

29B Universal Programming System



Operator's Manual

DATA I/O

Getting Started

Data I/O has made every attempt to ensure that the information in this document is accurate and complete. However, Data I/O assumes no liability for errors or for any damages that result from use of this document or the equipment which it accompanies.

Data I/O reserves the right to make changes to this document without notice of any time.

ORDERING INFORMATION

When ordering this manual use Part Number 984-0200-001.
Applies to: 298 Engineering Part Number 990-0015-020 and up.

Data I/O® is a registered trademark of Data I/O Corporation.
LogicBak™, GangBox™, MCSBak™ and Unix 28™ are all trademarks of Data I/O Corporation.


Copyright 1985, Data I/O Corporation. All rights reserved.

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.

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.

Symbols

 : This symbol appears on the equipment and it indicates that the user should consult the manual for further detail.

V ~ : This symbol stands for Vac. For example 120V ~ = 120 Vac.

Power Source

Check the voltage selector indicator (located on the rear panel) to verify that the product is configured for appropriate line voltage.

Grounding the Product

The product is grounded through the grounding conductor of the power cord. To avoid electric shock, plug the power cord into a properly wired and grounded receptacle only. Grounding this equipment is essential for its safe operation.

Power Cord

Use only the power cord specified for your equipment.

Fuse Replacement

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

Serviceing

To reduce the risk of electric shock, do not perform any serviceing other than that described in this manual.

Table of Contents

INTRODUCTION	1
Optional Features	1
Ordering	2
System Overview	2
298 Front Panel Operations	3
Specifications	6
Functional Specifications	8
Power Requirements	8
Physical and Environmental	9
Warranty and Customer Support	9
10	10
GETTING STARTED	11
Power Connection	11
Verifying/Changing the Operating Voltage	11
Verifying/Replacing the Line Fuse	11
Spare Line Fuse Tray	13
Grounding the Unit	14
Pak Installation	14
Powering Up	15
Sample Programming Section	16
17	17

PROGRAMMING	19
Overview	19
General Operational Notes	20
The Action Symbol	20
Aborting an Operation	20
Programming By-16 (16-bit) Devices	20
Setting the Beginning Address for an Operation	24
Setting the Block Size	22
Setting the Offset Address	22
Keying in Family and Pinout Codes	23
Sumcheck Display	23
Programming	24
Load RAM with Master Device Data	25
Load RAM from Serial Port	27
Program Device with RAM Data	29
Block Move	31
Output to Serial Port	32
Edit Operations	33
Editing Using a Hexadecimal Base	34
Verify Operations	35
Verify RAM Data Against Serial Port Data	36
Verify RAM Data Against Device Data	37

REMOTE CONTROL	39
System Setup	40
Transferring to and From Remote Control Mode	40
Setting Parameters	40
Serial Port Hookup	43
Computer Remote Control	47
Overview	47
Response Characters	48
Description of CRC Commands	50
CRC Interactive Commands	56
Error Status Word	58
System Remote Control	60
Command Protocol	60
Load RAM with Master Device Data	62
Program a Device with RAM Data	64
Load RAM From the Serial Port	66
Output RAM Data to the Serial Port	68
Block Move	70
Verify RAM Data Against Device Data	71
Verify RAM Data with Serial Port Data	73
Editing in SRC	75
Select Functions in SRC	76
Data Translation Formats	78
Introduction	78
Transition Formats	81
298/Handler Interface	108
Introduction	108
Compatibility	108
298 Handler Port Information	109

Table of Contents

SELECT FUNCTIONS	111
Introduction	111
Accessing Select Functions	113
Format Codes	114
RAM Data Manipulation	115
A2 Fill RAM	116
A5 Split RAM	117
A6 Shuffle RAM	118
Utility and Inquiry Commands	119
B2 System Configuration	120
F3 Data Lock On	121
Serial I/O Commands	122
B8 Format Number	123
D8 Record Size	126
D9 Null Count	127
FA Character Output	128
FC Remote On/Off	129
ERROR CODES	131
INDEX	137

Data I/O's 298 Universal Programmer reliably programs, tests and verifies most commercially available programmable memory (PROM) and logic devices. Standard features of the 298 include a 128K x 8 data RAM and RS-232C serial port and a 16-character alphanumeric display. The 298 also offers 29 systems, such as Intel, Texas Instruments, Motorola, Hewlett-Packard and Tektronix. In addition, the standard system includes two remote control protocols, Computer Remote Control (CRC) and System Remote Control (SRC).

This manual contains the operational procedures for the Model 298 Universal Programmer. Included in this manual are instructions on:

- **GETTING STARTED**—A sample session. Includes instructions on voltage selection and checking the back panel fuses, powering up the programmer, inserting a device, and programming the socketed device.
- **PROGRAMMING**—Information about programming devices, including a list of general programming notes. Also includes information on editing, verifying data integrity, and serial port data transfer.
- **REMOTE CONTROL**—Describes Serial I/O operation with the 298 programmer, includes CRC and SRC operation, information on RS-232 hookup, descriptions of data translation formats, and handler operation.
- **SELECT FUNCTIONS**—Details on Select Codes, two-digit hexadecimal codes that enable special programmer functions such as RAM manipulation and serial port special functions.
- **ERROR CODES**—Describes the 298's error code displays, and corrective action.
- **INDEX**—An alphabetical guide to all the major topics covered in the manual.

NOTE

The error codes provided in this manual are not accompanied by any service information. If you would like to receive maintenance data (circuit descriptions, schematics, calibration information and waveform photographs), please contact your nearest Data I/O sales representative. A list of representatives is included with the warranty information of the back of this manual.

Optional Features

The 298 also offers the following optional features. If you wish to purchase any of these features, contact your nearest Data I/O sales representative. A list of representatives is included at the back of this manual.

- HANDLER INTERFACE**—Allows you to connect the 298 to a device handler. The user or handler manufacturer supplies both the Pak to handler interface and port cable. Data I/O supplies the control port and compatible firmware for 298/handler operation.

When using the Exatron 2500 handler, the optional handler interface is not necessary. Exatron provides the interface necessary for operation with the 298.

- SERIAL PAPER TAPE READER**—Allows transfer of paper tape data to the 298.

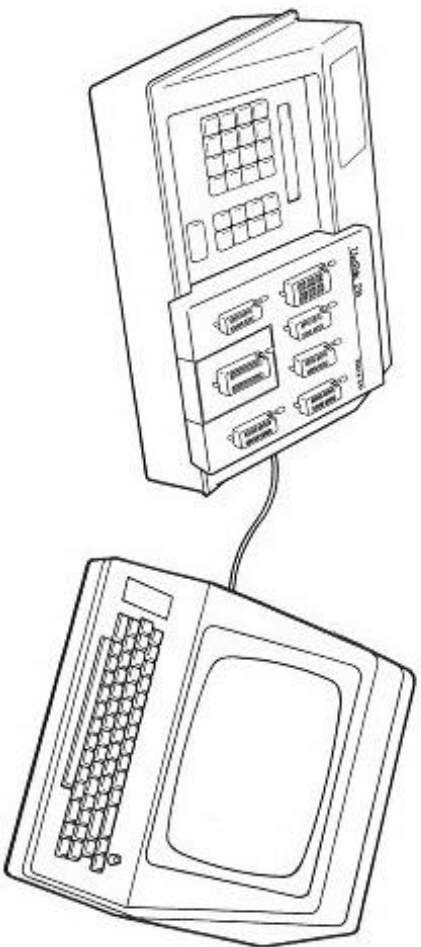
Ordering

Orders made with Data I/O must contain the following information:

- Description of the equipment
- Quantity of each item ordered
- Shipping and billing address of firm, including ZIP code
- Name of person ordering equipment
- Purchase order number
- Desired method of shipment

System Overview

The 298 provides a universal means of programming, testing, and verifying a variety of memory and logic devices. You can tailor the 298 to your programming needs by selecting the appropriate programming Pak and plugging it into the 298. The various programming components from which you can choose to complete your programming system are listed and shown in the accompanying figures.



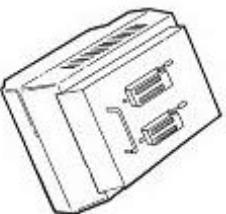
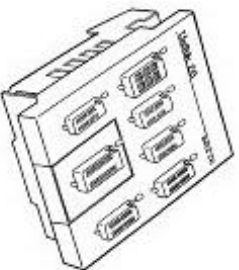
Remote Control*

The standard Computer Remote Control (CRC) allows you to save programs on disk or tape which may be downloaded to the 298. System Remote Control (SRC) allows you to send commands to the 298 from a terminal.

*Terminal not included.

System Overview (continued)

- Unipak 28—programs more than 800 devices, including MOS and CMOS EPROMs and EEPROMs, fuse link, AIM and DEAP bipolar PROMs. Programming algorithms are software selectable and no additional personally modules are required. Simple pinout cartridges are available for 40-pin microcomputers and parts with non-standard pinouts and unique package types (LCC, PLCC).
- LogicPak—combined with appropriate plug-in adapters, allows you to design, program and functionally test more than 160 different logic devices. The high-level software translates your logic design from truth tables or Boolean equations into the correct fuse map.
- MOSPak—programs more than 145 MOS EPROMs and EEPROMs, requiring no additional hardware.

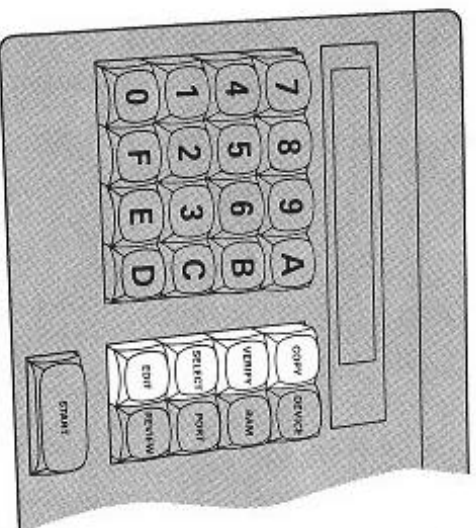


- GangPak—gives you the capability of programming, in a single operation, identical sets of MOS EPROMs or EEPROMs. Set programming allows you to partition a program into one or more sets of PROMs. The GangPak can also be used for conventional gang programming of up to eight devices at a time.
- Programming Modules—support device family applications not covered by the standard programming Paks.



29B Front Panel Operations

With the 29B, you can perform four basic operations: COPY, VERIFY, SELECT and EDIT. Each of these functions has its own key (see figure) on the programmer's front panel. In addition to these four, the programmer has three source/destination keys, a REVIEW and START key, and a hexadecimal keyboard for editing data and selecting special functions and parameters.



MODE (OPERATION) KEYS

COPY

Used to move a block of data to or from a serial port, RAM, or device. Works in conjunction with source/destination keys.

VERIFY

Used to make a byte-by-byte comparison of a block of data. Used with source/destination keys.

SELECT

Prepares the programmer to accept codes for select functions. See the select functions section for details.

EDIT

Allows viewing and changing of data at individually selected RAM address locations. See the programming section for details.

Specifications

The 298's specifications are listed below.

Functional Specifications

Functional specifications for the 298 are as follows:

- General Architecture: Microprocessor controlled (6808)
- Data RAM: 128K x 8
- Programming Support: GangPak, LogicPak, UniPak 28, MOSPak and programming modules
- Keyboard: 16-key hexadecimal and 9-key functional
- Display: 16-character alphanumeric
- Input/Output: Serial RS-232C and 20 mA current loop
- Baud Rates: 50, 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600, 19,200
- Remote Control: Computer Remote Control (CRC)
System Remote Control (SRC)
- Translation Formats: 29 available. See table below.
- Handler Capability: Optional handler port is available for binning and control signals.

Available Translation Formats for the 298 Universal Programmer

ASCII-810F	ASCII-Octal SMA	Intel MCS-86 Hexadecimal Object
ASCII-81LF	ASCII-Octal (Space)	MOS Technology
ASCII-BNPF	Binary	Motorola Exorciser
ASCII-Hex (Apostrophe)	BNPF (5-Level)	Motorola Exormax
ASCII-Hex (Comma)	Data I/O Data Control Unit (DCU)	RCA Cosmac
ASCII-Hex (Percent)	DEC Binary	Signetics Absolute Object
ASCII-Hex SMS	Fairchild Fairbug	Spectrum
ASCII-Hex (Space)	Hewlett-Packard 64000	Tektronix Hexadecimal
ASCII-Octal (Apostrophe)	Absolute	Extended Tektronix Hexadecimal
ASCII-Octal (Percent)	Intel Intellec 8MDS	Texas Instruments SDSMAC

Power Requirements

Power requirements for the 298 are as follows:

- Operating Voltages: 100, 120, 220 or 240 Vdc + 5% or -10%
- Frequency Range: 48-63 Hz
- Power Consumption: 415W/175 VA
- Fuse Protection: Primary and secondary fuse protection

Physical and Environmental

Physical and Environmental Requirements for the 298 are as follows:

- Dimensions: 31.1 x 15.2 x 27.3 cm (15 x 6 x 10.8 in.)
- Weight: 6.4 kg (14.1 lb)
- Operating Temperature: +5° to 45°C (41° to 113°F)
- Storage Temperature: -40° to 70°C (-40° to 158°F)
- Humidity: to 95%
- Operational Altitude: to 10,000 ft.

Warranty and Customer Support

Data I/O equipment is warranted against defects in materials and workmanship. The warranty period of one year, unless specified otherwise, begins when you receive the equipment. Refer to the warranty card inside the back cover of this manual for information on the length and conditions of the warranty. For warranty service, contact your nearest Data I/O customer support center.

Data I/O maintains customer support centers throughout the world, each staffed with factory-trained technicians to provide prompt, quality service. This includes not only repairs, but also calibration of all Data I/O products. A list of all Data I/O customer support centers is located in the back of this manual.

Getting Started

This section explains how to get started using your 298 programmer. Included here are complete procedures for powering up and for programming a device from the programmer's keyboard. For details on operating the programmer using a terminal or computer, see the remote control section.

This section includes the following information:

- Verifying/Checking the Operating Voltage and Line Fuse
- Installing a Programming Module/Pak
- Powering up the Programmer
- Sample Programming Session

Power Connection

Before applying power to your programmer, make sure that the operating voltage is correct, that the line fuse is intact, and that the unit is properly grounded. When you have checked that the above are in order, proceed to the next subsection, Pak Installation.

Verifying/Changing the Operating Voltage

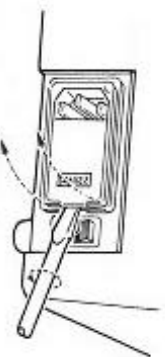
The factory has selected the proper voltage according to your specification. A voltage reading is visible through a window in the door that covers the voltage selector wheel, located on the back panel, as shown in the figure. This voltage should be the same as the line voltage on which the machine will operate. If the voltage that appears in the window is incorrect, change the operating voltage according to the following procedure.

CAUTION

This instrument may be damaged if operated with the wrong line voltage.

The procedures to verify and/or change the operating voltage are described here and illustrated in the figure.

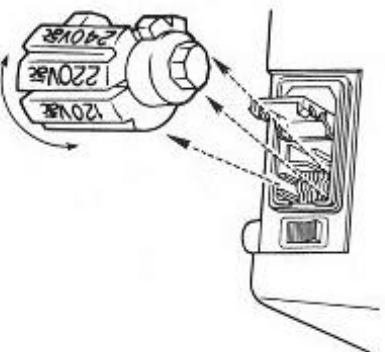
1. Disconnect the power cord.
2. Gently pry open the door that covers the voltage wheel selector with a flat-blade screwdriver.



3. Pull the voltage wheel selector out of its slot.
4. Rotate the selector until the correct operating voltage points toward you.
5. Insert the selector back into its slot.

NOTE

If you wish to access the line fuse at this point, proceed to step 2 in the next procedure.



6. Snap the door closed.
7. The correct voltage reading will now appear in the window.



VOLTAGE READING

Verifying/Replacing the Line Fuse

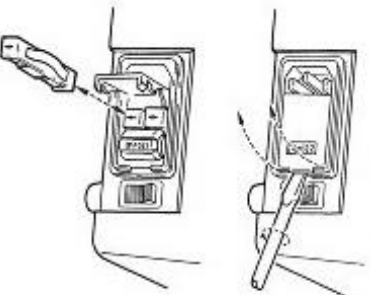
The line fuse is located behind the same door that covers the voltage wheel selector. Perform the following procedure to verify that the line fuse is correct and intact. In the event that the fuse is blown, replace it with one of the correct size. Procedure steps are illustrated in the figure.

1. Gently pry open the door that covers the fuse holder using a flat-blade screwdriver.

NOTE

There are two fuse receptacles, only the one on the bottom is connected to the programmer's circuitry. The top receptacle is a spare fuse tray. See the next subsection for more information on this spare tray.

2. Pull the bottom fuse holder out of its slot.
3. Check to determine whether the fuse is intact. If it is intact, proceed to step 4. If it is blown, install a new fuse. See table for line fuse ratings.



CAUTION

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

Operating Voltage	Current	Line Fuse Rating Voltage	Type	Data I/O Part Number
100	2.5A	250V	Slow-blow*	416-4240
120	2.5A	250V	Slow-blow*	416-4240
220	1.0A	250V	Slow-blow**	416-4571
240	1.0A	250V	Slow-blow**	416-4571

* Littelfuse type "313," Busman type "MDA,"

** Littelfuse type "218," Busman type "SDC," Schurter type "00-2504."

4. Insert the fuse holder into its slot so that the arrow on the fuse holder points in the same direction as the arrows on the door.
5. Snap the door closed.



Spare Line Fuse Tray

All 298 programmers are equipped with two line fuse trays (see the previous figure). The white fuse tray accepts $1/4 \times 1 \ 1/4$ inch fuses; the black tray accepts 5×20 millimeter fuses, commonly available in Europe. Only the bottom fuse receptacle is connected to the programmer's circuitry.

Grounding the Unit

The 298 is shipped with a three-wire power cable. This cable connects the chassis of the unit to the earth ground when the cable is connected to a three-wire (grounded) receptacle.

WARNING

Continuity of the grounding circuit is vital for the safe operation of the unit. Never operate this equipment with the grounding conductor disconnected.

Pak Installation

Any of the Data I/O programming Paks may be installed and removed with the programmer's power on; this feature allows you to retain data in the 298 RAM when changing programming Paks. If the programmer power is turned on before a Pak is installed, you will hear a "beep" until the Pak is properly installed.

CAUTION

Voltage transients can cause device damage. Be sure that all sockets are empty when switching power on or off or installing or removing the Pak.

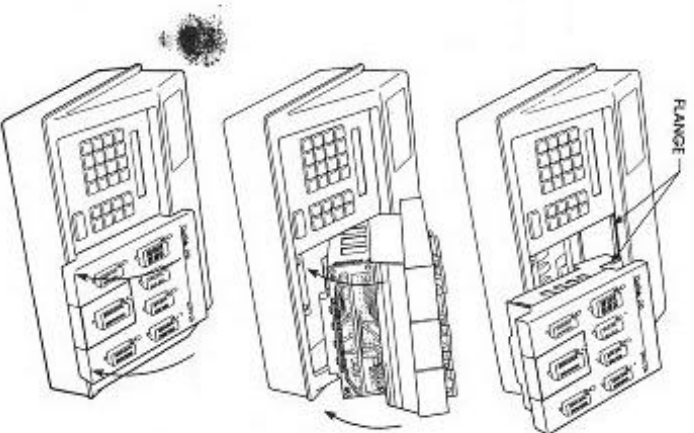
To install a Pak into the 298, refer to the figure and follow this installation procedure.

1. Slide the Pak into the opening in the programmer.
2. Tilt the Pak up and gently push it back to hook its flange over the top back edge of the programmer opening.
3. Lower the Pak into position as shown in the figure.

CAUTION

Be careful when inserting the Pak. If the connector at the bottom of the Pak (see figure) has bent contact pin(s), forcing the Pak could break the pin(s) or damage the connector.

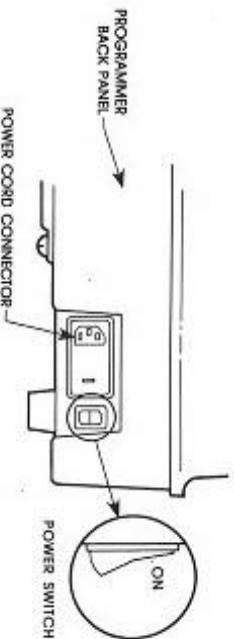
4. Press down gently on the front edge of the Pak to ensure a good connection.



Powering Up

The first step in getting started is powering up your 298 programmer. Use the following procedure.

1. Check to make sure the Pak's sockets are empty. If a device is in a socket, remove it.
2. Check to be sure the voltage selector is in the proper position. Plug the AC power cord into the rear of the programmer and into a power receptacle.
3. Press the power switch at the back of the programmer to the "ON" position (see figure).



When the programmer is powered up, it automatically performs the self-test routine, which initializes the programmer's hardware and checks the scratch RAM, firmware and data RAM. While the self-test is being performed, the programmer will display

SELF-TEST

The symbol at the right-most digit of the display is the "action symbol," which "notifies" when the programmer is performing an operation.

When the self-test has been successfully completed, the programmer will display

SYSTEM 298 V0N

"N" in the above display represents the 298's firmware version. For example, "V02" would denote version two firmware. This number is useful when contacting Data I/O Support Personnel.

If an error message is displayed, check the error codes section of this manual.

Sample Programming Section

The following steps describe how you would program a 2764 part using a master device (a part that has been previously programmed and is used as a "master" to program other parts). This procedure assumes that the programming data has already been created (using a development system) and transferred to the master device. For more details on device programming, see the programming section of this manual.

1. Plug power cord into the programmer and into a power receptacle.
2. Make sure all the device sockets are empty.
3. Power-up the programmer.

4. Press

to prepare the programmer to transfer the master device data to the programmer's data RAM. The programmer will display

FAM ^ 00 PIN 00

5. Press

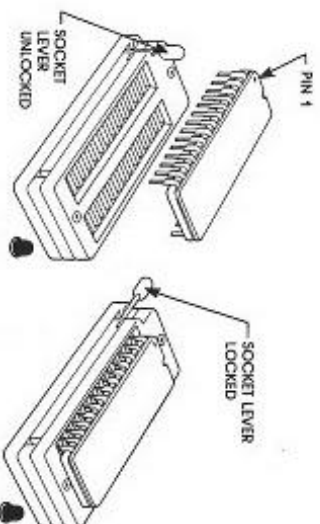
the family/pinout code for the 2764 part. The programmer will then display

FAM 79 PIN ^ 33

6. Lift up the lever on the socket that has an illuminated LED below it (see figure). Line up pin 1 of the device so that it is nearest the lever and set the device into the socket. Press down on the lever to lock the device in place.

Note

Orient LCC devices according to the arrow pointing to the left of the LCC socket.



7. Press . The programmer will display
- LOADING DEVICE**
- LOAD DONE XXXX**

NOTE

"XXXX" is the sumcheck of the device. See step 11 for more information.

8. Lift up the socket lever and remove the master device from the socket. The master device data is now transferred to RAM. The next part of the procedure transfers that data to the blank device.
9. Press to prepare the programmer to transfer the data to the blank device. The programmer will display

FAM 79 PIN **33**

10. Line up pin 1 of the blank device so that it is nearest the lever and set the device into the socket. Press down on the lever to lock the device in place.

11. Press . The programmer will display

TEST DEVICE

PROGRAM DEVICE

VERIFY DEVICE

PRG DONE 01 XXXX

NOTE

"XXXX" in the above display represents the device's sumcheck; the hexadecimal sum of all the bytes in the device. The number displayed should match the sumcheck displayed during step 7 of this procedure. "PRG DONE 01" means that 1 device has been programmed.

12. Lift up the socket lever and remove the device from the socket. The device is now programmed.
13. To program another device, simply place it in the socket and press START.

Programming



Programming

This section describes how to program, edit and verify data using the 298 programmer. Included in this part of the manual is the following information:

- **GENERAL OPERATIONAL NOTES**—Explains common symbols and messages encountered during programming operations. Read this subsection first, to familiarize yourself with the displays.
- **PROGRAMMING**—Describes Copy operations, which transfer programming data from a source to the 298's RAM and then to either the blank device to be programmed or the serial port.
- **EDIT OPERATIONS**—Explains use of the editing function, which lets you access and change programming data residing in the 298's RAM, before that data is programmed into a device. You may edit RAM data in three number bases: hexadecimal, octal or binary.
- **VERIFY OPERATIONS**—Describes how to verify data in two locations; for example, between a programmed device and the data in RAM. A Verify checks that the information in both locations is the same, thereby verifying the integrity of the data transfer.

Overview

Prior to programming a device, you must first load (also referred to as copy or transfer) the programmer's RAM with data to be programmed into a particular device. This data may be transferred to RAM from either the serial port, a "master" (previously programmed) device or may be keyed in by hand. You can then transfer that data to the socketed device.

After you transfer the data to the programmer's RAM, you may make any needed corrections by using the EDIT function. You may edit RAM data in any of three number bases: octal, hexadecimal or binary. After data has been transferred, you may check that the data was transferred correctly by using the Verify operation. A Verify compares the data in the two locations to make sure they match. Besides the Program, Edit and Verify operations described above, this section includes some general operating notes, which follow.

General Operational Notes

The following displays and notes are common to nearly all the Program/Verify procedures for the 298.

The Action Symbol

A special character is displayed by Data I/O programmers while an operation is being executed. This character is called an action symbol, and appears in the right-most character position of the display. For example, during power up, the programmer automatically performs a self-test routine. While this test is being executed, the programmer will display

SELF TEST 

The "hand" of the action symbol rotates several times while an operation is taking place, to indicate that the programmer is executing the operation.

Aborting an Operation

Most operations (except for a few special select functions) may be aborted by pressing one of the four mode keys (COPY, VERIFY, SELECT or EDIT). If an operation is in progress when one of these keys is pressed, the programmer will momentarily display an abort message. The programmer will then assume and display the mode selected. For example, if you abort an operation by pressing the SELECT key, the programmer will display

FUNCTION ABORT

and then

SELECT CODE 

Programming By-16 (16 bit) Devices

When the programmer is doing operations using normal 4- or 8-bit devices, the 298's RAM is configured to 128K x 8. For by-16 devices, the programmer's RAM is reconfigured to 64K x 16. Because word rather than byte addresses are used in this mode, only addresses 00000 thru 0FFFF may be accessed during a Program, Verify or Edit operation.

Setting the Beginning Address for an Operation

Although setting the beginning address is an optional procedure, you may use it to make transfers when an operation requires use of only a partial amount of a device, RAM or a serial downloaded file. When you have selected the source or destination of the data that you wish to transfer (by pressing the COPY, RAM or PORT key), the display will always prompt you for the specific address to begin copying from or to. For example, if you were copying master device data to the programmer's RAM, you would press



to instruct the programmer to copy data from the master device. The programmer would then display

DEV \ ADDR/SIZE

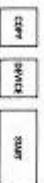
The prompt (\) preceding "ADDR" in the display means that you may change the beginning device address to any address within the range of the device's word limit. To change the beginning address, simply key in the hexadecimal address that you want to start copying from. If the hex value that appears in place of "ADDR" is correct, continue to the next step in the procedure. The default address is 00000.

NOTE

If you set the beginning address, you must also key in The block size (see next paragraph).

Setting the Block Size

You may set the size of the block of data you want to move in the same manner that you could change the beginning address. After you have selected the source of the data you wish to transfer and have pressed the START key, the display will always prompt you for the size of the data block that you wish to transfer. For example, if you were copying master device data to the programmer's RAM, you would press



The programmer would display

DEV ADDR ^ SIZE

The prompt (^) preceding "SIZE" in the above display means that you may change the size of the block of data you wish to transfer. To change the block size, simply key in the hexadecimal number of bytes that you want to transfer. If the hex value that appears in place of "SIZE" is correct, continue to the next step in the procedure. Default value for block size is the device size for device-related operations. For I/O transfers, default is 64K for 16-bit address formats; all of RAM for 17 bit or more address formats.

NOTE

You may specify the block size without setting the address.

Setting the Offset Address

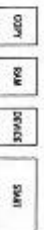
When transferring data through the serial port, you may specify the offset address if you only want to copy a certain block of the data. For example, if you set the offset at 001100, port address 001100 would be copied to address 00000 in RAM.

Keying In Family and Pinout Codes

NOTE

Some programming Paks skip the family/pinout display, and automatically select it for you. Check your Pak manual.

Each device that Data I/O equipment supports is represented by a hexadecimal two-digit family code and a two-digit pinout code. These codes identify the device to the programmer, to ensure that the proper programming pulses are applied to the part. When you are performing a Program or Verify operation, the display will always prompt you to enter this code. For example, if you were programming RAM data into a blank device, you would press



The programmer would then display

FAM \ 00 PIN 00

You would then key in the two-digit family code and two-digit pinout code for the device being programmed. Family and pinout codes are listed in the device table included with your Pak manual or on the Data I/O wall chart.

CAUTION

Be sure you enter only those family/pinout codes listed in the device table. Invalid codes may cause unpredictable results at the device socket, which may damage the device. Data I/O assumes no responsibility or liability for results produced by entry of illegal family/pinout code combinations.

Sumcheck Display

After the 298 has performed a Copy or Verify operation, a four-digit number is always displayed in the right-most display positions. This hexadecimal number, called a sumcheck, is used to verify the integrity of a data transfer. For example, if you copy RAM data to the serial port, at the end of the operation the programmer will display

OUTPUT DONE HHHH

Programming

The following pages explain how to transfer programming data to RAM and transfer that RAM data to the device to be programmed. A list of the programming operations is given below. The procedures described here are for front panel operation. For programming operations using remote control, see the remote control section of this manual.

Operation	Description
Load RAM With Master Device* Data	Used to transfer data from a master device to the programmer's RAM.
Load RAM From Serial Port	Transfers programming data from a remote system to the programmer's RAM.
Program Device With RAM Data	Copies the programming data from RAM to the device installed in the programming module socket.
Block Move	This operation moves a block of RAM data to another location in RAM.
Output RAM Data to Serial Port	Transfers the RAM data to a remote system via the serial port.

* A master device is a previously-programmed device whose data is used as a "sourcer" to program blank devices.

Load RAM With Master Device Data

This operation transfers programming data from a master device to the programmer RAM. When the data transfer is complete, the 298 calculates and displays the sumcheck (see step 8). Use the following procedure to load the 298 RAM with data from a master device using the programmer front panel.

Procedure	Keystroke	298 Displays
1. Select the copy operation.	<input type="button" value="COPY"/>	COPY DATA FROM
2. Select the source of the data to be transferred.	<input type="button" value="DEVICE"/>	DEV ^ ADDR / SIZE
3. Accept or (optionally) change the device address to begin copying from (default is zero) and/or the size of the block (in hex) to be transferred.	<input type="button" value="START"/> Or <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="START"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="X"/>	DEV ADDR ^ SIZE DEV XXXXXX/XXXXXX
NOTE To specify only block size, press START and then key in the size		
4. Select the destination for the data.	<input type="button" value="RAM"/>	CO DEV > RAM ^ ADDR
5. Accept or (optionally) change the begin RAM address.	<input type="button" value="START"/> Or <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="START"/>	FAM ^ 00 PIN 00 CO DEV > RAM ^ XXXXXX FAM ^ 00 PIN 00

Load RAM With Master Device Data (continued)

Procedure

Keystroke

298 Displays

- Enter the four hex-digit family/pinout code combination for the device to be copied. Family/pinout codes are listed in the device list included with your programming Pak.

x	x	x	x
---	---	---	---

FAM XX PIN ^ XX

NOTE

If the installed programming Pak does not require a family/pinout code, the programmer automatically skips this step.

- Insert and lock the master device into the socket that has an illuminated LED below it.



LOADING DEVICE

LOAD DONE HHHH

NOTE

"HHHH" denotes the device sumcheck, a four hexa-decimal digit summation of the data loaded. This

number should match the number displayed when you transferred this data to the device originally.

- Remove the master device from the socket. To repeat the load operation from another device with the same family and pinout codes, return to step 7.

Load RAM From Serial Port*

To transfer data received at the serial port to the programmer RAM, use this operation. When transfer is completed, the programmer calculates and displays the sumcheck of the transferred data and will signal an error if it does not match the one received with the data.

After setting up the serial port and selecting the appropriate data translation format (see remote control section), use the following procedure to load the 298 RAM with incoming serial port data.

Procedure	Keystroke	298 Displays							
1. Select the copy operation.	<input type="button" value="COPY"/>	COPY DATA FROM							
2. Select the source of the data to be transferred.	<input type="button" value="SERIAL"/>	POR ^ ADDR / SIZE							
3. Accept or (optionally) change begin address offset and/or block size. The block size should not exceed 10000H. Default is the first incoming address.	<input type="button" value="START"/>	POR ADDR ^ SIZE							
	OR <table border="1"> <tr> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> </tr> </table>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>
<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>		
<p><i>NOTE</i> To specify only the block size, just press START and key in the size. If a 64K-addressable translator has been selected, the block size should not exceed 10000H.</p>									
4. Select the destination for the data.	<input type="button" value="RAM"/>	CO POR > RAM ^ ADDR							

*The displays shown here may differ slightly if you are using a translator that can address over 64K bytes of data. For example, Extended Elektronik Hexadecimal or HP64000.

Load RAM From Serial Port (continued)

Procedure

Keystroke

298 Displays

5. Accept or (optionally) change the begin RAM address. The host system is now set up to download data to the 298. Data must be sent within 25 seconds or an error message will appear. To disable the 25 second timeout, use select function F9 (see select function section).

OR

INPUT PORT

E7

INPUT DONE HHHH

sumcheck

6. To repeat the load operation, press

Program Device With RAM Data

Before programming a device, the system automatically performs illegal bit tests and blank checks at nominal VCC, to verify the ability of the device to accept programming data. Data is then programmed into the device in the socket one byte at a time. This continues until all data bytes have been programmed into the device. After programming is completed, the data in the device is automatically compared (verified) with the RAM data to ensure correct programming.

Procedure	Keystroke	298 Displays
1. Select the copy operation.	<input type="text" value="COPY"/>	COPY DATA FROM
2. Select the source of the data to be transferred.	<input type="text" value="RAM"/>	RAM ^ ADDR ^ SIZE
3. Accept or (optionally) change the hex begin RAM address and/or block size. Defaults are zero and the word limit of the device.	<input type="text" value="START"/> <input type="text" value="OR"/> <input type="text" value="START"/>	RAM ADDR ^ SIZE RAM XXXXXX/XXXXXX
<p><i>NOTE</i> To specify only the block size, just press START and key in the size.</p>		
4. Select the destination for the data.	<input type="text" value="DEVICE"/>	CO RAM > DEV ^ ADDR
5. Accept or (optionally) change the hex begin device address.	<input type="text" value="START"/> <input type="text" value="OR"/> <input type="text" value="START"/>	FAM ^ 00 PIN 00

Program Device With RAM Data (continued)

Procedure

Keystroke

298 Displays

6. Enter the family/pinout code for the device.

X	X	X	X
---	---	---	---

FAM XX PIN ^ XX

7. Insert and lock the blank device into the socket with the illuminated LED below it. Execute the operation.

start

TEST DEVICE
 PROGRAM DEVICE
 VERIFY DEVICE

8. When the operation is complete, this display will appear. The surcheck should match the one displayed after the load to RAM operation. If an error code is displayed, check the error codes list.

PRG DONE 01 HHHH
 number of surcheck
 correctly-
 programmed
 devices

9. Remove the programmed device from the socket. To program additional identical devices using the data stored in RAM, return to step 7.

Block Move

A block move copies data in one block of RAM locations to another block of RAM locations, beginning of a defined address. Use the following front panel keyboard procedure to copy a block of data from one location in RAM to another location in RAM:

Procedure	Keystroke	298 Displays
1. Select the copy operation.	<input type="button" value="COPY"/>	COPY DATA FROM
2. Select the source of the data to be transferred.	<input type="button" value="RAM"/>	RAM ^ ADDR / SIZE
3. Accept or (optionally) change the hex begin address and block size. Default values are zero.	<input type="button" value="START"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="START"/>	RAM ADDR ^ SIZE
NOTE To specify only the block size, just press START and key in the size.	<input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="X"/>	RAM XXXXXX / XXXXXX
4. Select the destination for the data to be transferred.	<input type="button" value="RAM"/>	CO RAM ^ RAM ^ ADDR
5. Accept or (optionally) change the hex destination address.	<input type="button" value="START"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="X"/> <input type="button" value="START"/>	BLOCK MOVE [] BLOCK MOVE DONE

Output To Serial Port*

To transfer data to the serial port from the programmer RAM, use this operation. When transfer is completed, the programmer calculates and displays the sumcheck of the transferred data.

After setting up the serial port and selecting the appropriate data translation format (see remote control section), use the following procedure to transfer the data from the 298 RAM to the serial port using front panel control.

Procedure

Keystroke

298 Displays

1. Select the copy operation.
2. Select the source of the data to be transferred.

COPY DATA FROM

RAM_N ADDR_N/SIZE

3. Accept or (optionally) change begin RAM address and/or block size. ** Defaults are 00000 and all of RAM.

NOTE

To specify only the block size, just press START and key in the size.

RAM ADDR_N/SIZE

RAM XXXXX/XXXXX

4. Select the destination for the data.

CO RAM>POR_N ADDR

5. Accept or optionally change the offset address. Execute the operation.

OUTPUT PORT

OUTPUT DONE HHHH

sumcheck

6. To repeat the output operation, press

*The displays shown here may differ slightly if you are using a translator that can address over 64K bytes of data, for example, Intel MCS85, or Motorola Extormax.

**For 8- or 16-bit devices, the block size displayed represents the number of 16-bit words that will be transferred. For example, a block size of 8000 x 16 will output 64K.

Edit Operations

The following pages describe use of the programmer's Edit function. Using the EDIT key, you may (1) edit data in RAM before it is programmed into a device, and you may (2) observe and compare device data with RAM data. You may edit data in the programmer's RAM using any of three number bases: hexadecimal, octal or binary. (Hexadecimal is the default value.) The number base is determined by a select code [F5, F6 or F7]. Check the select codes section for the key sequence to enable a particular number base. When keying in data, you may only key in values allowed in that number base. For example, if you were editing data using the octal base, you could only use keys 0 through 7.

NOTE

*If you attempt to edit an address outside the range of the device, two asterisks will appear in the 298's display (D**). When editing data for a by-16 (16-bit) device, four asterisks will appear. The by-16 devices may only be edited in the hexadecimal number base.*

Editing Using A Hexadecimal Base

Use the following procedure to edit data in a RAM address using the hexadecimal number base:

Procedure	Keystroke	298 Displays
-----------	-----------	--------------

1. Select the edit operation.

EDIT ADDR ^ HHHHH

2. Key in the hex RAM address to be edited*.

EDIT ADDR ^ HHHHH
HHHHH DHH ^ RHH

3. Key in the hex data to be entered of this RAM address*. Use the REVIEW key to clear RAM.

HHHHH DHH ^ RHH
address device RAM
data data

NOTE

The new data is displayed following the ^R.

4. Press one of the following keys to increment (+ 1), or decrement (- 1) the edit address. Press EDIT to go to another RAM address.

increments (+ 1) to edit the next higher RAM location.
decrements (- 1) to edit the next lower RAM location.
allows selection of another RAM address for editing.

5. Press any other function (blue) key (COPY, VERIFY, SELECT) to exit from the EDIT operation.

*For by-16 (16-bit) devices, key in a 5-digit address and 4 digit data.
A by-16 edit display looks like this:

HHHHH DDDD ^ RRRR
by-16 device RAM
address data data
word word

Verify Operations

After the 295 has executed a Copy operation, you may check that the data was transferred correctly from RAM to the device or port by using the Verify operation. A Verify compares the RAM data with the port or device data, to make sure they match. The programmer always performs a two-pass verify (low and high voltage). If the data does not match on a device-to-RAM verify, the programmer will display the address location where the error occurs, and also indicate whether the high or low voltage verify failed. A port verify that is unsuccessful will cause the programmer to display

I/O VERIFY FAIL 52.

Verify RAM Data Against Serial Port Data

In addition to verifying against master device data, you may also use the verify operation to ensure that the information transferred through the serial port matches the RAM data. Use the following procedure to verify that the RAM data is the same as the serial port data.

Procedure	Keystroke	298 Displays																					
1. Select the verify operation.	<input type="button" value="VERIFY"/>	VERIFY DATA FROM																					
2. Select the source of the data to be verified.	<input type="button" value="RAM"/>	RAM ^ ADDR / SIZE																					
3. Accept or (optionally) change the begin address and/or block size of RAM.	<table border="0"> <tr> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="START"/></td> </tr> <tr> <td colspan="6"></td> <td>OR</td> </tr> <tr> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="START"/></td> </tr> </table>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="START"/>							OR	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="START"/>	RAM ADDR ^ SIZE RAM XXXXXX/XXXXXX
<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="START"/>																	
						OR																	
<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="START"/>																	
<i>NOTE</i> To specify only the block size, just press START and key in the size.	<input type="button" value="PORT"/>	VE RAM>POR ^ ADDR																					
4. Select the source of the data to be verified against the RAM data.	<input type="button" value="START"/>	VERIFY PORT <input type="checkbox"/>																					
5. Accept or optionally change the port address to begin verifying from.	<table border="0"> <tr> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="START"/></td> </tr> <tr> <td colspan="6"></td> <td>OR</td> </tr> <tr> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="START"/></td> </tr> </table>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="START"/>							OR	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="START"/>	VE POR DONE HHHH		
<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="START"/>																		
						OR																	
<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="X"/>	<input type="button" value="START"/>																		
6. If the data in RAM does not match the port data, the programmer will display this message		I/O VFY FAIL 52																					

Verify RAM Data Against Device Data

Use the following front panel keyboard procedure to verify that the data in the 298 RAM is the same as the data in the device:

Procedure	Keystroke	298 Displays												
1. Select the verify operation.	<input type="checkbox"/> YES/1	VERIFY DATA FROM												
2. Select the source of the data to be verified.	<input type="checkbox"/> RAM	RAM ^ ADDR / SIZE												
3. Accept or (optionally) change the begin RAM address and/or size of the block of RAM.	<table border="1"> <tr> <td><input type="checkbox"/> START</td> <td><input type="checkbox"/> OR</td> <td><input type="checkbox"/> START</td> <td>RAM ADDR ^ SIZE</td> </tr> <tr> <td><input type="checkbox"/> X</td> <td><input type="checkbox"/> X</td> <td><input type="checkbox"/> X</td> <td>RAM XXXXXX/XXXXXX</td> </tr> <tr> <td><input type="checkbox"/> X</td> <td><input type="checkbox"/> X</td> <td><input type="checkbox"/> X</td> <td><input type="checkbox"/> START</td> </tr> </table>	<input type="checkbox"/> START	<input type="checkbox"/> OR	<input type="checkbox"/> START	RAM ADDR ^ SIZE	<input type="checkbox"/> X	<input type="checkbox"/> X	<input type="checkbox"/> X	RAM XXXXXX/XXXXXX	<input type="checkbox"/> X	<input type="checkbox"/> X	<input type="checkbox"/> X	<input type="checkbox"/> START	
<input type="checkbox"/> START	<input type="checkbox"/> OR	<input type="checkbox"/> START	RAM ADDR ^ SIZE											
<input type="checkbox"/> X	<input type="checkbox"/> X	<input type="checkbox"/> X	RAM XXXXXX/XXXXXX											
<input type="checkbox"/> X	<input type="checkbox"/> X	<input type="checkbox"/> X	<input type="checkbox"/> START											
4. Select the source of the data to be verified against RAM data.	<input type="checkbox"/> DEVICE	VE RAM> DEV ^ ADDR												
5. Accept or (optionally) change begin device address.	<table border="1"> <tr> <td><input type="checkbox"/> START</td> <td><input type="checkbox"/> OR</td> <td><input type="checkbox"/> START</td> <td>FAM ^ PIN 00</td> </tr> <tr> <td><input type="checkbox"/> X</td> <td><input type="checkbox"/> X</td> <td><input type="checkbox"/> X</td> <td>FAM ^ PIN 00</td> </tr> </table>	<input type="checkbox"/> START	<input type="checkbox"/> OR	<input type="checkbox"/> START	FAM ^ PIN 00	<input type="checkbox"/> X	<input type="checkbox"/> X	<input type="checkbox"/> X	FAM ^ PIN 00					
<input type="checkbox"/> START	<input type="checkbox"/> OR	<input type="checkbox"/> START	FAM ^ PIN 00											
<input type="checkbox"/> X	<input type="checkbox"/> X	<input type="checkbox"/> X	FAM ^ PIN 00											

Verify RAM Data Against Device Data (continued)

Procedure

Keystroke

298 Displays

6. Enter the family/pinout code (see the list included with your programming Pck).

FAM ^ XX PIN XX

7. Insert and lock the device into the appropriate socket. Execute the operation.

VERIFY DEVICE
VE DEV DONE HHHH

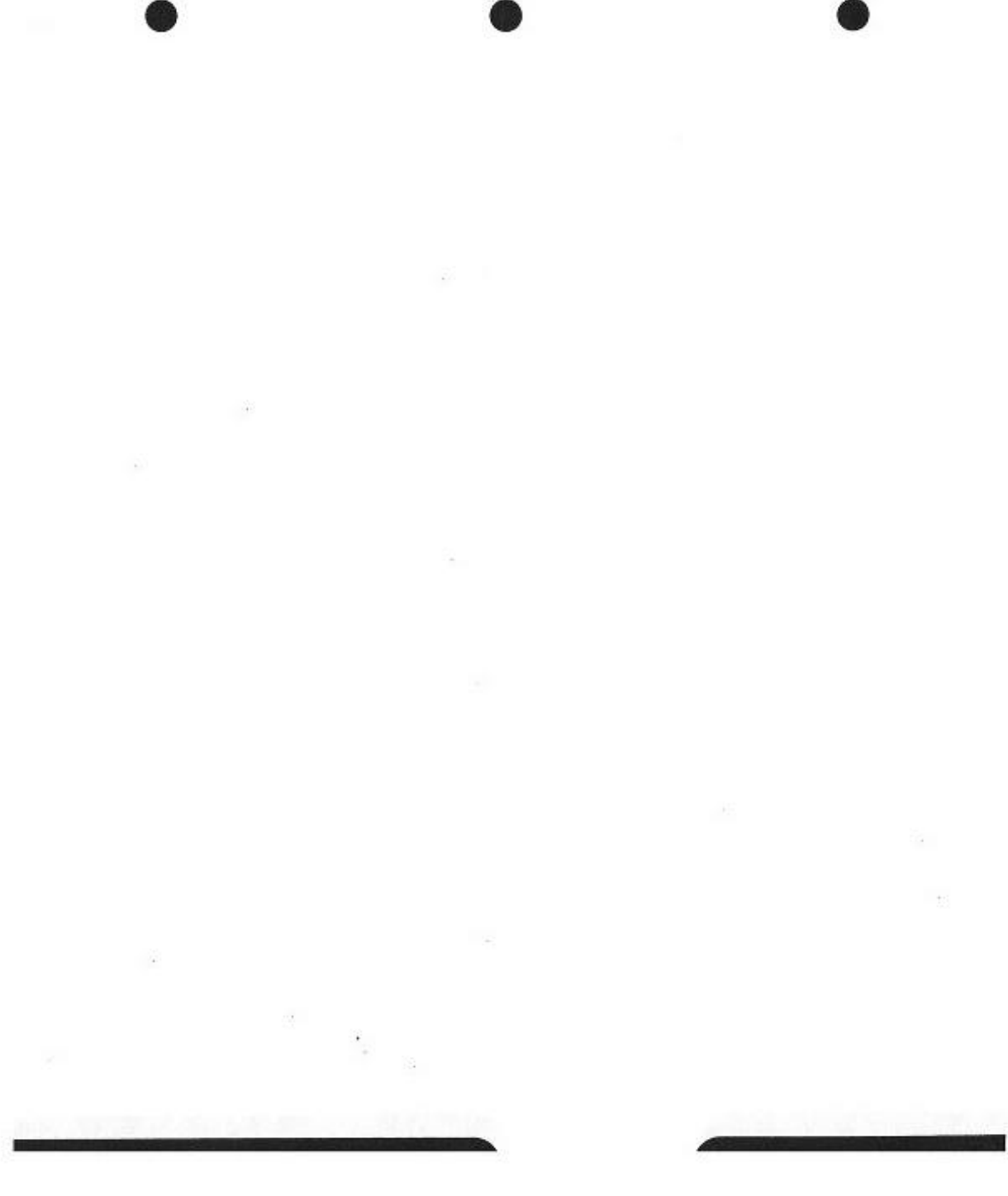
8. To verify additional identical devices using the data stored in RAM return to step 7. Otherwise, remove the device from the socket. If the device does not verify, the following displays will signal the errors. (Display shown on the second and third lines appears only when using a by-16 (16-bit) device.) Press START to display the next address that did not verify.

NOTE

The LogicPck verify error message is different. Refer to the LogicPck manual for the error message format.

VN HHHH DHH RHH
 verify test address device RAM
 that failed (1 or 2) data data

VN VERIFY FAIL
 verify test
 device
 that failed
 (1 or 2)
 HHHHH DDDD/RRRR
 address device RAM
 data data





The page is otherwise blank, with only a few scattered, very faint and illegible marks or artifacts visible.

Remote Control

This section of the manual contains the following information:

- SYSTEM SETUP**—Explains how to set up the 298 for remote control operation. Includes information on setting the baud rate, parity and stop bits, setting up the RS232 serial port and the optional paper tape reader.
- COMPUTER REMOTE CONTROL (CRC) OPERATION**—Describes operation of the 298 under CRC. Includes a complete command summary.
- SYSTEM REMOTE CONTROL (SRC) OPERATION**—Describes operation of the 298 under SRC. Includes key sequences required to execute programming operations while in SRC mode.
- DATA TRANSLATION FORMATS**—Defines the data translation formats compatible with the 298. Includes a complete list of all the formats, with examples of each.
- 298/HANDLER INTERFACE**—Describes use of the 298 with an IC handler. Includes handler port pin assignments and timing diagrams.

System Setup

This section explains how to set up your 298 programmer for serial I/O operation. Included is information on setting baud rate, parity and stop bits, and RS232 connector pin assignments.

Transferring to and From Remote Control Mode

You may transfer control of the 298 to a terminal by using hexadecimal codes F4 or FB. Code F4 enables Computer Remote Control (CRC) and code FB enables System Remote Control (SRC). The key sequences are listed in the Select Functions section of the manual.

When all terminal operations have been completed, you may return system control to the 298 by using another hexadecimal code.

CRC: To exit CRC via the 298's keyboard, press any of the four mode keys (COPY, VERIFY, SELECT or EDIT). To exit using the terminal's keyboard, press Z(CR) to exit. If the "-" (underscore) command has been executed, use Z(CR), remove and reinstall the Pdk and then either restart or power down.

SRC: To exit SRC, press any of the 298's four front panel keyboard mode keys (COPY, VERIFY, SELECT or EDIT); however, the port remains enabled until the programmer is powered down.

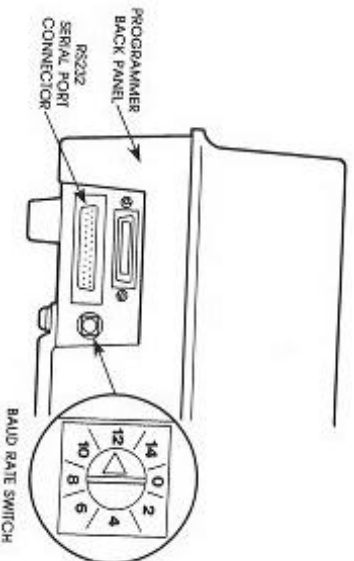
Setting Parameters

Before the 298 can operate with another system, three parameters must be set: parity, stop bits and the baud rate. These parameters must be the same for both systems.

Baud Rate

To set the baud rate, refer to the figure and follow this procedure:

1. Locate the baud rate rotary switch on the programmer's rear panel.
2. Locate the desired baud rate on the chart in the figure and the switch position required for that baud rate.
3. With a flat-blade screwdriver, turn the switch to the numbered switch position that corresponds to your desired baud rate.



BAUD RATE	SWITCH POSITION
50	0
75	1
110	2
134.5	3
150	4
300	5
600	6
1200	7
1800	8
2000	9
2400	10 or A
3600	11 or B
4800	12 or C
7200	13 or D
9600	14 or E
19200	15 or F

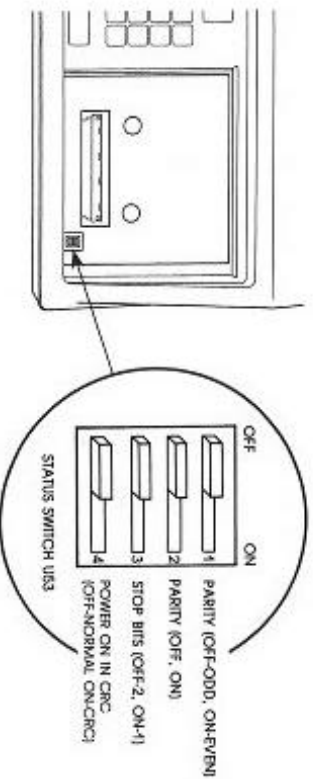
Parity and Stop Bits

To set parity and stop bits, refer to the figure and follow this procedure:

CAUTION

Make sure that all device(s) have been removed from the Pak socket(s) before removing the Pak or turning off the programmer. Voltage transients caused by removing the Pak could damage the device(s).

1. Turn off the programmer.
2. Remove the Pak from the programmer (see the Getting Started section).
3. Access the controller board's status switch (US3) through the cut-out in the lower right corner of the protective shield.



4. Flip the switches to the desired settings; the figure shows the switch positions set by the factory.
5. Reinstall the programming Pak and turn on the programmer.

NOTE

You may override the parity and stop bit settings by using CRC commands D, E, N, J or K. Check the I/O commands portion of the CRC Command Summary Table.

Serial Port Hookup Cabling

To connect the 298 to other instruments, you must use the serial interface connector pin assignment listed in the table. Any computer or any other peripheral device that interfaces to the 298's serial port must allow for duplex operation consistent with the 298 programmer's remote control software. System Remote Control (SRC) and Computer Remote Control (CRC) are half duplex. The figure shows sample interconnections to the serial interface for half/full duplex with handshake and without handshake and the current loop connection for full duplex and half duplex. See the baud rate switch illustration for the location of the RS232 serial port connector.

The 5-wire handshake interconnection may be used to download data at any supported baud rate when using a host system that recognizes the 5-wire protocol. The 5-wire handshake and control codes 1 or 2 must be used when transmitting formats 10, 11 or 89 at 9600 baud.

NOTE

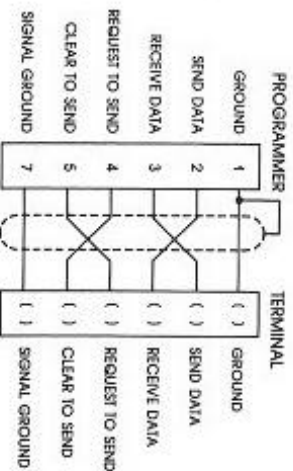
To reduce electromagnetic interference (EMI), we recommend using a shielded cable.

Serial Interface Connector Pin Assignment

Pin No	Signal Mnemonic	Description
1	Ground	In the RS232 environment, this line is common for the -12V and provides a safety ground connection to the RS232-compatible terminal. In the TTY environment, the -12V VDC signal line provides the signal return for a TTY terminal.
2	Send Data	Transmits data at RS232 voltage levels (+12V and -5V).
3	Receive Data	Accepts data at RS232 voltage levels.
4	Request To Send	This line is normally held high by the programmer. It goes low to inhibit data transmission from a remote source.
5	Clear to Send*	A high level on this line allows the programmer to transfer data. A low level inhibits data transfer.
6	Data Set Ready	Connected by internal jumper to Data Ready (pin 20).
7	Signal Ground	This line provides a common signal connection to the RS232 remote source.
8	Carrier Detect*	This line is positive when modem detects a carrier signal. This line is scripted by the programmer if used.
9	+24 Vdc	Available for external use if required (500 mA maximum). Not used.
10		Transmits data using active 20 mA current loop.
11	20 mA Send	Accepts data using active 20 mA current loop.
12	20 mA Receive	Receive Data on pin 12 is internally converted to RS232 levels. Output on pin 13 should be jumpered externally to Receive Data, pin 3.
13	Detect 20 mA	Not used.
14-19		Not used.
20	Data Ready	Connected by internal jumper to Data Set Ready, pin 6. A high level on this line from the RS232 data terminal indicates that the data terminal is ready.
21		Not used.
22	+5 Vdc	Available for external use if required (200 mA maximum).
23	-5 Vdc	Available for external use if required (200 mA maximum).
24-25		Not used.

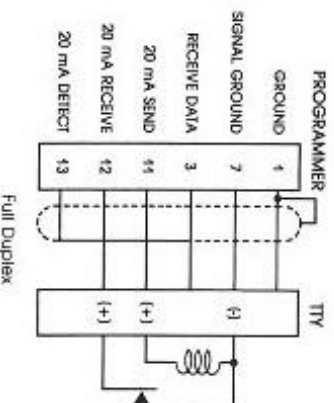
* Pins 5 and 8 have internal pull-ups and need no connection if unused.

RS232 Connection

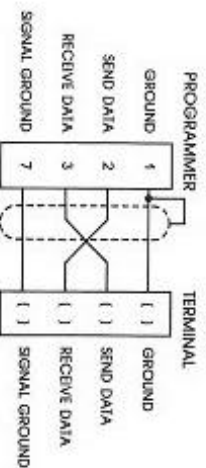


Half-Full Duplex, With Handshake

20 mA Current Loop Connection



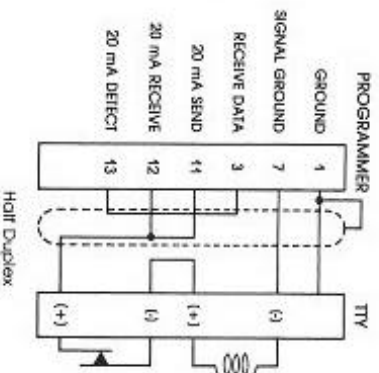
Full Duplex



Half-Full Duplex, Without Handshake

NOTES

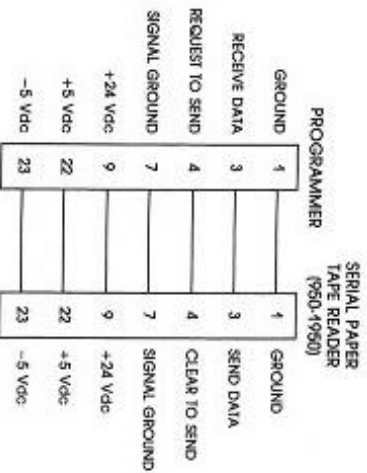
1. All signals are normed with respect to the originating unit.
2. All undesignated pins are to be left open.
3. For applications that do not require handshaking, the programmer's clear to send line is pulled up internally.
4. Host system's pin numbers may differ.



Half Duplex

Hooking up a Serial Paper Tape Reader

A Data I/O Serial Paper Tape Reader (950-1950) can be connected to your programmer. A direct connection to the programmer's serial port is made using the existing serial paper tape reader cable. It will connect according to the specifications in the figure. Set the baud rate at 2400. Two operations are possible using the serial paper tape reader: load RAM from the serial port and verify RAM from the serial port. See the programming section for the keyboard sequences to execute these operations.

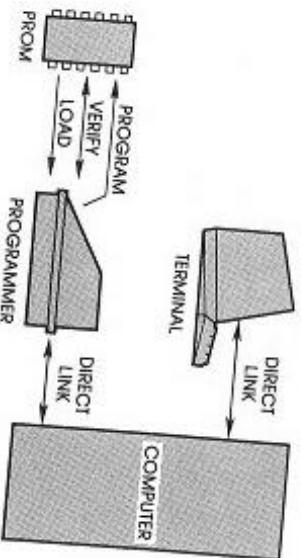


NOTE
All undesignated pins are to be left open.

Computer Remote Control Overview

Computer Remote Control (CRC) is designed to enable you to control the 298 Programmer by a user's computer. Linked directly to the programmer, the computer generates and sends commands to the programmer; determines variables for setting programming parameters (where needed), and reacts to information returned to it from the programmer. While these commands may be sent by the operator at a terminal, the commands and syntax described in this manual were designed to be easily incorporated into a computer program.

The figure below illustrates the basic components of the 298 under computer remote control (CRC). For interactive programs, the computer can both send messages to be displayed on the 298 and allow his computer to issue CRC commands and to interpret CRC responses.



Remote Control

Response Characters

The programmer sends a response character to the computer after every command; the table below summarizes these characters. A ">" symbol for a response means that the command was successfully executed. Whenever an error occurs, the 298 will send an F to the computer. The computer or the operator can respond by interrogating the programmer with the X or F command. The X command causes the programmer to send the computer a complete list of the last 16 error codes that have occurred. The F command codes all errors into a 32-bit error status word. A description of the error status word follows the CRC command description list.

Character	Name	Description
>	Prompt	Sent on entering remote control, after an ESCAPE or BREAK key has halted a command, or after a command has been successfully executed. The programmer then transmits a carriage return.
F	Foil	Informs the computer that the programmer has failed to execute the last command entered. The programmer then transmits a carriage return.
?	Question	Informs the computer that the programmer does not understand a command or the command was invalid. The programmer then transmits a carriage return.

CRC Command Summary

ASCII Character	Command Description	ASCII Character	Command Description
CR	Execute command	K	Set 2 stop bits
BREAK	Abort binary transfer	L	Load from device
ESCAPE	Abort	HH M	Select record size
H ^{1st}	Bitting control (IC handler use only)	N	Select no parity
% ^(e)	Handler start (IC handler use only)	O	Output Program
HHHHH :	Set begin device address	P	Swap nibbles
HHHHH :	Set block limit	R	Respond device
HHHHH <	Set begin RAM address	S	Checksum RAM
=	Disable timeout	T	Illegal bit test
HHHHH >	Shuffle RAM data	HH U	Set nulls
HHHHH ?	Split RAM data	V	Verify
HHHH @	Select family/pinout code for device	HHHH W	Set address offset
HH A	Select data translation format	X	Error code inquiry
B	Blank device test	Y	Parity error
C	Input compare	Z	Escape remote control
D	Select odd parity	[Family/pinout inquiry
E	Select even parity	\	RAM-RAM block move
F	Error status inquiry	HH]	Select function (see select functions section for list)
G	Software configuration	^	Clear all RAM
H(letters H)	No operation	~	Send ASCII character to computer
I	Input		Send ASCII characters to 298's display
J	Set 1 stop bit		

^(e)See 298/Handler discussion at end of this section.

NOTE

Other than the CRC No Operation command (which is the letter "H" on the keyboard), all "H" characters in the table denote a hexadecimal value to be keyed in.

Description of CRC Commands

A description of all the CRC commands is given in the following table. The commands are divided into six categories, shown in bold type. Handler commands are used only when an optional Handler is used with the 298. The following abbreviations are used in the response column of the table to indicate the terminal's display.

- > Prompt character on terminal display
- CRLF Carriage return followed by a line feed
- HHHH Data, expressed in hexadecimal

CRC Command Descriptions

Command	Name	Response	Description
CONTROL COMMANDS			
RETURN		>CRLF	Execute
ESCAPE		>CRLF	Abort.
BREAK		>CRLF	Abort binary transfer.
UTILITY COMMANDS			
G(CR)	Software Configuration	HHHH>CRLF	Sends a 4-digit hex number configuration representing the software revision of the programmer.
HHHHH <(CR)	Set Begin RAM	>CRLF	Defines the first RAM address for data transfers. Default is 00000.
HHHHH :(CR)	Set Block	>CRLF	Sets the number of bytes to be transferred or programmed. Default is the device size or the RAM limit less the begin RAM address. Must be set for a RAM-RAM block move.

Remote Control

Command	Name	Response	Description
UTILITY COMMANDS (continued)			
HHHHH : (CR)	Set Begin Device Address	> CRLF	Sets the first device address to be programmed or the RAM destination address in a RAM-RAM block move. Default is 00000.
HH J(CR)	Select Function	> CRLF	Accesses the external select codes (HH) in the extended software of some programming Poks.
S(CR)	Sumcheck	HHHH>CRLF	Calculates the sumcheck of the RAM data from the beginning RAM pointer up to the word limit of the selected device.
F(CR)	Error Status Inquiry	HHHHHHHH>CRLF	Returns a 32-bit word that codes the accumulated errors (see description following this table).
X(CR)	Error Code Inquiry	HHHH>CRLF	Outputs the error codes stored in the scratch-RAM.
H(CR)	No Operation	> CRLF	No-op command that returns a prompt character (>).
Z(CR)	Escape Remote Control	None	Returns control of the 298 to the front panel.
-(CR) 298 Keystroke	Send ASCII Characters		Allow computer to recognize the 298's keyboard entries and manipulate the programmer's display. See description following this table.
Y(CR) 298 Display (CR)	Receive ASCII Characters		

Command	Name	Response	Description
DEVICE COMMANDS			
T(CR)	Illegal-Bit Test	>CRLF	Test for an illegal bit in the device.
B(CR)	Blank Check	>CRLF	Checks that no bits are programmed in the device.
I(CR)	Family and Pinout	FFPP>CRLF	Sends a 4-digit hex code (FFPP) representing the Family and Pinout Code. Sends F when the code is not required.
FFPP @ (CR)	Select Family	>CRLF	Selects the 2-digit hex Family code (FF) and the 2-digit hex Pinout code (PP) required to program a particular device.
R(CR)	Respond	AAAA/B/C>CRLF	Indicates the device status. Outputs AAA(A)B(C) where: AAA or AAAA = the device word limit, B = the word size, and C = VOL (1)/VCH(0) status.
L(CR)	Load	>CRLF	Load the device data into the RAM.
P(CR)	Program	>CRLF	Program the RAM data into the device.
V(CR)	Verify	>CRLF	Verify the device data against the RAM data.

Command	Name	Response	Description
I/O COMMANDS			
D(CR)	Select Odd Parity	>CRLF	Sets odd parity. Default is the parity switch setting.
E(CR)	Select Even Parity	>CRLF	Sets even parity. Default is the parity switch setting.
N(CR)	Select No Parity	>CRLF	Sets no parity. Default is the parity switch setting.
J(CR)	Set 1 Stop Bit	>CRLF	Sets 1 stop bit. Default is the stop-bit switch setting.
K(CR)	Set 2 Stop Bits	>CRLF	Sets 2 stop bits. Default is the stop-bit switch setting.
CFF A(CR)	Select Transition Format	>CRLF	Defines the instrument control code (C)* and data transition format (FF) for IO data transfers. Default is instrument control code zero (0) and MDS technology format (#91).
HH M(CR)	Select Record Size	>CRLF	Defines (HH) the output record size. Default is 16 bytes per record (8 bytes per record in the Fairchild Fairbug format).
HH U(CR)	Set Nulls	>CRLF	Defines (HH) the number of nulls output after the carriage returns and enables the line feeds. Default is FF (no nulls) and no line feeds.

* See the data transition formats discussion in this section of the manual for details on the instrument control code.

Command	Name	Response	Description
I/O COMMANDS (continued)			
HHHH W(CR)	Set Address Offset	>CRLF	Defines the offset address on output and the value subtracted on input. Default is 0 on output or the first incoming address on input.
=(CR)	Disable Timeout	>CRLF	Disables the 25-second I/O timeout. Restored only if power-on.
I(CR)	Input	>CRLF ^(a)	Input data from the source to the RAM.
O(CR)	Output	>CRLF ^(a)	Outputs data from the RAM to the computer.
C(CR)	Compare	>CRLF ^(a)	Compares the RAM data with the data sent from the computer.
Y(CR)	Parity Error	HHHH>CRLF	Responds with the hex number of parity errors since the last Y command, power on, or parity command [D, E, or N].

(a) Response occurs after data transmission with the proper termination.

Remote Control

Command	Name	Response	Description
EDITING COMMANDS			
Q(CR)	Swap Nibbles	> CRLF	Exchanges the high- and low-order halves of every word in the RAM.
V(CR)	RAM-RAM Block Move	> CRLF	Initiates a data block transfer from one RAM location to another. Begin RAM address, block size, and begin device address must be set first.
HHHHH ?(CR)	Split RAM Data	> CRLF	Used for 16-bit microprocessor data. Splits the even- and odd-numbered bytes into two blocks, separated by a center point. HHHH, which must be a power of 2 between 0 and the RAM midpoint. Default is the RAM midpoint.
HHHHH >	Shuffle RAM Data	> CRLF	Used for 16-bit microprocessor data. Merges the block above the block below. The center point must be a power of 2 between 0 and the RAM midpoint. Default is the RAM midpoint.
^	Clear All RAM	> CRLF	Clears the programmers data RAM to all zeros.

CRC Interactive Commands

Two CRC commands allow the programmer's display and keyboard to be used as a terminal while in CRC mode. The computer can prompt the user via the 298's display, then read back the user's keyed-in response.

The Command

The (underscore) command allows the terminal to recognize the 298's front panel keystrokes. Each key on the 298 sends a corresponding ASCII code, listed in the following table.

NOTE

If this command has been executed, you cannot exit CRC by striking one of the 298's function keys. To exit, either (1) press the "Z" key on the terminal's keyboard, followed by a carriage return, or (2) remove and reinstall the Pak or (3) power down.

298 Key Pressed	Corresponding ASCII Code Sent
0-9	0-9
A-F	A-F
RAM	RA
PORT	PO
DEVICE	DE
COPY	CO
VERIFY	VE
SELECT	SE
EDIT	ED
START	ST
REVIEW	RE
No key pressed	Prompt only (>)

The \ Command

The \ (back apostrophe) command allows you to transmit ASCII characters to the 298's display. The valid ASCII codes and their corresponding hexadecimal codes are shown in the following table.

Hex Code Transmitted	ASCII Code Displayed on the 298
30H through 39H	0-9
41H through 5AH	A-Z
20H	Space
21H through 28H	Action symbol
29H	≡
2AH	*
2BH	+
2CH	' (apostrophe)
2DH	— (hyphen)
2EH	· (period)
2FH	/
3AH	— (short hyphen)
3BH	all display segments lit
3CH	<
3DH	=
3EH	>
3FH	?
40H	@
58H	<
59H	>
5AH	beeper ^(a)
07H	

^(a) This code may be sent along with a message, and does not represent a display character. If this code is sent by itself, it will simply cause the 298's beeper to sound briefly.

Error Status Word

After executing the CRC "F" command, a 32-bit error status word will be sent to the computer (or terminal). The error status word format is shown in the following table. The table shows that the eight-character word is broken into four two-character groups. The first two-character group defines receive errors, the second group defines programming errors, the third group defines I/O errors and the last group defines RAM errors. Each two character group contains a maximum of eight bits, with each bit representing the presence of the defined error or error type. Multiple errors within a character do occur; therefore, if all four bits within the character are used and present, the transmitted character would be 'f', while if there were no errors present for that character, zero would be transmitted.

Type of Error	Bit Number	Value	Description
RECEIVE ERRORS	31	8	ANY ERROR. If the word contains any errors, the most significant bit (bit 31) will be high
	30	4	Not used
	29	2	Not used
	28	1	Not used
	27	8	Not used
	26	4	Serial-overflow error (42)
	25	2	Serial-framing error (41, 43)
	24	1	Buffer overflow, i.e., >15 characters (48)
PROGRAMMING ERRORS	23	8	Any device-related error
	22	4	Start line not set high (26)
	21	2	Blocksize + begin device address exceeds device address
	20	1	Composite DAC error
	19	8	Device not blank (20)
	18	4	Illegal bit (21)
	17	2	Nonverity (23, 24, 29)
	16	1	Incomplete programming, or no programming module (22, 25, 30-39)

Type of Error	Bit Number	Value	Description
I/O ERRORS	15	8	I/O error (46, 50, 58, 59, 94, 95 or any I/O error)
	14	4	Not used
	13	2	Not used
	12	1	Compare error (52)
	11	8	Checksum error (82)
	10	4	Record-count error, MOS Technology (93)
			Address-check error, Signetics and Tek Hex (92)
			Record-type error, Intel Intellec 81MDS (94)
			Address error, Ia.>word limit (27, 28, 54, 56, 57, 95)
			Data not hexadecimal (84, 85, 91)
RAM ERRORS	9	2	Insufficient data received, ASCII-Hex and Octal (54)
	8	1	
	7	8	RAM-hardware error (64, 66 or any RAM error)
	6	4	Not used
	5	2	Blocksize + begin RAM address exceeds RAM address limit (27)
	4	1	Invalid center point for split or shuffle
	3	8	Illegal split or shuffle
	2	4	No RAM or insufficient RAM resident (61)
	1	2	RAM write error, or program-memory failure (63)
	0	1	RAM end not on 4K boundary (62, 69)

EXAMPLE: What errors are indicated in this error status word: 80C80081

8 — the word contains error information

0 — no receive errors

C — (= 8 + 4): 8 = Device-related error

4 = Start line not set high (error 26)

8 — device is not blank (error 20)

0 — no input errors

0 — no input errors

8 — RAM error (error 62, and possibly 64 and 66)

1 — RAM end is not on 4K boundary (error 62)

NOTES

1. The numbers in parentheses are 298 error codes defined in the error codes section.
2. An error can cause as many as 3 bits to be high; the bit which represents the error, the most significant bit of the 8-bit word in which the error bit occurs, and bit 31.
3. After being read, the error-status word resets to zeros.

System Remote Control

The programmer's System Remote Control (SRC) capability allows you control of the programmer's operation from a terminal. Once the controlling terminal has been properly interfaced to the programmer, select function F8 (Port Enable) must be entered from the 298's keyboard to enable SRC.

Command Protocol

The syntax for SRC is similar to that of the 298's front panel operations. When keying in commands from a terminal, the programmer recognizes the first two characters of each word (except REVIEW), as shown in the following SRC table.

NOTE

The format Menu can also be displayed by entering FO(CR).

Command Entry in SRC

298 Keyboard Command	Remote Control Command
Copy	CO(SP)
Verify	VE(SP)
Select	SE(SP)
Edit	ED(SP)
Device	DE(SP)
RAM	RA(SP)
Port	PO(SP)
Review	/

The space bar (denoted by the letters (SP) as shown in the table) is used after the command as a delimiter, setting the boundaries for that command. The programmer will not define the characters input until the space bar is entered. And, since the programmer only recognizes the first two characters of the command, some variations are possible. For example: the command COPY from DEVICE (XXXXX = Device Address, YYYYY = Size) to RAM (ZZZZZ = RAM Address) can be written with the same result in either of the following ways:

```
CO(SP)DE(SP)XXXXX(SP)YYYYY(SP)TO(SP)RA(SP)ZZZZZ(CR)
OR
COPY(SP)DEVICE(SP)XXXXX(SP)YYYYY(SP)TO(SP)RAM(SP)ZZZZZ(CR)
```

NOTE

The word "TO" must be keyed in before the destination when entering commands with SRC.

When entering data on the terminal, the slash (/) is used in place of the REVIEW key. When pressed, it causes the previous character(s), prior to a space bar entry, to be ignored. The characters are still displayed on the screen; however, all entries back to the previous space bar are ignored and the operator can enter the replacement characters.

You may enter SRC commands in two ways: direct or interactive. Direct entry uses the format shown in the above example. Entry of the device address, size and RAM address is optional. The interactive method streamlines the entries required. When a command is keyed in, the terminal's display is similar to that of the programmer. For example, if you key in CO(CR), the terminal will display COPY DATA FROM. The SRC command key sequences are shown on the following pages, both the 298s and the terminal's displays are shown.

NOTE

Just as with 298 front panel control, you can change the beginning address, block size and the destination address if needed. Read the general operating notes of the beginning of the programming section for more information. If you set a beginning address, you must also set the block size. Default for the beginning address is 00000.

Load RAM with Master Device Data

Use the following procedure to transfer programming data from a master device to the 298's RAM.

Procedure	Terminal Key Sequence	Terminal Display (2nd line)
-----------	-----------------------	-----------------------------

1. Select the copy operation.

298 (1st line)

COPY DATA FROM
COPY DATA FROM>

2. Select the source for the data transfer.

DEV ^ ADDR SIZE
DEV ADDR, SIZE>

3. Accept or (optionally) change the beginning device address and block size to copy. Default is 00000.

OR

X	X	X	X	X	X	
Y	Y	Y	Y	Y	Y	SPACE
	R	A				CR

CO DEV> RAM ^ ADDR
CO DEV> RAM ADDR>

4. Accept or (optionally) change the beginning RAM address. Default is 00000.

FAM ^ 00 PIN 00
FAM 00 PIN 00>

NOTE

If the programming Pak installed does not require entry of a family/pinout code, the 298 skips to the display shown in step 5.

Load RAM With Master Device Data (Continued)

Procedure	Terminal Key Sequence	298 (1st line) Terminal Display (2nd line)
5. Accept (or if display is wrong), key in the 4-digit family/pinout code for the device (check the device list received with your programming Pak).	<input type="text" value="CR"/> or <input type="text" value="F"/> <input type="text" value="F"/> <input type="text" value="P"/> <input type="text" value="P"/> <input type="text" value="CR"/>	LOAD DEVICE LOAD DEVICE>
6. Insert the master device into the socket with the illuminated LED below it.	<input type="text" value="CR"/>	¹ LOADING DEVICE <input checked="" type="checkbox"/> ¹ LOAD DONE HHHH <input type="checkbox"/> carriage return ² LOAD DONE HHHH> sumcheck of data transferred sumcheck of data transferred

7. Remove the master device. To repeat the operation, return to step 6.

¹298 Display
²Terminal Display

Program a Device With RAM Data

Use this procedure to transfer RAM data to a blank device to be programmed, using SRC operation.

- | Procedure | Terminal Key Sequence | Terminal Display (2nd line) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|--------------------------------|------------------------------------|------------------------------------|--------------------------------|---------------------------------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|------------------------------------|--------------------------------|--------------------------------|------------------------------------|--|--|--|--------------------------------|--------------------------------|---------------------------------|--|--|--|---|
| 1. Select the copy operation. | <input type="text" value="C"/> <input type="text" value="0"/> <input type="text" value="CR"/> | 29B (1st line)
COPY DATA FROM
COPY DATA FROM> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2. Select the source of the data to be copied. | <input type="text" value="R"/> <input type="text" value="A"/> <input type="text" value="CR"/> | RAM ^ ADDR SIZE
RAM ADDR, SIZE> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3. Accept or (optionally) change the beginning device address and block size to transfer. Default is 00000. | <table border="1"> <tr> <td><input type="text" value="T"/></td> <td><input type="text" value="0"/></td> <td><input type="text" value="SPACE"/></td> </tr> <tr> <td><input type="text" value="D"/></td> <td><input type="text" value="E"/></td> <td><input type="text" value="CR"/></td> </tr> </table> <p>OR</p> <table border="1"> <tr> <td><input type="text" value="X"/></td> <td><input type="text" value="X"/></td> <td><input type="text" value="X"/></td> <td><input type="text" value="X"/></td> <td><input type="text" value="X"/></td> <td><input type="text" value="1"/></td> </tr> <tr> <td><input type="text" value="Y"/></td> <td><input type="text" value="Y"/></td> <td><input type="text" value="Y"/></td> <td><input type="text" value="Y"/></td> <td><input type="text" value="Y"/></td> <td><input type="text" value="SPACE"/></td> </tr> <tr> <td><input type="text" value="T"/></td> <td><input type="text" value="0"/></td> <td><input type="text" value="SPACE"/></td> <td colspan="3"></td> </tr> <tr> <td><input type="text" value="D"/></td> <td><input type="text" value="E"/></td> <td><input type="text" value="CR"/></td> <td colspan="3"></td> </tr> </table> | <input type="text" value="T"/> | <input type="text" value="0"/> | <input type="text" value="SPACE"/> | <input type="text" value="D"/> | <input type="text" value="E"/> | <input type="text" value="CR"/> | <input type="text" value="X"/> | <input type="text" value="X"/> | <input type="text" value="X"/> | <input type="text" value="X"/> | <input type="text" value="X"/> | <input type="text" value="1"/> | <input type="text" value="Y"/> | <input type="text" value="Y"/> | <input type="text" value="Y"/> | <input type="text" value="Y"/> | <input type="text" value="Y"/> | <input type="text" value="SPACE"/> | <input type="text" value="T"/> | <input type="text" value="0"/> | <input type="text" value="SPACE"/> | | | | <input type="text" value="D"/> | <input type="text" value="E"/> | <input type="text" value="CR"/> | | | | CO RAM> DEV ^ ADDR
CO RAM> DEV ADDR> |
| <input type="text" value="T"/> | <input type="text" value="0"/> | <input type="text" value="SPACE"/> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <input type="text" value="D"/> | <input type="text" value="E"/> | <input type="text" value="CR"/> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <input type="text" value="X"/> | <input type="text" value="X"/> | <input type="text" value="X"/> | <input type="text" value="X"/> | <input type="text" value="X"/> | <input type="text" value="1"/> | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <input type="text" value="Y"/> | <input type="text" value="Y"/> | <input type="text" value="Y"/> | <input type="text" value="Y"/> | <input type="text" value="Y"/> | <input type="text" value="SPACE"/> | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <input type="text" value="T"/> | <input type="text" value="0"/> | <input type="text" value="SPACE"/> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <input type="text" value="D"/> | <input type="text" value="E"/> | <input type="text" value="CR"/> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Program a Device With RAM Data (Continued)

Procedure

Terminal Key Sequence

Terminal Display (2nd line)

298 (1st line)

4. Accept or (optionally) change the address to begin transferring to. Default is the first device address, 00000.

CR					
or	2	2	2	2	2
or	2	2	2	2	CR

FAM ^ 00 PIN 00
 FAM 00 PIN 00>

If you are using a programming Pak that does not require entry of a family/pairout code, the 298 skips to the display shown in step 5.

NOTE

5. Accept, or if display is incorrect, key in the 4-digit family/pairout code for the device you are programming (check the device list included with the Pak).

CR					
or	F	F	P	P	CR

PROGRAM DEVICE
 PROGRAM DEVICE>

6. Insert the blank device into the socket with the illuminated LED below it.

NOTE

"NW" in the display represents the number of devices programmed since power-up.

```

^TEST DEVICE      [ ]
^PROGRAM DEVICE  [ ]
^VERIFY DEVICE   [ ]
^PRG DONE NW HHHH
  sumcheck of [ ]
  carriage return
  ? PRG DONE NW HHHH>
  sumcheck of [ ]
  data transferred
  
```

7. Remove the programmed device. To program another, return to step 6.

¹ 298 Display

² Terminal Display

Load RAM From the Serial Port

Use the following key sequence to transfer data through the serial port to the 298's RAM, using SPC operation.

Procedure	Terminal Key Sequence	298 (1st line) Terminal Display (2nd line)
1. Set up the serial port according to the instructions given at the beginning of this section.		
2. Select the appropriate data transition format (listed in the following subsection) and enable that format using select code 83 (see select functions section for exact key sequence).	<input type="text" value="C"/> <input type="text" value="0"/> <input type="text" value="OK"/>	<i>COPY DATA FROM</i> <i>COPY DATA FROM-></i>
3. Select the copy operation.	<input type="text" value="P"/> <input type="text" value="0"/> <input type="text" value="OK"/>	<i>POR ^ ADDR SIZE</i> <i>POR ADDR, SIZE-></i>
4. Select the source of the data to be copied.		

Load RAM From the Serial Port (Continued)

Procedure

Terminal Key Sequence

298 (1st line)
Terminal Display (2nd line)

5. Accept or (optionally) change the beginning port address and size of the block to copy. Default is 00000.

1	0	SPACE
R	A	OK

CO POR> RAM^ ADDR
CO POR> RAM ADDR>

NOTE

When the family/pinout code for a by-16 (16-bit) device has been selected, the port block size is in terms of 16-bit data. A hex block size of 100 (by-16) will load 200 bytes of data.

X	X	X	X	X	1
Y	Y	Y	Y	Y	SPACE
1	0	SPACE	R	A	CR

OR

					CR
Z	Z	Z	Z	Z	
					CR

OR

6. Accept or (optionally) change the beginning RAM address to transfer the data to. Default is 00000. Instruct the host system to send the data file. If the timeout is not turned off (select function F9), the file must be sent within 25 seconds or an error message will be displayed.

1 INPUT PORT

1 INPUT DONE HHHH
 └─┬─┘
 sumcheck of
 data transferred

carriage return
2 INPUT DONE HHHH
 └─┬─┘
 sumcheck of
 data transferred

298 Display
Terminal Display

Output RAM Data to the Serial Port

Use the following procedure to output data through the serial port, using SRC operation.

Procedure	Terminal Key Sequence	Terminal Display (2nd line)
1. Set up the serial port using the procedures described in the first part of this section.		

2. Select the appropriate data translation format from the list in the following subsection. Enable that format using selector code 83 (see the select functions section of the manual for the key sequence).

C	D	CR
---	---	----

COPY DATA FROM

COPY DATA FROM>

3. Select the source of the data to be copied.

R	A	CR
---	---	----

RAM\ ADDR SIZE

RAM ADDR, SIZE>

Output RAM Data to the Serial Port (Continued)

Procedure

Terminal Key Sequence

29B (1st line)
Terminal Display (2nd line)

5. Accept or (optionally) change the beginning RAM address and block size. Default is 00000.

T	0	SPACE
P	0	CR

```
CO RAM> POR ^ ADDR
CO RAM> POR ADDR>
```

NOTE

When the family/pinout code for a by-16 (16-bit) device has been selected, the port block size is in terms of 16-bit data. A hex block size of 100 (by-16) will load 200 bytes of data.

X	X	X	X	X	1
Y	Y	Y	Y	Y	SPACE
T	0	SPACE	P	0	CR

6. Accept or (optionally) change the first port address to begin copying. Default is 00000.

CR					
OR					
Z	Z	Z	Z	Z	CR

```
^ OUTPUT PORT 
^ OUTPUT DONE HHHH
```

sumcheck of data transferred
formatted data file scrolls through on display; then

```
? OUTPUT DONE HHHH
sumcheck of data transferred
```

29B Display
Terminal Display

Block Move

Use the following procedure to move data from one block to another in RAM, using SRC operation.

Procedure

Terminal Key Sequence

298 (1st line)
Terminal Display (2nd line)

1. Select the copy operation.

COPY DATA FROM
COPY DATA FROM>

2. Select the source of the data to be copied.

RAM \ ADDR SIZE
RAM ADDR, SIZE>

3. Accept or (optionally) change the beginning RAM address and size of the block to copy. Default is 00000.

CO RAM> RAM \ ADDR
CO RAM> RAM ADDR>

OR

X	X	X	X	X	X	!
Y	Y	Y	Y	Y	Y	SPACE
T	0	SPACE	R	A	A	CR

4. Accept or (optionally) change the first address in RAM to which the block will be moved. Default is 00000.

' BLOCK MOVE

OR

' BLOCK MOVE DONE

terminal shows no change

'298 Display

Verify RAM Data Against Device Data

Use the following procedure to verify that the data in the 298's RAM is the same as that in the device using SRC operation.

- | Procedure | Terminal Key Sequence | 298 (1st line)
Terminal Display (2nd line) | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|--|---|-----------------------------------|--------------------------------------|--------------------------------------|----------------------------------|----------------------------------|-----------------------------------|--|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|--------------------------------------|----------------------------------|----------------------------------|--------------------------------------|----------------------------------|----------------------------------|-----------------------------------|---|
| 1. Select the verify operation. | <input type="button" value="V"/> <input type="button" value="E"/> <input type="button" value="CR"/> | VERIFY DATA FROM
VERIFY DATA FROM> | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2. Select the source of the data to be verified. | <input type="button" value="D"/> <input type="button" value="E"/> <input type="button" value="CR"/> | DEV \ ADDR / SIZE
DEV ADDR, SIZE> | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3. Accept or (optionally) change the beginning address and size of the block to verify. Default is 000000. | <table border="1"> <tr> <td><input type="button" value="T"/></td> <td><input type="button" value="0"/></td> <td><input type="button" value="SPACE"/></td> <td><input type="button" value="CR"/></td> </tr> <tr> <td><input type="button" value="R"/></td> <td><input type="button" value="A"/></td> <td><input type="button" value="CR"/></td> <td></td> </tr> </table> OR <table border="1"> <tr> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="X"/></td> <td><input type="button" value="3"/></td> </tr> <tr> <td><input type="button" value="Y"/></td> <td><input type="button" value="Y"/></td> <td><input type="button" value="Y"/></td> <td><input type="button" value="Y"/></td> <td><input type="button" value="Y"/></td> <td><input type="button" value="SPACE"/></td> </tr> <tr> <td><input type="button" value="T"/></td> <td><input type="button" value="0"/></td> <td><input type="button" value="SPACE"/></td> <td><input type="button" value="R"/></td> <td><input type="button" value="A"/></td> <td><input type="button" value="CR"/></td> </tr> </table> | <input type="button" value="T"/> | <input type="button" value="0"/> | <input type="button" value="SPACE"/> | <input type="button" value="CR"/> | <input type="button" value="R"/> | <input type="button" value="A"/> | <input type="button" value="CR"/> | | <input type="button" value="X"/> | <input type="button" value="X"/> | <input type="button" value="X"/> | <input type="button" value="X"/> | <input type="button" value="X"/> | <input type="button" value="3"/> | <input type="button" value="Y"/> | <input type="button" value="Y"/> | <input type="button" value="Y"/> | <input type="button" value="Y"/> | <input type="button" value="Y"/> | <input type="button" value="SPACE"/> | <input type="button" value="T"/> | <input type="button" value="0"/> | <input type="button" value="SPACE"/> | <input type="button" value="R"/> | <input type="button" value="A"/> | <input type="button" value="CR"/> | VE DEV> RAM \ ADDR
VE DEV> RAM ADDR> |
| <input type="button" value="T"/> | <input type="button" value="0"/> | <input type="button" value="SPACE"/> | <input type="button" value="CR"/> | | | | | | | | | | | | | | | | | | | | | | | | | |
| <input type="button" value="R"/> | <input type="button" value="A"/> | <input type="button" value="CR"/> | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <input type="button" value="X"/> | <input type="button" value="X"/> | <input type="button" value="X"/> | <input type="button" value="X"/> | <input type="button" value="X"/> | <input type="button" value="3"/> | | | | | | | | | | | | | | | | | | | | | | | |
| <input type="button" value="Y"/> | <input type="button" value="Y"/> | <input type="button" value="Y"/> | <input type="button" value="Y"/> | <input type="button" value="Y"/> | <input type="button" value="SPACE"/> | | | | | | | | | | | | | | | | | | | | | | | |
| <input type="button" value="T"/> | <input type="button" value="0"/> | <input type="button" value="SPACE"/> | <input type="button" value="R"/> | <input type="button" value="A"/> | <input type="button" value="CR"/> | | | | | | | | | | | | | | | | | | | | | | | |

Verify RAM Data Against Device Data (Continued)

Procedure

Terminal Key Sequence

Terminal Display (2nd line)

4. Accept or (optionally) change the first address in RAM with which the block will be verified. Default is 00000.

NOTE

If you are using a programming Pak that does not require entry of a family/pinout code, the 298 skips to the display shown in step 5.

CR	OR
Z	Z
Z	Z
Z	Z
Z	Z
Z	Z
CR	CR

298 (1st line)
 FAM ^ 00 PIN 00
 FAM 00 PIN 00

5. Accept, or if displayed code is incorrect, key in the 4-digit family/pinout code for the device you are verifying (check the device list included with the Pak).

CR	OR
F	F
F	F
P	P
P	P
CR	CR

VERIFY DEVICE
 VERIFY DEVICE>

6. Insert the device into the socket with the illuminated LED below it.

VERIFY DEVICE
 ^VE DONE HHHH
 carriage return
 ^VE DONE HHHH
 sumcheck of
 data transferred
 sumcheck of
 data transferred

7. Remove the device. To verify another, return to step 6.

¹298 Display
²Terminal Display

Verify RAM Data With Serial Port Data

Procedure	Terminal Key Sequence	Terminal Display (2nd line)
		298 (1st line)

1. Set up the serial port using the procedures described in the first part of this section.
2. Select the appropriate data translation format from the list in the following subsection. Enable that format using select code 83 (see the select functions section of the manual for the key sequence).
3. Select the verify operation.
4. Select the source of the data to be verified.

V	E	CR
---	---	----

```
VERIFY DATA FROM
VERIFY DATA FROM>
```

P	0	CR
---	---	----

```
POR\ ADDR SIZE
POR ADDR, SIZE>
```

Verify RAM Data With Serial Port Data (Continued)

Procedure

Terminal Key Sequence

298 (1st line)
Terminal Display (2nd line)

5. Accept or (optionally) change the beginning address and block size to verify. Default is 00000.

T	0	space
R	A	CR

VE POR> RAM ^ ADDR
VE POR> RAM, ADDR>

OR

X	X	X	X	X	X	!
Y	Y	Y	Y	Y	Y	space
T	0	space	R	A	CR	CR

6. Accept or (optionally) change the address to begin verifying with. Default is the first RAM address, 00000.

OR

Z	Z	Z	Z	Z	CR
---	---	---	---	---	----

¹VERIFY PORT
¹VE POR DONE HHHH

sumcheck of
verified data

²VE POR DONE HHHH

sumcheck of
verified data

¹298 Display
²Terminal Display

Editing in SRC

Use the following procedure to edit data in the 298's RAM, using SRC operation. * Exit the SRC editor by pressing the ESCAPE key.

Procedure	Terminal Key Sequence	Terminal Display (2nd line)
1. Select the edit mode.	<input type="text" value="E"/> <input type="text" value="D"/> <input type="text" value="CR"/>	298 (1st line) EDIT ADDR ^ 00000 EDIT ADDR 00000>
2. Key in the hexadecimal address that you wish to edit.	<input type="text" value="X"/> <input type="text" value="X"/> <input type="text" value="X"/> <input type="text" value="X"/> <input type="text" value="X"/> <input type="text" value="CR"/>	XXXXXX DHH ^ RHH XXXXXX DHH, RHH>*
3. Key in the new data to be placed at the displayed address. Next address will then automatically be displayed.	<input type="text" value="H"/> <input type="text" value="H"/> <input type="text" value="CR"/>	ZZZZ DHH RHH ZZZZ DHH, RHH>*

NOTE

To decrement (address-1) addresses, press the / key.

* By-to device editing causes a display of this format on the terminal:

```
ADDR DEV RAM NEW RAM
XXXXX HHHH HHHH >
```

Select Functions in SRC

The special codes described in the select functions section of this manual can also be invoked while in SRC mode.

To display all the select functions available, press



The following table lists the select functions and the codes used to enable them, from both the 298 and terminal keyboards.

Hex (298) Code	SRC (terminal) Code	Title
A1	SW	Swap Nibbles
A2	F	Fill RAM
A3	IN	Invert RAM
A4	CL	Clear All RAM
A5	SP	Split RAM
A6	SH	Shuffle RAM
B0	DE	Device Size
B1	SU	Sumcheck RAM
B2	SY	System Config
B3	FO	Format Number
B4	NO	Nonblank Fail
B9	DI	Display Test
C1	CA	Calibration
D7	LE	Leader Output
D8	SI	Size Record
D9	NU	Null Count
F0	PR	Program Count
F1	RE	Remote Mode
F3	LO	Lock Data On
F4	NI	Nibble Mode
F5	BI	Binary Base
F6	OC	Octal Base
F7	HE	Hex Base
F8	BY	Byte/Nib Mode
F9	TI	Timeout Off
FA	CH	Char Output
FB	EN	Enable Port
FC	RE	Remote On Off

You may enable select functions in two ways using SRC: either by keying in the 2-digit SRC code (followed by a carriage return) or by first pressing "SE" and then the hex code. The following example shows how to clear the 298's RAM (hex code A4) using both methods.

STANDARD ENTRY:

Press

the 298 will then display **CLEAR ALL RAM **** and the terminal will display a prompt (>).

ABBREVIATED ENTRY:

Press

The 298 will then display **CLEAR ALL RAM **** and the terminal will display a prompt (>).

Data Translation Formats

Introduction

This subsection defines the data translation formats available for the 298. The 298 is capable of interfacing with all RS232 serial equipment employing a data translation format described in this subsection.

Each data translation format is assigned a 2-digit code which the operator enters into the programmer (from the keyboard or in remote control, through the serial port) to send or receive data in that format. In addition to the data translation format code, there is a 1-digit instrument control code which specifies control characters to be transmitted to, or received from, peripheral instruments. In several cases, the 298's standard display symbols will be shortened to accommodate large address fields used with some translation formats. These are:

- Copy RAM to Port FORMAT: HP 64000 Absolute (Format #89)
 DISPLAY: RAM > POR ^ ZZZZZZ
- Copy RAM to Port FORMAT: Motorola Exormax (Format #87)
 DISPLAY: RAM > POR ^ ZZZZZ
- Copy Port to RAM FORMAT: HP 64000 Absolute (Format #89)
 DISPLAY: P ^ ZZZZZZZZZZZZZ
- Copy Port to RAM FORMAT: Motorola Exormax (Format #87)
 DISPLAY: P ^ ZZZZZZZZZZZZZ
- Copy Port to RAM FORMAT: Extended Tek Hex (Format #94)
 DISPLAY: P ^ LO ADDR/ZZZZZ
 P ^ HI ADDR/ZZZZZ
- Pressing REVIEW to review I/O parameters
 FORMAT: HP 64000 Absolute* and Motorola Exormax
 DISPLAY: XXXXXXXXXXXZZZZZ

* In the HP format, ZZZZZZ represents the six least significant digits in the 8-digit address field. If either of the two most significant digits in the field is not zero, the display will show POR instead of the address. To view the address, reinitiate the key sequence for the input or output operation.

Data Verification

For data verification, the 298 calculates a sumcheck of all data sent to or from the programmer. At the end of a successful input operation, the programmer will display the sumcheck of all data transferred. It will also compare any received sumcheck fields with its own calculation. If the two agree, the programmer will display the sumcheck; a mismatch will produce an error message. Output data is always followed by a sumcheck field which may be printed on disk or tapes for use in subsequent input operations.

Description of Format Codes

Each format is assigned a 2-digit data translation format code which the operator enters to instruct the programmer as to which format to use. In addition to this code, a 4-digit instrument control code may be used to specify control characters for peripheral equipment. The codes must be formatted as follows: *xyx*, where "x" is the instrument control code, "yy" the format code. If no codes are entered into the programmer, the current default values will be in effect.

The following list shows the instrument control codes, with the corresponding 298 action.

Control Code	Programmer Action
0	Sends data immediately and continuously until acknowledging a "reader off" code. It will then stop sending data until receiving a "reader on" code. Sending no control codes results in normal, uninterrupted transmission.
1	Sends "reader on" (ASCII DC1/Hex 11) when ready to receive data, and "reader off" (ASCII DC3/Hex 13) when all data is received. Also send "punch on" (ASCII DC2/Hex 12) before sending data, and "punch off" (ASCII DC4/Hex 14) after sending data.
2	Sends data after acknowledging a "reader on" (ASCII DC1/Hex 11), and stops sending data after acknowledging a "reader off" (ASCII DC3/Hex 13).

Leader and Null Output

A leader is a string of characters that is attached to the beginning and end of a data file. It is used to separate different files from one another and allows extra room which may be necessary for loading and unloading the data medium to or from equipment. For the 298, the leader is sent at the beginning and end of a data output operation. With one exception, this leader will always be comprised of carriage return [CR], a line feed [LF], and 50 nulls in succession.

Null count is the number of null characters in a string of characters between each record or line within a file. What actually comprises a data record depends upon the format that is being used. Records and lines can basically be thought of as separations of data within a file.

Null count is a parameter which can be defined by the 298 user for use with printers with a slow carriage return response time. The number of nulls can be set to any value from zero to 254 decimal (FE hexadecimal). With one exception, the string of characters actually sent between each and every record or line of the file includes a carriage return [CR], a line feed [LF], and the number of nulls defined by the null count.

The exception referred to above for the leader and the null count occurs when the user defines the null count equal to the value of "FF" hexadecimal (or 255 decimal). In this case, the leader is made up of a solitary carriage return (no line feed and no nulls). Also, the string separating the records of the file is a carriage return (no line feeds and no nulls).

Parity for the beginning and end leader is the same as the parity for the data within the file. The same is true for the carriage return [CR], line feed [LF] and nulls separating the records or lines of the file. They have the same parity as the data. However, it should be noted that binary formats (10, 41 and 89) do not have parity for their data; therefore, the leader, the carriage return, line feed, and nulls separating the records for these files have no parity.

Translation Formats

This section gives information on the translation formats available for input and output by the 298 listed by code number in numerical order. The table below provides a quick reference of all the translation formats and corresponding codes.

Format	Code	Format	Code
ASCII-BNPF	01 (05)**	RCA Cosmroc	70
ASCII-BHLF	02 (06)*	Fairchild Fairbug	80
ASCII-B40F	03 (07)*	MOS Technology	81
5-Level BNPF	08 (09)*	Motorola Exoriser	82
Binary	10	Intel Intelac 81MDS	83
DEC Binary	11	Signetics Absolute Object	85
Spectrum	12 (13)**	Tektronix Hexadecimal	86
ASCII-Octal (Space)	30 (35)**	Motorola Exormox	87
ASCII-Octal (Percent)	31 (36)**	Intel MCS-86 Hexadecimal Object	88
ASCII-Octal (Apostrophe)	32	Hewlett-Packard 64000 Absolute	89
ASCII-Octal SMS	37	Texas Instruments SDSMAC	90
ASCII-Hex (Space)	50 (55)**	DCU Format	93
ASCII-Hex (Percent)	51 (56)**	Extended Tektronix Hexadecimal	94
ASCII-Hex (Apostrophe)	52		
ASCII-Hex (Comma)	53 (58)**		
ASCII-Hex SMS	57		

* For transmission of data without start codes, these alternate data translation format codes are used.

** For transmission of data with the SOH (Ctrl. A) start codes, these alternate data translation format codes are used.

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 N's and P's, L's and H's, or 1's and 0's, respectively. See the figure. The ASCII Binary formats do not have addresses.

The figure 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 normally separated by spaces. Data can also be expressed in 4-bit words. Any other characters, such as carriage returns or line feeds, may be inserted between an "F" and the next "B". The start codes are a nonprintable STX, control B (or hex 02), and the end code is a nonprintable ETX, control C (or a hex 03).

NOTE

Data without a start code may be input to or output from the programmer by use of alternate data transition format codes. These are: ASCII-BMPF, 05, ASCII-BHLF, 06, ASCII-BIOF, 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. The entire data transfer can be aborted by pressing any mode key (COPY, VERIFY, SELECT, EDIT).

Data is output in 4-byte lines with a space between bytes.

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 is punched on 5-hole Telex paper tapes to be read by an ASCII-based reader that has an adjustable tape guide. The reader reads the tape as it would on 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. 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

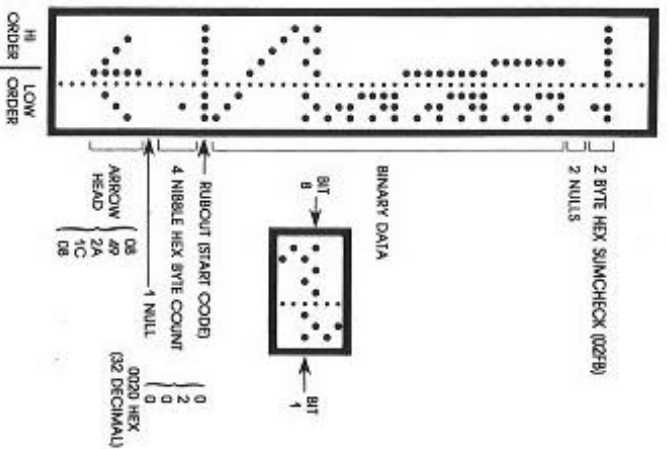
Data without a start code may be input to or output from the programmer by use of the alternate data transition format code, 09.

Binary Transfer, Code 10

Data transfer in the binary format consists of a stream of 8-bit data words preceded by a byte count and followed by a sumcheck. The binary format does not have addresses.

A paper tape generated by a programmer will contain 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 signalled by two nulls and a 2-byte sumcheck of the data field. Refer to the figure.

The programmer stores incoming binary data upon receipt of the start character. Data is stored in RAM starting at the first RAM address and ending at the last incoming data byte. Transmission may be aborted by pressing any mode key.



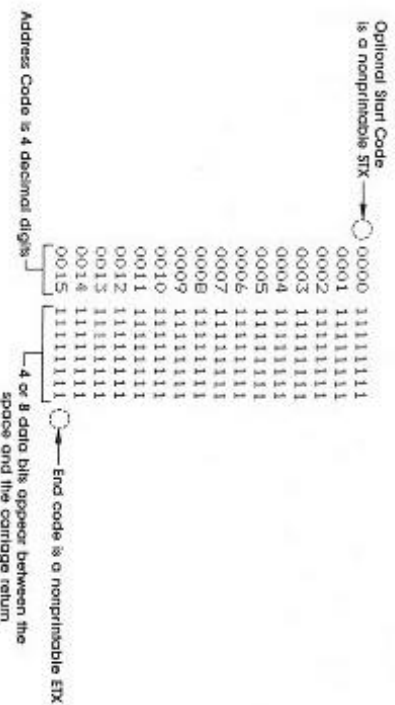
DEC Binary Format, Code 11

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. A tape output from the programmer will contain 32 rubouts in the leader. 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 1's and 0's. Each byte is preceded by an address.

The figure 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, control 8 (or hex 02), and the end code is a nonprintable ETX, control C (or hex 03).

**NOTE**

Data without a start code may be input to or output from the programmer by use of the alternate data transition format code, 13.

A single data byte can be aborted if the programmer receives an "E" character between a space and a carriage return. Data will continue to be stored in sequential RAM addresses. The entire data transfer can be aborted by pressing any mode key (COPY, VERIFY, SELECT or EDIT).

Data output to a printer will have one address and one byte of data on each line. The programmer first sends an STX (optionally), then the data, and finally an ETX.

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. The figure illustrates 4 data bytes coded in each of the 9 ASCII-Octal and Hex formats. Data in these formats is organized in sequential bytes separated by the execute character (space, percent, apostrophe, or comma). Characters immediately preceding the execute character are interpreted as data. ASCII-Hex and Octal formats can express 8-bit data, by 2 or 3 octal, or 1 or 2 hex 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.

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 "X", must contain 2 to 4 hex or 3 to 6 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.

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

NOTE

The checksum field consists of either 2-4 hex or 3-6 octal digits, sandwiched between "\$S" and "." characters. The checksum immediately follows an end code. It is optional in the input mode, but always included in the output mode. The most significant digit of the checksum may be 0 or 1 when expressing 16 bits as 6 octal characters.

Output is begun by invoking an output-to-port operation. The programmer divides the output data into 8-line blocks. Data transmission is begun with the start code, a nonprintable STX, optionally SOH.* Data blocks follow, each one prefaced by an address for the first data byte in the block. The end of transmission is signalled by the end code, a nonprintable ETX. Directly following the end code is a checksum 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 begins with a start record consisting of the start character (IM or ?M), an address field, and a space. See the figure.

The start character ?M is sent to the programmer only by a development system. This happens when the operator enters the interrogation ?M of a terminal (linked in parallel with the programmer to the development system), followed by the address in the development system memory where data transmission is to begin, followed by a number of bytes to be transferred, then by a carriage return.

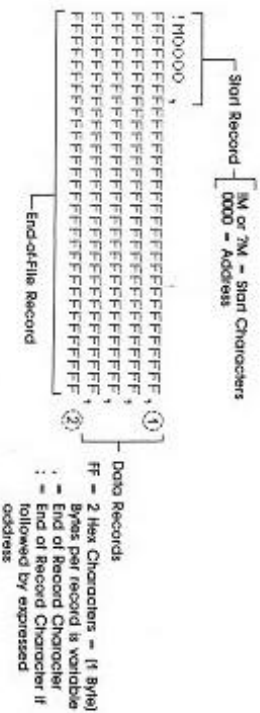
The development system responds by sending ?M to the programmer, followed by the starting address, and a data stream which conforms to the data input format described in the ASCII-Hex and Octal figure. Transmission stops when the specified number of bytes have been transmitted.

Address specification is required for only the first data byte in the transfer. An address must have 4 to 4 hex characters and be followed by a space. The programmer records the next hex 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. Any characters received between a comma or semicolon and a carriage return will be ignored by the programmer.

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.



LEGEND

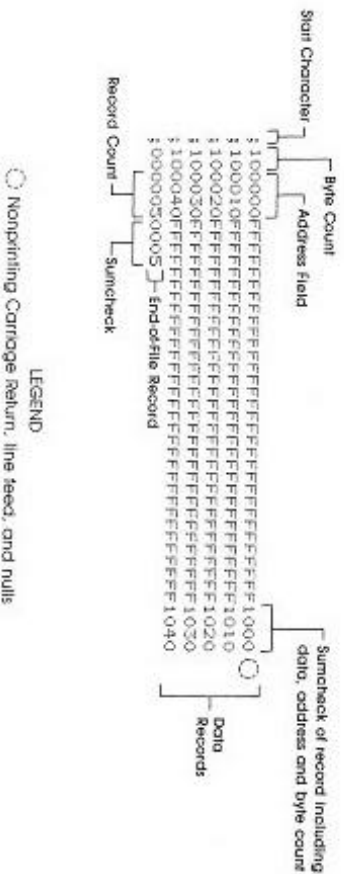
- ① Nonprinting line feed, carriage return, and nulls
- ② Nonprinting carriage return

MOS Technology Format, Code 81

The data in each record is 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.

The figure simulates 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 hex digits (0-9, 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.

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

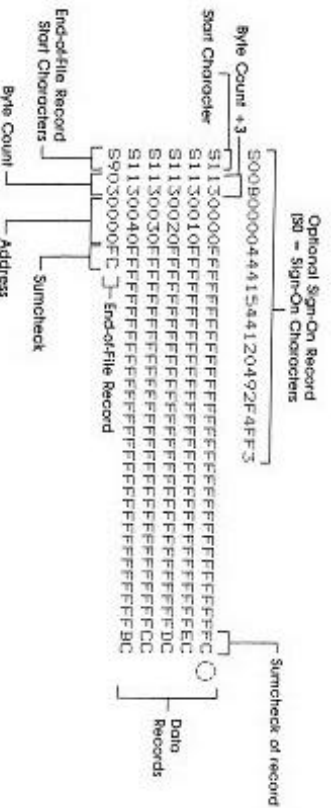


Motorola Exerciser Format, Code 82

Motorola Exerciser 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. The figure demonstrates a series of valid Motorola data records.

Each data record begins with the start characters "S1", the programmer will ignore all earlier characters. The third and fourth characters represent the byte count, which expresses the number of data, address and sumcheck 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 sumcheck, 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 (equal to "03"), the address (in hex) and a sumcheck.

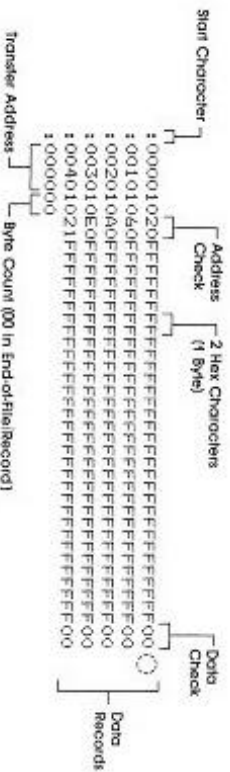


Signetics Absolute Object Format, Code 85

The figure shows the specifications of Signetics format files. The data in each record is sandwiched between a 9-character prefix and a 2-character suffix.

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 OR'ing 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 delta 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").

**LEGEND**

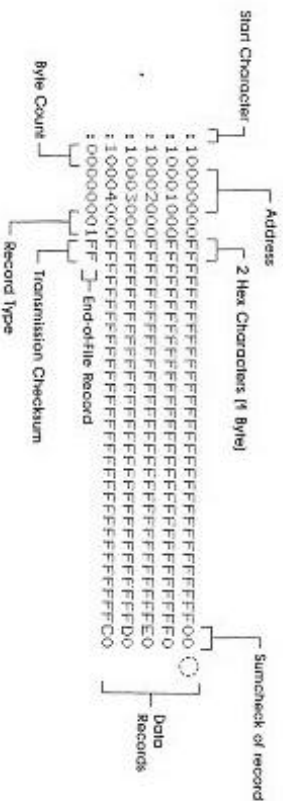
○ Nonprinting Carriage Return, line feed, and nulls

Intel Intellec 8MDS 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.

The figure 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 hex digits; the number of data bytes in each record must equal the byte count. Following the data bytes of each record is the sumcheck, the two's complement (in binary) of the preceding bytes (including the byte count, address and data bytes), expressed in hex.

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 sumcheck of the record.



LEGEND

○ Nonprinting Carriage Return, line feed, and nulls.

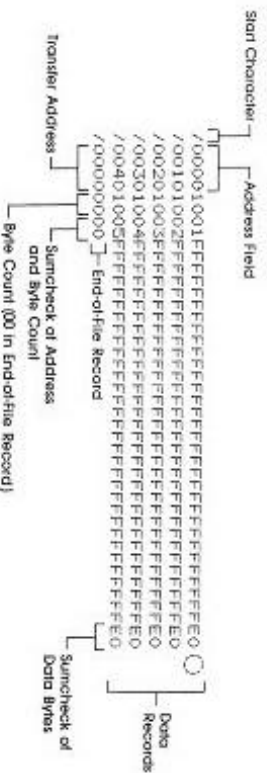
Tektronix Hexadecimal Format, Code 86

The figure illustrates a valid Tektronix data file. The data in each record is sandwiched between the start character (a slash) and a 2-character surcheck. 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 surcheck of the address and byte count. Data bytes follow, represented by pairs of hexadecimal characters. Succeeding the data bytes is their surcheck, an 8-bit sum, modulo 256, of the 4-bit hex values of the digits making up the data bytes. All records are followed by a carriage return.

Data is output from the programmer starting at the first RAM address and continuing until the number of bytes in the specified block have 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 surcheck of the transfer address and byte count.

An optional abort record contains 2 start characters (slashes), followed by an arbitrary string of ASCII characters.



LEGEND

○ Nonprinting Carriage Return, line feed, and nulls

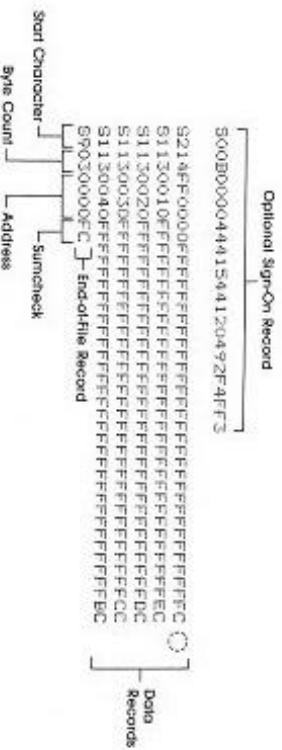
Motorola Exorimax Format, Code 87

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. The figure demonstrates a series of Motorola Exorimax data records.

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 hex FFFF). Data bytes follow, each represented by two 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 ones complement (in binary) of the preceding bytes in the record, including byte count, address and data bytes.

The end-of-file record begins with either an "S8" or "S9" start character. The start character must be "S9" if the previous data record started with an "S1"; otherwise, either "S8" or "S9" may be used.

Following the start characters are the byte count (equal to "03"), the address (equal to "0000") and a checksum.



LEGEND

○ Nonprinting Carriage Return, line feed, and nulls

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. The figure illustrates the sample records of this format.

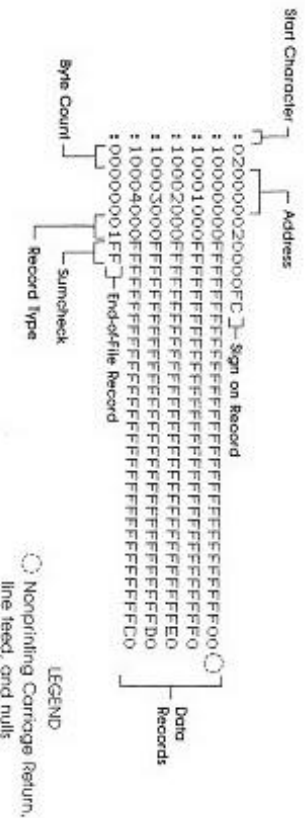
The four record types are:

- 00 = data record
- 01 = end record (signals end of file)
- 02 = extended address record (added to the offset to determine the absolute destination address)
- 03 = start record (ignored during input and not sent during output by Data I/O translator firmware)

Record type 00, data record, begins with the colon start character. This 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 twos complement (in binary) of the preceding bytes in the record, including byte count, address and data bytes.

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

Record type 02, the extended address record, defines bits 4 to 19 of the segment base address. It can appear randomly anywhere within the object file and in any order; i.e., it can be defined such that the data bytes at high addresses are sent before the bytes at lower addresses. The following example illustrates how the extended 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 FFBC

Solution:

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

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

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

offset address	1230	(upper 16 bits)
+ record address	0045	(lower 16 bits)
	<u>12345</u>	(20-bit address)

The address for the first data byte is therefore 12345.

NOTE

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

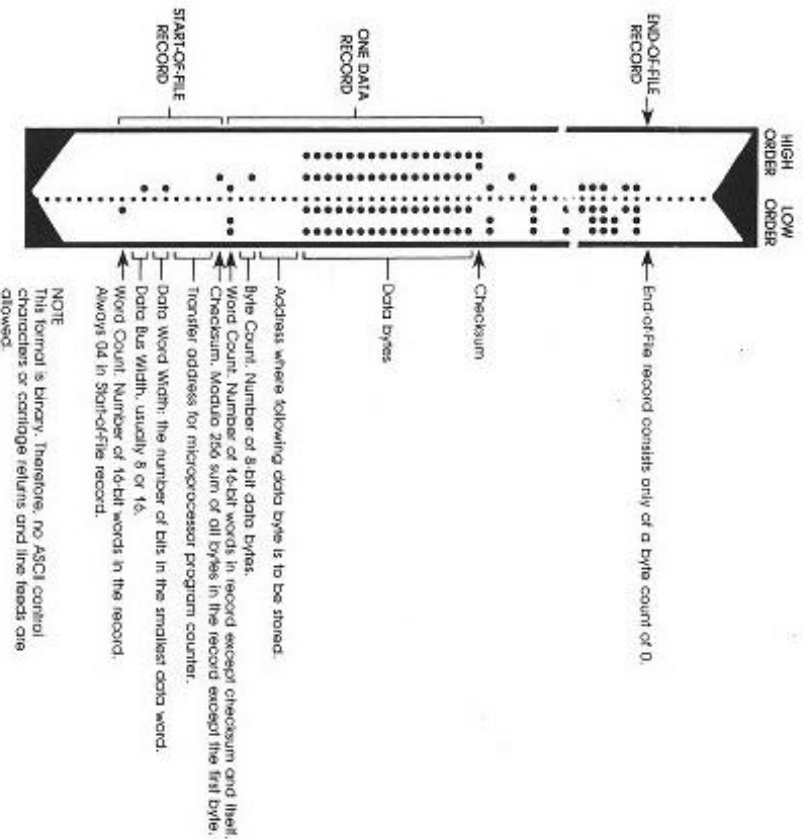
During output translation, the firmware will force the record size to 16 (decimal) if the record size is specified greater than 16. There is no such limitation for record sizes specified less than 16.

Hewlett-Packard 64000 Absolute Format, Code 89

Hewlett-Packard Absolute is a binary format with control and data-checking characters. See the figure. Data files begin with a Start-of-File record including the data bus width, data width base, transfer address, and a surcheck of the bytes in the record.

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

The end-of-file record consists only of a byte count, which is always zero.



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 the figure.

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 - always followed by a file header.
- 7 - always followed by a checksum which the programmer acknowledges.
- 8 - always followed by a checksum which the programmer ignores.
- 9 - always followed by a load address.
- B - always followed by 4 data characters.
- F - denotes the end of a data record.

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 16-bit 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 control S after the colon.

Data I/O DCU Format, Code 93

Remote Interface with Data I/O's Disk Control Unit (DCU) is possible using format 93. Access to disk data is made by keying in the part number and date for the stored data. Alphabetic keyboard entries are made possible using the 298's hexadecimal keys in conjunction with the SELECT key.

For example, during a Copy RAM to Port operation, the part number prompt, then the date prompt and finally the disk prompt would appear:

```
PNXXXXXXXXXXXXXXXXX  
DATEMMDDYY  
DISK, X
```

Hexadecimal values [0-9, A-F] for the part number date and disk are keyed in using the 298's keyboard. Letters G-Z and the dash (-) are keyed in by first pressing the "F" key, then the SELECT key, until the desired value is displayed. An "H" would be displayed by pressing "F SELECT SELECT". Press the REVIEW key to erase the current entry.

For more information, consult the DCU manual.

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. The header field contains the following information:

Item	Number 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	<ul style="list-style-type: none"> 0 = data record. 3 = symbol record 8 = termination record
Sumcheck	2	A 2-digit hex sum modulo 256 of all the values in the record except the % and the sumcheck itself.

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.

```

BLOCK LENGTH: 16H -21H
%1661C310D20202020202
SUMCHECK: 10H - 1+5+6+3+1+0+0+0+2+0+2+-
OBJECT CODE: 6 BYTES
HEADER CHARACTER
LOAD ADDRESS: 100 H
BLOCK TYPE: 6 (DATA)

```

When you are copying data to the port or to RAM, make sure to set the high-order address if the low-order is not at default value. Non-default addresses are entered as follows in a Copy Port to RAM operation.

Press . The 298 will display *P_N LO ADDR/SIZE*. Key in the low 8-digits of the address, and press . Next, key in the high 8-digits, and press .

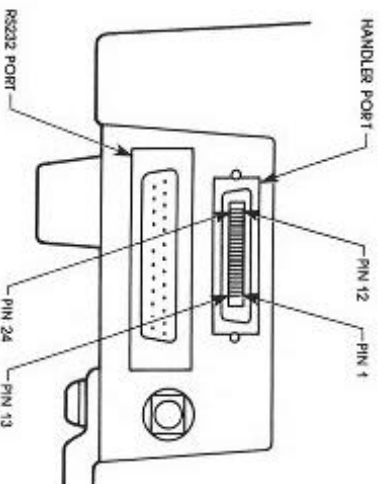
298B/Handler Interface

Introduction

This subsection describes how to operate the 298 with a commercially available handler. For more detailed information, consult your handler manual.

Compatibility

The 298 is capable of interfacing with two types of handlers: those that connect to the 298 via the handler port (see figure), and those that interface via the RS232 port.



Handler CRC Commands

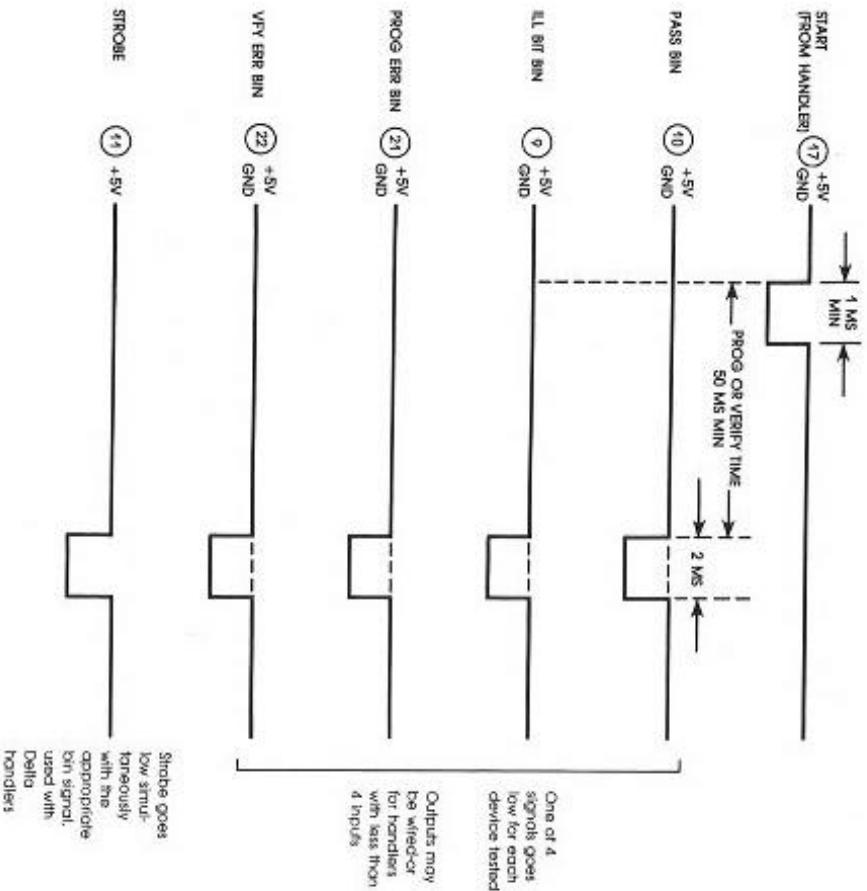
Two CRC commands, "%" and "H," are specific to operation with a handler. The "%" command puts the 298 into a waiting state. When either (1) the handler indicates a start condition or (2) the 298's START key is pressed, the programmer will output a prompt ">" symbol to the serial port. The "H" command, used for binning control, tells the 298 to distribute programmed and verified devices to the "H" bin (1 thru 4).

298 Handler Port Information

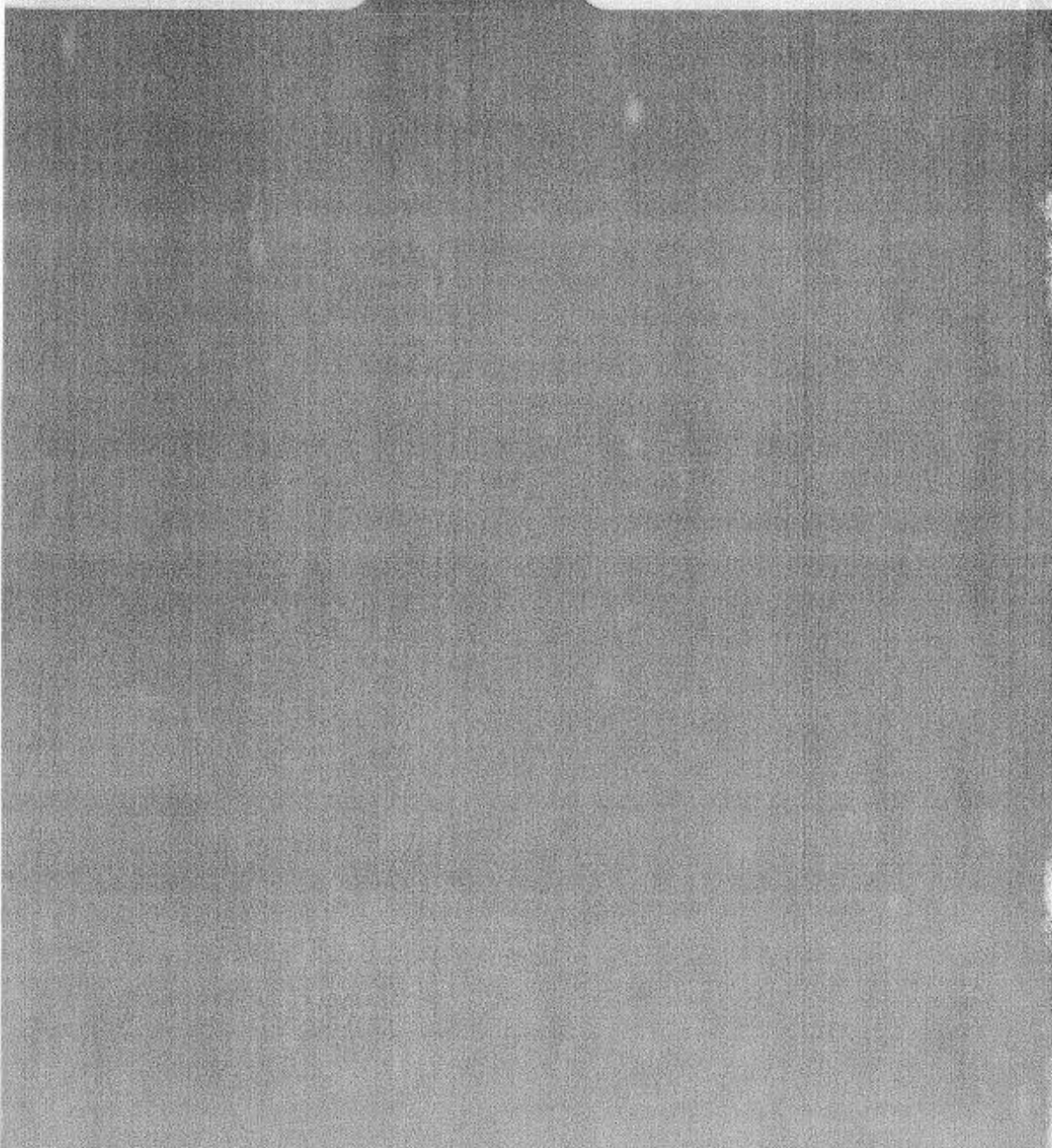
The following information is provided for 298 users who are making their own interface. The figure shows the port pin numbers; the table below defines the function of each.

The next page illustrates the handler control timing. To prevent device oscillation or other errors, it is recommended that the cable connecting the handler device contacts to the programming box socket be shielded and of minimal length.

Pin No.	Signal Mnemonic	Description
1,2	Ground	
3,4,5	+5V	Connected to the programmer's +5V logic supply. Available for external use if required (200mA max.).
6,7,8	Ground	
9	Illegal Bit Bin	Signal lines for selecting the appropriate handler bin. These open-collector outputs go to ground to 2 milliseconds when selected by the programmer.
10	Pass Bin	
11	Strobe	Signal used for handlers that need a strobe as well as a binning signal. This open-collector output goes to ground coincidentally with any of the four binning signals.
12-16	Ground	
17	Start	This input is internally pulled to +5V and should be grounded to inform the programmer of when a device is in place in the handler. This signal should not go high again until 1 millisecond after the previous binning signal goes high.
18, 19, 20	ID1, ID0, ID2	These inputs inform the programmer what is connected to the port. If pins 18, 19, 20 are all high, the 298 identifies the port as being not connected. If pin 19 is low, and 18 and 20 are high, the port is identified as being connected to a handler.
21	Programming Error Bin	Signal lines for selecting the appropriate handler bin. These open-collector outputs go to ground for 2 milliseconds when selected by the programmer.
22	Verify Error Bin	
23	Spare Bin	
24	Ground	For future use.



Select Functions



Select Functions

Introduction

The 298 offers special select functions that allow you to select several operating modes including: RAM data manipulation, utility and inquiry commands and serial I/O commands. The SELECT key plus two-character hexadecimal codes which are entered at the front panel keyboard enable the functions. The programmer signals that the select function has been performed by displaying two asterisks (**). In the last two display positions. Following is a complete list of the 298 select functions. The pages that follow describe these commands. See your programming Pdk manual for a list of select functions specific to the Pdk.

Select Functions

Select Function List

Command Group	Hex Code	Title
Format Codes		
	01-99	Data Transition Format Codes
RAM Data Manipulation		
	A1	Swap Nibbles
	A2	Fill RAM
	A3	Invert RAM
	A4	Clear All RAM
	A5	Split RAM
	A6	Shuffle RAM
Utility and Inquiry		
	B0	Device Size
	B1	Sumcheck RAM
	B2	System Configuration
	B4	Nonblank Fail
	B9	Display Test
	C1	Calibration
	F0	Program Count
	F3	Lock Data On
	F4	Nibble Mode
	F5	Binary Base
	F6	Octal Base
	F7	Hex Base
	F8	Byte/Nibble Mode
Serial I/O		
	83	Format Number
	D7	Leader Output
	D8	Size Record
	D9	Null Count
	F1	Remote Mode CRC
	F9	Timeout Off
	FA	Character Output
	FB	Enable Port
	FC	Remote On Off

Accessing Select Functions

The select functions may be accessed by either direct entry, stepping or scrolling.

- For direct entry, press SELECT. The 298 will display "SELECT CODE". Enter the hex code for the desired function or data translation format and then press START. (Some functions require the START key to be pressed twice.) The display will prompt any additional entries required, or indicate an invalid entry.
- To access the select functions by stepping, press the SELECT key repeatedly until the function desired is displayed. Then press the START key to initiate the operation. Press the REVIEW key to step backwards through the select functions. The functions are displayed in hexadecimal order.
- To access the select functions by scrolling, press SELECT and then START. Each function is momentarily displayed in turn. When the desired function is displayed press any key to stop the scrolling. To back up, press REVIEW. Once the desired select function is displayed, initiate the operation by pressing START.

The following pages list the key sequences required to perform each of the 298's select functions. Most of the functions may be performed by keying in



(where "HH" is the two-digit select function). Functions that may be keyed in using this sequence are grouped together; those requiring entry of additional data are listed on the following pages. The select functions are organized according to command group: Format Codes, RAM Data Manipulation, Utility and Inquiry and Serial I/O.

Format Codes

Data translation formats allow the 298 to send and receive data to and from other systems. You may key in directly the two-digit code for the data translation format you are using. A complete list of formats and their corresponding codes appears in the remote control section. To key in format codes directly, press

select	C	H	H	EXIT	SAVE
--------	---	---	---	------	------

(where "HH" is the format code and "C" is the optional instrument control code). The instrument control code specifies control characters used when sending or receiving data. For example, to select the binary (code 10) data translation format, press

select	0	1	0	EXIT	SAVE
--------	---	---	---	------	------

The 298 will then display **FORMAT NO 010 ****

NOTE

Refer to select code B3 for alternate key sequence to enable a format. Code B3 allows you to scroll through the list of format codes.

RAM Data Manipulation

The select codes listed in the following table are all executed using the following key sequence:



(where "HH" is the two-digit code given in the Hex Code column of the table). Three others (fill, split and shuffle RAM) require additional data entry and are listed on the following pages.

Hex Code	Command Title	Description
A1	Swap Nibbles	Exchanges high and low order nibbles of every byte. Used when manipulating 4-bit word data.
A3	Invert RAM	Performs the ones complement of 4 or 8 bits of each word (determined by the word size in effect). For example, changes all "1's" to "0's".
A4	Clear all RAM	Clear all RAM to zeroes. Useful before downloading port data to RAM.

A2 Fill RAM

The fill RAM function fills the 298's RAM with a specified pattern, from the edit address to the end of RAM. Use the following procedure to fill RAM from the last EDIT address to the end of RAM with variable hex data. The default value is 00. If select function F4 (nibble mode) has been specified, or if a 4-bit device is selected, it will fill only the lower order nibbles of RAM; otherwise, it will default to the word width of the selected device.

Procedure	Keystroke	298 Displays
1. Select the Fill RAM command operation.	<input type="button" value="select"/> <input type="button" value="A"/> <input type="button" value="2"/> <input type="button" value="start"/>	SELECT CODE ^ A2 FILL RAM ^ 00 A2
2. Enter the hex data to be placed in RAM.	<input type="button" value="H"/> <input type="button" value="H"/> <input type="button" value="start"/>	FILL RAM ^ HH A2
3. Write the hex data to RAM.	<input type="button" value="start"/>	FILL RAM ^ HH (action symbol rotates) FILL RAM

**

A5 Split RAM

The split RAM function is useful when working with 16-bit data. Split RAM is the inverse of the shuffle RAM operation. Use the following procedure to split odd- and even-addressed bytes in RAM about a center point, dividing them into two adjacent blocks occupying the same original amount of RAM. The center point must be a power of two between 0 and the RAM midpoint. The default center point is the midpoint of RAM.

Procedure

Keystroke

298 Displays

- Select the Split RAM operation.

SELECT	A	S
SAVE		

H	H	H	H	H	SAVE
---	---	---	---	---	------

SELECT CODE	^	A5	
SPLIT RAM	^	HHHHH	
SPLIT RAM			<input checked="" type="checkbox"/>
SPLIT RAM			**
- Set the hex midpoint (if the default or displayed value is correct, press START.)

NOTE

The execution time of this function is dependent on the size of the midpoint. The default setting requires the greatest amount of time.

A6 Shuffle RAM

As with the split RAM function, the shuffle RAM function is useful when working with 16-bit data. Data may be split, then transferred to two PPOMs. Use the following procedure to shuffle the block of RAM addresses immediately above the center point with the block below, placing the lower-block bytes at even-numbered addresses starting with 0 and the upper-block addresses of odd-numbered addresses starting with 1. The center point must be a power of two between 0 and the RAM midpoint. The default center point is the total RAM midpoint.

Procedure**Keystroke****298 Displays**

1. Select the Shuffle RAM command operation.



SELECT CODE ^ A6
SHUFFLE ^ HHHHHH

2. Set the hex midpoint (if the default or displayed value is correct, press SHUFF)



SHUFFLE
SHUFFLE **

NOTE

The execution time of this function is dependent on the size of the midpoint. The default setting requires the greatest amount of time.

Utility and Inquiry Commands

Utility and inquiry select functions allow you to access and/or display parameters such as the RAM sumcheck, device size or serial port word size. The select functions listed in the following table are all executed using the following key sequence:



(where "H" is the two digit code given in the Hex Code column of the table). The system configuration and data lock command (B2 and F3) require additional data be keyed in, and are listed on the following pages.

Hex Code	Command Title	Description
B0	Device Size	Displays the device size and word width, currently entered. A display of "00800 4" indicates a by-4, 800 hex address device.
B1	Sumcheck RAM	Displays the total RAM sumcheck.
B4	Nonblank Fail (handler option only)	Automatically fails nonblank devices when the 298 is equipped with the optional handler. The default condition is to attempt to program any nonblank devices detected; if the optional handler is not connected, the command has no effect on the system.
B9	Display Test	Illuminates all segments of the programmer's display for approximately 4 seconds.
F0	Program Count	After START is pressed, displays the number of devices programmed since the last power-up or reset. The program counter is reset to 00 after START is pressed again.
F4	Nibble Mode	Selects a 4-bit word size for I/O transfers, regardless of word size defined by default or the family/pairout code. The F8 select command returns to 8-bit word size.
F5	Binary Base*	Sets the number base for edit operations to binary.
F6	Octal Base*	Sets the number base for edit operations to octal.
F7	Hex Base*	Sets the number base for edit operations to hexadecimal.
F8	Byte/Nibble Mode	Nullifies F4, allowing word size defined by the selected device to take effect.

*By-16 devices may only be edited in hexadecimal.

B2 System Configuration

Use the following procedure to display the four-character system configuration code, used when contacting Data I/O Technical Support Personnel for software version determination.

Procedure

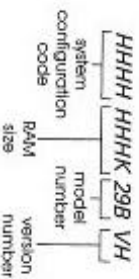
Keystroke

298 Displays

1. Select the System Configuration command.

SELECT CODE ^ B2

2. Display the System Configuration.



F3 Data Lock On

When the Data Lock On function is selected, the data in RAM is protected because the keys used to alter data are disabled. This allows an operator to program or verify without accidentally altering data. Only the functions listed below are permitted when the lock Data On function is selected:

- Copy operations that move data from RAM to the port or device
- Verify operations
- Abort the operation in progress
- Release data lock

Use the following procedure to select the Lock Data On function:

Procedure	Keystroke	298 Displays
1. Select the Data Lock On command.	<input type="button" value="SELECT"/> F <input type="button" value="3"/>	<i>SELECT CODE ^ F3</i>
2. Enable the Data Lock On command.	<input type="button" value="ENTER"/>	<i>LOCK DATA ON **</i>

The Data Lock On feature is now enabled. RAM altering operations cannot be performed until the Data Lock On function is released. Use the following procedure to release the Data lock On command:

Procedure	Keystroke	298 Displays
1. Select the release data lock operation.	<input type="button" value="SELECT"/> <input type="button" value="PASS"/>	<i>PASSWORD ?</i>
2. Release data lock. RAM data may now be altered.	<input type="button" value="ENTER"/>	<i>SELECT CODE ^</i>

Serial I/O Commands

Serial I/O select functions set parameters that are used in serial port operations; for example, to enable CRC and SRC mode.

The select codes listed in the following table are all executed using the following key sequence:



(where "HH" is the two-digit select code given in the Hex Code column of the table). I/O functions requiring additional data entry are listed on the following pages.

Hex Code	Command Title	Description
D7	Leader Output	Sends 50 nulls from the serial port, allowing separation of files.
F9	Timeout Off	Disables the 25-second timeout, active unless disabled during I/O operations. 25 seconds is the maximum time the 298 will wait when receiving or sending. Once the timeout has been disabled, the only way it can be enabled is to turn the instrument power off, and then reapply power.
FB	Enable Port (SRC enable)	Enables System Remote Control and Input Interrupt. This also forces the RTS line high at all times for remote control from peripherals requiring a hardware handshake. The default of power up is RTS low and System Remote Control and serial interrupts disabled.
F1	Computer Remote Control Mode	Enables Computer Remote Control (CRC) mode.

B3 Format Number (Direct Selection)

The data translation format can be selected in any of three ways. If the desired format number is known, the select mode can be accessed and the code entered directly. The indirect methods allow the operator to display the current or default entry and then change the entry by either stepping through the formats, or entering the format code.

Use the following procedure to directly enter a new data translation format number:

Procedure**Keystroke****298 Displays**

1. Select instrument control code if desired, and the format number (the first "N" represents the optional instrument control code, default 0 if not entered, the second and third numbers represent the format code).

select	N	N	N
--------	---	---	---

FORMAT NO \ ONNN

2. Display a mnemonic for the selected format, and enable that format:

swt	swt
-----	-----

XXXXXXXXXXXXXXXXXXHH
 |-----|
 selected format format number
 Instrument control code

FORMAT NO NNN **

B3 Format Number (Continued)

Use the following procedure to display and change the current data translation format number by scrolling through the list of available formats:

- | Procedure | Keystroke | 298 Displays |
|---|---|---|
| 1. Select the format number command. | <input type="button" value="SELECT"/> <input type="button" value="8"/> <input type="button" value="3"/> | <i>SELECT CODE</i> \wedge <i>B3</i> |
| 2. Display the mnemonic for the current or default format. If the displayed format is the desired format, press START . If it is not the desired format, continue with step 3. | <input type="button" value="START"/> | <pre> XXXXXXXXXXXXXXXXXXYYHH ----- current format ----- ----- instrument control code </pre> |
| 3. Press and release the SELECT key to step forward through the formats until the desired format appears on the keyboard display. Press the REVIEW key to step backwards through the formats to the desired format. | <input type="button" value="SELECT"/> | <pre> XXXXXXXXXXXXXXXXXXYYHH ----- selected format ----- ----- instrument control code </pre> |
| 4. When the displayed format is the one you want, press START . | <input type="button" value="START"/> | <i>FORMAT NO</i> <i>NNN</i> ** |

B3 Format Number (Continued)

Use the following procedure to display and change the current data transition format number:

Procedure**Keystroke****298 Displays**

1. Select the format number command.



SELECT CODE \ B3



2. Display the mnemonic for the current or default format. If the displayed format is the desired format press **START**. If it is not the desired format, continue with step 3.

XXXXXXXXXXXXXXXXXXXXYHH
 |-----|
 current format
 |-----|
 format number
 |-----|
 Instrument control code



FORMAT NO \ ONWN

3. Select the format number (the first "N" represents the optional Instrument control code, default 0 if not entered, the second and third numbers represent the format code).



XXXXXXXXXXXXXXXXXXXXYHH
 |-----|
 selected format
 |-----|
 format number
 |-----|
 Instrument control code

FORMAT NO MNW

**

4. Display a mnemonic for the selected format, and enable that format.

D8 Record Size

Standard (default) record size is 10 hexadecimal digits per line. If you wish to change this value, use this select code to change the number of characters sent before the next address.

Use the following procedure to change the number of bytes per data record on the serial output (the value entered must be in hexadecimal):

Procedure	Keystroke	298 Displays
1. Select the record size command. The default entry in the hex field is 10 (16 decimal).	<div style="display: flex; align-items: center; gap: 5px;"> <div style="border: 1px solid black; padding: 2px;">select</div> <div style="border: 1px solid black; padding: 2px;">0</div> <div style="border: 1px solid black; padding: 2px;">8</div> </div> <div style="display: flex; align-items: center; margin-top: 5px;"> <div style="border: 1px solid black; padding: 2px;">start</div> </div>	SELECT CODE ^ D8 SIZE REC ^ 10 D8
2. Set the record size (if the default value is correct, press START).	<div style="display: flex; align-items: center; gap: 5px;"> <div style="border: 1px solid black; padding: 2px;">H</div> <div style="border: 1px solid black; padding: 2px;">H</div> </div> <div style="display: flex; align-items: center; margin-top: 5px;"> <div style="border: 1px solid black; padding: 2px;">start</div> </div>	SIZE REC ^ HH SIZE RECORD D8 **

D9 Null Count



Use the following procedure to set the null count following each data record (carriage return and line feed) on the output. The hex entry selects from 0 (00) to 254 (FE) nulls. An entry of 255 (FF) sends no nulls, suppresses the line feed, and separates records by using a carriage return. The default entry on power-up is 1 (01) null.

Procedure**Keystroke****298 Displays**

- | | | |
|--|---|---|
| 1. Select the null count function. | <input type="button" value="select"/> <input type="button" value="D"/> <input type="button" value="9"/> | <i>SELECT CODE</i> \setminus <i>D9</i> |
| | <input type="button" value="start"/> | <i>NULL COUNT</i> \setminus <i>HH</i> <i>D9</i> |
| 2. Set the null count (default is 01) and execute the operation. | <input type="button" value="H"/> <input type="button" value="H"/> | <i>NULL COUNT</i> \setminus <i>HH</i> <i>D9</i> |
| | <input type="button" value="start"/> | <i>NULL COUNT</i> \setminus <i>HH</i> <i>D9</i> |

FA Char Output

One way of testing the serial port is to send a single ASCII character through, and then verify that it was transmitted. Use the following procedure to enter the hex code for an ASCII character which will be transmitted to the port each time START is pressed (this function is inhibited in system remote control):

Procedure	Keystroke	298 Displays
1. Select the character output command.		SELECT CODE ^ FA CHAR OUTPUT ^ 00
2. Enter the hex code for the ASCII character to be transmitted.		CHAR OUTPUT ^ HH

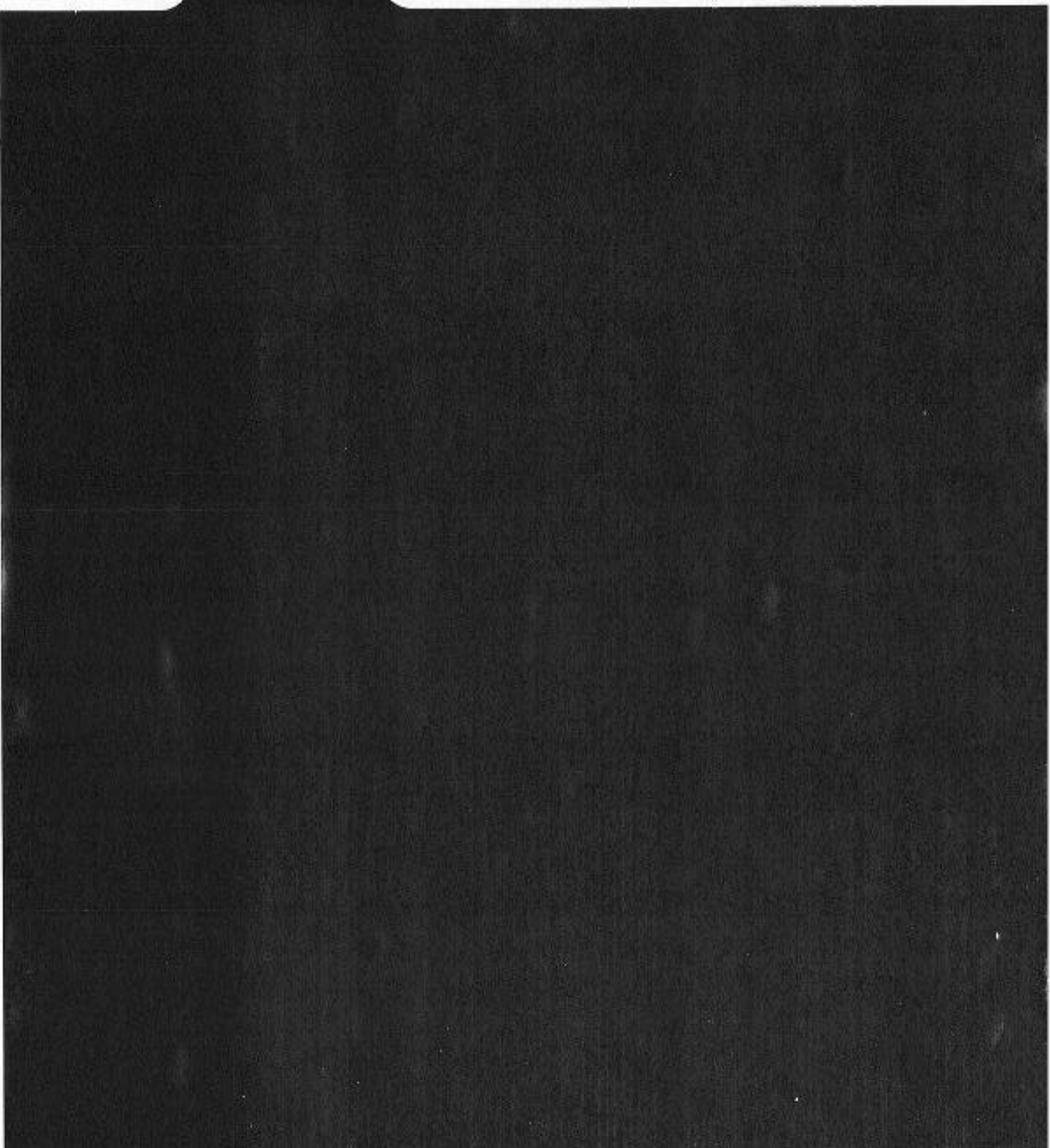
- To retransmit the same character, press the START key again. To transmit a different character, perform step 2 again.

FC Remote On/Off

If you want to use ASCII characters to enable or disable the port remotely, use select code FC. Use the following procedure to enter hex codes for ASCII characters that can be used to turn remote control on or off (may be used with CRC or STC).

Procedure	Keystroke	298 Displays
1. Select the remote on or off command.	<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="border: 1px solid black; padding: 2px;">SELECT</div> <div style="border: 1px solid black; padding: 2px;">F</div> <div style="border: 1px solid black; padding: 2px;">D</div> </div> <div style="margin-top: 5px; text-align: center;">START</div>	SELECT CODE ^ FC RMT ON OFF ^ HH HH
2. Select and enter the hex ON code.	<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="border: 1px solid black; padding: 2px;">H</div> <div style="border: 1px solid black; padding: 2px;">H</div> </div> <div style="margin-top: 5px; text-align: center;">START</div>	RMT ON OFF ^ HH HH
3. Select and enter the hex OFF code.	<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="border: 1px solid black; padding: 2px;">H</div> <div style="border: 1px solid black; padding: 2px;">H</div> </div> <div style="margin-top: 5px; text-align: center;">START</div>	RMT ON OFF **

Error Codes



Error Codes

NOTE

If you get a recurring error, call your local customer support center listed on the back of this manual.

Code	Name	Description	Corrective Action
15	SORC/DEST ERR	Illegal source/destination key sequence was entered.	Check key sequence and re-enter.
17	COMMAND ERROR	Illegal key sequence while in System Remote Control.	Check key sequence and re-enter.
18	ILL RAM PAGE	The operation would require a 16K RAM page change, which the present Pak software does not support.	Either modify Begin RAM or block size so that the first and last bytes of the operation reside in the same 16K range (0-3FFF, 4000-7FFF, 8000-BFFF, C000-FFFF), or contact your local Data I/O Service Center for a Pak software update.
20	NONBLANK	Device failed the blank test.	Press START to override this error and program the device.
21	ILLEGAL BIT	Not possible to program the device due to already programmed locations of incorrect polarity.	Erase the device if possible or discard it.
22	PROGRAM FAIL	The program electronics were unable to program the device.	Either the device is bad or the programming module is inoperative or out of calibration.

Code	Name	Description	Corrective Action
23	VERIFY FAIL 1	The device data was incorrect on the first pass of the automatic verify sequence during device programming.	This error indicates that the device failed the low voltage verify; the data in the part is not the same as the RAM data.
24	VERIFY FAIL 2	The device data was incorrect on the second pass of automatic verify sequence during programming.	This error indicates that the device failed the high voltage verify; data in the part is not the same as the RAM data.
25	NO PROG PAK	A device-related operation was attempted without any programming module installed.	Install the appropriate programming module.
26	PROG PAK RST	The programming electronics will not start operation due to a reset condition.	Usually an overcurrent caused by an incorrectly inserted or bad device in the socket.
27	RAM EXCEEDED	There is insufficient RAM. The total allotment of RAM resident is less than the word limit or block size, or the begin address is set too high.	Program smaller parts or specific lower beginning address. If enough RAM is installed, it may be faulty.
30-39	ERRORS	These are PAK-related errors.	Refer to your programming PAK manual.
41	FRAME ERR	The serial interface detected a start bit but the stop bit was incorrectly positioned.	Check the baud rate and stop bit switches or use handshake.
42	OVERRUN ERR	The serial interface received characters when the programmer was unable to service them.	Check the baud rate and stop bit switches or use hardware handshake. See the remote control section for instructions.
43	FRAME+OVR ERR	Combination of FRAME ERR 41 and OVERRUN ERR 42.	Check the baud rate and stop bit switches or use hardware handshake. See the remote control section for instructions.

Error Codes

Code	Name	Description	Corrective Action
46	I/O TIMEOUT	No character (or only nulls and rubouts) were received on serial input for 25 seconds after pressing the START key, or no characters could be transmitted for a period of 25 seconds due to the state of the handshake lines.	Check all connections; then restart the operation. I/O timeout can be disabled by select code P9, which will then allow more than 25 seconds for serial port inputs.
47	FAULTY ACIA	ACIA chip may have failed.	Contact your local Data I/O Service Center.
48	I/O OVERRUN	The serial port input buffer received too many characters after the handshake line informed the sending device to stop.	Make sure the handshake lines are hooked up and operative.
52	I/O VFY FAIL	The data from the serial port did not match the data in RAM.	Check and resend the data.
61	NO RAM	RAM error during self-test; first page of RAM.	Power off and then on again. If the error reoccurs, call your local Data I/O Service Center.
62	RAM BIT ERR	The highest RAM address in the programmer is not on a 1K boundary.	Replace faulty RAM or have the programmer serviced by your local Data I/O Service Center.
63	RAM WRITE ERR	The programmer is unable to write the intended data in RAM.	Failure of the associated RAM chip; replace the failed chip.
64	RAM DATA ERR	The programmer detected a spurious change in RAM data.	Reload data into RAM. If problem persists, service the programmer or notify your local Data I/O Service Center.
65	ERROR	The checksum of software residing in the installed programming module is in error.	Contact your local Data I/O Service Center.

Code	Name	Description	Corrective Action
66	IRQ ERR	The IRQ line to the processor was held low for no apparent reason.	Ignore. If the error persists, service the programmer.
67*	ERROR	Programmer received a non-valid command in Computer Remote Control (CRC).	Re-enter the command.
68	DATA LOCKED	Data locked via Select Function F3.	Use the password to release data lock.
69	RAM BANK ERR	RAM bank error	The address size is out of range for the Pak installed; reduce the block size to 4000. Refer to your programming Pak manual.
70-79	ERRORS	These are Pak-related errors. Refer to your programming Pak manual.	Refer to your programming Pak manual.
81	PARITY ERR	The incoming data has incorrect parity.	Check the parity switch and try again.
82	SUMCHK ERR	The checksum field received by the programmer does not agree with its own calculated checksum. For ASCII Binary formats, this error message indicates a missing F character.	Check all connections of units in the system, data format, and data source, and then try again.
83, 85-87	COMPOSITE ERRORS	A composite error occurs from any combination of errors 81, 82 and 84. These combinations are: Error 83 = errors 81 and 82 Error 85 = errors 81 and 84 Error 86 = errors 82 and 84 Error 87 = errors 81, 82 and 84	

*Remote Control only: will not occur during front panel operation, hence no front panel display.

Error Codes

Code	Name	Description	Corrective Action
84	INVALID DATA	The programmer received invalid or not enough data characters. Non-data characters (formats 4-3, 5-9, 12-13) Non-hex characters (formats 70, 81, 88, 90).	Check the connection of all units in the system, data format and data source, and then try again.
90	INVALID FORM	Non-existent I/O format is selected in Computer Remote Control (CRC).	Enter a legal format code.
91	I/O FORM ERR	The programmer received an invalid character in the address field.	Check the connection of all units in the system, data format, and data source, and then try again.
92	I/O FORM ERR	The address check was in error (Signetics Absolute Object and Tektronix Hexadecimal formats only).	Check the connection of units in the system, data format, and data source, and then try again.
93	I/O FORM ERR	The number of input records did not equal the Record Count (MOS Technology format only).	Check the connection of all units in the system, data format, and data source, and then try again.
94	BAD REC TYPE	The record type was in error. (Intel/Intellic 8/MDS, Intel MCS-86 and TL SDSMAC formats only.)	Check the connection of all units in the system, data format, and data source, and then try again.
95	FMAT EXCEEDED	Blocksize exceeds format	Reset the block size of the data being transferred; check programming section for specific key sequence. Maximum size for 64K addressable translators is 10000.

Code	Name	Description	Corrective Action
96*	ERROR	Illegal center point for RAM shuffle.	Select a new RAM centerpoint.
97	BLOCK MOVE ERR	Block Move was attempted outside RAM boundaries.	Select new RAM boundaries.
98	DEV EXCEEDED	Programming data exceeded the last device address.	Select new RAM boundaries.
A0-A9; B0-B9	ERRORS	These are Pak-related errors. Refer to your programming Pak manual.	Refer to your programming Pak manual.
YY XXXXX	DATA RAM ERR USABLE RAM	RAM error during self-test, outside first page of RAM.	YY represents the failed data; XXXXX, the amount of data RAM still usable. You may still perform operations; but the addressable RAM will depend on the number of address locations that are good. For example, if XXXXX=04000, the last address you could use would be 03FFF. The programmer will not allow access to bad RAM address locations.

* Remote Control only; will not occur during front panel operation, hence no front panel display.

