

**Operating Manual**

**for**

**RS PP39 Universal MOS Programmer**

**with**

**39M100 EPROM/EEPROM Module**

**and**

**39M200 Microprocessor Module**

**Manual Revision 2**

R.S. Components Ltd.  
P.O. Box 99  
Corby  
Northants  
NN17 9RS

Tel: 0536 201234  
Tlx: 342512

(Revision 2) Address-01



## RS PP39 CONTENTS

Section	1	GENERAL INTRODUCTION
	1.1	Introduction
	1.2	Modules
	1.3	The Keyboard
	1.4	Initial Setting-up Procedure
	1.5	Selection of Local and Remote Modes
		39M100
Section	2	39M100 INTRODUCTION
	2.1	The 39M100 Module
	2.2	RAM Operating Structure
	2.3	List of 'Set' Commands
Section	3	SELECTING A DEVICE
	3.1	Selecting a Device
	3.2	List of Devices and Device Codes
	3.3	Electronic Identifier
Section	4	BIT MODE
	4.1	Selection of Bit Mode Configuration
	4.2	8-Bit Mode
	4.3	Gang Mode
	4.4	16-Bit Mode
	4.5	32-Bit Mode
Section	5	DEVICE FUNCTIONS
	5.1	Load
	5.2	Empty Test
	5.3	Pre-Program Bit Test
	5.4	Programming
	5.5	Verify
	5.6	Access Time Testing
	5.7	Checksum and Cyclic Redundancy Check (CRC)
	5.8	Device/RAM Address Limits
	5.9	Save and Recall Machine Configurations
Section	6	RAM FUNCTIONS
	6.1	Interlace *
	6.2	List and Edit
	6.3	Insert
	6.4	Delete
	6.5	Block Move
	6.6	Filling the RAM
	6.7	String Search

## 39M200

Section	7	39M200 INTRODUCTION
	7.1	39M200 Module
	7.2	RAM operating structure
	7.3	List of 'SET' Commands
Section	8	SELECTING A DEVICE
	8.1	Device Type Selection
	8.2	List of Devices and Device Codes
Section	9	DEVICE FUNCTIONS
	9.1	Load
	9.2	Empty Test
	9.3	Programming
	9.3.1	Verify Pass-Security Bit
	9.3.2	In-program Verify
	9.4	Checksum
	9.5	Device/RAM address limits
	9.6	Save and Recall Machine Configurations
	9.7	68705 Device
Section	10	RAM FUNCTIONS
	10.1	List
	10.2	Edit
	10.3	Insert
	10.4	Delete
	10.5	Block move
	10.6	Filling the RAM
	10.7	String Search

## 39M100 and 39M200

Section	11	INTERFACE
	11.1	Setting the Input/Output Interface Parameters
	11.2	Input/Output Operations
	11.2.1	Input Parameters
	11.2.2	Output Parameters
	11.3	Error Reporting on Input/Output
Section	12	FORMAT DESCRIPTIONS
	12.1	Interface Formats
	12.1.1	Intellec
	12.1.2	Extended Intellec
	12.1.3	Hex ASCII
	12.1.4	Exorcisor
	12.1.5	Extended Exorcisor
	12.1.6	Tek-Hex
	12.1.7	Extended Tek-Hex
	12.1.8	PPX or Stag Hex
	12.1.9	Binary, DEC Binary and Binary Rubout

Section	13	RS232C HARDWARE DESCRIPTIONS
	13.1	RS232C Interface Port Connections
	13.2	XON/XOFF (3 wire cable form)
	13.3	Hardware Handshake (7 or 8 wire cable form)
	13.4	Non-Standard Connections
Section	14	REMOTE CONTROL
	14.1	Selecting Remote Control
	14.2	Remote Control Commands
	14.3	Remote Error Words and Codes
Section	15	SPECIFICATION
	15.1	The ASCII code
	15.2	Specification



# **SECTION 1**





# **1.-GENERAL INTRODUCTION**



## 1.1 INTRODUCTION

The PP39 is a Universal MOS Programmer, which in conjunction with its family of modules is capable of supporting all MOS erasable PROM and MICRO devices in NMOS and CMOS technology.

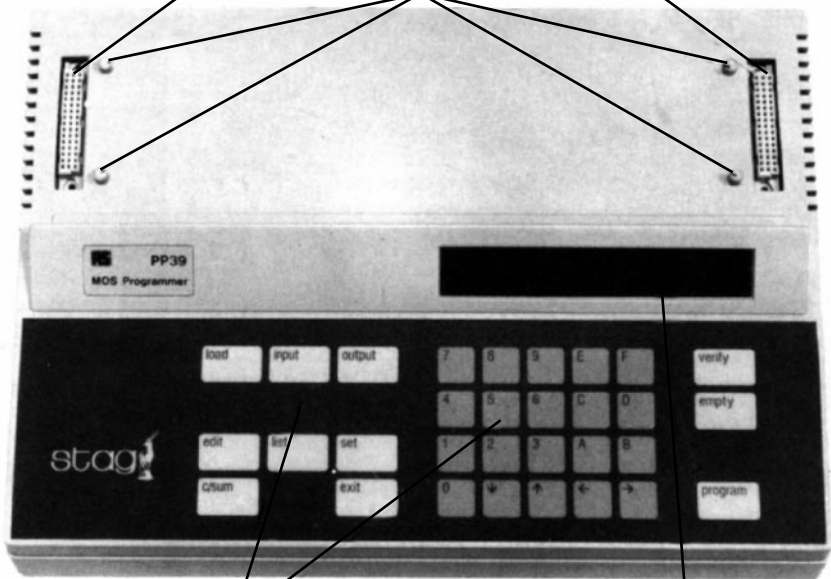
The Programmer is software controlled using a single level module approach. This ensures flexibility and ease of upgrade for future devices; whereby the module alone can be returned for software upgrades. (For urgent programming needs a module exchange plan is available).

The PP39 can be operated in 'LOCAL' mode or it can be linked to a computer via the serial RS232C interface port enabling 'REMOTE' operation of the machine.

### RS PP39 MAINFRAME

Two polarised sockets provide interconnections from the main frame to the PP39 modules.

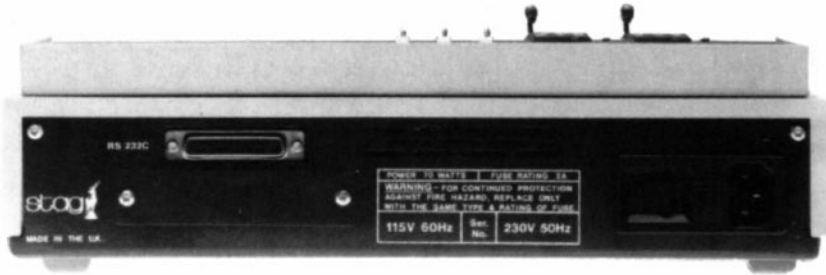
Four locating pins provide automatic module alignment.



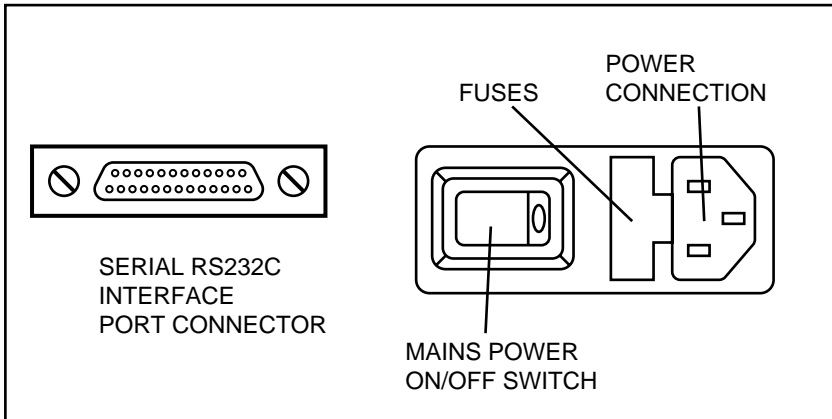
Keyboard: For data entry and operating programmer functions.

16-Character Green alphanumeric display

# 39M 100 EPROM & EEPROM MODULE



## SPECIFICATIONS

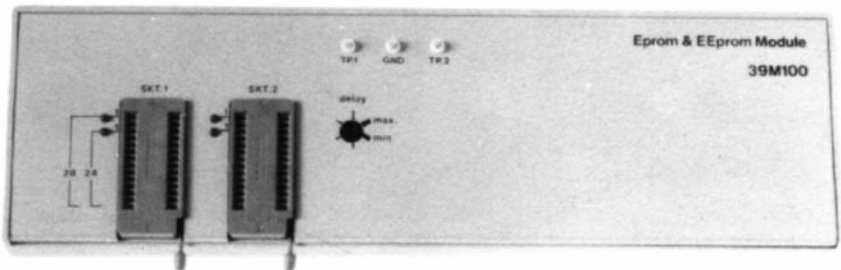


## 1.2 MODULES

A variety of modules is available to plug into the main frame. This guarantees future flexibility to support new devices as they are developed.

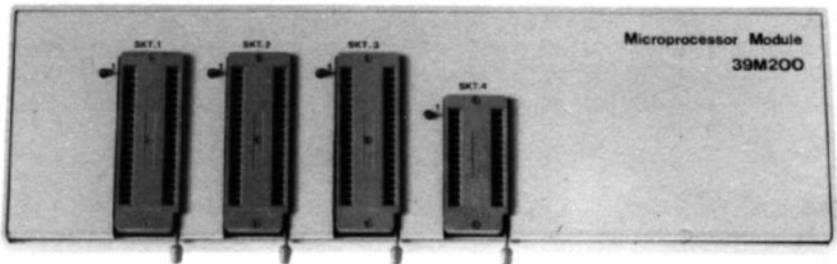
### 39M100 – EPROM/EEPROM

This module supports NMOS and CMOS, EPROM and EEPROM devices in both 24 and 28 pin DIPs packages. The module features algorithms for fast programming and it supports Silicon Signature\* technology for automatic device identification. Sockets are provided for set programming of two devices simultaneously or they can be used as a mini-ganger. The PP39 can be configured as an 8, 16 or 32 bit machine. Access time tests can be performed and an auto-recall feature is incorporated where pre-set parameters can be recalled from a non-volatile memory at any time for ease of use.



### 39M200 MICROPROCESSOR MODULE

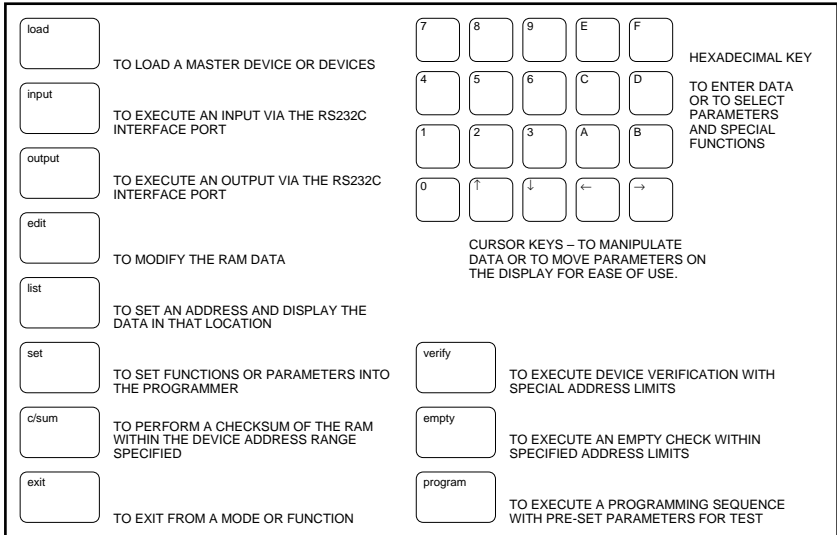
This module will program single chip microprocessors containing EPROM from AMD, Intel, NEC and Motorola in both 40 pin and 28 pin DIPs packages.





## 1.3 THE KEYBOARD

For data entry and operating programmer functions



## OPTIONAL FEATURES

### RAM EXTENSION BOARD

The PP39 is presently supplied as standard with 512K bits of RAM. As larger devices become available this will prove to be inadequate and more RAM will be required.

Therefore the PP39 has the facility to have its RAM expanded to 1M-bits, 2M-bits, 4M-bits and beyond.





## 1.4 INITIAL SETTING UP PROCEDURE

Before attempting to apply power to your PP39 Programmer ensure that it is set to the correct operating voltage for your power source. The voltage setting is printed on the rear panel.

1. Plug the supplied Power cord into the rear panel socket
2. Apply power to the machine from the mains power source
3. Power-up the machine using the ON/OFF switch on the rear panel

After "POWER UP" and without a 39-Module inserted the display will read:



MODULE 7

The mainframe software revision can now be ascertained prior to the module being inserted simply by pressing the key marked 'SET' followed by the key marked '6', e.g.



PP39 155 04

to remove press 'EXIT'

In order to make this manual as straightforward as possible the action of pressing the key marked 'SET' followed by another key or keys will be abbreviated to a single instruction e.g. 'SET 6', 'SET F3', 'SET INPUT' etc.

### Note

To ensure correct initialisation, power down before inserting a module. Always wait five seconds before applying power again.

## INTRODUCTION OF A 39-MODULE TO THE MAIN FRAME

Having completed the setting up procedure the PP39 is ready to receive its 39-Module. Controlling software for the machine resides in the selected module, therefore the operation of the Programmer is dependent upon the type of 39-Module plugged into the main frame.

On power-up the programmer will be configured automatically to what it was before the machine was last powered down.

This ensures that once any machine parameter has been set-up, it needn't be reset every time the machine is switched on.

For instance, if the machine was previously used in the 'LOCAL' mode with the 39M100 Eprom Module inserted, the initial configuration of the machine will be set on power-up and the display will show the last entered manufacturer, device type and selected mode, such as:



AMD 2764 GANG

To determine the software revision of the Module press 'SET 6' and the display will show:



MODULE M100 07

To remove, press 'EXIT'.

## 1.5 SELECTION OF 'LOCAL' OR 'REMOTE' MODES

The Programmer will be in either 'Local' or 'Remote' on power up.

### To Select Local Mode

When the machine is in remote mode on power-up the display will show manufacturer, device type and remote mode itself. For instance:

Manufacturer	Device type	Remote mode
5E0	5516A	REM

To exit from remote one of two sequences can be performed:

- (i) If the programmer is connected via the I/O port to a computer or terminal keyboard then the sequence of pressing Key 'Z' followed by Key 'RETURN' will bring control back to 'local' on the PP39 keyboard.
- (ii) If the PP39 is in stand alone mode on power-up but, still under the 'remote' setting, the operator must power down wait five seconds and then power up again with the 'EXIT' key depressed to reach 'local' mode.

When either sequence (i) or (ii) is performed the display will show manufacturer, device type and bit mode for instance a typical 'local' mode setting for the 39M100 might be:

Manufacturer	Device type	Bit mode
5E0	5516A	M0

In local mode all functions of the PP39 are controlled from its own keyboard.

## REMOTE CONTROL

### To select remote control

Press set 2 and the display will show:



REMOTE PRESS SET

By pressing set again, the display will show the manufacturer, device type and remote mode.

For instance:

Manufacturer	Device type	Remote mode
5EQ	5516A	REM

In the remote mode, the PP39 operates under remote control from a computer or a terminal. The keyboard of the PP39 is inoperative at this time and the display will only show information as requested under remote control.

## **SECTION 2**



**39M100**









## 2.2 RAM OPERATING STRUCTURE



## 2.3 LIST OF 'SET' COMMANDS

set 0	Allows user to scan and select various manufacturers and device types
set 1	Selects interface parameters Format, Baud Rate, Word Length, Stop Bits, Parity
set 2	Sets programmer into 'Remote' control. (To return to Local Mode: Power up with exit key depressed)
set 3	Selects mode of machine, i.e. 8-Bit, 16-Bit, 32-Bit or GANG operating mode.
set 4	Displays RAM size in hexadecimal
set 5	RAM data complemented from lower to upper address limit
set 6	Displays module software revision if module is plugged in, or main frame software revision, if no module is plugged in.
set 7	Verifies device under access time control
set 8	Calculates and displays CRC (Cyclic Redundancy Check)
set A1	To A9 Saves machine configuration (up to nine sets)

## LIST OF 'SET' COMMANDS (continued)

set B1      To      B9      Recalls previously saved machine configurations (up to 9 sets)

set F0      Fills entire RAM with 00

set FF      Fills entire RAM with FF

set F1      Audible Alarm: To indicate end of program, test, or as a warning using a combination of bleeps and tones. SET F1 both enables and disables this function.

set F2      Fills RAM with arbitrary variable from lower to upper address limit

set F4      Block Move: A block of data with pre-selected address limits can be copied and then re-located at another address within the RAM.

set F6      Defines RAM and device address ranges for all functions which operate on the device

set F7      Access time calibration – Provides repetitive waveform for access time calibration

set input      Input – Enters input address offset

set output      Output – Enters output address offset, start address and stop address

## LIST OF 'SET' COMMANDS (continued)

set E1

Electronic Identifier: Mode (i) Two Key Operation

set E2

Electronic Identifier: Mode (ii) Single Key Operation

set 9

String Search

set FE

Applicable to MOTOROLA 2816 ONLY – Erases Device





## **SECTION 3**



### 3.1 SELECTING A DEVICE

#### Selecting the device using a 4 digit code

The complete range of devices supported by the 39M100 is stored in the module. Each individual device has its own four digit code. (See device code list Section 3.2)

SET 0 – Allows code selection


SEQUENCE: Prior to SET '0' the display will show the last entered configuration

For example:



The display shows the manufacturer 'AMD', the device code '2716', and the bit mode 'MB'.

By pressing SET '0' the device code of this configuration will be displayed:



The display shows the text 'DEVICE CODE' followed by the device code '9F42'.

When the new device code to be entered is already known. (For instance AF44 is the code for a Fujitsu 2732 EPROM device.) Then the AF44 can be entered directly onto the display from the keyboard replacing the old code:



The display shows the text 'DEVICE CODE' followed by the new device code 'AF44'.

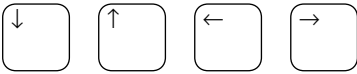
The selection sequence can be completed by pressing EXIT whereby the new manufacturer and device type are displayed along with the bit mode:



The display shows the manufacturer 'FUU', the device code '2732', and the bit mode 'MB'.

## Scanning device types and manufacturers by use of cursor keys

When a device code is not known or if the user wishes to scan the devices available, selection can be made via the cursor keys:

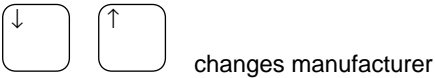


By pressing SET '0' the code of the last used device is displayed:

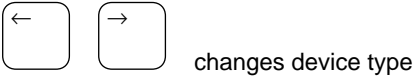


The manufacturer and device type can be changed by use of the cursor keys:

The up/down keys scan the range of manufacturers.



The left/right keys scan the device range of a particular manufacturer.



### 3.2 LIST OF DEVICES AND DEVICE CODES

This list of parts is supported by the 39M100 Module. Each device carries a four digit code. The first two digits define the manufacturer of the device and the second two digits refer to the device type.

#### EPROMS

Manufacturer	Device	Device Code	Device Size (Hex)
AMD	2716	9F 42	800
	2732	9F 44	1000
	2732A	9F C4	1000
	2764	9F 4A	2000
	2764A	9F CA	2000
	27128	9F 4B	4000
	27128A	9F CB	4000
	27256	9F 4C	8000
	27512	9F 4D	10000
Fujitsu	2716	AF 42	800
	2732	AF 44	1000
	2732A	AF C4	1000
	2764	AF 4A	2000
	27128	AF 4B	4000
	27256	AF 4C	8000
	27C256	AF DC	8000
	27C128	AF DB	4000
	27C256A	AF CC	8000
	27C512	AF DD	10000
General Instruments	27C64	02 DA	2000
	27HC64	02 DA	2000
	27256	02 4C	8000
	27C256	02 DC	8000
Hitachi	2716	BF 42	800
	2532	BF 43	1000
	2732	BF 44	1000
	2732A	BF C4	1000
	2764	BF 4A	2000
	27C64	BF DA	2000
	27128	BF 4B	4000
	27128A	BF CB	4000
	27256	BF 4C	8000
	27C256	BF DC	8000
	27512	BF 4D	10000

**LIST OF DEVICES AND DEVICE CODES (continued)**

<b>Manufacturer</b>	<b>Device</b>	<b>Device Code</b>	<b>Device Size (Hex)</b>
Intel	2716	6F 42	800
	2732	6F 44	1000
	2732A	6F C4	1000
	2764	6F 4A	2000
	2764A	6F CA	2000
	27128A	6F CB	4000
	27128B	6F CB	4000
	27C64	6F DA	2000
	27128	6F 4B	4000
	27C128	6F DB	4000
	27256	6F 4C	8000
	27256 **	6F FC	8000
	27C256**	6F DC	8000
	27512	6F 4D	10000
	27512 **	6F FD	10000
	27513	6F CD	10000
	27513 **	6F FE	10000
	87C64	6F EA	2000
87C128	6F EB	4000	
87C256	6F EC	8000	
Mitsubishi	2716	DF 42	800
	2732	DF 44	1000
	2732A	DF C4	1000
	2764	DF 4A	2000
	27128	DF 4B	4000
	27C128	DF DB	4000
	27256	DF 4C	8000
	27512	DF 4D	10000
Motorola	2716	7F 42	800
	2532	7F 43	1000
	2732	7F 44	1000
	68764	7F 45	2000
	68766	7F 47	2000
National	2716	3F 42	800
	2732	3F 44	1000
	27C32H	3F 46	1000
	27C32B	3F D5	1000
	2732A	3F C4	1000
	2764	3F 4A	2000
	27128	3F 4B	4000
	27C16	3F D2	800
	27C32	3F D4	1000
	27C64	3F DA	2000
	27CP128	3F DB	4000
	27C256	3F DC	8000
	27C512	3F DD	10000

**LIST OF DEVICES AND DEVICE CODES (continued)**

<b>Manufacturer</b>	<b>Device</b>	<b>Device Code</b>	<b>Device Size (Hex)</b>
NEC	2716	CF 42	800
	2732	CF 44	1000
	2732A	CF C4	1000
	2764	CF 4A	2000
	27C64	CF DA	2000
	27128	CF 4B	4000
	27256	CF 4C	8000
	27C256	CF DC	8000
	27C256A	CF CC	8000
	27C512	CF DD	10000
Oki	2716	08 42	800
	2532	08 43	1000
	2732	08 44	1000
	2732A	08 C4	1000
	2764	08 4A	2000
	27128	08 4B	4000
SEEQ	5133	FF 4A	2000
	2764	FF 4A	2000
	5143	FF 4B	4000
	27128	FF 4B	4000
	27256	FF 4C	8000
	27C256	FF DC	8000
SGS	2716	8F 42	800
	2532	8F 43	1000
	2732A	8F C4	1000
	2764	8F 4A	2000
	2764A	8F CA	2000
	27128A	8F CB	4000
	27256	8F 4C	8000
Signetics	27C64A	1F DA	2000
	87C64A	1F EA	2000
SMOS	27C64	0F DA	2000
	27128	0F 4B	4000
	27C256	0F DC	8000

**LIST OF DEVICES AND DEVICE CODES (continued)**

<b>Manufacturer</b>	<b>Device</b>	<b>Device Code</b>	<b>Device Size (Hex)</b>
Texas Instruments	2516	4F 42	800
	2532	4F 43	1000
	2532A	4F 41	1000
	2732A	4F C4	1000
	2564	4F 47	2000
	2764	4F 4A	2000
	27128	4F 4B	4000
	27128A	4F CB	4000
	27C128	4F DB	4000
	27256	4F 4C	8000
	27C256	4F DC	8000
	27C512	4F DD	10000
Toshiba	2732	EF 44	1000
	2732A	EF C4	1000
	2764	EF 4A	2000
	2764A	EF CA	2000
	27128	EF 4B	4000
	27128A	EF CB	4000
	27256	EF 4C	8000
	27256A	EF CC	8000
	57256	EF DC	8000
VTI	27C64	04 DA	2000
	27C128	04 DB	4000
	27C256	04 DC	8000
Waferscale	27C64	0B DA	2000
	57C64	0B DA	2000
	27C128	0B DB	4000
	57C49	0B 06	
E100	2516	0E 42	800
	2716	0E 42	800
	2532	0E 43	1000
	2732	0E 44	1000
	2564	0E 47	2000
	2764	0E 4A	2000
	87C64	0E EA	2000
	27128	0E 4B	4000
	87C128	0E EB	4000
	27256	0E 4C	8000
	87C256	0E EC	8000
	27512	0E 4D	10000

\*\* Indicates Quick Pulse Programming Parts



## LIST OF DEVICES AND DEVICE CODES (continued)

### EEPROMS

Manufacturer	Device	Device Code	Device Size (Hex)
AMD	9864	9F 5A	2000
	2817A	9F 58	800
	2864B	9F 5C	2000
Exel	2816A	01 52	800
	2817A	01 58	800
	2864A	01 5C	2000
	2865A	01 5C	2000
Hitachi	48016	BF 51	800
	58064	BF 5A	2000
	58C65	BF 7C	2000
Intel	2816A	6F 52	800
	2817 *	6F 53	800
Motorola	2816	7F 52	800
	2817	7F 53	800
National	9716	3F 51	800
	9817	3F 58	800
	98C64	3F 7C	2000
NEC	28C64	CF 7C	2000
Rockwell	2816A	06 52	800
	87C32	06 D4	1000
Samsung	2816A	09 52	800
	2864A	09 5C	2000
	2865A	09 5C	2000
SEEQ	5516A	FF 52	800
	2816A	FF 52	800
	5213	FF 53	800
	52B13	FF 54	800
	52B13H	FF 56	800
	52B33	FF 5A	2000
	52B33H	FF 5E	2000
	2817A	FF 58	800
	2864	FF 5B	2000
	28C256	FF 7F	8000
	28C64	FF 7C	2000
SMOS	2864	0F 5B	2000

## LIST OF DEVICES AND DEVICE CODES (continued)

<b>Manufacturer</b>	<b>Device</b>	<b>Device Code</b>	<b>Device Size (Hex)</b>
Xicor	X2804A	07 50	400
	X2816A	07 52	800
	2816H	07 56	800
	2816B	07 56	800
	28C16	07 76	800
	2864A	07 5C	2000
	2864B	07 5C	2000
	28C64	07 7C	2000
	2864H	07 5B	2000
	28256	07 5F	8000
	28C256	07 7F	8000

\* Intel 2817 requires AM100 6F53 Adaptor

### 3.3 ELECTRONIC IDENTIFIER

#### Important Note:

Devices which do not contain an Electronic Identifier may be irreparably damaged if they are used in the Silicon Sig mode.

Electronic Identifier is a term used to describe a code mask programmed into a PROM which identifies the device type and manufacturer. The code is stored outside the normal memory array and is accessed by applying 12 Volts to address line A9. This allows the PP39 to directly identify any device containing an Electronic Identifier and thus eliminate the need for the user to select the device type.

The PP39 presently uses two modes of Silicon Sig operation both of which only work with 28 pin devices.

#### Mode (i): Two Key Operation

On pressing SET E1 the display will show "SILICON SIG" alongside the selected bit-mode:



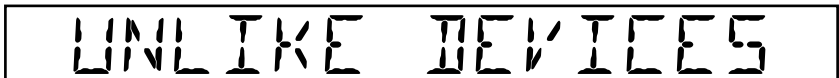
SILICON SIG M8

If any device function key is pressed e.g. Program, Load etc. the PP39 will first attempt to read the signature of any devices present. If no code can be read or the code is not found in the PP39's list of valid codes the display will show:



SI SIG NOT FOUND

If two devices are successfully recognised but are incompatible i.e. they use different programming algorithms the display will show:



UNLIKE DEVICES

If neither of the above two fault conditions occur then the manufacturer and the device type will be displayed. In the case of devices in both sockets the manufacturer and device code of the device in the left socket will be displayed.

To execute the function the specified 'device function key' must be pressed again e.g. Prog, Load etc.

To exit from the Silicon Sig mode select a device using SET 0 in the usual manner.

### Mode (ii): Single Key Operation

Pressing SET E2 will again display "SILICON SIG" alongside the selected bit-mode.



SILICON SIG M8

Operation is similar to the previously described mode except that the PP39 rather than stopping to display the manufacturer and device type continues straight on to execute the selected function.

To exit from the Silicon Sig mode select a device using SET 0 in the usual manner.

## **SECTION 4**



## BIT MODES

### 4.1 SELECTION OF BIT MODE CONFIGURATION

The last used bit mode will already be displayed on power up along with manufacturer and device type. For instance:



AMD 2764 M16

By pressing SET 3 the bit mode along is shown:



MODE--16 BIT

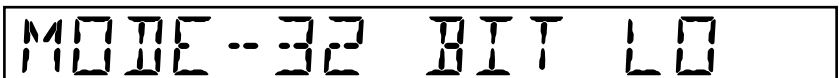
The range of modes can now be scanned by pressing either the up or down cursor keys:



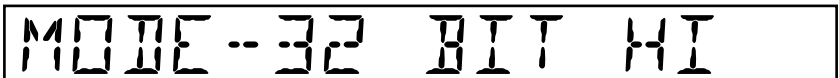
MODE--8 BIT



MODE--16 BIT



MODE--32 BIT LO

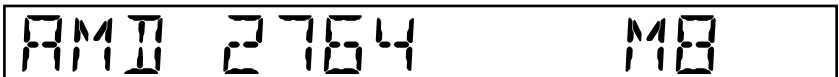


MODE--32 BIT HI



MODE--6ANG

When selection is made press EXIT for operation in chosen mode, e.g.:



AMD 2764 M8





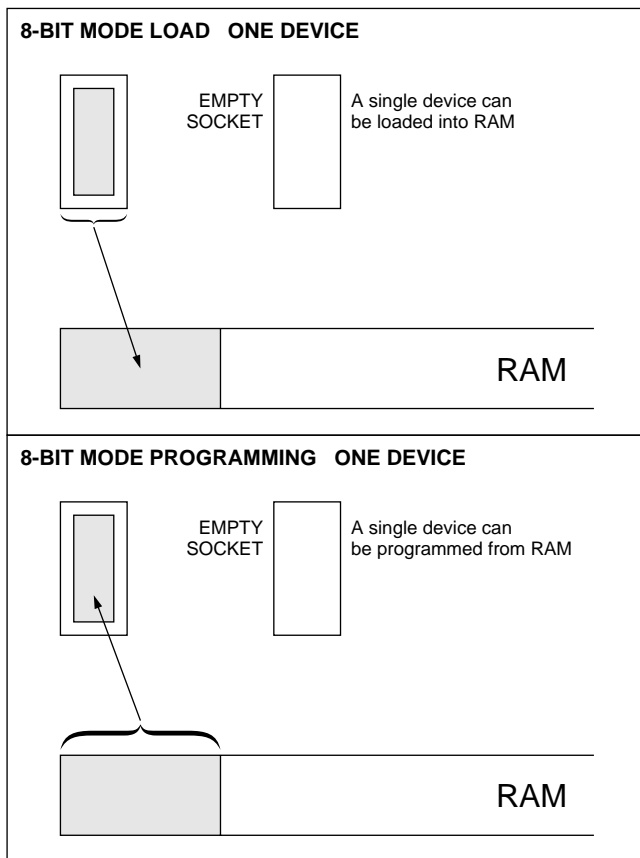
## 4.2 8-BIT MODE

In the 8-Bit mode, the programmer is configured to handle 8-bit data as single devices or in sets of two.

### One Device

Either left or right ZIF can be used. If only one device is to undergo program or load

- (i) The PP39 can detect a single device in a particular socket.
- (ii) The information in the device can be loaded into a specifically located section of the RAM.
- (iii) The information to be programmed into the device comes from a specifically located section of the RAM.
- (iv) These specific sections of the RAM are pre-selectable.



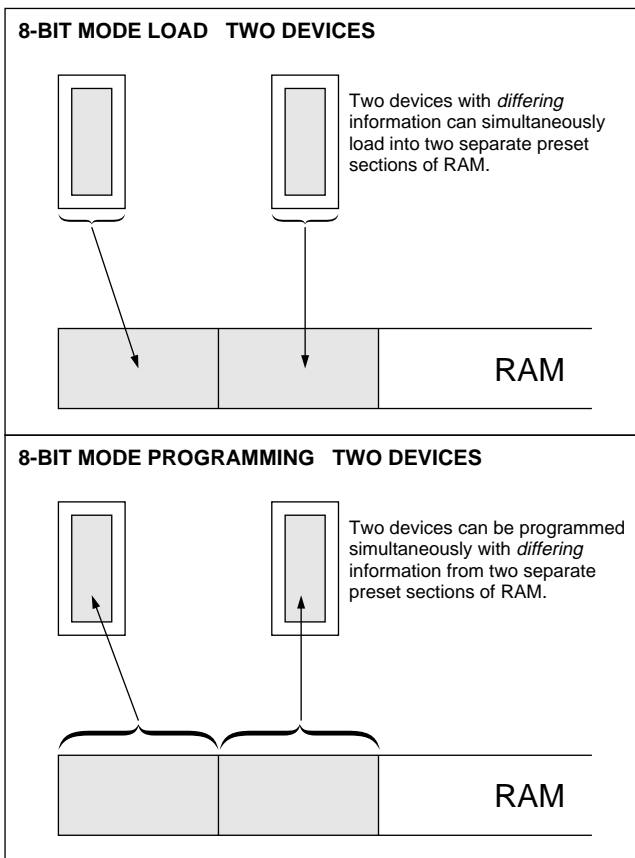
## 8-Bit Mode

When two devices are to undergo program or load naturally both sockets will be used.

### Two devices

- (i) The PP39 can detect devices in both sockets.
- (ii) The information in the two individual devices can be loaded into two separate but specifically located sections of the RAM.
- (iii) The information to be programmed into the two individual devices comes from two separate but specifically located sections of the RAM.
- (iv) These two specifically located sections of the RAM are pre-selectable.

Pre-selection of RAM address ranges also applies to 16-bit, 32-bit and gang mode.



## DISPLAY

### 8-Bit Mode

By pressing Key 'List' a visual display of information in TWO HEX CHARACTERS can be shown at a specific address.

ADDRESS (ZERO)	TWO HEX CHARACTER
000000	12

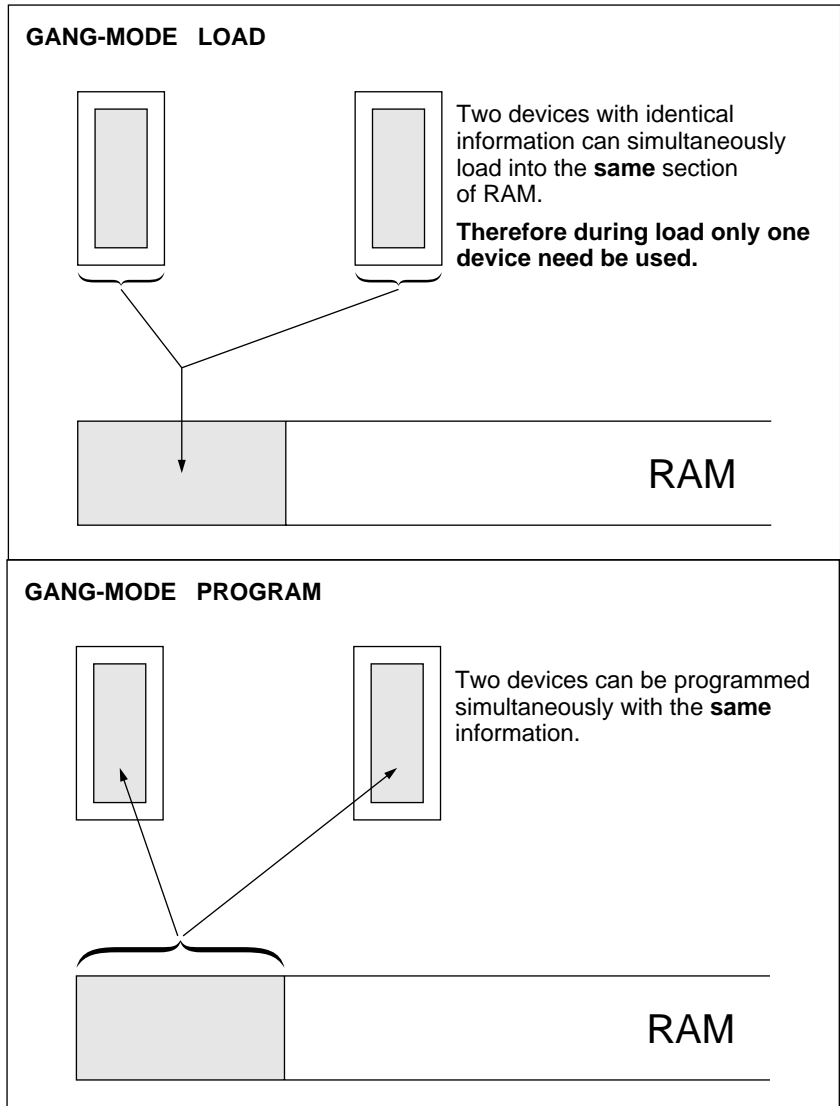
(Under the 'List' function THE ADDRESS RANGE can be scanned by use of the hex keyboard and cursor keys).



### 4.3 GANG MODE

**LOAD:** Two devices with identical information can simultaneously load into the same section of the RAM. **Therefore during 'Load' either socket can be used.**

**PROGRAM:** Two devices can be programmed simultaneously with identical information.





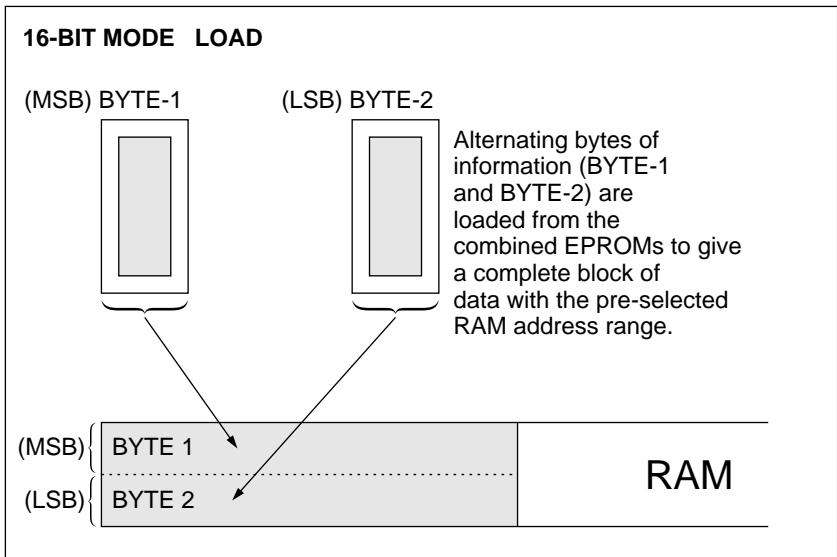
#### 4.4 16-BIT MODE

**LOAD:** The two EPROMs combined can load over a common address range within the RAM.

Alternating bytes of information (BYTE-1 and BYTE-2) are loaded from the combined EPROMs to give a complete block of data within the pre-selected RAM address range.

BYTE-1 Will reside in the left hand ZIF socket. It represents the (MSB) "most significant byte" of a 16-Bit parallel word.

BYTE-2 Will reside in the right hand ZIF socket. It represents the (LSB) "least significant byte" of a 16-bit parallel word.



MSB – The most significant BYTE in binary code

LSB – The least significant BYTE in binary code

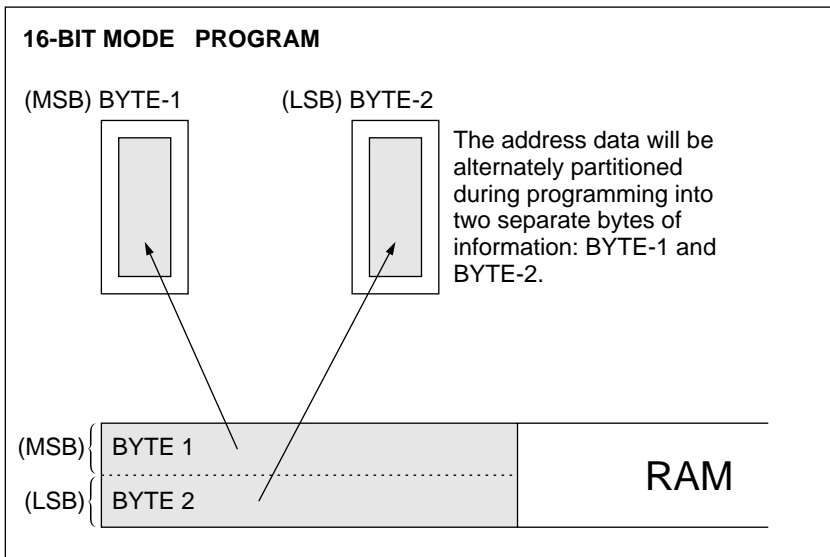
## 16-BIT MODE

**PROGRAM:** The two EPROMs can be programmed from a common address range within the RAM.

The address data will be alternately partitioned during programming into two separate bytes of information: BYTE-1 and BYTE-2.

BYTE-1 Will reside in the left hand ZIF socket. It represents the (MSB) "most significant byte" of a 16-Bit parallel word.

BYTE-2 Will reside in the right hand ZIF socket. It represents the (LSB) "least significant byte" of a 16-bit parallel word.



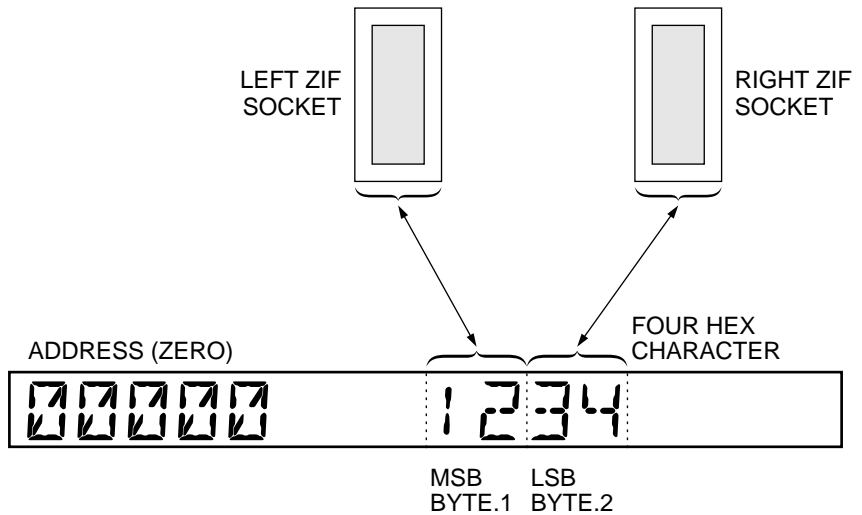
MSB – The most significant BYTE in binary code

LSB – The least significant BYTE in binary code



## DISPLAY

The two bytes can be **displayed** as four hex characters by pressing key 'list'.



(Under the 'List' function THE ADDRESS RANGE can be scanned by use of the hex keyboard and cursor keys) see section 'LIST'.



#### 4.5 32-BIT MODE

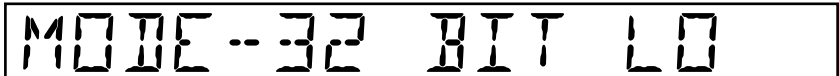
The machine can be configured to handle 32-Bit information

##### PROGRAM

- (i) Four EPROMs can be programmed from a common address range within the RAM.
- (ii) Data programmed using the 32-Bit word is divided into four separate bytes: BYTE-1, BYTE-2, BYTE-3 and BYTE-4.
- (iii) Each BYTE will program into one of the four individual EPROMs.
- (iv) Two programming operations occur.
- (v) The two operations are called:

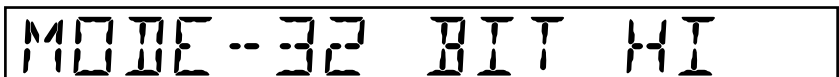
##### LOAD

- (i) Four EPROMs can load into a common address range within the RAM.
- (ii) Data loaded from the four EPROMs using the 32-Bit word is divided into four separate bytes: BYTE-1, BYTE-2, BYTE-3 and BYTE-4.
- (iii) Each BYTE is divided from one of the four EPROMs.
- (iv) Two loading operations occur.
- (v) The two operations are called:



MODE--32 BIT LO

AND

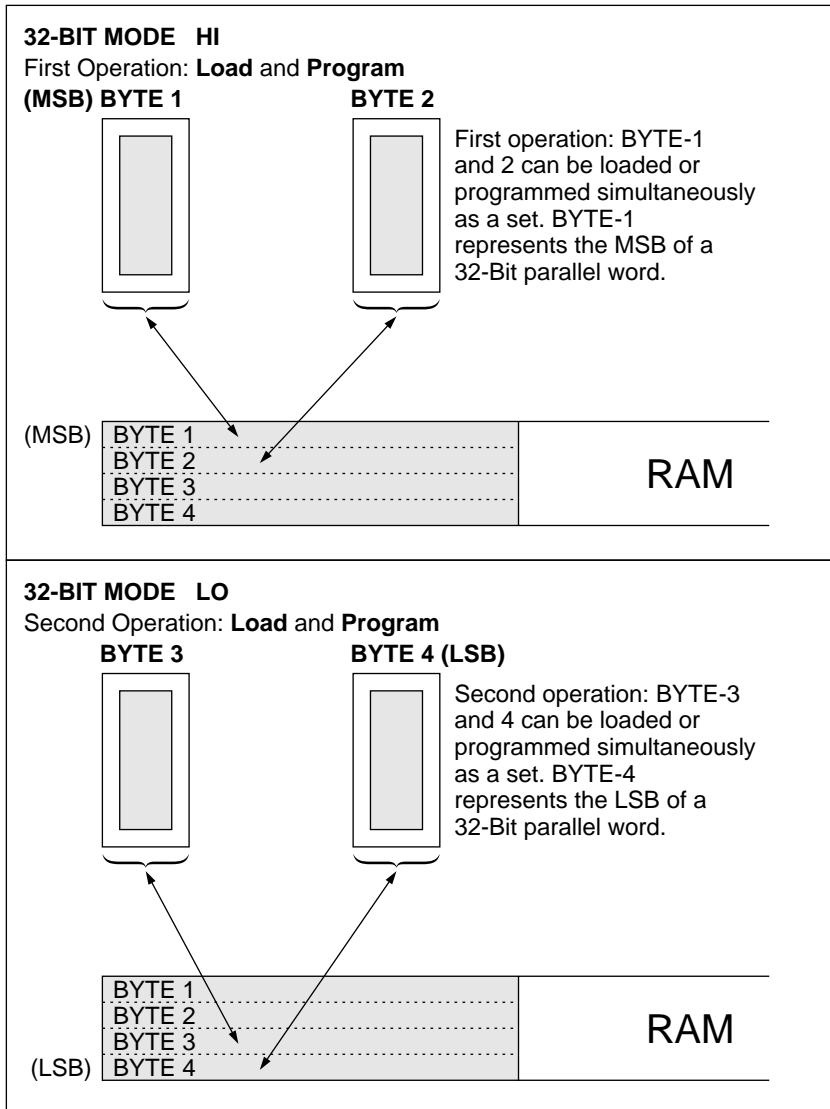


MODE--32 BIT HI

- (vi) 32-BIT HI represents BYTE 1 and BYTE 2.
  - (a) BYTE-1 resides in the left hand ZIF socket.
  - (b) BYTE-2 resides in the right hand ZIF socket.
- (vii) 32-Bit LO represents BYTE-3 and BYTE-4.
  - (a) BYTE-3 resides in the left hand ZIF socket.
  - (b) BYTE-4 resides in the right hand ZIF socket.

## 32-BIT MODE

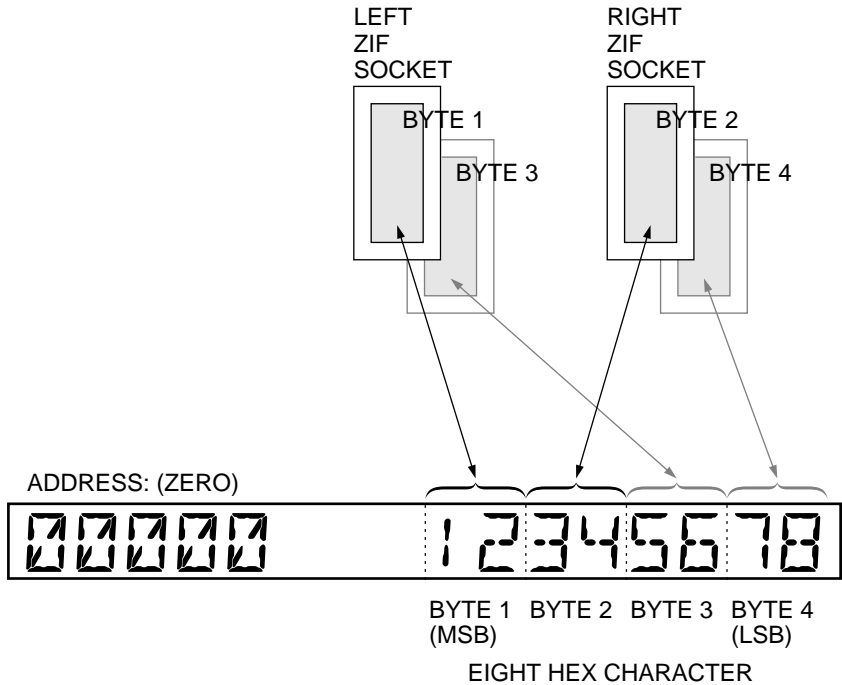
Division of RAM during Load and Program



MSB – The most significant BYTE in binary code.  
LSB – The least significant BYTE in binary code.

## DISPLAY

The four Bytes can be **displayed** as an eight hex character by pressing Key 'List'.



(Under the 'List' function THE ADDRESS RANGE can be scanned by use of the hex keyboard and cursor keys).



## **SECTION 5**





## DEVICE FUNCTIONS

### 5.1 LOAD

#### Loading the RAM from a 'master' PROM


Insert the master device or devices into the ZIF sockets. Press the Load key. On completion of load the display will show:



6C90 CSUM 18E0

BOTH ZIF SOCKETS IN USE

The checksum will be displayed on the left as well as on the right hand side of the display, assuming both sockets are in use. However, if only one device is inserted in either of the ZIF sockets then the checksum will appear on either the left or the right of the display corresponding to the socket used:



6C90 CSUM

LEFT ZIF  
SOCKET IN USE



CSUM 18E0

RIGHT ZIF  
SOCKET IN USE

The programmer has the ability to detect empty sockets and therefore only the ZIF socket in use will be shown on the display.




## PROGRAMMING SEQUENCE

### 5.2 EMPTY TEST

If required an 'empty test' can be applied to the device or devices in the ZIF sockets prior to programming. This can be done by pressing the 'empty' key. The device or devices will be examined for the unblown state (FF); if both are entirely empty the display will show:

BOTH ZIF SOCKETS IN USE



PASS EMPTY PASS

LEFT ZIF  
SOCKET


RIGHT ZIF  
SOCKET

If a device were to fail the 'empty test' the display would show:

(i) The first location where a discrepancy occurs; (ii) The unblown state of the selected device; (iii) The EPROM data at that particular location.

For instance:

BOTH ZIF SOCKETS IN USE



0000 RR-FF RP-2E

LOCATION  
(ZERO)

UNBLOWN  
STATE (FF)

RIGHT DEVICE  
CONTAINING DATA (2E)

In this example the device in the right hand ZIF socket has failed. The device contains data '2E' and not the unblown state 'FF'. This discrepancy occurs at the first location – 0000 (ZERO).

Continually pressing 'empty' will allow the whole device to be tested for the empty state, and each successive failure will be displayed.



If the empty test passes or is unnecessary the programming can begin.

Pressing the program key will automatically execute the 'program' sequence to the manufacturer's specifications with pre-program (Bit Test) and in-program (Verify) device tests.

### 5.3 PRE-PROGRAM BIT TEST

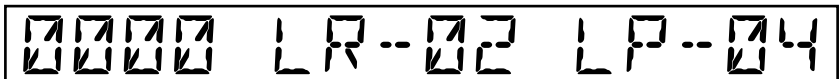
The PP39 automatically checks that the pattern already within the device is able to be programmed with the intended data from the RAM.

If a device were to fail a bit test the display would show:

(i) The first location where a discrepancy occurs; (ii) The RAM data at that location; (iii) The PROM data at that particular location.

For instance:

BOTH ZIF SOCKETS IN USE



LOCATION  
(ZERO)

LEFT RAM (02)

LEFT PROM (04)

In this example the device in the left hand ZIF socket has failed. It contains the data '04' compared to the RAM data '02'. The discrepancy occurs at location 0000 ZERO.



## 5.4 PROGRAMMING

Once the device has passed the bit test, programming of that device will start.

To provide an indication of how far programming has progressed at any given time the address being programmed is simultaneously displayed, for example:

COUNTER

PROGRAMMING 0078

FOUR DIGIT ADDRESS

In the case of the larger devices which use a fast algorithm only the two most significant digits of the address are displayed.

COUNTER

PROGRAMMING 2A

TWO MOST SIGNIFICANT  
DIGITS OF THE ADDRESS

If the data to be programmed into a particular location is the same as the unblown state of that device, the programming sequence will automatically skip to the next location. This function speeds up programming considerably where large sections of the device are to remain empty.

At the end of programming an automatic verify check is done on the whole device. If 'device data' and 'RAM data' are identical the display will show:

PASS VERIFY PASS

LEFT ZIF SOCKETRIGHT ZIF SOCKET

If at any time during programming the EXIT key is pressed programming will stop and a verify within the selected address limits of the device will be done.

Should the PROGRAMMING fail, by pressing 'program' again the PROGRAMMING function will continue from the next location after the failure.

NOTE: PROGRAMMING THE MOTOROLA 2816.

To program data into a specific location in a MOTOROLA 2816 requires the location to be in the EMPTY state.

For instance:

If new data is to be programmed into the device at a previously unprogrammed location, then PROGRAMMING can be carried out in the normal manner.

If, however, it is required to program new data into a location that has already been programmed then the device will have to be set to the EMPTY state prior to programming.

To do this:

Press SET FE

After the data within the 2816 has been erased, the programmer will carry out an 'EMPTY' test. If the device is empty the message 'PASS EMPTY' will be shown on the display and programming can be carried out.



## 5.5 IN-PROGRAM VERIFY

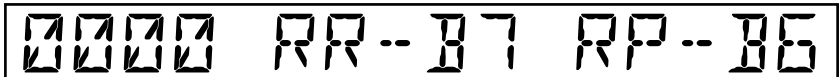
This is a feature whereby each location, as it is programmed, is checked to see that it is identical to the corresponding data byte in the RAM.

If a device were to fail, the display would show:

- (i) the first location where a discrepancy occurs
- (ii) the RAM data at that location
- (iii) the PROM data at that location

For instance:

BOTH ZIF SOCKETS IN USE



LOCATION  
(ZERO)

RIGHT RAM (B7)

RIGHT PROM (B6)

In this example the device in the right hand ZIF socket has failed. It contains the data B6 compared to the RAM data B7. The discrepancy occurs at location 0000 (ZERO).

### MANUAL VERIFY

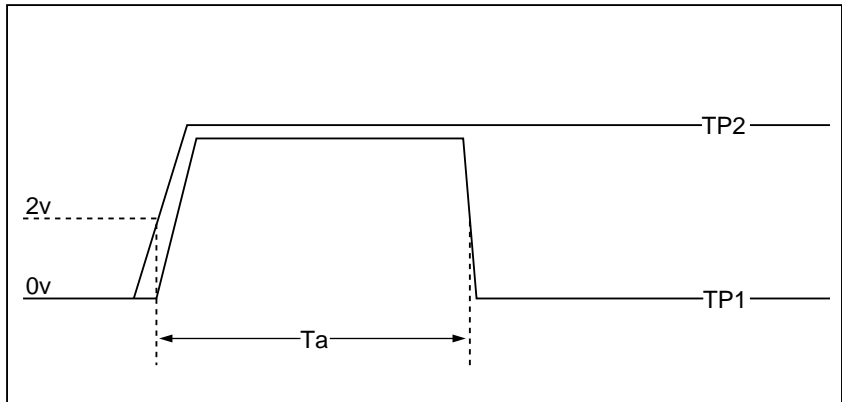
By pressing the 'verify' key a manual verify can be applied at any time. Continually pressing 'Verify' will allow the whole device to be tested and each successive failure will be displayed.



## 5.6 ACCESS TIME TESTING

The Access time test verifies device data against RAM data, where the device data is read by the PP39 a preset delay after the device address lines have changed. If the delay is adjusted until the device just passes then the delay is a direct measure of the access time of the device.

To use this facility on the PP39 an oscilloscope is required to display the outputs from the module terminals TP1 and TP2. The scope should be set up using the access time calibrate function SET F7. Adjust the scope (with a timebase of 100 ns) to give the display as shown below.



The access time testing feature may be used in two ways:

1. Direct measurement of the access time of a device
2. The screening of devices to ensure they meet a predetermined access time.

### Direct Measurement

Having checked that the device will pass VERIFY and, that the delay pot on the module is turned fully counter-clockwise to point 'min', press 'SET 7', the device will fail. Continue pressing SET 7 turning the delay pot clockwise until the device passes.

If SET F7 is now pressed with the scope set up as described above, the access time may be easily read as  $T_a$ .

### Screening of Devices

Ensure the RAM contains the correct device data. Using the SET F7 function, calibrate  $T_a$  using the delay pot on the module to the desired maximum access time. Devices may now be tested using SET 7.



## 5.7 CHECKSUM and CRC

To do a checksum press the c'sum key.

When the programmer is configured to the Gang Mode the display will show a single checksum for both devices for example:



CSUM            5E86

In any other mode two checksums will be displayed whether one or two devices are in use for example:



9F4E    CSUM    1F58

### CYCLIC REDUNDANCY CHECK (CRC)

Cyclic Redundancy Check applies a continuous process of shifting and addition to the PROM data. This yields a coded representation of the data which is sensitive to the ordering of the data bytes unlike checksum which only considers their value.

By pressing the SET 8 when two devices are in use the display will show:



018A    CRC    50C0

As with checksum the CRC function can distinguish between different modes and sockets that are not in use therefore the display will follow a similar format.

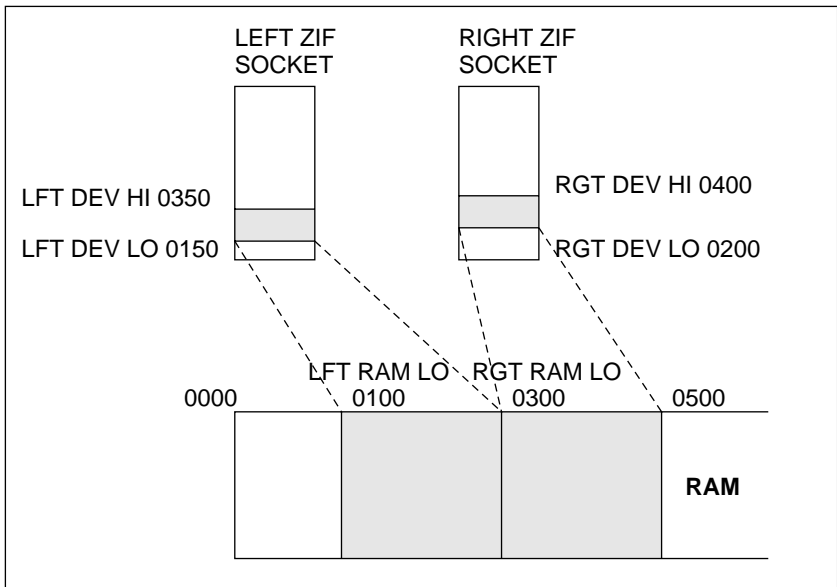


## 5.8 DEVICE/RAM ADDRESS LIMITS (SET F6)

All functions of the PP39 which operate on a device or devices have 6 associated parameters which may be altered by the user. Additionally CRC and checksum which operate on the RAM have their address limits defined by these same 6 parameters.

Address limits for both devices and RAM form these parameters.

Examples of the address limits for the two devices and the RAM can be shown in diagram form:



There are two address limits which can be selected for a single device, these are called Address Low (0150) and Address High (0350). When two devices are in use this figure becomes four as an Address Low (0200) and an Address High (0400) can be specified on the second device.

The RAM has two Address Lows. The left RAM Low (0100) corresponds to the left device and the right RAM Low (0300) corresponds to the right device.

(The RAM has no high addresses as data loaded or programmed will automatically default to the size of the data block specified within the device or devices at the start address pre-selected within the RAM.

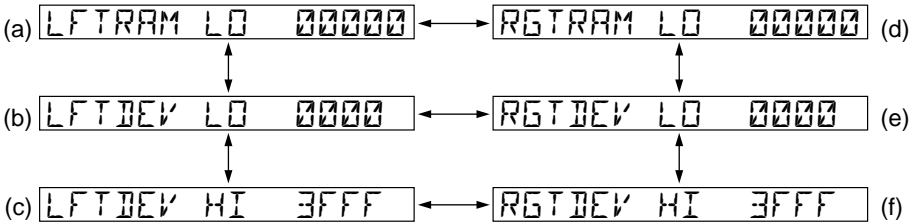
## SETTING UP THE RAM AND THE DEVICE ADDRESS LIMITS

To set address limits press set F6  
The display will first show:



By using the up or down cursor keys, the 3 address limits available for the RAM and device in the left hand ZIF sockets will be displayed. By pressing the right hand cursor key and then the up and down cursor keys the 3 address limits available for the RAM and the device in the right hand socket will be displayed. The left and right hand cursor keys will allow interchange between both devices.

The initial displays show all 6 parameters in the \* default state:



\* The default state in the 8 Bit mode differs from all other modes.  
The 'Right RAM Low' defaults to the device size plus 1. For example:





- (a) This display shows the left RAM low which has defaulted to ZERO. An offset can be selected by use of the hex-keyboard for example 00100:

LFT RAM LO 00100

- (b) By pressing the down cursor key the left device low will be displayed defaulted to ZERO also. A lower address limit can be selected by use of the hex-keyboard for example 0150.

LFT DEV LO 0150

- (c) By pressing the down cursor key again the left device high will be displayed this time defaulted to the size of the device (for example a 27128 device has a capacity of 3FFF). A new upper address limit can be selected by use of the keyboard for example 0350.

LFT DEV HI 0350

- (d), (e) and (f) The user can select address limits for the right device and RAM in the same manner for example:

RGT RAM LO 00300

RGT DEV LO 0200

RGT DEV HI 0400

### **32-BIT MODE: ADDITIONAL PARAMETERS**

In both 32-Bit Modes, two more parameters become available, these select which byte of the word is programmed into which socket.

The parameters are called LEFT BYTE and RIGHT BYTE. They are selected in the same manner as the other parameters.

The default values of these parameters for 32-Bit Mode Hi are:

LEFT BYTE 1                      RIGHT BYTE 2

The default values of these parameters for 32-Bit Mode Lo are:

LEFT BYTE 3                      RIGHT BYTE 4

Any of the four default values can be changed; by using the hex-keys 1, 2, 3 or 4.

## 5.9 SAVE AND RECALL MACHINE CONFIGURATIONS

### 'SAVE' Machine Configurations

Up to 9 different pre-set configurations can be saved in the machine for recall later. Therefore different users can protect their pre-set conditions and recall them later. To save a set of parameters press 'Set A1'. The commands for the 9 sets of configurations are 'Set A1' through to 'Set A9' inclusive.

### 'RECALL' Machine Configurations

Press 'Set B1' to recall previous pre-set configurations saved with A1. Similarly for other recall configurations B2 to B9.

### List of Save and Recall Parameters

(1) **SET F6 RAM/Device Address Limits:**

LEFT RAM LOW  
LEFT DEVICE LOW  
LEFT DEVICE HIGH  
RIGHT RAM LOW  
RIGHT DEVICE LOW  
RIGHT DEVICE HIGH

(2) **SET 1 Interface Parameters:**

FORMAT  
BAUD RATE  
WORD LENGTH  
NUMBER OF STOP BITS  
PARITY

(3) **SET 0 Device Type Selection\*** See list of devices and device codes for the 39M100 Mod.

(4) **SET 3 Bit Mode:**

GANG  
8-BIT  
16-BIT  
32-BIT LOW  
32-BIT HIGH

(5) **SET INPUT, SET OUTPUT I/O Offset and Address Limits:**

INPUT OFFSET  
OUTPUT OFFSET  
OUTPUT START ADDRESS  
OUTPUT STOP ADDRESS



## **SECTION 6**



## **RAM FUNCTIONS**

### **6.1 INTERLACE\***

Interlace\* is the operating concept embodied within the PP39's software which allows easy handling of 8, 16 and 32-bit data.

Previously, when loading 16 or 32-bit data into a programmer RAM the 'split' function had to be employed to re-arrange the data into a suitable form for programming PROMs.

In the 16-bit mode the 'split' function takes a specified block of RAM data and manipulates it to form two new blocks half the size of the original. One will contain all the odd-addressed bytes from the original block and the other will contain all the even-addressed bytes.

For 32-bit data this 'split' operation would have to be done twice, forming four new blocks of data. Each block would be one quarter of the size of the original block.

To reform the original block of data the inverse function of 'split', shuffle is used. On the PP39 neither of these functions is required to configure the RAM. The software in the PP39 removes the need for these functions by making the apparent RAM word-size selectable as either 8, 16 or 32 bits.





## 6.2 LIST AND EDIT

The data field displayed will vary depending upon which machine configuration is in use.

- (a) 8-Bit mode and Gang mode will display 1 byte of data (2 characters).
  - (b) 16-Bit mode will display 2 bytes of data (4 characters).
  - (c) 32-Bit mode low and high will display 4 bytes of data (8 characters).
- For convenience the examples used in the edit routines section will be in the 8 bit mode.

List, Edit, Insert and Delete are integrated functions.

### LIST

This is a feature enabling the data content of the RAM to be scanned on the display. Without the danger of changing the RAM data.

This can be selected by pressing the list key: the first address will be displayed with the data within the first address.

FOR EXAMPLE:

LOCATION (ZERO)	DATA
00000	FF

The address can be scanned in two ways:

1. By use of the cursor keys:



- (a) By using the right/left cursor keys the address can be incremented or decremented a single location at a time.
  - (b) By using the up/down cursor keys the address can be incremented or decremented  $16_{10}$  locations at a time.
2. Any address within RAM limits can be directly entered by use of the hex-keyboard.

For example:

SELECTED ADDRESS	DATA
01FF0	29

## EDIT

This is a feature whereby the actual content of the RAM can be directly modified by using the keyboard.

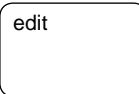
The edit mode can be selected in two ways.

- (a) By pressing the edit key when the machine is in the normal operating mode.
- (b) By pressing the edit key when the machine is in the list mode. (The list mode can be reselected in the same manner).

When switching from the list to the edit mode or vice versa the address and data being displayed will be unaffected.

For example:

	LOCATION	DATA
LIST	01FF0	29



	LOCATION	E DENOTES EDIT	DATA
EDIT	01FF0	E	29

The data '29' at location '01FF0' can now be changed by use of the hex keyboard into for instance A3:

	LOCATION	NEW DATA
	01FF0	E A3

As with 'list' the data can be scanned by use of the cursor keys; when selection of address is made the information can again be changed by use of the hex-keyboard.

Alternatively and usually more quickly an address can be directly entered by switching back to the 'List mode' and using the hex-keyboard to select the location. Switching back to the Edit mode will not corrupt this information.

### 6.3 INSERT

Insert is part of the edit mode and can be selected by pressing the edit key once, when the machine is in the edit mode.

Information can be inserted into a particular location within the RAM. The existing data content in and above the selected address is repositioned one location higher. Apart from this shift in location the existing data remains the same.

For example:

LOCATION      I DENOTES INSERT      DATA

01FF0	I	29
-------	---	----

By pressing the SET key all data inclusive of location 01FF0 and above is repositioned one location higher:

NEXT LOCATION UP

01FF1	I	29
-------	---	----

Having pressed the set key, '00' will be inserted into the selected address.

01FF0	I	00
-------	---	----

By use of the hex-keyboard the chosen data can now be inserted for instance A6:

01FF0	I	A6
-------	---	----

Other than the user of the set key, operation in the Insert mode remains the same as when in the ordinary edit mode.

For graphic example see next page.

**A GRAPHIC EXAMPLE OF HOW THE INSERT FUNCTION WORKS IS SHOWN BELOW:**

**INITIAL STATUS:**

	01FED	01FEE	01FEF	01FF0	01FF1	01FF2	01FF3
RAM DATA	6 C	8 8	4 0	2 9	E 3	7 9	F 3

CURRENTLY DISPLAYED LOCATION

By pressing the SET key all data inclusive of location 01FF0 and above is repositioned one location higher. At the displayed location, '00' will be automatically inserted:

	01FED	01FEE	01FEF	01FF0	01FF1	01FF2	01FF3
RAM DATA	6 C	8 8	4 0	0 0	2 9	E 3	7 9

CURRENTLY DISPLAYED LOCATION

DATA REPOSITIONED ONE LOCATION HIGHER

By use of the hex-keyboard the chosen data A6 can be entered at location 01FF0:

	01FED	01FEE	01FEF	01FF0	01FF1	01FF2	01FF3
RAM DATA	6 C	8 8	4 0	A 6	2 9	E 3	7 9

'A6' ENTERED

CURRENTLY DISPLAYED LOCATION

## 6.4 DELETE

Delete is also part of the edit mode and can be selected by pressing the edit key twice when the machine is in the edit mode. Delete is the opposite function to insert whereby data is removed 'from' a particular location.

The data above the selected deletion address is repositioned one location lower.

For example: 5B is the data to be deleted.

LOCATION    **D DENOTES DELETE**    DATA

00200	D	5B
-------	---	----

By pressing the SET key all data above but 'not' inclusive of location 00200 is automatically brought down one location. The information previously at address 00201 replaces 'Data 5B' at location 00200.

For example:

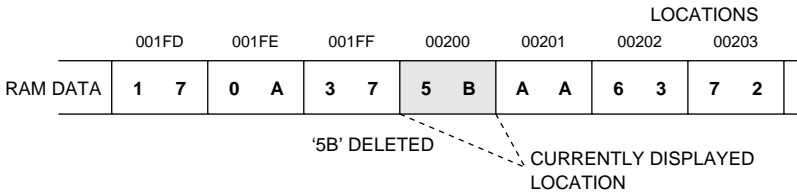
00200	D	AA
-------	---	----

Other than the use of the set key, operation in the delete mode remains the same as when in the ordinary edit mode.

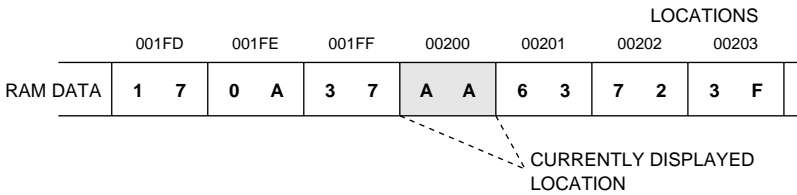
For graphic example see next page.

**A GRAPHIC EXAMPLE OF HOW THE DELETE FUNCTION WORKS IS SHOWN BELOW:**

**INITIAL STATUS:**



By pressing the SET key all data above the displayed location 00200 is brought down one location. (All data below the displayed location is left unaffected).



## 6.5 BLOCK MODE (SET F4)

### SETTING ADDRESS LIMITS

This is a feature enabling a block of data with pre-selected address limits to be relocated at another address within the RAM, without destroying the original data.

Selection of this function is made by pressing SET F4.

The display will show:

ADDRESS LOW	ZERO
A00R LO	00000

This defines the lower limit of the block in RAM to be re-located.  
(Defaults to 0000)

The new lower RAM limit can be entered using the hex-keyboard.

For example 00100:

	NEW LOWER RAM LIMIT
A00R LO	00100

If the down cursor is pressed the display will show:

ADDRESS HIGH	SIZE OF SELECTED DEVICE
A00R HI	03FFF

This defines the upper limit of the block in RAM to be relocated.  
(Defaults to selected device size).

A new value for the upper RAM limit can be entered using the hex-keyboard.

For example 00300:

	NEW UPPER RAM LIMIT
A00R HI	00300

## LOWER LIMIT OF RE-LOCATED DATA

By pressing the down cursor key again the display will show:

TO ADDRESS



TO ADDR 00000

This defines the lower RAM limit of where the block of data is to be re-located (Defaults to 0000).

The re-located lower RAM limit can be entered using the hex-keyboard.

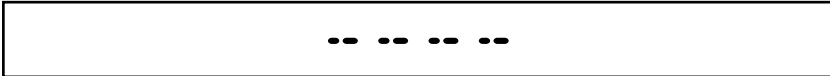
For example 00500:

LOWER LIMIT OF THE  
NEW BLOCK OF DATA



TO ADDR 00500

Pressing the exit key will initiate the block-move function. A series of dashes will be displayed indicating the function is in progress:



---

The PP39 will automatically return to the normal operating mode.







## 6.6 FILLING THE RAM

By pressing SET FF the RAM will be entirely filled with F's.

By pressing SET F0 the RAM will be entirely filled with 0's (Zeros).

By pressing SET 5 the RAM data will be complemented. (1's complement).

### FILLING THE RAM WITH AN ARBITRARY VARIABLE\* (SET F2)

This function enables the user to fill the RAM with an arbitrary variable of their own choosing.

The variable will be identically repeated at every word within address limits specified by the user.

\*The variable can be of differing word length depending on which machine configuration is in use, i.e.:

2 Characters (1 Byte)

Can be used in the 8-bit mode and Gang mode.

4 Characters (2 Bytes)

Can be used in the 16-bit mode.

8 Characters (4 Bytes)

Can be used in the 32-bit mode low and high.

For convenience the example used will stay in the 8-bit mode.

Pressing SET F2 will display the lower address limit, which defaults to ZERO:

ADDRESS LOW	LOCATION ZERO
A000	L0 000000

The new lower address limit can be selected by using the hex-keyboard for instance 00600:

LOCATION

ADDR LO 00600

The upper address limit can be shown by pressing the up cursor key, this also defaults to ZERO:

ADDRESS HIGH                      LOCATION ZERO

ADDR HI 00000

The new upper limit can be selected using the hex-keyboard for instance 01000:

LOCATION

ADDR HI 01000

The arbitrary variable can be entered by pressing the up-cursor again to display.

DATA 00

The data selection can be made by using the hex-keyboard for instance A1:

ARBITRARY VARIABLE

DATA A1

Pressing 'SET' alone will implement this selection.

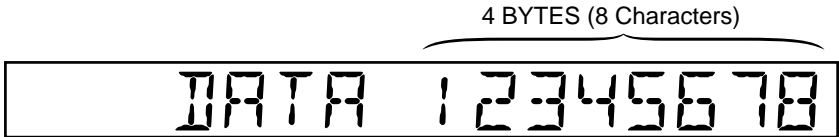
Every byte of RAM within and inclusive of the specified address limits of 00600 low to 01000 high is filled with 'A1'.

\*A larger number of up to 8 digits can be used to fill the RAM in 16 or 32 bit mode. This facility can be used in conjunction with modes of smaller word size.

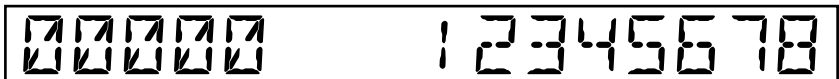
For instance with the 8-bit mode:

Selection of arbitrary variable made in 32-Bit Mode

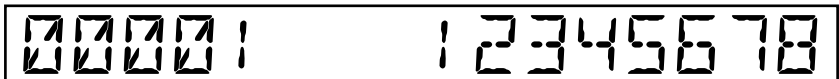
For example:



The display using LIST at location 00000 and 00001 will be:



and





## 6.7 STRING SEARCH

This function allows the RAM data to be searched for a particular string of data.

Press SET 9 to display:



The lower address limit of the area of RAM to be searched is now displayed defaulted to zero. It can be altered using values input from the keyboard.

To display the upper limit:

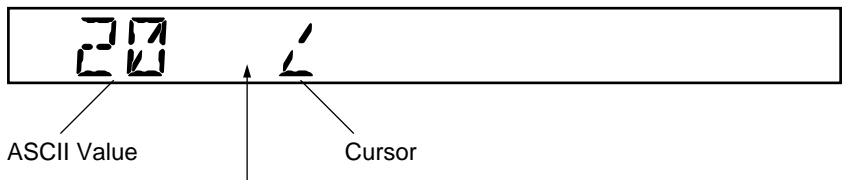
Press ↑ or ↓



The upper address limit is shown defaulted to the size of the pre-selected device, and like the lower limit it can be altered using values input from the keyboard.

Once the address limits have been set:

Press SET to display:



The default state of the string is now shown. To the extreme left of the display is the ASCII code equivalent of the character displayed on the immediate left of the cursor. In this case the space character is displayed.

To increment or decrement the ASCII value and hence alter the character displayed:

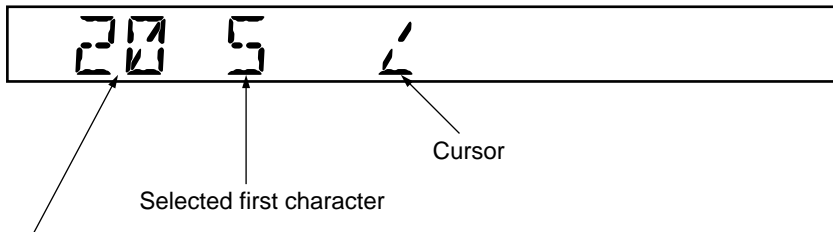
Press ↑ or ↓

Alternatively and quicker, the ASCII value if known can be entered directly from the keyboard.

Note: Due to the limitations of the display some of the characters cannot be represented accurately. Their value will however remain valid.

To move the cursor one space to the right and allow selection of the next character:

Press →



ASCII value of character to immediate left of the cursor (in this case 'space')

The second character can now be selected in the manner previously described. In this way a string of upto 11 characters (or data bytes) can be entered.

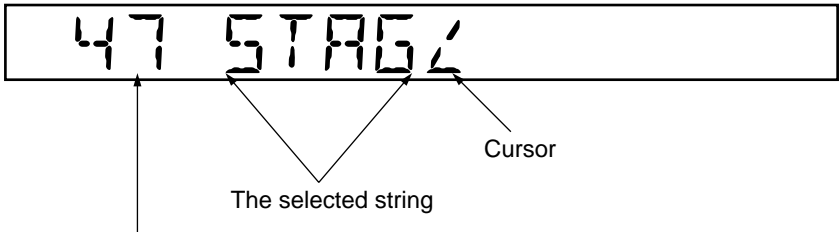
When the desired string has been selected, to implement the String Search:

Press SET

If a corresponding string is located within the specified area of RAM, then the message 'FOUND AT' and the address of the first occurrence will be displayed. Every subsequent occurrence can be located by continually pressing SET until the entire specified area of RAM has been searched.

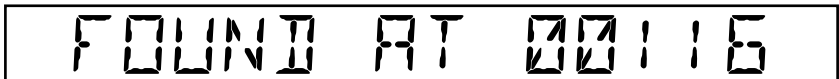


For instance:



The ASCII value of the character to the immediate left of the cursor. (In this case 'G')

The above string was searched for and the display showed the following message:



This means that the first occurrence of the string was found at location 00116.

If the string had not been found within the specified area of RAM the display would have shown:



If a string has been entered and only part of it is used, then moving the cursor to the left will restrict the string to the desired length. The original string will be retained however in its entirety, and moving the cursor to the right will display it again.

Any entered string will be retained until the PP39 is powered down.

To abort the String Search at any time.

Press EXIT



## **SECTION 7**



**39M200**













### 7.3 LIST OF 'SET' COMMANDS

set 0	Allows user to scan and select various manufacturers and device types
set 1	Selects interface parameters Format, Baud Rate, Word Length, Stop Bits, Parity
set 2	Sets programmer into 'Remote' control. (To return to Local Mode: Power up with exit key depressed)
set 4	Displays RAM size in hexadecimal
set 5	RAM data complemented from lower to upper address limit
set 6	Displays module software revision if module is plugged in, or main frame software revision, if no module is plugged in.
set 8	Calculates and displays CRC (Cyclic Redundancy Check)
set A1	To A9 Saves machine configuration (up to nine sets)

## LIST OF 'SET' COMMANDS (continued)

set B1      To      B9      Recalls previously saved machine configurations (up to 9 sets)

set F0      Fills entire RAM with 00

set FF      Fills entire RAM with FF

set F1      Audible Alarm: To indicate end of program, test, or as a warning using a combination of bleeps and tones. SET F1 both enables and disables this function.

set F2      Fills RAM with arbitrary variable from lower to upper address limit

set F3      Set Security fuse: the fuse can be set to 'BLOWN' on 'INTACT' by use of the up/down arrow keys.

set F4      Re-Locate RAM data. A block of data with pre-selected address limits can be copied and then re-located at another address within the RAM.

set F6      Defines RAM and device address ranges for all functions which operate on the device

set input      Input – Enters input address offset

set output      Output – Enters output address offset, start address and stop address

## **SECTION 8**



## SELECTING A DEVICE

### 8.1 DEVICE TYPE SELECTION

#### Selecting the device using a 4 digit code

The complete range of devices supported by the 39M200 is stored in the module. Each individual device has its own four digit code. (See device code list Section 8.2).

SET 0 – Allows code selection

SEQUENCE: Prior to SET '0' the display will show the last entered configuration

For example:



A 7-segment display showing the text 'AMD 9761H' in a monospaced font. The 'A' is formed by segments 1, 2, 3, 4, 5, and 7. The 'M' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The 'D' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The space is formed by segments 1, 2, 3, 4, 5, 6, and 7. The '9' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The '7' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The '6' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The '1' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The 'H' is formed by segments 1, 2, 3, 4, 5, 6, and 7.

By pressing SET '0' the device code of this configuration will be displayed:




A 7-segment display showing the text 'DEVICE CODE 9F8F' in a monospaced font. The 'D' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The 'E' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The 'V' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The 'I' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The 'C' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The 'O' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The space is formed by segments 1, 2, 3, 4, 5, 6, and 7. The '9' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The 'F' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The '8' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The 'F' is formed by segments 1, 2, 3, 4, 5, 6, and 7.

When the new device code to be entered is already known, (for instance 7F91 is the code for a Motorola 68701), then the 7F91 can be entered directly onto the display from the keyboard replacing the old code:



A 7-segment display showing the text 'DEVICE CODE 7F91' in a monospaced font. The 'D' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The 'E' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The 'V' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The 'I' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The 'C' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The 'O' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The space is formed by segments 1, 2, 3, 4, 5, 6, and 7. The '7' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The 'F' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The '9' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The '1' is formed by segments 1, 2, 3, 4, 5, 6, and 7.

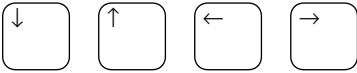
The selection sequence can be completed by pressing EXIT whereby the new manufacturer and device type are displayed along with the bit mode:



A 7-segment display showing the text 'MOT 68701' in a monospaced font. The 'M' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The 'O' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The 'T' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The space is formed by segments 1, 2, 3, 4, 5, 6, and 7. The '6' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The '8' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The '7' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The '0' is formed by segments 1, 2, 3, 4, 5, 6, and 7. The '1' is formed by segments 1, 2, 3, 4, 5, 6, and 7.

## Scanning device types and manufacturers by use of cursor keys

When a device code is not known or if the user wishes to scan the devices available, selection can be made via the cursor keys:

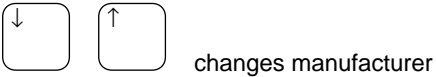


By pressing SET '0' the code of the last used device is displayed:

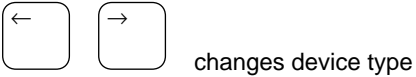


The manufacturer and device type can be changed by use of the cursor keys:

The up/down keys scan the range of manufacturers.



The left/right keys scan the device range of a particular manufacturer.





## 8.2 LIST OF DEVICES AND DEVICE CODES FOR THE 39M200 MODULE

This list of parts is supported by the 39M200 module and is stored in the module's software. Every programmable part carries a four digit code. The first two digits define the manufacturer of the device while the second two digits describe the device type.

<b>Manufacturer</b>	<b>Device</b>	<b>4 Digit Device Code</b>	<b>Device Size (Hex)</b>
AMD	8751H 9761H	9F 8D 9F 8F	1000 2000
INTEL	8741A 8742 8748 8748H 8749H 8744 8751 8751H 8755	6F 81 6F 82 6F 86 6F 88 6F 8B 6F 84 6F 8C 6F 8D 6F 8E	400 800 400 400 800 1000 1000 1000 800
MOTOROLA	68701 6870104 68705P3 68705P5 68705U3 68705U5 68705R3 68705R5	7F 91 7F 93 7F 95 7F 96 7F 98 7F 99 7F 9A 7F 9B	800 1000 800 800 1000 1000 1000 1000
NEC	8741A 8748 8748H 8749H 8755A	CC 81 CC 86 CC 88 CC 8B CC 8E	400 400 400 800 800



## **SECTION 9**



## DEVICE FUNCTIONS

### 9.1 LOAD

#### Loading the RAM from the Device

Insert the device into the ZIF socket.

Press LOAD

On completion of LOAD the display will show:



**Note:** It is impossible to load the RAM from a Motorola 68705 device or from a device which has had its security fuse blown. (See section 5.7).




## 9.2 PROGRAMMING SEQUENCE

### Empty Test

If required an 'empty test' can be applied to the device in the ZIF socket prior to programming. To do this:

Press 'empty'

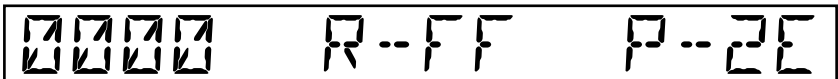
The device will be examined for the unprogrammed state (FF). If it is empty the display will show:



EMPTY PASS

If the device fails the 'empty test' the display will show:

i. The first location where a discrepancy occurs, ii. The unblown state of the selected device, iii. The EPROM data at that particular location:



0000 R--FF P--2E

LOCATION  
(ZERO)

UNBLOWN  
STATE (FF)

DEVICE  
CONTAINING DATA (2E)

In this example the device has failed the test because it contains the data '2E' and not the unblown state 'FF'. This discrepancy occurs at the first location – 0000 (Zero).

If the empty test passes or is unnecessary the programming can begin.

Pressing the program key will automatically execute the 'program' sequence to the manufacturer's specifications with pre-program (Bit Test) and in-program (Verify) device tests.

### PRE-PROGRAM BIT TEST

The PP39 automatically checks that the pattern already within the device is able to be programmed with the intended data from the RAM.

If a device were to fail a bit test the display would show:

(i) The first location where a discrepancy occurs; (ii) The RAM data at that location; (iii) The EPROM data at that particular location.

For instance:

0000	R--02	P--04
LOCATION (ZERO)	RAM (02)	PROM (04)

In this example the device has failed the test because it contains the data '04' compared to the RAM data '02'. The discrepancy occurs at location 0000 ZERO.



### 9.3 PROGRAMMING

Once the device has passed the bit test, programming of that device will start.

To provide an indication of how far programming has progressed at any given time the address being programmed is simultaneously displayed, for example:

COUNTER



FOUR DIGIT ADDRESS

In the case of the larger devices which use a fast algorithm only the two most significant digits of the address are displayed.


COUNTER



TWO MOST SIGNIFICANT  
DIGITS OF THE ADDRESS

If the data to be programmed into a particular location is the same as the unblown state of that device, the programming sequence will automatically skip to the next location. This function speeds up programming considerably where large sections of the device are to remain empty.

At the end of programming an automatic verify check is done on the whole device. If 'device data' and 'RAM data' are identical the display will show:



If at any time during programming the EXIT key is pressed programming will stop and a verify within the selected address limits of the device will be done.



### 9.3.1 Verify

#### Verify Pass-Security Bit

If the device has passed the verify test and the security bit was set to 'BLOWN', the display will show:



#### Verify Fail

Should the device fail the verify test, the security bit will not be blown.

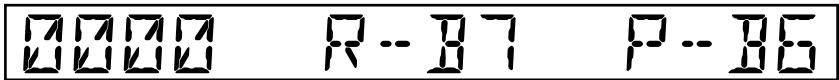
#### IN-PROGRAM VERIFY

A feature whereby each location programmed is immediately checked to see that is identical to the corresponding data byte in RAM.

If a device were to fail a verify, the display would show:

(i) The first location where a discrepancy occurs; (ii) The RAM data at that location; (iii) The 'EPROM' data at that particular location.

For instance:



LOCATION  
(ZERO)

RAM (B7)

PROM (B6)

In this example the device contains the data B6 compared to the RAM data B7. The discrepancy occurs at location 0000 (ZERO).

#### MANUAL VERIFY

By pressing the 'verify' key a manual verify can be applied at any time.

Automatic and manual verify operations are identical.

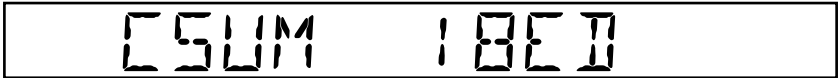


#### 9.4 CHECKSUM

To do a checksum

Press 'C/Sum'

The display will show:



CSUM 18E0

#### CYCLIC REDUNDANCY CHECK (CRC)

The Cyclic Redundancy Check applies a continuous process of shifting and addition to the RAM data. This yields a coded representation of the data which is sensitive to the ordering of the data bytes unlike the checksum, which only considers their values.

Press 'SET 8'

The display will show:



CRC 6C90

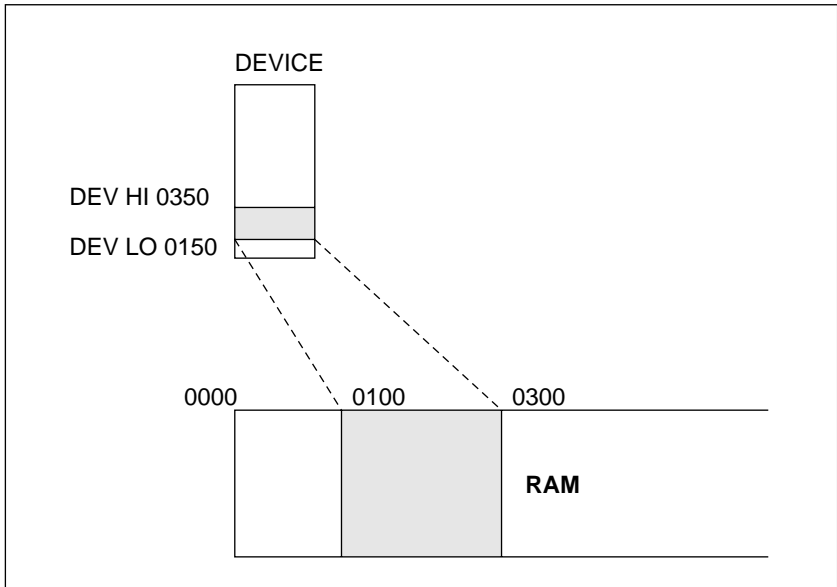


## 9.5 DEVICE/RAM ADDRESS LIMITS (SET F6)

All functions of the PP39 which operate on a device have 3 associated parameters which may be altered by the user. Additionally CRC and checksum which operate on the RAM have their address limits defined by these same 3 parameters.

Address limits for both devices and RAM form these parameters.

Examples of the address limits for the device and the RAM can be shown in diagrammatic form:



There are two address limits which can be selected for a single device, these are called Address Low (0150) and Address High (0350).

The RAM has an Address Low but no Address High. Data loaded or programmed will automatically default to the size of the data block specified within the device at the start address pre-selected within the RAM.

## SETTING UP THE RAM AND THE DEVICE ADDRESS LIMITS

To set address limits:

Press: Set F6

The display will show:



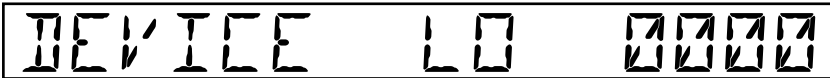
A rectangular display box containing the text "RAM LO" followed by a space and then "000000". The text is in a monospaced, digital font.

The RAM Low defaults to zero, but an offset can be selected by use of the hex-keyboard, for example 00100:



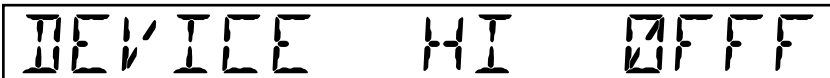
A rectangular display box containing the text "RAM LO" followed by a space and then "001000". The text is in a monospaced, digital font.

By using the up/down arrow keys, the 2 address limits available for the device can be displayed.

(a) 

A rectangular display box containing the text "DEVICE LO" followed by a space and then "000000". The text is in a monospaced, digital font.

The address limit defaults to zero, but can be altered by use of the hex-keyboard.

(b) 

A rectangular display box containing the text "DEVICE HI" followed by a space and then "0FFFF". The text is in a monospaced, digital font.

The address limit defaults to the size of the device, but can be altered by use of the hex-keyboard.



## 9.6 SAVE AND RECALL MACHINE CONFIGURATIONS

### 'SAVE' Machine Configurations

Up to 9 different pre-set configurations can be saved in the machine for recall later. Therefore different users can protect their pre-set conditions and recall them later. To save a set of parameters:

Press 'Set A1'

The commands for the 9 sets of configurations are 'Set A1' through to 'Set A9' inclusive.

### 'RECALL' Machine Configurations

Press 'Set B1'

To recall previous pre-set configurations saved with A1. Similarly for other recall configurations B2 to B9.

### List of Save and Recall Parameters

(1) **SET F6 RAM/Device Address Limits:**

RAM LOW  
DEVICE LOW  
DEVICE HIGH

(2) **SET 1 Interface Parameters:**

FORMAT  
BAUD RATE  
WORD LENGTH  
NUMBER OF STOP BITS  
PARITY

(3) **SET 0 Device Type Selection**

\*See list of devices and device codes  
for the 39M200 Module.

(5) **SET INPUT, SET OUTPUT I/O Offset and Address Limits:**

INPUT OFFSET  
OUTPUT OFFSET  
OUTPUT START ADDRESS  
OUTPUT STOP ADDRESS



## 9.7 68705 Devices

The 68705s are self-programming, WRITE ONLY devices. This means that they cannot be loaded, bit checked, empty checked or verified. They can only program themselves with data stored in the module's static RAMs followed by a self verify. As a result, the device functions for this family differ from those described earlier in the section.

### **Load, Empty, Verify**

On pressing any one of these function keys, the message 'Not Applicable' will be displayed, and the warning failure 'Bleep' will sound.

### **Pre-Program Bit Test, In-Program Verify**

These functions are inoperative for 68705 devices.

### **Programming**

No running count is displayed while programming is in progress.

### **Device/RAM Address Limits (SET F6)**

It is possible to alter all three parameters, but only RAM Low actually takes effect.



## **SECTION 10**



## RAM FUNCTIONS

### 10.1 KEYBOARD EDIT ROUTINES

#### LIST

This is a feature enabling the data content of the RAM to be scanned on the display without the danger of changing the RAM data.

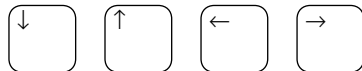
This can be selected by pressing the list key: the first address will be displayed with the data within the first address.

FOR EXAMPLE:

LOCATION (ZERO)	DATA
00000	FF

The address can be scanned in two ways:

1. By use of the cursor keys:



- (a) By using the right/left cursor keys the address can be incremented or decremented a single location at a time.
  - (b) By using the up/down cursor keys the address can be incremented or decremented  $16_{10}$  locations at a time.
2. Any address within RAM limits can be directly entered by use of the hex-keyboard.

For example:

SELECTED ADDRESS	DATA
01FF0	29





## 10.2 EDIT

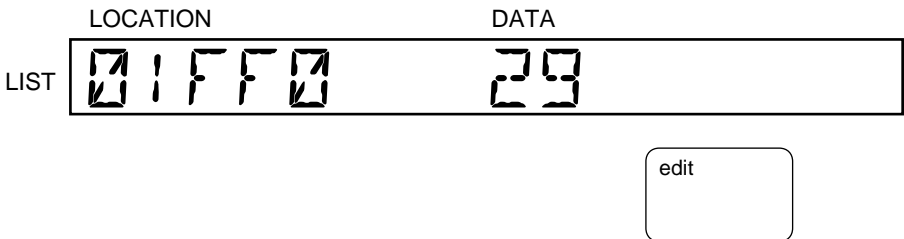
This is a feature whereby the actual content of the RAM can be directly modified by using the keyboard.

The edit mode can be selected in two ways.

- (a) By pressing the edit key when the machine is in the normal operating mode.
- (b) By pressing the edit key when the machine is in the list mode. (The list mode can be reselected in the same manner).

When switching from the list to the edit mode or vice versa the address and data being displayed will be unaffected.

For example:



The data '29' at location '01FF0' can now be changed by use of the hex keyboard into for instance A3:



As with 'list' the data can be scanned by use of the cursor keys; when selection of address is made the information can again be changed by use of the hex-keyboard.

Alternatively and usually more quickly an address can be directly entered by switching back to the 'List mode' and using the hex-keyboard to select the location. Switching back to the Edit mode will not corrupt this information.



### 10.3 INSERT

Insert is part of the edit mode and can be selected by pressing the edit key once, when the machine is in the edit mode.

Information can be inserted into a particular location within the RAM. The existing data content in and above the selected address is repositioned one location higher. Apart from this shift in location the existing data remains the same.

For example:

LOCATION      I DENOTES INSERT      DATA

01FF0	I	29
-------	---	----

By pressing the SET key all data inclusive of location 01FF0 and above is repositioned one location higher:

NEXT LOCATION UP

01FF1	I	29
-------	---	----

Having pressed the set key, '00' will be inserted into the selected address.

01FF0	I	00
-------	---	----

By use of the hex-keyboard the chosen data can now be inserted for instance A6:

01FF0	I	A6
-------	---	----

Other than the user of the set key, operation in the Insert mode remains the same as when in the ordinary edit mode.

For graphic example see next page.

**A GRAPHIC EXAMPLE OF HOW THE INSERT FUNCTION WORKS IS SHOWN BELOW:**

**INITIAL STATUS:**

	01FED	01FEE	01FEF	01FF0	01FF1	01FF2	01FF3
RAM DATA	6 C	8 8	4 0	2 9	E 3	7 9	F 3

CURRENTLY DISPLAYED LOCATION

By pressing the SET key all data inclusive of location 01FF0 and above is repositioned one location higher. At the displayed location, '00' will be automatically inserted:

	01FED	01FEE	01FEF	01FF0	01FF1	01FF2	01FF3
RAM DATA	6 C	8 8	4 0	0 0	2 9	E 3	7 9

CURRENTLY DISPLAYED LOCATION

DATA REPOSITIONED ONE LOCATION HIGHER

By use of the hex-keyboard the chosen data A6 can be entered at location 01FF0:

	01FED	01FEE	01FEF	01FF0	01FF1	01FF2	01FF3
RAM DATA	6 C	8 8	4 0	A 6	2 9	E 3	7 9

'A6' ENTERED

CURRENTLY DISPLAYED LOCATION

## 10.4 DELETE

Delete is also part of the edit mode and can be selected by pressing the edit key twice when the machine is in the edit mode. Delete is the opposite function to insert whereby data is removed 'from' a particular location.

The data above the selected deletion address is repositioned one location lower.

For example: 5B is the data to be deleted.

LOCATION    **D DENOTES DELETE**    DATA

00200	D	5B
-------	---	----

By pressing the SET key all data above but 'not' inclusive of location 00200 is automatically brought down one location. The information previously at address 00201 replaces 'Data 5B' at location 00200.

For example:

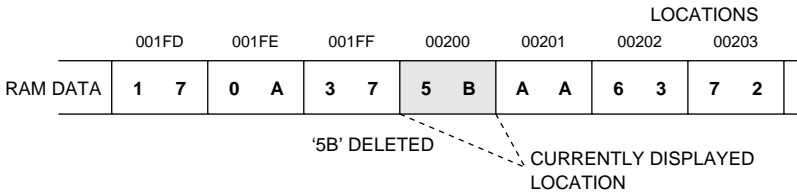
00200	D	AA
-------	---	----

Other than the use of the set key, operation in the delete mode remains the same as when in the ordinary edit mode.

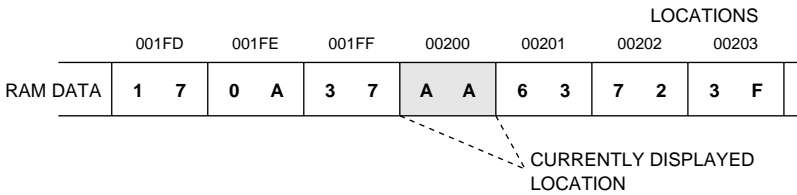
For graphic example see next page.

**A GRAPHIC EXAMPLE OF HOW THE DELETE FUNCTION WORKS IS SHOWN BELOW:**

**INITIAL STATUS:**



By pressing the SET key all data above the displayed location 00200 is brought down one location. (All data below the displayed location is left unaffected).



## 10.5 BLOCK MODE (SET F4)

### SETTING ADDRESS LIMITS

This is a feature enabling a block of data with pre-selected address limits to be relocated at another address within the RAM, without destroying the original data.

Selection of this function is made by pressing SET F4.

The display will show:

ADDRESS LOW	ZERO
A00R LO	000000

This defines the lower limit of the block in RAM to be re-located.  
(Defaults to 0000)

The new lower RAM limit can be entered using the hex-keyboard.

For example 00100:

	NEW LOWER RAM LIMIT
A00R LO	001000

If the down cursor is pressed the display will show:

ADDRESS HIGH	SIZE OF SELECTED DEVICE
A00R HI	03FFF

This defines the upper limit of the block in RAM to be relocated.  
(Defaults to selected device size).

A new value for the upper RAM limit can be entered using the hex-keyboard.

For example 00300:

	NEW UPPER RAM LIMIT
A00R HI	003000

## LOWER LIMIT OF RE-LOCATED DATA

By pressing the down cursor key again the display will show:

TO ADDRESS



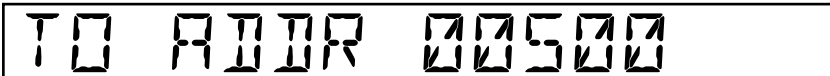
TO ADDR 00000

This defines the lower RAM limit of where the block of data is to be re-located (Defaults to 0000).

The re-located lower RAM limit can be entered using the hex-keyboard.


For example 00500:

LOWER LIMIT OF THE  
NEW BLOCK OF DATA



TO ADDR 00500

Pressing the exit key will initiate the block-move function. A series of dashes will be displayed indicating the function is in progress:



---

The PP39 will automatically return to the normal operating mode.







## 10.6 FILLING THE RAM

By pressing SET FF the RAM will be entirely filled with F's.

By pressing SET F0 the RAM will be entirely filled with 0's (Zeros).

By pressing SET 5 the RAM data will be complemented. (1's complement).

### FILLING THE RAM WITH AN ARBITRARY VARIABLE\* (SET F2)

This function enables the user to fill the RAM with an arbitrary variable of their own choosing.

Pressing SET F2 will display the lower address limit, which defaults to ZERO:

ADDRESS LOW	LOCATION ZERO
ADDR	L0 000000

The new lower address limit can be selected by using the hex-keyboard for instance 00600:

LOCATION

ADDR LO 00600

The upper address limit can be shown by pressing the up cursor key, this also defaults to ZERO:

ADDRESS HIGH                      LOCATION ZERO

ADDR HI 00000

The new upper limit can be selected using the hex-keyboard for instance 01000:

LOCATION

ADDR HI 01000

The arbitrary variable can be entered by pressing the up-cursor again to display.

DATA 00

The data selection can be made by using the hex-keyboard for instance A1:

ARBITRARY VARIABLE

DATA A1

Pressing 'SET' alone will implement this selection.

Every byte of RAM within and inclusive of the specified address limits of 00600 low to 01000 high is filled with 'A1'.

## 10.7 STRING SEARCH

This function allows the RAM data to be searched for a particular string. Press SET 9. The display will show:



The lower address limit for the search can be altered using the hex-keyboard.

To alter the upper address limits:

Press ↑ or ↓

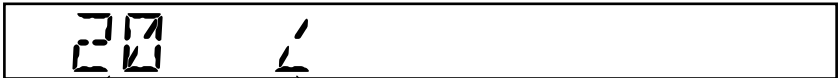
The display will show:



The values will default to the size of the device, but may be altered using the hex-keyboard.

Once the address limits have been set, press SET

The display will show:



ASCII Value

Cursor

On the left hand side of the display appears the ASCII equivalent of the character next to the cursor. (In the example above this is a space).

Press ↑ or ↓

To increment or decrement the ASCII value. The character next to the cursor will change automatically. It is possible to enter the ASCII value directly if known.

Press ← or →

to move the cursor left and right.

Note: Some of the characters cannot be exactly reproduced on the display and hence may be unrecognisable. Their value will however, remain valid.

Press SET

The specified string is searched for and if found, the display will show 'FOUND AT' and the location. If the string search fails, the display will simply show 'NOT FOUND'.

To increment or decrement the ASCII value and hence alter the character displayed:

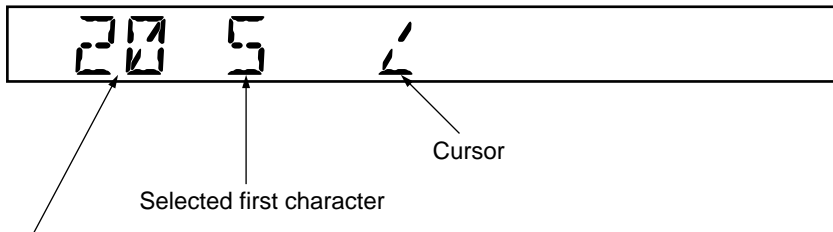
Press ↑ or ↓

Alternatively and quicker, the ASCII value if known can be entered directly from the keyboard.

Note: Due to the limitations of the display some of the characters cannot be represented accurately. Their value will however remain valid.

To move the cursor one space to the right and allow selection of the next character:

Press →



ASCII value of character to immediate left of the cursor (in this case 'space')

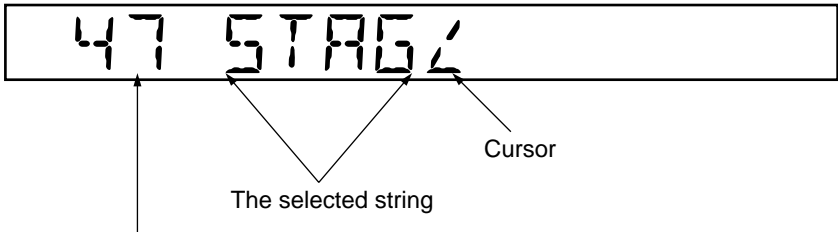
The second character can now be selected in the manner previously described. In this way a string of upto 11 characters (or data bytes) can be entered.

When the desired string has been selected, to implement the String Search:

Press SET

If a corresponding string is located within the specified area of RAM, then the message 'FOUND AT' and the address of the first occurrence will be displayed. Every subsequent occurrence can be located by continually pressing SET until the entire specified area of RAM has been searched.

For instance:



The ASCII value of the character to the immediate left of the cursor. (In this case 'G')

The above string was searched for and the display showed the following message:



This means that the first occurrence of the string was found at location 00116.

If the string had not been found within the specified area of RAM the display would have shown:



If a string has been entered and only part of it is used, then moving the cursor to the left will restrict the string to the desired length. The original string will be retained however in its entirety, and moving the cursor to the right will display it again.

Any entered string will be retained until the PP39 is powered down.

To abort the String Search at any time.

Press EXIT





## **SECTION 11**



## INTERFACE

### 11.1 SETTING THE I/O INTERFACE PARAMETERS

On power-up the I/O defaults to the last used I/O parameter.

This default function is programmed into the Non-Volatile RAM and can be displayed by pressing SET 1.

For instance:

INT	9600	8	2	EP
-----	------	---	---	----

There are five categories of I/O interface parameter available for selection on the PP39 Programmer. These are: Format, Baud Rate, Word Length, Number of Stop Bits and Parity.

They correspond to the display in this manner:

FORMAT	BAUD RATE	WORD LENGTH	No. OF STOP BITS	PARITY
INT	9600	8	2	EP



































## **SECTION 12**



## FORMAT DESCRIPTIONS

### 12.1 INTERFACE FORMATS (INTRODUCTION)

There are eleven formats available on the PP39, these are:

INT	=	INTELLEC
XINT	=	EXTENDED INTELLEC
HASC	=	HEX ASCII
XOR	=	EXORCISOR
XXOR	=	EXTENDED EXORCISOR
TEK	=	TEK HEX
XTEK	=	EXTENDED TEK
PPX	=	STAG HEX *
BIN	=	BINARY
DBIN	=	DEC BINARY
BINR	=	BINARY RUBOUT

#### STANDARD FORMATS

There are three standard manufacturer formats these are: INTELLEC, EXORCISOR and TEK HEX which are used on most development systems.

#### EXTENDED FORMATS

There are three protracted versions of the standard formats these are: EXTENDED INTELLEC, EXTENDED EXORCISOR and EXTENDED TEK. The extended formats can be used when a larger address field is required.

#### HEX ASCII

The Hex ASCII format is the original base version of the standard formats. It lacks the facility of an address field and a checksum.

#### PPX (Stag Hex) \*

The PPX format differs from the HEX ASCII in that it has an address field and terminates with a checksum of total bytes.

#### BINARY

The Binary format is the most fundamental of all formats and can be used where fast data transfers are required. It has no facility for address, byte count or checksum.

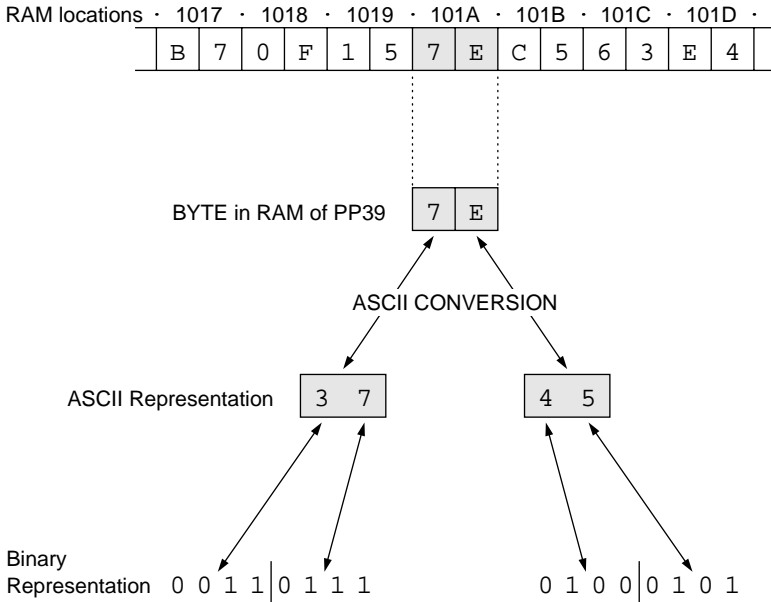
#### BINARY RUBOUT

BINARY RUBOUT is similar to BINARY apart from the inclusion of the rubout character (FF) at the start of the data.

#### DEC BINARY

This is an improvement of binary in that it has a single address and a single checksum for the entire block of data.

# STRUCTURE AND CONVERSION OF DATA BETWEEN SERIAL SIGNAL AND THE PP39 RAM



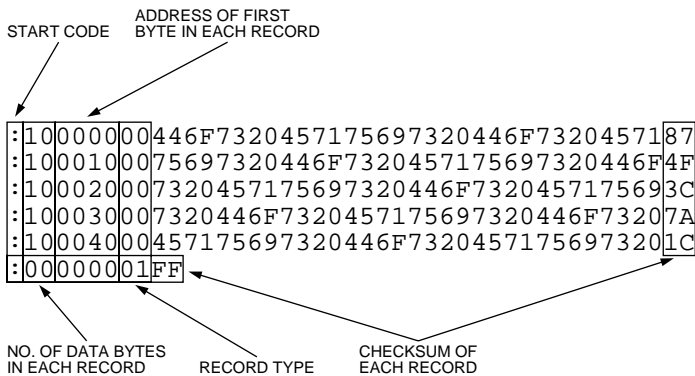
### 12.1.1 INTELLEC

The Intellec format when displayed consists of:

- a. A start code, i.e. (a colon):
- b. The sum of the number of bytes in an individual record, e.g. 10
- c. The address of the first byte of data in an individual record, e.g. 0000.
- d. The record types, i.e. 00 – Data Record  
01 – End Record.
- e. Data in bytes, e.g. 44 6F 73 20 45 71
- f. Checksum of an individual record, e.g. 87

For example:

START ADDRESS:	0000
STOP ADDRESS:	004F
OFFSET:	0000



## CALCULATION OF THE INTELLEC\* CHECKSUM

:10000000446F7320457175697320446F7320457187  
:01001000757A  
:00000001FF

Example: THE SECOND 'DATA RECORD' OF THE ABOVE FORMAT.

- (i) This is: :01 00 10 00 75 7A
- (ii) The start code and the checksum are removed: :7A
- (iii) Five Bytes remain: 01 00 10 00 75
- (iv) These are added together:  $01 + 00 + 10 + 00 + 75 = 86$
- (v) The total '71' is converted into Binary:                   8       6  
                                  1000   0110
- (vi) The Binary figure is reversed. This is known as a complement:       7       9  
                                  0111   1001
- (vii) A one is added to this complement. This addition forms a "two's complement":       7       A  
                                  0111   1010
- (viii) 7A is the checksum as above: :01 00 10 00 75 (7A)

\* This calculation also applies to the extended version.

When addition of information occurs in longer records the checksum may consist of more than one byte. When this occurs the least significant byte is always selected to undergo the above calculation.



## 12.1.2 EXTENDED INTELLEC

The extended Intellec format when displayed consists of:

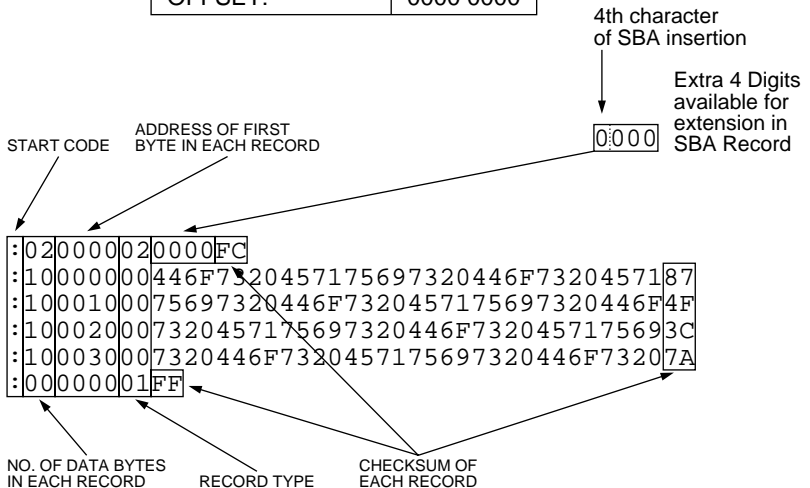
- a. A start code, i.e. (a colon):
- b. The sum of the number of bytes in an individual record, e.g. 10
- c. The address of the first byte of data in an individual record, e.g. 0000.
- d. The record types, i.e. 00 – Data Record  
01 – End Record.  
02 – 'Segment Base Address' record (SBA)\*

\* The SBA is the record that displays, the intellec extension. This is achieved by the provision of an extra digit which corresponds to the 4th character of the SBA insertion. This 4th character is effectively the extension which lengthens the standard (FFFF) limitation, into the Extended Intellec (FFFFF).

- e. Data (in bytes) e.g. 44 6F 73 20
- f. A checksum of an individual record, e.g. 87

For example:

START ADDRESS:	0000
STOP ADDRESS:	003F
OFFSET:	0000 0000



## SBA REPETITION

In some operations where an offset is in use the SBA can be displayed twice.

When the address field passes the maximum quantity for a four digit figure, i.e. (FFFF), a second SBA record is specified.

For example:

START ADDRESS:	FFA0
STOP ADDRESS:	FFFF
OFFSET:	0000 0018

```

:020018020000E4 — SBA RECORD
A:0FFB800FF00FF00FF00FF00FF00FF00FF00FF0041
:10FFC800FF00FF00FF00FF00FF00FF00FF0031
:10FFD800FF00FF00FF00FF00FF00FF00FF0021
:10FFE800FF00FF00FF00FF00FF00FF00FF0011
:08FFF800FF00FF00FF00FF0005
:020000021000EC — NEW SBA RECORD
B:8000000FF00FF00FF00FF00E8
:10000800FF00FF00FF00FF00FF00FF00FFF1
:00FFB80148
    
```

MAXIMUM ADDRESS  
FOR FOUR DIGITS (FFFF)

The SBA is added to the address field in the following fashion:

EXTENSION DIGIT	<b>B</b>
1 000	SBA INSERTION
+ 0000	ADDRESS FIELD
= 10000	

EXTENSION DIGIT	<b>A</b>
0 000	SBA INSERTION
+ FF B8	ADDRESS FIELD
= 0 FF B8	

If required by the user the remaining 3 digits of the SBA insertion can be non zero.

### 12.1.3 HEX ASCII

The Hex ASCII format when displayed consists of:

DATA ALONE

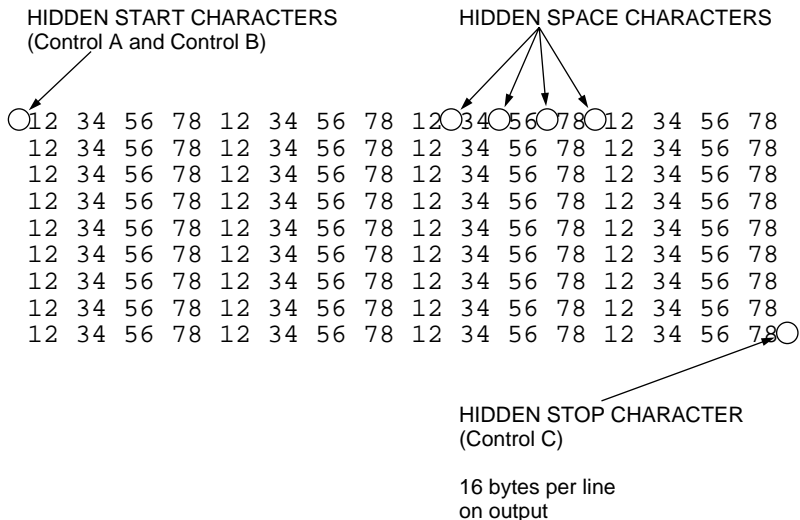
However, invisible instructions are necessary for operation. These are:

- (i) Two hidden start characters known as Control A and Control B.  
(01: ASCII code, SOH: ASCII character and 02: ASCII code, STX: ASCII character).
- (ii) A hidden stop character known as Control C.  
(03: ASCII code, ETX: ASCII character).
- (iii) A hidden 'space' character between data bytes.  
(20: ASCII code, SP: ASCII character).

For example:

START ADDRESS:	0000
STOP ADDRESS:	008F

OFFSET: NONE REQUIRED AS  
HEX ASCII ALWAYS LOADS AT ZERO





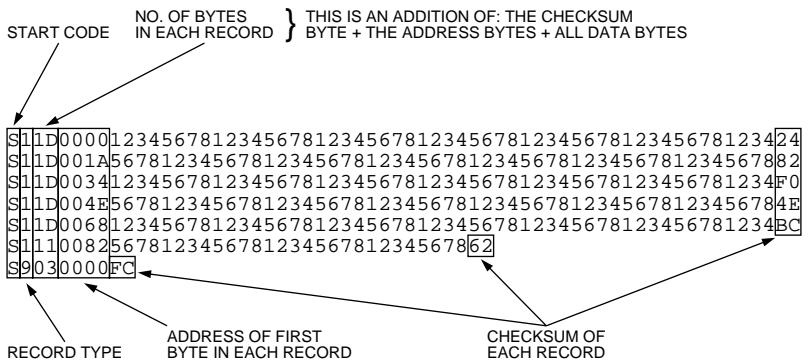
## 12.1.4 EXORCISOR

The Exorcisor format consists of:

- a. A start code, i.e. S
- b. The record types, i.e. 1 – Data Record  
9 – End Record
- c. The sum of the number of bytes in an individual record, e.g. 1D
- d. The address of the first byte of data in an individual record, e.g. 0000
- e. Data in bytes, e.g. 12 34 56 78
- f. Checksum of an individual record, e.g. A4

For example:

START ADDRESS:	0000
STOP ADDRESS:	008F
OFFSET:	0000



## CALCULATION OF THE EXORCISOR\* CHECKSUM

S11D00001234567812345678123456781234567812345678123424  
S104001A568B  
S9030000FC

Example: THE SECOND 'DATA RECORD' OF THE ABOVE FORMAT.

- (i) This is: S1 04 00 1A 56 8B
- (ii) The start code, the record type and the checksum are removed: S1 8B
- (iii) Four Bytes remain: 04 00 1A 56
- (iv) These are added together:  $04 + 00 + 1A + 56 = 74$
- (v) The total '74' is converted into Binary:
- |      |      |
|------|------|
| 7    | 4    |
| 0111 | 0100 |
- (vi) The Binary figure is reversed. This is known as a complement:
- |      |      |
|------|------|
| 8    | B    |
| 1000 | 1011 |
- (vii) 8B corresponds to the checksum as above: S1 04 00 1A 56 (8B)

When no additional figures are added to this calculation it is called a 1's (One's) complement.

\* This calculation also applies to the extended version.

When addition of information occurs in longer records the checksum may consist of more than one byte. When this occurs the least significant byte is always selected to undergo the above calculation.

### 12.1.5 EXTENDED EXORCISOR

The Extended Exorcisor is identical to the standard version when displayed up to the point that the data's address goes beyond FFFF and thus requires a 5th digit, e.g. 10000. To compensate for this addition an extra byte is added to the address giving 010000.

When this occurs the record type changes:

The data record changes from 1 to 2  
and the end record changes from 9 to 8

Similarly when the data address goes beyond FFFFFF a 7th digit is required and likewise a byte is added giving the address 8 characters: 01000000.

When this occurs:

The data record changes from 2 to 3  
and the end record changes from 8 to 7.

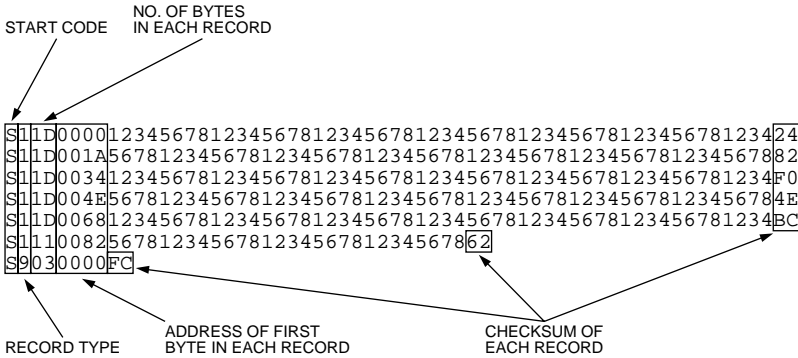
The extended exorcisor when displayed consists of:

- a. A start code, i.e. S
- b. The record types, i.e. 1 – Data Record (Four character address)  
9 – End Record (Four character address)  
  
2 – Data Record (Six character address)  
8 – End Record (Six character address)  
  
3 – Data Record (Eight character address)  
7 – End Record (Eight character address)
- c. The sum of the number of bytes in an individual record, e.g. 1D
- d. The address of the first byte of data in an individual record, e.g.  
0000, 010000, 01000000  
  
Data in bytes, e.g. 12 34 56 78  
  
Checksum of an individual record: 24

1 – Data Record (Four Character Address) }  
 9 – End Record (Four Character Address) } 2 BYTES

For example:

START ADDRESS:	0000
STOP ADDRESS:	008F
OFFSET:	0000 0000



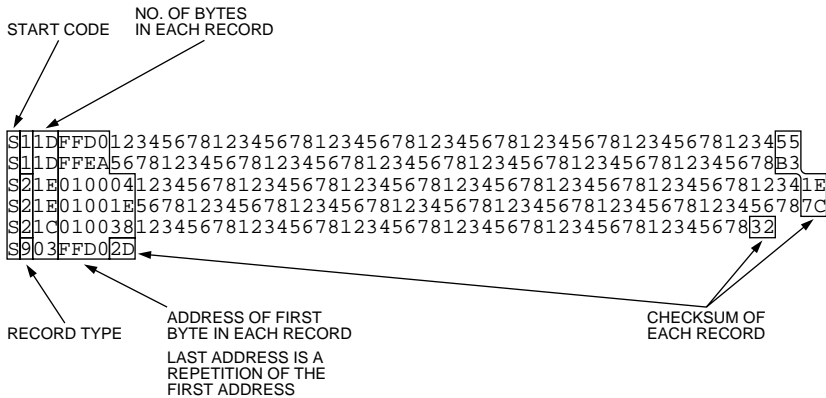
1 – Data Record (Four Character Address) }  
 9 – End Record (Four Character Address) } 2 BYTES

The Extended Exorcisor format stays identical in layout to that of the standard when the address field stays below FFFF.



**TRANSITION FROM 2 BYTE ADDRESS (4 CHARACTERS)  
THROUGH TO 3 BYTE ADDRESS (6 CHARACTERS)**

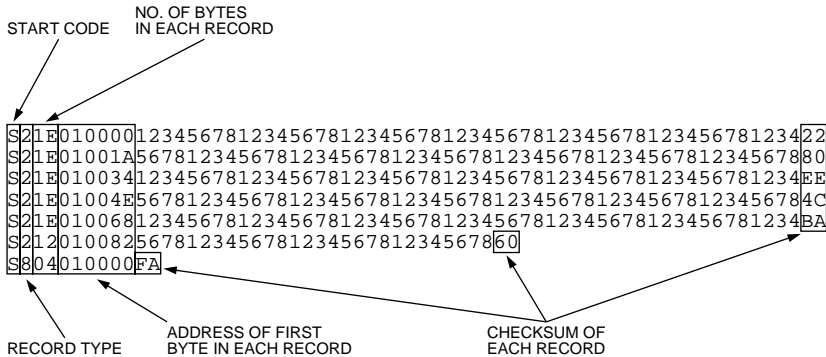
START ADDRESS:	FF80
STOP ADDRESS:	FFFF
OFFSET:	00000050



2 – Data Record (Six Character Address) }  
8 – End Record (Six Character Address) } 3 BYTES

For example:

START ADDRESS:	0000
STOP ADDRESS:	008F
OFFSET:	00010000

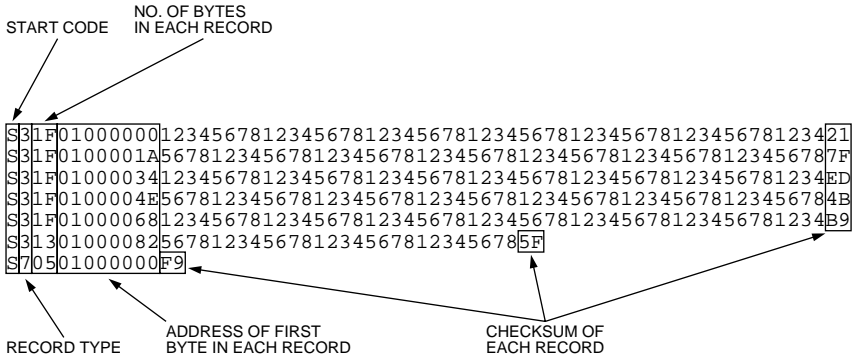


2 – Data Record (Six Character Address) }  
8 – End Record (Six Character Address) } 3 BYTES

3 – Data Record (Eight Character Address) }  
 7 – End Record (Eight Character Address) } 4 BYTES

For example:

START ADDRESS:	0000
STOP ADDRESS:	008F
OFFSET:	01000000



3 – Data Record (Eight Character Address) }  
 7 – End Record (Eight Character Address) } 4 BYTES

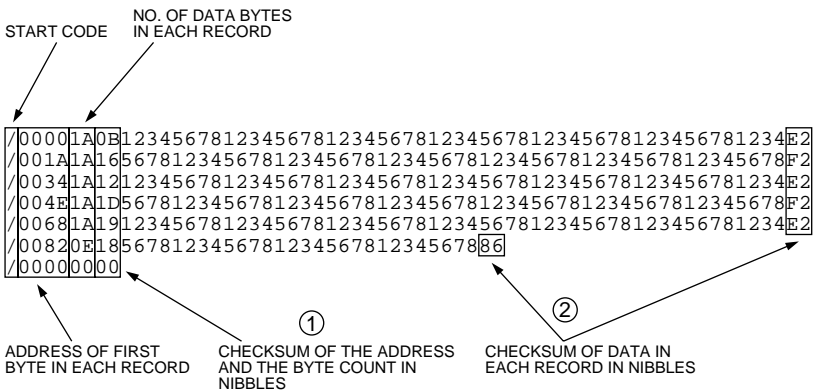
### 12.1.6 TEK HEX

The Tek Hex format when displayed consists of:

- A start code, i.e. /
- The address of the first byte of data in an individual record, e.g. 0000
- The sum of the number of bytes in an individual record, e.g. 1A
- Checksum 1 which is a nibble addition of the address (4 characters) and the byte count (2 characters), e.g. 0B
- Data in bytes, e.g. 12 34 56 78
- Checksum 2 which is a nibble addition of all data.
- An end record which automatically stops the operation when 00 is specified in the byte count (c).

For example:

START ADDRESS:	0000
STOP ADDRESS:	008F
OFFSET:	0000



## CALCULATION OF TEK HEX CHECKSUMS

Unlike the other PP39 formats, the Tek Hex has two checksums which are both the result of nibble additions, as opposed to byte additions.

Checksum 1 is a nibble addition of the 'address' and the 'byte count' which make 6 characters in total.

Checksum 2 is a nibble addition of the data alone.

```
/00001A0B1234567812345678123456781234567812345678123456781234E2
CHECKSUM 1 |CHECKSUM 2 6781234567812345678123456781234567812345678F2
/0034030A12345615
/00000000
```

Example: THE THIRD 'DATA RECORD' OF THE ABOVE FORMAT.

### CHECKSUM 1

- (i) This is: /10034030A
- (ii) The start code and the checksum are removed: /0A
- (iii) Six nibbles remain: 003403
- (iv) They are added together:  $0 + 0 + 3 + 4 + 0 + 3 = A$
- (v) 0A is the checksum which is displayed in byte form as above: /1003403 (0A)

### CHECKSUM 2

- (i) This is: 12345615
- (ii) The checksum is removed: 15
- (iii) Six nibbles remain: 123456
- (iv) These are added together:  $1 + 2 + 3 + 4 + 5 + 6 = 15$
- (v) 15 is the checksum as above 123456 (15)

When addition of nibble information occurs in longer records the checksum may consist of more than one byte. When this occurs the least significant byte is always selected to undergo the above calculation.

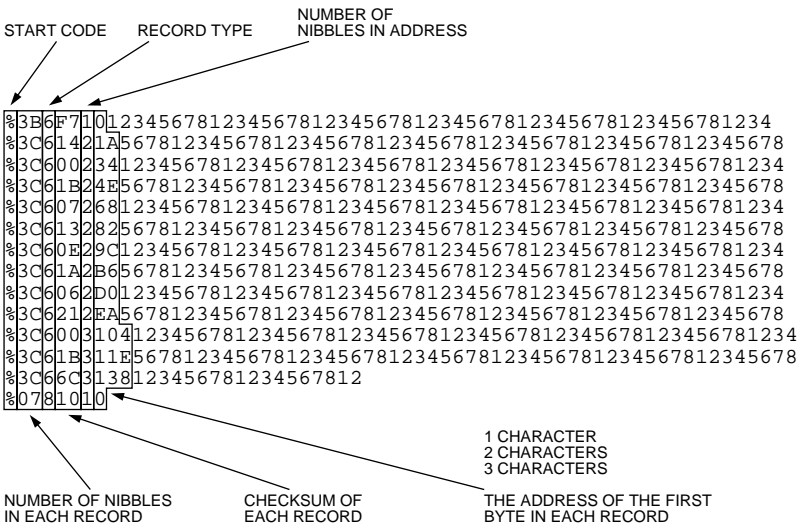
## 12.1.7 EXTENDED TEK HEX

The Extended Tek Hex when displayed consists of:

- A start code: % (percentage)
- A count of the nibbles in an individual record, e.g. 3B
- The record types, i.e. 6 – Data Record  
8 – End Record
- A checksum of the whole of an individual record excluding the %, e.g. F7
- \* The number of nibbles comprising – “the address of the first byte in each record”, e.g. 1, 2, 3 etc.
- The address of the first byte of data in an individual record, e.g. 0, 1A, 104

For example:

START ADDRESS:	0000
STOP ADDRESS:	0140
OFFSET:	0000 0000



\* Sections (e) and (f) are integrated:

As the operation progresses the address field lengthens. More characters are added to show this expansion. The nibble count of section (e) reflects this, e.g.:

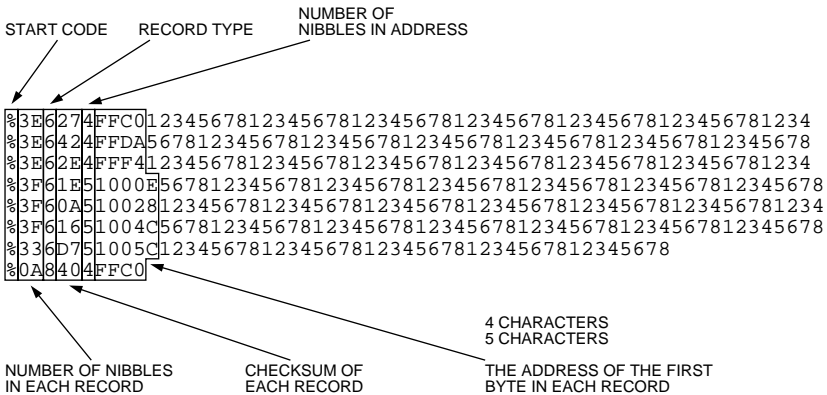
2/1A                      6/100000                      A/1B4625DC95  
 2 Characters              6 Characters              A Characters (10 in Decimal)

The nibble count has the facility to rise to 'F' making a 15 (DECIMAL) character address field possible.

**EXTENDED TEK HEX WITH AN OFFSET, DISPLAYING TRANSITION FROM 4 CHARACTER ADDRESS FIELD TO 5 CHARACTER ADDRESS FIELD.**

For example:

START ADDRESS:	0000
STOP ADDRESS:	00AF
OFFSET:	0000 FFC0



## CALCULATION OF THE EXTENDED TEK HEX CHECKSUM

Unlike the standard version the Extended Tek Hex has only one checksum.

```
%3B6F710123456781234567812345678123456781234567812345678123456781234  
%3C61421A567812345678123456781234567812345678123456781234567812345678  
%0A61C23412  
%0781010
```

Example: THE THIRD LINE OF THE ABOVE FORMAT.

- (i) This is: % 0A61C23412
- (ii) The start code and the checksum are removed: % 1C
- (iii) Eight nibbles remain: 0A623412
- (iv) These are added together:  $0 + A + 6 + 2 + 3 + 4 + 1 + 2 = 1C$
- (v) 1C is the checksum as above: % 0A6 1C 23412

When addition of nibble information occurs in longer records the checksum may consist of more than one byte. When this occurs the least significant byte is always selected to undergo the above calculation.





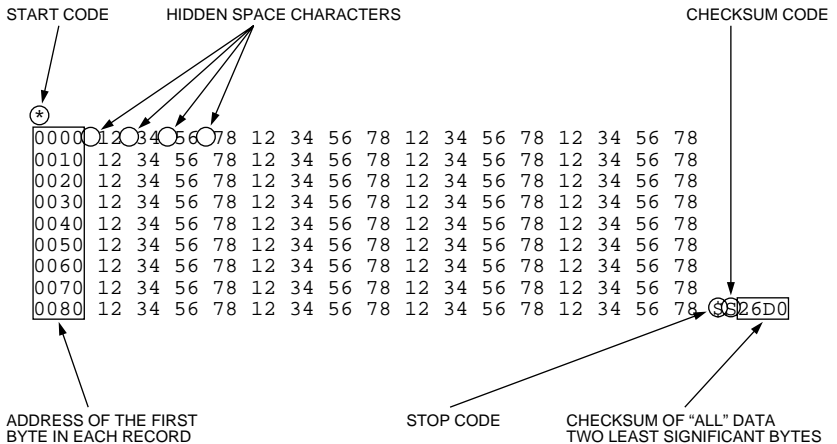
### 12.1.8 PPX or (STAG HEX) \*

The PPX format when displayed consists of:

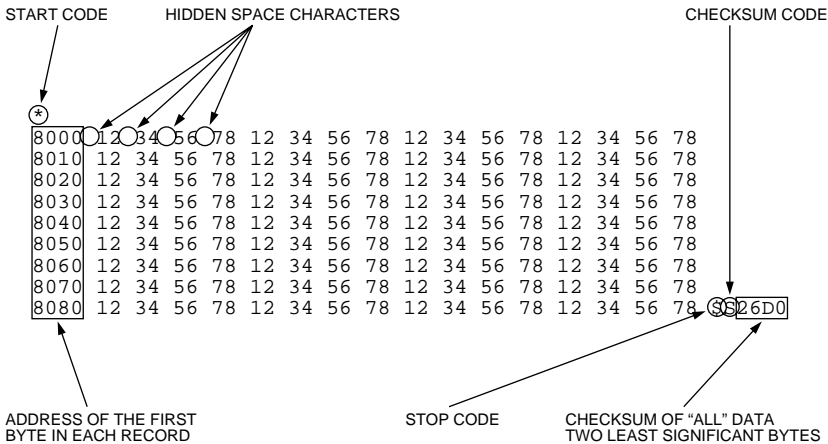
- a. A start code, i.e. \* (an asterisk, 2A – ASCII Code)
- b. The address of the first byte of data in an individual record, e.g. 1000
- c. Data in bytes, e.g. 12 34 56 78
- d. A stop code, i.e. \$ (a dollar sign, 24 – ASCII Code)
- e. A checksum of all data over the entire address range. (The displayed checksum is the two least significant bytes.)
- f. A checksum start code: S
- g. An invisible space character between data bytes (20 – ASCII Code)

For example:

START ADDRESS:	0000
STOP ADDRESS:	008F
OFFSET:	0000



## AND WITH AN OFFSET OF 8000



## CALCULATION OF THE PPX CHECKSUM

"Data alone", in bytes over the entire address range (as opposed to individual records) is added together to give the checksum. The address is not included in this calculation.

```
*
0000 12 34 56 78 $S0114
```

Example: THE SEGMENT OF DATA ABOVE

- (i) This is: \*0000 12 34 56 78 \$S0114
- (ii) The start code, the address, the stop code, the checksum code and the checksum are removed: \*0000 \$S0114
- (iii) Four bytes remain: 12 34 56 78
- (iv) These are added together: 12 + 34 + 56 + 78 = 114
- (v) 114 is the checksum which is displayed in two byte form as above: \*0000 12 34 56 78 \$S0114

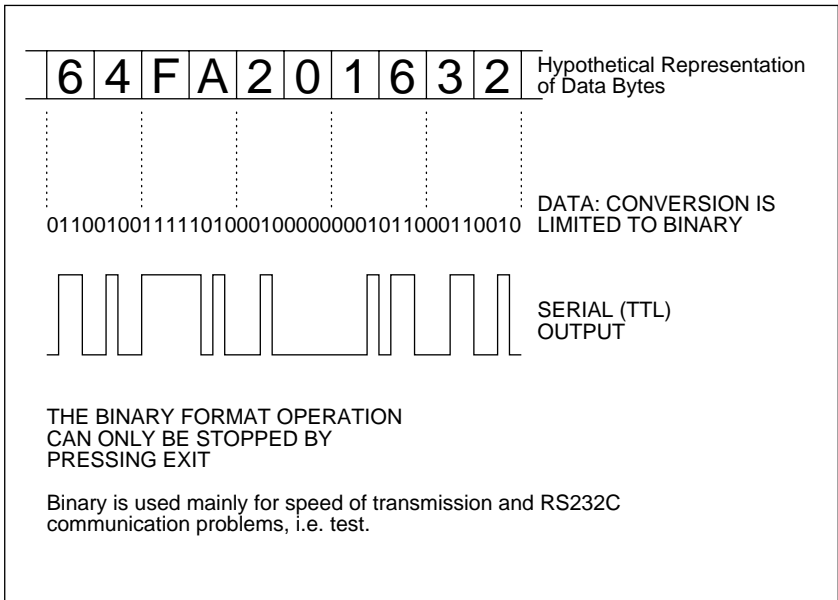
As the PPX checksum is an addition of all data the total will invariably constitute more than two bytes. When this occurs the least significant two bytes are always selected to undergo the above calculation.

## 12.1.9 BINARY, DEC BINARY and BINARY RUBOUT

Binary, DEC Binary and Binary Rubout are the most fundamental of all formats. ASCII code conversion never occurs. Information is therefore limited to the interpretation of pulses via the RS232C interface port into either ONES or ZEROS. Hence 'Binary'. A visual display is not possible, however a simple graphical representation can be made.

### BINARY

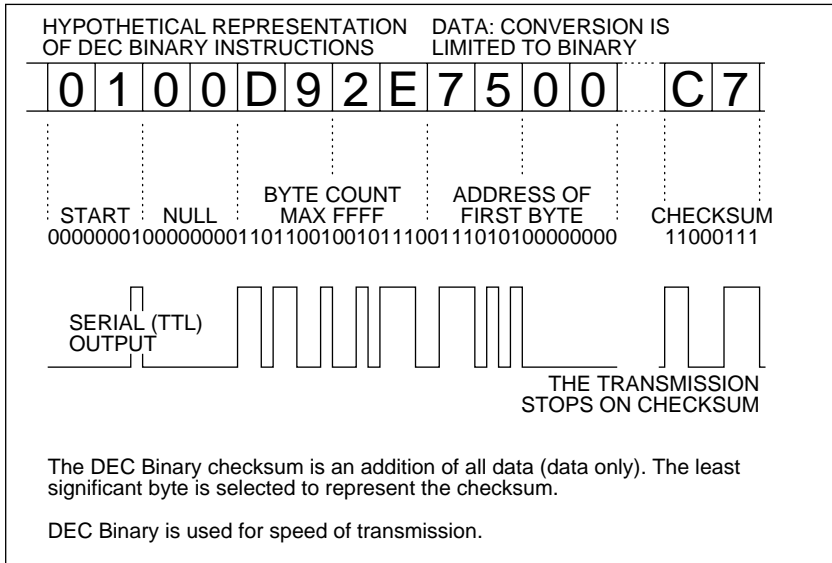
Binary is data only. It is devoid of a start code, address, stop code and checksum.



## DEC BINARY

DEC Binary is an improvement of Binary. It has a start code, a null prior to transmission, a byte count, a single address and a single checksum of all data. It also has the facility for an offset to be set.

For example:

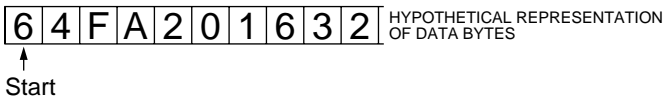


## BINARY RUBOUT

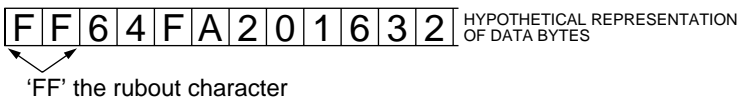
Binary Rubout is similar to Binary in that it is devoid of Address, Stop Code and Checksum. The data is preceded however, by the Rubout character (FF).

For example:

If a string of Binary data is represented thus:



then the same data in Binary Rubout format would be represented thus:



## **SECTION 13**

















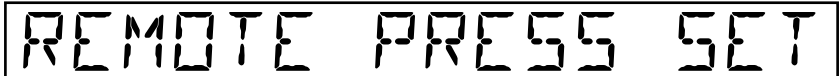
## **SECTION 14**



## REMOTE CONTROL

### 14.1 SELECTING REMOTE CONTROL

To select remote control Press Set 2  
The display will show:



REMOTE PRESS SET

By pressing set again, the display will show manufacturer, device type and remote mode.

For instance:

Manufacturer	Device type	Remote mode
SE0	5516A	REM

In the remote mode the PP39 operates under remote control from a computer or a terminal. The keyboard of the PP39 is inoperative at this time and the display will only show information as requested under remote control.





## 14.2 REMOTE CONTROL COMMANDS

h = one hex digit

- RETURN    Executes a command for instance G RETURN, A6AF< RETURN, [ RETURN, 11A RETURN etc.
- ESC        Aborts a command.
- G          Software revision number. This command issues a 4-digit hex number representing the software configuration in the programmer.
- Z          Exits from remote control.

### SET UP FOR LOAD AND PROGRAM

#### Device Type Selection

- hhhh@    \* A four digit code sets up programming for a particular device. (The first two digits represent the manufacturer code and the second two represent the pin out codes).
- [          \* The programmer sends a four digit hex code of the device in use. (The first two digits represent the manufacturer code and the second two represent the pin out code).
- T          Test for illegal bit in the device.
- B          Blank check, sees that no bits are programmed in the device.
- R          Respond indicates device status for instance: 0FFF/8/0>:  
The first 4 digits reflect the working RAM limit relevant to the device. The 5th digit is the byte size measured in bits. The 6th digit reflects the unprogrammed state of the device selected. The 6th digit can be either 1 or 0.  
1 = Unprogrammed state 00.  
0 = Unprogrammed state FF.

#### Selection of bit mode configuration

1M]	GANG MODE
2M]	8-BIT MODE
3M]	16-BIT MODE
4M]	32-BIT MODE LOW
5M]	32-BIT MODE HIGH

\* See: LIST OF DEVICES AND DEVICE CODES FOR THE 39M100 MODULE

## Set Up for Load and Program (Cont.)

### Device/RAM address limits

#### RAM low address <

hhhh< This defines the lower address limit in RAM

#### Device high address ;

hhhh; This sets the number of bytes of data to be transferred, therefore effectively defining the upper address limit.

#### Device low address :

hhhh: This defines the lower address limit within the device or devices.

NOTE: The above commands may specify either the left or right ZIF socket using the suffix L or R. For instance hhhhL; or hhhhR<. If no suffix is included the left socket will be assumed therefore, hhhh;

L	LOADS device data into RAM.
P	PROGRAMS RAM data into device.
V	VERIFY device against RAM.
S	CHECKSUM causes programmer to calculate checksum of RAM data.
RS	RIGHT CHECKSUM causes programmer to calculate checksum of RAM data for right socket.
LS	LEFT CHECKSUM causes programmer to calculate checksum of RAM data for left socket.
00 ^	Fills RAM with 00s
FF ^	Fills RAM with FFs

By initiating either the load or program operation, data transference will commence between the RAM and devices inclusive of any selected parameters specified above.

## SET UP FOR INPUT AND OUTPUT

### Selection of Translation Formats A

10A	Binary
11A	DEC Binary
12A	Binary Rubout
50A	Hex-ASCII (space)
51A	Hex-ASCII (percent)
52A	Hex-ASCII (apostrophe)
53A	Hex-ASCII (comma)
59A	PPX (Stag Hex *)
82A	Exorcisor
83A	Intellec
86A	Tek-Hex
92A OR 87A	Extended Exorcisor
93A OR 88A	Extended Intellec
96A	Extended Tek-Hex

} All covered by  
standard hex-ASCII

### Input/Output Address Limits

#### Lower Address Limit

hhhh<        This gives a four digit figure defining the lower address limit.

#### Upper Address Limit

hhhh;        This sets the number of bytes of data to be transferred, therefore effectively defining the upper address limit.

#### Input Output Offset

hhhh hhhhW    This defines the offset required for data transference in both input and output, 4 or 8 digits can be specified.

NOTE: The above commands may specify either the left or right ZIF socket using the suffix L or R. For instance hhhhL; or hhhhR<. If no suffix is included the left socket will be assumed therefore, hhhh;.

I              This inputs data from computer to RAM

O              This outputs data from RAM to computer

By initiating either the input or output operation data transference will commence, inclusive of any specified parameters above.

## **ERROR RESPONSES**

- F Error-status inquiry returns a 32-bit word that codes errors accumulated. Error-status word returns to zero after interrogation. (See PP39 remote error words)
- X Error-code inquiry. Programmer outputs error codes stored in scratch-RAM and then clears them from memory. (See PP39 remote error codes)
- H No operation. This is a null command and always returns a prompt character ( > ).

## **PROGRAMMER RESPONSES**

- > Prompt character. Informs the computer that the programmer has successfully executed a command.
- F Fail character. Informs the computer that the programmer has failed to execute the last-entered command.
- ? Question mark. Informs the computer that the programmer does not understand a command.

### 14.3 REMOTE ERROR WORD -F-

#### BIT

#### NUMBER

#### RECEIVER ERRORS

- 31 If any error has occurred, this bit is set
- 30 Not used
- 29 Not used
- 28 Not used

27

26 Serial-overflow error (42)

25 Serial-framing error (41, 43)

24 Command-buffer overflow, i.e. > 18 characters (48)

#### PROGRAMMING ERRORS

23 Any device-related error

22 Device appears faulty to the machine electronics (26)

21  $L2 + L3 > \text{Device}$

20 Not used

19 Device not blank (20)

18 Illegal bit (21)

17 Non verify (23)

16 Incomplete programming or invalid device (22)

#### I/O ERRORS

15 If any I/O error has occurred, this bit is set

14 Not used

13 Not used

12 Not used

11 Checksum error (82)

10 Not used

9 Address error, i.e. > word limit

8 Data not hexadecimal where expected (84)

#### RAM ERRORS

7 RAM – hardware error

6 Not used

5  $L2 + L3 > \text{RAM}$

4 Not used

3 Not used

2 No RAM or insufficient RAM resident

1 RAM write error, or program-memory failure

0 Not used

## INTERPRETATION OF THE ERROR STATUS WORD

EXAMPLE: 80C80084

- 8 – The word contains error information
- 0 – No receive errors
- C – (= 8 + 4); 8 = Device error  
4 = Start line not set high
- 8 – Device is not blank
- 0 – No input errors
- 0 – No input errors
- 8 – RAM error
- 4 – Insufficient RAM resident

## REMOTE ERROR CODES – 'X' remote code PP39

Code	Name	Description
20	Blank check Error	Device not blank
21	Illegal bit Error	
22	Programming Error	The device selected could not be programmed
23	Verify Error	
26	Device Faulty	Either faulty part or reversed part
41	Framing Error	
42	Overrun Error	
43	Framing and Overrun Error	
48	Buffer Overflow	
50	No Data Input	Because of address errors or an invalid format selected
81	Parity Error	
82	Checksum	
84	Invalid Data	

## **SECTION 15**





## 15.1 THE ASCII CODE

ASCII Code	Character	ASCII Code	Character	ASCII Code	Character
00	NUL	2B	+	56	V
01	SOH	2C	,	57	W
02	STX	2D	-	58	X
03	ETX	2E	.	59	Y
04	EOT	2F	/	5A	Z
05	ENQ	30	0	5B	[
06	ACK	31	1	5C	\
07	BEL	32	2	5D	]
08	BS	33	3	5E	^ (↑)
09	HT	34	4	5F	_ (←)
0A	LF	35	5	60	`
0B	VT	36	6	61	a
0C	FF	37	7	62	b
0D	CR	38	8	63	c
0E	SO	39	9	64	d
0F	SI	3A	:	65	e
10	DLE	3B	;	66	f
11	DC1 (X-ON)	3C	<	67	g
12	DC2 (TAPE)	3D	=	68	h
13	DC3 (X-OFF)	3E	>	69	i
14	DC4	3F	?	6A	j
15	NAK	40	@	6B	k
16	SYN	41	A	6C	l
17	ETB	42	B	6D	m
18	CAN	43	C	6E	n
19	EM	44	D	6F	o
1A	SUB	45	E	70	p
1B	ESC	46	F	71	q
1C	FS	47	G	72	r
1D	GS	48	H	73	s
1E	RS	49	I	74	t
1F	US	4A	J	75	u
20	SP	4B	K	76	v
21	!	4C	L	77	w
22	"	4D	M	78	x
23	£	4E	N	79	y
24	\$	4F	O	7A	z
25	%	50	P	7B	{
26	&	51	Q	7C	
27	'	52	R	7D	} (ALT MODE)
28	(	53	S	7E	~
29	)	54	T	7F	DEL (RUB OUT)
2A	*	55	U		



## 15.2 SPECIFICATION

Programming Support:	39M100 Module supports 24 and 28 pin EPROMs and EEPROMs.  39M200 Module supports 40 pin Microprocessors.
User RAM:	64K x 8 (512 bits) Expansion RAM to standard 1M bits and 2M bits
Keyboard:	16 Hexadecimal keys, 4 cursor keys and 11 function keys
Display:	16 character alpha numeric green vacuum fluorescent display
Auto Recall:	Up to 9 complete machine configurations may be stored in non volatile memory and recalled at any time. Parameters include device type, I/O format, RS232C baud rate, address range etc.
Zif Socket Test:	Tests zif socket for poor connections or faulty device.
Device Test:	Empty, Verify and Illegal Bit
Access Time Test:	Variable access time test 100-600ns
Programming Speed:	High speed programming algorithms are used where applicable.
Auto Select:	The 39M100 module supports Silicon Signature* and Intelligent Identifier* coded devices.

I/O Interface:	RS232C with full handshake XON/XOFF, device control on input, keyboard entry of parameters and transmission rates up to 19,200 baud. Full remote control.
I/O Formats:	Supports all commonly used I/O formats including extended formats, e.g. Intel-hex, Tek-hex, Extended Tek-hex, Motorola S-record, Hex-ASCII, Stag-hex, Binary, DEC Binary and Binary Rubout.
Audible Alarm:	Software selectable to indicate end of program test or as a warning.
Set Programming:	Will program two devices simultaneously with different data for 16 bit applications. The machine is also configured to program 32 bit sets.
Edit Functions:	String Search, Insert, Delete, Block move, Complement, Interlace, Fill RAM with test pattern etc.
Self-Test:	Automatically runs self-test program on power-up.
Operating Voltages:	100-130V 200-260V 60/50Hz
Power Consumption:	70 Watts
Physical Specification:	Width: 315mm; Height: 90mm; Depth: 225mm; Weight: 2.5Kg

Interlace\* and Stag Hex\* are tradenames of Stag.  
 Silicon Signature\* is a tradename of the SEEQ Corporation.  
 Intelligent Identifier\* is a tradename of the Intel Corporation.

Stag reserve the right to alter design and specifications without prior notice in pursuit of a policy for continuous improvement.