Operating Manual
for the
P800
Production Programmer

Stag Programmers Ltd.,
Silver Court,
Watchmead,
Welwyn Garden City,
Herts AL7 1LT, UK.
Tel: +44 1707 332148
Fax: +44 1707 371503

sales@stag.co.uk
www.stag.co.uk

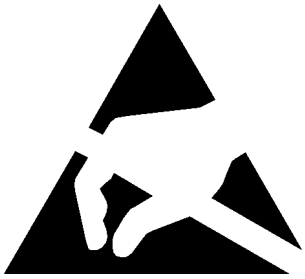
803 1182 Issue.2
CIS 3012

EMC

The P800 programmer complies with the Generic Emission standard EN50081-1 class B and the Generic Immunity standard EN50082-1. It meets the requirements of FCC Part15, subpart J class A.

Safety

This product is grounded through the grounding conductor of the power cord. NEVER operate this equipment with the grounding conductor disconnected. Replace the fuse only with a fuse of the specified voltage, current and type ratings.



Caution. Programmable devices are sensitive to Electro-Static Discharge. Observe ESD precautions when handling devices.

Contents

1 Introduction

The P800 Programmer is a multi socket production programmer, which is extremely simple to use. In local operation, all functions are accessed directly from the keypad in conjunction with menus and prompts displayed on the in-build LCD display.

1.1 Top Panel

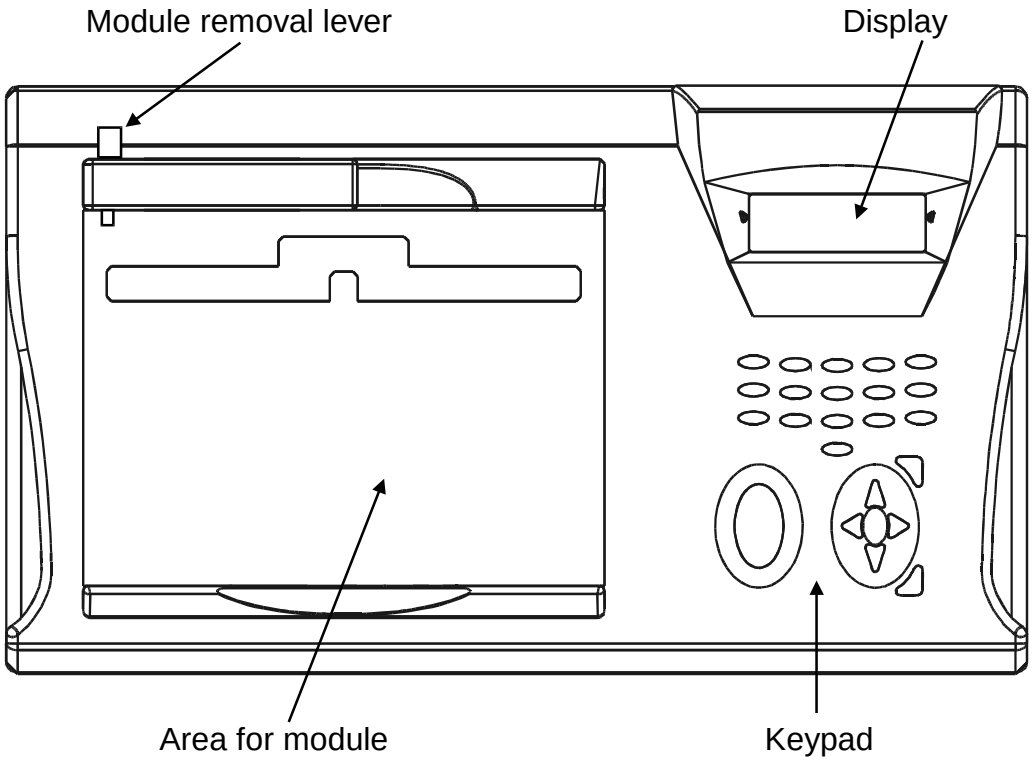


figure 1

1.2 Rear Panel

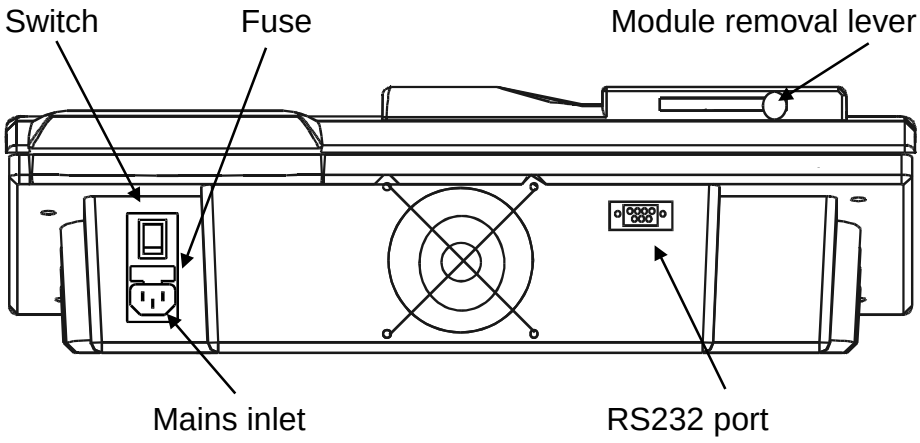


figure 2

1.3 Before powering up...

The mains voltage can be either nominally 115V or 230V without the need for voltage selection.
The power connector `block' houses the ON/OFF switch, the fuse 2A(T) (Slow blow) and the mains inlet - see figure 2.

! The unit should NOT be powered up or down with a device in any of the sockets.

Wait a minimum of 5 seconds after powering down, before powering back up.

1.4 Keypad

CRC	calculates a Cyclic Redundancy Check of data in RAM.
CSUM	calculates the checksum of data in RAM.
DATA	performs additional manipulation functions of data in RAM.
DEV	select a device.
EDIT	to manually edit data in RAM.
EMPTY	to perform an empty-check on a device or devices.
ENTER	to accept a mode or function setting.
EXIT	to exit from a mode or function.
INPUT	to input data from the serial or parallel port into RAM.
I/O	to set all input/output parameters.
LIMITS	to set the default limits for RAM and device data.
LOAD	to load data from a master device or devices into RAM.
MISC	to perform miscellaneous additional functions.
MODE	to set the bit-mode, e.g.: 8, 16 or 32, gang or set.
OUTPUT	to output data from RAM to the serial or parallel port.
PROGRAM PROG	} use either key to program data from RAM into devices.
SEQ	to set the programming sequence.
VERIFY	to compare data in RAM against data in a device or devices.
↑	to scroll data up the menu.
↓	to scroll data down the menu.
←	to move the cursor left or display previous option.
→	to move cursor right or display next option.

The keys labelled 0-9, A-F are also used to enter numeric data when this is required.

1.5 Modules

One module can be fitted at a time. Modules can either program 4 or 8 parts at a time. The 8 socket modules only support gang mode, but 4 socket modules can support different bit modes see section 2.3 (below).
When a part is selected that the module supports, the Busy LED will go out.

! If the BUSY LED is red, parts must not be inserted into the sockets.

To remove a module move the lever behind the module (see figure 1) towards the middle of the module and remove the module using the indents at the front and back of the module. Move the lever back to the left hand side.

To insert a module, ensure the module removal lever behind the module is to the left. Push module into place.

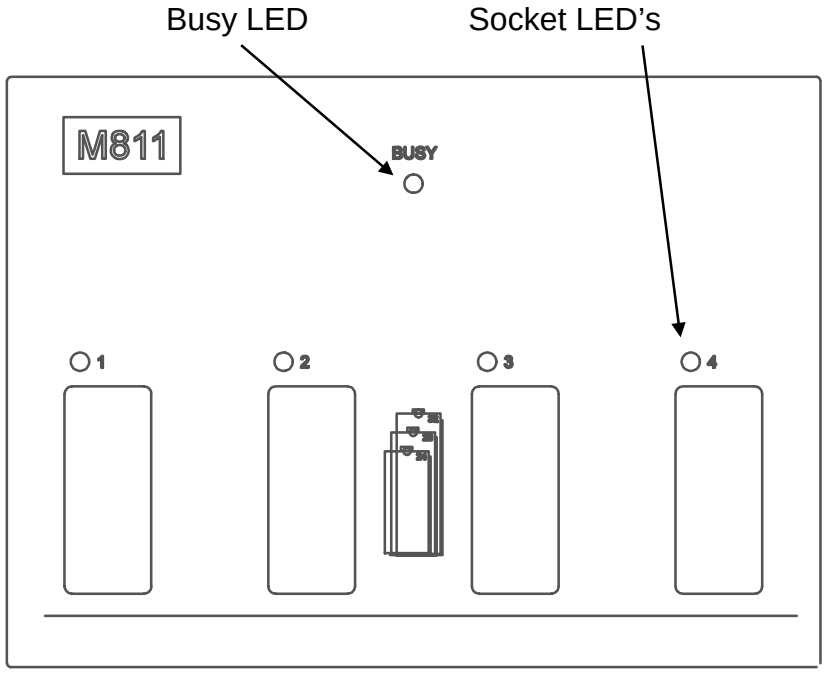


figure 3

The module can be inserted or removed while the unit is powered up, but should not be inserted or removed while the unit is performing a device function (e.g. load or program). This is indicated by the busy LED being green.

2 Local Operation

All functions are menu driven. Use the ↑ and ↓ keys to select the required option, then press ENTER.

The option which will be selected is always the 2nd row on the display. This is indicated by the arrows formed in the display surround.

To abort from the menu:

EXIT. (The pointing finger symbol denotes a dedicated function key press).

2.1 Device Selection

DEV

Once DEV has been pressed the device code may be entered directly using 0-9, A-F. To edit the code use the ← and → keys. When correct press ENTER.

Or

use the ↑ and ↓ keys to select the required manufacturer, then press ENTER. Now use the ← and → keys to select the required family or size of device, then the ↑ and ↓ keys for the exact device, finally press ENTER.

When selecting a new device, the bit mode may change, the device limits will be reset to default values, and the security setting will be reset to unprogrammed.

2.2 Device Functions

Device functions will first perform a reverse part check to ensure that the devices are the correct way round. If a part is faulty it may also fail this test. If this passes, a connect test is performed, to ensure that devices are present in the correct sockets.
See the marking on the module for correct orientation of the package.
DIL devices should be inserted towards the front of the ZIF with pin 1 towards the rear of the machine.

! **Before a device function is executed the user should ensure that all devices are the same as the device selected. (See also Section 2.4.3 Electronic Identifier)**

DIL devices should be inserted into the socket with the ZIF handle up; the handle should then be lowered. The handle should be raised before the part is removed.

! **If the BUSY LED is red, parts must not be inserted into the sockets on that module (see section 1.5 Modules)**

The BUSY LED will go green, during a device function.

! Devices or modules should not be removed, or added during a device function, while the BUSY LED is green.

At the end of the function the display will indicate if the function passed or failed. All devices that pass will have a green LED lit adjacent to them; all devices that fail will have a red LED lit adjacent to them.
If a device fails, the red LED will light as soon as the failure is detected but the programmer will continue until the function is completed, or all devices fail.
If a device is incorrectly placed in a socket, the P800 may not be able to detect its presence, so the adjacent LED will be turned off. The isolation will ensure that the other parts program correctly, but it is possible that the part in the adjacent socket may not be programmed, this would be indicated by a red LED in that position.

If a device function is selected, when no, or an incorrect module is connected, then the function will abort with the message
'No or incorrect Module Requires: xxxx'
where xxxx is the name of the required module.

2.2.1 Loading the P800's RAM from Master Devices

Master devices should be placed in the sockets starting at socket 1 on the left hand side module.
The number of parts required depends on the mode selected (see Section 2.3).

LOAD

On pressing LOAD the data in the device will be copied into the P800's internal RAM.
Remove master devices from the sockets after loading.

2.2.2 Verify

This function compares the contents of RAM with the data in the device(s). Some devices will be verified twice at different VCC values as directed by the manufacturer's specification. This will also happen if marginal verify is selected (see Section 2.4.2).
Ensure devices to be checked are inserted

VERIFY

2.2.3 Empty

This function will check that the devices are unprogrammed. If an electrically erasable part is selected, the part can be erased during programming, so new devices may not be shipped in their empty state.
Ensure devices to be checked are inserted

EMPTY

2.2.4 Program

This function initiates the automatic programming sequence. The device is first checked with the pre-program check & erased if necessary (see Section 2.4.1), programmed with the data in the RAM to the manufacturer's specification, verified, then the security fuses may be blown if applicable (see Section 2.4.4).
Ensure devices to be programmed are inserted

PROGRAM or PROG

Note there is no difference between pressing the PROG or PROGRAM buttons either can be used.

2.3 Bit modes

The 8 Socket modules (named G8xx), are gang only (i.e. all sockets programmed with the same data). 4 socket modules support different bit modes, as listed below:

For 8 bit parts

Mode	Description
8 bit gang	4 sets of 1 part
8 bit 2of2	2 sets of 2 parts
8 bit 1of4	1 set of 4 parts
16 bit gang	2 sets of 2 parts with 16 bit wide data
16 bit 1of4	1 set of 4 parts with 16 bit wide data
32 bit gang	1 set of 4 parts with 32 bit wide data

For 16 bits parts

Mode	Description
16 bit gang	4 sets of 1 part
16 bit 2of2	2 sets of 2 parts
16 bit 1of4	1 set of 4 parts
32 bit gang	2 sets of 2 parts with 32 bit wide data

The bit mode is used in all device functions (e.g. Load or Program), and can also be used when calculating the checksum and CRC.

MODE then select required mode.
Note that if the device is 16 bits wide then two bytes of RAM are required to store each device word. This can be done either high byte first/low byte last (the default), or else low byte first/high byte last. Having selected the bit mode as detailed below you will then be asked to specify the byte order. To do this use the ← and → keys to make the selection, then press ENTER when ready.

The mode may be changed on power up, or when a new device is selected, for example if the mode requires more RAM than the P800 has. Not all modes are available for all parts.

2.3.1 8-bit Gang

In 8 bit GANG mode all sockets map to the same RAM image. Device data is stored in one contiguous region of RAM. Loading is done from socket 1.

2.3.2 8-bit 2of2 (2 sets of 2 parts)

In 2of2 set mode data in the RAM is stored in two blocks, one block for sockets 1 and 3, followed by one block for sockets 2 and 4. The block size is the size of the device selected.
- Sockets 1 and 3 will program/verify with data starting at RAM START.
- Sockets 2 and4 will program/verify with data starting at (RAM START + device size).

Loading will over write any existing data in the RAM for all sockets. Devices are required in sockets 1 and 2.

2.3.3 8-bit 1of4 (1 set of 4 parts)

In 1of4 set mode data in the RAM is stored in four blocks, one block for each socket. The block size is the size of the device selected.
- Socket 1 will program/verify with data starting at RAM START.
- Socket 2 will program/verify with data starting at (RAM START + device size).
- Socket 3 will program/verify with data starting at (RAM START + (device size x 2)).
- Socket 4 will program/verify with data starting at (RAM START + (device size x 3)).

Loading will over write any existing data in the RAM for all sockets. 3 or 4 devices may be present, as long as there are devices in sockets 1, 2 and 3.

2.3.4 16-bit Gang

In 16-bit GANG mode groups of 16 bit data may be programmed.
For an 8-bit wide device the mapping will be as follows:

Socket	1	2	3	4
Image	A	B	A	B

The data for image A will be stored at even locations in RAM, while the data for image B will be stored at odd locations in RAM. Loading is done sockets 1 and 2 only.

In the case of 16-bit wide devices all 8 sockets will program with the same data. Two bytes of RAM are required for each device word, and the order in which these are stored is (by default) high byte first, low byte last. This default may be modified by the user (see section 2.3 above). Loading is done from socket 1.

2.3.5 16-bit 2of2 (2 sets of 2 parts)

This mode is available for 16 bit wide parts only.
In 2of2 set mode data in the RAM is stored in two blocks, one block for sockets 1 and 3, followed by one block for sockets 2 and 4. The block size is the size of the device selected.
- Sockets 1 and 3 will program/verify with data starting at RAM START.
- Sockets 2 and4 will program/verify with data starting at (RAM START + device size).

Loading will over write any existing data in the RAM for all sockets. Devices are required in sockets 1 and 2.

2.3.6 16-bit 1of4 (1 set of 4 parts)

For 8-bit wide devices data in the RAM is stored in two blocks, one block for the socket pair 1 and 2, with the second block for the socket pair 3 and 4. The block size is twice the size of the device selected.
- Socket 1 will program/verify with data starting at RAM START.
- Socket 2 will program/verify with data starting at (RAM START + 1).
- Socket 3 will program/verify with data starting at (RAM START + (device size x 2)).
- Socket 4 will program/verify with data starting at (RAM START + (device size x 2) +1).

Loading requires 4 devices, one in all 4 sockets.

For 16 bit parts data in the RAM is stored in four blocks, one block for each socket. The block size is the size of the device selected.
- Socket 1 will program/verify with data starting at RAM START.
- Socket 2 will program/verify with data starting at (RAM START + device size).
- Socket 3 will program/verify with data starting at (RAM START + (device size x 2)).
- Socket 4 will program/verify with data starting at (RAM START + (device size x 3)).

Loading will over write any existing data in the RAM for all sockets. 3 or 4 devices may be present, as long as there are devices in sockets 1, 2 and 3.

2.3.7 32-bit Gang

In 32 BIT GANG mode, groups of 32 bit data may be programmed.
For an 8-bit wide device the mapping will be as follows:

Socket	1	2	3	4
Image	A	B	C	D

The data for image A will be stored at addresses ending 0, 4, 8 and C (hex) the data for image B will be stored at addresses ending 1, 5, 9 and D (hex), the data for image C will be

stored at addresses ending 2, 6, A and E (hex) and the data for image D will be stored at addresses ending 3, 7, B and F (hex).
Loading will over write any existing data in the RAM for all sockets. 4 devices must be present.

For a 16-bit wide device the mapping will be as follows:

Socket	1	2	3	4
Image	A	B	A	B

The data for image A will be stored at addresses ending 0-1, 4-5, 8-9, C-D (hex), the data for image B will be stored at addresses ending 2-3, 6-7, A-B, E-F (hex). Note that two bytes of RAM are required for each device word. The order in which these are stored is (by default) high byte first, low byte last. This default may be modified by the user (see section 2.3 above).
Loading is done from the sockets 1 and 2 only.

2.4 Programming Sequence

This allows the user to define what functions are performed when a device operation is required.

Press SEQ

Sub-menus are selected using the ↑ and ↓ keys, then pressing ENTER.

2.4.1 Pre-program checks

SEQ then select PRE-PROGRAM

Before a device is programmed, the device can be automatically checked with either an empty check or an illegal bit check or neither.
The empty check checks that the device (within the specified device limits) is empty.
The Illegal bit check checks that the device has no bits which are programmed and required to be empty by the RAM data.
Select using the ↑ and ↓ keys, then press ENTER.

On a electrically erasable device (e.g. EEPROM, Flash), the illegal bit check is used to unlock the device, where applicable.
On parts that erase in blocks (e.g. Flash), the illegal bit check is used to check if erase is necessary, and then erase the part.

! on EEPROM and Flash devices, the illegal bit check should be enabled, unless the parts are guaranteed empty, and unlocked.

2.4.2 Marginal verify testing

SEQ then select MARGINAL TESTING

After programming, during illegal bit test, and when the VERIFY key is pressed the device is verified with the RAM. This can either be done at the manufacturer's recommended VCC voltages (Marginal verify disabled), or at +/- 10% of the nominal VCC voltage (Marginal verify enabled).
Note: The lower marginal voltage is also used during empty testing and illegal bit testing.

Select the required option using ← and → , then press ENTER.

2.4.3 Electronic Identifier

SEQ then select ELECTRONIC ID

An Electronic Identifier exists in some EPROM and Flash devices. It can be used to check or select a device before load/verify/empty check or program.
Three options are given: check, automatic, none.

NONE	will not check the electronic identifier unless required by the programming algorithm.
CHECK	will check that the device(s) in the socket(s) are the same as that selected. If not, the error message WRONG PART will be displayed. NO SIG will be displayed if no signature can be read from the device.
AUTOMATIC	will read the identifier and try to select the correct device code to match. It can only select devices of the same family as that already selected. If different family devices are inserted then the error message MISMATCHED PARTS will be displayed.

Select the required option using the ↑ and ↓ keys, then press ENTER.

2.4.4 Security fuses

SEQ then select SECURITY

Some devices have one or more security fuses, which can either secure the part, or disable writing to it. The user can select to program them or leave them intact using the ← and → keys followed by ENTER. With devices that have more than one security fuse they can be selected using the ↑ and ↓ keys to display the other fuses, ENTER is then pressed once to enter all the fuses.
On some EEPROMs and Flash the security feature can be used to make the device, or blocks of the device write protected.

To find out more details of what each fuse does, for a particular part, see the back of the device support list.
The security setting is reset to not secure when the P800 is powered up, or a new device is selected, except when selected via AUTOmatic electronic ID.

2.4.5 Display Checksums

SEQ then select DISPLAY CHECKSUMS

After programming, verifying, and loading, the RAM checksums may be displayed.
Note: The checksums displayed are of the RAM not the device, and so should not be used to check if a part verified, or programmed correctly.

Select the required option using ← and → , then press ENTER.

After a device function, the checksums for sockets 5, 6, 7, 8 can be displayed by pressing →.

2.5 Device Limits

All device functions (e.g. Load or Program) have 3 associated parameters:

DEV START	the device address from which the function should start;
DEV STOP	the device address at which the function should stop;
RAM START	the RAM address from which the function should start;

These are also used when calculating the checksum and CRC and can be altered by the user.

LIMITS

Enter the addresses (in Hexadecimal) using 0-9, A-F.
The cursor can be moved using the ↑ and ↓ keys.
When correct press ENTER.

If invalid addresses are chosen (e.g. DEV START higher than DEV STOP) the ENTER key will not let the user out of the function. If the EXIT key is pressed the limits will not be changed.
This function can be disabled: see section 2.9.3 for details.

! **The default limits for a device will be used when a new device, or new mode is selected.**

2.6 RAM Functions

2.6.1 Listing and changing the RAM

EDIT

The editor displays 4 addresses, one on each line. Each is in the following format:
aaaaaaaa hh ddd c
where:

aaaaaaaa	is the RAM address in hex;
hh	is the hex value stored at that location;
ddd	is the decimal value stored at the same location;
c	is the ASCII for that byte if printable (if not, a SQUARE character is displayed).

The address can be changed using 0-9, A-F and by moving the cursor using ← and →. To edit the data move the cursor right to the hex or decimal data fields, then overwrite the data. To edit the next or previous byte use ↑ or ↓.
When complete press ENTER then EXIT.
Data can be listed by changing the address as above and then pressing ENTER, or by using the ↑ and ↓ to view the previous or next location. Press EXIT when finished.
This function can be disabled: see section 2.9.3 for details.

2.6.2 RAM data manipulation

These functions can be disabled: see section 2.9.3 for details.
The following functions can be performed on the P800's internal RAM:

- FILL RAM
- BLOCK MOVE
- INSERT BYTES
- DELETE BYTES
- COMPLEMENT RAM
- STRING SEARCH

DATA

Select the function required using the ↑ and ↓ keys, then press ENTER.

2.6.3 Fill the RAM

This function allows you to fill the contents of RAM between selected limits with a selected bit pattern.

DATA then select FILL RAM

On selecting `FILL RAM' the following options are available:

Fill with Zeros	(fill the RAM with 00 hex)
Fill with Ones	(fill the RAM with FF hex)
Fill with Empty	(fill the RAM with the empty state of the selected device)
Fill with Pattern	(fill the RAM with a user defined pattern)

Select the option required using the ↑ and ↓ keys, then press ENTER.
If `Fill with Pattern' is selected the desired pattern should be entered in hex using 0-9, A-F. The ← and → keys may be used to move the cursor to edit the pattern. The ASCII value of the hex numbers is displayed underneath (if a printable value is entered).
When correct press ENTER.

! Note that patterns are only considered legal if they are 1, 2 or 4 hex characters long.

All the options will then ask for the address range over which the fill is to take place. The options are:

ARBITRARY LIMITS
ENTIRE MEMORY
DEVICE LIMITS

Select the option required using the ↑ and ↓ keys, then press ENTER.

The three functions available are as follows:

ENTIRE MEMORY:	This function fills the entire RAM with the specified pattern.
DEVICE LIMITS:	This will only fill the RAM used for the selected part, taking account of the selected device limits (see Section 2.6) and mode (see Section 2.3). Note that if the device address limits are set to only partially cover the device then this function may actually fill non-contiguous regions of RAM.
ARBITRARY LIMITS:	This function will enable the user to fill RAM between entirely arbitrary RAM limits. On selecting this option the address limits should be entered in hex using 0-9, A-F and ↑, ↓, ← and → to move the cursor as required. When correct press ENTER.

2.6.4 Move a block of data

DATA then select BLOCK MOVE

This function allows data to be moved from one section of RAM to another.
There are no restrictions on the positioning of either the source block or the destination block, other than that they must both fit within the physical available RAM. Source and destination blocks may even overlap, should this be required.
On selecting `BLOCK MOVE' the RAM address of BLOCK START, BLOCK END, and DESTINATION should be entered in hex using 0-9, A-F and ↑↓ ← and → to move the cursor as required. When correct press ENTER.

2.6.5 Inserting bytes into RAM

DATA then select INSERT BYTES

This function allows a pattern of bytes to be inserted into RAM at a specific location.
All data at or beyond (i.e. at higher addresses than) the insertion address will be moved upward in memory by the number of bytes inserted. No data bytes are overwritten at the insertion position - instead they move up to make room for the new data.

! Note that as a result of this operation, the very last byte(s) in memory will be lost.

First enter the address in RAM to insert the first byte, use 0-9, A-F, and ↑ ↓ ← and → to move the cursor. When correct press ENTER.
Then the desired pattern should be entered in hex using 0-9, A-F. The ← and → keys may be used to move the cursor to edit the pattern. The ASCII value of the hex numbers is displayed underneath (if a printable value is entered). Up to 32 characters may be entered.
When correct press ENTER.

2.6.6 Deleting bytes from RAM

DATA then select DELETE BYTES

This function allows a number of bytes to be deleted from RAM. All data at or beyond (i.e. at higher addresses than) the deletion address will be moved down in memory by the number of bytes specified.
Enter the address in RAM to delete the first byte and the number of bytes to be deleted (in hex), use 0-9, A-F, and ↑ ↓ ← and → to move the cursor. When correct press ENTER.

2.6.7 Complementing the RAM

DATA then select COMPLEMENT RAM

This function allows the data in RAM to be complemented between selected limits. This means that every binary 1 in the RAM data is changed to a binary 0, and vice versa.
The address range over which the complement is to take place should then be selected. The options are:

ARBITRARY LIMITS
ENTIRE MEMORY
DEVICE LIMITS

Select the option required using the ↑ and ↓ keys, the press ENTER.
The three functions available are as follows:

ENTIRE MEMORY:	This function changes the entire RAM with the specified pattern.

DEVICE LIMITS:	This will only change the RAM used for the selected part, taking account of the selected device limits (see Section 2.6) and mode (see Section 2.3). Note that if the device address limits are set to only partially cover the device then this function may actually fill non-contiguous regions of RAM.
ARBITRARY LIMITS:	This function will enable the user to change the RAM between entirely arbitrary RAM limits. On selecting this option the address limits should be entered in hex using 0-9, A-F and ↑, ↓, ← and → to move the cursor as required. When correct press ENTER.

2.6.8 Search the RAM for a data sequence (STRING SEARCH)

This function allows you to search for a string of bytes within specified RAM limits.

DATA then select STRING SEARCH

The desired pattern should be entered in hex using 0-9, A-F. The ← and → keys may be used to move the cursor to edit the pattern. The ASCII values of the hex numbers are displayed underneath (if printable values are entered). Up to 32 characters may be entered. When correct press ENTER.
The address range over which the search is to take place should then be selected. The options are:

ARBITRARY LIMITS
ENTIRE MEMORY
DEVICE LIMITS

Select the option required using the ↑ ↓ keys, then press ENTER.

The three functions available are as follows:

ENTIRE MEMORY:	This function searches the entire RAM with the specified pattern.
DEVICE LIMITS:	This will only search the RAM used for the selected part, taking account of the selected device limits (see Section 2.6) and mode (see Section 2.3). Note that if the device address limits are set to only partially cover the device then this function may actually fill non-contiguous regions of RAM.
ARBITRARY LIMITS:	This function will enable the user to search the RAM between entirely arbitrary RAM limits. On selecting this option the address limits should be entered in hex using 0-9, A-F and ↑, ↓, ← and → to move the cursor as required. When correct press ENTER.

2.6.9 Checksum of RAM data

CSUM

This function will display the checksum of the whole RAM, the device(s)' limits of RAM, or arbitrary limits defined by the user. Select the option required. If the device limits are chosen, then a checksum of each sockets 1 to 4 will be displayed, calculated according to the current bit mode. On a gang only module, only 1 checksum will be displayed, as all the sockets are the same.

2.6.10 Cyclic Redundancy Check of RAM data

CRC

Cyclic Redundancy Check provides a better representation of the RAM data than a checksum as it takes account of the order of the data. The format is the same as checksum.

2.7 Setting up the I/O

The I/O ports can be used to input and output data to or from the P800's internal RAM. They can also be used to remotely control the unit.

2.7.1 Data Transfer Formats

This function enables the user to select the data format for input and output.

I/O then select FORMAT.

A list of available I/O formats is displayed. These can be scrolled up and down using the ↑ and ↓ keys. Typically, you might have a choice between: STAG HEX, BINARY, STAG BINARY, ASCII HEX SPACE, INTEL 16 BIT, INTEL 32 BIT, MOTOROLA S-REC. The actual list of formats available will vary, depending on the device currently selected.
Select the required format using the ↑ and ↓ keys, then press ENTER.

2.7.2 Selecting and setting up the port.

I/O then select PORT

A list of parameters is displayed. These can be scrolled up and down using the ↑ and ↓ keys. The option may be changed using the ← and → keys. When the whole menu is set-up as required press ENTER.

SPEED: The speed of the RS232 port may be set to:
1200, 2400, 3600, 4800, 9600, 19K2, 38K4 or 115K2 baud.

STOP BITS: The number of stop bits transmitted after each byte on the RS232 port may be set to 1 or 2.

2.7.3 Beeper Control

After each function the beeper will sound to indicate pass or fail (2 beeps for pass, 5 beeps for fail). This function may be disabled or enabled.

I/O then select BLEEP

Select disabled or enabled using the ← and → keys followed by ENTER.
You can also have the beeper sound for each key press. Select disabled or enabled using the ← and → keys followed by ENTER.

2.7.4 Entering Remote Control

To put the unit into remote control:

I/O then select REMOTE CONTROL then press

I/O.

To quit from remote back into local mode, power down the unit, then power up with the EXIT key pressed. The P800 will beep twice to indicate the P800 is returning to local operation.
See also Section 3 - Remote Control.

2.8 Transferring DATA via the ports

Before loading or outputting data via the ports, it is first necessary to ensure the following:
the correct interface format is selected (see Section 2.7.1);
and that the correct port set-up is selected (see Section 2.7.2).
When using RS232 the interface must be used with either hardware handshaking or the Xon/Xoff protocol.

2.8.1 Receiving data from the port

INPUT

On pressing INPUT, three further options may be entered OFFSET, RAM START and RAM STOP.
They are used to define where in RAM to store the data. The OFFSET value is subtracted from the address of the incoming data and the RAM ADDRESS is added on. Data beyond the RAM STOP address will be truncated.
A rotating wheel is displayed to indicate that data is being received.

2.8.2 Transmitting data to the RS232 port

OUTPUT

On pressing OUTPUT, three further options may be entered OFFSET, RAM START and RAM STOP.
OFFSET is used to generate the first transmitted address.
RAM START gives the location to find the first byte of data, then the transmitted address and the RAM address are incremented until the RAM address equals RAM STOP.
A rotating wheel is displayed to indicate that data are being transmitted.

2.9 Miscellaneous Set-ups and Functions

2.9.1 Machine's statistics

MISC then select STATISTICS

This function will show the following information:

Flash software revision/ Boot ROM revision.
The RAM size (in bytes);
The flash size (in bytes);

2.9.2 Updating the software

To update the software, first load the software from STAG into the unit's RAM. This can be done using either the I/O ports or by loading master devices.

MISC then select UPDATE

P800 will then check that the data in the RAM has the correct format and CRC. The Flash will then be updated. When complete the unit will re-start itself, as if just powered up.

2.9.3 Ram Lock

MISC then select RAM LOCK

The following functions can be locked:

EDIT
DATA
LIMITS

When locked, any of these function, when pressed, will display FUNCTION DISABLED. This will help prevent inexperienced operators from changing the RAM contents. The RAM can still be changed using, for example, Load, or Input.

Select the Lock RAM Functions, or Enable RAM Functions using ← and → , then press ENTER. If the status is to be changed, then the security code is asked for. This can be any 6 digit hexadecimal number. Units are shipped with a code of 000000. The ram lock status will only be changed if the code is correct.

2.9.4 Ram Lock Code

MISC then select RAM LOCK CODE

This function allows the Ram Lock Code to be changed. First the user will be asked for the current security code, if this is entered correctly, then the new code can be entered. The P800 will then ask you to repeat the code, to ensure it was entered correctly. If it was then the new code will be set.

It is the users responsibility to ensure the code is not forgotten

3 Remote Operation of P800

P800 may be controlled remotely through the RS232 port.
The unit is put into remote mode by a key sequence in local mode (see Section 2.7.4). On power down the mode of operation is remembered so it will power back up still in remote, unless the self test fails.
To return to local, either issue the Z command or power up with the EXIT key pressed.

See also StagCom and/or Production Manager.

3.1 Remote Control Commands

Remote control commands are case insensitive, and so may be transmitted in either upper, lower or mixed case. Spaces and tabs are ignored (The only exception to these rules is the remote control `D' command).
In the following table, anything printed in UPPER CASE should be sent literally, while anything in lower case represents a parameter which you should substitute with an appropriate value. Some of the commands cause P800 to transmit information back to the host, others do not.
P800's response is followed immediately by a carriage-return, linefeed, status-code (see Section 3.2), carriage-return, linefeed, prompt (a greater-than symbol).

S0 manufacturer device	Set the programmer for the specified manufacturer and device. Each of the parameters consists of exactly three hexadecimal characters.
S1 format	Set the I/O format. The parameter is a single ASCII character, and may be one of the following: 4 (Intel hex); 5 (Motorola S-Record); 8 (Stag- hex); 9 (ASCII-hex-space); A (Stag Binary); D (Binary); I (Intel 32). Note that many of these formats may be illegal for a given device. Additional formats may be added to this list by Stag at a later date.
S3 security	Set the security flags. The parameter is a hexadecimal number between 0 and FFFFFFFF. Each bit corresponds to one security bit for the currently selected device: bit 0 corresponds to fuse 0, through to bit 31 corresponding to fuse 31. Note that not all devices support security, and the number of security fuses varies with the selected device.
S4	Fill the RAM between RAM-START and RAM-STOP with the device's unprogrammed state.
SM bitmode	Set the bit-mode. For 8-bit wide devices, legal combinations are: 010408 (8 bit gang); 020208 (8 bit 2of2), 040108 (8 bit 1of4), 010216 (16 bit gang), 020116 (16 bit 1of4), 010132 (32 bit gang).

	For 16-bit wide devices, legal combinations are: 010416 (16 bit gang); 020216 (16 bit 2of2), 040116 (16 bit 1of4), 010232 (32 bit gang).
SR ram_start	Set the RAM-START address to the specified hexadecimal value. Note that this operation is not carried out immediately, but instead is deferred until after the SE command is issued - therefore you MUST supply these commands in the order: SR followed by SE. It is legal to omit the ram_start parameter if you do not wish to modify it but intend to use SE.
SE ram_stop	Set the RAM-STOP address to the specified hexadecimal value. Note that this command also makes permanent the RAM-START address specified by a previous SR command, therefore you MUST supply these commands in the order SR followed by SE. It is legal to omit the ram_stop parameter if you do not wish to modify it but intend to use SR.
SD device_start	Set the DEVICE-START address to the specified hexadecimal value. Note that this command also sets the DEVICE-STOP address to:(DEVICE-START + (RAM-range / bit-mode-width), so you MUST set the RAM-START, the RAM-STOP and the bit mode BEFORE using this command. It is legal to omit the device_start parameter if you do not wish to modify it but need to modify the device stop address (having previously modified the RAM range).
SQ ram_start	Set the destination RAM-START address to the specified hexadecimal value. This is only used by the BM command, and should be set before the BM command is executed.
SO offset	Set the I/O offset to the specified hexadecimal value.
SB parity	Set the least significant byte (16-bit EPROMs or micros) to odd or even RAM addresses. The single parameter may be ASCII 'O' (odd) or 'E' (even).
SS socket	Set the socket number to which checksum and CRC will refer.
ST margin_mode	Set marginal testing to on (1) or off (0).
SY eid_mode	Set the electronic identifier mode to OFF (0), CHECK (1) or AUTOMATIC (2).
PPS	Set Pre Program checks. 0=No test, 1=Bit check, 2=Empty check.
R0	Read the manufacturer and device code. This command outputs a six character hexadecimal number consisting of the Stag manufacturer and device codes for the currently selected device.
R1	Read the interface format. This command outputs a single ASCII character representing the currently selected I/O format, which will correspond to one of the options available for the S1 command.

R3	Read the security fuse setting. The response is an 8 digit hexadecimal number. Each binary bit corresponds to one security bit for the currently selected device: bit 0 corresponds to fuse 0, through to bit 31 corresponding to fuse 31. Note that not all devices support security, and the number of security fuses varies with the selected device.
R4	Reads the CRC. This command outputs a four character hexadecimal number corresponding to the currently selected socket (see SS and RS).
R5	Read the RAM size. The output is a six character hexadecimal number representing the topmost RAM address available.
R6	Read the Flash software revision number. The output consists of the ASCII string "136-" (which identifies this product as the P800) followed by the flash revision number, followed by a '/', then the boot ROM revision.
R7	Read the checksum. This command outputs a four character hexadecimal number, corresponding to the currently selected socket (see SS and RS).
R9	Read device description. Output consists of three fields separated by a "/" character. The first field is the maximum possible (hexadecimal) device address; the second field is the (decimal) device width measured in bits; and the last field is the empty state of the device - ('0' meaning all zeros; '1' meaning all ones; and '2' meaning unknown or indeterminate).
RM	Read the current bit mode. Output consists of six digits. For 8-bit wide devices, legal combinations are: 010408 (8 bit gang); 020208 (8 bit 2of2), 040108 (8 bit 1of4), 010216 (16 bit gang), 020116 (16 bit 1of4), 010132 (32 bit gang). For 16-bit wide devices, legal combinations are: 010416 (16 bit gang); 020216 (16 bit 2of2), 040116 (16 bit 1of4), 010232 (32 bit gang).
RR	Read the current RAM-START address. Output consists of six hexadecimal characters.
RE	Read the current RAM-STOP address. Output consists of six hexadecimal characters.
RD	Read the current DEVICE-START address. Output consists of six hexadecimal characters.
RO	Read the current I/O offset. Output consists of eight hexadecimal characters.
RQ	Read the current destination RAM-START address. Output consists of six hexadecimal characters.
RB	Read the current odd/even setting for 16-bit wide devices. Returns ASCII 'O' for least significant byte odd, or 'E' for least significant byte even.

RS	Read the currently selected socket number.
RT	Read the current marginal-verify setting. Output is ASCII '0' for off, or '1' for on.
RY	Read current electronic identifier setting. Output is ASCII '0' for Off, '1' for Check or '2' for Automatic.
RF	Read failing sockets. After a device operation you can find out which sockets failed using this command. The output is a hexadecimal number consisting of two characters. Bit zero corresponds to socket one, and so on through to bit seven which corresponds to socket eight. Each bit will be set if the part in that socket failed.
RG	Read active sockets. After a device operation you can find out which sockets were had parts present using this command. The output is a hexadecimal number consisting of two characters. Bit zero corresponds to socket one, and so on through to bit seven which corresponds to socket eight. Each bit will be set if the corresponding socket had a part present.
RA	Read module codes. Output is a six character hexadecimal number. The first 2 digits will be 02, followed by two digits representing the left module code currently fitted. The last 2 digits represent the right socket module currently fitted If no module is fitted then the output will be 'FF'.
RP	Read Flash PROM size. Output is a six character hexadecimal number representing the amount of Flash PROM currently installed in your machine.
RH	Returns the device type/package style in the format aa/bbbb, where aa is the device type EPROM=1, EEPROM=2, PROM=3, MICRO=4.
PPR	Read Pre Program setting. 0=No test, 1=Bit check, 2=Empty check.
P0	Program devices with pre-program illegal bit check.
P1	Program devices with no pre-program check.
L	Load devices.
E	Empty check devices.
V	Verify devices.
I	Input data from the currently selected port using the currently selected I/O format into the P800's RAM, using the currently selected RAM-START, RAM-STOP and I/O-OFFSET settings.
O	Output data from the P800's RAM to the currently selected port using the currently selected I/O format, and using the currently selected RAM-START, RAM-STOP and I/O-OFFSET settings.
F pattern	Fill RAM between the current RAM-START and RAM-STOP limits using the specified pattern. The pattern must consist of either 2, 4 or 8 hexadecimal characters.

BM	Block move between previously defined limits
H number	Sound the horn the specified number of times. The single parameter should be a decimal number between 1 and 20.
Dstring	Display a string on P800's LCD. This is the only command for which spaces and tabs are not ignored, and for which, case is important. There should be no space between the D and the first character of the string. The string may not contain linefeeds or carriage-returns - it may however contain the following commands: 0x0C — Clear Screen 0x1B[y;zH — Move Cursor to row y (value 0 to 3) column z (value 0 to 39) 0x1B[J — Erase current line 0x1B[0~ — Turn on cursor 0x1B[1~ — Turn off cursor where 0x0C, 0x1B are hex values.
K	Wait for any key on the P800's keypad to be pressed.
Z	Exit remote control mode.

3.2 Status Codes

Status codes returned by the P800 consist of two hexadecimal characters.
The following responses may be obtained:

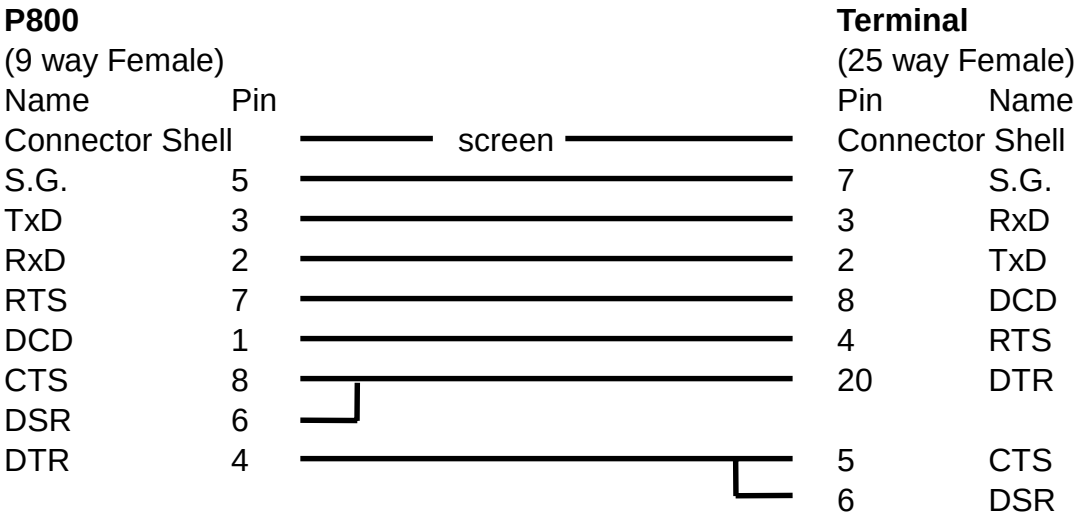
00 Command executed successfully.
01 Device failed to program.
02 Device failed to verify.
03 Device failed illegal bit test.
04 Device failed empty test.
05 Device failed connect test.
06 Device reversed or faulty.
08 Warning: Command executed successfully, but something minor went wrong.
09 Security bit(s) failed to program.
0A Illegal or unrecognised command.
0B Error in inputting data.
0D Ram failure.
10 Wrong module
11 Wrong Part: Electronic Identifier check failed.
12 Different parts in different sockets
13 Illegal or out of range address.
15 No Signature: Electronic Identifier check failed.
1C Out of memory.
1E Function aborted.
1F Command syntax error.
20 Wrong Bit mode

4 Pinouts for port connector

4.1.1 The RS232 connector pinout

Pin	Name	Function
3	TxD	Send data, P800 transmits data on this pin.
2	RxD	Receive data, P800 receives data on this pin.
7	RTS	Driven high by P800 when it wants to transmit.
8	CTS	Must be high or open to permit P800 to send.
6	DSR	Must be high or open to permit P800 to send.
5	GND	Signal ground.
1	DCD	Sent low by P800 when not ready to receive.
4	DTR	Sent low by P800 when not ready to receive.

4.1.2 Typical RS232 Cable Configuration



Note the cable should be fully shielded and incorporate a ferrite at the P800 end of the cable.

Note: For systems using Xon/Xoff protocol a three wire connection is sufficient. Resistors in the P800 will hold handshake lines in the correct polarity. A 3 wire connection may not work reliably with Windows / Windows95 applications, including SCOM WIN, SCOM WIN95.

A1 Increasing the RAM Size of P800

The RAM fitted to P800 can be increased in size from 4Mbyte to 16Mbyte as detailed below:

! Power down the P800 and disconnect from supply before proceeding with this operation

Remove the 6 fixing screws from the rear panel- see figure 4 for location of screws.

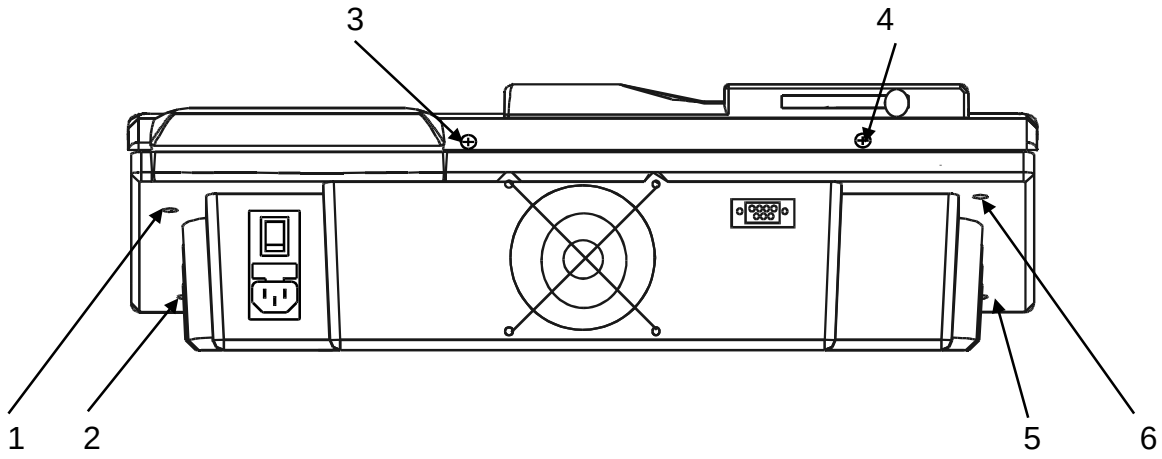


figure 4

Separate the base from the top cover. Disconnect the fan cable and the power cable from the main P.C.B..-see figure 5 for location of connectors

Remove the SIMM and replace with a bigger size.
All SIMMs should be of a Stag recommended type.

! SIMMs are static sensitive - observe suitable handling precautions

To remove existing SIMM, lever clips out sideways and tilt SIMM backwards to disengage. Pull SIMM upwards to remove from connector slot.
To insert a new SIMM, insert into connector slot, ensuring that the cut out corner is at the left end. Tip forwards and ensure both side clips fully engage (see figure 5).
Replace fan and power cable connectors. Click the top and bottom mouldings together and replace the 3 screws to fasten top cover to base. Connect to power supply and power-up.
P800 will recognise the increase in RAM size and perform a full RAM check. Subsequent power-ups will perform the standard self-test sequence.

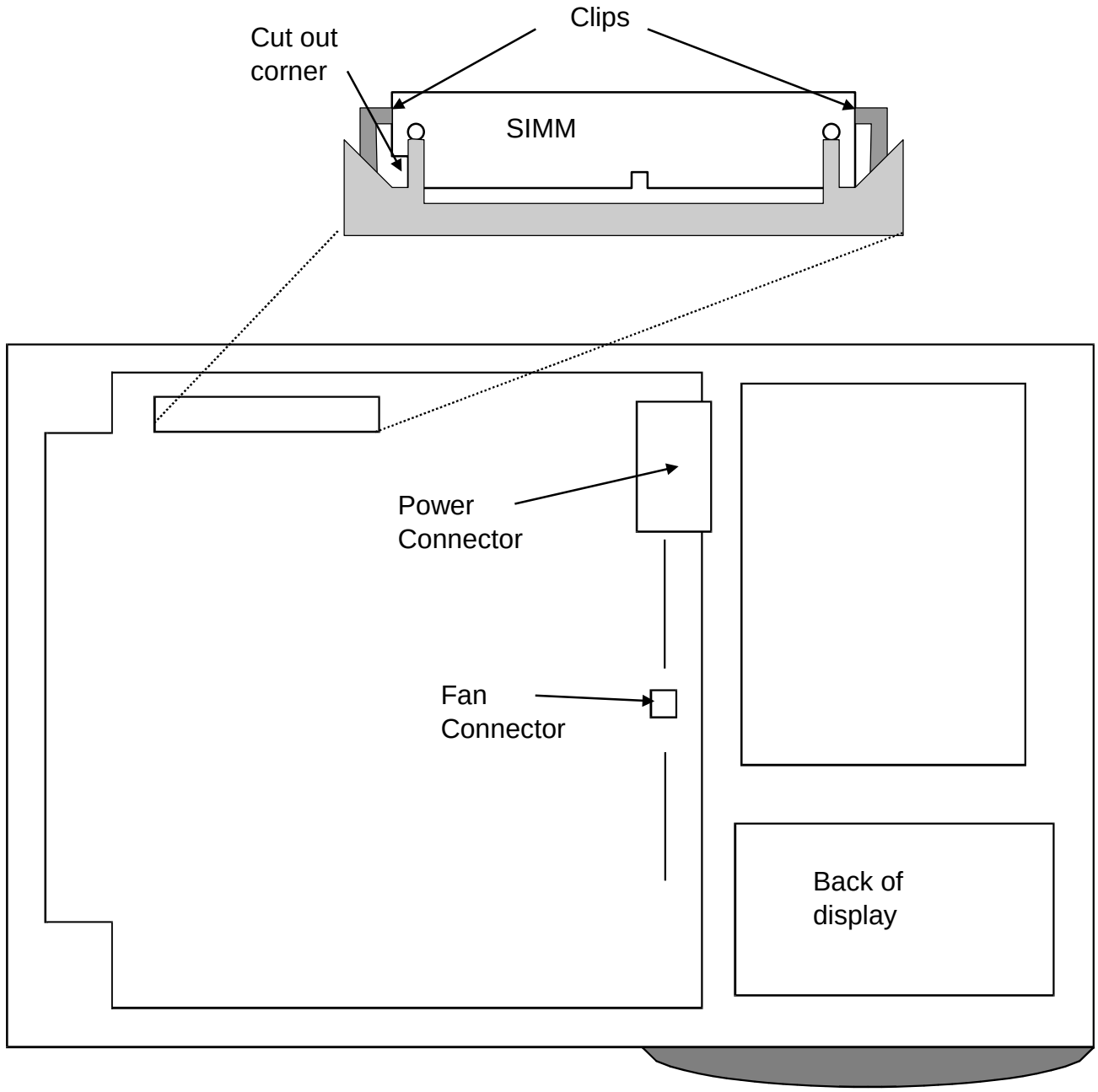

Clips
Cut out
corner
SIMM
Power
Connector
Fan
Connector
Back of
display

figure 5

A2 Specification

Device Support:	CMOS PROMs, EPROMs, FLASH, EEPROMs, Selected Micros.
Display:	20 x 4 character, alphanumeric back lit LCD.
Keypad:	Silicon rubber, tactile; 23 keys including dedicated program and cursor keys.
RAM:	32M bits standard (4M bytes) expandable to 128M bits. Standard 72 pin SIMM modules.
LEDs:	Bi-chromatic: red to indicate fail or green to indicate pass adjacent to each socket. Busy will be green to indicate the sockets are busy, Red to indicate the module is incorrect for the currently selected part.
Ports:	1x RS232C - 1200 to 115K2 baud hardware handshake and Xon/Xoff.
I/O Formats:	All popular including Intellec, Hex ASCII, Motorola S-Record, Binary, etc.
Remote Control:	Standard (optional StagCom software for PC available).
Alarm:	User selectable - to announce end of operation and/or on a key press.

ENVIRONMENTAL

Power:	90-263VAC 40 - 70 Hz, 60 Watts.	
Size:	440mm x 130mm x 260mm (17.2" x 5.2" x 10.4").	
Operating temperature:	0^0C - 35^0C.	
Humidity:	20-80% non-condensing.	
EMC:	**Emissions**	Complies with EN50081-1 class B EN55022 class B
	Immunity	Complies with EN50082-1
Safety:	Complies with EN60950, IEC950, CSA C22-950	

A3 Error Messages

The following is a list of error messages that the programmer can display in local operation, together with a list of common causes.

"BIT TEST FAIL"
One or more of the inserted devices had a bit programmed, which is required to be unprogrammed (see section 2.4.1).
In a flash or block erase part, BIT TEST FAIL can be returned if the limits are set to none block boundaries, so that erasing the block would erase more that the current limits.

"CODE INCORRECT"
The security code for RAM lock was entered incorrectly.

"CONNECT ERROR"
No devices inserted,
Device(s) incorrectly seated in their sockets
Device(s) fitted in the wrong part of the ZIF socket (see section 2.2)
Device(s) fitted the wrong way round.
Lever on a ZIF socket in incorrect position
Device(s) in the wrong position for load (see section 2.2.1)
Device(s) faulty.
See also the device support list for part specific instructions on inserting parts.

"EMPTY FAIL"
One or more of the inserted parts was found to be not empty, or would not erase.

"FUNCTION DISABLED"
Ram lock has been selected see section 2.9.3.

"INVALID STATUS CODE"
Software is corrupted - contact your authorised distributor.

"LOAD ERROR"
An error was detected on the data inputted from the port, or the input was aborted.

"MISMATCHED PARTS"
When Electronic ID is set to auto, and parts of incompatible programming algorithms are inserted, and a programming function is selected.
On some parts, 2 dies are marked externally the same but program differently. On these parts it is necessary for the programmer to read an electronic signature, to determine how to program the part. If parts of different dies are detected, then the programmer will return MIS MATCHED PARTS. The parts must be programmed in two lots.

"No or incorrect Module"
There is no module fitted, that is correct for the currently selected part, or the module is incorrectly fitted.

"No Software in RAM"
Upon trying to update the software, the RAM did not contain the update software, or it was corrupted (see section 2.9.2)

"Not Applicable"
The function is not applicable for the currently selected part, e.g. in the security menu on a part without security.
The user tried to output from the parallel port.

“OUT OF MEMORY”
The current selected part, and bit mode requires more memory, than the machine has, either select a different bit mode, or add more RAM.

"PROGRAM FAIL"
This is returned if one or more parts failed to program correctly. Possible reasons include:
- Part incorrectly inserted.
- Part faulty
- Part not empty, or locked and illegal bit not selected in SEQ, Pre-program menu (see section 2.4.1)
- Part locked or secured.
- Wrong device selected.
See also the device support list for part specific instructions.

"REVERSED/FAULTY PART"
A part was inserted incorrectly or the wrong way round, or the device is faulty.

"SECURITY FAIL"
The security fuse(s) have failed to program or erase correctly.

"SIGNATURE UNKNOWN"
The electronic ID of a part was not recognised, or the currently selected part does not have a signature.

"SOFTWARE ERROR"
Software is corrupted - contact your authorised distributor.

"UNABLE TO STORE DATA"
Hardware problem with internal EEPROM - contact your authorised distributor.

"VERIFY FAIL"
One or more parts did not verify with the RAM (see section 2.2.2)

"WARNING! RAM EXCEEDED"
More ram is required for the size of part that is fitted. Depending on the selected bit mode (see section 2.3), up to 8 times the size of the device is required in RAM.

"WRONG BIT MODE"
This message is displayed, if a G8xx (gang only) module, is used when an incorrect bit mode is selected.

"WRONG MODULE"
The module is incorrect for the selected part.

"WRONG PART"
When Electronic ID is set to check, and parts are inserted that are different to the selected part.