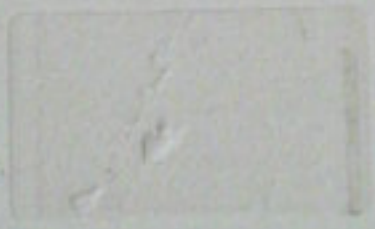
PP40 Series Programmers



STAG PROGRAMMERS

Operating Manual

for

PP40, PP41 and PP42

Gang MOS Programmers

Staples Corner London

Stag Programmers Ltd
Silver Court, Watchmead
Welwyn Garden City
Hertfordshire
AL7 1LT
United Kingdom

Tel: (0707) 332148
Fax: (0707) 371503

01707 863400

Stag Microsystems Inc
1600 Wyatt Drive
Santa Clara
CA 95054

USA

Tel: (408) 988-1118
Fax: (408) 988 1232

803 1008 rev 3

CONTENTS

PP40, PP41 and PP42

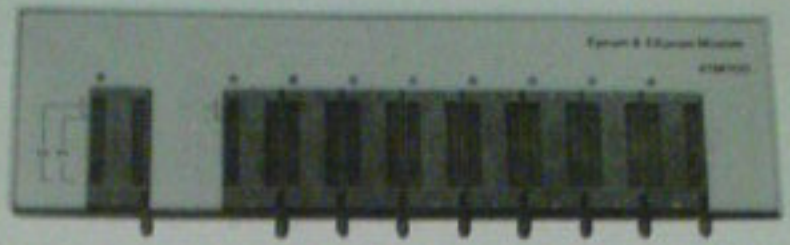
PP41 and PP42 only

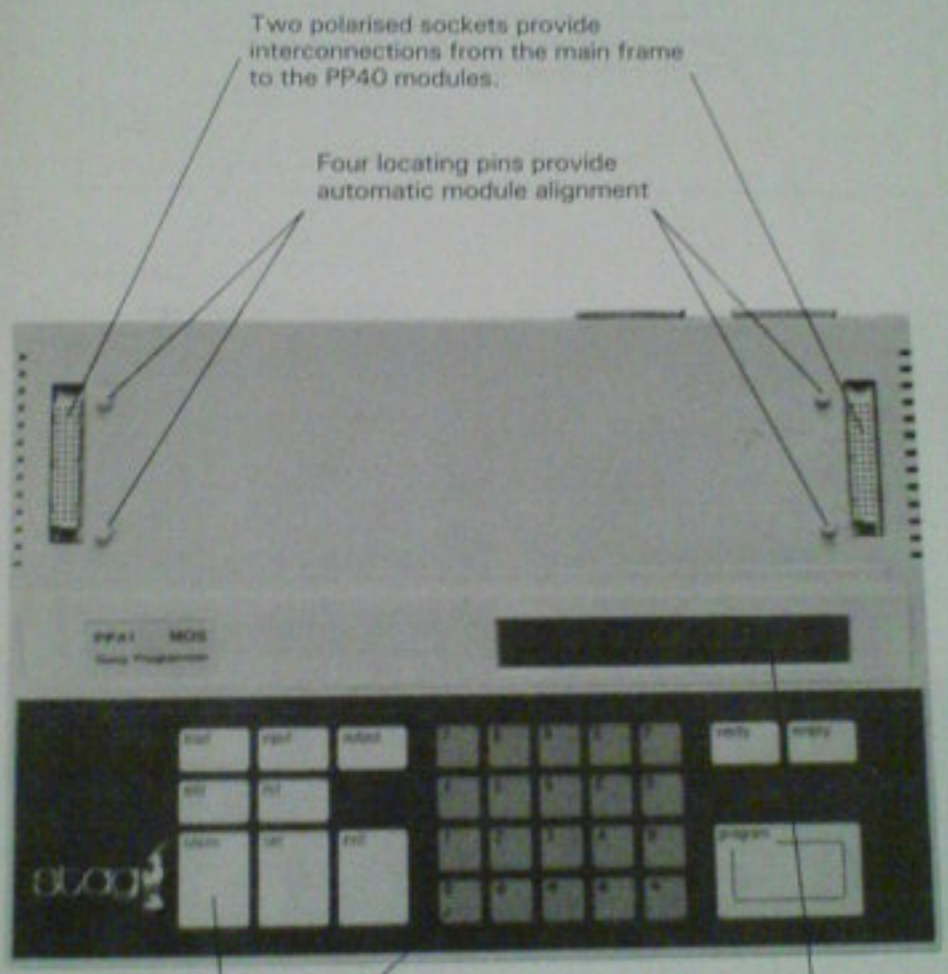
1.1 INTRODUCTION

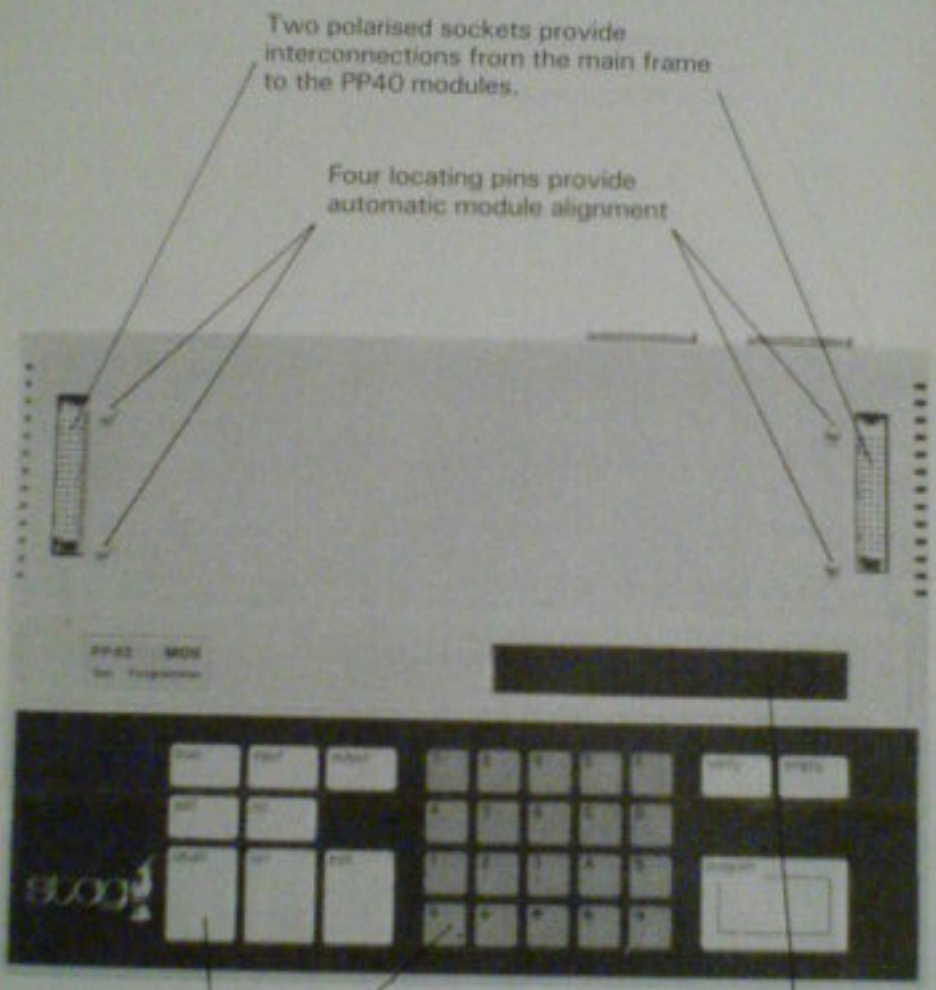
The PP40, PP41 and PP42 are high speed gang programmers, capable of supporting EPROMs and EEPROMs in CMOS and NMOS technology.

The programmers are software controlled using a single level module approach and have no need of adaptor modules. This ensures flexibility and ease of upgrade for future devices. The modules support NMOS and CMOS EPROM and EEPROM devices in both 24 and 28 pin DIL packages. They feature algorithms for fast programming and support Electronic Identifier technology for automatic device identification. The sockets all have bi-coloured LEDs to enhance socket status indication and error reporting. An auto-recall feature is incorporated whereby pre-set parameters are recalled from a non-volatile memory on power-up. These are the device and the I/O parameters.

PP41 MAINFRAME



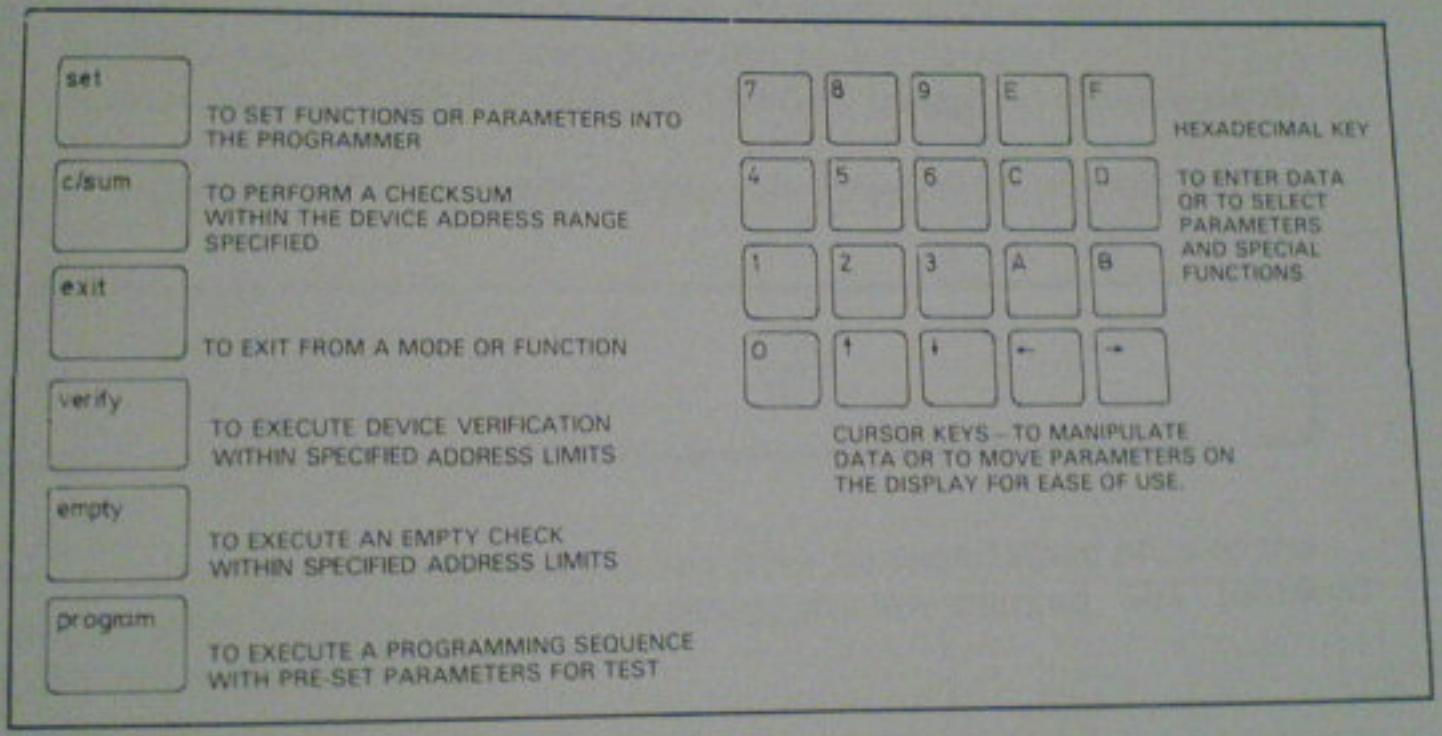



Two polarised sockets provide interconnections from the main frame to the PP40 modules.
Four locating pins provide automatic module alignment
Keyboard: For data entry and operating programmer functions
16-Character green alphanumeric display

1.3 The Keyboard

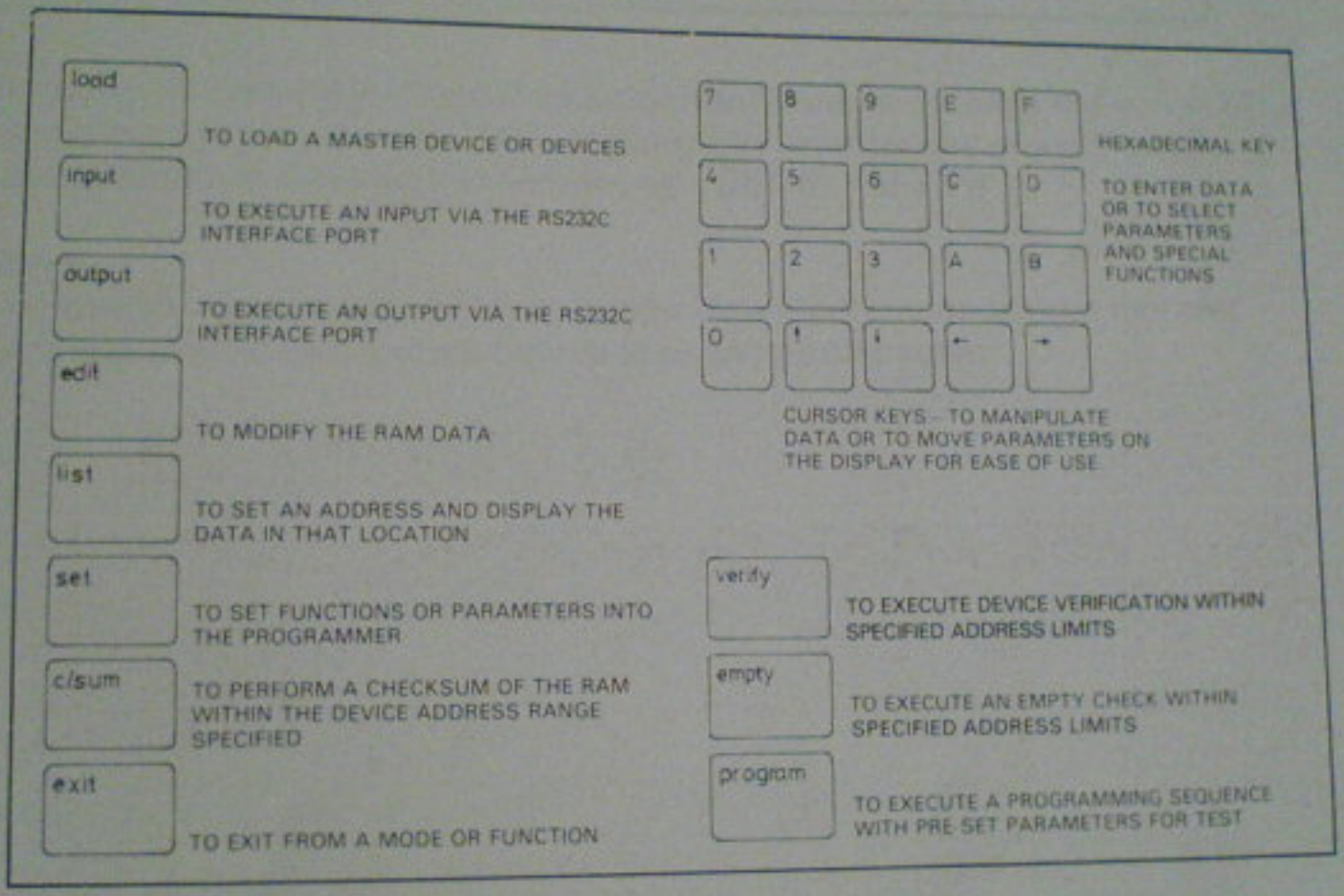
For data entry and operating programmer functions.

PP40



PP41/PP42

PP41/PP42

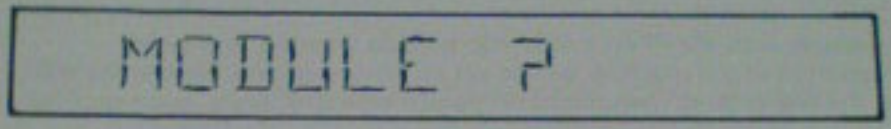

load
TO LOAD A MASTER DEVICE OR DEVICES
input
TO EXECUTE AN INPUT VIA THE RS232C INTERFACE PORT
output
TO EXECUTE AN OUTPUT VIA THE RS232C INTERFACE PORT
edit
TO MODIFY THE RAM DATA
list
TO SET AN ADDRESS AND DISPLAY THE DATA IN THAT LOCATION
set
TO SET FUNCTIONS OR PARAMETERS INTO THE PROGRAMMER
c/sum
TO PERFORM A CHECKSUM OF THE RAM WITHIN THE DEVICE ADDRESS RANGE SPECIFIED
exit
TO EXIT FROM A MODE OR FUNCTION
7 8 9 E F
4 5 6 C D
1 2 3 A B
0
HEXADECIMAL KEY
TO ENTER DATA OR TO SELECT PARAMETERS AND SPECIAL FUNCTIONS
CURSOR KEYS – TO MANIPULATE DATA OR TO MOVE PARAMETERS ON THE DISPLAY FOR EASE OF USE
verify
TO EXECUTE DEVICE VERIFICATION WITHIN SPECIFIED ADDRESS LIMITS
empty
TO EXECUTE AN EMPTY CHECK WITHIN SPECIFIED ADDRESS LIMITS
program
TO EXECUTE A PROGRAMMING SEQUENCE WITH PRE SET PARAMETERS FOR TEST

1.4 INITIAL SETTING UP PROCEDURE

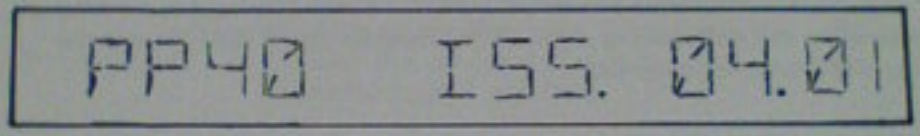
Before attempting to apply power to your programmer ensure that it is set to the correct operating voltage for your power source. The voltage setting will be printed on the rear panel.

1. Plug the supplied power cable into the back panel socket.
2. Apply power to the machine from the power source.
3. Switch on the machine using the ON/OFF switch on the rear panel.

After "POWER ON" and without a Module inserted the display will read:



The main frame 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', eg:



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, eg. 'SET 6', 'SET F6'.

Note
To ensure correct initialization, power down before inserting a module. Always wait two seconds before applying power again.

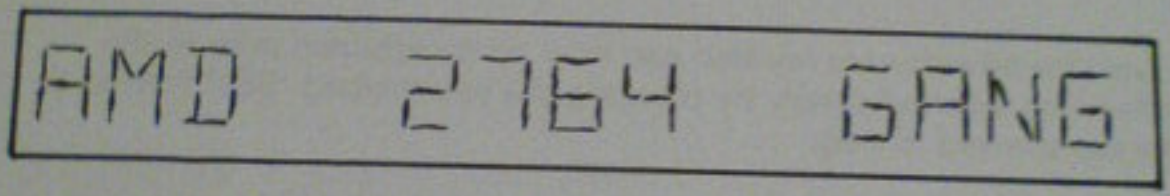
INTRODUCTION OF A MODULE TO THE MAIN FRAME

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

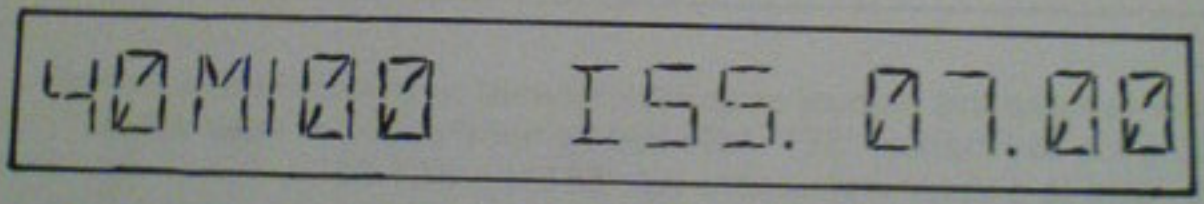
On power-up the programmer will be automatically configured to what it was before the machine was last switched off.

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

For example with the PP40's 40M100 module inserted, the initial configuration of the machine will be set on power-up and the display will show the last entered manufacturer, device type and mode, such as:



To determine the software revision of the module press 'SET 6' and the display will show, for example:

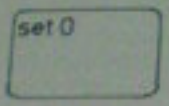


To remove press 'EXIT'.

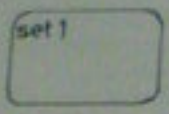
1.5 PP40 LIST OF 'SET' COMMANDS

set 0	Allows user to scan and select various manufacturers and device types.
set6	Displays module software revision if module is plugged in, or main frame software revision, if no module is plugged in.
set8	Calculates and displays CRC (Cyclic Redundancy Check)
setEI	Allows the user to enter the two-key operation of the Electronic Identifier mode. The device signature is read and the manufacturer and device type are displayed. To execute the function the specified device function key must be pressed again, eg. Prog.
setE2	Allows the user to enter the single-key operation of the Electronic Identifier mode. The device signature is read and the PP40 continues straight on to execute the selected function.
setF1	Audible Alarm: To indicate end of program, test, or as a warning using a combination of bleeps and tones. SET F1 can either enable or disable this function.
setF6	Defines device address range.

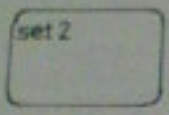
PP41/PP42 LIST OF SET COMMANDS



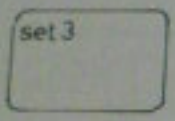
Allows user to scan and select various manufacturers and device types.



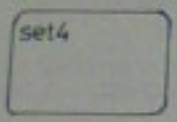
Selects interface parameters: Baud Rate, Word Length, Stop Bits, Parity, Format, Control Z and Pass-Through.



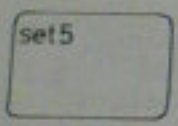
Sets programmer into 'Remote Control'. (To return to Local Mode: Power up with exit key depressed).



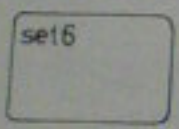
Selects the Bit Mode on the PP42.



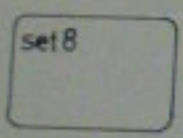
Displays RAM size in hexadecimal.



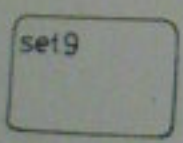
Data complemented throughout entire RAM.



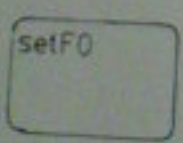
Displays module software revision if module is plugged in, or main frame software revision, if no module is plugged in.



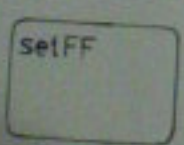
Calculates and displays CRC (Cyclic Redundancy Check).



String Search: The RAM is searched for a specified string of data.



Fills entire RAM with 00.



Fills entire RAM with FF.

PP41/PP42 LIST OF 'SET' COMMANDS – Continued

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 across an arbitrary address range.
setF4	Re-locate RAM data. A block of data with pre-selected address limits can be copied and then re-located at another address within the RAM.
setF6	Defines RAM and device address ranges for all functions which operate on the device.
set input	Enters input address offset, start and stop addresses and port selection.
set output	Enters output address offset, start address and stop address and port selection.

2.1 DEVICE TYPE SELECTION

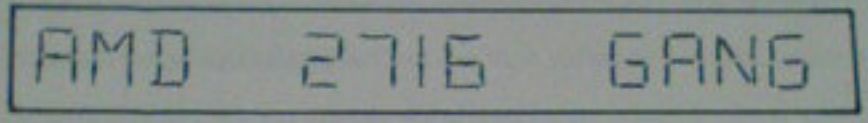
Selecting the device using a 4 digit code

The complete range of devices supported by the programmer's module is stored in the EPROM. Each individual device has its own four digit code. (See device code list).

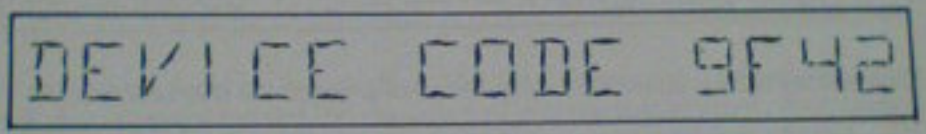
SET 0—Allows code selection

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

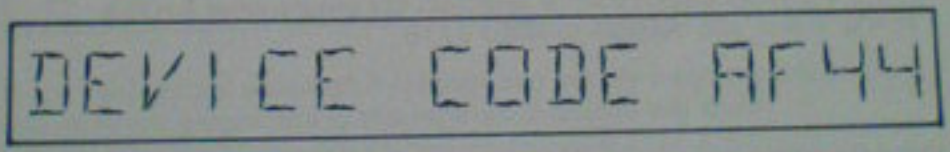
For example:



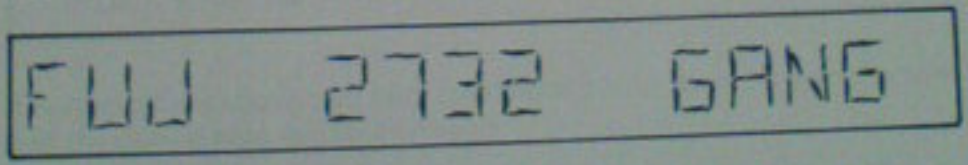
By pressing SET '0' the device code of this configuration will be displayed:



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

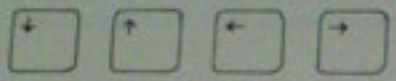


The selection sequence can be completed by pressing EXIT whereupon the new manufacturer and device type are displayed.

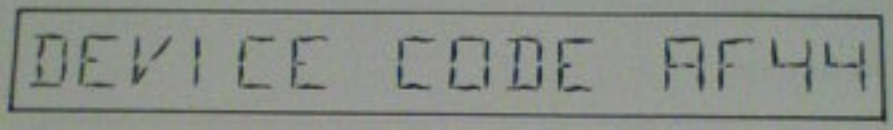


Scanning device types and manufacturers by use of cursor keys

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:

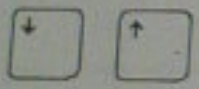


By pressing SET '0' the code of the last used device is displayed:

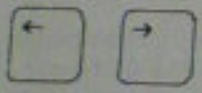


The manufacturer and device type can be changed by use of the cursor keys:

The up/down keys scan the range of manufacturers.

 changes manufacturer

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

 changes device type

Note: If an invalid device code is selected, the programmer emits a warning 'bleep' and defaults to AMD 2716.

2.2 ELECTRONIC IDENTIFIER

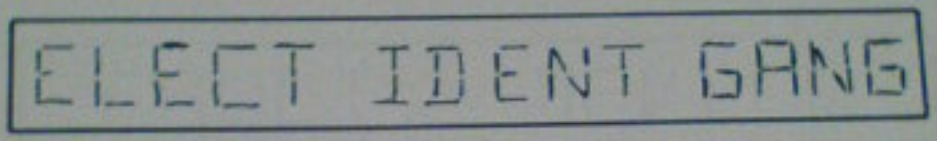
Important Note:
Devices which do not contain an Electronic Identifier can be irreparably damaged if they are used in the Electronic Identifier mode.

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 programmers directly to identify any devices containing an Electronic Identifier and thus eliminates the need for the user to select the device type.

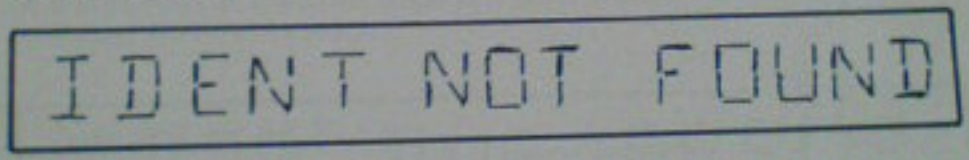
The programmers presently uses two modes of Electronic Identifier operation both of which only work with 28 pin devices.

Mode (i): Two Key Operation

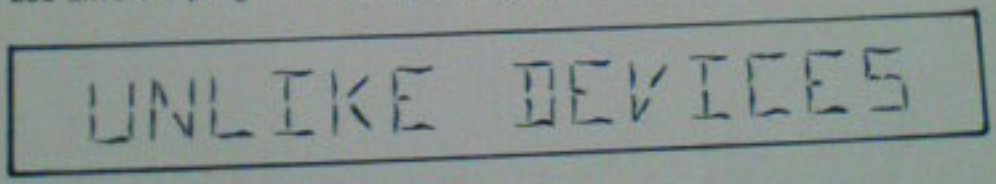
On pressing SET E1 the display will show:



If any device function key such as 'Program' is pressed, the programmer 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 programmer's list of valid codes the display will show:



If any devices are successfully recognised but are incompatable ie. they use different programming algorithms the display will show:



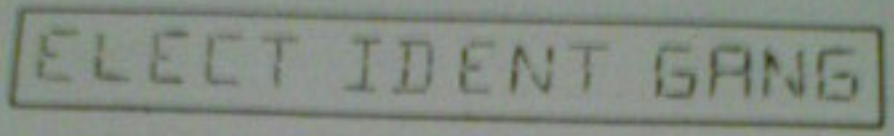
If neither of the above two fault conditions occurs then the manufacturer and the device type will be displayed.

To execute the function the specified 'device function key' must be pressed again eg. Prog

To exit from the Electronic Identifier mode select a device using SET 0 in the usual manner.

Mode 00: Single Key Operation

Pressing SET E2 will again display "ELECT IDENT".



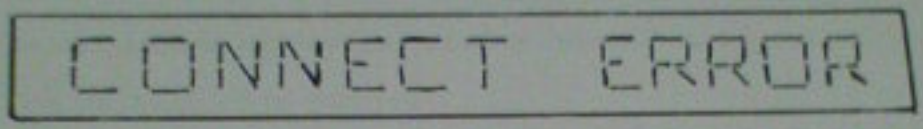
Operation is similar to the previously described mode except that rather than stopping to display the manufacture and device type the programmer continues straight on to execute the selected function.

To exit from the Electronic Identifier mode select a device using SET 0 in the usual manner.

3.1 ERROR DETECTION

Connect Errors

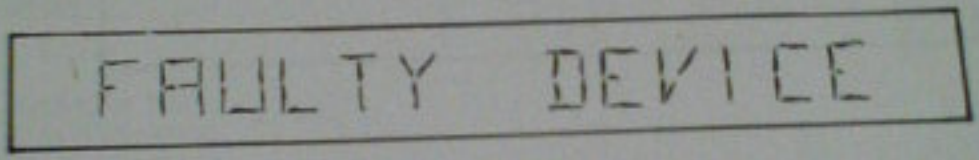
The programmers have the ability to detect connect errors but the selected operation will not be interrupted unless the master socket or all the slave sockets have connect errors. In such a case the display will show:



Red LEDs indicate the failing socket(s).

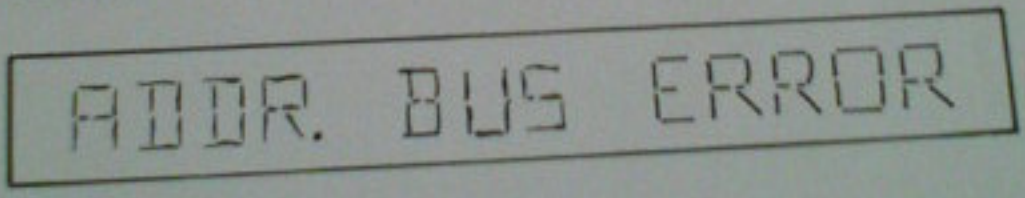
Reversed or Faulty Device

If just one reversed or faulty device is detected, the selected operation stops immediately with the message:



Device Address Bus Check

If a fault such as a shorted address line is detected the selected operation will stop with the message:



3.2 PP41 and PP42 only

Note: For a full explanation of Bit mode configuration on the PP42 the user is advised to turn to section 10.

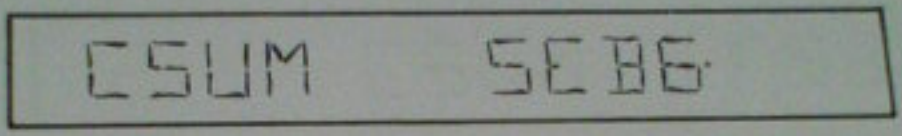
LOAD

Loading the RAM from a 'master' PROM

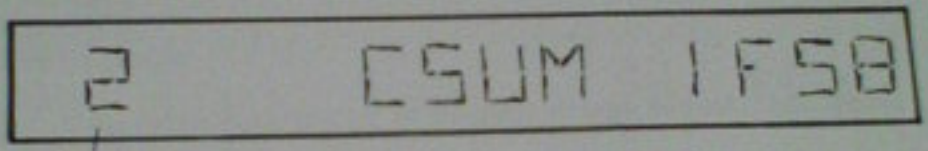
Insert the master devices into the master sockets. Press the load key.

The checksum will be displayed:

PP41



PP42



a number from 1 to 8 denoting section of RAM allocated to device

Press ← or → to obtain a checksum of the next master socket (if applicable).

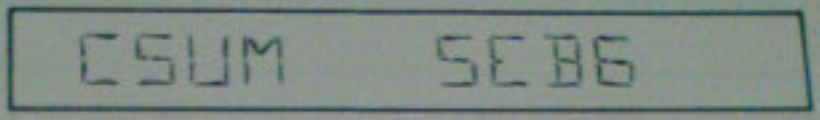
3.3 CHECKSUM

Press C/sum

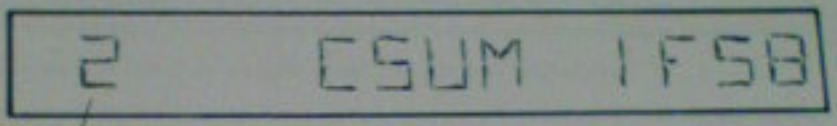
to perform a checksum of the master device (PP40) or RAM (PP41, PP42)

The display will show:

PP40/ PP41



PP42



a number from 1 to 8 denoting section of RAM allocated to device

Press [→] to obtain a checksum of the next master socket if applicable.

CYCLIC REDUNDANCY CHECK (CRC)

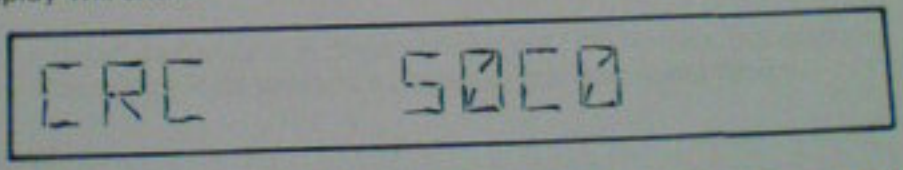
The Cyclic Redundancy check applies a continuous process of shifting and addition to the PROM data. This yields a coded representation of the data which is sensitive to the ordering of the data bytes, unlike the checksum which only considers their values.

Press SET 8

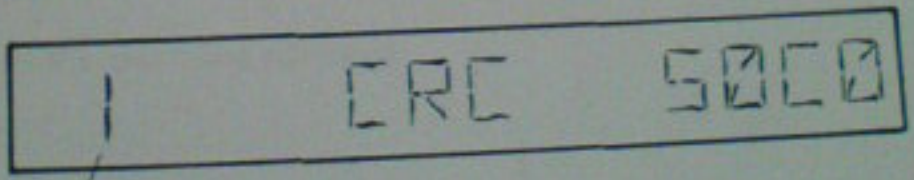
to perform a Cyclic Redundancy check on the master device (PP40) or RAM (PP41, PP42)

The display will show:

PP40/ PP41



PP42



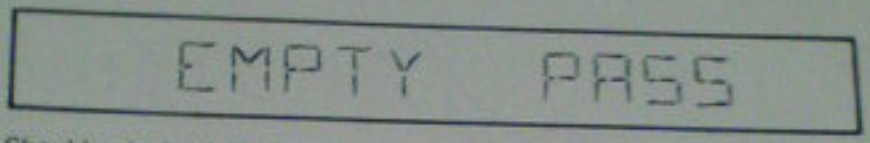
a number from 1 to 8 denoting section of RAM allocated to device

Press [→] to obtain a Cyclic Redundancy check of the next master socket if

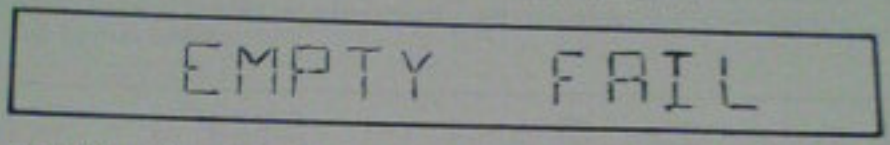
3.4 PROGRAMMING SEQUENCE

Empty Test

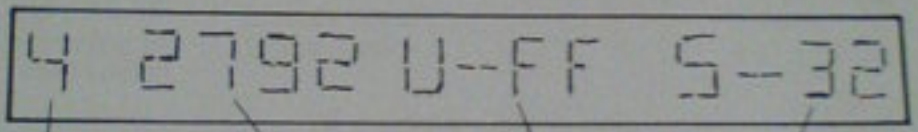
If required an 'empty test' can be applied to the device or devices in the slave sockets prior to programming. This can be done by pressing the 'empty' key. The device or devices will be examined for the unprogrammed state and if they are entirely empty the display will show thus:



Should a device fail the 'empty test' the display will show:



Red LEDs indicate the failed device(s). The cursor keys can be used to move to each in turn, and the display will change to give information of the format below:



a number from 1 to 8 indicating the failed socket

4-digit device address at which failure occurs

unprogrammed state (expected data)

Data actually found

s = slave device

While the failed device data is displayed the red LED flashes, but stops flashing when -> or <- is pressed to move to the next failed device.

Press EXIT

to skip to the next failing addresses.

Press EXIT

to return to Select Device mode.

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

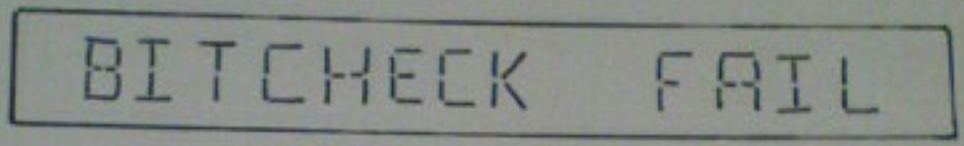
Pressing the program key will automatically execute the 'program' sequence to the manufacturers' specifications with pre-program (Bit Test) and in-program (Verify) device tests.

3.5 PRE-PROGRAM BIT TEST

The programmers automatically check that the pattern already within the slave device is able to be programmed with the intended data from the master device (PP40), or RAM (PP41, PP42).

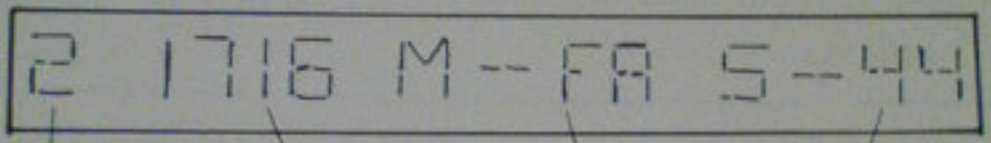
If a device passes the bit test, programming begins automatically.

Should a device fail the bit test, the display will show:



Red LEDs indicate the failed device(s). The cursor keys can be used to move to each in turn, and the display will change to give information of the format below:

PP40



a number from 1 to 8 indicating the failed socket

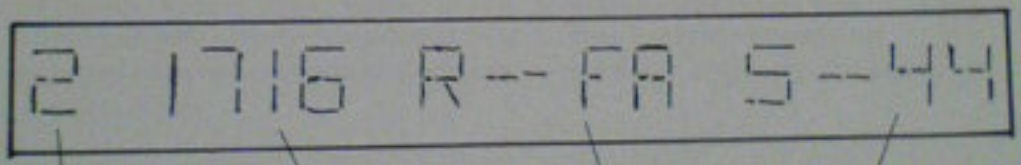
4-digit device address at which failure occurs

Expected Data

Data actually found

m = master s = slave

PP41/ PP42



a number from 1 to 8 indicating the failed socket

4-digit device address at which a failure occurs

Expected Data

Data actually found

R = RAM S = slave

When the failed device data is displayed the red LED flashes, but stops flashing when -> or <- is pressed to move to the next failed device.

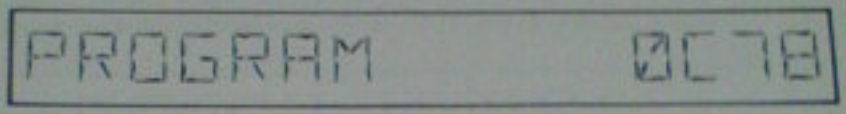
Press set
to skip to the next failing addresses.

3.6 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 displayed; for example:

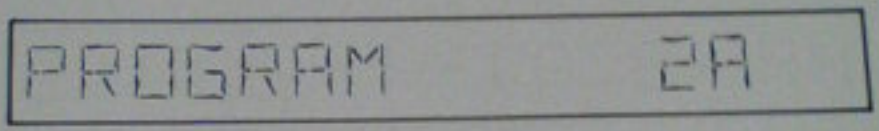
COUNTER



FOUR DIGIT ADDRESS

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

COUNTER



TWO MOST SIGNIFICANT DIGITS OF THE ADDRESS

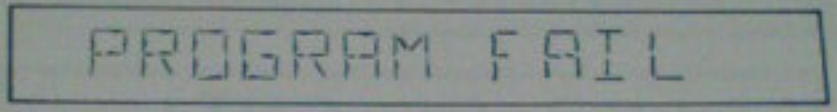
If the data to be programmed into a particular location is the same as the unblown state of that device, the programming sequence will automatically skip to the next location. This function speeds up programming considerably where large sections of the device are to remain empty.

3.7 IN-PROGRAM VERIFY

The algorithms of certain devices are such that an in-program verify is performed. This is a feature whereby each location is checked to see that its data is identical to the corresponding data in the master device (PP40) or the RAM (PP41, PP42).

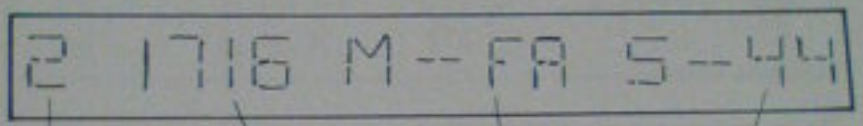
If a device passes the in-program verify at all locations an automatic verify is performed.

Should a device fail the in-program verify, the display will show:



Red LEDs indicate the failed device(s). The cursor keys can be used to move to each in turn, and the display will change to give information of the format below:

PP40



a number from 1 to 8

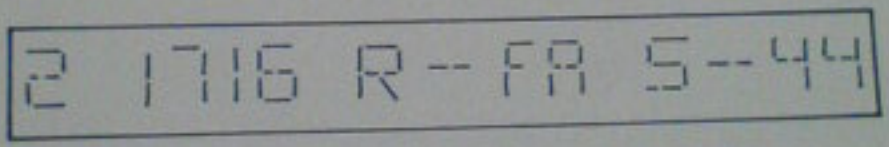
4-digit device address at which failure occurs

Expected data

Data actually found

m = master s = slave

PP41/ PP42



R = RAM s = slave

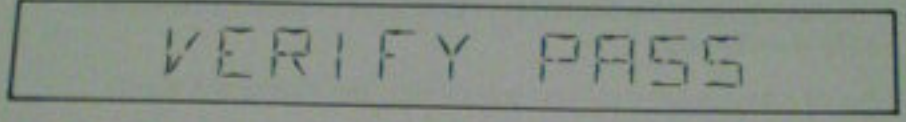
While the failed device data is displayed the red LED flashes but stops flashing when -> or <- is pressed to move to the next failed device.

Press Set
to skip to the next failing addresses.

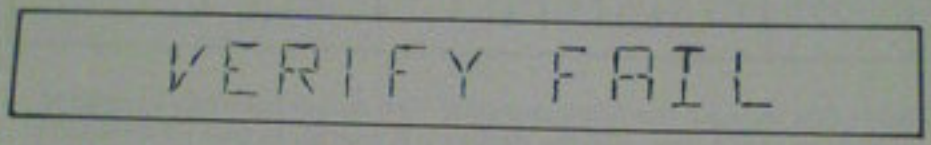
AUTOMATIC VERIFY

The automatic verify function is particularly useful when the algorithm of a device precludes the operation of the in-program verify.

If a device passes automatic verify the display will show:



A failure will cause the display to show:



Detailed information about the failure takes the same format as for the in-program verify.

MANUAL VERIFY

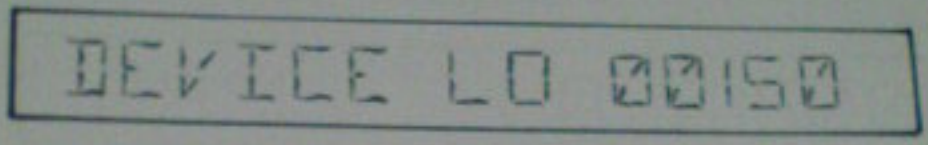
This function can be applied at any time and not just during programming by pressing the 'verify' key. Its operation and display is identical to automatic verify.

3.8 DEVICE ADDRESS LIMITS (SET F6) – PP40

All device-related functions on the PP40 are defined by two parameters: the address limits. These are Address High and Address Low.

Press SET F6

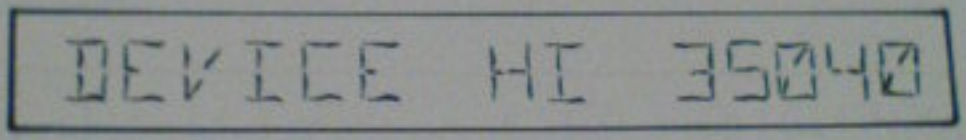
to set the address limits. The display will show:



A new address limit can be entered using the hexadecimal keyboard.

Press ↓ or ↑

to display:



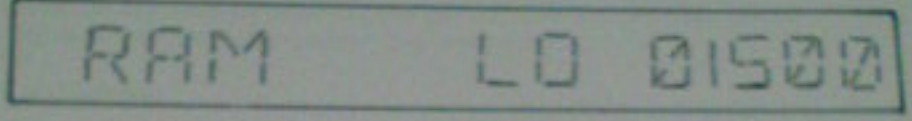
A new address limit can be entered using the hexadecimal keyboard.

3.9 DEVICE ADDRESS LIMITS (SET F6) – PP41 and PP42

All device-related functions on the PP41 and PP42 are defined by three parameters: the address limits. These are Address High, Address Low and RAM Low.

Press SET F6

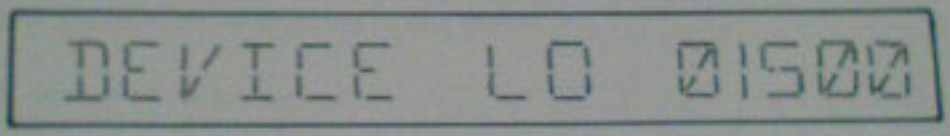
to set the address limits. The display will show:



A new address limit can be entered using the hexadecimal keypad.

Press [↓]

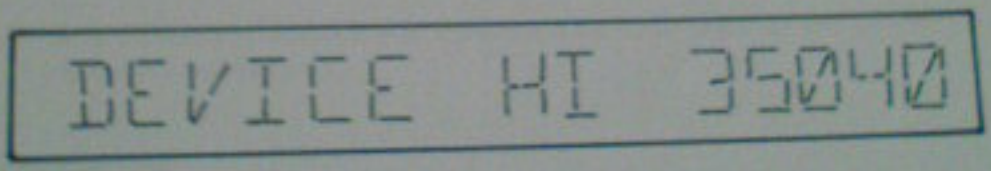
to display:



A new address limit can be entered using the hexadecimal keypad.

Press [↓]

to display:



The upper limit defaults to the size of the device but a new limit can be entered using the hexadecimal keypad.

Pressing [↑] displays the address limits in the reverse order.

4.1 KEYBOARD EDIT ROUTINES

The comprehensive editor on the PP41 and PP42 enables the following functions:

LIST

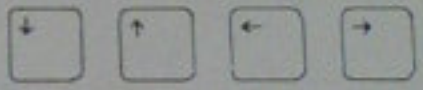
This is a feature enabling the data content of the RAM to be scanned on the display, without the danger of changing the RAM data.

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

For example:

LOCATION (ZERO)	DATA
00000	FF

The address can be scanned in two ways:

1. By use of the cursor keys: 

   (a) By using the right/left cursor keys the address can be incremented or decremented a single location at a time.

   (b) By using the up/down cursor keys the address can be incremented or decremented 16_{10} locations at a time.

2. Any address with RAM limits can be directly entered by use of the hexadecimal keypad.

For example:

SELECTED ADDRESS	DATA
01FF0	29

*Note: if 16 or 32 bit configurations are selected on the PP42 the address shown will be the true address and the data will be four or eight digits in size.

4.2 EDIT

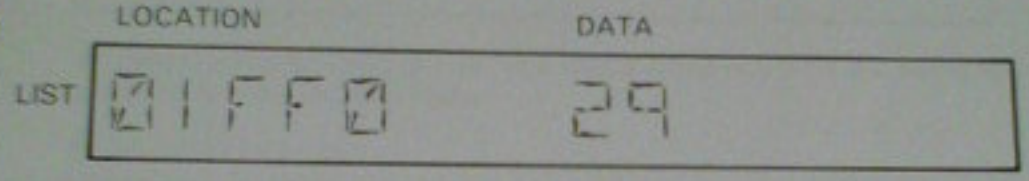
This is a feature whereby the actual content of the RAM can be directly modified by using the keyboard.

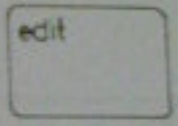
The edit mode can be selected in two ways.

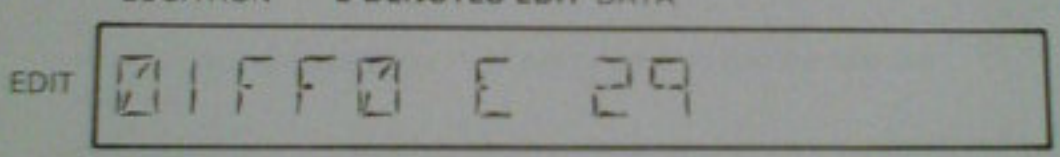
(a) By pressing the edit key when the machine is in the normal operating mode.
(b) By pressing the edit key when the machine is in the list mode. (The list mode can be reselected in the same manner).

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

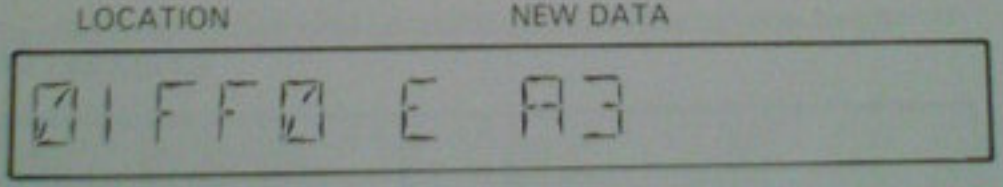
For example:







The data '29' at location '01FF0' can now be changed by use of the hexadecimal keypad into A3, for example.



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 hexadecimal keypad.

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

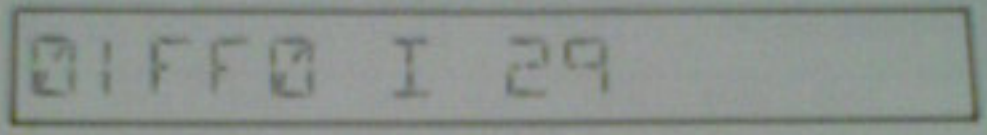
4.3 INSERT

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

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

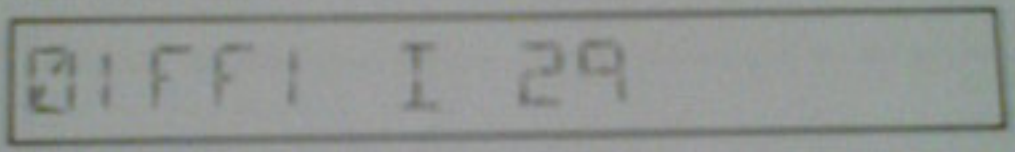
For example:

LOCATION **I DENOTES INSERT** DATA

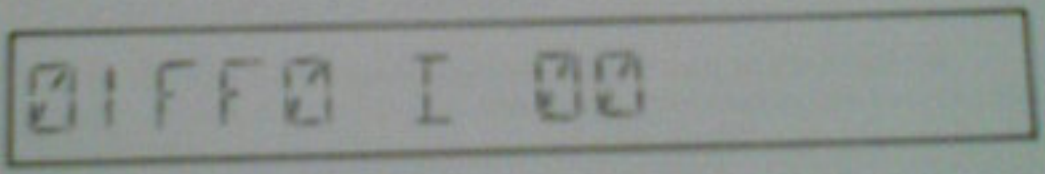


By pressing the SET key all data inclusive of location 01FF0 and above is repositioned one location higher

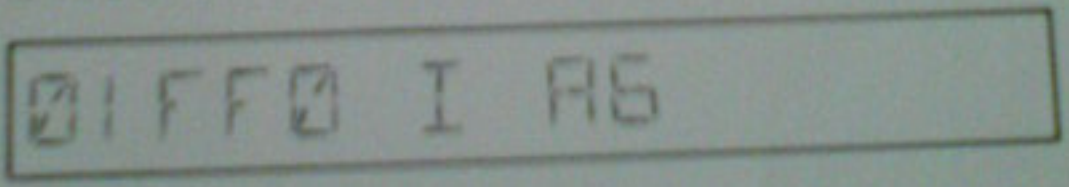
NEXT LOCATION UP



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



By use of the hexadecimal keypad the chosen data can now be inserted for instance A6:

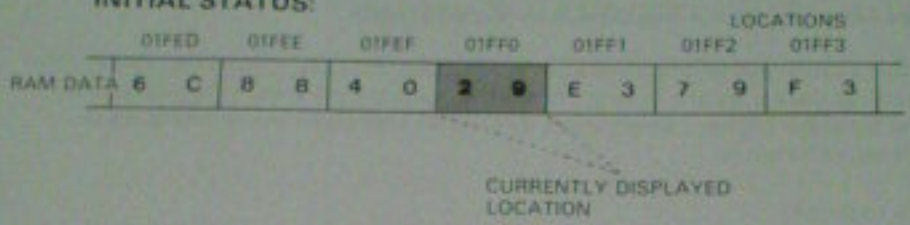


Other than the use of the set key, operation in the Insert mode remains the same as when in the ordinary edit mode.
For graphic example see next page

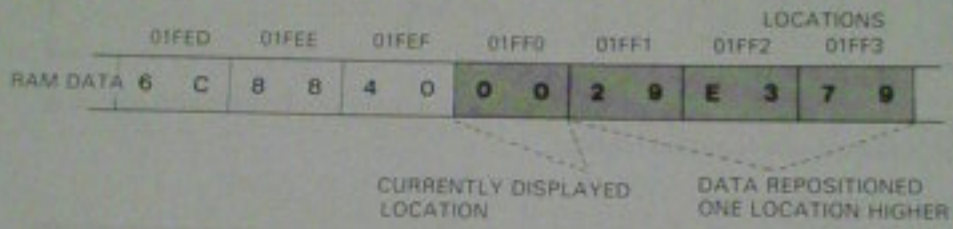
If 16 or 32 bit configurations are selected on the PP42 the address

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

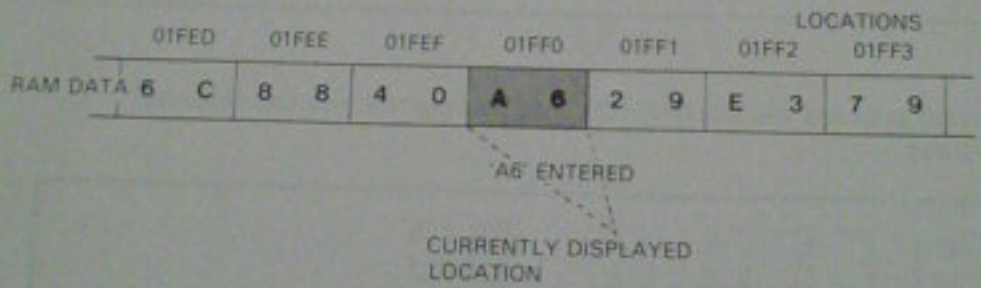
INITIAL STATUS:



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:



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



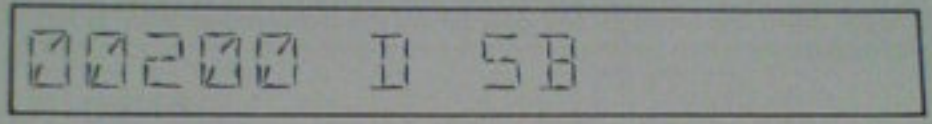
4.4 DELETE

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

The data above the selection address is repositioned one location lower.

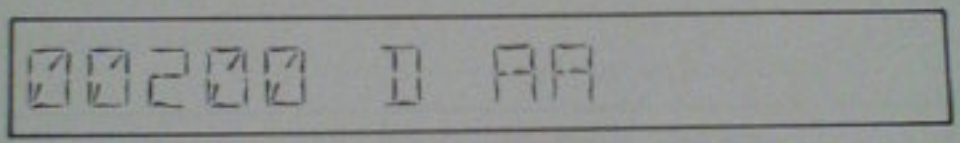
For example: 5B is the data to be deleted.

LOCATION **D DENOTES DELETE** DATA



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:



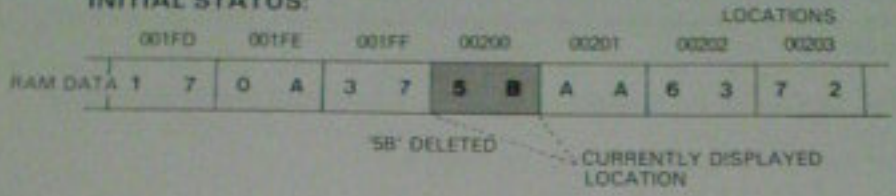
Other than the use of the set key, operation in the delete mode remains the same as when in the ordinary edit mode.

For graphic example see next page.

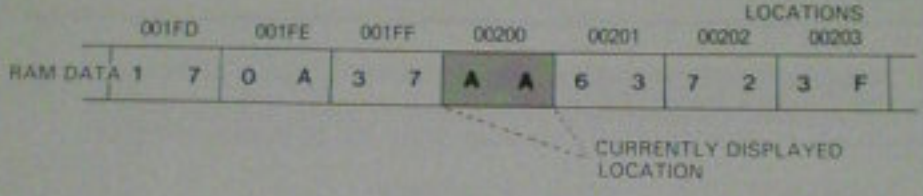
*Note: if 16 or 32 bit configurations are selected on the PP42 the address shown will be the true address and the data will be four or eight digits in size.

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

INITIAL STATUS:



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



4.5 BLOCK MOVE (SET F4)

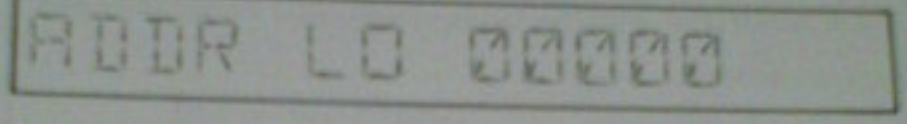
SETTING ADDRESS LIMITS

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

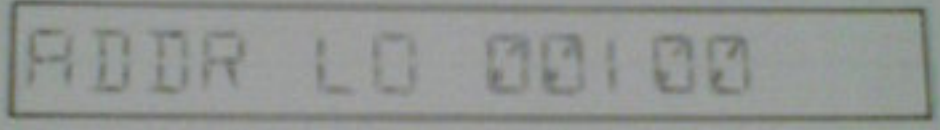


This defines the lower limit of the block in RAM to be re-located. (Defaults to 0000)

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

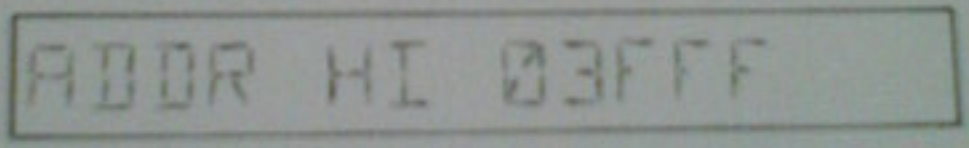
For example 00100:

NEW LOWER RAM LIMIT



If [↓] is pressed the display will show:

ADDRESS HIGH SIZE OF SELECTED DEVICE

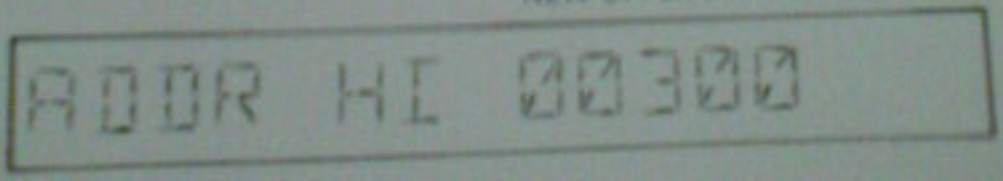


This defines the upper limit of the block in RAM to be relocated. (Defaults to selected device size)

A new value for the upper RAM limit can be entered using the hexadecimal keypad.

For example 00300

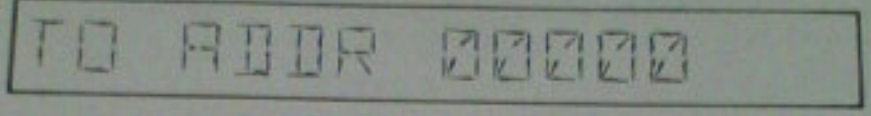
NEW UPPER RAM LIMIT



LOWER LIMIT OF RE-LOCATED DATA

By pressing [↓] again the display will show:

TO ADDRESS

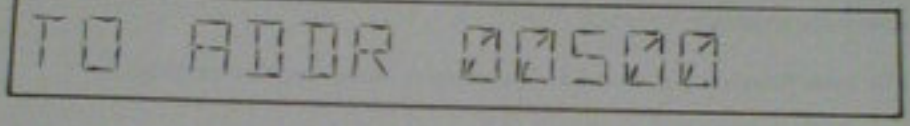


This defines the lower RAM limit of where the block of data is to be re-located (Defaults to 0000).

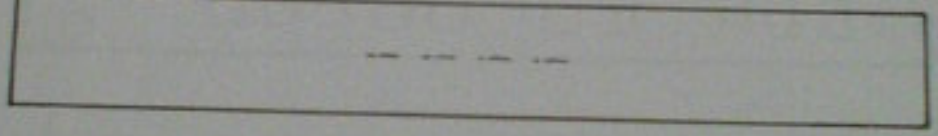
The re-located lower RAM limit can be entered using the hex-keyboard.

For example 00500:

LOWER LIMIT OF THE
NEW BLOCK OF DATA

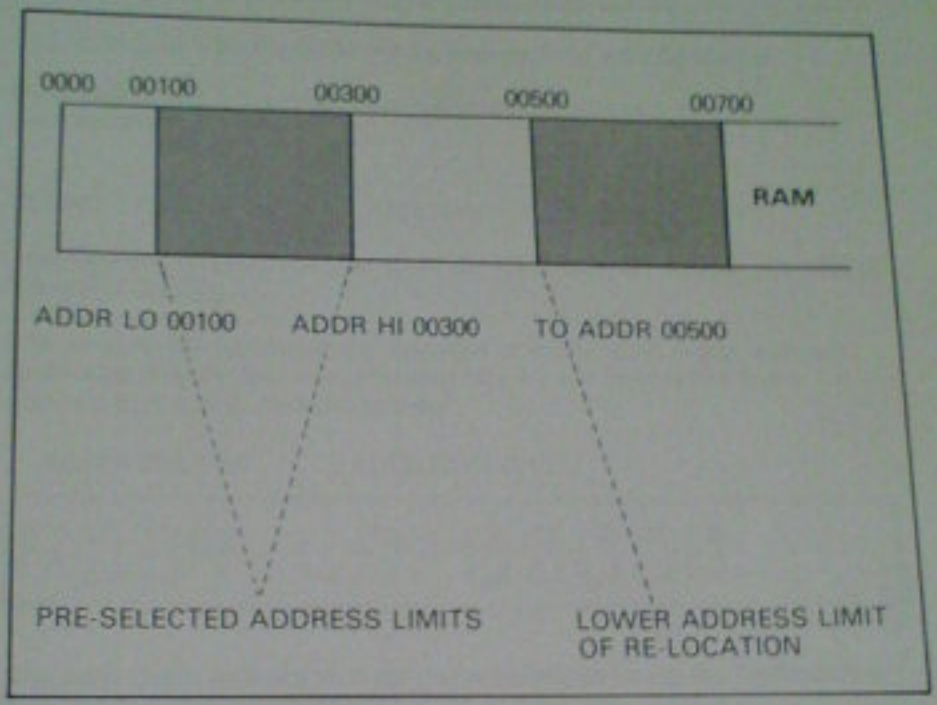


Pressing the exit key will initiate the block-move function. A series of dashes will be displayed indicating the function is in progress:



The programmer will automatically return to the normal operating mode.

A GRAPHIC EXAMPLE OF HOW THE BLOCK MOVE FUNCTION WORKS IS SHOWN BELOW:



4.6 FILLING THE RAM

By pressing SET FF the RAM will be entirely filled with Fs.

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

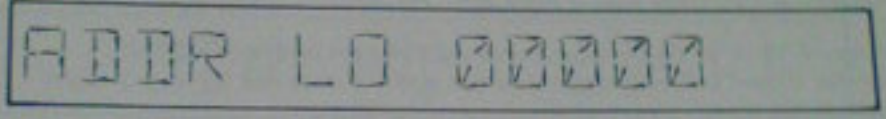
By pressing SET 5 the RAM data will be complemented. (1's complement).

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

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

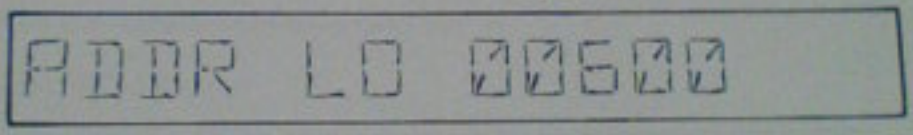
The variable will be identically repeated at every word within address limits specified by the user. Pressing SET F2 will display the lower address limit which defaults to zero:

ADDRESS LOW LOCATION ZERO



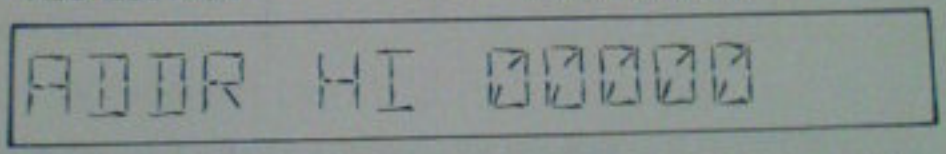
The new lower address limit can be selected by using the hexadecimal keypad, for example 00600:

LOCATION



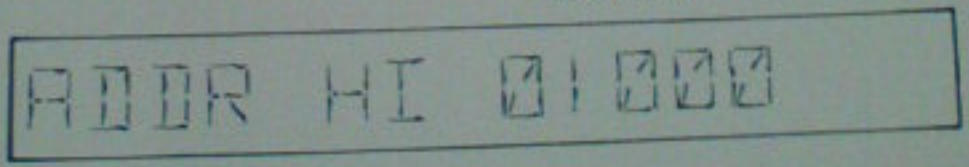
The upper address limit can be shown by pressing [↑] and this also defaults to the device size.

ADDRESS HIGH LOCATION ZERO

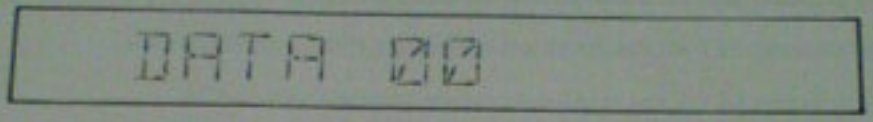


The new upper limit can be selected using the hexadecimal keypad, for example 01000:

LOCATION

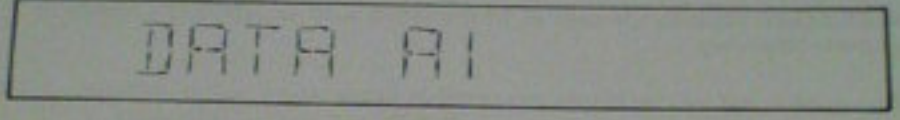


The arbitary variable can be entered by pressing the [↑] again to display:



The data selection can be made by using the hexadecimal keypad, for example A1:

ARBITARY VARIABLE



Pressing 'SET' alone will implement this selection.

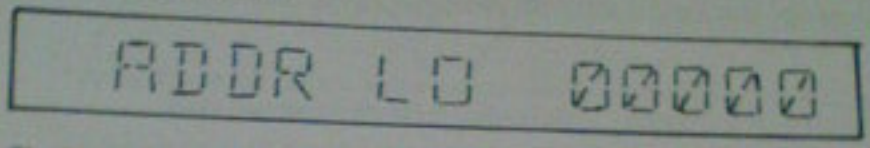
Every byte of RAM within and inclusive of the specified address limits of 00600 low to 01000 high is filled with 'A1'.

*Note: if 16 or 32 bit configurations are selected on the PP42 the address shown will be the true address and the data will be four or eight digits in size.

4.7 STRING SEARCH

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

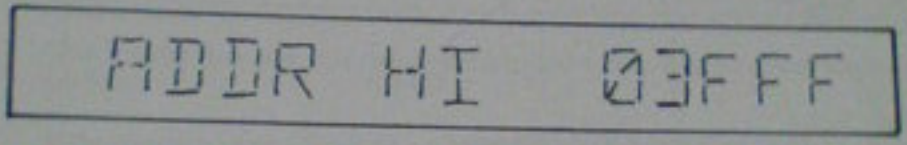
Press SET 9 to display:



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 keypad.

To display the upper limit:

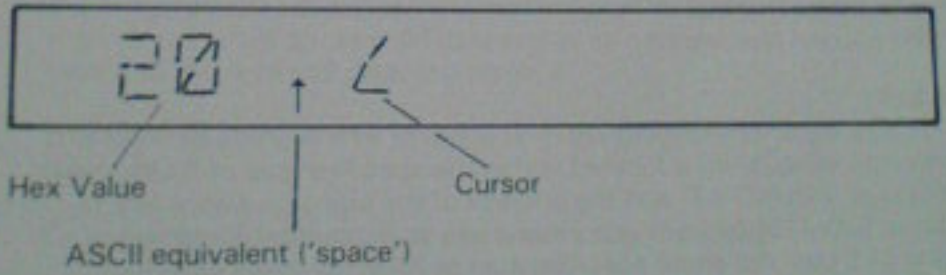
Press ↑ or ↓



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 keypad.

Once the address limits have been set:

Press SET to display:



To the extreme left of the display is the hex equivalent of the ASCII character displayed on the immediate left of the cursor. In this case the space character is displayed.

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

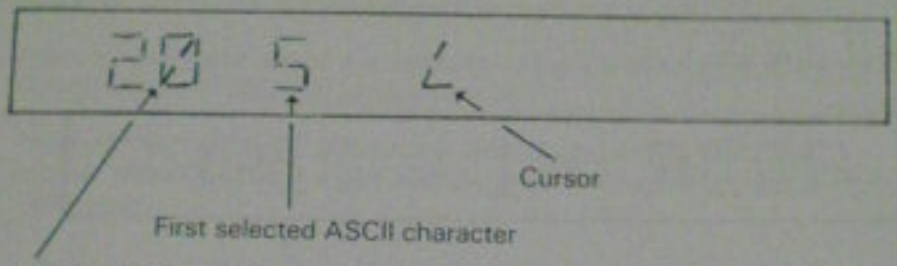
Press [↑] or [↓]

Alternatively and more quickly, the hex value 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 ASCII character:

Press [→]



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

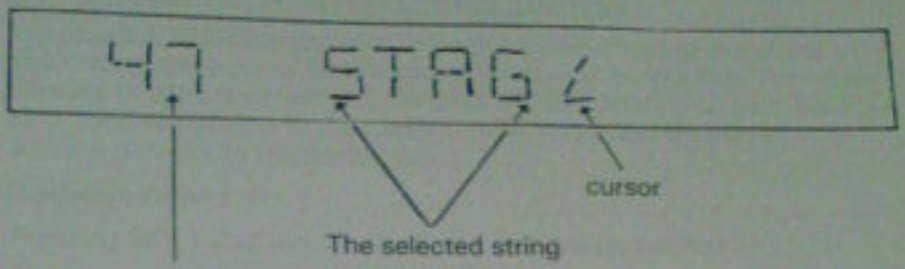
The second character can now be selected in the manner previously described. In this way a string of up to 11 characters (or data bytes) can be entered.

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

Press SET

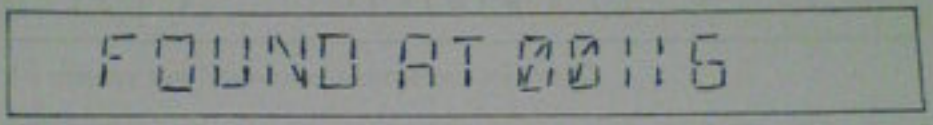
If a corresponding string is located within the specified area of RAM, then the message 'FOUND AT' and the address of the first occurrence will be displayed. Every subsequent occurrence can be located by continually pressing SET until the entire specified area of RAM has been searched.

For instance:



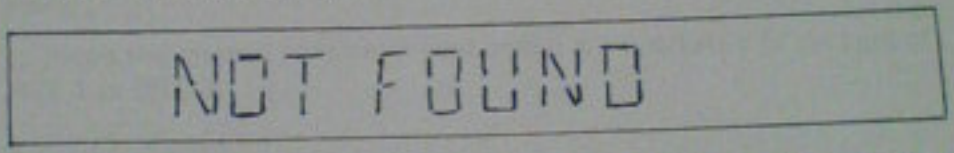
The ASCII value of the character to the immediate left of the cursor. (In this case 'G')

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 movifng 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 programmer is powered down.

To abort the String Search at any time.

Press EXIT.

5.1 SETTING THE I/O INTERFACE PARAMETERS

It is possible to initiate an input or output without setting any of the associated parameters. Simply press the input or output key. The offset defaults to the offset last set, the start address defaults to zero, the input stop address defaults to the maximum RAM size and the output stop address defaults to the device size.

Hardware Parameters

Pressing SET 1 displays the hardware parameters for Port 1:

PORT NUMBER	BAUD RATE	WORD LENGTH	NO. OF STOP BITS	PARITY
PRT1.	9600	8	2	EP

To display the hardware parameters for Port 2:

Press SET 1

followed by Key 2 or just key 2 if SET has already been pressed.

The parameters for both ports can be changed. This is done by use of the horizontal cursor keys.

Press ←

to move the chosen parameter field until it is immediately to the right of PRT 1 or PRT 2.

Press ↑ or ↓

to modify the parameter.

Software Parameters

Pressing SET 1 followed by Key 3 displays the software parameters:

SOFTWARE PARAMETER	FORMAT	PASS-THROUGH OPTION	CONTROL Z OPTION
SWR.	INT	TRSP	CZ

Press ↑ or ↓

to select the required format.

Press ←

to enable selection of Pass-Through.

Press ↑ or ↓

to select Transparent (Pass-Through mode) or Normal (non-Pass-Through mode)

Press [*]

to enable the selection of Control Z

Press [↑] or [↓]

to select Control Z

On completion of parameter selection press the Exit key to return to the initial power up display.

Display Abbreviations

PRT	=	Port
TRSP	=	Transparent
NRML	=	Normal
SWR	=	Software
CZ	=	Control Z

The selection available in each category is shown in this table.

FORMAT	BAUD RATE	WORD LENGTH	NUMBER OF STOP BITS	PARITY
INT	38K4	8	2	—
HASC	19K2	7	1	EP
XOR	9600	6		OP
TEK	4800	5		
XTEK	2400			
PPX	2000			
BIN	1800			
DBIN	1200			
MOST	600			
	300			
	150			
	110			
	75			
	50			

FORMAT		
Key to Abbreviations		
INT	=	INTELLEC
HASC	=	HEX ASCII
XOR	=	EXORCISOR
TEK	=	TEK HEX
XTEK	=	EXTENDED TEK
PPX	=	STAG HEX
BIN	=	BINARY
DBIN	=	DEC BINARY
MOST	=	MOS TECHNOLOGY

PARITY		
Key to Abbreviations		
— —	=	NO PARITY CHECK
EP	=	EVEN PARITY
OP	=	ODD PARITY

5.2 Setting the I/O Address Parameters

The pre-settable address parameters are:

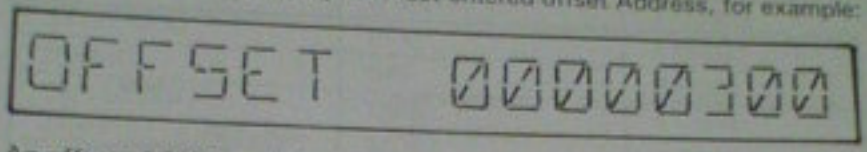
INPUT ADDRESS OFFSET
INPUT START ADDRESS
INPUT STOP ADDRESS

OUTPUT ADDRESS OFFSET
OUTPUT START ADDRESS
OUTPUT STOP ADDRESS

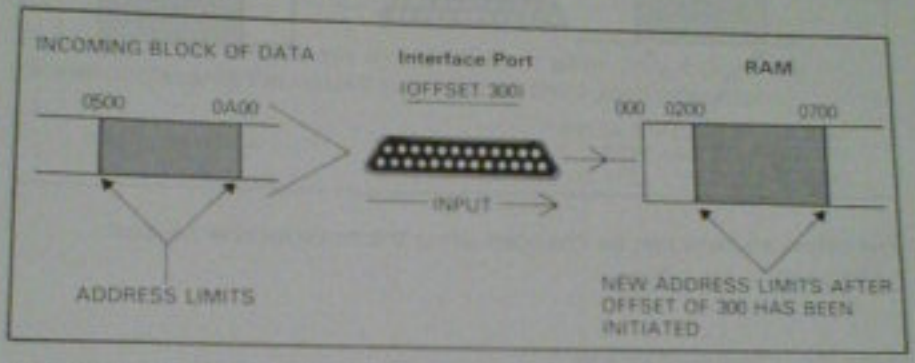
Input Parameters

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

Pressing 'Set Input' displays the last-entered offset Address, for example:



An offset of 300 would look like this:



The offset address can be changed using the hexadecimal keypad.
00000000 = No Offset

Press [↓] followed by Key 1

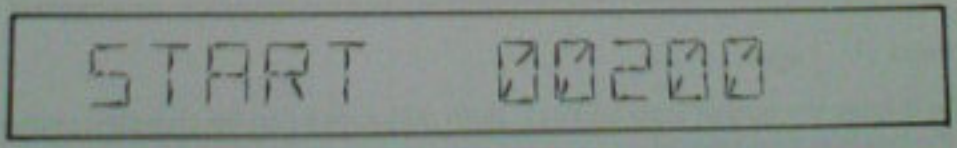
to select Port 1 for the data input.

or Press [↓] followed by Key 2

to select Port 2 for the data input.

Press [↓]

to display the Input start address.

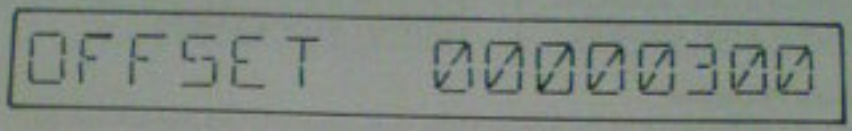


Press [↓] again

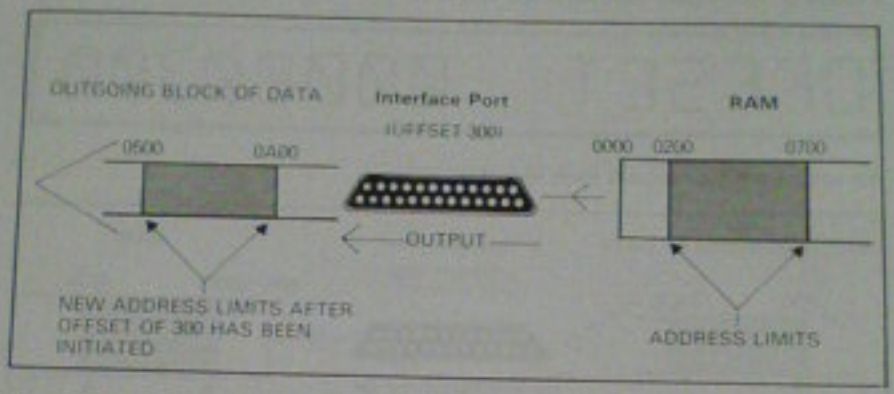
to display the Input stop address. Both addresses can be changed using the hexadecimal keypad.
Pressing the input key initiates the input operation.

Output Parameters

An outgoing block of data originating from the RAM can be re-located at a higher address, using an output offset. Presing 'Set Output' displays the last-entered Offset Address, for example:



An offset of 300 would look like this:

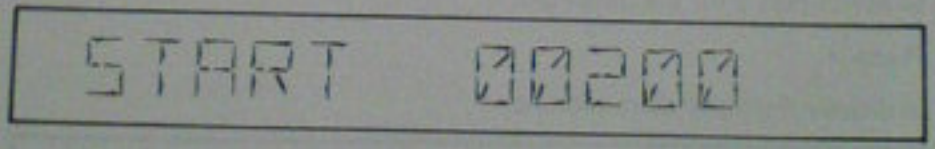


The offset address can be changed using the hexadecimal keypad.
00000000 = No Offset

Press [↓] followed by Key 1
to select Port 1 for the data output.

or Press [↓] followed by Key 2
to select Port 2 for the data output.

Press [↓]
to display the output start address.



Press [↓] again
to display the output stop address. Both addresses can be changed using the hexadecimal keypad.

FORMAT OFFSET TYPES

FORMAT	DISPLAY ABBREVIATION	NO. OF RELEVANT DIGITS IN ADDRESS FRAME
INTELLEC	INT	5 Digits
HEX ASCII	HASC	0 Digits
MOTOROLA S-RECORD	XOR	8 Digits
TEK-HEX	TEK	4 Digits
EXTENDED TEK	XTEK	17 Digits
STAG HEX	PPX	4 Digits
BINARY	BIN	0 Digits
DEC BINARY	DBIN	4 Digits
MOS TECHNOLOGY	MOST	4 Digits

Eight characters are always displayed even when only 4 digits are required. Pressing the output key initiates the output operation.

5.3 ERROR REPORTING ON INPUT/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 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.

6.1 TRANSLATION FORMATS (INTRODUCTION)

There are nine formats available on the PP41 and PP42:

INT	=	INTELLEC
HASC	=	HEX ASCII
XOR	=	MOTOROLA S-RECORD
MOST	=	MOS TECHNOLOGY
TEK	=	TEK HEX
XTEK	=	EXTENDED TEK
PPX	=	STAG HEX
BIN	=	BINARY
DBIN	=	DEC BINARY

STANDARD FORMATS

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

EXTENDED FORMATS

There is one protracted version of the standard formats: EXTENDED TEK. The extended format 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.

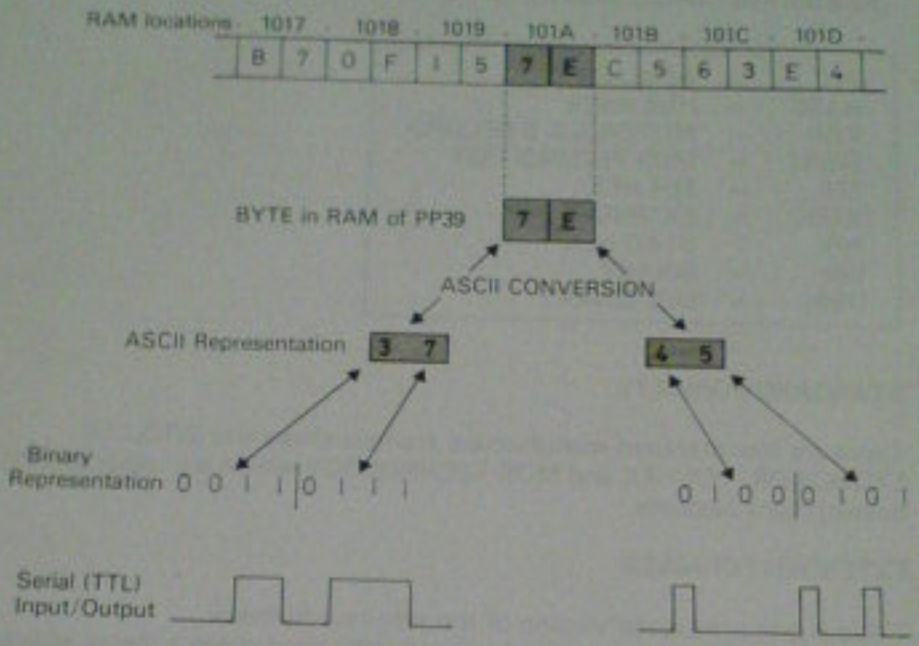
BINARY

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.

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 PROGRAMMER'S RAM



6.2 INTELLEC

The intellec format when displayed consists of:

a. A start code, i.e. (a colon);

b. The sum of the number of bytes in an individual record, e.g. 1A.

c. The address of the first byte of data in an individual record, e.g. 0000.

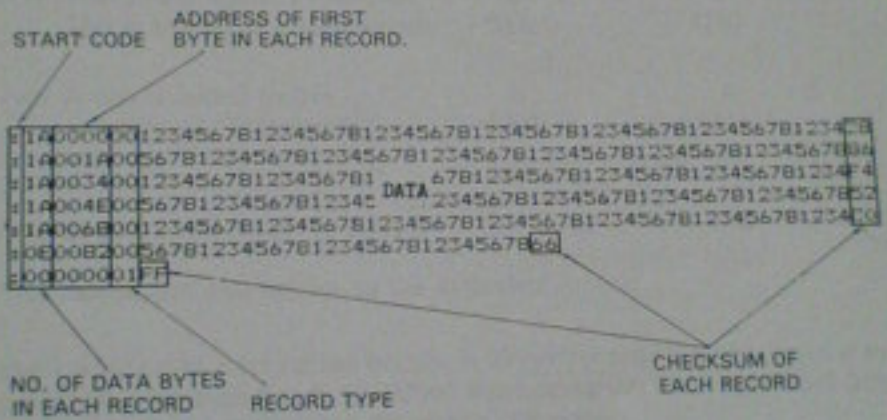
d. The record types, i.e. 00-Data Record
01-End Record.

e. Data in bytes, e.g. 12 34 56 78.

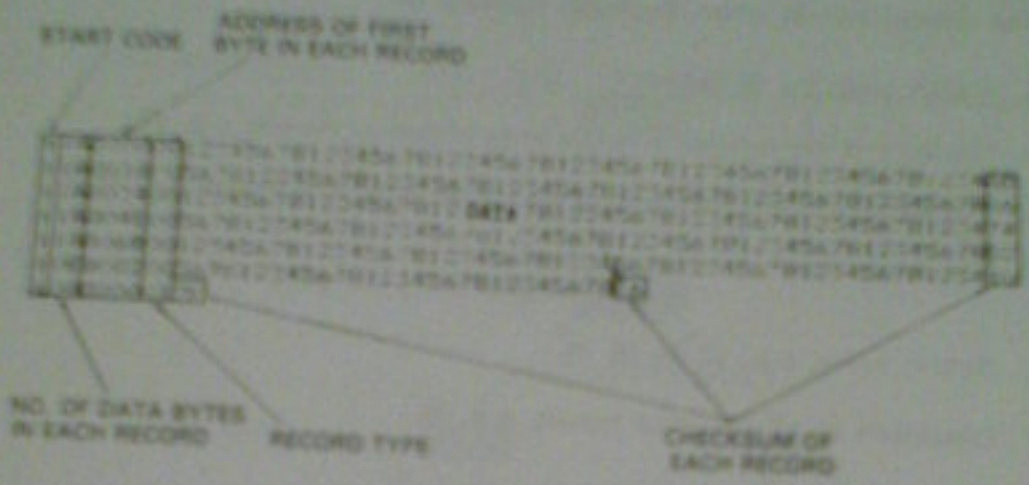
f. Checksum of an individual record, e.g. 28.

For example:

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



AND WITH AN OFFSET OF (8000)



CALCULATION OF THE INTELLEC* CHECKSUM

:1400000012345678123456781234567812345678123456781234567812345678123428
:00000001FF

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

(i) This is: :01 00 1A 00 56 8F

(ii) The start code and the checksum are removed: :8F

(iii) Five Bytes remain: 01 00 1A 00 56

(iv) These are added together: 01 + 00 + 1A + 00 + 56 = 71

(v) The total '71' is converted into Binary: 7 0111 1 0001

(vi) The Binary figure is reversed. This is known as a complement: 8 1000 E 1110

(vii) A one is added to this compliment. This addition forms a "2's complement": 8 1000 F 1111

(viii) 8F is the checksum as above: :01 00 1A 00 56 (8F)

*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.

6.3 HEX ASCII

The Hex ASCII format when displayed consists of:

DATA ALONE

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

(A hidden start character known as Control B.
(02: ASCII Code, STX ASCII character).

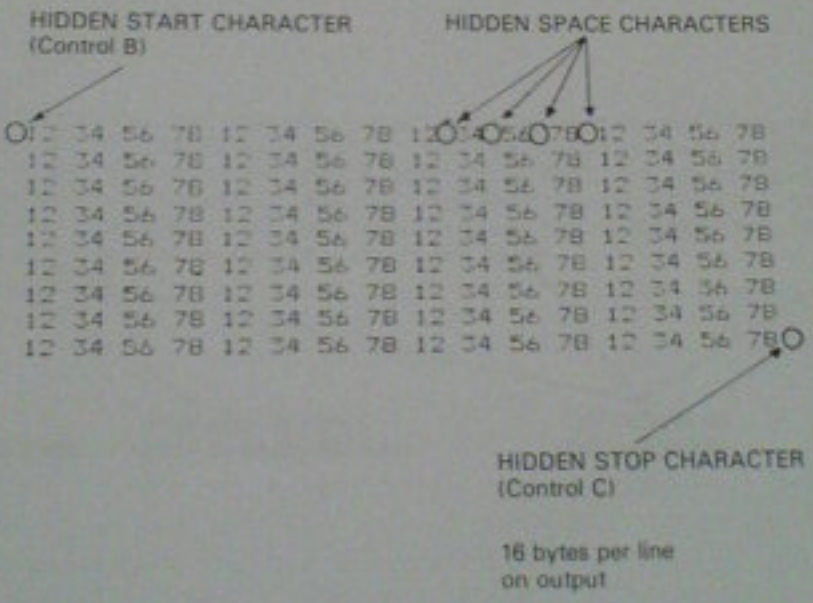
A hidden stop character known as Control C.
(03: ASCII Code, ETX ASCII character)

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



16 bytes per line
on output

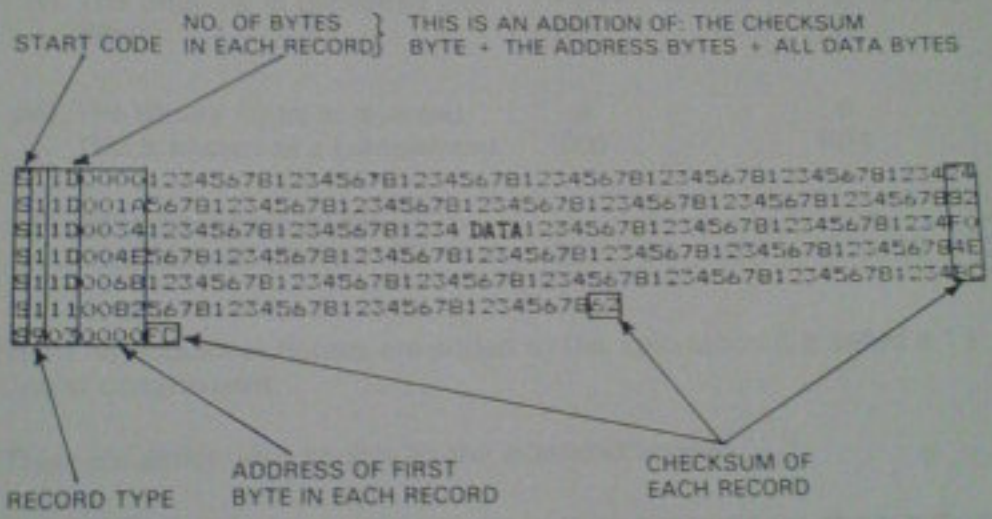
6.4 MOTOROLA S-RECORD

The Motorola S-Record when displayed consists of:

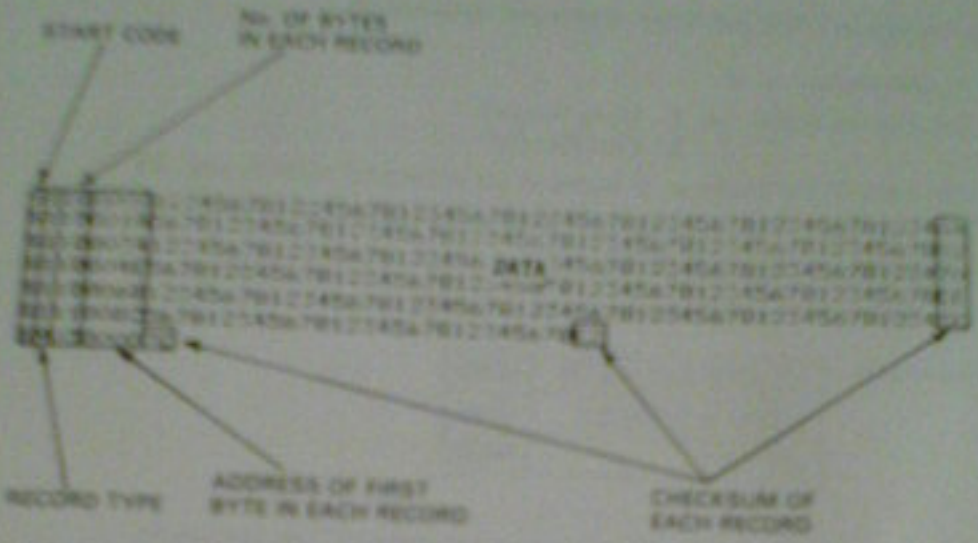
a. A start code, i.e. S
b. The record types, i.e. 1 - Data Record
   9 - End Record
c. The sum of the number of bytes in an individual record, e.g. 1D
d. The address of the first byte of data in an individual record, e.g. 0000
e. Data in bytes, e.g. 12 34 56 78
f. Checksum of an individual record, e.g.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

For example:

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



AND WITH AN OFFSET OF (8000)



CALCULATION OF THE EXORCISOR* CHECKSUM

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

(i) This is: S1 04 00 1A 56 8B

(ii) The start code, the record type and the checksum are removed: S1 8B

(iii) Four Bytes remain: 04 00 1A 56

(iv) These are added together: 04 + 00 + 1A + 56 = 74

(v) The total '74' is converted into Binary:

7	4
0111	0100

(vi) The Binary figure is reversed. This is known as a complement:

8	B
1000	1011

(vii) 8B corresponds to the checksum as above:

S1	04	00	1A	56	(8B)

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

*This calculation also applies to the extended version.

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

6.5 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

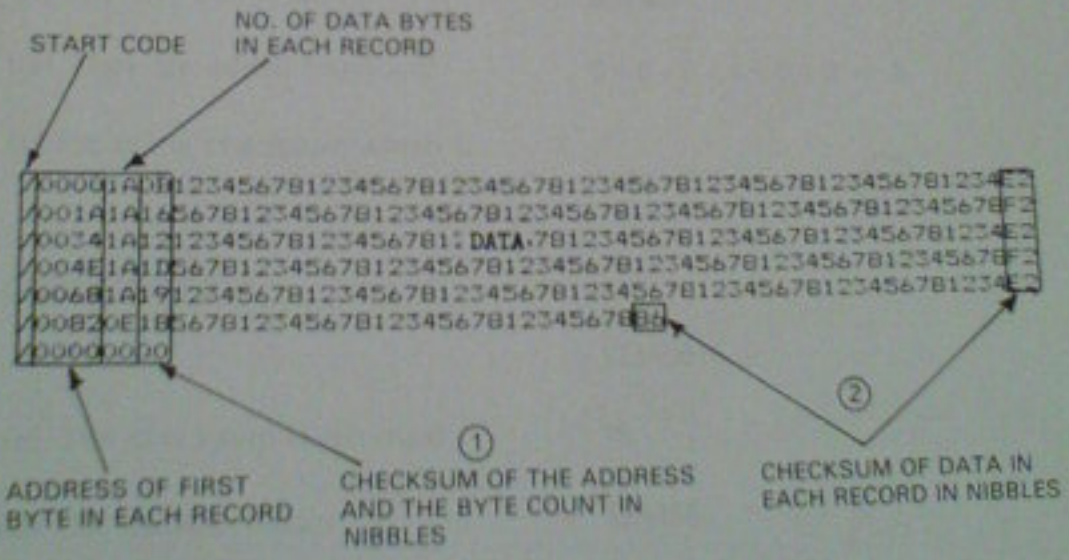
e. Data in bytes, e.g. 12 34 56 78

f. Checksum 2 which is a nibble addition of all data.

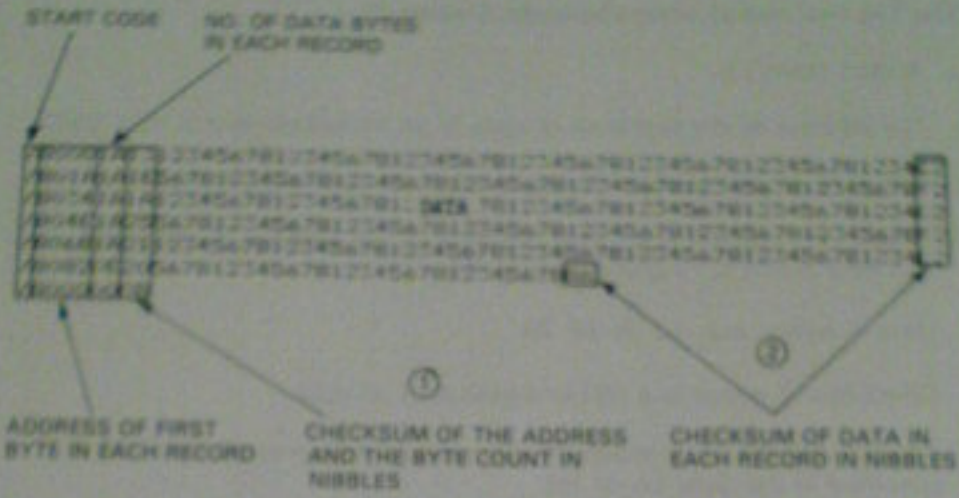
g. An end record which automatically stops the operation when 00 is specified in the byte count (c).

For example:

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



AND WITH AN OFFSET OF (8000)

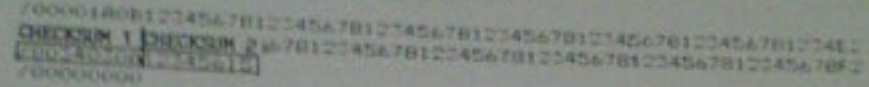


CALCULATION OF TEK HEX CHECKSUMS

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

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

Checksum 2 is a nibble addition of data alone.



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.

6.6 EXTENDED TEK HEX

The Extended Tek Hex when displayed consists of:

a. A start code: % (percentage)

b. A count of the nibbles in an individual record, e.g. 3B

c. The record types, i.e. 6—Data Record
8—End Record

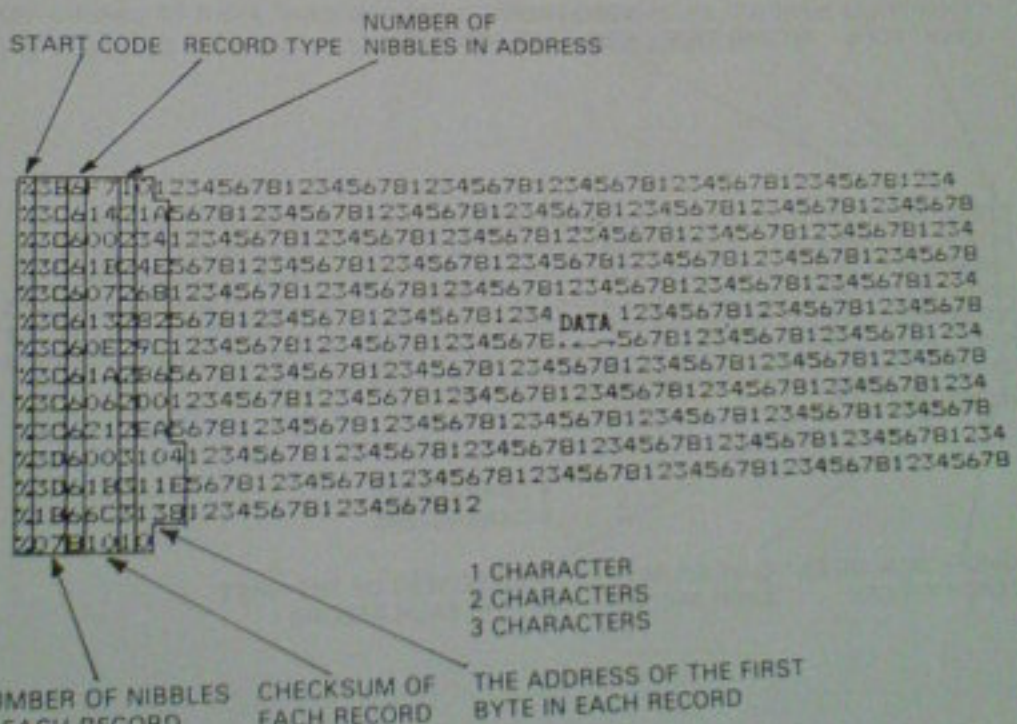
d. A checksum of the whole of an individual record excluding the %, e.g. F7

e. *The number of nibbles comprising—"the address of the first byte in each record", e.g. 1, 2, 3 etc.

f. The address of the first byte of data in an individual record, e.g. 0, 1A, 104

For example:

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



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

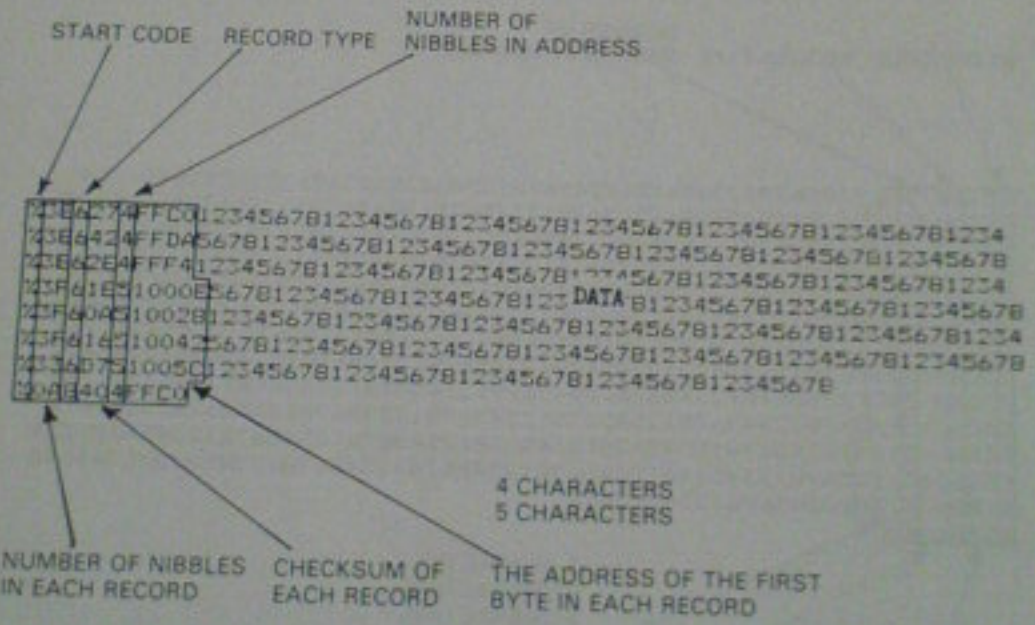
2/1A	6/100000	A/1B4625DC95
2 Characters	6 Characters	A Characters (10 in Decimal)

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

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

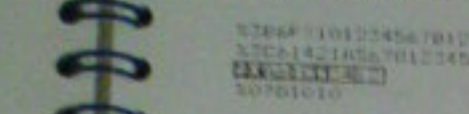
For example:

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



CALCULATION OF THE EXTENDED TEK HEX CHECKSUM

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



Example: THE THIRD LINE OF THE ABOVE FORMAT.

(i) This is: % 0A61C23412

(ii) The start code and the checksum are removed: % 1C

(iii) Eight nibbles remain: 0A623412

(iv) These are added together: 0 + A + 6 + 2 + 3 + 4 + 1 + 2 = 1C

(v) 1C is the checksum as above; % 0A6 (1C) 23412

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

6.7 PPX or (STAG HEX)

The PPX format when displayed consists of:

a. A start code, i.e. * (an asterisk, 2A – ASCII Code)

b. The address of the first byte of data in an individual record, e.g. 0000

c. Data in bytes, e.g. 12 34 56 78

d. A stop code, i.e. $ (a dollar sign, 24 – ASCII Code)

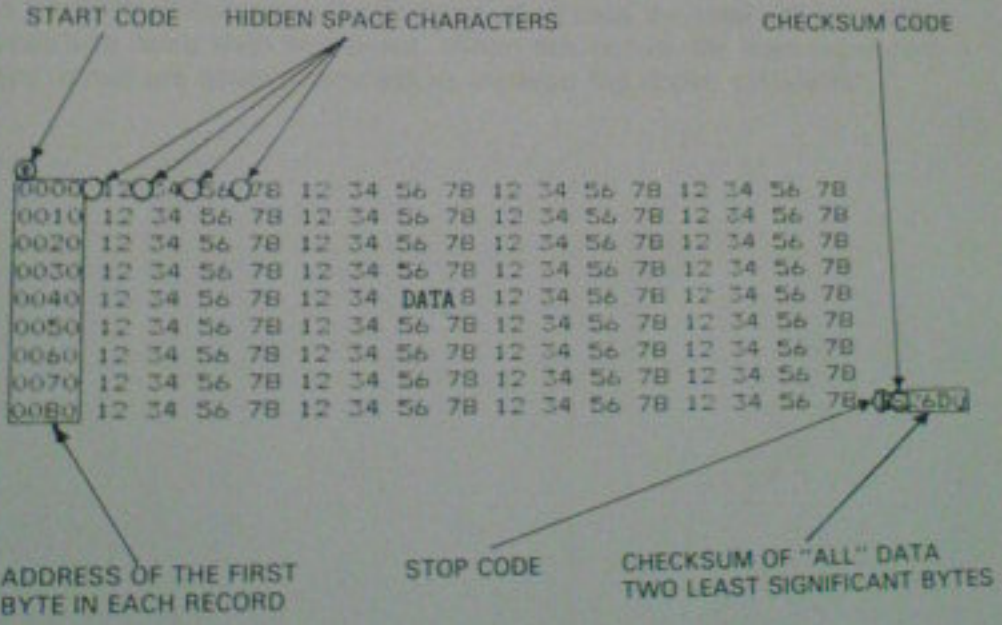
e. A checksum start code: S

f. A checksum of all data over the entire address range. (The displayed checksum is the two least significant bytes.)

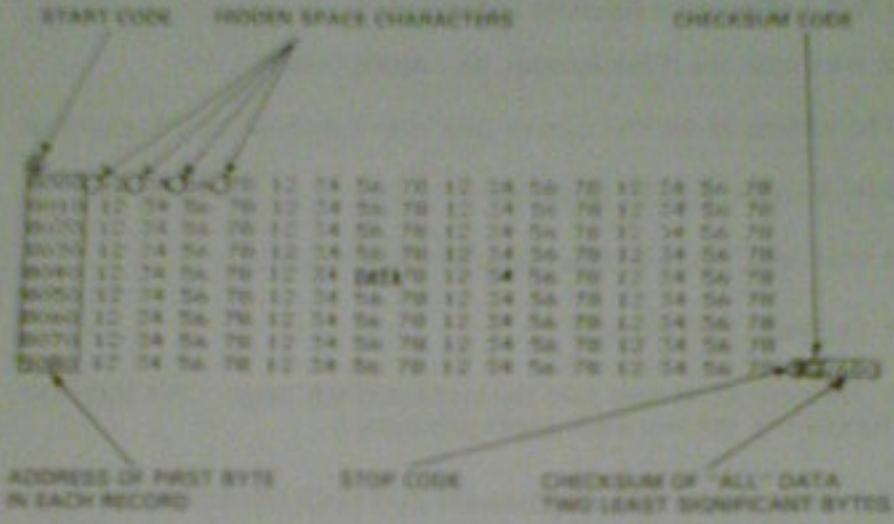
g. An invisible space character between data bytes (20-ASCII Code)

For example:

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

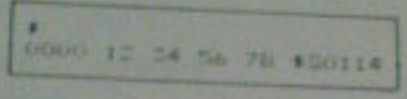


And with an Offset of 8000



CALCULATION OF THE PPX CHECKSUM

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



Example: THE SEGMENT OF DATA ABOVE

(i) This is: *0000 1 34 56 78 $S0114

(ii) The start code, the address, the stop code, the checksum code and the checksum are removed: *0000 $S0114

(iii) Four bytes remain: 12 34 56 78

(iv) These are added together: 12 + 34 + 56 + 78 = 114

(v) 114 is the checksum which is displayed in two byte form as above: *0000 12 34 56 78 $S 0114

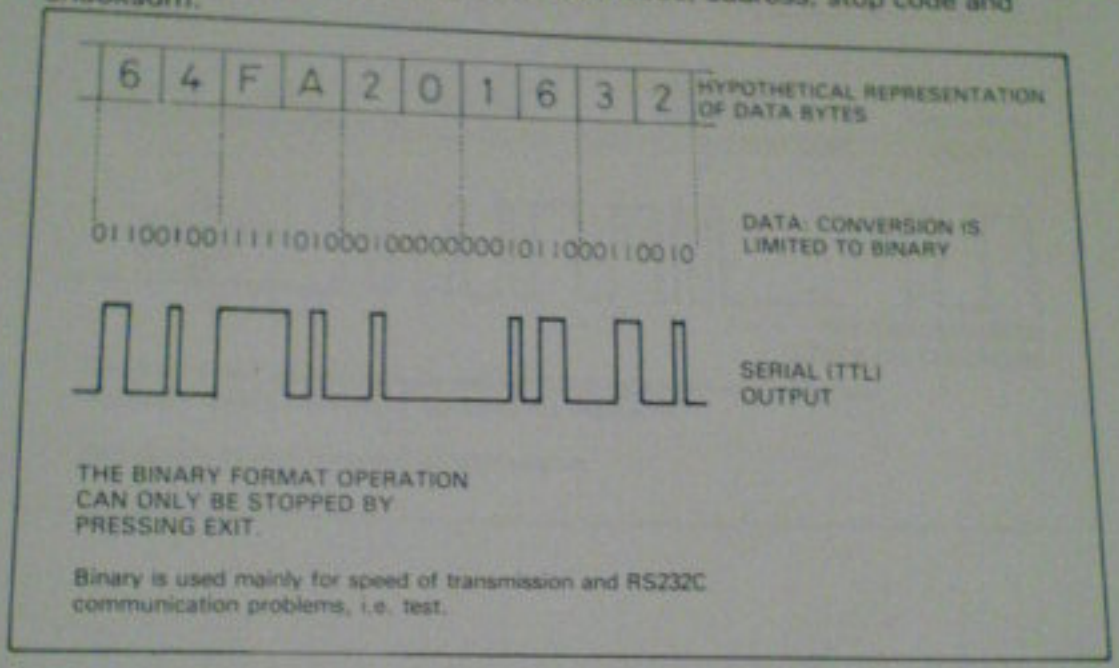
As the PPX checksum is an addition of all data the total will invariably constitute more than two bytes. When this occurs the least significant 'two' bytes are always selected to undergo the above calculation.

6.8 BINARY AND DEC BINARY

Binary and DEC Binary 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 ONE'S or ZERO's. Hence 'Binary'. A visual display is not possible.

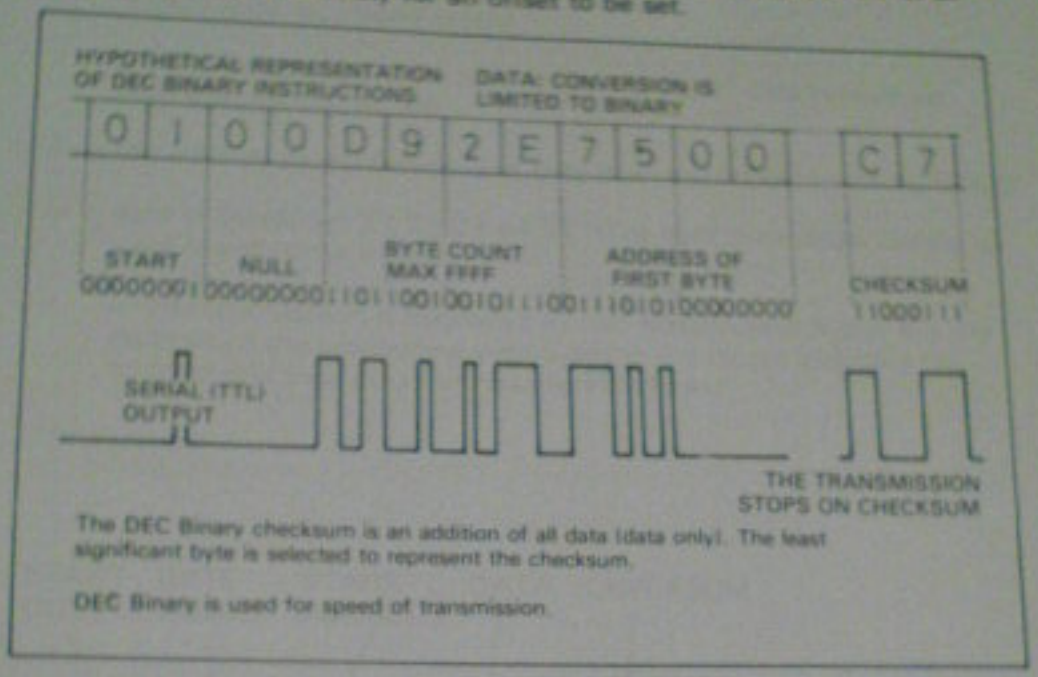
6.8.1 BINARY

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



6.8.2 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.



The DEC Binary checksum is an addition of all data (data only). The least significant byte is selected to represent the checksum.

DEC Binary is used for speed of transmission.

6.9 MOS-TECHNOLOGY

The MOS-TECHNOLOGY format consists of:

i. a start code, ie: ; (semi-colon)

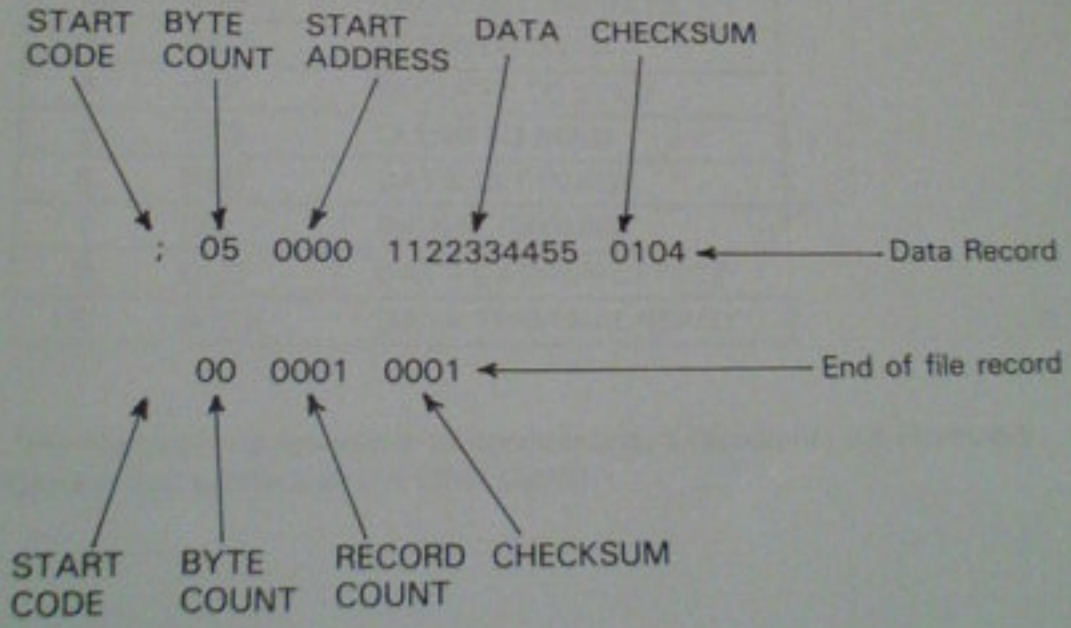
ii a byte count – that is the sum of the number of data bytes in an individual record, eg: 05

iii the address of the first byte of data in an individual record, eg: 0000

iv data in bytes, eg: 11 22 33 44 55. (The data bytes must consist of valid hexadecimal digits).

v. a checksum which is displayed as two hexadecimal bytes. It is the addition of the preceeding data bytes in the record including the address and byte count in hexadecimal form.

For example:



Calculation of MOS-Technology checksum

```
; 05 0000 11 22 33 44 55 0104
; 00 0001 0001
```

Example: the first line of the above format.

i. this is ; 05 0000 11 22 33 44 55 0104

ii. the start code and the checksum are removed: ; 0104

iii. this leaves:
the byte count: 05
the address of the first byte in the record: 0000
and five data bytes: 11 22 33 44 55

iv. these are added together:
05 + 0000 + 11 + 22 + 33 + 44 + 55 = 0104

v. 0104 is the checksum as above: ; 0500001122334455 0104

7.1 RS232C INTERFACE PORT CONNECTIONS

The PP41 and PP42 are linked to peripheral equipment via their two RS232C interface ports.

LINK-UP TO PERIPHERAL EQUIPMENT

There are two distinct types of machine:

(i) Data Terminal Equipment (DTE)

(ii) Data Communication Equipment (DCE)

The PP41 and PP42 fall into both categories: Port 1 is configured as DTE, Port 2 is configured as DCE.

On the 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.

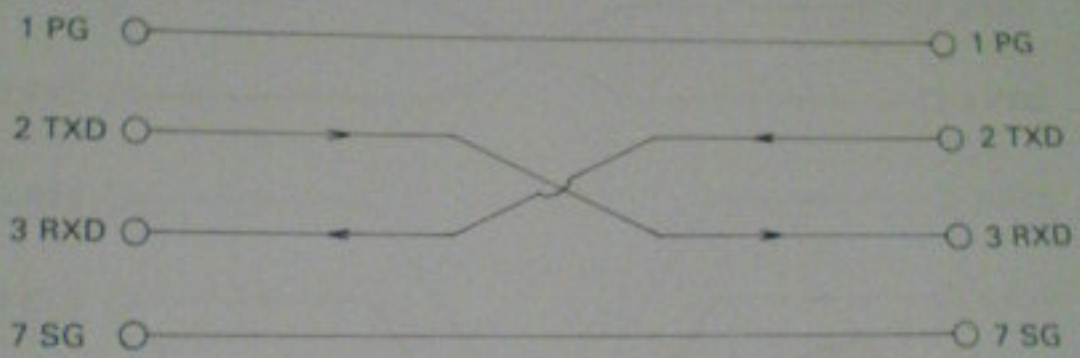
7.2 CONNECTION TYPES

The programmers support 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.

XON/XOFF (3 wire cable-form connection)

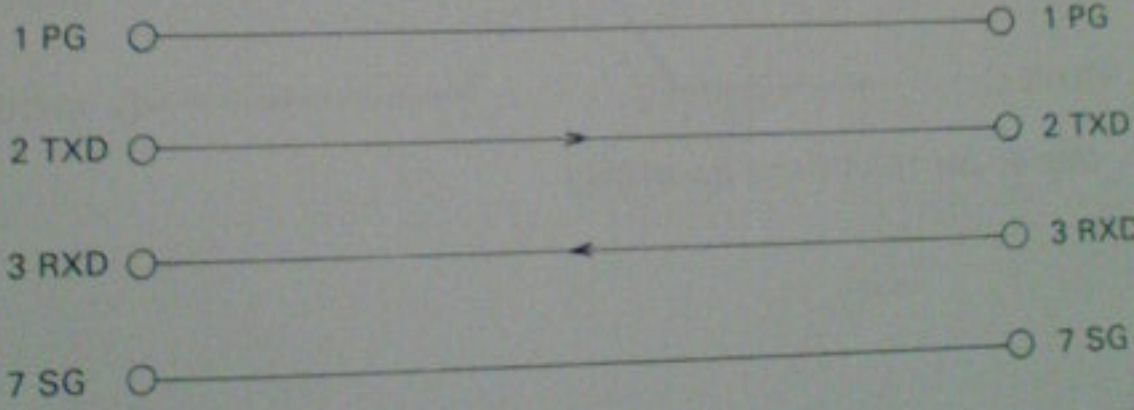
a. For connection of two alike machines a 'cross over' is required.

DCE to DCE and DTE to DTE:



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

DCE to DTE and DTE to DCE: Straight

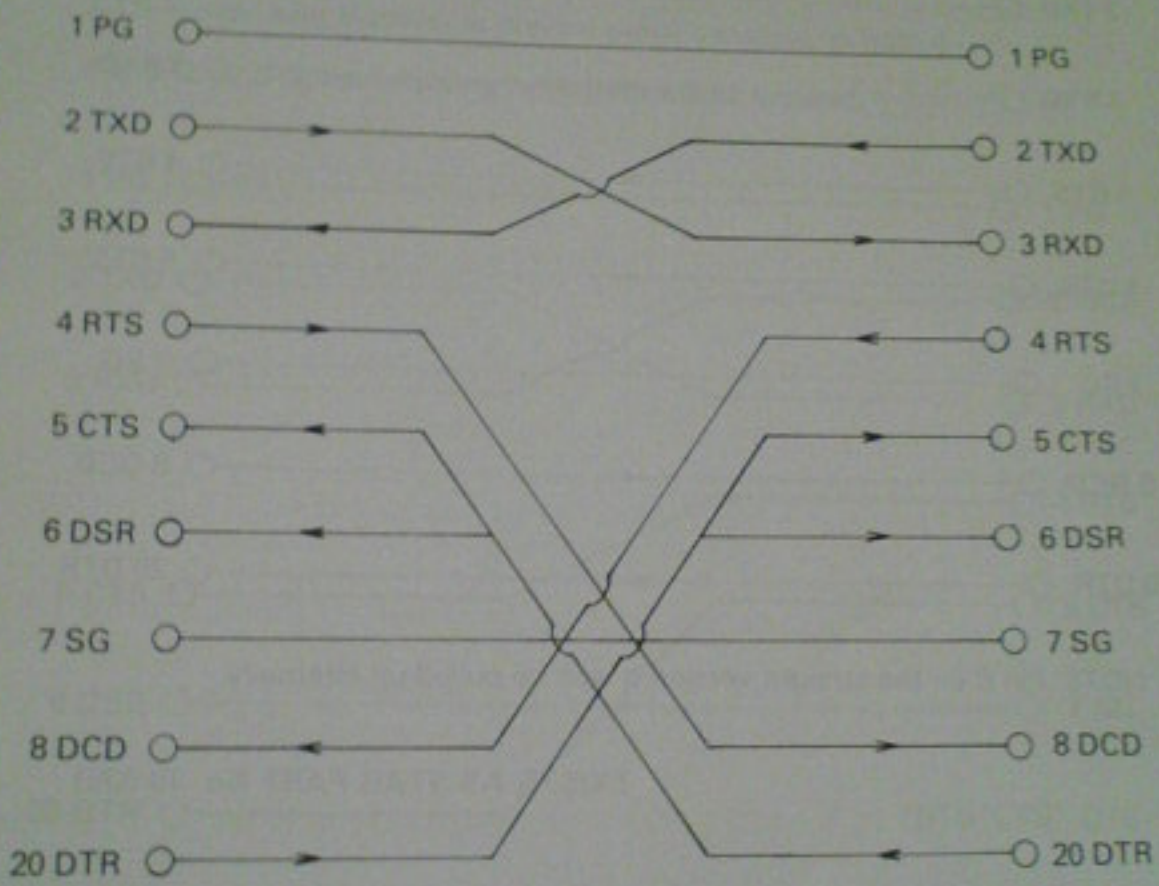


NOTE: Some machines do not have internal pull-ups and require extra connections within the cable form. Pull-ups may be required on pins 5, 6 and 8 of the external device if it is DTE or pins 4, 6 and 20 if it is DCE.

7.3 HARDWARE HANDSHAKE (7 OR 8 WIRE CABLE-FORM)

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

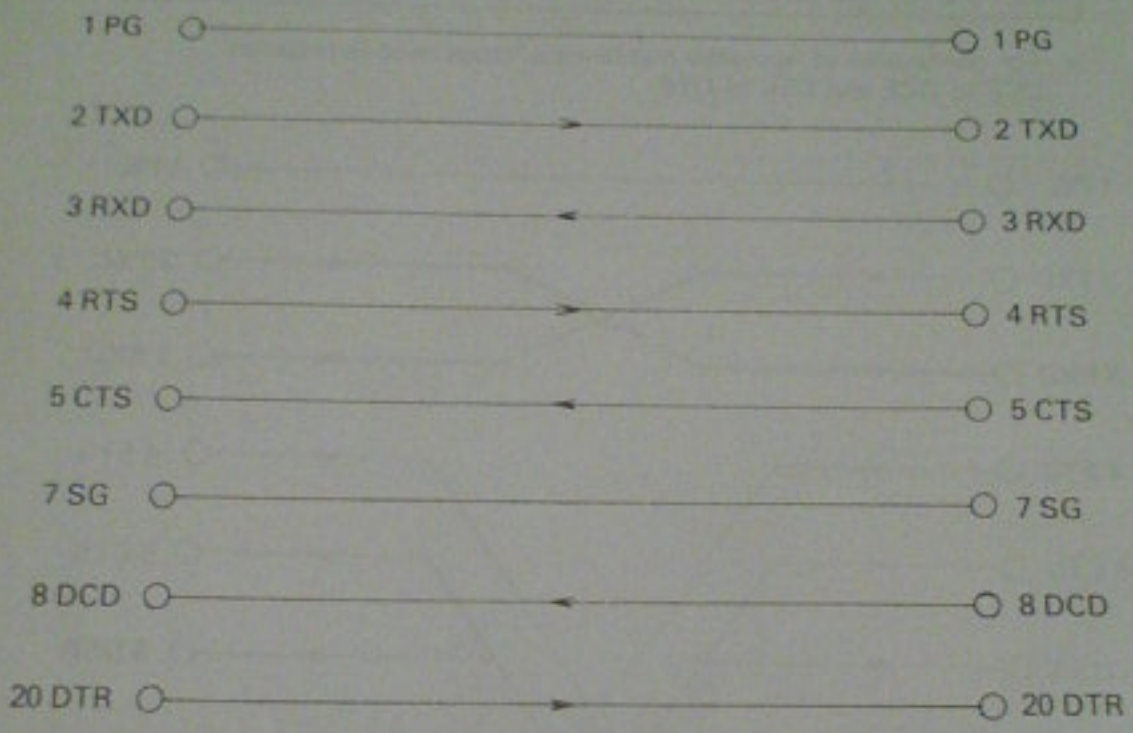
a. For connection of two alike machines a 'cross over' is required.
DCE to DCE and DTE to DTE:



EXISTS AS STAG PART No. 10-0250

b. For connection of two unalike 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.

EXISTS AS STAG PART No. 10-0251

7.4 NON-STANDARD CONNECTIONS

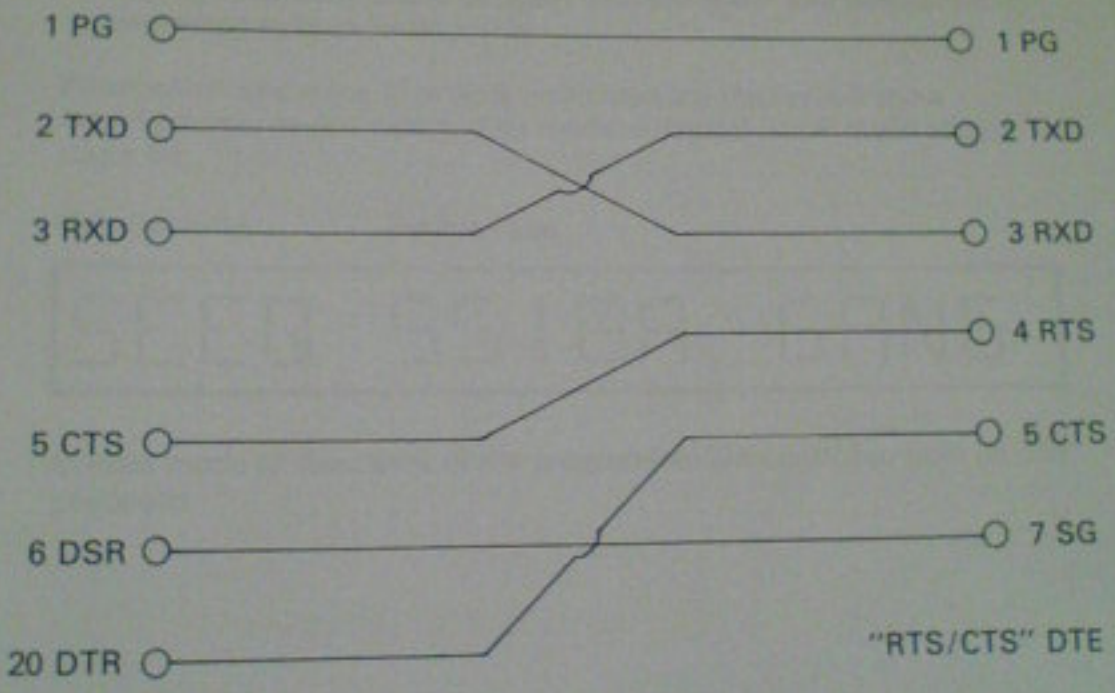
HARDWARE HANDSHAKE (5 WIRE CABLE-FORM)

This is a link-up to an unalike connector using "CTS-RTS" handshaking, it is made possible by the fabrication of a non standard cable-form.

RTS is a signal normally made active, but it remains inactive to inhibit data transmission to it, from external devices.

CTS is a pin kept inactive, to prevent signal transmission from it.

For DTE to DTE the following cable-form will be required.



8.1 SELECTION OF 'LOCAL' OR 'REMOTE' MODE

To select Local Mode

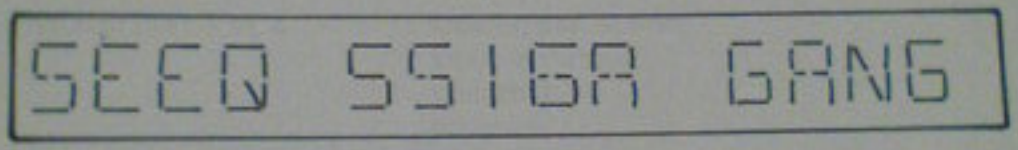
If the machine is in remote mode on power-up, one of two sequences can be performed:

(i) If the programmer is connected via the remote I/O port to a computer or terminal keyboard then the sequence of pressing Key 'Z' followed by the 'RETURN' key will bring control back to 'local' on the programmer's keyboard.

(ii) If the programmer is in stand-alone mode on power-up, but still under the 'remote' setting, the operator must power down, wait two seconds and then power up again with the 'EXIT' key simultaneously depressed to read 'local' mode.

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

Manufacturer Device Type

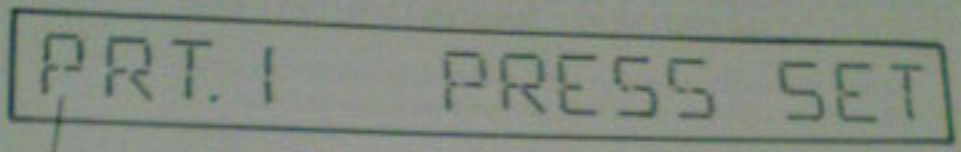


In local mode all functions of the programmer are controlled from its own keyboard.

8.2 REMOTE CONTROL

To select remote control Press Set 2

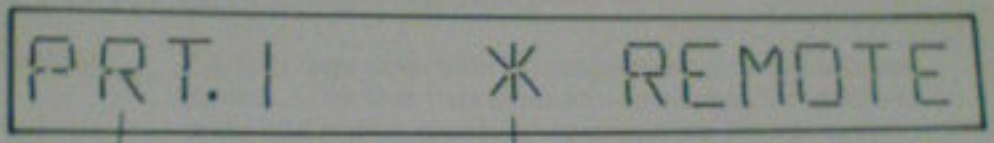
The display will show:



Defaults to last port selected

Press ↑ or ↓ to change to Port 2

By pressisng set again, the display will show:

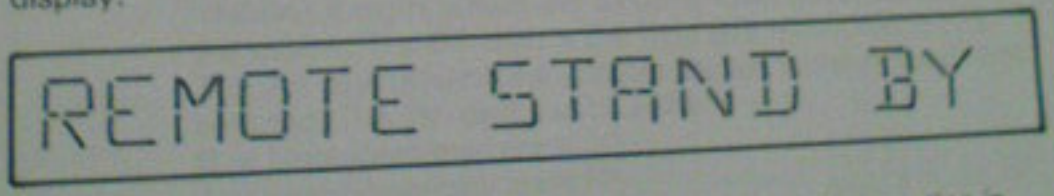


Remote port 1 or 2

Moving star shows input or output operation in progress

In remote mode the programmer is operated remotely from a computer or terminal. The programmer's keyboard is inoperative at this time.

Note: If Pass-through has been selected then pressing Set 2 gives the display:



A full description of Pass-through and its remote mode of operation is found in Section 9.

REMOTE CONTROL COMMANDS

h = one hex digit

RETURN	Executes a command for instance G RETURN, A6AF< RETURN,I RETURN, 11A RETURN etc.
G	Software revision number. This command issues a 4-digit hex number representing the software configuration in the programmer.
Z	Exits from remote control.
H	No operation. This is a null command and always returns a prompt character (>)

SET UP FOR DEVICE/RAM FUNCTIONS

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 code).
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: 00FFF/8/0>: The first 5 digits reflect the working RAM limit relevant to the device. The 6th digit is the byte size measured in bits. The 7th digit reflects the unprogrammed state of the device selected. The 7th digit can be either 1 or 0. 0 = Unprogrammed state 00. 1 = Unprogrammed state FF.

Set up for device/RAM functions (Continued)

Device/RAM address limits.

RAM low address

hhhhh < This defines the lower address limit in RAM.

Block Size

hhhhh; This defines the block size within the device or devices.

Device start address

hhhhh: This defines the start address for the device or devices.

h = one hex character. One to five characters may be specified when setting the address limits.

L	LOADS device data into RAM.
P	PROGRAMS RAM data into device.
V	VERIFIES device against RAM.
S	CHECKSUM causes programmer to calculate checksum of RAM data.

Note: on the PP42 a # command can precede the device address or checksum to indicate a particular socket. For example 6 # hhhhh: sets the lower address limit in the sixth device. If the command is omitted, the function defaults to socket 1 (leftmost socket).

By initiating the load, program, verify, bit check, empty check or checksum 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
50A	HEX-ASCII (Space)
51A	HEX-ASCII (Percent)
52A	HEX-ASCII (Apostrophe)
53A	HEX-ASCII (Comma)
59A	STAG HEX
82A	MOTOROLA S-RECORD
83A	INTELLEC 86
86A	TEK HEX
96A	EXTENDED TEK HEX
81A	MOS TECHNOLOGY

All covered by standard HEX-ASCII

Input/Output Address Limits

h hhhh< Sets a five digit figure defining the lower address limit.

h hhhh; Sets the number of bytes of data to be transferred, which in effect defines the upper address limit.

hhhh hhhhW Sets the offset required for data transference for both INPUT and OUTPUT.

I INPUTS data from the computer/terminal to the programmer's RAM

O OUTPUTS data from the programmer to the computer/terminal.

By initiating either the INPUT or the OUTPUT operation data transfer will commence, inclusive of any pre-selected parameters specified above.

To select Port 1 or Port 2 preceed I or O with a '1' or a '2'. If the port number is not specified the programmer defaults to the last selected port.

ERROR RESPONSES

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

PROGRAMMER RESPONSES

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

CR = Carriage return.

8.4 Remote Error Codes

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
48	Buffer Overflow	
82	Checksum	
84	Invalid Data	

REMOTE ERROR WORD

BIT NUMBER	
	RECEIVE ERRORS
31	If any error has occurred, this bit is set
30	Not used
29	Not used
28	Not used
27	Not used
26	Serial-overrun error (42)
25	Serial-framing error (41, 43)
24	Command-buffer overflow, i.e. > 16 characters (48)
	PROGRAMMING ERRORS
23	Any device-related error
22	Device appears faulty to the machines electronics (26)
21	Device start and Block size > Device max. address
20	Not used
19	Device not blank (20)
18	Illegal bit (21)
17	Non verify (23)
16	Incomplete programming or invalid device (22)
	I/O ERRORS
15	If any I/O error has occurred, this bit is set
14	Not used
13	Not used
12	Not used
11	Checksum error (82)
10	Not used
9	Address error, i.e.> word limit
8	Data not hexadecimal where expected (84)
	RAM ERRORS
7	RAM – hardware error
6	Not used
5	RAM start and Block size > RAM max. address
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

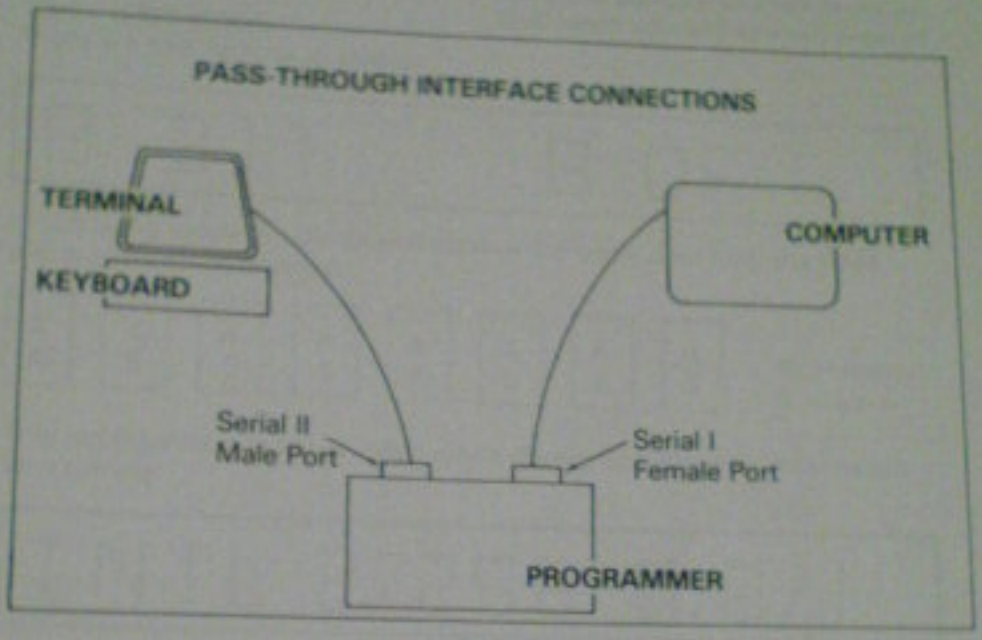
8 — RAM error

4 — Insufficient RAM resident

PASS-THROUGH

The PASS-THROUGH mode allows communication between a terminal, the programmer and a computer by utilizing only one connection on the terminal and the computer.

Note: In order for PASS-THROUGH to work the computer must be connected to the programmer via the RS232(1) female port (configured as DTE) and the terminal must be connected via the RS232(2) male port (configured as DCE).



Pass-Through mode is selected by pressing Set 1 followed by Key 3, ← and ↑ or ↓

There are two modes of operation for Pass-Through:
(i) Normal Mode
(ii) Remote Mode

9.1 (i) Normal Mode

The programmer appears to be transparent in data transfer between the terminal and the computer. Pass-Through continues until the input or output key is pressed on the programmer, causing it to display the message 'Stand By'. Nothing further is transmitted until either the programmer receives a 'Control O' sequence from the computer or its own 'Set' Key is depressed. The input or output function is then implemented.

When the input or output is finished or the exit key is used to abort the function, the programmer goes back to Pass-Through.

9.2 (iii) Remote Mode

Press Set 2
to enter Remote.
If Pass-Through (i.e. Transparent) has been selected, the programmer displays the message 'Remote Stand By' until it receives a 'Control O', Carriage Return' from the computer. On receipt of this key sequence, the programmer ceases to Pass-Through, and goes into Remote mode under computer control. Operating continues in Remote Control until the sequence 'Z, Carriage Return' from the computer, when the programmer returns to Remote Standby and starts to Pass-Through again.
To leave Remote Standby press 'Exit'.

BIT MODES AND SET PROGRAMMING

The PP42 can be configured to 8, 16 or 32 bit mode using Stag's "Interlace" concept. A wide range of set programming options are available, selected by pressing:

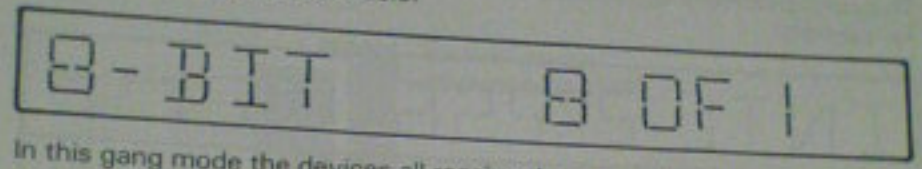
SET 3

The display will show the last-entered bit mode configuration. To change configuration press ↓ or ↑

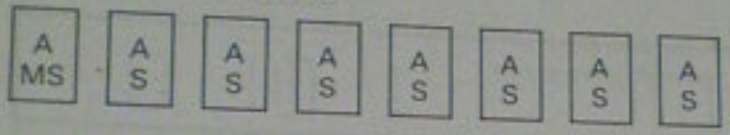
10.1 8-BIT MODE

There are four options available.

(i)

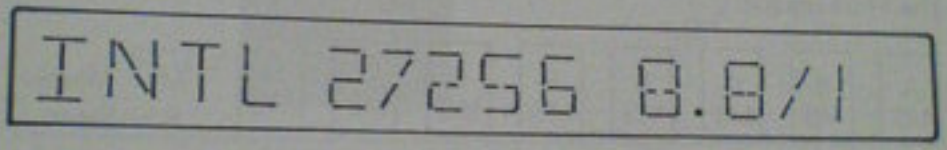


In this gang mode the devices all receive the same data from the RAM and are therefore identical.

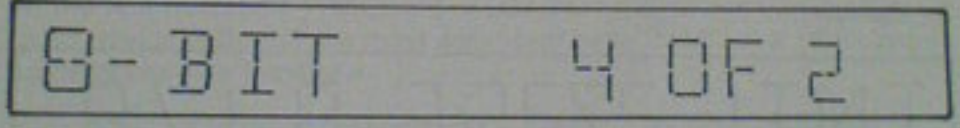


M = master,
S = slave

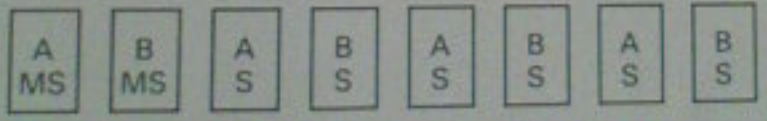
This configuration is abbreviated when displayed with other information, for example:



(ii)

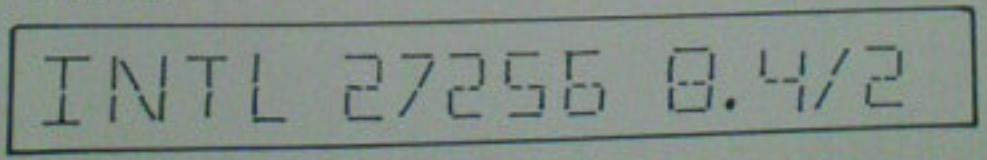


This mode consists of four identical sets of two devices. The two devices within the set differ from each other in the data which they receive from the RAM.



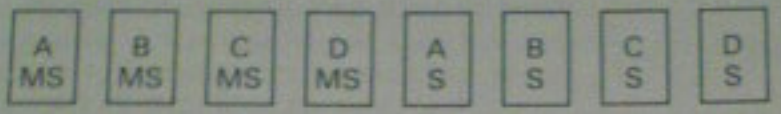
M = master,
S = slave

This configuration is abbreviated when displayed with other information, for example:



(iii) 8-BIT 2 OF 4

This mode consists of two identical sets of four devices. The four devices within the set differ from each other in the data which they receive from the RAM.



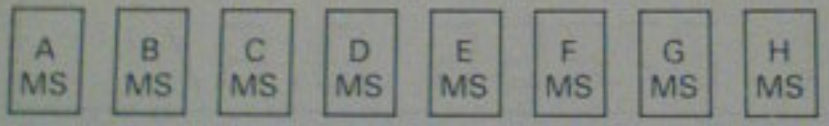
M = master,
S = slave

This configuration is abbreviated when displayed with other information, for example:

INTL 27256 8.2/4

(iv) 8-BIT 1 OF 8

In this mode the devices all receive different data from the RAM. Each is therefore unique.



M = Master
S = Slave

This configuration is abbreviated when displayed with other information, for example:

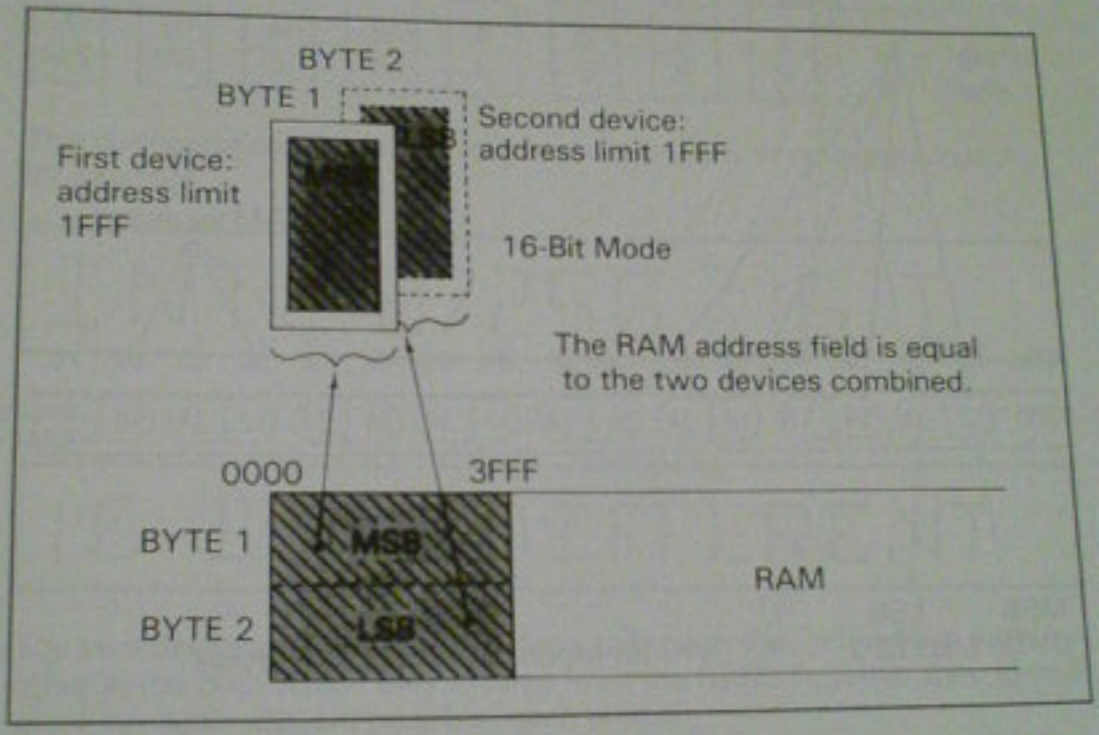
INTL 27256 8.1/8

10.2 16-BIT MODE

In this mode 16 data bits are split between two eight-bit devices.

A graphic example of how the 16-Bit Mode works on both LOAD and PROGRAM is shown below:

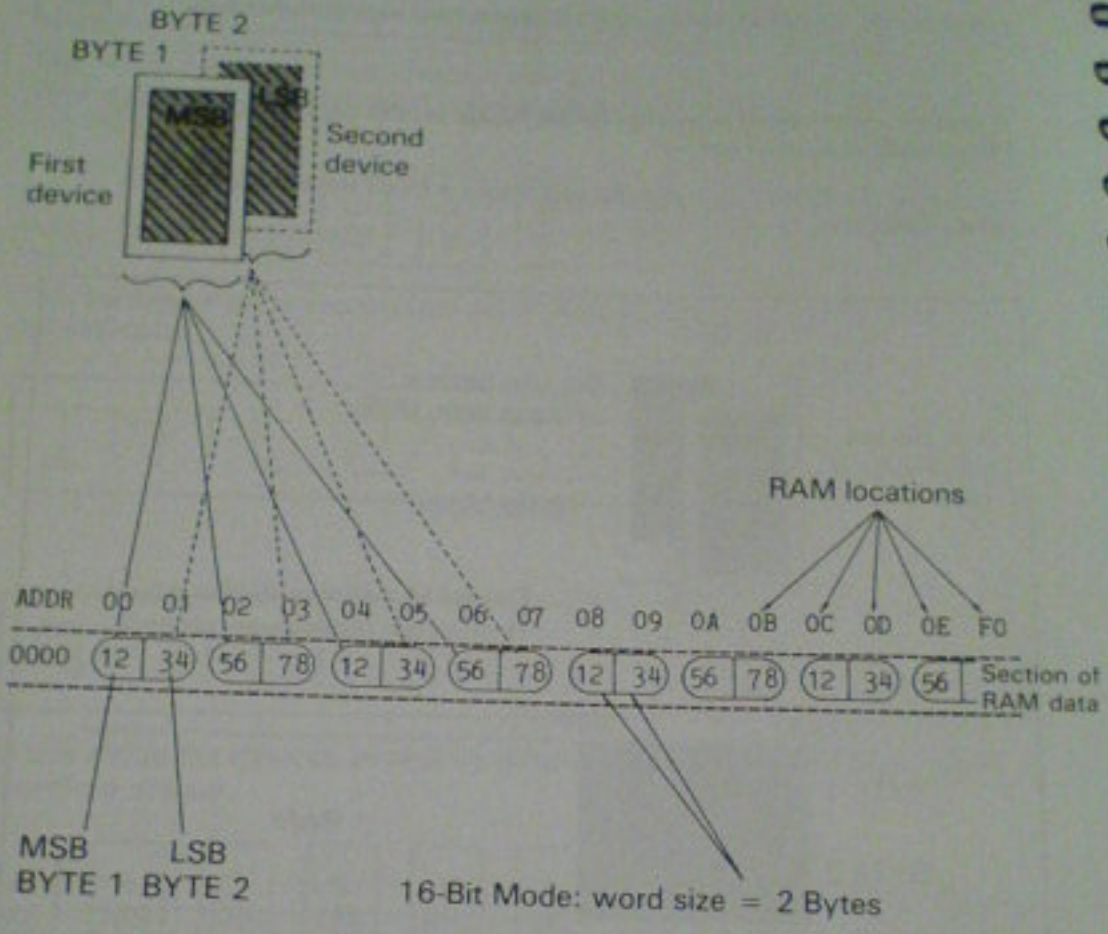
For clarity the following example will show a fixed device size: 1FFF (2764 EPROM)



In the first operation: PROGRAM or LOAD is made to and from a single 8-Bit device with only the most significant byte "MSB" of a 16-Bit Word undergoing data transfer.

In the second operation: PROGRAM or LOAD is made to and from a single 8-Bit device with only the least significant byte "LSB" of a 16-Bit word undergoing data transfer.

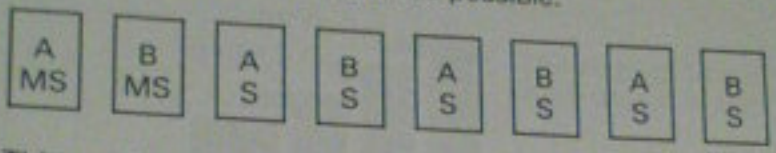
Two bytes of data make up one word in the 16-Bit Mode:



There are two options available:

(i) 16 BIT IDENTICAL

The two devices function interactively, although they differ from each other in the data which they receive from the RAM. Together they form a set. Four such identical sets are possible.



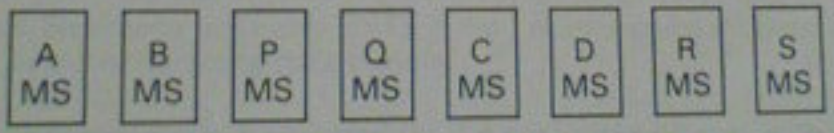
M = Master
S = Slave

This configuration is abbreviated when displayed with other information, for example:

INTL 27256 16.I

(ii) 16-BIT DIFFERENT

The two devices function interactively, although they differ from each other in the data which they receive from the RAM. Together they form a set. Four such sets are possible, each unique.



M = Master
S = Slave

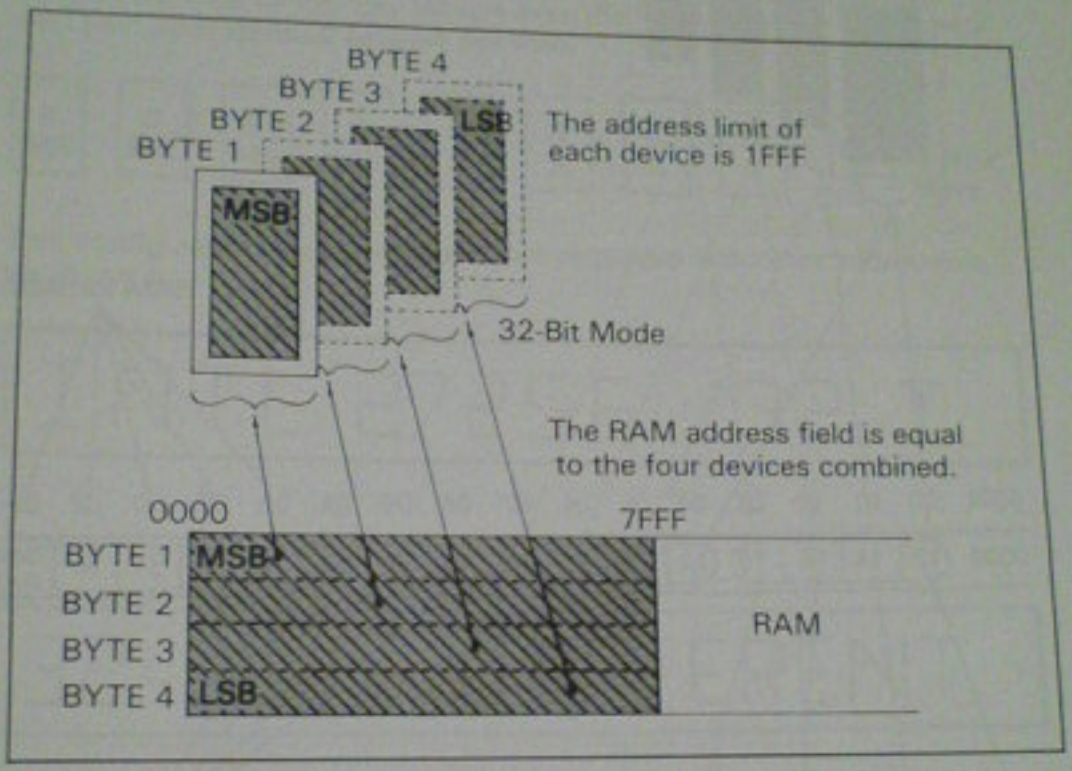
This configuration is abbreviated when displayed with other information, for example:

INTL 27256 16.D

10.3 32-BIT MODE

In this mode 32 bit data bits are split between four eight-bit devices.

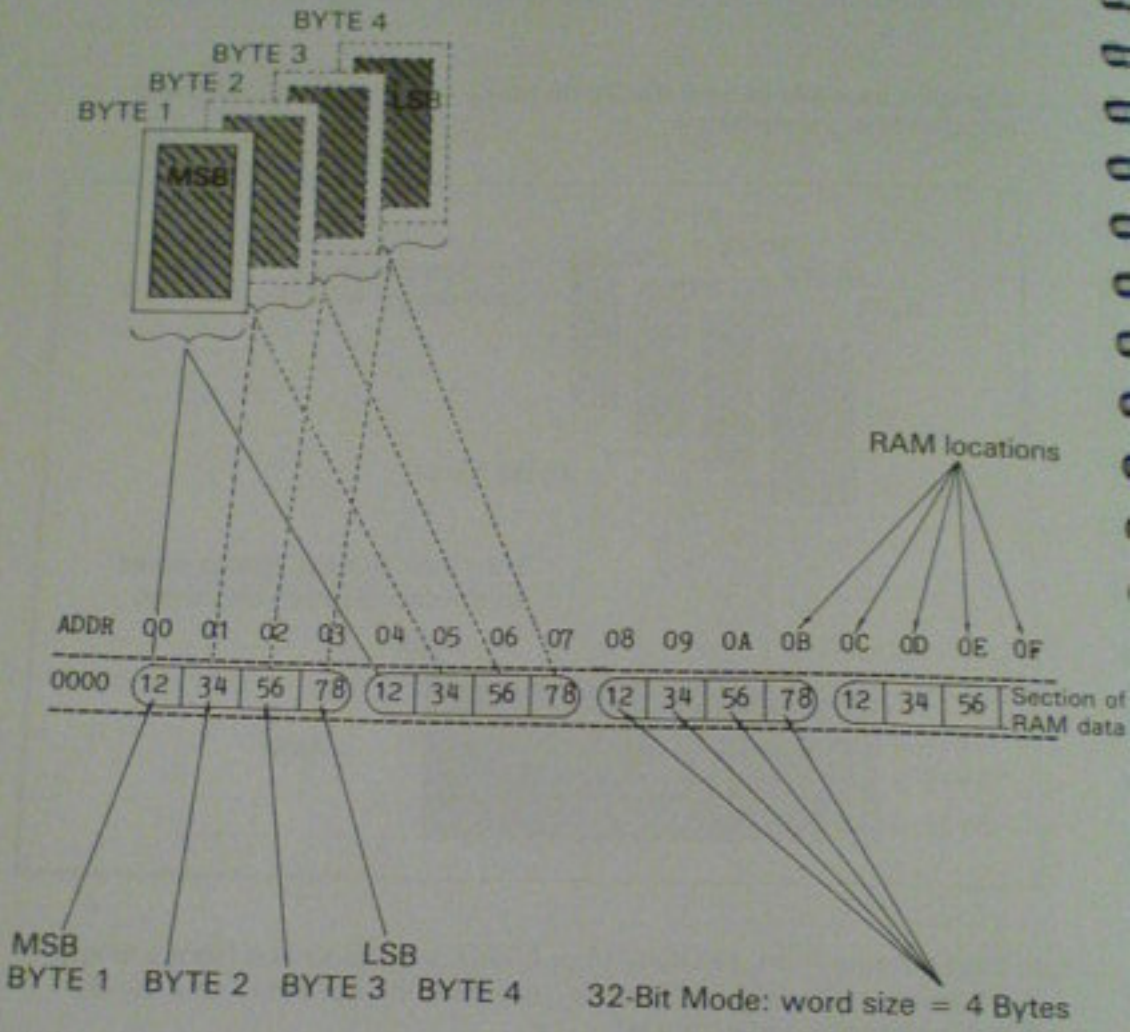
A graphic example of how the 32-Bit Mode works on both LOAD and PROGRAM is shown below:



In the first operation: PROGRAM or LOAD is made to and from a single 8-Bit device with only BYTE 1 the "most significant byte" of a 32-Bit Word undergoing data transfer.

This sequence is repeated for the next three devices with bytes 2, 3 and 4 respectively undergoing data transfer.

Four bytes of data make up one word in the 32-Bit Mode:

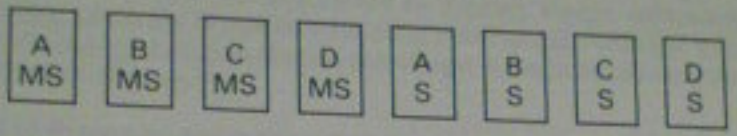


MSB – The "most significant BYTE" in binary code
LSB – The "least significant BYTE" in binary code

There are two options available:

(i) 32-BIT IDENTICAL

The four devices function interactively, although they differ from each other in the data which they receive from the RAM. Together they form a set. Two such identical sets are possible.

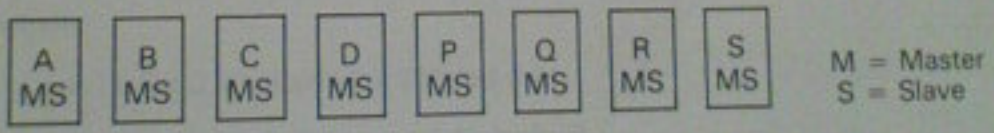


This configuration is abbreviated when displayed with other information, for example:

INTL 27256 32.I

(ii) 32-BIT DIFFERENT

The four devices function interactively, although they differ from each other in the data which they receive from the RAM. Together they form a set. Two such sets are possible, each unique.



This configuration is abbreviated when displayed with other information, for example:

INTL 27256 32.D

PP40 Series Addendum

Note: These additions will be incorporated into the PP40 series manual at the next revisional reprint.

Additions to Existing Manual (PP40, PP41 and PP42 Rev.1)

Additional 'Set' Commands - pages 1.5-01 and 1.5-03

Set Verify: This allows selection of 'Normal' verify and 'Margin' verify. Normal verification is carried out with Vcc at a steady 5V (typical). Margin verification is carried out with Vcc being taken 5% higher and 5% lower than normal. Press Set followed by Verify. Interchange between Normal and Margin is made by pressing the vertical cursor keys. Press Exit to confirm.

Set Program: Allows the selection of pre-program checks. The options are: illegal bit; empty and none. Press Set followed by Program. Interchange between the options is made by pressing the vertical cursor keys. Press Exit to confirm.

Software Data Protection for E^2 devices - 100/101 modules: Press Set F3, use left/right arrow keys to toggle the setting. Press Exit.

Remote Commands for Filling the PP41/PP42 RAM

The commands are: FF^ - to fill with FFs

00^ - to fill with 00s

Remote Commands for Setting the PP42 Set Configuration

The set configuration command takes the form:

ABCDD22]

where:

A - can be I or D for Identical or Different data.

B - is the number of devices per set (in decimal).

C - number of sets (in decimal).

DD - bit mode (expressed in hexadecimal e.g. 08,10 or 20).

22] - command

For example:

I160822] - 8-bit gang mode with 6 devices.

D221022] - 16-bit mode with two different sets of two devices each.

Erratum

Page 4.5-02 Substitute the word 'set' for the word 'exit' on line six (under second display diagram).

40M101, 41M101 and 42M101 Modules

Description

These three modules program MOS PROMs, EPROMs and EEPROMs in 24,28 and 32-pin DIL packages.

Operation

These modules function in a similar manner to the 40M100, 41M100 and 42M100 modules respectively as detailed in the manual apart from one function.

Set DE - to erase Seeq 48128 devices.

Press Set followed by D and E. If the wrong device type is selected, the message 'NOT APPLICABLE' will be returned. If the correct device type is selected, erasure will take place.

Warning: Extreme care should be taken to ensure that only Seeq 48128 devices are socketed when the Device Erase function is used. Failure to comply may result in damaged devices for which Stag and the semiconductor manufacturers can take no responsibility.

40M102 and 41M102 Modules

Description

These modules program 40-pin DIL EPROMs and EEPROMs from most major manufacturers.

Operation

These modules function in essentially the same way as the 40M100 and 41M100 as detailed in the manual apart from one major difference. The 'M100 modules handle 8-bit data but the 'M102 modules handle 16-bit data. In the case of the 41M102 where there is a RAM editor present, the data displayed in functions such as List, Edit, Delete etc. will be in the form of a double byte (4 hex. digits) for any given address.

'Byte Swap' for 16-bit devices

Select the 16-bit device, e.g. 27C1024

Press SET F6

Press the down arrow key 3 times

Use the right and left arrow keys to swap the Hi / Lo settings

Press EXIT

This will not affect the data in RAM, but will change the order in which data is presented to the device when programming.

40M103/41M103

These modules support MROM (masked ROM) pinout EPROMs.

Operationally these are very similar to the other 40 series modules, but see Byte Swap above.

41M200 - compatible with PP41 and PP42 mainframes.

Description

The 41M200 module will gang program 40-pin DIL, single chip microcomputers with data loaded into RAM from a master micro, a master EPROM, direct keyboard entry or via one of the dual RS232C ports.

Operation

The 41M200 operates in almost the same way as the 41M100 but for the following exceptions:

Loading of Data from a Master Device

Data can be loaded from a master micro or from a master EPROM. To select between the two master sockets, press Set followed by Load. This will display the default state 'MASTER MICRO', which indicates that the 40-pin micro socket is to be used. To interchange betwwen either of the sockets, use the vertical cursor keys. Press Exit to confirm. A green LED will illuminate adjacent to Pin 1 of the relevant socket. To load the data, insert the master device and press Load. Data will be loaded from addresses corresponding to the size of the selected micro to be programmed. The address limits can however be altered - see Section 3.9 of the manual.

Note: When loading data from a master EPROM, only 2764 and 27128 devices should be used. The master EPROM should be the only device socketed during loading.

Electronic Identifier

There is no Electronic Identifier function on the 41M200.

Margin Verify

There is no margin verify function on the 41M200

Security Bit Status and Encryption Table

Certain devices such as the 8751H have a security bit. If this bit (bit 1) is blown, the device will function but the data cannot be read and no further programming of the device can be carried out. For devices with two security bits such as the 87C51, operation is slightly different. Blowing bit 1 will allow the device data to be read but will inhibit further programming and blowing bit 2 will not allow the device data to be read.

Devices such as the 87C51 and the 8752BH support a data encryption facility. This enables data within the device to be 'exclusively NORed' with a 32 byte encryption table before being read. The encryption code is entered into the programmer's memory immediately after the data to be programmed.

For example:	8752BH
Device Address Lo:	0000h
Device Address Hi:	1FFFh
RAM Address Lo:	0000h
32 Byte Encryption Table:	2000h - 2031h (inclusive)

or:

Device Address Lo:	0000h
Device Address Hi:	0FFFh
RAM Address Lo:	0000h
32 Byte Encryption Table:	1000h - 1031 (inclusive).

To select the security bit and encryption status option, press Set F3. To interchange between bit 1, bit 2 or Encryption, press the vertical cursor keys. To interchange between 'blown' and 'intact', press the horizontal cursor keys. Press Exit to confirm.

The selected security bits will be blown after the device has verified following programming. The display will return the message 'VERIFIED/SECURED'.

Interface Formats (Introduction)

There are thirteen formats available on the PP41/42, 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
SBIN	=	STAG BINARY
MOST	=	MOS TECHNOLOGY

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.

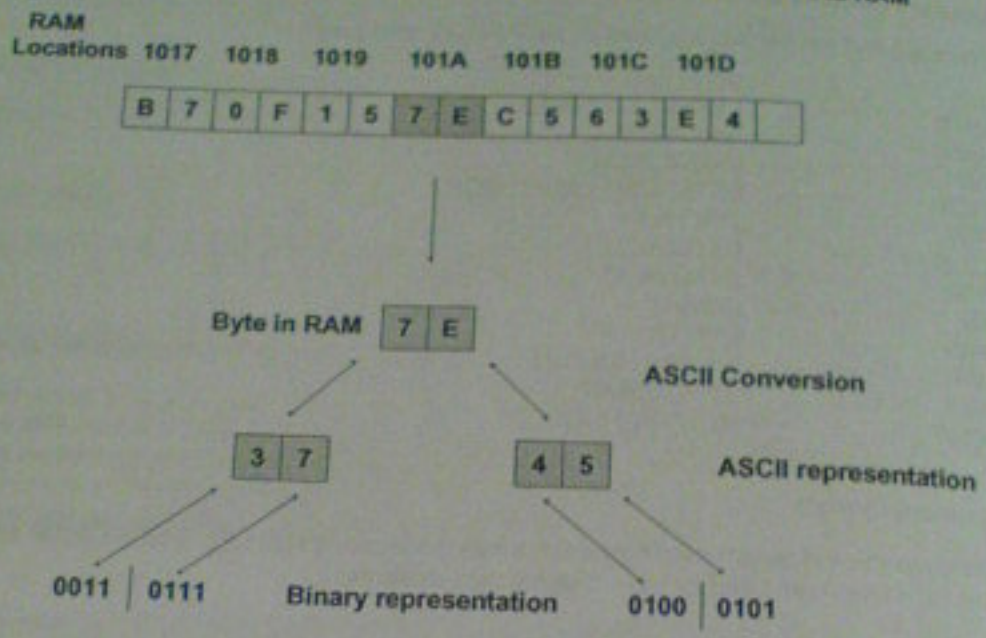
Binary Rubout

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 PP41/42 RAM



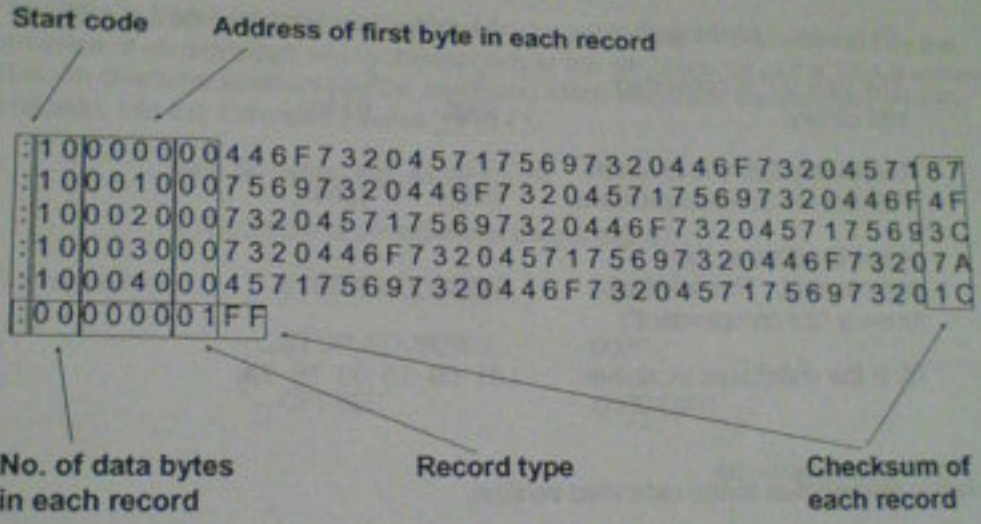
Intellec

The Intellec format when displayed consists of:

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

For example:

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



Calculation of the Intellec* Checksum

```
:100000000446F7320457175697320446F7320457187
: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 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.

Extended Intellec

The extended Intellec format when displayed consists of

a. A start code, i.e. (colon)

b. The sum of the number of bytes in a particular record, e.g. 10

c. The address of the first byte of data in a individual record, e.g. 0000

d. The record types, i.e.

00 - Data Record

01 - End Record

02 - 'Segment Base Address' record (SBA)*

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

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

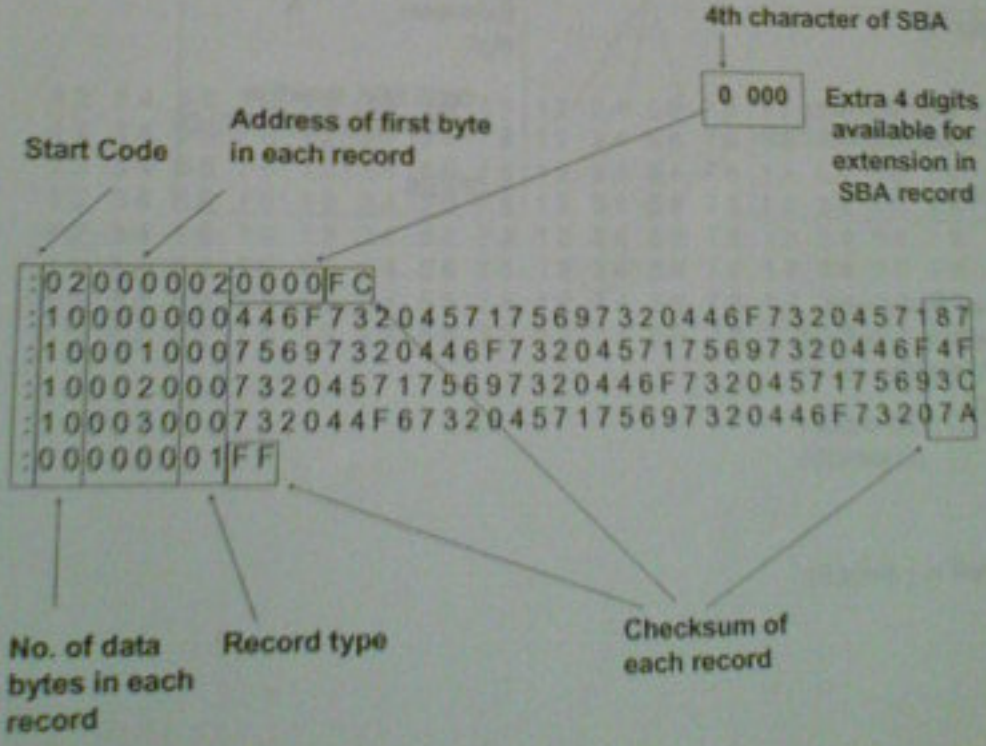
f. A checksum of an individual record e.g. 87

For example:

START ADDRESS: 0000

STOP ADDRESS: 003F

OFFSET: 0000 0000



SBA Repetition

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

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

For example:

START ADDRESS: FFA0

STOP ADDRESS: FFFF

OFFSET: 0000 0018

```
:0200180200000E4 ---------- SBA RECORD
A
:10FFB800FF00FF00FF00FF00FF00FF00FF00FF00FF0041
:10FFC800FF00FF00FF00FF00FF00FF00FF00FF00FF0031
:10FFD800FF00FF00FF00FF00FF00FF00FF00FF00FF0021
:10FFE800FF00FF00FF00FF00FF00FF00FF00FF00FF0011
:08FFF800FF00FF00FF00FF0005
:0200000210000EC ---------- NEW SBA RECORD
B
:0800000000FF00FF00FF00FF00E8
:0000800FF00FF00FF00FF00FF00FF00FF00FFFFFF1
:00FFB80148
```

Maximum address for four digits (FFFF)

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

Extension digit	B
1000 SBA Insertion	
+ 0000 ADDRESS FIELD	
= 10000	

Extension digit	A
0000 SBA Insertion	
+ FFB8 ADDRESS FIELD	
= 0FFB8	

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

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).

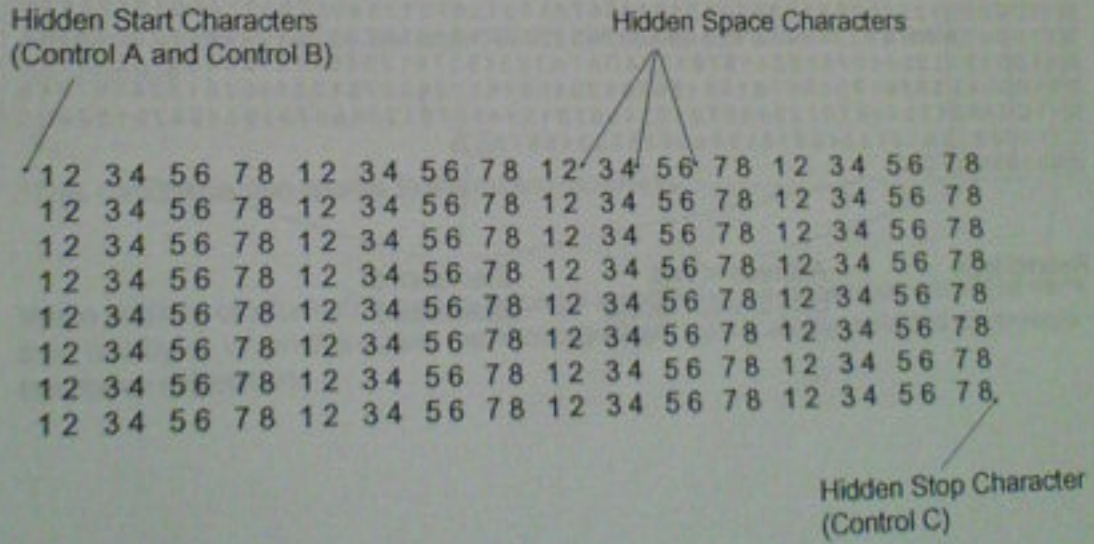
(ii) A hidden stop character known as Control C. (03: ASCII code, ETX: ASCII character).

(ii) 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



16 bytes per line on outp

Exorcisor

The Exorcisor format consists of:

a. A start code, i.e. S

b. The record types, i.e. 1 - Data Record
9 - End Record

c. The sum of the number of bytes in an individual record, e.g. 1D

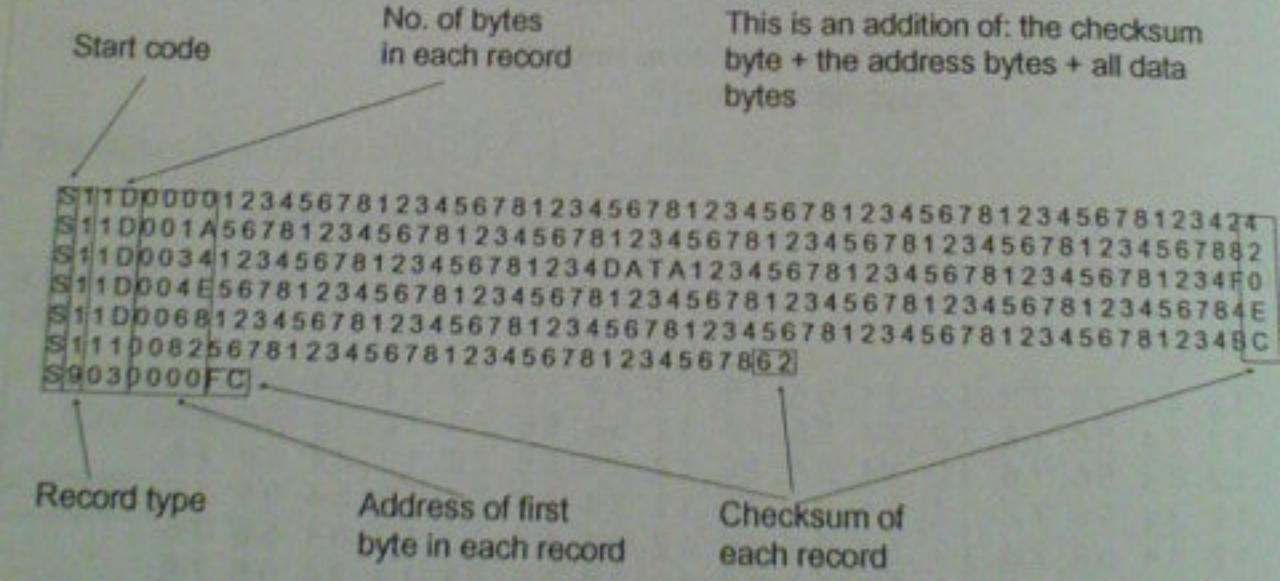
d. The address of the first byte of data in an individual record, e.g. 0000

e. Data in bytes, e.g. 12 34 56 78

f. Checksum of an individual record, e.g. A4

For example:

Start Address:	0000
Stop Address:	008F
Offset:	0000



Calculation of the Exorcisor* Checksum

```
S1100001234567812345678123456781234567812345678123456781234567812345678123424
S104001A568B
S90300000FC
```

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

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

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

*This calculation also applies to the extended version.

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

Extended Exorcisor

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

When this occurs the record type changes:

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

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

When this occurs:

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

The extended exorcisor when displayed consists of:

a. A start code. i.e. S

b. The record types, i.e.

1 - Data Record (Four Character address)
9 - End Record (Four character address)

2 - Data Record (Six character address)
8 - End Record (Six character address)

3 - Data Record (Eight character address)
7 - End Record (Eight character address)

c. The sum of the number of bytes in an individual record, e.g. 1D

d. The address of the first byte of data in an individual record, e.g. 0000, 010000, 01000000.

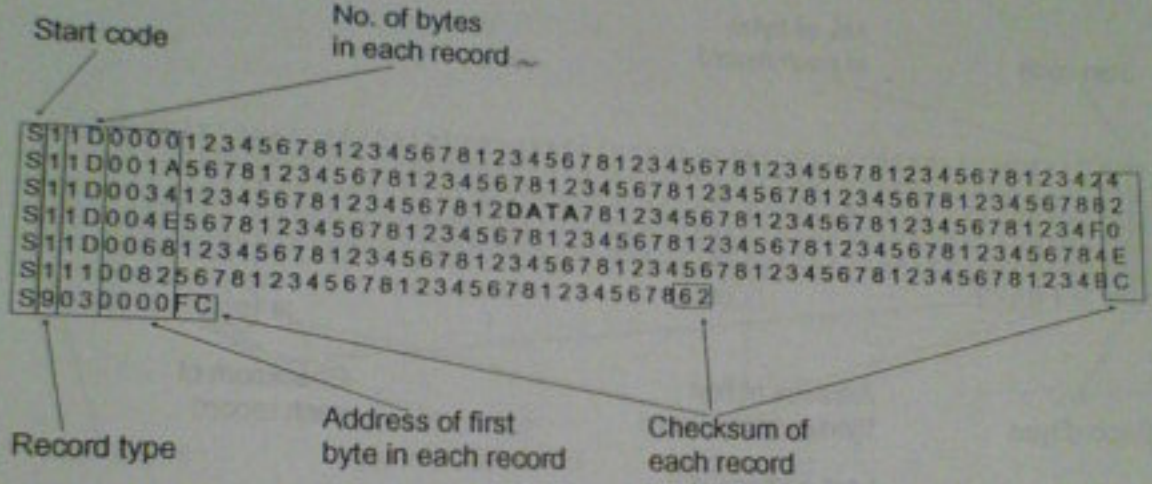
Data in bytes, e.g. 12 34 56 78

Checksum of an individual record: 24

1 - Data Record (Four Character Address)
9 - End Record (Four Character Address) } 2 bytes

For example:

start address: 0000
end address: OO8F
offset: 0000 0000

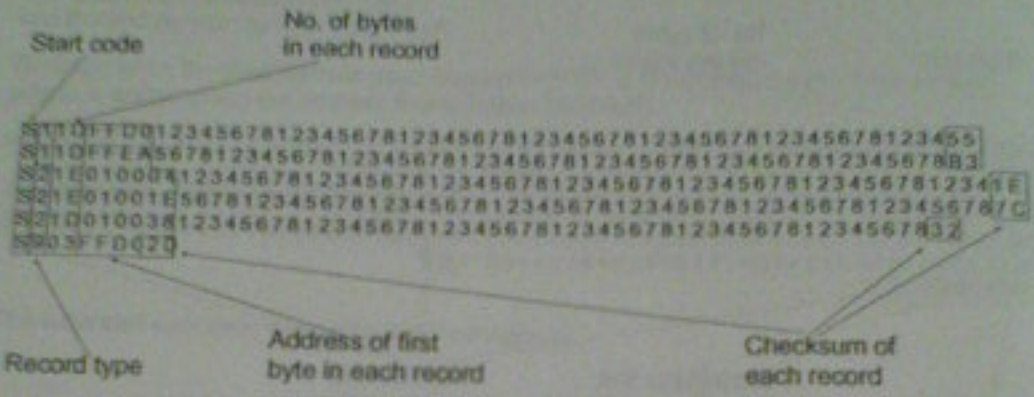


1 - Data Record (Four Character Address)
9 - End Record (Four Character Address) } 2 bytes

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

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

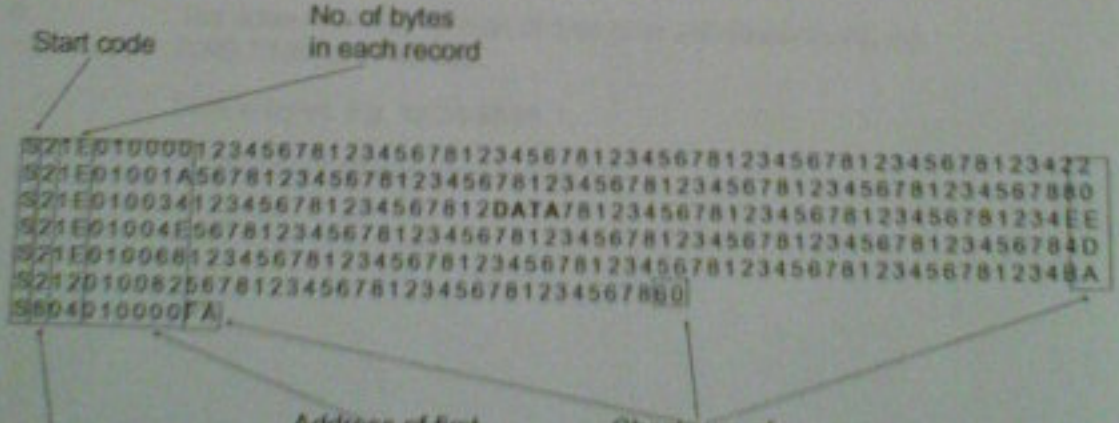
Start Address:	FF80
Stop Address:	FFFF
Offset:	00000050



2 - Data Record (Six character Address)
8 - End Record (Six Character Address) } 3 bytes

For example:

start address:	0000
stop address:	008F
offset:	00010000



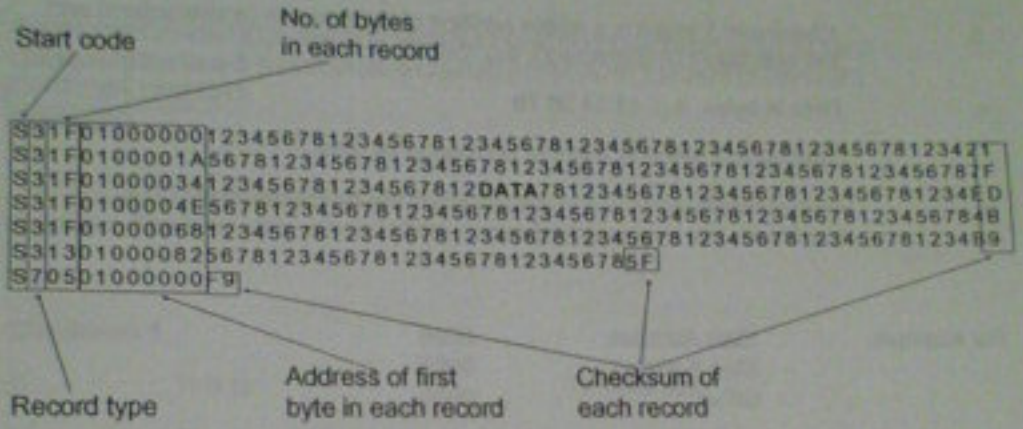
2 - Data Record (Six Character Address)
6 - End Record (Six Character Address) } 3 bytes

3 - Data Record (Eight Character Address)
7 - End Record (Eight Character Address)

1.4 bytes

For example:

Start Address: 0000
Stop Address: 008F
Offset: 01000000



3 - Data Record (Eight Character Address)
7 - End Record (Eight Character Address)

} 4 bytes

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. OB

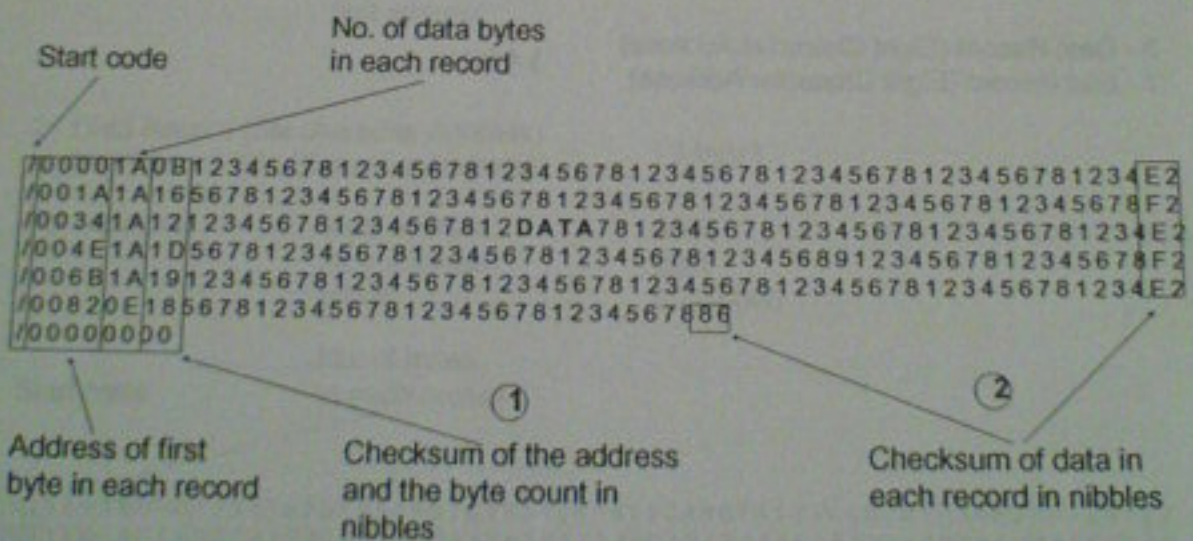
e. Data in bytes, e.g. 12 34 56 78

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 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.

/00001A0B12345678123456781234567812345678123456781234567812345678123456781234E2
Checksum1Checksum2 67812345678123456781234567812345678123456781234567812345678F2
/0034030A12345615
/00000000

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

Checksum 1

(i)	This is:	/10034030a
(ii)	The start code and the checksum are removed:	/0A
(iii)	Six nibbles remain:	0034003
(iv)	The 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.

Extended Tek Hex.

The Extended Tek Hex when displayed consists of:

a. A start code: % (percentage)

b. A count of the nibbles in an individual record, e.g. 3B

c. The record types, i.e.
6 - Data Record
8 - End Report

d. A checksum of the whole of an individual record excluding the %, e.g. F7

e. *The number of nibbles comprising - "the address of the first byte in each record", e.g. 1, 2, 3, etc.

f. 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

Start code

Record type

Number of nibbles in address

```
%3B6F7100123456781234567812345678123456781234567812345678123456781234
%3D6142100A5678123456781234567812345678123456781234567812345678123456
%3D6002341234567812345678123456781234567812345678123456781234567812345
%3D61E24E5678123456781234567812345678123456781234567812345678123456781
%3D6072681234567812345678123456781234567812345678123456781234567812345
%3D613282567812345678123456781234DATA12345678123456781234567812345678
%3D60E29C1234567812345678123456781234567812345678123456781234567812345
%3D61A2B65678123456781234567812345678123456781234567812345678123456781
%3D6062D01234567812345678123456781234567812345678123456781234567812345
%3D6212EA5678123456781234567812345678123456781234567812345678123456781
%3D6003104123456781234567812345678123456781234567812345678123456781234
%3D61B311E5678123456781234567812345678123456781234567812345678123456781
%1866C3138123456781234567812
%07B101D
```

1 Character
2 Characters
3 Characters

Number of nibbles in each record

Checksum of each record

The address of the first byte in each record

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

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

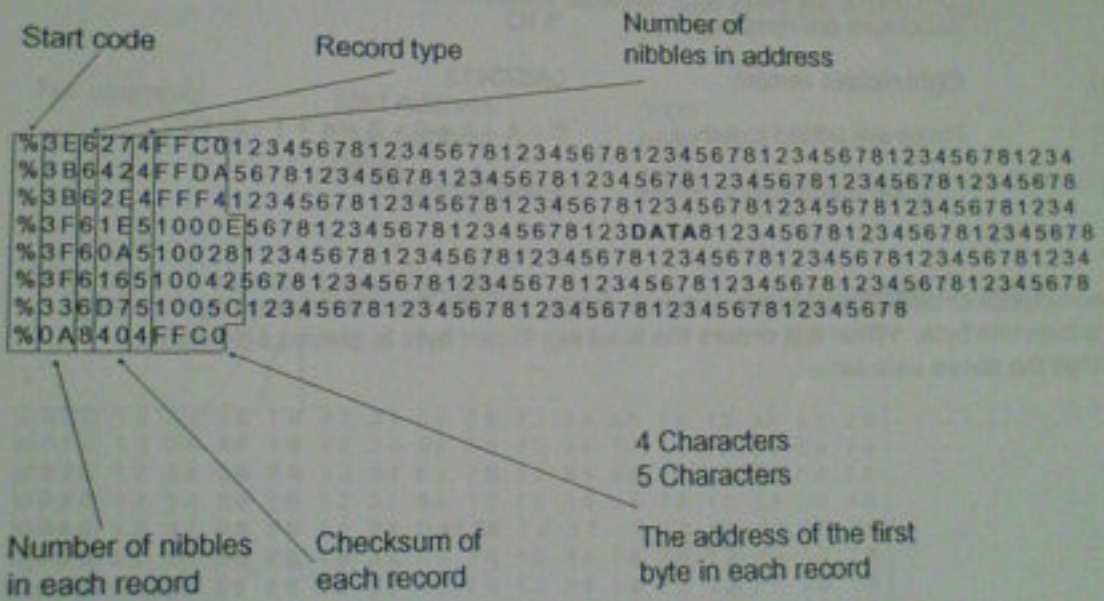
2/1A	6/1000000	A/1B4625DC95
2 Characters	6 Characters	A Characters (10 in Decimal)

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

Extended Tek. Hex. with an Offset, Displaying Transition from 4 Character Address Field to 5 Character Address Field.

For example:

Start Address:	0000
Stop Address:	00AF
Offset:	0000 FFC0



Calculation of the Extended Tek Hex. Checksum

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

```
%386F710123456781234567812345678123456781234567812345678123456781234
%3C61421A5678123456781234567812345678123456781234567812345678
%0A61C23412
%07881010
```

Example: The third line of the above format

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

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

PPX (or STAG HEX.*)

The PPX format when displayed consists of:

a. A start code, i.e. * (an asterisk, 2A-ASCII Code)

b. The address of the first byte of data in an individual record, e.g. 0000

c. Data in bytes, e.g. 12 34 56 78

d. A stop code, i.e. $ (a dollar sign, 24-ASCII Code)

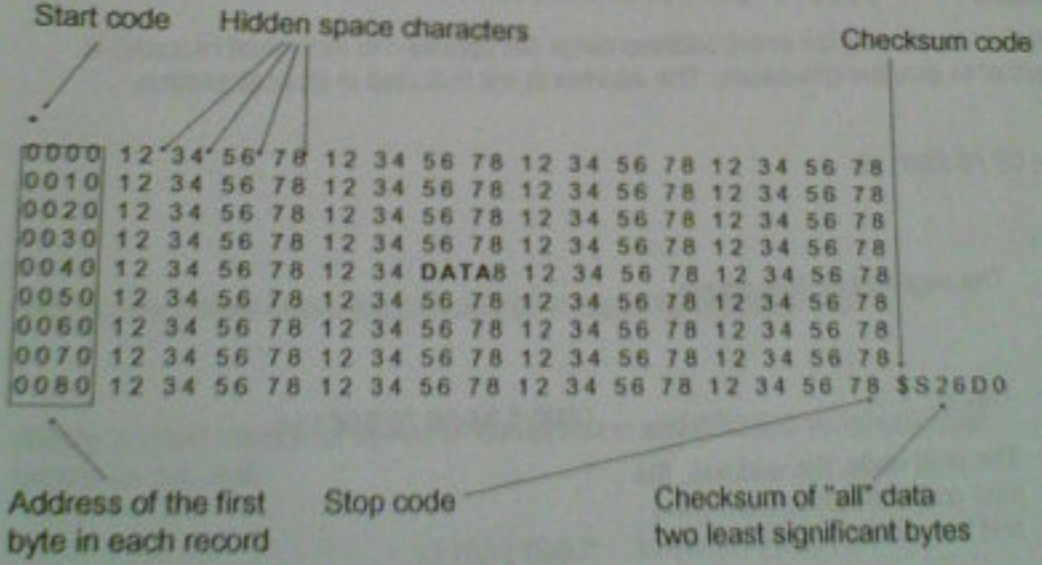
e. A checksum of all data over the entire address range. (The displayed checksum is the two least significant bytes.)

f. A checksum start code: S

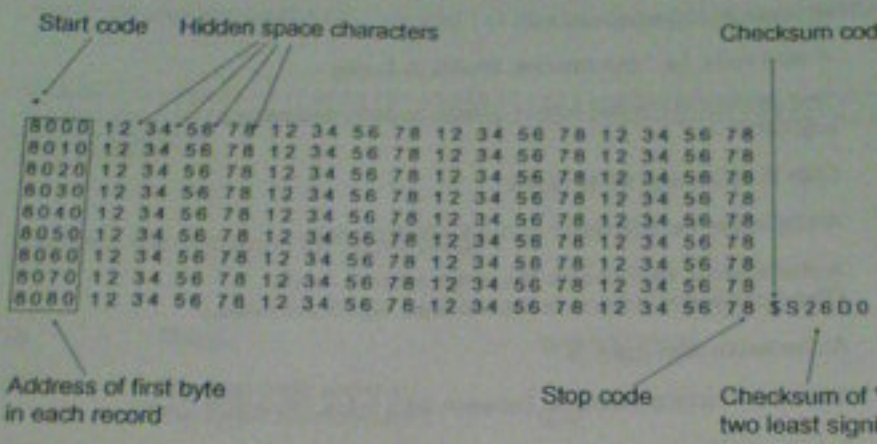
g. An invisible space character between data bytes (20-ASCII Code)

For example:

Start Address:	0000
Stop Address:	008F
Offset:	0000



And with an Offset of 8000



Calculation of the PPX Checksum

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

*

0000 12 34 56 78 $S0114

Example: The segment of data above

(i)	This is: *	0000 1 34 56 78 $S0114
(ii)	The start code, the address, the stop code, the checksum code and the checksum are removed:	*0000 $S0114
(iii)	Four bytes remain:	12 34 56 78
(iv)	These are added together:	12+34+56+78=114
(v)	114 is the checksum which is displayed in two byte form as above:	*0000 12 34 56 78 $S0114

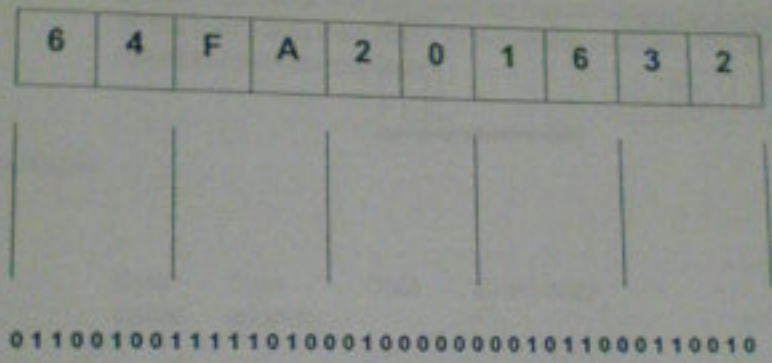
As the PPX checksum is an addition of all data the total will invariably constitute more than two bytes. When this occurs the least significant 'two' bytes are always selected to undergo the above calculation.

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.



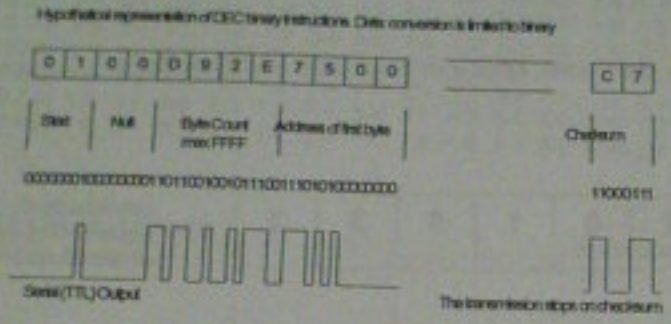
The Binary format operation can only be stopped by pressing 'EXIT'.

Binary is used mainly for speed of transmission and RS232C communication problems, i.e. test.

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:

64FA201632 Hypothetical Representation of Data Bytes

Start

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

7F64FA201632 Hypothetical Representation of Data Bytes

7F the Rubout character

MOS TECHNOLOGY

The MOS-TECHNOLOGY format consists of:

(i) A start code, i.e.: ; (semi-colon)

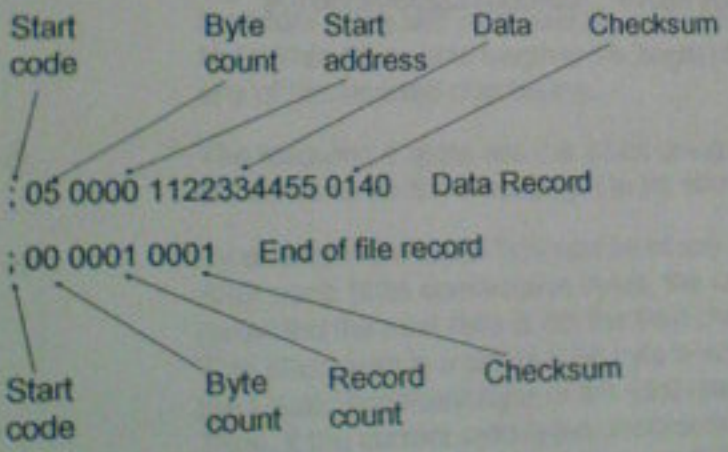
(ii) A byte count-that is the sum of the number of data bytes in an individual record, e.g.: 05

(iii) The address of the first byte of data in an individual record, e.g.: 0000

(iv) Data in bytes, e.g.: 11 22 33 44 55. (The data bytes must consist of valid hexadecimal digits).

(v) A checksum which is displayed as two hexadecimal bytes. It is the addition of the preceding data bytes in the record including the address and byte count in hexadecimal form.

For example:



Calculation of MOS-Technology checksum

```
; 05 0000 11 22 33 44 55  0104
; 00 0001 0001
```

Example: the first line of the above format

(i) This is: ; 05 0000 11 22 33 44 55 0104

(ii) The start code and the checksum are removed: ; 0104

(iii) This leaves:

the byte count:	05
the address of the first byte in the record:	0000
five data bytes:	11 22 33 44 55

(iv) These are added together: 05+0000+11+22+33+44+55=0104

(v) 0104 is the checksum as above: 05000011223344550104

Stag Binary

Stag Binary is for the rapid transfer of large amounts of data with error detection. It allows a textual header for identification purposes only. The format allows multiple blocks of data up to FFFFFFFFh with any offset up to FFFFFFFFh, and these blocks may be in any order.

The file is terminated with a NULL block with zero data length.

Each individual block has a terminating checksum and for blocks greater than 1024 bytes, an additional embedded checksum is provided for rapid error detection.

The Stag Binary format is as follows:

a) The file can have any amount of header information (for file identification, etc) as long as it does not contain any **'binary 1 bytes' (00000001 binary / 01 hex.)**

b) Block start. There can be any number of blocks terminated by a null block. The file format proper starts with the first **'binary 1 byte'**. The block checksum is calculated from the next byte.

c) The next 4 bytes are the 32-bit unsigned length of all the data for this block, where the first byte is the Most Significant Byte and the last byte is the Least Significant Byte. If all 4 bytes are zero (00000000) then this is a null block and therefore is the last block and the end of the transmission.
Note: This 4 byte data length is the length of the data only and does not include any of the inserted checksums.

d) The following 4 bytes are the 32-bit unsigned offset of the data. The format is the same as for the data length in (c) above.

e) Next follows the data. This can be of any length from 0 to FFFFFFFFh.
After each 1024 consecutive bytes, the current negated checksum is inserted, (provided the next byte is not the final checksum anyway).
This checksum is a single 8-bit byte which is the negation (2's complement) of the Least Significant Byte of the total checksum so far.
Thus, if the current calculated checksum is added to this inserted negated checksum, the result should be zero. If it is not, the recipient has the option to abort reception or at least to warn of the error. (In a practical system, if an error is detected, it may be prudent to keep loading to prevent the sender from 'locking-up'. Without this feature, a very long file could be corrupted in the first second, and not noticed until the end of transmission.

f) The last byte in the block is the 8-bit negated checksum of all bytes immediately after (but not including) the start byte and not including the inserted checksums. The checksum therefore includes: the 4 byte data length; the four byte offset and the data only. If this is added to the checksum calculated by the receiver, the result should be zero.
This is the end of one block. If it is not a null block, the cycle continues with the next block.

Interblock data is allowed if required providing it does not contain any **'binary 1 bytes' (00000001 binary / 01 hex.)** Interblock data is not included in the checksum.

A sample short file of only one data byte would look as follows:

(Hex.)	Interpretation	
46	F	(ASCII)
69	i	(ASCII)
6C	l	(ASCII)
65	e	(ASCII)
20		(ASCII space character)
31	1	(ASCII)
01	start byte of one block (00000001 binary)	
	(Checksum starts from next byte)	
00	first byte of data length (Most Significant Byte)	
00	second byte of data length	
00	third byte of data length	
01	last byte of data length (LSB) (indicates 1 byte of data in this example)	
00	first byte of offset (MSB)	
00	second byte of offset	
00	third byte of offset	
00	last byte of offset (LSB) (indicates no offset in this example)	
02	the data byte (Checksum stops with previous byte)	
FD	Negation of checksum (This is the end of one block)	

There could be some interblock data here. It must not contain any binary 1 bytes (00000001 binary / 01 hex.)

01	start byte of block
00	data format as before. Checksum starts here.
00	
00	
00	4 zeros means this is the terminator (null) block
00	Since this is the last block, the offset is irrelevant but must be read
00	and used to update the checksum to ensure that this is a genuine
00	null block and not a corrupted data block
00	
00	negation of checksum and file end.

Note: A **FREE** set of converter programs is available on IBM PC/AT disk by applying to Stag either in the U.K. or the U.S.A.

A request may be made to the same addresses for the 'C' language source code.


STAG