

# **Operating Manual**

## **for**

# **PP38 MOS Programmer**

Stag Electronic Designs Ltd.  
Stag House  
Tewin Court  
Tewin Road  
Welwyn Garden City  
Hertfordshire AL7 1AU  
Telephone (0707) 332148  
Telex 8953451

# PP38 MOS Programmer

## CONTENTS

Section 1	General Introduction
1.1	Introduction
1.2	The Keyboard
1.3	Initial Setting-up Procedure
1.4	Selection of 'Local' or 'Remote' Modes
1.5	Correct Operation of ZIF Sockets
1.6	RAM Operating Structure
1.7	List of 'Set' Commands
Section 2.	Device Selection
2.1	Selecting a Device
2.2	Electronic Identifier
Section 3.	Bit Modes
3.1	Section of Bit Mode Configuration
3.2	8-Bit Mode
3.3	Gang Mode
3.4	16-Bit Mode
3.5	32-Bit Mode
Section 4.	Device Functions
4.1	Load
4.2	Empty Test
4.3	Pre-program Bit Test
4.4	Programming
4.5	Verify
4.6	Checksum and CRC
4.7	Device/RAM Address Limits
4.8	Save and Recall Machine Configurations
Section 5.	RAM Functions
5.1	List and Edit
5.2	Insert
5.3	Delete
5.4	Block Move
5.5	Filling the RAM
5.6	Complement
5.7	String Search
Section 6.	Interface
6.1	Setting the Interface Parameters
6.2	Input/Output Operation

## Contents

Section 7.	Format Descriptions
7.1.1	Intellec
7.1.2	Extended Intellec
7.1.3	Hex. ASCII
7.1.4	Exorcisor
7.1.5	Extended Exorcisor
7.1.6	Tek Hex.
7.1.7	Extended Tek Hex.
7.1.8	PPX (or Stag Hex. *)
7.1.9	Binary, DEC Binary and Binary Rubout
Section 8.	RS232C Hardware Description
8.1	RS232C Interface Port Connections
8.2	XON/XOFF (3 wire cable-form connection)
8.3	Hardware Handshake (7 or 8 wire cableform)
8.4	Connection to IBM *PC and AT
Section 9	Remote Control
9.1	Selecting Remote Control
9.2	Remote Control Commands
9.3	Remote Error Word-F-
Section 10	Worldwide Sales and Service Support

APPENDIX I  
PP38 Device Support (Rev 1.1)  
(28th March 1988)

Checklist

When all devices up to and including this issue have been correctly inserted, the complete Device Support List for the PP38 should consist of the following pages:

Page 1	Issue 1.0
Page 2	Issue 1.0
Page 3	Issue 1.0
Page 4	Issue 1.0

# PP38 Device Support

## AMD

Device	Code	Rev
2716.....	9P42...	1.0
2732.....	9P44...	1.0
2832A.....	9P44...	1.0
2764.....	9P4A...	1.0
2764A.....	9P4A...	1.0
27128.....	9P4B...	1.0
27128A.....	9P4B...	1.0
27256.....	9P4C...	1.0
27512.....	9P4D...	1.0
2817A.....	9P58...	1.0
2864B.....	9P5C...	1.0
9864.....	9P5A...	1.0

## HITACHI

Device	Code	Rev
2532.....	BP42...	1.0
2716.....	BP42...	1.0
2732.....	BP44...	1.0
2732A.....	BP44...	1.0
2764.....	BP4A...	1.0
2764A.....	BP4A...	1.0
27C64.....	BP4A...	1.0
27128.....	BP4B...	1.0
27128A.....	BP4B...	1.0
27256.....	BP4C...	1.0
27C256.....	BP4C...	1.0
27512.....	BP4D...	1.0
48016.....	BP51...	1.0
58064.....	BP5A...	1.0
58065.....	BP7C...	1.0

## EXEL

Device	Code	Rev
2816A.....	0152...	1.0
2817A.....	0158...	1.0
2864A.....	015C...	1.0
2865A.....	015C...	1.0

## ICT

Device	Code	Rev
27C64A.....	0CDA...	1.0

## FUJITSU

Device	Code	Rev
2716.....	AP42...	1.0
2732.....	AP44...	1.0
2732A.....	AP44...	1.0
2764.....	AP4A...	1.0
27128.....	AP4B...	1.0
27C128.....	AP4B...	1.0
27256.....	AP4C...	1.0
27C256.....	AP4C...	1.0
27C256A.....	AP4D...	1.0
27C512.....	AP4D...	1.0

## INTEL

Device	Code	Rev
2716.....	6P42...	1.0
2732.....	6P44...	1.0
2732A.....	6P44...	1.0
2764.....	6P4A...	1.0
27128.....	6P4B...	1.0
2764A.....	6P4A...	1.0
27128A.....	6P4B...	1.0
27128B.....	6P4B...	1.0
27C64.....	6P4A...	1.0
27C128.....	6P4B...	1.0
27256...*	6P4C...	1.0
27256..QP...	6PFC...	1.0
27C256.....	6PDC...	1.0
27512...*	6P4D...	1.0
27512..QP...	6PFD...	1.0
27513...*	6PFD...	1.0
27513..QP...	6PFE...	1.0
2816A.....	6P52...	1.0
2816D.....	6P57...	1.0
2817...*	6P53...	1.0
2817A.....	6P58...	1.0
2864A.....	6P5C...	1.0
87C64.....	6PEA...	1.0
87C256.....	6PEC...	1.0

## GENERAL

## INSTRUMENTS

Device	Code	Rev
27C64.....	02DA...	1.0
27HC64.....	02DA...	1.0
27256.....	024C...	1.0
27C256.....	02DC...	1.0

# PP3B Device Support

## MITSUBISHI

Device	Code	Rev
2716.....	DF42...	1.0
2732.....	DF44...	1.0
2732A.....	DFC4...	1.0
2764.....	DF4A...	1.0
27128.....	DF4B...	1.0
27C128.....	DFDB...	1.0
27256.....	DF4C...	1.0
27512.....	DF4D...	1.0

Device	Code	Rev
2716.....	CF42...	1.0
2732.....	CF44...	1.0
2732A.....	CFC4...	1.0
2764.....	CF4A...	1.0
27C64.....	CFDA...	1.0
27128.....	CF4B...	1.0
27256.....	CF4C...	1.0
27C256.....	CFDC...	1.0
27C256A.....	CFDB...	1.0
27C512.....	CFDD...	1.0
28C64.....	CF7C...	1.0

## MOTOROLA

Device	Code	Rev
2532.....	7F43...	1.0
2716.....	7F42...	1.0
2732.....	7F44...	1.0
2816.....	7F47...	1.0
2817.....	7F53...	1.0
68764.....	7F45...	1.0
68766.....	7F47...	1.0

## OKI

Device	Code	Rev
2532.....	0843...	1.0
2716.....	0842...	1.0
2732.....	0844...	1.0
2732A.....	08C4...	1.0
2764.....	084A...	1.0
27128.....	084B...	1.0

## NATIONAL

Device	Code	Rev
2716.....	3F42...	1.0
27C16.....	3FD2...	1.0
2732.....	3F44...	1.0
27C32.....	3FD4...	1.0
27C32H.....	3FD6...	1.0
27C32B.....	3FD5...	1.0
2732A.....	3FC4...	1.0
2764.....	3F4A...	1.0
27128.....	3F4B...	1.0
27C64.....	3FDA...	1.0
27CP128.....	3FDB...	1.0
27C256.....	3FDC...	1.0
27C512.....	3FDD...	1.0
9816.....	3F51...	1.0
9817.....	3F58...	1.0
98C64.....	3F7C...	1.0

## ROCKWELL

Device	Code	Rev
2816A.....	0652...	1.0
87C32.....	06D4...	1.0

## SAMSUNG

Device	Code	Rev
2816A.....	0952...	1.0
2864A.....	095C...	1.0
2865A.....	095C...	1.0



# PP38 Device Support

## SEEQ

Device	Code	Rev
2764.....	FF4A...	1.0
27128.....	FF4B...	1.0
27256.....	FF4C...	1.0
27C256.....	FFDC...	1.0
2816A.....	FF52...	1.0
2817A.....	FF5B...	1.0
2864.....	FF5B...	1.0
28C64.....	FF7F...	1.0
28C256.....	FF7C...	1.0
5213.....	FF54...	1.0
52B13.....	FF54...	1.0
52B13H.....	FF56...	1.0
52B33.....	FF5A...	1.0
52B33H.....	FF5B...	1.0

## TEXAS

### INSTRUMENTS

Device	Code	Rev
2516.....	4F42...	1.0
2532.....	4F43...	1.0
2532A.....	4F41...	1.0
2564.....	4F47...	1.0
2732A.....	4FC4...	1.0
2764.....	4F4A...	1.0
27128.....	4F4B...	1.0
27128A.....	4FCB...	1.0
27C128.....	4FDB...	1.0
27256.....	4F4C...	1.0
27C256.....	4FDC...	1.0
27C512.....	4FDD...	1.0

## SGS

Device	Code	Rev
2532.....	8F43...	1.0
2716.....	8F43...	1.0
2732A.....	8FC4...	1.0
2764.....	8F4A...	1.0
2764A.....	8FCA...	1.0
27128A.....	8FCB...	1.0
27256.....	8F4C...	1.0

## TOSHIBA

Device	Code	Rev
2732.....	EF44...	1.0
2732A.....	EFC4...	1.0
2764.....	EF4A...	1.0
2764A.....	EFCA...	1.0
27128.....	EF4B...	1.0
27128A.....	EFCB...	1.0
27256.....	EF4C...	1.0
27256A.....	EFCC...	1.0
57256.....	EFDC...	1.0

## SIGNETICS

Device	Code	Rev
27C64A.....	1F0A...	1.0
B7C64.....	1FEA...	1.0

## VLSI

Device	Code	Rev
27C64.....	04DA...	1.0
27C128.....	04DB...	1.0
27C256.....	04DC...	1.0

## SMOS

Device	Code	Rev
27C64.....	0F0A...	1.0
27128.....	0F4B...	1.0
27C256.....	0FDC...	1.0
2864.....	0F5B...	1.0

## WAFERSCALE

Device	Code	Rev
57C64.....	0BDA...	1.0
57C128.....	0BDB...	1.0
57C49.....	0BAC...	1.0



# PP38 Device Support

## XICOR

Device	Code	Rev
--------	------	-----

2804A.....	0750...	1.0
2816A.....	0752...	1.0
2816H.....	0757...	1.0
2816H.....	0757...	1.0
28C16.....	0776...	1.0
2864A.....	075C...	1.0
2864B.....	075D...	1.0
28C64.....	077C...	1.0
2864H.....	075B...	1.0
28256.....	075F...	1.0
28C256.....	077F...	1.0

## EMULATOR

Device	Code	Rev
--------	------	-----

2516.....	OE42...	1.0
2532.....	OE43...	1.0
2564.....	OE47...	1.0
2716.....	OE42...	1.0
2732.....	OE44...	1.0
2764.....	OE4A...	1.0
27128.....	OE4B...	1.0
27256.....	OE4C...	1.0
27512.....	OE4D...	1.0
87C64.....	OEEA...	1.0
87C128.....	OEEB...	1.0
87C256.....	OEEC...	1.0

## SECTION 1

stag

---

Sophisticated systems for the discerning engineer.



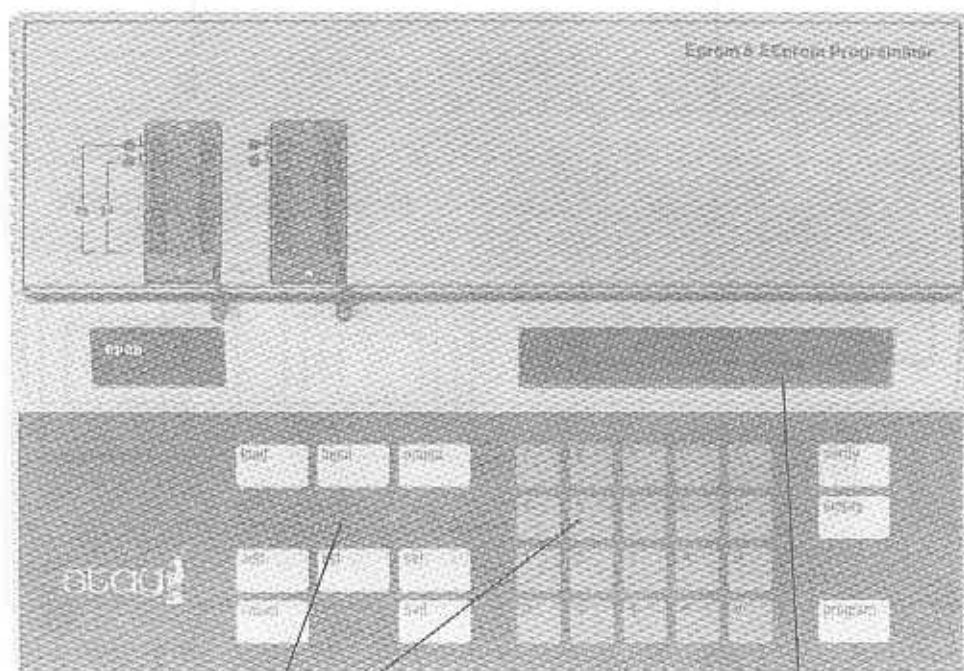
## 1 General Introduction

### 1.1 Introduction

The PP38 is a Universal MOS Programmer which is capable of supporting all MOS erasable PROM devices in NMOS and CMOS technology.

The PP38 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.

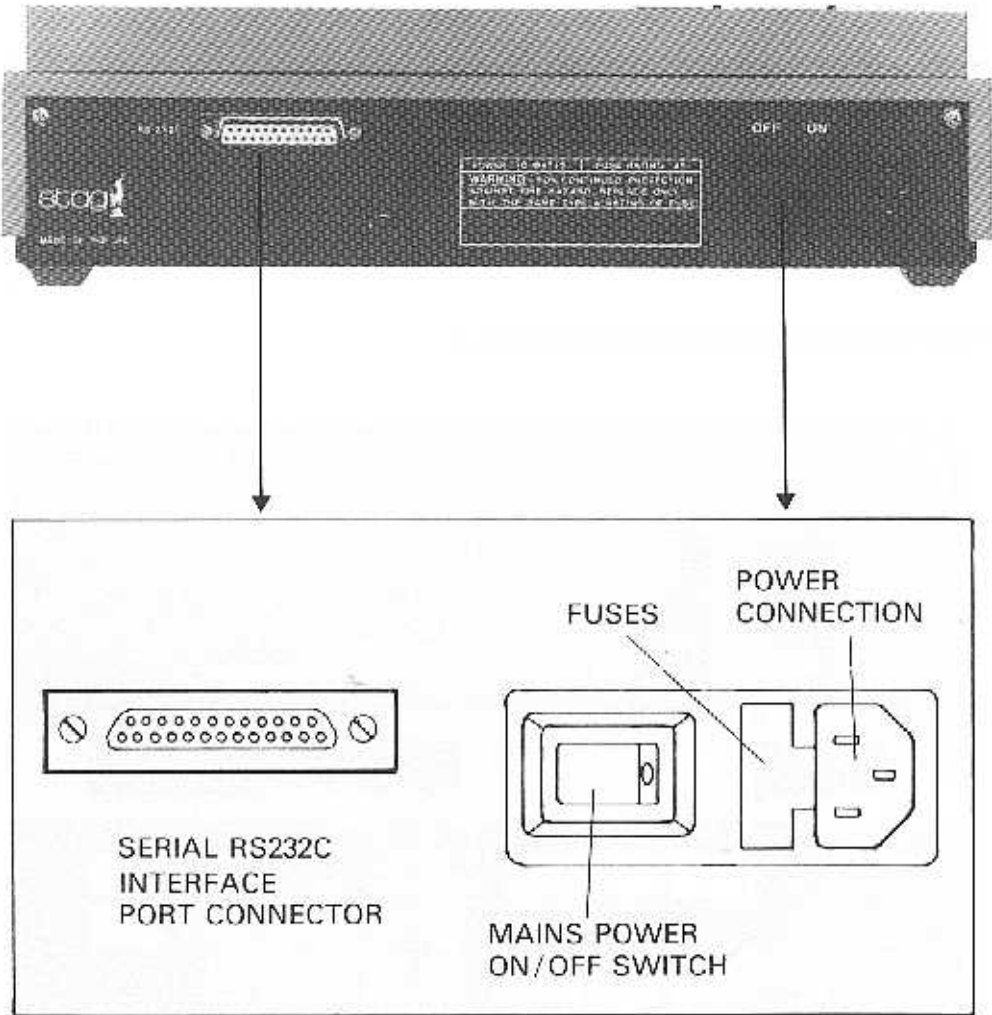
#### PP38 EPROM and EEPROM Programmer



Keyboard: For data entry and operating programmer functions.

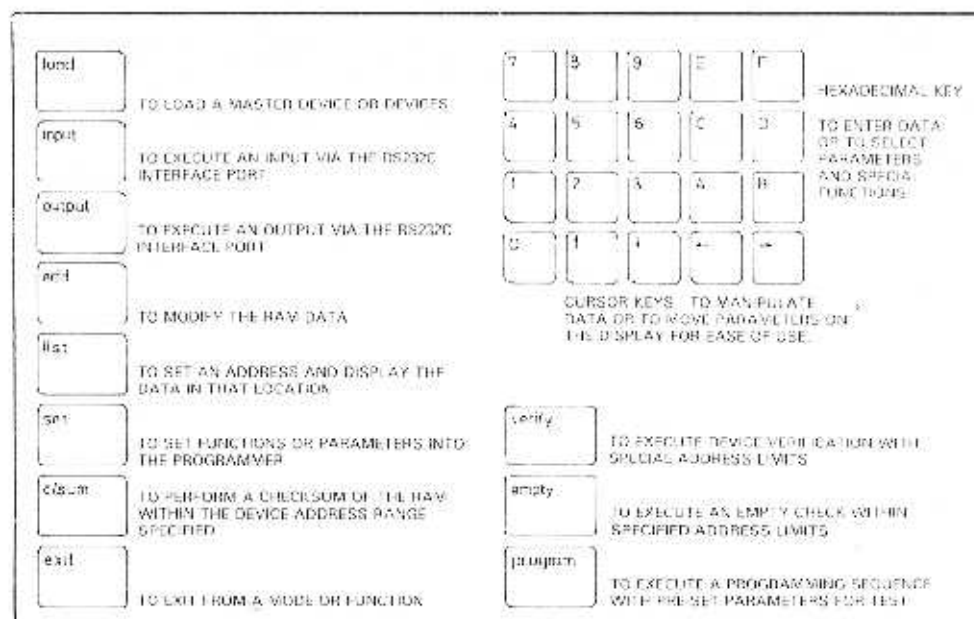
16-Character  
Green alphanumeric display

Rear Panel of PP38



## 1.2 The Keyboard

For data entry and operating programmer functions

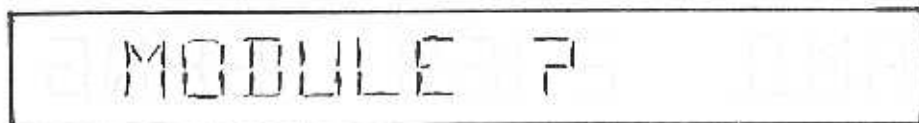


### 1.3 Initial Setting-up Procedure

Before attempting to apply power to your PP38 Programmer ensure that it is set to the correct operating voltage for your power source. The voltage setting is printed on the rear panel.

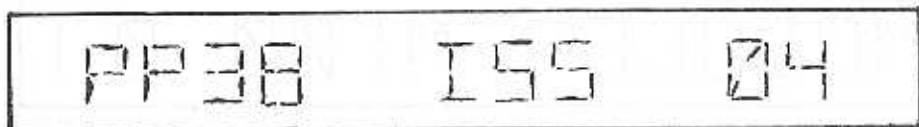
1. Plug the supplied power cord into the rear panel socket.
2. Apply power to the machine from the mains power source
3. Power-up the machine using the ON/OFF switch on the rear panel

After 'power-up' and without the 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.



PP38 ISS 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 initialization, power down before inserting a module. Always wait five seconds before applying power again.

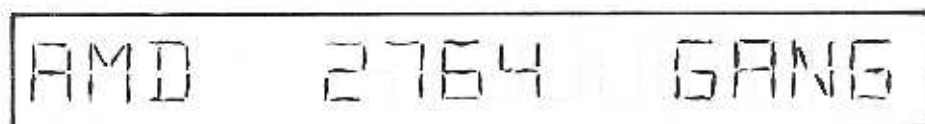
## Introduction of a Module to the Mainframe

Having completed the setting up procedure the PP38 is ready to receive its module.

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

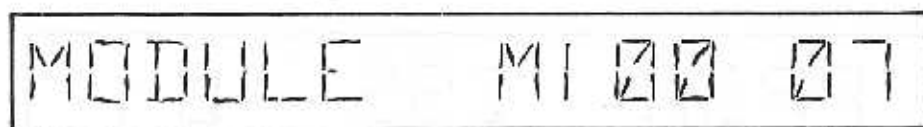
This ensures that once any machine parameter has been set-up, it needn't be reset every time the machine is switched on.

For instance, if the machine was previously used in the 'LOCAL' mode, 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 M1 00 07

To remove, press 'EXIT'.

## 1.4 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.
SEED	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 'RETURN' will bring control back to 'local' on the PP38 keyboard.
- (ii) If the PP38 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 PP38 might be:

Manufacturer.	Device type.	Bit mode.
SEED	5516A	MB

In local mode all functions of the PP38 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.
---------------	--------------	--------------

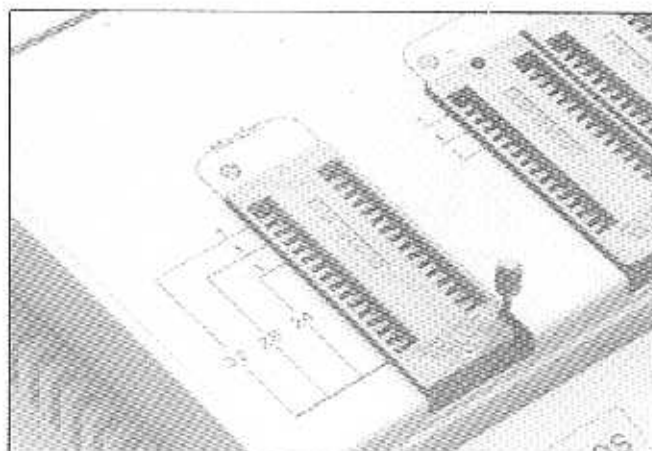
SEED	SS16A	REM
------	-------	-----

In the remote mode the PP38 operates under remote control from a computer or a terminal. The keyboard of the PP38 is inoperative at this time and the display will only show information as requested under remote control.

## 1.5 Correct Operation of ZIF Sockets

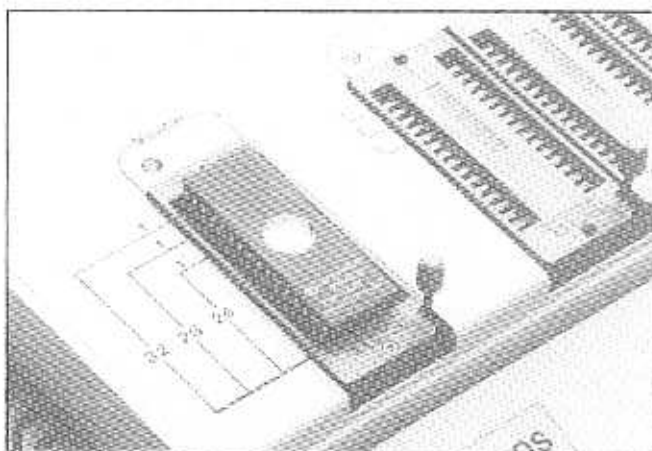
1. Empty ZIF socket.

Lever in open (up) position.



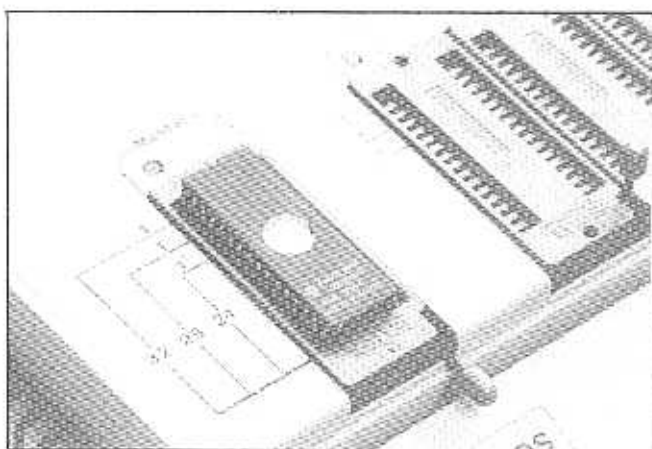
2. Device inserted.

Socket open.

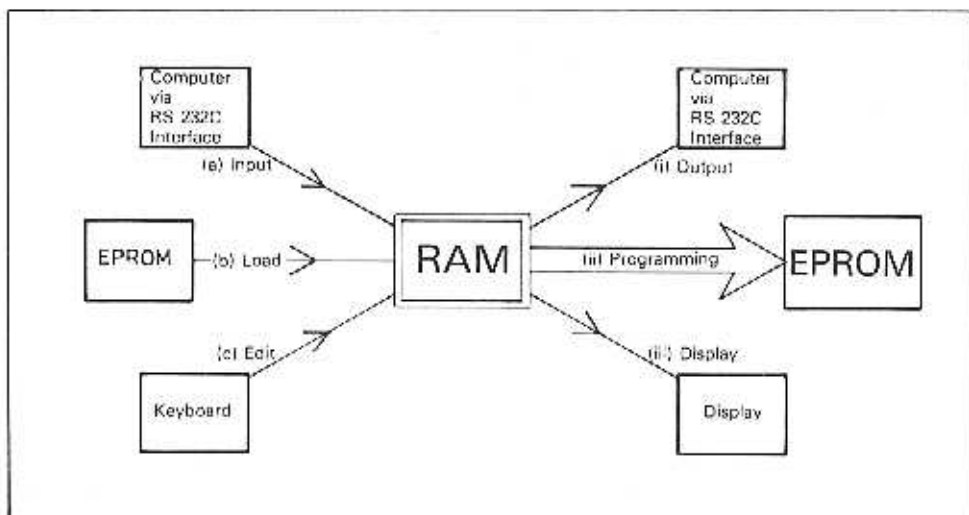


3. Device inserted.

Lever in closed (down) position.



## 1.6 RAM Operating Structure



Data can be loaded into the RAM of the PP38 from 3 different sources, these are:

- (a) **INPUT:** From a computer, terminal or microprocessor development system through the serial RS232C interface port.
- (b) **LOAD:** From a master device or devices (plugged into either of the module ZIF sockets) i.e. an EPROM or EEPROM.
- (c) **EDIT:** Controlled from the PP38 keyboard. Where data can be entered directly, or revisional adjustments can be made to the RAM data.

Data can be transferred from RAM in three ways:

- (i) **OUTPUT:** From RAM through the serial RS232C interface port to a computer, terminal or microprocessor development system.
- (ii) **PROGRAMMING:** The title function whereby data is finally conveyed from the RAM to an empty device or devices in the ZIF sockets.
- (iii) **DISPLAY:** Visual confirmation of operations undertaken.

## 1.7 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.

set4

Displays RAM size in hexadecimal

set5

RAM data complemented from lower to upper address limit

set6

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

set8

Calculates and displays CRC (Cyclic Redundancy Check)

set A1

To

A9

Saves machine configuration  
(up to nine sets)

## List of 'Set' Commands (continued)

**setBI** To **B9** Recalls previously saved machine configurations (up to 9 sets)

**setF0** Fills entire RAM with 00

**setFF** Fills entire RAM with FF

**setF1** 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.

**setF2** Fills RAM with arbitrary variable from lower to upper address limit

**setF4** Block Move: A block of data with pre-selected address limits can be copied and then re-located at another address within the RAM.

**setF6** 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

## List of 'Set' Commands (continued)

setEI

Electronic Identifier: Mode (i) Two Key Operation

setE2

Electronic Identifier: Mode (ii) Single Key Operation

set9

String Search

setFE

Applicable to MOTOROLA 2816 ONLY—Erases Device

## SECTION 2

stag

---

Sophisticated systems for the discerning engineer.



## 2 Device Selection

### 2.1 Selecting a Device

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

The complete list of devices supported by the PP38 is stored in the programmer. Each individual device has its own four digit code.

SET 0—Allows code selection

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

For example:

AMD	2716	MB
-----	------	----

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

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), it can be entered directly onto the display from the keyboard replacing the old code:

DEVICE CODE	AF44
-------------	------

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

FUJ	2732	MB
-----	------	----

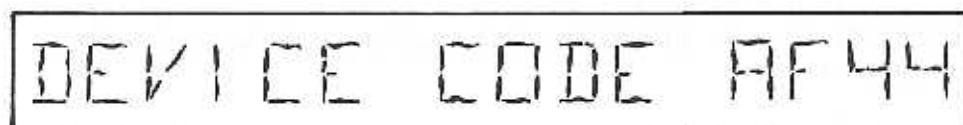


## Scanning device types and manufacturers by use of cursor keys

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:

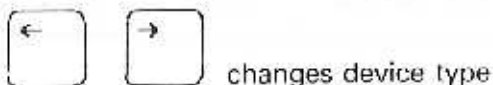


The manufacturer and device type can be changed by use of the cursor keys:

The up/down keys scan the range of manufacturers.



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



## 2.2 Electronic Identifier

### Important Note:

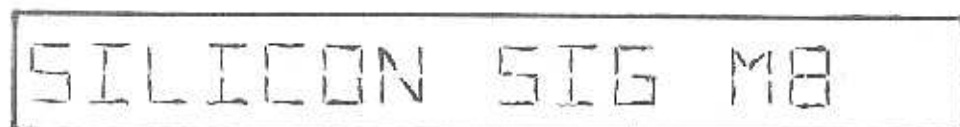
Devices which do not contain an Electronic Identifier may be irreparably damaged if they are used in the Silicon Sig mode.

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 PP38 to directly identify any device containing an Electronic Identifier and thus eliminate the need for the user to select the device type.

The PP38 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" adjacent to the selected bit-mode:



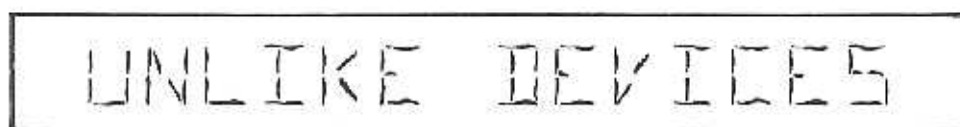
SILICON SIG M8

If any device function key is pressed e.g. Program, Load etc. the PP38 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 PP38'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

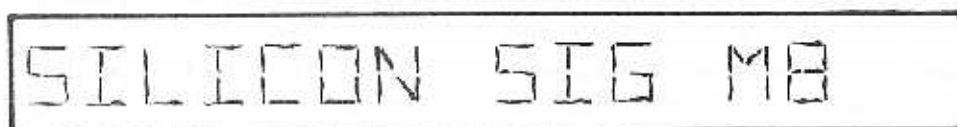
If neither of the above two fault conditions occur then the manufacturer and the device type will be displayed. In the case of devices in both sockets the manufacturer and device code of the device in the left socket will be displayed.

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" adjacent to the selected bit-mode.



SILICON SIG MB

Operation is similar to the previously described mode except that the PP38 rather than stopping to display the manufacture 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 3

stag

---

Sophisticated systems for the discerning engineer.



### 3 Bit Modes

#### 3.1 Selection of Bit Mode Configuration

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

AMD 2764 M16

By pressing SET 3 the bit mode alone 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-GANG

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

AMD 2764 M8

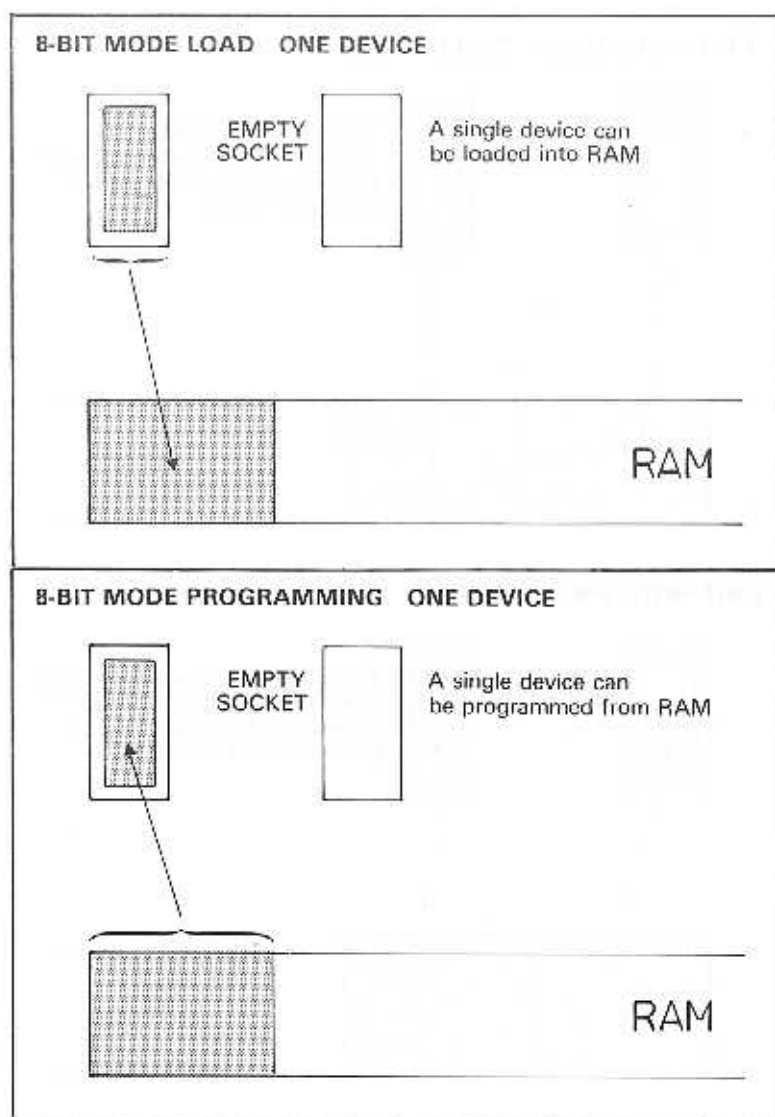
### 3.2 8-Bit Mode

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 PP38 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.



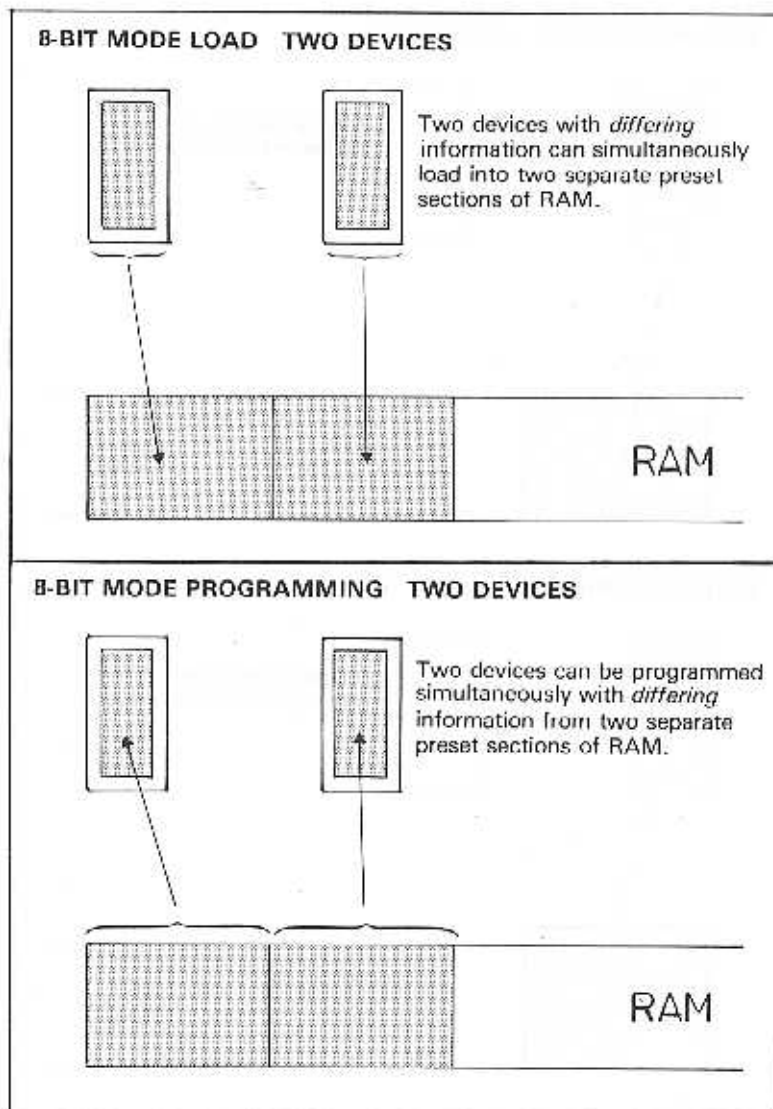
## 8-Bit Mode

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

### Two devices

- (i) The PP38 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.



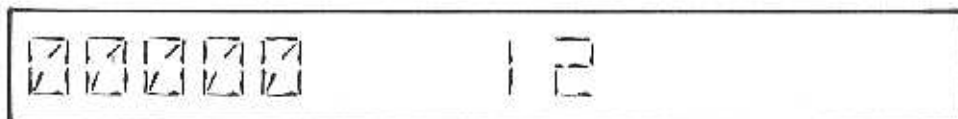
## Display

### 8-Bit Mode

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

ADDRESS (ZERO)

TWO HEX CHARACTER



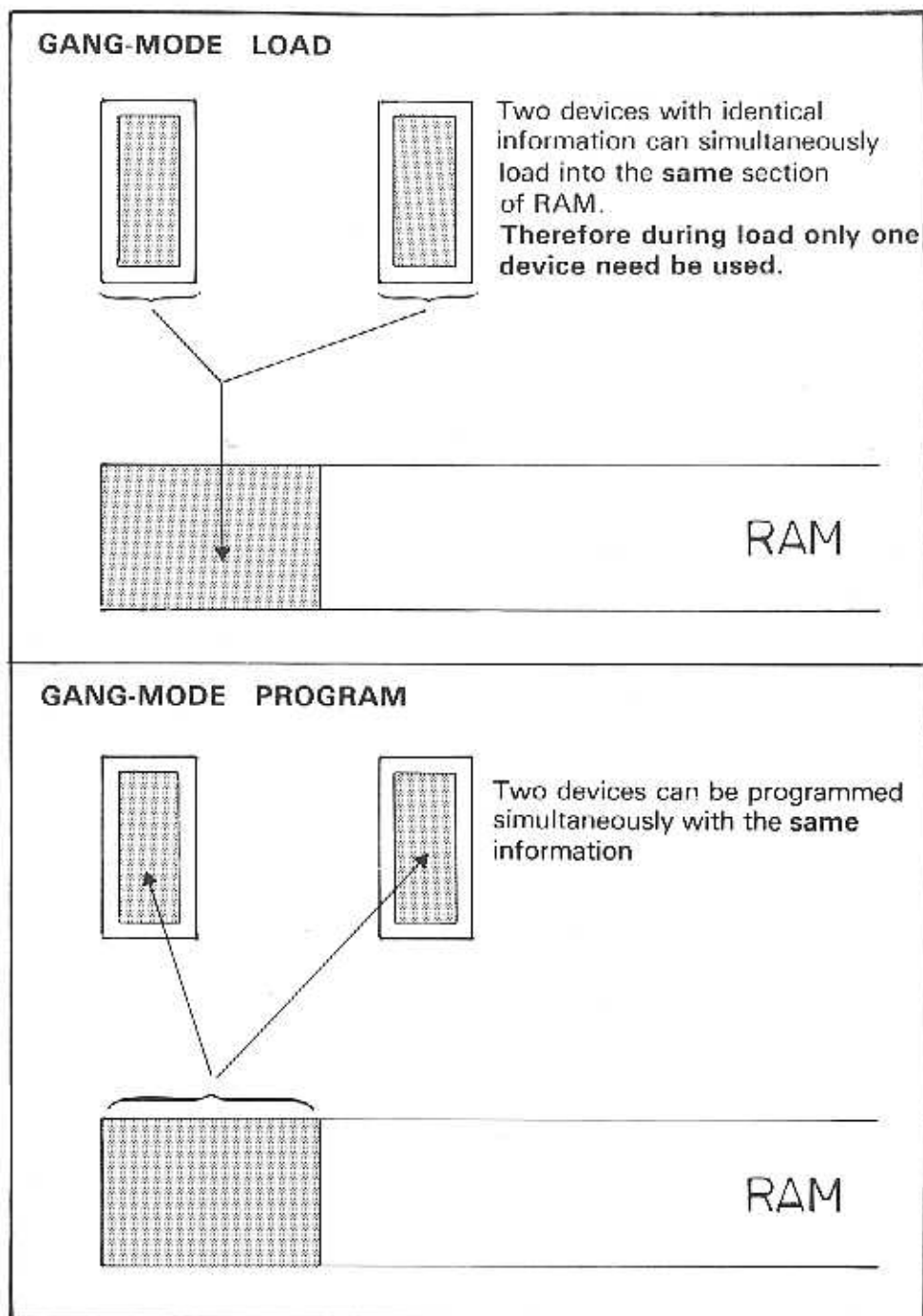
(Under the 'List' function THE ADDRESS RANGE can be scanned by use of the hex keyboard and cursor keys).



### 3.3 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.



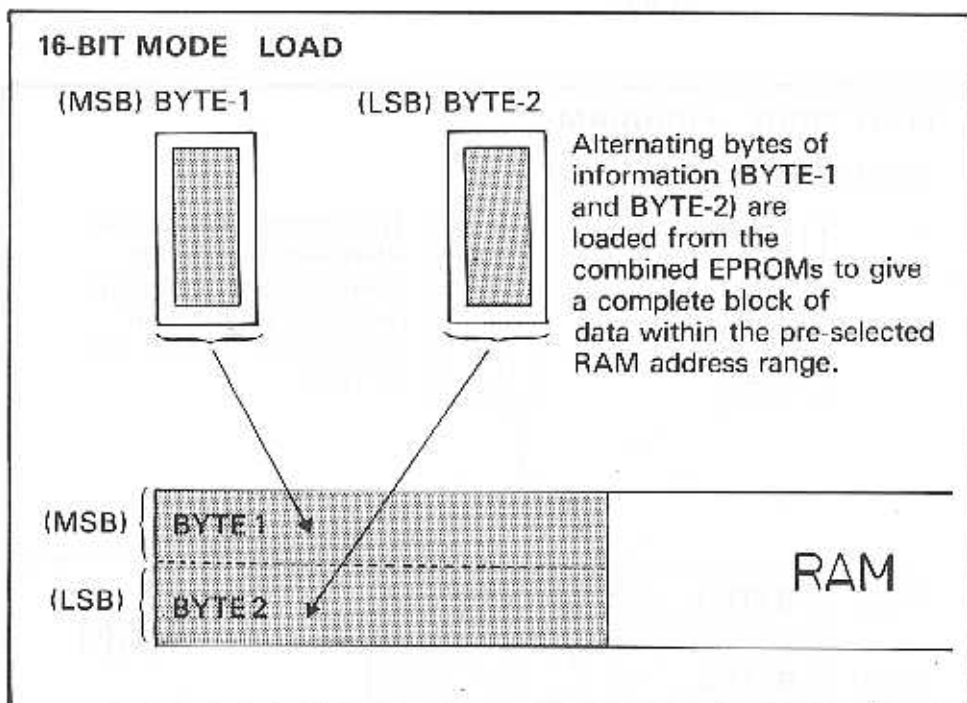
### 3.4 16-Bit Mode

**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

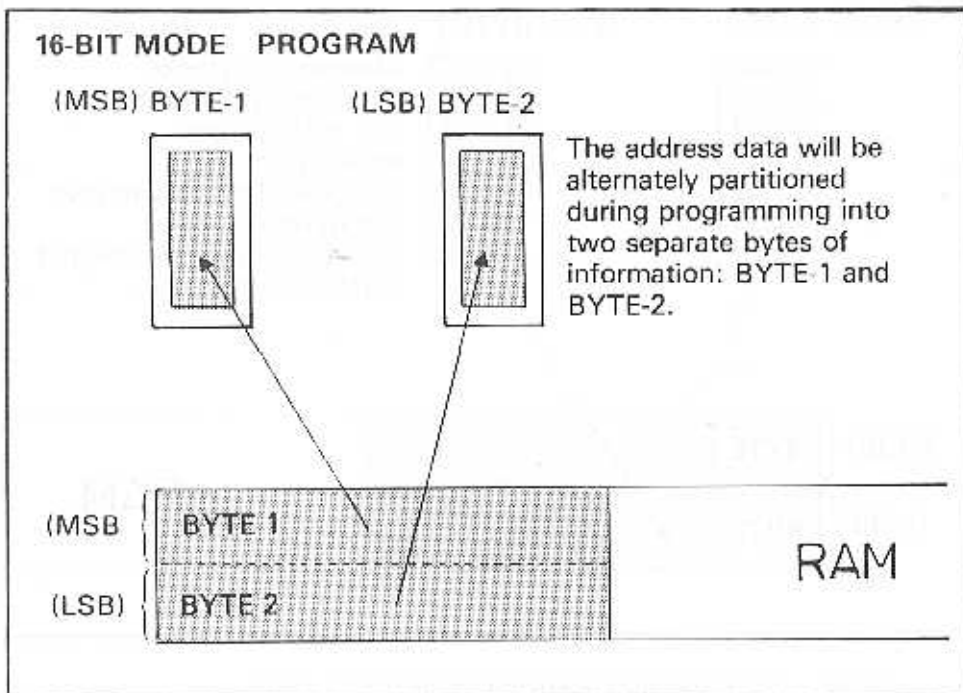
## 16-Bit Mode

**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.

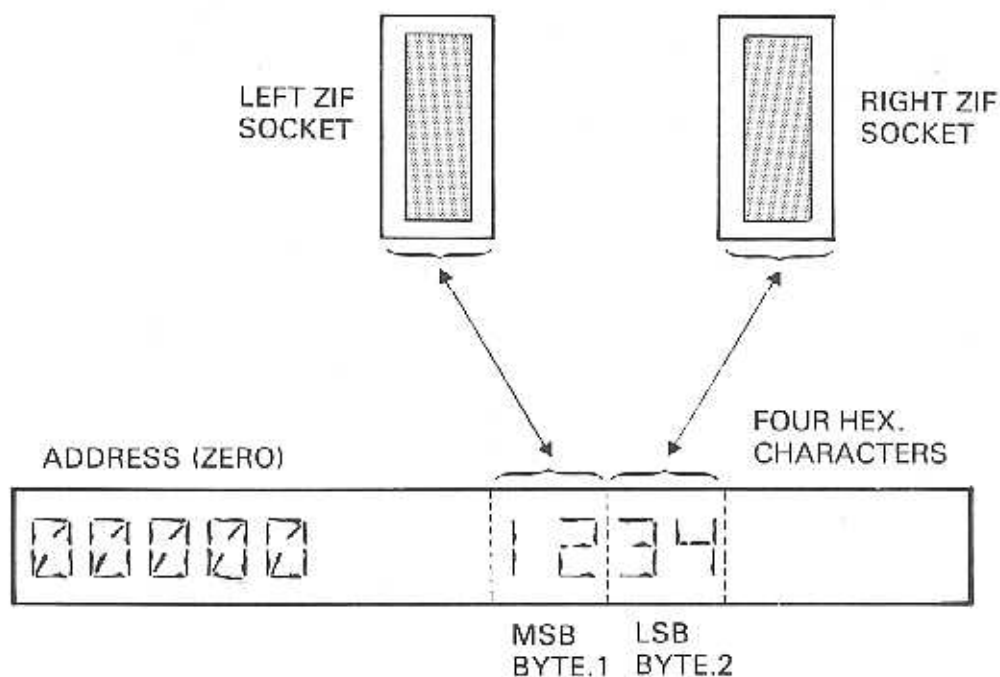


MSB—The most significant BYTE in binary code

LSB—The least significant BYTE in binary code

## Display

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



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

### 3.5 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

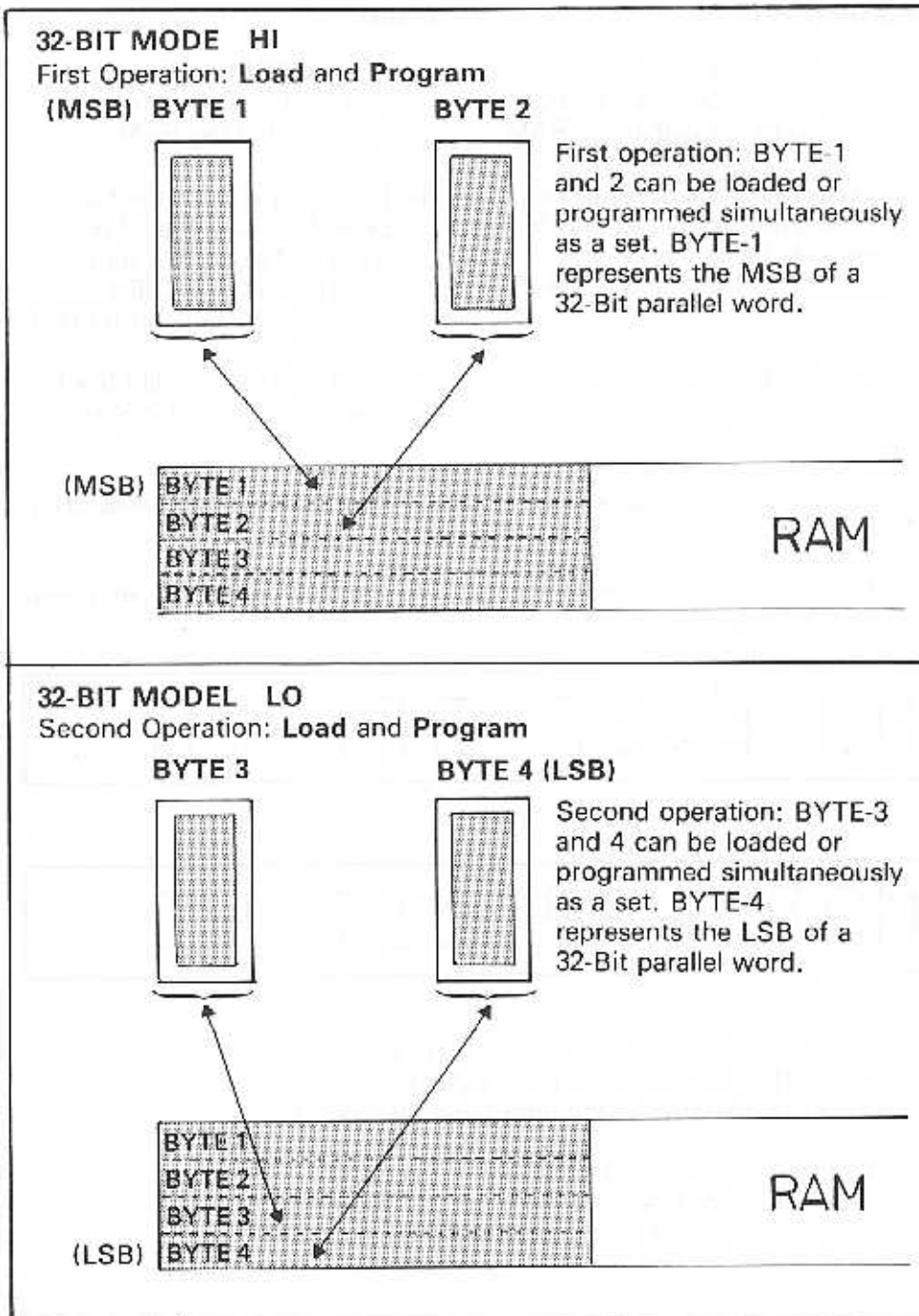
AND

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

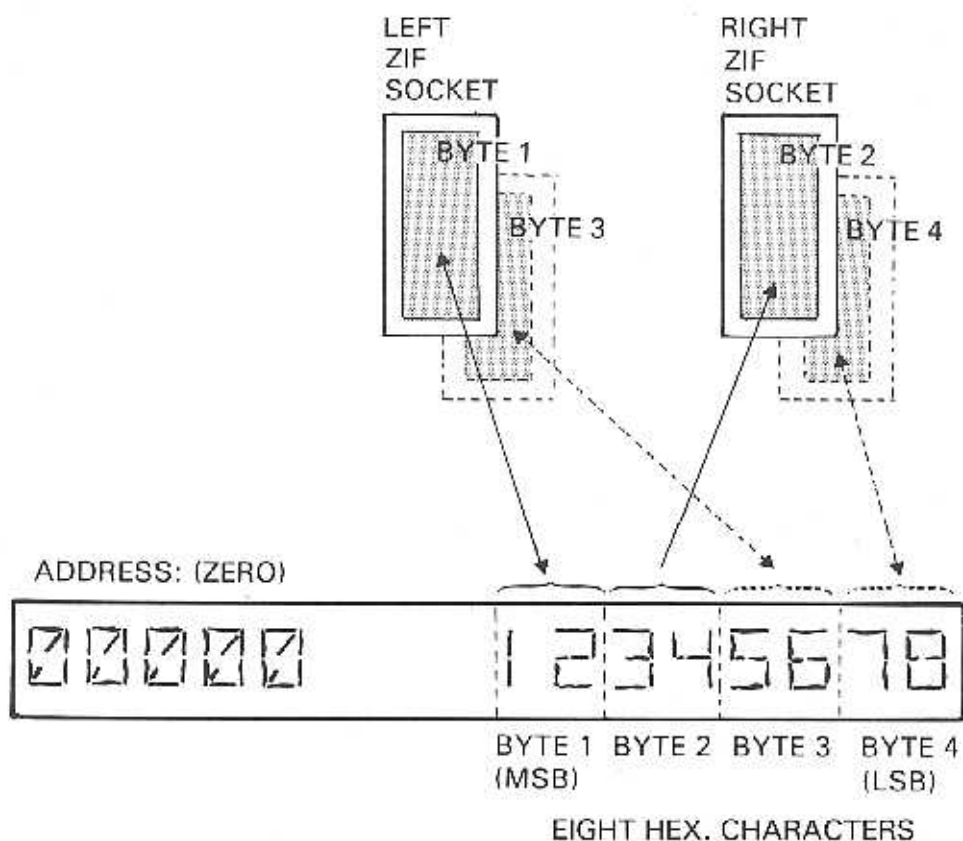


MSB—The most significant BYTE in binary code.

LSB—The least significant BYTE in binary code.

## Display

The four Bytes can be **displayed** as eight hex. characters by pressing 'List'.



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

## SECTION 4

stag

---

Sophisticated systems for the discerning engineer.





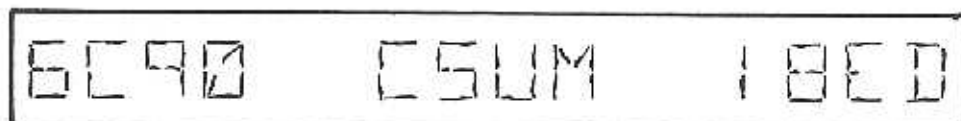
## 4 Device Functions

### 4.1 Load

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

To set Device/RAM Address Limits, see Section 4.7.

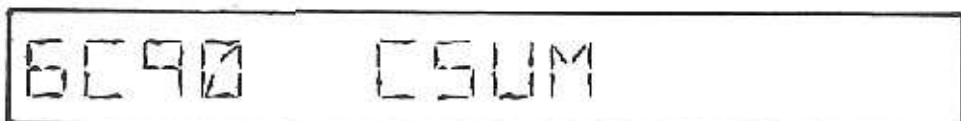
Insert the master device or devices into the ZIF sockets. Press the load key. On completion of load the display will show:



6C90 CSUM 18ED

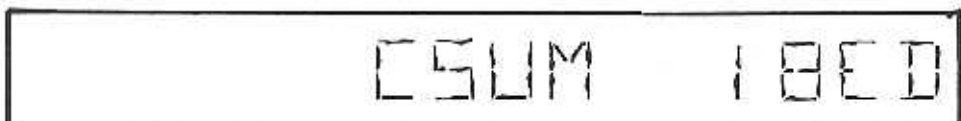
BOTH ZIF SOCKETS IN USE

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 18ED

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.

## 4.2 Empty Test

To set Device/RAM Address Limits, see Section 4.7.

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.

## Programming Sequence

If the empty test passes or is unnecessary the programming can begin. To set Device/RAM Address Limits, see Section 4.7.

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.

### 4.3 Pre-program Bit Test

The PP38 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:

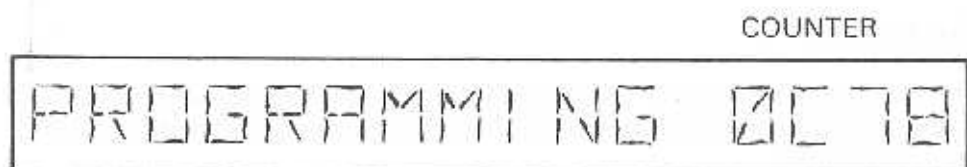
BOTH ZIF SOCKETS IN USE		
0000	LR-02	LP-04
LOCATION (ZERO)	LEFT RAM (02)	LEFT PROM (04)

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.

#### 4.4 Programming

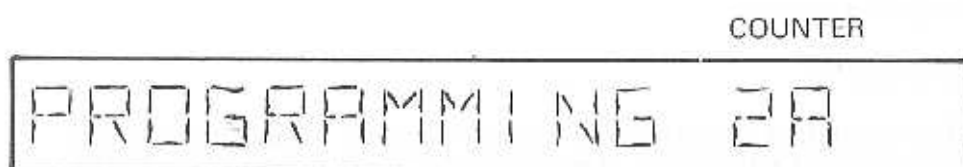
Once the device has passed the bit test, programming of that device will start.

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



FOUR DIGIT ADDRESS

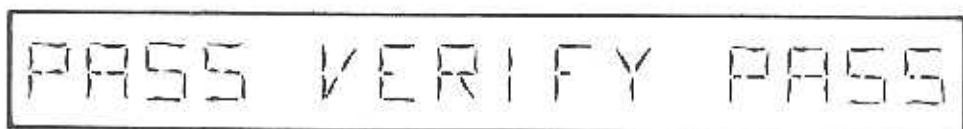
In the case of the larger devices which use a fast algorithm only the two most significant digits of the address are displayed.



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:



LEFT ZIF SOCKET

RIGHT 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 executed.

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.

## 4.5 Verify

### In-program Verify

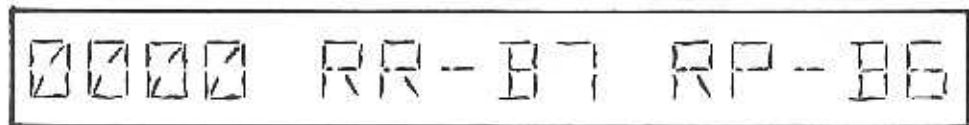
This is a feature where each location, as it is programmed, is checked to see that it is identical to the corresponding data byte in the RAM.

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.

## 4.6 Checksum and CRC

### Checksum

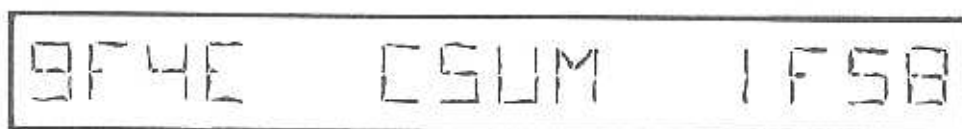
To do a checksum press the c'sum key.

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:

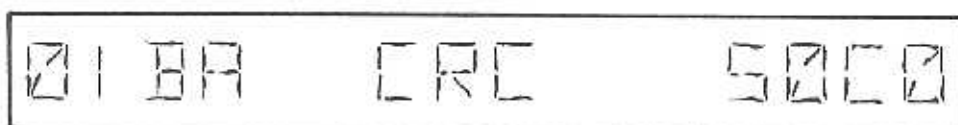


9F4E CSUM 1F5B

### 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 values.

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



018A CRC 50C0

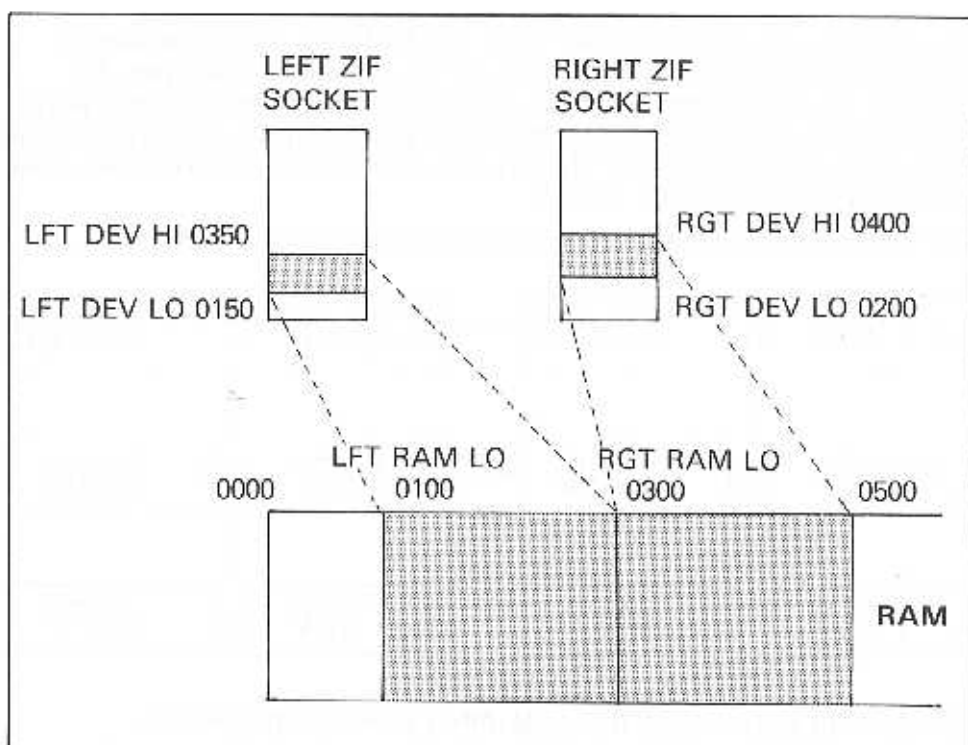
As with checksum the CRC function can distinguish between different modes and sockets that are not in use, therefore the display will follow a similar format.

#### 4.7 Device/RAM Address Limits (Set F6)

All functions of the PP38 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 RAM and the Device Address Limits

To set address limits press set F6

The display will first show:

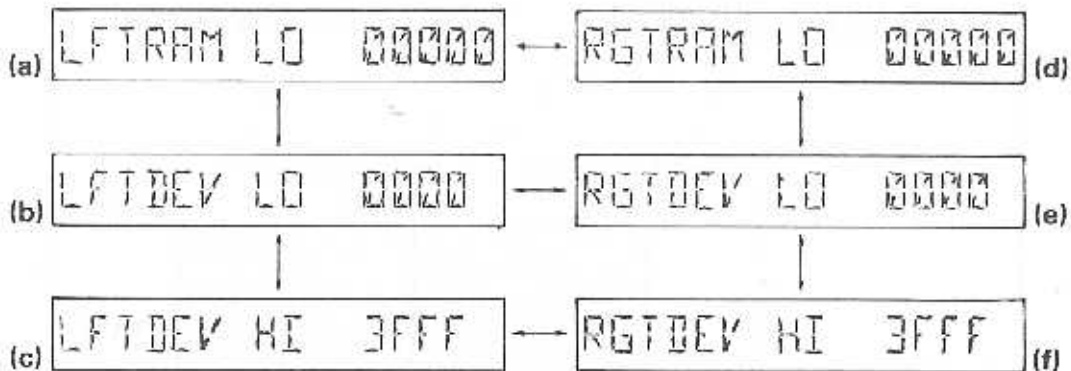
LEFT RAM LOW

ZERO

LFTRAM LO 000000

By using the up or down cursor keys, the 3 address limits available for the RAM and device in the left hand ZIF socket will be displayed. By pressing the right hand cursor key and then the up or down cursor keys the 3 address limits available for the RAM and the device in the right hand ZIF 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:

RGT RAM LO 040000

- (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.

## **4.8 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 configuration are 'Set A1' through to 'Set A9' inclusive.

### **'RECALL' Machine Configuration**

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

#### **(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 5

stag

---

Sophisticated systems for the discerning engineer.



## 5 RAM Functions

### 5.1 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
00000000	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

edit

	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.

## 5.2 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 use 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	B 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:

														LOCATIONS	
		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

### 5.3 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										LOCATIONS				
001FD		001FE		001FF		00200		00201		00202		00203		
RAM DATA	1	7	0	A	3	7	5	B	A	A	6	3	7	2

'5B' DELETED

CURRENTLY DISPLAYED LOCATION

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).

	001FD		001FE		001FF		00200		00201		00202		00203	
RAM DATA	1	7	0	A	3	7	A	A	6	3	7	2	3	F

CURRENTLY DISPLAYED LOCATION

## 5.4 Block Move (Set F4)

### SETTING ADDRESS LIMITS

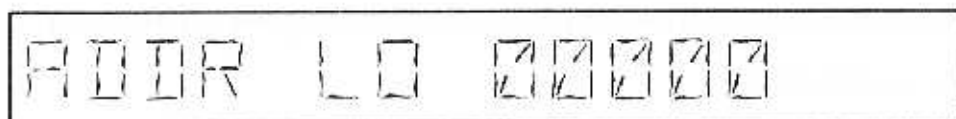
This is a feature enabling a block of data with pre-selected address limits to be relocated at another address within the RAM, without destroying the original data.

Selection of this function is made by pressing SET F4.

The display will show:

ADDRESS LOW

ZERO



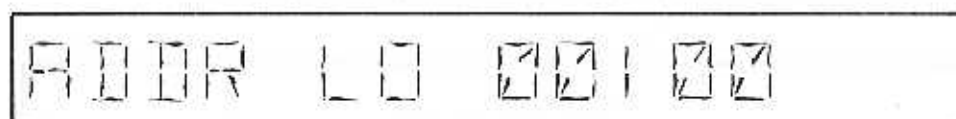
A hex display with two sections. The left section is labeled 'ADDRESS LOW' and shows '0000'. The right section is labeled 'ZERO' and shows '000000'.

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

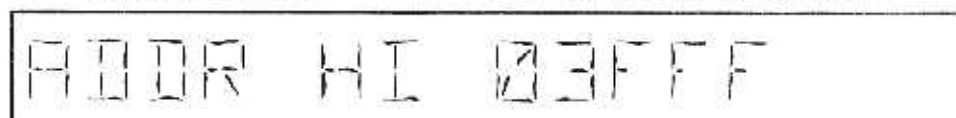


A hex display with two sections. The left section is labeled 'ADDRESS LOW' and shows '00100'. The right section is labeled 'NEW LOWER RAM LIMIT' and shows '000000'.

If the down cursor is pressed the display will show:

ADDRESS HIGH

SIZE OF SELECTED DEVICE



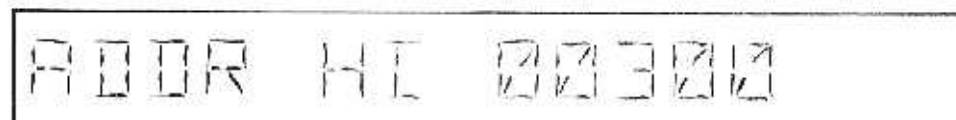
A hex display with two sections. The left section is labeled 'ADDRESS HIGH' and shows '0000'. The right section is labeled 'SIZE OF SELECTED DEVICE' and shows '03FFFF'.

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



A hex display with two sections. The left section is labeled 'ADDRESS HIGH' and shows '00300'. The right section is labeled 'NEW UPPER RAM LIMIT' and shows '000300'.

### Lower limit of Re-located Data

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

TO ADDRESS

TO ADDR 000000

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

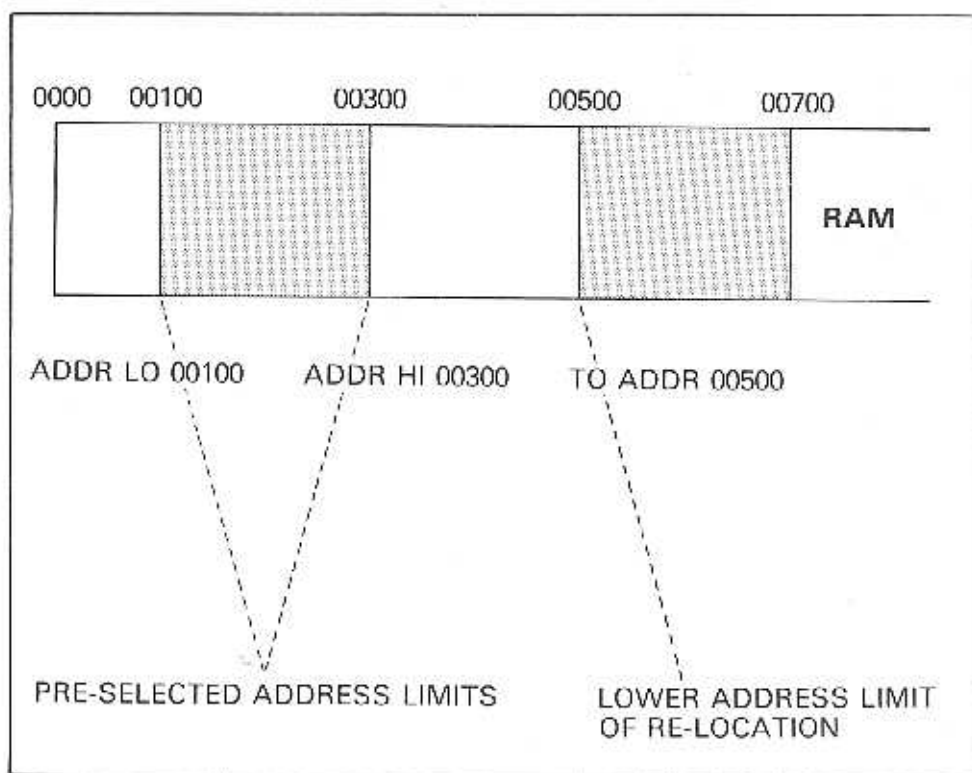
TO ADDR 00500

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

-----

The PP38 will automatically return to the normal operating mode.

A graphic example of how the Block Move function works is shown below:



## 5.5 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).

### 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 upon 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
0000	00000000

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:

4 BYTES (8 Characters)

DATA 12345678

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

00000 12345678

and

00001 12345678

## 5.6 Complement

Complement is a modification of RAM data made by inverting the individual bits that make up the data bytes.

Complement will change 'ones' to 'zeros' and 'zeros' to 'ones'.

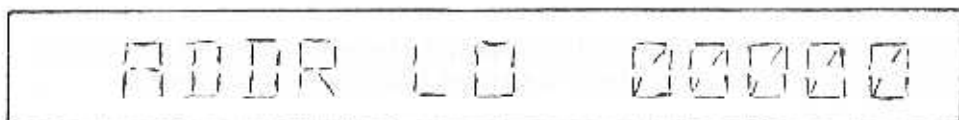
To complement the RAM data:

Press Set 5.

## 5.7 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:

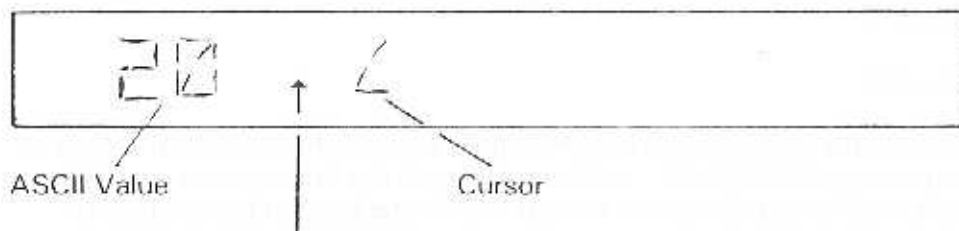
Press ↑ or ↓



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:

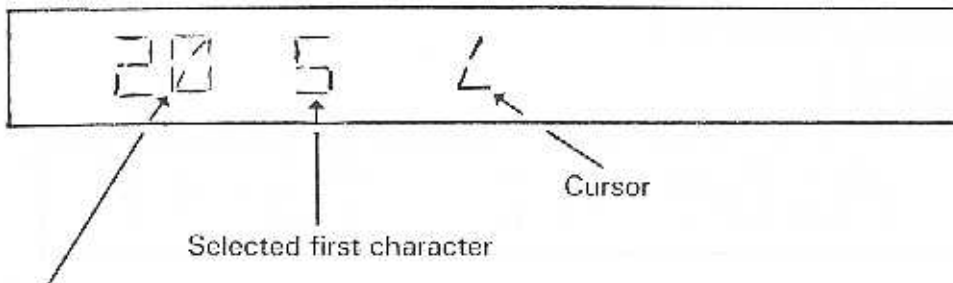
Press ↑ or ↓

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')

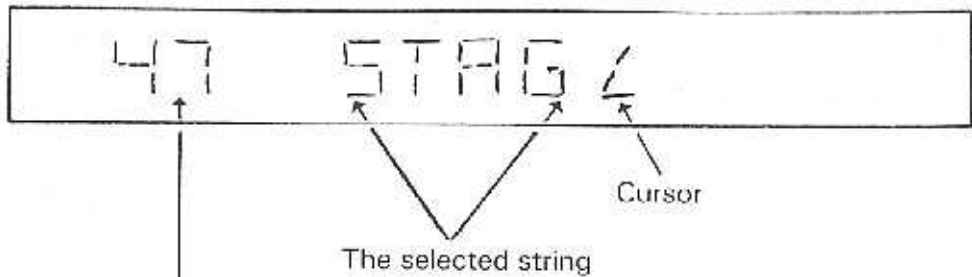
The second character can now be selected in the manner previously described. In this way a string of upto 11 characters (or data bytes) can be entered.

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

Press SET

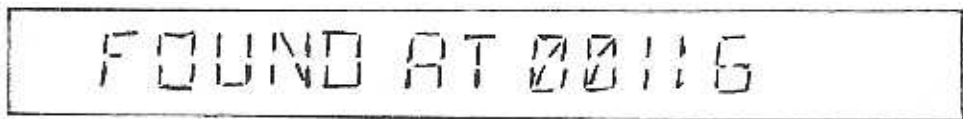
If a corresponding string is located within the specified area of RAM, then the message 'FOUND AT' and the address of the first occurrence will be displayed. Every subsequent occurrence can be located by continually pressing SET until the entire specified area of RAM has been searched.

For instance:



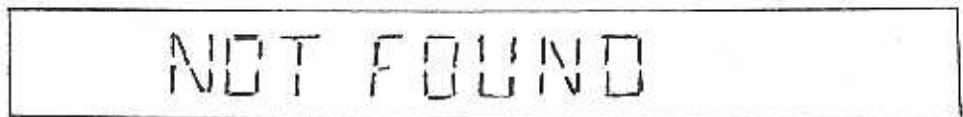
The ASCII value of the character to the immediate left of the cursor. (In this case 'G')

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 to be 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 PP38 is powered down.

To abort the String Search at any time.

Press EXIT

## SECTION 6

stag

---

Sophisticated systems for the discerning engineer.



## 6 Interface

### 6.1 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 SET1.

For instance:

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

There are five categories of I/O interface parameter available for selection on the PP38 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

The selection available in each category is shown in this table

FORMAT	BAUD RATE	WORD LENGTH	NUMBER OF STOP BITS	PARITY
INT	19K2	8	2	—
XINT	9600	7	1	SP
HASC	7200	6		MP
XOR	4800	5		EP
XXOR	3600			OP
TEK	2400			
XTEK	1800			
PPX	1200			
BIN	600			
DBIN	300			
BINR	150			
	135			
	110			
	75			
	50			

FORMAT	
Key to Abbreviations	
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

PARITY	
Key to Abbreviations	
—	= NO PARITY
SP	= SPACE PARITY
MP	= MARK PARITY
EP	= EVEN PARITY
OP	= ODD PARITY



## Selection

Selection can be made by using the Cursor Keys:



To change one of the five parameter fields it is first necessary to rotate the chosen field to the left-most area of the display.

This can be done by using the horizontal cursor keys.



For instance:

PARAMETER SELECTED  
FOR CHANGE

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

LEFT MOST  
AREA OF DISPLAY

Once the chosen parameter is positioned on the left of the screen it can be modified by using the up/down cursor keys:



For instance:

NEW SELECTION

BAUD RATE	WORD LENGTH	No. OF STOP BITS	PARITY	FORMAT
7200	8	2	EP	INT

LEFT MOST  
AREA OF DISPLAY

All values of each parameter can be scanned and chosen in this way from the table stored in the programmer.

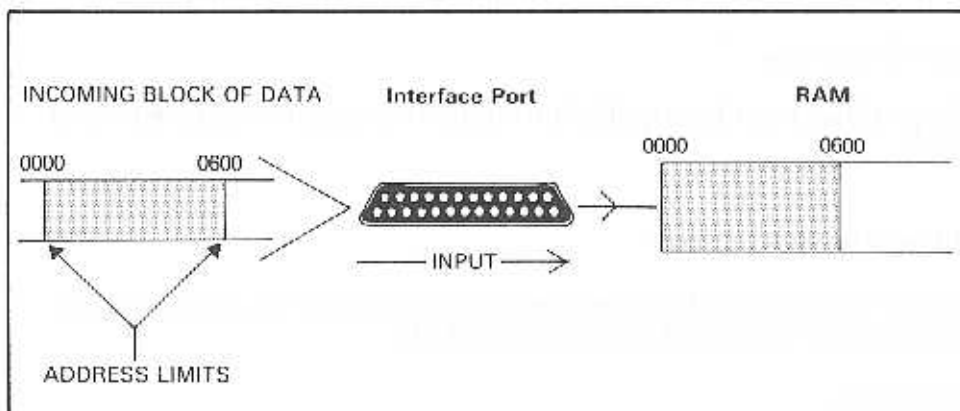
On completion of parameter selection – press EXIT key to return to the normal operation mode.

## 6.2 Input/Output Operation

### Input

Pressing the input key alone initiates the input operation.

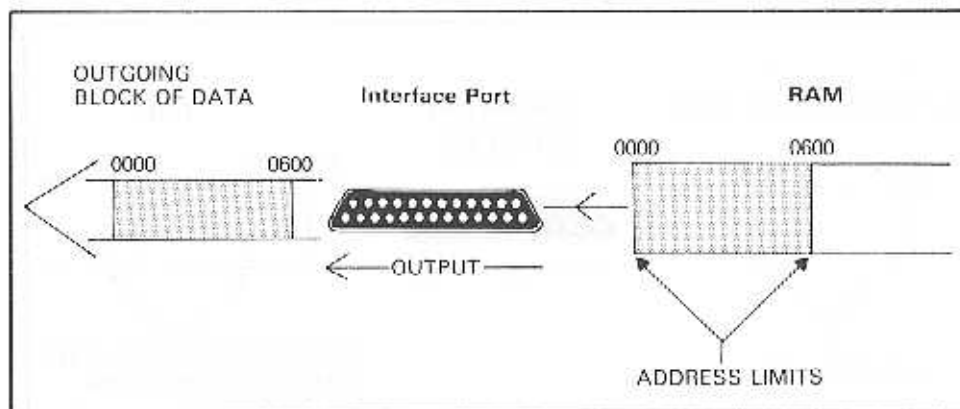
### Operation of Input



### Output

Pressing the output key alone initiates the output operation.

### Operation of Output



## To Pre-set I/O Address Parameters

The pre-settable address parameters are:

INPUT ADDRESS OFFSET
----------------------

OUTPUT ADDRESS OFFSET
-----------------------

OUTPUT START ADDRESS
----------------------

OUTPUT STOP ADDRESS
---------------------

### Input Parameters

Only an offset need be specified on input. This operation is called 'input offset'.

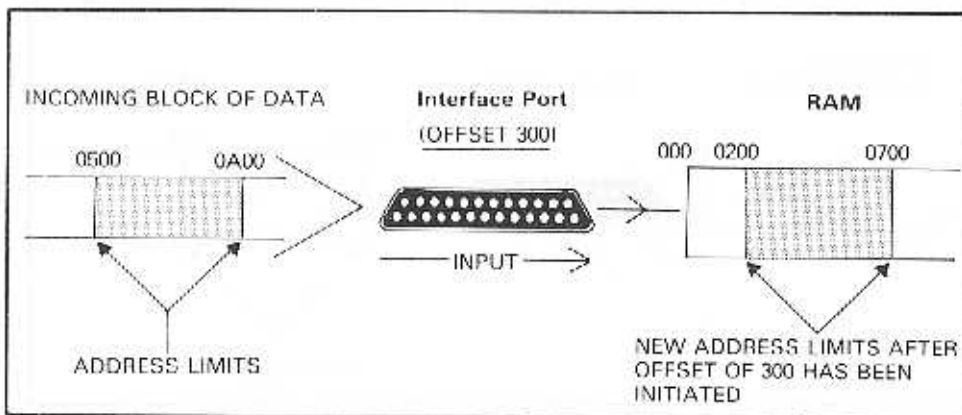
### Structure of 'Input Offset'

An incoming block of data originating from peripheral equipment can be re-located at a lower address within the RAM.

For example:

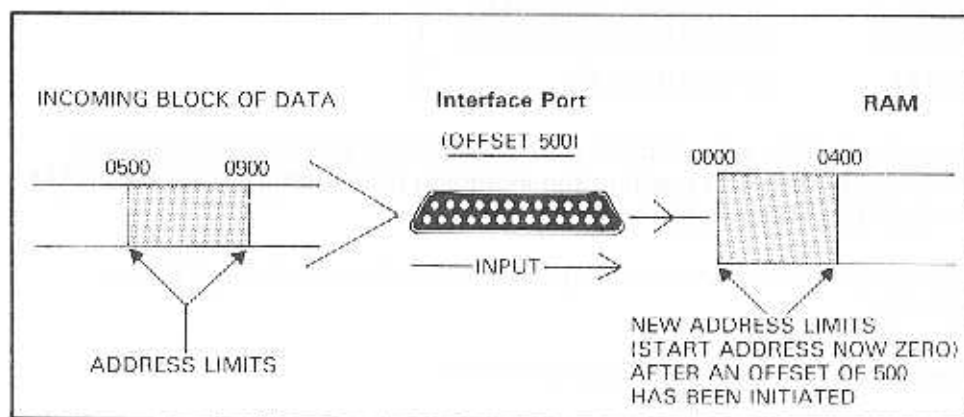
OFFSET      0300
------------------

An offset selection of 300 would look like this:



A more widely used example is:

When an offset is specified so that the start address of the incoming block of data becomes 0000 (ZERO) in the RAM, i.e.:



#### Setting-up the Input Offset

By pressing set input the last entered offset will be displayed for instance:

OFFSET 1386

OR

OFFSET 0000 = NO OFFSET

By use of the keyboard the new offset can be entered: e.g. offset 1A00.

OFFSET 1A00

Pressing the input key after entering the offset address will initiate the input operation, whereby an incoming block or blocks of data will be entered into the RAM via the RS232C port from peripheral equipment with an offset of 1A00.

## Extended Formats (Input Offset)

The extended formats available are:

XINT	—	EXTENDED INTELLEC
XXOR	—	EXTENDED EXORCISOR
XTEK	—	EXTENDED TEK

The extended formats contain a larger address field than the standard formats. Therefore to position the incoming data within the PP38's RAM an extended offset needs to be specified.

This can be shown when setting the offset with an extended format selected.

It will be displayed as an eight digit figure:

OFFSET	1	2	3	4	5	6	7	8
--------	---	---	---	---	---	---	---	---

This offset can be changed in the same manner as the four digit offsets by use of the hexadecimal keyboard.

## Formats Without Address Fields

As the formats Hex ASCII (HASC), Binary (BIN) or Binary Rubout (BINR) do not contain an address field, no offset is required. Therefore the display will show:

NO OFFSET REQ'D

When an input operation occurs under one of these formats the incoming data will always be loaded into RAM starting at address ZERO.

## Format Offset Types

FORMAT	DISPLAY ABBREVIATION	OFFSET SIZE
INTELLEC	INT	4 Digits
EXTENDED INTELLEC	XINT	8 Digits
HEX ASCII	HASC	Not Required
EXORCISOR	XOR	4 Digits
EXTENDED EXORCISOR	XXOR	8 Digits
TEK HEX	TEK	4 Digits
EXTENDED TEK-HEX	XTEK	8 Digits
STAG-HEX*	PPX	4 Digits
BINARY	BIN	Not Required
DEC BINARY	DBIN	4 Digits
BINARY RUBOUT	BINR	Not Required

(THIS TABLE ALSO APPLIES TO OUTPUT)

## Output Parameters

When using output, three parameters are available.

These are:

- |     |                       |
|-----|-----------------------|
| (a) | OUTPUT ADDRESS OFFSET |
| (b) | OUTPUT START ADDRESS  |
| (c) | OUTPUT STOP ADDRESS   |

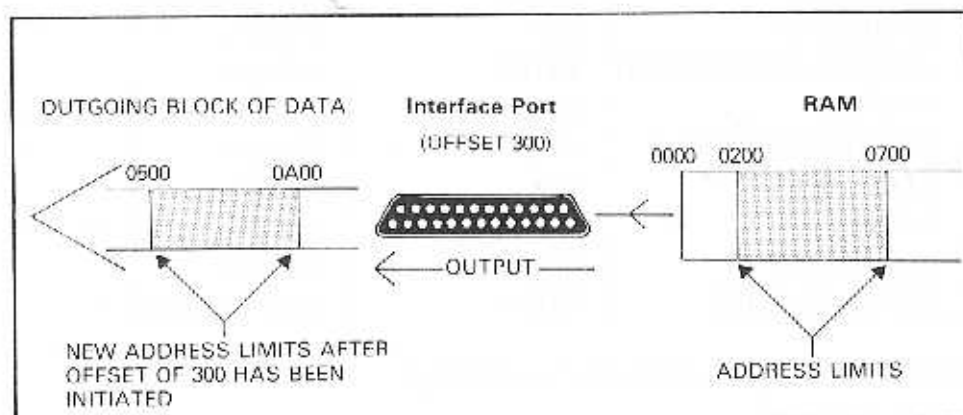
### (a) Output Address Offset

A block of data originating from the RAM can be re-located at a higher address.

For example:

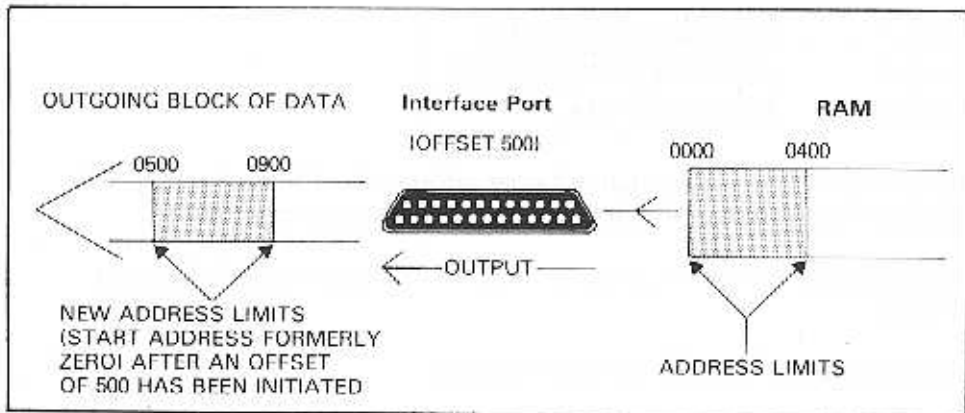
OFFSET      0300
------------------

An offset of 300 would look like this:



**A more widely used example is:**

When relocation to a higher address occurs from a start address of 0000 in the RAM:



**Setting-up the Output Offset**

By pressing Set Output the last entered offset will be displayed for instance:

OFFSET 1386

OR

OFFSET 0000 = NO OFFSET

By use of the keyboard the new offset can be displayed:  
e.g. offset 1A00

OFFSET 1A00

Pressing the output key after selecting the offset address will initiate the output operation whereby an outgoing block or blocks of data will be re-located via the RS232C.



## Extended Formats (Output Offset)

The extended formats available are:

XINT	—	EXTENDED INTELLEC
XXOR	—	EXTENDED EXORCISOR
XTEK	—	EXTENDED TEK

The extended formats contain a larger address field than the standard formats, therefore a correspondingly large offset is required.

This can be shown when setting the offset with an extended format selected.

It will be displayed as an eight digit figure:

OFFSET	1	2	3	4	5	6	7	8
--------	---	---	---	---	---	---	---	---

This offset can be changed in the same manner as the four digit offsets by use of the hexadecimal keyboard.

## Formats Without Address Fields

As the formats Hex ASCII (HASC), Binary (BIN) or Binary Rubout (BINR) do not contain an address field, no offset is required. Therefore the display will show:

NO OFFSET REQ'D

When an output operation occurs under one of these formats the outgoing data will always be dumped from RAM starting at address ZERO.

## Format Offset Types

FORMAT	DISPLAY ABBREVIATION	OFFSET SIZE
INTELLEC	INT	4 Digits
EXTENDED INTELLEC	XINT	8 Digits
HEX ASCII	HASC	Not Required
EXORCISOR	XOR	4 Digits
EXTENDED EXORCISOR	XXOR	8 Digits
TEK-HEX	TEK	4 Digits
EXTENDED TEK-HEX	XTEK	8 Digits
STAG HEX *	PPX	4 Digits
BINARY	BIN	Not Required
DEC BINARY	DBIN	4 Digits
BINARY RUBOUT	BINR	Not Required

(THIS TABLE ALSO APPLIES TO INPUT)

## Output Start Address and Output Stop Address

In addition to an offset, a start and stop address can be used on output.

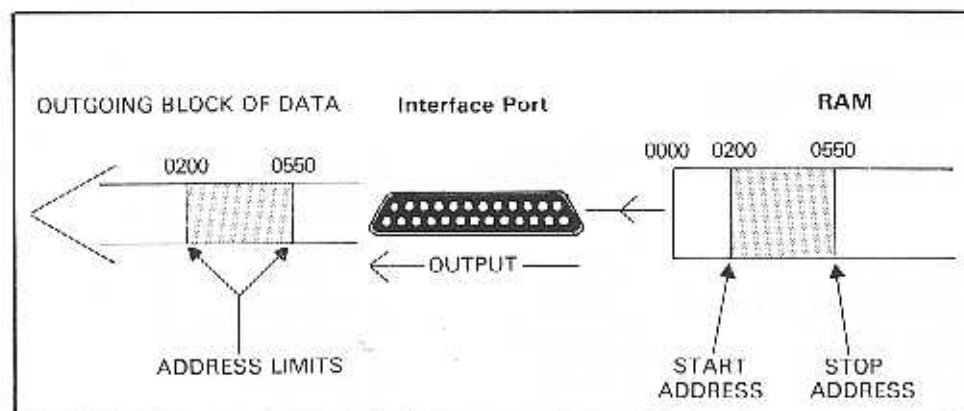
### Structure of Output Start and Stop Addresses

The block of data to be output within the RAM can be specified by using start and stop addresses.

For example:

START ADDRESS 0200

STOP ADDRESS 0550



### Setting-up Start and Stop Addresses

The display will first show the last entered offset, e.g.

OFFSET 0000

By pressing either the up or down cursor keys



the last entered start and stop addresses will be displayed, e.g.:

START 00000

STOP 00FFF

CORRESPONDS TO  
EPROM SIZE 2732

The stop address will default to the selected device size whenever a device is selected or the mode changed. For example if INTEL 2764 is selected the STOP address will be set to IFFF.

By use of the keyboard the new start and stop addresses will be displayed, e.g.

START 00200

STOP 00550

## Error Reporting on Input and Output

The following table shows the 12 possible error messages that will be displayed instead of the checksum on completion of either input or output or when 'exit' is pressed.

### Reported at the end of data transfer

- |                   |  |
|-------------------|--|
| (1) PARITY ERROR  | — A parity error was detected.   |
| (2) FRAMING ERROR | — A pulse on the serial signal was not of an acceptable size.                        |
| (3) PTY/FMG ERROR | — Parity/Framing: A combined parity and framing error was detected.                  |
| (4) OVERRUN ERROR | — Data was lost due to overwriting of secondary information in UART                  |
| (5) PTY/OVN ERROR | — Parity/Overrun: A combined parity and overrun error was detected.                  |
| (6) FMG/OVN ERROR | — Framing/Overrun: A combined framing and overrun error was detected.                |
| (7) PY/FR/O ERROR | — Parity/Framing/Overrun: A combined parity, framing and overrun error was detected. |
| (8) L CSUM ERROR  | — Line Checksum Error: A checksum failure in a record (line) was detected.           |
| (9) NON-HEX ERROR | — A non-hex character was received where a hex character was expected.               |

### Reported when exit key is pressed

- |                    |                             |
|--------------------|-----------------------------|
| (10) CTS LOW ERROR | — Pin 5 Clear to Send       |
| (11) DSR LOW ERROR | — Pin 6 Data set ready      |
| (12) DCD LOW ERROR | — Pin 8 Data carrier detect |

The pin indicated on the PP38 RS232C connector was held low by an external device, hence impairing communications.

## SECTION 7

stag

---

Sophisticated systems for the discerning engineer.



## 7 Format Descriptions

### 7.1 Interface Formats (Introduction)

There are eleven formats available on the PP38, 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.

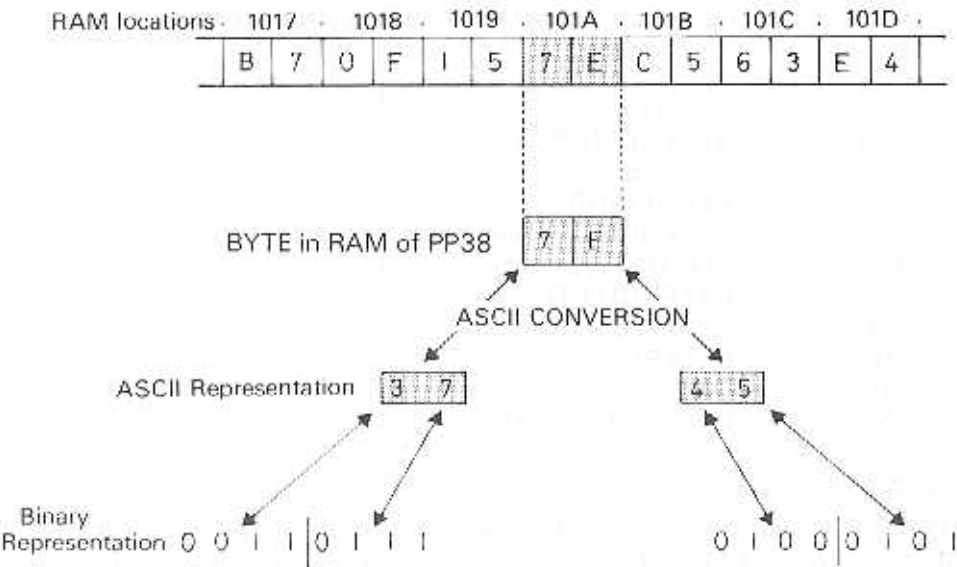
#### Binary Rubout

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 PP38 RAM





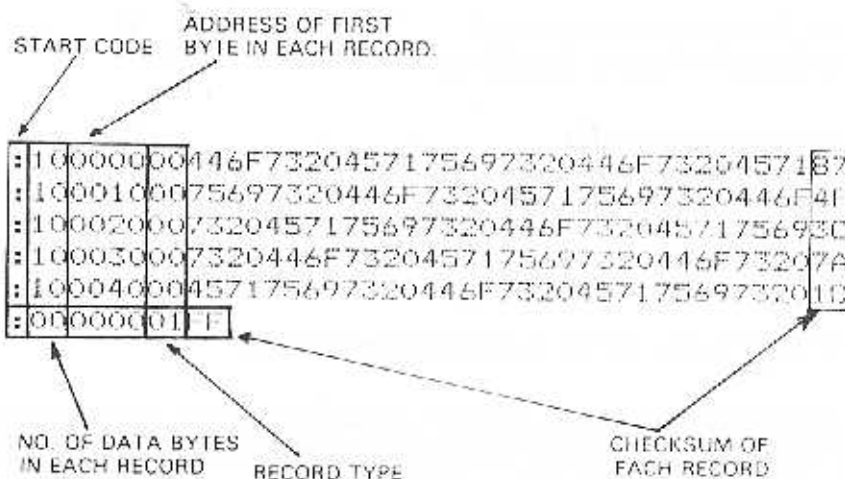
### 7.1.1 Intellec

The Intellec format when displayed consists of:

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

For example:

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



## Calculation of the Intellec\* Checksum

:10000000446F7320457175697320446F73204571B7  
 :01001000757A  
 :00000001FF

Example: The second 'data record' of the above format.

- (i) This is: :01 00 01 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 "2'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.

## 7.1.2 Extended Intellec

The extended Intellec format when displayed consists of:

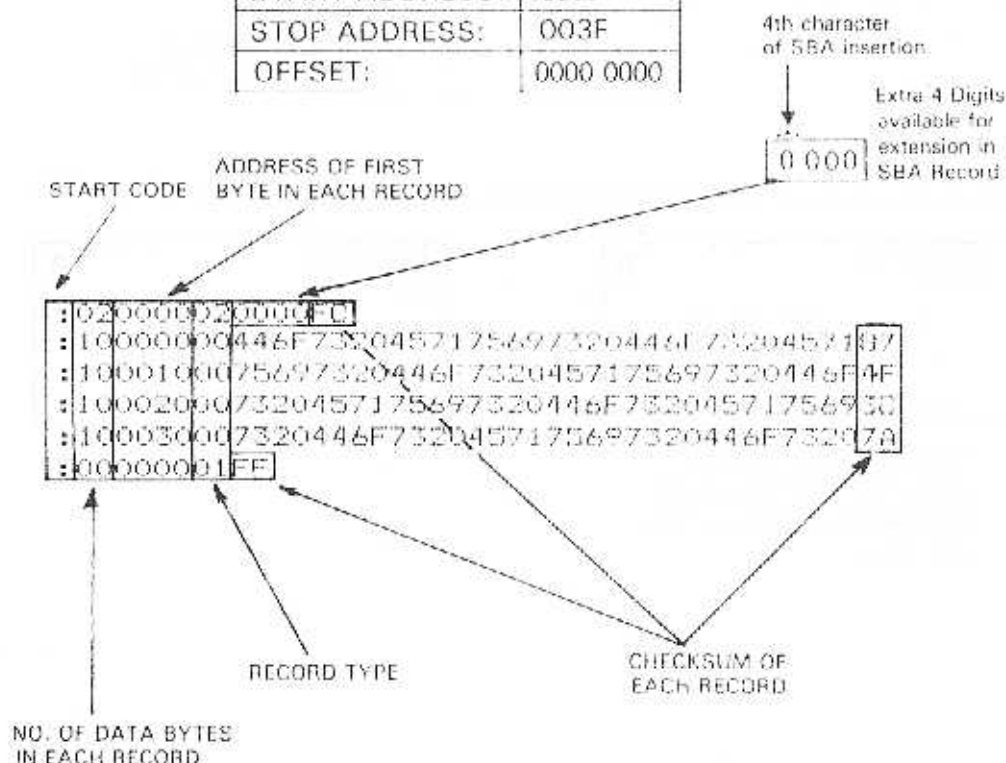
- A start code, i.e. (colon)
- The sum of the number of bytes in a particular record, e.g. 10
- The address of the first byte of data in an individual record, e.g. 0000
- The record types, i.e. 00 - Data Record  
01 - End Record  
02 - 'Segment Base Address' record (SBA)\*

\*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).

- Data (in bytes) e.g. 44 6F 73 20
- 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



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

EXTENSION DIGIT	<b>B</b>
1000 SBA INSERTION	
+ 0000 ADDRESS FIELD	
= 10000	

EXTENSION DIGIT	<b>A</b>
0000 SBA INSERTION	
+ FFBB ADDRESS FIELD	
= 0FFBB	

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

### 7.1.3 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







### 7.1.5 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:

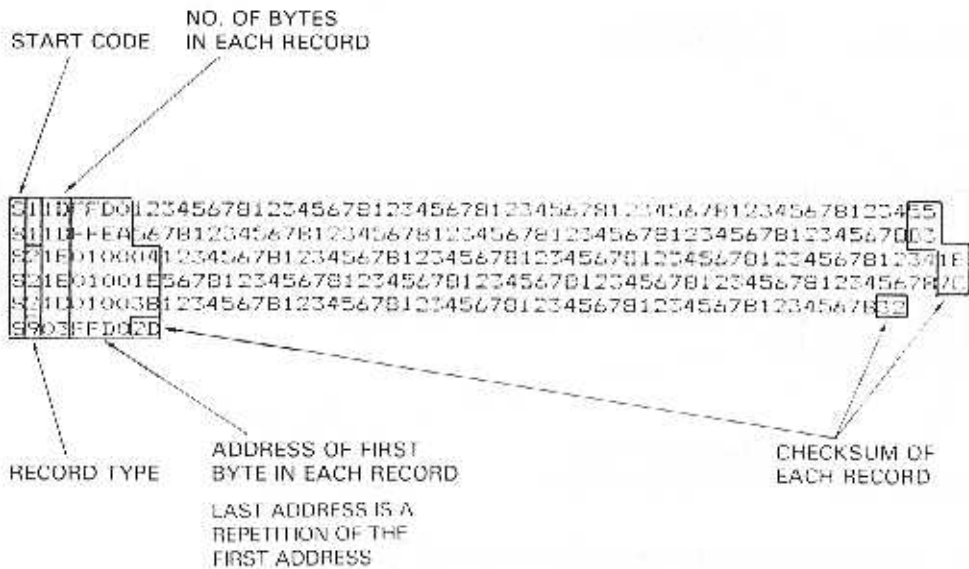
- a. A start code, i.e. S
- b. The record types, i.e. 1—Data Record (Four Character address)  
9—End Record (Four Character address)  
  
2—Data Record (Six character address)  
8—End Record (Six character address)  
  
3—Data record (Eight character address)  
7—End Record (Eight character address)
- 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





**Transition from 2 Byte Address (4 Characters)  
Through to 3 Byte Address (6 Characters).**

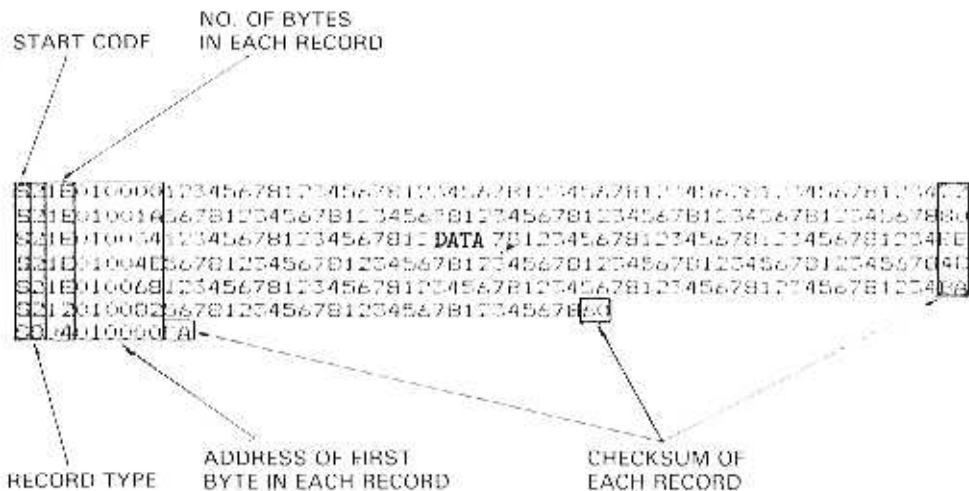
START ADDRESS:	FF80
STOP ADDRESS:	FFFF
OFFSET:	00000050



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

For example:

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

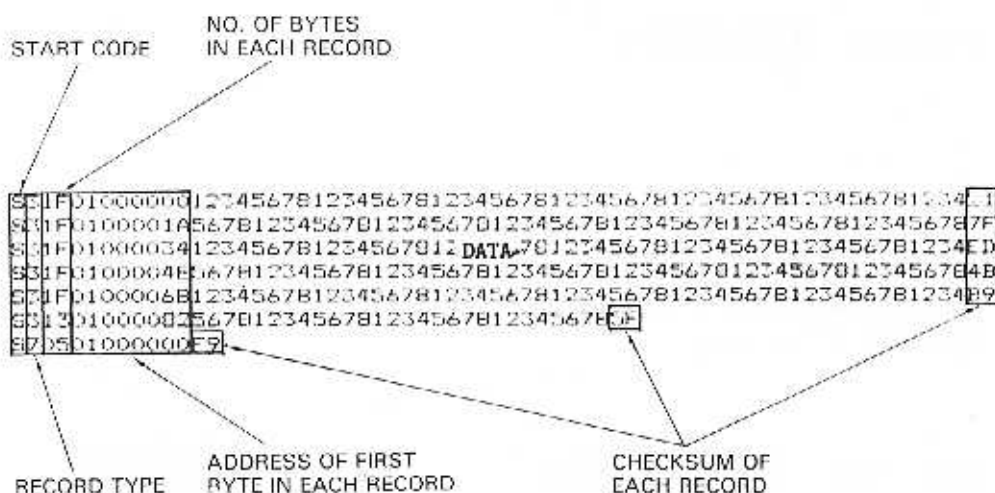


2—Data Record (Six Character Address)	} 3 BYTES
8—End Record (Six Character Address)	

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

For example:

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



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

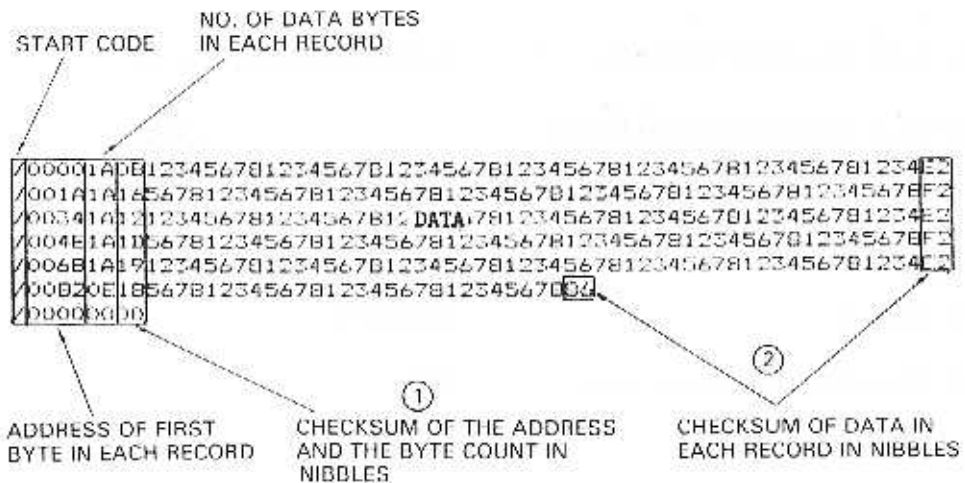
### 7.1.6 Tek Hex.

The Tek Hex format when displayed consists of:

- a. A start code, i.e. /
- b. The address of the first byte of data in an individual record, e.g. 0000
- c. The sum of the number of data bytes in an individual record, e.g. 1A
- d. Checksum 1 which is a nibble addition of the address (4 characters) and the byte count (2 characters), e.g. 0B
- e. Data in bytes, e.g. 12 34 56 78
- f. Checksum 2 which is a nibble addition of all data.
- g. 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 the Tek Hex. Checksums

Unlike the other PP38 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 data alone.

```
700001A0B12345678123456781234567812345678123456781234E0  
CHECKSUM 1 CHECKSUM 2 i678123456781234567812345678123456781234567DF2  
70034030A12345615  
700b000000
```

Example: The third 'data record' of the above format

### Checksum 1

(i) This is: /10034030A

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.

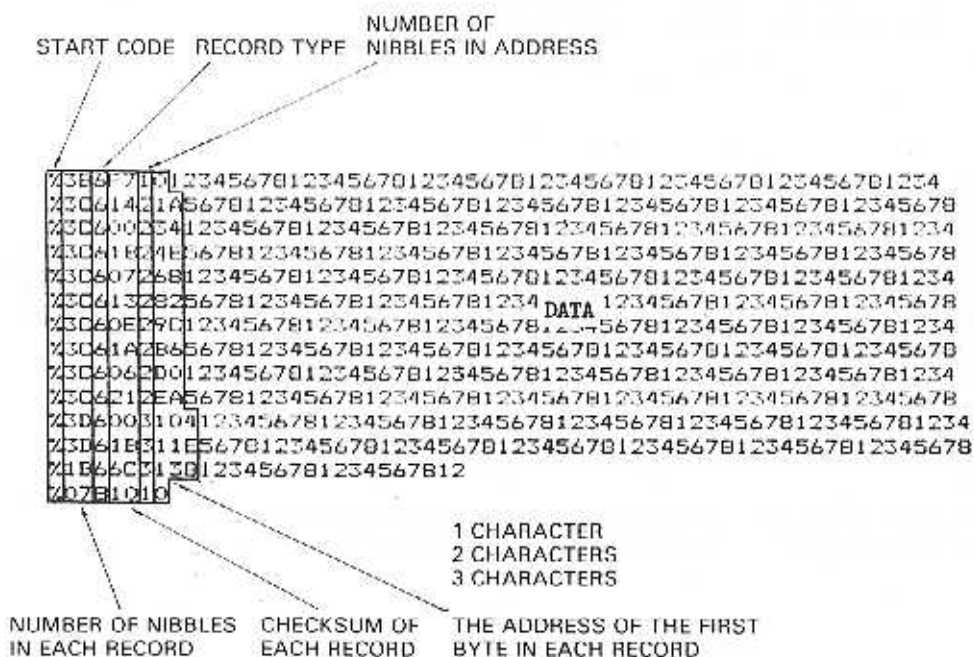
### 7.1.7 Extended Tek Hex.

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

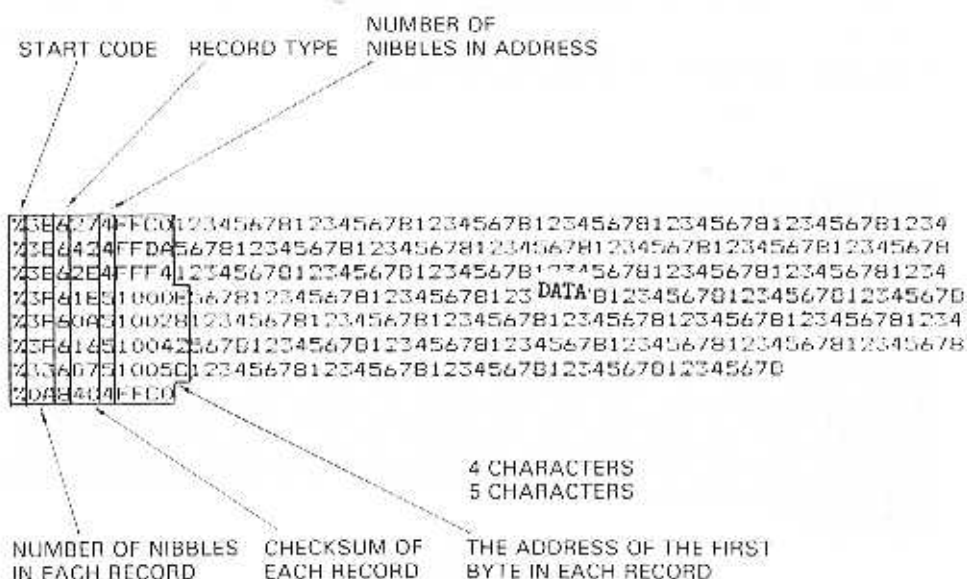


\*Sections (e) and (f) are integrated:

2/1A 6/100000 A/1B4625DC95

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

For example:



### Calculation of the Extended Tek Hex. Checksum

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

73B6F71012345678123456781234567812345678123456781234  
73C61421A56781234567812345678123456781234567812345678  
**73D6A6B6C6D6E6F**  
7078d61010

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 <u>1C</u> 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.



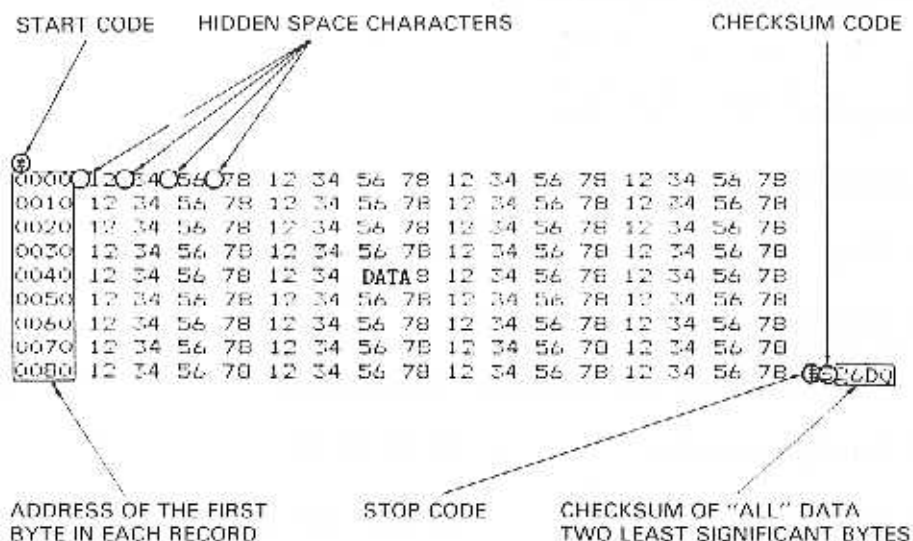
### 7.1.8 PPX (or STAG HEX.\*)

The PPX format when displayed consists of:

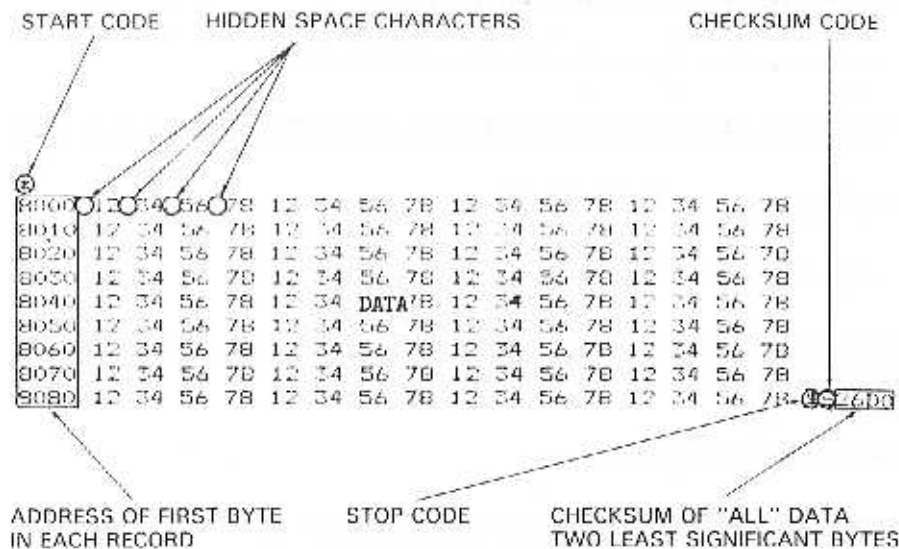
- A start code, i.e. \* (an asterisk, 2A—ASCII Code)
- The address of the first byte of data in an individual record, e.g. 0000
- Data in bytes, e.g. 12 34 56 78
- A stop code, i.e. \$ (a dollar sign, 24—ASCII Code)
- A checksum of all data over the entire address range. (The displayed checksum is the two least significant bytes.)
- A checksum start code: S
- 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

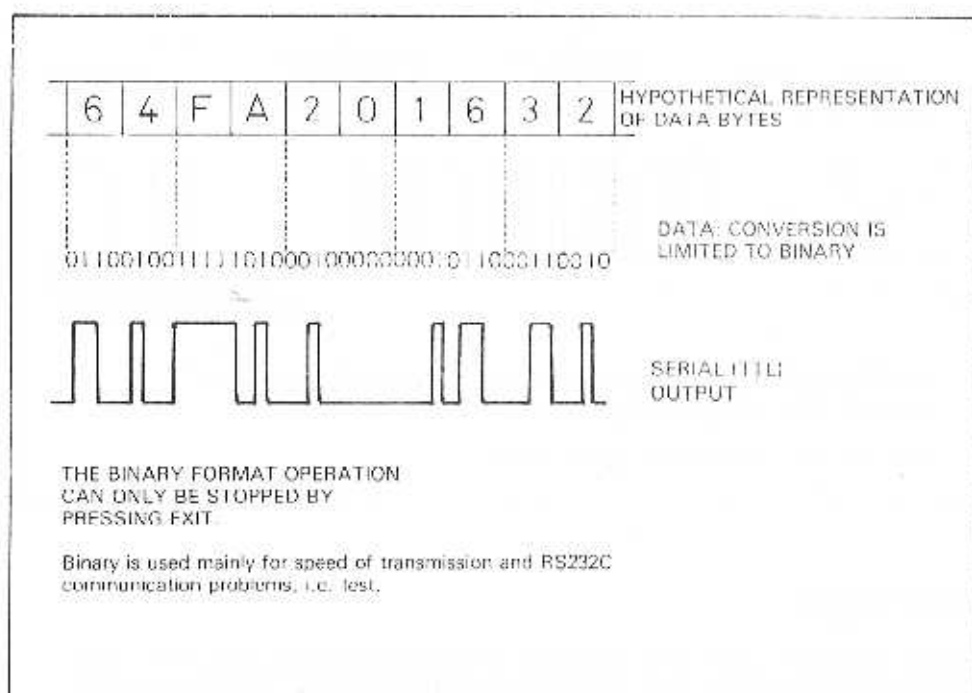


### 7.1.9 Binary, DEC Binary and Binary Rubout

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

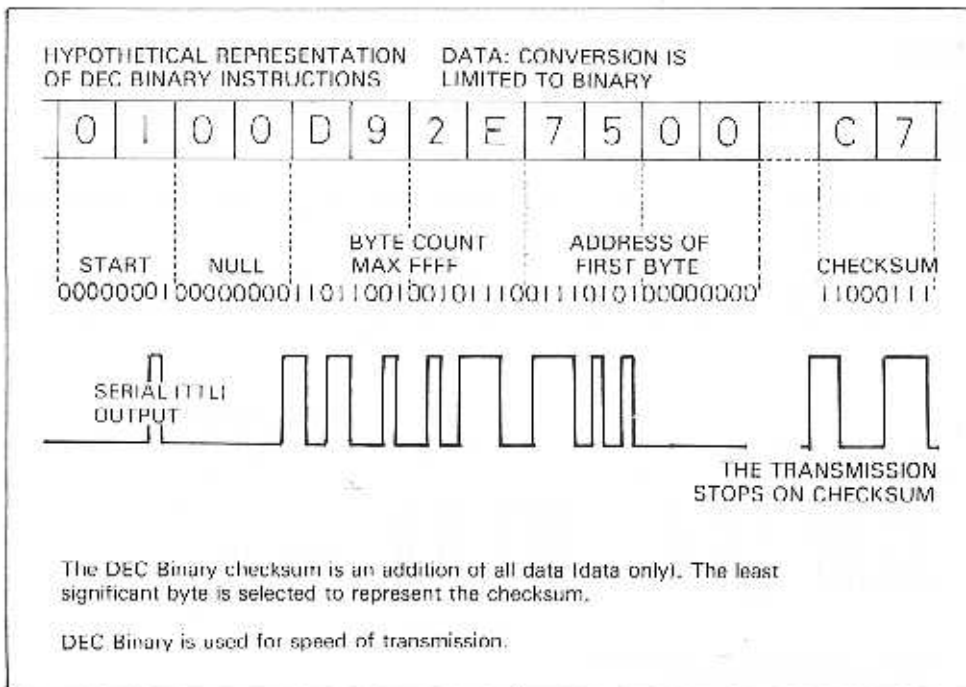
Binary is data only. It is devoid of a start code, address, stop code and checksum.



## 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:

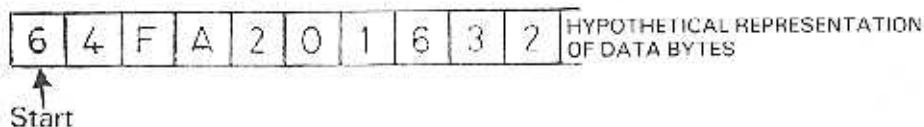


## Binary Rubout

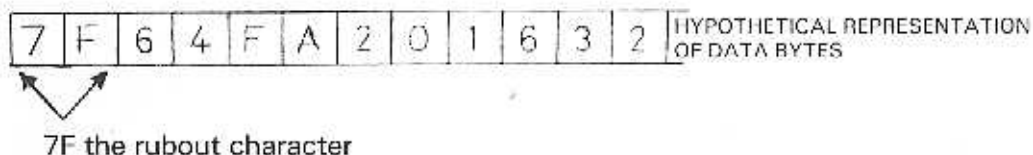
Binary Rubout is similar to Binary in that it is devoid of Address, Stop Code and Checksum. The data is preceded however, by the Rubout character (FF).

For example:

If a string of Binary data is represented thus:



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



## SECTION 8

stag

---

Sophisticated systems for the discerning engineer.



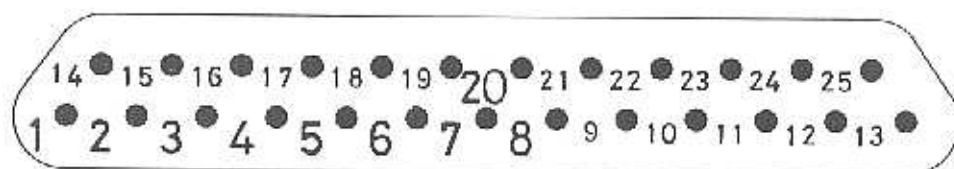
## 8 RS232C Hardware Descriptions

### 8.1 RS232C Interface Port Connections

The PP38 is linked to peripheral equipment via its RS232C interface port.

The RS232C port is a D type female connector which is able to accept a 25 pin male connector. Only 9 of the 25 available pins play an active role in data transfer.

These are: Numbers 1, 2, 3, 4, 5, 6, 7, 8 and 20.



1.*	PG	PROTECTIVE GROUND
2.	TXD	TRANSMITTED DATA
3.	RXD	RECEIVED DATA
4.	RTS	REQUEST TO SEND
5.	CTS	CLEAR TO SEND
6.	DSR	DATA SET READY
7.	SG	SIGNAL GROUND
8.	DCD	DATA CARRIER DETECT
20.	D.T.R.	DATA TERMINAL READY

\*Pin Number 1 is present in all connections, it represents the Protective Ground surrounding all the other cables.

#### Link-up of the PP38 to Peripheral Equipment

There are two distinctive types of machine:

- (i) Data Terminal Equipment (DTE)  
The PP38 itself falls into this category, plus terminals, tape reader etc.
- (ii) Data Communication Equipment (DCE)  
Computers etc.

However, connection types between these two machine categories is not so distinct.

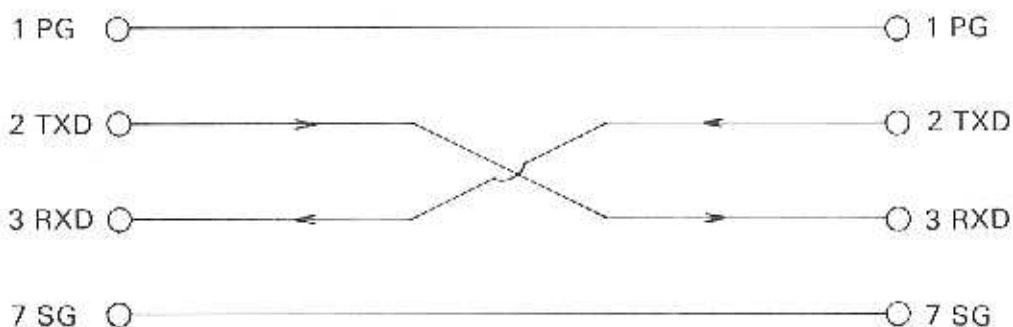
#### Connection Types

The PP38 supports the two most popular types of connection, these are: XON/XOFF (3 wire cable-form connection) and (7/8 wire cable form) hardware handshake. The most straightforward of these two is the XON/XOFF.

## 8.2 XON/XOFF (3 wire cable-form connection)

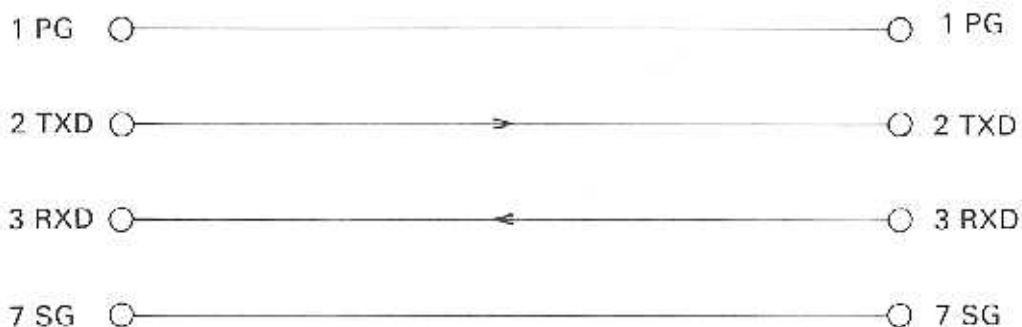
- a. For connection of two alike machines a 'cross over' is required.

DCE to DCE and DTE to DTE:



- b. For connection of two unlike machines 'no' cross over is required.

DCE to DTE and DTE to DCE: Straight

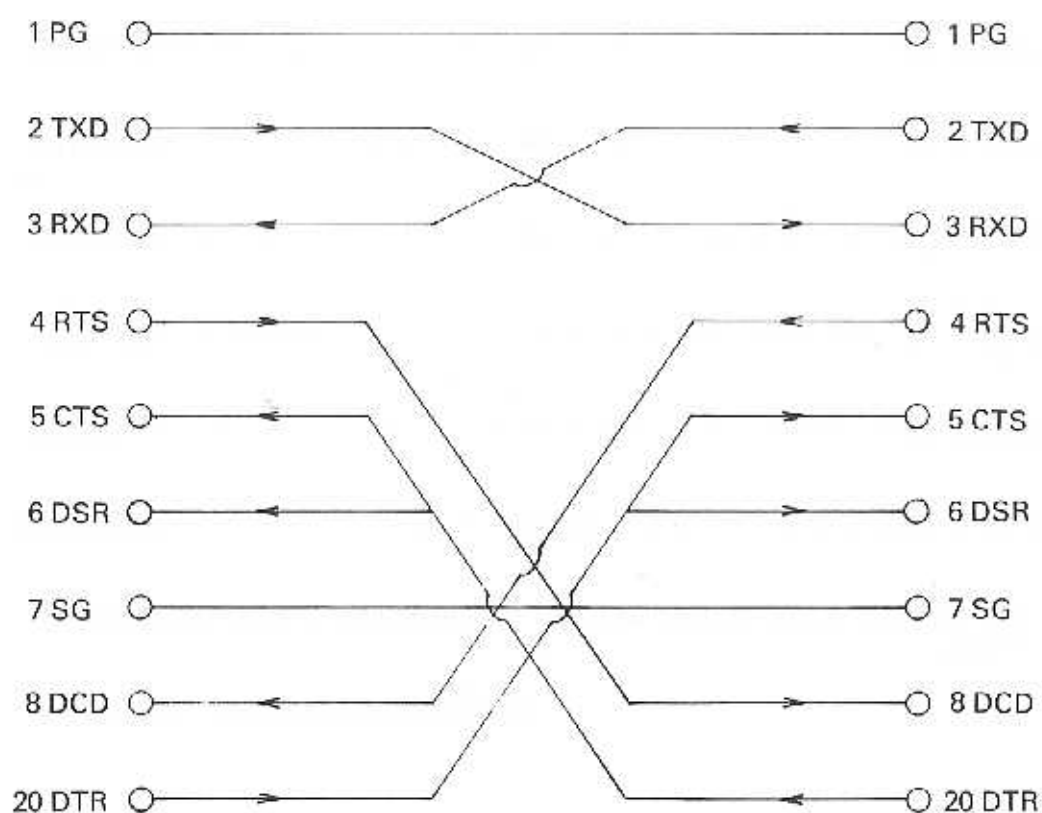


NOTE: Some machines do not have internal pull-ups and require extra connections within the cable form. Pull-ups may be required on pins 5, 6 and 8 of the external device if it is DTE or pins 4, 6 and 20 if it is DCE.

### 8.3 Hardware Handshake (7 or 8 wire cableform)

DCE TO DCE AND DTE TO DTE	CROSSOVER	8 WIRE CABLE-FORM
DCE TO DTE AND DTE TO DCE	STRAIGHT	7 WIRE CABLE-FORM

- a. For connection of two alike machines a 'cross over' is required.  
DCE to DCE and DTE to DTE:

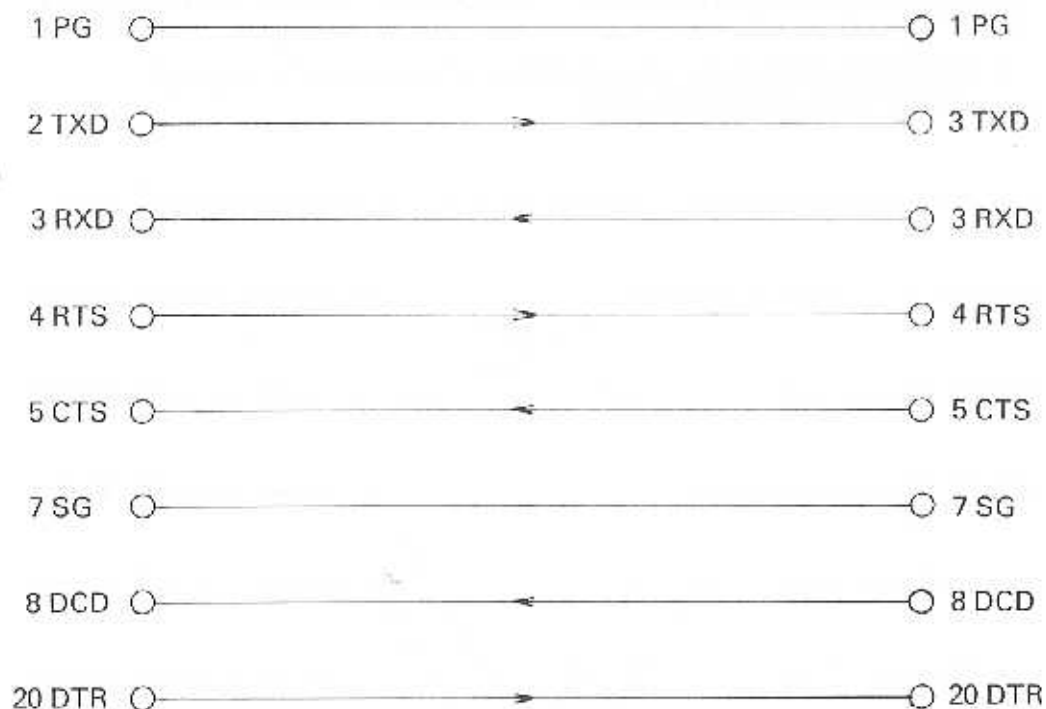


EXISTS AS STAG PART No. 10-0250



b. For connection of two unlike machines 'no' cross over is required.

DCE to DTE and DTE to DCE: Straight



NOTE: Pin 6 on the straight version b. will be pulled-up internally by the PP38.

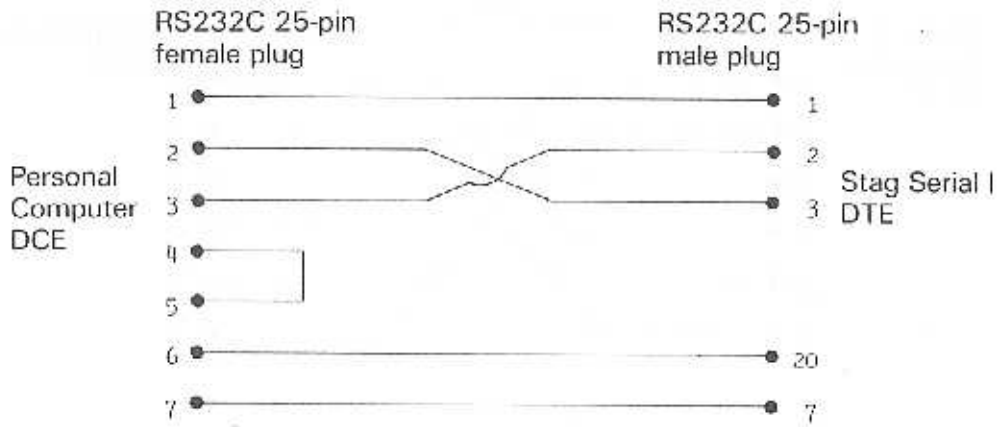
EXISTS AS STAG PART No. 10-0251

## 8.4 Connection to IBM\*PC and AT

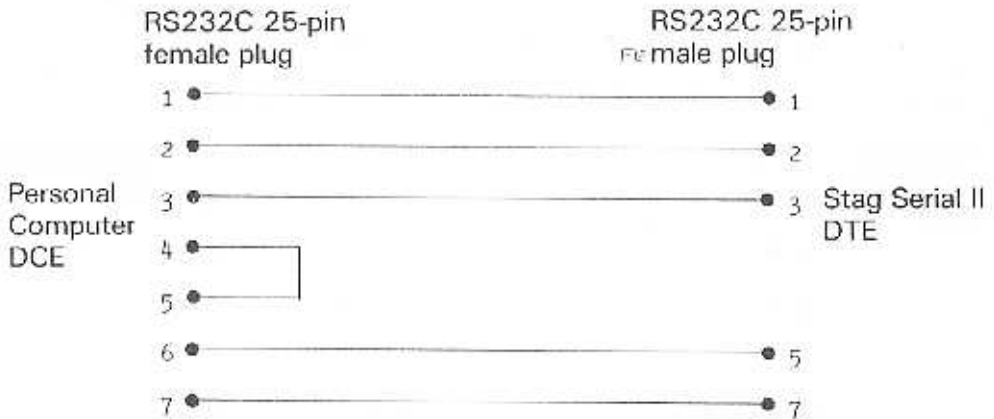
### PC to Programmer RS232C interface connections

#### XON/XOFF

IBM-PC connected to the female (serial I) interface on a Stag Programmer.

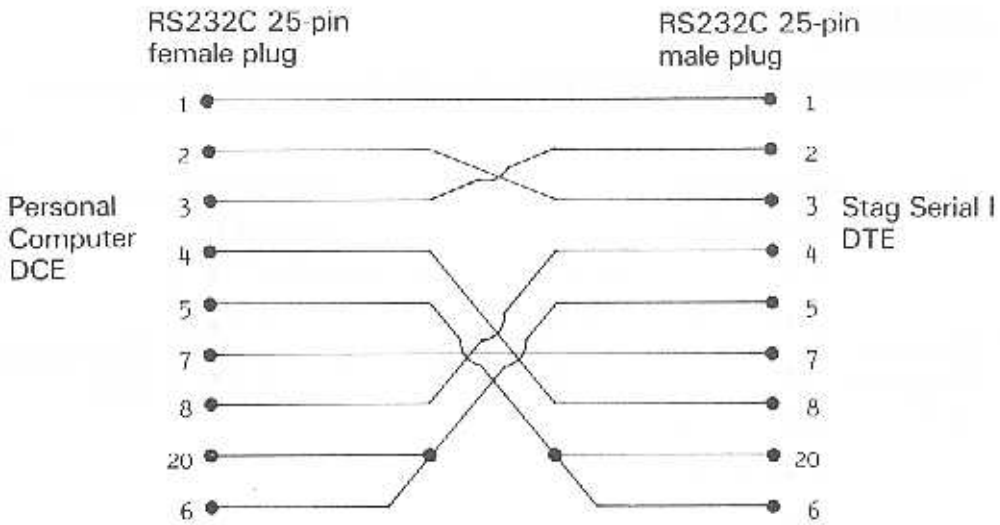


IBM-PC connected to the male (serial II) interface on a Stag programmer.

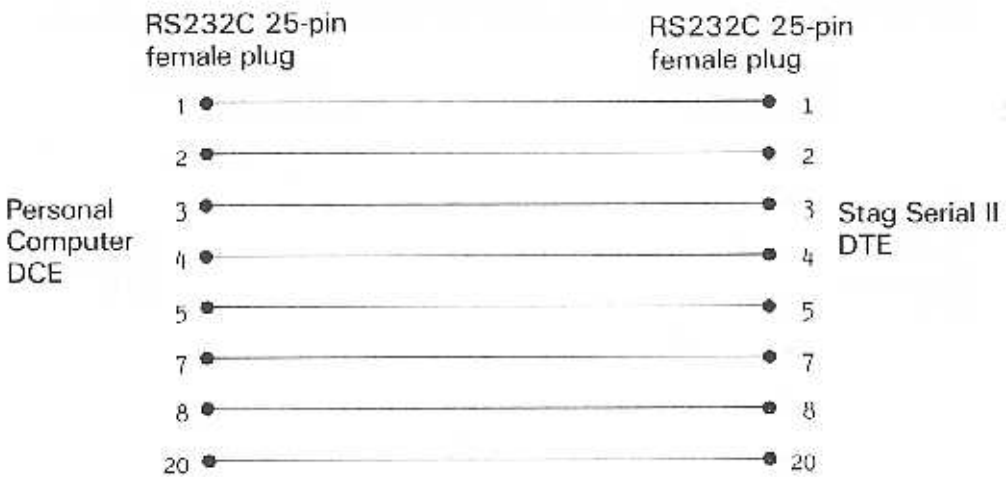


## Hardware Handshake

IBM-PC connected to the female (serial I) interface on a Stag programmer.

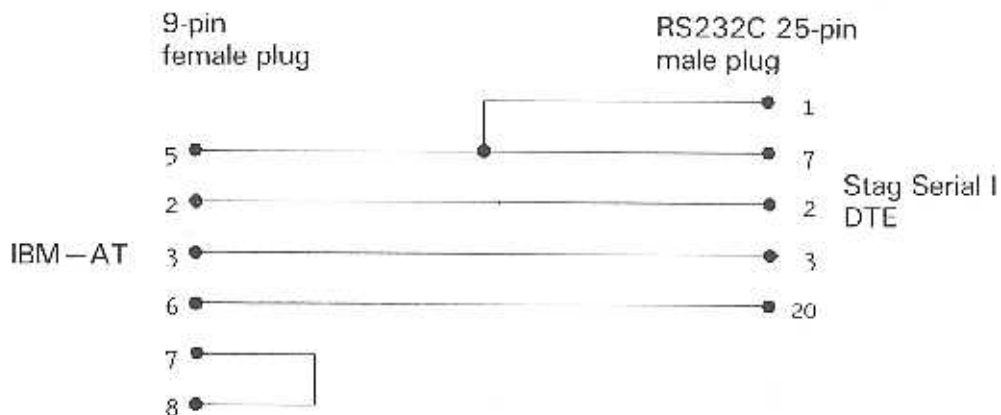


IBM-PC connected to the male (serial II) interface on a Stag programmer

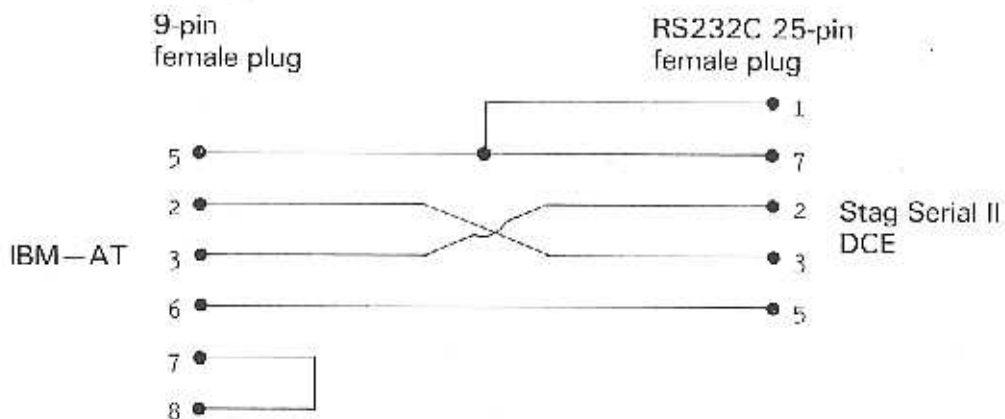


## IBM—AT connections

IBM—AT connected to the female (serial I) interface on a Stag programmer



IBM—AT connected to the male (serial II) interface on a Stag programmer.



## SECTION 9

stag

---

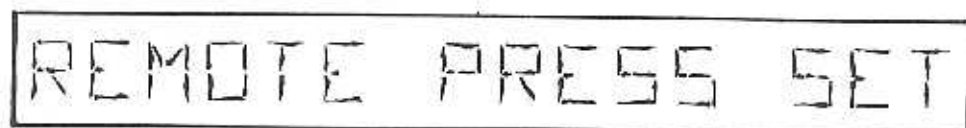
Sophisticated systems for the discerning engineer.



## 9 Remote Control

### 9.1 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.
SEEQ	SS16A	REM

In the remote mode the PP38 operates under remote control from a computer or a terminal. The keyboard of the PP38 is inoperative at this time and the display will only show information as requested under remote control.

## 9.2 Remote Control Commands

h – one hex digit

**RETURN** Executes a command for instance G RETURN, A6AF< RETURN, I 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).

**I** \*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'**

## 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 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 PP38 remote error words)
- X Error-code inquiry. Programmer outputs error codes stored in scratch-RAM and then clears them from memory. (See PP38 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.

### 9.3 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)
	<b>Programming Errors</b>
23	Any device-related error
22	Device appears faulty to the machine's electronics (26)
21	L2 + L3> Device
20	Not used
19	Device not blank (20)
18	Illegal bit (21)
17	Non verify (23)
16	Incomplete programming or invalid device (22)
	<b>I/O Errors</b>
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)
	<b>RAM Errors</b>
7	RAM — hardware error
6	Not used
5	L2 + L3> 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 PP38

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	Overflow Error	
43	Framing and Overflow 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 10

stag

---

Sophisticated systems for the discerning engineer.



## 10 Worldwide Sales and Service Support

Stag provides full Sales and Service support in the following countries:

Australia	Netherlands
Austria	New Zealand
Belgium	Northern Ireland/Eire
Canada	Norway
Denmark	Portugal
Finland	Singapore
France	Spain
Greece	South Africa
Hong Kong	Sweden
India	Switzerland
Israel	Taiwan
Italy	United Kingdom
Japan	United States
Korea	West Germany

Should you require information regarding your nearest Stag representative, do not hesitate to contact us at one of the following addresses:

### United Kingdom

Stag Electronic  
Designs Ltd.  
Tewin Court  
Welwyn Garden City  
Herts. AL7 1AU  
United Kingdom  
Tel: (0707) 332148  
Tlx: 8953451  
Fax: (0707) 371503

### U.S.A. (West Coast)

Stag Microsystems Inc.  
1600 Wyatt Drive  
Santa Clara, CA 95054  
U.S.A.  
Tel: (408) 988-1118  
Fax: (408) 988-1232

### U.S.A. (East Coast)

Stag Microsystems Inc.  
3 Northern Blvd,  
Amherst, NH 03031  
U.S.A.  
Tel: (603) 673-4380  
Fax: (603) 673-1915