuration.
2. **Programming Site**—Programming modules, the DIP Base, and the PLCC Base fit here, connecting the pin drivers in the pin driver head to the socketed device.

Specifications

Functional	**RAM**	8 MB standard (on units shipped after July 1997)
	Disk Format, Floppy	Double-sided, Quad-density 3.5-inch disk with 135 tracks per inch. 1.44MB formatted
	Disk Format, MSM (optional hard drive)	Minimum of 80MB
	Controller	Motorola 68000 16-bit microprocessor
	Terminal Support	Interfaces with ANSI 3.64 compatible terminals, IBM PCs and compatibles running a terminal emulator program, and many popular ASCII terminals
	Communication Standard	RS-232C
	Data transfer rate	110 to 19.2 K baud (up to 115.2K baud using TaskLink)
Power Requirements	**Operating Voltages**	90 to 264 Vac
	Frequency Range	50–60 Hz
	Power Consumption	150 VA maximum
	Input Current	1.5A maximum
Physical and Environmental	**Dimensions**	Control unit: 17.15h x 33.65w x 28.5d cm 6.75h x 13.25w x 11.25d in. Pin driver head: 8.9h x 27.3w cm 3.5h x 10.75w in.
	Weight	7.7 kg (17 lbs.)
	Temperature	Operating: +5° to 40°C (+40° to 105°F) Storage: +5° to 50°C (+40° to 122°F)
	Transportation:	-40° to +55°C (-40° to +130°F)
	Relative Humidity	Operating: to 80% noncondensing Storage: to 90% noncondensing
	Altitude	Operating: to 15,000 meters

Safety

AutoSite is certified by UL, CSA, and TUV to comply with the following safety standards:

Underwriters Laboratories—UL 1950

Canadian Standards Association—CSA C22.2 No. 231

Technischer Überwachungsverein—
TUV GS-Mark Certification
EN60950

Electrostatic Discharge (ESD)

IEC 801-2 (± 8 kV)

Certificate of RFI/ EMI Compliance

The following paragraph applies to all units shipped after January 1996:

Data I/O certifies that the AutoSite 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*

WARNING: This equipment is a class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

Performance Verification

AutoSite 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. Data I/O recommends that you cycle power AND run a complete self-test cycle at least every three months. Contact Data I/O for information on checking the reference voltage and the master clock.

To ensure that your AutoSite continues to meet product performance specifications, Data I/O recommends that your programmer be returned to an authorized Data I/O Service Center every twelve months for a complete performance evaluation.

Options

The items listed below complement the AutoSite Automated Production Programmer. For more information, or to order an item below, contact Data I/O Customer Support as listed in the Preface.

Keep Current Subscription Service

Data I/O offers a one-year subscription to keep your programmer and documentation up-to-date with the latest features and device support. This subscription also incorporates manufacturer-recommended changes to existing device support to maintain optimum yields, throughput, and long-term reliability.

In addition, you receive immediate access to new and updated programming algorithms via our Keep Current Bulletin Board System (BBS)—up to three months before the algorithms are available in an update kit.

For more information, see the Keep Current documentation located behind the Keep Current tab, or contact Data I/O Customer Support.

TaskLink Software

TaskLink is PC/programmer interface software designed for use with AutoSite and other Data I/O programmers. TaskLink runs on an IBM PC (or compatible) and allows you to control AutoSite from a personal computer for streamlined and enhanced programming operations. TaskLink features automatic programming file configuration, full-screen editing, and error-logging. TaskLink also features a windowed interface, extensive online context-sensitive help, and full mouse support.

88-pin Upgrade Kit

The 88-pin upgrade kit contains all the hardware and software required to convert your 44-pin AutoSite into an 88-pin AutoSite.

Mass Storage Module

The Mass Storage Module provides the storage space required to store the growing number of device programming algorithms for AutoSite. In addition, the extra storage space provided by the Mass Storage Module allows for additional commands and features to be added without hampering normal operation in a production environment. Specifically, the MSM virtually eliminates the need to swap disks under normal programming operation.

Once the MSM is installed, the AutoSite system software can easily be installed on the MSM. During normal operation, the MSM is virtually transparent.

Stand-alone Kit

Provides DIP and PLCC support to the AutoSite programmer in stand-alone mode. Includes a DIP Base, a PLCC Base, and a set of MatchBook device carriers that provide support for 20-pin to 84-pin PLCCs. The stand-alone kit is best suited for diagnostics and single-device programming.

Additional Programming Modules

The table below lists the different programming modules available for the AutoSite programmer. The table also shows which Data I/O handlers support a particular programming module.

Device Type	ProMaster 2500	ProMaster 3000	ProMaster 7000/7500
150-mil SOIC			
220-mil SOIC			
300-mil SOIC			
350-mil SOIC			
450-mil SOIC			
530-mil SOIC			
300-mil DIP			
600-mil DIP			
20-pin PLCC			
28-pin PLCC			
32-pin PLCC			
44-pin PLCC			
52-pin PLCC			
68-pin PLCC			
84-pin PLCC			

For example, AutoSite supports 84-pin PLCCs when used with a ProMaster 3000, ProMaster 7000, or ProMaster 7500 handler. For more information or for a list of the current programming modules, contact Data I/O Customer Support as listed in the Preface.

Note: For AutoSite use with a non-ProMaster handler, refer to the documentation supplied with your handler or contact your handler manufacturer for information on available programming modules.

2 Setup and Installation

This chapter describes how to set up AutoSite and get it working with your equipment. Before you read this chapter, make sure you have read the previous chapter, “Introduction.”

This chapter guides you through configuring the hardware, connecting AutoSite to a device handler, installing the system software, and powering up AutoSite for the first time. The installation process is divided into the following steps:

Once you get AutoSite set up and installed, you will want to refer to Chapter 3, “Operation,” for information on commonly performed tasks such as changing programming modules and adding device support packages.

Before You Begin

Before you begin the setup and installation, make sure you read and understand the terms of the Software License Agreement, which is printed on the outside of the envelope containing the AutoSite disks.

Connect AutoSite to a ProMaster 2000

This section describes how to connect AutoSite to a ProMaster 2000 handler. The installation is divided into two main steps:

- Attach the AutoSite control unit to the 2000
- Attach the AutoSite pin driver head to the 2000

Figure 2-1 is a flowchart that illustrates the general flow of the installation procedures contained in this section.

What You Need

In addition to the contents of the Installation Kit, you will need the following to connect AutoSite to a ProMaster 2000:

- Programming module
- Grounded wrist strap
- Antistatic workstation
- 5/32-inch hex driver
- Flatblade screwdriver
- #2 Phillips screwdriver

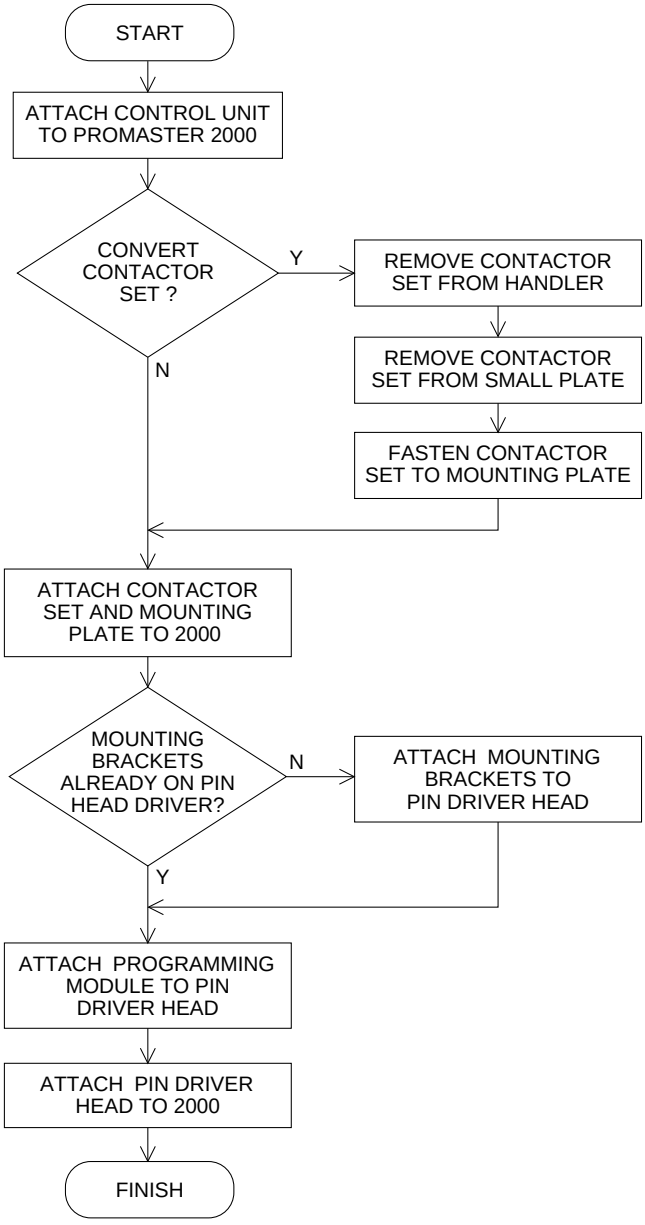
Safety Information

This information is provided as a supplement to the Safety Summary at the beginning of this manual.

The circuitry housed inside the pin driver head and the control unit and the devices AutoSite programs are static sensitive and can be damaged by electrostatic discharge (ESD). To help minimize the effects of ESD, we suggest you wear an antistatic wrist strap while you follow the procedures described in this section.

For best performance, the antistatic wrist strap should be connected to a properly grounded antistatic workstation and the wrist strap should contain a 1MΩ (minimum) to 10MΩ (maximum) isolating resistor.

Figure 2-1
Flowchart of the Installation Process for the ProMaster 2000



Attaching the Control Unit

Connect the AutoSite control unit to a ProMaster 2000 as follows:

1. Unplug the power cord from the AutoSite control unit.
2. *(For control units without connector brackets at ports J1 and J2)* Make sure the 50-pin cable and the 68-pin cable are disconnected from the control unit and the pin driver head. The cables and the ports to which they connect are shown in Figures 1-1 and 1-3.

   (For control units with connector brackets at ports J1 and J2) Make sure the 50-pin cable and the 68-pin cable are disconnected from the pin driver head.
3. Locate the two flathead screws shown in Figure 2-2 and remove them from the control unit. Set these screws aside; you will need them later.
4. As shown in Figure 2-2, position the control unit mounting plate against the control unit so the countersunk holes on the control unit mounting plate are facing away from the control unit. Also, make sure that the narrow end of the control unit mounting plate is pointing toward the disk drive on the control unit.

   Secure the control unit mounting plate to the control unit with the two flathead screws you removed in step 3. Tighten the screws with a #2 Phillips screwdriver. Do not overtighten the screws.

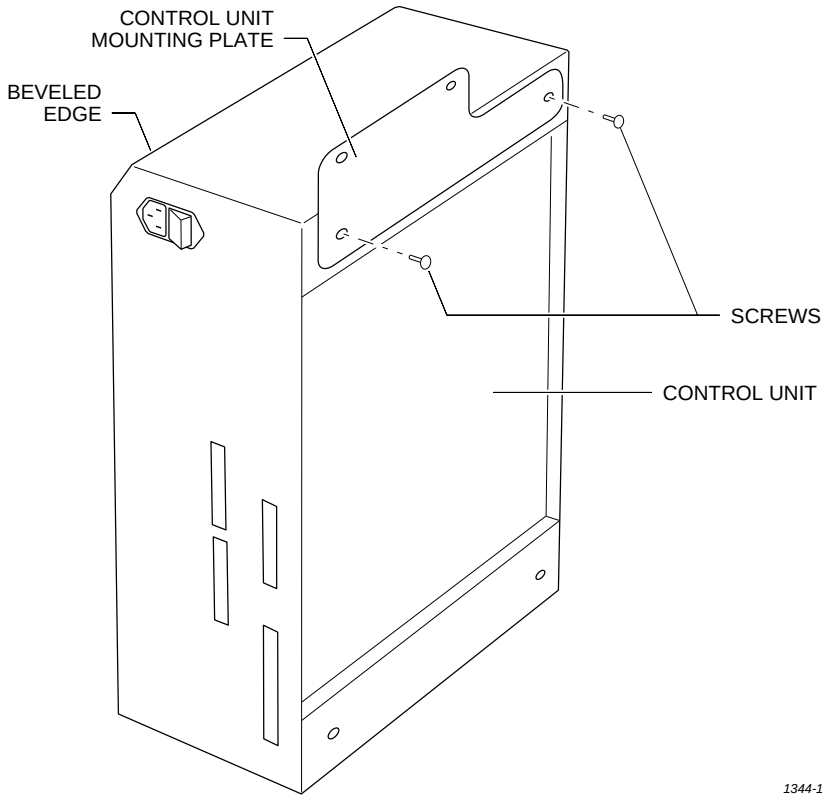
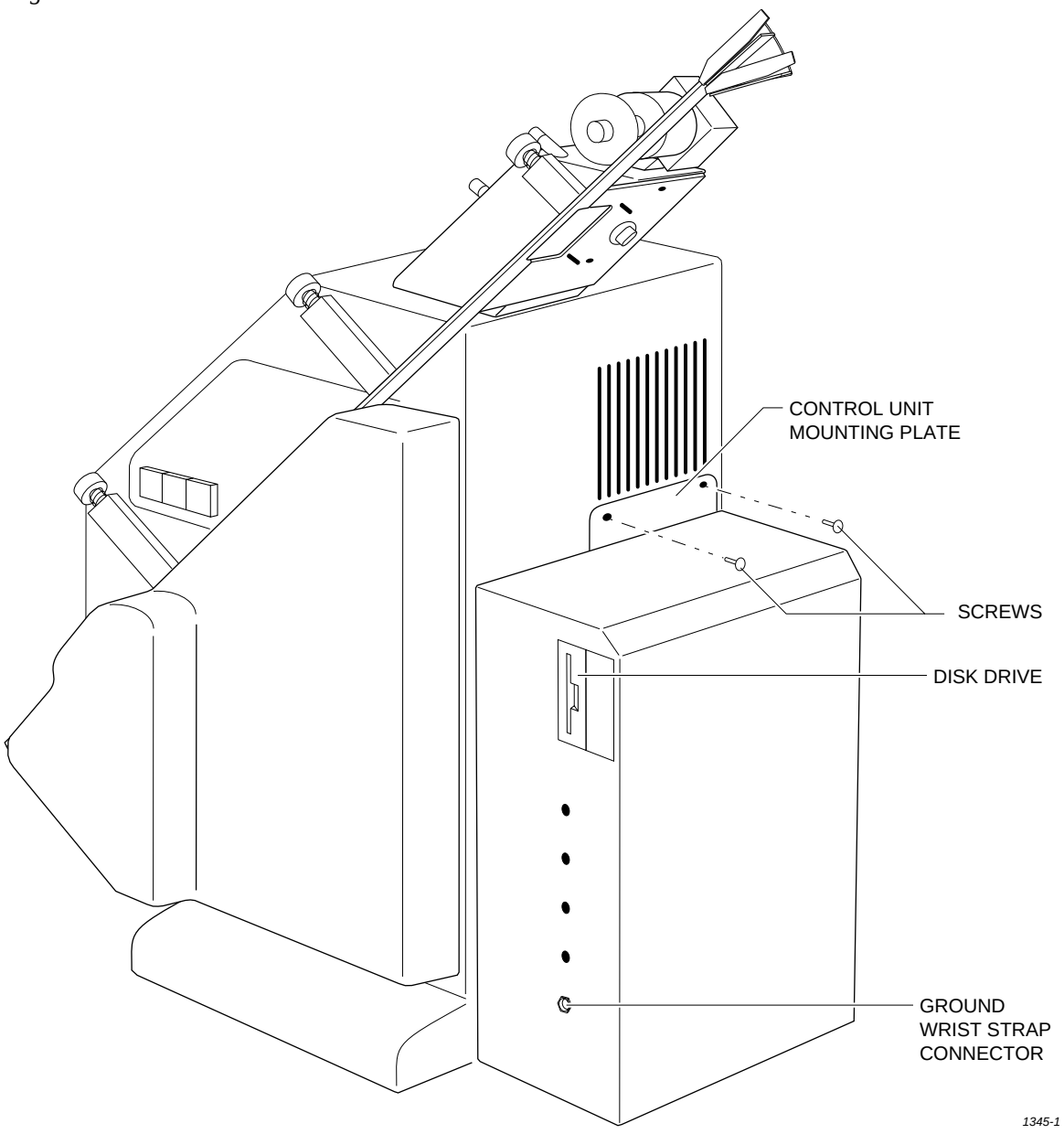


Figure 2-2
Attaching the Control Unit Mounting Plate to the Control Unit

5. Move the four rubber pads from the bottom of the control unit to the edge of the control unit opposite the bevelled edge.
6. Locate the fan opening on the back of the 2000. Using a 5/32-inch hex driver, remove the two screws on the bottom edge of the fan opening. Discard these screws; you will not need them later. Position the control unit against the back of the handler as shown in Figure 2-3. The control unit should be sitting on its four rubber pads.
7. Attach the control unit to the handler with the two buttonhead screws provided. Tighten the screws with a 5/32-inch hex driver. Do not overtighten the screws.

Figure 2-3
Attaching the Control Unit to the ProMaster 2000



Converting a Contactor Set

This section describes how to convert a contactor set for use with AutoSite.

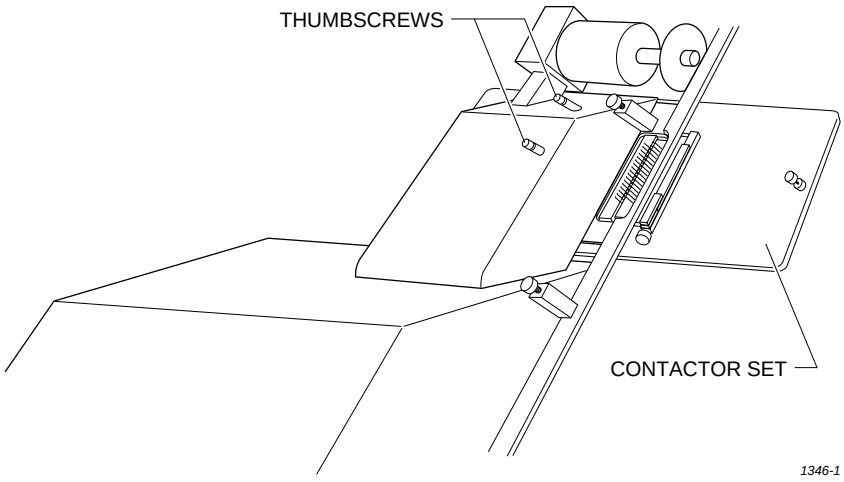
Note: If you purchased AutoSite and the ProMaster 2000 at the same time, each contactor set sent with the 2000 is ready for use with AutoSite. Go to step 6 to continue with the setup and installation.

If you purchased AutoSite and the 2000 separately (i.e., you are connecting AutoSite to an existing 2000), you must convert each contactor set you have before you can use it with AutoSite.

Follow the steps below to convert a contactor set for use with AutoSite:

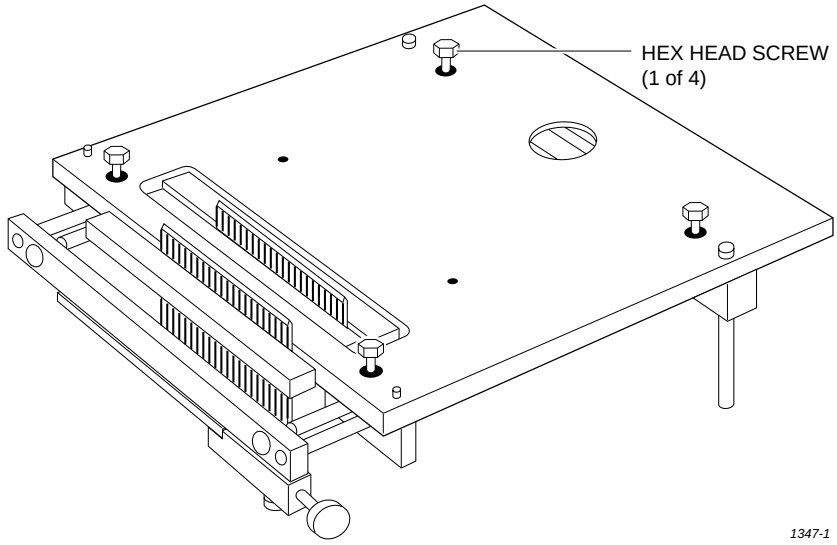
1. Unplug the power cord from the AutoSite control unit.
2. Remove the contactor set from the 2000 by loosening the two thumbscrews shown in Figure 2-4. With the thumbscrews loosened, lower the contactor set from the 2000.

Figure 2-4
Removing the Contactor Set from the ProMaster 2000



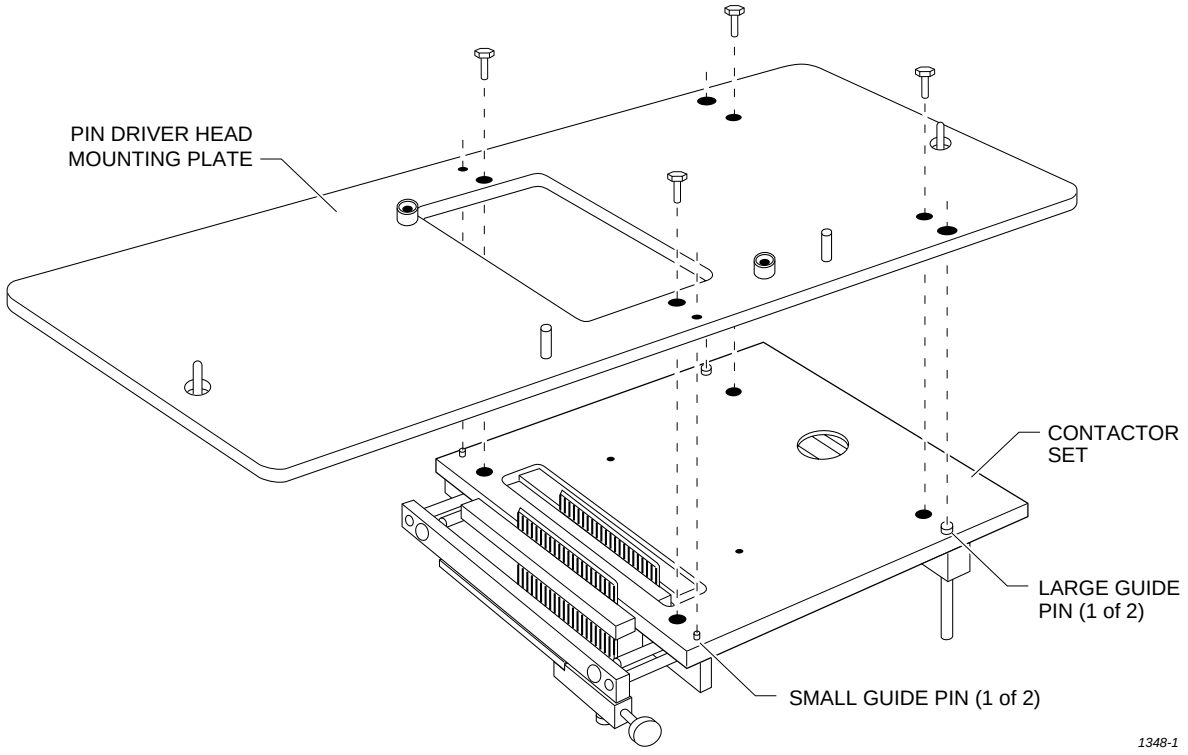
3. Using a 5/32-inch hex driver, remove the four hex head screws shown in Figure 2-5. Discard these screws; you will not need them later.

Figure 2-5
Removing Four Hexhead Bolts from the Contactor Set



4. Position the pin driver head mounting plate on the contactor set as shown in Figure 2-6. Make sure the guide pins on the contactor set align with the proper guide holes on the pin driver head mounting plate.

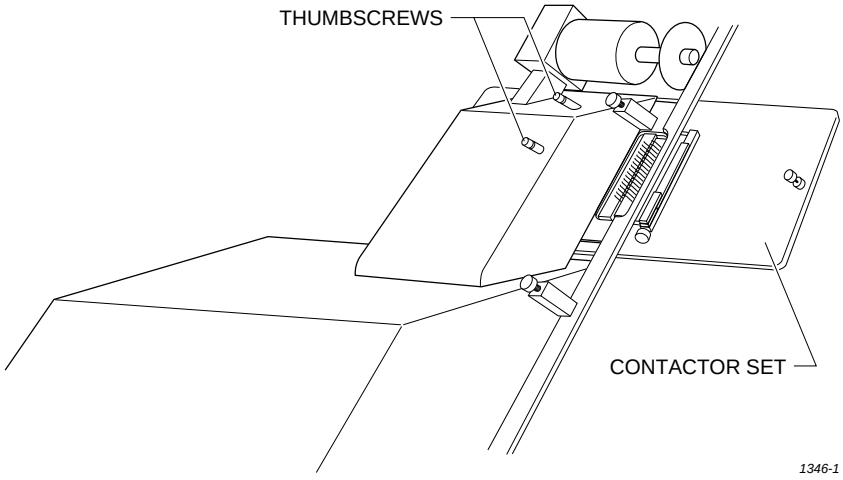
Figure 2-6
Attaching the Contactor Set to the Mounting Plate



5. Using a 5/32-inch hex driver, secure the contactor set to the pin driver head mounting plate with the four hex head screws provided.
6. Position the pin driver head mounting plate up to the 2000. Finger tighten the thumbscrews on the 2000 to secure the pin driver head mounting plate to the 2000.

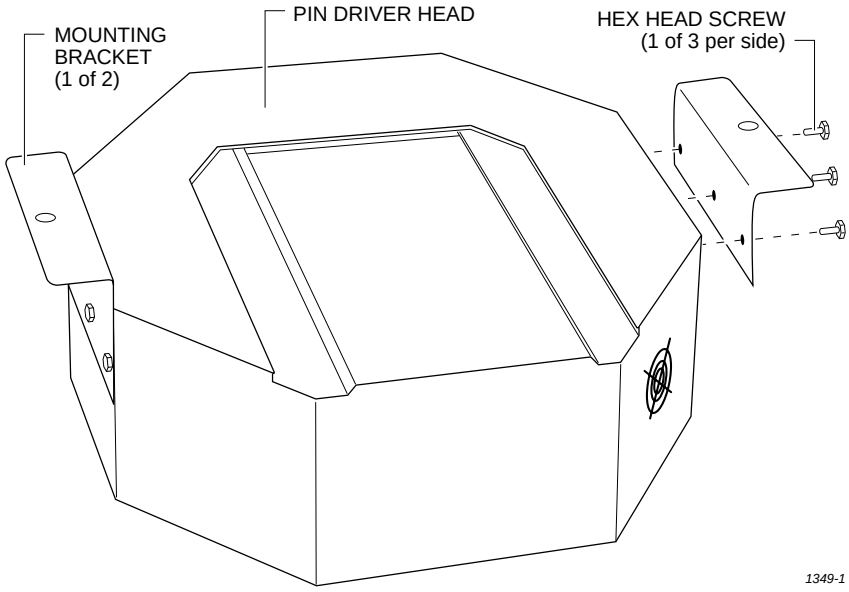
CAUTION: Do not touch or damage the gold pins on the contactor set.

Figure 2-7
Attaching the Pin Driver Head Mounting Plate to the ProMaster 2000



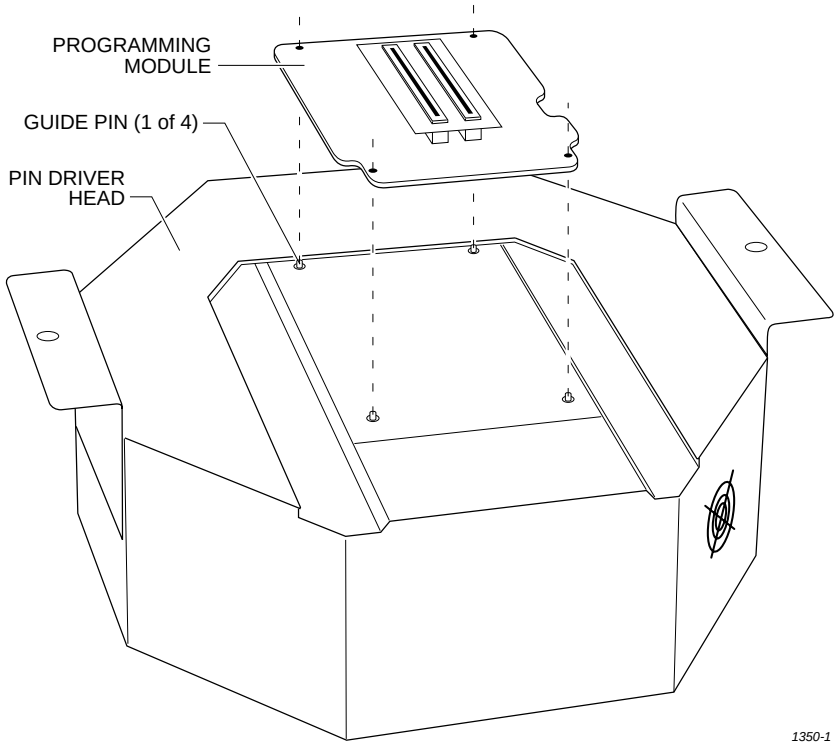
7. Using a 5/32-inch hex driver, attach the two mounting brackets to the pin driver head with the six hex head screws provided. Position the mounting brackets as shown in Figure 2-8.

Figure 2-8
Attaching the Mounting Brackets to the Pin Driver Head



8. Select a programming module that matches the contactor set you attached to the pin driver head mounting plate in steps 4 and 5. As shown in Figure 2-9, set the programming module onto the pin driver head, making sure the guide pins on the pin driver head line up with the guide holes in the programming module.

Figure 2-9
Aligning the Programming Module on the Pin Driver Head



9. As shown in Figure 2-10, slide the clamp ring onto the pin driver head, securing the programming module in place.

CAUTION: You may have to push down on the programming module while sliding the clamp ring onto the pin driver head.

Do not use the device socket on the programming module as a leverage point. You can damage the device socket by applying any sort of force to it.

You will feel and hear a "click" from the clamp ring when the programming module is properly secured to the pin driver head.

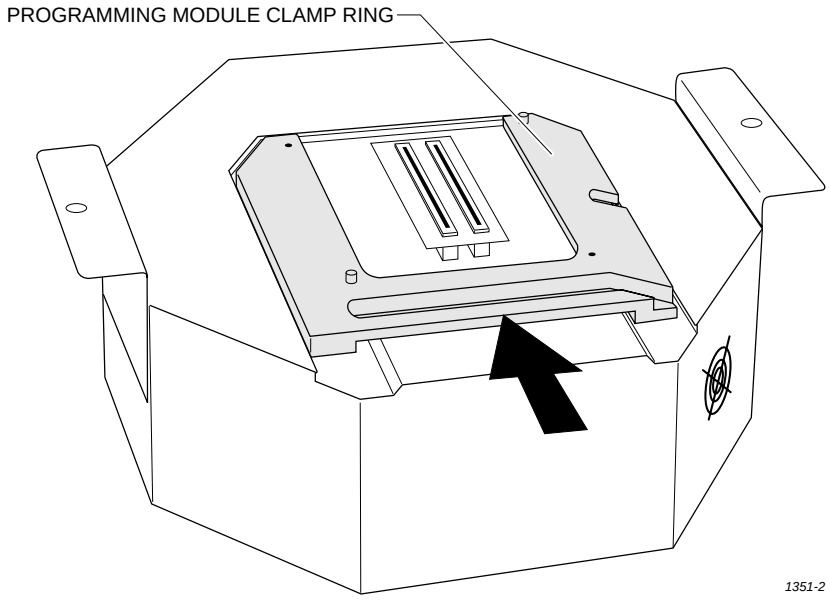


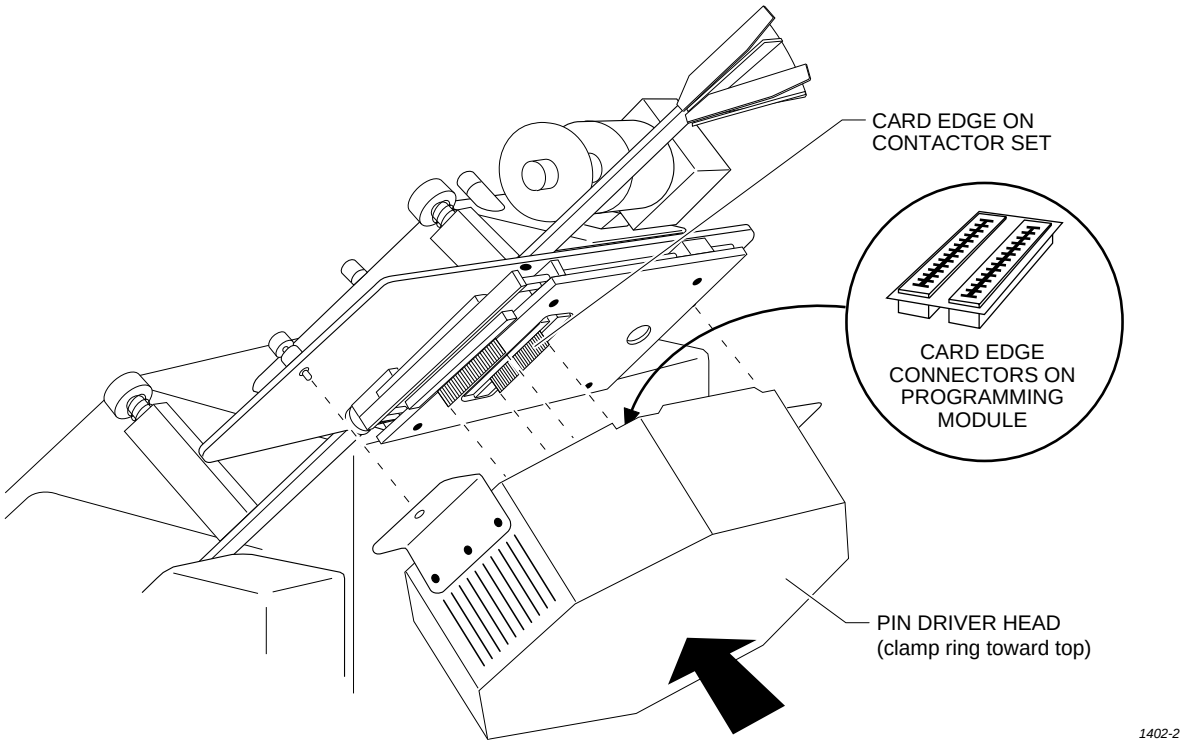
Figure 2-10
Securing a Programming Module to the Pin Driver Head

Attaching the Pin Driver Head

Connect the pin driver head to the 2000 as follows:

10. Position the pin driver head against the 2000 so the handle on the clamp ring points toward the top of the 2000.
11. Align the card edge on the contactor set with the card edge connectors on the programming module. Gently push the pin driver head onto the 2000.

Figure 2-11
Securing the Pin Driver Head to the ProMaster 2000



12. Tighten the thumbscrews on the pin driver head mounting plate, securing the pin driver head to the Handler. You might have to use a flatblade screwdriver to finish tightening the thumbscrews.

CAUTION: To prevent damage to the edge connectors, and to ensure solid contact, we suggest you alternate tightening the left and right thumbscrews until the pin driver head is completely fastened to the handler.

13. *(For control units and pin driver heads without connector brackets at ports J1 and J2)* Connect the 50-pin cable, and the 68-pin cable to the AutoSite control unit and the pin driver head. The 50-pin cable and 68-pin cable and the ports to which they connect are shown in Figures 1-1 and 1-3.

(For control units and pin driver heads with connector brackets at ports J1 and J2.) Remove the two screws that hold the connector shell together at the unconnected end of the 50-pin cable. Plug the 50-pin cable into the appropriate port on the pin driver head and fasten it to the connector bracket by aligning the holes in the bracket with the holes on the connector shell and reinserting the screws through the holes. Tighten the screws.

Repeat the procedure for the 68-pin cable.

Checking the Installation

When properly connected to the handler, the mounting brackets attached to the pin driver head will be flush against the pin driver head mounting plate.

If the mounting brackets are not flush against the pin driver head mounting plate, detach the pin driver head from the pin driver head mounting plate and go back to step 10.

You are finished connecting AutoSite to your 2000. Go to the section titled "Power Up AutoSite," on page 2-27, to continue with the installation.

Connect AutoSite to a ProMaster 3000 or ProMaster 7000 Handler

Note: The procedure for connecting AutoSite to a ProMaster 3000 is almost the same as the procedure for connecting AutoSite to a ProMaster 7000. Differences between the two procedures will be pointed out at the appropriate times in this section.

This section describes how to install AutoSite in a ProMaster 3000 handler. The installation is divided into four main steps:

- Removing the programmer shelf from the 3000
- Rerouting the optics on the 3000
- Installing the AutoSite control unit in the 3000
- Attaching the AutoSite pin driver head to the 3000

Figure 2-12 is a flowchart that illustrates the general flow of the installation procedures contained in this section.

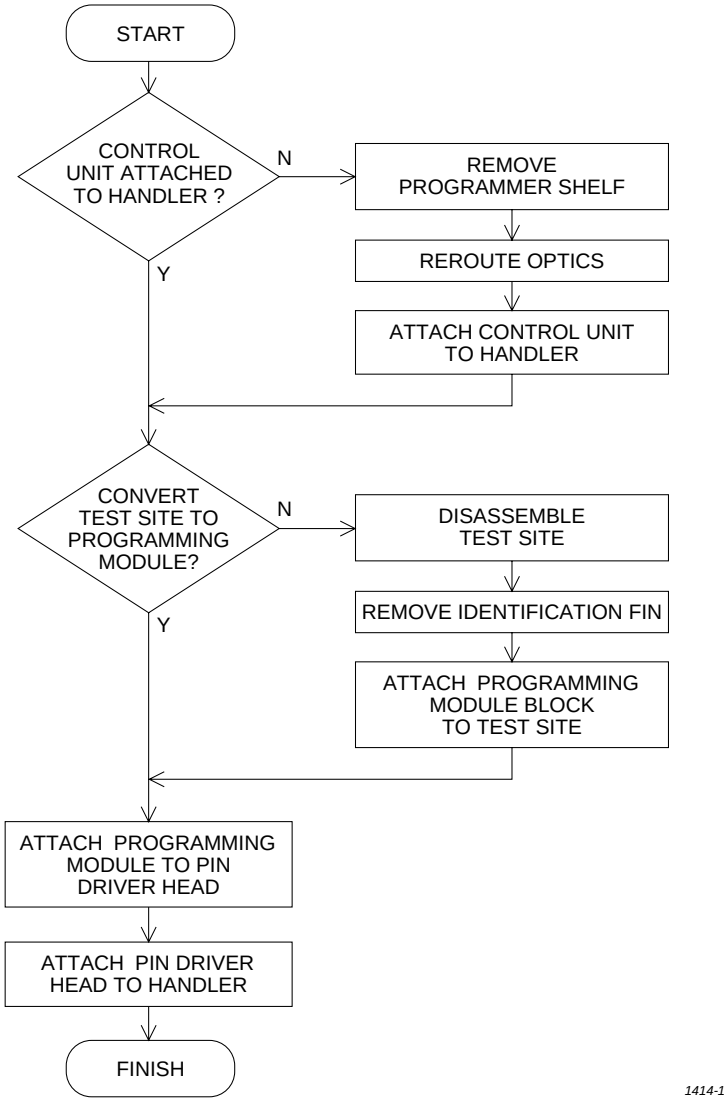
Safety Information

This information is provided as a supplement to the Safety Summary at the beginning of this manual.

The circuitry housed inside the pin driver head and the control unit, and the devices AutoSite programs are static sensitive and can be damaged by electrostatic discharge (ESD). To help minimize the effects of ESD, we suggest you wear an antistatic wrist strap while you follow the procedures described in this section.

For best performance, the antistatic wrist strap should be connected to a properly grounded antistatic workstation and the wrist strap should contain a 1MΩ (minimum) to 10MΩ (maximum) isolating resistor.

Figure 2-12
Flowchart of the Installation Process for the ProMaster 3000 or the ProMaster 7000



Before You Begin

Depending on when you purchased your ProMaster 3000 (or ProMaster 7000) and AutoSite, you might not have to install AutoSite into your handler. All new ProMaster 3000 and ProMaster 7000 handler systems come with the AutoSite programmer installed. If this is the case, skip this section and continue to the section titled "Connect the Pin Driver Head," on page 2-21.

Some older ProMaster 3000 and ProMaster 7000 handlers were sold without the AutoSite, in which case you have to install the AutoSite control unit into your handler. In this case, continue with the section titled "What You Need."

Note: All directional references—front, back, left, right, up, and down—are as if you are looking at the front of the handler.

What You Need

In addition to the contents of the Installation Kit, you will need the following tools and equipment to help you install AutoSite in your handler.

- Programming module
- Grounded wrist strap
- Antistatic workstation
- 5/32-inch hex driver
- 5/64-inch hex driver
- Flat blade screwdriver
- #2 Phillips screwdriver
- Small wire cutters

Remove the Programmer Shelf

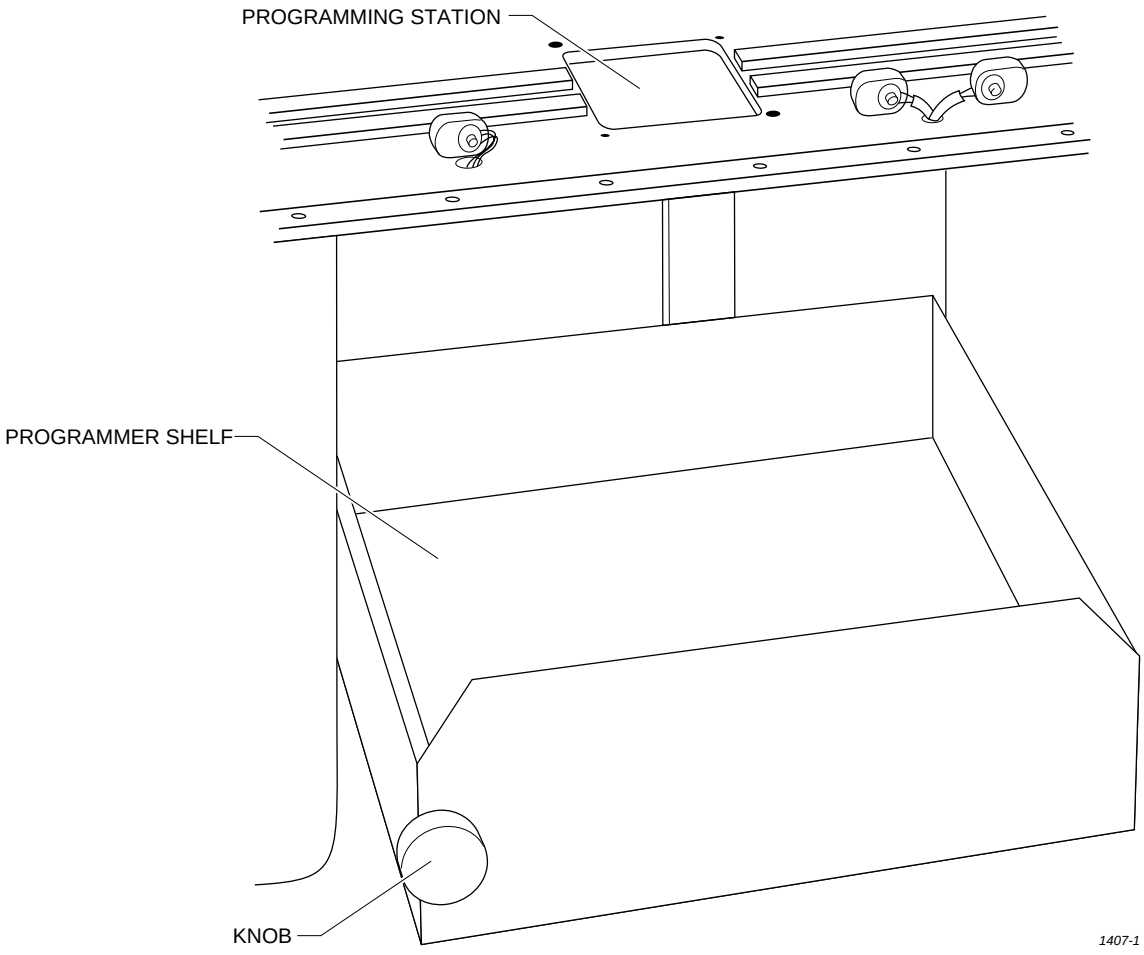
This section describes how to remove the programmer shelf from the 3000. This section does not apply to a 7000. If you are installing AutoSite into a 7000, skip to the section titled "Reroute the Optics," on page 2-17.

CAUTION: In this section, you will be working with the programmer shelf on the 3000. The shelf is supported by an air shock and can spring up when you turn the knob or when you detach the shock from the programmer shelf.

Use caution when adjusting the programmer shelf or working with the air shock.

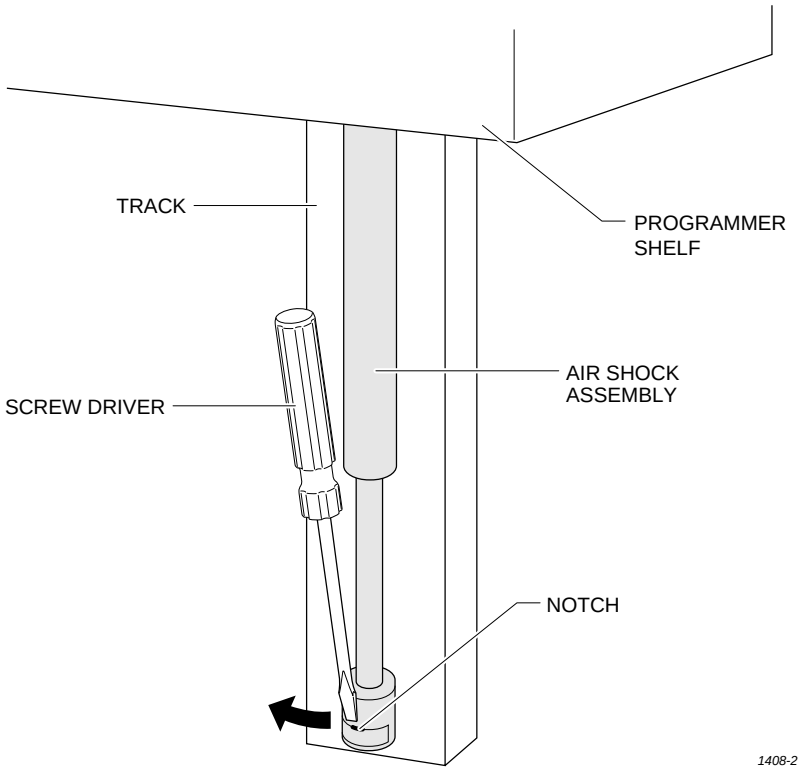
1. Power down the handler and the programmer.
2. Loosen the knob on the programmer shelf and push the shelf to its bottom-most position. Tighten the knob to lock the shelf in place.
3. Remove any programmer, such as an AutoSite, from the programmer shelf in the handler.
4. Remove any tests sites, programming fixtures, or performance boards from the programming station on the handler.
5. Remove the metal underplate from the 3000 by lifting the underplate up and sliding it toward you. Discard the metal underplate; you will not need it later.
6. Loosen the knob on the programmer shelf and raise the shelf to its topmost position. Tighten the knob to lock the shelf in place.

Figure 2-13
Lowering the Programmer Shelf



7. Insert a flat blade screwdriver into the notch shown in Figure 2-14. Disconnect the bottom of the air shock by twisting the blade of the screwdriver and pulling the bottom of the air shock toward you.

Figure 2-14
Disconnecting the Air Shock



8. Loosen the knob on the programmer shelf and push the programmer shelf to its bottom-most position.
9. Using a 5/64-inch hex driver, remove the three hex head screws on the right side of the back of the programmer shelf. Discard these screws; you will not need them later.

Note: When you remove the last of the three screws, the shelf guide and air shock assembly will slide off the track.

10. After you have removed all three screws, remove the shelf guide from the track. Set the shelf guide and the air shock aside; you will need them later.
11. Remove the knob and shaft from the programmer shelf by turning the knob counterclockwise. Discard the knob and shaft; you will not need them later.

12. Using a 5/32-inch hex driver, remove the eight hex head screws on the left side of the programmer shelf. Set these screws aside; you will need them later.

Note: Support the shelf before you remove the last few screws. The shelf will fall from the 3000 when you remove the last screw.

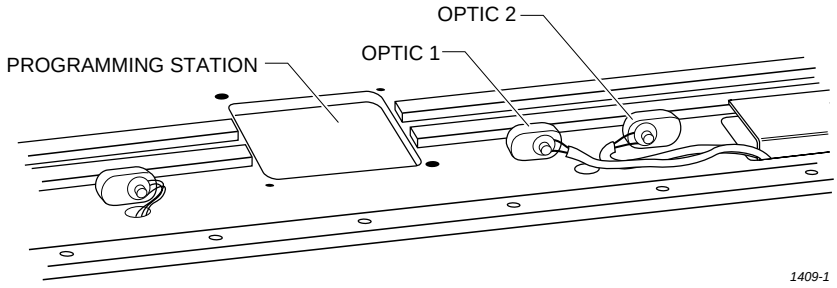
13. After you have removed all eight screws, remove the programmer shelf and discard it. You will no longer need the programmer shelf.

Reroute the Optics

Follow the steps below to reroute the optics on the 3000 (or 7000):

1. Locate the two optics shown in Figure 2-15.

Figure 2-15
Rerouting Two Optics



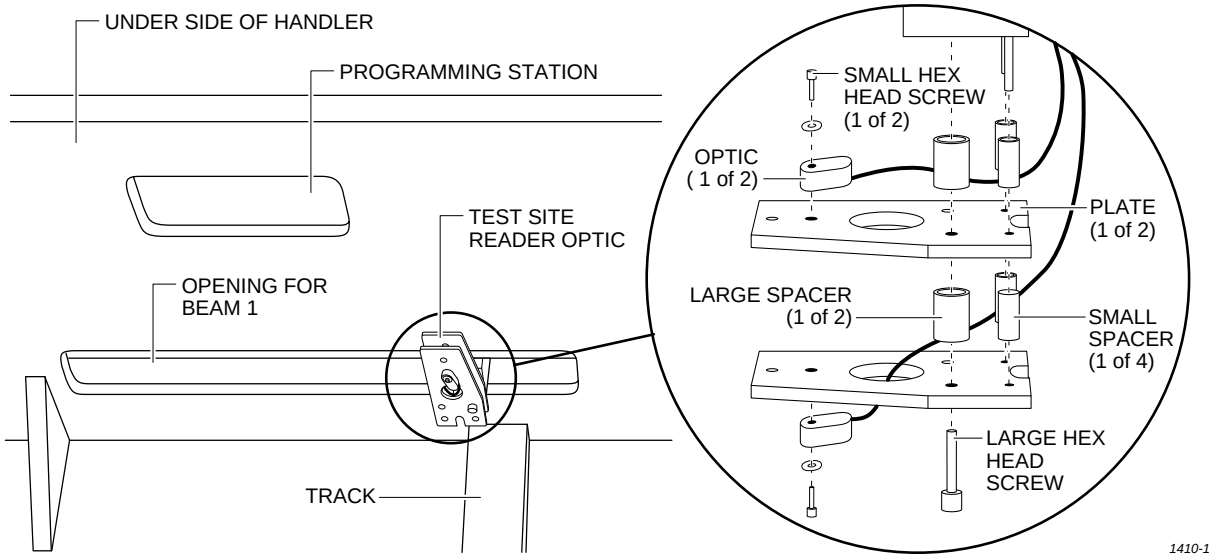
2. Working with one optic at a time, follow the steps below:
   a. Trace the optic wire back to its connector.
   b. Disconnect the connector.
   c. Feed the optic wire back through the hole so the optic wire is above the programming track.
   d. Feed the optic wire down through the new opening.
   e. Reconnect the optic wire to the connector.

Note: You may have to cut a cable tie to free up enough of the optic wire to allow you to reroute the optic through the new opening.

3. Repeat the procedure described in step 2 with the other optic shown in Figure 2-15.
4. Using a cable tie, secure the rerouted optic wires to an adjacent wire bundle.

5. Locate the test site reader optic, which is shown in Figure 2-16.

Figure 2-16
The Location of the Test Site Reader Optic



6. Using a 5/64-inch hex driver, remove the lower optic from the lower plate. Discard the hex head screw and the washer; you will not need them later.
7. Using a hex driver, loosen, but do not remove, the large hex screw shown in Figure 2-16.
8. Hold the lower plate in place while you remove the hex screw from the handler. When you remove the screw, remove the plate and the three lower standoffs from the handler. Discard the lower plate and the lower standoffs; you will not need them later.
9. Remove the upper plate and the three upper standoffs from the handler. Discard the upper standoffs; you will not need them later.
10. Using a 5/64-inch hex driver, remove the upper optic from the upper plate. Discard the upper plate, the hex head screw, and the washer; you will not need them later.
11. Using a cable tie, secure the two optic read heads together.
12. Make certain that you secure the optics so they do not interfere with the full travel of beam 1 to both ends of its run. Push beam 1 all the way to the right and left to check the optics do not interfere with the full travel of the beam.

Install the Control Unit

Follow the steps below to install the control unit into the 3000. This section does not apply to a 7000. If you are installing AutoSite into a 7000, set the control unit in the programmer shelf and continue with the section titled "Convert a a Test Site to a Programming Module."

CAUTION: In this section, you will be working with the programmer shelf on the 3000. The shelf is supported by an air shock and can spring up when you turn the knob or when you detach the shock from the programmer shelf.

Use caution when adjusting the programmer shelf or working with the air shock.

1. Using the eight new 5/32-inch screws supplied in the installation kit, fasten the new control unit mounting plate to the handler. Position the mounting plate so the beveled edge points to the left.

Note: For best results, we suggest that you fasten one screw half way. Then, lift the mounting plate up a few inches and fasten the remaining seven screw.

2. Insert the threaded end of the new, shorter shaft into the control unit mounting plate. Thread the shaft into the shelf lock, which is visible through the oval cutout in the mounting plate.
3. Position the conversion plate onto the shelf guide so the three-holed side of the plate lines up with the three holes in the shelf guide. Also, check that the two-holed side of the conversion plate is on the same side of the shelf guide as the air shock. Finally, make sure the three counter-sunk holes on the conversion plate are facing away from the air shock.
4. Using a #2 Phillips screwdriver and the three flathead screws provided, fasten the conversion plate to the shelf guide.

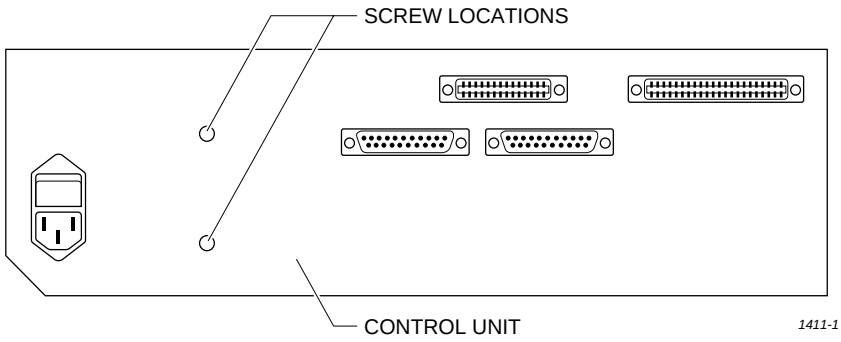


Figure 2-17
The Location of the Two Conversion Plate Screw Holes

5. Using a 5/64-inch hex driver and the two hex head screws provided, fasten the conversion plate onto the back of the AutoSite control unit.
6. Slide the control unit and the shelf guide onto the track. Make sure the bearings on the shelf guide are properly positioned on the track.
7. Using a #2 Phillips screwdriver and the four flathead screws provided, fasten the left side of the control unit to the mounting plate.
8. Slide the control unit up and down the track, checking for proper alignment and installation.
9. Loosen the knob on the control unit and raise the control unit to its topmost position.
10. Reconnect the air shock to the 3000.
11. Slide the control unit up and down the track, checking for proper installation of the air shock.
12. Loosen the knob on the control unit and lower the control unit to its bottom-most position.
13. Install the new metal underplate on the 3000.

Convert a Test Site to a Programming Module

If you are installing AutoSite into an existing handler (i.e., you did not buy the handler and AutoSite at the same time), you will need to convert your existing test sites into AutoSite compatible programming modules.

If you purchased AutoSite and the 3000 (or 7000) at the same time, each programming module sent with the 3000 is ready for use with AutoSite. Go to the section titled “Connect the Pin Driver Head,” on page 2-21, to continue with the setup and installation.

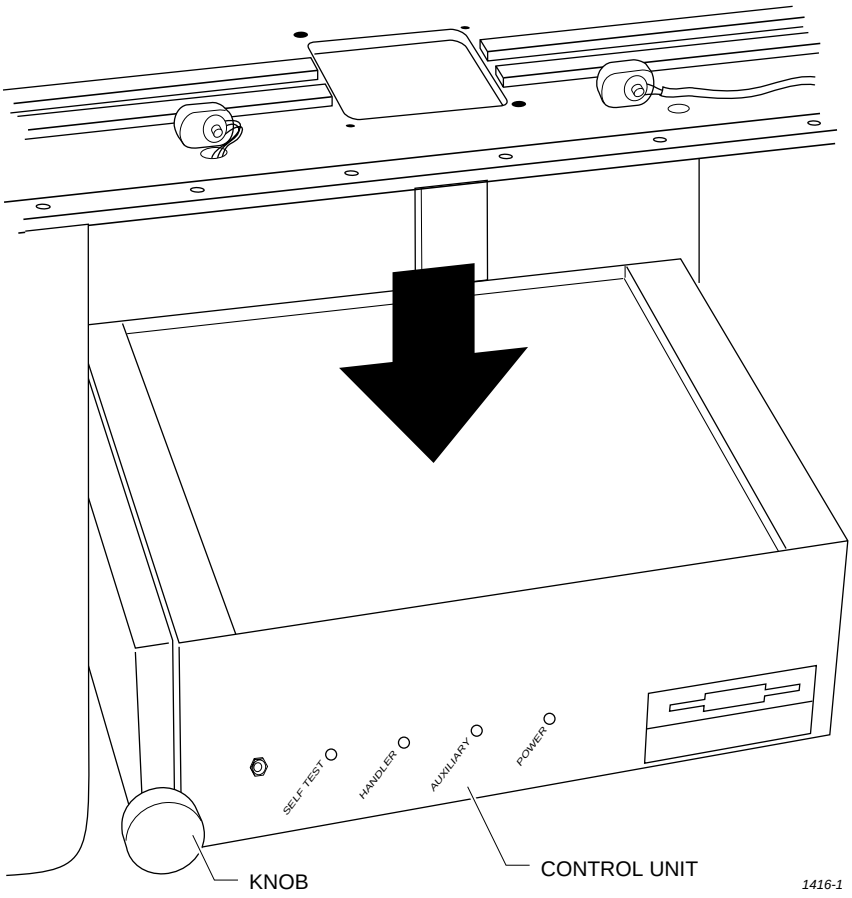
For more information, see the documentation supplied with the Conversion Kit.

Connect the Pin Driver Head

Connect the AutoSite pin driver head to a ProMaster 3000 handler as follows:

1. Loosen the knob and move the control unit down, giving you more room to work.

Figure 2-18
Lowering the Control Unit



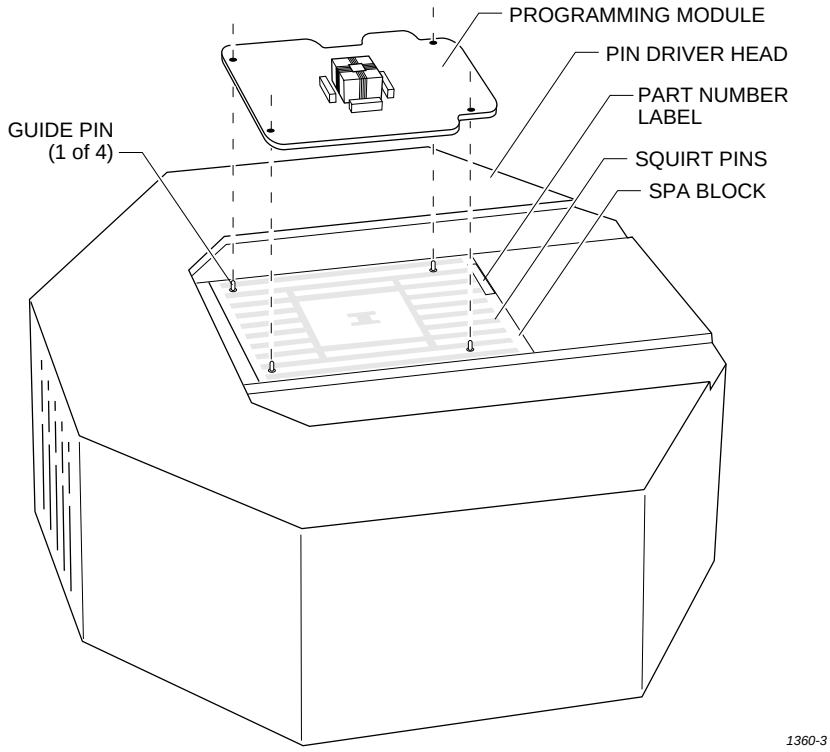
2. Remove any test site from the programming station on the 3000.
3. Verify that the 50-pin cable and the 68-pin cable are connected and properly fastened to the AutoSite control unit. Do not connect the other end of the 50-pin and 68-pin cables yet. The 50-pin cable and 68-pin cable and the ports they connect to are shown in Figures 1-1 and 1-3.
4. *(For control units without connector brackets at ports J1 and J2.)* The 50-pin and 68-pin cables click when properly connected.
5. Connect a 25-pin RS-232C serial cable to the Handler port on the AutoSite control unit. Refer to Figure 1-3 for the location of the Handler port.
6. Connect the other end of the serial cable to a serial port on the PC.

Note: See the section in this chapter titled "More About Cables" if you are using a serial port with nonstandard pinouts or if you want to build your own serial cable.

7. If the clamp ring is installed on the pin driver head, slide the clamp ring off and set it aside; you will need it later. The clamp ring is shown in Figure 2-10.

8. As shown in Figure 2-19, set a programming module onto the pin driver head, making sure the guide pins on the pin driver head line up with the guide holes in the programming module.

Figure 2-19
Aligning a Programming Module on the Pin Driver Head



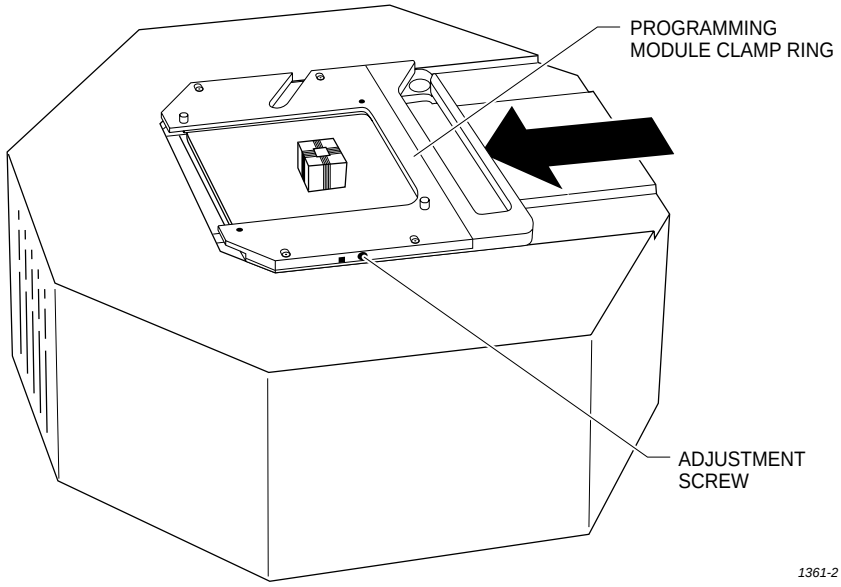
9. As shown in Figure 2-20, slide the clamp ring onto the pin driver head, securing the new programming module in place.

CAUTION: You may have to push down on the programming module while sliding the clamp ring onto the pin driver head.

Do not use the device socket on the programming module as a leverage point. You can damage the device socket by applying any sort of force to it.

You will feel and hear a "click" from the clamp ring when the programming module is properly secured to the pin driver head.

Figure 2-20
Securing a Programming Module to the Pin Driver Head



10. *(For pin driver heads without connector brackets at ports J1 and J2.)* Connect the other end of the 50-pin and 68-pin cables to the pin driver head. The 50-pin and 68-pin cables click when properly connected to the pin driver head.

    (For pin driver heads with connector brackets at ports J1 and J2.) Remove the two screws that hold the connector shell together at the unconnected end of the 50-pin cable. Plug the 50-pin cable into the appropriate port on the pin driver head and fasten it to the connector bracket by aligning the holes in the bracket with the holes on the connector shell and reinserting the screw through the holes. Tighten the screws.

    Repeat the procedure for the 68-pin cable.

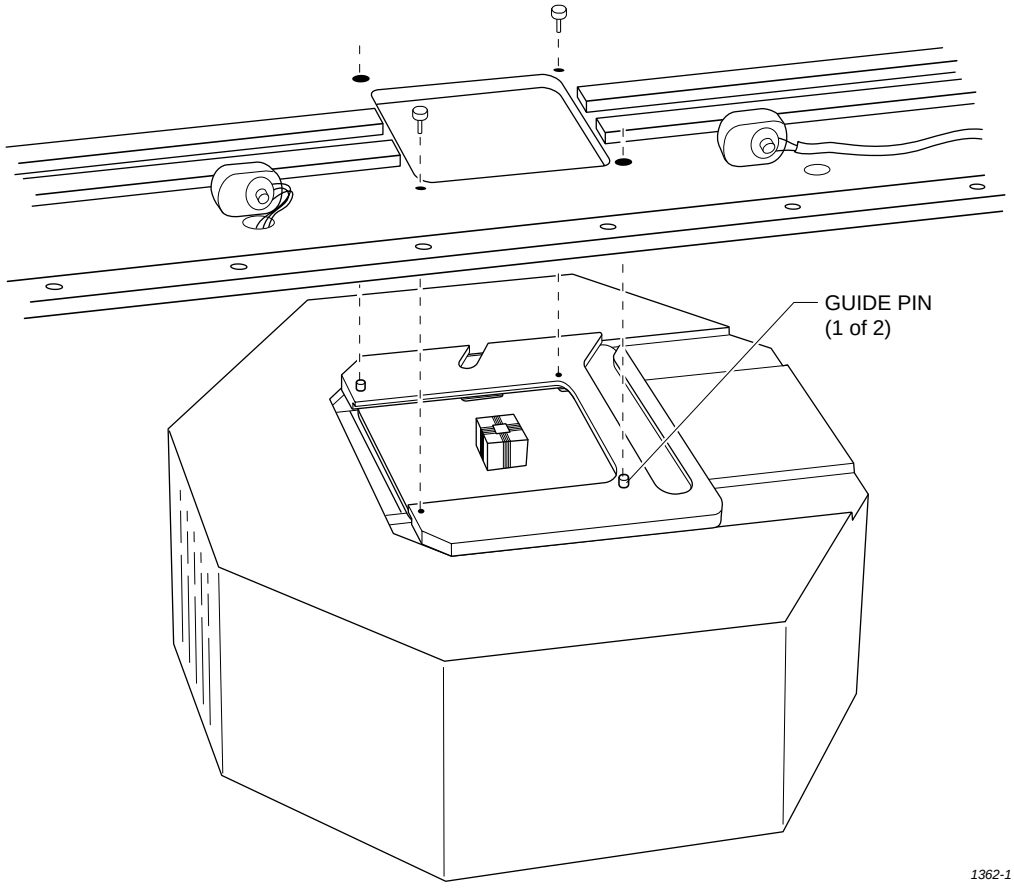
11. Lift up the hood on the handler.

Note: Read the next two steps before proceeding with either step.

12. Position the pin driver head below the programming station with the clamp ring to the right. Line up the guide pins on the programming module with the guide pin holes in the handler.

13. Using the two pin driver head thumbscrews provided, secure the pin driver head to the handler. See Figure 2-21 for the location of the holes for the thumbscrews.

Figure 2-21
Securing the Pin Driver Head to the Handler



14. Lower the hood on the handler.

Checking the Installation

When properly connected to the handler, the clamp ring will be flush against the handler.

If the clamp ring is not flush against the handler, remove the pin driver head from the handler and go back to step 8.

You are finished connecting AutoSite to your ProMaster 3000 (or ProMaster 7000). Go to the section titled “Power Up AutoSite” to continue with the installation.

Connect AutoSite to a Non-ProMaster Handler

This section describes how to connect AutoSite to a handler. The installation is divided into two main steps:

- Attach the AutoSite control unit to the handler
- Attach the AutoSite pin driver head to the handler

Before you begin, you must set up your handler and install your handler control software. Refer to the documentation that came with your handler and handler control software for information.

What You Need

In addition to the contents of the Installation Kit, which is supplied by the handler manufacturer, you will need the following to connect AutoSite to a handler:

- Grounded wrist strap

Note: Contact the handler manufacturer if you did not receive hardware for connecting AutoSite to your handler.

Safety Information

This information is provided as a supplement to the Safety Summary at the beginning of this manual.

The circuitry housed inside the pin driver head and the control unit, and the devices AutoSite programs are static sensitive and can be damaged by electrostatic discharge (ESD). To help minimize the effects of ESD, we suggest you wear an antistatic wrist strap while you follow the procedures described in this section.

For best performance, the antistatic wrist strap should be connected to a properly grounded antistatic workstation and the wrist strap should contain a 1MΩ (minimum) to 10MΩ (maximum) isolating resistor. We suggest you connect your antistatic wrist strap to the grounding terminal on the front of the AutoSite control unit. The grounding terminal is shown in Figure 1-2.

Attaching the Control Unit

Connect the AutoSite control unit to a handler as follows:

1. Verify that the 50-pin cable and the 68-pin cable are connected and properly fastened to the AutoSite control unit. Do not connect the other end of the 50-pin and 68-pin cables yet. The 50-pin cable and 68-pin cable and the ports to which they connect are shown in Figure 1-1 and Figure 1-3.

   (For control units without connector brackets at ports J1 and J2.) The 50-pin and 68-pin cables click when properly connected.

2. If you will be controlling AutoSite from a PC, connect a 25-pin RS-232C serial cable to the Handler port on the AutoSite control unit. Refer to Figure 1-3 for the location of the Handler port.

   Connect the other end of the serial cable to a serial port on the PC.

Note: See the section in this chapter titled "More About Cables" if you are using a serial port with nonstandard pinouts or if you want to build your own serial cable.

3. Depending on the type of handler you are using, you may be able to attach the control unit to the handler. Refer to the documentation provided with your handler for information on connecting the AutoSite control unit to the handler.

Note: Because of space limitations and other restrictions, you may not be able to attach AutoSite to some handlers. In this case, simply place the control unit in a convenient position near the handler. See the handler documentation for more information.

Connect the Pin Driver Head

Connect the AutoSite pin driver head as follows:

Note: The four standoffs located around the SPA block on the pin driver head (as shown in Figure 2-22) may cause interference if the AutoSite is installed on a non-ProMaster handler. If interference occurs, remove the standoffs using a 1/4-inch nut driver, or a 1/4-inch open-end box wrench or the equivalent.

4. *(For pin driver heads without connector brackets at ports J1 and J2.)* Connect the other end of the 50-pin and 68-pin cables to the pin driver head. The 50-pin and 68-pin cables click when properly connected to the pin driver head.

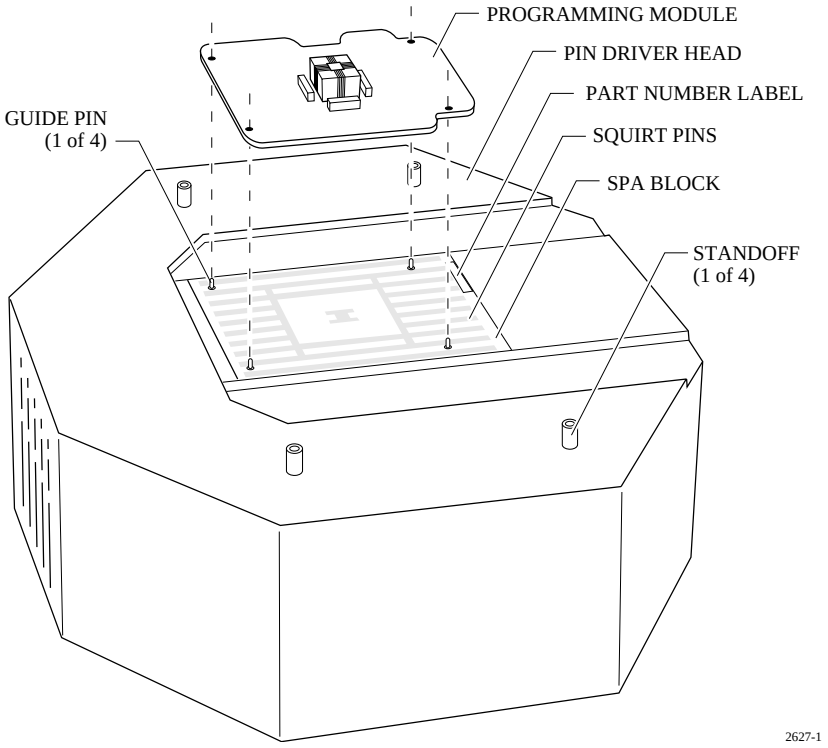
   (For pin driver heads with connector brackets at ports J1 and J2.) Remove the two screws that hold the connector shell together at the unconnected end of the 50-pin cable. Plug the 50-pin cable into the appropriate port on the pin driver head and fasten it to the connector bracket by aligning the holes in the bracket with the holes on the connector shell and reinserting the screws through the holes. Tighten the screws.

   Repeat the procedure for the 68-pin cable.

5. Refer to the documentation shipped with your handler for information on installing a programming module. We suggest you keep the following in mind while installing a programming module:
   - Make sure the handler is idle
   - Wear a properly grounded antistatic wrist strap while working with the programming module and the pin driver head
   - Do not touch the gold pins that are exposed when you remove a programming module from the pin driver head
   - Do not block the fan on the side of the pin driver head
   - Make sure the guide pins on the pin driver head line up with the guide holes in the programming module. See Figure 2-22 for an example.
   - Do not use the device socket or connectors on the programming module as a leverage point.

Note: The programming module you install may not look like the programming module shown in Figure 2-22.

Figure 2-22
Aligning a Programming Module to the Pin Driver Head



Power Up AutoSite

Safety Information

This information is provided as a supplement to the Safety Summary at the beginning of this manual.

The circuitry housed inside the pin driver head and the control unit, and the devices AutoSite programs are static sensitive and can be damaged by electrostatic discharge (ESD). To help minimize the effects of ESD, we suggest you wear an antistatic wrist strap while you follow the procedures described in this section.

For best performance, the antistatic wrist strap should be connected to a properly grounded antistatic workstation and the wrist strap should contain a 1MΩ (minimum) to 10MΩ (maximum) isolating resistor. We suggest you connect your antistatic wrist strap to the grounding terminal on the front of the AutoSite control unit. The grounding terminal is shown in Figure 1-2.

About the Programmer Disks

The disk labeled **Boot Files** contains the AutoSite operating system. The disk must be in the disk drive when you boot AutoSite.

Any disk labeled **System Files** contains the system files necessary for many operations after the programmer has booted. If prompted to insert the System Files disk, you can insert any one of the disks labeled “System Files.”

The **Algorithm** disks contain all the programming algorithms for the devices currently supported by AutoSite.

If you have a hard drive (also known as a Mass Storage Module, or MSM) installed in your AutoSite, you can boot directly from the hard drive after you install the system and algorithm files on the hard drive. Ensure that no disks are in the floppy drive when you boot up.

Note: Do not attempt to use the Boot Files disk or the Algorithm disks with more than one AutoSite. Each disk is intended to work with only one particular AutoSite.

Once you get AutoSite working, make a backup copy of each disk. The "Finish Up" section of this chapter describes how to make a backup copies of the AutoSite disks.

Power Up AutoSite

To power up AutoSite, follow the steps below.

1. Connect one end of the ac line cord to the ac receptacle on the rear panel of AutoSite and the other end to a properly grounded ac outlet.

   AutoSite contains a switching power supply that configures itself to operate on the proper voltage. The power supply accepts voltages ranging from 90 to 264 Vac and frequencies ranging from 48 to 63 Hz.

WARNING: To ensure proper grounding, and to avoid the hazard of electrical shock, connect AutoSite to ONLY a properly grounded ac outlet.

2. Make sure a programming module (or Base) is installed in AutoSite. Also, make sure the device socket in the programming module (or Base) is empty.

CAUTION: Leaving a device in the socket during powerup could damage the device.

3. Power up the handler.
4. Power up the PC you will use to control AutoSite. Start the programmer control software, such as TaskLink, you will use to control AutoSite or start the handler control software you will use to control AutoSite.
5. If you do not have an MSM, insert the Boot Files disk into the AutoSite disk drive.
6. Power up AutoSite. The power switch is located on the back panel of the control unit, next to the ac receptacle, and is pictured in Figure 1-3.

   When you turn the power switch on, the Power LED should light. If it doesn't, turn AutoSite off, check the power connections, and turn AutoSite on again.

Note: Do not remove the Boot Files disk while either the Self Test or disk drive LED is lit. If you remove the Boot Files disk during powerup, you will need to reboot AutoSite.

Did AutoSite Pass Self-test?

While powering up, AutoSite performs a powerup self-test. AutoSite has completed powerup when the self-test LED and disk drive LED are off.

Note: The powerup self-test takes approximately three minutes on a 44-pin AutoSite and four minutes on an 88-pin AutoSite.

If the self-test LED goes off, the power-up self-test did not detect any system problems. Go to the "Are the Right LEDs Lit?" section.

If the power-up self-test detected anything wrong, the four LEDs on the front panel of the control unit illuminate in different patterns, telling you what the self-test found. The different combinations of blinking LEDs are explained below.

	----------Indicator----------			
Power	**Auxiliary**	**Handler**	**Self-Test**	**Description**
On	Off	Off	On	Self-test in progress. No Error condition.
On	On	Off	Off	Handler port OK; self-test finished. No Error condition.
On	Off	On	Off	Auxiliary and Handler ports OK; self-test finished. No Error condition.
On	On	On	Off	Auxiliary and Handler ports OK; self-test finished. No Error condition.
Off	X	X	Off	Power supply problem
On	Blinking	X	On	CPU or EPROM problem or corrupt ID PAL
Off	X	X	On	Power supply problem
On	Blinking	Off	On	CPU or EPROM problem
On	Off	Blinking	On	Bad system RAM problem
On	Blinking	Blinking	On	Serial port DUART problem
On	Blinking	Blinking	Blinking	Power supply problem. Make sure voltage selector is set correctly.
X	X	X	Blinking	LCA problem

Note: X = don't care condition.

If one or more of the LEDs is blinking after the self-test, there may be a faulty circuit board in AutoSite. Contact Data I/O Customer Support for more information.

Are the Right LEDs Lit?

If AutoSite completed the self-test successfully, the Power LED is lit. Also, if you have equipment connected to the Auxiliary and Handler ports, the corresponding LEDs should be lit. If all the right LEDs are lit, AutoSite has powered up successfully. Go to the "Finish Up" section.

If the Handler LED and/or Auxiliary LED should be lit and are not, check the connections between AutoSite and the connected equipment.

Checking the Connections

Sometimes problems are caused by unconnected cables. Turn AutoSite off and check all of the following:

- Power cords—Are they all plugged into a live outlet and into the equipment?
- Cables—Is each cable that connects AutoSite to another piece of equipment connected properly? Is each cable connected to the proper port?
- Algorithm/System disk—Is the disk inserted properly?
- Programming module—Is there a programming module (or Base) inserted into AutoSite? Is it inserted properly? Is the programming module (or Base) empty? The programming module (or Base) must be empty for AutoSite to boot up properly.

After checking everything described above, reboot AutoSite and go to the beginning of the "Power Up AutoSite" section.

Insert Algorithm Disk

At various times, such as when you select a device, AutoSite accesses the Algorithm disk. You might be prompted to insert a different Algorithm disk than the one currently in the programmer. If so, insert the disk for which you are prompted. For instance, if AutoSite prompts you to `Insert Algorithm Set 3 Disk`, insert the Algorithm disk that contains Algorithm Set 3. If AutoSite prompts you to `Insert Algorithm Disk`, insert any one of the Algorithm disks.

Note: If a Mass Storage Module is installed and has been updated with the current algorithms, you do not need to insert an Algorithm disk in the disk drive.

Finish Up

Establishing Communication

By this point, AutoSite should have completed and passed a power-up self-test. To establish communication with AutoSite, do the following:

1. After AutoSite has powered up, start your terminal or terminal emulator (such as TaskLink) that you are using to communicate with AutoSite. This terminal or terminal emulator must be capable of emulating one of the following terminal types:
   - ANSI 3.64 compatible terminals
   - DEC VT-100 compatible terminals (supported by TaskLink)
   - Qume QVT-101 compatible terminals
   - TELEVIDEO TVI--910 compatible terminals
   - Wyse WY-30 compatible terminals

   If you are using handler control software, consult the documentation to see if it supports one of the terminal types listed above.
2. If you are using terminal emulation software, configure your software to match the following parameters:
   - 9600 baud
   - No parity
   - 8 data bits
   - 1 stop bit

For more information, refer to the documentation supplied with the software you will be using to control AutoSite.

If you are using TaskLink, the communication parameters listed above can be set from the **Options** menu, in the **Programmer Port** dialog box.

TaskLink

If you are controlling AutoSite from TaskLink, make sure TaskLink is running and press **CTRL + F1** to check communication between TaskLink and AutoSite.

If TaskLink responds with `Contact With Programmer Established`, then TaskLink and AutoSite are communicating. If TaskLink cannot contact AutoSite, check the cables connecting AutoSite to the PC and then see the TaskLink documentation for more information.

Note: Unless otherwise noted, all references to TaskLink in this manual refer to the DOS TaskLink product. TaskLink for Windows performs similar operations using different user interface commands. Refer to the ***TaskLink for Windows Getting Started Guide*** *and online Help for additional information.*

Ways to Control AutoSite

Once communication with AutoSite is established, you could control AutoSite in any of the following ways:

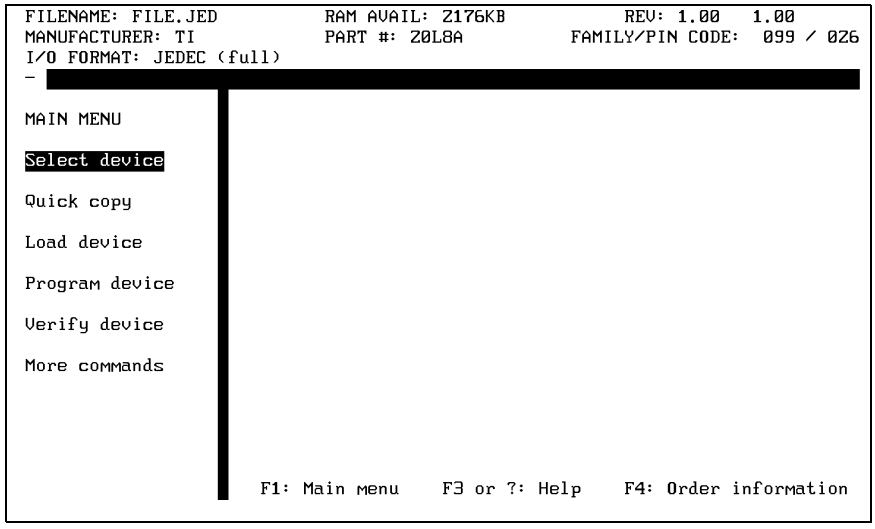
- **Terminal Mode**—This interface is the standard menu system that is built into AutoSite.
- **CRC Mode**—This mode uses Computer Remote Control codes to instruct AutoSite. In order to use this mode, AutoSite must be running in Remote Mode. CRC mode is described in Appendix A of this manual.
- **Custom Software Interface or TaskLink**—This interface is a custom interface or a PC-based menu interface (such as TaskLink, which offers a full menu-driven user interface). For more information about TaskLink, refer to the documentation included with TaskLink or contact Customer Support.

More About Terminal Mode

In most cases, the emulation software must be set to **full screen** terminal mode before you can view **terminal mode**. If you are using TaskLink, start TaskLink and select **Programmer Interface** from the **Utilities** menu to access terminal mode. See your TaskLink documentation for more information.

While in terminal mode, you can access the Main Menu, shown in Figure 2-23, which is the starting point from which you can select commands.

Figure 2-23
The AutoSite Main Menu



Selecting a Command in Terminal Mode

You can select a command while in terminal mode by using one of two methods:

- Type the first letter of the command, or
- Move the cursor to the menu item and press **ENTER**.

If you get lost in the menu structure, press **F1** to return to the Main Menu.

Backing Up the AutoSite Disks

When you have successfully powered up AutoSite, and you have established communication between AutoSite and TaskLink, we suggest you make a backup copy of your AutoSite disks.

If you have access to a DOS-based PC with a 1.44MB disk drive, use the DOS DISKCOPY command to make a copy of your disks.

CAUTION: Make sure you use DISKCOPY and not COPY. The backup must be an exact, bit-for-bit, sector-for-sector copy of the original. Store the backup copy in a safe place.

To use the DISKCOPY command:

1. Insert an AutoSite disk (the "source" disk) into you PC's disk drive.
2. At the DOS prompt, type **diskcopy *a*: *a*:**, where *a* represents the letter of the disk drive into which the AutoSite disk has been inserted, and press ENTER.

   DOS will copy a portion of the "source" disk into RAM. It will then ask for the "target" disk, the disk to which you want to copy the "source" information.
3. Remove the "source" disk and insert a formatted floppy disk into your disk drive. DOS will then copy the information it has stored in RAM onto the "target" diskette.

Note: You may be prompted to switch diskettes more than once.

What to Do Next Time

Next time you power up AutoSite, you probably do not need to follow all the steps outlined in this chapter. Listed below are the normal steps for preparing for another session on AutoSite.

Note: If you have not used AutoSite for a while, or if you suspect AutoSite might have been moved from one area to another, follow the procedure below before you use AutoSite.

To prepare AutoSite for another session, follow the procedure below:

1. Check the power cords and cables between AutoSite and the connected equipment.
2. If you are controlling AutoSite from a PC, make sure it is on and that the controlling software, such as TaskLink, is running.
3. If a Mass Storage Module (hard drive) is not installed, insert the AutoSite Boot Files disk into the disk drive.
4. Select and insert a programming module (or Base) into the pin driver head. Make sure the programming module (or Base) is locked in place.
5. Connect the pin driver head to the handler.

6. Power up AutoSite.
7. Verify that AutoSite and the controlling software are communicating. If you are using TaskLink, press **CTRL** + **F1** to check communication between TaskLink and AutoSite. If TaskLink displays `Contact With Programmer Established`, then TaskLink and AutoSite are communicating.

You are now ready to begin a new session on AutoSite.

More About Cables

This section describes the pin assignments AutoSite uses on its serial ports. Use the information on this page and the following two pages to build serial cables to connect to AutoSite.

SmartPort

AutoSite is compatible with both Data Terminal Equipment (DTE) and Data Communications Equipment (DCE). AutoSite's SmartPort feature automatically toggles between DTE and DCE until a connection is established.

Making Your Own Cable

AutoSite receives commands and sends responses through an RS-232C port using a 25-pin D connector in two possible configurations: either DTE or DCE. The connections are shown in Figure 2-24.

Pin Functions When In DTE Mode

The following table explains the function of the connector pins on the Auxiliary and Handler ports when they are configured as DTE ports.

Pin	Function	Description
1	Ground	Provides a safety ground connection
2	Transmit Data	Carried the transmitted data
3	Receive Data	Carries the received data
4	Request to Send	This line is held high by AutoSite
5*	Clear to Send	A high on this line enables AutoSite to transmit data. (Used for hardware handshaking.) A low inhibits data transmission from AutoSite.
6*	Data Set Ready	This line is held high when the remote source is ready to send or receive data. A low inhibits data transmission from AutoSite.
7	Signal Ground	Provides a reference ground for all signals on the cable.
8*	Data Carrier Detect	This line is held high when the modem detects a carrier. A low on this line inhibits AutoSite from transmitting data.

9-19	No Connection	
20	Data Terminal Ready	This line is pulled high by AutoSite to indicate it is ready to receive data. This line is pulled low to signal the PC to stop sending data. (Used for hardware handshaking.)
21-25	No Connection	

* *If these lines are not connected, AutoSite considers them high and functions normally.*

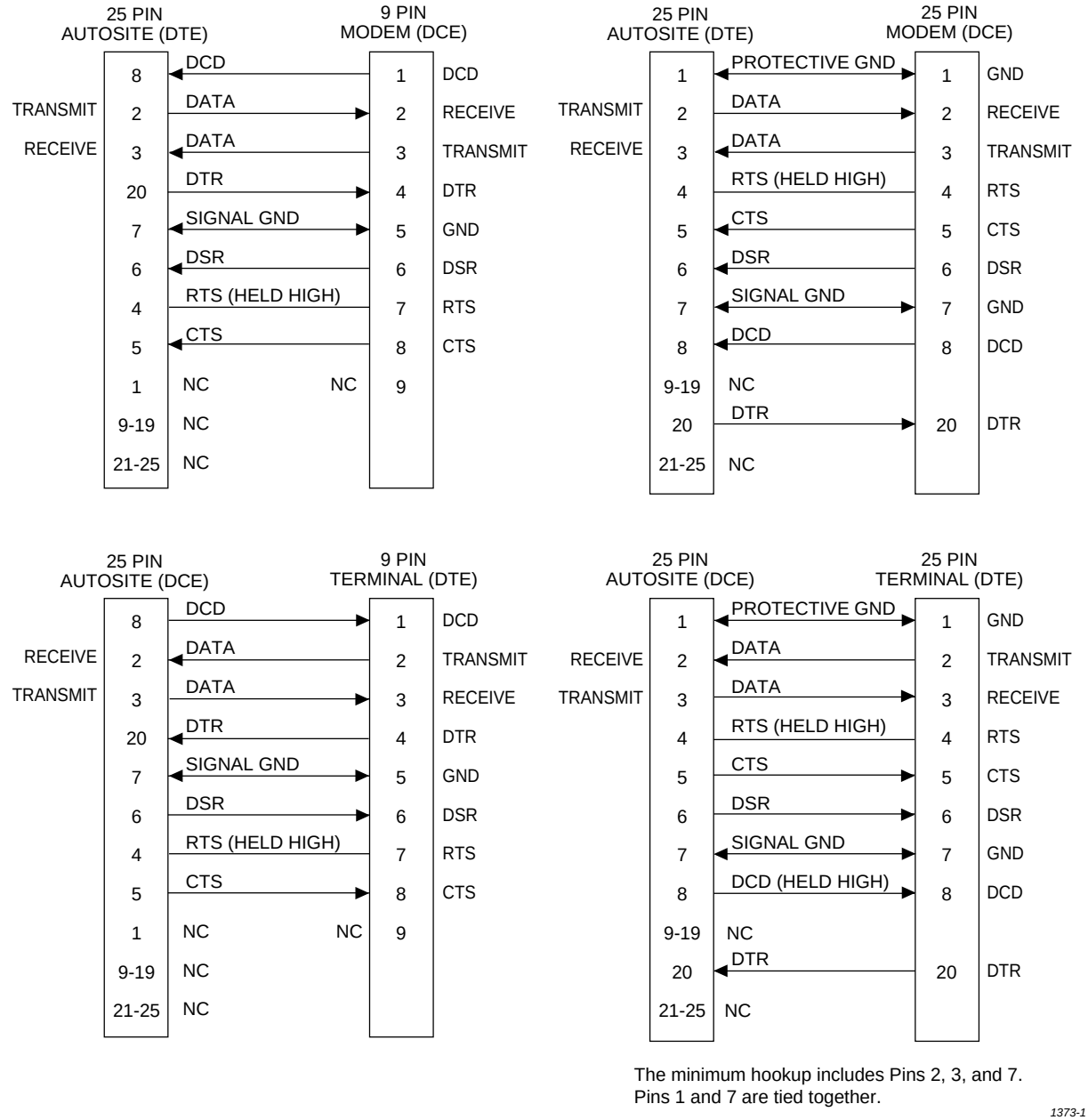
Pin Functions When In DCE Mode

The following table explains the function of the connector pins on the Auxiliary and Handler ports when they are configured as DCE ports.

Pin	Function	Description
1	Ground	Provides a safety ground connection
2	Receive Data	Carries the received data from the DTE device to AutoSite.
3	Transmit Data	Carries the transmitted data from AutoSite to the DTE device.
4	Request to Send	This line is held high by AutoSite.
5	Clear to Send	A high on this line from AutoSite means that it is ready to receive data. (Used for hardware handshaking.)
6	Data Set Ready	This line is held high when AutoSite is ready to transfer data.
7	Signal Ground	Provides a reference ground for all signals on the cable.
8	Data Carrier	This line is held high by AutoSite.
9-19	No Connection	
20*	Data Terminal	A high on this line enables AutoSite to transmit data. (Used for hardware handshaking.) A low inhibits data transmission from AutoSite.
21-25	No Connection	

* *If this line is not connected, AutoSite considers it high and functions normally.*

Figure 2-24
Pin Designations for RS-232C Serial Port Connection



The minimum hookup includes Pins 2, 3, and 7.
Pins 1 and 7 are tied together.

1373-1

3 Operation

This chapter describes how to perform various day-to-day procedures on AutoSite, including the following:

Starting AutoSite

To prepare AutoSite for a session, follow the procedure below.

Note: If you have not used AutoSite for a while, or if you suspect AutoSite might have been moved from one area to another, follow the procedure below before you use AutoSite.

1. Ensure that AutoSite is set up as described in Chapter 2, "Setup and Installation."
2. Select and insert a programming module (or Base) into the pin driver head. Make sure the programming module (or Base) is locked in place.
3. Connect the pin driver head to the handler.
4. Power up AutoSite. (The powerup operation is described in detail on page 2-28.)
5. Verify that AutoSite and the controlling software are communicating.

You are now ready to begin a session on AutoSite.

Changing a Programming Module: Overview

The procedure for changing a programming module depends on the manufacturer of the handler you are using with AutoSite. See the documentation supplied with your handler for information on changing a programming module.

To prevent damage to AutoSite and to make changing programming modules easier, we suggest that you keep the following items in mind while changing a programming module:

- Make sure the handler is idle
- Wear a properly grounded antistatic wrist strap while working with the programming module and the pin driver head
- Do not touch the gold pins exposed when you remove a programming module from the pin driver head
- Do not block the fan on the side of the pin driver head
- If necessary, disconnect the 50-pin and 68-pin cables from the pin driver head to gain better access to the pin driver head
- Do not use the device socket or connectors on the programming module as a leverage point
- Store programming modules in a static safe area when not in use or when removed from the pin driver head

Note: You may need to change hardware on your handler when you change programming modules on AutoSite. See your handler manual for more information.

Changing a Programming Module on a ProMaster 2000

Safety Information

This information is provided as a supplement to the Safety Summary at the beginning of this manual.

The circuitry housed inside the pin driver head and the control unit, and the devices AutoSite programs are static sensitive and can be damaged by electrostatic discharge (ESD). To help minimize the effects of ESD, we suggest you wear an antistatic wrist strap while you follow the procedures described in this section.

For best performance, the antistatic wrist strap should be connected to a properly grounded antistatic workstation and the wrist strap should contain a 1 MΩ (minimum) to 10 MΩ (maximum) isolating resistor.

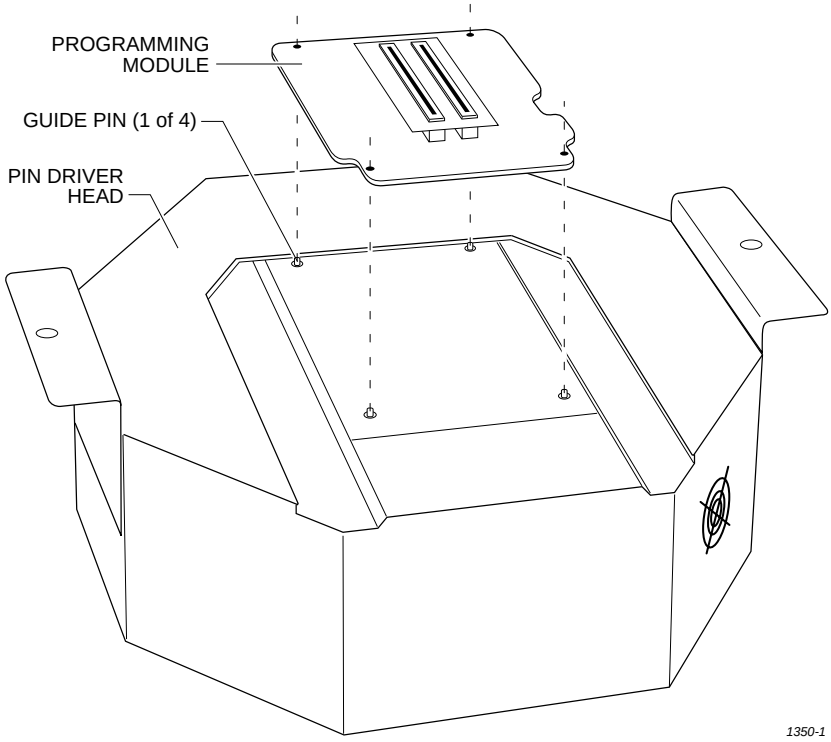
Follow the steps below to change the programming module on a ProMaster 2000.

1. Make sure the handler is idle.
2. Clear all devices from the handler.
3. Loosen the thumbscrews that hold the pin driver head to the pin driver head mounting plate and carefully remove the pin driver head from the pin driver head mounting plate. The thumbscrews are located at each end of the pin driver head mounting plate.
4. Slide the clamp ring off the pin driver head. Set the clamp ring aside; you will need it later.
5. Remove the old programming module. Set the old programming module aside.

CAUTION: Do not touch the pins that are exposed when you remove the programming module.

6. As shown in Figure 3-1, set the new programming module onto the pin driver head, making sure the guide pins on the pin driver head line up with the guide holes in the programming module.

Figure 3-1
Aligning the Programming Module on the Pin Driver Head



Note: If you are using a programming module for the first time, and you purchased the programming module after you purchased your AutoSite, (i.e., you didn't order the programming module and the AutoSite at the same time) you must perform a one-time procedure to unlock the programming module before you can use it.

Continue with the installation. At the appropriate time, a note in this manual will tell you when to unlock the new programming module.

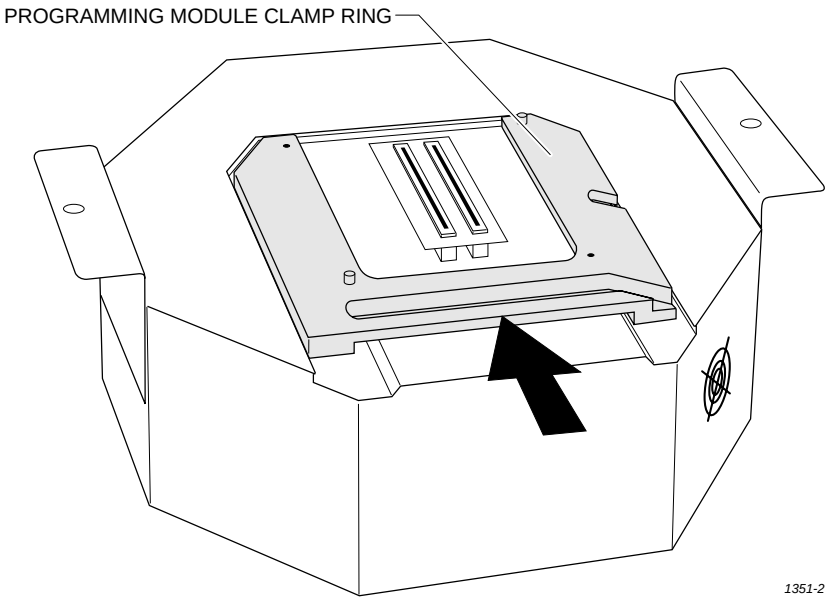
7. As shown in Figure 3-2, slide the clamp ring onto the pin driver head, securing the programming module in place.

CAUTION: You may have to push down on the programming module while sliding the clamp ring onto the pin driver head.

Do not use the device socket on the programming module as a leverage point. You can damage the device socket by applying any sort of force to it.

You will feel and hear a "click" from the clamp ring when the programming module is properly secured to the pin driver head.

Figure 3-2
Securing a Programming Module to the ProMaster 2000



8. Remove the contactor set from the handler by loosening the two thumbscrews shown in Figure 3-3. Set the contactor set aside.

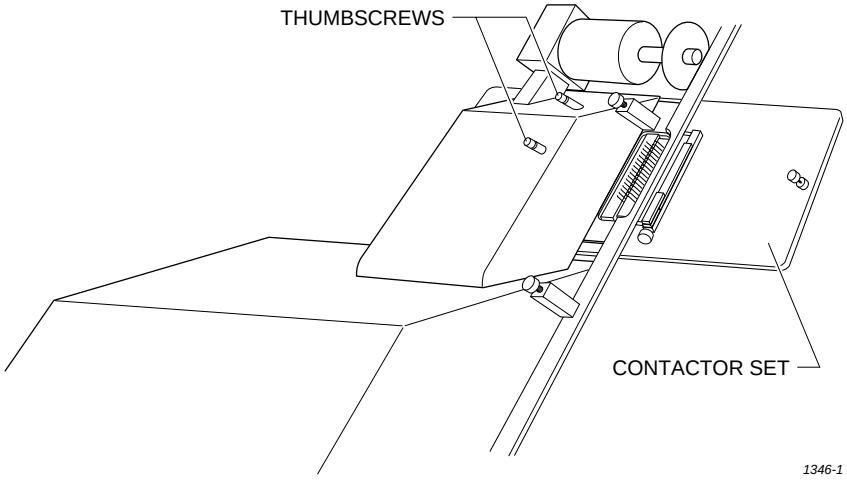
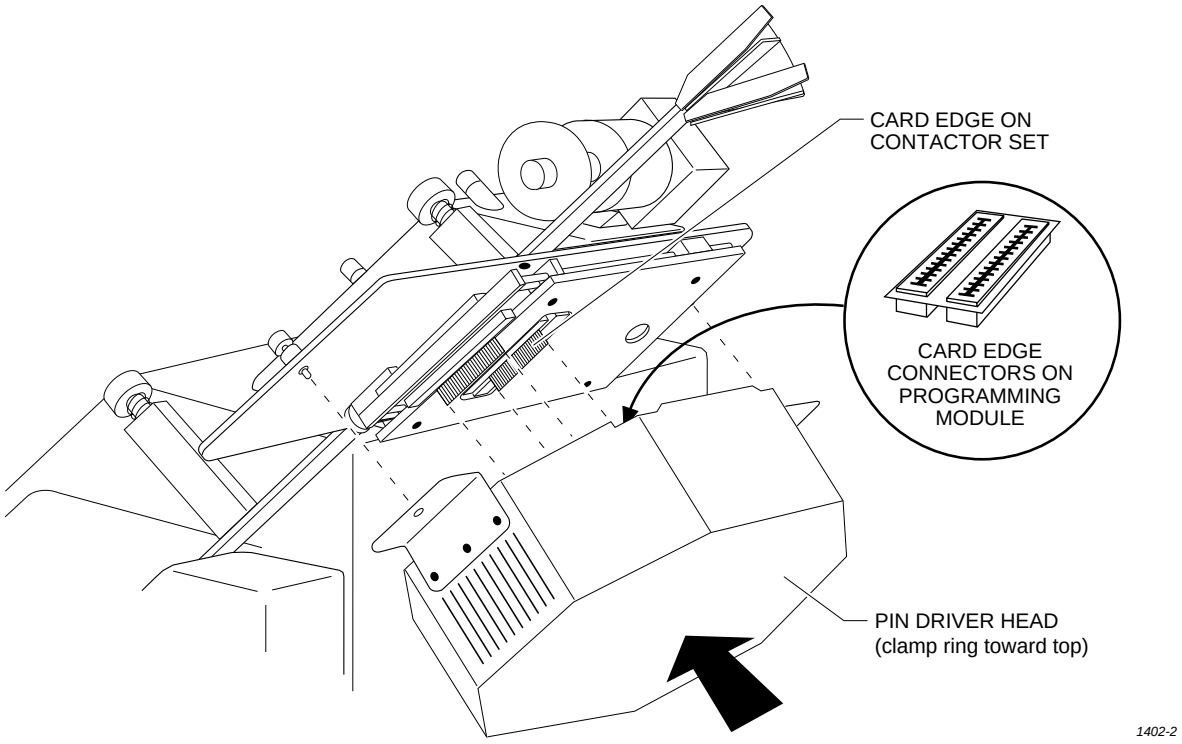


Figure 3-3
Removing the Contactor Set from the 2000

9. Replace the old contactor set with the contactor set that matches the programming module you installed in step 6. Position the pin driver head mounting plate to the handler. Finger tighten the thumbscrews on the handler to secure the pin driver head mounting plate to the handler.
10. Position the pin driver head to the handler so that the handle on the clamp ring is pointing toward the top of the handler.
11. As shown in Figure 3-4, align the card edge on the contactor set with the card edge connectors on the programming module. Gently push the pin driver head onto the handler.

Figure 3-4
Aligning the Pin Driver Head with the 2000



12. Tighten the thumbscrews on the pin driver head mounting plate, securing the pin driver head to the handler. You might have to use a flatblade screwdriver to finish tightening the thumbscrews.

CAUTION: To prevent damage to the edge connectors, and to ensure solid contact, we suggest you alternate tightening the left and right thumbscrews until the pin driver head is completely fastened to the 2000.

When properly connected to the handler, the mounting brackets attached to the pin driver head will be flush against the pin driver head mounting plate.

You are finished changing the programming module.

Changing a Programming Module on a ProMaster 3000, 7000, or 7500 Handler

Safety Information

This information is provided as a supplement to the Safety Summary at the beginning of this manual.

The circuitry housed inside the pin driver head and the control unit and the devices AutoSite programs are static sensitive and can be damaged by electrostatic discharge (ESD). To help minimize the effects of ESD, we suggest you wear an antistatic wrist strap while you follow the procedures described in this section.

For best performance, the antistatic wrist strap should be connected to a properly grounded antistatic workstation and the wrist strap should contain a 1 MΩ (minimum) to 10 MΩ (maximum) isolating resistor.

Follow the steps below to change the programming module on a ProMaster 3000, 7000, or 7500 handler. Instructions which pertain to a specific handler are noted in the text.

1. Make sure the handler is idle.
2. Clear all devices from the handler.
3. Turn the knob on the control unit and push the control unit down so you have more room to work (3000 and 7000 only).

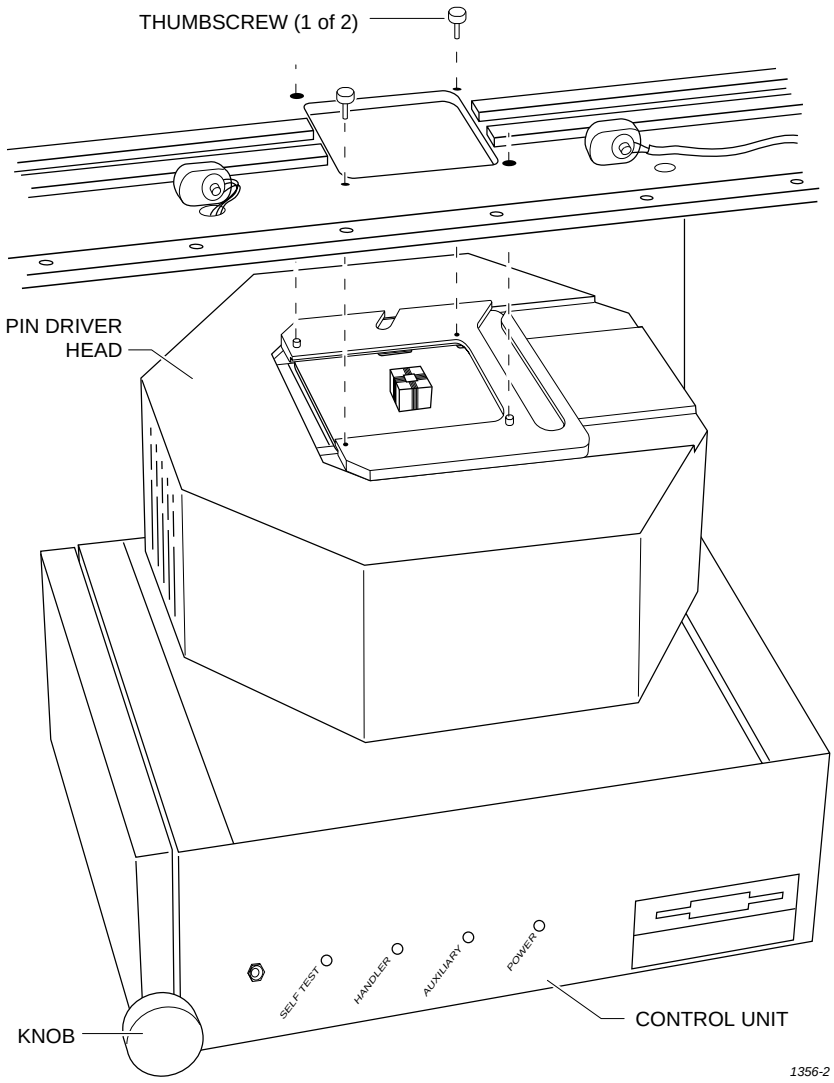
CAUTION: In this section, you will be working with the control unit, which is connected to the 3000. The control unit is supported by an air shock and can spring up when you loosen the knob.

Use caution when adjusting the height of the control unit or working with the air shock.

4. Raise the hood on the handler.

5. While using one hand to support the pin driver head, loosen the two thumb screws shown in Figure 3-5 and lower the pin driver head from the handler. Set the pin driver head on the control unit.

Figure 3-5
Removing the Pin Driver Head from the Handler



6. Slide the clamp ring off the pin driver head. Set the clamp ring aside; you will need it later.

7. Remove the programming module from the pin driver head. Set the programming module aside.

   7500 only: Repeat this procedure for the second pin driver head.

CAUTION: Do not touch the pins that are exposed when you remove the programming module.

Select the new programming module to be installed from the *Device List* disk, as described in your handler manual. If the module needs to have the jumpers reconfigured, or if you are not certain how to determine if it needs to be reconfigured, refer to the "Operation" chapter of your handler manual.

8. As shown in Figure 3-6, set the new programming module onto the pin driver head, making sure the guide pins on the pin driver head line up with the guide holes in the programming module.

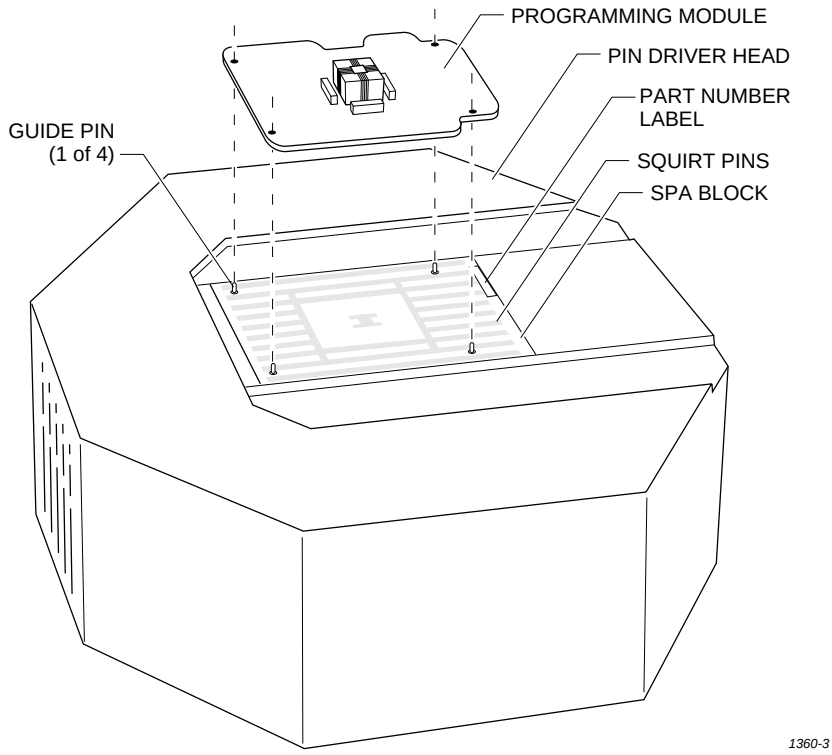


Figure 3-6
Aligning the Programming Module on the Pin Driver Head

9. As shown in Figure 3-7, slide the clamp ring onto the pin driver head, securing the new programming module in place.

   7500 only: Install the module for the AutoSite.

CAUTION: To ensure good contact between the module and the spring pins on the pin driver head, you may have to push down slightly on the programming module while sliding the clamp ring over the board. Apply downward pressure on the board, not the programming module's socket. Pressing down on the socket could damage the pins or contacts.

You will feel and hear a "click" from the clamp ring when the programming module is properly secured to the pin driver head.

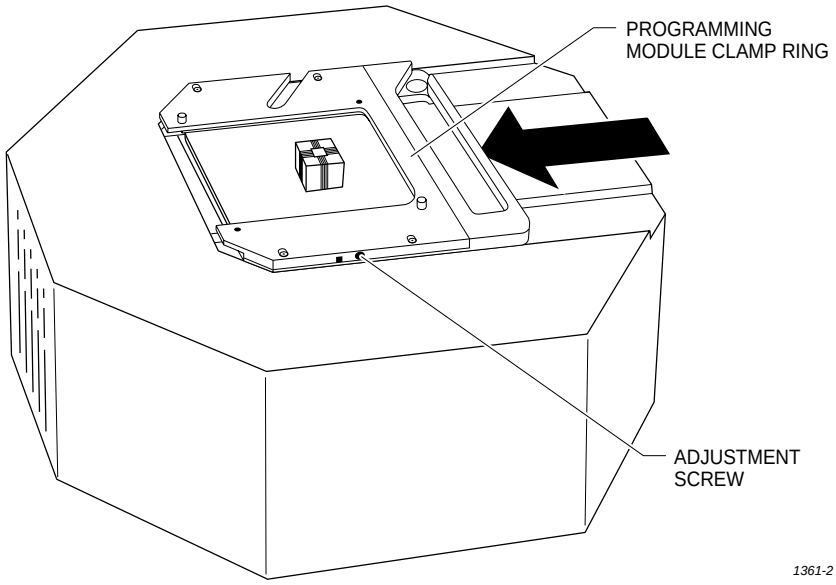


Figure 3-7
Securing a Programming Module to the Pin Driver Head

Note: Before continuing with the next step, we suggest that you change the chuck on the handler. See the "Removing and Installing Chucks" section of the handler manual for more information.

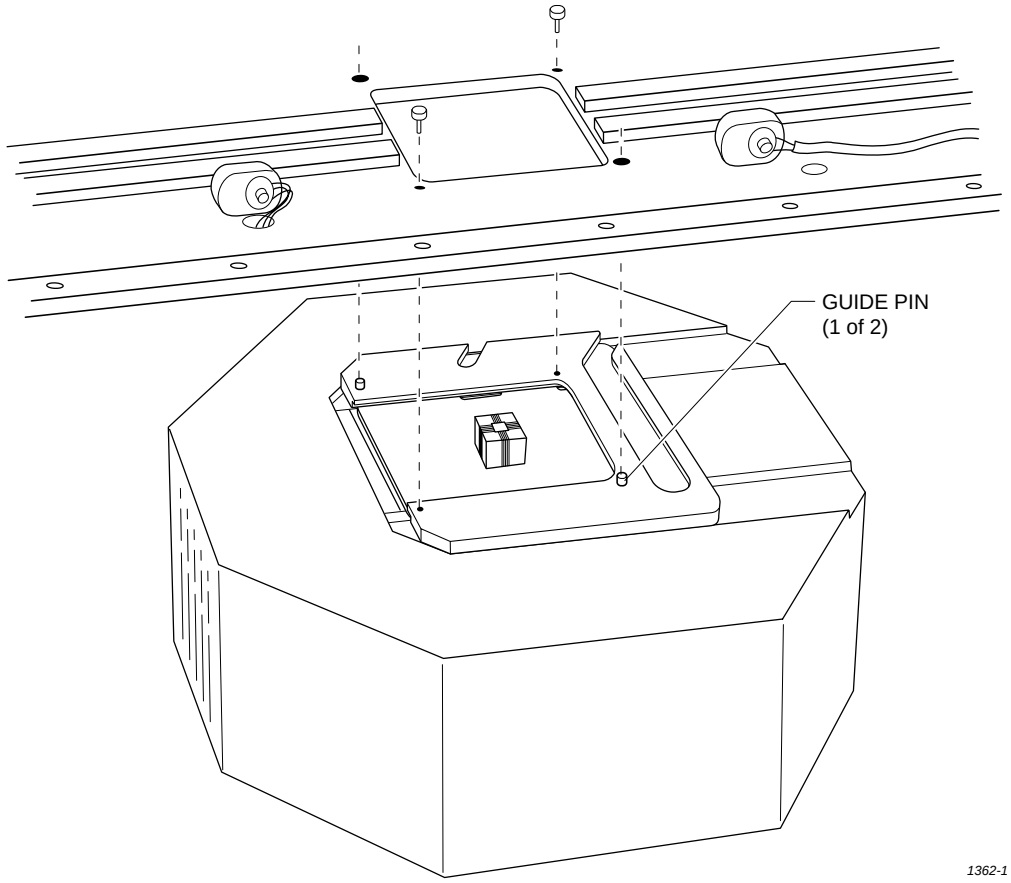
10. Position the pin driver head against the bottom of the handler. Using the guide pins on the clamp ring, align the pin driver head with the handler. Tighten the two thumbscrews to secure the pin driver head to the handler.

Note: When properly connected to the handler, the clamp ring will be flush against the handler.

7500 only: Replace the second pin driver head.

CAUTION: When the pin driver head is fully in position under the main plate, make sure that no wires or air lines are pinched by the head.

Figure 3-8
Securing the Pin Driver Head to the Handler



11. Readjust the height of the control unit (3000 and 7000 only).

You are finished changing the programming module.

Inserting the DIP or PLCC Base

This section explains how to insert the DIP and PLCC Bases into AutoSite.

About the Base

Similar to a programming module, the DIP Base and PLCC Base serve as the interface between a device and AutoSite. The DIP and PLCC Bases are designed to help isolate programming problems and hardware problems. For more information, see the section "Isolating Programming Problems."

Inserting a Base

Follow the procedure below to insert a Base into AutoSite.

Note: You can install and remove a Base with the power on as long as you are not performing a device operation.

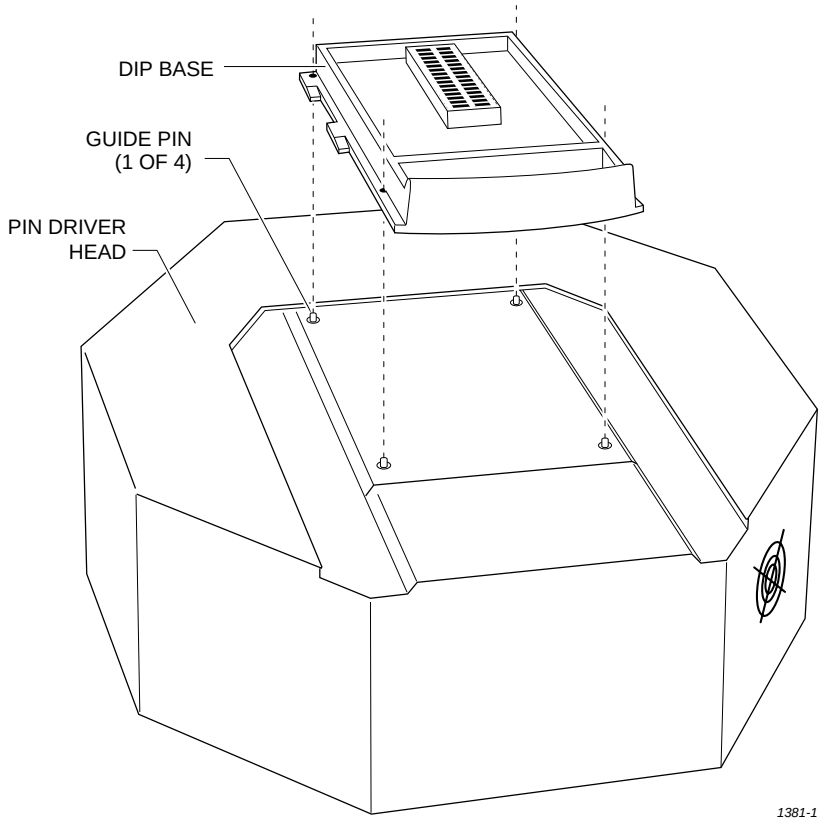
1. Make sure the handler is idle.
2. Clear all devices from the handler.
3. If the pin driver head is attached to a handler, remove the pin driver head from the handler. See the handler manual for more information.
4. If a programming module is installed in the pin driver head, remove the programming module from the pin driver head. Set the programming module aside. See the handler manual for more information.

CAUTION: Do not touch the pins that are exposed when you remove the programming module.

5. Slide the compression handle onto the pin driver head. The compression handle is shown in Figure 1-1.

6. As shown in Figure 3-9, set the Base onto the pin driver head, making sure the guide pins on the pin driver head line up with the guide holes in the Base.

Figure 3-9
Aligning the Base on the Pin Driver Head



7. Squeeze the Base and the clamp ring together, securing the Base to the pin driver head. You do not need to use excessive force.

CAUTION: You can damage AutoSite by squeezing too hard.

With the Base installed in the pin driver head, you can perform the following procedures:

- Update AutoSite to a new version of system software. See the *User Notes and Update Instructions* that accompany the new software.
- Program PLCC devices one at a time. See the section titled “Inserting PLCC Devices and Using MatchBooks” for more information.

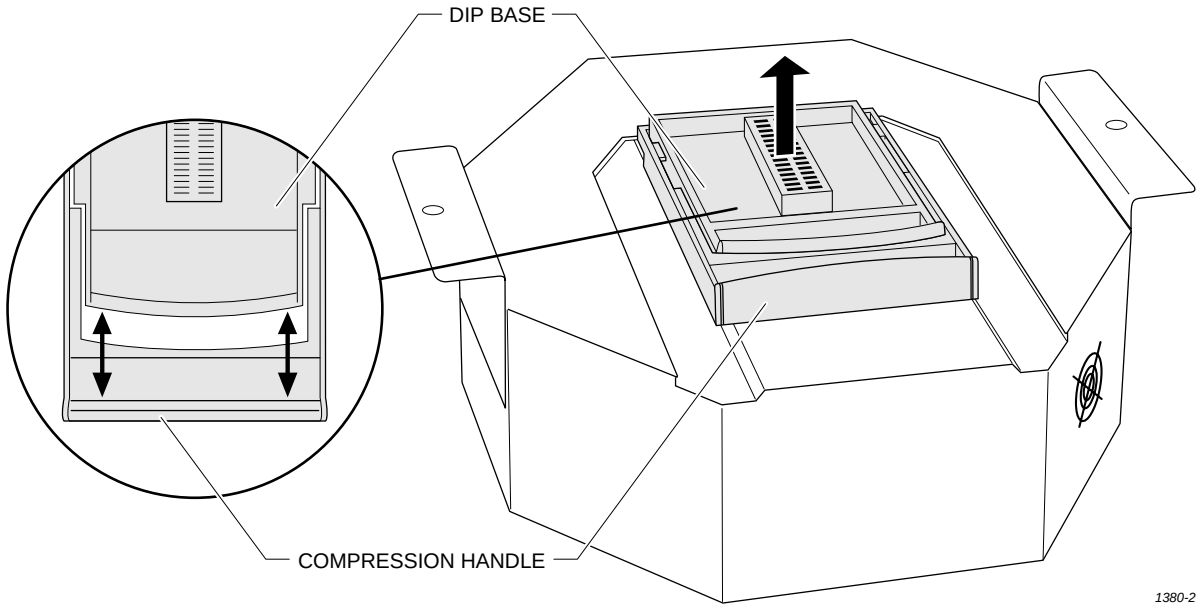
Removing a Base

When removing a Base from AutoSite, be sure to apply even pressure while moving the handles apart. If you exert uneven pressure on the handles, you could damage the clamp ring.

To remove a Base, follow the steps described below.

1. As shown in Figure 3-10, remove the Base by separating the handles on the Base and the clamp ring with your thumbs and fingers.

Figure 3-10
Removing a Base



2. Lift the Base up and out of the pin driver head. Store the Base in a safe place.

CAUTION: Do not touch the pins that are exposed when you remove the Base.

Once the Base is removed from AutoSite, we suggest you reinstall a programming module in the pin driver head. For more information on installing a programming module and reconnecting the pin driver head to a handler, see the section titled “Changing a Programming Module” earlier in this chapter.

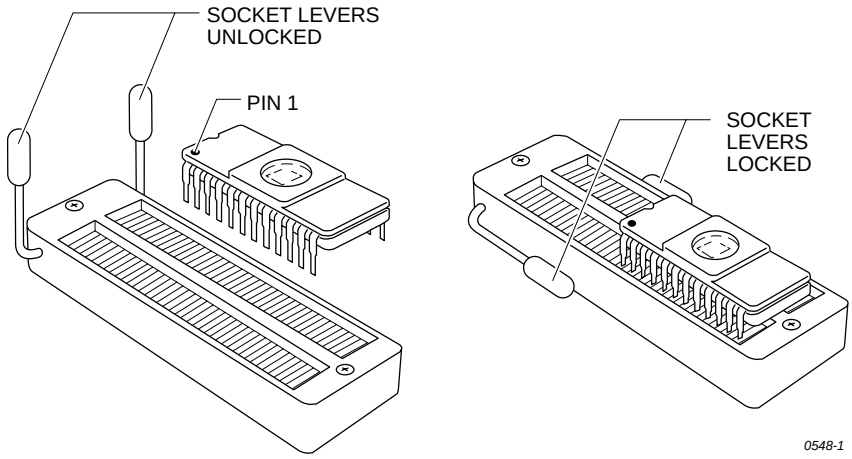
Inserting a DIP Device

To insert a DIP device into the DIP Base, follow the steps below:

1. Unlock the socket on the DIP Base by pulling up on the socket lever.
2. Insert the DIP device into the socket. Make sure the device is bottom justified. If the device is not bottom justified, AutoSite will not be able to read or program the device.
3. Lock the device into place by pressing the socket lever down.

Note: Insert DIP devices into AutoSite AFTER you have installed the DIP Base in AutoSite.

Figure 3-11
Inserting a DIP Device into the DIP Base



Removing a DIP Device

To remove a DIP device from the DIP Base, follow the steps below:

1. Make sure AutoSite has finished programming and testing the device in the DIP Base.
2. Unlock the socket on the DIP Base by pulling up on the socket lever.
3. Remove the device by lifting it out of the socket. Set the device on an antistatic surface or in antistatic foam.

Note: Remove DIP devices from the DIP Base before you remove the DIP Base from AutoSite.

Inserting PLCC Devices and Using MatchBooks

Read this section if you will be using the Stand Alone Kit (a PLCC Base and a set of MatchBooks) to program PLCC devices.

Data I/O has developed a new method to accommodate the special nature of PLCC packages. The method uses the MatchBook, which holds the PLCC device in place on the PLCC Base. When the device is locked into place, the conductive pad on the bottom of the PLCC Base forms a conductive path between the pin drivers in the pin driver head and the device in the MatchBook.

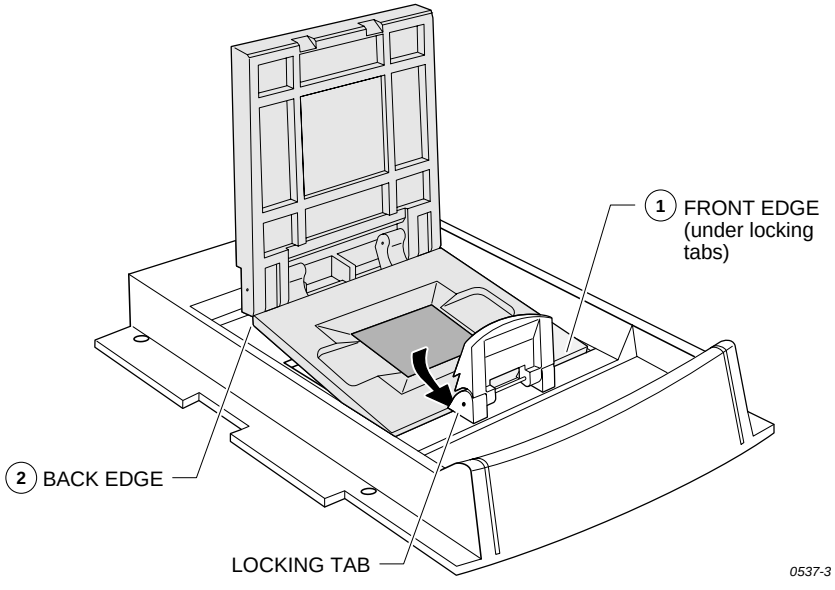
MatchBooks do away with clumsy and expensive sockets and adapters, and make inserting and removing surface mount devices much easier and faster. MatchBooks eliminate the guesswork involved when you insert a PLCC device into a socket. All you have to do is align pin 1 and set the device in the MatchBook.

Inserting a Device Into a MatchBook

The instructions below explain how to use a MatchBook and how to insert and remove a device from a MatchBook.

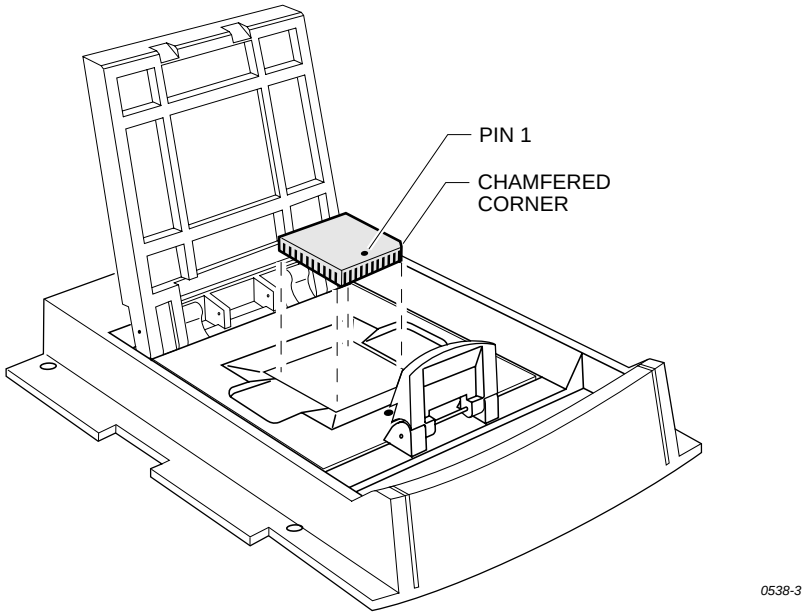
1. Insert the PLCC Base into AutoSite. Lock the Base into place. See the section titled “Inserting the DIP or PLCC Base” for information on inserting the PLCC Base.
2. Select the MatchBook for the device you are going to use.
3. Insert the MatchBook into the PLCC Base as shown in Figure 3-12. First, set the front edge of the MatchBook onto the PLCC Base. Then lower the back edge of the MatchBook into place on the Base.

Figure 3-12
Inserting a MatchBook into the PLCC Base



4. Insert the device into the MatchBook, as shown in Figure 3-13. Make sure you insert the device according to the alignment diagram on the MatchBook. Close the MatchBook.

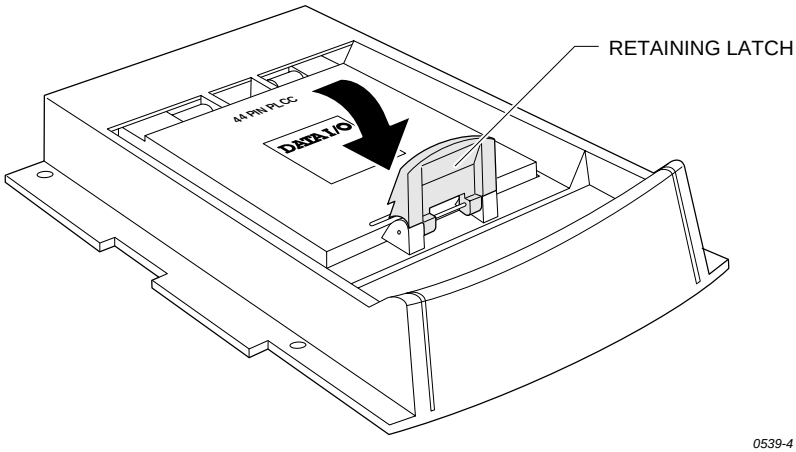
Figure 3-13
Inserting a Device into the PLCC Base



Note: Position the device so pin 1 is near to the retaining clip. There is a small dot molded into each MatchBook to help you align your device. Each MatchBook also has a beveled corner to help you align devices with a chamfered corner to indicate pin 1.

5. Finally, close the MatchBook and press the retaining latch forward with your thumb until the latch snaps into place, as shown in Figure 3-14.

Figure 3-14
Closing the MatchBook



Removing a Device From a MatchBook

To remove a device from the MatchBook, unsnap the retaining latch, open the MatchBook, and lift out the device.

Preventive Maintenance

Conductive Pad

The conductive pad, the material the MatchBook rests on, should be kept free of dirt to keep yields high and prolong the life of the pad. We recommend that you inspect and clean the pad at least every 1000 device insertions or monthly, whichever comes first.

Note: After a number of insertions, you may notice an indentation in the middle of the conductive pad. The indentation is normal and does not degrade the contact resistance or the performance of the MatchBook. It is also normal for the pad to show signs of discoloration as it is used.

The life of the pad is dependent on proper care as well as the pin count and package type of the device being used. Not all devices have the same tolerances, and use of each device type may result in different life cycles for the pad. If you experience an increase in device insertion errors or continuity errors, or if you experience a sudden drop in programming yields, the pad may need to be replaced.

Cleaning

Blow air over the pad to clean it. If you use compressed air, direct the air stream from the front or back of the Base.

Note: To avoid lifting the pad off the circuit board, do not blow air from the side of the pad.

To further clean the pad, apply a small amount of isopropyl alcohol on a cotton swab and gently wipe off the pad to dislodge dirt. Make sure the pad is clear of any cotton filaments left over after cleaning.

CAUTION: Do not clean the pad with any petroleum- or freon-based products. These substances will cause premature deterioration of the pad material.

Replacement Pad Kits

The Base has been designed to allow you to replace the pads quickly and easily and to minimize downtime. To order a replacement pad kit, contact Data I/O Customer Support as listed in the Preface.

SPA Block and Base

For optimal performance, keep the SPA block (see Figure 3-6) and bases clean. The following messages during device operations could result from dirt in the SPA block or base adapter.

```
ID Error
Continuity Error
Base Adapter not Installed
Device Insertion Error
Overcurrent Error
Base/Adapter Relay Failure
```

SPA Block Cleaning

To avoid error conditions caused by dirty or worn SPA blocks, we recommend that you perform the following preventive maintenance procedures.

- Keep the SPA block covered with a base or programming module when not in use. To prevent base adapters from contaminating the SPA block, store them in an uncontaminated area.

- Each time you remove or replace a programming module, clean the SPA block with a brush.
- Inspect the SPA block and base adapter for dirt weekly. Follow the steps below to clean the SPA block.
  1. Blow compressed air across the SPA block.
  2. Mildly dampen a small section of a lint-free cloth with a DeOxit pen (Data I/O P/N 570-5500-901) and gently rub the dampened cloth across all the pins on the SPA block.
  3. Using a clean section of the lint-free cloth, gently wipe the surface again.
  4. To ensure that all pins are properly positioned in their receptacles, push a programming module down on the SPA pins a few times and check that the pins spring up to their normal upright position.

Base Cleaning

Weekly inspect the bases for dirt. Clean the surface, including the underside, with filtered compressed air. Clean with a lint-free cloth dampened with DeoxIT if needed.

Isolating Programming Problems

If you are experiencing less than optimal yields when programming a

DIP or PLCC device, we suggest that you remove the AutoSite pin driver head from the handler and try programming the device in the DIP or PLCC Base.

If the device programs successfully in the Base, there could a problem with the handler or with the programming module supplied by the handler manufacturer. Refer to your handler manual for cleaning, maintenance, and testing information.

If you experience programming problems when programming the device in the Base, try to isolate the nature of the problem. For example, does the problem occur during programming or during post-programming testing? If you are unable to correct the problem yourself, contact Data I/O Customer Support as listed in the Preface of this manual.

Updating the MSM (Mass Storage Module)

The Mass Storage Module is an internal hard drive that can be installed in AutoSite. The Mass Storage Module allows you to store system and algorithm files in the programmer, which can speed up the start-up and disk access routines.

Installation

The MSM comes installed in all new AutoSites. If your older AutoSite does not have an MSM installed, you can purchase one for your AutoSite. Installation instructions are shipped with the Mass Storage Module.

Updating Software

To update AutoSite system software and algorithm files on your MSM, refer to the User Notes accompanying the new software.

Booting from MSM

After installing the system software on the MSM, you can boot directly from the MSM by doing the following:

1. Make sure that drive A is empty.
2. Reboot or powerup AutoSite.

When AutoSite boots up and does not find a disk in drive A, it boots using the system software located on your MSM.

Storage Capacity

The MSM is partitioned into four logical drives, as follows:

Drive	Storage	Max. No. of Files	Data Type
C	31 MB	512	User Data
D	31 MB	512	User Data
H	7 MB	320	System Data
I	10 MB	320	System Data

Storage Suggestions

Drives C and D are reserved for user data; drives H and I are reserved for system use. Although drives H and I can be written to and read from, we STRONGLY suggest you use only C and D to store your data.

Limitations

Other than the file operations listed below, drives C and D on the MSM can be used for all file operations that can be done with a floppy drive:

File Operation	Comments
Format Disk	Format only C and D. Formatting H or I will render your programmer inoperative until you restore the system software from floppy.
Duplicate Disk	Drives C, D, H, and I cannot be duplicated using the Duplicate Disk command.

Backing up the MSM

We STRONGLY suggest that you periodically make backup copies of the files on the C and D partitions of the MSM. If the MSM were to "crash," you could restore your data files to C and D from your backups, and you could restore the system files from the AutoSite disks.

Adding a New Programming Module

To add device support to your AutoSite, simply plug in a new programming module and you are ready to use the new device support. To add additional programming capabilities to your AutoSite, simply purchase a new programming module and insert it in AutoSite. The "Options" section of Chapter 1 lists the available programming modules.

Self-test

The Self-test command allows you to test circuits and subsystems in AutoSite, verifying proper operation or isolating possible problem areas.

An automatic self-test is performed each time AutoSite is powered up.

Stopping Self-test

You can stop a self-test anytime during its operation. Press **CTRL** + **Z** to stop a self-test.

Running Self-test

To perform a self-test, perform the following steps.

Note: Unless otherwise noted, all references to TaskLink in this manual refer to the DOS TaskLink product. TaskLink for Windows performs similar operations using different user interface commands. Refer to the ***TaskLink for Windows Getting Started Guide*** *and online Help for additional information.*

1. Start your terminal emulation software. If you are using TaskLink, start TaskLink in administrator mode by entering **tl a** at the DOS prompt. If you are already running TaskLink in operator mode, exit TaskLink and restart it in administrator mode.
2. If you are using TaskLink, skip to step 8. Otherwise, reboot AutoSite by cycling power.
3. Wait for AutoSite to complete the powerup self-test. The Self-test LED goes out when the powerup self-test is complete.

Note: The powerup self-test takes approximately four minutes.

4. When the self-test is complete, you will see either the powerup screen or a single angle bracket prompt ">". If you see the powerup screen, go to step 6.
5. Type **Z** ↵ to display the powerup screen.
6. AutoSite displays the powerup screen and the following prompt:

   ```
   Current terminal type = DEC VT100 (ANSI 3.64)

   Do you want to select a new terminal type? (Y/N)
   ```

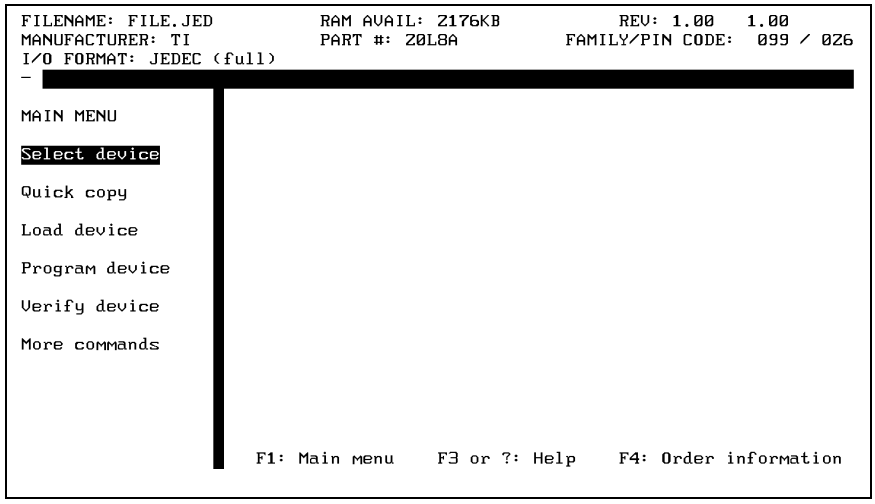
   If the current terminal type matches the terminal type you are using, press **N** ↵ and go to step 8.

   If the current terminal type does not match the terminal type you are using, press **Y** ↵. AutoSite displays the default and current terminal types and a list of the available terminal types. Select a terminal type that matches the terminal you are using and press ↵.
7. AutoSite responds with the following prompt:

   ```
   Save terminal type as power on default? (Y/N)
   ```

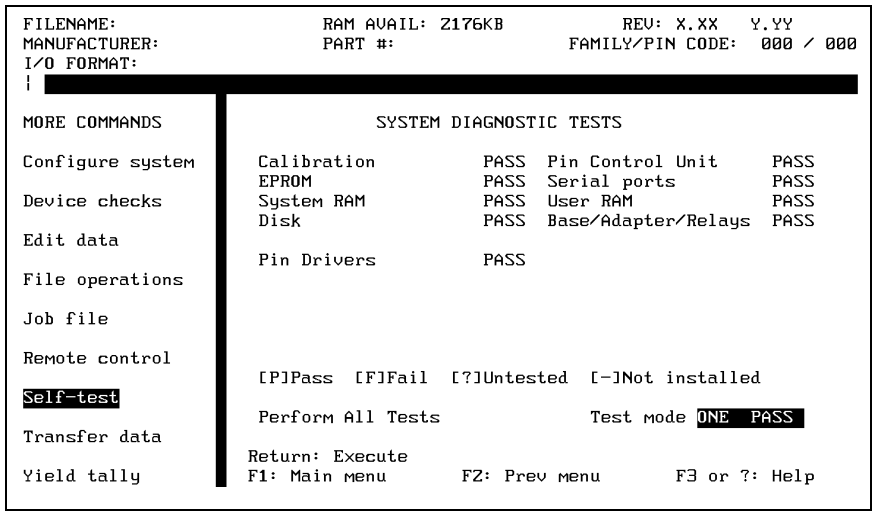
   Press **Y** ↵. AutoSite saves your current terminal type as the powerup default and then displays the Main Menu, which is shown in Figure 3-15.

Figure 3-15
The AutoSite Main Menu



8. Make sure the device socket is empty.
9. If you are using TaskLink, select **Programmer Interface** from the **Utilities** menu.
10. From the Main Menu, press **F1**, **M**, **S** to display the Self-test screen, which is shown in Figure 3-16.

Figure 3-16
The Self-test Screen



Note: The Base / Adapter / Relays *field will display* ???? *if this test has not explicitly been run from the Self-test screen for the current base. The Base/Adapter/Relay test is not run during powerup self-test.*

In addition, the screen shown above may differ slightly from the screen shown on your terminal.

11. Select the test mode. You can select either one-pass or continuous testing. To toggle modes, move the cursor to the Test Mode field and press **SPACE.**

    One Pass testing runs the specified test once. Continuous testing runs the specified test until there is a failure or until you stop the test by pressing **CTRL** + **Z**.

Note: There may be a delay before AutoSite responds to the Ctrl-Z if you are testing system RAM or user RAM.

CAUTION: Executing the System RAM test or the User RAM test erases all data in RAM.

12. To test all hardware, move the cursor to the **Perform All Tests** prompt and press ↵. To test a particular item, move the cursor to the desired test and press ↵.

Manipulating Keep Current Algorithm Files

The Keep Current service allows you to download new and updated device algorithms from the Keep Current Bulletin Board System. Keep Current algorithms are made available to Keep Current subscribers before they are incorporated in an update kit.

See the *Keep Current User Manual* for more information and for instructions on downloading Keep Current algorithms.

Accessing the Keep Current Menu

Before you can manipulate a Keep Current algorithm file, you must access the Keep Current menu, which can only be done with a full screen terminal that emulates one of the following:

- ANSI 3.64 compatible terminals
- DEC VT-100 compatible terminals (supported by TaskLink)
- Qume QVT-101 compatible terminals
- TELEVIDEO TVI--910 compatible terminals
- Wyse WY-30 compatible terminals

If you are using handler control software, consult the documentation to see if it supports one of the terminal types listed above.

If you are using TaskLink, your system should already be configured to access the Keep Current menu; skip to step 9 on the next page. Otherwise, do the following:

1. Connect the terminal to one of the serial ports on the control unit. The serial ports are shown in Figure 1-4.
2. Configure your terminal or terminal emulation software to match the following parameters:
   - 9600 baud
   - No parity
   - 8 data bits
   - 1 stop bit

   See the documentation supplied with your terminal or terminal emulation software for more information.
3. Power up AutoSite if it is not already powered up. If you power up AutoSite, wait for AutoSite to complete the powerup self-test. The self-test LED on the control unit goes out when the self-test is complete.

Note: The powerup self-test takes approximately three minutes on a 44-pin AutoSite and four minutes on an 88-pin AutoSite.

4. When the self-test is complete, you will see either the powerup screen or a single angle bracket prompt (`>`). If you see the powerup screen, go to step 6.
5. Type **Z** ↵ to display the powerup screen.
6. AutoSite displays the powerup screen and the following prompt:

   ```
   Current terminal type = DEC VT100 (ANSI 3.64)

   Do you want to select a new terminal type? (Y/N)
   ```

   If the current terminal type matches the terminal type you are using, press **N** ↵ and go to step 8.

   If the current terminal type does not match the terminal type you are using, press **Y** ↵. AutoSite displays the default and current terminal types and a list of the available terminal types. Select a terminal type that matches the terminal you are using and press ↵.
7. AutoSite responds with the following prompt:

   ```
   Save terminal type as power on default? (Y/N)
   ```

   Press **Y** ↵. AutoSite saves your current terminal type as the power on default and then displays the Main Menu, which is shown in Figure 3-15.
8. From the Main Menu, press **M**, **C**, **K** to get to the Keep Current menu.

9. If you are using TaskLink, access the Keep Current menu by doing the following:
   - Start TaskLink in administrator mode by entering **tl a** at the DOS prompt. If you are already running TaskLink in operator mode, exit TaskLink and restart it in administrator mode.
   - Select **Programmer Interface** from the **Utilities** menu.
   - If you are entering the programmer interface for the first time since powering up your programmer, the banner screen appears. Press ↵ to get to the programmer's Main Menu.
   - From the programmer's Main Menu, press **F1**, **M**, **C**, **K** to get to the Keep Current menu.

Viewing Keep Current Files

This section describes how to view a list of all the Keep Current algorithm files found on the Algorithm disk. Each Keep Current algorithm file has the extension .KCx, where *x* is a number ranging from 0 to 9.

Note: When viewing a list of Keep Current algorithms, AutoSite does not check for compatibility between the installed version of system software and the individual Keep Current algorithms.

To view a list of Keep Current algorithm files found on the installed disk, follow these steps:

1. Make sure the handler is idle.

2. Access the Keep Current menu as described in the section titled "Accessing the Keep Current Menu."

3. Insert the disk with the Keep Current algorithm files you want to view into the disk drive.

   Typically, Keep Current algorithms are downloaded to a separate disk. If you wish to manipulate a Keep Current algorithm, the Keep Current algorithm must be copied to the appropriate Algorithm / System disk.

   If you are performing these operations using the MSM, the Keep Current algorithm and system (.sys) files must be copied to the **I:** directory. Use the More Commands / File Operations / Copy File command.

4. Press **V**. AutoSite displays up to 10 files at one time. If there are more than 10 files, press **CTRL** + **N** to display the next page of files. Press **CTRL** + **P** to display the previous page of files.

   If you want to view files on another disk, press **F2**, insert another disk, and return to the beginning of this step.

5. When you are finished, restart your handler control software as normal or, if you are using TaskLink, press **ALT** + **F1** to return to TaskLink.

Replacing an Algorithm

This section describes how to substitute a Keep Current algorithm for an algorithm included on the Algorithm disk.

To substitute an algorithm, follow these steps:

1. Make sure the handler is idle.
2. Ensure that the Keep Current algorithm you intend to manipulate is on the same disk as the System algorithm that will be replaced.
3. Access the Keep Current menu as described in the section titled "Accessing the Keep Current Menu."
4. Insert the Algorithm / System disk on which the Keep Current algorithm is located into the disk drive.

Note: Normally, the Keep Current algorithm files are on the Algorithm disks.

5. From the Keep Current menu, press **R**. AutoSite displays up to 10 files at one time. If there are more than 10 files, press **CTRL** + **N** to display the next page of files. Press **CTRL** + **P** to display the previous page of files.

   The files displayed in reverse video—that is, black text on a white background—are Keep Current algorithms that have already replaced algorithms on the Algorithm disk.

   If no compatible Keep Current algorithm files are found, AutoSite displays the following message:

   ```
   Insert Keep Current algorithm disk
   ```

   Insert an Algorithm disk that contains Keep Current algorithms and repeat this step.

Note: For this command, AutoSite displays only the algorithms that are compatible with the installed version of system software.

6. If you want to view files on another disk, press **F2**, insert another disk, and return to the beginning of step 4.
7. Enter the number of the Keep Current algorithm corresponding to the algorithm you want to replace (substitute).

Note: The maximum number of replaced algorithms is 10.

8. Press ↵ to replace (substitute) the algorithm on the Algorithm / System disk with the Keep Current algorithm you selected in step 7. If you do not want to replace the algorithm, press **F2** to return to the Keep Current menu.

Note: If you accidentally replace an algorithm, use the Restore command to undo the replacement. See the section titled "Restoring a Replaced Algorithm" for more information.

After replacing an algorithm, AutoSite displays the replaced Keep Current algorithm in reverse video.

9. When you are finished, restart your handler control software as normal or, if you are using TaskLink, press **ALT** + **F1** to return to TaskLink.

Restoring a Replaced Algorithm

This section describes how to restore an algorithm that was previously replaced with a Keep Current algorithm. To restore an algorithm to its previous state, follow these steps:

1. Make sure the handler is idle.
2. Access the Keep Current menu as described in the section titled "Accessing the Keep Current Menu."
3. Insert the disk that contains the Keep Current algorithm file(s) you previously replaced and that you want to undo.
4. Press **R**. AutoSite displays up to 10 files at one time. If there are more than 10 files, press **CTRL** + **N** to display the next page of files. Press **CTRL** + **P** to display the previous page of files.

   The files displayed in reverse video are Keep Current algorithms that have already replaced algorithms on the Algorithm disk.

   If no compatible Keep Current algorithm files are found, or if there are no Keep Current algorithm files on the disk, AutoSite displays the following message:

   ```
   Insert Keep Current algorithm disk
   ```

   Insert a disk that contains Keep Current algorithms and repeat this step.

Note: For this command, AutoSite displays only those algorithms that are compatible with the installed version of system software.

5. If you do not see the algorithm you want to restore, press **F2**, insert another disk, and return to the beginning of step 4.
6. Move the cursor to the Replace / Restore field and enter the number corresponding to the file you want to restore.

7. Press ↵ to restore the algorithm on the Algorithm disk to its original state. If you do not want to restore the file, press **F2** to return to the Keep Current menu.

Note: If you accidentally restore an algorithm, use the Replace command to undo the restoration. See the section titled "Replacing an Algorithm" for more information.

After restoring the algorithm, AutoSite displays the restored algorithm in normal video.

8. When you are finished, restart your handler control software as normal or, if you are using TaskLink, press **ALT** + **F1** to return to TaskLink.

Deleting Keep Current Files

This section describes how to delete a Keep Current algorithm file from a disk.

Note: For this command, AutoSite displays the Keep Current algorithms it finds without checking for compatibility with the installed version of system software.

To delete a Keep Current algorithm file from a disk, follow these steps:

1. Make sure the handler is idle.

Note: The Delete command is different from the Purge command. The Delete command deletes one file at a time, regardless of whether the algorithms are compatible with the current version of installed software or not. The Purge command deletes all Keep Current algorithms that are not compatible with the current version of installed software.

If you want to delete all Keep Current algorithms that are not compatible with the current version of installed software, see the section titled "Purging Keep Current Files" for more information.

You should not delete an algorithm file unless you want all algorithms in the file to be deleted.

2. Access the Keep Current menu as described in the section titled "Accessing the Keep Current Menu."
3. Insert the disk with the Keep Current algorithm files you want to delete into the disk drive.
4. Press **D**. AutoSite displays up to 10 files at one time. If there are more than 10 files, press **CTRL** + **N** to display the next page of files. Press **CTRL** + **P** to display the previous page of files.

   If you do not see the file(s) you want to delete, press **F2**, insert another disk, and return to the beginning of this step.
5. Move the cursor to the Delete field and enter the number corresponding to the file you want to delete.

6. Move the cursor to the `Are You Sure` field and press **Y**.

CAUTION: Deleting a file deletes all the algorithms in that file. Do not delete a file unless you are sure you want all algorithms in the file deleted.

If you do not want to delete the file, do not press ENTER.

7. To delete the file you selected in step 5, press ↵. If you do not want to delete the file, press **F2** to return to the Keep Current menu.
8. When you are finished, restart your handler control software as normal or, if you are using TaskLink, press **ALT** + **F1** to return to TaskLink.

Purging Keep Current Files

This section describes how to purge all Keep Current algorithm files that are not compatible with the current version of installed system software from a disk.

To purge incompatible Keep Current algorithms from a disk, follow these steps:

1. Make sure the handler is idle.

Note: The Purge command is different from the Delete command. The Purge command deletes all Keep Current algorithm files that are not compatible with the current version of installed software. The Delete command allows you to delete one file at a time, regardless of whether the Keep Current algorithm is compatible with the current version of installed software or not.

If you want to delete specific Keep Current algorithms, see the section titled "Deleting Keep Current Files" for more information.

2. Access the Keep Current menu as described in the section titled "Accessing the Keep Current Menu."
3. Insert the disk with the Keep Current algorithm files you want to purge into the disk drive.
4. Press **P**. AutoSite displays up to 10 outdated Keep Current algorithm files at one time. If there are more than 10 files, press **CTRL** + **N** to display the next page of files. Press **CTRL** + **P** to display the previous page of files.

   If you do not want to purge all of these files, press **F2** to return to the Keep Current menu.

   If no outdated files are on the disk, the message `Insert Keep Current Algorithm Disk` appears.
5. Move the cursor to the `Are You Sure` field and press **Y**.

CAUTION: If you do not want to purge all the files listed, do not press ENTER.

6. To purge the files listed on the screen, press ↵. If you do not want to purge the files, press **F2** to return to the Keep Current menu.

   If you press ↵, AutoSite purges the Keep Current algorithm files listed on the screen. When finished purging the files, AutoSite displays the next screen of outdated files to purge.

   If no more outdated Keep Current algorithm files are left on the disk, AutoSite returns to the Keep Current menu.

7. When you are finished, restart your handler control software as normal or, if you are using TaskLink, press **ALT** + **F1** to return to TaskLink.

A Computer Remote Control

The programmer can be controlled via a host computer using Computer Remote Control (CRC) protocol. CRC commands have been designed to be incorporated into a remote computer software program (driver) which will allow an operator to control the programmer. The driver generates commands and sends them to the programmer, which executes the commands. The programmer then returns a response character, and in some cases, data. The driver reacts to the response and uses it to generate messages and prompts for the user.

---

*Note: You do **not** need to use CRC if you are using TaskLink or are accessing the programmer's built-in menu system (using HiTerm or a similar product to communicate with the programmer). CRC commands offer you an alternative, allowing you to create your own custom interface with the programmer.*

This chapter is not intended to be a complete guide to using CRC commands. For a more detailed explanation of CRC commands, refer to the "UniSystem Computer Remote Control" Application Note available from Customer Support.

This chapter contains the following information:

- **System Setup**—Explains how to set up the programmer for remote control operation. Includes information on entering and exiting CRC mode.
- **CRC Summary**—Gives a listing of the available CRC commands.

Which Driver to Use?

If you are using CRC commands, you must use a driver program to send the CRC commands and receive the programmer's responses. You can either write your own software driver or use an existing driver (such as the **terminal.exe** program included with Windows).

System Setup

The programmer receives CRC commands and sends responses to the host computer through an RS-232C port using a 25-pin D connector in two possible configurations: either DTE or DCE. Only the Handler port supports CRC operation.

The pin designations for the Handler port are shown in the "More About Cables" section of Chapter 2. Included in that section is a table of pin definitions which explain the function of each pin for the two serial port configurations.

To ensure correct operation of the Handler port with the host computer, set the parameters for the Handler port according to the host computer requirements.

Entering CRC Mode

Follow the steps below to enter CRC mode at powerup.

AutoSite enters CRC mode during powerup if it detects equipment connected to the Handler port.

Note: CRC operations must be run from the Handler port; AutoSite does not support powerup CRC on the Auxiliary port.

To ensure correct operation of the Handler port with the host computer, set the parameters for the Handler port according to the host computer requirements.

Halting CRC Operations

To halt any command or any ongoing CRC operation, use one of the following commands from the Handler port. Neither of the following two commands requires that you press ↵. Both commands are immediate and both terminate any preceding command operation.

ASCII Command	Hex Code	Description
ESC	1B	Causes AutoSite to unconditionally halt any operation except a binary transfer.
BREAK	n/a	Causes AutoSite to unconditionally halt any operation in progress. This includes all data communications transfers. The data line must be held in the spacing condition for 110 ms to 700 ms.

CRC Default Settings

When CRC mode is entered, certain defaults are set prior to Autosite's accepting any commands. The default settings are outlined below:

Description	Setting
Upload / download port	Handler port
Data source / destination	RAM
Security fuse data (0 or 1)	0
Program security fuse	No
Reject option (commercial or single)	Commercial
Logic verification option	All
Number of verify passes (0, 1, or 2)	2
Fill RAM before downloading	No
Illegal bit check option	No
Blank check option	No
Enable yield tally option	No
EE bulk erase option	No
Odd / even byte swap for 16 bit option	No
JEDEC I / O translate DIP / LCC option	Yes
Continuity check option	Yes
Compare electronic signature	Yes
Host command	Blank
I / O address offset	0
I / O format	MOS technology (format 81)
Instrument control code (0, 1, 2)	0
I / O timeout	30 seconds
Upload wait	0 seconds
Number of nulls	255
Serial set auto-increment mode	No
Programming mode	Single device
Total set size	1
Upload EOF delimiter flag	Disabled
Download EOF delimiter flag	Disabled

If you exit remote mode using the **Z** command, AutoSite's parameters are set to what they were before you entered remote mode. If you exit using **CTRL** + **Z**, AutoSite's parameters are NOT changed.

CRC Commands

CRC commands are a set of simplified commands for AutoSite. The commands are designed to be received from a controlling computer. Because the commands are so simplified, they can be cryptic.

CRC Command Summary

You send CRC commands to AutoSite by typing the command and then pressing the ↵ key. When AutoSite receives a CRC command, the command is executed and a response is sent back, followed by a carriage return. If the response is an F, an error occurred. If the response is a ?, AutoSite did not understand the command. If the response is a >, the normal CRC prompt, the command executed properly. Some commands respond with both a value and the prompt. For example, AutoSite might return `00284295>` when you send the Calculate Sumcheck command. In this case, the `00284295` is the sumcheck and the `>` indicates that the command executed properly. The **I**, **O**, and **C** commands perform any data transfer prior to sending the response.

Each command in the CRC command set is summarized in the following tables. For a more detailed explanation of CRC commands, refer to the "UniSystem Computer Remote Control" Application Note available from Customer Support. The command tables are broken up into standard and extended CRC commands. Standard CRC commands are commonly used commands, such as load, program, and verify. Extended CRC commands are more specific device-related commands, such as Set Security Fuse, Fill Fuse Map, and Set Vector Test Options.

Note: While in CRC mode, AutoSite recognizes only uppercase characters.

Except where noted, the commands use the following notation conventions:

- lowercase alphabetic characters indicate arguments that must be specified
- *h* represents a hexadecimal digit.
- *n* represents a decimal digit.
- *xxx...xxxx* represents a string of characters.

For example, ***nn02*** indicates that you may precede the **02** command with two decimal digits.

Summary of Standard CRC Commands

Command	Description	Response
–	Invert RAM	>
hhhhhh:	Select device begin address	>
hhhhhh;	Select memory block size	>
hhhhhh<	Select memory begin address	>
nn=	Select I/O timeout	>
fffppp@ or *ffpp*@	Select device type	>
cffA	Enter translation format	>
B	Blank check	>
C	Compare to port	>
D	Set odd parity	>
E	Set even parity	>
F	Error status inquiry	HHHHHHHH>
G	Configuration inquiry	DD>
H	No operation	>
I	Input from port	>
J	Set 1 stop bit	>
K	Set 2 stop bits	>
L	Load RAM from device	>
*hh*M	Enter record size	>
N	Set no parity	>
O	Output to port	>
P	Program device	>
Q	Swap nibbles	>
R	Return status of device	AAAA/BB/C>
S	View sumcheck	HHHH>
T	Illegal-bit test	>
*hh*U	Set nulls	>
V	Verify device	>
*hhhhhhhh*W	Set I/O offset	>
X or *n*X	Error code inquiry	HH....HH>
Y	Display parity errors	HHHH>
Z	Exit remote control	none
[	View device family/pinout code	FFFPPP>
\	Move memory block	>
hh^	Clear/fill RAM with data	>

Summary of Extended CRC Commands

Command	Description	Response
01]	Display system configuration	SSSS / AAAA / MM / PP / II / JJ>
*nn*02]	Set upload wait time	>
*n*03]	Set device ID verify option	HHHHHHHH> or
*nn*04]	Set Handler port baud rate	>
*xxx...xxxx*05]	Set host command	>
*n*06]	Select data bits	>
*n*07]	Set next set member	>
*n*08]	Select programmer mode	>
*xx*09]	Set set size	>
*nn*22]	Set data word width	>
*n*23]	Select number of verify passes	>
*n*24]	Select security fuse programming option	>
*n*26]	Specify logic verify options	>
*n*27]	Set / clear enable / disable sec. fuse	>
*n*28]	Fill fuse map	>
*n*29]	Set reject count option	>
*hhh*2A] or *hh*2A]	Enable programming options	>
*hhh*2B] or *hh*2B]	Disable programming options	>
*nhh*2C]	Select memory fill option	>
*hh*2D]	Vector test options	
*nn*2F]	Return 8-character sumcheck	HHHHHHHH>
*xxx...xxxx*30]	Set data file name	>
*n*31]	Set data source / destination	>
*xxx...xxxx*33]	Select device manufacturer	>
*xxx...xxxx*34]	Select device part number	>
*xxx...xxxx*38]	Load file from disk	>
*xxx...xxxx*3B]	Delete disk file	>

Command	Description	Response
*n*3C]	Set data transfer port	>
*xxx...xxxx*3E]	Select Keep Current file	>
39]	Delete all RAM files	>
40] or *n*40]	Upload device information	See Application Note
*n*41]	Perform self-tests and report results	AAA...AA>
43]	Upload yield tally	See Application Note
46]	Clear yield tally	>
49]	Suspend CRC mode	Displays terminal screen
*n*4A]	Get filename from disk	AAA...AA>
*n*4D	Select algorithms type	>
*n*4F	Set RAM device selection	>
*n*52]	Select algorithm media (floppy disk or MSM)	>
*xxx...xxxx*53]	Save RAM data to disk file	>
54]	Upload device footnote	See Application Note
55]	Upload device-specific message	See Application Note
56]	Upload memory verify failure	*dd*PAAAAAAAAHH*hh*
57]	Returns checksum of last operation	>
58]	Upload system ID	HHHH HHHH HHHH>
*n*59]	Enable / disable capacitor configuration test 0=enable; 1=disable	>
5A]	Display list of parameters	See Application Note
5B]	Clear vector data	>
5C]	Load system files for CM algorithm disk	>
5D]	Write system files to CM disk	>
5E]	Write algorithms to CM disk	>
*n*5F]	Select CM algorithm drive for creating CM algorithms	>
60]	Get number of sectors	dd>

Command	Description	Response
*n*61]	Get sector configuration settings	HHHH HHHH>
*nhhhhhhhh*62]	Set sector configuration settings	>
63]	Reboot the programmer	
*xxx...xxxx*64]	Select device part number for CM (use *xxx...xxxx*33] to select the manufacturer)	>
A65]	Return the software version number	
*n*66]	Set Abort on Empty Socket	>
*n*67]	Set Checksum calculation word size	>
A7	Swap bytes	>
DC]	Device check	See Application Note
EB]	Input JEDEC data from host	>
EC]	Output JEDEC data to host	>
FC]	Restore CRC entry default parameter	>
FD]	Restore user-defined CRC parameters	>
FE]	Save user-defined CRC parameters	>

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**	Format	**Code**
ASCII-BNPF	01 (05*)	RCA Cosmac	70
ASCII-BHLF	02 (06*)	Fairchild Fairbug	80
ASCII-B10F	03 (07*)	MOS Technology	81
Texas Instruments SDSMAC (320)	04	Motorola EXORcisor	82
		Intel Intellec 8/MDS	83
5-level BNPF	08 (09*)	Signetic Absolute Object	85
Formatted Binary	10	Tektronix Hexadecimal	86
DEC Binary	11	Motorola EXORmacs	87
Spectrum	12 (13*)	Intel MCS-86 Hex Object	88
POF	14	Hewlett-Packard 64000 Absolute	89
Absolute Binary	16		
LOF	17	Texas Instruments SDSMAC	90
ASCII-Octal Space	30 (35**)	JEDEC format (full)	91
ASCII-Octal Percent	31 (36**)	JEDEC format (Kernal	92
ASCII-Octal Apostrophe	32	Tektronix Hexadecimal Extended)	94
ASCII-Octal SMS	37		
ASCII-Hex Space	50 (55**)	Motorola 32 bit (S3 record)	95
ASCII-Hex Percent	51 (56**)	Hewlett-Packard UNIX Format	96
ASCII-Hex Apostrophe	52		
ASCII-Hex SMS	57	Intel OMF 386	97
ASCII-Hex Comma	53 (58**)	Intel OMF 286	98
		Intel Hex-32	99

* *This alternate code is used to transfer data without the STX start code and the ETX end code.*

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

Instrument Control Codes

The instrument control code is a 1-digit number that signals or controls data transfers. Specifically, the instrument control code can be used to implement a form of remote control that provides peripherals with flow control beyond that provided by software handshaking. When using computer remote control, the instrument control code is sent immediately preceding the 2-digit format code. The three values of the instrument control code and associated functions are described below.

0-Handshake Off	Input Function:	Send X-OFF to stop the incoming transmission. Send X-ON to resume transmission.
	Output Function:	Data transmission will be halted upon receipt of an X-OFF character; transmission will resume upon receipt of an X-ON character.
1-Handshake On	Input Function:	Transmit an X-ON character when ready to receive data; transmit X-OFF if the receiver buffer is full; transmit an X-ON if the receiver buffer is empty; transmit an X-OFF after all the data are received.
	Output Function:	Transmit a PUNCH-ON character prior to data transmission. Data transmission will be halted upon receipt of an X-OFF character and will resume upon receipt of an X-ON character. A PUNCH-OFF character is sent when the transmission is completed.
2-X-ON/X-OFF	Input Function:	Send X-OFF to stop the incoming transmission. Send X-ON to resume transmission.
	Output Function:	Transmit data only after receiving an X-ON character. Data transmission will be halted upon receipt of an X-OFF character; transmission will resume upon receipt of an X-ON character.

Note: X-ON character is a CTRL-Q, or 11 hex.
X-OFF character is a CTRL-S, or 13 hex.
PUNCH-ON character is a CTRL-R, or 12 hex.
PUNCH-OFF character is a CTRL-T, or 14 hex.

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.

Hardware Handshaking

Hardware handshaking may be used if compatible with the host interface by connecting the appropriate lines at the serial port interface.

Hardware handshake (CTS / DTR) is enabled as the default. However, if those signals aren't connected, the programming electronics sense this and communicate using software handshake (XON / XOFF). The programmer always uses software handshake regardless of whether hardware handshake is enabled.

Leader/Trailer

During output of all formats except 89 (HP 64000), a 50-character leader precedes the formatted data and a 50-character trailer follows. This leader / trailer consists of null characters. If the null count parameter is set to FF hex, then the leader / trailer is skipped. To set the null count, go to the More Commands / Configure / Edit / Communication Parameters screen and set the Number of Nulls parameter. If in CRC, use the CRC **U** command to set the null count.

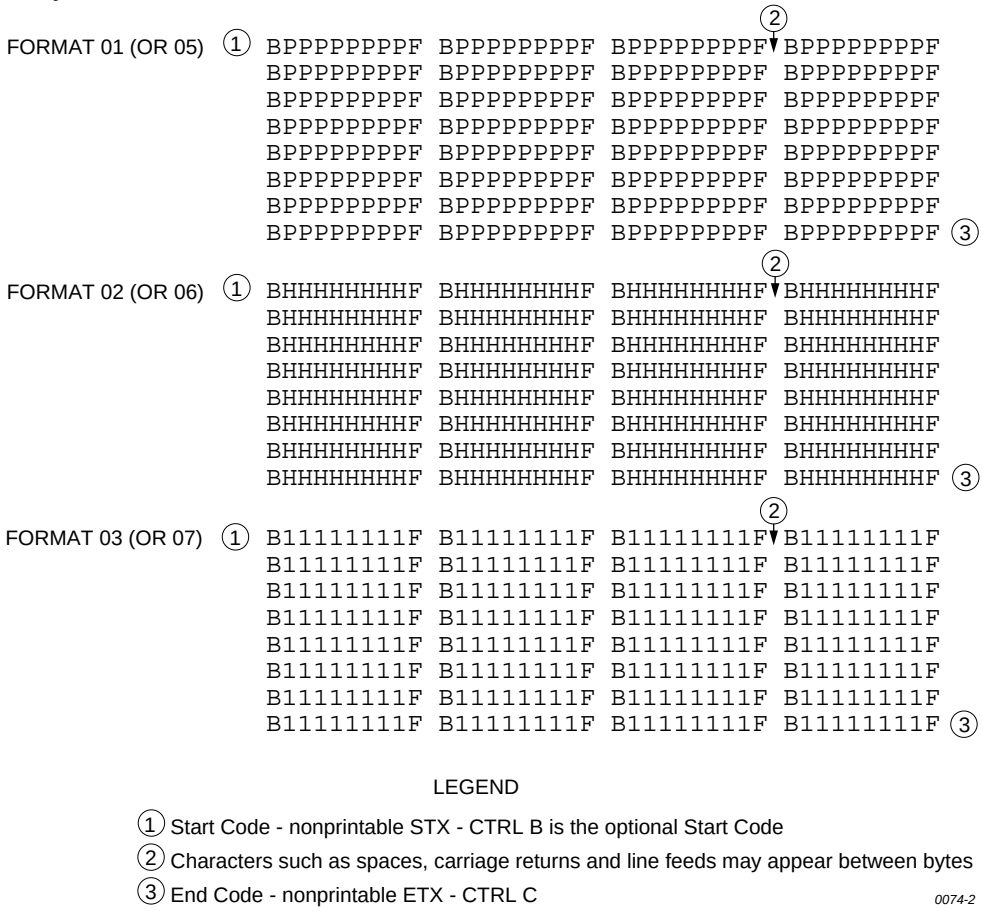
Note: Formats 10, 11, and 89 do not function properly unless you select NO parity and 8-bit data.

ASCII Binary Format, Codes 01, 02, and 03(or 05, 06, and 07)

In these formats, bytes are recorded in ASCII codes with binary digits represented by Ns and Ps, Ls and Hs, or 1s and 0s, respectively. See Figure B-1. The ASCII Binary formats do not have addresses.

Figure B-1 shows sample data bytes coded in each of the three ASCII Binary formats. Incoming bytes are stored in RAM sequentially starting at the first RAM address. Bytes are sandwiched between B and F characters and are separated by spaces.

Figure B-1
An Example of ASCII Binary Format



LEGEND

① Start Code - nonprintable STX - CTRL B is the optional Start Code

② Characters such as spaces, carriage returns and line feeds may appear between bytes

③ End Code - nonprintable ETX - CTRL C

0074-2

Data can also be expressed in 4-bit words. The programmer generates the 4-bit format on upload if the data word width is 4 bits. Any other characters, such as carriage returns or line feeds, may be inserted between an F and the next B.

The start code is a nonprintable STX, which is a CTRL-B (the same as a hex 02). The end code is a nonprintable ETX, which is a CTRL-C (the same as a hex 03).

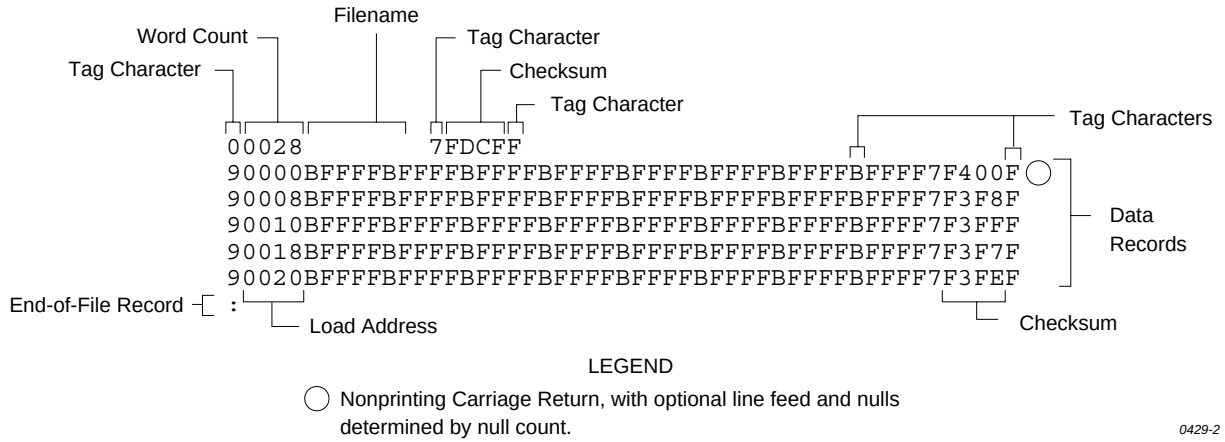
Note: Data without a start or end code may be input to or output from the programmer by use of alternate data translation format codes. These are ASCII-BNPF, 05; ASCII-BHLF, 06; ASCII-B10F, 07.

A single data byte can be aborted if the programmer receives an E character between B and F characters. Data will continue to be stored in sequential RAM addresses. Data are output in 4-byte lines with a space between bytes.

Texas Instruments SDSMAC Format (320), Code 04

Data files in the SDSMAC (320) format consist of a start-of-file record, data records, and an end-of-file record. See Figure B-2. The format is used for Texas Instruments' 320 line of processors. It is very similar to format 90; the only difference is that the address fields represent 16-bit data words rather than bytes

Figure B-2
An Example of TI SDSMAC Format



Each record is composed of a series of small fields, each initiated by a tag character. the programmer recognizes and acknowledges the following tag characters:

0 or K—followed by a file header.

7—followed by a checksum which the programmer acknowledges.

8—followed by a checksum which the programmer ignores.

9—followed by a load address which represents a word location.

B—followed by 4 data characters (16-bit word).

F—denotes the end of a data record.

*—followed by 2 data characters.

The start-of-file record begins with a tag character and a 12-character file header. The first four characters are the word count of the 16-bit data words; the remaining file header characters are the name of the file and may be any ASCII characters (in hex notation). Next come interspersed address fields and data fields (each with tag characters). The address fields represent 16-bit words. If any data fields appear before the first address field in the file, the first of those data fields is assigned to address 0000. Address fields may be expressed for any data word, but none are required.

The record ends with a checksum field initiated by the tag character 7 or 8, a 4-character checksum, and the tag character F. The checksum is the two's complement of the sum of the 8-bit ASCII values of the characters, beginning with the first tag character and ending with the checksum tag character (7 or 8).

Data records follow the same format as the start-of-file record but do not contain a file header. The end-of-file record consists of a colon (:) only. The output translator sends a CTRL-S after the colon.

During download or input from disk operations the destination address for the data is calculated in the following manner:

Memory address =
(load address x 2) – I/O address offset + begin address

During upload or output to disk operations the load address sent with each data record is calculated in the following manner:

Load address = I/O address offset / 2

The Memory begin address, I/O address offset, and User data size parameters represent bytes and must be even values for this format. The upload record size must also be even for this format (default is 16).

Note: If the data will be programmed into a 16-bit device to be used in a TMS320 processor-based system, the odd/even byte swap switch must be enabled.

The 5-Level BNPF Format, Codes 08 or 09

Except for the start and end codes, the same character set and specifications are used for the ASCII-BNPF and 5-level BNPF formats.

Data for input to the programmer are punched on 5-hole Telex paper tapes to be read by any paper tape reader that has an adjustable tape guide. The reader reads the tape as it would an 8-level tape, recording the 5 holes that are on the tape as 5 bits of data. The 3 most significant bits are recorded as if they were holes on an 8-level tape. Tape generated from a telex machine using this format can be input directly to a serial paper tape reader interfaced to the programmer. the programmer's software converts the resulting 8-bit codes into valid data for entry in RAM.

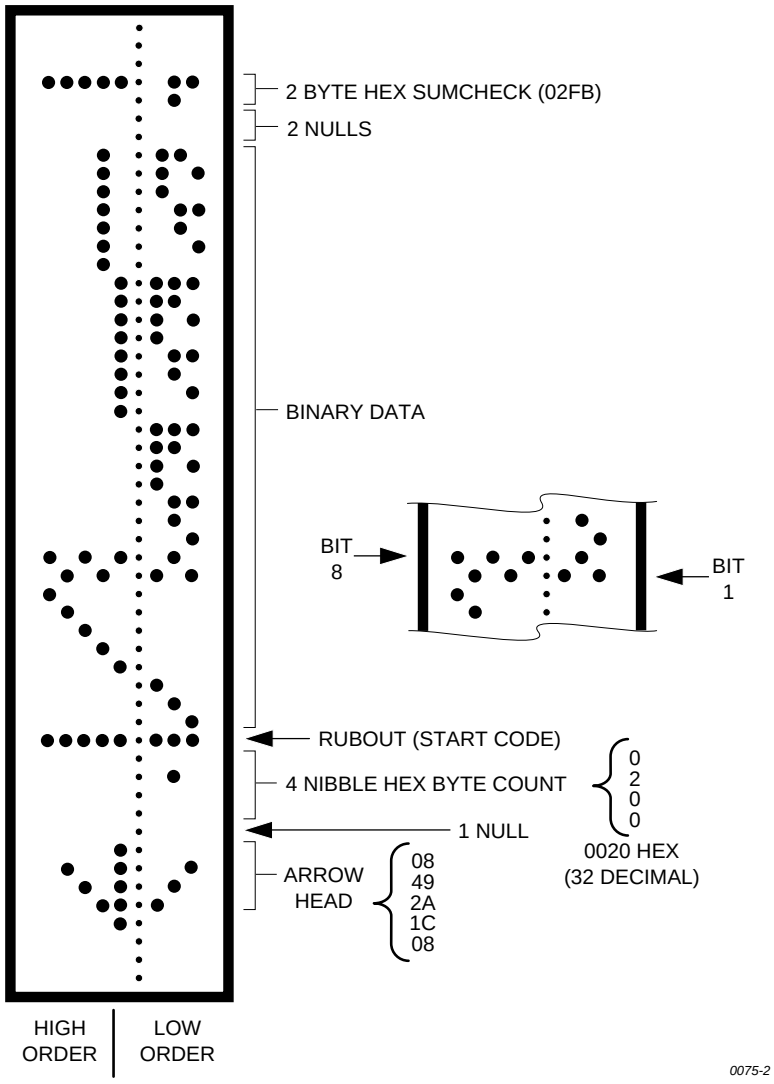
The start code for the format is a left parenthesis, (Figs K on a telex machine), and the end code is a right parenthesis, (Figs L on a telex machine). The 5-level BNPF format does not have addresses.

Note: Data without a start or end code may be input to or output from the programmer by use of the alternate data translation format code, 09. This format accepts an abort character (10 hex) to abort the transmission.

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-3. The Formatted Binary format does not have addresses.

Figure B-3
An Example of Formatted Binary Format

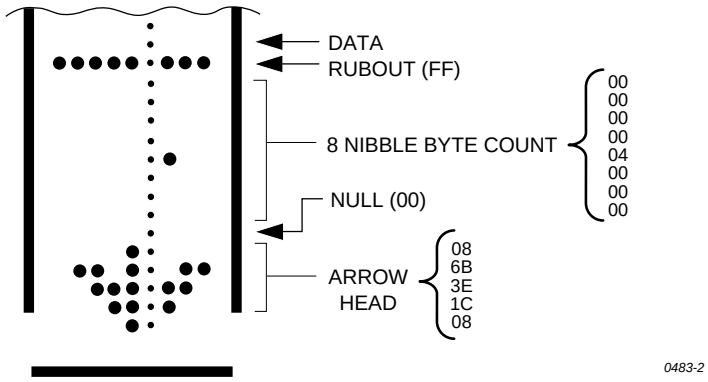


The programmer stores incoming binary data upon receipt of the start character. Data are stored in RAM starting at the first RAM address specified by the Memory Begin Address parameter and ending at the last incoming data byte.

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-4.

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-4
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.

DEC Binary Format, Code 11

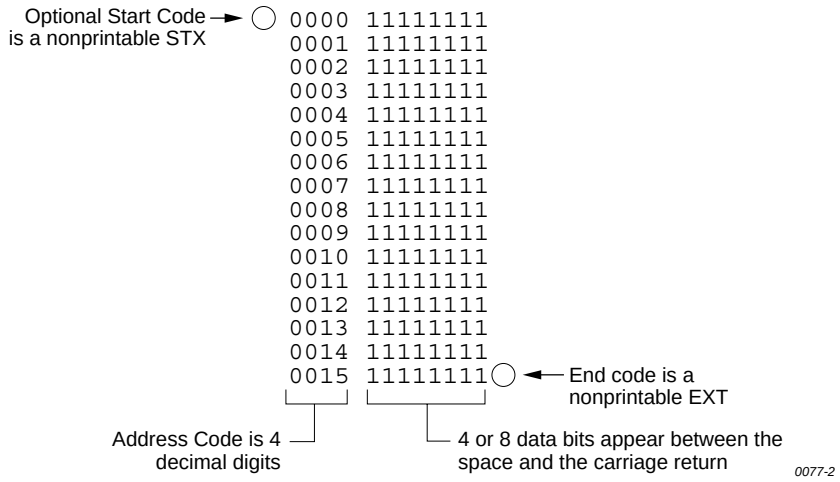
Data transmission in the DEC Binary format is a stream of 8-bit data words with no control characters except the start code. The start code is one null preceded by at least one rubout. The DEC Binary format does not have addresses.

Spectrum Format, Codes 12 or 13

In this format, bytes are recorded in ASCII codes with binary digits represented by 1s and 0s. During output, each byte is preceded by a decimal address.

Figure B-5 shows sample data bytes coded in the Spectrum format. Bytes are sandwiched between the space and carriage return characters and are normally separated by line feeds. The start code is a nonprintable STX, CTRL-B (or hex 02), and the end code is a nonprintable ETX, CTRL-C (or hex 03).

Figure B-5
An Example of Spectrum Format



Note: Data without a start or end code may be input to or output from the programmer by use of the alternate data translation format code, 13

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 B-34 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:

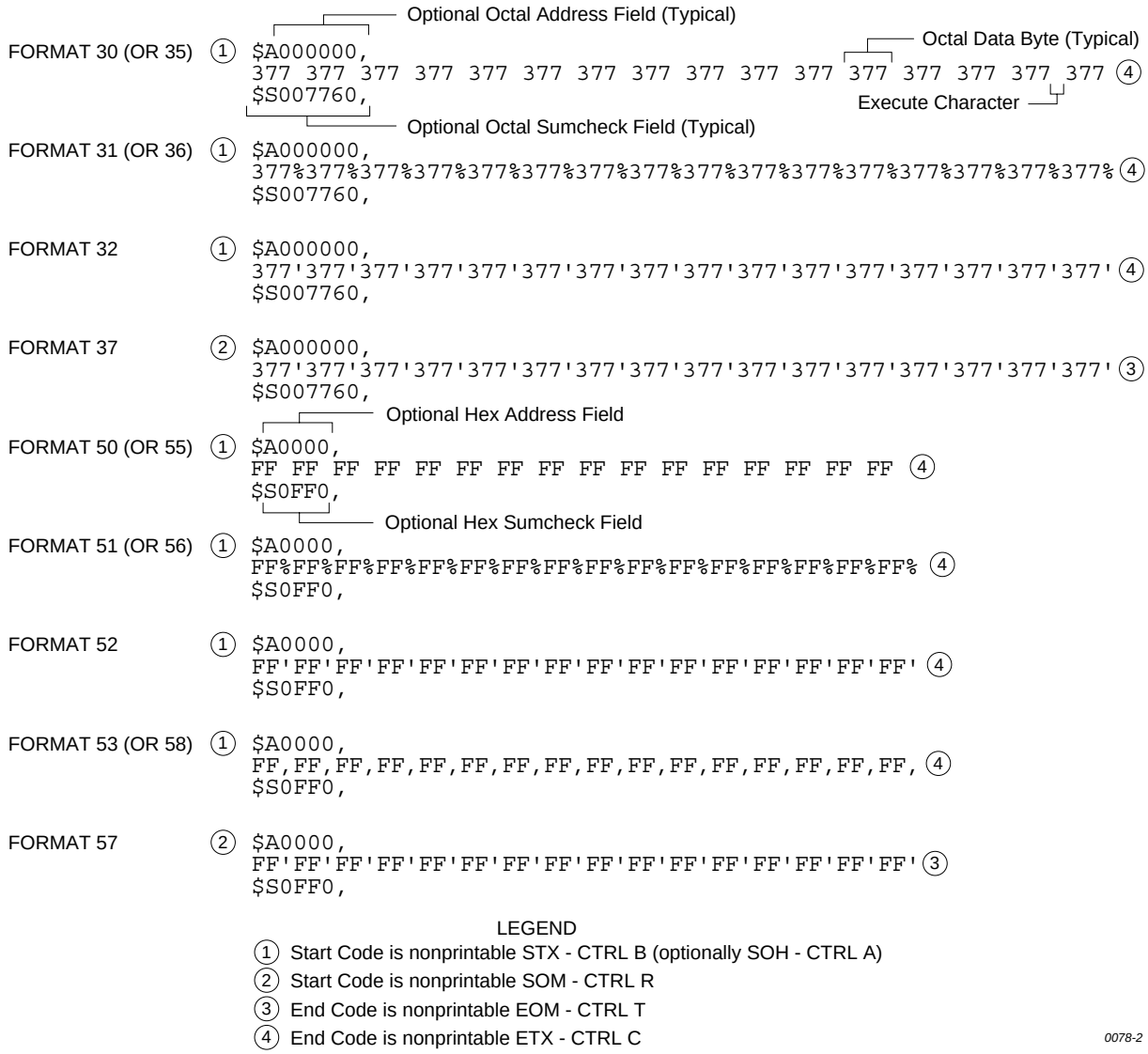
<STX>	*	Start of Data (ASCII Ctrl-B, 0x02 hex)
C	*	Fuse Checksum
K		Fuse data, followed by control words and pulse link cycles
N	*	Notes Field
QB		Number of bits per word
QC		Number of control words at the end of each K field
QF		Number of Fuses in Device (# of K fields)
QM		Number of macro cells in 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 Octal and Hex Formats, Codes 30-37 and 50-58

Each of these formats has a start and end code, and similar address and checksum specifications. Figure B-6 illustrates 4 data bytes coded in each of the 9 ASCII Octal and 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 Octal and Hex formats can express 8-bit data, by 3 octal or 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-6
An Example of ASCII Octal and Hex Formats



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 or 3 to 11 octal 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 either 2-4 hex or 3-6 octal 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 most significant digit of the sumcheck may be 0 or 1 when expressing 16 bits as 6 octal characters.

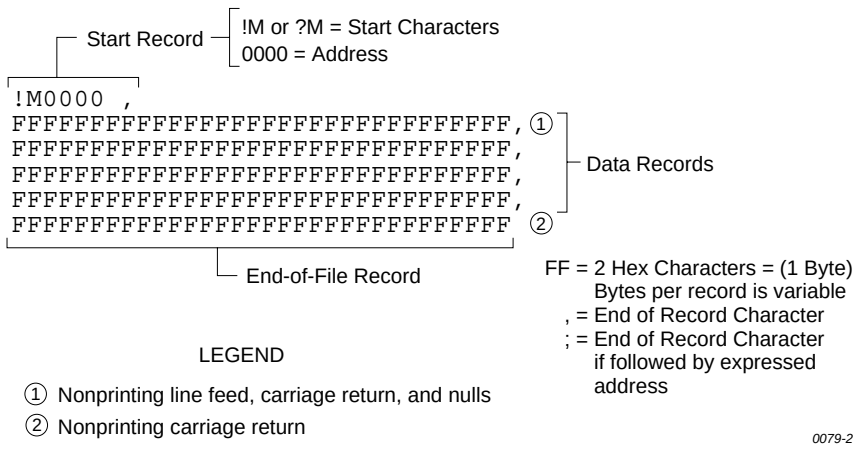
the programmer divides the output data into 8-line blocks. Data transmission is begun with the start code, a nonprintable STX character, or optionally, SOH.* Data blocks follow, each one prefaced by an address for the first data byte in the block. The end of transmission is signaled by the end code,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

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

RCA Cosmac Format, Code 70

Data in this format begin with a start record consisting of the start character (!M or ?M), an address field, and a space. See Figure B-7.

Figure B-7
An Example of RCA Cosmac Format



The start character ?M is sent to the programmer by a development system, followed by the starting address and a data stream which conforms to the data input format described in the ASCII-Hex and Octal figure. Transmission stops when the specified number of bytes has been transmitted.

Address specification is required for only the first data byte in the transfer. An address must have 1 to 4 hex characters and must be followed by a space. The programmer records the next hexadecimal character after the space as the start of the first data byte. (A carriage return must follow the space if the start code ?M is used.) Succeeding bytes are recorded sequentially.

Each data record is followed by a comma if the next record is not preceded by an address, or by a semicolon if it starts with an address. Records consist of data bytes expressed as 2 hexadecimal characters and followed by either a comma or semicolon, and a carriage return. The programmer ignores any characters received between a comma or semicolon and a carriage return.

The carriage return character is significant to this format because it can signal either the continuation or the end of data flow; if the carriage return is preceded by a comma or semicolon, more data must follow; the absence of a comma or semicolon before the carriage return indicates the end of transmission.

Output data records are followed by either a comma or a semicolon and a carriage return. The start-of-file records are expressed exactly as for input.

Fairchild Fairbug, Code 80

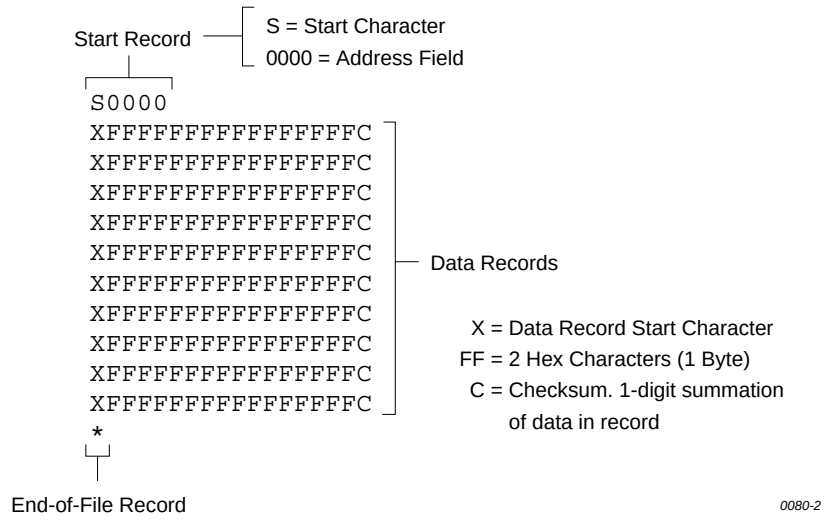
In the Fairbug format, input and output requirements are identical; both have 8-byte records and identical control characters. Figure B-8 shows a Fairbug data file. A file begins with a 5-character prefix and ends with a 1-character suffix. The start-of-file character is an S, followed by the address of the first data byte. Each data byte is represented by 2 hexadecimal characters. The programmer will ignore all characters received prior to the first S.

Note: Address specification is optional in this format; a record with no address directly follows the previous record.

Each data record begins with an X and always contains 8 data bytes. A 1-digit hexadecimal checksum follows the data in each data record. The checksum represents, in hexadecimal notation, the sum of the binary equivalents of the 16 digits in the record; the half carry from the fourth bit is ignored.

The programmer ignores any character (except for address characters and the asterisk character, which terminates the data transfer) between a checksum and the start character of the next data record. This space can be used for comments.

Figure B-8
An Example of Fairchild Fairbug



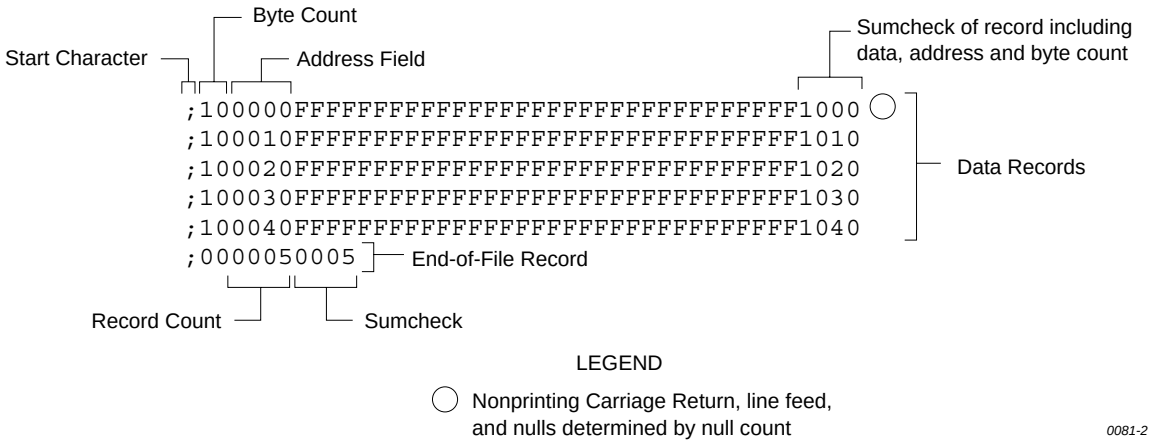
The last record consists of an asterisk only, which indicates the end of file.

MOS Technology Format, Code 81

The data in each record are sandwiched between a 7-character prefix and a 4-character suffix. The number of data bytes in each record must be indicated by the byte count in the prefix. The input file can be divided into records of various lengths.

Figure B-9 shows a series of valid data records. Each data record begins with a semicolon. The programmer will ignore all characters received prior to the first semicolon. All other characters in a valid record must be valid hexadecimal digits (0-9 and A-F). A 2-digit byte count follows the start character. The byte count, expressed in hexadecimal digits, must equal the number of data bytes in the record. The byte count is greater than zero in the data records, and equals zero (00) in the end-of-file record. The next 4 digits make up the address of the first data byte in the record. Data bytes follow, each represented by 2 hexadecimal digits. The end-of-file record consists of the semicolon start character, followed by a 00 byte count, the record count, and a checksum.

Figure B-9
An Example of MOS Technology Format

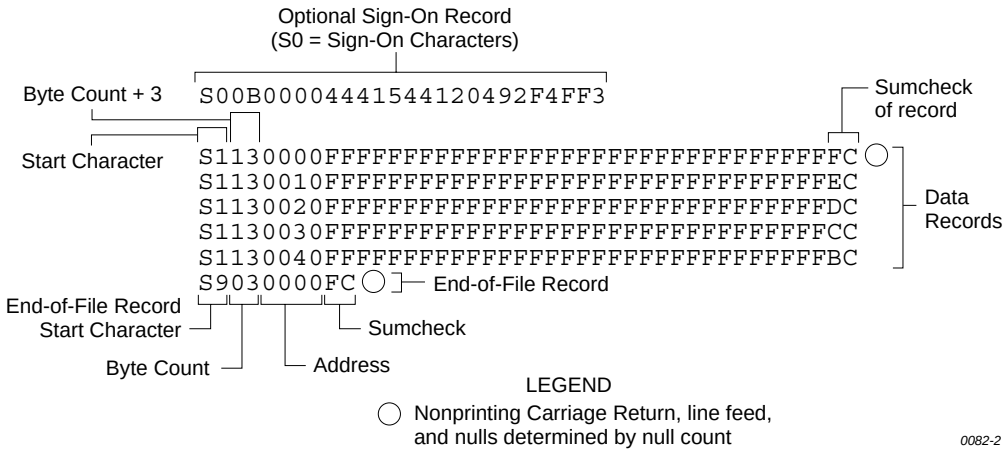


The checksum, which follows each data record, is a 2-byte binary summation of the preceding bytes in the record (including the address and byte count), in hexadecimal notation.

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-10 shows a series of valid Motorola data records.

Figure B-10
An Example of Motorola EXORciser Format



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.

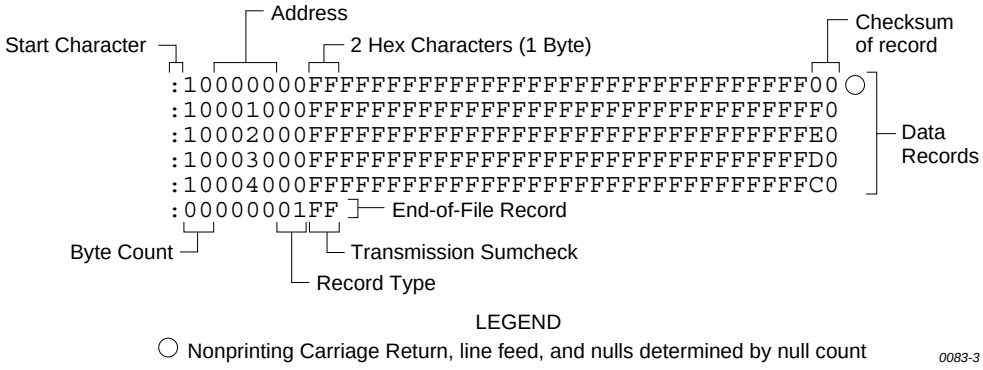
The end-of-file record consists of the start characters S9, the byte count, the address (in hex), and a checksum. The maximum record length is 250 data bytes.

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-11 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-11
An Example of Intel Intellec 8/MDS Format

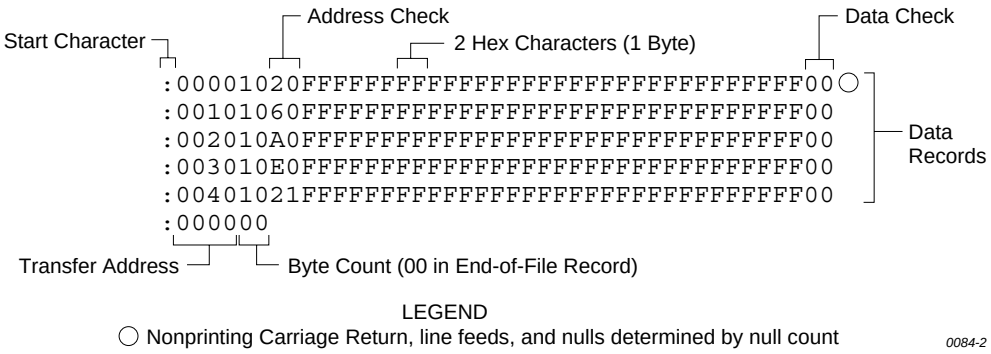


The end-of-file record consists of the colon start character, the byte count (equal to 00), the address, the record type (equal to 01), and the checksum of the record.

Signetics Absolute Object Format, Code 85

Figure B-12 shows the specifications of Signetics format files. The data in each record are sandwiched between a 9-character prefix and a 2-character suffix.

Figure B-12
An Example of Signetics Absolute Object Format



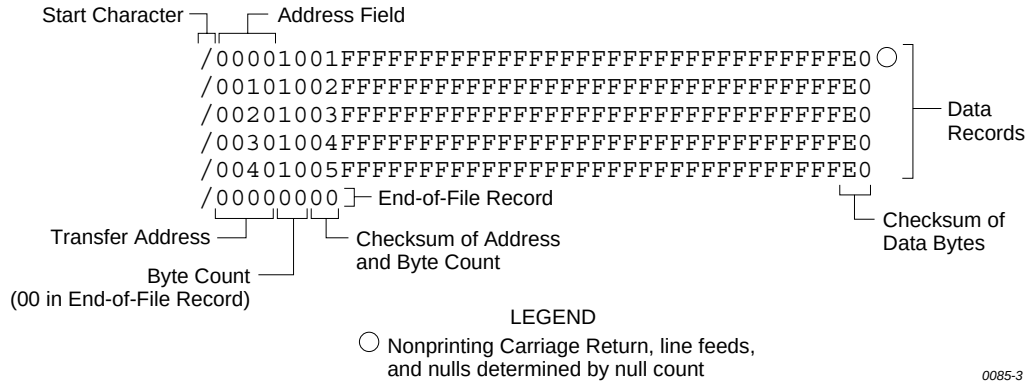
The start character is a colon. This is followed by the address, the byte count, and a 2-digit address check. The address check is calculated by exclusive ORing every byte with the previous one, then rotating left one bit. Data is represented by pairs of hexadecimal characters. The byte count must equal the number of data bytes in the record. The suffix is a 2-character data check, calculated using the same operations described for the address check.

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

Tektronix Hexadecimal Format, Code 86

Figure B-13 illustrates a valid Tektronix data file. The data in each record are sandwiched between the start character (a slash) and a 2-character checksum. Following the start character, the next 4 characters of the prefix express the address of the first data byte. The address is followed by a byte count, which represents the number of data bytes in the record, and by a checksum of the address and byte count. Data bytes follow, represented by pairs of hexadecimal characters. Succeeding the data bytes is their checksum, an 8-bit sum, modulo 256, of the 4-bit hexadecimal values of the digits making up the data bytes. All records are followed by a carriage return.

Figure B-13
An Example of Tektronix Hex Format



Data are output from the programmer starting at the first RAM address and continuing until the number of bytes in the specified block has been transmitted. The programmer divides output data into records prefaced by a start character and an address field for the first byte in the record.

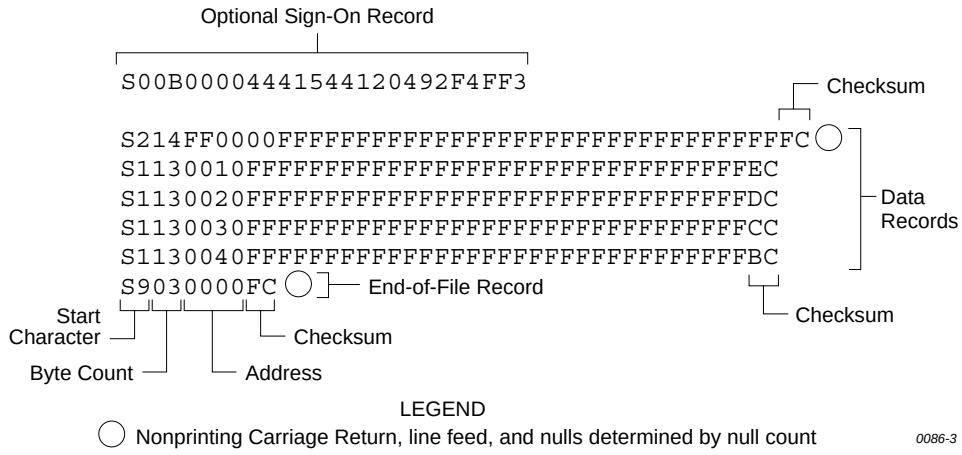
The end-of-file record consists of a start character (slash), followed by the transfer address, the byte count (equal to 00), and the checksum of the transfer address and byte count.

An optional abort record contains 2 start characters (slashes), followed by an arbitrary string of ASCII characters. Any characters between a carriage return and a / are ignored.

Motorola EXORmacs 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-14 shows a series of Motorola EXORmacs data records.

Figure B-14
An Example of Motorola EXORmacs Format



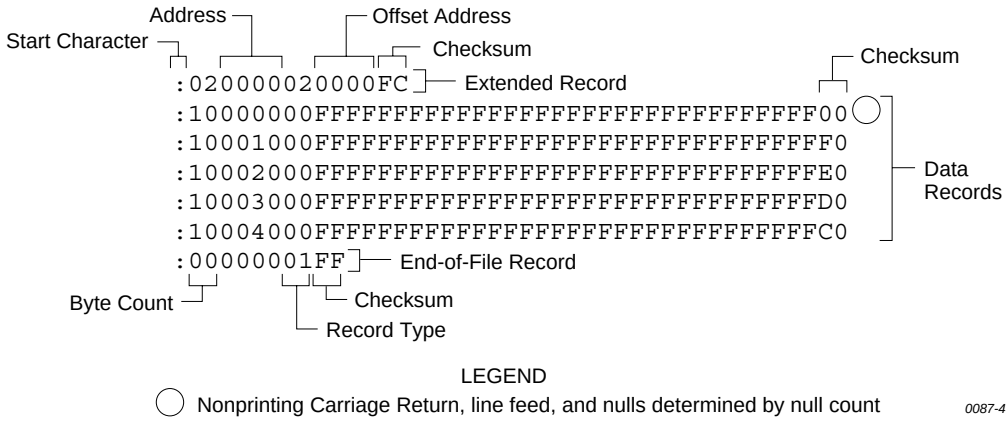
Each data record begins with the start characters S1 or S2: S1 if the following address field has 4 characters, S2 if it has 6 characters. The third and fourth characters represent the byte count, which expresses the number of data, address, and checksum bytes in the record. The address of the first 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-15 shows a sample record of this format.

Figure B-15
An Example of Intel MCS-86 Hex Object



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, as shown below:

```
  1234     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.

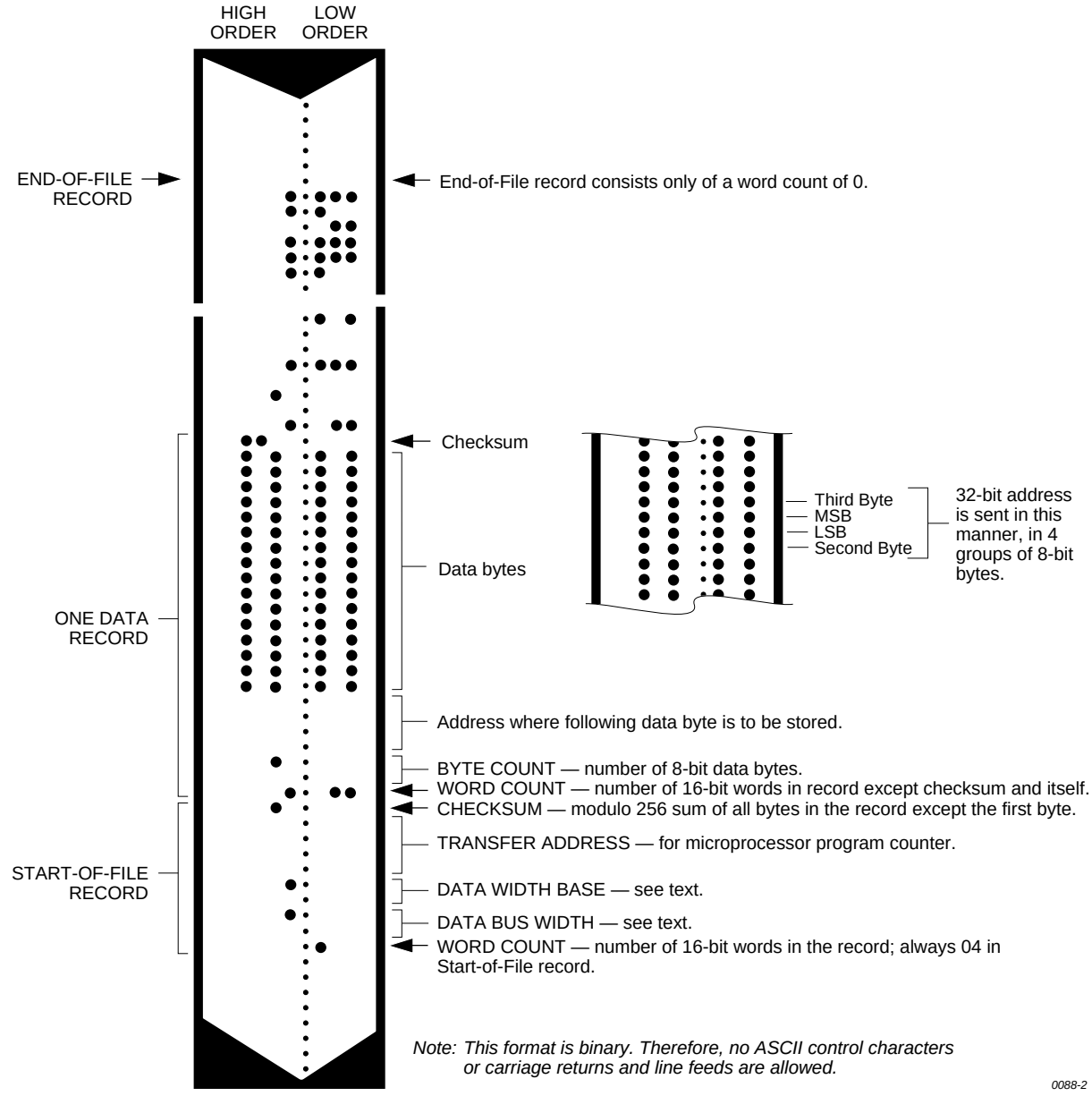
03-Start Record

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

Hewlett-Packard 64000 Absolute Format, Code 89

Hewlett-Packard Absolute is a binary format with control and data-checking characters. See Figure B-16.

Figure B-16
An Example of HP 64000 Absolute Format



Data files begin with a Start-of-file record, which includes the Data Bus Width, Data Width Base, Transfer Address, and a checksum of the bytes in the record.

The Data Bus Width represents the width of the target system's bus (in bits). The Data Width Base represents the smallest addressable entity used by the target microprocessor.

The Data Bus Width and Data Width Base are not used by the programmer during download. During upload, the Data Bus Width will be set to the current Data Word Width, and the Data Width Base will be set to 8. The Transfer Address is not used by the programmer.

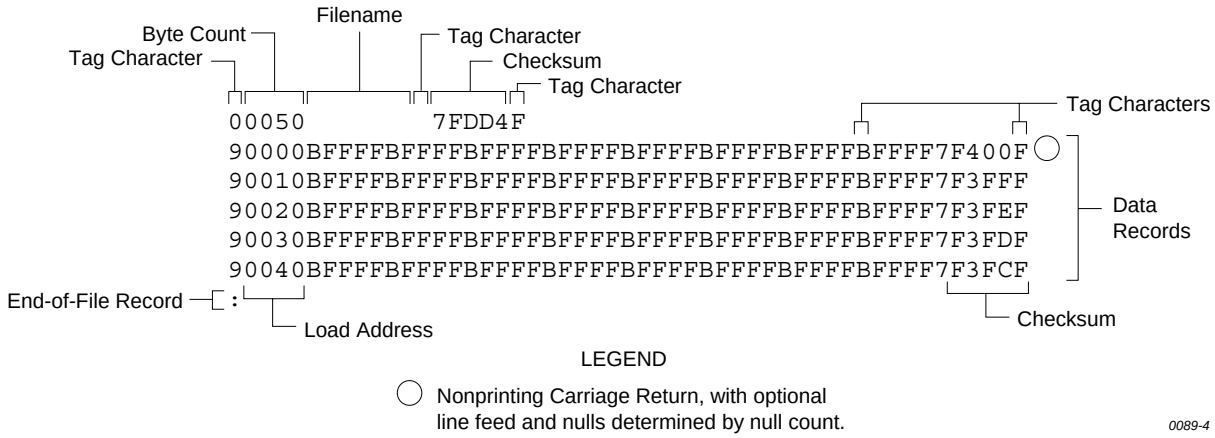
Data records follow the Start-of-file record. Each begins with 2 byte counts: the first expresses the number of 16-bit bytes in the record, not including the checksum and itself; the second expresses the number of 8-bit data bytes in the record. Next comes a 32-bit address, which specifies the storage location of the following data byte. Data bytes follow; after the last data byte is a checksum of every byte in the record except the first byte, which is the word count.

The End-of-file record consists of a one byte word count, which is always zero. Leader and trailer nulls, normally 50 each, are suppressed in this translation format.

Texas Instruments SDSMAC Format, Code 90

Data files in the SDSMAC format consist of a start-of-file record, data records, and an end-of-file record. See Figure B-17.

Figure B-17
An Example of TI SDSMAC Format



Each record is composed of a series of small fields, each initiated by a tag character. The programmer recognizes and acknowledges the following tag characters:

0 or K—followed by a file header.

7—followed by a checksum which the programmer acknowledges.

8—followed by a checksum which the programmer ignores.

9—followed by a load address.

B—followed by 4 data characters.

F—denotes the end of a data record.

*—followed by 2 data characters.

The start-of-file record begins with a tag character and a 12-character file header. The first four characters are the byte count of the data bytes; the remaining file header characters are the name of the file and may be any ASCII characters (in hex notation). Next come interspersed address fields and data fields (each with tag characters). If any data fields appear before the first address field in the file, the first of those data fields is assigned to address 0000. Address fields may be expressed for any data byte, but none are required.

The record ends with a checksum field initiated by the tag character 7 or 8, a 4-character checksum, and the tag character F. The checksum is the two's complement of the sum of the 8-bit ASCII values of the characters, beginning with the first tag character and ending with the checksum tag character (7 or 8).

Data records follow the same format as the start-of-file record but do not contain a file header. The end-of-file record consists of a colon (:) only. The output translator sends a CTRL-S after the colon.

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.

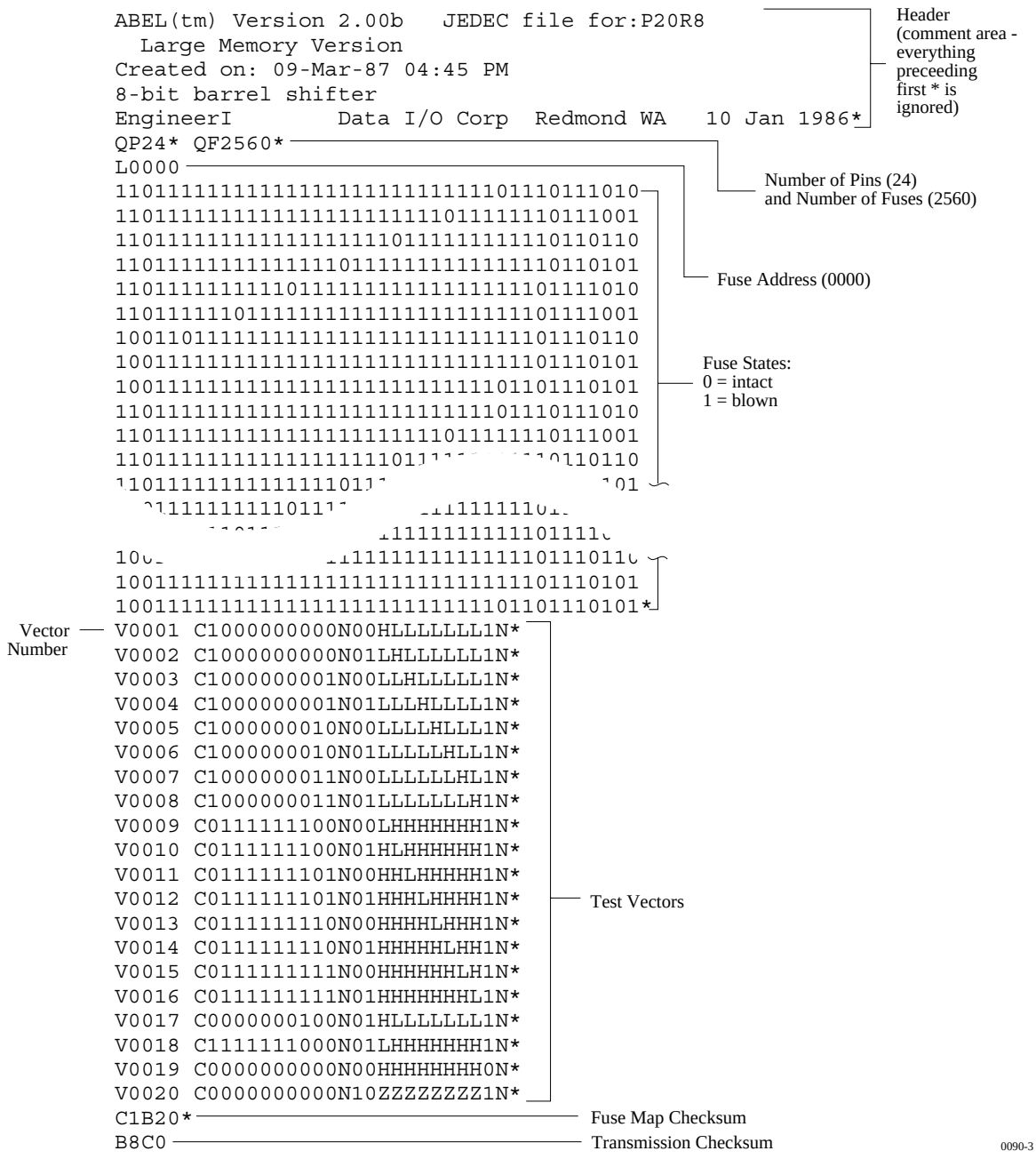
Some computer systems do not allow you to control what characters are sent, especially at the end of a line. You should set up the equipment so that it will accept a dummy value of 0000 as a valid checksum. This zero checksum is a way of disabling the transmission checksum while still keeping within 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-18. Each of the fields is described on the following pages.

Figure B-18
An Example of JEDEC Full Format



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

01011101011010000100100100 10*

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 111000HLHHNNNNNNNNNNN*

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 2.5, 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
10101100000000000000000*
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
10101100000000000000000*
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:

```
V0001 C01010101NHLLLHHLHLN *
V0002 C01011111NHLLHLLLHLN *
V0003 C10010111NZZZZZZZZZN *
V0004 C01010100NFLHHLFFLLN *
```

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-19 shows an example of a kernel JEDEC transmission.

Figure B-19
An Example of JEDEC Kernel Mode Format

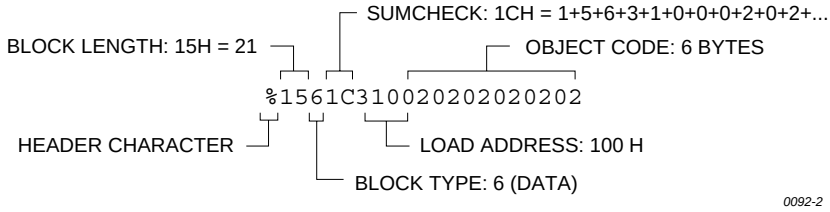
```
<STX>
Acme Logic Design  Jane Engineer   Feb. 29 1983
Widget Decode  756-AB-3456 Rev C Device Mullard 12AX7*

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

Extended Tektronix Hexadecimal Format, Code 94

The Extended Tektronix Hexadecimal format has three types of records: data, symbol, and termination records. The data record contains the object code. Information about a program section is contained in the symbol record (the programmer ignores symbol records), and the termination record signifies the end of a module. The data record (see sample below) contains a header field, a load address, and the object code. Figure B-20 lists the information contained in the header field.

Figure B-20
An Example of Tektronix Extended Format



Item	No. of ASCII Characters	Description
%	1	Signifies that the record is the Extended Tek Hex format.
Block length	2	Number of characters in the record, minus the %.
Block type	1	6 = data record 3 = symbol record (ignored by the programmer) 8 = termination record
Checksum	2	A 2-digit hex sum, modulo 256, of all the values in the record except the % and the checksum.

Character Values for Checksum Computation

The number of fields in the file will vary, depending on whether a data or a termination block is sent. Both data and termination blocks have a 6-character header and a 2-to-17 character address.

Character(s)	Value (decimal)	Character(s)	Value (decimal)
0 . . 9	0 . . 9	. (period)	38
A . . Z	10 . . 35	_(underline)	39
$	36	a . . z	40 . . 65
%	37		

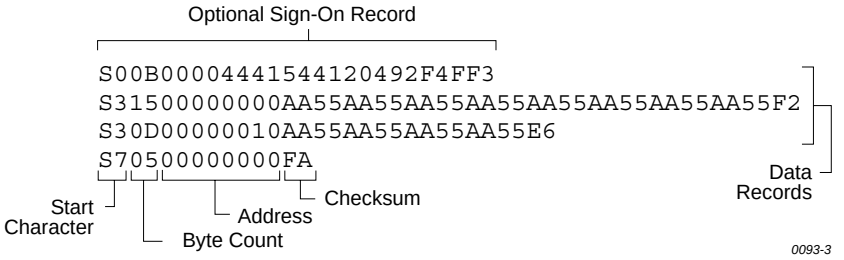
The load address determines where the object code will be located. This is a variable length number that may contain up to 17 characters. The first number determines the address length, with a zero signifying a length of 16. The remaining characters of the data record contain the object code, 2 characters per byte.

When you copy data to the port or to RAM, set the high-order address if the low-order is not at the default value.

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-21 shows a sample of the Motorola 32-bit format.

Figure B-21
An Example of Motorola S3 Format



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.

Hewlett-Packard UNIX Format, Code 96

This format divides the data file into data records, each with a maximum size of 250 bytes not including header information. An ID header is added to the beginning of the first record. Each subsequent record has its own header section. The section at the beginning of the file contains the following elements: the header 8004, filename, byte count for the processor information record, and the processor information record.

The header 8004 identifies the type of file being transferred. The first byte of this header (80) indicates that this file is binary, and the 04 indicates the type of file (absolute).

The ID header is followed by a 16-byte filename (not used by the programmer).

Next is the byte count, which indicates the size (minus one) of the Processor Information Record that follows. The Processor Information Record is divided into the following data words: Data Bus Width, Data Width Base, Transfer Address LS (least significant), and Transfer Address MS (most significant).

The Data Bus Width represents the width of the target system's bus (in bits). The Data Width Base represents the smallest addressable entity used by the target microprocessor.

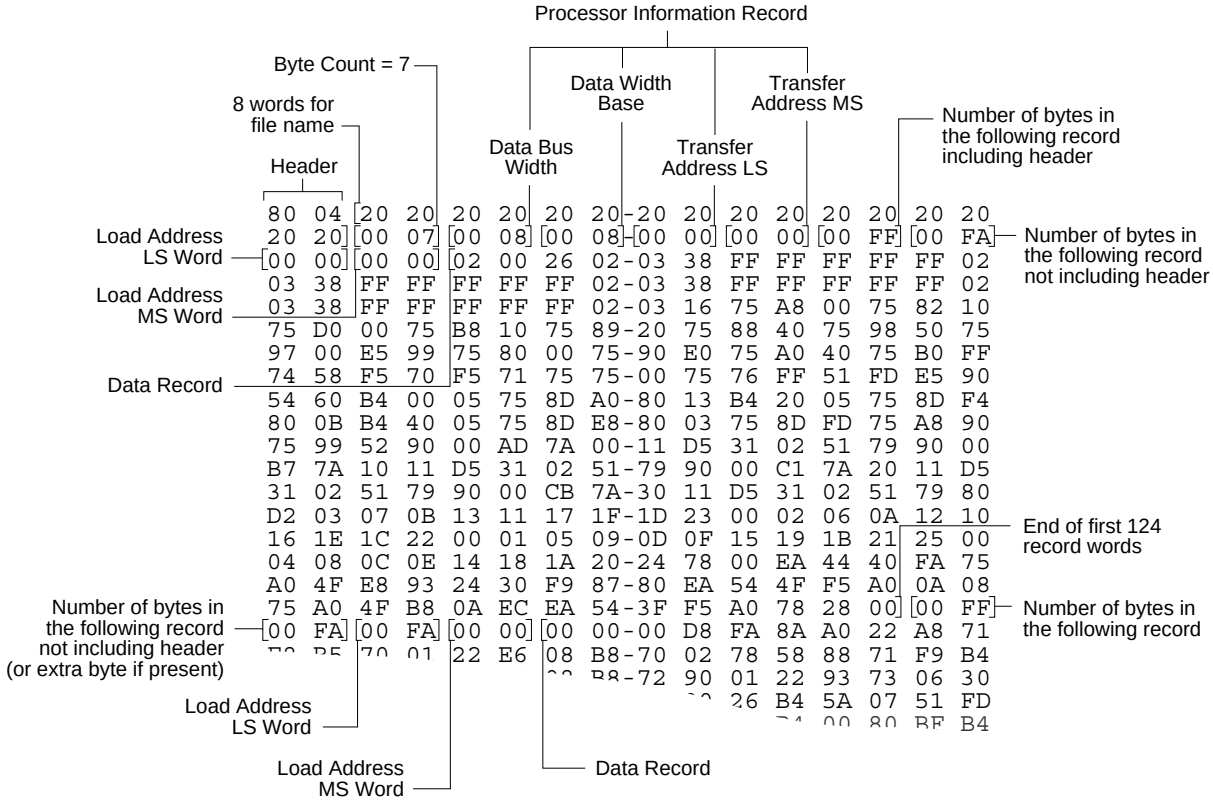
The Data Bus Width and Data Width Base are not used by the programmer during download. During upload, the Data Bus Width will be set to the current Data Word Width, and the Data Width Base will be set to 8. The Transfer Address LS and Transfer Address MS are not used by the programmer.

The data records consist of a header (8 bytes) and the data bytes. The first 2 bytes of the header indicate the size of the data record including the header (minus one). If the number of data bytes in the data record (not including the header) is odd, one extra byte will be added to the data record to ensure that an even number of data bytes exist in the data record. The maximum value for this field is 00FF hex. The next two bytes indicate the number of actual data bytes in the record, not including the header bytes and the extra byte (if present). The maximum value for this field is 00FA hex. The 4 bytes that follow represent the destination address for the data in this record. The rest of the bytes in the record are the data bytes.

This format has no end of file identifier.

The record length during upload is not affected by the upload record size parameter in the Configure / Edit / Communication screen. It is automatically set to transfer records using the maximum size (250 bytes) except for the last record. The size of the last record will be set according to the remaining number of data bytes.

Figure B-22
Hewlett-Packard 64000 Unix Format



This data translation format was generated by a "dump utility" for illustrative purposes. Actual data files are in binary code and are typically generated by the appropriate development software.

0474-2

Intel OMF386 Format, Code 97

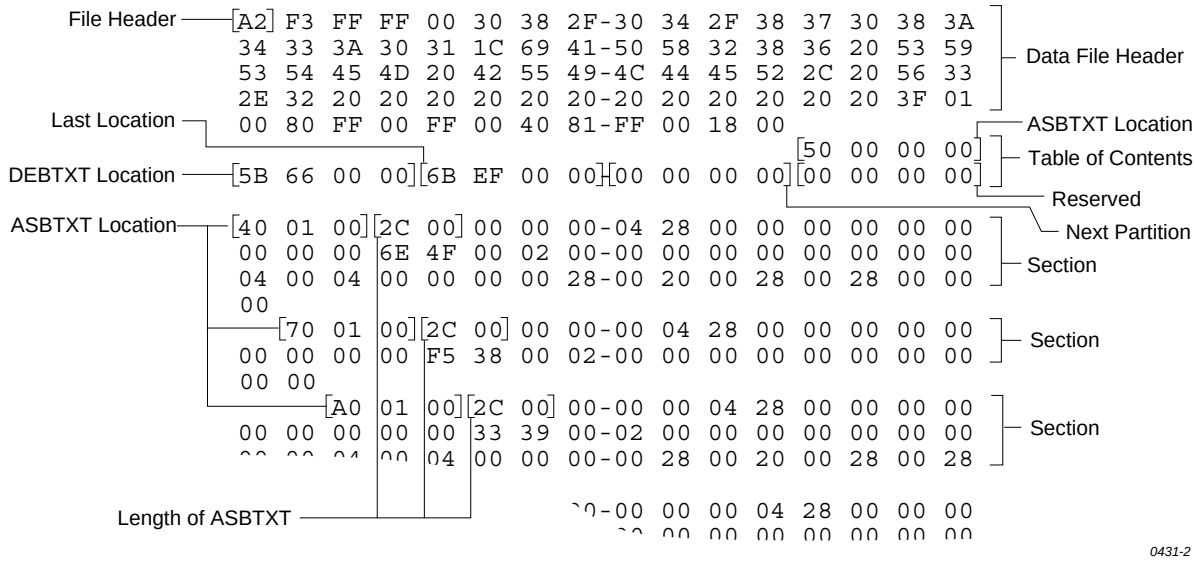
This data translation format is considered by Intel to be proprietary information. Contact your local Intel representative or call (408) 987-8080 for information about the structure of this format.

Intel OMF286 Format, Code 98

The Intel OMF286 format is a dynamically allocatable file format.

This format has three basic parts: the file header, data file module, and a 1-byte checksum. The file header is hexadecimal number (A2) that identifies this file as an Intel OMF 286 format file. See Figure B-23.

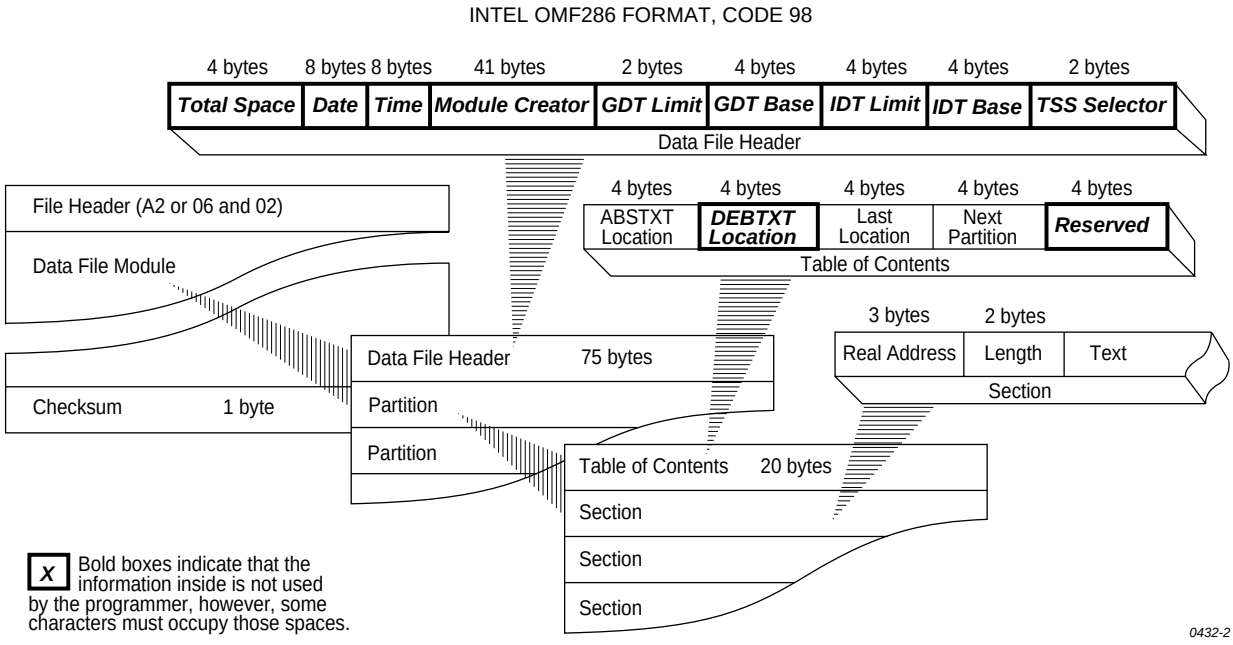
Figure B-23
A Sample of the Intel OMF286 Format



The first 75 bytes of the data file module is the data file header. The header information is generated and used by the development system and is not used by the programmer, although some characters must fill those bytes. The rest of the data file module consists of one partition.

The partition begins with a 20 byte table of contents. The table of contents specifies the locations of ABSTXT (absolute text), DEBTXT (debug text), the last location of this partition, and the location of the next partition. The OMF286 format consists of only one partition so this field will be zeros. The rest of the partition consists of sections. The actual data are located in the sections. The first 3 bytes in each section specify the real address of the text. The next 2 bytes state the length of the text, and the remainder of the section is the text (or data). Following the final section of the final partition is a 1-byte checksum representing the complement of the sum of all the bytes in the file, including the header. The sum of the checksum byte and the calculated checksum for the file should equal zero. The programmer ignores this checksum.

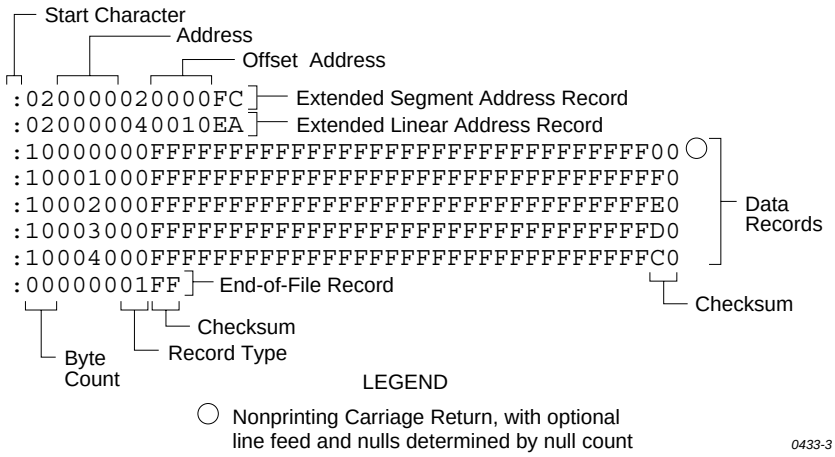
Figure B-24
A Close-up of the Intel OMF286 Format



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-25 illustrates the sample records of this format.

Figure B-25
An Example of the Intel Hex-32 Format



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)
01-03	ASCII (BNPF, BHLF, and B10F)	N/A
04	Texas Instruments SDSMAC (320)	1FFFF (FFFF words)
05-07	ASCII (BNPF, BHLF, and B10F)	N/A
11	DEC Binary	N/A
12-13	Spectrum	270F
16	Absolute Binary	N/A
17	LOF	N/A
30-32	ASCII-Octal (Space, Percent, and Apostrophe)	3FFFF (777777 octal)
35-37	ASCII-Octal (Space, Percent, and SMS)	3FFFF (777777 octal)
50-52	ASCII-Hex (Space, Percent, and Apostrophe)	FFFF
55-58	ASCII-Hex (Space, Percent, SMS, and Comma)	FFFF
70	RCA Cosmac	FFFF
80	Fairchild Fairbug	FFFF
81	MOS Technology	FFFF
82	Motorola EXORciser	FFFF
83	Intel Intellec 8/MDS	FFFF
85	Signetics Absolute Object	FFFF
86	Tektronix Hexadecimal	FFFF
87	Motorola EXORmacs	FFFFFF
88	Intel MCS-86 Hex Object	FFFFF
89	Hewlett-Packard 64000 Absolute	FFFFFFFF
90	Texas Instruments SDSMAC	FFFF
91, 92	JEDEC (Full and Kernel)	N/A
94	Tektronix Hexadecimal Extended	FFFFFFFF
95	Motorola 32 bit (S3 record)	FFFFFFFF
96	Hewlett-Packard UNIX Format	FFFFFFFF
97	Intel OMF 386	FFFFFFFF
98	Intel OMF 286	FFFFFF
99	Intel Hex-32	FFFFFFFF

Keep Current Subscription Service

The Keep Current™ subscription service keeps your programmer up-to-date with the latest features and device support. You gain immediate access to new and improved programming algorithms via the Keep Current Library accessible through the Internet or via the Data I/O BBS.

Semiconductor companies constantly introduce new devices and issue specification changes for existing devices. Incorporating these changes swiftly into your programming system ensures that you obtain the highest programming yields and best device reliability possible. Periodic update kits incorporate all changes since the previous update.

The Keep Current device support files are located on the Data I/O Bulletin Board System, on the Data I/O Web page, and through anonymous FTP.

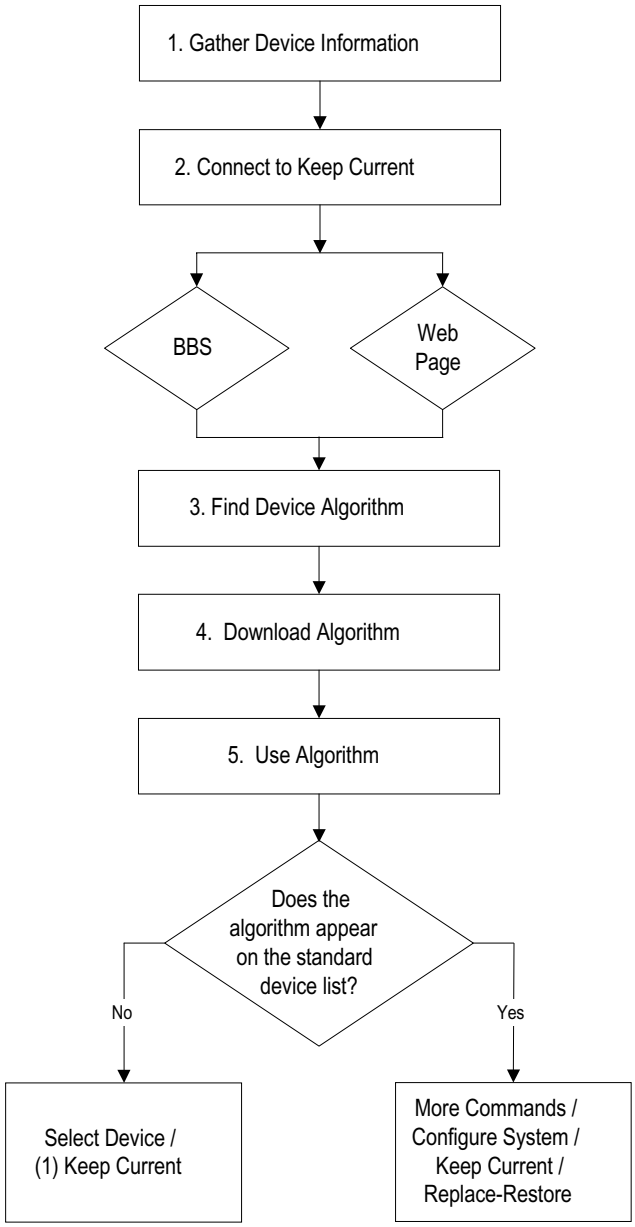
Computer Requirements

To access and download the Keep Current files, you need the following:

- Ability to create 3.5-inch DOS disks: 720KB if you are using a UniSite, 1.44MB for all other programmers.
- The ability to connect to the Keep Current Library through a modem or the Internet.

Note: The modem for BBS connection must be capable of handling 2400 baud or greater. Modem speeds less than 2400 baud are no longer supported.

Procedure Overview


1. Gather Device Information
2. Connect to Keep Current
BBS
Web Page
3. Find Device Algorithm
4. Download Algorithm
5. Use Algorithm
Does the algorithm appear on the standard device list?
No
Yes
Select Device / (1) Keep Current
More Commands / Configure System / Keep Current / Replace-Restore

1. Gather Information

Knowing the following information about the devices you will be programming will enable you to find the correct algorithm once you are connected.

- Manufacturer (*example*: AMD)
- Device name (*example*: 27c1024)
- Package type (*example*: 48-pin PLCC)
- Current version of the programmer software (*example*: 5.5)

2. Connect to Keep Current

Using the BBS

Use the following procedure to log on to the Keep Current BBS:

1. **Modem:** Call **425-882-3211**.
   Internet: Using telnet, go to **bbs.data-io.com**.
2. If this is the first time you have called, when the logon screen appears, type **new** at the prompt to create a new account and provide the requested information.

Note: Be sure to write down your user name and password for future reference. This is your personal BBS account and should not be shared.

3. Press **L** to go to the Library menu.
4. Press **S** to select the Keep Current Library.
5. Press **F** to search for algorithms.

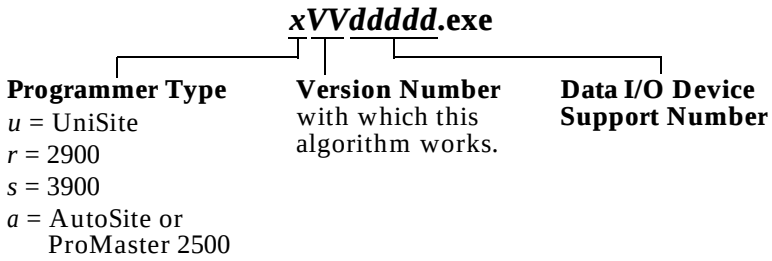
Using the Web

The Data I/O Home Page is located at www.data-io.com. Click on the Keep Current image from the Home Page or from the Programmer Device Support page.

3. Find Device Algorithm

When you reach the Keep Current Library, select the correct algorithm. Algorithms are arranged by programmer and system software version.

Keep Current filenames are represented as follows:



Each Keep Current algorithm is designed to work with a particular version of system software. Only algorithms that are compatible with the installed version of system software are displayed on the programmer's Keep Current Part List screen.

A Keep Current algorithm and a version of your programmer's system software are compatible when the numbers to the left and immediate right of the decimal point match, as shown in the following example:

Algorithm Version	System Software Version	Compatible?
3.51	3.5	Yes
3.7	3.7	Yes
3.6	3.7	No

Note: Keep Current algorithms are valid for only one major release of software because they will be included with the next release of system software.

4. Download Algorithm

Algorithms come in a self-extracting file format. Place the Keep Current file on a floppy disk that has been formatted on your programmer, and then expand the file by running it. The following files should be created:

File Name	Description of File
xVVddddd.KCx	Algorithm
xVVddddd.txt	Instructions on use
adapters. sys*	System adapters
devfnote.sys*	Device notes

* *Optional file*

Label the disk **Keep Current** and specify the version number to avoid mismatched software version numbers you need to use the disk again.

You can also create these files in a sub-directory and transfer the files to a disk using your programmer in terminal mode. Use the **More Commands/Transfer/Download** screen, select format **16** (Absolute Binary), and select **Disk** as the destination.

5. Use Algorithm

Before you use a Keep Current algorithm, determine whether or not it appears on the programmer's standard device list and follow the directions in the appropriate section below.

On Standard List

If the algorithm is listed in the standard device list, in terminal mode use **More Commands / Configure System / Keep Current / Replace-Restore.** This command adds the Keep Current algorithm to the device list, ensuring that the latest algorithm is available to all programmer users.

From this point forward, the operation is the same as using a regular algorithm. Refer to your programmer User Manual for instructions.

Not on Standard List

If the algorithm is not listed in the standard device list, in terminal mode use **Select Device / (1) Keep Current**. The algorithm can be selected from the Select Device menu, but it cannot be added to the device list.

From this point forward, the operation is the same as using a regular algorithm. Refer to your programmer User Manual for instructions.

Sample Keep Current Scenario

The following example illustrates a typical Keep Current scenario:

1. In May, you update your system software to version *X*.4. At the same time, you enroll in the Keep Current Subscription Service.
2. In June, Cruft Technologies announces a new device, the Cruft 1263.
3. A week later, Data I/O announces support for the Cruft 1263 and places a Keep Current algorithm for the Cruft 1263 on the Keep Current BBS and the Data I/O Web page.
4. The next day you connect to the Keep Current Library via the BBS or the Web page and download the new algorithm for the Cruft 1263.
5. In August, Data I/O releases version *X*.5 system software, complete with the new algorithm for the Cruft 1263.
6. You update your programmer to version *X*.5 system software. The algorithm for the Cruft 1263 is part of the system software.

Glossary

Action Symbol — Found in the upper left-hand corner of the screen, the action symbol rotates to indicate that the programmer is performing an operation.

Address — A coded instruction designating the location of data or program segments in storage.

Address Offset — A value subtracted from addresses during input translation and then added to addresses during output translation.

Algorithm — The software file containing information 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.

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.

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.

Communications Parameters — The various settings that determine the I/O characteristics of your 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.

Compensated Vector Test A device test that enables load compensation on PLD output pins under test during vector testing. This may eliminate structured test error when testing PLDs sensitive to output loading, where many of the devices register transitions simultaneously.

Computer Remote Control A command set that may be used to operate a programmer remotely. These commands are usually the basis for external programmer drivers, which may operate a programmer from a PC or other host. See also Remote Mode.

Continuity Check A device check that tests for open device pins before performing a device operation.

CRC An acronym for Computer Remote Control. See Remote Mode and Computer Remote Control.

Cross Programming A programming operation that allows a single generic programmable logic device (PLD) to be configured as any one of many PLD architectures. Consequently, the generic device can take on the function of many subset devices. As an example, a 16V8 generic PLD can be configured as a 16R4, 16R8, 16L8, etc.

Data Bits A communication parameter that specifies the number of bits per byte.

Data Word Width The word width of the data to be used during a device operation. For 8-bit (or above) devices, the maximum 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 you are sending something. The “something” you are sending is almost always data. The destination can be RAM, a disk file, or one of the programmer’s serial 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 Support Packages The method of device support for AutoSite. The device algorithms have been organized according to package type and pin count.

Device Operation Usually a term that refers to loading, programming, or verifying. However, 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 A type of device package. An acronym for Dual In-line Package.

Download Data A file operation that moves a data file from a host computer to the programmer's RAM or disk.

Download Echoing Displays the data being downloaded.

Download Host Command A command that is sent from the programmer to the host during a download. The command tells the host to begin sending data to the programmer.

E-MICRO An acronym for Programmable Microcontroller. A type of device technology.

EPROM An acronym for Erasable Programmable Read-Only Memory. (Usually refers to UV erasable memories.)

EEPROM An acronym for Electronically 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 identify 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.

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.

File Transfer Operations An operation involving the transfer of data between the programmer and a host. Upload and download are file transfer operations.

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 valid filenames would be **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.

Handshaking The required sequence of signals for communication between two units. The I/O bus protocol for a unit defines its handshaking requirements. This is especially true for asynchronous I/O systems in which each signal requires a response to complete an I/O operation.

High-speed Logic Drivers A device test that increases the speed of the logic transitions between 0 to 1 and 1 to 0 of the test vector input states. This test is a diagnostic tool designed to help debug and classify test vector failures. Specifically, this test is designed to help identify vector transitions that are speed dependent.

Host A micro-mini, or mainframe computer used to control AutoSite in Remote mode. You must use a software driver, such as Data I/O's TaskLink, to allow the computer to communicate with AutoSite.

Host Command (download & upload) The command that is sent from AutoSite to the host system during uploading/downloading. See Download Host Command and Upload Host Command.

I/O Address Offset This value influences the beginning address where data is stored during a file transfer operation. For uploads, the I/O Offset represents the address to start loading a formatted data file. For downloads, the I/O Offset is subtracted from the beginning address in the formatted data file. The result is then added to the memory begin address to determine where the block of data is loaded.

I/O Timeout The amount of time that AutoSite will wait for a data transfer to begin.

I/O Translation Format See Data Translation Format.

Illegal Bit An illegal bit is 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.

Instrument Control Code A 1-digit number that signals or controls data transfers. It also implements a form of remote control that provides peripherals with flow control beyond that provided by software handshaking.

JEDEC Joint Electron Design Engineering Committee: a committee of programmer and semiconductor manufacturers that provides common standards for programmable issues. Examples of these 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 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.

LCA An acronym for Logic Cell Array.

LCC An acronym for Leadless Chip Carrier—a type of device package. A 4-sided device package with pads on the underside. Typically, the LCC is used in surface mount applications.

LED An acronym for Light Emitting diode. AutoSite has five LEDs: four on the front panel and one on the disk drive.

Load Data A device operation that moves device data into AutoSite. You can load AutoSite with data from a device, from AutoSite's internal disk drive, or from a serial port (for example, from the Handler port).

Logic Verification After programming a device, you can select test vector verification, fuse verification, or both types of verification.

Master Device A device that contains data you wish to program into another device. For example, you would load data from a master device and then program that data into a blank device.

MatchBook A new type of socketing technology that makes handling surface-mount devices easier. MatchBook device carriers are used with AutoSite's stand alone kit for single device programming.

Memory Begin Address The first address, in hex, of the first byte of data to be used in device operations. If the data source / destination is RAM, the memory begin address is a RAM address. If the data source / destination is disk, the memory begin address is the offset for a disk file.

Next Device Used during serial set programming, this value specifies the next device in the set. For example, if you are using 8-bit devices and have specified a word width of 16 bits, it will require two devices 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. AutoSite 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.

Output Record Size The number of data bytes contained in each data record during upload.

Overblow A condition in which fuses are blown that should not have been.

Overblown Fuse A fuse that has been over-programmed such that the surrounding area may have been damaged or such that 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.

Parallel Test Vector Application — Use of internal registers to hold and release a full set of test vectors (e.g., 20 for a 10-input 10-output device) at once. In contrast to serial application, parallel does not require accommodations for clocking contention, and parallel better matches in-circuit PLD operation and board test suites.

Part Number — The number on the device. For example, if you are using an Intel 27C256, then the part number of the device is 27C256.

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 device package with J-shaped leads extending from four sides downward, used for surface mount applications.

PLD — An acronym for Programmable Logic Device. A particular type of programmable integrated circuit. Architectures range from being very simple to very complex. Most PLDs contain two levels of logic, an AND array followed by an OR array.

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.

Program — The controlled application of electrical pulses to program specific fuses or cells in a device.

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 and ICs that are user configurable. PLDFs and PGAs are examples of programmable integrated circuits.

Programming Module — The interface between AutoSite and the device. The programming module routes the signals from AutoSite's pin driver head to the pins on the device.

Reboot The process of re-initializing the programmer. After rebooting, the programmer is in the same state as 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 you select.

Remote Control Mode AutoSite is controlled from a host running a driver program, such as a PC running TaskLink. Device data files can be stored on AutoSite's disk and on the host.

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 Device A procedure that tells AutoSite what device you will be using. You can select a device in one of two ways: by entering the family / pinout code, or by selecting the manufacturer and the device part number.

Self-test A built-in self-diagnosis command that allows you to test various circuits and subsystems in AutoSite, 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.

Serial Test Vector Application The process of applying test vectors in a serial fashion, one input at a time.

Serial Vector Test A device test that applies test vector input states serially, starting with pin one and stepping through the remaining pins. This test is a diagnostic tool designed to help debug and classify test vector failure. Specifically, this test is designed to isolate test vectors that are sequence dependent.

Set Programming A type of programming in which a large data file is partitioned and programmed into multiple memory devices.

SmartPort A feature of AutoSite that automatically detects and adjusts AutoSite to the presence of DCE / DTE protocol.

Source The place from which something comes. The "something" the source is sending is almost always data. The source can be RAM, a disk file, or one of AutoSite's serial ports.

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. A test vector is

	one such string, e.g., 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, e.g., that a 10-bit counter is counting to 10. Design vectors are used in both preprogramming simulation and 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, e.g., 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 you 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, you can calculate the individual sumcheck of the device and the sumcheck of the entire set.
Terminal Emulator	A program to enable a PC or other computer to act as an ASCII terminal. Allows a PC to be used to communicate with a programmer in terminal mode or with a mainframe.
Test Vector	Test vectors functionally test the device, using structured test vectors stored in memory or in a disk file.
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/LCC Vectors	See JEDEC I/O translate DIP/LCC 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.
Transmit Pacing	The number of milliseconds AutoSite will insert as a time-delay between characters transmitted to the host computer during uploading. The time delay is specified in tenths of milliseconds.
Underblow	A condition in which fuses that should have been blown or programmed were not.

Underblown Fuse — A fuse that did not disconnect as per manufacturer's specification. These fuses may test properly, but tend to be more prone to grow back when in circuit, rendering the PLD useless.

Universal PLD Programmer — A programmer that can apply power, ground, and any programming pulse required to program any fuse technology device.

Universal Pin Driver — A pin driver with the ability to supply power and ground to every pin. With Universal Pin Drivers, you can program and test devices without having to use pin out adapters and characterizers.

Upload Data — A file transfer operation that involves sending data from the programmer to a host.

Upload Host Command — A command that is sent from AutoSite to the host during an upload. The command tells the host what to do with the incoming data.

Upload Wait — The length of time AutoSite will wait before it begins sending data to the host computer after the host upload command is sent.

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. 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 RAM — The RAM in AutoSite. 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 disk 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 5.5V and the second pass at 4.5V.

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.

Wildcard Used when entering filenames, a wild card represents one or more characters in a filename. For example, **27*.dat** can represent both **27512.dat** and **27128.dat**.

Yield The percentage of successfully programmed devices.

Yield Tally The yield tally function keeps track of the programming statistics for the last 16 types of devices programmed. The following statistics are kept for each device type: the manufacturer name and part number, the family / pinout code, the number of devices attempted, the number of devices that programmed successfully, the number of devices that failed non-blank test or illegal bit check, the number of devices that failed to verify, the number of devices that could not be programmed because they contained bits that required more programming pulses than were specified, and, for logic devices only, the number of devices that failed structured vector test.

ZIF Socket An acronym for Zero Insertion Force. A socket into which the device can be dropped and engaged via a lever.

Index

A

B

C

D

E

F

G

H

I

K

L

M

O

P

R

S

T

U

V

W