DATAMAN S3

Owner's Manual

Dataman Programmmers Ltd
Station Road
Maiden Newton
DORCHESTER
Dorset DT2 0AE
United Kingdom

Phone	(0300) 320719
Fax	(0300) 321012
Bulletin Board	(0300) 321095
Telex	418442

Issue 2

Table of Contents

Introduction to S3 1

Check List of Parts and Accessories 2
Confirming that S3 works 2
Loading the HELP ROM 3

Fundamentals 5

Automatic Power-down 5
The PROM Socket 5

Computer Operation 7

Glossary: Words and Concepts 8

Paged Mode EPROMS like 27513 11
Function Keys 12
Beeping 12

LOAD 13

BURN 15

SPLIT 17

How memory splitting works . . 18

SHUFFLE 19

HELP 20

PRETEST 22

COMPARE 24

RECEIVE 26

Audible Output 27
Returning to Command Mode. . 27

SEND 28

Audible Output 29

CONFIGURE 30

PROM Type. 31
File Type. 32
Baud Rate. 32

MOVE 34

SWAP 35

FILL 36

BOOST CHARGE 37

EMULATE 39

EMULate from SECTOR zero. . . 40

CHECKSUM 41

SEEK 43

EDIT 44

Stand-Alone Editing. 44
Remote Editing 45
DUMP 46

ESC edit(system) 47

F1 - DUMP ROM 49

F2 - QUICK-BURN 50

F3 - CHECKSUM ROM 52

F4 - MAKE HELP 53

F5 - SYSTEM CHECK 55

Block Diagram 56

BIOS Mode 57

RS232 Serial Interface 59

Alterations to the Standard Interfac59

RS232 Interface Lead 60

Baud Rates 61

Interfacing with a Computer. 64

Terminal Emulating Programs. . 65
Communications Software . . . 66
File Formats. 67

INTEL Format. 68

Example 68

MOTOROLA S Format 69

Example 69

TEKHEX Format 70

Example 70

ASCII Format 71

Example 71

BINARY Format 72

Example 72

Battery and Charger 73

Battery Charger. 74

Boost Charge Current 74
Starting a Boost Charge. 75
Charging from a Bench Supply. . 76
Important Charger Modification . 77
How to Detect a Defective Battery. 78

Memory Emulation 79

Benefits of memory emulation . . 79
How it works 80
Microsystem memory selection . 80
Correct Prototype Design 81
Emulating RAM. 82
Power Usage when Emulating. . 82
Line Terminations of the EMULead 82
NOTE - Emulation Addressing. . 83

Changing System Variables 84

S3 VARIABLES 85

Derived from PULSE table . . . 85
Derived from OR/AND table . . 86
System defaults 86
Silent Serial Transmissions . . . 87

Programming Algorithms 88

How to Design New Algorithms 89

PROM Addressing 89
Programming states 90
Instructions 90

The Algorithm Table 92

The Pulse Table 94

EPROM Pulse Table. 94
EEPROM Pulse Table 95

The OR/AND Table 96

The OR/AND Chart. 98

Using the Modules 99

Installing a Module. 99
Removing a Module. 99

Page Editing 100

32 Pin Module 102

Keeping Up-to-Date. 102
Programming in PAGES. 102
Programming Methods. 103

40 Pin Module 104

How S3 handles 16 bit words . 104

MCS-48 Series Module 106

Using the Module 106

MCS-51 Series Module 108

Using the Module 108
Additional features 108

MCS51 Security Mode 110

MCS51 Burn Security 111

MCS51 Burn Encryption 112

MCS51 Comp Encrypted 114

MCS51 Load Encrypted 115

The EPLD Package 116

EPLD Module 1 116
EPLD Module 2 116
Logic Compilers 117
The Fuse Array 118
Unwanted routines 119
BEWARE - HIGH VOLTAGES . . . 119

EPLD Load 121

EPLD Burn 122

EPLD Test 123

EPLD Compare 124

EPLD Receive 125

EPLD Send 127

EPLD Configure 128

F1 - EPLD Status 129

F2 - EPLD Blow Security 131

F3 - EPLD Power Modes 133

F4 - EPLD Make Help 135

Software Upgrades 136

BURN visual feedback 136
Sector confusion cleared up . . 136
Silent Transmissions 137
NICAD care 137

FLASH ROM Programming 138

S3's programming procedure . . 138

Texas TMS27C292 140

Programming a 27C292 140

Intel's 2816 141

Programming the 2816 141

GETTING STARTED

HOW TO COPY EPROMS

This guide is for those who wish to use the S3 to copy a Master EPROM into blank EPROMs of the same type.

There are four stages:

- (1) CONFIGuration of S3
- (2) LOADing the Master
- (3) TESTing the Blanks
- (4) BURNing the Copies

CONFIGURATION

Place a blank EPROM, with the notch at the top, into the ZIF socket on S3. If it is a 24 pin device, it should be placed at the lower end of the socket.

Press the <confg> key, then the a key.

If S3 displays the EPROM type, press the <ESC> key and proceed to the 'LOAD' instructions below.

If S3 requires you to find the EPROM type manually, it will display the following message.

```
DATAMAN S3 HELP 2.4
>CONFIG
UNKNOWN, PRESS ESC _
```

Press the <ESC> key ONCE and proceed as follows.

Manual Identification

The EPROM type will be printed on the EPROM. It is commonly one of 2716, 2732, 2764, 27128, 27256, or 27512, often interspersed with letters. To find the correct programming algorithm for the EPROM, look in the S3 Owner's manual in the section, just after the index, entitled 'EPROM CONFIGURATION TABLE'. This is a complete list of the devices which can be programmed by S3 and it corresponds with the standard library which S3 holds in memory. The devices are grouped by manufacturer for ease of reference. Find the type number in the first column of the table, and read off the algorithm name from the third column.

On S3, using the left and right arrow keys scroll backwards and forwards through the list of algorithms until the algorithm name you want is shown in the display. Then press the <ESC> key and proceed to the 'LOAD' instructions.

LOAD

Remove the blank EPROM from the socket, and insert the Master. Press the <LOAD> key. For a 2764 device, for example, S3 will display the following message.

```
>CONFIG
13 2764-FAST 21.0V
>LOAD 2764-FAST
SECTOR 00=0000,1FFF
```

The sector number should be 00, it should be followed by an '=', and the two addresses should correspond to the limits of the currently selected device.

The limit addresses are:

Device	Address Limits
2716	0000,07FF
2732	0000,0FFF
2764	0000,1FFF
27128	0000,3FFF
27256	0000,7FFF
27512	0000,FFFF

Once you are satisfied that the correct information is displayed press the <ENTER> key, and S3 will read the EPROM's data into its memory.

To check that the data has been read correctly, press the <COMP> key then the <ENTER> key. The S3 should display 'SAME', indicating that the data in memory is the same as the data in the EPROM.

- NOTE: It is a good idea to label the Master EPROM with its checksum. The checksum can be obtained by pressing the F3 key, which will return a six digit hexadecimal number. Once recorded, this number can be used for comparison in future checks.

TEST

Before attempting to copy data into an EPROM it is important to test that the device is either blank or, at least, able to receive data. To do this, remove the Master from the ZIF socket and replace it with the 'blank' EPROM. Press the <TEST> key then the <ENTER> key. The S3 should display 'BLANK ROM' or 'WILL BURN'. If either of these messages is displayed, proceed to the 'BURN' instructions below, if not, the EPROM should be erased and the test repeated until a correct message is obtained.

BURN

Press the <BURN> key then the <ENTER> key. S3 programs the EPROM. During the BURN cycle, S3 displays the current address at which it is working.The time taken varies from one device to the next, it may be as short as 17 seconds or as long as 14 minutes.

On completion of the BURN cycle, S3 executes an automatic COMPare. During this time it will beep at half second intervals. If the EPROM has been programmed correctly, S3 displays 'SAME'.

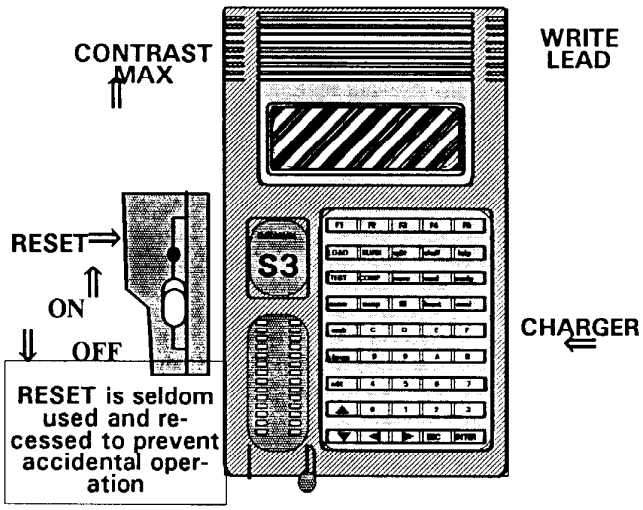
To program more blank EPROMs, simply repeat the 'TEST' and 'BURN' instructions.

- NOTE: When copying programs into EPROMs of varying manufacturer it is important to check the programming algorithm and re-configure S3 if necessary.The consequences of using the wrong algorithm range from unreliable programming, to device destruction.

If, after following the instructions, you experience difficulties with copying EPROMs then please contact Dataman Technical Support Department.

Introduction to S3

DATAMAN's S3 is a Personal Development Tool for Microsystem Designers, which emulates and/or programs EPROMS, EEPROMS and other devices. It is battery-powered, with 64k bytes of continuous memory which retains data and configuration even when switched off. The programming facilities include EPROMS of the 27 series, such as 2716 or 27513, and the 25 series, such as 2532 or 2564— and most EEPROMS, including 28, 52, 55 & 98 series. It also provides direct plug-in emulation for all these devices by means of a 24 or 28 pin emulator lead. Many other devices can be programmed, such as single-chip microprocessors, but some require a plug-in adaptor.



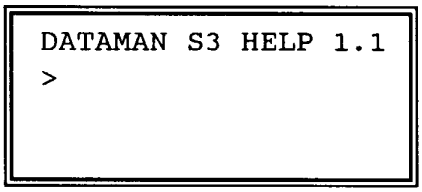
S3 has a keypad and an 80 character liquid crystal display. Also it may be used via an RS232 link with a computer. It performs like a debugger, such as MSDOS's DEBUG or CP/M's DDT, with viewing and editing of memory contents. The keyboard and terminal modes are active simultaneously - S3 responds to either source.

Check List of Parts and Accessories

1. Manual.
2. Dataman S3.
3. Write Lead 2mm plug to Minihook.
4. EMULead - Ribbon cable with 28 pin dual-in-line plug.
5. HELP ROM.
6. Mains charger.

Confirming that S3 works

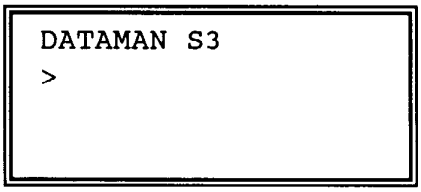
Switch it on and rotate the display thumb-wheel away from you to its full extent, which darkens the display. Adjust the display for best contrast. (Depends on angle and temperature). If a HELP program is loaded, S3 will display the version number:—



If you see this message then you may carry on and use S3. Store the HELP ROM somewhere safe, preferably in a piece of metal foil or conductive foam.

Loading the HELP ROM

Reloading S3's software program from HELP ROM is not something you should need to do to a brand new product. In fact you should never need to do it at all, except when a new version of the working program is to be loaded. The procedure is as follows: Press the RESET button by poking something of small diameter through the hole in the case above the ON/OFF switch - the write lead plug is suitable. You will see this message:



You will notice that the version of HELP is not identified, because there is no HELP program loaded.

If there is nothing in the display turn up the contrast. If still nothing, press the RESET button. If this does not work, perhaps the battery is totally flat. Switch off, then plug the charger in with the RESET button depressed. Leave it for a few minutes then try again. When you see the message, press **help**

```
DATAMAN S3
>INSTALL HELP
```

Put your HELP ROM in the socket and press the ENTER key. S3 will load the program contained in the HELP ROM. Then it will restart and run the program. If the HELP program loads correctly, S3 will introduce it and display the version number.

When the RESET button is pressed, S3 returns to low-level BIOS MODE, running in the microprocessor's masked ROM. It is not intended that you should use S3 in BIOS MODE - although it will perform a limited number of instructions without a HELP program. If you want to know about BIOS MODE there is more later.

NB. In theory you will never need to load the HELP program. It will be present in memory when you buy your S3 — and it will remain there, because S3's memory is permanent and continuous, for both programs and data. But there is the possibility that you will want to run different software in S3 at some future time, or make your own custom version.

Fundamentals

The manual is laid out to make it as painless as possible for you to extract the information you need. It proceeds from the general to the particular, so you should read all the first part in sequence and later parts when required. Functions are explained under headings in the order they appear on the keypad. Deeper technical details are handled at the end.

Automatic Power-down

S3 goes to sleep if ignored—after 30 minutes it turns the display off and enters a powered-down mode. For the last 15 seconds it makes beep-beep noises. It will switch itself off at the end of this time if you do not switch it off first. If you press a key during the beeping, power-down is prevented. No data is lost by power-down, but you have to actually switch-off-and-on-again to get the power back.

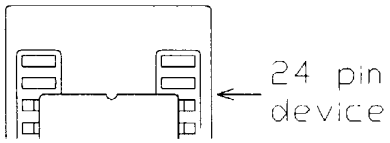
S3 also powers down when it believes that the battery is getting too low—less than 8.4 volts. At this level, data and program can be preserved, but nothing else works. The only cure is to charge the battery.

The PROM Socket

The ZIF (Zero-Insertion-Force) socket is used to hold PROMS when programming. The ZIF is also used to load new programs into the TPA (Transient Program Area). S3 can use PROMS much the same as a computer uses a disk-drive: they are a permanent storage medium, which contains programs to be loaded into

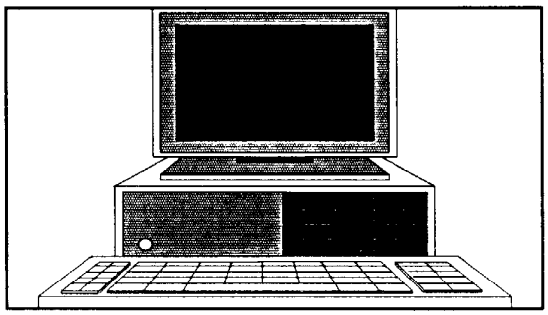
system RAM—such as the program loaded from the HELP ROM.

When the socket is not being addressed, no power is applied to it. PROMS can be inserted at any time, except during an operation like HELP, LOAD, BURN, TEST or COMP, which obviously read the PROM. When S3 is waiting for a command, or performing any function which does not involve reading a PROM, the socket is "cold"—it has no supply voltages. Even when the socket is being addressed directly, it is only powered-up for sufficient time to read the data, because PROMS take power from the battery.



PROMS with 24 pins must be inserted at the "bottom end" of the socket - the upper two pin-sockets of each side should be empty.

Computer Operation



There are two ways of using S3. The obvious way is to enter commands by pressing keys and reading the 80-character LCD. The other way is to attach an RS232 interface lead and enter commands from your computer and see the results on the screen. There are only minor differences between Stand-alone and Remote operation, and each key on S3's keypad has an equivalent two character serial command. S3 always responds to the requesting device: keypad commands produce responses in the LCD, computer commands produce screen responses. S3 will respond to both devices from the command prompt; no switching of modes is necessary. If you disconnect the RS232 interface during a command it may be necessary to switch-off-and-on-again to regain control at the keyboard. **Once a function has been entered from either the computer interface or the keypad, the other Device is ignored. Otherwise there would be confusion!**

Glossary: Words and Concepts

WYSIWYG (pronounced Whizzi-wig) is an acronym for What You See Is What You Get. It is a convenient way of expressing the principle of communication used in S3. What you see in the display is intended to be a true record of events.

The message **ESC** in the display means that the **ESC**ape key was pressed during or before execution; the previous command was not completed or maybe not even started.

The **ZIF** means the Zero-Insertion-Force socket on the front panel.

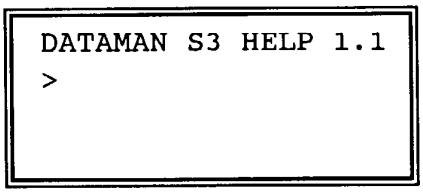
The **Keypad** refers to S3's 45 keys. The **Keypad** repeats automatically, when a key is held down. The delay after the first entry is longer than that after subsequent entries, to prevent false repetition. When data is being entered by repetition, the flashing block cursor changes to a steady underline cursor, so that progress is easier to follow.

The **Keyboard** refers to the remote terminal.

The word **Key** means input from either Keyboard or Keypad.

The **LCD** is S3's own Liquid Crystal Display. The **Screen** means the remote terminal screen.

The **Display** means either the terminal screen or S3's liquid crystal. Outputs are shown boxed in the text, meaning that this is literally what you will see:



The **Command Line** means the display line which starts with a prompt **>** .

An operation will be performed if you press ENTER, or be aborted if you press ESCape.

Commands which are non-destructive (do not change anything) are actioned as soon as you press the key, without waiting for ENTER.

A **Digit** is always Hexadecimal - never Decimal.

An **Address** defines one location in memory (expressed as 4 digits).

A **Parameter** is a set of **Digits**, two for a **Byte**, four for an **Address**.

A **Block** means contiguous bytes of memory from **Start** address to **End** address inclusive.

Backspace and space are used to edit parameters in the command line from the terminal keyboard, equivalent to ⊟ **and** ⊟ **on the keypad.**

ESCape aborts a command, even if it is already running. That part of the command which is already done cannot be undone, of course.

ENTER (or RETURN) accepts a command as seen.

ESCape or **ENTER** may be pressed anywhere in the command line. If the parameters have

been altered, ENTER accepts them as seen but ESCape restores the originals. One exception to this rule is SECTOR parameters, as described below, which retain their new values even if the ESCape key is pressed and the command aborted.

A **SECTOR** is a Block equal in size to the PROM as configured, entered as two Digits, numbered from 00. e.g. if configured for 2764 then SECTOR 00 extends from 0000 to 1FFF and SECTOR 03 extends from 6000 to 7FFF, inclusive.

SECTORS are convenient for programs which are larger than a single PROM, and must be programmed into, or copied from, a SET of PROMS.

```
>
>LOAD 2764-FUJ
SECTOR 03=6000,7FFF
```

The START and END addresses normally define the whole SECTOR. They can be edited to limit the effect of functions to less than a whole PROM or a whole SECTOR. There are rules for editing the SECTOR parameter line:

- If the SECTOR is changed to another value, then the START and END addresses will change automatically to the new boundaries.

- If the START address is changed to fall in a different SECTOR then the SECTOR changes and the END address adopts the new boundary.
- If either the START or the END address is moved away from the boundary, the = sign will change to a **X**.
- S3 refuses to accept a value for the END address outside the current SECTOR.

Only one set of SECTOR, START and END addresses are stored and shared between those functions which use these parameters.

Pressing the ESCape key when editing the SECTOR parameter line aborts the command, but leaves the SECTOR, START and END as seen—it does not restore their original values.

Paged Mode EPROMS like 27513

A **Page** has a special meaning for EPROMS like the 27513, which is divided into pages of 16K bytes. Pages are selected by writing the page number on the data bus. PAGE is output after the SECTOR parameter line, when relevant, and may be edited in the same way. The 27513 mentioned has pages 0 to 3, but the software covers pages 0 to 7 to accommodate bigger PROMS. If the user specifies page numbers which do not exist in the PROM, the highest select-bit is ignored.

```
BURN 27513-INT
12.5V TO PIN 22
SECTOR 00x0123,07FF
PAGE=0
```

Each PAGE is handled like a separate PROM. Programming does not continue from one PAGE to another. Different SECTORS must be programmed into different PAGES and each operation must be individually specified.

Like the SECTOR parameter, the chosen PAGE remains set even if the command is aborted by pressing the ESCape after changing it.

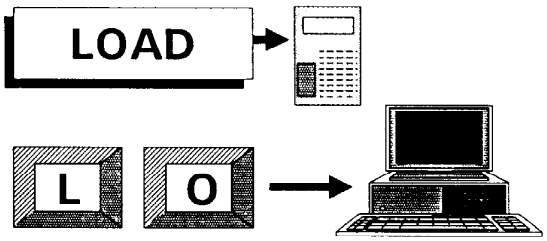
Function Keys

The function keys F1 to F5 are used for routines which are not an essential part of every S3 program. Later versions of the HELP ROM may offer different functions for these keys.

Beeping

S3 has a variety of noises: a single tone is made on acceptance of a key, a double tone as a rebuke that a key is not acceptable. During the execution of those commands which take several seconds, tones are emitted every half-second to tell you that the program is working—except in the BURN routine. Serial transmissions are audible and the baud-rate is easy to recognise. The system has variables which can be modified to alter or remove the tones made by the beeper.

LOAD




```
>
>LOAD 2764A-AMD
SECTOR 01=2000,3FFF
```

LOAD (SECTOR) (START) (END)

See Glossary for the special meaning of SECTOR, START and END.

LOAD copies the contents of the PROM in the ZIF into USER-RAM. The area copied and the destination is defined by the START and END addresses.

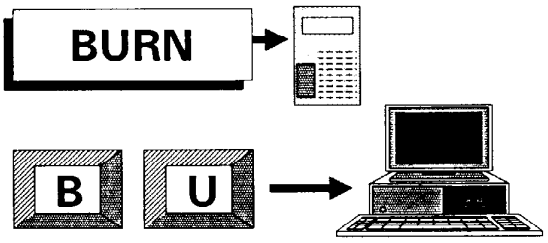
S3 must be configured for the right type of PROM. LOADing memory does not apply program voltages or program pulses to the PROM. PROMS of the same size and type have different configurations for programming, but they can be read by the same procedure. If the only difference is the manufacturer and you

intend to copy say, 27256 PROMS, of one manufacturer into similar blank PROMS made by another, then it is okay to leave configuration set to the PROMS you are actually programming. You need not change configuration just to read the originals.

A warning message gives the PROM type. If the ESCape key is pressed the process is aborted. If the ENTER key is pressed the process goes ahead.

The example given will copy a 2764 in the ZIF, addresses 0000 to 1FFF to USER-RAM addresses 2000 to 3FFF.

BURN




```
>BURN 2764-FAST
21.0v to PIN 1
SECTOR 01x3230,3FFF
```

BURN (SECTOR) (START) (END)

See Glossary for the special meaning of SECTOR, START and END.

BURN programs a PROM in the ZIF with the contents of the USER-RAM between the START and END addresses.

Chip-erase is not applied to EEPROMS if they cannot be completely erased, i.e. if less than a whole SECTOR is defined.

If S3 powers down during BURN, because lack of battery-power is sensed, the PROM will still be programmed correctly up to the point where the program aborted.

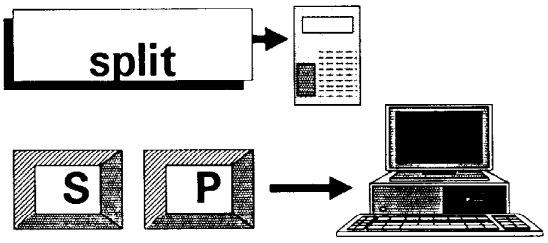
S3 must be configured for the right PROM. A warning message gives the PROM type, the program voltage and the pin to which it is applied. If these are wrong, the PROM could be destroyed. **Note** that pin number refers to the PROM, which can have 24 pins, not the socket which has 28.

If the ESCape key is pressed whilst editing the parameters or even after the programming starts, the process is aborted at that point.

To start the BURN cycle, press the ENTER key. When the cycle is complete, the program jumps to COMPARE for verification.

The example given will program a 2764 in the ZIF, addresses 1230 to 1FFF with the contents of USER-RAM addresses 3230 to 3FFF. The rest of the PROM is ignored during programming and subsequent testing.

SPLIT




```
>
>
>SPLIT
```

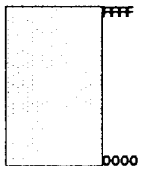
Puts all the odd bytes in the top half of memory and all the even bytes in the bottom half. Split moves all 64K, (except location 0000 and location FFFF!)

When a microsystem has a bus which is 16 bits or 32 bits wide, then it will address more than one ROM in parallel. But the assembler produces the code in serial fashion, with high bytes and low bytes alternating. It is possible that S3 will receive serial code and must put every other byte in a different PROM. Splitting the code moves the bytes to a contiguous SECTOR of memory which may be burned directly into PROM.

How memory splitting works

Example: Fitting 64K x 8 file into 16K x 32

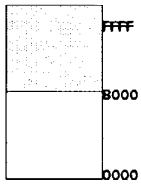
Load 64K



Split memory

ODD address code

EVEN address code



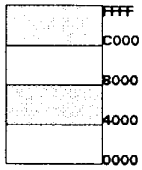
Split memory again

Sector 3 = 0003, 0007 &c.

Sector 2 = 0002, 0006 &c.

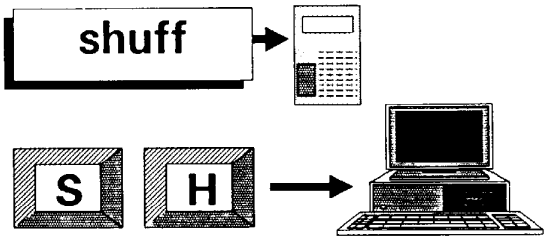
Sector 1 = 0001, 0005 &c.

Sector 0 = 0000, 0004 &c.



Program SET of four 27128 with Sectors 0,1,2,3

SHUFFLE

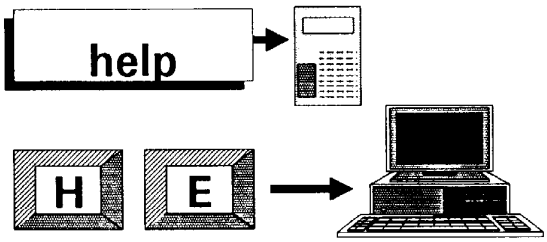



```
>
>
>SHUFFLE
```

Shuffle is the opposite of split. The top half of memory becomes the odd bytes, the bottom half the even bytes. Every byte in memory is moved, except the first and last—0000 and FFFF.

If PROMS are to be loaded into USER RAM and the code inspected or disassembled there, or sent to a remote computer through the RS232 interface, in a 16 or 32 bit system it must be combined so that the bytes are successive. This can be achieved by loading the different PROMS to different SECTORS and shuffling. The destination is defined by the START and END addresses.

HELP




```
>
>
>INSTALL HELP
```

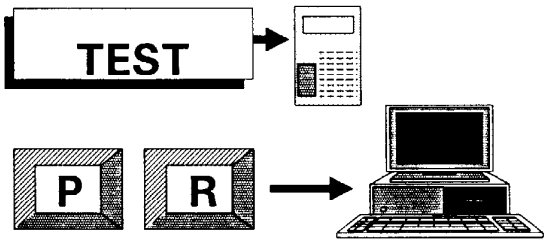
HELP is used to COPY new software from a "HELP ROM" in the ZIF socket to the Transient Program Area. The program runs automatically when it has loaded.

When you see the message, place a HELP ROM in the front panel socket and press ENTER. To abort the command, press ESCape.

The HELP instruction does not move all the code in the HELP ROM into the TPA area: if it did the stack would be overwritten in the process and the program would crash. Instead 1FD8 et. seq. contains pointers which show which code must be moved. Each pointer is prefixed by a 42 byte: 42 is used to indicate

that there is a block to be copied (Why 42? Well, any byte could be used and 42 has no real significance, except to the software engineer who is a Douglas Adams fan....) The four bytes following give START and END addresses of the block. If there is another block then follows another 42 byte, followed by another START and END, and so on. When the HELP program has been loaded, S3 resets and runs it.

PRETEST




```
>
>PRETEST 2764-50mS
SECTOR 04=8000,9FFF
```

PRETEST (SECTOR) (START) (END)

See Glossary for the special meaning of SECTOR, START and END and editing of the parameter line.

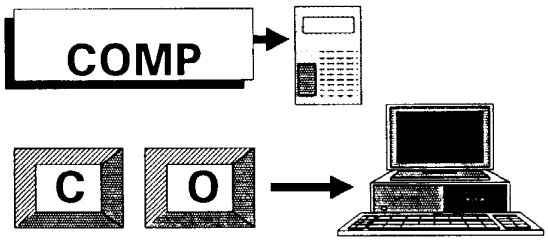
PRETEST compares a PROM in the ZIF with the contents of the USER-RAM between the START and END addresses. The purpose is to check whether EPROMS must be erased with UV light prior to programming. Bits are erased all high, and may only be programmed from high to low. Locations which cannot be programmed are reported. If a PROM contains locations which are already programmed, but the PROM will accept the new program, the

message **WILL BURN** is displayed If the PROM is actually blank—it contains all FF bytes—then the message **BLANK ROM** is displayed: a message you will also see if the ZIF is empty.

S3 must be configured for the right type of PROM. A warning message gives the PROM type. If the ESCape key is pressed, whilst editing the parameters, the process is aborted. To start the PRETEST, press the ENTER key.

The example given will PRETEST a 2764, addresses 0000 to 1FFF to see whether it will correctly program with the contents of USER-RAM addresses 8000 to 9FFF.

COMPARE




```
>
>COMPARE 27256-QP
SECTOR 00=0000,7FFF
```

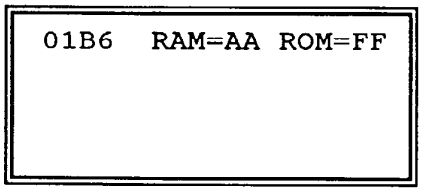
COMPARE (SECTOR) (START) (END)

See Glossary for the special meaning of SECTOR, START and END and editing of the parameter line.

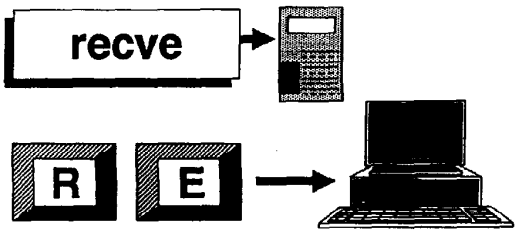
COMPARE matches all data in the PROM with the contents of the USER-RAM between the START and END addresses. Locations which do not match are reported. If the PROM matches the block exactly, the message SAME is displayed.

S3 must be configured for the right type of PROM. A warning message gives the PROM type. If the ESCape key is pressed, whilst editing the parameters, or after any reported

mismatch the process is aborted. To start the COMPARE, or restart after a reported mismatch, press the ENTER key. A mismatch between PROM and USER-RAM is displayed RAM ADDRESS, RAM BYTE, ROM BYTE as follows:



RECEIVE




```
>
>
>R-ASCII 0123,4567
```

S3 receives files in INTEL, MOTOROLA, TEKHEX, ASCII or BINARY format, as defined by the CONFIG routine. You are prompted for START and END addresses in ASCII and BINARY only, because these transmissions do not contain any destination for the data. If the transmission continues after the specified END address, the remainder of the file is "run out" i.e. received, but discarded.

The **keypad** ESCape key will abort transmission whilst waiting for a file, or during the reception of a file. After the ESCape key, S3 waits for a second to make sure that no more data will be sent—

otherwise such data would be misinterpreted as keyboard instructions. Also the sending terminal might lock-up if RTS is not held true until the End-Of-File is reached. If more data is sent then S3 stays in the routine—the user should stop the file transmission from the sending end.

Error-check failures built into the file-format are reported e.g. Checksum errors are reported together with the address at which the error was detected, which is usually the end of the row in which the error occurred. When a file is received correctly S3 displays the last address where code was stored.

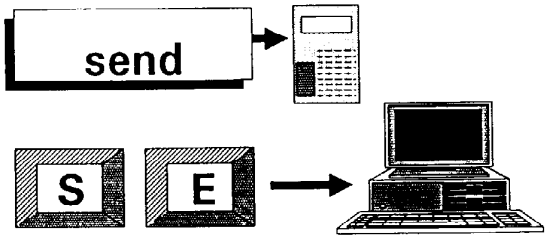
Audible Output

S3 makes a noise when receiving data. When setting-up this provides useful feedback, but later it might prove annoying. The noise can be turned off by editing HELP RAM location E002 from 5F to 7F. To avoid having to make this change every time you reload your HELP ROM, you can make your own version - see page 53

Returning to Command Mode.

Before returning to command mode, S3 waits until the data input has been inactive for at least one second. This is to "run-out" any extra data to avoiding hanging-up the sending computer. When sending from a batch-file, a delay of over one second must be inserted before the next command, to prevent S3 seeing it as "garbage".

SEND




```
>
>
>S-INTEL 0000,1FFF
```

SEND (START),(END)

Transmits the block from START to END inclusive through the serial interface. The FILE FORMAT and BAUD RATE are chosen in the CONFIG routine.

If CTS is true the transmission proceeds. If CTS is FALSE S3 displays "IGNORE CTS", if ENTER is pressed at this point the transmission proceeds regardless of CTS, but if ESCape is pressed the transmission is aborted. Assuming that the transmission is proceeding with regard to CTS, if CTS becomes false the message AWAITING CTS is displayed until CTS

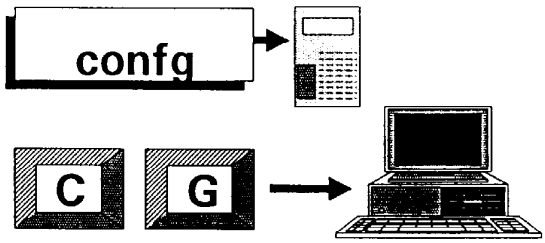
becomes true again. These messages go to the LCD, of course, not the terminal.

At any point the transmission may be aborted by the user pressing the keypad ESCape key. ESCape is polled at the end of each line, so it is necessary to hold it down for long enough for the program to see it, and at low baud-rates this takes a little while. The transmission sends an End-Of-File character (1A) and exits from the routine.

Audible Output

S3 makes a noise during data transmission. When setting-up this provides useful feed-back, but later it might prove annoying. The noise can be turned off by editing BIT 1 of location FFA4 of the TPA to 0. This location is usually 3, so set it to 1. (BIT 0 controls whether the CTS line is checked or not during transmission of data).

CONFIGURE




```
>
>CONFIG
37 2804-GI     12.0V
```

(ALGORITHM) (PROM TYPE/MANUFACTURER) (PROGRAM VOLTAGE)

The CONFIGURE routine sets the PROGRAMMING ALGORITHM, FILE TYPE and RS232 BAUD RATE, in that order.

It is possible to ESCape at any point but any changes stand - the settings do not revert to original values. Remember that S3 works on the WYSIWIG principle—What You See Is What You Get. If you read something in the display, then it is true. Therefore it is not necessary to proceed to set up FILE FORMAT or BAUD RATE just to change the PROM.

Choosing by scrolling the choices through the display works well for a human interface. However to allow configuration from a batch-file or macro-maker, such as "Prokey" or "Smartkey", each item is assigned a two digit hex number, starting from 00.

PROM Type.

There are three ways of configuring S3 to read and program a PROM correctly:

1. If you know the hex number of the algorithm, enter it. The number is shown when you scroll through the ALGORITHMS as described below, and listed in the ALGORITHM TABLE which goes with the version of HELP you are using.

2. Use the ⇦ and ⇨ keys to scroll through the list of ALGORITHMS. (At a remote terminal, use BACKSPACE/SPACE).

You have to know the part-number and the program-voltage of the PROM - if you are unsure, get a data sheet from the manufacturer.

3. Press the ⇧ key,('H' key of a terminal) to read the "Intelligent Identifier" in the PROM by raising Address line 9 to 12 volts. "Intelligent Identifier" is also called "Silicon Signature" and other names by manufacturers. It consists of two bytes, the first gives the MAKER, the second the PROM type. If S3 reads a signature it cannot find in the table, the message "UNKNOWN" appears.

File Type.

```
>CONFIG
3E 5516-10ms
FILE: INTEL
```

S3 will receive files serially transmitted in a standard format. The formats supported are INTEL, MOTOROLA, TEKHEX, ASCII and BINARY (numbers 00-04). A detailed description of each format is given later.

The ⊟ and ⊟(BACKSPACE/SPACE) scroll through the different formats and set-up the one displayed. You must leave the right format in the display—what you see is what you get, even if you press ESCape instead of ENTER.

Baud Rate.

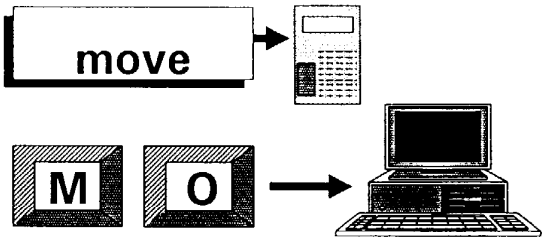
Baud rates of 300, 600, 1200, 2400, 4800 or 9600 may be chosen (numbers 00-05).

```
3E 5516-10ms
FILE: INTEL
BAUD: 9600
AUTO - HIT SPACEBAR
```

The ⊟ and ⊟ (BACKSPACE/SPACE) scroll through the choices, and you may use ESCape

or ENTER to complete the set-up. Auto-selection of Baud Rate is possible too. At this point, if your computer is running a COMMS program in terminal mode and is attached to S3 you can set baud-rate by pressing the SPACEBAR on the computer's keyboard.

MOVE

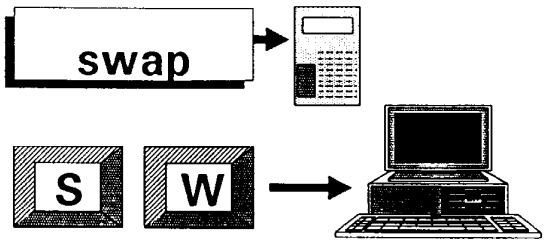



```
>
>
>MOVE 0000,1FFF,3000
```

MOVE (START),(END),(DEST)

Move memory block within USER RAM. START and END define the block, inclusively. Blocks can be moved forwards or backwards. Overlapping causes no problem. The contents of the original block are left unchanged - except where the destination block overlaps.

SWAP



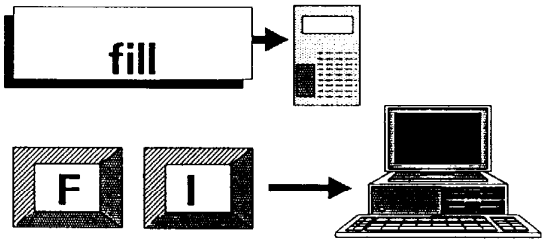

```
>
>
>SWAP FE20,FE30,FF00
```

SWAP (START1),(END),(START2)

Exchanges the code in two equal length memory blocks.

The block from hex address FROM (START1) to hex address (END) is exchanged with a block of same length starting at hex address (START2). Swapping overlapping blocks does not do anything particularly useful, but you are not prevented from doing it. Swapping twice without changing parameters will restore the original order of code, provided there was no overlapping.

FILL

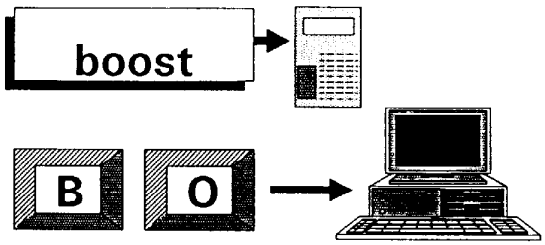



```
>
>
>FILL 0000,1FFF,AA
```

FILL (START),(END),(BYTE)

Fills each byte of memory with hex value (BYTE) between hex address (START) and hex address (END) inclusive. FILL works "round the corner" i.e. if (START) is greater than (END) then addresses between (END) and (START) are not filled - but the remainder of memory is filled, the effect being that you can fill memory excepting the excluded block.

BOOST CHARGE




```
>
BOOST CHARGE IS ON
ENTER=ON ESC=OFF
```

The discussion which follows assumes that the charger is connected and switched-on.

When BOOST is OFF, 50ma flows constantly into S3, whether the system is operating normally or in standby mode. This is the 14 hour charge rate. If you work a 10 hour day, then the battery will recharge fully if left on from the end of one day to the start of the next.

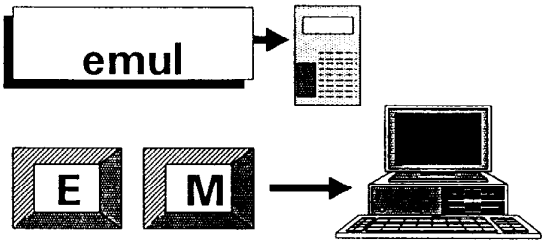
When BOOST is ON, a total of 225ma flows into S3, which will fully recharge the battery from flat in 3 hours. Reasonable working capacity will be restored in much less time. BOOST charging is provided for situations where the battery has be come discharged and

must be refreshed quickly, but it does no harm to the battery—it may even be beneficial.

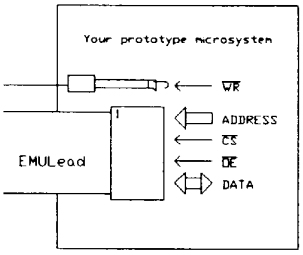
When S3 is in use the operating current will reduce the charge that the battery receives. Editing, for example, takes 10ma and the battery will only receive the remaining 40ma. Emulating and programming take more than 50ma, so the battery will not receive any charge—but the battery drain will be reduced by 50ma.

See later section for fuller discussion of battery capacity and recharging

EMULATE




```
>
>
>EMULATE 2764
```



WR	OE	CS	
1	0	0	READING
0	1	0	WRITING
NA	22	20	pin no. of 28
NA	20	18	pin no. of 24

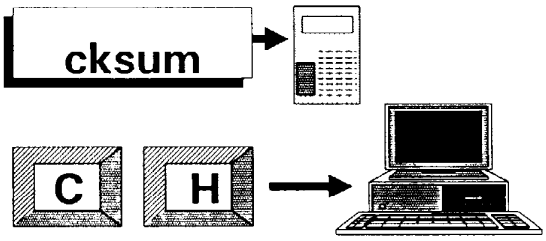
Emulates the PROM that was set in the CONFIGURE routine. If the Write Lead is connected to the microprocessor then writing into USER RAM through the EMULead is possible. The ESCape key stops emulation and returns control to the key-

board. (other keys may do the same.) See later section for full discussion of memory-emulation.

EMULate from SECTOR zero.

The EMULead does not decode the PROM address - any more than the PROM itself does. It sees the lowest address of the PROM at location zero in the RAM. If the SECTOR of memory which is intended for emulation does not happen to start at location zero, then it should be moved (or swapped) to location zero, for emulation to work.

CHECKSUM




```
>
>
>CHECKSUM 2000,3FFF
```

CHECKSUM (START),(END)

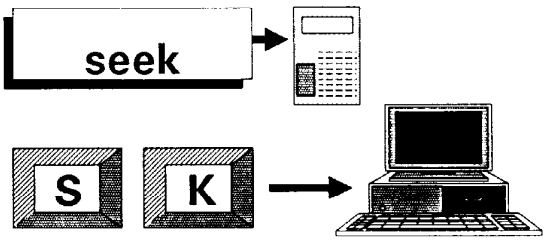
Adds together all bytes in a chosen block from START to END inclusive and presents the answer as a six digit hex number.

A CHECKSUM provides a "signature" for identifying and labelling programmed PROMS. Master PROMS should be labelled with their CHECKSUM to ensure that they have not been over-programmed by mistake - and neither gained nor lost bits whilst in storage. The CHECKSUM of a device which has programmed wrongly, gives useful information: if it is too low than the device has extra bits programmed and was possibly under-erased. If

the CHECKSUM is too high then the device has bits which will not program - it may be damaged or the wrong algorithm is being used.

A PROM used as a MASTER to prepare copies for sale as firmware should be marked with its checksum. The checksum of the slaves should be taken and matched against the master.

SEEK



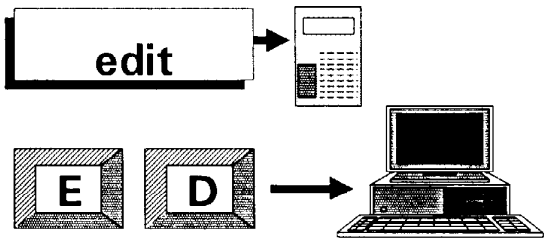

```
>
>SEEK 0123,4567
12,34,X3,A5,XX,FF
```

SEEK (START),(END),(6 BYTES)

Searches through a block of code from (START) to (END) for a sequence of up to six bytes. Bytes are pairs of hex characters, or "wild" characters, which can have any value. **Wild characters are entered by pressing the SEEK key again (or terminal X key); they are represented by X in the seek line.** Wild characters may be used freely for half bytes or full bytes.

All the matching addresses are listed in the display, when they are found. If there are many matches, use ESCape to terminate SEEK when you have seen enough.

EDIT




```
>
>
>EDIT 1A00
```

EDIT has been split into two different utilities, to make best use of the LCD and remote terminal.

Stand-Alone Editing.

S3 uses all of the LCD in the edit routine.

```
ADDR=1A00 ABCD EFGH
4142 4344 4546 4748
494A 4B4C 4D4E 4F50
5152 5354 5556 5758
```

The top line shows the cursor address, which changes when you press one of the four cursor keys or when you enter data. On the right of the top line is the current line in ASCII. (To translate to ASCII, bit 7 is disregarded). Values below 20H and 7FH are represented by a . (point). The cursor appears as a flashing block, which changes to an underline cursor when you hold a key down to move quickly through memory. If you move off the top or bottom of the screen the display scrolls up or down by one line. Editing is immediate: once the code is changed, there is no way of recovering the original.

Remote Editing

The remote terminal screen has more columns than the LCD and will look like this:

0000 42 43 44 45 46 47 48 49

The ADDRESS is followed by the DATA in eight memory locations. As usual with terminal editing the SPACE/BACKSPACE keys move the cursor through the data. Hex keys 0-9 and A-F will change the data.

When the ENTER key is pressed the ASCII equivalent of the lines is shown at the end and the next eight bytes presented for editing:

0000 42 43 44 45 46 47 48 49 BCDEFGHI

0008 00 B1 46 92 8F FF 91 0C

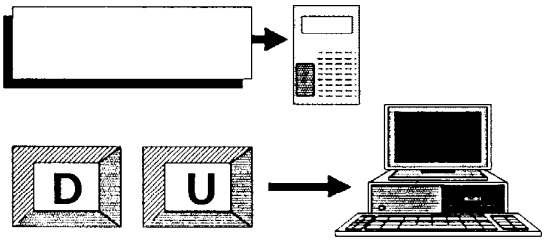
Only ASCII characters 20H to 7EH are sent through the interface, after the MSB has been stripped. The others are sent as a point (full stop), because control characters will be inter-

preted by the terminal instead of being printed. e.g. 7FH usually performs a RUBOUT.

To get out of the edit routine, press the ESCape key.

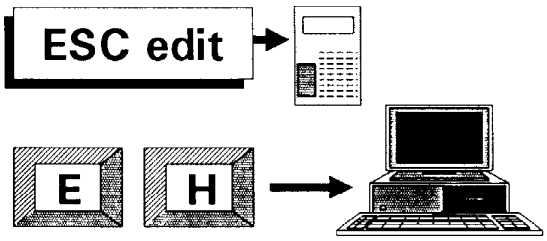
DUMP

Dump works only via terminal and is similar to



EDIT above.
It dumps 8 bytes per line in HEX with ASCII equivalents. Inputs required are START address and number of bytes—up to FFH or 255.

ESC edit(system)




```
>
>
*EDIT 1A00
```

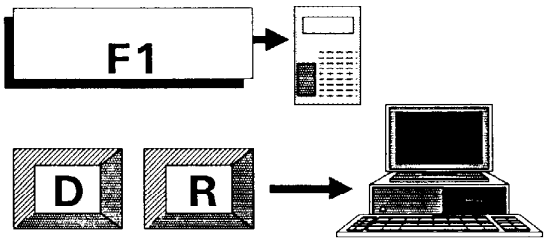
System memory space can be seen and altered using a "shifted" version of edit. System memory space is where the working program is stored when loaded from a HELP ROM, in the Transient Program Area which extends from E000 to FFFF. If the ESCape key is pressed from the command prompt, the prompt changes from a > to a *. EDIT then works on system memory. The LCD behaves exactly as the normal edit routine, except that there is a full-stop in the lower right corner of the screen.

WARNING *EDIT allows changes to the working program and system variables. S3 can be

crashed by editing the running program, if indiscriminate changes are made to the stack or code. Do not edit between 1000 & E000 because odd displays and other unwanted effects may be the result. Regard the TPA as a "NO GO" area unless you are certain of what you are doing. You can recover from any changes by reloading the HELP ROM.

```
ADDR=E000 DATA MAN
4441 5441 4D41 4E20
53B3 0838 3838 0C06
0129 1267 0008 85F9.
```

F1 - DUMP ROM

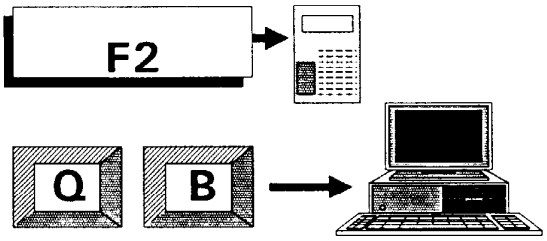



```
>
>
>DUMP  2532 1A00
```

```
ADDR=1A00 ABCD EFGH
4142 4344 4546 4748
494A 4B4C 4D4E 4F50
5152 5354 5556 5758
```

This function is similar to edit, except that it works directly on the device in the ZIF socket. For economy of code we have used the edit routine. You cannot, of course, change any of the data. Any apparent alterations only affect the buffer and will change back when the display scrolls.

F2 - QUICK-BURN




```
>Q-BURN 27256-FAST
12.5V TO PIN 1
SECTOR 00=0000,7FFF
```

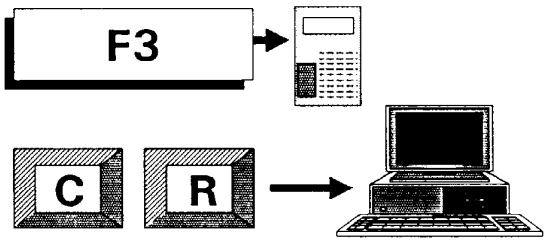
QUICK-BURN burns a PROM more quickly when it has only a few locations which require over-programming. It is strictly for development purposes. Each address is examined and skipped if the data matches. EEPROMS are not chip-erased. Locations which would not pass PRETEST will not program, of course.

QUICK-BURN is not always quicker: if the PROM requires complete programming or a lot of changes, the usual BURN routine will be quicker. PROMS which usually program with QuickPulse or FlashRite Algorithms show no significant improvement in programming time—the compare-cycle takes just as long as

programming. However, for making a few byte changes in a PROM which usually takes a minute or more to program, QUICK-BURN saves time.

QUICK-BURN is not recommended for programming PROMS which will be sold—a location which pre-compares cannot be guaranteed to have been previously programmed correctly.

F3 - CHECKSUM ROM

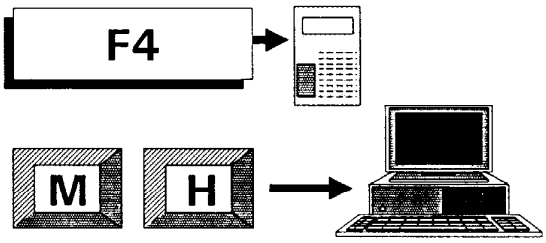



```
>CHECKSUM 2532
SUM = 01FEA9
>
```

Performs a checksum on all the locations of a ROM in the ZIF socket. The CHECKSUM ROM function allows the user to take the checksum of a ROM device without first loading it into RAM, thus avoiding over-writing USER MEMORY and the SECTOR PARAMETERS.

The PROM TYPE as specified in CONFIG is displayed.

F4 - MAKE HELP




```
>
>
>MAKE HELP
```

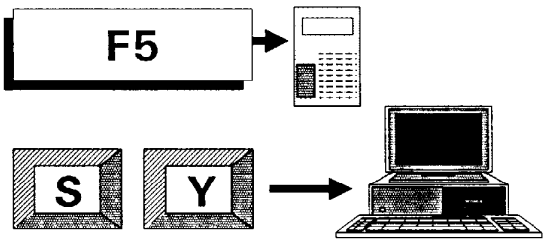
MAKEHELP lets you make a HELP ROM, with your own preferences incorporated. The code in the TPA area is moved into USER RAM, from E000 upwards, and certain framing information (42 bytes—see HELP) added. All you have to do after MAKEHELP is to configure for a 2764, then program it with SECTOR 7 from E000 TO FFFF.

Such a HELP ROM will restore the ALGORITHM, BAUD-RATE, FILE-TYPE and other system-variables which were set when you made it. Command parameters are not restored.

Making a HELP ROM with your own configuration is easy, and useful if you have to lend your

S3 to a colleague, to restore all your own preferences when you get it back (if you do get it back).

F5 - SYSTEM CHECK



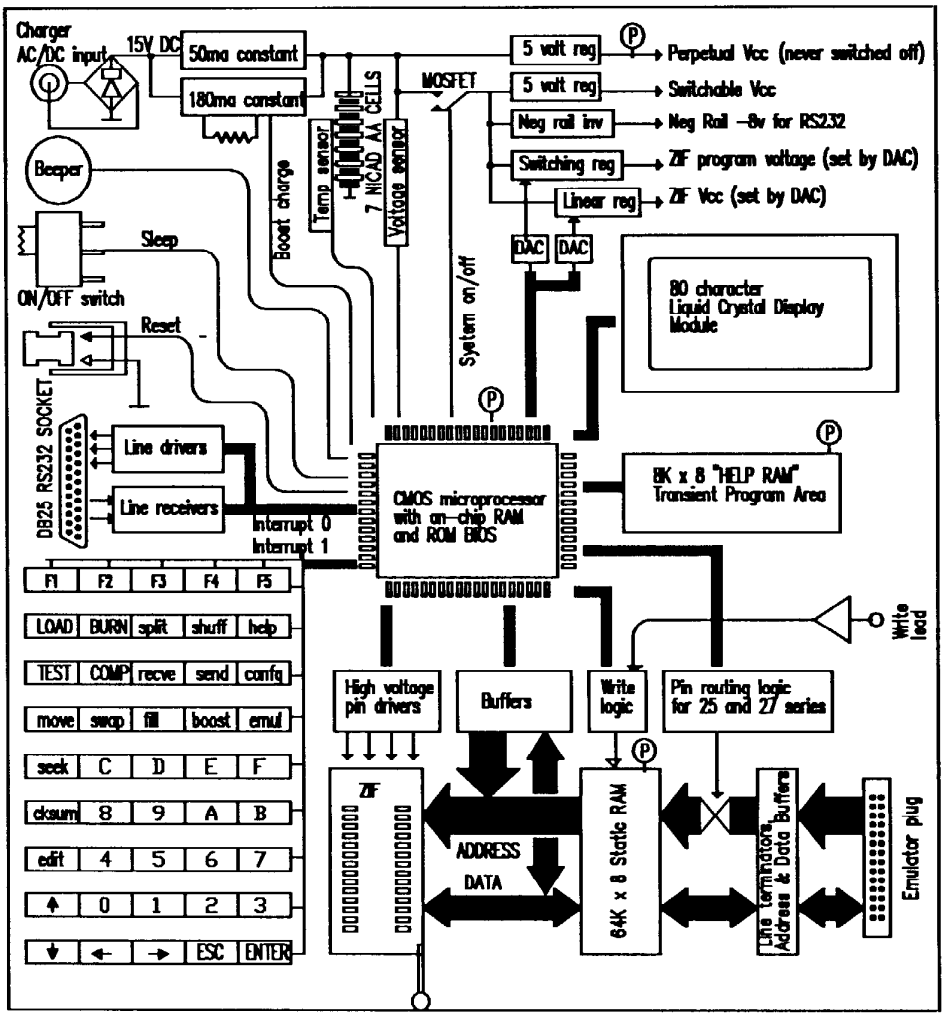

```
SYSTEM CHECK
BOOST CHARGE IS ON
INITIAL TEMP = 20
BATT=9.4V TEMP=22
```

SYSTEM CHECK continuously monitors battery voltage and temperature

If the boost charge is on, you will see the battery voltage and temperature rise as the battery charges. When the battery is fully charged the boost will switch-off.

To terminate the SYSTEM CHECK mode, press ESCape.

Block Diagram



BIOS Mode

This section is written for sake of completeness - it is unlikely that anyone not developing new software for S3 will care to use the BIOS mode.

If RESET is pressed S3 returns to a program running in the masked ROM of the microcontroller—this is called BIOS mode (Basic Input/Output System). Whilst S3 will run in this BIOS mode with a reduced instruction set, it is not intended to be used without a program loaded into the TPA (Transient Program Area). The BIOS contains subroutines which are used to handle input and outputs - RS232, Keyboard, Display etc. BIOS mode is used only for development of new programs. The TPA is used for temporary storage only. Displayed messages are reduced to the same two letters required at the remote RS232 keyboard. Some keyboard functions work only in a limited way - CONFG and SEND do not work at all. S3 will only program a 2764 type PROM, at 21 volts Vpp, the type used as a HELP ROM. RECEIVE works only in INTELHEX at 9600 baud, with two stop-bits, not one. The function keys do not function.

When S3 is first switched on, it examines a bit in a register which is retained whilst sleeping in power-down mode and decides whether it is a WARM START i.e. whether there is a HELP program to run or whether a COLD START into the BIOS program is necessary.

The HELP command loads a program into the Transient Program Area from E000 to FFFF,

from a ROM in the ZIF socket. The HELP ROM must be capable of loading after the RESET and HELP buttons are pressed. The BIOS configures S3 to handle a 2764, so the HELP ROM must be a 2764 or a PROM which will load from a socket configured as a 2764 e.g. Address range 0000-1FFF of a 27128 or 4000-5FFF of a 27256.

RS232 Serial Interface

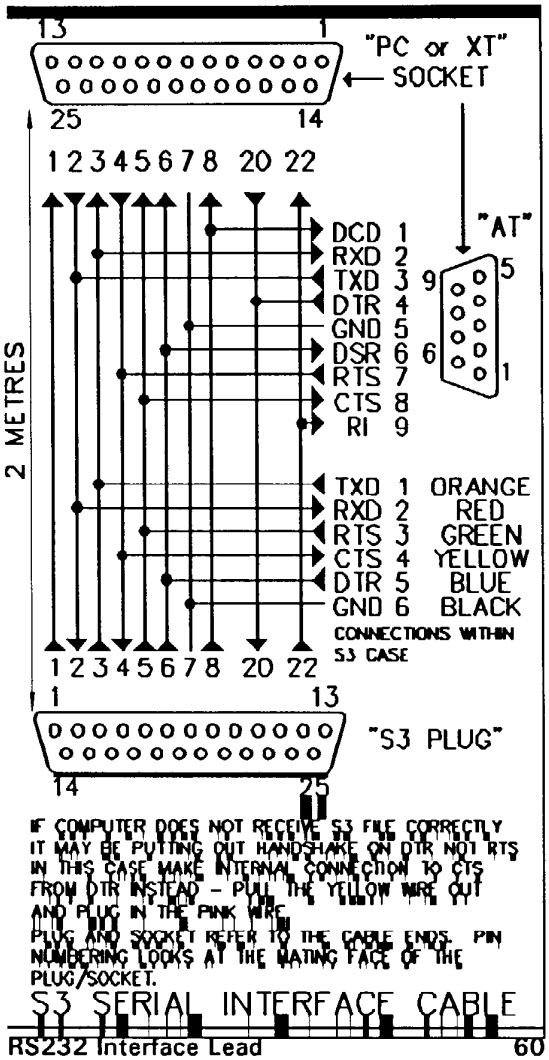
The serial RS232 interface uses an IC made by Motorola, part number MC145406, which meets RS232 standards of voltage and current on inputs and outputs.

Current can be sourced by the outputs into any load on the RS232 interface, but it is unlikely that such current will exceed 5ma. Nevertheless this current comes from the battery, so for maximum time between recharges, connect only those outputs which are needed. When S3 powers down, the RS232 voltages fall to zero.

Alterations to the Standard Interface

Switching the connections around is easy. If you take the back off your S3 you will find that the wires are pushed into a labelled connector. The DB25 socket can be exchanged for a plug, if you prefer. We have made our implementation as forgiving as we know how, easy to alter and willing to work under most circumstances. Even experienced engineers have trouble getting RS232 to work, because there are more exceptions than rules.

RS232 Interface Lead



Baud Rates

The baud-rate of a transmission is the reciprocal of the time used to send one bit. Asynchronous serial transmissions as commonly used by computers have extra bits to frame and check the data: a START bit, then the DATA bits, then an optional PARITY bit and then one or more STOP bits. Baud-rate indicates the fastest possible transmission speed: it does not indicate the actual transmission speed. You can send one byte per fortnight at any baud-rate. When a computer has reached its limiting speed for processing the data, increasing the baud-rate will not make it send or receive any faster—the gaps between characters just get bigger. Systems which claim very high baud rates—19200 or 38400—are often slow at sending data files. To prevent the sending device supplying data faster than the receiving device can digest it, the receiver prompts with a signal **Request-To-Send,** which the sender sees as **Clear-to-send** These signals are called **handshaking**. Leads used for serial communications must connect the handshaking signals properly, to prevent data getting lost. (Falling in the Bit-Bucket, as they say). As S3 can actually receive data-files at 9600 baud with 1 stop bit, the highest rate usually offered by a computer, without stopping the transmission, which means that there will be no problem even if **RTS** is not connected.

To establish the fact that a connection exists between the sender and receiver, the sender puts out a signal **DTR (Data Terminal Ready)** and looks for a signal **DSR (Data Set Ready).**

The receiving device puts out a signal **RTS (Request to send)** when it is ready to receive data, and the sending device looks for a signal **CTS (Clear to send)** before it sends data. When two computers are talking then **DTR** on one is connected to **DSR** on the other, **RTS** is connected to **CTS**. However it is rarely that easy. Most computers have signals named as above, but you can't guarantee they will work as described. The actual signal which starts and stops data output is likely either **DSR** or **CTS** and, because these are both inputs and on adjacent pins 5 & 6, it usually does no harm to connect them both to the incoming **RTS**. S3 has one active handshake signal in each direction, normally connected as **CTS** and **RTS**. **DTR** is provided too and is always true - it proves only that the cable is connected. The signal at the computer end which requests data is usually **RTS** or **DTR**, and you can probe these with a meter to find which one is at a high level. If they are both high then it is likely that **DTR** is wired at a positive level, and **RTS** is the active stop/go signal.

It may be possible to manage without any active handshaking, but some computers lock-up if they do not see the correct signals. S3 doesn't lock, it only warns you that no handshaking is present, and lets you carry on and receive data with no handshake. The implementation is as follows: at switch-on, S3 looks for **CTS**. If **CTS** is present then S3 puts out the introductory message to the serial interface as well as putting it in the LCD. Whether or not **CTS** was present at switch-on, S3 polls both devices, the keypad/LCD and serial channel for commands. When a com-

mand is received then the responses go to the requesting device.

Note: *Both devices are only polled simultaneously at the command prompt. If S3 is in the middle of doing something requested by the RS232, the keypad is ignored, and vice versa. No other method makes sense.*

S3 receives commands through the serial interface in an interactive way i.e. it expects you to wait for the resulting output. You normally expect computers programs to work like that and when entering commands and data by hand it causes no problem. However, in the rare circumstance that you wish to send S3 commands from a batch-file which does not wait for results, you must make sure that S3's **RTS** signal handshakes with your computer to prevent it sending commands faster than S3 can action them.

When sending files, remember that most computers cannot process data as fast as S3 can. To be safe, send at a slow baud rate or connect the **RTS** handshake line. You can usually see a bad transmission if you inspect the file after receipt - some of the lines will be short and contain bad characters.

Interfacing with a Computer.

Almost every S3 user will need to make a serial link with a computer at some time. Computers usually have a serial port, through which file transfers can be made, in much the same way as files can be transferred between disks. In fact the operating system file-copying routines can specify the serial device: MSDOS uses COM1: or AUX:, CP/M refers to RDR: and PUN: for example the command to transfer a file to S3 from an IBM-type PC is:

```
COPY FILE.HEX COM1:
```

The baud-rate, word-length, stop-bits and parity setting must have been set previously to the correct values. The command to do this is:

```
MODE COM1:9600,N,8,1
```

S3 must be set similarly, and the file-format must expect, in this case, an IntelHEX file which is decoded as it is received. Downloading files from the operating system is likely to work with no handshaking problems, because S3 will receive any file at full speed.

Sending files back is usually not so easy. Computers seem to implement handshaking properly on output, but not on input. It is surprisingly difficult to get any information on

this subject: the manufacturer's data tells you the names of the signals, but does not tell you that they do not work. Experiments show that the input buffer overflows at some point, usually at 64K, when the system transfers the buffer contents to disk without putting out a signal to stop the input. 64K characters is not 64K bytes, because a HEX file contains two ASCII characters for every data byte plus addresses, checksums and other odds and ends. In fact it is more like 26K bytes. For small PROMS this is enough. It is possible to send the whole 64K as 3 chunks, then patch it together with a word-processor and take out the two spurious End-of-File lines. **A much better solution is to use some kind of COMMS or TERMINAL program.**

Terminal Emulating Programs.

A TERMINAL sends the information you type at the keyboard through the serial port. It displays what comes back through serial port on your screen. When S3 is connected to your computer running a terminal program, it might seem that what you type appears on your screen: that is not true. What your see is what S3 chooses to send you in response: sometimes this is what you typed: sometimes it is not. Terminal programs usually let you send and receive files as well, with handshaking properly implemented, and that is all that is required for complete control of S3 by your computer. You might have a terminal program already, without knowing it. There is no such program as a part of the MSDOS or CP/M operating system, but many computer manufacturers provide one on the system disk:

Epson do, for example. Some software programs have a terminal mode built-in: Open Access, HomeBase, WordStar 2000+ etc.

Communications Software

A COMMS program is basically the same as a terminal program, but it has extra features which control a MODEM for sending and receiving "Electronic Mail" by telephone. If you buy a MODEM you are likely to get a Comms package bundled with it. With this combination you have access to a whole new world: PRESTEL, TELECOM GOLD, BULLETIN BOARDS, COMPUTER CONFERENCES etc. On Bulletin Boards there is useful software which can be downloaded and used with S3, such as cross-assemblers, code-editors and even Terminal and Comms programs. These are usually "shareware", which means the author expects a donation from you if you find his software useful. You can get the same programs on disk for a small copying charge from the Public Domain software libraries. Many of these programs are superior to those sold by the big software houses for hundreds of dollars. We recommend ProComm or Telix for use with S3, both available from Public Domain software/shareware libraries, whose addresses will be found in many computer magazines.

File Formats.

S3 sends and receives orthodox computer files.

Only a programmer with an unusual amount of patience would wish to enter a large quantity of code into S3 by keying it in hexadecimal numbers. Microsystems have in the past been developed by "hand-assembly": the translation of microprocessor instructions into machine code mentally without benefit of an assembler, writing the instructions into memory in hexadecimal using a keyboard and repetitively trying out the program until it works. In fact S3's ancestors, Softy1 and Softy2 were developed that way. Most programmers these days would use an assembler which permits the entry of code as instruction-mnemonics. The assembler creates a file of machine-code automatically, but the file is not always actionable code. Actionable object code which is placed in memory exactly as it is received is called BINARY format. Transmission formats usually have a certain amount of extra information, for example the ADDRESS to start loading the data, CHECKSUM bytes to validate transmission etc. S3 receives files in common formats, such as INTELHEX, MOTOROLA S, TEKHEX, ASCII or BINARY.

INTEL Format.

Lines of ASCII characters, expressing bytes as ASCII pairs, terminated with Carriage return and Linefeed.(0D,0A). The transmission is terminated by ASCII End-of-File (1A).

Pos	Character	–	*Remarks*
1	:	–	*Colon*
2,3	Byte	–	*Byte count in hex, max 20 (64 ASCII characters)*
4-7	Address	–	*Most significant byte first*
8,9	Byte	–	*Record type 00 = Data, 01 = End of File*
10-N	Data		
N+1, N+2	Checksum	–	*Inverted sum of all bytes in the line (forget the colon, linefeed and return) that is, make the line add up to 00.*

Example

```
:1100000044415441 4D414E2053332053455249414C73
:00000001FF
```

MOTOROLA S Format

Lines of ASCII characters, expressing bytes as ASCII pairs, terminated with Carriage return and Linefeed.(0D,0A). The transmission is terminated by ASCII End-of-File (1A).

Position	Character	–	*Remarks*
1	S	–	*letter S indicates start-of-record*
2	0,1 or 9	–	*Record type:* *0 = Header, may be followed by comments* *1 = Data* *9 = End-of-file*
3,4	Byte	–	*Byte COUNT in hex of rest-of-line (multiply by two for number of characters).*
5-8	2 bytes	–	*Memory ADDRESS for inserting data. MSB first*
9-N	Bytes	–	*DATA*
N+1, N+2	Checksum	–	*One's complement of sum of DATA, ADDRESS and COUNT.*

Example

S11400004441544 14D414E2053332053455249414C6F

S9030000FC

TEKHEX Format

Lines of ASCII characters, expressing bytes as ASCII pairs, terminated with Carriage return and Linefeed.(0D,0A). The transmission is terminated by ASCII End-of-File (1A).

Position	Character	–	*Remarks*
1	/	–	*Slash character for start of line*
2-5	2 Bytes	–	*Address, MSB first*
6,7	Byte	–	*Number of data bytes - not checksums.*
8,9	Byte	–	*Checksum of ADDRESS and COUNT by character in hex.(not by byte).*
10-N	Data	–	*Data bytes as ASCII pairs*
N+1, N+2	Byte	–	*Checksum of Data by character, not as bytes*

The terminating block has zero in the byte count field.

Example

```
/0000110244415441 4D414E205333205345524941
4C8F
/01000001
```

ASCII Format

Each data byte is translated into two hexadecimal ASCII characters, sent Most Significant Nybble first. ASCII is similar to INTEL, MOTOROLA and TEKHEX, except that nothing but the raw data is sent - no addresses, no checksums. After 32 bytes or 64 characters a LINEFEED and CARRIAGE RETURN are sent (0D 0A), which enables the transmission to be received by a serial printer ora terminal. At the end of the transmission an End-of-File character (1A) is sent.

ASCII is normally sent and received as an 8 bit transmission. The MSB is masked low when receiving, placed low when sending. If the sending terminal must use a 7 bit transmission, this will only work if a redundant bit is added, such as a parity bit, or two stop bits are specified.

When receiving, S3 asks for START and END addresses where the block will be stored.

NOTE. *INTEL, MOTOROLA and TEKHEX transmissions can be inspected by receiving them in ASCII format. Then you can see delimiters, addresses and checksums.*

Example

44415441 4D41 4E20 53332053 45524941
4C

BINARY Format

The file is sent as a succession of bytes. The data is not processed at all. S3 asks for START and END addresses when receiving. An ASCII End-of-File character is not sent when the format is set to BINARY because it would be received as data. Sending BINARY files from S3 is possible, but may be difficult to implement at the receiving end. The computer must expect a pre-defined size of block and know where to put it. When receiving a block in BINARY format, S3 has no way of knowing when the computer has finished sending. You must press ESCape when the transmission is ended.

NOTE Computers will not send a BINARY file through their serial interface, using PIP or COPY routines, unless you add a suffix: COPY needs /B, PIP needs [O]. Otherwise the transmission is aborted when an end-of-file (1A) character is seen.

Example

DATAMAN S3 SERIAL

The data in the example above is the same as in the other file examples, transmitted in BINARY CODE. BINARY reception is useful to check other file formats: every character that is sent appears in S3's RAM, so you can see what is really being received.

Battery and Charger

S3 is powered by a rechargeable battery of nickel-cadmium cells providing 8.4 volts or more at 500ma/hr. The battery is capable of the following typical performance when fully charged:

Standby 25ua – *about 12 weeks retention of program, data and configuration.*

Viewing 12ma – *40 or 50 hours of editing.(up to 25% less via RS232)*

Burning 180ma – *2 or 3 hours programming - the actual number of PROMS varies widely from about 100 oldest-type to 1000 latest-type*

Emulating 70ma – *8 or 12 hours emulating (depends on amount of access that the target system makes and the load it places on the data lines).*

Real work is a combination of activities, of course. Early warning of battery discharge and automatic shut-off are incorporated, and operate if the battery falls below 8.4 volts. Shutting-off is orderly—there is no harm to any device being programmed and no loss of data, provided S3 is recharged within a few days.

Battery Charger.

The charger will replenish the battery in about 14 hours. For emergency use there is also a boost charge mode which will put working-capacity into the battery in a quarter-hour, and fully charge it in three hours. Boost charging is thus perfectly safe and S3 may be used normally whilst being charged by either method. The charge input circuit is designed to work with almost any kind of supply over a wide range of voltages, both ac and dc, so that the engineer has a good chance of recharging on-site.

The normal charging circuit is working all the time the charger is connected, supplying a constant-current of 50 ma. When S3 is in STANDBY mode, nearly all of this 50ma flows into the battery, which will reach a full state of charge in 14 hours. If S3 is in use, the battery will receive 50ma less the current which S3 is using. For example, when editing S3 uses about 12ma so the battery gets the remaining 38ma. When programming or emulating, S3 might use more than 50ma, but the battery drain is reduced by this amount if the charger is connected. Current flows either into or out-of the battery depending on S3's requirement. If the charger is connected then the current requirement for all activities is reduced by 50ma.

Boost Charge Current

If S3 is in BOOST mode then an additional 180ma constant current flows making 230ma. If S3 is in use, the BOOST current is switched off when a command is being actioned which

prevents temperature monitoring. *If the battery voltage is below about 9 volts, the boost mode turns on automatically: this ensures that high current activities i.e. programming and emulating can be performed continuously when the charger is attached.*

The internal circuitry prevents the batteries overcharging. While capacity remains they are capable of absorbing high currents, but when fully charged the current is not stored— it is dissipated as heat. S3 monitors battery temperature, looking for a 5°C rise at which point the high current is turned off. S3 remains on when BOOST charging - but will turns itself off automatically when the job is done. You can safely leave S3 alone when BOOST charging or continue to use it.

If the message HEAT appears when BOOST is requested, it means that the temperature is too high or too low to be monitored. (below 5°C or above 45°C). The method used for detecting temperature is to measure the voltage drop across 3 silicon diodes, which are in physical contact with the battery, which does not make an accurate thermometer - but it is good enough to observe a 5° rise. S3 does not permit fast charging of batteries which are outside the 5° and 45° limits, because battery manufacturers do not recommend it.

Starting a Boost Charge.

When entering BOOST mode, S3 turns on the high current, then monitors the battery-voltage for a second, looking for a rise. Normally fast charging will produce a voltage rise. If a rise is not seen the message NO CHARGER? is

displayed, and you are given the opportunity to ESCape or ENTER—and plug in the charger, if it was forgotten. A battery which has recently been on-charge may give the same message, because its voltage has risen already, but that does not guarantee that it is fully-charged, therefore you are permitted to override the warning message and charge anyway.

One BOOST CHARGE is enough. It is possible to raise the battery temperature by BOOST charging three or four times in succession, which raises 5 degrees each time. **This does not put extra charge in the battery or achieve anything at all.**

Charging from a Bench Supply.

You can also charge your S3 from a bench supply or from any **A.C.** or **D.C.** supply which meets voltage and current output conditions. The regulators are self-limiting and it is unlikely that any harm will result. Too high an input voltage, however, will produce a large voltage drop across the regulators which, in turn, will raise their temperature to 150°C. At this point the regulator will go into thermal-limiting-mode and cut the charge current down, which defeats the object. Using a bench supply it is best to start at 17 volts and reduce voltage until the current falls slightly, showing that the voltage across the regulator is at a minimum.

A.C. is rectified and D.C. is repolarised by a bridge in the charging circuit. It does not matter if positive and negative are the wrong way round. (but you must avoid ground conflict if charging and emulating at the same time, if the circuit and the charger are both grounded.)

Inputting A.C. reduces the charge to 40% of the D.C. level, because the 100hz rectified pulses are only at a higher voltage than the battery for 40% of the time. Most manufacturers say that pulse charging like this is better for the battery. A.C. charging reduces the battery charge current to 20ma trickle and 100ma boost. NICAD manufacturers also claim that "deep cycling", meaning full charges and discharges, is good for the battery; apparently this discourages physical change, the formation of dendrites, which reduce capacity. It is electronics lore that NICADS "learn" the capacity that their job requires and refuse to give more.

CAUTION When boost charging, power-down does not happen until the boost charge is completed. If you leave S3 boost-charging and then switch off the charger at the mains by mistake, S3 will never see a full-charge; therefore it will not switch-off until the battery falls to the low voltage cut-off point.

Important Charger Modification

Latest versions of S3 (sold after 20th March 88) incorporate automatic fast charging if the battery voltage is less than 9 volts. When the battery is in good condition the voltage rises over 10 volts on slow or boost charge. The reasons for this modification are as follows:

1. Some examples of S3 could get into a mode which failed to recharge a completely flat battery. The reason was that the charge current of 50ma can be dissipated internally at a low voltage level if the system does not get the 5 volts it needs to power-up properly

and sort things out. This can usually be cured by holding down the RESET button for the first few seconds that the charger was plugged-in. Nevertheless it caused some telephone calls to our service department. *If you run the battery completely flat, so that nothing appears in the display, press the reset button when you plug in the charger.*

2. Emulation and Programming switch-off the boost-charge because S3 cannot monitor battery temperature. S3 would power-down when the battery voltage dropped, but this is now prevented if the charger is attached, because the low-voltage boost takes over.

How to Detect a Defective Battery.

The symptoms of a defective battery is that it does not reach normal output voltage and constantly charges at the boost rate. In these circumstances the battery must dissipate the 2 watts delivered by the charger, which produces a noticeable 15 degree rise in temperature whilst the voltage remains below 9 volts. The combination of high temperature and low voltage is obvious when SYSTEM CHECK is used, and the battery should be replaced as soon as possible.

Memory Emulation

S3 emulates ROM and RAM, and may be used to modify or develop code. The emulation-lead (supplied) should be plugged into the target system ROM or RAM socket before issuing the emulate command. Pin-out configuration of EPROMS is automatic, Chip Enable and Output Enable signals are implemented correctly, both for 25 series and 27 series EPROMS. This technique is an enhancement of ROM-emulation—we call it Memory-Emulation.

Benefits of memory emulation .

1. It is universal. You can use it with any microprocessor.

2. The equipment costs less than a Microprocessor Develoment System. The only other piece of kit you need is a computer and a cross-assembler.

3. The target system behaves like the "real thing".

4. The software runs at full-speed.

5. Memory contents can be inspected, edited and the program started again very quickly.

6. It is very cheap and effective with single-chip microcontrollers which have a piggy-back version. You plug it straight into the back of the microprocessor.

7. If you need to follow the code in real-time then you can use another universal tool—a logic analyser.

8. It is good for making modifications to existing systems, such as changing messages and odd values, especially where some trial and error is involved. The "tweak values and try it again" process works very well.

9. When successful the program can be transferred to an EPROM immediately.

How it works

The method used is to assemble the program on a computer and download in one of the standard file-formats into the emulator. The Memory-Emulator will do most things that an MDS can do, and most Microprocessor Development Systems do not have editors and code manipulators which are as powerful as those of S3. S3 will not do anything microprocessor-specific—but it is not usually too difficult to write a routine in the code of the target-microprocessor, whereby a subroutine-call or software-interrupt may be substituted for any instruction, and the microprocessors registers dumped to a specific area of USER RAM where they can be inspected. The microprocessort in the target system can WRITE as well as READ. It it true that PROMS do not have a WRITE input, but that is provided separately on a flying lead.

Microsystem memory selection

In a microsystem, ROM and RAM are selected by decoding the address bus to derive unique CHIP SELECT signals. Each chip has its own slot in addressing space. ROM and RAM output their data on the system bus when their address is selected and the microprocessor's

READ line becomes true: the READ line should be connected to the OUTPUT ENABLE lines of all chips in the system. The microprocessor sees no difference between ROM and RAM, when reading. During a WRITE cycle the READ line stays false and no memory chip can output data on the bus. This prevents conflict with the data written by the microprocessor. The WRITE output of the micro is connected to the WRITE input of the RAM. When the WRITE signal becomes true the write-data output by the microprocessor is written to the location in RAM specified by the address bus. The micro would happily try to write to ROM, if required to do so, but nothing would happen because the ROM has no WRITE input.

Correct Prototype Design

There are many microsystems around which are incorrectly designed - usually without any good reason. A common mistake is to select the chips by their OE inputs or to connect OE to CS—or even to ground. Such systems use more current than they should. There is a period of conflict in every cycle, in which both the micro and a memory chip are enabled on the bus together, and this causes current surges and voltage spikes. Some memory parts require longer access times when recovering from such conflict. Access time is also made worse if the address does not produce CS prior to OE—access time from OE is always shorter. ROM emulation will work in such systems - but RAM emulation cannot work if the system does not apply CS without OE—nor would it work with a real RAM. It is the User's responsibility to ensure that Chip

Enable is TRUE, and Output Enable FALSE when writing.

Emulating RAM.

Byte-Wide RAMS may also be emulated. If the EMULead is to be plugged into the socket meant for a STATIC RAM, the CONFIGURE routine must be set to a PROM which has similar address and data pin-out. The write-line must not be connected through the EMU-Lead, but must go directly to the Flying Write Lead. One way to do this might be to cut the Write-pin off a spare EMULead, used only for this purpose. Another way is to use an intermediate socket plugged into the EMULead, which has the write-pin removed.

Power Usage when Emulating.

The time-out termination is disabled while emulating, but the low-battery termination still works. Emulation places an extra current-drain on the battery: the actual current depending on the load driven by the data-pins and the amount of time the internal memory is enabled. 70 milliamps is typical which is about 7hrs on a full charge. If the charger is plugged-in then there is no time-limit. The ESCape key will terminate the emulate mode.

Line Terminations of the EMULead

All the DATA, ADDRESS and CONTROL lines are connected to logic inputs or outputs of 74HC series gates via series 150Ω resistors within S3: this minimises over- and under-shoot without compromising output drive. To prevent the lines floating when disconnected,

which would cause unnecessary power dissipation, all lines have 100KΩ resistors pulling then to the 5 volt or GND rails. To minimize current-drain when emulating, it is preferable that inputs should be driven right to the 5v and 0v rails. Drive current taken from the data-outputs should be kept to a minimum.

S3 is impedance-matched to normal drive conditions, e.g. direct connections to logic gates. Emulation may not work if driven with high-impedance lines - via series resistors, for example.

NOTE - Emulation Addressing.

The pins of the EMULead are routed to the RAM memory through buffers. When address 0000 and the Chip-Select and Output Enable signals are applied, the data on address 0000 of the RAM appears on the databus. PAGED and SECTORED addressing, other than PAGE 0, SECTOR 0, is not supported.

Changing System Variables

This section might have been called "Fiddlers' Corner". It is for those who are familiar with S3 and would like to make alterations to some features, such as "I wish it would power-down after five minutes, rather than a half-hour." or "I wish keys would repeat quicker" or "I would like to change the noises it makes". Many system variables can be changed, and a personal HELP ROM can be made with the preferred values.

If you press the ESCape key from the command prompt, you will see that the prompt changes to a * instead of > When the prompt is an asterisk you have access to a version of the edit routines which can modify the program and variables.

This is a system version of the normal EDIT routine, which permits editing of the HELP RAM area and the System Variables. Any changes that you make are strictly temporary, and will revert to the default values if you press RESET and load the HELP ROM again. You can keep your changes by making a new HELP ROM with the MAKEHELP facility.

WARNING - BE CAREFUL. You have to know what you are doing when you make direct changes to the software. Incorrect alterations to pulse-lengths and voltages could destroy PROMS instead of programming them. If you want to cancel any changes you have made, press RESET and reload the HELP ROM.

S3 VARIABLES

ADDR	DEFAULT	
Derived from PULSE table		
FF82	75	– *Program voltage applied to ZIF when programming. First 2 bits give pin number. Remainder gives voltage in steps of 0.4 volts. 75 = 53*0.4 = 21.2v*
FF83	1A	– *Vcc supply voltage applied to ZIF in steps of 5/64 volts + constant of 3v. 1A = 26*5/64 + 3 = 5.03v*
FF84	27	– *Vcc supply voltage applied to ZIF during burn in steps of 5/64 volts + 3v 27 = 39*5/64 + 3 = 6.05v*
FF85	0F	– *Maximum number of program pulses before aborting 0F = 15 pulses 00 means not interactive*
FF86	50	– *Length of programming pulse time = 38 + (12*pp)uS*

		*50=80*12 + 38 =998uS*
FF87	04	– *Multiply factor for over-pulse*
FF88	00	– *Addition factor for over-pulse*

Derived from OR/AND table

FF89	E2	– *OR for idle state*
FF8A	FA	– *AND for idle state*
FF8B	A2	– *OR for programming state*
FF8C	BA	– *AND for programming state*
FF8D	E0	– *OR for pre-verify state*
FF8E	F8	– *AND for pre-verify state*
FF8F	E0	– *OR byte for read state*
FF90	F8	– *AND byte for read state*
FF91	0C	– *Written to EMULATCH to emulate current device*
FFA0	70	– *AND for verify state*
FFA1	E4	– *OR for verify state*

System defaults

FF92	00	– *File format 0=Intel, 1=Motorola*
FF93	05	– *Baud rate 0=300, 1=600, 2=1200, 3=2400, 4=4800, 5=9600*

FF94	**0D**	– *Start bit-time of RS232*
FF95	**09**	– *Half bit-time of RS232*
FF96	**2A**	– *Battery-low power-off volts v* 0.2v) 2A = 42*0.2 = 8.4v*
FF98	**1E**	– *Number of minutes before power down, 00 = 1 minute, 1E = 32 min*
FF9D	**08**	– *Sets key-down to key-repeat time*
FF9E	**03**	– *Sets key-repeat frequency*
FFA2	**2C**	– *Audible High tone*
FFA3	**32**	– *Audible Low tone*
FFA4	**03**	– *BIT 0 isCTS flag—if 0 CTS is never checked BIT 1 is audible transmission flag - see below*

Silent Serial Transmissions

S3 makes noises when uploading and downloading files. At first, this is quite useful because it indicates that a transmission is actually taking place. The tones also give some feedback about conditions and quality. After a while, some users will prefer to turn these noises off. Here's how:

Set bit 1 of location FFA4 to 0. (SEND)

Change location E002 from 5F to 7F. (RECEIVE)

Programming Algorithms

Old EPROMS have single pulse algorithms, such as one 50ms pulse per location, which suits products from many different manufacturers. These are identified by pulse length e.g. 2716-50ms.

The next generation of EPROMS use a FAST programming algorithm. Vcc is raised to 6 volts and each location hit with a succession of 1ms pulses until it is seen to program. Then it is overprogrammed with a pulse three or four times the sum of the pulses already applied. Most manufacturers specify 15 tries per location - but in fact all locations seem to program in one or two tries. This algorithm is described by the general name FAST, e.g 2764 FAST.

Algorithms for the latest generation of EPROMS tend to be unique to the manufacturer. These are identified by manufacturer's name, e.g. 2716B NS means National Semiconductor's interactive programming algorithm, which uses 0.5ms pulses with Vpp equal to 13 volts. NOTE that older EPROMS from NS do not use the NS algorithm: the NMC27C16 uses the 2716-50ms algorithm, the NMC27C16H uses the 2716-10ms algorithm. Only the NMC27C16B uses interactive programming and the 2716B NS algorithm.

The algorithm is displayed according to the principles set out above, and the high program voltage and high-voltage pin is given too. PROMS will be damaged by high-voltages applied to the wrong pins.

How to Design New Algorithms

A set of tables is enclosed which relate to the latest version of the HELP ROM, as supplied with your S3, and the notes which follow should be studied in conjuction with those tables. It must be assumed that the designer of algorithms has some knowledge of PROM programming, especially on subjects such as **Interactive Programming, Silicon Signature, Data Polling etc. which are covered in manufacturers' literature.**

All the tables contain the ADDRESS and the actual OBJECT CODE in hex, followed by the data from which the code is derived. Study the OBJECT CODE for existing PROMS, until you can follow the derivation from the data, before attempting to modify or create algorithms.

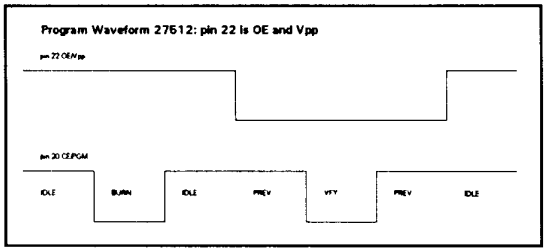
PROM Addressing

All DATA lines and ADDRESS lines 0 - 10 are applied to the same pins for all PROMS. A11 to A15, OE, CE, PGM etc. are routed in various ways.

BIT	LINE	PIN	FUNCTIONS
7	A15	1	A15, Vpp
6	A14	27	A14, PGM
5	A13	26	Vcc on when 0
4	A12	2	A12, CE2
3	A11	23	A11, A12, Vpp
2	A10	20	A11, CE, PGM
1	A9	22	OE, Vpp, PGM
0	A8	22	Vpp on when 1

To accommodate these differences S3 manipulates the high byte of the address bus. The address is first OR'd then AND'd with a byte from a table of OR/AND bytes. The actual address line can only pass if OR'd with 0 and AND'd with 1, otherwise it is forced high or low. There are OR/AND bytes for each of the states applied to the socket: IDLE, BURN, PRE-VERIFY, VERIFY and READ.

Programming states



The significance of IDLE, BURN, PRE-VERIFY and VERIFY states should be made clear by the diagram. The READ state is the one normally used for accessing the PROM in the LOAD, COMP etc. routines: for most PROMS it is the same as the VERIFY state in the burn routine.

Instructions

Copy your HELP ROM into USER RAM, SECTOR 7, using MAKEHELP. Find the tables, using EDIT. Work out whether your new PROM can use existing numbered entries in the PULSE and OR/AND table - if not, edit the UNUSED ones. The simplest method of making

a new ALGORITHM is to convert one of the spare USER entries.

When you are satisfied with your alterations, make a new HELP ROM in a 2764 by programming it with SECTOR 7.

Load the new HELP ROM by pressing the HELP key. Make sure that the CONFIG routine still works and contains your new PROM. Check your new algorithm, preferably using an oscilloscope and DVM before trying it on a real PROM.

The Algorithm Table

ALG is the Algorithm Number as shown by the CONFIG routine, which also shows **MESSAGE**.

ADDR is the actual location in the **HELP RAM** which contains the **OBJECT CODE**.

OBJECT CODE is the actual data to be found at **ADDR** in the HELP RAM, being a translated version of the parameters below.

FLAGS are coded instructions to the program: The higher nybble tells the program to call subroutines to manipulate addresses and selects in a special way:
1 = 2532,
2 = 2564,

25 series PROMS require A11 on pin 20 and A12 on pin 23 of the socket.
3 = 8764,
4 = 87256

87 series PROMS require that $\overline{CE}$ performs an Address Latch Enable function, because the address bus is latched into the PROM by a pulse on this line.
8= Paged PROMS like 27513

PAGE mode PROMS, such as the 27513, made up of pages which contain 16K bytes, require that the page is written into the PROM on the data bus.

The lower nybble gives the number of byte-pairs of Device Identifiers in the SIGN field.

MESSAGE is the Algorithm Name as displayed by S3 in the config routine: the object code is in ASCII. It has 10 chars max including a

hyphen, only 7 of which can come before the hyphen. The message is truncated before the hyphen for EMUL, CHECKROM and DUMPROM because there is insufficient space in the command line. The ^ character means that the MSB of the previous character is set, telling the program that it is the last byte of MESSAGE.

SIZE is the size of the PROM in Binary Coded Decimal e.g. a 27256 uses 32.

PULSE is a hex pointer to the PULSE TABLE.

OR/AND is a hex pointer to the OR/AND TABLE.

SIGN (signature) consists of the byte pairs, in BCD, of Device Identifiers which are masked into the PROM. The number of pairs is specified by the second character of FLAG

A number of entries are available for modification by the user. The table could be shifted or extended in memory if there is free space. All fields must be represented. The size of each table-entry depends on the number of SIGN pairs, set by the second character of FLAGS, and the length of MESSAGE.

The Pulse Table

EEPROMS and EPROMS are mixed in the same pulse table. However, the headings are different for EEPROMS so the table is separated to make it easier to follow.

EPROM Pulse Table.

NO is the pointer used in the ALGORITHM TABLE to choose the entry.

ADDR is the place in the HELP RAM where the entry will be found.

OBJECT CODE is the actual data in HELP RAM.

Vpp is the Program Voltage expressed in tenths of a volt, shown in decimal but coded in HEX.

PIN is the pin number of the CHIP (not the ZIF) to which Vpp is applied. Coded in BCD.

Vcc, Vcp are the supply voltage when Reading and Programming. The voltage is calculated in steps of 5/64 volts from a base of 3 volts. The value is one point larger for 24 pin PROMS to compensate for the 0.1 volt drop across the switching transistor.

SYMBOL	DECIMAL	- *HEX*
5v	26	- *1A*
6v	39	- *27*
6.25v	42	- *2A*

HITS is the number of pulses applied in interactive algorithms before aborting, shown in DECIMAL but coded in HEX.

LENGTH is the program pulse in microseconds, shown and coded as two bytes in BCD.

MP and AP are used to calculate the over-programming pulse. The pulse is MP (Multiplied Pulse) times the sum of previous pulses, or AP (Additional Pulse) times a single pulse depending on which has a non-zero value.

EEPROM Pulse Table

Vpp is the voltage used for programming or chip-erasure, in tenth-volts. e.g. D2H = 210 = 21v.

PIN is the CHIP pin number held at Vpp when programming. 00 means high-volts used for ERASE, but not programming.

FLAGS are used to pass instructions to the program:
BIT 7 SET if chip-erase required
BIT 4 SET if byte-pre-erase required
BIT 3 SET for chip erase on ZIF pin 23
BIT 2 SET for chip erase on ZIF pin 22
BIT 1 SET for chip erase on ZIF pin 1
BIT 0 SET for chip erase on ZIF pin 24

PULSE is two BCD bytes giving pulse-length in multiples of 100uS, if 0000 then Data Polling is active.

PAGE is page size in HEX - some EEPROMS write data in pages of 16, 32 or 64 bytes, which speeds programming.

NU is a byte which is not used for EEPROMS.

EE is set at 80H to show the device is an EEPROM.

The OR/AND Table

NO is the pointer used in the ALGORITHM TABLE to choose the entry.

EMUL is the byte written to the EMULatch which configures pins when emulating. The EMbyte is written to an eight bit latch. It routes the pins of the EMULead to the ADDRESS and CONTROL lines which are connected to the USER RAM. The EMbyte causes A11 and A12 to be exchanged for 25 series PROMS and implements the extra Chip Select on pin 27 for a 2564 - and so forth.

Bit	Name	– *Remarks*
7	'27512'	– *Connects E1 to A15. If 0, A15 = 0. 1 for 27512 else 0.*
6	'> = 27256'	*Connects E27 to A14. If 0, A14 = 0. 1 for 27512, 27256 else 0.*
5	'> = 27128'	*Connects E26 to A13. If 0, A13 = 0. 1 for 27512, 27256, 27128 else 0.*
4	'2564'	– *Connects E23 to A12. 1 for 2564 only, else 0.*
3	'NOT 25'	– *0 FOR 2516, 2716, 2532, 2564. 1 if A11 on E23.*
2	'EM 25'	– *0 for 2532, 2564 chips with A11 on E20.*
1	'EM 27'	– *0 for 27 series incl 2716 & 2516 with $\overline{CE}$ on E20.*

0 'UNUSED' *This output of the latch is not connected .*

IDLE, BURN, PRE-VFY, READ and VERIFY bring about the different states which are applied to the program socket. The values are successively OR'd, then AND'd with the High Byte of the address to be applied.

VERIFY bytes have different purpose for EEPROM: they are written to ZIF latch to chip-erase.

The VERIFY state has no special meaning for EEPROMS so the VERIFY bytes are used to define the PRE-ERASE and ERASE states. Those EEPROMS which have chip-erase, must set-up the PRE-ERASE state before the high voltage is applied, then the ERASE state is applied.

The OR/AND Chart.

This chart shows how the OR and AND bytes are derived. It is in fact taken from the spreadsheet which calculates the HEX values from the binary states prescribed.

The ADDRESS is applied to the program socket by means of a 16 bit latch.

The LOW BYTE of the latch, A0 to A7, is always applied to the same pins.

The HIGH BYTE of the latch has address lines, which pass through, marked X, other lines must be forced HIGH or LOW during the program cycle. Two logic functions are perfomed on the address: an OR, then an AND. The address only passes if OR'd with 0, then AND'd with 1. All other OR and AND conditions force the pin to 1 or 0.

Bit 5 applies supply voltage to pin 26 of the socket - the supply of a 24 pin chip. It should be 0 for 24 pin chips, 1 for 28 pin chips.

BIT 0 applies the Vpp (high voltage) to pin 22. If BIT 0 is 1, then BIT 1, which is the signal to PIN 22, should be 1 too.

VERIFY has no meaning for EEPROMS, so these bytes are used to define the correct state of the pins for CHIP-ERASE. The sequence is that the PRE-ERASE state is applied, then the high voltage, then the ERASE state is applied.

Using the Modules

Installing a Module.

Install the new software by placing the HELP ROM which came with the module in S3's ZIF socket then pressing HELP followed by ENTER.

Insert the module's 28 pin plug into S3's ZIF socket with the box to the left of S3.

Insert devices into the ZIF socket with pin 1 at the top.

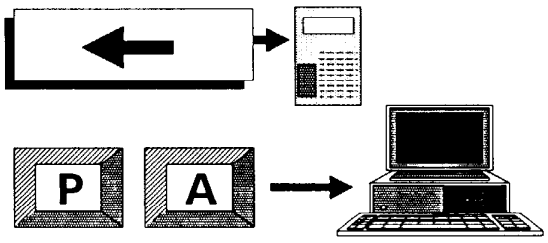
The most noticeable difference between the module's software and the original software is the range of PROMS or devices covered, as shown by CONFIG. Some, if not all, of the original PROMS are left out, to make room for the new devices. Any which are retained can be read or programmed without the module. **Do not put 28 pin or 24 pin PROMS, which still appear in the CONFIG list, into a 32 pin or 40 pin module — remove the module and put them directly into S3.**

Apart from the choice of devices in the CONFIG routine, the workings of S3 remain much the same. Any differences are documented in the specific module's literature.

Removing a Module.

Restore S3's working program by installing the original HELP ROM. Changing S3's program by loading different HELP ROMS does not affect the contents of the 64K of USER-RAM.

Page Editing




```
>
>
>PAGE=0
```

The word PAGE is used in the S3 manual to refer to a page-mode EPROM like the 27513 which programs in 16K blocks. Each block can be selected by its PAGE number.

When modules are used which program EPROMS larger than 64K, which is the size of S3's USER RAM buffer, PAGE is used to mean the particular 64K block which is being addressed.

A new keyboard function has been added to the '32 PIN' and '40 PIN' software to allow

more convenient PAGE swapping using the ⏎ key. This also works for PAGE mode EPROMS like the 27513 when the 32 pin or 40 pin module HELP ROM is loaded.

Pages from 0 to 7 can be selected—more than is necessary currently but if devices with more pages are released these may be programmed without software upgrades.

32 Pin Module

This module is for programming 32 pin EPROMs. Direct in-circuit emulation of these 32 pin devices is not, of course, possible, because 64Kbytes is the limit of S3's USER RAM BUFFER.

AMD	Am27C010
Fujitsu	MBM27C10001
Hitachi	HN27C101G, HN27C301G
Intel	D27010
National	NMC27C1023
NEC	u P D 2 7 C 1 0 0 0 D , uPD27C1001D,uPD27C2001D
TI	TMS27C010
Toshiba	TC571000D, TC571001D

Keeping Up-to-Date.

New devices will be added to the software as they appear on the market, including two and four megabit parts. Dataman software is continually upgraded; algorithms are added as new devices appear on the market and useful suggestions from our users are incorporated. Upgrades can be obtained for free by down-loading from the Dataman bulletin board or for a nominal charge in a PROM.

Programming in PAGES.

Since these large devices have more locations than S3's 64K bytes of user-RAM they are dealt

with in PAGES. e.g. a one megabit part has two PAGES, page 0 and page 1. See the notes in the manual about PAGE editing.

Programming Methods.

The high speed page programming mode is implemented wherever appropriate. For example, NEC's two megabit 27C2001 will program with less than one minute of burn time. In the unlikely event that the user has modified the BURN start address to a location not divisible by four (the BURN page size), S3 will action a byte-at-a-time burn algorithm. This produces a noticeable increase in programming time.

40 Pin Module

This module is for programming 40 pin EPROM-S. Direct In-circuit emulation of 40 pin devices is not possible, because 64Kbytes is the limit of S3's USER RAM BUFFER.

AMD	Am27C1024
Fujitsu	MBM27C1024
Hitachi	HN27C1024G
Intel	D27210
National	NMC27C1024
NEC	uPD27C1024D
Toshiba	TC571024D

New devices will be added to the software as they appear on the market .

How S3 handles 16 bit words

Two of S3's USER-RAM BYTES are used to program each 16 bit EPROM WORD. In this context a BYTE means 8 bits and a WORD 16 bits. Even-addressed BYTES are burnt into the EPROM's higher 8 bits and odd-addressed BYTES into the lower 8 bits to form a 16 bit WORD.

e.g. U-RAM byte at 0010 = EPROM high 8 bits at 0005 and U-RAM byte at 0011 = EPROM low 8 bits at 0005.

These 64K WORD EPROMs require more locations than S3's 64K bytes of User-RAM so they are dealt with in PAGES. Page 0 covers the lower 32K words and PAGE 1 the top 32K.

Only one SECTOR (00) exists in S3'3 USER-RAM, but addresses can be given upper and lower limits. Note that addresses relate to S3 USER-RAM addressing not the EPROM e.g.

```
>LOAD 27210-QP
SECTOR 00x0002,0011
PAGE=0
```

This will load the words from 0001 to 0005 of the EPROM into 0002 to 0011 of S3's user-RAM.

S3 will round sector addresses so that whole EPROM words are always dealt with e.g.

```
>LOAD 27210-QP
SECTOR 00x0001,0004
PAGE=0
```

This will load the words from 0000 to 0002 of the EPROM into 0000 to 0005 of S3's user-RAM.

MCS-48 Series Module

This module is for programming MCS-48 series of micro-controllers.

Intel	D8748H, D8749H, P8748H, P8749H, D8741A, D8742, P8741AH, P8742AH
NEC	8748, 8749

Using the Module

Most of the chips in the MCS-48 series use high voltages for the loading routines (LOAD, COMPARE and PRETEST) as well as the BURN routine. These high voltages vary from chip to chip so be sure to be configured properly when using any of the ZIF-accessing routines.

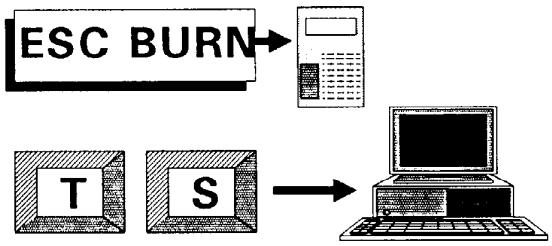
When configuring the device type S3 displays the highest voltage which will be applied to the chip when programming. This is to help you select the correct algorithm.

This high voltage value is used in the warning message when you are about to BURN, LOAD, PRETEST or COMPARE.

The MCS-48 series devices are unusual in that they program from low to high. i.e. If you load an erased chip you will find all 00s rather than the FFs that you are used to seeing in erased PROMs.

Obviously S3 cannot EMULATE these devices, hence this function is disallowed. All other keyboard functions work as expected.

The OTP (One Time Programmable) plastic chips (P8741AH and P8742AH) have a signature mode allowing various features (see Intel's Microprocessor and Peripheral Handbook - Volume 2) including a security mode so you can hide your code from prying eyes.




```
>
>
*SIGNATURE MODE ON
```

S3 will toggle in and out of this mode (TS stands for Toggle Signature mode). When the mode is on, the signature bytes, (there are 32 of them) can be observed or burnt as appropriate in the normal way.

MCS-51 Series Module

This module is for programming the MCS-51 series of micro-controllers.

Intel	8751H, 8751BH, 87C51, 87C51FA, 87C51FB, 8752BH, 87C252
AMD	8751H, 8753H, Am9761H, 87C51, 87C521, 87C541,
Mullard	SC87C51B,

Using the Module

When configuring the device type S3 displays the highest voltage which will be applied to the chip when programming. This is to help you select the correct algorithm.

This high voltage value is also displayed in a warning message when you are about to perform an operation applying high voltages to the device. These operations are BURN, BURN ENCRYPTION, and BLOW SECURITY.

Obviously S3 cannot EMULATE these devices, hence this function is disallowed. All other keyboard functions work as expected.

Additional features

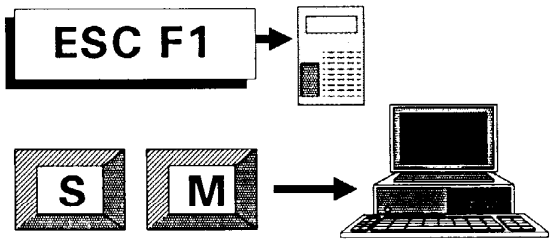
In addition to S3's normal operating modes, extra functions are included to support the enhanced features of some MCS-51 series devices. These functions are listed overleaf.

Not all devices provide these features and S3 operates accordingly (check the appropriate data sheet). If a device has a particular feature, S3 will implement it. If the device does not support that feature S3 gives the message 'NOT AVAILABLE'.

The new features provided by MCS-51 help ROM are summarised below and described in the following sections.

- SECURITY MODE
- BURN SECURITY
- BURN ENCRYPTION
- COMPARE ENCRYPTED
- LOAD ENCRYPTED

MCS51 Security Mode

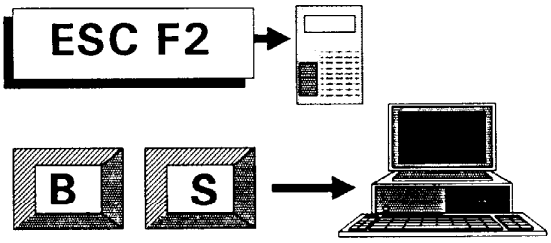



```
>
*SECURITY MODE
LB = 1, LB2 = 0
```

This function allows the desired combination of lock (security) bits to be set. A '1' indicates that the bit is asserted and will be programmed. Note that this feature is only available for devices which have more than one lock bit.

This function does not actually blow the security feature, this must be done using the 'BURN SECURITY' command.

MCS51 Burn Security

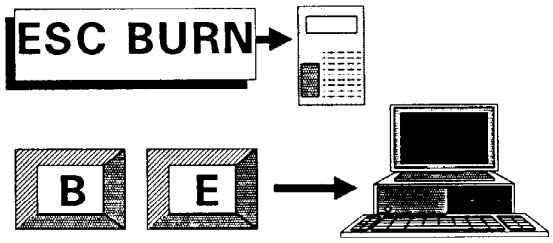



```
*BURN SECURITY
87C51-INT
HIGH VOLTAGE = 12.7V
SECURE
```

This command burns the lock bits activating the desired security feature. Devices with more than one lock bit must first have the desired mode selected using the 'SECURITY MODE' command. Devices having a single lock bit can be secured using this command only, no special mode is required.

Because this process applies high voltages to the device a warning is given. Permanent damage can result if the configuration is for the wrong part.

MCS51 Burn Encryption

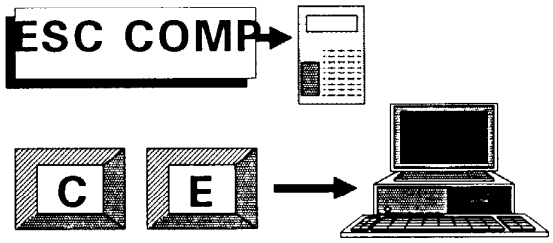



```
*BURN  ENCRYPTI ON
87C51B-MUL
HIGH VOLTAGE=12.7V
ARRAY ADDRESS=1000
SAME
```

This function burns the encryption bytes from the array location given. The default array address is the next location after the normal data block. For example, assuming the Mullard 87C51B above was programmed in sector 00, the last programmable location was 0FFFH. The default start address for the encryption data bytes would then be 1000H as shown.

The encryption array can not be verified directly and successful operation is tested using the 'COMPARE ENCRYPTED' function. Therefore the data in the relevant sector of S3 must be the same as that previously programmed into the device.

MCS51 Comp Encrypted

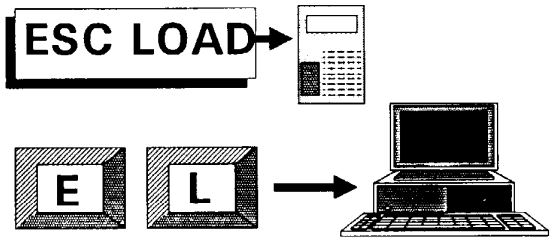



```
*COMPARE  ENCRYPTED
87C51FA-INT
SECTOR 00=0000,1FFF
ARRAY ADDRESS=2000
SAME
```

This function operates in the same way as 'COMPARE' but uses the encryption bytes at the array address to decipher the data from the device.

The encryption array location is selected using the same method as for the 'BURN ENCRYPTION' command.

MCS51 Load Encrypted




```
*LOAD  ENCRYPTED
87C51FB-INT
SECTOR 00=0000,3FFF
ARRAY ADDRESS=4000
```

This command loads data from the device using the encryption array bytes to decipher the data. The array address is selected in the same way as for the other encryption operations.

Note that this command does not read the encryption array, this cannot be done. It assumes that the encryption bytes are already known and loads the original uncoded data. If the encryption array is unknown or incorrect, the loaded data is meaningless.

The EPLD Package

This packages includes

- Two EPLD modules
- Two HELP ROMs

The two modules each have separate software. Install the HELP ROM that corresponds to the module you intend to use.

EPLD Module 1

Install the EPLD1 HELP ROM.

Devices covered by this module;

ALTERA	EP300, EP310, EP320
CYPRESS	PALC16L8, PALC16R4, PALC16R6, PALC16R8, PALC20G10, PALC22V10
GOULD	PEEL18CV8
ICT	PEEL18CV8
INTEL	5C0315C032
MMI/AMD	PALC22V10
TEXAS	TICPAL22V10

EPLD Module 2

Install the EPLD2 HELP ROM.

Devices covered by this module;

ALTERA	EP600, EP610, EP900, EP910
INTEL	5C060, 5C090

Logic Compilers

You will need a logic compiler, they make EPLD design a simple and quick process. Logic compilers allow you to enter your application in the form of a high-level program. This algorithm is compiled and optimised to produce a JEDEC file which can be sent straight to S3 and burnt into an EPLD.

Manufacturers' compilers tend to support only their own chips. Some manufacturers all but give their software away to help promote their chips, others don't.

If you are trying to hold down the cost of your development system use EPLDs supported by free software. Some problems will be more suited to a particular device and big tasks will require more little chips than big ones, but any part will do the job!

If you are new to the idea of EPLD design get the two AMD/MMI PAL handbooks. These contain an excellent introduction to the subject and a manual for PLPL, this is their logic compiler which comes on eight floppies and is available for a modest sum.

UK contact is:

A.M.D., Mail Operations, P.O. Box 4, Westbury-on-Trym, BRISTOL, BS9 3DS.
Tel: 04862 22121

ICT give away their compiler. U.K. distributor is:

SEQUOIA, Unit 5, Bennet Place, READING, Berks Tel: 0734 311822, Fax: 0734 312676

There is another completely different approach to programmable logic design. You lay down schematic logic circuits on your computer screen. This schematic is captured and compiled and then blown into an EPLD. This provides various advantages over traditional design methods (quicker than wire-wrapping, reduced chip count, design security and so on), but it is not as powerful as the algorithmic method.

Intel are freely distributing an 'EPLD Designer Kit' which supports their 5C032 (equivalent to Altera's EP320). This software supports the schematic capture method.

UK contact is:

Intel Corporation (UK) Ltd, Piper's Way, Swindon, Wiltshire England SN3 1RJ
Tel: 0793 696000.

The Fuse Array

You may never need to inspect or edit fuses individually. Usually you work at a high level with a logic compiler and let the compiler do all the hard work.

CMOS EPLDs don't actually have fuses, however the erasable cells are still referred to as fuses.

After loading an EPLD or receiving a JEDEC file S3's FUSE ARRAY is filled with data. The array is stored in USER-RAM from 0000 onwards. Each location will contain 00 or 01 depending on whether the fuse is blown or not.

Fuse values are stored in addresses as specified by the JEDEC map of the EPLD.

JEDEC maps often specify fuse positions in decimal. If you want to inspect a particular fuse using EDIT you will have to convert the decimal value to HEX.

Unwanted routines

Various routines which are not relevant to EPLD programming have been left in. SPLIT and SHUFFLE for instance.

What is the point of this? The code for these routines is contained within the MASK-ROM BIOS; therefore it costs us nothing to leave them in. Somebody, somewhere, for some obscure reason may use SPLIT one day while using the EPLD software! So we left it in.

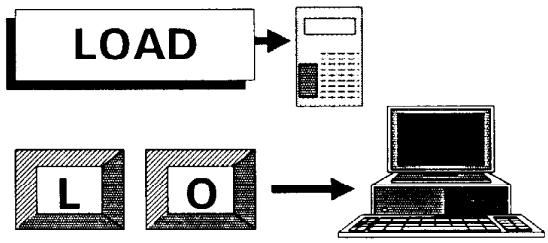
BEWARE - HIGH VOLTAGES

All EPLD accessing routines use high voltages; LOAD, PRE-TEST, BURN, COMPARE, STATUS and BLOW SECURITY. So they are all potentially damaging, not to your equipment, but to your chips.

- Use the correct module as specified on screen when you first install the HELP-ROM or when S3 is turned on.
- Be sure your algorithm matches the EPLD you are using.
- Insert chips with pin 1 towards the top of the ZIF socket. If the device is smaller than the ZIF put it as low in the socket as possible.

All the ZIF-accessing routines require more than one key-press before they will start. Having pressed, say LOAD, you still have the options to proceed by pressing ENTER, or to escape by pressing [ESC].

EPLD Load

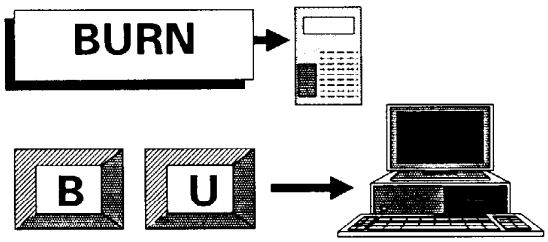



```
>
>LOAD
CYPRESS PALC16R8
```

Press ENTER to proceed or ESC to escape.

Loads the data from an EPLD into S3's fuse array which is stored at the beginning of USER-RAM.

EPLD Burn



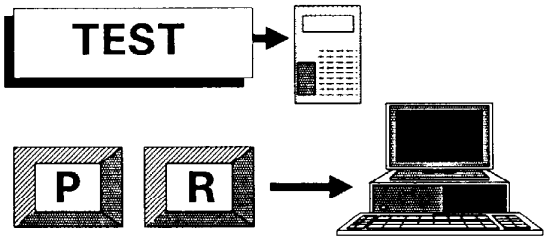

```
>
>BURN
INTEL 5C031
```

Press ENTER to proceed or ESC to escape.

Burns the fuse array data into an EPLD. After burning the device is automatically compared with the fuse array.

If a location will not burn **FAILED** is displayed before comparing.

EPLD Test




```
>PRETEST
MMI PALC22V10
Will burn
```

Press ENTER to proceed or [ESC] to escape.

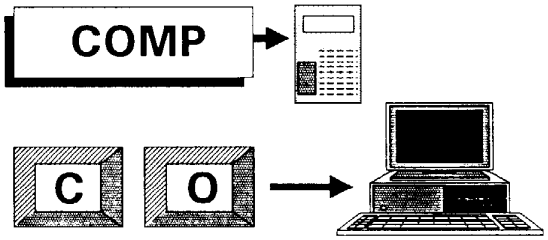
EPLDs can be pretested before burning. It is rare to re-program EPLDs, the pretest is mainly for checking that the device is a blank one.

Like EPROMs, EPLDs can only be blown in one direction: usually the fuses are low after erasure.

If the data will burn then **Will burn** is displayed, if not then **Cannot burn**.

N.B. A secure UV-erasable device cannot be burnt.

EPLD Compare

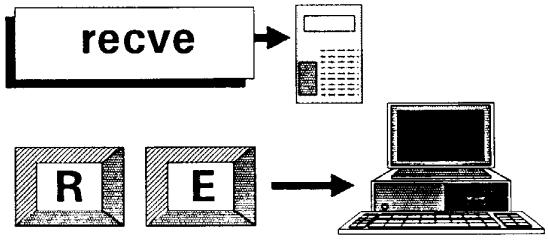



```
COMPARE
ICT PEEL18CV8
0002   RAM=0 PLD=1
0007   RAM=1 PLD=0
```

Press ENTER to proceed or ESC to escape.

Compares the EPLD fuse data with the fuse array which is stored at the beginning of USER-RAM. Any differences are shown: if there are none, **Same** is displayed.

EPLD Receive




```
>
>RECEIVE
ALTERA EP900
```

Receives a JEDEC file via the RS232 port. S3 must be configured to the required EPLD type before the file is received; use CONFIG to set up the device type and the baud rate.

The received file is stored as a fuse array from 0000 onwards in USER-RAM. This array can be inspected and even modified by using EDIT, but usually the data will be burnt straight into a chip.

If the EPLD has TURBO and MISER modes these bits will usually be included in the JEDEC file by the logic compiler. If not, S3 will default TURBO OFF and MISER ON.

RECEIVE expects the JEDEC file to be started with an STX (02) character and ended with an ETX (03). Having received a file enclosed between these two characters, S3 displays **JEDEC RECEIVED** then goes on to check it and fill the USER-RAM fuse array. If the file does not include STX and ETX characters S3 stays in the RECEIVE routine. Press the S3 [ESC] key to escape.

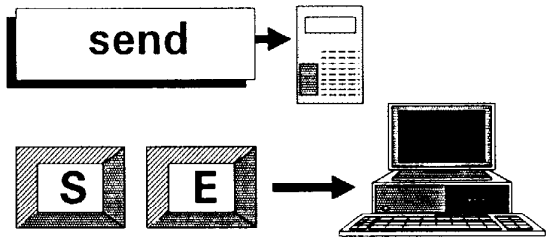
S3's receive routine checks the transmitted *fuse checksum* and *transmission checksum* as specified in the JEDEC standard. If the sums are wrong **FUSE CHECKSUM ERROR** or **XMIT CHECKSUM ERROR** is reported. If a *transmission checksum* of 0000 is received no error report is made.

Even if there are checksum errors the file is processed and the USER-RAM fuse array is set up.

If you are having reception problems install S3's original HELP-ROM. Set it up to receive binary files and send your JEDEC file. Now SEEK for STX and ETX. Are they being sent?

Some people object to S3's reception noises. This can be prevented by editing E002 from 5F to 7F. Edit HELP-RAM directly or load up your HELP-ROM, modify the location and BURN yourself a 'quiet' HELP-ROM.

EPLD Send




```
>
>SEND
ALTERA EP600
```

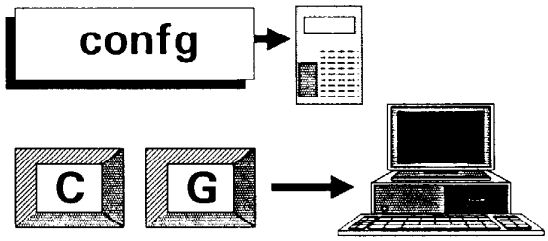
The fuse array in USER-RAM is converted to a JEDEC file and sent down the RS232 line. S3 must be configured to the required EPLD type before the file is sent; use CONFIG to set up the device type and the baud rate.

Fuse checksums and transmission checksums are calculated and sent, as specified in the JEDEC standard.

Transmissions are terminated with a 1A character. Your PC will recognise this as the end of the file.

To escape SEND, hold down the S3 ESC key for a while. S3 checks for the escape key at the end of each JEDEC row.

EPLD Configure



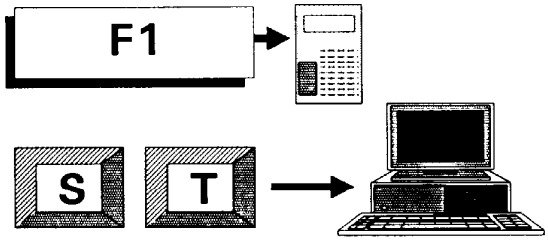

```
>CONFIG
MANUF: MMI
DEVICE: PALC22V10
```

For setting up EPLD algorithm and RS232 BAUD rate. First select the manufacturer, they are listed in alphabetical order. Use ← and → to scroll through (or BACKSPACE/SPACE if using a terminal). When the required manufacturer is displayed press ENTER.

Now select the device in the same way. If you wish to set up the RS232 baud-rate press ENTER, if not press ESC.

RS232 baud-rate is set in the same way as for an ordinary EPROM.

F1 - EPLD Status




```
>STATUS
TEXAS TICPAL22V10
Fuse checksum = 6A60
```

Press ENTER to proceed or ESC to escape.

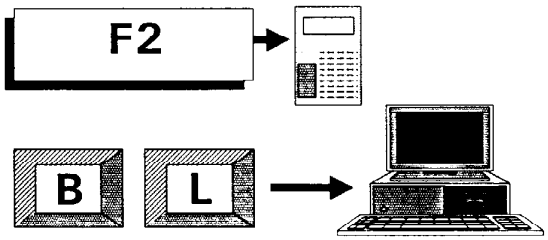
Use STATUS if you find a 'stray' chip and need to identify it. This function inspects the fuses of the EPLD in the ZIF socket; the USER-RAM fuse array is unchanged. The displayed fuse checksum value is calculated as specified by the JEDEC standard for serial communication.

It is worth knowing the fuse checksums of your production masters. If you STATUS check the master and its checksum is unchanged you can be all but certain that it is valid. Better than sending out hundreds of faulty products!

If the EPLD is secure its fuses cannot be seen so the fuse checksum cannot be calculated but **Secure** is displayed.

The security bits of the Altera and Intel EPLDs cannot be checked except when they are being blown for the first time since the chip was erased. Try to STATUS check a secure one and you will get a meaningless fuse checksum report. S3 cannot know whether the device is secure or not so it will calculate and report a sum regardless.

F2 - EPLD Blow Security




```
>BLOW SECURITY
ICT PEEL18CV8
Secure
```

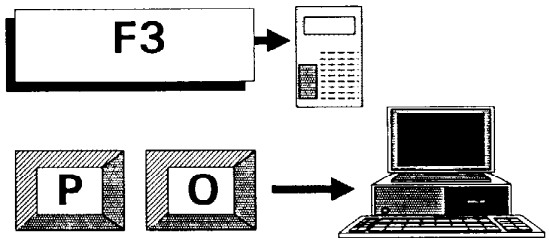
Press ENTER to proceed or ESC to escape.

Blows the EPLD security bits. This makes it impossible for your design to be inspected - using S3 or anything else. After blowing the fuses the STATUS is automatically checked, so you can be sure that the chip really is secure.

The security bits of the Altera and Intel EPLDs can not be checked except when they are being blown for the first time since the chip was erased. These devices don't run STATUS after BLOW SECURITY. Security bits are checked directly after they are blown; the

message is **Secure.** In the unusual circumstance of the security bits not blowing, **Failed** would be reported.

F3 - EPLD Power Modes




```
>POWER MODES
ALTERA EP320
Turbo:OFF  Miser:ON
```

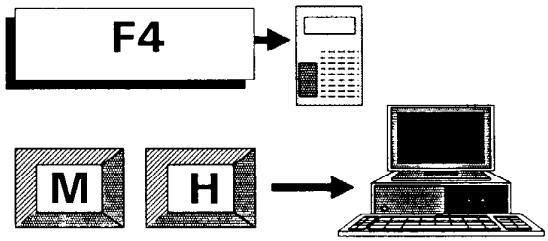
This function allows the TURBO and MISER bits in the USER-RAM fuse array to be toggled. These bits could be modified by directly editing the fuse array; the POWER MODES function is included for convenience.

Use ⊟ (or SPACEBAR of a terminal) to toggle the mode. Press ENTER to accept the entry. Press ESC if both modes are as required.

S3 must be configured to the required EPLD, so that it knows whether the chip has any power modes and if so, where the power bits are in the fuse array. Most EPLDs don't have these modes. In these cases **None** will be displayed. Some have only a TURBO mode, then S3 will leave the **Miser** message out.

Turning both TURBO and MISER modes on together is forbidden in the manufacturers data sheets. S3 does not prevent this but displays this message as you leave the POWER MODES function:
WARNING: both modes on, not recommended

F4 - EPLD Make Help




```
>
>
>MAKE HELP
```

This works in the same in the same way as it does in the original software.

"But how do you program a 2764 with an EPLD programmer?", you ask. There is no problem. Run MAKE HELP. Now install the original EPROM-programming HELP-ROM; the HELP-RAM data is not corrupted by this process. Configure S3 for your 2764 and burn your own customised EPLD HELP-ROM.

Software Upgrades

Dataman places the most current versions of S3 software on its bulletin board. These can be freely down-loaded and distributed on:

(0300) 21095 300/1200/2400/4800/9600 N,8,1.

(HST and V.42bis supported.)

The changes that require more explanation are included below.

BURN visual feedback

While burning, the address reached is displayed. Each time a 256 byte page boundary is crossed the address is updated.

```
>BURN 2764-INT
12.5V TO PIN 1
SECTOR 07=E000,FFFF
ADDR=E300
```

Sector confusion cleared up

The confusion of some S3 users led us to make a slight modification. CONFIG used to leave the sector START and END addresses unchanged, even if a PROM of a different size was specified. Some people did not realise this and would mistakenly only program part of their PROM. *SECTORS are explained on pages 10 and 11.*

The upgraded CONFIG routine always sets the START and END addresses to the limits of the sector. So, after CONFIG the whole PROM is always mapped out.

Silent Transmissions

When sending or receiving S3 produces audible feedback. This is very useful when you are trying to establish communications with another machine.

However some people would prefer either *send* or *receive* to be quiet.

The send noise can be turned off by setting bit 1 of location FFA4 to 0. The receive noise can be turned off by editing HELP-RAM location E002 from 5F to 7F. These changes will work for all S3's module software as well.

You may want to make a HELP-ROM which causes S3 to default to silent transmissions. Do this by editing HELP-RAM and then using the MAKE HELP function. Or you could load up the current HELP-ROM and edit the location in USER-RAM and then burn this into another 2764.

NICAD care

It's not a good idea to keep nicads permanently topped up. They will eventually lose their ability to hold charge. Every 40 or so charge/discharge cycles its worth allowing S3's battery to be discharged. So, on these occasions, keep using it until the message **ABORT - BATTERY LOW** appears, before recharging.

FLASH ROM Programming

27-series and 28-series FLASH ROMs can be programmed by S3. New software is required; this is supplied in a HELP-ROM.

27-series	27F64, 27F256
28-series	28F256, 28F512, 28F010

The 27-series will program directly in S3's ZIF socket. The larger 28-series devices require S3's 32-pin module.

S3's programming procedure

```
>BURN 28F010
12.0V TO PIN 1
SECTOR 00=0000,FFFF
Clearing ADDR=FF00
Erasing ADDR=FF00
Burning ADDR=1200
```

- Checks to see if device is empty, if it is, jumps straight to the burn routine.
- Before erasing 'clears' all locations - programs them all to zero. **Clearing** is displayed.
- Erases the chip; **Erasing** displayed.
- Check that device is completely erased.
- Burn the PROM; **Burning** is displayed.

Note: If you QUICK-BURN a FLASH ROM the erase section is skipped. The same applies if you burn just part of a device by altering the sector addresses.

```
>BURN 27F64
12.8V TO PIN 1
SECTOR 00x1600,1891
```

Texas TMS27C292

A HELP ROM is available that allows Texas Instrument's TMS27C292 to be programmed.

The software in this HELP ROM includes all the EPROM algorithms. The EEPROM algorithms have been left out to make space for the software required to support the 27C292.

Programming a 27C292

- Install the 27C292 HELP ROM.
- Configure S3 to the 27C292 algorithm. This is at the end of the algorithm list, number 30.
- Put the PROM in the bottom of S3's ZIF socket with Pin 1 towards the top.

Note: Erased 27C292s don't look blank. If you load them you will see data, not FFs. So PRE-TEST does not work on these chips.

Intel's 2816

This EEPROM requires a special module and HELP ROM to program it.

Programming the 2816

- Plug the module into S3's ZIF socket.
- Install the HELP ROM

```
DATAMAN S3 HELP 2.3i
>
>
>
```

- Configure to algorithm 03

```
CONFIG
03 2816     21.0V
>
>
```

- If the chip needs erasing then do this by burning all its locations with FFs.
- Now load the required data into USER RAM and burn it into the device.

Index

A

Algorithm 30
Algorithm Table 92 - 93
Algorithms, Designing 89 - 91
ASCII File Format 71

B

Battery
- Charging 37, 73 - 78
- Charging from a bench supply 76
- Completely flat 78
- Monitoring Volts/Temp 55
- Time between charges 73

Baud Rate 61 - 63
- Discussion 60
- Selection 33

Beeper 12
BINARY File Format 72
BIOS MODE 4, 57 - 58
Blocks
- Checksum of 41
- Exchanging 35
- Explanation 9
- Filling with specific byte 36
- Moving 34
- Overlapping 35

BOOST charge 37 - 38
BURN 15 - 16

C

Charging Battery 37, 73
CHECKSUM 41 - 42
    without loading 52
CHEKSUM ROM 52
Circuit Block Diagram 56
Command Line 9
Communications Software 66
COMPARE 24 - 25
Computer Operation 7
CONFIGURE 30 - 33
Current consumption 73

D

Data Polling 89
Diagram, Block Circuit 56
Display
    ASCII characters 45
DUMP
    Dumping RAM to terminal 46
    Dumping ROM to S3 49

E

EDIT 46
    Editing the TPA 47 - 48, 84
    Remote editing 45

Stand-Alone editing 44
EMULATE Proms 39 - 40
Emulating RAM 82
EMULead 39
EMULead line termination 82
EPLD
Burn 122
Compare 124
Configure 128
F1 - Status 129 - 130
F2 - Blow Security 131 - 132
F3 - Power Modes 133 - 134
F4 - Make Help 135
Load 121
Receive 125 - 126
Send 127
Test 123
EPLD Package
Fuse Array 118
High Voltages 119
Logic Compilers 117
Module 1 116
Module 2 116
ESC edit system RAM 47 - 48
ESC message 8

F

Fiddler's Corner 84
File Formats 67
ASCII 71
Binary 72
Configuring 32

Discussion 64
Intel 68
Motorola 69
TekHex 70
Files
Sending and receiving 63
FILL 36
Flash ROM Programming 138 - 139
FUNCTION KEYS
F1 = DUMP ROM 49
F2 = QUICKBURN 50 - 51
F3 = CHECKSUM ROM 52
F4 = MAKE HELP 53 - 54
F5 = SYSTEM CHECK 55
Fuse Array - EPLDs 118

G

Glossary of terms 8 - 12

H

HELP 20 - 21
HELP ROM 58
Copying to TPA 6
Making your own custom 53

I

INTEL HEX File Format 68
Intelligent Identifier 31
Interfacing with a computer 64 - 67

K

KEY
- ENTER 9
- ESC 9
- FUNCTION 12
- RESET 57

Keypad repeating 8

L

LCD
- Liquid Crystal Display 8

LOAD 13 - 14
Logic Compilers 117

M

MAKEHELP -F4 53
MCS-48 106
MCS-51 Additional Features
- Burn Encryption 112
- Burn Security 111
- Compare Encrypted 114
- Load Encrypted 115
- Security Mode 110

Memory Emulation
- Discussion 79 - 83
- How memory is selected 80

MODULES
- 32 pin 102 - 103
- 40 pin 104 - 105
- EPLDs 116 - 120

Installing 99
MCS-48 series 106 - 107
MCS-51 series 108 - 109
Removing 99
MOTOROLA File Format 69
MOVE 34

O

OR/AND Chart 98
OR/AND Table 96 - 97

P

Page editing 12, 100 - 101
Power-down, automatic 5
Power-up 57
Programming Algorithms 88
PROM
24 pin 6
Blank test 23
Checksum without loading 52
Configuration, automatic 31
Configuring 31
Page Mode 11
Quickburn Utility 50
Viewing without loading 49
PROM
Addressing 89
PROM emulation 39
Pulse Table 94 - 95

Q

QUICKBURN - F2 50

R

RECEIVE 26 - 27
RECEIVE silently 87
RS232
- Altering interface signals 59
- Automatic Baud Rate Selection 32
- Baud Rate Selection 32
- Connection and disconnection 7
- CTS, RTS and other signals 61
- Discussion 59
- Dumping to terminal 46
- Editing 45
- Receiving data files 26
- Remote operation 7
- Sending files 28

RS232 Interface Lead 60

S

Searching User Ram 43
SECTOR 10
- Editing 10

SECTOR Zero
- Emulation from 40

SEEK 43
SEND 28 - 29
SEND silently 87
Serial Interface 32

SHUFFLE 19
Silicon Signature 31, 89
Socket, ZIF 5
SPLIT 17 - 18
SWAP 35
SYSTEM CHECK - F5 55
SYSTEM VARIABLES 84
List of locations and meanings 85 - 87

T

TEKHEX File Format 70
Terminal Editing 9
Terminal Emulating Programs 65
TEST 22 - 23
TPA Editing 47 - 48

U

UPGRADES
Battery Abort Message 137
Burn - Visual Feedback 136
Config - Sector Limits 136
Flash ROM Programming 138
Intel 2816 141
Silent Transmission 137
TMS27C292 140

W

WYSIWYG 8

Z

ZIF socket 6