Owner's Manual

SA-10 • SA-20

NEEDHAM'S ELECTRONICS, INC.
4630 Beloit Dr., Suite 20
Sacramento, CA 95838
(916) 924-8037

Warranty and Disclaimer

Warranty

Needham's Electronics warrants the SA-10 and SA-20 device programmers against defects in materials and workmanship for a period of two (2) years from the date of purchase.

If you discover a defect, Needham's Electronics will, at its option, repair, replace, or refund the purchase price of the programmer at no charge to you, provided that you return it with a copy of the bill of sale to Needham's Electronics within the warranty period.

This warranty does not apply if the programmer has been modified or has been damaged by accident, abuse, or misuse.

Disclaimer of Liability

Needham's Electronics is not responsible for special, incidental, or consequential damages resulting from any breach of warranty, or under any legal theory, including lost profits, downtime, goodwill, damage to or replacement of equipment and property, and any costs of recovering, reprogramming, or reproducing any program or data stored in or used with Needham's Electronics products.

Table of Contents

Table of Contents

Table of Contents

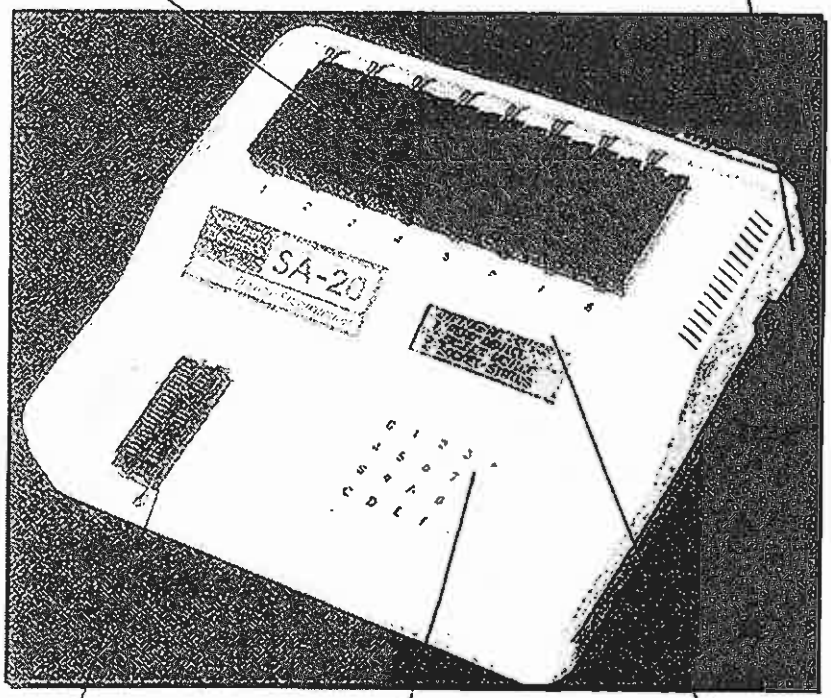
Chapter 1

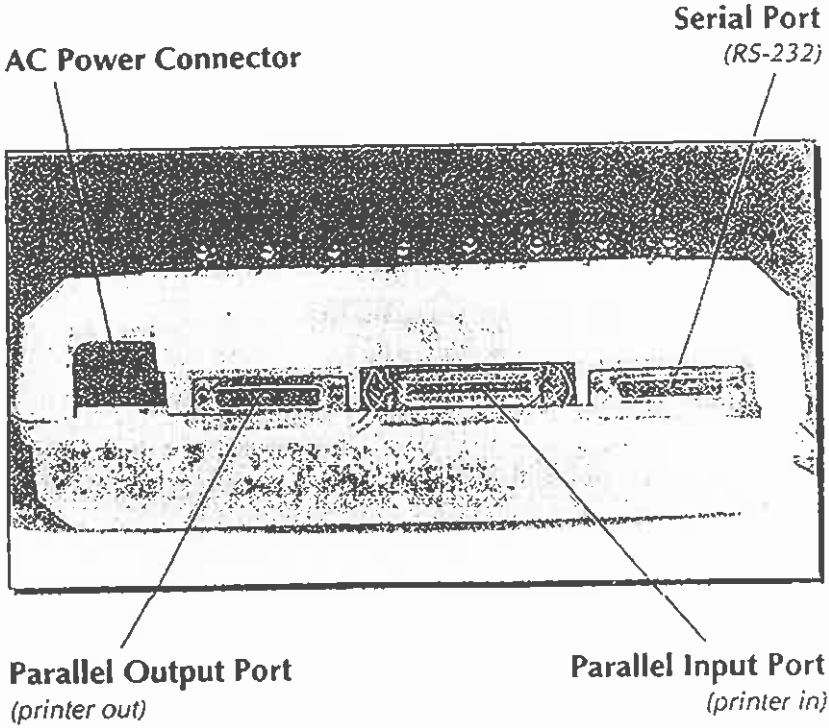
Tutorial

The tutorial chapter introduces you to the processes of reading and programming devices. It will demonstrate how to read and program devices in a simple, step-by-step manner, allowing you to make immediate use of your programmer.

Before starting the tutorial, however, the text describes the SA's basic controls, keypad, and display conventions.

In the following chapters, you will learn how to use your SA's additional features and intricacies.

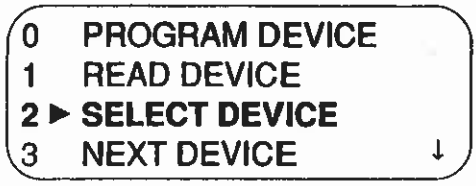

Sockets 1-8
(present on SA-20 only)
Power Switch
SA-20
Socket Zero
Keypad
LCD Display
(4 line x 20 character)


AC Power Connector
Serial Port
(RS-232)
Parallel Output Port
(printer out)
Parallel Input Port
(printer in)

Keypad & Display Conventions

Before we start the tutorial, you need to know several things about the keypad and display found on your SA-10 or SA-20:

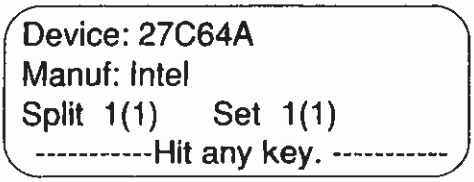
1) To choose an item from a menu, you may either use the ↑ and ↓ arrows or simply press the number(s) that correspond to the item. For instance, to choose Select Device from the Main menu, you could press ↓ twice and then press ↵, or you could just press the "2" key. Although pressing the corresponding number is faster, some people prefer using the arrow keys.



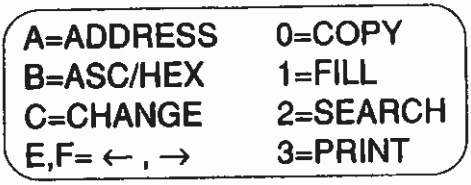
2) Press ↵ to indicate that you are making a selection or giving the okay to proceed with a process, such as programming.

3) To "escape" from the current display, press "*". Doing so will place you at the previous display or at the Main menu, depending upon where you were when you pressed "*".

4) a) Press "D" to see the current device settings. The display below shows the settings that you'll see:



4) b) "D" acts as a help key during certain processes. For instance, if you press "D" while in the editor, you will see the following help display:



These simple conventions are true throughout the SA's design, making use of the SA simple and consistent.

Now that you know these conventions, we are ready to start the tutorial.

Copying a Device

In the following tutorial, you will learn how to copy the data from one device onto a blank device. The process of copying devices involves all of the major functions – reading, programming, and verifying.

The tutorial uses two 27C64A EPROMs for the examples. One 27C64A should contain data to be copied; it will be referred to as the "original." The other 27C64A should be erased; it will be referred to as the "erased" 27C64A.

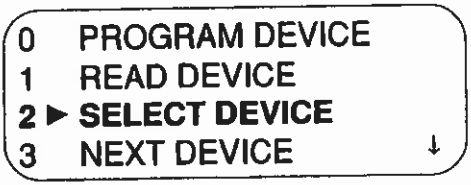
The 27C64A was chosen for the examples because it is common and because it supports the "A9 Identifier" method of selecting device type.

The illustrations below show how to insert devices into the sockets found on your SA-10 or SA-20:

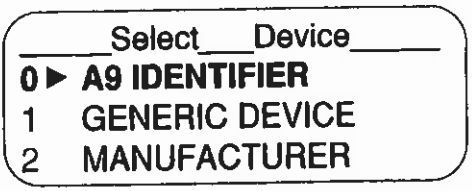
Device Insertion into Socket Zero

Device Insertion into Sockets 1-8

Before you can program a device, you must first select what type of device you will be programming. When you select the device type, your SA will determine the device's size, programming voltages, and manufacturer's programming algorithm. To select the device, choose ***Select Device*** from the Main menu:



Then, from the Select Device menu, you can choose one of three methods to select a device.



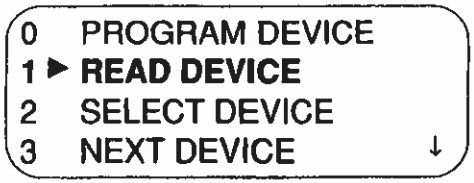
For the purpose of our tutorial, we will use the A9 identifier method. This method of selecting device type reads a binary code from the device that tells the SA the size, manufacturer, and programming algorithm of the device.

Insert the original 27C64A into socket zero (the socket to the left of the keypad) and choose ***A9 Identifier***.

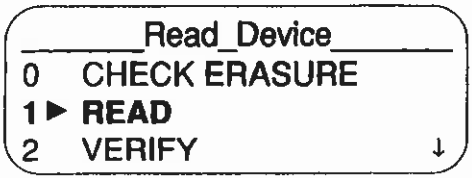
Note: If you have an SA-20, you may insert the 27C64A into any socket and the SA will find it, but you should not insert devices into both socket zero and another socket.

Reading a Device

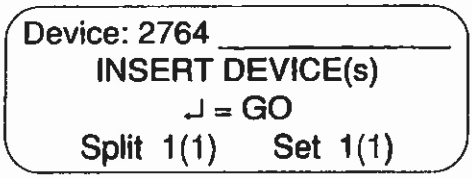
Once you have selected a device type, you can read your device into the SA's data buffer. To do so, choose ***Read Device*** from the Main menu:



Then choose ***Read*** from the Read Device menu:

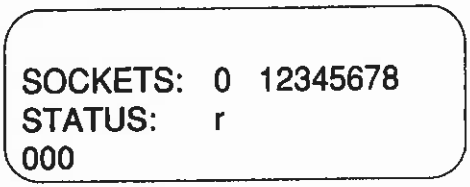


After choosing Read, you will see the display shown below:



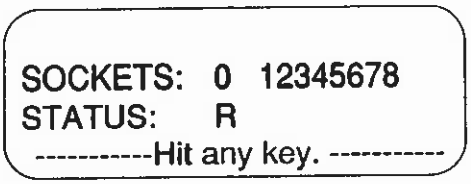
At this point, the SA is waiting for you to insert the device to read. Assuming you did not remove the original 27C64A from its socket, simply press ↵. The SA will read the contents of the original 27C64A into its buffer.

During the reading process, you will see the following display:



The "r" under "0" indicates that the SA is reading the device in socket zero. The three digit number in the lower left-hand corner is the page count; it shows how many pages have been read (a page is 256 bytes).

When the SA is finished reading, the above display will change to the following:



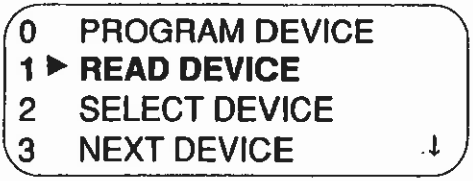
The uppercase "R" means that the SA is finished reading the device. If an error occurred during reading, a blinking "**X**" will be shown instead.

Assuming the SA read your 27C64A without a problem, press ↵ to return to the Main menu.

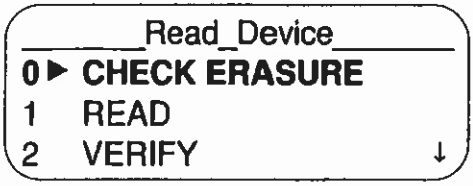
Checking Erasure

Now that we have read the contents of the original 27C64A into the data buffer, we are ready to make the copy. Remove the original 27C64A from its socket and replace it with the erased 27C64A.

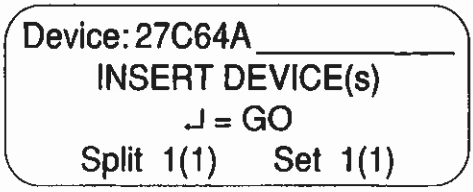
Before attempting to program a device, you should verify erasure. To check erasure, choose ***Read Device*** from the Main menu:



Then choose ***Check Erasure*** from the Read Device menu:

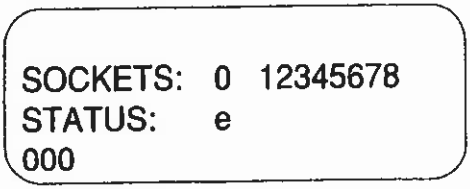


After choosing Check Erasure, you will see the display shown below:



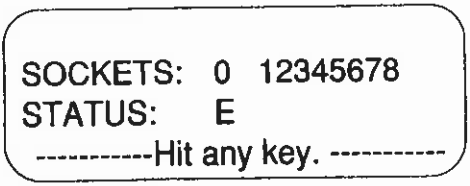
At this point, the SA is waiting for you to insert the device to check. Press ↵ to start the process.

During the checking process, you will see the following display:



The "e" under "0" indicates that the SA is checking the device in socket zero. The three digit number in the lower left-hand corner is the page count; it shows how many pages have been checked (a page is 256 bytes).

When the SA is finished checking, the above display will change to the following:

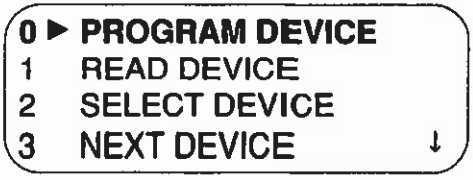


The uppercase "E" means that the device is erased. If the device was not completely erased, a blinking "X" will be shown instead.

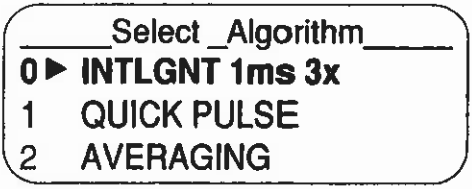
Press ↵ to return to the Main menu.

Now that we have determined that the 27C64A is indeed erased, we can program it with the data from the SA's buffer.

To program the 27C64A, choose ***Program Device*** from the Main menu:



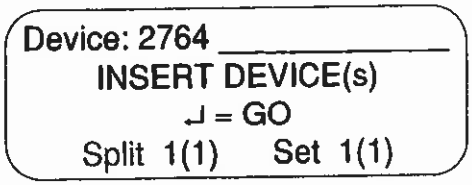
After choosing Program Device, you'll see a list of programming algorithms for the device:



The algorithms above were true for the particular 27C64A that we used when writing this tutorial, but may change with other device manufacturers and device types (we used a Signetics 27C64A-25). The first algorithm is the algorithm recommended by the device manufacturer. The other algorithms provide faster programming and are especially useful in prototyping applications.

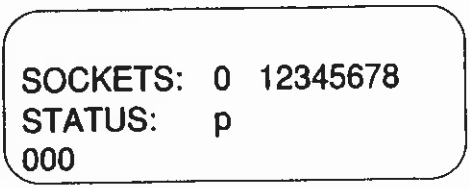
For the purpose of this tutorial, choose whatever algorithm is listed first.

After choosing the algorithm, you will see the display shown below:



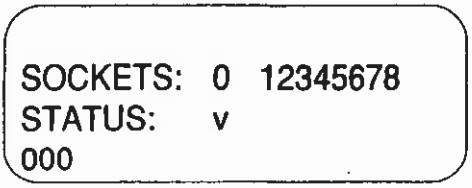
At this point, the SA is waiting for you to insert the device to be programmed. Assuming you did not remove the erased 27C64A from its socket, simply press ↵. The SA will program the device with the data read from the original 27C64A.

While the 27C64A is being programmed, you will see the following display:

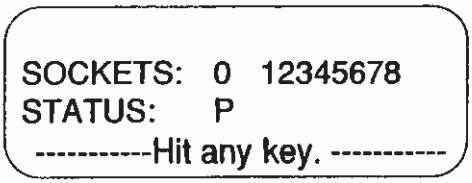


The "p" under "0" indicates that the SA is programming the device in socket zero. As before, the three digit number in the lower left-hand corner is the page count; it shows how many pages have been programmed (a page is 256 bytes).

When the SA is finished programming, it will verify that the data in the device equals the data in its buffer:



The "v" under "0" indicates that the SA is verifying the device in socket zero. If the device passes verification, you will see the following display:



The uppercase "P" means that the device was programmed successfully. If programming was not successful, a blinking "X" will be shown instead.

Assuming your 27C64A was programmed successfully, press ↵ to return to the Main menu. If it did not pass, erase the device or use another erased 27C64A. Then, repeat the verify, erasure, and programming portions of the tutorial.

This supplement is designed to address some common questions regarding use of the SA-10/20 Programmer. It gives a brief overview of the programmer in stand-alone and PC controlled operation.

Upon power up the first menu will display company name, current software version and the amount of memory installed. After this display a series of audible tones will be heard while the Programmer attempts to initialize communication with a PC through the SA.EXE communication software. If the programmer cannot initialize communication it will default to a stand-alone mode and all operations must be performed from the keypad of the Programmer.

In stand-alone mode the LCD will show a 4 line window in the current menu. Arrows on the right side of the display will indicate if there are additional menu options above or below the current window being displayed. The star key functions like ESC. For example, you would use the star key to escape back to the previous menu level. The right angle arrow key functions as an Enter or Ret. Menu selections can be made by either pressing the appropriate menu number or by arrowing up or down to the desired selection and pressing Enter.

A common misconception with regards to the SA-20 Programmer is the relationship between socket 0 and sockets 1-8. During an operation such as reading or programming the Programmer scans the sockets to locate a device. If a device is found in socket 0 the Programmer will ignore sockets 1-8. If no device is found in socket 0 the Programmer will continue to

scan the upper sockets consecutively, until a device is found. Therefore it is possible to read the "master" eprom into the buffer from any socket on the Programmer. Once a "master" file has been created in the buffer you can program as many blank devices as are desired. It is not necessary to define how many devices are installed in the upper sockets. (See manual for Split/Set programming)

The floppy disk inside the manual contains the PC communications software. The file is labeled SA.EXE. Parallel and serial communications are supported. The amount of time to transfer data using parallel or serial at 115.2K baud is roughly the same. The program defaults to parallel communication. A standard centronics parallel printer cable can be used to communicate via parallel. Attach the printer end of the cable to the PRINTER IN connector at the back of the Programmer. All pins on the parallel cable are necessary for communication. It is possible to have a parallel cable that does not work with the Programmer but works well with a printer. All pins on the cable must be functioning properly to allow communications.

For serial communications you must first change the default setting in the SA.EXE program. To do this press F3 and then ESC. Attach the serial cable to the SERIAL IN port on the back of the Programmer. Please note that all serial cables are not the same. The communication software for serial uses pins 2, 3, and 7 of the serial port. If the serial port on your PC is a DB25 pins 2 and 3 run straight through. If the serial port is DB9 pins 2 and 3 will have to be swapped within the cable itself.

Once your cable is installed and the correct communication mode is set in the software, the SA10/20 upon power up should automatically initialize communication. If the Programmer was in stand-alone mode it is necessary to reinitialize communication. You can do this by powering down and back up or press 8 from the main menu on the Programmer keypad. It is not necessary to modify the File I/O parameters in the config menu if the SA-10/20 communication software is being used.

Differences between PC I/O ports may make it necessary to modify the software timing delay in the communication software to, achieve smooth bi-directional communication. Indication of insufficient delay are incomplete menus on the PC, incorrect correspondence of the cursor on the screen or the LCD or no communication at all. Too high a software delay may result in additional characters on the screen after the menus are drawn.

To modify the timing delay press F4. The default setting is 0000. It is recommended that this number be increased by increments of 100 until solid communication is achieved. It is necessary to reinitialize communication with the Programmer each time the delay is modified. Escape from F4 menu and press 8 on the SA keypad. Once optimum communication is achieved, return to F4 menu and press S on the PC keyboard to save this configuration.

To transfer files, without using the PC communication software, it is necessary to set the communication parameters in the I/O portion of the Configuration

menu. Refer to the manual contents for the pins used in serial transfer. The Programmer must be able to control the host computer via pin 6 of the SA serial port. It is recommended that all files transferred, without using the communication software, be in Intel hex or Motorola S format to provide adequate error checking.

Chapter 2

Programming

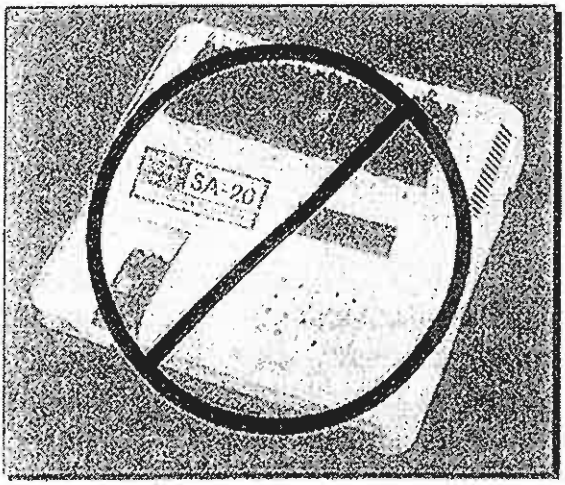
This chapter explains how to use the various programming features of your SA-10 or SA-20. Your SA is capable of programming many devices, including EEPROMs, Flash EEPROMs, and EPROMs ranging from the common 2716 to the (4 Megabit 27C040 or 574000). In addition to programming many parts, the SA can program these parts in sets and splits.

Even with its extensive capabilities, however, the SA is simple and intuitive to use.

Device Placement

When programming devices, certain rules and circumstances need to be considered when placing devices in the SA's ZIF sockets.

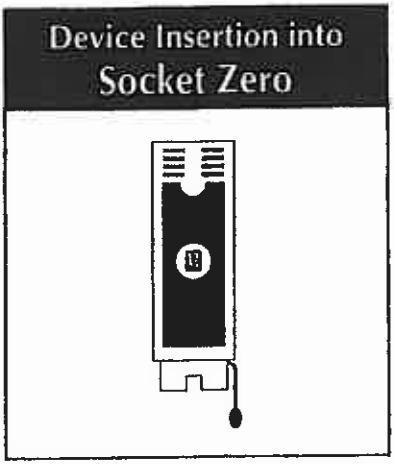
Most importantly, you should not place devices in both socket zero (the socket to the left of the keypad) and sockets 1-8. Doing so will not damage the parts, but the SA will only access socket zero, possibly giving erronious results.



Do not place devices in both socket zero and sockets 1-8

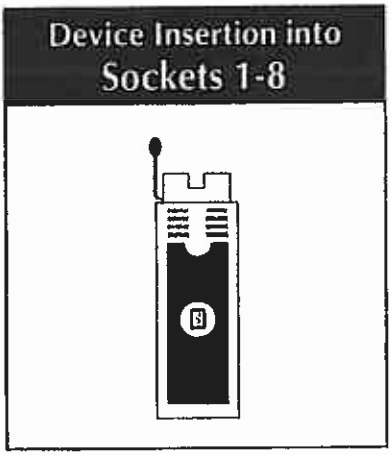
The following pages show several examples of the relationship between programming and device placement. The last example describes what to do if one device in a group fails during programming.

The following examples show programming sets and splits. If you are unfamiliar with sets and splits, refer to the next chapter, Data Buffer, for an explanation of each.



Socket zero is located to the left of the keypad and is present on both the SA-10 and SA-20. Socket zero's handle points toward the operator, which may be confusing if you have an SA-20.

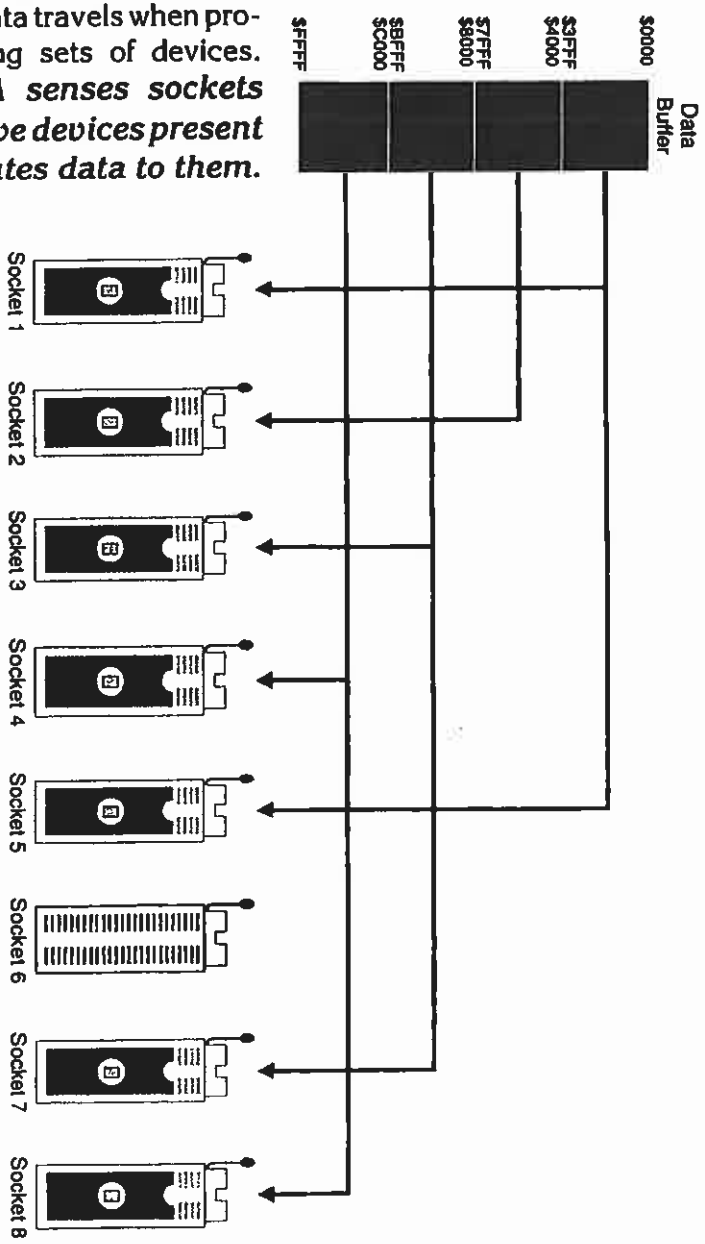
The SA-20 has eight addtional sockets, the handles of which point away from the operator.



To ensure that devices are inserted properly, just remember that they should always face "notch up," toward the back of the SA. In addition, devices should be placed in lowest possible position in the socket, closest to you.

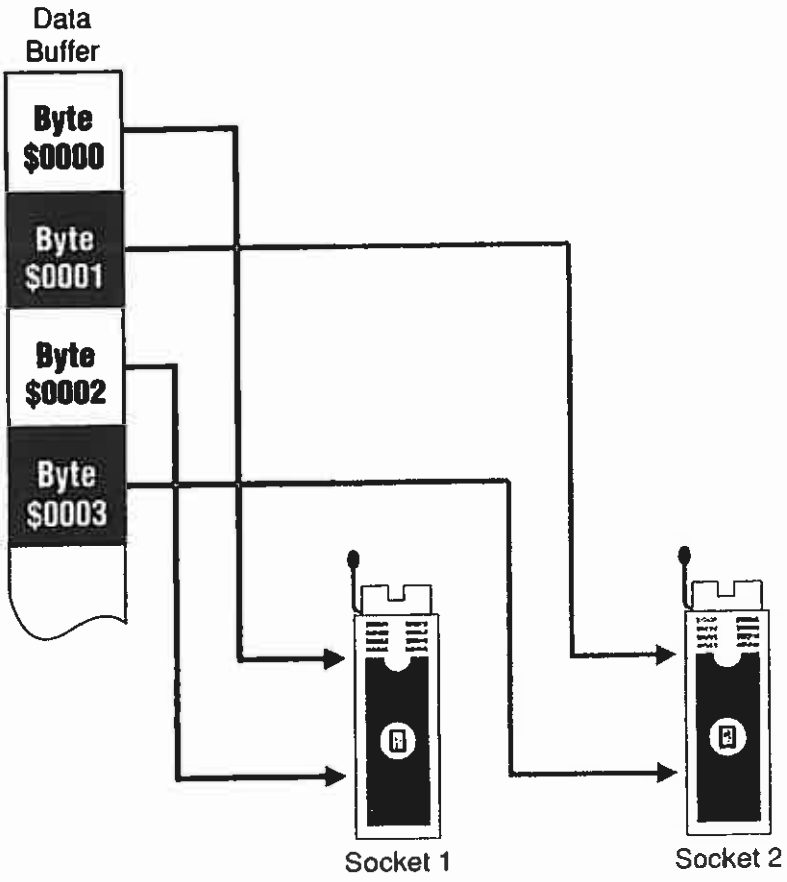
Programming Sets

This diagram illustrates where data travels when programming sets of devices. ***The SA senses sockets that have devices present and routes data to them.***



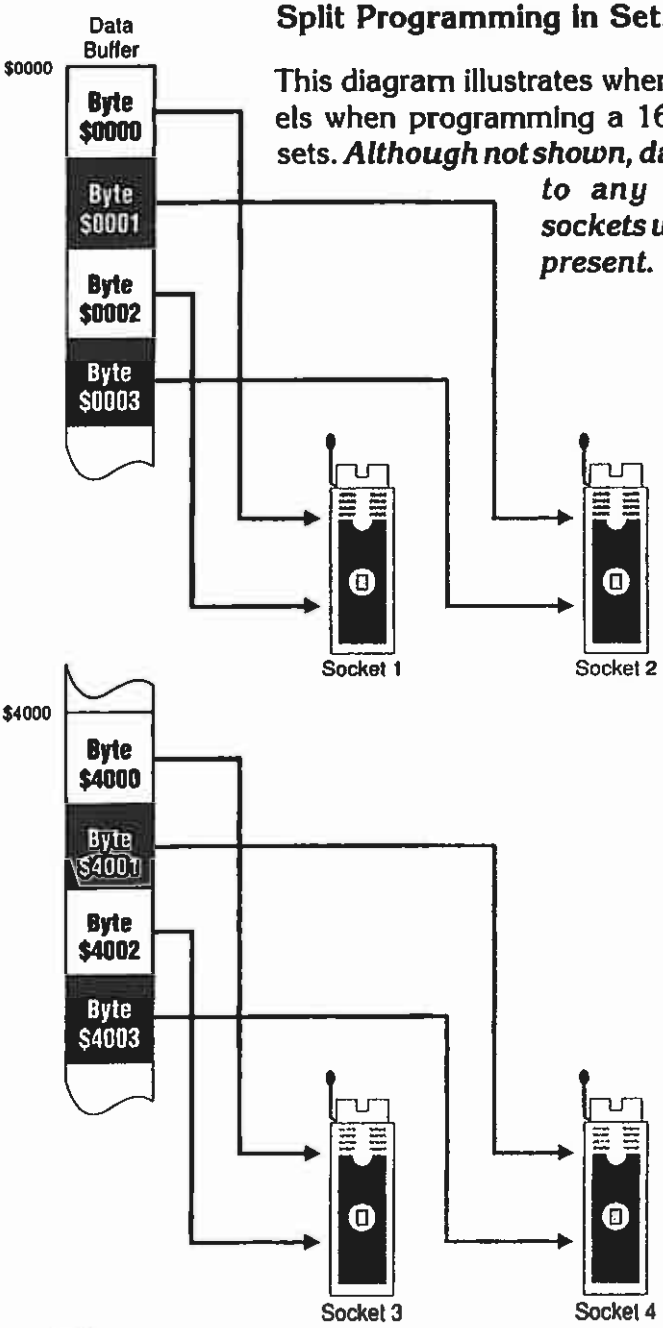
Split Programming - 16 Bit

This diagram illustrates where data travels when programming a 16 bit split. ***Although not shown, data is routed to any additional sockets with devices present.***



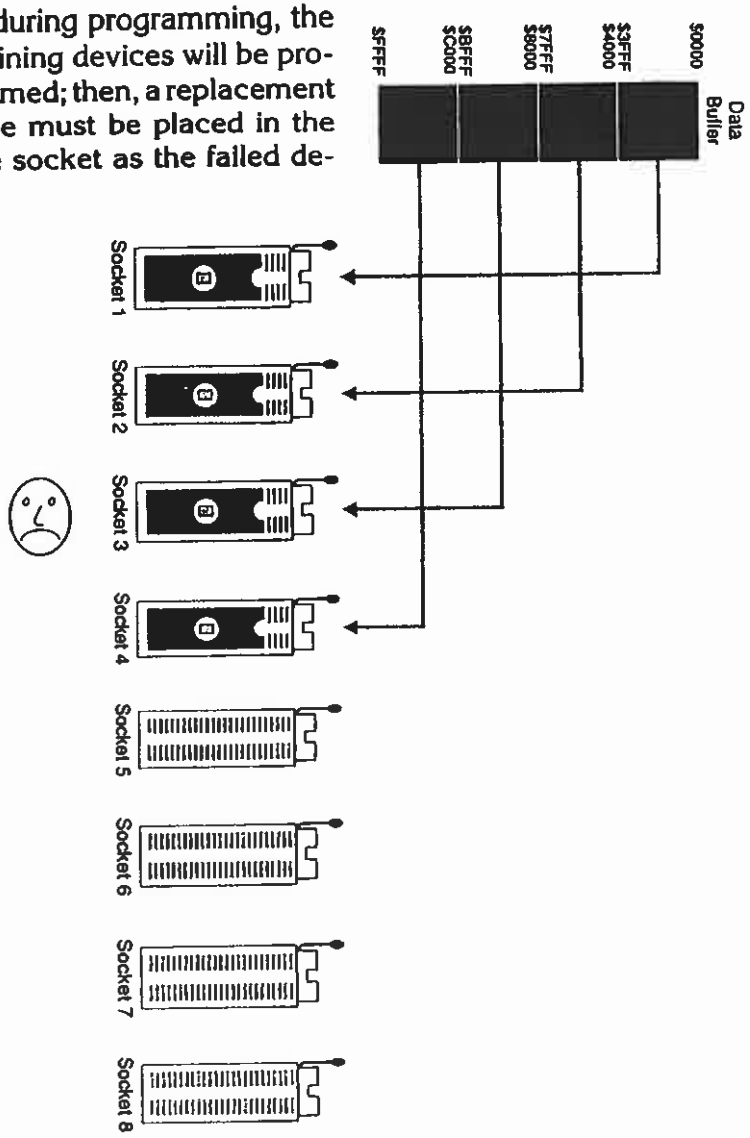
Split Programming in Sets

This diagram illustrates where data travels when programming a 16 bit split in sets. ***Although not shown, data is routed to any additional sockets with devices present.***



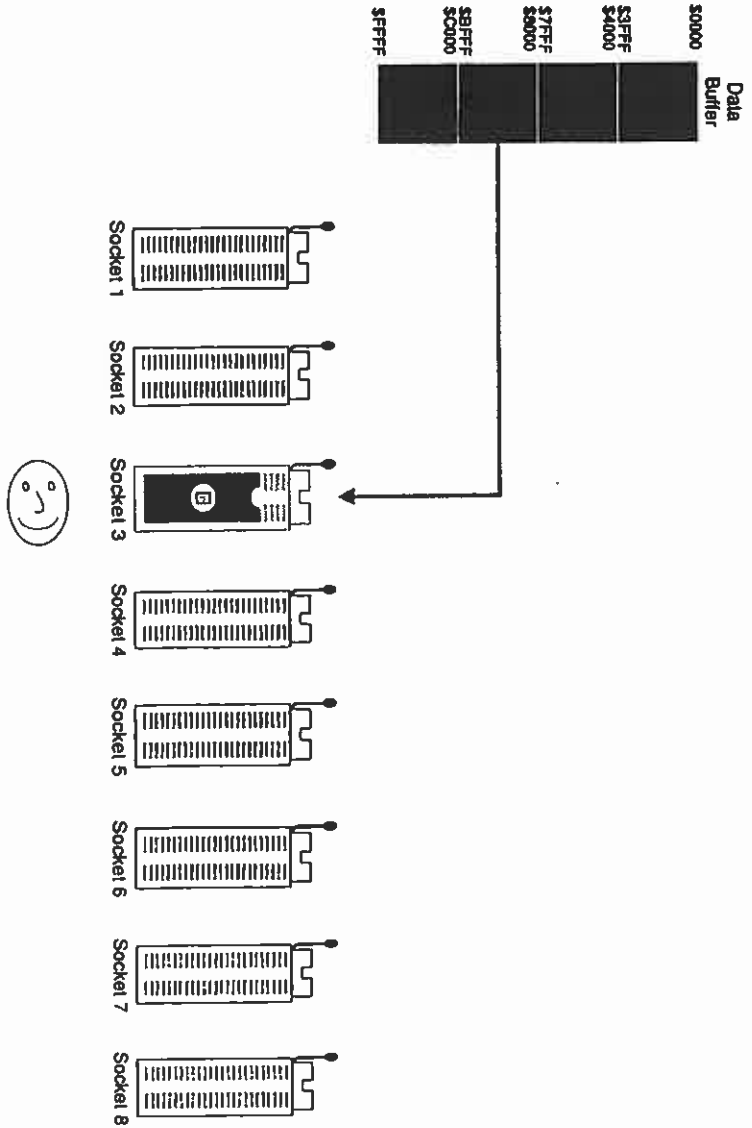
Re-programming Part of a Set or Split

As illustrated in these diagrams, data is routed to sockets in sequential order (the first block of data goes to the first socket, the second block to the second, etc.). If a device in a group fails during programming, the remaining devices will be programmed; then, a replacement device must be placed in the same socket as the failed device.

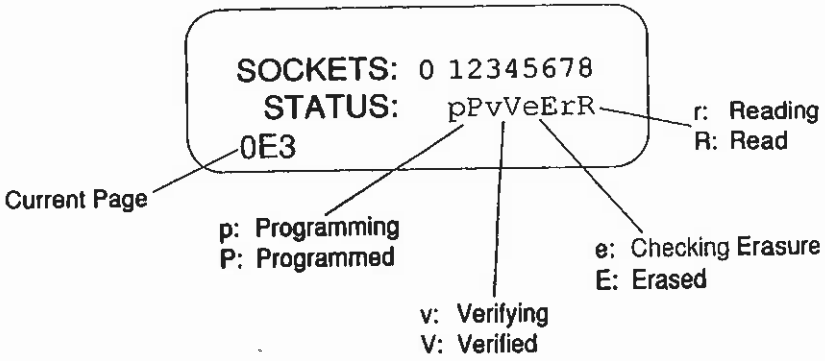


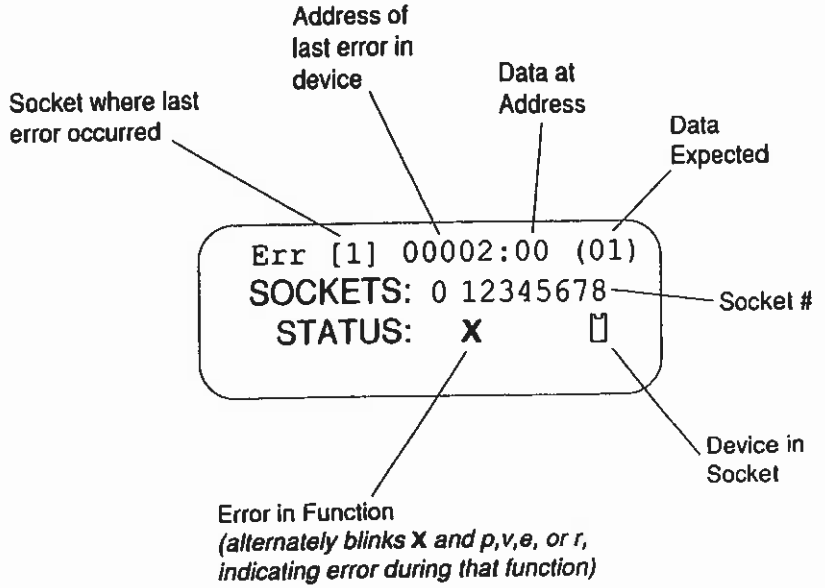
Re-programming Part of a Set or Split

With the device in the proper socket, simply repeat the programming process. The SA will ignore empty sockets, thereby programming only the replacement device.



When programming devices, you'll see various symbols and displays that indicate the status of each function. The following text shows the various symbols and their meanings:





r: Reading device in corresponding socket
R: Device read

e: Checking erasure of device in corresponding socket
E: Device erased

v: Verifying that data in device equals data in SA's buffer
V: Verified

p: Programming device with data from SA's buffer
P: Device programmed

Current Page:
Indicates current page of operation. The page is the address without the lower eight bits; therefore, each page represents 256 bytes
(Example: E3B7 = page E3)

Socket where last error occurred:
Number of socket holding device where last error occurred

Address of last error in device:
Address of last error in device indicated

Data at Address:
Data found at address of last error

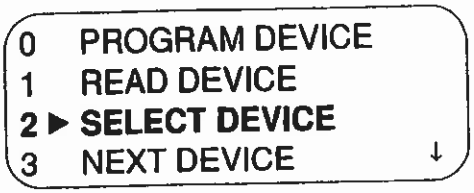
Data Expected:
Data expected at address of last error
(during verify erase, erase value is expected)
(during all other operations, corresponding data from SA's buffer is expected)

Socket #:
Socket number *(0 is single socket)*

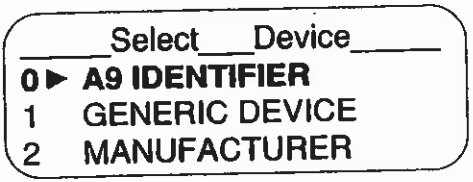
Device in Socket:
Device symbol indicated device present

Error in Function:
When an error occurs during a function (programming, verifying, checking erasure, or reading), the SA will alternatley blink an **X** and the letter corresponding to the function in which the error occurred

Before you can program a device, you must first select what type of device you will be programming. When you select the device type, your SA will determine the device's size, programming voltages, and manufacturer's programming algorithm. To select the device, choose ***Select Device*** from the Main menu:



Then, from the Select Device menu, you can choose one of three methods to select a device.

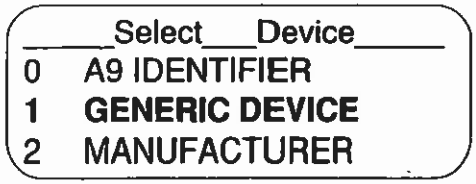


The first and easiest method is by checking the device's *A9 Identifier*, which tells the SA the size, manufacturer, and programming algorithm of the deivce. To use the A9 identifier, insert your device into the SA's socket and choose ***A9 Identifier***. If you have an SA-20 and insert more than one device, the SA will check the first device, ignoring any devices inserted in higher-numbered sockets. We recommend that you avoid inserting different device types, as doing so may result in damage to the devices different than the identified device *(make sure that all devices inserted are of the same manufacturer and type)*.

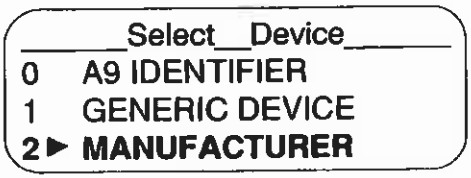
On a technical level, the A9 identifier method allows the reading out of a binary code from a device that identifies the device manufacturer and type. The code is read out by applying 12V to address line A9 of the device. The A9 identifier method works with most devices designed since 1984. In addition, most devices that have the suffix A (2764A, etc.) or that are 256K bits or larger will work.

If the code returned by the device is not in the SA's table of codes, the SA will not be able to determine the device size and programming algorithm.

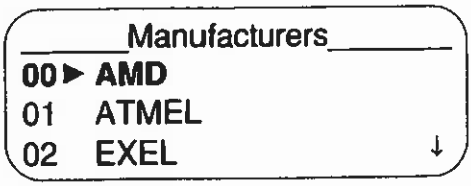
If the SA can not use the A9 identifier method because the device does not support it or because the SA does not know the device's code, you will need to use one of the two methods described below.



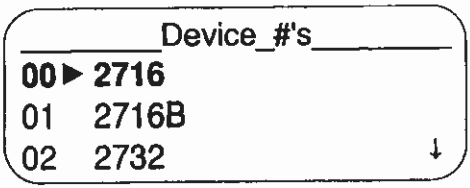
The second method of selecting a device is to choose ***Generic Device*** from the Select Device menu. You will see a list of generic device numbers (2716, 27128, etc.) from which to choose. Once you choose a device number, the SA will know the size, programming voltage, and programming algorithm for the device. However, the voltage and algorithm may differ slightly from the manufacturer's specifications.



The third method of selecting a device is to choose ***Manufacturer***. When you choose manufacturer, you will see a list of manufacturers as shown below:

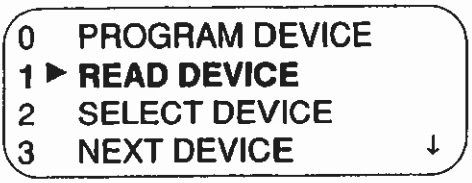


After choosing the manufacturer, you will see a list of device numbers:

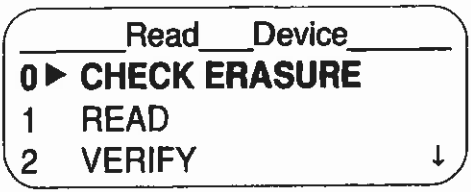


Once you choose a device number, the SA will know the size, programming voltage, and programming algorithm specified by the manufacturer.

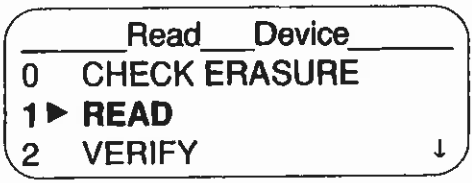
After you have completed one of the device selection methods, you can read or program the devices.



Once you have selected a device type, you can read your device into the SA's data buffer. Choose ***Read Device*** from the Main menu.



To check whether or not a device is erased, choose ***Check Erasure*** from the Read Device menu. Check Erasure checks for the device's erase value ($FF for most devices). *On differential cell technology, erasure may be undeterminable.*

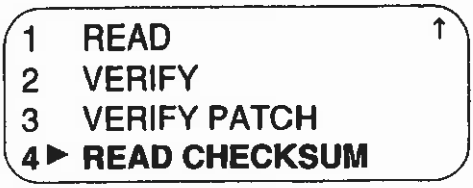


To read a device, choose ***Read***. This reads the device into the SA's data buffer. If set and/or split options are selected, the data will be copied into the buffer according to these options (see Data Buffer for an explanation of set and split).

To verify that a device equals the SA's data buffer, choose ***Verify***. This compares the contents of the device to the contents of the SA's data buffer. If set and/or split options are selected, the data will be compared according to these options. *A verify is automatically performed after programming.*

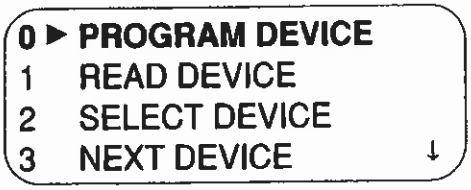
With most devices, an erased bit (usually a 1) can be changed to a programmed bit (usually a 0) without first erasing the device. Programming this way is called patching.

To find if patching is possible, choose ***Verify Patch***. The SA will compare each bit in its buffer to the corresponding bit in the device. If the contents of the device can be changed to equal the contents of the SA's buffer by simply programming certain bits, then patching is possible. If just one bit needs to be erased, patching cannot be done.

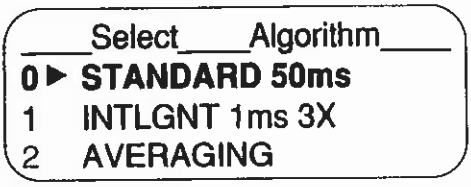


To verify that the data in a device is correct, a checksum is sometimes used. A checksum is the sum of all the data in the device. For instance, if you had a device with five locations and the locations had the values 1, 2, 3, 4, 5, then the checksum for the device would be $0F (1+2+3+4+5).

To determine the checksum for a device, choose ***Read Checksum***.



To program a device, choose ***Program Device*** from the Main menu.



After choosing Program Device, you will see a list of programming algorithms for the device. The first algorithm is the algorithm recommended by the device manufacturer. Any algorithms listed after the first one are not specifically recommended by the manufacturer, but are useful for faster programming in prototype situations. The relative speeds of each algorithm are shown below:

Quick Pulse	1X
Averaging	4X
Intelligent (1ms 2X)	30X
Intelligent (1ms 3X)	40X
Intelligent (1ms 4X)	50X
Standard 50ms	500X

Once you have chosen a programming algorithm, the SA will ask you to insert the device and press ↵ to start programming.

If set and/or split options are selected, the data will be copied into the buffer according to these options.

Chapter 3

Data Buffer

The data buffer is one of the most important parts of your SA device programmer. The buffer acts as the "middle man" between your source data and the device to be programmed. Data is transferred to the buffer from one of several sources, usually a pre-programmed device or personal computer, and then programmed to the device. While the process is simple, there are various programming modes and pointers that may need to be adjusted to fit your particular application.

This chapter is divided into three sections, each of which describes a different aspect of the data buffer.

The first section introduces the theory of the data buffer. Theory includes such things as buffer size and illustrations of set and split programming.

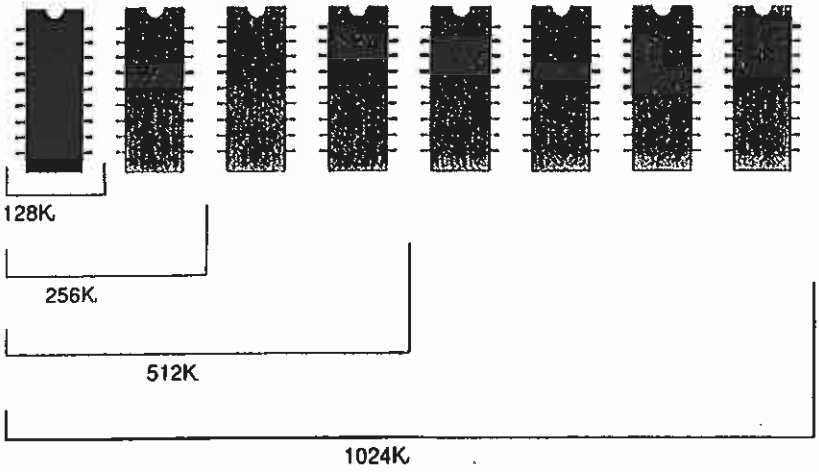
The second section explains how to implement the functions described in theory in the first section. Specific menu choices and displays are included in this section.

The third section explains how to send and receive data to and from a PC.

Data Buffer

Section 1: Theory

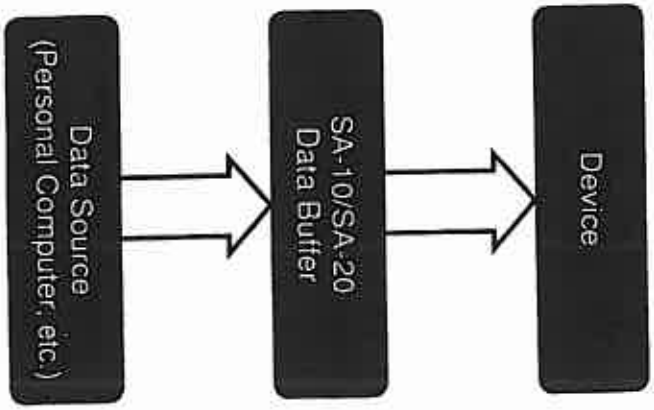
Memory Configurations



Your SA-10/SA-20 was shipped with 128K of RAM for use as the data buffer, enough for most applications. However, by simply adding memory chips, it is possible to have up to 1 megabytes of buffer space. Added buffer space provides faster programming and easier downloading of large amounts of data.

The diagram above shows the configurations available. The first number in each category is true when using 1 megabit chips.

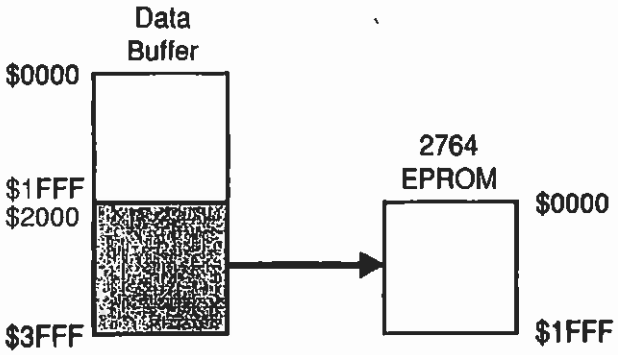
Please see the Upgrade section for information on installing additional memory.



When downloading data to be programmed onto a device, the data follows the path shown above.

The data originates from a source (usually a personal computer), is then downloaded to the SA's buffer, and is finally transferred to the device being programmed.

It is important to note that no device may be programmed without the data going through the buffer first. For instance, you cannot copy data directly from one device to another; instead, you must read the first device into the buffer, then program the second device from the buffer.



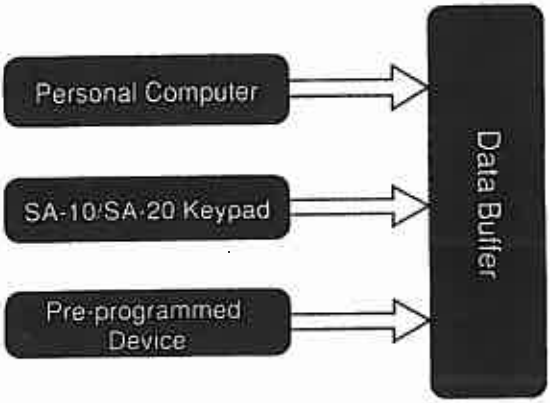
When programming a device, you may have more data in the buffer than will fit in the device. When this happens, you need to set pointers that tell the SA what portion of your data buffer to program to the device.

The drawing below shows the five pointers used in programming:

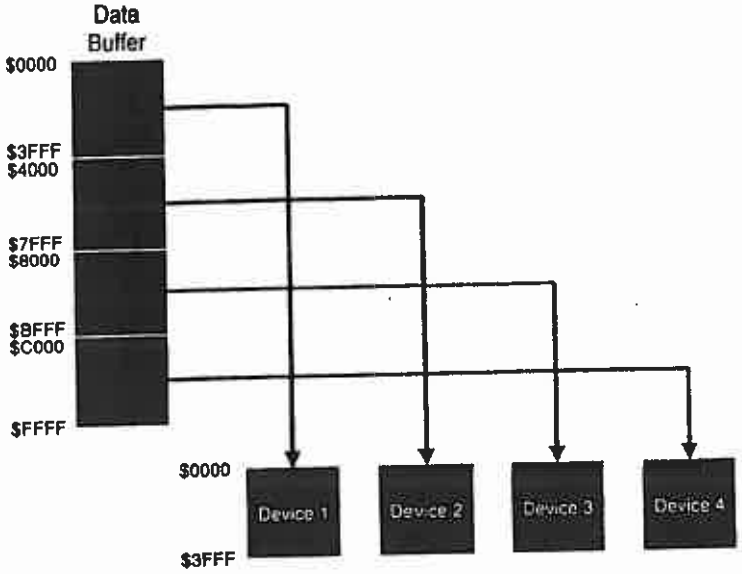
```
_____Set_Pointers_____
BUFFER:02000,03FFF
___DEVICE:00000,01FFF___
LENGTH:02000
```

In the example above, the second half of 16 kilobytes of data is being programmed onto a 2764 EPROM, a device that can only hold 8 kilobytes.

When you change a pointer setting, all other pointers change accordingly. In the example above, if you changed the length to 01000, the buffer end pointer would change to 02FFF and the device end pointer would change to 00FFF.

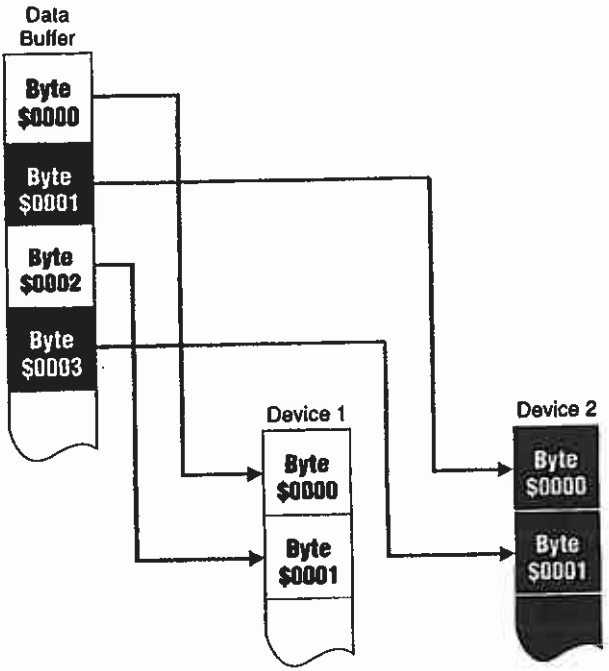


Three sources can download data to the buffer or change data already in the buffer. As stated earlier, a personal computer is the most common source. However, data can be generated or modified from the SA's built-in keypad or read from an previously programmed device.



Occasionally, you will need to separate your data into smaller sections for programming. The resultant group of programmed devices is called a set.

In the above example, we have 64 kilobytes of data being programmed onto a set of four 16 kilobyte devices.

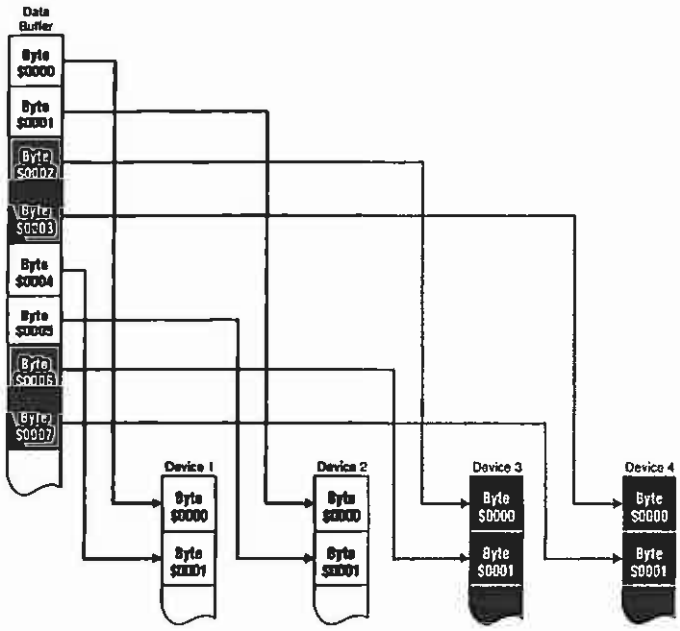


If you need to program devices for use in a 16 bit environment, you will need to split your data onto two devices.

When you perform a 16 bit split, the first device is programmed with all data having an even address (0,2,4,...) and the second device is programmed with all data having an odd address (1,3,5,...).

When used in a 16 bit system, the two devices act as one, since they share common address and enable lines. When enabled, data from the same locations in each device combines to form a 16 bit word.

Note that when using a 16 bit split, the devices being programmed need only be half the size of the original data. This is true because only half of your data is programmed onto each device.

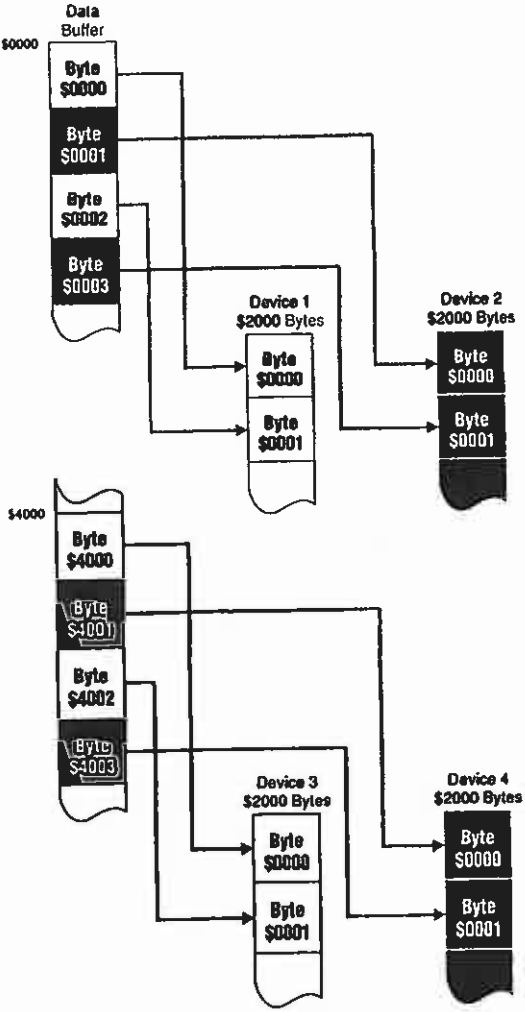


If you need to program devices for use in a 32 bit environment, you will need to split your data onto four devices.

When you perform a 32 bit split, the first device is programmed with the first of every four bytes (0,4,8,...), the second device with the second of every four bytes (1,5,9,...), the third device with the third of every four bytes (2,6,10,...), and the fourth device with the fourth of every four bytes (3,7,11,...).

When used in a 32 bit system, the four devices act as one. When enabled, data from the same locations in each device combines to form a 32 bit longword.

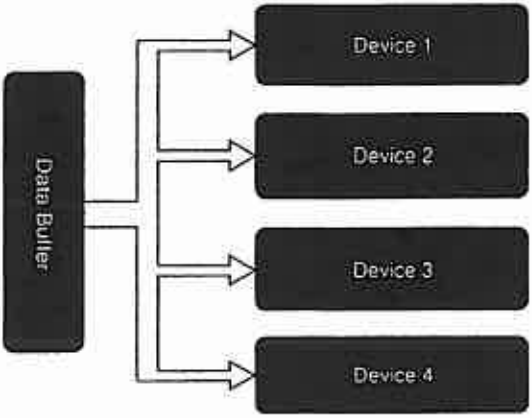
Note that when using a 32 bit split, the devices being programmed need only be one-fourth the size of the original data. This is true because only one-fourth of your data is programmed onto each device.



If you need to program devices for use in a 16 bit environment, but your devices are too small to hold the split data, then you will need to use a split set approach.

As with a normal 16 bit split, all even bytes are programmed onto the first device and all odd bytes are programmed onto the second device. However, once the first set of devices is full, the process continues with additional sets, until all data is transferred.

A 32 bit split set is also possible, with the only difference being that each set is comprised of four devices rather than two.



If you purchased the SA-20, you have the added capability of programming up to eight devices.

Multiple programming makes set and split programming more convenient and less time consuming.

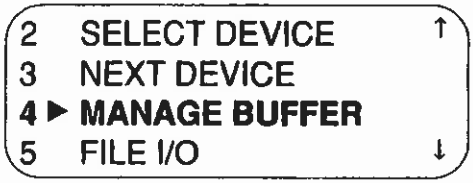
In addition, multiple programming allows you to simultaneously program many devices with the same data. This is particularly useful in production and large prototyping applications.

Section 2: Menus and Displays

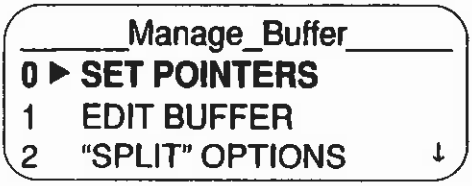
Setting Programming Pointers

When programming a device, you may have more data in the buffer than will fit in the device. When this happens, you need to set pointers that tell the SA what portion of your data buffer to program to the device.

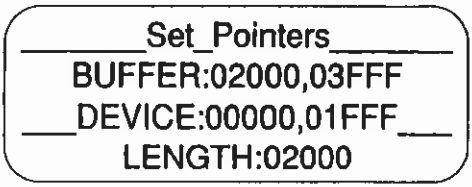
To set the pointers, first choose ***Manage Buffer*** from the Main menu:



Then choose ***Set Pointers*** from the Manage Buffer menu:

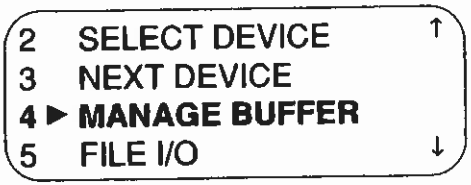


You will then see the pointers shown below:

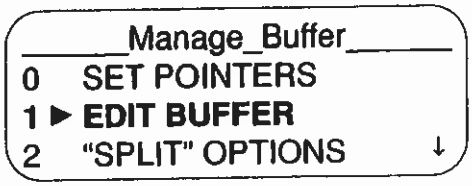


The figures shown are the same ones used in the Theory section of this chapter: the second half of 16 kilobytes of data is being programmed onto a 2764 EPROM, a device that can only hold 8 kilobytes.

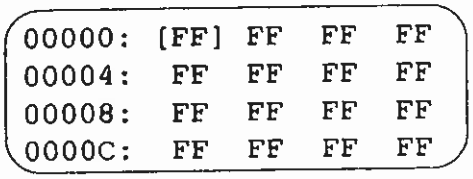
Your SA allows you to directly access the data stored in its buffer for manual editing. To enter the editor, choose ***Manage Buffer*** from the Main menu:



Then choose ***Edit Buffer*** from the Manage Buffer menu:



After choosing Edit Buffer, you will be in the editor:



The editor allows you to edit on a byte-by-byte level or on a global level. Byte level editing allows you to enter individual bytes as ASCII or hex values; global editing functions allow you to copy, fill, or search any portion of the buffer.

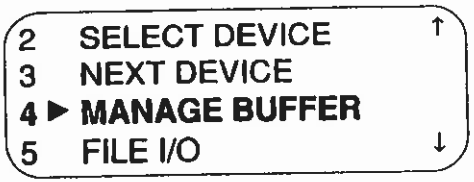
The listing on the next page shows the keys used to operate the functions of the editor.

While in the editor, the following keys may be used for the indicated purposes:

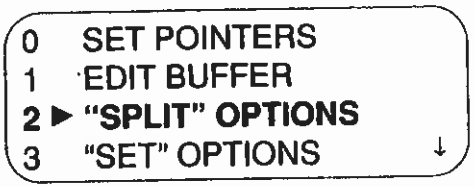
- A Edit Address (editor displays data starting at specified address)
- B Switch display between ASCII and hex
- C Change byte at cursor to new value
- E Move cursor left one space
- F Move cursor right one space
- 0 Copy portion of buffer to another location in buffer
- 1 Fill portion of buffer with specific values
- 2 Search portion of buffer for specific values
- 3 Print portion of buffer

Pressing "D" while in the editor will display a help menu that shows the above keys.

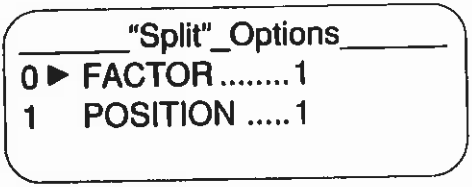
To change the settings for split programming, first choose ***Manage Buffer*** from the Main menu:



Then choose ***"Split" Options*** from the Manage Buffer menu:



After choosing "Split" Options, you will see this display:



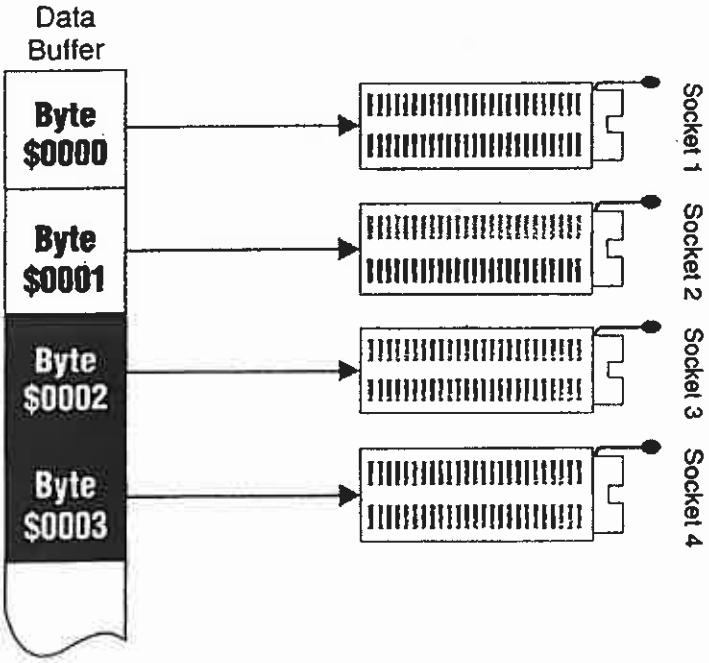
The Factor parameter is the ratio of bytes stored to bytes skipped. A factor of 1 indicates no split, all data in the buffer will go to one device. A factor of 2 indicates a 16 bit split, all even bytes will go to one device and all odd bytes will go to another. A factor of 4 indicates a 32 bit split. See to the Theory section of this chapter for illustrations of split programming.

The second value, Position, is explained on the next page.

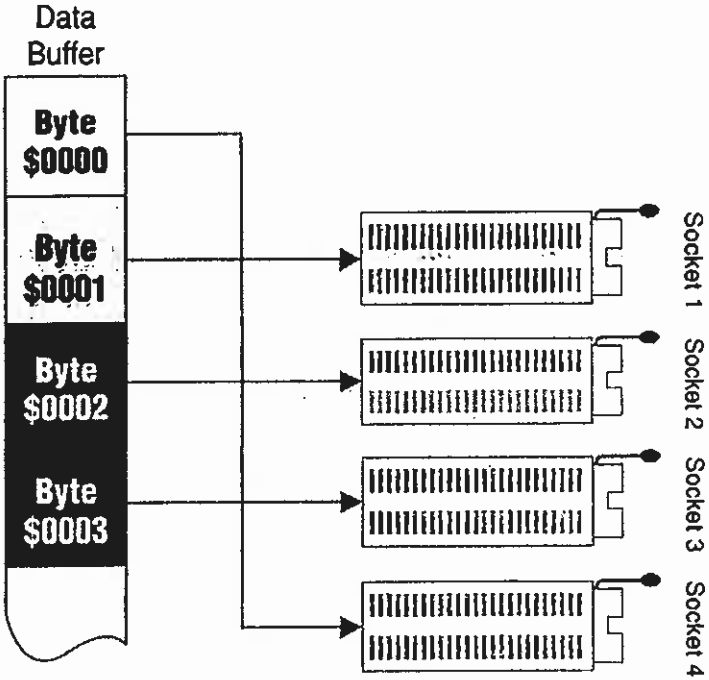
The Position setting determines which SA socket will receive the first portion of a split, which will receive the second, etc. In most cases, you will want socket 1 to receive the first portion, so you will leave Position set at 1. However, if you already have the first device programmed, you may want to change Position so that the first portion of data is skipped and directed to the last socket.

The illustrations below and on the following page show the relationship between Position and data:

32 Bit Split
Position 1 (normal setting)



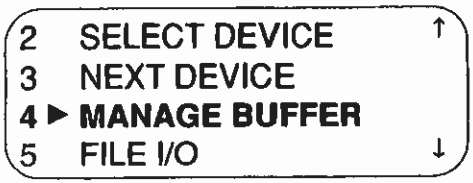
32 Bit Split
Position 2



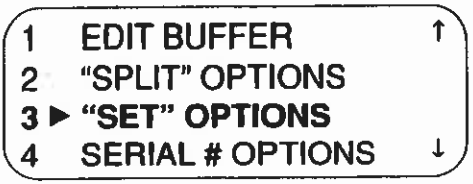
This and the previous diagram illustrate the SA's ability to begin programming at any selected position. This is useful for set and split programming applications.

Changing Set Settings

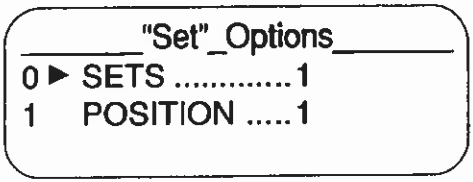
To change the settings for set programming, first choose ***Manage Buffer*** from the Main menu:



Then choose ***"Set" Options*** from the Manage Buffer menu:



After choosing "Split" Options, you'll see this display:



The Sets parameter tells the SA how many devices you want to use to accomplish set programming. For instance, if you needed to program 16K of data onto 2K devices, you would specify 8 sets.

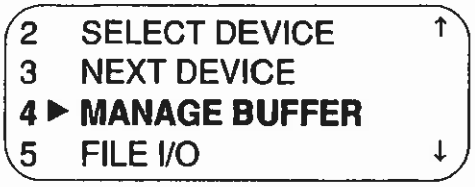
Since the SA knows the capacity of each device, it will simply program as much data as it can onto each, updating the appropriate programming pointers as it does so.

The second parameter under Set Options is Position. This setting determines which SA socket will receive the first portion of data, which will receive the second, etc. In most cases, you will want socket 1 to receive the first portion, so you'll leave Position set at 1. However, if you already have the first device programmed, you may want to change Position so that the first portion of data is skipped and directed to the last socket.

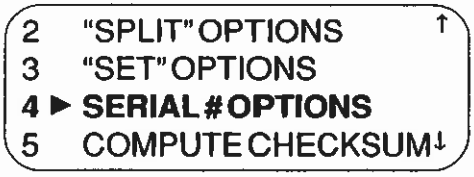
See the previous pages on Changing Split Settings for illustrations of different Position settings.

Serial Number Options

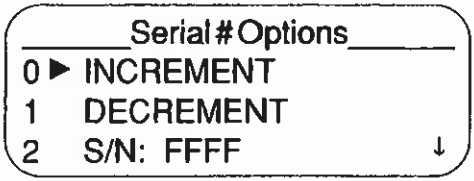
Your SA can perform several functions to aide in serializing devices to be programmed. To get to the serial number options display, first choose ***Manage Buffer*** from the Main menu:



Then choose ***Serial # Options*** from the Manage Buffer menu:



After choosing Serial # Options, you will see this display:



The directions on the next page describe how to use these options to add serial numbers to devices being programmed.

To assign serial numbers to devices, follow these steps:

1) Choose ***Address*** from the Serial # Options menu. You will then be able to enter a buffer address. The four bytes starting at this address form the serial number.

2) After entering an address, you will be taken back to the Serial # Options menu. The S/N line will display the serial number at the address given in step 1. If you want a different serial number, choose ***S/N*** from the menu and enter the new value.

3) Press '*' twice to return to the Main menu, then program the first device.

4) After programming, choose ***Manage Buffer*** from the Main menu and then ***Serial # Options*** from the Manage Buffer menu. This will return you to the Serial # Options display.

5) Choose ***Increment*** or ***Decrement***. Doing so will increment or decrement the value stored at the serial number address.

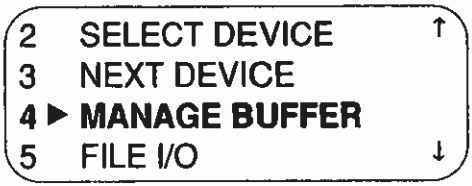
6) Repeat steps 3-5 for each device to be given a serial number.

While the process above may seem somewhat tedious, it goes quickly.

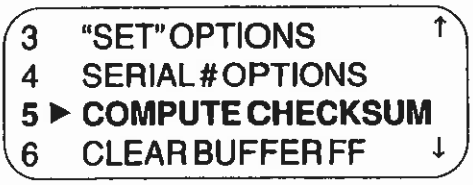
To make serialization easier, you may want to write a macro to perform the above operations (*see Chapter 4 for information on macros*).

Occasionally, you will find the need to calculate a checksum of the data in the SA's buffer. Checksums are usually used as a means of ensuring accuracy; the individual bytes in a device should always add up to the same value if the device was programmed correctly.

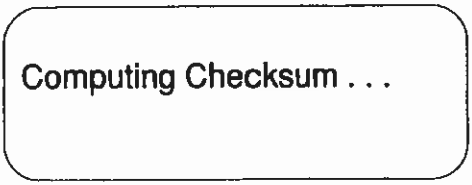
To compute the checksum of the data in the buffer, first choose ***Manage Buffer*** from the Main menu:



Then choose ***Compute Checksum*** from the Manage Buffer menu:

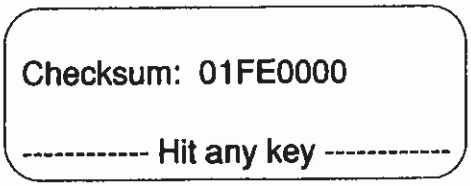


After choosing Compute Checksum, you will see this display:



The SA will compute the checksum for the portion of the data buffer specified by the programming pointers *(see Setting Programming Pointers at the beginning of this section)*.

After several moments, the display will show the checksum:



Pressing any key will return you to the Manage Buffer menu.

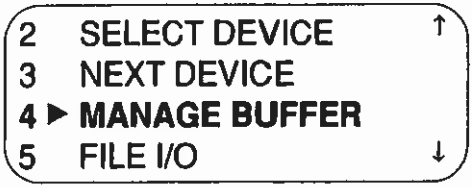
The SA can also compute the checksum for device(s) in the SA sockets without modifying the buffer contents.

To compute the checksum for device(s) in the SA socket(s) first choose ***Read*** from the main menu, then choose read ***Checksum.***

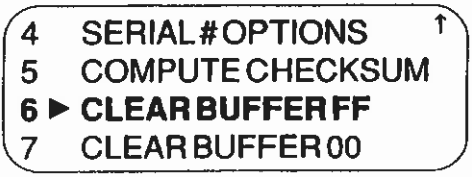
Clearing the Buffer

The last two functions of the Manage Buffer menu are Clear Buffer FF and Clear Buffer 00. As their names imply, they clear the buffer to FF or 00. However, these functions clear the *entire* buffer, not just the portion designated by programming pointers, so be careful when using these functions.

To access these functions, first choose ***Manage Buffer*** from the Main menu:



Then choose ***Clear Buffer FF*** or ***Clear Buffer 00*** from the Manage Buffer menu:



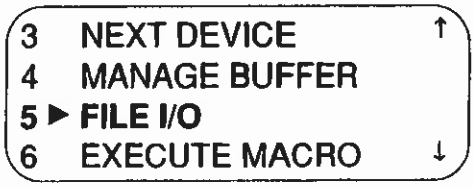
Section 3: File I/O with a PC

Communication with a PC

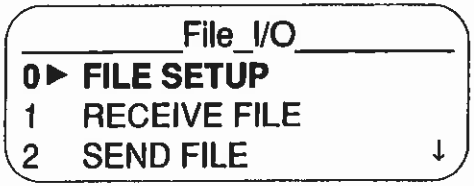
When sending or receiving data to or from a PC, you will need to set several parameters if any of the following applies:

1) Data to be downloaded from PC is larger than the SA's buffer
2) Only part of the data will be sent or received
3) Only part of the SA's buffer will be used to receive data

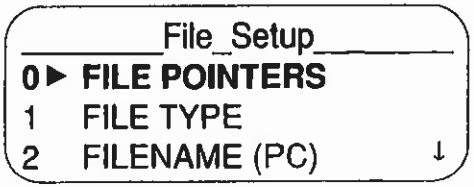
The three parameters associated with communication are Buffer pointers, File pointers, and Shuffle. To access these parameters, choose ***File I/O*** from the Main menu:



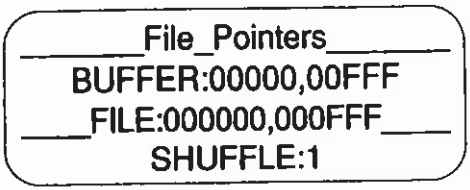
Then choose ***File Setup*** from the File I/O menu:



Finally, choose ***File Pointers*** from the File Setup menu:



After choosing File Pointers, you will see the display shown below:



When sending, the Buffer pointers determine which part of the SA's buffer will be sent. When receiving, the Buffer pointers determine which part of the SA's buffer will be used to store incoming data from the PC. In the above example, 4K of buffer space has been set aside.

When sending, the File pointers dictate the address stored with the data received by the PC. When receiving, the File pointers determine which part of the file will be downloaded.

Shuffle is the ratio of bytes stored to bytes skipped. A shuffle of 1 stores all bytes, a shuffle of 2 stores every other byte, etc. *(think of shuffle as the opposite of split)*. Shuffle is useful when the following circumstances are true:

1) Data to download is too large to fit in the SA's buffer

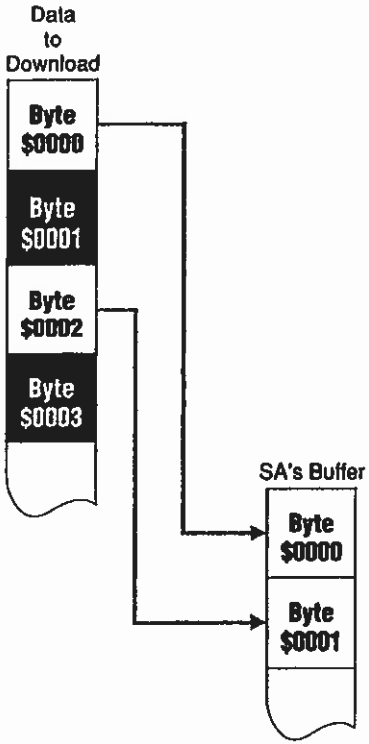
2) Data will be used in a 16 or 32 bit environment, making split programming necessary. Since all of the data cannot be downloaded into the buffer, however, all devices cannot be programmed at once.

When these circumstances are true, shuffle allows you to first download the data for the first device, then the data for the second device, etc.

Note that shuffle has no affect when sending data to the PC.

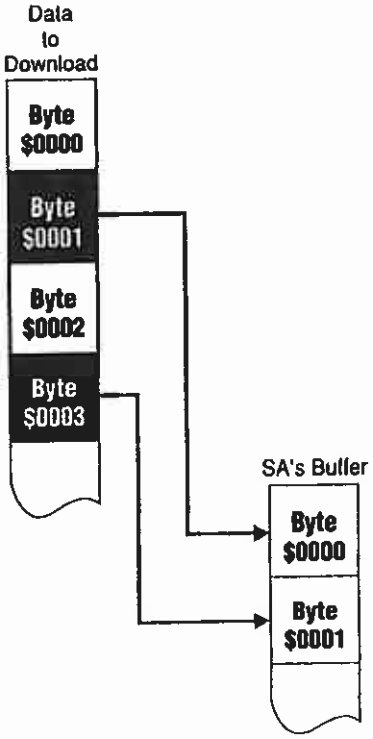
In the example below, 2,048K of data is being downloaded from the PC into a 1,024K buffer. Because shuffle is set to 2, the SA will only store the even bytes, thereby taking only 1 megabyte of buffer space.

```
File_Pointers
BUFFER:00000,7FFFF
FILE:000000,0FFFFF
SHUFFLE:2
```



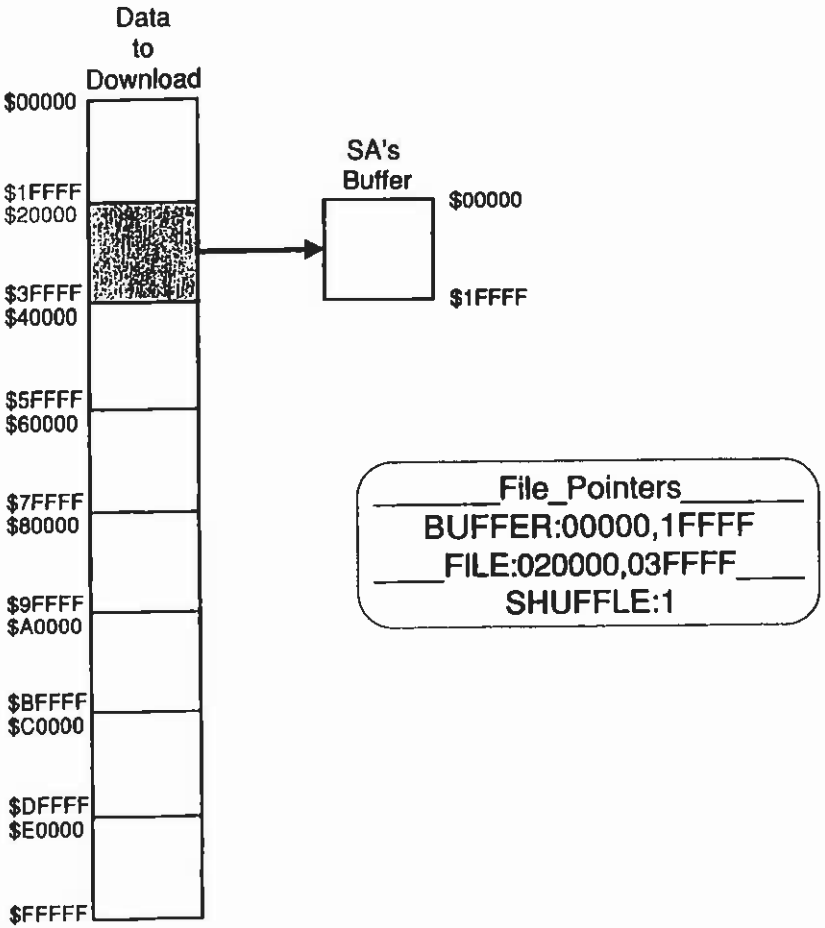
In the example below, 2,048K of data is being downloaded into a 1,024K buffer. Like the example on the previous page, shuffle is set to 2. However, the file start pointer has been changed to 000001, meaning that the SA will store the odd bytes rather than the even bytes.

```
_____File_Pointers_____
BUFFER:00000,7FFFF
____FILE:000001,0FFFFF____
SHUFFLE:2
```

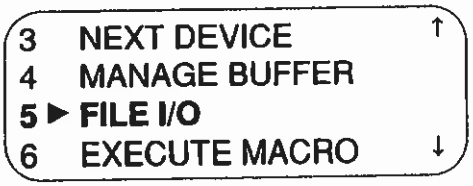


Partial Downloading

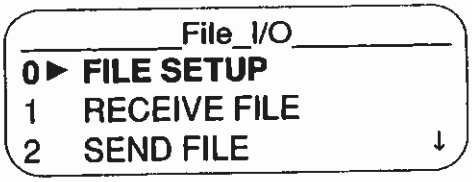
The other common use of the File Pointers is to designate part of a file to be downloaded. For instance, if you have 1,024K of data to download, but you only have 128K of SA buffer space, you'll need to download the data in eight separate parts. In the example below, the pointers are set to download the second eighth of the data.



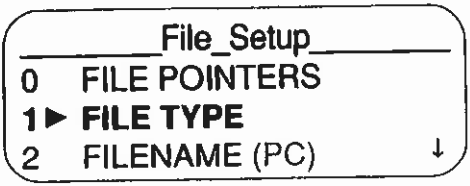
When sending or receiving data, you'll need to specify the proper file type. To set the file type, first choose ***File I/O*** from the Main menu:



Then choose ***File Setup*** from the File I/O menu:

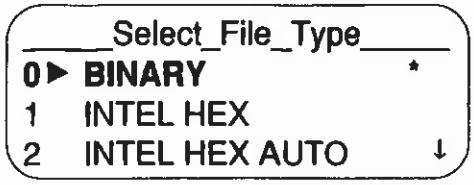


Finally, choose ***File Type*** from the File Setup menu:



After choosing File Type, you will see the display shown on the next page.

Selecting File Type



To select a file type, use the ↑ and ↓ keys to scroll through the display, then press ↵ when you have found the file type you want. As always, the "*" indicates the current setting.

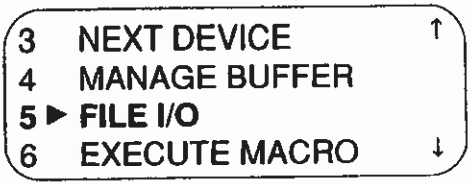
The file types available are:

- Binary
- Intel Hex
- Intel Hex Auto
- Motorola S
- Motorola S Auto
- Hex (ASCII)

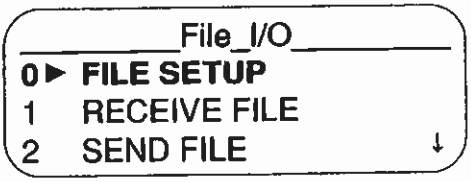
Intel Hex and Motorola S file types include an address in the data that specifies where the data is to be placed when loaded into a system. This address may cause problems when the data is downloaded to the SA, however, since the address may specify a location not within the range of a particular device to be programmed.

If this problem is suspected, use the Intel Hex and Motorola S "auto" file types. These function just as their normal counterparts, except that they perform address range limiting. If an address is specified that's too large, it simply "wraps around" to the beginning of the device. For instance, if an address is specified that's one byte beyond the end of the device, the data will be stored in the first location of the device. To prevent "wrap around," be sure to set the File Pointers for an area at least the size of the data to download, if not larger.

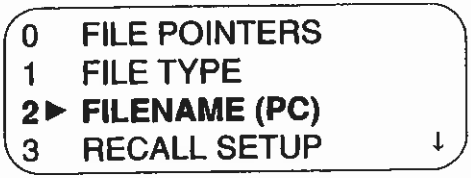
When sending or receiving data, you'll need to specify the filename used by the PC. To set the filename, first choose ***File I/O*** from the Main menu:



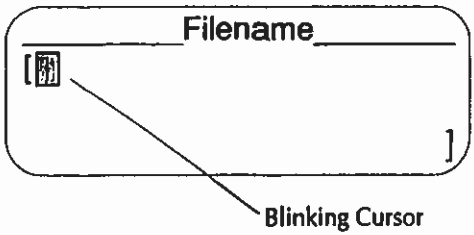
Then choose ***File Setup*** from the File I/O menu:



Finally, choose ***Filename (PC)*** from the File Setup menu:



After choosing Filename, you will see the display shown on the next page.



This display allows you to enter the filename.

To select the desired letter, number, or symbol, use the ↑ and ↓ keys. When the cursor shows the desired character, press "F" to move to the next space.

While entering the filename, the following keys may be used for the indicated purposes:

0 Insert
1 Delete

E Move Left
F Move Right

↑ Increment Character
↓ Decrement Character

↵ Press when finished with name

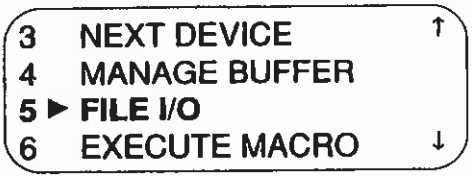
D Display help menu

Pressing "D" while editing the filename will display a help menu that shows the above keys.

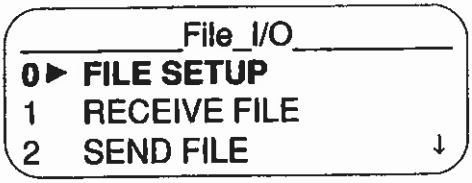
When finished entering the filename, press ↵.

Your SA can save up to ten separate File I/O Setup records, making file I/O much easier. File I/O records contain such information as file pointers, file type, and filename.

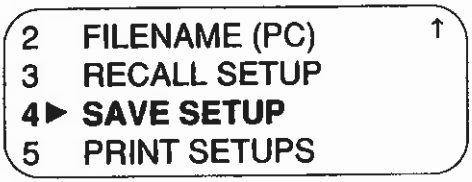
To save the current settings, first choose ***File I/O*** from the Main menu:



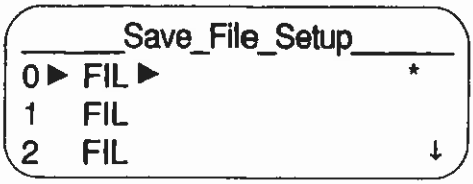
Then choose ***File Setup*** from the File I/O menu:



Finally, choose ***Save Setup*** from the File Setup menu:



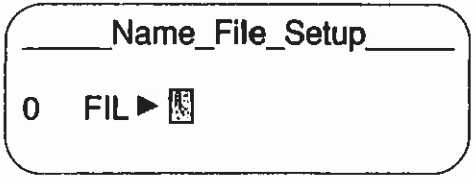
After choosing Save Setup, you will see the display shown on the next page.



This display is used to designate where the current file I/O settings will reside in the record. If all ten "slots" are already occupied, then you will have to overwrite one of them.

As with other displays, the "*" indicates the currently selected slot. To select another slot for the current settings, use the ↑ and ↓ keys to scroll through the display, then press ↵ when you've found the slot you want. If you press ↵ while at a slot that's already occupied, the new settings will replace the old.

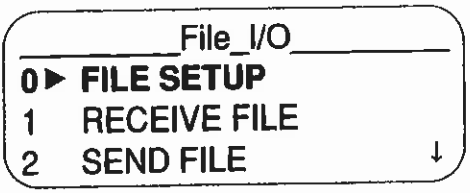
After designating one of the ten available "slots" for the new file I/O settings, you will be asked to name the new record:



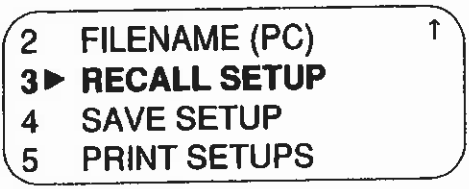
If the new settings are replacing old ones, then the old settings' name will appear as the default name. Pressing ↵ will leave the name as it appears.

The same keys are used to edit the name as are used to edit the PC filename (see page 3-33).

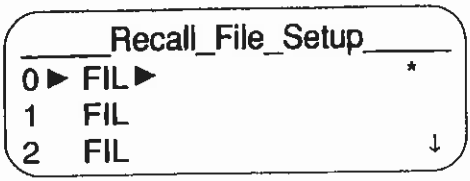
To recall a previously saved File I/O record, choose ***File Setup*** from the File I/O menu:



Then, choose ***Recall Setup*** from the File Setup menu:



After choosing Recall Setup, you will see the display shown on the next page.

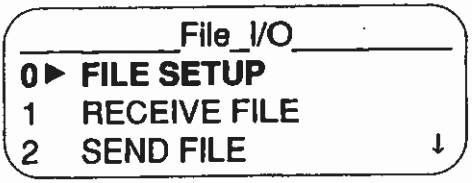


To select a setup record, use the ↑ and ↓ keys to scroll through the display, then press ↵ when you've found the record to recall.

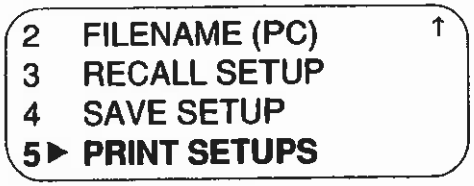
Once recalled, the setups in the record become the current file I/O settings.

Printing File I/O Setups

If you have a printer connected to your SA, you can print your file I/O setup records for easy reference. To print the setups, choose ***File Setup*** from the File I/O menu:

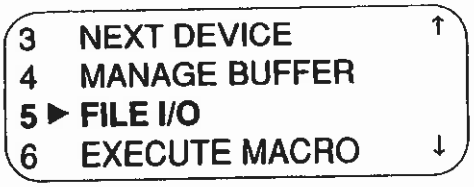


Then, choose ***Print Setups*** from the File Setup menu:

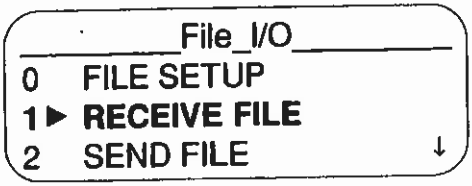


After you choose Print Setups, the SA will start printing the ten file I/O setup records. To cancel printing, press "*".

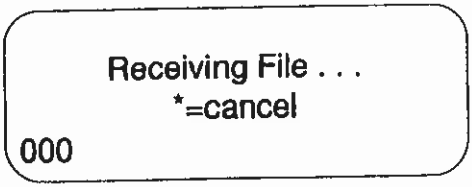
After setting the various parameters associated with File I/O, such as File I/O pointers, file type, and filename, you are ready to send or receive data. To start either process, first choose ***File I/O*** from the Main menu:



Then choose ***Receive File*** or ***Send File**** from the File I/O menu:



While sending or receiving, the SA will show the following display:

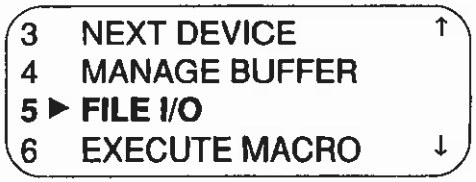


The three digit number in the lower left-hand corner of the display indicates the number of pages (256 byte sections) sent or received.

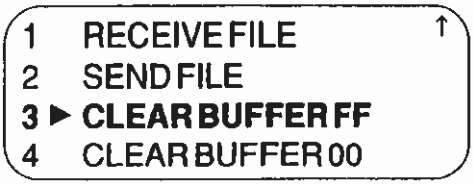
Clearing the Buffer

The last two functions of the File I/O menu are Clear Buffer FF and Clear Buffer 00. As their names imply, they clear the buffer to FF or 00. However, these functions clear the ***entire*** buffer, not just the portion designated by programming pointers, so be careful when using these functions.

To access these functions, first choose ***File I/O*** from the Main menu:



Then choose ***Clear Buffer FF*** or ***Clear Buffer 00*** from the File I/O menu:



Chapter 4

Macros

A macro is a collection of keystrokes that can be executed as if the keys were pressed on the SA's keypad. This powerful function allows you to automate repetitive tasks, making your SA easier and faster to use.

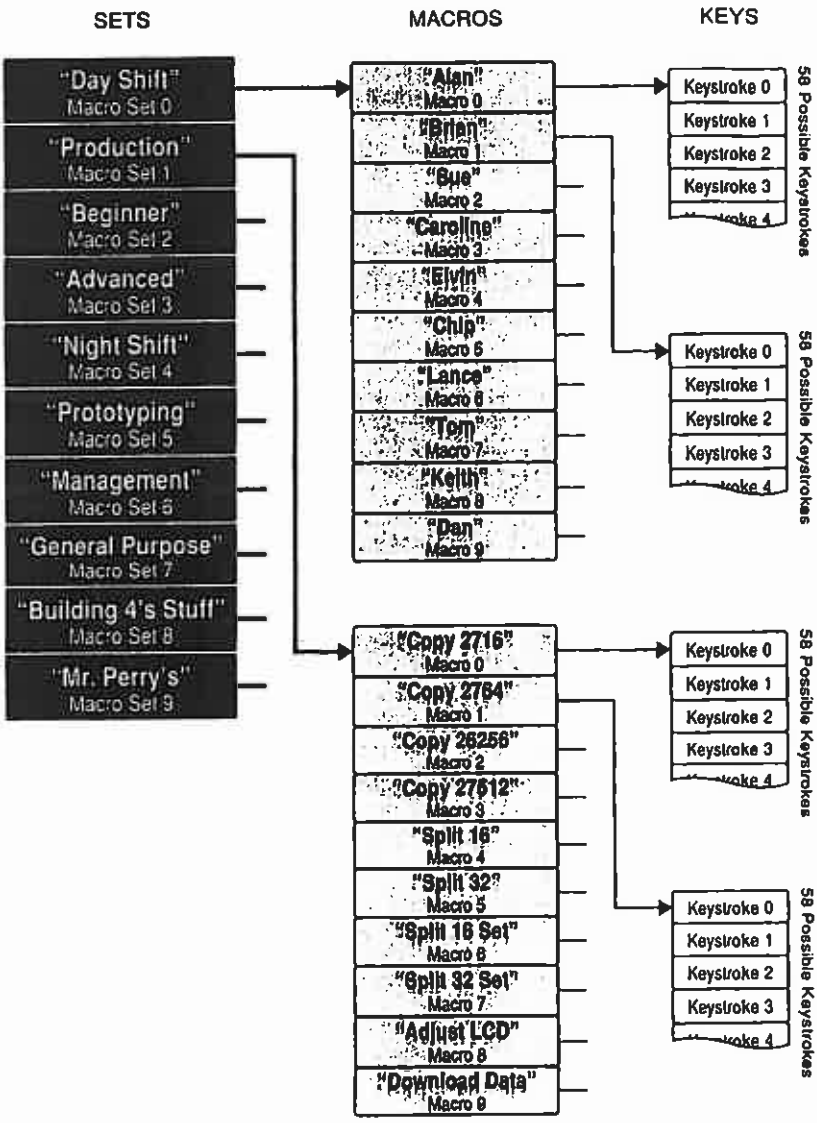
Macro Hierarchy

As stated on the previous page, a macro is simply a collection of keystrokes that can executed as if from the keypad.

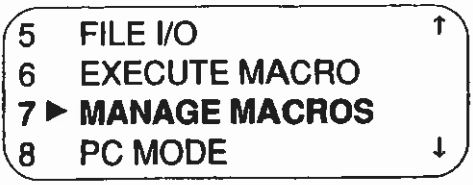
The SA has ten macro sets, each set containing ten macros. Each macro may contain up to 58 keystrokes. So, with ten sets of ten macros, each macro with 58 keys, you can define as many as 5,800 keys.

The diagram on the following page shows the relationship between sets, macros, and actual keystrokes. Following the diagram is a discussion on how to create and use macros.

Macro Hierarchy

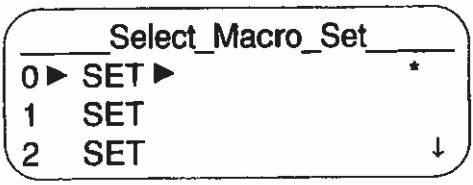


To create a macro, choose ***Manage Macros*** from the Main menu.



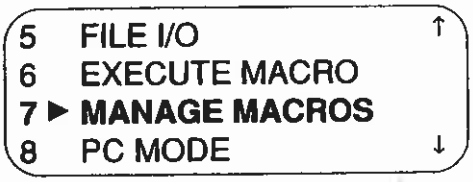
The first step in creating a macro is to select one of the ten possible macro sets, so choose ***Select Macro Set*** from the Manage Macros menu.

After choosing Select Macro Set, you will see the display shown below:

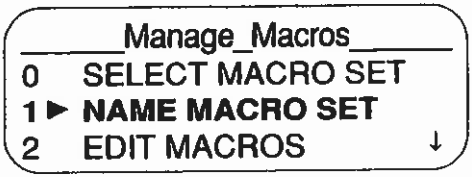


This display is used to select one of the macro sets. The "*" indicates the currently selected set. When the SA is turned on, the first set is automatically selected. To select another set, use the ↑ and ↓ keys to scroll through the display, then press ↵ when you have found the set you want.

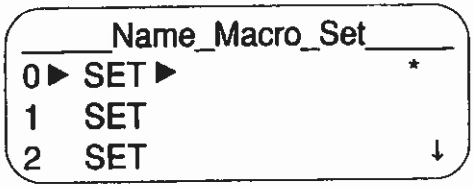
After selecting a macro set, you will be taken back to the Main menu. To return to the macros menu, again choose ***Manage Macros.***



The second step in creating a macro is to name the selected macro set. Choose ***Name Macro Set*** from the Manage Macros menu.

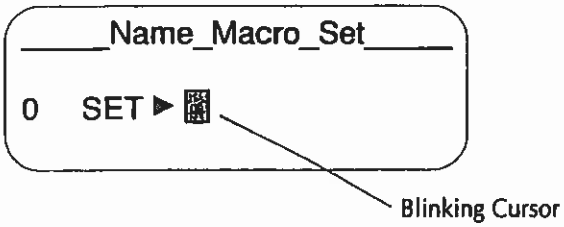


After selecting Name Macro Set, you will see the display shown below:



This display is used to name one of the macro sets. As with the Select Macro Set menu, the "*" indicates the currently selected set. To select another set, use the ↑ and ↓ keys to scroll through the display, then press ↵ when you have found the set you want.

After selecting the set to be named, you will see the following:



This display allows you to enter the name of the selected macro set.

To select the desired letter, number, or symbol, use the ↑ and ↓ keys. When the cursor shows the desired character, press "F" to move to the next space.

While entering the name, the following keys may be used for the indicated purposes:

0 Insert
1 Delete

E Move Left
F Move Right

↑ Increment Character
↓ Decrement Character

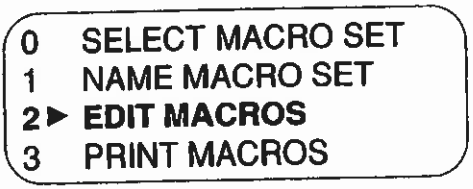
↵ Press when finished with name

D Display help menu

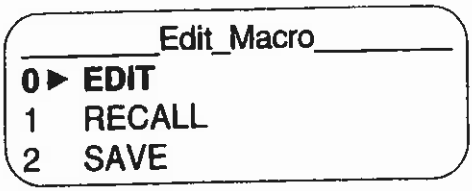
Pressing "D" while editing the name will display a help menu that shows the above keys.

When finished entering the name, press ↵.

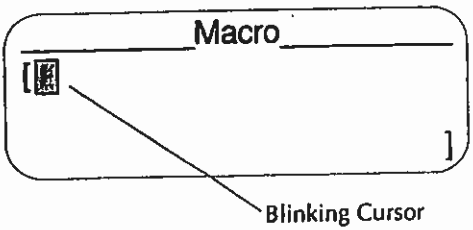
After selecting and naming a macro set, you can enter the actual macro. To do so, choose ***Edit Macros*** from the Manage Macros menu.



Then choose ***Edit*** from the Edit Macro menu.



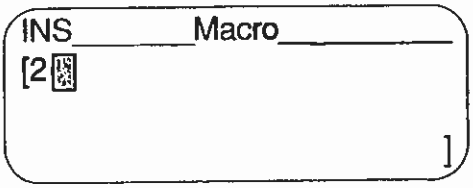
After choosing Edit, you will see the following display:



This display is used to enter the actual macro keystrokes. Instructions on entering the keystrokes appear on the next page.

To enter a keystroke, press "0". You will see the letters "INS" appear at the top of the display; this indicates that the SA is waiting for the actual keystroke to insert.

After pressing "0", press the key that you want to insert. The selected key will appear and the cursor will move to the next position.



Insert mode is cancelled after each keystroke, so you must press "0" before each key.

In addition to entering actual keys, you may also enter the special "K" character. The "K" character causes the SA to pause while executing a macro, so that you can input data by pressing a key on the SA's keypad. For each key that will be pressed, one "K" should be inserted. For instance, if the macro pauses at the Generic Part Number menu, you will need two K's, since two digits are used to specify a part number.

While entering the macro, the following keys may be used for the indicated purposes:

0 Insert
1 Delete
2 Insert "K" (pause for key)
E Move Left
F Move Right

↵ Press when finished with macro

D Display help menu

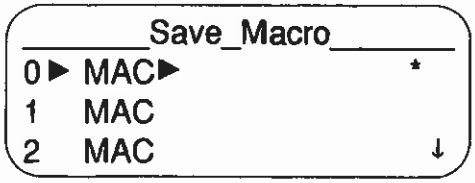
When finished entering the macro, press ↵.

Note that macros always start execution from the Main menu. When executed, the macro on the previous page would choose Select Device (option #2) from the Main menu. Since the macro only has one entry, its execution would stop there, leaving you at the Select Device menu.

After entering a macro, you will be returned to the Edit Macro menu. To save the newly created macro, choose ***Save***:

```
_____Edit_Macro_____
0   EDIT
1   RECALL
2 ► SAVE
```

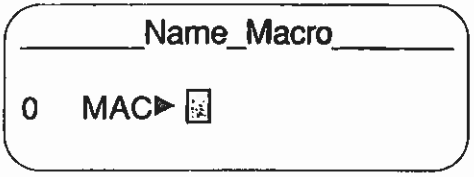
After choosing Save, you will see the display shown on the next page.



This display is used to designate where the newly-created macro will reside in the currently selected macro set (remember, each macro set can contain up to ten macros). If all ten "slots" are already occupied by other macros, then you will have to overwrite one of them or select another macro set.

As with other displays, the "*" indicates the currently selected macro (actually, the slot containing the macro). To select another slot for the new macro, use the ↑ and ↓ keys to scroll through the display, then press ↵ when you have found the slot you want. If you press ↵ while at a slot already occupied by a macro, the new macro will replace the old macro.

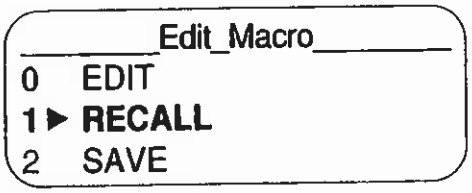
After designating one of the ten available macro "slots" for the new macro, you'll be asked to name the new macro:



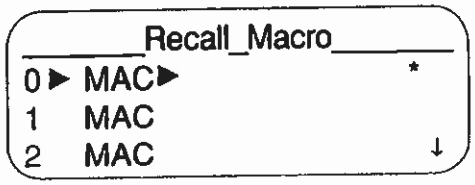
If the new macro is replacing an old macro, then the old macro's name will appear as the default name. Pressing ↵ will leave the name as it appears.

The same keys are used to edit a macro name as are used to edit the name of a macro set (see page 4-5).

If you need to edit a previously saved macro, you may do so by choosing ***Recall*** from the Edit Macro menu.:



You will then see the following display. This display allows you to select which macro to recall:

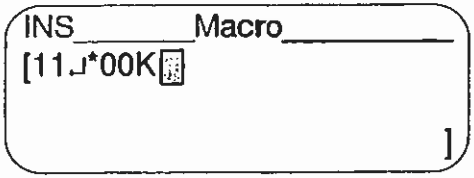


To select a macro, use the ↑ and ↓ keys to scroll through the display, then press ↵ when you have found the macro to recall.

Once recalled, a macro becomes the "currently selected" macro and may be edited and saved just as a new macro.

Sample Macro

The following sample macro sums up all that we have learned about entering macros. It is a simple copying macro that reads a device and then pauses so you can insert a blank device before programming:



When executed, the macro performs the following functions:

1: Choose READ DEVICE from Main menu
1: Choose READ from Read Device menu
↵: Start Reading Data from Device
*: Return to Main menu
0: Choose PROGRAM DEVICE from Main menu
0: Choose STANDARD 50ms from Select Algorithm menu
K: Pause; wait for keypress. After keypress, start programming

When executed, the macro will read the original device and then pause while you remove the original and insert a blank device to be programmed. When the blank device is inserted, press ↵; the SA will then program the device. At this point, the macro is finished executing and control is returned to the SA's keypad.

After programming the blank device, the SA will display the status of the programming process and then wait for a keypress before returning to the Main menu.

Note that you must select the device type before using the above macro.

If you have a printer connected to your SA, you can print your macros for easy reference. To print the macros, choose ***Print Macros*** from the Manage Macros menu:

```
0   SELECT MACRO SET
1   NAME MACRO SET
2   EDIT MACROS
3 ► PRINT MACROS
```

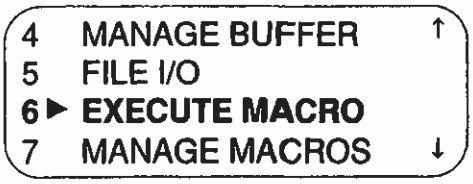
After choosing Print Macros, you will see the following display:

```
_____Print_Macros_____
0 ► CURRENT SET
1   ALL SETS
```

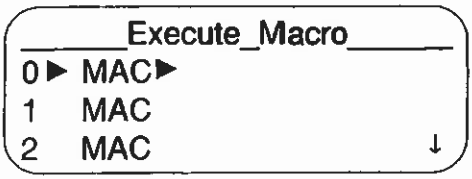
To print the currently selected macro set, choose ***Current Set***. To print all macro sets, choose ***All Sets***.

After receiving your selection, the SA will start sending macro data to the printer port. To cancel printing, press "*".

To execute a macro, choose ***Execute Macro*** from the Main menu:



The SA will then show the following display, from which you may select the appropriate macro:



To select a macro, use the ↑ and ↓ keys to scroll through the display, then press ↵ when you've found the macro you want. The selected macro will start executing after you press ↵.

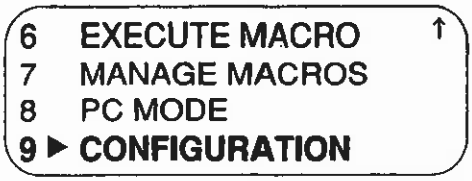
Remember that there are ten macro sets available. The macros shown in the Execute Macro menu are from the currently selected set. If the macro you want does not appear in the menu, the wrong set may be selected. For instructions on selecting another set, see page 4-3.

Chapter 5

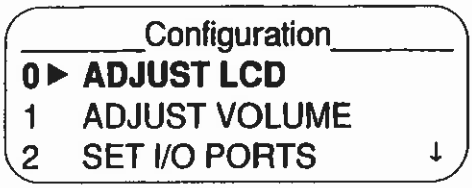
User Configuration

There are a number of aspects of your SA that can be set to suit particular needs, such as LCD viewing angle, speaker volume, and I/O port settings. As a whole, these settings form the SA's configuration.

The following chapter describes how to adjust and save configuration settings. In addition, this chapter descirbes how to upgrade the SA's internal firmware.

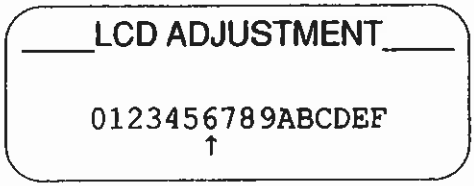


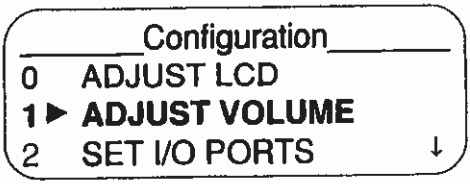
To change a configuration setting, choose ***Configuration*** from the Main menu.



From the Configuration menu, you can choose one of eight options.

To change the LCD screen viewing angle, choose ***Adjust LCD***. You will see the numbers 0-F with an arrow pointing to the current LCD setting. To change the setting, press the desired number key (0-F) or use the ↑ and ↓ keys to step through the settings.

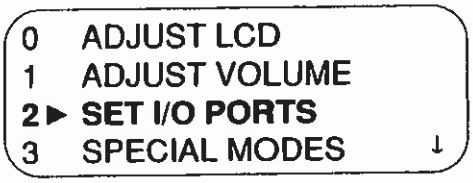




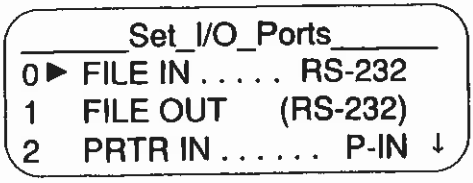
To change the SA's speaker volume, choose ***Adjust Volume.*** Like the LCD adjustment, you will see the numbers 0-F with an arrow pointing to the current volume setting. To change the setting, press the desired number key (0-F) or use the ↑ and ↓ keys to step through the settings.

The SA has three built-in I/O ports for connection to a printer and host computer. The first port Is a bi-directional parallel port that may be connected to a printer or host computer. The second port is a bi-directional serial port that may also be connected to a printer or host computer. The third port is a uni-directional parallel port that can only be connected to a printer.

With three ports, the SA can accomodate most needs. For instance, a host computer could be connected via the serial port and a printer via the parallel out port. When used as a printer buffer, the SA will input the serial data and send it out over the parallel port.



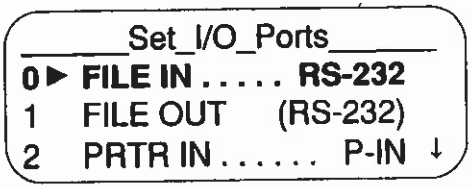
To set the SA's file I/O and printer ports, choose ***Set I/O Ports*** from the Configuration menu. You will then see the following display:



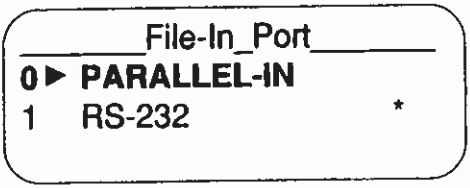
This display allows you to designate the function of each port. If the port is a serial port, you may also adjust the settings used for serial communication, such as baud rate, number of stop bits, etc.

Listed to right of each function is the port currently in use by that function. In the example display, file in and out functions are being carried out over the RS-232 serial port, while printer operation is taking place over the parallel ports.

To change a function's port or port settings, choose the function from the menu. For example, to change the File In port from serial to parallel, choose ***File In*** from the Set I/O Ports menu:



After choosing File In, you will see the display shown below:



Choose ***Parallel In*** to switch the File In port to the parallel port; as always, the * indicates the currently active selection. After choosing Parallel In, you'll be returned to the Set I/O Ports menu.

If you instead select a serial port, you will see an additional display as shown on the next page.

```
_______File-In_Port_______
BAUD▶19.2K      WDLEN:8
STP:1  PAR: OFF     FLW:D
--- E,F = ← → --0=RESET ---
```

If you select a serial port, you will see this display. It allows you to change the settings associated with serial communications.

To change a setting, use the ↑ and ↓ arrows. To move to another setting, use the E and F keys to move left and right. Press ↵ when you are finished adjusting the settings, or press 0 to reset the settings to their default values.

The following chart defines each of the settings shown in the above display:

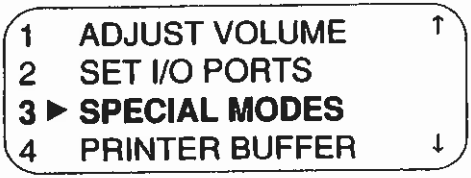
BAUD:	Rate of transmission and reception of data (in bits per second)
WDLEN:	Word length (in bits)
STP:	Number of stop bits used (in bits)
PAR:	Parity (on or off)
FLW:	Transmission protocall used (DTR or XON/XOFF)

When connecting the SA to a PC compatible via serial white using the communication software, only pins 2, 3 and 7 are necessary. Be sure, however, that the cable used is the 1:1 or null modem type. Some printer cables reverse pins 2 and 3 within the cable and will not operate with the SA.

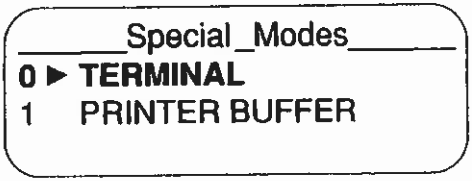
If you are connecting the SA to another type of computer in the terminal mode more lines are necessary. Please refer to page 6-3.

If you have a computer that is not yet supported by the supplied software (Apple, CP/M, etc.), you can still use the SA with your computer by connecting it as a modem and envoking Terminal Mode. When in Terminal Mode, the SA accepts keystrokes from the your computer as if from its keypad; in addition, the SA sends the same data to your computer that it displays on its LCD screen.

To use the SA this way, connect the SA's serial port to your computer's modem port (RS-232) and then run your favorite communicatons software. On the SA, choose ***Special Modes*** from the Configuration menu:



Then choose ***Terminal*** from the Special Modes menu:



Note that when using Terminal Mode, the baud rate, stop bits, and parity settings must be identical for both your computer and the SA. See the previous page for a description of setting baud rate, stop bits, and parity for the SA. The method of adjusting these settings varies for different communications software, so refer to your communications software manual for instructions on adjusting the settings for your computer.

Since the SA will often send full screens to your computer, you should use the highest possible baud rate that both the computer and SA support. In most cases, this will be 19.2K baud.

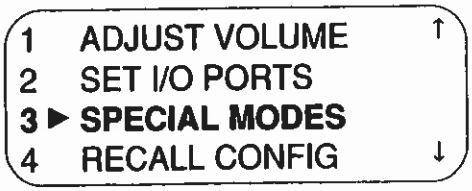
While in the terminal mode, the following keys will replace the SA keys not available from the computer keyboard:

(Shift) I ..... up arrow key
(Shift) M ..... down arrow key
Backspace ..... (*)

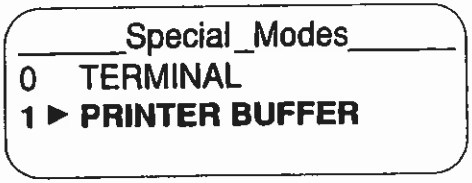
When you are not using your SA to program devices, it can function as a simple printer buffer. When acting as a printer buffer, the SA accepts data from one of the input ports, stores the data in it's memory, and then outputs the data to a connected printer. Since the SA can accept data much faster than the printer can print it, this greatly reduces the time that your computer has to wait for printing.

All of the SA's internal memory is used as the buffer, so a standard SA provides 128K of buffer space, enough to store about 64 pages. Of course, if you expand the memory of your SA, it will be able to store more pages (512 pages with 1 megabyte; 2048 pages with 4 megabytes).

To activate the printer buffer function, choose ***Special Modes*** from the Configuration menu:



Then choose ***Printer Buffer*** from the Special Modes menu:



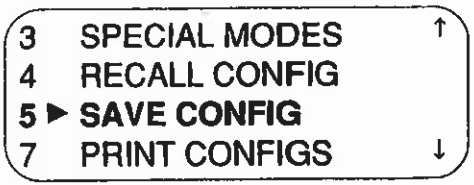
While the printer buffer is in effect, the SA will display the amount of data received and the maximum amount it can hold. To cancel printer buffer mode, press "*".

Using Configuration Records

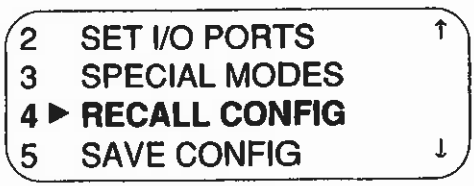
Your SA is capable of storing up to ten *configuration records* for your convenience. Configuration records contain various settings that affect the operation of the SA. The following settings are stored in the configuration record:

- Device Type
- Macro Information
- Set & Split Options
- LCD Adjustment
- Pointer Values
- I/O Port Settings
- Current Serial Number
- Volume Level

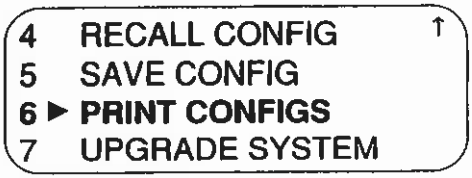
To save the configurations of your SA, choose ***Save Config*** from the Configuration menu. You will then select one of ten available configuration records, give the record a name, and then press ↵ to save the configuration in the SA's internal EEPROM.



To later recall one of the ten configuration records from the SA's internal EEPROM, choose ***Recall Config***.



To print the available configuration records on a printer, choose ***Print Configs***. This will print the name and data stored for each configuration record.

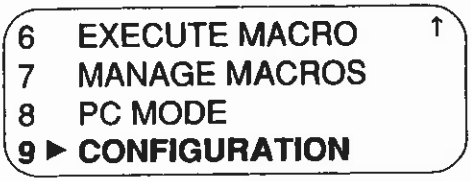


Upgrading the SA's Firmware

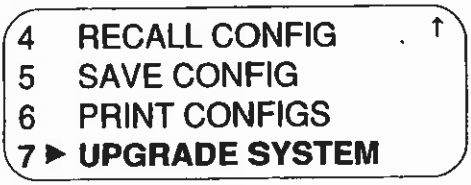
When firmware upgrades are offered by Needham's, these upgrades will be provided on diskette (formatted for the IBM PC and other popular computers).

Using the diskette, load the upgraded software into your personal computer and then download the software to the SA just as you would any other data.

To tell the SA that the data received is upgraded software, choose ***Configuration*** from the Main menu:



Then choose ***Upgrade System*** from the Configuration menu:



Chapter 6

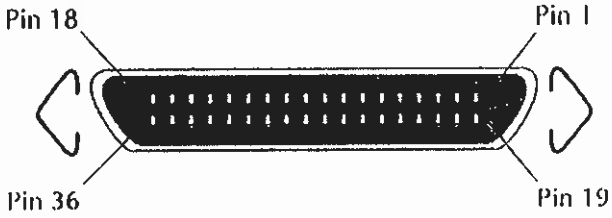
I/O Ports

Your SA device programmer is equipped with several I/O ports that allow you to connect the SA to various input and output devices, such as a personal computer, printer, and modem.

This chapter gives the pin-outs for each port.

Parallel Output Port (printer out)

Pin	1:	STB
	2:	D0
	3:	D1
	4:	D2
	5:	D3
	6:	D4
	7:	D5
	8:	D6
	9:	D7
	10:	ACK
	11:	BUSY
	12:	NC
	13:	NC
	14:	NC
	15:	NC
	16:	NC
	17:	GND
	18:	GND
	19:	GND
	20:	GND
	21:	GND
	22:	GND
	23:	GND
	24:	GND
	25:	GND



Pin	1:	STB	Pin	19:	GND
	2:	D0		20:	GND
	3:	D1		21:	GND
	4:	D2		22:	GND
	5:	D3		23:	GND
	6:	D4		24:	GND
	7:	D5		25:	GND
	8:	D6		26:	GND
	9:	D7		27:	GND
	10:	ACK		28:	GND
	11:	BUSY		29:	GND
	12:	PAPER OUT		30:	GND
	13:	SELECT		31:	NC
	14:	GND		32:	SELECT
	15:	NC		33:	GND
	16:	GND		34:	NC
	17:	GND		35:	NC
	18:	NC		36:	NC

Pin	1:	GND
	2:	RECEIVE
	3:	TRANSMIT
	4:	CTS
	5:	RTS
	6:	DTS
	7:	GND
	8:	NC
	9:	NC
	10:	NC
	11:	NC
	12:	NC
	13:	NC
	14:	NC
	15:	NC
	16:	NC
	17:	NC
	18:	NC
	19:	NC
	20:	DSR
	21:	NC
	22:	NC
	23:	NC
	24:	NC
	25:	NC

Chapter 7

Upgrading

As your needs increase and software improves, you will need to upgrade your SA. The three aspects of upgrading are downloading new software, adding RAM, and changing the EPROM & Flash EEPROM that contain the SA's software.

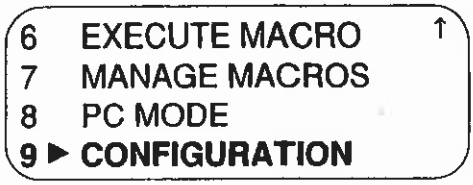
This chapter shows how to download new software and how to open your SA to insert/remove various chips.

Upgrading the SA's Firmware

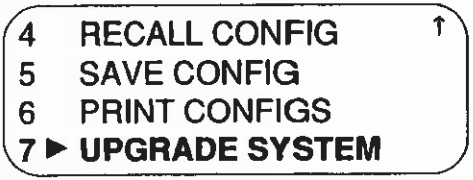
When firmware upgrades are offered by Needham's, these upgrades will be available on diskette (formatted for the IBM PC and other popular computers).

Using the diskette, load the upgraded software into your personal computer and then download the software to the SA just as you would any other data.

To tell the SA that the data received is upgraded software, choose ***Configuration*** from the Main menu:

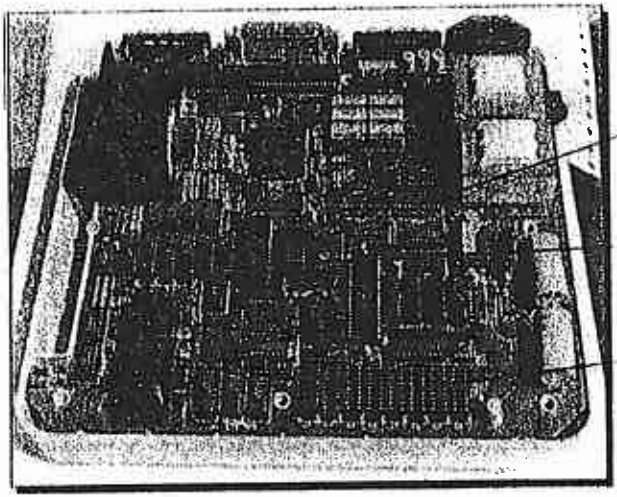


Then choose ***Upgrade System*** from the Configuration menu:



See pages 7-5 & 7-6 for further instructions.

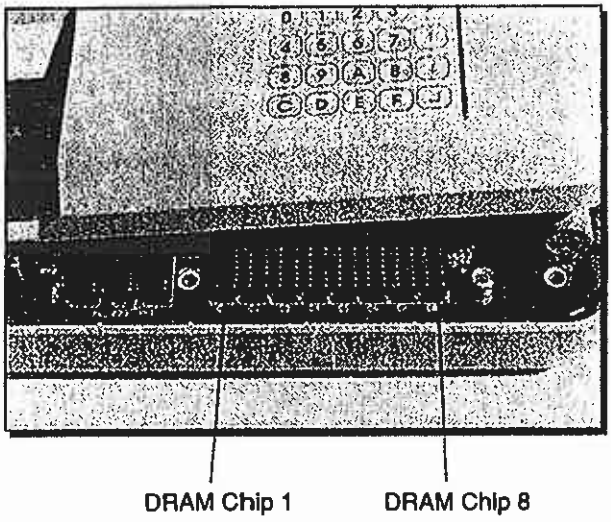
To add RAM or change the EPROM & Flash EEPROM, you must first open the SA's case. The photo below shows an SA opened to allow access to the various chips.



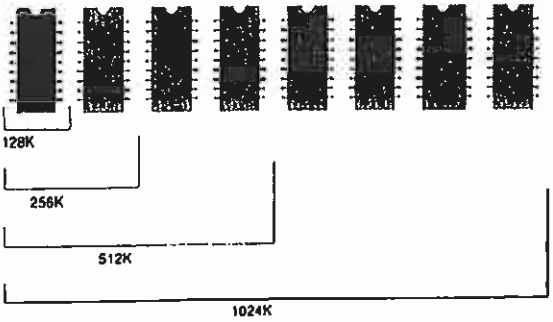
Note that the top half of the case should not be completely removed (specifically, the cables should not be disconnected). Instead, as in the photo on the following page, the top should be offset slightly to expose the parts to be inserted/replaced; this is easier and prevents the cables from being damaged or reconnected incorrectly.

Adding RAM

Your SA-10/SA-20 was shipped with 128K of RAM for use as the data buffer, enough for most applications. However, by simply adding memory chips, it is possible to have up to 1 megabyte of buffer space. Added buffer space provides faster programming and easier downloading of large amounts of data. The photo below shows the location of the DRAM chips that comprise the SA's memory.



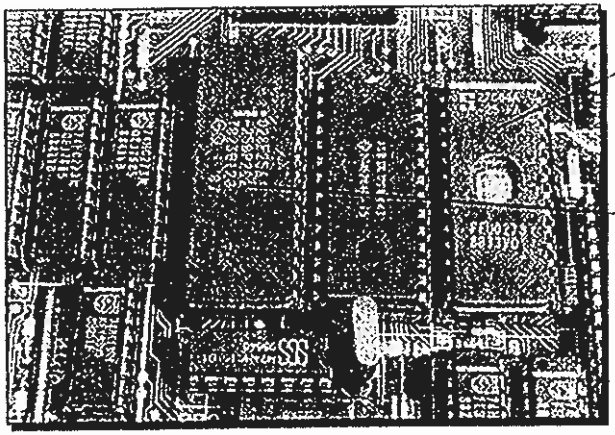
As explained in the Data Buffer chapter, you can use 1 megabit chips in any of the configurations shown below.



The first number in each pair applies if using 1 megabit chips.

Most software updates will be accomplished by downloading new code to the SA from a computer. However, the SA's software may eventually become too large to fit in the current EPROM and Flash EEPROM, requiring you to replace these chips.

The photo below shows the location of the EPROM and Flash EEPROM .



UPGRADING THE FIRMWARE FROM A FILE

The internal firmware for the SA-10/20 can be upgraded without disassembly of the unit. An upgrade file, provided by Needham's Electronics, can be downloaded from an IBM PC compatible computer into the SA-10/20 and from this file the internal Flash device can be rewritten. Please note that when rewriting the internal Flash device, it will erase all macros, configurations, and file setups stored internal to the Stand Alone. You may wish to print these out or write them down before upgrading.

When downloading the upgrade software from a disk, the file will be in the firmware subdirectory of the SA directory. It can be recognized by the .bin extension and the SAxxx prefix. At the time of this writing the current software is version 1.49. The upgrade file would then be listed as "SA149.bin".

This file can be downloaded into the programmer like any other file. From the main menu go into the (5) File I/O section, and then into (0) File Setup. Be certain that the buffer and file pointers in the file in the (0) File Pointer section begin at 00 and extend at least to FFFF. Be sure that the (1) File Type is set for Binary, and that the correct path and file name are input in the (2) File Name menu. A common path from a floppy drive would be: "A:\SA\firmware\SA149.bin". Once this is input and Enter is pressed, escape once to go to the File I/O menu and press (1) to Receive a file. This will transfer the data from the PC to the SA-10/20. Once the file is received, chose main menu option (9) Configuration and then (7) to rewrite the Flash device.

PRINTING THE .HEX FILE TO UPGRADE

It is possible for the internal Flash device to become corrupted during the upgrading procedure or normal operations due to a power glitch. During power up procedures, a check of the validity of the internal Flash is performed, and if found to be corrupted, the SA-10/20 should default to a mode where the statement "waiting for system download on parallel" will appear on the screen. It is also possible that a partial corruption of the Flash can result in a blank screen or errant data on the LCD display. Holding down the * key while turning the unit on should display the "waiting for system download in parallel" message. At this point it is possible to rewrite the Flash memory using the Print command on the PC and the .hex file. The SA-10/20 must be hooked to the PC via the parallel port, not serial, and the PRINT command must be used. If the latest firmware is available the procedure would be as follows:

With the "waiting for system ..." message on the LCD display, connected to a PC via the parallel port, type the following;

Print SA149.HEX

This is assuming that you are using DOS and are withing the directory that contains the SAxxx.HEX file. The LCD display should change to "Receiving Data" and then to "Upgrading Flash". If problems should continue please contact Needham's Electronics.

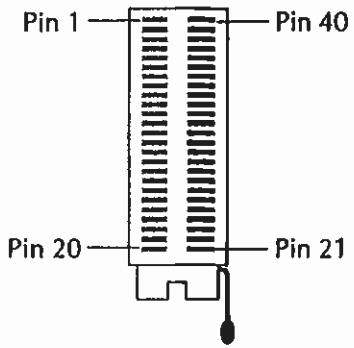
Chapter 8

General Information

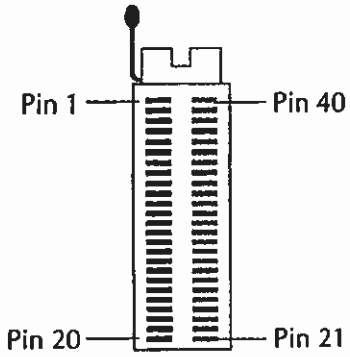
This chapter contains information about the SA's electrical characteristics, including socket pin-outs and power requirements. Also included are weight and Flash EEPROM lifespan figures.

A printable file of the full electrical schematics of the SA can be found on the communication software disc provided with the programmer.

Socket Pin-Outs (socket zero)



Pin	1:	GND	Pin	40:	5V
	2:	GND		39:	5V
	3:	EXTRA PIN		38:	5V
	4:	EXTRA PIN		37:	NC
	5:	A19/Vpp		36:	Vcc (32 pin dev.)
	6:	A16		35:	A18
	7:	A15/Vpp		34:	A17/Vcc
	8:	A12		33:	A14
	9:	A7		32:	A13/Vcc
	10:	A6		31:	A8
	11:	A5		30:	A9/12V
	12:	A4		29:	A11/Vpp
	13:	A3		28:	OE/Vpp
	14:	A2		27:	A10
	15:	A1		26:	CE
	16:	A0		25:	D7
	17:	D0		24:	D6
	18:	D1		23:	D5
	19:	D2		22:	D4
	20:	GND		21:	D3



Pin	1:	NC	Pin	40:	5V
	2:	NC		39:	GND
	3:	NC		38:	NC
	4:	NC		37:	NC
	5:	A19/Vpp		36:	Vcc (32 pin dev.)
	6:	A16		35:	A18
	7:	A15/Vpp		34:	A17/Vcc
	8:	A12		33:	A14
	9:	A7		32:	A13/Vcc
	10:	A6		31:	A8
	11:	A5		30:	A9/12V
	12:	A4		29:	A11/Vpp
	13:	A3		28:	OE/Vpp
	14:	A2		27:	A10
	15:	A1		26:	CE
	16:	A0		25:	D7
	17:	D0		24:	D6
	18:	D1		23:	D5
	19:	D2		22:	D4
	20:	GND		21:	D3

Power requirements ..... 120 volts AC @ 1 amp

Unit weight (SA-10) ..... 7 pounds
Unit weight (SA-20) ..... 8 pounds

Flash EEPROM Lifespan ..... 100 write cycles

Chapter 9

Appendices

Appendix A: *List of Devices by Manufacturer*

Appendix B: *ASCII Chart*

Appendix A: List of Devices

The following list shows the devices that have been tested on the SA device programmer. Devices not shown are likely to work, although not specifically tested. To use a device that is not shown, choose the device's "generic" part number when selecting device type.

Generic	AMD	Atmel	Exel
2716	2716	27HC64	2804A
2732	2716B	27C256	2816A
2732A	2732	27HC256	
2764	2732A	27C256R	
27128	2732B	27C512	
27128A	2764	27C512R	
27256	2764A	27C513	
27512	27128	27C513R	
	27128A	27C010	
	27256	2804A	
	27C256	2816A	
	27512	2817A	
	27C512	28C64	
	27C010		

Fujitsu	Hitachi	Intel	Mitsubishi
2732	2716	2716	2716
2764	2732	2732	
27C128	2732A	2732A	
27C256	2764	2764	
27C512	27128	2764A	
	27128A	P2764A	
	27256	27C64	
	27C256	P27C64	
	27512	27128	
	27C101	27128A	
	27C301	27128B	
		P27128A	
		27256	
		27C256	
		27C256-V	
		27512	
		27010	
		27C010	
		27011	
		27C011	
		27C020	
		27C040	

Motorola	National Semiconductor	NEC
68766	2716	2716
	27C16	2732
	27C32	2732A
	27C32B	2764
	27C64	27C64
	27C64B	27128
	27CP128	27C256
	27C128B	27C256A
	27C256	27C512
	27C512	27C1000
	27C010	27C1001
	27C020	27C2001

Seeq	Signetics	Texas Inst.	SGS
2764	27C64A	2732	2716
27128	27C256	2732A	2732
27C256	27HC641	27C32	27C64A
2804A		2764	27256
2816A		27C64	27C256
2817A		27C128	27512
2864		27C256	
28C64		27C512	
28C256		27C010	

Toshiba	Wafer Scale	Epson	
2764	57C49	32SEC0	34 PIN
2764A		64SEC0	CARDS
27128		128SEC1	
27128A		128IEC1	40 PIN
27256		256IEC0	CARDS
27256D		512IEC0	
57256D		RBC008	RAM
57256AD		RBC016	CARDS
27C512		RBC032	
571000		RBC064	
571001		RBC128	
574000		RBC256	
		RBC512	
		065HE	50 PIN
		129HE	CARDS
		257HE	
		513HE	

ASCII (hex)	Character	ASCII (hex)	Character
00	NUL	21	!
01	SOH	22	"
02	STX	23	#
03	ETX	24	$
04	EOT	25	%
05	ENQ	26	&
06	ACK	27	'
07	BELL	28	(
08	BCKSP	29	)
09	HT	2A	*
0A	LF	2B	+
0B	VT	2C	,
0C	FF	2D	
0D	CR	2E	.
0E	SO	2F	/
0F	SI	30	0
10	DLE	31	1
11	DC1	32	2
12	DC2	33	3
13	DC3	34	4
14	DC4	35	5
15	NAK	36	6
16	SYN	37	7
17	ETB	38	8
18	CANCEL	39	9
19	EM	3A	:
1A	SUB	3B	;
1B	ESC	3C	<
1C	FS	3D	=
1D	GS	3E	>
1E	RS	3F	?
1F	US	40	@
20	SPACE	41	A

ASCII (hex)	Character	ASCII (hex)	Character
42	B	63	c
43	C	64	d
44	D	65	e
45	E	66	f
46	F	67	g
47	G	68	h
48	H	69	i
49	I	6A	j
4A	J	6B	k
4B	K	6C	l
4C	L	6D	m
4D	M	6E	n
4E	N	6F	o
4F	O	70	p
50	P	71	q
51	Q	72	r
52	R	73	s
53	S	74	t
54	T	75	u
55	U	76	v
56	V	77	w
57	W	78	x
58	X	79	y
59	Y	7A	z
5A	Z	7B	{
5B	[	7C	|
5C	\	7D	}
5D	]	7E	~
5E	^	7F	DEL
5F	_		
60	`		
61	a		
62	b		

Index