GP Industrial Electronics

XP640 EPROM Programmer

User Manual

Contents

Introduction

XP640 EPROM Programmer

The XP640 EPROM Programmer is designed to keep you ahead in the fast moving world of programmable device technology. It combines both a reliable EPROM duplicator, RAM editor, video display and comprehensive input/output to make it one of the most sophisticated machines available anywhere.
The RAM editor can be "locked out" at any time to make the XP640 a very easy to use EPROM workstation. This allows the machine to be used by unskilled personnel. for low volume production runs.
The XP640 works equally well in either true stand-alone mode or connected to your computer or development system. Once connected, data can be transferred between the two machines and the programmer can be remotely controlled to make it an integral part of your workstation.

XU620 Universal Programming Module

The XP640 EPROM Programmer is expandable with the XU620 Universal Programming module to support:

- Bipolar PROM from all major manufacturers.
- Single Chip EPROM Microcomputers.
- Programmable Array Logic (PALs).

XM512 EPROM Emulation Module

The emulation option provides up to 64k x 8 of emulation memory. Two modules can be connected for 16-bit emulation. Data can be written from the target side to allow its use with microprocessor emulators.

1. General Operating Instructions

Supply voltage

Machines supplied in the UK and Europe are set to operate at 240v, 50Hz supply. A mains cable is supplied with the machine. The cores of the cable are colour coded as follows:

Live: Brown
Neutral: Blue
Earth: Green/Yellow.

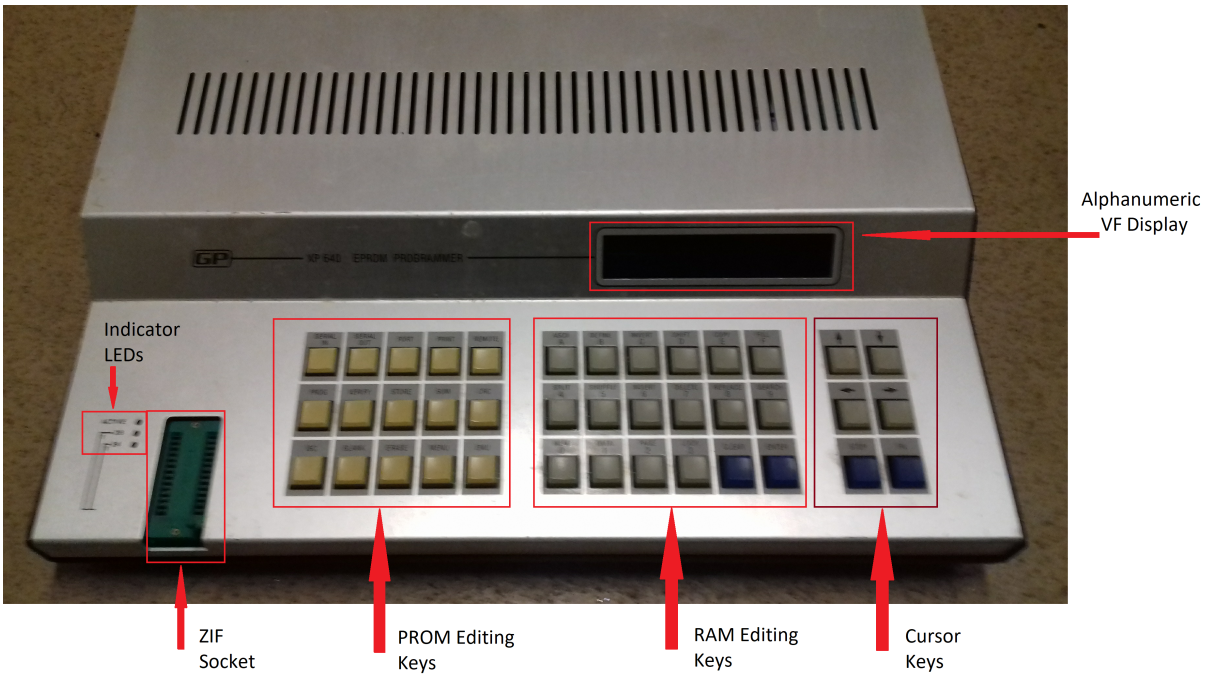
The mains cable plugs into the XP640 via the fused connector located on the right-hand side, rear of the unit. The unit is protected by a 500mA Anti-surge fuse located in the mains connector. Ensure mains voltage is disconnected before attempting to replace the fuse.

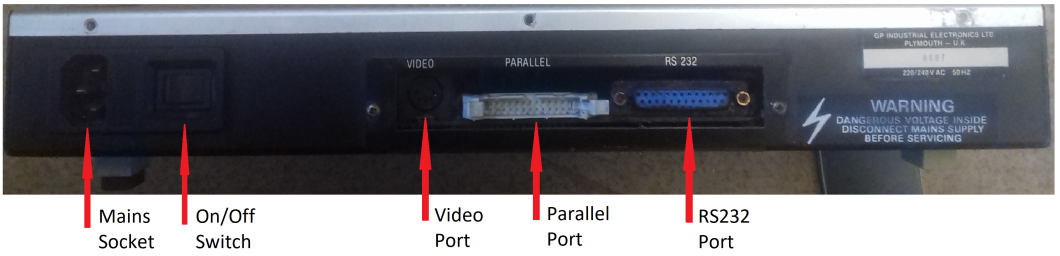
Using the Machine – Points to note

To ensure trouble free operation, please observe the following points:

- Operate the machine on a vibration free surface.
- Do not locate the machine near any source of heat or in direct sunlight.
- Ensure no metal parts can fall into the machine.
- Disconnect from the mains supply when not in use.
- **DO NOT** switch the machine on or off with EPROM devices in the ZIF socket.
- Check the device type setting when inserting an EPROM into the ZIF socket.
- Periodically clean the ZIF socket with a stiff bristle brush to ensure good contact.
- Never force an EPROM into or out of the ZIF socket – It is a zero insertion force socket.

Layout of the XP640





The Keypad

The keypad is divided into three separate sections.

- The right hand section is used for cursor and keyboard control.
- The centre section for the Hex editor.
- The left hand section is subdivided into input/output and PROM function keys.

The key functions are described in detail in the later sections of the manual.

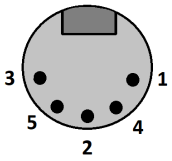
16 Character Alphanumeric Display

This is the on-board display and allows the XP640 to be used without a video monitor. It is used to display keyboard commands, messages, address and data information. The display usually shows the cursor address (current address of interest) and RAM and PROM data at that address, as illustrated below :

0000	FF	32	READY
Cursor Address	RAM Data	PROM Data	Message

Video Display

A composite video output is provided at the rear of the XP640.



1 Video out
2 Signal ground
3 Video out
4 +5v - do not connect
5 Lightpen - do not connect

Video Display Format

The video display is divided into four sections:

- A status section showing selected device type and input/output parameters.
- Data entry line – similar to the on-board line display.
- Address and data display showing the cursor address and PROM and RAM data at that address and the ASCII equivalent of the RAM byte.
- A hex dump of 255 bytes with on-screen cursor.

Discrete LED Indicators

The programming socket has 3 LEDs associated with it. The **active** LED indicates when power is applied to the ZIF socket. EPROMs should not be inserted or removed when the LED is on (the socket can be powered down by pressing the STOP key). The other two LEDs indicate the location of pin 1 for the selected device, depending on whether it has 24- or 28-pins.

Firmware Version number

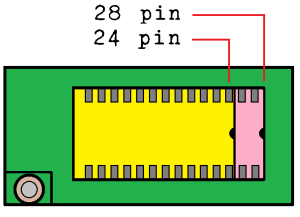
When power is first applied to the XP640 an 'INITIALISE BUSY' message is displayed whilst it performs a system self check. When complete the firmware version number is displayed in the message 'XP640 V X.Y READY' where X.Y is the version number.

Zero Insertion Force Socket

The socket is a zero insertion force type and will give reliable service provided it is kept clean and used in the proper way. The diagram below shows the correct way to load a PROM into the socket. The ZIF is designed to accommodate both 24- and 28-pin PROMs.

Lever open, insert or remove PROM

Lever closed, PROM is firmly held



2. Hex Editor

XP640 RAM Editing Functions

This section gives a detailed description of the XP640 editing facilities taken one key at a time. Examples are given on the use of each key by itself, and in conjunction with other keys. The table below gives a list of the available RAM editing facilities.

KEY	DESCRIPTION
STOP	Power down ZIF socket, return to normal mode.
HEX	Hexadecimal data keys
FN	Function key to activate editing keys
CURSOR	Move cursor up, down, left & right
ENTER	Load hex entry from display buffer
CLEAR	Clear last hex entry
MEM	Move cursor to memory address
DATA	Change hex data
PAGE	Select a 255 byte page
ASCII	Display ASCII dump on-screen
DEFINE	Define a RAM block for editing & PROM functions
INVERT	Invert data in RAM block
SHIFT	Shift data with cursor keys or to any address
COPY	Copy source block to destination
FILL	Fill block with a data value
SPLIT	16 bit to 8 bit split
SHUFFLE	8 bit to 16 bit shuffle
INSERT	Insert data at address
DELETE	Delete data at address
REPLACE	Change data strings to new strings
SEARCH	Find occurrence of data string
LOCK	Lock or unlock RAM editor

Note:

In the examples which follow, 'DISPLAY' means the on-board fluorescent display. The video display gives similar messages, but in expanded form.

STOP

This will stop any function and return the machine to normal mode, ready to accept new keyboard commands. After STOP, the ZIF socket is powered down and any previously defined block is now undefined.

HEX KEYS (0123456789ABCDEF)

These lower case keys are only enabled when the XP640 requires entry of hexadecimal data, otherwise they are not directly accessible.

FN

This key is used to enable any editing key and must be used prior to any RAM editing function. Its use prevents unintentional or accidental use of the editor.

Example: Select page 34 for display

KEYPRESS	DISPLAY	MEANING
FN	FN	RAM editor is enabled
PAGE	PAGE_	prompt for page number
34	PAGE 34	enter the page number
ENTER	1400 FF -	cursor address is 3400, RAM data FF, no PROM data

CURSOR

These are the arrow keys and can be used at any time to move the cursor up, down, left or right. Pressing the key once will move the cursor one position. Holding the cursor key down will continuously move the cursor as required.

Example: Move cursor right, left, down, up

KEYPRESS	DISPLAY	MEANING
Right Arrow	0001 FF FF	Increment cursor address by one
Left Arrow	0010 FF FF	Decrement cursor address by one
Down Arrow	0011 FF FF	Increment cursor address by 16 (1 screen line)
Up Arrow	0000 FF FF	Decrement cursor address by 16 (1 screen line)

Note:

- The cursor address is shown followed by RAM data and PROM data, both are hex FF in this example.
- The video always shows a dump of RAM data. PROM data at the corresponding RAM cursor address is also shown. If the PROM data is shown as '–' then the cursor is outside the range of the selected EPROM (i.e. no PROM data is available).

ENTER

This is used during the course of hexadecimal data entry. E.g. address and data information, Fill parameter, lock code. It is also an implied 'YES' key to reply to questions asked by the XP640. The XP640 will only act on the data entry once the ENTER key has been pressed.

CLEAR

This can be used to clear a hex entry E.g. if a mistake has been made. it is also used as an implied 'NO' key for use in response to questions asked by the XP640.

Example: Move cursor address to 013F and correct the mistake.

KEYPRESS	DISPLAY	MEANING
(FN) MEM	ADDRESS_	Prompt for memory address
013c	ADDRESS 013C	Enter address, but last digit is incorrect
CLEAR	ADDRESS 013_	Last entry cleared
F		OK, now enter the correct digit
ENTER	013F DE FF	Cursor address is 013F, RAM data DE, PROM data FF

MEM (Memory address)

This moves the cursor to any RAM address within the 64k x 8 user RAM. The base address of the RAM is 0000 and corresponds to PROM (ZIF) address 0000. The last address of the RAM is FFFF. The last address of the PROM depends on the size of the device selected.

Example: Move cursor to address FFFF.

KEYPRESS	DISPLAY	MEANING
(FN) MEM	ADDRESS_	Prompt for new cursor address
FFFF	ADDRESS FFFF	Enter the address
ENTER	FFFF FF –	Cursor address is FFFF, RAM data FF, but no PROM data because the selected device is smaller than the RAM

Note:

- Blanks are shown in the PROM data field if the cursor address is outside the range of the PROM.
- The cursor can also be moved with the cursor keys or the page select key.
- If no hex address entry is made & ENTER is pressed, the XP640 will substitute 0000 for the required address as shown below.

KEYPRESS	DISPLAY	MEANING
(FN) MEM	ADDRESS_	Prompt for new cursor address
ENTER	0000 FF FF	No address entered, so XP640 substitutes 0000 at new cursor address. RAM and PROM data is FF

DATA

This command allows keyboard entry of hex data at the cursor address.

Example: Change data at address 8000, 8001, 8002 to 01, 23, 45

KEYPRESS	DISPLAY	MEANING
(FN) MEM	ADDRESS_	Prompt for new cursor address
8000	ADDRESS 8000	Enter the address
ENTER	8000 FF – READY	cursor address is 8000, RAM data is FF
(FN) DATA	8000 FF_	prompt for data entry
01	8000 FF 01	enter the data
ENTER	8001 FF – _	cursor moved to next address, enter data
23	8001 FF 23	keep entering data, cursor will increment after each entry
ENTER	8002 FF – _	
45	8002 FF 45	cursor address is 8000, RAM data is FF
ENTER	8003 FF – _	
STOP	8003 FF – READY	data entry terminated

If no hex entry is made & the ENTER key is pressed, the XP640 will substitute 00 for the data as shown below.

KEYPRESS	DISPLAY	MEANING
(FN) DATA	8003 FF_	prompt for data entry
ENTER	8004 FF – _	no data entry made so XP640 enters 00 and increments cursor
Left Arrow	8003 00 – _	review data entry – change data entry is required
STOP	8003 00 – READY	data entry terminated

Note:

- The cursor keys can be used during data entry to move to a new address.
- Data entry mode is terminated with the STOP key.

PAGE

This is almost identical to the MEM key, but positions the cursor at the start of a 256 byte page. The cursor is placed at the top left of the video.

Example: Select page 83 (put cursor at address 8300)

KEYPRESS	DISPLAY	MEANING
(FN) PAGE	PAGE_	Prompt for page number
83	PAGE 83	Enter page
ENTER	8300 49 – READY	Cursor moved to address 8300, RAM data is 49, No PROM data available (Blanks in the PROM data field)

ASCII

Provides an ASCII dump of the on-screen hex dump. The cursor position is shown by a corresponding cursor (inverted video) in the ASCII dump. This function is only usable with a video monitor connected to the XP640.

DEFINE

This is the powerful block define function for use with many of the PROM functions and editing keys. It defines the start and end address of a RAM block.

There are two different ways to define a block: using the hex keys or using the cursor keys.

Using the Hex keys.

Example: Define the block 0000 – 1FFF using the hex keys

KEYPRESS	DISPLAY	MEANING
(FN) DEFINE	DEFINE_	Prompt for start address of RAM block
0000	BLOCK 0000_	Enter the start address
ENTER	BLOCK 0000-	Prompt for end address
1FFF	BLOCK 0000-1FFF_	Enter the end address
ENTER	BLOCK 0000-1FFF	Block is now defined
(FN)	FN 0000-1FFF	Pressing FN before a command shows block limits.
STOP	875D 43 – READY	Terminate edit command

Note:

- If a block is defined then the (FN) key will display the block limits before an editing command. (This is a reminder that those editing functions that can act on a block will do so).
- The cursor can be moved through a defined block, the READY message being replaced by DEF-D as a reminder that the block is defined.
- If a block has been defined, the PROM function keys will prompt for a ROM start address, and the function will act on the block.
- When STOP is pressed, the block is undefined, but the previously entered limits are still available and can be recalled by the key sequence (FN) DEFINE ENTER (i.e. define a block without manually entering limits will define the block using the last entered limits).
- A block need not be defined to use the block editing functions (INVERT, SHIFT, COPY, FILL) as these will prompt for start and end addresses if there is no previously defined block but a PROM function will not prompt for block addresses since their block limits are taken to be the PROM start and end addresses (see PROM function).
- A block remains defined until STOP is pressed.
- The cursor is not part of the block unless it is used to define the block, as shown in the following example.

The previous example is typical for defining large data blocks for use with the PROM functions e.g. block program, copy a PROM block to RAM etc.

Using the cursor.

The following example shows how the cursor can be used to define blocks.

Example: Define the block 1FFF–2000 using the cursor

KEYPRESS	DISPLAY	MEANING
(FN) MEM	ADDRESS_	Prompt for new cursor address
1FFF	ADDRESS 1FFF	
ENTER	1FFF FF FF READY	Put cursor at 1FFF
(FN) DEFINE	DEFINE_	Prompt for block start
RIGHT ARROW	2000 FF FF 1FFF	Move cursor, XP640 fixes address 1FFF as start of the block
ENTER	BLOCK 1FFF 2000	The block has been defined
STOP	875D 43 – READY	Terminate edit command

Note:

- The cursor can be moved in any direction to define a block.

INVERT

Inverts data in a RAM block. This is useful for micro-systems which have inverting buffers on the data bus.

Example: Invert the data in the block 0000 0011

KEYPRESS	DISPLAY	MEANING
(FN) INVERT	DEFINE_	Prompt for start address of block to be inverted
0000	BLOCK 0000	Enter the start address
ENTER	BLOCK 0000-_	Prompt for end address
0011	BLOCK 0000-0011	
ENTER	BLOCK 0000-0011	Block defined
	INVERTING	Busy inverting
	DONE	Function complete

Note:

- In the example, the block was defined as part of the INVERT function, however if the block had previously been defined (using DEFINE), then no prompts would have appeared for the block limits.
- The block remains defined until STOP is pressed.

SHIFT

This function shifts a defined block through memory using the cursor keys or direct to the cursor address. Data is shifted without overwriting or loss of data. Data in front of the block is transferred to the other side as the block moves through the RAM.

Example: Shift the block 0000 0001 to address F000.

KEYPRESS	DISPLAY	MEANING
(FN) MEM	ADDRESS_	Prompt for new cursor address
F000	ADDRESS F000	
ENTER	F000 C3 READY	Put the cursor at F000
(FN) SHIFT	DEFINE_	Prompt for block start
0000	BLOCK 0000	
ENTER	BLOCK 0000_	Prompt for block end
0001	BLOCK 0000-0001	
ENTER	BLOCK 0000-0001	Define the block.
	SHIFT TO F000	Data can be shifted to cursor position (see note)
ENTER		
	BUSY F000 D9 –DONE	Shift complete

Note:

- When the message 'SHIFT TO' is displayed, a hex entry can be made as the address to where the block is to be shifted, pressing ENTER (as in the example), the cursor is used as the shift address.
- For small shift movements the cursor keys can be used to move the block when the 'SHIFT TO' message is displayed.
- The block could have been defined using the define function.
- When shift is complete, the block remains defined until the STOP key is depressed.

COPY

This command will copy blocks of data within the RAM. When a copy has been completed, the source data has not been changed, but has been duplicated at the destination address. The copy command is 'intelligent' in that if the destination block overlaps the source block, then a complete copy is made at the destination, the source overlap obviously having been overwritten. The data block can be defined as part of the COPY command. or using the DEFINE function.

Example: Copy the block from 0000-0800 to the area starting at address 1000.

KEYPRESS	DISPLAY	MEANING
(FN) COPY	DEFINE_	Prompt for source block start address
0000	BL0CK 0000	
ENTER	BLOCK 0000_	Prompt for block end
0800	BLOCK 0000-0800	
ENTER	COPY TO F002_	Prompt for destination address or option to use the cursor address (F002)
1000	COPY TO 1000	But enter address 1000 as required
ENTER	BUSY	
	1000 F4 1A DONE	Block has been copied, cursor is at 1000, RAM and PROM data are different

Note:

- In this example the cursor was at address F002 and would have been used as the destination address if ENTER had been pressed when the 'COPY TO F002' prompt had appeared.

FILL

Memory fill is used to fill all or part of the RAM with a specific value.

Example: Fill the RAM block 0123-0234 with 0A

KEYPRESS	DISPLAY	MEANING
(FN) FILL	DEFINE_	Prompt for block start
0123	BL0CK 0123	
ENTER	BLOCK 0123_	Prompt for block end
0234	BLOCK 0123-0234_	
ENTER	BLOCK 0123-0234	define the block
	FILL WITH -	prompt for fill parameter
0A	FILL WITH OA	
ENTER	BUSY	
	0123 0A DONE	Block is filled with 0A , cursor is at the start of block

Note:

- The block could have been defined using the DEFINE function.
- The filled block remains defined until the STOP key is pressed.

SPLIT

This divides the RAM block as specified by the device type selection into two. All data at even addresses is stored in the lower half of the block, and all odd address data is stored in the top half. The effect is that if 16 bit data had been loaded into the RAM (from the serial port) it can be split so that 2 EPROMs can be programmed : one containing the data at even addresses, the other containing data at odd addresses.

SHUFFLE

This is the converse of SPLIT. The effect of shuffle is to interleave the data in the top half of the block with data in the lower half i.e. a 16bit to 8bit shuffle. The block limits are defined by the device selected from the menu.

INSERT (also see DELETE)

Inserts a free byte (FF) at any address in the RAM. The XP640 searches the RAM starting at the current cursor address for the occurrence of 5 unused bytes (5 bytes at FF). If free space is found, the first byte at FF is shifted back through the intervening data to the cursor address. The data at this address can now be modified using the DATA function. Once the INSERT mode has been entered, pressing ENTER will insert free bytes as often as required. if there are no free bytes or the RAM is completely cleared, a 'NO SPACE' message is displayed. To exit from INSERT mode, press STOP.

Example: Insert data at address 0010. This example assumes the RAM is completely filled with data except for 5 free bytes starting at address 0010.

KEYPRESS	DISPLAY	MEANING
	0010 00 FF READY	Position cursor at the insert address
(FN) INSERT	BUSY	locating free bytes
	0010 FF FF INS	insert complete
ENTER	0010 FF FF BUSY	locating free bytes
	NO SPACE	no free bytes available

Note:

- No data has been lost or added. The first FF in the 5 byte block has been shifted through memory to the cursor address, no further insertions were possible because there was no more free space.

DELETE (also see INSERT)

Deletes any byte in RAM provided there are at least 5 bytes of free space above the delete address. The XP640 will search for free bytes starting at the cursor address and working to the top of the RAM. 0nce found., the data at the cursor address will be deleted intervening data will be shifted down one address and an FF will be added to the free bytes block.

Example: Delete data at 0005. This example assumes the RAM is completely filled with FF except for a data block at 0000 0007.

KEYPRESS	DISPLAY	MEANING
	0005 00 FF READY	Cursor is at the delete address
(FN) DELETE	0005 0 FF DEL'	Delete first byte
ENTER	0005 0 FF DEL'	Delete again
ENTER	0005 0 FF DEL'	And again
ENTER	0005 FF FF BUSY	
	NO SPACE	No more deletions possible, all data from cursor to top of RAM is at FF

REPLACE (also see SEARCH)

Replaces a data string with a new data string. Any number of occurrences of a string can be found, (see SEARCH) and changed to the new string. Maximum string length is 10 bytes. The search for strings begins at the cursor address and works towards the top of the RAM.

Example: Replace the data strings at 0010, 0020 to 45, 67. This example assumes that the RAM is filled with FF except for the 2 strings of 01, 23 at addresses 0010, 0020.

KEYPRESS	DISPLAY	MEANING
	0000 FF FF READY	Position cursor to start of RAM to begin search
(FN) REPLACE	FIND_	Prompt for string to be found
0123	FIND 01 23	
ENTER	REPLACE WITH _	Prompt for new string data
4557	REPLACEWITH 45 67	
ENTER	HOWMANY SWOPS_	Prompt for the number of string changes
2	HOWMANY SWOPS2	Prompt for new string data
ENTER	BUSY	Busy searching
	DONE	All required strings have been replaced
STOP	0020 45 FF READY	cursor is at the start of the last string to be replaced

Note:

- The maximum string length that can be changed is 10 bytes.
- Any number of strings can be replaced.

SEARCH (also see REPLACE)

Searches the RAM for the occurrence of a specified data string. The search starts at the current cursor address and proceeds until a match is found with the specified string. Subsequent or previous string occurrences can be found by using the cursor right and cursor left keys.

Example: Search the RAM for the data strings 30, 31 . This example assumes that the RAM is filled with FF except for two strings of 30, 31 at addresses 0010, 0020.

KEYPRESS	DISPLAY	MEANING
	0000 FF FF READY	Position cursor at RAM start
(FN) SEARCH	FIND_	Prompt for string data
30 31	FIND 30 31 _	
ENTER	BUSY	search for first string
	0010 30 FF NEXT	found it at 0010
RIGHT ARROW	0020 30 FF NEXT	next string found
RIGHT ARROW	0020 10 FF BUSY	
	DATA NOT FOUND	no more strings in RAM

Note:

- The maximum string length that can be searched for is 10 bytes.

LOCK

This useful command will lock out the RAM edit or to prevent accidental use or use by unauthorised personnel. The PROM functions and cursor keys are not inhibited. A 4 digit code is required to lock and unlock the editor.

Example: Lock and unlock the editor with code 01 21

KEYPRESS	DISPLAY	MEANING
(FN) LOCK	LOCK_	Prompt for code
0121	LOCK-0121	
ENTER	0020 30 FF READY	Editor is locked out
(FN)	UNLOCK-	Pressing FN asks for unlock code
0121	0020 10 FF READY	Editor unlocked

PRINT

This key outputs data in the currently selected format via the parallel port. The key requests start and end addresses, and for records with address fields it also asks for an offset. Once all parameters have been entered, it will print the data.

3. PROM Functions

The table below briefly describes the PROM function keys, a detailed explanation is given later in this section. Each function (except BLANK, ERASE, MENU, EMU) operates on a user defined block of data in the RAM and device socket. If no block has been defined, then the function operates on the whole device and its corresponding RAM area.

KEY	DESCRIPTION
IBC	Perform an illegal bit check on the PROM using RAM block data.
CRC	Calculate the CRC value for the complete PROM or a specified RAM block.
SUM	Calculate the checksum of the complete PROM or a specified RAM block.
STORE	Copy PROM data starting at the specified address to the RAM block.
VERIFY	Verify PR0M against RAM and show error data.
PROGRAM	Program the PROM at any specified address with the RAM block.
BLANK	Performs a blank check on the entire device.
ERASE	Electrically erase EEPROM.
MENU	Device table.
EMU	Emulation function.

Note:

- The block is defined using the DEFINE key and defines a RAM block.
- If no block is defined, the function operates on the whole PROM and RAM areas.
- If the PROM start address is outside the range of the selected device it will be rejected and requested again.

MENU (Device Selection, Electronic Identifiers)

Device Selection

The XP640 must be set up to correspond to the particular type of EPROM to be read or programmed. The device type is selected using the MENU key and the cursor up, down keys or hex keys. By pressing the MENU key the machine displays the current EPROM selected. The XP640 when supplied as new will default to 2764 at power on however this default value can be changed at any time. (see SET PARAMETERS).
Depressing either the cursor up or cursor down keys will step the display through the EPROM list. Once the required device appears in the display, Press ENTER to select it. A Device selection can also be made using the hex keys followed by ENTER.

The currently selected device number always appears in the status section of the video display. To select the correct device from the device menu, refer to the two tables listed overleaf.

PROM manufacturers are listed on the left hand side of the page, and their respective devices are listed to the right. The correct selection for the XP640 is listed at the top of the page in the line labelled DEVICE MENU. Some devices are apparently duplicated in the device menu. E.g. 2764N, 2764I, 2764A & 2764Q. The suffixes (N, I, A or Q) refers to the programming method required by those devices as stipulated by the EPROM manufacturers.

KEY	DESCRIPTION
N	Normal program (50ms pulse)
I	Intelligent programming.
A	INTEL 'A' version of standard part.
Q	Fujitsu Quick Pro programming method.

It is important to match the XP640 with the devices you are programming E.g. A 2764A does **NOT** program in the same way as a 2764. Damage to the devices or inadequate programming may result if the incorrect setting is used.

Electronic Identifier

EPROM's now provide high speed programming algorithms along with electronic identifiers (E.g. INTEL's intelligent identifier, SEEQ's silicon signature) . These identifiers are provided to match the selected device to the correct high speed. algorithm. Its main use is to prevent the use of a high speed programming algorithm on non-intelligent devices (and thereby possibly under-program the device). If an intelligent device is selected from the menu, the user is given the option to use the electronic algorithm identifier. In response to the prompt 'AUTO SELECT ?', key CLEAR for no (don't use the identifier) or ENTER for yes (use identifier).

PROG (Program)

Programs the PROM with RAM block data after performing an illegal bit check to test for programmability. Once programming is complete, the PROM is verified and a checksum displayed.

Example: Program entire PROM. The PROM is programmed with data starting at PROM & RAM address 0000.

KEYPRESS	DISPLAY	MEANING
STOP	READY	Remove any block definition & power down the ZIF socket.
PROG	BIT CHECKFAIL	Fail bit check, ZIF is powered down.
	PROGRAM BUSY	Bit check pass, program cycle in progress.
	PROGRAM - F01E	Programming complete, verify pass, checksum displayed.
	PROGRAM FAIL	Fail to Program, enter verify mode.
	0024 00 FF VMODE	First error is at RAM & PROM address 0024, RAM data is 00, PROM data is FF. (Error data is available because RAM & ROM start addresses are the same).

Note:

- See VERIFY function for a description of VMODE.

Example Program the RAM block 8000 8010 into the PROM starting at PROM address 0000.

KEYPRESS	DISPLAY	MEANING
(FN) DEFINE	DEFINE _	Prompt for start address of block.
8000	BLOCK 8000	
ENTER	BLOCK 8000 _	Prompt for end address.
8010	BLOCK 8000-8010_	
ENTER	BLOCK 8000-8010	RAM block defined.
PROG	ROM START _	Prompt for ROM address.
0000	ROM START-0000	
ENTER	PROGRAM BUSY	Block program in progress.
	PROGRAM = CD0F	Program PASS & block checksum displayed.

VERIFY

Compares a user-defined RAM block with the PROM. If no block is defined, then the entire PROM is verified against RAM data starting at address 0000. If the RAM and PROM contain identical data then a PASS

message is displayed. If the PROM falls to verify then verify mode is entered to display error data.
Once in verify mode the following points apply:

- If the cursor lies outside the RAM area corresponding to the PROM it is automatically moved to address 0000.
- The search for errors always starts from the current cursor position proceeding to the top of RAM.
- The display shows the first error encountered and this is the new cursor position.
- All errors on a video page are shown, and these are shown as highlighted bytes, the cursor being shown as a highlighted nibble.
- The cursor left and cursor right keys can be used to move to the previous or next error occurrence -if no more errors are present the display will show an 'OUT OF RAM' message.
- If a block has been defined and verify errors are present, the search for the first error always starts from the start address of the block. Only block data is shown, other bytes not in the block are shown as blanks on-screen.
- Provided that the RAM block start address is the same as the PROM start address, then actual error data is shown. If the blocks are at different addresses, the RAM error address is always shown along with RAM and PROM data at that same address.

Example: Verify a complete PROM against RAM data.

KEYPRESS	DISPLAY	MEANING
STOP	READY	Remove any unwanted block definition.
VERIFY	VERIFY PASS	RAM and PROM contain identical data
	VERIFY FAIL	Errors are present
	1024 00 FF VMODE	The search started from the current cursor position. First error is at the new cursor address at 1024, RAM data is 00, PROM data is FF.

Note:

- Use the cursor left key to view previous errors and the cursor right key to view next errors.

Part of a PROM can be verified with any user specified RAM block as illustrated in the following example.

Example: Verify the RAM block 8000 8010 with a 2716 EPROM starting at PROM address 0000.

KEYPRESS	DISPLAY	MEANING
(FN) DEFINE	DEFINE _	Prompt for RAM block start address
8000	BLOCK 8000	Enter start address
ENTER	BLOCK 8000 _	Prompt for end address
8010	BLOCK 8000-8010_	Enter end address
ENTER	BLOCK 8000-8010	Block is now defined.
VERIFY	ROM START _	Prompt for PROM start address
0000	ROM START-0000	Enter PROM start
ENTER	VERIFY BUSY	Comparing PROM and RAM data
	VERIFY PASS	Verify complete
	VERIFY FAIL	Verify mode entered, first error is at RAM address 8001
	8001 00 00 VMODE	(PROM address 0001). This is the first error address to be found, but no error data is available because the RAM block start and PROM start addresses are different.

STORE

Stores data from the PROM to the RAM, verifies PROM against RAM data, then calculates and displays a checksum. A complete device can be stored, or part of a device may be stored using the DEFINE function.

Example: STORE a 2764 into the RAM.

KEYPRESS	DISPLAY	MEANING
STOP	READY	Un-define any RAM block
STORE	STORE BUSY	Copy the PROM to the RAM starting at RAM address 0000, then verify and checksum the PROM
	STORE 2D9A	Store successful & checksum displayed
	STORE FAIL	Store unsuccessful (fail verify)
	0044 00 FF VMODE	Verify mode entered use cursor left or right keys to view error data. (First error is at 0044, RAM data 00, PROM data FF).

Note:

- For a description of the verify mode and how error data is displayed, see VERIFY function.

Part of a PROM can be stored to any RAM start address as shown below:

Example: STORE PROM block 0010-001F to RAM block 2030-203F.

KEYPRESS	DISPLAY	MEANING
(FN) DEFINE	DEFINE _	Prompt for RAM block start address.
2030	BLOCK 2030	Enter block start.
ENTER	BLOCK 2030 _	Prompt for RAM block end address.
203F	BLOCK 2030-203F	Enter the block end.
ENTER	BLOCK 2030-203F	Block defined.
STORE	ROM START_	Prompt for PROM start of block.
0010	ROM START 0010	Enter PROM start.
ENTER	STORE BUSY	Stores the data block & verifies RAM.
	C'SUM 0769	Store successful, display RAM block checksum.
	STORE FAIL	Store unsuccessful.
	2034 AA AA VMODE	Error is at address 2034, but both PROM and RAM data are AA at this address (actual data cannot be shown because the RAM & PROM blocks do not start at the same address).

Note:

- See DEFINE function for more details on block defining.
- See VERIFY function for details of VMODE(verify node).

SUM (Checksum)

Calculates the 2 byte checksum of any length RAM block or of the entire PROM. The checksum is the 16 bit addition of all the bytes in the block. The carry from the 16th bit is discarded to give a 2 byte value.

Example: Calculate the checksum for the RAM block 0000-1FFF.

KEYPRESS	DISPLAY	MEANING
(FN) DEFINE	DEFINE _	Prompt for start address of block.
0000	BLOCK 0000	Enter the start address.
ENTER	BLOCK 0000 _	Prompt for RAM block end address.
1FFF	BLOCK 0000-1FFF	Enter end address.
ENTER	BLOCK 0000-1FFF	Block now defined and highlighted on-screen.
SUM	RAM C'SUM BUSY	Calculate checksum.
	RAM C'SUM = FO1E	Display checksum (F01E in this case).

Note:

- Once a block has been defined it is highlighted on screen and shown by DEF-D on the fluorescent display. To clear the block definition, press STOP.
- The block could have been defined using the cursor keys. (See DEFINE function).

A complete device can be quickly checksummed as shown in the following example:

Example: Calculate the checksum of a 2716 EPROM (select 2716 from the device menu).

KEYPRESS	DISPLAY	MEANING
STOP	READY	Un-define any unwanted block.
SUM	CHECKSUM BUSY	Calculating the checksum.
	CHECKSUM 25AD	Display checksum (25AD in this case)

Note:

- To calculate the PROM checksum no block must be defined a defined block operates on the RAM, not the PROM.

CRC (Cyclic Redundancy Check)

The cyclic redundancy check is a complex algorithm which produces a unique number to 'describe' a block of data. it is similar in many respects to a checksum, but is more reliable as a check value, since any changes in the data will always produce a new CRC value (this is not always the case with checksum). The CRC function will calculate a value for any length RAM block or produce a value for the entire PROM.

Example: Calculate the CRC for the RAM block 0100-01FF.

KEYPRESS	DISPLAY	MEANING
(FN) DEFINE	DEFINE _	Prompt for start address of block.
0100	BLOCK 0100	Enter block start
ENTER	BLOCK 0100 _	Prompt for end address of block.
01FF	BLOCK 0100-01FF	Enter block end address.
ENTER	BLOCK 0100-01FF	Block now defined & highlighted on screen.
CRC	RAM CRC BUSY	Calculating CRC of block.
	RAM CRC = EF57	CRC for the RAM block is displayed (EF57 in this case)

Note:

- The block remains defined unless the STOP key is pressed.
- The block could have been defined using the cursor keys (See DEFINE section).

Example: Calculate the CRC of a 27128 EPROM (27128 selected from device menu).

KEYPRESS	DISPLAY	MEANING
(STOP	READY	Clear any defined block.
CRC	CRC BUSY	Calculating PROM CRC.
	CRC = E5CF	CRC for the PROM is displayed (E5CF in this case).

Note:

- To calculate the PROM CRC, no block must be previously defined since a defined block operates on the RAM, not the PROM.

IBC (Illegal Bit Check)

Performs an illegal bit cheek on the PROM using RAM block data starting at a specified PROM start address, The IBC is a check for programmability it checks that all bits in the device can be set to the required pattern in the RAM. '0' '1 ' A programmed bit cannot be set to a 1 without exposure to UV light (EPROMs), or electrical erasure (EEPROMs).

Example: Illegal bit check an entire device with RAM data starting at address 0000. (Select the required device from the menu).

KEYPRESS	DISPLAY	MEANING
STOP	READY	ZIF powered down, insert device, un-define any RAM block.
IBC	BIT CHECKBUSY	Perform IBC on PROM using RAM data starting at 0000.
	BIT CHECKPASS	Device can be programmed with RAM data.
	BIT CHECKFAIL	Fail IBC.

An illegal bit check can also be performed using the block DEFINE function:

Example: Illegal bit check a PROM block starting at PROM address 0200 using a pre-defined RAM block at 0400-0500.

KEYPRESS	DISPLAY	MEANING
(FN) DEFINE	DEFINE	Prompt for RAM block start address.
0400	BLOCK 0400	Enter start address.
ENTER	BLOCK 0400 _	Prompt for RAM block end address.
0500	BLOCK 0400-0500	Enter end address.
ENTER	BLOCK 0400-0500-	Block defined.
IBC	ROMSTART _	Prompt for ROM start address.
0200	ROM START-0200	Enter PROM start.
ENTER	BIT CHECKBUSY-	Performing bit check.
	BIT CHECKPASS	PROM can be programmed successfully.
	BIT CHECKFAIL	Fail illegal bit check.

Note:

- See DEFINE function for details of block defining.

BLANK

Performs a blank check on the selected device. If all bytes in the selected device are Hex FF, a PASS message is displayed.

ERASE

Electrically erases the selected EEPROM then performs a blank check to give a PASS or FAII message. The device type selected must be an EEPROM any other selection will give an error message.

EMU (Emulate)

Sends RAM data to the optional XM512 Emulator Module via the parallel port. The data sent is the same length as the currently selected device and starts at address 0000. Typical time to transfer the entire RAM (64k x 8) is 6 seconds.

4. XP640 Interfaces

XP640 Serial Data Transfers

Introduction

The XP640 has a bidirectional RS232C port as standard . This port nay be used to receive data for device programming, from a host computer, transmit data to a host computer or printer, or used as a communications link to an RS 232C terminal for remote operation. The RS232C port will support transmission/reception baud rates between 110 and 19200 baud. The data may be received in any one of 15 formats, and transmitted in 16.

Serial Data Transfer Formats

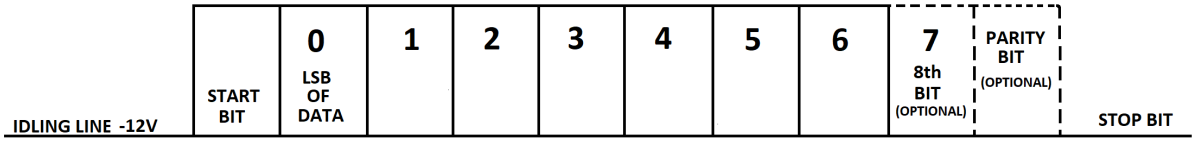
1. MOS TECHNOLOGY
2. SIGNETICS ABSOLUTE
3. TEKTRONIX HEXADECIMAL
4. BINARY
5. DEC BINARY
6. ASCII HEX COMMA
7. ASCII HEX APOSTROPHE
8. ASCII HEX PERCENT
9. ASCII HEX SPACE
10. BIOF
11. BHLF
12. BPNF
13. LIST (Output only)
14. MOTOROLA EXORCISER
15. INTEL HEX
16. GP BINARY

These formats are all described in detail in Appendix A.
A full specification of the serial formats will be found in the appendix. The speed , format, word format and handshaking selections are made using the XP640 menu selection.

Word Format

XP640 word format	
START BITS	1
STOP BITS	1 or 2
DATA BITS	7/8
PARITY	ON/OFF/ ODD/EVEN

The Serial Word:



Handshaking

The XP640 uses hardware handshaking via CTS/DTR and DSR/RTS. When the XP640 is receiving the DTR and RTS line (pin 20 and pin 4) must be used to control the data flow into the programmer. A high level (+12V) on the RTS & DTR line indicates ready to receive. A low level (-12V) indicates not ready. Before the XP640 will output data, the input handshake lines CTS and DSR, (pins 5 & 6) must taken to a high level (> 5V).If a handshake line changes state during a byte, the XP640 expects the transfer to continue until the end of the next stop bit.

Serial Output

The serial output key instructs the programmer to output data. The programmer will prompt for start and end addresses for the data to be transmitted Once the limits have been entered, the XP640 will prompt for an offset address. This address is added to the actual address of the data and transmitted in the address field of those formats that have address information. Once this has been entered, the XP640 will either transmit the data and display the message 'DONE SOUT' or it will show 'TIMEOUT ERROR' if the handshake lines are preventing serial output.

Serial Input

The serial input key instructs the XP640 to load data from the RS232 port into the RAM in the currently selected data format. Once the key has been pressed the XP640 will prompt for an 'OFFSET ADDRESS'. This is either taken as the start address of data for formats with no address information, or it is added to the address of those formats which include address information. If the currently selected format does not have a byte count facility' the XP640 will prompt for a "length" of the data being input. Once these parameters have been entered the XP640 will load in the data and display 'DONE'. If for any reason no data is transmitted to the XP640, it will display 'TIMEOUT ERROR'.

Remote Operation of the XP640

Pressing the Remote Key on the XP640 causes the XP640 to transfer control to the RS232 port. The first operation of the remote mode for the XP640 is to send out a menu of possible commands. All communication is at the settings previously defined from the port menu. Once the Command Menu has been sent, the XP640 outputs the '>' prompt indicating that it is ready to receive a command. Commands are entered by typing all or part of the menu commands, followed by a carriage return. If you enter an ambiguous command the XP640 will interpret it as being the first matching command in the Menu.

The Command Menu is listed below:

COMMAND	OPERATION
MENU	Define a block
SHIFT	Shift a block
FILL	Fill a block
MERGE	Combine 16 bit data
DELETE	Delete byte at cursor
FIND	Find string
DATA	Data entry
DUMP	Hex dump of memory
INVERT	Complement memory
COPY	Copy a block
SPLIT	Split 16 bit data
INSERT	insert FF at current cursor
REPLACE	Replace string
MEM	Define cursor address
PAGE	Define current page
PRINT	Parallel print
SOUT	Serial output
VERIFY	Verify device against RAM
CHECKSUM	Checksum
BITCHECK	Illegal Bit Check
ERASE	Erase EEPROM devices
PARALLEL SELECT	Select list format
STATUS	List XP640 status
SIN	Serial input
PROGRAM	Program device
STORE	Copy device data into RAM
CRC	Cyclic redundancy cheek
BLANKCHECK	Blank check
DEVICE SELECT	Device selection
EMULATE	Emulation function
LOCAL	Return command to XP640

All functions work in the same way as in the local mode, with the following addition:

Cursor Keys

The cursor keys are implemented as:

H	Cursor Right
G	Cursor Left
T	Cursor Up
Y	Cursor Down

A function may be terminated by keying 'Q', to which the XP640 will reply 'ABORTED' and then it will redisplay the prompt. Selection of device and parallel formats is made by typing in the name of the device format after selecting the selection mode, The XP640 confirms this selection by displaying your choice.
The ready prompt is displayed together with cursor and block information.

AAAA Cursor address	DD RAM data	PP PROM data	XTZ&B Machine status
BLOCK WXYZ-ABCD	Block limits		

DUMP

This is used to display hexadecimal data. It prompts for start and end addresses, once given it will print Hex data on the screen. Dump may be interrupted by keying CTRL–S which will cause the display to stop at the end of the current line. The dump may then be resumed with a carriage return or ended with 'Q'.

Internal Parameter Set-Up

All parameters of the XP640 operating system (other than the device type) are set up using the port key. The parameter selection is menu driven with the menus being visible on both the video display and the vacuum fluorescent display. On the video display a complete menu is displayed, with the current selection indicated by a cursor on the active line of the menu. The vacuum fluorescent display shows only the current line. On the left hand side of each menu line is a 2 digit number, this gives the line number of the menu entry (in Hexadecimal) . Selection from the menu may be made in one of two ways:

Method 1:

Use the Up and Down cursor keys to select the required line of the menu and press ENTER to select it.

Method 2

Press the HEX keys to select the desired line number. As soon as the first hex key is pressed the display shows 'SELECT _' . The CLEAR and ENTER keys are used as for the all other hex entry. If an invalid selection is made, the XP640 will beep and re-prompt with 'SELECT _'.

All of the sub menus return control to the the main menu.

To return to the XP640 ready mode, options 7 or 8 should be selected.

Main Port Configuration Menu

Cursor Keys		
00	BAUD RATE	Set up serial speed
01	SERIAL FORMAT	Select serial data transfer format
02	PARALLEL FORMAT	Select print data format
03	WORD FORMAT	Set up serial word format
04	EMULATION	Select 8 or 16 bit emulation
05	KEYBEEP	Switch key beep on/off
06	STATUS	Display of current status. (Nothing may be changed)
07	CALIBRATE	Calibrate procedure
08	SET PARAMETERS	Save parameters in internal EEPROM and return to command
09	END	Return to command level

The baud rate, serial format & parallel options present lists of speeds/formats which may selected.

Word Format The word format option goes through a series of questions. These are:

Word Format Options	
DATA BITS –	Answer 7 or 8
STOP BITS –	Answer 1 or 2
TEST PARITY ? –	ENTER= YES, CLEAR = NO
ODD PARITY ? –	Only if YES to above, then ENTER = YES, CLEAR = NO
HANDSHAKE ? –	ENTER =YES, CTEAR = NO
Returns to main menu	

Emulation The emulation option asks whether emulation is 8 or 16 bits. The appropriate value should be entered.

Keybeep The key beep option asks: KEYBEEP ON ? (Enter ȲES, CLEAR ÑO)

Status The status option gives the following display:

00	Device type
01	Baud rate
02	Message saying 'SERIAL'. (aids use with fluorescent display)
03	Serial format
04	Message saying 'PARALLEL'. (aids use with fluorescent display)
05	Parallel format
06	Stop bits
07	Data bits
08	Parity
09	Handshake
0A	Emulation

The cursor keys may be scrolled through the display to show the current parameters. Selecting any option causes a return to the main menu.

Calibrate The calibrate option allows the user to check the internal voltages of the XP640. See Section 4 below on calibration.

Set Parameters Saves the selection made in the internal EEPROM so that they will always be recalled on power up.

End Configures the machine with the new parameters, but does not save them.

The Printer Interface

General

The XP640 printer interface is a parallel interface. It is compatible with the Centronics type port which the majority of printers are equipped.

The data is transmitted in standard ASCII code with the 8th bit set to a zero. Carriage Returns and Line Feeds are sent at the end of each line.

Connection

The printer port is the 26-pin IDC connector on the rear of the XP640. It may be connected to any Centronics type printer via an IDC/CENTRONICS cable.
The pin out of the connector is shown in the table below:

Connector Pin Out

Pin	Signal	Pin	Signal
1	STROBE	14	Twisted Pair Ground(pin 1)
2	DATA 1	15	Twisted Pair Ground(pin 2)
3	DATA 2	16	Twisted Pair Ground(pin 3)
4	DATA 3	17	Twisted Pair Ground(pin 4)
5	DATA 4	18	Twisted Pair Ground(pin 5)
6	DATA 5	19	Twisted Pair Ground(pin 6)
7	DATA 6	20	Twisted Pair Ground(pin 7)
8	DATA 7	21	Twisted Pair Ground(pin 8)
9	DATA 8	22	Twisted Pair Ground(pin 9)
10	NC	23	Twisted Pair Ground(pin 10)
11	BUSY	24	Twisted Pair Ground(pin 11)
12	NC	25	GND
13	NC	26	NC

Description of Centronics Port Signals

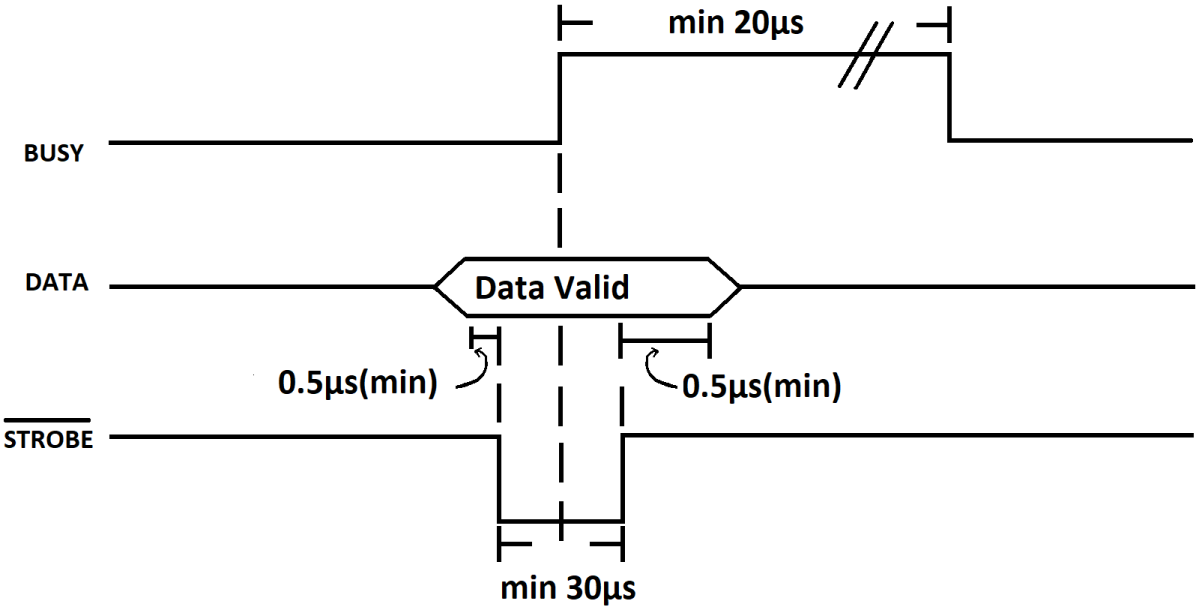
STROBE This is an active low output signal which is output to indicate that there is valid data on the port.

BUSY When this input is high the XP640 will not output data. It is used to indicate that the printer is not ready to receive data.

DATA 1-8 These lines carry the output data.

GND All of the ground lines are linked to the XP640 system ground. Problems with parallel interfaces often stem from bad grounds, hence ensure that all grounds are connected.

Centronics Port Timing Diagram



RS232 Connector Pin out

Pin	Name	Direction	Description
1	Shield		Protective ground.
2	TXD	OUT	Output data from P9000.
3	RXD	IN	Input data from P9000.
4	RTS	OUT	Paired with DTR.
5	CTS	IN	Handshaking input (controls data output).
7	GND		Signal ground.
20	DTR	Out	Handshaking output (controls data input).

Calibration Procedure

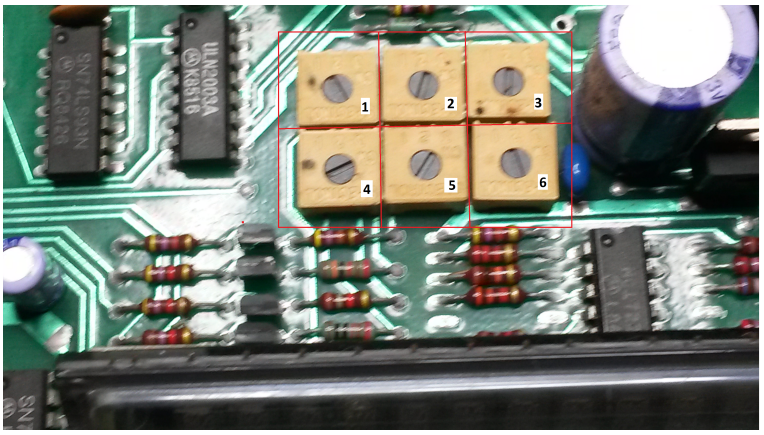
The XP640 is a precision made machine. All timing for program pulses, set up times etc. are software controlled by the Z80 microprocessor and are therefore crystal controlled and fixed. The power supply voltages are pre set and computer tested before they leave the factory. These voltages may need adjustment from time to time. Before attempting to calibrate the XP640, first check that it is required:
Select CALIBRATE from the port menu. Follow the sequence of steps listed below and measure the voltage as specified. Move to the next step by pressing the UP ARROW key. To exit from calibrate mode, press 'STOP'.
If one or more of the measured voltages are outside those specified in the table then repeat the procedure and adjust the pre set potentiometers numbered below.
To gain access to the potentiometers, remove the XP640 top cover. Please follow the instructions on its removal as given in the XP640 Service Manual.

- There are dangerous voltages inside the XP640 and calibration should only be carried out by a competent electronics engineer or technician.
- When reassembling the XP640 please follow the procedure given in the Service Manual.
- Damage caused by incorrect calibration or inexpert dismantling of the XP640 will void the warranty.

Calibration table:

Step	Pin	Lo Limit	Hi Limit	Adjust Pot
One	28	5.90	6.20	1
Two	28	4.80	5.20	4
Three	1	24.70	25.50	6
Four	1	20.70	21.50	1
Five	1	11.70	12.40	2
Six	1	4.80	5.20	5
Seven				Measure the TTL pulses on pin 27 of the copy socket to be of 1ms (approx) mark/space ratio. This checks that the system clock (crystal controlled) is OK to guarantee software timing. **No adjustment is possible or should ever be necessary.**

Potentiometer Identification



A. Serial Data Transfer Formats

Intel Hex Data Format

General

The Intel Hex format is a widely used format for the transfer of binary data. It transmits the data as short data records in ASCII code each, record having a checksum in order to ensure integrity of the data. There are several record types within the definition of Intel Hex, but the XP640 only uses three of them. These are: type 00 – data record, type 01 – the end of file record and type 02 – the extended address record. If the XP640 receives any other records it just discards them.

Intel Data Record Format (type 00)

Byte	
1	Colon (:) delimiter
2 – 3	Number of binary bytes of data in this record. The maximum is 32 binary bytes (64 ASCII bytes).
4 – 5	Most significant byte of the start address of the data.
6 – 7	Least significant byte of the start address of the data.
8 – 9	ASCII zeroes. The 'record type' for a data record.
10 –	Data bytes. Each binary byte is sent as two ASCII characters each one representing one nibble of the Hex representation of the byte.
Last two bytes	Checksum of all bytes in the record, excluding the delimiter, carriage return and line feed.The checksum is the negative of the modulo 256 binary sum of all of the bytes in the record.
CR/LF	Carriage return/line feed.

Intel Data Record Format (type 02)

Byte	
1	Colon (:) delimiter
2 – 3	'02' The record length.
4 – 5	ASCII zeroes.
6 – 7	Record type '02'.
8 – 9	USBA Upper segment base address. (the top 16 bits of a 24 bit address) It is used in Intel's 16 bit data records. If no 02 records are sent, the USBA is set to zero. If an USBA is specified, then the bottom 12 bits are added to the offset address of the data records.
10 – 11	Checksum of all bytes in the record, excluding the delimiter carriage return and line feed. The C'sum is the negative of the modulo 256 binary sum of all of the bytes in the record.
CR/LF	Carriage return/line feed.

Intel End of File Record (type 01)

Byte	
1	Colon (:) delimiter
2 – 3	ASCII zeroes.
4 – 5	Most significant byte of transfer address (Not used by XP540 ; Set to zeroes).
6 – 7	Least significant byte of transfer address (Not used by XP640 ; Set to zeroes).
8 – 9	Record type 01. Indicates end of record.
10 – 11	Checksum.
CR/LF	Carriage return/line feed.

Note: All ASCII code is sent as seven bits.

The data stream 25 45 AF B1 D0 77 to be sent to start at address 0000.

The Record would be: : 05 00 00 00 25 45 AF B1 D0 77 EB 0D 0A

Which may be broken down as :

Delimiter	:
Number of Bytes in the Record	06
Start Address High	00
Start Address Low	00
Record Type	00
Data	23 45 AF B1 D0 77
Checksum	EB
CR	0D
LF	0A

	Where the Checksum is calculated as follows:
CS	= 06+00+00+00+25+45+AF+B1+D0+77 = 315
Modulo 256	= 15
Negative	= EB

Note:	The above checksum calculation was performed in Hexadecimal.

Upper Segment Base Addresses (USBA)

The Intel Hex records which may be received by the XP640 may be either the standard 8 bit format (record types 0 & 1) or the extended 16 bit format (additional record type z). The USBA is a 16 bit number which is used to set the current segment base. (This terminology is derived from the Intel 8086). In effect this means that the 16 bit number is shifted right four times and added to the 16 bit address of the type 00 data records. This results in a 24 bit address. The XP640 actually uses the 16 least significant bits only.

Upper Segment Base Addresses (USBA)

USBA	=	1263h
Address in data record	=	3334h
Actual Address of data	=	12340h
		+ 3334h
		15674h

In the XP640 this would be	5674h

Motorola Exorcisor Format

General

The Motorola format provides for the transmission of data in printable ASCII format. The data is divided into records. The XP640 recognises and uses three types of record„ these are: 'S1' and 'S2' – the data records, and 'S9' – the end of file record.

Exorcisor Data Record Format (type 'S1')

Byte	
1	'S' character delimiter
2	ASCII 1. The record type for data.
3 – 4	Byte count. The number of binary data bytes in the record plus three (1 for checksum and 2 for address).
5 – 6	Most significant byte of the start address of the data record.
7 – 8	Least significant byte of the start address of the data record.
9	Data bytes. Each byte is sent as two ASCII characters, each representing one nibble of the hex representation of the byte.
Last two bytes	Checksum of all bytes in the record excluding the delimiter and record type. The checksum is the 2's complement (NOT) of the modulo 256 binary sum of the bytes in the record.
CR/LF	CR and LF are output from the XP640, but are not checked when input.

Exorcisor Data Record Format (type 'S2')

Byte	
1	'S' character delimiter
2	ASCII 2. The record type for data.
3 – 4	Most significant byte of start address of the data record.
5 – 6	Next most significant byte of the start address of the data record.
7 – 8	Least significant byte of the start address of the data record.
9	Data bytes. Each byte is sent as two ASCII characters, each representing one nibble of the hex representation of the byte.
Last two bytes	Checksum of all bytes in the record excluding the delimiter and record type. The checksum is the 2's complement (NOT) of the modulo 256 binary sum of the bytes in the record.
CR/LF	CR and LF are output from the XP640, but are not checked when input.

Exorcisor End of File Record

Byte	
1	'S' character delimiter
2	ASCII 9. Indicates end of file record.
3 – 4	Byte count 03 in end of file record.
5 – 6	Most significant byte of start address (not used in the XP640; set to zero).
7 – 8	Least significant byte of start address (not used by the XP640; set to zero).
9 – 10	Checksum.
CR/LF	Carriage return and line feed are output from the XP640, but are not checked when input.

To send data 67 A0 4A 2B to start at 213F would be:

S1 07 21 3F 67 A0 4A 2B 1C 0D 0A

Which consists of:	
Delimiter	S
Record Type	1
Byte Count (Data + 3)	07
Start Address High	21
Start Address Low	3F
Data	67
	A0
	4A
	2B
Check sum	1C
CR	0D
LF	0A
Where the Checksum is calculated as follows:	
CS = 07+21+3F+67+10+4A+2B	1E3
Modulo 256	E3
1's Complement	1C

N. B. : The calculations were performed in hex

GP Binary Format

General

This is a simple format devised by GP specifically for users writing there own formats. It is designed to be as simple as reasonably possible to write drivers/ receivers for. All data is sent in 8 bit binary, LSB first.

Format of GP Binary Record

The data is preceded by a 4 byte block consisting of a block-length and a checksum:

Byte	
1	Least significant byte of the block length.
2	Most significant byte of block length.
3	Least significant byte of the checksum.
4	Most significant byte of the checksum.
5 –	Data bytes.

The block length is the number of bytes in the data record.
The checksum is the modulo 65536 binary sum of the data being transferred.

A GP Binary record to send the following data 23 67 8F 2A would be:

Low Block Length	04
High Block Length	00
Low part of Checksum	43
High part of Checksum	01
Data	23
	67
	8F
	2A

Where the checksum was calculated as follows:

CS 23+67+8F+2A=143

N.B.: The above calculation was performed in Hexadecimal

Serial List Format

General

This is an output only format designed primarily to drive a serial printer. Data is output as ASCII characters in rows of 16 characters, each row being preceded by the address of the first character in the row. Each row is terminated by carriage return and line feed.. The data is sent in blocks of 256 bytes. After every third block a form feed is sent to prevent data being printed on the perforations of the paper.

Example of serial list output

```
0000  E4 AA CD 00 99 C9 E5 F5 E1 F1 4F 7D ED CF 21 01
0010  21 FF FF 0A E4 C4 01 C9 22 FD 22 E4 14 C3 FF FF
                           .
                           .
                           .
01E0  A4 B9 27 C9 22 FD 14 99 4F 6C CF FF FF FF FF FF
01F0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

Tektronix Hex Format (TekHex)

General

This format provides for the transfer of data blocked into records of printable ASCII characters. There are 2 types of records used and recognised by the XP640. These are the data record and the end of file record.

TekHex Data Record.

Byte	
1	'/' delimiter
2 – 3	Most significant byte of the start address of the data record.
4 – 5	Least significant byte of the start address of the data record.
6 – 7	Byte count. The number of binary data bytes in the record.
8 – 9	First checksum, sum of all bytes, modulo 256 of the six hex digits of the load address and byte count.
10 –	Data bytes. Each byte is sent as two ASCII characters, each representing one nibble of the hex representation of the byte.
Last two bytes	Checksum of all the data bytes in the record, calculated as the modulo 256 sum of all the nibbles making up the data bytes.
CR/LF	CR and LF are output from the XP640, but are not checked when input.

TekHex End of File Record.

Byte	
1	'/' delimiter
2 – 3	MSB of the start address of the data record (not used in the XP640; set to zero).
4 – 5	LSB of the start address of the data record (not used in the XP640; set to zero).
6 – 7	Byte count. 00 in end of file record.
8	(not used in the XP640; set to zero).
9 – 10	Checksum of all bytes in the record excluding the delimiter and record type. The checksum is the modulo 256 binary sum of the **nibbles** making up the bytes in the record.
CR/LF	CR and LF are output from the XP640, but are not checked when input.

To send the data 25 00 A8 A9 17 04 the data format would look like:

/000006062500A8A91704160D0A

Which consists of:

Delimiter	'/'
Start Address	0000
Byte Count	06
Checksum of Address field	06
Data	25 00 A8 A9 17 04
Checksum	16
CR	0D
LF	0A

Where the checksums were calculated as follows:

Address Checksum	0+0+0+0+6 = 6h
Data Checksum	2+5+0+0+A+8+A+9+1+7+0+4 = 36h

MOS Technology Format

General

In this format the data is divided into records and sent as printable ASCII characters. There are two types of record used and recognised by the XP640. These are the data record and the end of file record.

MOS Data Record.

Byte	
1	’; ’ delimiter
2 – 3	Byte count. The number of binary data bytes in the record.
4 – 5	Most significant byte of the start address of the data record.
6 – 7	Least significant byte of the start address of the data record.
8 –	Data bytes. Each byte is sent as two ASCII characters, each representing one nibble of the Hex representation of the byte.
Last four bytes	Checksum. Sum of all data bytes in the record. It is the modulo 65536 binary sum of all bytes in the record including the block length and address, but excluding the delimiter and checksum. It is transmitted high byte then low byte.
CR/LF	CR and LF are output from the XP640, but are not checked when input.

MOS End of file record.

Byte	
1	’; ’ delimiter
2 – 3	Byte count. 00 in end of file record.
4 – 5	Most significant byte of sum of total bytes sent in all records.
6 – 7	Least significant byte of sum of total bytes sent in all records.
8 – 9	MSB of checksum
10 – 11	LSB of the checksum of all bytes in the record excluding the delimiter and record type. The checksum is the modulo 65536 binary sum of the bytes in the record.
CR/LF	CR and LF are output from the XP640, but are not checked when input.

To send the data 86 AF E5 64 98 99 99 00 the MOS record would be:

;08 00 00 86 AF E5 64 98 99 99 00 04 48 0D 0A

Which consists of:

Delimiter	’;’
Byte Count	08
Start Address	0000
Data	86
	AF
	E5
	64
	98
	99
	99
	00
Checksum	0448

The checksum is calculated as follows:

Checksum	86+AF+E5+64+98+99+99+00 = 0448

Signetics Absolute Data Transmission Format

General

In this format data is divided into records of printable ASCII characters. The XP640 uses and recognises two types of data record. The data record and the end of file record.

Signetics Absolute Data Record.

Byte	
1	':' delimiter
2 – 3	Most significant byte of the start address of the data record.
4 – 5	Least significant byte of the start address of the data record.
6 – 7	Byte count. The number of binary data bytes in the record.
8 – 9	Checksum of the bytes in the address and data fields calculated by EXORing each byte with the previous byte, then rotating the resultant byte left one bit.
10	Data bytes. Each byte is sent as two ASCII characters, each representing one nibble of the Hex representation of the byte.
Last two bytes	Checksum. Sum of all data bytes in the record, the checksum is calculated in the same way as the first checksum.
CR/LF	CR and LF are output from the XP640, but are not checked when input.

Signetics Absolute End of File Record

Byte	
1	':' delimiter
2 – 3	MSB of the start address of the data record. (not used in the XP640 ; set to zero).
4 – 5	LSB of the start address of the data record. (not used by the XP640 ; set to zero).
6 – 7	Byte count. 00 in end of file record.
8 – 9	Checksum of the bytes in the address and data fields calculated by EXORing each byte with the previous byte, then rotating the resultant byte left one bit.
CR/LF	CR and LF are output from the XP640, but are not checked when input.

To send the data 23 EE F1 2A D4 55 99 the MOS record would be:

;:00 00 07 0E 23 EE F1 2A D4 55 99 46 0D 0A

Which consists of:

Delimiter	':'
Start Address	0000
Byte Count	07
First Checksum	0E
Data	23
	EE
	F1
	2A
	D4
	55
	99
Second Checksum	46

	The checksums are calculated as follows:
1st checksum	(((00 Xor 00)*2 Xor 00)*2 Xor 07)*2 = 0E
2nd checksum	((((((23 Xor EE)*2 Xor F1)*2 Xor 2A)*2 Xor D4)*2 Xor 55)*2 Xor 99)*2 = 46

The ASCII Space, Comma, Apostrophe and Percent

General

Data in these formats is transmitted in sequential, two character groups representing hex bytes followed by the execute code 'space', 'percent', 'apostrophe' or 'comma'. Data may be transmitted as either 4 or 8 bits. The XP640 assumes that the two characters prior to the execute code were a valid character. If only one character was received prior to the execute code then a leading zero is assumed.

When the XP640 is receiving in these formats, it recognises 3 types of information; these are Address information, Data and Checksum. The data transmission must be preceded with an <STX> character (02h) which may then be followed immediately with data or by an address field. The transmission must be terminated with an <ETX> (03h) followed by either a checksum field or at least 16 nulls.

Data field

Each time an execute code is received the two previous bytes are assumed to be valid data. If there have not been two valid ASCII Hex bytes prior to the execute code then the programmer assumes leading zeroes. Address field 'rt' ,A" When the XP640 receives a followed by an it then expects 4 ASCII Hex digits giving the address of the first data field. This address must be terminated by a comma (except in the 'Comma' format where it is terminated by a full stop).The input data will then be loaded, starting at this address.

Checksum field

The data field must be terminated with an <ETX> this may optionally be followed with a checksum. The checksum is expected as $ followed by 'S' followed by the four bytes of the checksum. The checksum must be terminated with a comma (or for the comma format a full stop). The checksum is calculated as the modulo 65536 sum of all of the data sent since the previous <STX>. If the checksum is not sent then at least 16 characters must follow the <STX> to prevent a time-out error.

ASCII SPACE data transmission

<STX>$A000,<CR><LF>
31 FF 77 C7 FF FE 76.....<ETX><CR><LF>
$S1234,<CR><LF>

ASCII COMMA data transmission

<STX>$A0000.<CR><LF>
31,FF,77,C7,FF,FE,76.....<ETX><CR><LF>
$S1234.<CR><LF>

ASCII PERCENT data transmission

<STX>$A0000.<CR><LF>
31%FF%77%C7%FF%FE%76.....<ETX><CR><LF>
$S1234,<CR><LF>

ASCII APOSTROPHE data transmission

<STX>$A0000.<CR><LF>
31'FF'77'C7'FF'FE'76.....<ETX><CR><LF>
$S1234,<CR><LF>

ASCII BPNF, BHLF & B10F Formats

General

In these formats each byte of data is transmitted as an ASCII 'B' followed by eight ASCII bytes representing the bits of the data byte. Zeroes and ones are represented respectively in the two formats by: 'N', 'P' or 'H', 'L' or '1', '0'. Each byte is terminated with the ASCII character The data is transmitted least significant bit first. The entire data stream must be started with a non-printable <STX> and ended with a non-printable <ETX>. The data output from the XP640 is formatted to suit a list device by outputting a space between each byte, and a <CR><LF> at the end of each line of six bytes.

An example data stream 0F 84 73 21 is shown in each format below:

BPNF format.

<STX>BPPPPNNNNF BNNPNNNNPF BPPNNPPPNF BPNNNNPNNF<ETX>

BHLF format.

<STX>BHHHHLLLLF BLLHLLLLHF BHHLTHHHLF BHLLLLHLLF <ETX>

B10F format.

<STX>B11110000F B00100001F B11001110F B10000100F<ETX>

DEC Binary and Binary formats

General

In both of these formats data is transmitted as a string of binary information. The only difference in the two formats is the start of record. For Binary the record starts with any number of nulls followed by a rubout (FFh). In DEC binary the format starts with any number of rubouts followed by a null. The data after the record start is a string of binary data with no checksum or byte counts and no print formatting. As there is no end of file delimiter, the receiving machine must have been told how many bytes to expect. In the XP640 this is entered from the keyboard.

B. PROM Device Table

Devices recognised by the XP640

Manufacturer	2508	2708A	2708B	2716	2815	2816	48016	9716	2532	2732	2732A	2564
AMD				2716DC						2732DC	2732ADC	
Eurotechnique					ET2716					ET2732		
Fujitsu				8516						MBM2732	MBM2732A	
Hitachi					HN462716		HN48016		HN482532	HN482732		
Intel		2758A	2758B	2716	2815	2816						
Mitsubishi				M5L2716K					M5L2732K			
Motorola				MCM2716 MCM27A16					MCM2532			
National				MM2716 NM27C16		NMC2816		NMC9716	NMC2532	NMC2732 NMC27C32		
NEC				UPD2716D						UPD2732D	UPD2732AD	
OKI				2716						2732A		
Rockwell					R5213					R87C32		
SGS					2816A 5516A							
Texas Inst	TMS2508									TMS2532	TMS2732	TMS2564
Toshiba											TMM2732D	

Devices recognised by the XP640

Manufacturer	2764N	2764I	2764A	2764Q	27128N	27128A	27128I	27128Q	27256I	27256Q	27512I
AMD	2764DC					27128DC			27256DC		27512DC
Eurotechnique	ET2764										
Fujitsu				MBM2764 MBM27C64				MBM27128		MBM27C256	
Hitachi	HN27C64 HN482764				HN4827128						
Intel	2764		2764A			27128			27256		
Mitsubishi		M5L2764K									
NEC	UPD2764D UPD27C64					UPD27128D					
OKI	MSM2764RS										
Rockwell	R2764 R87C64										
SEEQ		2764				27128					
SGS	M2764										
Texas Inst	TMS2764										
Toshiba	TMM2732D				TMM27128D						

Alphabetical Index