2

Front Panel Operation

2. FRONT PANEL OPERATION

This section describes how to blank check, program and verify parts using the front panel keys and display of the Model 212 Programmer. Also provided are instructions for using the parallel port and the RS-232C serial interface port in conjunction with the front panel keys, i.e., setting the RS-232C communications protocol and performing download and upload operations using the RS-232 and parallel ports. The information in this section is divided into the following subsections:

□ GENERAL OPERATING NOTES — Provides general information pertaining to performance of many of the 212 operations and functions. Read this section before attempting any 212 operations.

□ CHECKING FOR NON-BLANK DEVICES — Describes how to verify that a device is completely blank (contains no data).

□ PROGRAMMING FROM A MASTER — Describes load and program operations required to program devices.

□ VERIFYING PROGRAMMED PARTS — Describes how to verify the data programmed into a device against the contents of the programmer's memory.

□ SETTING COMMUNICATIONS PROTOCOL — Describes how to set the RS-232C communications protocol, such as the baud rate and the number of stop bits, for the 212.

□ DOWNLOADING DATA using the **RS-232 port** — Describes how to prepare the programmer to receive and store data transferred from a remote data source (computer or development system) to the programmer's RAM.

□ UPLOADING DATA using the **RS-232 port** — Describes how to perform the transfer of data from the programmer's memory to a host computer or other development system.

□ DOWNLOADING DATA using the **parallel port** — Describes how to prepare the programmer to receive and store data transferred from a remote data source (computer or development system) to the programmer's RAM.

□ UPLOADING DATA using the **parallel port** — Describes how to perform the transfer of data from the programmer's memory to a host computer or other development system.

□ EDITING RAM DATA — Describes how to edit the contents of the programmer's data memory (RAM).

GENERAL OPERATING NOTES

The following notes explain features, functions, and displays that are common to nearly all of the front panel operations.

Family/Pinout Codes and Device Part Numbers

The required programming algorithm and the pin assignments of devices differ greatly. To program a device via the front panel, the device must be selected using the keypad on the programmer front panel. Device selection may be made by selecting the manufacturer and part number from the 212 menus, or by entering a specific family/pinout code on the front panel keypad (A family/pinout code is a unique number assigned to individual manufacturer's devices by Data I/O). The operating procedures presented in this section allow you to select the device by either method.

NOTE

The device's electronic identifier may also be used to identify the device to the programmer (see the subsection "Devices with Electronic Identifiers" page 2-6) .

To allow for additional family/pinout codes in the future, Data I/O has gone to a six-digit family/pinout code on the front panel menu. Four- or six-digit codes are accepted using the following method:

Procedure	Example 212 Displays
• To enter family/pinout code AF/33, press A, F, ENTER, then press 3, 3, ENTER	
The display reads	LOAD FROM MASTER AF/33
or	
Enter Ø, A, F, Ø, 3, 3	
The display reads	LOAD FROM MASTER ØAF/Ø33
• To enter six-digit family/pinout codes, enter the six digits in order as above	

The family/pinout code for each programmable device is contained in the Device List accompanying this manual. In the menus and also in the Device List, the device manufacturers are listed alphabetically. The parts for each device manufacturer are listed in order of device size, with multiple devices of the same size listed in numerical order.

The Action Display

A special action display appears on the programmer during the execution of certain operations, such as self test, program, and verify. This display consists of decimal points advancing across the lower portion of the front panel display and indicates that the programmer is executing the operation.

Aborting an Operation

Most operations may be aborted by pressing one of the hexadecimal keys. When an operation is halted in this manner, the programmer beeps and reverts to the currently selected mode; e.g., if a hexadecimal key is pressed during a "Load from Master" operation, the programmer beeps and the display returns to

```
LOAD FROM MASTER
```

If a hexadecimal key is accepted as input, press a scroll key to abort the operation.

Error Indicators

If the selected device operation is successfully executed, a completion message appears such as

```
    PROGRAM
SUMCHECK = ØØ1AE5
```

A pass signal (two slow beeps) sounds and the LED above the socket containing the device lights green. However, if execution is unsuccessful for any device, a failure signal (three quick beeps) sounds, the LED above the failed device lights red, and a failure message appears as in the example below. (Refer to the Error Messages section at the back of the manual for a description of the error messages.)

```
    PROGRAM
BAD INSERTION 37
```

Devices with Electronic Identifiers

To benefit from the electronic identifier feature built into many devices, an electronic identifier mode can be selected that causes the programmer to automatically select the correct programming algorithm and pin configuration. To select the electronic identifier mode when blank checking, loading, programming, or verifying a device, enter "ØFFØFF" or "FFFFFF" when prompted for the family and pinout code, or press a scroll key, scroll to the top of the device list, and select "Electronic ID."

When the electronic identifier mode is selected, the programmer determines the programming algorithm and pin configuration of the device that is installed in the socket. If the electronic identifier mode is selected and the device installed does not contain an electronic ID, an error message will be displayed and the electronic ID mode can not be used for the installed device. If an operation is completed successfully using the electronic ID mode, the electronic ID family and pinout code becomes the default family and pinout code.

CHECKING FOR NON-BLANK DEVICES

Blank Check can be used prior to programming operations to ensure that no data has been previously written to a device. The **Blank Check** function verifies that no data exists at any address of the installed device. Perform the blank check as follows:

NOTE

The 212 Displays shown in this manual are examples only. The devices shown in the displays may be different than the devices appearing on your printed Device List or in the menus on your unit since the number of supported devices changes periodically.

Procedure	Example 212 Displays
1. Scroll to the BLANK CHECK function.	`BLANK CHECK`

2. Press ENTER.	`BLANK CHECK`
	`F/P CODE 79/33`

3. If the correct family/pinout code is displayed, press ENTER and go to step 7. If the correct code is not displayed, key in the 6-digit family/pinout code for the device (e.g., ØAFØ33), and skip to step 7	`BLANK CHECK`
	`F/P CODE ØAF/Ø33`

or	

press a scroll key and then scroll to the device manufacturer (e.g., AMD).	`BLANK CHECK`
	`AMD`

Procedure	Example 212 Displays
4. Press ENTER when the correct manufacturer is displayed.	BLANK CHECK AMD 2716
5. Scroll to the part number of the device (e.g., 2764).	BLANK CHECK AMD 2764
6. Press ENTER to select the part number.	INSERT DEVICE AMD 2764
7. Insert the device into the socket with the illuminated LED and press ENTER to initiate the check.	BLANK CHECK ....
If the blank check is successful for the installed device, the socket lamp lights green and the display reads	BLANK CHECK OK
If the blank check operation is unsuccessful for the installed device, the socket lamp lights red and the display reads	BLANK CHECK NONBLANK 20
8. Lift the socket lever and remove the device from the socket.	
9. To blank check another device that uses the same family/pinout code (see the Device List for correct family and pinout codes), press ENTER and return to step 7. To exit the Blank Check operation, press a scroll key.	

PROGRAMMING FROM A MASTER

The Model 212 Multi Programmer can be used to program one device at a time with the data contained in a single master device, or with data downloaded to the programmer via the I/O (RS-232C or parallel) port. The following paragraphs describe the programming operation using a master device. If you choose to download the program data to the programmer's memory instead of loading from a master device, refer also to Downloading Data (RS-232C or parallel port).

Programming a device consists of two basic operations:

□ **Load From Master** - Loading the data from a master device to the programmer's memory (or downloading the data from a remote source such as a host computer — refer to Downloading Data).

□ **Program** - Programs the device with the data copied to the programmer memory.

Loading the Data from a Master Device

The first step in programming a device is to load the data from the master device. The **Load From Master** operation copies the programming data from the master device to the programmer's memory. When the transfer is complete, the programmer calculates and displays the sumcheck of the data. Use the following procedure to load the data from the master device.

Procedure	Example 212 Displays
1. Scroll to the LOAD FROM MASTER function.	LOAD FROM MASTER
2. Press ENTER.	LOAD FROM MASTER F/P CODE FF/FF
3. If the correct family/pinout code is displayed, press ENTER and go to step 7. If the correct code is not displayed, key in the 6-digit family/pinout code for the device (e.g., ØAFØ33), and skip to step 7	LOAD FROM MASTER F/P CODE ØAF/Ø33
or	
press a scroll key and then scroll to the device manufacturer (e.g., AMD).	LOAD FROM MASTER AMD
4. Press ENTER when the correct manufacturer is displayed.	LOAD FROM MASTER AMD 2716

Procedure	Example 212 Displays
5. Scroll to the part number of the device (e.g., 2764).	LOAD FROM MASTER AMD 2764
6. Press ENTER to select the part number.	INSERT DEVICE AMD 2764
7. Insert the master device into the socket with the illuminated LED and press ENTER to initiate the load operation.	LOAD FROM MASTER • • • • • • •
If the load operation is successful, the socket lamp lights green. Note the checksum displayed on the programmer for later verification.	LOAD FROM MASTER SUMCHECK = 00BA25
If the load operation is unsuccessful, the socket lamp lights red and the programmer displays an error message. (See the Error Messages section for an explanation of the error message displayed).	

Programming a Device

After the master data has been loaded into programmer RAM from a master device or downloaded to the programmer, use the following procedure to program devices with the data. The **Program** operation copies the data from the programmer's memory to the installed device. When the programming operation is complete, the programmer calculates and displays the sumcheck of the data.

Procedure	Example 212 Displays
1. Scroll to the PROGRAM function.	`PROGRAM`

2. Press ENTER.
If the data to be programmed was just copied from the master device and the master is the same type of device as the device to be programmed, press ENTER and skip to step 7. Otherwise, proceed to select the device type for the part to be verified.
	`PROGRAM`
	`F/P CODE AF/33`

3. Key in the 6-digit family/pinout code for the device (e.g., 0AF/033) and skip to step 7. or press a scroll key and then scroll to the device manufacturer (e.g., AMD).	`PROGRAM` `AMD`

Procedure	Example 212 Displays
4. Press ENTER when the correct manufacturer is displayed.	PROGRAM AMD 2716
5. Scroll to the part number of the device (e.g., 2764).	PROGRAM AMD 2764
6. Press ENTER to select the part number.	PROGRAM AMD 2764

7. Insert the blank device into the socket and press ENTER to initiate the program operation.

   If the program operation is successful, the socket lamp lights green. Note that the sumcheck displayed on the programmer matches that displayed at the end of the load operation.

   If the program operation is unsuccessful, the socket lamp lights red and the programmer displays an error message. (See the Error Messages section for an explanation of the error message displayed.)

VERIFYING PROGRAMMED PARTS

The **Verify** operation allows you to check programmed devices to make sure that the data in a device matches the data in the programmer's memory. The verify function compares device data with the data read into the programmer's memory during the preceding load or download operation.

The verify operation uses the VccH and VccL and number of passes recommended by the manufacturer of the device being verified. VccH and VccL levels can be selected specifically when using Terminal Remote Control or Computer Remote Control.

Before you can verify the data within a device, you must first make sure that the master data is contained in the programmer's memory. You can load the master data into the programmer's memory by performing either the Load from Master operation described previously in this section.

With the master data contained in the programmer's memory, perform the verify operation as follows:

Procedure	Example 212 Displays
1. Scroll to the VERIFY function.	VERIFY
2. Press ENTER. If the data to be verified was just copied from the master device and the device to be verified is the same type of device as the master, press ENTER and skip to step 7. Otherwise, proceed to select the device type for the part to be verified.	VERIFY F/P CODE AF/33

Procedure	Example 212 Displays
3. Key in the 6-digit family/pinout code for the device (e.g., ØAFØ33) and skip to step 7	VERIFY F/P CODE ØAF/Ø33
or	
press a scroll key and then scroll to the device manufacturer (e.g., AMD).	VERIFY AMD
4. Press ENTER when the correct manufacturer is displayed.	VERIFY AMD 2716
5. Scroll to the part number of the device (e.g., 2764).	VERIFY AMD 2764
6. Press ENTER to select the part number.	INSERT DEVICE AMD 2764

Procedure	Example 212 Displays
7. Insert the device to be verified into the socket with the illuminated LED and press ENTER to initiate the verify operation.	VERIFY ....
If the verify operation is successful, the socket lamp lights green and the sumcheck of the data is displayed	VERIFY SUMCHECK=ØØBA25
For unsuccessful verify operations, the programmer display differs for 8-bit devices and 16-bit devices as follows:	
If an **8-bit device** verify operation is unsuccessful, the programmer will display the address location where the error occurs, the RAM data, and the device data.	VERIFY XXXXX R:XX D:XX (5-digit addr.)(RAM Data)(Device Data)
Pressing ENTER advances to the next address where RAM data and device data don't match. When the programmer verifies the complete device, the programmer will respond with a Verify Error Message.	VERIFY VERIFY ERROR NN (Where NN equals the error message)

	Procedure	Example 212 Displays
	If a **16-bit device** verify operation is unsuccessful, the programmer LCD display is not large enough to display the entire data word. Therefore, the display will first indicate the address and the low byte data	VERIFY XXXXX LO R:XX D:XX (5-digit addr.)(Byte)(RAM Data)(Device Data)
	Pressing ENTER will display the high byte data.	VERIFY XXXXX HI R:XX D:XX (5-digit addr.)(Byte)(RAM Data)(Device Data)
	Pressing ENTER advances to the next address where RAM data and device data don't match. When the programmer verifies the complete device the programmer will respond with a Verify Error Message.	VERIFY VERIFY ERROR NN
8.	Lift the socket lever and remove the device from the socket.	
9.	To verify another device of the same type (using the same family and pinout codes) against the same master data, press ENTER and return to step 7. To exit the Verify operation, press the scroll key.	

SETTING COMMUNICATIONS PROTOCOL (RS-232 Only)

The programmer can be made to select the correct communications protocol automatically (see Terminal Remote Control or Computer Remote Control) or you can change the communications protocol from the power-up default values by using the following procedure. The power-up default values are:

baud rate = 9600 parity check = none number of data bits = 8 number of stop bits = 1

Procedure	Example 212 Displays
1. Scroll to the RS232 PORT function.	`RS232 PORT`

2. Press ENTER.	`RS232 PORT`
	`COMPUTER CONTROL`

3. Scroll to the PORT SETTING function.	`RS232 PORT`
	`PORT SETTING`

4. Press ENTER.	`PORT SETTING`
	`BAUD RATE = 9600`

5. Scroll to the desired data transfer speed and press ENTER.	`PORT SETTING`
	`PARITY CHK.=NONE`

Procedure	Example 212 Displays

6. Scroll to the desired parity and press ENTER.	`PORT SETTING`
	`DATA BITS = 8`

7. Scroll to the desired number of data bits and press ENTER.	`PORT SETTING`
	`STOP BIT = 1`

8. Scroll to the desired number of stop bits and press ENTER.	`RS232  PORT`

This completes the setting of the communications protocol for the serial I/O port.

DOWNLOADING DATA USING RS-232 PORT

The 212 can receive and store data transferred from a host computer into its memory over a serial communications link. (Refer to the Getting Started section for information on serial I/O connections.) This feature may be used to download data from a computer, or other software development system, to the programmer in preparation for programming parts. The programmer can accept the data in any of the data translation formats shown on the 212 menus. Descriptions of each of these formats are provided in the Computer Remote Control section of this manual.

When initiating a download operation, the programmer prompts for the entry of three parameters in addition to the data translation format. These parameters are:

□ the beginning RAM address

□ the block size

□ the offset address

Selection of the beginning RAM address allows you to make transfers of data to a specified start address in the programmer's memory. For example, if you specify a beginning address of 1000H (hexadecimal), the data transferred from the external data source will be placed in the programmer's memory starting at location 1000H instead of 0000H.

Selection of the block size allows you to specify how much data is to be transferred to the programmer's memory. For example, if you specify a block size of 2000H, the subsequent data transfer operation would store 8192 (2000H) bytes of data in the programmer's memory. To disable the selected block size and load data to the end of RAM (or the end of the data file, whichever is smaller), enter 0000H. The power-up default block size is 0000H.

Selection of the offset address allows you to specify the first data record of the data file in the host computer to be downloaded. Each data record in the data file contains a "record address" which specifies the address of the first data byte in the record. The record address is assigned when the program is compiled into formatted object code. During a download operation, the offset address specifies the record address of the first data byte that will be stored in RAM. For example, if you specify an offset address of 2000H, and a beginning RAM address of 1000H, the data bytes contained in the record with record address 2000H (the offset address) will be stored in RAM starting at the beginning RAM address (1000H). Data bytes contained in records with subsequent addresses will be stored in subsequent RAM addresses, up through the block size specified. Data records with record addresses less than the offset address will not be stored in RAM. The default offset address, "FFFFFFFF," has special meaning to the translators and causes the first data byte received to be stored in the beginning RAM address and subsequent data bytes received to be stored in subsequent RAM addresses.

After you have hooked up the programmer to the host computer and set the proper communications protocol (as described in the previous subsection), perform a **Download** operation as follows:

Procedure	**Example 212 Displays**
1. Scroll to the RS232 PORT function.	RS232 PORT
2. Press ENTER.	RS232 PORT COMPUTER CONTROL
3. Scroll to the DOWNLOAD function.	RS232 PORT DOWNLOAD
4. Press ENTER.	DOWNLOAD FORMAT INTEL INTELLEC 8
5. Scroll to the desired data translation format (e.g., Motorola Exormax) and press ENTER.	DOWNLOAD EXORMAX BE RAM AD=ØØØØØØ
6. Press ENTER to select the displayed beginning RAM address (the first programmer memory address that the downloaded data will be transferred to) or key in the hexadecimal beginning address and press ENTER.	DOWNLOAD EXORMAX BLCK SIZE=ØØØØØØ

Procedure	Example 212 Displays
7. Press ENTER to select the displayed block size, or key in the block size of the data to be downloaded and press ENTER. A block size of zero loads the data file up to the end of programmer memory or up to the end of the data file, whichever is smaller.	DOWNLOAD EXORMAX OFFSET =FFFFFFFF
8. Press ENTER to select the displayed offset address or key in the offset address (record address) of the first data byte to be downloaded and press ENTER. (The default offset address, FFFFFFFF, will cause the first data byte received to be stored at the beginning RAM address.)	LOADING PORT
9. Initiate the download of the data from the host computer. When the download is complete, the sumcheck of the data received by the programmer will be displayed.	LOADING PORT . . . . LOADING COMPLETE SUMCHECK = ØØ9FFØ

UPLOADING DATA USING RS-232 PORT

The 212 can be used to transfer device data trom its memory to a computer over a serial communications link. (Refer to the Getting Started section for information on serial I/O connections.) This feature may be used to upload data from the programmer's memory for the purposes of storing the data or editing it and then downloading it back to the programmer's memory. The programmer can transfer the data in any of the data translation formats shown on the 212 menus. Descriptions of each of these formats are provided in the Computer Remote Control section of this manual.

When initiating an upload operation, the programmer prompts for the entry of three parameters in addition to the data translation format. These parameters are:

□ the beginning RAM address

□ the block size

□ the offset address

Selection of the beginning RAM address allows you to make transfers that use only a portion of data contained in the programmer's memory. For example, if you specify a beginning address of 8000H (hexadecimal), the data transfer operation will start with programmer RAM address 8000H (32,768 decimal) instead of the default beginning address. The power-up default beginning RAM address is 0000H.

Selection of the block size allows you to specify how much of the data contained in the programmer's memory is to be transferred, starting at the specified beginning address. For example, if you specify a beginning address of 8000H and a block size of 1000H, the subsequent data transfer operation would transfer only the data contained in addresses 8000H through 8FFFH. If you do not specify a block size, the default block size is the size of the current device times the set size specified in the last load or download operation.

Selection of the offset address allows you to select the address which will be assigned to the first data byte uploaded to the host computer. The data in programmer RAM is translated into the specified data translation format prior to being sent through the RS-232C port. The data is formatted into data records which include a "record address." The record address is the address assigned to the first byte of data contained in that data record. The offset address specifies the first record address of the block of data being transferred. For example, if you specify a beginning RAM address of 8000H and an offset address of 2000H, then the data from programmer RAM address 8000H will be assigned a record address of 2000H. Consecutive RAM data bytes following the first data byte will be assigned consecutive record addresses (2001H, 2002H, and so on).

After hooking up the programmer to the host computer and setting the correct communications protocol, perform an **Upload** operation as follows:

Procedure	**Example** **212 Displays**
1. Scroll to the RS232 PORT function.	RS232 PORT
2. Press ENTER.	RS232 PORT COMPUTER CONTROL
3. Scroll to the UPLOAD function.	RS232 PORT UPLOAD
4. Press ENTER.	UPLOAD FORMAT INTEL INTELLEC 8
5. Scroll to the desired data translation format (e.g., Intel MCS-86 Hexadecimal) and press ENTER.	UPLOAD MCS-86 BE RAM AD=ØØØØØØ
6. Press ENTER to select the displayed beginning RAM address, or key in the hexadecimal beginning address of the data in the programmer's memory to be uploaded and press ENTER.	UPLOAD MCS-86 BLCK SIZE=ØØØØØØ

7. Press ENTER to select the displayed block size, or key in the block size of the data to be uploaded and press ENTER. (Entering a block size of 0000 selects the current device size times the set size specified in the last load or program operation.)

```
 UPLOAD MCS-86
OFFSET =FFFFFFFF
```

---

8. Press ENTER to select the displayed offset address, or key in the offset address for the data and press ENTER. (The default offset address, FFFFFFFF, causes address 0000 to be assigned to the first data byte sent by the programmer.)

```
  SENDING PORT
...........
```

When the upload operation is complete, the programmer displays the sumcheck of the data that was sent over the serial RS-232C port in hexadecimal notation.

```
SENDING COMPLETE
SUMCHECK = ØØ9FFØ
```

DOWNLOADING DATA USING PARALLEL PORT

The 212 can receive and store data transferred from a host computer into its memory over a parallel communications link. (Refer to the Getting Started section for information on parallel I/O connections.) This feature may be used to download data from a computer, or other software development system, to the programmer in preparation for programming parts. The programmer can accept the data in any of the data translation formats shown on the 212 menus. Descriptions of each of these formats are provided in the Computer Remote Control section of this manual.

When initiating a download operation, the programmer prompts for the entry of three parameters in addition to the data translation format. These parameters are:

□ the beginning RAM address

□ the block size

□ the offset address

Selection of the beginning RAM address allows you to make transfers of data to a specified start address in the programmer's memory. For example, if you specify a beginning address of 1000H (hexadecimal), the data transferred from the external data source will be placed in the programmer's memory starting at location 1000H instead of 0000H.

Selection of the block size allows you to specify how much data is to be transferred to the programmer's memory. For example, if you specify a block size of 2000H, the subsequent data transfer operation would store 8192 (2000H) bytes of data in the programmer's memory. To disable the selected block size and load data to the end of RAM (or the end of the data file, whichever is smaller), enter 0000H. The power-up default block size is 0000H.

Selection of the offset address allows you to specify the first data record of the data file in the host computer to be downloaded. Each data record in the data file contains a "record address" which specifies the address of the first data byte in the record. The record address is assigned when the program is compiled into formatted object code. During a download operation, the offset address specifies the record address of the first data byte that will be stored in RAM. For example, if you specify an offset address of 2000H, and a beginning RAM address of 1000H, the data bytes contained in the record with record address 2000H (the offset address) will be stored in RAM starting at the beginning RAM address (1000H). Data bytes contained in records with subsequent addresses will be stored in subsequent RAM addresses, up through the block size specified. Data records with record addresses less than the offset address will not be stored in RAM. The default offset address, "FFFFFFFF," has special meaning to the translators and causes the first data byte received to be stored in the beginning RAM address and subsequent data bytes received to be stored in subsequent RAM addresses.

After you have hooked up the programmer to the host computer use the **RECEIVE PORT** function to perform a download operation as follows:

Procedure	Example 212 Display
1. Scroll to the PARALLEL PORT function.	`PARALLEL PORT`

2. Press ENTER.	`PARALLEL PORT`
	`RECEIVE PORT`

3. Press ENTER.	`RECEIVE FORMAT`
	`INTEL INTELLEC 8`

4. Scroll to the desired data translation format (e.g., Motorola Exormax) and press ENTER.	`RECEIVE EXORMAX`
	`BE RAM AD=000000`

5. Press ENTER to select the displayed beginning RAM address (the first programmer memory address that the downloaded data will be transferred to) or key in the hexadecimal beginning address and press ENTER.	`DOWNLOAD EXORMAX`
	`BLCK SIZE=000000`

Procedure	Example 212 Displays
6. Press ENTER to select the displayed block size, or key in the block size of the data to be downloaded and press ENTER. A block size of zero loads the data file up to the end of programmer memory or up to the end of the data file, whichever is smaller.	RECEIVE EXORMAX OFFSET =FFFFFFFF
7. Press ENTER to select the displayed offset address or key in the offset address (record address) of the first data record to be downloaded and press ENTER. (The default offset address, FFFFFFFF, will cause the first data byte received to be stored in the beginning RAM address.)	RECEIVING PORT
8. Initiate the download of the data from the host computer. When the download is complete, the sumcheck of the data received by the programmer will be displayed.	RECEIVING PORT . . . . RECEIVE COMPLETE SUMCHECK = ØØ9FFØ

UPLOADING DATA USING PARALLEL PORT

The 212 can be used to transfer device data from its memory to a computer over a parellel communications link. (Refer to the Getting Started section for information on serial I/O connections.) This feature may be used to upload data from the programmer's memory for the purposes of storing the data or editing it and then downloading it back to the programmer's memory. The programmer can transfer the data in any of the data translation formats shown on the 212 menus. Descriptions of each of these formats are provided in the Computer Remote Control section of this manual.

When initiating an upload operation, the programmer prompts for the entry of three parameters in addition to the data translation format. These parameters are:

□ the beginning RAM address

□ the block size

□ the offset address

Selection of the beginning RAM address allows you to make transfers that use only a portion of data contained in the programmer's memory. For example, if you specify a beginning address of 8000H (hexadecimal), the data transfer operation will start with programmer RAM address 8000H (32,768 decimal) instead of the default beginning address. The power-up default beginning RAM address is 0000H.

Selection of the block size allows you to specify how much of the data contained in the programmer's memory is to be transferred, starting at the specified beginning address. For example, if you specify a beginning address of 8000H and a block size of 1000H, the subsequent data transfer operation would transfer only the data contained in addresses 8000H through 8FFFH. If you do not specify a block size, the default block size is the size of the current device times the set size specified in the last load or download operation.

Selection of the offset address allows you to select the address which will be assigned to the first data byte uploaded to the host computer. The data in programmer RAM is translated into the specified data translation format prior to being sent through the parallel port. The data is formatted into data records which include a "record address." The record address is the address assigned to the first byte of data contained in that data record. The offset address specifies the first record address of the block of data being transferred. For example, if you specify a beginning RAM address of 8000H and an offset address of 2000H, then the data from programmer RAM address 8000H will be assigned a record address of 2000H. Consecutive RAM data bytes following the first data byte will be assigned consecutive record addresses (2001H, 2002H, and so on).

After hooking up the programmer to the host computer, use the **SEND PORT** function to perform an upload operation as follows:

Procedure	Example 212 Displays
1. Scroll to the PARALLEL PORT function.	`PARALLEL PORT`

2. Press ENTER.	`PARALLEL PORT`
	`RECEIVE PORT`

3. Scroll to the SEND function.	`PARALLEL PORT`
	`SEND PORT`

4. Press ENTER.	`SEND FORMAT`
	`INTEL INTELLEC 8`

5. Scroll to the desired data translation format (e.g., Intel MCS-86 Hexadecimal) and press ENTER.	`SEND MCS-86`
	`BE RAM AD=000000`

6. Press ENTER to select the displayed beginning RAM address, or key in the hexadecimal beginning address of the data in the programmer's memory to be uploaded and press ENTER.	`SEND MCS-86`
	`BLCK SIZE=000000`

Procedure	Example 212 Displays
7. Press ENTER to select the displayed block size, or key in the block size of the data to be uploaded and press ENTER. (Entering a block size of 0000 selects the current device size times the set size specified in the last load or download operation.)	SEND MCS-86 OFFSET =FFFFFFFF
8. Press ENTER to select the displayed offset address, or key in the offset address for the data and press ENTER. (The default offset address, FFFFFFFF, causes address 0000 to be assigned to the first data byte sent by the programmer.)	SENDING PORT ....
When the upload operation is complete, the programmer displays the sumcheck of the data that was sent over the parallel port in hexadecimal notation.	SENDING COMPLETE SUMCHECK = ØØ9FFØ

EDITING RAM DATA

After the master program data has been loaded into the programmer's RAM, that data can be altered, added to, or deleted from using the **Edit** functions. The data can also be searched through and some special programming features can be changed. The **Edit** functions are summarized below (Note that the editing functions differ slightly between the EPROM and microprocessor modules). They are listed below in the same order that they occur in the 212 menus and are also discussed in more detail (in the same order) in the pages that follow.

EDITING FUNCTIONS FOR EPROM AND MICROPROCESSOR MODULES

□ COMPLEMENT—inverts each byte of data in a range of RAM.

□ INSERT—causes the programmer to enter insert mode and allow you to insert data bytes into consecutive addresses one-by-one. Existing data bytes are shifted up to higher addresses to accommodate the new data.

□ BLOCK INSERT—inserts one value into a range of addresses. Existing data bytes are shifted up to higher addresses to accommodate the new data.

□ DELETE—causes the programmer to enter delete mode and allow you to delete data bytes.

□ BLOCK DELETE—deletes a block of data.

□ BLOCK FILL—fills a range of addresses with a single value. The data originally contained in the filled addresses is overwritten.

□ DATA SEARCH—searches for up to eight bytes of data in a specified range of RAM.

□ BLOCK COPY—copies a block of data from one location in RAM to another, overwriting the data originally stored in the addresses copied to (destination addresses).

□ DATA EDIT - allows you to enter an editing mode which allows you to change data bytes one-by-one.

□ OPTION EDIT - allows you to change special programming functions, such as enabling and disabling electronic ID checking, enabling and disabling device testing, specifying an initial byte, and enabling or disabling synchronous programming (for Cypress devices).

□ BYTE SWAP - use the byte swap function to swap the contents of memory byte pairs over a specified range.

□ SHUFFLE - combines two blocks of 8-bit wide data into one block of 16-bit-wide words for subsequent uploading or programming in 16-bit-wide format.

□ SPLIT - splits a block of 16-bit-wide data into two adjacent 8-bit-wide blocks that take up the same amount of programmer RAM. Split always starts at address 0.

EDITING FUNCTIONS FOR MICROPROCESSOR MODULES ONLY

□ ENCRYPTION EDIT - allows you to edit the 32 bytes that are reserved for encryption.

□ SIGNATURE EDIT - allows you to edit the 2 bytes that are reserved for user signature.

Complementing Data

Use the **Complement** function to invert the data contained in a specified range of memory (a ØØ will become an FF; an FF will become a ØØ; an EE will become a 11; an 11 will become an EE, and so on). Data complements are based on 8-bit-word size.

Procedure	Example 212 Displays
1. Scroll to the EDIT function.	EDIT
2. Press ENTER.	EDIT COMPLEMENT
3. Scroll to the COMPLEMENT function (if necessary) and press ENTER.	COMPLEMENT START ADD=000000
4. Key in the first address of the range of data you want to complement and press ENTER. The default start address is 0.	COMPLEMENT END ADD=03FFFF
5. Key in the last address of the range of data you want to complement and press ENTER. The default end address is the end of memory (03FFFF for a programmer with 256K of memory).	COMPLEMENT . . . .
When the complement is complete, the display will read	COMPLEMENT COMPLETED

Inserting Single Data Bytes

Use the **Insert** function to insert data bytes one at a time into consecutive addresses. (To insert the same value into a block of memory, use the Block Insert function.) Executing the Insert function causes the programmer to go into an insert mode. You may continue inserting data bytes into consecutive memory addresses until you press a scroll key. Pressing either of the scroll keys will cause the programmer to exit insert mode and return to the main menu. The original data at the address where new data is inserted is shifted up in memory (to higher addresses) to accommodate the new data. Data at the end of memory is lost.

Procedure	Example 212 Displays
1. Scroll to the EDIT function.	`EDIT`
2. Press ENTER.	`EDIT` `COMPLEMENT`

3. Scroll to the INSERT function and press ENTER.	`INSERT`
	`ADDRESS = 000000`

4. Key in the address of the memory location where you want to begin inserting data (e.g., 1000) and press ENTER. The default start address of insertion is 0.	`ADDRESS = 001000`
	`INSERT DATA = __`

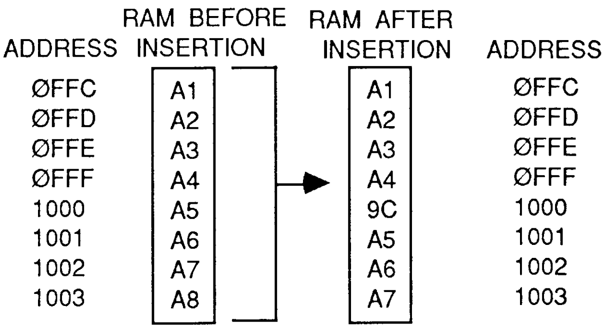
Procedure	Example 212 Displays

5. Key in the data value that you want to insert at the address displayed on the top line of the display.	`ADDRESS = 001000`
	`INSERT DATA = 9C`

6. Press ENTER to insert the data into memory. When you press ENTER, the programmer will display the next consecutive address, at which the next keyed in byte of data will be inserted. To continue inserting data into consecutive addresses, return to step 5. To exit insert mode and return to the main menu, press either of the scroll keys.	`ADDRESS = 001001`
	`INSERT DATA = __`

Example

Insert data 9C at address 1000:



ADDRESS	RAM BEFORE INSERTION	RAM AFTER INSERTION	ADDRESS
ØFFC	A1	A1	ØFFC
ØFFD	A2	A2	ØFFD
ØFFE	A3	A3	ØFFE
ØFFF	A4	A4	ØFFF
1000	A5	9C	1000
1001	A6	A5	1001
1002	A7	A6	1002
1003	A8	A7	1003

Inserting a Single Value into a Block of RAM

Use the **Block Insert** function to insert one value into a range of memory addresses. The original data in the range where new values are inserted is shifted up in memory (to higher addresses) to accommodate the new data. Data at the end of memory is lost. (To overwrite a range of memory addresses with a value, use the Block Fill function.)

Procedure	Example 212 Displays
1. Scroll to the EDIT function.	`EDIT`
2. Press ENTER.	`EDIT` `COMPLEMENT`

3. Scroll to the BLOCK INSERT function and press ENTER.	`BLOCK INSERT`
	`START ADD=000000`

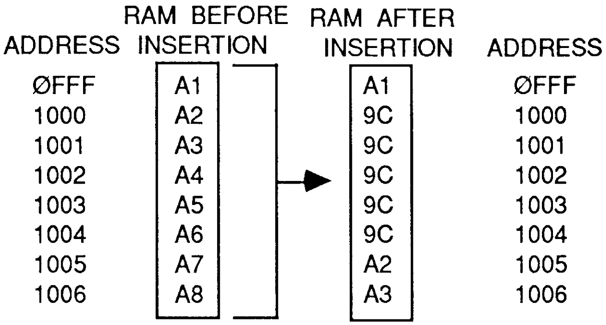
4. Key in the first address of the range of memory in which data is to be inserted and press ENTER. The default start address is 0.	`BLOCK INSERT`
	`END ADD=03FFFF`

5. Key in the last address of the range of memory in which data is to be inserted and press ENTER. The default end address is the end of memory (03FFFF for a programmer with 256K of memory).	`BLOCK INSERT`
	`DATA = __`

Procedure	Example 212 Displays
6. Key in a one byte value to be inserted into the specified range of memory.	BLOCK INSERT DATA = 9C
7. Press ENTER to execute the Block Insert function.	BLOCK INSERT ....
When the insert function is complete, the display will read	BLOCK INSERT COMPLETED

Example

Insert data 9C into block from 1000 through 1004:



ADDRESS	RAM BEFORE INSERTION	RAM AFTER INSERTION	ADDRESS
ØFFF	A1	A1	ØFFF
1000	A2	9C	1000
1001	A3	9C	1001
1002	A4	9C	1002
1003	A5	9C	1003
1004	A6	9C	1004
1005	A7	A2	1005
1006	A8	A3	1006

Deleting Single Data Bytes

Use the **Delete** function to delete data bytes one at a time. Data following the deleted data byte is shifted down to fill in the deleted address and 00 is filled in the last byte of RAM. Executing the Delete function causes the programmer to go into a delete mode. You may continue deleting successive data bytes until you press a scroll key. Pressing a scroll key causes the programmer to exit delete mode and return to the main menu.

Procedure	**Example 212 Displays**
1. Scroll to the EDIT function.	`EDIT`
2. Press ENTER.	`EDIT` `COMPLEMENT`

3. Scroll to the DELETE function and press ENTER.	`DELETE`
	`ADDRESS = 000000`

4. Key in the address of the memory location where you want to begin deleting data (e.g., 1000) and press ENTER. The default start address of deletion is 0. The current value of the data contained in the specified address is displayed on the top line of the display.	`DELETE DATA = 93`
	`ADDRESS = 001000`

Procedure	Example 212 Displays

5. Press ENTER to delete the current contents of the displayed address. The next data byte is shifted down to the current address displayed. (The same address is displayed but it contains the next data byte.)	`DELETE DATA = 94`
	`ADDRESS = 001000`

6. To continue deleting successive data bytes, continue repeating step 5. To exit delete mode and return to the main menu, press either of the scroll keys.	

Example

Delete data at address 1000:

ADDRESS	RAM BEFORE DELETION	RAM AFTER DELETION	ADDRESS
ØFFF	A1	A1	ØFFF
1000	A2	A3	1000
1001	A3	A4	1001
1002	A4	A5	1002
1003	A5	•	•
•	•	•	•
•	•	FF	3FFFE
3FFFF	FF	00	3FFFF

Deleting a Block of Data

Use the **Block Delete** function to delete a block of data stored in the programmer's RAM. Data following the deleted block is shifted down to fill in the deleted range and 00 is filled in the vacated RAM addresses at the end of memory.

Procedure	Example 212 Displays
1. Scroll to the EDIT function.	EDIT
2. Press ENTER.	EDIT COMPLEMENT
3. Scroll to the BLOCK DELETE function and press ENTER.	BLOCK DELETE START ADD=000000
4. Key in the address of the first data byte to be deleted (e.g., 1000) and press ENTER. The default start address is 0.	BLOCK DELETE END ADD=03FFFF
5. Key in the address of the last data byte to be deleted (e.g., 1FFF). The default end address is the end of RAM.	BLOCK DELETE END ADD=001FFF
6. Press ENTER to execute the Block Delete function.	BLOCK DELETE . . . .
When the delete function is complete, the display will read	BLOCK DELETE COMPLETED

Filling a Block of Memory

Use the **Block Fill** function to overwrite the data contained in a range of RAM with a single value (data byte).

Procedure	Example 212 Displays
1. Scroll to the EDIT function.	EDIT
2. Press ENTER.	EDIT COMPLEMENT
3. Scroll to the BLOCK FILL function and press ENTER.	BLOCK FILL START ADD=000000
4. Key in the first address of the range of memory to be overwritten by the new data value (e.g., 1000) and press ENTER. The default start address is 0.	BLOCK FILL END ADD=03FFFF
5. Key in the last address of the range of memory to be overwritten by the new data value (e.g., 1FFF) and press ENTER. The default end address is the end of RAM (3FFFF for a programmer with 256K of RAM).	BLOCK FILL DATA = __
6. Key in a one byte value to be written into the specified range of memory (e.g., 9C).	BLOCK FILL DATA = 9C
7. Press ENTER to execute the Block Fill function.	BLOCK FILL . . . .
When the Block Fill function is complete, the display will read	BLOCK FILL COMPLETED

Searching for Data

Use the **Data Search** function to search for up to eight bytes of data in any specified range of memory. You must specify a range of memory to search that is larger than the block of data bytes to be searched for or the data search will not be executed.

Procedure	Example 212 Displays
1. Scroll to the EDIT function.	`EDIT`
2. Press ENTER.	`EDIT` `COMPLEMENT`

3. Scroll to the DATA SEARCH function and press ENTER.	`DATA SEARCH`
	`START ADD=000000`

4. Key in the address of the memory location where you want the data search to begin (e.g., 1000) and press ENTER. The default start address is 0.	`DATA SEARCH`
	`END ADD=03FFFF`

5. Key in the address of the memory location where you want the data search to end (e.g., 1FFF) and press ENTER. The default end address is the end of RAM (3FFFF for a programmer with 256K of RAM).	`DATA = __`

Procedure	Example 212 Displays
6. Key in the value of a byte to be searched for in the specified range of memory and press ENTER to enter that byte in the search string. (The comma will be provided by the programmer when you press ENTER.)	DATA = 01,
7. To enter the next byte of data to be searched for, return to step 6, or press ENTER to execute the Data Search function for the data already entered. You may enter up to eight bytes of data to be searched for. After pressing ENTER to accept the eighth byte, the search will begin immediately and you will not have to press ENTER a second time. A match will occur only if the data is found in the same order as the typed in string of data.	DATA = 01,02,03, 04,05,06,07,08
8. The programmer will search the specified range of memory for the data string entered on the display. If the data is found, the programmer will display the address of the first byte of the string found. You can then press ENTER to continue searching for other occurrences of the data in the specified range, or you can return to the main menu by pressing either scroll key.	DATA MATCH ADDRESS = 10FF
If the data was not found in the specified range (or the data string was longer than the specified range) the programmer will display a fail message. Press any key to return to the main menu.	DATA SEARCH FAIL

Copying Data from One RAM Location to Another

Use the **Block Copy** function to copy data from one RAM location to another. The original data contained in the addresses to which the data was copied is overwritten by the copied data. Also make sure that the destination address of the block to be copied plus the block size of the block to be copied does not exceed programmer RAM (3FFFF for a programmer with 256K of memory).

Procedure	Example 212 Displays
1. Scroll to the EDIT function.	EDIT
2. Press ENTER.	EDIT COMPLEMENT
3. Scroll to the BLOCK COPY function and press ENTER.	BLOCK COPY START ADD=000000
4. Key in the address of the first data byte you want to copy and press ENTER. The default start address is 0.	BLOCK COPY END ADD=03FFFF
5. Key in the address of the last byte of data you want to copy and press ENTER. (If you want to copy only one byte of data, type in the same value for the end address as you did for the start address.) The default end address is the end of RAM.	BLOCK COPY DESTN ADD=000000

Procedure	Example 212 Displays

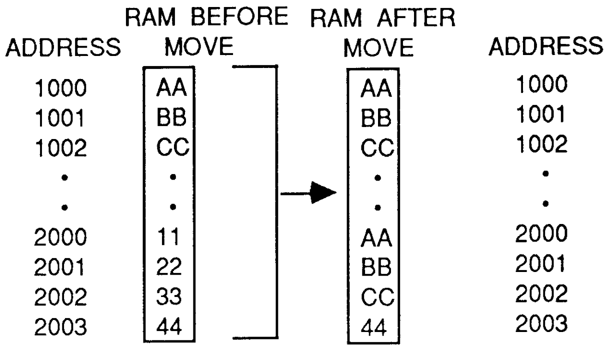
6. Key in the address to which the first byte of data will be copied (destination address) (e.g., 2000). The rest of the data to be copied will be copied into consecutive addresses following the destination address.	`BLOCK COPY`
	`DESTN ADD=002000`

7. Press ENTER to execute the Block Copy function.	`BLOCK COPY`
	`........`

When the copy is complete, the display will read	`BLOCK COPY`
	`COMPLETED`

Example

Copy block 1000 through 1002 to address 2000:



Editing Single Data Bytes

Use the **Data Edit** function to edit individual bytes of data stored in the programmer's RAM. Executing this command causes the programmer to go into a special edit mode that will allow you to display and edit data bytes one-by-one until you press a scroll key.

Procedure	Example 212 Displays
1. Scroll to the EDIT function.	EDIT
2. Press ENTER.	EDIT SPLIT
3. Scroll to the DATA EDIT function and press ENTER.	DATA EDIT ADDRESS = 0000
4. Key in the address of the first byte you want to edit (e.g., 1000) and press ENTER. The default address to start editing at is 0. The current value of the data at the address displayed is shown on the second line of the display.	ADDRESS = 10000 DATA 05-->_ _
5. To change the current data byte (as in the example), key in the new value and press ENTER. To leave the current data byte unchanged and advance to the next byte to be edited, press ENTER without keying in a new value.	ADDRESS = 10000 DATA 05-->9C

Procedure	Example 212 Displays

6. After pressing ENTER, the next address will be displayed on the top line of the display and the current value of the data at that address will be displayed on the second line of the display. To edit the displyed data byte, repeat step 5. To exit the edit mode and return to the main menu, press either scroll key. If you key in a new value and then press a scroll key before press ENTER, the keyed in data will be ignored and the current data byte will remain unchanged.	`ADDRESS = 1001`
	`DATA 06--> _ _`

Changing Special Programming Options

Use the **Option Edit** function to change the special programming functions. For instance, use this function to enable or disable electronic ID checking. If electronic ID checking is enabled, the 212 will check to make sure that the installed device's electronic ID matches the device type selected during a device operation (such as programming). If a mismatch occurs, an error message will be displayed and the operation will be halted. Option Edit also allows you to use special features available on Cypress devices. These features are initialize byte programming and synchronous programming. See the device manufacturer's documentation for information on these device features.

Note

*The **Option Edit** function differs for EPROM and microprocessor modules. For instance, use this function to enable or disable **electronic ID checking** for both EPROM and microprocessor modules; use this function to enable or disable **device testing** for the microprocessor module only; use this function to enable or disable **synchronous programming** for the EPROM module only; and use this function to set the **initialize byte** for the EPROM module only.*

Enabling/Disabling Electronic ID Checking

Within **Option Edit**, use the **ID EN / DIS** function to enable or disable electronic ID checking for EPROM and microprocessor modules, as follows:

Procedure	Example 212 Displays
1. Scroll to the EDIT function.	`EDIT`
2. Press ENTER.	`EDIT` `COMPLEMENT`

3. Scroll to the OPTION EDIT function and press ENTER.	`OPTION EDIT`
	`ID EN/DIS`

4. Scroll to the ID Enable/Disable function (if necessary) and press ENTER. If electronic ID checking is currently enabled "ENABLE" will appear on the lower portion of the display. If electronic ID checking is disabled, "DISABLE" will appear on the display. Electronic ID checking is enabled automatically on power up.	`ID EN/DIS`
	`ENABLE`

5. To change the current status of electronic ID checking, press a scroll key and then press ENTER. To leave the current status unchanged, press ENTER without pressing a scroll key, or scroll back to the previous value and press ENTER.	`ID EN/DIS`
	`DISABLE`

Enabling/Disabling Device Testing

Within **OPTION EDIT**, use the **EN / DIS DEV TEST** function to enable or disable device testing for the microprocessor module only, as follows:

Procedure	Example 212 Displays
1. Scroll to the EDIT function.	EDIT
2. Press ENTER.	EDIT COMPLEMENT
3. Scroll to the OPTION EDIT function and press ENTER.	OPTION EDIT ID EN/DIS
4. Scroll to the EN/DIS DEV TEST function (if necessary) and press ENTER. If Device Testing is currently enabled "DISABLE" will appear on the lower portion of the display. If Device Testing is disabled, "ENABLE" will appear on the display. Device Testing is automatically disabled on power up.	EN/DIS DEV TEST DISABLE
5. To change the current status of Device Testing, press a scroll key and then press ENTER. To leave the current status unchanged, press ENTER without pressing a scroll key, or scroll back to the previous value and press ENTER.	EN/DIS DEV TEST ENABLE

Enabling/Disabling Synchronous Programming

Within **OPTION EDIT**, use the **SYNCH EN / DIS** function to enable or disable synchronous programming for the EPROM module only, as follows:

Procedure	Example 212 Displays
1. Scroll to the EDIT function.	`EDIT`
2. Press ENTER.	`EDIT` `COMPLEMENT`

3. Scroll to the OPTION EDIT function and press ENTER.	`OPTION EDIT`
	`ID EN/DIS`

4. Scroll to the SYNCH EN/DIS function and press ENTER. If synchronous programming is currently disabled "DISABLE" will appear on the lower portion of the display. If synchronous programming is enabled, "ENABLE" will appear on the display. Synchronous programming is disabled automatically on power up.	`SYNCH EN/DIS`
	`DISABLE`

5. To change the current status of synchronous programming, press a scroll key and then press ENTER. To leave the current status unchanged, press ENTER without pressing a scroll key, or scroll back to the previous value and press ENTER.	`SYNCH EN/DIS`
	`ENABLE`

Setting the Initial Byte

Within **OPTION EDIT**, use the **INITIAL BYTE** function (for the EPROM module only) to set the value of the initiaizel byte to be programmed into devices supporting the initialize byte feature, as follows:

Procedure	Example 212 Displays
1. Scroll to the EDIT function.	EDIT
2. Press ENTER.	EDIT COMPLEMENT
3. Scroll to the OPTION EDIT function and press ENTER.	OPTION EDIT ID EN/DIS
4. Scroll to the Initial Byte function and press ENTER.	INITIAL BYTE DATA 00 --> _ _
5. Type in the value of the initial byte data (e.g., B5) and press ENTER. To leave the value unchanged, press a scroll key. The default data value is 00 (zero).	INITIAL BYTE DATA 00 --> B5

Encryption Edit

Use the **Encryption Edit** function for the microprocessor module only to edit the 32 bytes of key data reserved in the programmer's RAM. The clear data is encrypted with the key data to provide the cypher data that will be stored when the device is programmed. The default key data is FF.

Procedure	Example 212 Displays
1. Scroll to the EDIT function.	`EDIT`
2. Press ENTER.	`EDIT` `COMPLEMENT`

3. Scroll to the ENCRYPTION EDIT function and press ENTER.	`ENCRYPTION EDIT`
	`ENCRYP. ADDR = 00`

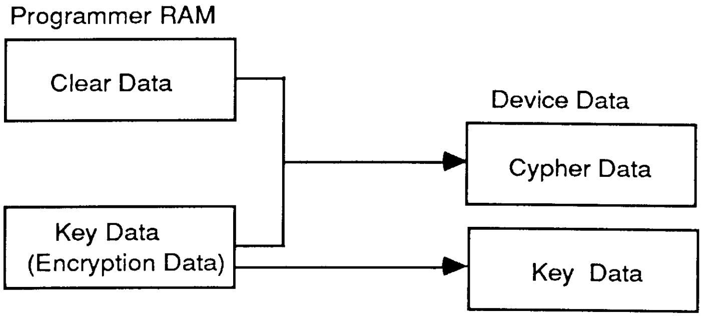
4. Key in the address of the first data byte you want to edit (e.g., 10) and press ENTER. The default address to start editing at is 0. The current value of the data at the address displayed is shown on the second line of the display (e.g., 05).	`ENCRYP. ADDR = 10`
	`DATA 05 --> _ _`

5. To change the current data byte (as in the example), key in the new value and press ENTER. To leave the current data byte unchanged and advance to the next byte to be edited, press ENTER without keying in a new value.	`ENCRYP. ADDR = 10`
	`DATA 05 --> 9C`

Procedure	Example 212 Displays

6. After pressing ENTER, the next address will be displayed on the top line of the display and the current value of the data at that address will be displayed on the second line of the display. To edit the displayed data byte, repeat step 5. To exit the ENCRYPTION EDIT mode and return to the main menu, press either scroll key. If you key in a new value and then press a scroll key before pressing ENTER, the keyed in data will be ignored and the current data byte will remain unchanged.	`ENCRYP. ADDR = 11`
	`DATA 06 --> _ _`

Example



Signature Edit

Use the **Signature Edit** function for the microprocessor module only to edit the 2 bytes of data reserved in the programmer's RAM. Executing this command causes the programmer to enter a special edit mode which allows you to display and edit the data bytes.

Procedure	Example 212 Displays
1. Scroll to the EDIT function.	`EDIT`
2. Press ENTER.	`EDIT` `COMPLEMENT`

3. Scroll to the SIGNATURE EDIT function and press ENTER.	`SIGNATURE EDIT`
	`SIGNA. ADDR = 0`

4. Key in the address of the first data byte you want to edit (e.g., 0 or 1) and press ENTER. The default address to start editing at is 0. The current value of the data at the address displayed is shown on the second line of the display (e.g., 05).	`SIGNA. ADDR = 0`
	`DATA 05 --> _ _`

5. To change the current data byte (as in the example), key in the new value and press ENTER. To leave the current data byte unchanged and advance to the next byte to be edited, press ENTER without keying in a new value.	`SIGNA. ADDR = 0`
	`DATA 05 --> 9C`

Procedure	Example 212 Displays

6. After pressing ENTER, the next address will be displayed on the top line of the display and the current value of the data at that address will be displayed on the second line of the display. To edit the displayed data byte, repeat step 5. To exit the SIGNATURE EDIT mode and return to the main menu, press either scroll key. If you key in a new value and then press a scroll key before pressing ENTER, the keyed in data will be ignored and the current data byte will remain unchanged.	`SIGNA. ADDR = 1`
	`DATA 06 --> _ _`

Byte Swap Data

Use the **Byte Swap** function to swap the contents of adjacent memory byte pairs over a specified range. This is useful when programming 16-bit devices to properly assign the high/low order bytes in the 16-bit word.

Procedure	Example 212 Displays
1. Scroll to the EDIT function.	`EDIT`
2. Press ENTER.	`EDIT` `COMPLEMENT`

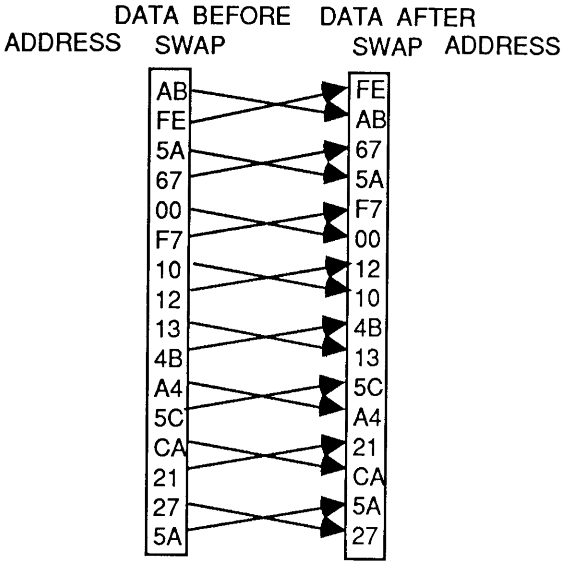
3. Scroll to the BYTE SWAP function (if necessary) and press ENTER.	`BYTE SWAP`
	`START ADD=000000`

4. Key in the first address of the range of data you want to swap and press ENTER. The default start address is 0.	`BYTE SWAP`
	`END ADD=03FFFF`

5. Key in the last address of the range of data you want to byte swap and press ENTER. The default end address is the end of memory (03FFFF for a programmer with 256K of memory).	`BYTE SWAP`
	`......`

When the byte swap is complete, the display will read	`BYTE SWAP`
	`COMPLETED`

Example


DATA BEFORE SWAP
ADDRESS
DATA AFTER SWAP
ADDRESS
AB
FE
5A
67
00
F7
10
12
13
4B
A4
5C
CA
21
27
5A
FE
AB
67
5A
F7
00
12
10
4B
13
5C
A4
21
CA
5A
27

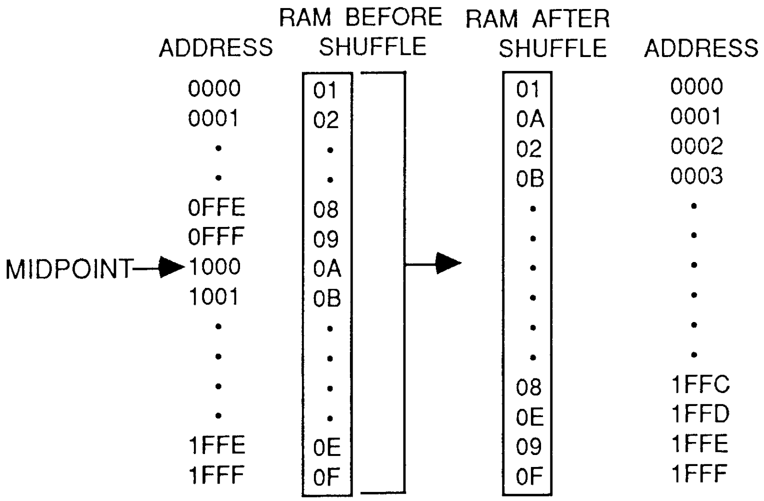
Shuffling Data from 8-bit-wide to 16-bit-wide Format

Use the **Shuffle** function to converge two adjacent blocks of 8-bit-wide data into one block of 16-bit wide data occupying the same block of RAM. This operation reverses the split operation, shuffling the two blocks of data together exactly the same way you would shuffle two halves of a deck of cards together (alternating data bytes of one block with data bytes of the other block). The data bytes of the block below a specified midpoint are placed in even-numbered addresses, starting with address 0, and the data bytes of the block at and above the specified midpoint are placed in odd-numbered addresses, starting with address 1. This command should be used to reunite 16-bit-wide data that was loaded into memory from two 8-bit-wide devices for subsequent uploading and editing on a development system or host computer in 16-bit-wide format. You can reverse the "Shuffle" function using the "Split" function.

Procedure	Example 212 Displays
1. Scroll to the EDIT function.	EDIT
2. Press ENTER.	EDIT COMPLEMENT
3. Scroll to the SHUFFLE function and press ENTER.	SHUFFLE MID ADR.=020000
4. Key in the midpoint of the data block to be shuffled (e.g., 1000). The first byte after the midpoint will be the byte placed into address 1. The midpoint must be a power of two between 2H and the RAM midpoint (20000H). The default midpoint is the midpoint of RAM.	SHUFFLE MID ADR.=1000

Procedure	Example 212 Displays
5. Press ENTER to execute the command.	
After the shuffle has been completed, the display will read	SHUFFLE • • • • • • SHUFFLE COMPLETED

Example



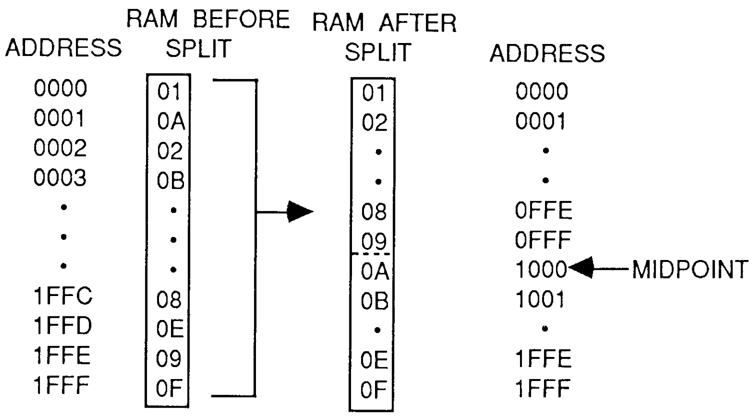
Splitting 16-bit-wide Data

Use the **Split** function to split a block of 16-bit-wide data into two adjacent 8-bit-wide blocks occupying the same block of programmer RAM. The odd-addressed bytes and even-addressed bytes are divided into two blocks around a specified midpoint, with the odd-addressed bytes starting at the midpoint and the even-addressed bytes starting at address 0 (see illustration). This command should be used when the data to be programmed into blank parts is downloaded from the development system in 16-bit-wide format and must be programmed into two 8-bit-wide parts that will be accessed as one 16-bit-wide word. You can reverse the Split function by using the Shuffle function.

Procedure	Example 212 Displays
1. Scroll to the EDIT function.	EDIT
2. Press ENTER.	EDIT COMPLEMENT
3. Scroll to the SPLIT function (if necessary) and press ENTER.	SPLIT MID ADR. =020000
4. Key in a midpoint around which the data will be split (e.g., 1000). The midpoint must be a power of two between 2H and the RAM midpoint (20000H for 256K RAM). The first odd-addressed byte of data will be moved to the address at the midpoint. The default midpoint is the mldpoint of RAM.	SPLIT MIDDLE ADR.=1000

Procedure	Example 212 Displays
5. Press ENTER to execute the command.	SPLIT . . . .
After the split has been completed, the display will read	SPLIT COMPLETED

Example


ADDRESS
RAM BEFORE SPLIT
RAM AFTER SPLIT
ADDRESS
0000
0001
0002
0003
1FFC
1FFD
1FFE
1FFF
01
0A
02
0B
08
0E
09
0F
01
02
08
09
0A
0B
0E
0F
0000
0001
0FFE
0FFF
1000
1001
1FFE
1FFF
MIDPOINT