

S 2 0 0 0 I N T E L L I G E N T P R O G R A M M E R

U S E R M A N U A L

Stack Limited
Unit 8
Wedgwood Road Industrial Estate
Bicester
Oxon. OX6 7UL

CONTENTS

1.0	General Operation	1.1
1.1	Power Requirements	1.1
1.2	28 Pin ZIF Sockets	1.1
1.3	Device Type Selection	1.1
1.3.1	Device Selection Table	1.1
1.3.2	Selecting Standard Programming Algorithms	1.1
1.3.3	Selecting Fast Programming Algorithms	1.1
1.3.4	Automatic Device Selection	1.2
1.4	Indicators	1.2
1.5	Audible Indicator	1.3
2.0	Stand-Alone Operation	2.1
2.1	Blank Check	2.1
2.2	Verify	2.1
2.3	Program	2.1
2.4	Erasing EEPROMS	2.2
2.5	Connecting a Monitor	2.2
3.0	Remote Operation via the RS-232 Interface	3.1
3.1	General	3.1
3.2	Summary of Commands	3.1
3.3	The Internal RAM and Socket Memory Map	3.1
3.4	Description Of Remote Commands	3.2
3.5	Command Syntax	3.5
3.5.1	Commands Operating on Data	3.5
3.5.2	Commands initialising The Programmer	3.7
3.6	Operations on Sets Of Devices	3.7
3.7	Indicator	3.9
4.0	The RAM Editor	4.1
4.1	Summary of Editor Commands	4.1
4.2	Description of Editor Commands	4.1
5.0	The RS-232 Interface	5.1
5.1	Baud Rate Selection (7 way Bit Switch)	5.1
5.2	Data Format Selection (6 way Bit Switch)	5.1
5.3	Connector Assignments	5.1
5.4	Handshaking Protocols	5.1
5.4.1	X-ON/X-OFF Protocol	5.1
5.4.2	CTS/DTR Protocol	5.2
5.5	Interfacing With External Equipment	5.2

Appendices

A1	Data Formats	A1.1
A2	Error Reporting	A2.1
A2.1	Error Indications	A2.1
A2.2	Error Messages	A2.2
A3	Device Codes For Manufacturers' Devices	A3.1

1.0 General Operation

1.1 Power Requirements

The programmer is supplied pre-wired for operation at either 220-240 VAC or 110-120 VAC. Check the label on the rear of the programmer shows the correct voltage for your area. The programmer consumes less than 60 Watts maximum (standby operation is typically 4 Watts). The programmer is supplied ready fused. Ensure replacement fuses are of the same value as the original fuse.

When the power is switched on the programmer self tests the memory and indicators, and fills the data memory with 'FF's. All parameters are set to their default value.

1.2 28 Pin ZIF Sockets

The master and copy sockets are fully protected and powered down during insertion. Power is not applied until a device operation cycle has started.

24 pin devices must be inserted in the lower part of the socket.

1.3 Device Type Selection

The programmer allows rapid selection of the correct device type using 2 device selection keys, in conjunction with a 2-digit LED display.

1.3.1 Device Selection Table

The device selection table is printed on the front panel of the programmer under the 6 function keys. To select the required device type and programming algorithm simply enter the 2-digit code from the table corresponding to the device to be programmed. This code is altered using the device select keys to increment or decrement the present number. Holding either key down continuously will cause the display to increment/decrement automatically. Selection of an unallocated code is indicated by 2 flashing decimal points on the display. The 3 function keys and RS-232 interface are inhibited until a valid device code has been selected.

1.3.2 Selecting Standard Programming Algorithms

Some of the device types have only one programming algorithm (e.g. EEPROMS) and therefore only one device code. The 27XX series of EPROMS, however, can be programmed with one or more fast programming algorithms or by the single 50 millisecond pulse method. The 50mS pulse method is selected by entering the code, in the range 00-09, for the required device. E.g. To program a 2764A with a 50 mS pulse select code 05.

1.3.3 Selecting Fast Programming Algorithms

A number of manufacturers now recommend fast programming algorithms to minimise the time required to program the larger memory devices. To select the required fast programming algorithm first check the manufacturer name against the second column in the device table:

10-19	Hitachi, Intel, Mitsubishi, NEC, Seeq, Synertek, Texas
20-29	AMD
30-39	Fujitsu

E.g. The device codes for programming using the Intel programming algorithms are in the range 10-19.

To select the correct code for a particular device (e.g. 27128) refer back to the first column in the table and use that number as an offset into the fast programming range e.g. 27128 using Intel algorithm = device code 16.

Note that some manufacturers recommend fast programming algorithms for some of their devices, and not others. If you are unsure of the correct code to use then check the list of device codes in appendix 3.

IMPORTANT NOTE: FAST PROGRAMMING OF 2716 AND 2732 DEVICES

Manufacturers have only devised fast programming algorithms for use with their latest larger capacity devices. Small devices such as the 2716 and 2732 do not currently have an algorithm which is supported by any manufacturer. Often where the device is to have only a short term use (e.g. in product development) the efficiency of fast programming is still desirable. To meet this requirement the programmer offers a fast programming algorithm for these 2 devices (codes 11 and 12) but neither Stack nor the manufacturer will guarantee the long term integrity of devices programmed in this way.

1.3.4 Automatic Device Selection

An increasing number of manufacturers are incorporating intelligent identifiers into their devices. These identifiers can be read by the programmer and allow the programmer to automatically select the correct device type and programming algorithm. By setting the programmer to Auto-Select mode (device code 99) the identifiers will be read out of each device at the start of each operation. During the operation the device code selected by the programmer using the identifier will be displayed. At completion of each operation the display will return to '99'.

Note: The programmer will check to see if all the devices inserted are of the same type. If different device types are found then the programmer will assume that the left most device is correct (usually the master device) and will indicate as incorrect any devices which are different.

1.4 Indicators

The programmer has eleven tri-colour LEDs for operation and device status indications, and a 2 digit, 7-segment display for device selection. The operation of the device select LEDs is covered in the section on device selection.

Eight tri-colour LEDs indicate the individual copy sockets status as follows:

GREEN: Indicates that the device in this socket, on completion of the previous operation, has passed (i.e. no error detected).

RED: Indicates that the device in this socket failed to complete the previous operation (see section on error indications).

AMBER: Indicates that an operation is in progress and that all sockets are powered up. **WARNING:** under no circumstances must devices be inserted or removed from the sockets whilst the indicators show amber. If it is necessary to remove a device before the completion of an operation then press the RESET key before attempting to remove the device.

BLANK: Indicates that the programmer has not found a device in this socket. This will usually be the case for empty sockets only, and allows the programmer to function faster by only operating on those sockets with devices. If the indicator remains blank when the socket is occupied, then this indicates a faulty device (see section on error indications).

The three main function keys, BLANK CHECK, PROGRAM, and VERIFY each have an associated tri-colour LED indicator which give: operation in progress - amber; operation completed OK - green; and operation failed - red indications. Blank check and verify operations initiate single operations such that only the respective LED will be illuminated during, and on completion of, an operation. The program operation causes the programmer to execute a multi-operation sequence consisting of blank check, program and verify. As each operation is commenced the associated LED will illuminate. On successful completion all 3 LEDs will be green. If any operation has failed the LED specific to that operation will be red along with the PROGRAM LED.

1.5 Audible Indicator

The programmer incorporates a buzzer to give an audible indication of the completion of operations as follows:

- Single short beep = Reset - occurs on power up or after RESET key pressed.
- Three short beeps = Pass - occurs on completion of operation.
- Single long beep = Fail - occurs on termination of operation.

Note: Failures such as 'misinserted devices' and 'shorted power rail' cause the operation to terminate early, and should be re-started with the faulty device removed.

2.0 Stand Alone Operation

There are 3 main function keys on the programmer for initiating blank check, verify and program operations. Ensure that the correct device type is selected before using these keys.

2.1 Blank Check

Pressing the BLANK CHECK key once causes the programmer to check that all the devices in the copy sockets are in the blank (erased) condition. If all devices pass then the BLANK CHECK LED will be set to green. If any single device fails then the BLANK CHECK LED will be set to red, regardless of any other devices which pass. (Devices which have passed/failed are indicated by green/red LEDs respectively). Empty sockets are not checked and are indicated by blank LEDs. If blank check is performed without any devices in the copy sockets an error will be indicated and the BLANK CHECK LED set red, with all copy socket LEDs blank (see error indications in Appendix 2 for other error conditions).

2.2 Verify

Pressing the VERIFY key causes the programmer to check all devices in the copy sockets have the same contents as the device in the master socket. If all devices pass then the VERIFY LED will be set to green. If any single device fails then the VERIFY LED will be set to red, regardless of any other devices which pass. (Devices which have passed/failed are indicated by green/red LEDs respectively). Empty sockets are not checked and are indicated by blank LEDs. If a verify is performed without any devices in the master or copy sockets an error will be indicated and the VERIFY LED set red, with all copy socket LEDs blank (see error indications in Appendix 2 for other error conditions).

2.3 Program

Pressing the PROGRAM key causes the programmer to start a sequence of 3 operations; blank check, program and verify. For these operations to be successful there must be a device in the master socket and at least one device in the copy sockets. If the master or copy sockets are empty then an error will be indicated by a red PROGRAM LED.

BLANK CHECK: The first operation in the sequence is identical to the normal blank check. Whilst the copy sockets are being checked for blank devices both the BLANK CHECK and PROGRAM LEDs will be amber. If the operation is successful then the BLANK CHECK LED will be set to green and the programmer will proceed to the program operation. If the blank check operation fails, the program sequence will pause with the BLANK CHECK LED red and the PROGRAM LED amber. This indicates that one or more devices in the copy sockets are not completely blank. If these are faulty devices they should be replaced, and the programmer reset with the RESET key before recommencing the program operation. If the devices are known to be not blank, but require further programming, simply press the PROGRAM key a second time. The programmer will now check the devices in the copy sockets for illegal bits. If any device fails the illegal bit check, the program sequence is aborted with the PROGRAM and BLANK CHECK LEDs red. If there are no illegal bits the programmer commences programming.

PROGRAM: During the programming operation the BLANK CHECK LED will be green and the PROGRAM LED will be amber. It is at this time that new data will be copied into the devices from the master device. The programmer carries out a check on each location as it is programmed. If a location fails to program then the programmer will not attempt to program any further locations in that device.

VERIFY: After the new data has been programmed into copy devices the programmer then performs a final verify operation. During this operation both the PROGRAM and VERIFY LEDS will be amber. The verify operation is identical to the normal verify operation.

On successful completion of the programming sequence the LEDS above the function keys will all be green, showing that each stage of programming has been completed.

2.4 Erasing EEPROMS

Many EEPROMS offer a fast Chip Erase function. This completely erases the devices to their blank state (usually FFH) in the time taken to program a single location. By erasing the EEPROM in this way before programming the device the programming time can be reduced by up to 50% on some devices.

To erase the EEPROMS place the non-blank devices into the copy sockets and start the normal program sequence by pressing the PROGRAM key. The programmer will start the blank check operation and fail the devices. It will then pause with the BLANK CHECK LED amber and the PROGRAM LED red (as for normal program operation). If the BLANK CHECK key is now pressed this will cause the devices to be erased.

Once erased, the EEPROMS can then be programmed with data by pressing the program key.

Note 1: Devices which do not support a Chip Erase function will have blank data (FFH) programmed into each location by the normal programming method.

Note 2: If the PROGRAM key is pressed again instead of the BLANK CHECK key the programmer will continue programming the non-blank devices and will, where necessary, automatically erase the individual bytes before entering the required data.

2.5 Connecting a Monitor

During stand-alone operation a monitor, VDU or printer can optionally be used to display error messages. The monitoring device is connected to the programmer via the RS-232 interface. The error messages are described in Appendix 2.

3.0 Remote Operation via the RS-232 Interface

3.1 General

In addition to the front panel controls, the programmer can be controlled from a development system or a dumb terminal via the RS-232 interface. The commands enable devices in the copy sockets to be programmed and verified with the data in the internal RAM, and blank checked and erased (EEPROMS only). The data is transferred to the RAM by downloading from a development system via the RS-232 interface, or by reading devices in the sockets.

Use of the editor allows data items to be changed, inserted or deleted. The RAM can be searched to find a specific data string (see section 4 on RAM editor).

In addition, utility commands are available for manipulating blocks of data.

3.2 Summary of Commands

Command	Abbr.	Description
Blank	BL	Checks devices in copy sockets are unprogrammed
Calculate	CA	Transmits difference between 2 addresses
Checksum	CH	Transmits 4 hex digit checksum for stored data in RAM
Combine	CO	Combines odd and even byte blocks into single data block
Device	DE	Selects device code
Display	DI	Displays data stored in RAM via RS-232 in fixed format
Dump	DU	Transmits data in RAM via interface in selected format
Edit	ED	Invokes integral data editor
Erase	ER	Blanks locations of the devices (applies to EEPROMS)
Fill	FI	Stores hex byte into locations in RAM
Format	FO	Selects data format used for input/output via RS-232
Invert	IN	Stores 1's complement of data in RAM
Load	LO	Reads data from RS-232 in selected format into RAM
Move	MO	Copies block of data from one area of RAM to another
Origin	OR	Specifies the address at which internal RAM starts
Program	PR	Blank checks devices, programs and verifies them from RAM
Read	RE	Copies data from master or copy sockets to RAM
Split	SP	Separates odd and even bytes of data into 2 blocks
Verify	VE	Compares copy devices with RAM
Write	WR	Programs copy devices with data in RAM

3.3 The Internal RAM and Socket Memory Map

The programmer has 40K bytes of data RAM available to the user as standard. The addresses of the RAM block and the master and copy sockets can be set to map the target system. Using the ORIGIN command, the first address in the RAM and sockets can be set anywhere between 0000H and 1FF6000H. On power up the RAM is set to occupy addresses 0000H to 9FFFH (i.e. the ORIGIN is set to 0000H).

3.4 Description of Remote Commands

The parameters are all entered in hex and have the following meanings:

XX Data byte used by the command (default value = FFH).
 YY Decimal number used by the command (no default value).
 NNNNN Number of bytes for commands to operate on (default value is memory size of the current device type selected).
 ADDR1 Start address for source block (default value is the current value of ORIGIN).
 ADDR2 Start address for destination block (default value = ADDR1).

Parameters shown in parentheses are optional and assume the above default values. Parameters outside parentheses must be entered to construct a legal command.

List of commands:

BLANK (NNNNN ADDR1)

Checks that NNNNN bytes starting from ADDR1 in the copy sockets are unprogrammed (i.e. data = FFH). This operation can be used on sets of devices (see section on sets of devices).

CALCULATE ADDR1 ADDR2

Calculates the difference between 2 addresses ADDR1 and ADDR2 and transmits the result via the RS-232 interface. It can be used to calculate the number of bytes in a block, given the first and last addresses.

CHECKSUM (NNNNN ADDR1)

Calculates the CCITT standard cyclic redundancy checksum on the data in the RAM starting at ADDR1 for NNNNN locations, and transmits the 4 hex digit result via the RS-232 interface.

COMBINE (NNNNN ADDR1)

Combines 2 blocks of NNNNN/2 bytes each in RAM, starting at ADDR1 and ADDR1+NNNNN/2 into a single block of NNNNN bytes starting at ADDR1. The lower block become the even bytes and the upper block the odd bytes. E.g.:

->DISPLAY 10

```

      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0000 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF .."3DUfw.....

```

->COMBINE 10

->DISPLAY 10

```

      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0000 00 88 11 99 22 AA 33 BB 44 CC 55 DD 66 EE 77 FF ....".3.D.f.w...

```

->

This command is the opposite of the SPLIT command.

DEVICE (YY)

Selects the device code YY if YY is entered. When YY is not entered the programmer returns the device code currently selected (defaults to 01 on power up). The device code is entered and displayed in decimal.

DISPLAY (NNNN ADDR1)

Displays the data stored in NNNN bytes of RAM starting at ADDR1. The data are displayed in both hex and ASCII format. Data corresponding to non-printable ASCII characters are displayed as a period. For example:

->DISPLAY 20

```

      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0000 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF  .."3DUfw.....
0010 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF  .."3DUfw.....
->
```

The ESC character can be sent to abort output, and X-ON/X-OFF flow control can be used if the protocol is enabled (see section 5.4).

DUMP (NNNN ADDR1)

Transmits NNNN bytes of data stored in RAM starting at address ADDR1 via the RS-232 interface in the currently selected format. In formats which include a 16 or 24 bit address field, the highest address being dumped must be 16 or 24 bits or less respectively (i.e. less than 1000H and 100000H respectively).

EDIT (ADDR1)

Invokes the RAM data editor, allowing access to the editor commands. The editor cursor is set to ADDR1 and the current line of data is displayed. E.g.:

->EDIT 1008

```

      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
1000 00 11 22 33 44 55 66 77>88 99 AA BB CC DD EE FF  .."3DUfw.....
:
```

ERASE (NNNN ADDR1)

Sets NNNN bytes starting at ADDR1 to FFH in the devices in the copy sockets. This is only available when the device type currently selected is an EEPROM. This operation can be used on sets of devices (see section on sets of devices).

FILL (XX NNNN ADDR1)

Stores the hex byte XX into NNNN bytes of RAM, starting at ADDR1.

FORMAT (YY)

Selects the load and dump data format to YY (see Appendix 1 for Data Formats) if YY is entered. If YY is not entered the programmer returns the data format currently selected (defaults to 01 on power up). The format number is entered and displayed in decimal.

INVERT (XX NNNN ADDR1)

Exclusive-ORs NNNN bytes of data in RAM starting at ADDR1 with the hex byte XX. If XX = FFH (the default value) then the data bytes are inverted.

LOAD (NNNNN ADDR1)

Reads data from the RS-232 interface in the currently selected data format and stores in RAM at the required addresses. Loading will be aborted if there is a checksum error, or a load address outside the RAM addresses is encountered, and an error message is issued. Formats 12 and 13 allow the number of bytes to be loaded and the start address in RAM to be specified.

MOVE NNNNN ADDR1 ADDR2

Copies NNNNN bytes of data from a block starting at ADDR1 in RAM to a block starting at ADDR2 in RAM.

ORIGIN (ADDR1)

Sets the address of the first byte of RAM and the first address in the ZIF sockets if ADDR1 is entered. If ADDR1 is not entered the programmer returns the current value of the first address. ADDR1 can be specified in the range 0000H to 1FF6000H.

PROGRAM (NNNNN ADDR1 ADDR2)

Copies NNNNN bytes from the master data in RAM starting at ADDR1 to devices in the copy sockets starting at ADDR2. Before programming the devices, the specified locations are blank checked. After programming, the specified locations are verified against the RAM data. This operation can be used on sets of devices (see section on Sets of Devices).

READ (NNNNN ADDR1 ADDR2)

Copies NNNNN bytes of data from a device or devices in the ZIF sockets starting at ADDR1 into RAM starting at ADDR2. This operation can be used with sets of devices (see section on sets of devices). If the set size is 1, the device to be read should be in the master socket. For set sizes of 2 or greater, the devices should be placed in the copy sockets.

SPLIT (NNNNN ADDR1)

Splits a block of NNNNN bytes starting at ADDR1 in RAM into 2 blocks of even and odd bytes. The even bytes start at ADDR1, and the odd bytes start at ADDR1+NNNNN/2. E.g.:

->DISPLAY 10

```

      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0000 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF  .."3DUfw.....

```

->SPLIT 10

->DISPLAY 10

```

      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0000 00 22 44 66 88 AA CC EE 11 33 55 77 99 BB DD FF  ."Df.....3Uw....

```

->

This command is the opposite of the COMBINE command.

VERIFY (NNNNN ADDR1 ADDR2)

Verifies NNNNN bytes in devices in copy sockets starting at ADDR2 with the master data in RAM starting at ADDR1. This operation can be used with sets of devices (see section on Sets of Devices).

WRITE (NNNNN ADDR1 ADDR2)

Copies NNNNN bytes from the master data in RAM starting at ADDR1 to devices in the copy sockets starting at ADDR2. The devices are not blank checked or verified, but there is a byte verify when using some fast programming algorithms.

3.5 Command Syntax

Commands are received by the programmer via the RS-232 interface. On power up, and at completion of each command the programmer issues a prompt:

->

Commands are entered after the prompt. The command comprises a valid command name optionally followed by data parameters. Upper and lower case letters may be entered, and 'backspace' can be used to delete incorrect entries. Names can be shortened, when entered, to a minimum of 2 letters and they will still be understood by the programmer. Entries which cannot be understood are echoed with a '?'. The following are all examples of valid commands:

```
->PROGRAM
->program
->PROG
->pr
->
```

A number of commands can be entered on a single line, and they will be executed in sequence. For example:

```
->ORIGIN C000 FORMAT 2 DEVICE 16 LOAD PROGRAM
->
```

This could be output from a development system before downloading some data, and at completion of downloading, a 27128 would be programmed. It could, of course, be shortened to:

```
->OR C000 FO 2 DE 16 LO PR
->
```

Commands which output characters (e.g. DISPLAY, DUMP) can be controlled by X-ON/X-OFF handshaking or CTS/DTR handshaking (see section 5.4). Output can be aborted by sending the ESC character to the programmer.

The parameters required by the commands depend on the type of command. There are principally 2 types: to operate on data, and initialise the programmer.

3.5.1 Commands Operating on Data

All of these commands operate on a continuous data block. The user may optionally specify the data block by entering, after the command name, its size (number of bytes) and the address of the first byte in the block. The numbers are entered in hexadecimal and separated by spaces. For example, to calculate the checksum on 4000H bytes starting at address 0000H, the user enters:

```
->CHECKSUM 4000 0
->
```

If the user doesn't specify one or both of the parameters after the command, the programmer will assign default values. The default value for the number of bytes is the size of the device type currently selected. The default value of the start address is the same as the first location of the RAM, i.e. equal to the ORIGIN. Note that the start address cannot be entered without the number of bytes, since the command expects the first number after the command, if entered, to be the block size.

With the device type set to, say, 06, for a 27128, the above command could also be entered as follows, and still have the same effect:

```
->CHECKSUM
->CH 4000
->
```

Commands which operate on a block of data, and can have the number of bytes and start address optionally specified are:

BLANK	COMBINE	DUMP	SPLIT
CHECKSUM	DISPLAY	ERASE	

Some commands move or copy a block of data, and the user may additionally specify the destination address of the first byte in the block. For example to copy a block of 4000H bytes of data in RAM starting at address 1000H into one or more devices in copy sockets, starting at 0000H, the user enters:

```
->PROGRAM 4000 1000 0
->
```

Again the programmer will assign default values to unspecified parameters. As a default the destination address is set to the same value as the start address. With the device type set to 06 (27128) for example, the following commands could all be entered as follows with the same effect:

```
->PROGRAM 4000 0 0
->PROGRAM 4000 0
->PR 4000
->PR
->
```

Commands which have parameters in this format are:

MOVE *	READ	VERIFY	WRITE
PROGRAM			

* The MOVE command must have all the parameters specified to form a valid command.

The FILL and INVERT commands can have a data byte specified (default value is FFH) and this should be entered before the number of bytes. E.g.:

```
->FILL 00 4000 0
->INVERT FF 4000 0
->
```

3.5.2 Commands Initialising the Programmer

These commands set values within the programmer and expect a single number to follow the command. E.g.:

```
->DEVICE 6
->ORIGIN C000
->FORMAT 2
->
```

If a number does not follow the command, the current value of the parameter is transmitted by the programmer.

3.6 Operations on Sets Of Devices

The programmer normally copies the same data into 1 or more devices. Using the remote commands the programmer can be instructed to copy different data into adjacent devices, to generate a set of devices in a single operation. The number of devices in a set can be in the range 1 to 8. For example a program of 4000H bytes can be copied into a set of 8 x 2716's (2K devices), a set of 4 x 2732's (4K devices) or a set of 2 x 2764's (8K devices).

The devices are treated as contiguous blocks of memory with ascending addresses from left to right. For example for a set of 8 x 2716's, the addresses of the 8 devices are:

Set No.>	1							
	<							>
	1	2	3	4	5	6	7	8
	0000- 07FF	0800- 0FFF	1000- 17FF	1800- 1FFF	2000- 27FF	2800- 2FFF	3000- 37FF	3800- 3FFF

(For sets of less than 8 devices, the highest number sockets are left empty).

For sets of 3 or 4 devices, a second set can also be programmed simultaneously, whilst up to 4 sets of 2 devices can be programmed together.

For example, 2 sets of 4 devices can be programmed as:

Set No.>	1				>	2				>
	<				<				>	
	1	2	3	4	1	2	3	4		
	0000- 0FFF	1000- 1FFF	2000- 2FFF	3000- 3FFF	0000- 0FFF	1000- 1FFF	2000- 2FFF	3000- 3FFF		

(For sets of 3 devices, the sockets labelled '4' are left empty).

Up to 4 sets of 2 x 2764's are programmed as:

Set No.>	< 1 >	< 2 >	< 3 >	< 4 >
	1	2	1	2
	0000- 1FFF	2000- 3FFF	0000- 1FFF	2000- 3FFF
	1	2	1	2
	0000- 1FFF	2000- 3FFF	0000- 1FFF	2000- 3FFF
	1	2	1	2
	0000- 1FFF	2000- 3FFF	0000- 1FFF	2000- 3FFF

Note: The addresses shown in the above examples are true only when the ORIGIN is set to 0000H. The addresses for the sockets are offset when the ORIGIN is set to any other value (see section 3.3).

The following remote commands will operate on sets of devices:

BLANK	PROGRAM	VERIFY
ERASE	READ	WRITE

The number of devices in a set is determined automatically once a command has been entered, with its associated parameters. The number of bytes and the start address in the ZIF sockets defines a data block. The top address in the data block defines how many devices are in a set. For example, to program two 8K devices with a 16K block of data, enter:

```
->PROGRAM 4000
->
```

The block size does not have to be an exact multiple of the device size. The command:

```
->PROGRAM 2800 800
->
```

would also define the set size as 2 (for 8K devices) and program data from addresses 0800H to 2FFFH into the same addresses in the devices in the copy sockets.

Devices for 16 bit processors can be programmed in a single operation by treating them as sets of devices. Data is loaded into the programmer from, say, a development system, and then SPLIT on a device boundary, and copied into set(s) of devices.

For example, to generate a set of 2764 devices for an 1800H word program for a 16 bit processor 3000H bytes are loaded into the programmer RAM from address 0000H (with the ORIGIN set to 0000H), then the data is SPLIT into even and odd address bytes on a device boundary:

```
->SPLIT 4000
->
```

The even and odd bytes are programmed in a single operation using:

```
->PROGRAM 4000
->
```

One or more sets of devices will be programmed in the copy sockets as:

Set No.>	< 1 >	< 2 >	< 3 >	< 4 >				
	EVEN	ODD	EVEN	ODD	EVEN	ODD	EVEN	ODD
	0000- 1FFF	0000- 1FFF	0000- 1FFF	0000- 1FFF	0000- 1FFF	0000- 1FFF	0000- 1FFF	0000- 1FFF

If the program is copied into 2732's the commands are the same as above, and the devices are:

Set No.>	< 1 >				< 2 >			
	EVEN	EVEN	ODD	ODD	EVEN	EVEN	ODD	ODD
	0000- 0FFF	1000- 1FFF	0000- 0FFF	1000- 1FFF	0000- 0FFF	1000- 1FFF	0000- 0FFF	1000- 1FFF

3.7 Indicators

Remote commands which access devices in the ZIF sockets update the tri-colour LEDS during, and at completion of, the command. The operation of the indicators is the same as for stand-alone operation of the programmer, and is detailed in section 1.4.

The audible buzzer also indicates pass or fail at completion of these operations.

4.0 The RAM Editor

The editor is entered using the EDIT command. The cursor address can optionally be specified after EDIT, otherwise editing commences at the start of the RAM.

Whilst in the editor the prompt is a colon. Only editor commands are available whilst in the editor. These are:

DISPLAY ERASE FIND GOTO INSERT QUIT REPLACE

The command names can be shortened, when entered, to a minimum of a single letter, and they will still be understood by the programmer.

4.1 Summary of Editor Commands

Command	Abbr.	Description
Display	D	Displays data with cursor and allows data modification
Erase	E	Removes data bytes from memory
Find	F	Finds a specified data string
Goto	G	Moves the editor cursor to the specified location
Insert	I	Inserts a specified data string into memory after cursor
Quit	Q	Exits from the editor mode
Replace	R	Used with Find to change specified data strings

4.2 Description of Editor Commands

The parameters are all entered in hex and have the following meanings:

NNNNN Number of bytes for command to operate on (no default value).
string Hex data string.

Parameters shown in parentheses are optional. Parameters outside the parentheses must be entered to construct a legal command.

List of commands:

DISPLAY (NNNNN)

Displays NNNNN bytes from the current cursor position, if NNNNN is entered. E.g.

->EDIT 1008

0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF

1000 00 11 22 33 44 55 66 77>88 99 AA BB CC DD EE FF .."3DUfw.....

: DISPLAY 14

1000 00 11 22 33 44 55 66 77>88 99 AA BB CC DD EE FF .."3DUfw.....

1010 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF .."3DUfw.....

:

If the number of bytes is not entered, the data is displayed on a byte by byte basis, allowing data to be changed. To change a data byte the new hex value is typed in alongside the existing value, followed by a carriage return. To leave a byte unchanged, simply enter a carriage return. To exit from the display routine enter any non-hex character followed by carriage return. For example:

```
: DISPLAY
  1008 88 01
  1009 99
  100A AA 23
  100B BB .
: DISPLAY 4

  1000 00 11 22 33 44 55 66 77 01 99 23>BB CC DD EE FF .."3DUfw..#.....
:
```

ERASE NNNNN

Deletes NNNNN bytes starting at the cursor position. All remaining data to the end of the RAM are moved to replace the deleted bytes. The vacant bytes at the end of the RAM are filled with FFH.

FIND (string)

Searches through the RAM for the specified hex data string from the cursor to the end of the RAM. The string can be up to 32 bytes. If a string is not entered, the FIND command uses the 'find' string last entered, allowing repeated searches for a single string.

GOTO ADDR1

Moves the editor cursor to ADDR1, and displays the line containing the cursor.

INSERT (string)

Inserts the specified data string, starting at the current cursor position. All existing data from the cursor to the end of the RAM are moved to accommodate the inserted string, and data at the end of the RAM are lost. If a data string is not entered, the INSERT command uses the 'insert' string last entered, allowing repeated inserts of a data string. This command can be used in conjunction with the FIND command to find and insert different strings.

Note: The 'insert' string is held in the same buffer as the 'replace' string.

QUIT

Exits from the editor.

REPLACE (string)

Replaces the string at the cursor, found with the FIND command, with the specified string. Existing data from the cursor to the end of the RAM are moved to accommodate the replacement. If data bytes are moved towards the cursor the vacant bytes at the end of the RAM are filled with FFH. If data bytes are moved towards the end of the RAM then data at the end of the RAM are lost. If a data string is not specified, the REPLACE command uses the 'replace' string last entered, allowing repeated FIND and REPLACE operations.

Note: The 'replace' string is held in the same buffer as the 'insert' string.

5.0 The RS-232 Interface

The programmer is provided with an RS-232C compatible serial interface. The interface is accessible via the standard 25 way 'D' connector mounted on the rear of the programmer. The 13 bit switches to the left of the connector select baud rate, data format and handshake protocol as follows:

5.1 Baud Rate Selection (7 way Bit switch)

Switch No.>	1	2	3	4	5	6	7
Baud Rate >	19200	9600	4800	2400	1200	600	300

Select a baud rate by setting the appropriate switch closed/down. All other switches should be open/up.

5.2 Data format Selection (6 way Bit Switch)

Switch No.>	1	2	3	4	5	6
Parameter >	Parity	Parity	Stop Bits	Data Bits	Duplex	X-on/Xoff
Open/Up	Enabled	Odd	1	7	Half	Disabled
Closed/Down	Disabled	Even	2	8	Full	Enabled

5.3 Connector Assignments

Pin No.	Signal Description	Input/Output to programmer
1	Earth	---
2	Transmitted Data	Input
3	Received Data	Output
5	Clear to Send	Output
6	Data Set Ready	Output (always high)
7	Data Earth	---
8	Data Carrier Detect	Output (always high)
20	Data Terminal Ready	Input

Note: DTR has an internal pull up resistor and will default to the ON condition.

5.4 Handshaking Protocols

The programmer supports 2 handshaking protocols: X-ON/X-OFF and CTS/DTR. The X-ON/X-OFF protocol can be disabled using the 6 way bit switch on the rear of the programmer.

5.4.1 X-ON/X-OFF Protocol

During output of characters the programmer will respond to an X-OFF character (DC3) and pause the output, waiting for an X-ON character (DC1) before recommencing output.

The Programmer does not support X-ON/X-OFF handshaking for incoming data.

The X-ON/X-OFF handshaking protocol requires only a 3 wire interface to be implemented.

5.4.2 CTS/DTR Protocol

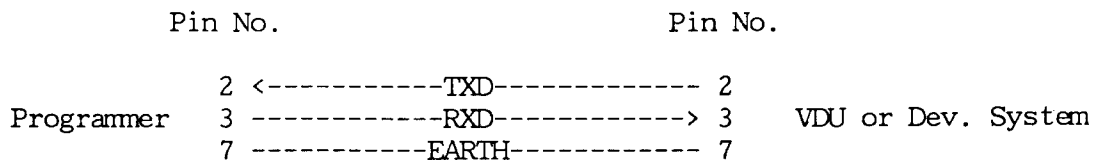
Output of data from the programmer can be controlled using the DTR signal line (pin 20). When DTR is pulled to a low potential, the programmer will halt output, until DTR is pulled high.

The programmer controls the input of data and command lines using the CTS line. When CTS is pulled to a low potential the programmer is not able to accept input. A character which is being transmitted to the programmer at the time CTS is pulled low will be held in the input buffer, until the programmer is ready to read it.

5.5 Interfacing with External Equipment

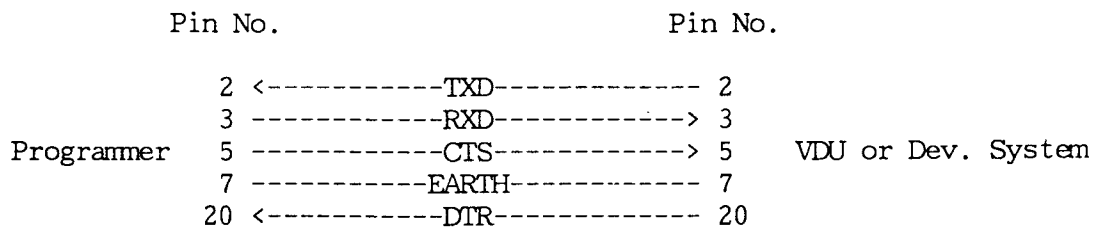
The RS-232 interface on the programmer is configured to enable terminals and VDUs to be connected directly, without concern for the interface signals.

If the external equipment is configured as a terminal, and does not require hardware handshaking (CTS/DTR) a 3 wire interface is sufficient, connecting the signal earth and 2 data signal lines as follows:



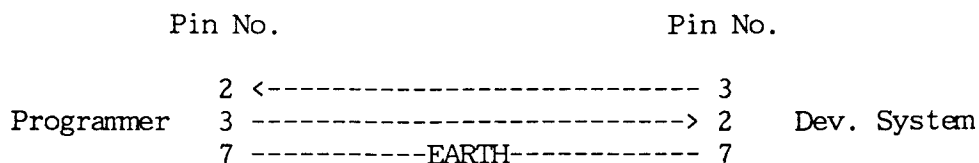
The programmer requires no further connections, but the external equipment may require pins 4 and 5 to be connected together, and/or pins 6, 8 and 20 to be connected together.

Where the programmer is to hardware handshake with the equipment a 5 wire interface is required:



Again pins 6, 8 and 20 may need to be connected together at the external equipment.

When the external equipment normally expects a terminal to be connected to it the signal lines need to be 'swapped' over. When hardware handshaking is not required a 3 wire interface as follows can be used:



The external equipment may require pins 4 and 5 and/or pins 6, 8 and 20 to be connected together.

When handshaking is required a 5 wire interface is required:

	Pin No.		Pin No.	
	2	<-----	3	
	3	----->	2	
Programmer	5	----->	20	Dev. System
	7	-----EARTH-----	7	
	20	<-----	5	

Again pins 6, 8 and 20 may need connecting together at the external equipment.

Appendix 1 Data Formats

Format 1 Intel Intellec

Example:

```
:10FFF000112233445566778899AABBCCDDEEFF0009
:020000021000EC
:050000001122334455FC
:00000001FF
```

Char No.	Description
1	Record Mark: A 'colon' marks the beginning of a record.
2-3	Byte Count: Hex representation of the number of data bytes in the record. A byte count of zero indicates an end-of-file record.
4-7	Address Field: Hex representation of address to load first data byte, with following data bytes being loaded into successive addresses.
8-9	Record Type: Data records are type 00. An end-of-file record is 01. An Extended Address Record is 02.
10-N	Data Bytes: Each data byte is represented by 2 hex characters. The 2 data bytes in an extended address record represent A4-A19 of the segment address with A0-A3 = 0.
N+1 - N+2	Checksum: 2 character hex checksum which is the 2's complement of the sum of all the bytes in the record except the colon and the checksum, evaluated modulo 256. The sum of all the bytes in the record plus the checksum is zero.
N+3 - N+4	Carriage return, line feed.

Format 2 Motorola 'S' Records

S0, S1, S2 and S9 records can be received by the programmer. S1, S2 and S9 records are transmitted.

Example:

```
S1130000112233445566778899AABBCCDDEEFF00F4
S2090100101122334455E6
S9030000FC
```

S1 Rec. Char No	S2 Rec. Char No	S9 Rec. Char No	Description
1	1	1	Record Mark: An 'S' marks the beginning of a record.
2	2	2	Record Type: Type 1 records and type 2 records are data records having 16 bit and 24 bit address fields respectively. Type 9 records are end-of-file records.
3-4	3-4	3-4	Record Length: Hex representation of number of bytes in record from and excluding byte count, to and including checksum.
5-8	5-10	5-8	Address Field: Hex representation of address to load first data byte, with following data bytes being loaded into successive locations.

9-N	11-N	Data Bytes: Each data byte is represented by 2 hex characters.
N+1-N+2	N+1-N+2 9-10	Checksum: A 2 character hex checksum which is the 1's complement of the sum of all the bytes in the record except the record type and the checksum, evaluated modulo 256.
N+3-N+4	N+3-N+4 11-12	Carriage return, line feed.

The programmer will dump all S2 records if the last byte specified has an address of greater than 16 bits, otherwise all S1 records will be transmitted. An S9 end-of-file record must be received by the programmer to finish loading of data.

Format 3 Tektronix Hexadecimal

Example:

```
/00001001112233445566778899AABBCCDDEEFF00F0
/0010050611223344551E
/00000000
```

Char No.	Description
1	Record Mark: A '/' marks the beginning of a record.
2-5	Address field: Hex representation of address to load first data byte, with following data bytes being loaded into successive locations.
6-7	Byte Count: Hex representation of number of data bytes in record. A byte count of zero indicates an end-of-file record.
8-9	Checksum: 2 character hex checksum representing the sum of the hex values of the 6 digits 2-7, evaluated modulo 256.
10-N	Data Bytes: Each data byte is represented by 2 hex characters.
N+1 - N+2	Checksum: A 2 character hex checksum representing the sum of the hex values of the digits of the data bytes, evaluated modulo 256.
N+3 - N+4	Carriage return, line feed.

Format 4 RCA Cosmac

Example:

```
!M
0000 112233445566778899AABBCCDDEEFF00;
0010 1122334455
```

Char No.	Description
1-2	File Mark: The '!M' characters mark the beginning of the file.
3-4	Carriage return, line feed (optional when loading data into the programmer).
5-8	Address Field: Hex representation of address to load first data byte, with following data bytes being loaded into successive locations (the address field is optional in successive records when loading into the programmer - see character N+1).
9	'Space' character.
10-N	Data Bytes: Each data byte is represented by 2 hex characters.

- N+1 End-of-record Mark: A ';' indicates another record follows, with an expressed address. A ',' indicates another record follows, without an address. No character indicates end-of-file. (Either ';' or ',' can be received when loading data into the programmer).
- N+3 - N+4 Carriage return, line feed.

Format 5 ASCII-Hex (Space)

Example:

```
[STX][SOH]$A0000,
11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00
11 22 33 44 55 [ETX]
$$S08F7,
```

Char No.	Description
1-2	File Mark: Non-printable STX and SOH characters mark the beginning of the file (either or both of these characters are optional when loading data into the programmer).
3-4	Address Mark: The '\$A' characters indicate the start address follows.
5-8	Address Field: Hex representation of address to load first data byte, with following data bytes being loaded into successive locations.
9	Data Mark: A ',' character indicates beginning of data (optional when loading data into the programmer).
10-11	Carriage return, line feed (optional when loading data into programmer).
12-N	Data Bytes: Each byte is represented by 2 hex characters followed by a space character. Carriage returns and line feeds can be inserted between any space and the next data byte.
N+1	Terminate Mark: Non-printable ETX character after a space marks the end of data.
N+2 - N+3	Carriage return, line feed (optional when loading data into programmer).
N+4 - N+5	Checksum Mark: The '\$S' characters indicate the checksum follows.
N+6 - N+9	Checksum: A 4 character checksum representing the sum of the hex data bytes, evaluated modulo 65,536.
N+10	File Mark: A ',' indicates end-of-file.
N+11-N+12	Carriage return, line feed.

Format 6 ASCII-Hex (Comma)

Example:

```
[STX][SOH]$A0000,
11,22,33,44,55,66,77,88,99,AA,BB,CC,DD,EE,FF,00,
11,22,33,44,55,[ETX]
$$S08F7,
```

See Ascii-Hex (Space) for description.

Format 7 ASCII-Hex (Apostrophe)

Example:

```
[STX][SCH]$A0000,
11'22'33'44'55'66'77'88'99'AA'BB'CC'DD'EE'FF'00'
11'22'33'44'55'[ETX]
$S08F7,
```

See ASCII-Hex (Space) for description.

Format 8 ASCII-Hex (Percent)

Example:

```
[STX][SCH]$A0000,
11%22%33%44%55%66%77%88%99%AA%BB%CC%DD%EE%FF%00%
11%22%33%44%55'[ETX]
$S08F7,
```

See ASCII-Hex (Space) for description.

Format 9 ASCII-BHLF

Example:

```
[STX]BLIHLLIHF BLIHLLIHLF BLIHLLIHF BLHLLIHLF
BLHLHLHLHF [ETX]
```

Char No.

Description

- | | |
|-----|--|
| 1 | File Mark: Non-printable STX character marks the beginning of the file (optional when downloading data into programmer). |
| 2-N | Data Bytes: Each byte is represented by 10 characters. A 'B' character indicates the start of a data byte, 'H' and 'L' characters indicate '1' and '0' respectively for the 8 bits of the data byte, and an 'F' character indicates the end of a data byte. 'Space' characters and carriage return, line feeds are inserted between data bytes for clarity (optional when loading data into programmer). |
| N+1 | File Mark: Non-printable ETX character marks the end-of file. |

Format 10 ASCII-BNPF

Example:

```
[STX]BNNPNNNPF BNNPNNNPNF BNNPPNNPPF BNPNNNNPNNF
BNPNPNPNPF [ETX]
```

See ASCII-BHLF for description.

Format 11 ASCII-B10F

Example:

```
[STX]B00010001F B00100010F B00110011F B01000100F
B01010101F [ETX]
```

See ASCII-BHLF for description.

Format 12 Binary

The RS232 interface must be set to 8 data bits, no parity for binary file transfer.

When loading data in format 12, the number of data bytes and start address can optionally be entered following the LOAD command. Otherwise the programmer defaults to the current device size and the start of RAM.

Char No.	Description
1	File Mark: FFH character marks the beginning of the file.
2-N	Data Bytes: Each data byte is represented by 1 character.
N+1	Checksum: A one byte checksum which is the sum of all of the data bytes evaluated modulo 256.

Format 13 DEC Binary

The RS232 interface must be set to 8 data bits, no parity for binary file transfer.

The byte count and start address of the data to be downloaded are included in the first bytes of the data block, as defined below.

Char No.	Description
1	File Start Mark: 01H character marks the beginning of the file.
2	Null Mark: 00H character must follow the File Start Mark.
3-4	Byte Count: The number of data bytes in the file excluding the checksum byte (includes the File Start Mark etc.) Character 3 is the low 8 bits of the Byte Count.
5-6	Address Field: The address to load the first data byte, with subsequent data bytes being loaded into successive locations. Character 5 is the low 8 bits of the Address Field.
7-N	Data Bytes: Each Data Byte is represented by 1 character.
N+1	Checksum: A one byte checksum which is the sum of all of the data bytes, evaluated modulo 256.

Format 14 Extended DEC Binary

The format is similar to DEC Binary with the Address Field and Byte Count extended to 24 bit numbers. The byte count and start address of the data to be downloaded are included in the first bytes of the data block, as defined below.

The RS232 interface must be set to 8 data bits, no parity for binary file transfer.

Char No.	Description
1	File Start Mark: 01H character marks the beginning of the file.
2	Null Mark: 00H character must follow the File Start Mark.
3-5	Byte Count: The number of data bytes in the file excluding the checksum byte (includes the File Start Mark etc.) Character 3 is the low 8 bits of the Byte Count, and character 5 is the most significant 8 bits.
6-8	Address Field: The address to load the first data byte, with subsequent data bytes being loaded into successive locations. Character 6 is the low 8 bits of the Address Field, with character 8 the most significant 8 bits.
9-N	Data Bytes: Each Data Byte is represented by 1 character.
N+1	Checksum: A one byte checksum which is the sum of all of the data bytes, evaluated modulo 256.

Appendix 2 Error Reporting

A2.1 Error Indications

During all operations the programmer carries out device fault and operation error checking. The results of these tests are displayed on the 8 copy socket LEDS and 3 function key LEDS. Also a buzzer gives an indication of pass or fail conditions.

When an operation is initiated the error checking sequence is:

- 1 Vcc and Vpp Overcurrent checks (continuous throughout operation)
- 2 Misinserted Devices Check
- 3 Missing Devices Check
- 4 Data Line Drive Check

And at completion of the operation:

- 5 Failed Devices Check

Errors corresponding with the above sequence are displayed as follows:

Vcc AND Vpp Overcurrent Check: As the sockets are powered up, and thereafter continuously throughout the operation, they are checked for an overcurrent condition on both the Vcc and Vpp pins. If an overcurrent condition is detected the operation is immediately aborted, and the sockets are powered down. A Vcc overcurrent condition is indicated by a single red LED on the failed device. The function LED will be blank. A Vpp overcurrent condition will be indicated by all the copy socket LEDS being set to red, with the function LED blank. The normal cause of an overcurrent condition is device misinsertion. Check all devices for correct alignment and orientation.

Misinserted Devices Check: A check is made to ensure that the data bus and address lines are not shorted. If a short is present on these buses then it will be indicated by red LEDS on all the copy sockets, with a red LED above the associated function key.

Missing Devices Check: If an operation is started without the necessary devices present in the master and/or copy sockets the programmer will indicate an error by setting all the copy socket LEDS blank, with the function LED red.

Data Line Drive Check: Each device is checked for its ability to drive to a logic high (2.4V) on the data bus. Devices failing this test will be ignored for the rest of the operation, and be indicated by a blank LED above the copy socket on completion.

Failed Devices Check: If a device fails during an operation the LED above the copy socket of the failed device will be set to red, with the function LED also set to red.

Table of Error Indications:

Description of Error	Copy LEDES	Funcn. LEDES	RS232 Error Message
Vcc Overcurrent	Single Red	Blank	shorted power rail
Vpp Overcurrent	All Red	Blank	shorted power rail
Misinserted Devices	All Red	Red	device misinserted
Missing Devices	Blank	Red	device missing
Data Line Drive	Blank	Green	None
Failed Devices	1-8 Red	Red	device failed

A2.2 Error Messages

Error Message

Cause of Error

<bad data>

A parameter following a command cannot be accepted for one of the following reasons:

- Number is not hexadecimal
- Number is too large to be a data byte
- Can't SPLIT or COMBINE an odd number of bytes or less than 4 bytes
- Invalid format code
- String contains non-hexadecimal character

<byte count out of range>

The command cannot be accepted for one of the following reasons:

- Byte count parameter is 0 or negative
- Byte count parameter is larger than RAM size
- Byte count parameter is larger than a set of 8 devices for the device type currently selected
- Number of bytes to INSERT or REPLACE in the editor is too large for the RAM size

<address out of range>

An address parameter following the command cannot be accepted for one of the following reasons:

- The address is less than the ORIGIN address
- The address is larger than the top RAM address
- The address is negative
- The address, in conjunction with the byte count, specifies the top address of the data block to be greater than the top RAM address
- The address, in conjunction with the byte count, specifies the top address of the data block to be greater than the top address in a set of 8 devices for the device type currently selected
- The ORIGIN address is specified such that the top RAM address is larger than 0FFFFFFFH
- An address received while loading data into the programmer is less than the ORIGIN address or greater than the top RAM address
- The address, in conjunction with the byte count, specifies the top address of the data block to be too large for the address field of the data format currently selected, when transmitting data using the DUMP command

Error Message	Cause of Error
<parameter missing>	<ul style="list-style-type: none"> - A parameter which must be entered to make a valid command is missing - A character which is expected, when data is being loaded into the programmer, is missing
<string not found>	The FIND command is unable to match the specified string with the data in the RAM
<checksum error>	The checksum received when loading data into the RAM is not the same as the checksum calculated by the programmer
<device failed>	One or more devices have failed the operation just selected. The LEDS on the front panel will indicate the faulty device(s)
<master missing>	The master socket is empty during an operation which must have a master device
<device missing>	<p>During an operation the programmer found a copy socket empty for one of the following reasons:</p> <ul style="list-style-type: none"> - All copy sockets are empty - A device which is to be accessed in a set of devices is missing - No device is present when the programmer is determining the default device size with the device code set to 99 (automatic select using intelligent identifiers)
<incorrect device type>	<ul style="list-style-type: none"> - When operating with the device code set to 99 (automatic select using intelligent identifiers) the devices in the ZIF sockets do not all have the same identifier codes. (Both the device and manufacturer codes must be the same). - The device type currently selected cannot be erased - The device code selected with the DEVICE command is not available
<shorted power rail>	The Vcc or Vpp power rails are taking excessive current. The LEDS on the front panel will indicate the faulty device(s). Excess current drawn on pin 24 in automatic select using intelligent identifiers will also cause error
<device misinserted>	A device has been inserted backwards, or misjustified, or 2 or more of the pins on the device are shorted together illegally
<error code n>	This error is a result of self-test checks that the programmer executes continuously. If this error occurs, note down the value of the code n, the use of the programmer at the time of the error code, and contact Stack

Appendix 3 Device Codes for Manufacturers' Devices

Device Part Number	Device Code	Device Part Number	Device Code
Advanced Micro Devices		Fujitsu	
AM2716	01	8516	01
AM9716	01	MBM2716	01
AM2716B	21,99	MBM8532	02
AM2732	02	MBM2732	02
AM2732A	03	MBM2732A	03
AM2732B	23,99	MBM2764	34
AM2764	24	MBM27C64	34
AM2764A	25,99	MBM27128	36
AM2764A OTP	25,99	MBM27256	18
AM27128	26	MBM27C256	38
AM27128A	27,99	MBM27512	39
AM27128A OTP	27,99	MBM27C512	39
AM27256	28,99		
AM27256 OTP	28,99	General Instrument	
AM27512	29,99	27C64	15,99
		27256	18,99
AM9864	68	27C256	18,99
		27C512(0)	19,99
Atmel		27C512(2)	59
AT27C256	18,99		
AT27C512	19,99	Hitachi	
AT27C513	59,99	HN462716	01
		HN462732	02
Electronic Arrays		HN482732A	03
2716	01	HN462532	42
		HN482764	14
Electronic Designs		HN482764P	14
EDH7816N	06	HN27C64	14
EDH7816C	06	HN4827128	16
		HN27128A	17,99
Eurotechniques		HN27128AP	17,99
ET2716	01	HN27256	18,99
ETC2716	01	HN27256P	18,99
ET2732	02	HN27C256	18,99
ETC2732	02	HN27512	19,99
		HN27512P	19,99
Fairchild Semiconductor		HN27C101	*10,99
F2716	01	HN48016	66
F2732	02	Hyundai	
		HY27C64	25,99

* Using the S2030 Programming Adaptor

Device Part Number	Device Code	Device Part Number	Device Code
Intel		National Semiconductor	
2758	00	MM2758A	00
2758S1865	49	MM2758B	49
2716	01	MM2716	01
2732	02	NMC2716	01
2732A	03,99	NMC6716	01
P2732A	03,99	NMC27C16	01
2764	14,99	NMC27C16B	51
2764A	15,99	NMC2532	42
27C64	15,99	NMC2732	02
27128	16,99	NMC27C32	02
27128A	17,99	NMC27C32B	53,99
27128B	17,99	NMC27C64	55,99
27256	18,99	NMC27C256	58,99
27C256	18,99		
27512	19,99	NMC9716	74
27513	59,99	NMC9817	75
27011	10,99	NMC98C64	76
27010	*10,99		
		NCR	
2816A	60		
2817A	61	52817	78
Mitsubishi		NEC	
M5L2716	01	uPD2716	01
M5L2732	02	uPD8716	01
M5L2764	14	uPD2732	02
M5L27128	16	uPD2732A	03
M5M27C128	16	uPD2764	14
M5L27256	18	uPD27C64	14
		uPD27128	16
Motorola		uPD27256	38
		uPD27C256	38
MCM2758	49	uPD27256A	18
MCM2716	44	uPD27C256A	18
MCM2532	45	uPD27C512	19
MCM68732-0	46		
MCM68732-1	47	OKI	
MCM68764	48		
MCM68766	48	MSM2758	00
		MSM2716	01
Mullard		MSM2732	02
		MSM2732A	03
27C256	18	MSM2532	42
		MSM2764	04
		MSM27128	06

* Using the S2030 Programming Adaptor

Device Part Number	Device Code	Device Part Number	Device Code
SEEQ		Texas Instruments	
2764	14,99	TMS2508	40
5133	14,99	TMS2516	41
27128	16,99	TMS2732	02
5143	16,99	TMS2732A	13
27256	18,99	TMS2532	42
27C256	18,99	TMS25L32	42
		TMS2764	14
52B13	70,99	TMS2564	43
52B13H	70,99	TMS27128	16
52B33	71,99	TMS27C128	17
52B33H	71,99	TMS27256	18,99
2816A	72,99	TMS27C256	18,99
2816AH	72,99		
2817A	73,99	Thomson EFCIS	
2817AH	73,99		
SGS		EF2516	41
		EF2532	42
		EF68764	48
M2716	01	Toshiba	
M2532	02		
M2732A	03		
M2764	14	TMM323	01
M2764A	15,99	TMM2732	02
M27128A	17,99	TMM2764	14
M27256	18,99	TMM2764A	15,99
M27512	19,99	TMM27128	16
		TMM27128A	17,99
Signetics		TMM27256	38,99
		TC57256	38,99
27C64	04	TMM27256A	18,99
27128	06	TC57256A	18,99
27C128	06	TMM27512	19,99
		TC571000	*10,99
Synertek		VLSI Technology	
SY2316	01		
SY2332	02	VT27C64	15,99
SY2333	02	VT27C128	17,99
SY2364	14	VT27C256	18,99
SY2365	14	VT27C512	19,99
SY23128	16		
		Xicor	
		X2816A	72
		X2864A	76

* Using the S2030 Programming Adaptor

Appendix 4 Device Codes for Single Chip Microcomputers

Device Part	Device Code	Adaptor Number	Comments
Advanced Micro Devices			
8751H	84	ST2021	No Security
8751H	85	ST2021	Security Bit Set
9761H	86	ST2021	No Security
9761H	87	ST2021	Security Bit Set
Fujitsu			
MBL8742N/H	96	ST2020	
MBL8749N/H	96	ST2020	
Hitachi			
HD63701V	18	ST2022	
HD63701X	03	ST2023	
HD63701Y	18	ST2025	
HD63705V	18	ST2024	
Intel			
8041A	91	ST2020	Read Only NOT Master
8042A	92	ST2020	Read Only NOT Master
8044H	80	ST2021	Read Only or Master
8048	91	ST2020	Read Only NOT Master
8048H	91	ST2020	Read Only NOT Master
8049H	92	ST2020	Read Only NOT Master
8050H	93	ST2020	Read Only NOT Master
8051	80	ST2021	Read Only or Master
8051H	81	ST2021	Read Only or Master
8052H	82	ST2021	Read Only or Master
8741A	94	ST2020	
8742	96	ST2020	
8744H	80	ST2021	No Security
8744H	81	ST2021	Security Bit Set
8748	94	ST2020	
8748H	95	ST2020	
8749H	96	ST2020	
8751	80	ST2021	No Security
8751H	80	ST2021	No Security
8751H	81	ST2021	Security Bit Set

Device Part	Device Code	Adaptor Number	Comments
NEC			
uPD8041A	91	ST2020	Read Only NOT Master
uPD8048	91	ST2020	Read Only NOT Master
uPD8049	92	ST2020	Read Only NOT Master
uPD8741A	94	ST2020	
uPD8748	94	ST2020	
uPD8748H	95	ST2020	
uPD8749H	96	ST2020	

Appendix 5 Programmer Service Return Form

Please complete this form and enclose a copy with the programmer.
 Return the programmer to :- STACK Ltd. Unit 8. Wedgwood Rd. Bicester.
 Oxon. OX6 7UL. Tel (0869) 240404.
 IMPORTANT: If the fault is associated with incorrect programming, then to aid the repair of the programmer, please enclose at least 2 of the devices which fail to programme.

SERVICE RETURN FORM				
Name:		Company:		
Address:				
.				
Post Code:		Tel: (. . . .).		Ex. . . .
Details of goods returned:		Date: . . ./ . . ./ . . .		
Model No:		Serial No:		
Mains cable Y / N		User Manual Y / N		
Devices TypeNo/Qty///				
Other				
Details of Fault:		(Delete where NOT appropriate).		
Programmer used with		(Computer, VDU, Stand-Alone).		
Devices programmed from		(Master, RAM).		
Device(s) programmed				
Device Manufacturer				
Device Code(s) used				
Fault Description:				
STACK use only: Service No:		Date / /		

Photocopy only, keep this copy as master.