GP INDUSTRIAL ELECTRONICS LTD

EPROM PROGRAMMER - EMULATOR

TYPE : EP4000

Instruction Manual

TABLE OF CONTENTS

INTRODUCTION

The EP4000 has been designed as a flexible system for programming and emulating all the popular NMOS EPROMs. It can be used as a straight forward programmer with ease of operator use, to give PASS/FAIL information for programming , verify and checksum.
The programming power can be expanded with the range of bipolar programming modules and the 2764 socket adaptor.

At the same time, the machines emulation capability, editing facilities and input/output makes this an ideal system for aiding in the development of programs in a high level language, or assembly language on a larger system. Then downloading the code to the EP4000 and using the emulation mode to debug the final stages. Alternatively the program can be developed in hex, entirely on the machine and run from prototype hardware. In both cases when the program has been developed, it can be transferred to an EPROM for use by the outside system, independent of the EP4000.

ABOUT THE MANUAL

The manual has been written as a guide to the use of the EP4000 EPROM Emulator Programmer. All aspects of the machine have been covered in detail, to allow anybody unfamiliar with EPROM programmers to be able to use it in the field.
Section 1 gives details on setting up the machine for use along with operating, care and maintenance instructions.
Section 2 gives a listing of status codes and a brief outline of the keyfunctions, controls and display.
Section 3 describes all the key functions along with examples of use.
Section 4 describes the operation of the cassette interface - setting up and using the port.
Section 5 gives details of the 8 bit handshake parallel port available.
Section 6 shows the method by which serial data transfers can be made, using any one of the 3 serial lines.
Section 7 details aspects of EPROM Emulation, giving 3 design examples to illustrate the use of this facility.
Section 8 gives general information on EPROM Programming and erasing and gives examples of the use of the programming facility, including block programming.

Appendix A gives brief details of the accessories available for the machine - please note that some of these are not yet in production, as indicated on the current price list.
Further appendices will be sent in the future to give users information on software and any further accessories which become available.

The design and specification of the machine are subject to continuous development, and improvement, consequently the machine may incorporate changes in detail from the information contained in the manual.
Any amendments will be issued at regular intervals between manual reprints.

SECTION 1

OPERATING INSTRUCTIONS

1.00 GENERAL

Before switching the machine on, it is generally recommended that sections 1, 2, and 3 be read right through, and that some time be spent on becoming familiar with the controls.

1.01 SUPPLY VOLTAGE

Machines supplied in the UK are ready to operate from the 240v, 50Hz supply. A ready wired cable, complete with fused plug is supplied. The cores of the cable are colour coded as follows:

Line Brown Neutral Blue Earth Green/Yellow

Note: If the case is to be opened for any reason - disconnect the unit from the mains supply before doing so.

1.02 OPERATING PRECAUTIONS

To ensure continued troble free operation, please note the following points.

1/ Operate the machine on a vibration free surface.
2/ Do not operate the machine in direct sunlight, or near any source of heat.
3/ Ensure the ventilation holes are not covered, and that external air flow is not restricted.
4/ Ensure no metal parts can fall into the machine - E.g. through the keypad aperture.
5/ Never switch the machine on or off with a device in the zero insertion force programming socket. This could destroy the device and possibly damage the machine.
6/ Always check the device selector setting, and key 'RESET' when inserting or removing EPROMs from the programming socket.
7/ Disconnect the unit from the mains supply when not in use.

1.03 CARE OF THE MACHINE

Treat the machine with the respect it, and any other item of equipment deserves. Observe the following points, to keep the machine in good condition.

1/ Periodically clean the ZIF with a stiff bristle brush, to ensure good contact and trouble free operation.
2/ Never force a device into or out of the ZIF - the socket is a zero insertion force device.

3/ If the outer surfaces require cleaning then use a damp cloth. Do not use volatile solvents - these will damage the plastic parts and the surface finish.

4/ Cover the machine when it is not in use, to prevent the ingress of dust and metal particles. (Disconnect from mains supply when not in use.)

1.04 SETTING UP AND SWITCHING ON

Connect a TV or video monitor to the EP4000. (See section 2 for connection data). Connect the machine to the mains supply and switch on. The LED display will show C1 3000 FF to indicate mode C1, and that the cursor is at address 3000, and data at that address is FF. Address 3000 is the start of the main RAM, and the entire RAM has been cleared to FF (erased state of EPROM).
Adjust the TVpicture with the height, width and position controls to view all the information. No adjustment is required if you are using the VM10 video monitor.
The ZIF LED is off to indicate that the socket is powered down, and ready to receive an EPROM. Check the device selector setting before inserting an EPROM into the socket.
The machine is now ready for use.

1.05 SERVICING AND MAINTENANCE

If the machine needs returning to us for repair or servicing under service contract or warranty, please ensure that:

1/ The machine is well packed for transit. Enclose it in a polythene cover and surround with soft packing material inside a strong cardboard box.

2/ Enclose a brief description of the fault.

3/ Enclose full name and address.

4/ State where and when purchased.

SECTION 2

Abbreviations & Definitions

2.00 Key Labels & Status Codes

The listing below explains the many status codes produced by the EP4000 and the abbreviations used for the keys. The list should be used as a guide when using the machine. The keyfunctions are described in detail in section 3.

	Key	Meaning	Status Code	Comments
	FN	function	FN	used to select uppercase keyfunctions
	TO	TO	to	data relocation. object/target definition
	DMA	Direct memory access	--	EPROM or RAM emulation facility
	RST	reset	C1	halts internal microprocessor
*	PAD1	scratchpad 1	P1	define memory area 4000-4FFF
	BLCK	block	bL	block memory area
*	DISP	displaeement	dP	displacement from previous keypress of cursor
	MEM	Memory address	Ad	define 4 digit hex address
*	CLR	clear	CL	clear target to FF
	↑	Move cursor	--	move cursor up (16 address move)
*	PRINT	print	--	dump object to EP printer
	←	move cursor	--	move cursor back
*	PAD2	scratchpad 2	P2	define RAM area 7000-73FF
	DEF	define	dF	define (block) mode
*	CSR2	Cursor 2	C2	screen scroll (C2 mode)
	CSR1	Cursor 1	C1	define on-screen cursor
*	SHIFT	shift	SH	shift (block) mode
	↓	move cursor	--	move cursor down (16 address move)
*	MTCH	match	HL	highlight (match) on-screen byte
*	S IN	Serial in	Si (FL)	prepare to receive data
*	S OUT	Serial out	(FL)	dump data via serial port
*	P IN	Parallel in	Pi	receive data from parallel port
*	P OUT	Parallel out	--	dump data to parallel port
*	C IN	Cassette in	Ci (FL)	receive data from cassette
*	C OUT	Cassette out	Co (FL)	dump data to cassette
*	PAGE	Page	PG	select 512 bytes for viewing on screen
*	RAM	Ram	rA	define RAM area 3000-3FFF
*	PROM	EPROM (in ZIF)	Pr	define EPROM area in accordance with device selector
*	EXEC	Execute	EC	execute (target) function
*	PROG	program	(PASS/FAIL)	program EPROM with RAM data
*	VFY	Verify	(PASS/FAIL)	compare EPROM and RAM data
*	CHCK	Checksum	CS	count bit set to logic 1 in EPROM
*	STOR	Store	(PASS/FAIL)	store EPROM data in RAM, then verify
	→	move cursor	--	move cursor forward

Upper case keys are indicated by *
Hex keys are lower case labelled 0-9, A-F used for direct data entry.

2.01 SPECIAL STATUS CODES

Some special codes are generated during the course of operation, usually when the machine requires further information - these are:

Status Code	Meaning
Ad	request for 4 digit hex address for new C1 position
Pg	request for 2 digit page number. The page number is the first 2 digits of the address of the first byte to appear on screen
Fn	next key entry will be upper case
HL	enter 2 digits for highlighting byte (match function)
FL	request for input/output code. May be 2 or 4 digits
no	indicates key sequence could corrupt internal data, and control sections, and is therefore not allowed
PASS	indicates good store, verify, program or load
FAIL	indicates bad store, program, verify or load

A blank display indicates the machine is busy or in the EPROM emulation mode. When busy, the machine does not respond to key strokes except RESET (RST).

2.02 DEVICE SELECTOR

This is the panel mounted 8 way rotary switch. It selects the device type required for EPROM Emulation and programming. It is a software polled switch, which is read every time a keypress is made. The position labelled 'EXT' is used for some external programming modules.

Selector Position	Voltage	Size (bytes)
2704	3 voltage rail	512
2708		1k
2716(3)		2k
2508/2758	single rail 5v	1k
2516/2716		2k
2532		4k
2732		4k
EXT	Depends upon external module	

Once the switch is read, power is applied to the programming socket as indicated by the ZIF LED (LED on = power on) and the chip select lines, high order address lines and extra power rails are configured correctly for the required device. At the same time, the emulation buffers are also configured ready for DMA.

2.03 PROGRAMMING SOCKET

The zero insertion force (ZIF) socket is a 24 pin, lever operated device designed for the rapid and easy insertion of EPROMs without any damage. The position of pin 1 is shown on the panel by a black dot next to the lever. Always ensure the EPROM is correctly orientated, and correctly seated before the lever is closed, otherwise the device,and the EP4000 may be damaged when the ZIF is powered up. When power is applied to the ZIF the LED will light. The socket can be powered down at any time by keying RESET. When using the socket to read or program EPROMs, always adopt the following procedure.

Device insertion

1/ Power down the socket by keying RESET

2/ Set the device selector to the required device type

3/ Insert device and close the lever -- check the device is in the correct way around, and that it is properly seated

4/ The socket will power automatically when any key is pressed.

Device removal

1/ Power down the socket by keying RESET

2/ Remove the device

2.04 LED DISPLAY

The 8 digit seven segment red LED display gives a constant readout of information to the user. The display is divided into 3 sections -

1/ Status (2 digits)
Gives information regarding keypress and mode. A complete list of status codes is given in 2.00

2/ Address (4 digits)
Usually displays the address at which cursor 1 is pointing.
Also used for information request entries under the FL status code and displays PASS or FAIL on completion of a store, program, verify or data load from the input ports.

3/ Data (2 digits)
Usually displays the data at cursor 1 address as a pair of hex digits.
The 'NO' status code will appear in this section of the display.

2.05 VIDEO DISPLAY

Two video outputs are provided

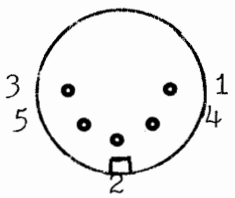
1/ Modulated video for connection to a T.V. receiver. The socket ia an 'aerial' type, located at the side of the machine. Tune the T.V. to channel 35 for a clear sharp display.

2/ Composite video for connection to a video monitor. The socket is a DIN type located at the rear right of the machine (as viewed from machine front). The centre pin is 0v and the video signal is available from any other pin (0.5v - 3.5v pp).

A lot of information is presented by the EP4000 for the video, and the height, width, and position controls should be used to view it all. (If you are using the VM10 monitor, then no adjustment is necessary - but use the 'out' position on the input selector of this monitor).

The information presented is a memory map of 512 consecutive bytes. These are arranged as 32 lines of 16 bytes, where each byte is a pair of hexadecimal digits. The cursor is the inverted video square whose address is shown in the LED display. The screen is divided into 4 shaded bands of 128 bytes each, to aid estimation of data block size. The cursor can be moved to any address location within the EP4000 memory - i.e. 120 screen pages are available for viewing (although many of these are not used).

5 PIN DIN VIDEO SOCKET (pin connections)



Socket view

(pin numbers are DIN standard)

pin 1 - video out
pin 2 - 0v
pin 3 - video out
pin 4 - video out
pin 5 - video out
shield - 0v

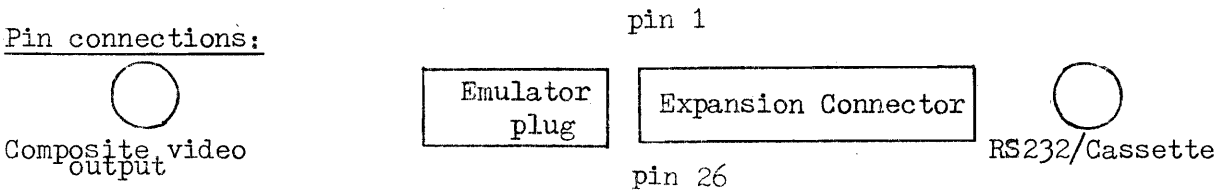
2.06 EMULATION PLUG

This is the 26 pin white plug at the rear of the machine, used for EPROM emulation. Input loading is 1 LS TTL load, and output leakage current is 10uA. The plug is only active when in the DMA mode. In order to achieve maximum speed of access time, no protection is provided on the plug, so care must be taken to ensure that the simulator cable is correctly inserted into the external host system. Take special care to plug the cable the correct way around. It is recommended that you use the standard simulator cable, buffer pod, or multi-EPROM simulator adaptor when emulating with the EP4000.

2.07 EXPANSION CONNECTOR

This is the 50 way plug at the rear of the machine. It gives direct access to the machine buses, the parallel port, and the TTL and 20mA serial input/output lines.

Pin connections:



Connections viewed from rear of the machine

pin number	Name	Pin number	Name
1	A7	26	Vcc
2	A6	27	A8
3	A5	28	A9
4	A4	29	Vcc
5	A3	30	NMREQ
6	A2	31	A10
7	D7	32	A1
8	D6	33	A0
9	D5	34	D0
10	D4	35	D1
11	D3	36	D2
12	0v	37	SIM R/W
13	SIM OE	38	A11
14	NWDS	39	NRDS
15	PB7	40	SIM DIR
16	A12	41	A13
17	A14	42	A15
18	PB6	43	CDMA
19	PA7	44	PA0
20	PA6	45	PA1
21	PA5	46	PA2
22	PA4	47	PA3
23	20mA input(return)	48	20mA input(supply)
24	TTL serial O/P	49	20mA out (return)
25	TTL serial I/P	50	20mA out (supply)

KEY

Pin	Description	Input/Output
D0-D7	Data bus	in/out
A0-A15	Address bus	out
PA0-PA7	Parallel port	in/out
PB6-PB7	Parallel handshake control	in/out
NWDS	Write strobe active low	out
NRDS	Read strobe active low	out
NMREQ	Memory request active low	out
CDMA	DMA Control	out
SIM OE	Simulation RAM O/P enable	in
SIM R/W	Simulation RAM write enable	in
SIM DIR	Simulation DATA buffer direction control	in
Vcc	+5v supply	out
0v	zero volts	out
NC	not connected	

Pin connection View from machine rear

Machine top

A7	A6	A5	A4	A3	A2	A1	A0	D0	D1	D2	GND	NC
NC	A8	A9	D	C	B	A	D7	D6	D5	D4	D3	NC

Key A0-A9 address lines from host system
D0-D7 data bus from EP4000
NC not connected
Inputs A,B,C,D depends on selected device type as shown below.

Device type selected	A(18)	B(19)	C(20)	D(21)
2704	0v	+12v	$\overline{CS}$	-5v
2708	0v	+12v	$\overline{CS}$	-5v
2716(3)	$\overline{CS}$	+12v	A10	-5v
2508/2758	$\overline{OE}$	0v	$\overline{CS}$	+5v
2716/2516	$\overline{OE}$	A10	$\overline{CS}$	+5v
2532	A11	A10	$\overline{CS}$	+5v
2732	$\overline{OE}$	A10	$\overline{CS}$	A11

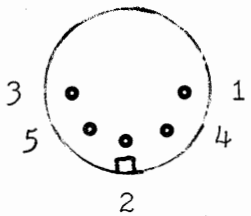
Pins A,B,C, and D are automatically configured by the EP4000 for the correct device type. The -5v and +12v for 3 rail devices are supplied from the host system, not the EP4000. These lines are used as reference levels by the EP4000.

CAUTION:
1/ Ensure the simulator cable is properly connected at both ends, ensuring correct orientation.
2/ Do not short any pin at any time - especially the data pins - short circuit current will destroy the line driver.

2.08 RS232/ CASSETTE SOCKET

This is the 5 pin socket located at the rear right of the machine (as viewed from rear) close to the 50 way plug.

Connection data



Socket view
(pin numbers are DIN standard)

pin 1 - Cassette input.
pin 2 - 0v
pin 3 - Cassette output
pin 4 - RS232 output (-5v, +5v)
pin 5 - RS232 input
shield - 0v

See 'serial data transfers' and 'cassette interface' for details

SECTION 3

3.00 Keyfunctions

This section gives a detailed description of the EP4000 keyfunctions, taken one key at a time. Examples are given in the use of each key by itself, and in conjunction with other keys.

3.01 Reset (RST)

Stops the internal microprocessor from executing the control program for as long as the key is held. The programming socket is powered down, as indicated by the ZIF LED. All functions are stopped, including the emulation facility. Releasing the key starts the machine in normal mode, as indicated by C1 in the status window. Cursor 2 is rewritten clearing any highlights, but C1 remains at the same address. Any function modes such as block define, shift, program, input/output are stopped and cleared to C1 mode. RAM data is not affected.
The serial output line will go to the mark condition (logic 1).

NOTE: If the ZIF contains an EPROM and reset is keyed, the socket will power down, so if you are viewing any page in the range 60 - 6F, because the socket is effectively empty (ZIF powered down), hex FF will be shown in all screen locations.
Powering the ZIF (by depressing any key) will have no effect, unless the page is rewritten by calling page 6X or by FN FN. (X = any hex key to define page number).

3.02 Hexadecimal Keys 0123456789ABCDEF

These are lower case keys for entering machine code at the C1 position shown on screen. The address of C1 is shown in the LED display.
These keys are also used for data entry when requested by the machine. For example - page selection, cassette file number, in/out speed and port number and cursor 1 address selection.
In normal mode when a 2 digit hex entry has been made at a particular C1 address, the cursor will automatically increment to the next address.

3.03 Function (FN)

This key is similar to shift or 'uppercase' on a typewriter or terminal and allows most of the keys to have 2 functions (except RST, TO and DMA). Example: Key FN 9 will select 'RAM' as the keyfunction and this should be thought of as Function RAM rather than FN 9 so that key sequences flow in a logical manner.

Double keying the function key, i.e. FN FN is a function in itself which will rewrite the screen and clear any highlights. This function is similar to RESET except that the programming socket is not powered down.

3.04 Cursor Control Keys

These are the four arrow keys, and can be used in four possible modes:

(1) C1 Mode
In this mode, the arrow keys control the movement of the on-screen cursor whose address is shown on the LED display. The cursor can be moved forward or back by 1 address or up/down (line jumping) by 16 addresses.

Example: Key CSR1 - the display blanks momentarily whilst a screen re-write is made. This key selects cursor 1 (C1) mode.
Now key ↓ and hold the key down. The cursor will jump a line at a time and fall off the screen bottom, reappearing at the top of the next 512 bytes, after a screen re-write. Releasing the key at any time will stop the cursor at the required address.

NOTE: It is not necessary to key CSR1 every time the cursor needs to be moved - this merely puts the machine into C1 mode, having cleared any other mode it may have been in.

(2) C2 Mode
The information presented on screen represents 512 bytes of memory, and can be considered to be a 'window' into the memory.
We shall call this window Cursor 2 or C2.
C2 mode allows the window to be moved up or down through the memory using the ↑ or ↓ keys. This is similar to a scroll function and allows the boundary of 2 consecutive pages of 512 bytes to be moved into the main screen area.

Example: Key FN CSR2 - this puts the machine into C2 mode. Now hold down the ↓ key. The screen is being re-written, so that C2 moves through the memory. Note that C1 remains in the same relative position on screen, but is of course moving through the memory.
To escape from C2 mode, key Reset, FN FN, or CSR1. This will put the EP4000 in C1 (normal) mode.

(3) dF (define) Mode
A block of data can be 'defined' in this mode, where the block length limit is 4k x 8. A block so defined can be shifted, cleared, relocated or treated in much the same way as any other memory area.
Cursor movement backwards in this mode will define the required block by highlighting it (reverse video). C1 address is shown in the display and it is the start of the block.
The end of the block is that address where C1 was when the define mode was entered.

Example: Key DEF and move the cursor back or up - The cursor encloses the required block. Moving the cursor forward will shorten the block as needed.

(4) SH (shift) Mode
In this mode the previously defined data block can be shifted with the cursor control keys. Data is shifted through RAM without overwriting or loss of data. Data in front of the block is transferred to the other side.
NOTE: The longer the block, the longer the delay before any apparent action takes place. This is because the data is moved in the RAM, and when complete C2 is updated to show the new block position. If no block has been defined, then the data at C1 will be shifted.

Example: Define a block as in (3) previously. Key shift ↓ .
The block will shift through memory line at a time for as long as the key is held.

To escape from any mode, Key FN FN, reset or select any other mode E.g. CSR1 (C1 mode).

3.05 TO Data Relocation

This is the data relocation key used to move data to or from an input/output port, or copy data to another memory area.

Example: PAGE 00 TO RAM will copy the data on page 00, (512 bytes) to the RAM area starting at 3000(H). The TO key is used to define PAGE 00 as the object and the RAM as the target area, to indicate the direction of data transfer.
This key is usefull for input/output control.

Example: SIN TO MEM 3010 will set the machine up to transfer data from the serial port to the RAM area, starting at address 3010(H).

3.06 DMA Emulation Mode (Direct Memory Access).

When the DMA key is pressed and released the internal microprocessor is disabled and isolated from the memory area.
The simulator cable buffers are enabled, to allow an external system to run the program in the EP4000's RAM. The machine now looks like an EPROM of the type selected by the device selector. The chip select and address lines have been automatically configured by the internal microprocessor to suit the selected device.
Prior to keying DMA, the external system should be held reset, or disabled, so that when the EP4000's RAM is accessed, the program is executed in a logical manner when the external system is enabled.

3.07 MEM Memory Address

This allows cursor 1 to be moved to any location within the 64k addressed by the internal microprocessor with the exception of the 4k block starting at 5000(H). This block is reserved for the EPROM configuration for the ZIF and emulator.

Example: MEM 3201 will move C1 to address 3201(H).
The correct page in which this address is located is now on screen.

MEM also serves as a specific memory area where the start of the area is C1 address and the end address is that of the 4k block in which C1 resides.

Example: PAGE 00 TO MEM 3200 will copy the data on page 00 to the RAM starting at address 3200. MEM has been defined as a target by the TO key.

Example: MEM 3800 TO MEM 3000 will copy the data from 3800-3FFF to 3000-3FFF. i.e. create two copies of the same 2k block of data 3800-3FFF and 3000-37FF.

Example: CLEAR MEM 3201 will clear all data at addresses 3201-3FFF to FF (erased state of EPROM).

Example: MEM 3800 TO PROM will transfer data from 3800-3FFF to the EPROM in the ZIF. This is useful when transfering data from one 2732 device into two 2716's. (The LED display will show when programming is complete).

3.08 PAGE Screen Page Selection

The screen area called cursor 2 (C2) can be described as a 'window' into the EP4000 memory area. C2 contains 512 consecutive bytes arranged as 32 lines of 16 bytes, divided into 4 shaded bands, each of 128 bytes. Any one of 240 screen pages can be called to the screen.

Example: PAGE 10 will copy the data from 1000-11FF to the screen. C1 has been assigned to address 1000(H).

Page can also be used as a memory area for relocation, programming etc.

Example: CLEAR PAGE 37 will write FF's to all locations on page 37, and at the same time call page 37 to the screen.
C1 is at 3700.

The high part of the page number describes a 4k block and the low part divides the 4k into 16 areas of 256 bytes.

3.09 CURSOR 1 (CSR1)

Depressing this key will define the area from C1 address to the end of the 4k block in which C1 resides as an object or target area for data movement etc. At the same time the machine will go to C1 mode (i.e. Cursor 1 is controlled by the arrow keys all the shift,C2 or define modes are cleared).

Example: If the machine is in C2 mode (screen scroll), then the arrow keys, up and down, will scroll the screen.
Keying CSR1 will now make the arrow keys move C1 only.

Example: CLR CSR1 (clear cursor 1) will clear the memory area between C1 and the end of the 4k block provided C1 is in a RAM area.

3.10 CURSOR 2 (CSR2)

This is the C2 mode key used for screen scrolling.
It enables the user to move the screen 'window' up or down through the EP4000 memory with the arrow keys. This is useful when working near the bottom of one page and the top of the next page. -- by moving C2 the area being worked on can be moved into the main area of the screen.

Example: CSR2 ↑ (Cursor 2 up) will move the memory window (C2) up through memory. To exit from this mode, key any mode key, FN FN, or RESET.

In addition to scrolling, keying CSR2 will also define C2 as a data block object or target. Once any operation has been performed on this block, the machine will exit from the scroll function and revert to C1 mode.
NOTE: The data block defined is that at the time of CSR2 keypress -- not that as viewed on screen after a scroll.

Example: CLR CSR2 (Clear Cursor 2) clears C2 contents to FF and the EP4000 is now in C1 mode.

3.11 SCRATCHPAD RAM 1 (PAD1)

This defines the area 4000-4FFF as an object or target area.
This area can contain a 2732 EPROM or 4k x 8 RAM and is not available on standard machines.

3.12 SCRATCHPAD RAM 2 (PAD2)

Scratchpad RAM 2 is a 1k block starting at 7000. The PAD2 key will define this RAM as an object or target. Scratchpad 2 is useful for the temporary storage of many data blocks for later recall to the main RAM area or for direct transfer to EPROM.

Example: Key PROM TO PAD2 will copy PROM data into scratchpad 2. Data will be overwritten if the PROM is larger than 1k.

Example: Key PAGE 30 TO MEM 7200 will copy page 30 to the scratchpad 2 starting at address 7200. In this way, up to 2 pages, or many previously defined data blocks can be temporarily stored for later retrieval.

3.13 RAM

Defines the 4k RAM area 3000-3FFF as an object or target.
This is the emulation RAM accessible via the 26way simulator plug under the control of the DMA key.

Example: Key RAM TO PROM will copy the RAM contents to the ZIF socket (i.e. program a 4k device).

Example: Key PROM TO RAM will transfer the PROM contents to the RAM starting at address 3000(H).

Example: Key CLR RAM (clear RAM) will clear the RAM to all FF. (Erased state of EPROM).

3.14 PROM

This defines the programming socket addresses as a target area for EPROM programming or an object area for data movement from the ZIF. The address limits of the PROM depend on the device selector position and therefore the device type.
The limits for the devices are shown below:

DEVICE	ADDRESS	BYTES
2704	6000-61FF	512
2708	6000-63FF	1k
2716(3)	6000-67FF	2k
2508/2758	6000-63FF	1k
2716/2516	6000-67FF	2k
2532	6000-6FFF	4k
2732	6000-6FFF	4k
EXT	depends upon the external programming adaptor	

Example: Key PROM TO RAM copies PROM data to the RAM starting at address 3000(H).

Example: Key PAGE 00 TO PROM will program the selected device with page 00 data.

3.15 BLOCK (BLCK)

This will fix the previously defined block as an object area only. Once the block has been relocated or cleared etc., the EP4000 reverts to normal C1 mode and the block limits disappear. If no block has been defined and block (BLCK) is keyed, the machine will show this by indicating NO on the display.

Example: Define a block as described in 3.04(3).

Key BLCK TO MEM 7100. This key sequence will copy the defined block into the scratchpad 2 RAM starting at address 7100. The original data block has not been lost but the block limits have disappeared.

Example: During the course of program development it is often useful to transfer a small program patch into a blank or partially programmed EPROM.
Key BLOCK TO MEM6101 transfers the block data into an EPROM of selected type, the start address being 6101(H).
When the programming cycle is complete the EP4000 will verify the block data only and indicate pass or fail.

Example: CLR BLCK (Clear block) will clear the defined block to FF.

If the data block defining is not required, it can be cleared by keying FN FN. (Block data is not lost -- only the defining limits).

3.16 INTERNAL (INT)

Defines a 2k block from 2801-2FFF. This area is read only memory and is useful for execution of a user program in 2732. (Half the 2732 has to be commited to a monitor program)

Example: EXEC INT (Execute internal memory). The internal microprocessor will execute the program at 2801. If there is no program the EP4000 will show this by indicating NO.
Data can be transferred from the INT memory using the TO key.

3.17 EXTERNAL (EXT)

Defines a 2k block of memory, or devices external to the EP4000 at F000-F7FF. This is used by some programming modules but could contain a user program in ROM or extra RAM as a scratchpad.

3.18 DISPLACEMENT (DISP)

This is useful for calculating relative addresses for jumps etc. When keyed, the EP4000 will calculate the address difference between the current C1 position and the C1 position when DISP was last keyed. The displacement is shown in the address section of the LED display for 5 seconds. The display then reverts to showing C1 address and data.

3.19 CLEAR (CLR)

An erased or new EPROM has all bits set to logic 1, so every location is FF (hex). For this reason, the clear function will set selected areas to FF as required.

Example: Key CLR RAM clears RAM to all FF.

Example: Key CLR BLCK (clear block) provides a useful means of clearing areas within a program, as specified by the block.

3.20 PRINT

This causes data to be sent to the parallel port as ASCII characters suitable for sending to a Centronics type printer - E.g. SEIKOSHA 100.
Example: Key PRINT PAGE 10 will send the 512 bytes on page 10 to the printer using handshake control.
Recommended connector for EP4000 to Centronics type printer is CR1 Printer interface cable, available as an accessory.

3.21 MATCH

Allows the user to highlight specific bytes on-screen for locating program areas. As many bytes as required may be highlighted.

Example: Key MTCH FF (match FF byte).
All on screen bytes which are at FF will be shown in reverse video.
The highlighting can be cleared by keying FN FN or RESET.

3.22 EXECUTE (EXEC)

This command key instructs the internal microprocessor to execute the program starting at the specified object address.
This facility allows the user to run programs from the programming socket, RAM area, INT, EXT, PAD1, PAD2 or any address (MEM) for specific tasks. E.g. Serial data transfers to customer formats.

The program must start with an 08 (NOP) instruction in order to start execution. If the program starts with any other instruction, the EP4000 will not execute the program, indicate NO and revert to C1 mode.
The internal microprocessor used is the INS8060.

Example: EXEC MEM 6000 will start program execution from address 6000 (ZIF socket) provided this location contains an 08 byte.
If the user program requires a return to the monitor, it should return using an XPPC3 instruction to 0000(H).

3.23 DEFINE (DEF)

Depressing this key identifies C1 address as the end of a datablock. Backward or upward cursor movement will extend the cursor to define the block. The current cursor position as shown in the LED display is the block start address. Moving the cursor forward will reduce the block length as required. The block so defined can now be cleared, shifted, relocated etc.

Example: Move C1 to address 3100.
Key DEF and move the cursor up to 3000. The block is between 3000 and 3010. Now move the cursor to 3010. This shortens the block to 3010-3100.
The block can, for example, be cleared by keying CLR BLCK (clear block).
To exit from the define mode, key any other mode (C1, C2, SHIFT) or FN FN or RESET.

3.24 SHIFT

This transfers the cursor control keys to the shift mode to move a defined data block. Data is moved through the RAM area without over-writing -- i.e. data is transferred from one to the other side of the block.

This is useful for shifting data to other areas for program rearrangement, and for shifting spaces into the program if data has to be inserted.

Example: Define the block as in 3.23. Key SHIFT and move the block forward with the cursor control keys.
The block can be shifted up/down, left/right. If a block is moved accross a 4k boundary, the highlighting will vanish to indicate the fact. The larger the data block being shifted, the longer will be the delay before any action is seen on-screen. This is due to the microprocessor shifting the data block and then updating the screen when done.

3.25 PROGRAM EPROM (PROG)

This is the EP4000 EPROM programming facility. When keyed, the machine will check the device in the ZIF socket to ensure it is blank before starting the programming cycle.
During programming data is transferred from the RAM area defined by the device selector to the EPROM (See table below).
When programming is complete, the RAM area contents are compared with the EPROM contents and the result is indicated by PASS or FAIL in the LED display.

DEVICE SELECTOR	RAM AREA TRANSFERRED	EPROM ADDRESS
2704	3400-35FF	6000-61FF
2708	3400-37FF	6000-63FF
2716(3)	3000-37FF	6000-67FF
2508/2758	3000-33FF	6000-63FF
2716/2516	3000-37FF	6000-67FF
2532	3000-3FFF	6000-6FFF
2732	3000-3FFF	6000-6FFF
EXT	depends upon external programming module	

The RAM area used for each device for programming is the same as that used for emulating that device. This means that once the program has been proved using the emulation facility, it can be transferred directly to EPROM for use by the external system independent of the EP4000.

3.26 VERIFY EPROM (VFY)

Compares the EPROM contents with it's specified RAM area. PASS will be shown on the LED display if the data is identical, else FAIL will be shown, & the EP4000 will highlight any discrepancy bytes on the page being viewed. Highlighting will only be observed if the page being viewed is in RAM corresponding to the selected EPROM. To view discepancies on further pages, the page should be called up and the EPROM verified again.

3.27 CHECKSUM EPROM (CHCK)

This will calculate a 16 bit addition (2 byte checksum) of the data in EPROM defined by the device selector. It provides a compare function without having to modify the RAM contents. The checksum is shown in the address section of the LED display (4 digits).

Example: With a master EPROM in the ZIF, key CHCK.
The checksum result is noted from the LED display.
Replace the master with a copy and key CHCK. If the resulting checksum is the same, one can be fairly confident that the contents of the 2 devices are the same. -- you cannot be absolutely sure, because a byte for byte comparison has not been made, only a bit count.

This key also doubles as a 'blank check' since the bit count of blank devices is that listed below:

Device	Checksum
2704	FE00
2708	FC00
2716(3)	F800
2508/2758	FC00
2716/2516	F800
2532	F000
2732	F000

3.28 STORE EPROM DATA (STOR)

Copies the EPROM contents to the RAM area. The addresses at which the copy is made is the same as that used for EPROM emulation and programming and depends upon the device selector. See table below:

Device	Copied to:
2704	3400-35FF
2708	3400-37FF
2716(3)	3000-37FF
2508/2758	3000-33FF
2716/2516	3000-37FF
2532	3000-3FFF
2732	3000-3FFF
EXT	Depends upon the external module being used

When the data is copied, the EP4000 will verify that the data was correctly transferred, and indicate pass/fail in the LED display.- along with discepancy byte highlighting on-screen.
NOTE: An empty ZIF socket looks like a blank EPROM, so storing an empty socket will clear the RAM area in accordance with the device selector setting.

3.29 SERIAL INPUT (SIN)

Provides a method of loading data to the RAM area of the machine from the serial input line. The input is treated in a similar way to memory areas, where the target area is defined by the TO key.

Example: Key SIN TO RAM , the machine is now set to receive data, but requests further information by showing FL in the status window. The data required is a 4 digit hex 'input word' used to define the transfer speed, port and number of data bits. The table below should be used to select the correct word to your requirement.

INPUT WORD

DIGIT 1	BAUD RATE	DIGIT 2	PORT	DIGIT 3	DIGIT 4	DATA BITS
0	110	0	Cassette	Enter 0	1	1
1	300	1	20mA Loop	(not used	2	2
2	600	2	RS232	on standard	3	3
3	1200	3	TTL	machines)	4	4
4	2400				5	5
5	4800				6	6
6	6400				7	7
7	9600				8	8

The machine expects a transfer of object code (binary data).
Each byte being sent must start with 1 start bit, no parity needed, and end with 1 or more stop bits.
The data block to be sent must be preceded by a 2 byte checksum and 2 byte block length.

START	LO	HI	LO	HI	DATA BLOCK	END

Block length Checksum

When the input word has been entered, the EP4000 will look at the serial line and wait for the first start bit. When the data has been received, the machine will do a checksum on the data and compare it with the received checksum, indicating PASS or FAIL in the LED display.
The video display is updated when the transfer is complete.

3.30 SERIAL OUTPUT (SOUT)

Allows any length data block up to 4k to be transmitted via the serial line to peripheral devices. Data is transmitted in the format: 1 start bit, 8 data bits, and 1 or more stop bits. Data is transmitted as object code (binary data).

Example: PAGE 30 TO SOUT. prepares the machine to transfer the data on page 30 to the serial output line.
The machine requests an output word to descibe the speed and number of stop bits.

OUTPUT WORD

DIGIT 1	SPEED	DIGIT 2	DIGIT 3	DIGIT 4	STOP BITS
8	110	Enter 0	Enter 0	9	1
9	300	"	"	A	2
A	600	"	"	B	3
B	1200	"	"	C	4
C	2400	"	"	D	5
D	4800	"	"	E	6
E	6400	"	"	F	7
F	9600	"	"		

No port number is required because data is transmitted from all ports simultaneously. The data block is sent after a 2 byte block length and 2 byte checksum in the same fashion as serial input. When the machine is outputting data, the LED display will blank to indicate 'busy'. When the transfer is complete the EP4000 will be in C1 mode.

3.31 PARALLEL INPUT (PIN)

Provides a high speed parallel data transfer facility.
The port is viewed by the EP4000 as a memory area, where the TO key defines it as the object area.

Example: Key PIN TO RAM. The display blanks to indicate 'busy' and is waiting for the first handshake transaction to take place.

Binary data should be set up on the Port A input lines. When the strobe signal (PB7) is taken low by the peripheral device, data will be latched onto the port and the 'input buffer Full' line (PB6) will go high to indicate data is received. When the data has been read from the port and stored at the target address, PB6 will go low to indicate that the next transaction can take place.

3.32 PARALLEL OUTPUT (POUT)

Data can be moved from a defined object area to the parallel port (which has been defined as the target by the TO key).

Example: MEM 3200 TO POUT transfers data from 3200-3FFF to the parallel port by handshake mode.

When the acknowledge (PB7) signal is low, the EP4000 will output binary data. 8 bits in parallel to port A.
It then issues an output buffer full (PB6) signal to strobe data into the peripheral device (active low). When the PB7 is taken high to indicate completion of the transfer, PB6 will also go high. When PB7 is again low, the next transfer will take place.

THE CASSETTE INTERFACE

4.00 GENERAL

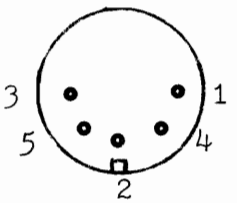
The interface provides an easy method of storing RAM data or EPROM sets onto audio cassette tape. The maximum capacity per load, or dump is 4k bytes. The encoding method used is not standard, and has been designed to eliminate, as far as possible, the effect of 'drop out', tape speed fluctuation, AVC and volume setting.
Essentially the system works by transmitting a logic 1 as a single cycle of one frequency, and a logic 0 as a single cycle of a different frequency. The incoming waveform is sampled in a similar way to bit serial transfers, except that each bit is synchronised, and not each series of bits.

4.01 CASSETTE RECORDER

Many cassette recorders have been tried with the EP4000. Some give good results, others very poor results. - We particularly recommend the ALBA models R29 and R32 as these gave excellent results and are very inexpensive (around £13).
To ensure consistent data recovery, ensure the tape heads are kept clean and use a good quality tape. Data recovery was found to be sensitive to volume control setting on all but the ALBA models. Automatic volume control (AVC) has no effect on the transmission or data recovery.

4.02 INTERCONNECTION

Connection to the cassette recorder is made from the RS232/cassette DIN socket at the rear right of the EP4000. Use a 3 or 5 pin 180deg DIN plug for connection with a 2 wire screened cable



Socket view
(pin numbers are standard DIN)

pin 1 - Input from cassette (playback)
pin 2 - 0v common
pin 3 - output to cassette (record)
pin 4 - not used on cassette - do not connect
pin 5 - " " "

NOTES:
1/ The playback line should be taken from the earpiece socket on the cassette machine, or if this is not available, direct from the internal loudspeaker.

2/ Not all recorder manufacturers follow the DIN standard socket wiring. Connection to the recorder should be checked.

3/ The record line should go to the 'phono' or 'line' socket on the recorder.

4.03 FILE NUMBER

This is a 2 digit hex code (i.e. a single byte) entered via the keypad prior to a dump or a load. It is used to identify a particular program recorded onto tape. Any code may be used except 0F (hex) which is used for AVC defeat. It follows that up to 255 files of up to 4k each may be stored on cassette (i.e. approximately 1 mega byte of data).
During the course of a load from cassette, the EP4000 will, if necessary, search the whole tape for a predefined file number, before actually loading data to RAM.
This is useful if many small program patches have been stored close together on tape, without any voice identification.

4.04 DATA FORMAT

Prior to transmission, a count is made of the number of bytes in the data block to be sent, and this is stored as a 2 byte 'block length'. Also calculated is a count of the number of bits set to logic 1 in the data, and this is saved as a 2 byte checksum. These 4 bytes are stored on the tape along with the file number and the actual data block.

Tape format

0F bytes	File	Block length	Checksum	Data block

Preceding the file number is 256 0F (hex) bytes. The tone produced is used to defeat the AVC of the recorder, before valid information is actually transmitted. These bytes are also used to synchronise data input.
During the course of a load after the correct file number has been found, the 2 byte block length is counted down to zero to indicate when the load is complete. When all the data is received, the EP4000 will sum the bits set to logic 1, and compare the result with the checksum received from the tape. A good load will be indicated in the LED display by PASS or FAIL if the checksum comparison was different.

4.05 RECORDING DATA ONTO CASSETTE

The interface is controlled from the keypad. Any memory area may be dumped onto cassette as follows:

Example: Dumping page 20 contents to cassette
Key PAGE 20 TO COUT - page 20 has been defined as the object area to be dumped to cassette. This is an EPROM monitor area and is now shown on-screen. The machine is now requesting the 2 digit file number by showing FL in the status display.

Start the cassette recording and enter the 2 digits (01 say). The display will blank to indicate 'busy' and start dumping data to the cassette - you may or may not hear the recording from your cassette (this depends on your recorder). When done, the EP4000 will be in C1 mode. Transmission of a screen page will take about 6 seconds.

4.06 LOADING DATA FROM CASSETTE

A similar procedure is adopted for program loading..

Example: Loading pre-recorded data to RAM starting at address 3200. Key CIN TO MEM 3200. The RAM address 3200 is defined by the TO key as the target start address and the cassette interface as the object area. Again the machine requests the file number by indicating FL in the status window. Key 01 (say) and start the cassette recorder on playback. The EP4000 searches the tape for the 01 file number and when found, will start to load data. Once done, PASS or FAIL will be indicated if the load was good or bad.

Some initial setting up regarding the cassette recorder volume control setting may be required. This is easily done by attempting to load a small program recorded many times. Vary the volume setting slightly for each attempted load, until the correct setting is found. Once the correct setting has been found (usually low to middle volume) the EP4000 will consistently load data reliably with no error rate.

SECTION 5

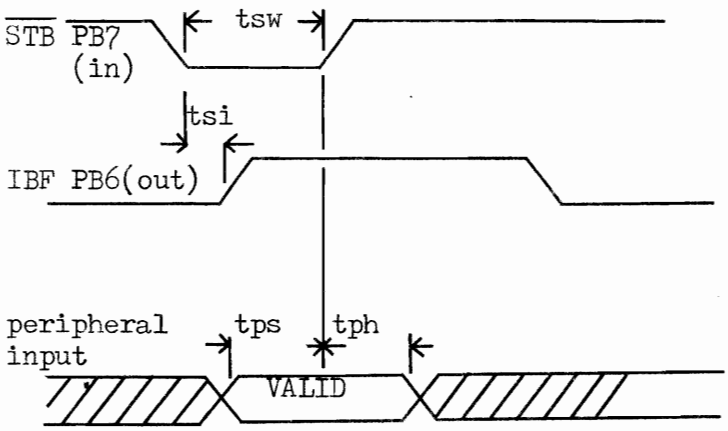
PARALLEL PORT

5.00 GENERAL

The 8 bit parallel port is avaiable from the 50 way expansion connector - See 2.07 for connection data. The monitor program written for the port will handle binary data transfers by hand-shaking. The 2 handshake bits are PB6 and PB7.
Operation of the port is controlled from the keypad.

5.01 STROBED INPUT.

This allows data to be read from a peripheral in a 2 step transaction. First the peripheral sets up data on the port, PA0 - PA7, strobes it into the input latch and notifies the microprocessor that data is ready to be read. Second, the processor reads the contents of the latch and resets the handshake signals for the next transaction to take place. The timing diagram below illustrates the procedure.



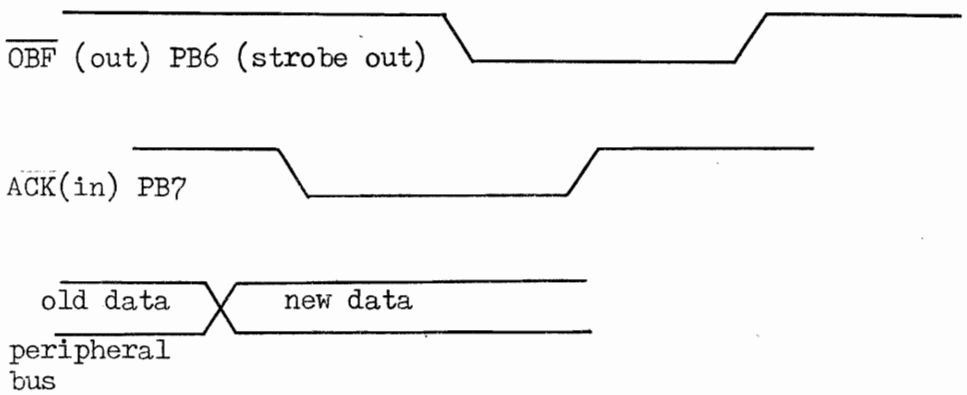
tsw	$\overline{\text{STB}}$ pulse width	300ns typical
tsi	$\overline{\text{STB}}$ to IBF delay	200ns typical
tps	data setup	150ns typical
tph	data hold	50ns typical

The strobe signal, $\overline{\text{STB}}$, is an active low signal generated by the peripheral to signify that data is valid on the trailing edge of this pulse. The strobe is connected to PB7. Data is latched into the input buffer on the trailing edge of $\overline{\text{STB}}$.
The input buffer full signal, IBF, is an output from the port controller on PB6. IBF is set by the leading edge of $\overline{\text{STB}}$ and is reset when the internal microprocessor has read the data from the port.

IBF high tells the peripheral that peripheral data is latched and waiting to be read by the EP4000. IBF goes low when the EP4000 has read the data and informs the peripheral that the next transaction can take place.

5.02 STROBED OUTPUT.

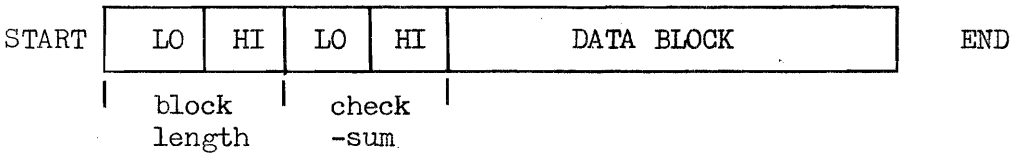
The port allows output of data to an asynchronous peripheral from the EP4000. The CPU writes data into the output latch which creates a handshake signal to strobe data into the peripheral's port.
The other signal involved in the transaction informs the EP4000 that data transfer is complete and that the next operation can take place.
The timing diagram below illustrates the process.



The output buffer full signal, $\overline{OBF}$, is an active low strobe generated by the EP4000, and should be used to strobe data into the peripherals input latch. $\overline{OBF}$ returns high when the answering acknowledge signal, ACK is also high. The ACK signal is an active high signal issued by the peripheral used to tell the EP4000 that data has been latched into the peripheral's buffer. ACK returns low when the peripheral requires the next transaction to take place.

5.03 DATA TRANSFER FORMAT

Because the software to operate the parallel port has been written for binary data transfers between the EP4000 and some external system, rather than transfer from the EP4000 and a printer (say), a simple format has been devised where information is appended to the data block being transmitted or received.
This takes the form of a 2 byte block length and a 2 byte checksum. The block length is used by the EP4000 as a counter to indicate when the transmission is complete, and the checksum is used in a comparison with a checksum performed on the inputted data block to indicate whether or not data was received correctly. - this will be shown as 'PASS' or 'FAIL' in the LED display. The format is shown below.



data transfer format

5.04 CONNECTION TO THE PORT

This can be made using a 50 way 'mass termination' socket e.g. Speedblock, blue Macs, Scotchflex etc., to connect to the 50 way expansion connector. Alternatively, MOLEX crimp sockets can be used as an inexpensive and convenient method, which allows the other pins on the expansion plug to be free for connection to other devices.
Cables should be kept away from mains lines and should not be longer than 2 feet.

5.05 ELECTRICAL CHARACTERISTICS

Parameter	Conditions	Min	Max	Units
V1H Logic 1 input voltage		2.0	Vcc+0.5	V
V1L Logic 0 input voltage		-0.5	0.8	V
VOH Logic 1 output voltage	IoH=-100uA	2.4		V
VOL Logic 0 output voltage	IoL= 2.0mA		0.4	V
IL1 Input load current	Vin=0v to 5.25v		±10	uA
ILO Output leakage current	High impedance		±10	uA

5.06 OPERATING THE PORT

Example: Loading data from the peripheral bus to the RAM starting at address 3020.
Key: PIN TO MEM 3020. This sets the port to input mode and is ready to receive data. PB7 has been configured as an input for the $\overline{\text{STB}}$ line and PB6, as an output for the IBF line.
The first 2 bytes of data strobed into the port are the block length, and this is counted down for each byte sent in the data block. When the count reaches zero, the transmission is complete, and the EP4000 ignores any further data placed on the peripheral bus. A checksum is now performed on the stored data, and this is compared with the 2 checksum bytes received from the peripheral. Identical checksums will be shown by PASS in the LED display, else FAIL if the checksums are different. The EP4000 is now in the C1 mode and ready for use.

Example: Writing data to the peripheral port. The object area of memory is the programming socket.
Key: PROM TO POUT. The EP4000 performs a checksum and block length calculation of the data in the EPROM socket.
PB6 is configured as an output and PB7 an input. The first byte (block length LO) is sent when PB7 is set low by the peripheral. Data is sent thereafter in accordance with the handshake mode described above. When all the data has been transmitted, the LED display will be restored and the machine is in C1 mode.

SECTION 6

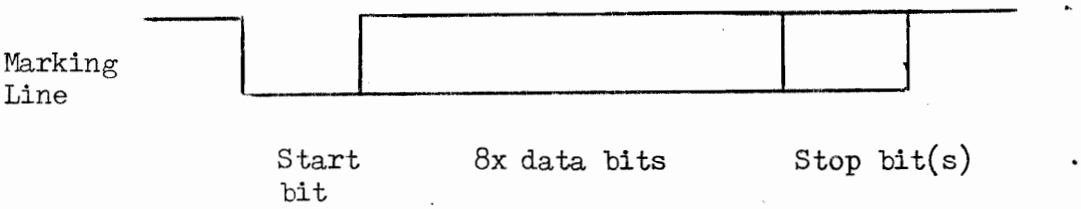
SERIAL DATA TRANSFERS

6.00 GENERAL

Serial data transfers made by the EP4000 are made in asynchronous format - i.e. no common clock is shared by the EP4000 and the communicating device. The serial data transfer program residing in the EP4000 has been written to send and receive binary data - i.e. object code of 8 bits. The machine will receive ASCII files, but it will assume that this is object code and no conversion of the ASCII data to object code will take place. Setting up for data transfers is made using the key pad to enter an input or output word to define various paramters. 3 ports are available - these are TTL, 20mA loop, and RS232.

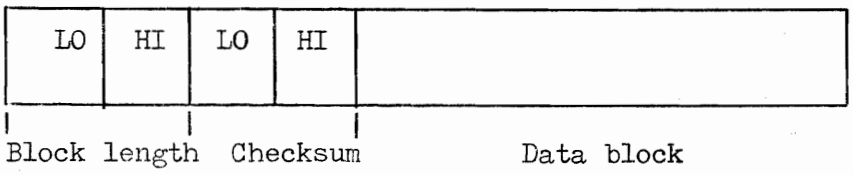
6.01 WORD FORMAT

Each data word sent is composed of 1 start bit, 8 data bits and 1 or more stop bits. Transmissions received by the EP4000 should follow the same format.



6.02 BLOCK LENGTH AND CHECKSUM

Prior to sending a data block the EP4000 will calculate the number of bytes in the block and store the 2 byte result as a block length. At the same time a 16 bit addition of all data in the block is calculated and stored as a two byte checksum.
These 4 bytes precede the data block in all bit serial transmissions. The format is shown below.



The block length and checksum may or may not be used by the receiving system. A sending device, making a transmission to the EP4000 should be arranged to append these bytes to the data block. The block length is used by the EP4000 as a counter to detect when the transmission is complete. The checksum is used to verify that the data has been properly received - since the machine performs a checksum on the stored data and compares this with the transmitted checksum. The result of the comparsion is shown by PASS or FAIL in the status display, and this also indicates the end of the data transfer.

6.03 PORTS

Three ports are available to the user to suit any particular application. All the output lines are driven by a common serial output line, so it is possible to transfer data to 3 different peripherals. The port used for serial input has to be selected to drive the serial input line to the microprocessor, and this is done via the keypad.
The 3 ports are;

1/ RS232
This port is usually used in noisy environments, or where the communicating device is remote from the EP4000, and long cable lengths are involved. The input port will accept any 'standard' RS232 level. The output provides marking (binary 1) of -5v and spacing (binary 0) of +5v, from a constant current source.

2/ TTL This is useful for communication between the EP4000 and a similar device located close to the machine. Short cables should be used, and should not be run close to mains cables. The output line is driven by a TTL compatible CMOS buffer to give marking of 4.5v and spacing of 0.3v. The output line represents an LS TTL load to the sending device.

3/ 20mA CURRENT LOOP
This provides an alternative to the RS232 port for use in noisy environments or where long cable runs are needed. The input line is opto-isolated and protected with a polarity compliance diode. The output provides 20mA to drive a similar device. Mark, logic 1 is 20mA and space, logic 0 is 0mA.

6.04 INPUT AND OUTPUT WORDS (SPEED SELECTION)

Once the EP4000 has been set up to make a serial transfer, it will request an input or output word to define the speed of transfer (baud rate), the number of stop and data bits, and the type of serial port being used for receiving. The word is entered by the keypad when the FL prompt is shown, and is a 4 digit hex entry.
The tables below show the possible combinations for selection.

INPUT WORD TABLE

Digit 1	Baud rate	Digit 2	Port select	Digit 3	Digit 4	Data bits
0	110	0	(cassette)	Enter 0	1	1
1	300	1	20mA loop	(not used)	2	2
2	600	2	RS232		3	3
3	1200	3	TTL		4	4
4	2400				5	5
5	4800				6	6
6	6400				7	7
7	9600				8	8

OUTPUT WORD TABLE

Digit 1	Baud Rate	Digit 2	Digit 3	Digit 4	Data bits	& Stop Bits
8	110	Enter 0	Enter 0	9	8	1
9	300	(not used)	(not used)	A	8	2
A	600			B	8	3
B	1200			C	8	4
C	2400			D	8	5
D	4800			E	8	6
E	6400			F	8	7
F	9600					

Example: Serial input at 1200 baud from RS 232 port, 8 data bits being sent (number of stop bits is irrelevant as these are discarded). When the FL prompt is shown, the input word entered should be 3208.

6.05 SERIAL DATA TRANSFER EXAMPLES

The TO key is used by the EP4000 to distinguish between an object memory area and a target area. The SIN key is used to define the serial port as the object 'area' and the SOUT key to define it as a target.

Example: Transfer EPROM data to the serial port. Transmission will be at 2400 baud. 1 stop bit per word.
Key RESET to set the output line to the marking condition.
Key PROM TO SOUT to set the machine for a transfer. The output word entry is requested by FL in the LED display.
Key C009 as the output word. There is a delay before the transmission begins, whilst the EP4000 calculates the blocklength and checksum bytes which will be sent before the PROM data.

Example: Receive a 2k block of data from the RS232 port and store it in the RAM. Baud rate is 1200. 8 data bits per word sent.
Key SIN TO RAM. This prepares the machine to receive from the serial port. The input word entry is requested by FL in the LED display.
Key 3208 and transmit data from the external system. The EP4000 will load the data and store it in the RAM starting at address 3000.
The first 2 bytes received are the block length - 2000 (10), 0800(H) in this example and when this has been counted down to zero, the EP4000 will ignore any further data appearing at the port - i.e. the transmission is complete. A checksum is performed on the 2k data block and this is compared with the 2nd pair of bytes received (checksum). The result of the comparison is shown as PASS or FAIL in the LED display.

Notes: 1/ In the example above, the 2k block length is 0800 in hex, and this should be sent in accordance with the format shown in 6.02. i.e. 1st byte is 00, 2nd byte is 08.

2/ Connection data for the ports is given in section 2.

3/ Although the 9600 baud rate has been included, some difficulty may be experienced in using this speed - this is due to the relation between the software timing and monostable pulse width variation.

EP4000 Instruction Manual Supplement A

Two new features have been added to the EP4000 Emulator Programmer. These are - the addition of an ASCII-Hex serial input routine and a parallel port printer output routine.

A1 ASCII-Hex Serial Input

This has been included in addition to the existing binary transfer format. The type of format required is selected using the input word (See section 6.04 in the manual).
Digit 3 has been reassigned so that a '1' selects the ASCII-Hex format, and a '0' selects the binary format.

A2 Format (based on the MOSTEK Standard output definition)

Two types of record are recognised by the EP4000. These are the data record (type 00) and the end of file record (type 01).
Each record begins with a colon as delimiter and ends with a carriage return, line feed. All information is in ASCII.

A3 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	MSB of start address
6-7	LSB of start address
8-9	ASCII zeros This is the record type for data
10-	Data bytes

Last two bytes - Checksum of all bytes, except the delimiter, carriage return, line feed. (The checksum is the negetive of the binary sum of all the bytes in the record).
CRLF Carriage return, line feed.

A4 End of File record (type 01)

Byte 1	Colon (:) delimiter
2-3	ASCII zeros
4-5	MSB of transfer address
6-7	LSB of transfer address
8-9	Record type 01
10-11	Checksum
CRLF	Carriage return, line feed.

A5 Notes

1/ Bytes 4-5, 6-7 in both record types are not used by the EP4000 in any way, but must be sent as part of the transfer
2/ The start address for the transfer is defined by the user with the keypad. Data records are assumed to contain contiguous data blocks. (They will be loaded one after the other after the start address).
3/ The data transfer is ended when the end of file record is received.
4/ If the checksum calculation fails at any time during a transfer, the EP4000 will abort and show FAIL in the LED display. When all data has been properly received, including the end of file record, the EP4000 will display PASS.
5/ The maximum speed of transfer using the ASCII-Hex format is 2400 baud.

A6 Example of use

1/ Data in a development system is to be sent to the EP4000 for programming into a 2716 EPROM.
Speed is 1200 baud, using the ASCII-Hex format
Key: SIN TO RAM sets up the system for transfer from the serial port to the RAM starting at address 3000.
The EP4000 requests 'FL' in the status display for the serial input word.

Key: 3218

```
|||└---- 8 data bits  1 start bit
||└----- ASCII-Hex format
|└------ Port 2 -  RS232
└------- Speed  1200 baud
```

Data has now been loaded to the correct area (3000-37FF) for transfer to a 2716 EPROM using the 'PROG' key.

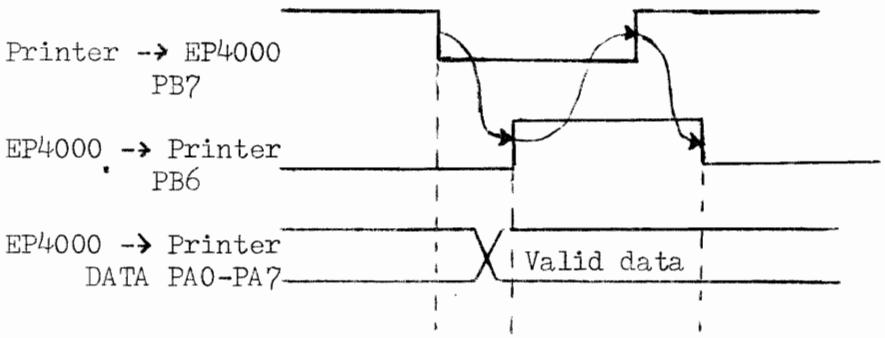
A7 Printer Output

This works with the 'PRINT' key in the same way as the other function keys-
e.g. PRINT RAM
PRINT PROM
PRINT BLOCK
PRINT PAGE 31 (etc)
The print-out appears in the same format as the video display.

A8 Printer Interface Specification

1/ The routine has been written to run printers with a parallel interface See section 2.07 of the manual for connector data). Handshake signals are dircetly compatible with the SHARP MZ-80P3 printer, whilst other printers may need some signal inversion.

2/ Signal Timing:



PB7 is set low by the printer when it is ready to receive data.
The EP4000 then sets up valid data on the peripheral port and sets PB6 high (strobe) until PB7 is set high by the printer to indicate data has been received. The handshake lines are now ready for the next transaction.
The printer need only recognise the upper case ASCII Characters 0-9 , A-F and the carriage return and space characters.

SECTION 7

EPROM EMULATION

7.00 GENERAL

The principle of EPROM Emulation is that an EPROM device, currently sitting in the addressing space of a system may be directly replaced by a machine, which looks to that system, exactly like an EPROM. The difference between the actual device and the emulator is that data can be easily, and quickly written to the machine, from a wide variety of sources. This means that quick program changes can be written, entered from the keypad, serial or parallel port, and run in real time, at full speed, without the lengthy process of EPROM programming and erasing.
To illustrate this process, several application examples are given below:

7.01 EPROM EMULATION - EXAMPLES

Example 1: To produce a video circuit for the display of 200 different alpha-numeric characters. Character frequency is to be 1MHz, so a 2716 EPROM could be used as the character generator. No development system other than the EP4000 is available.

Once the hardware has been built, it can be easily debugged. All that is needed now is the character generator, residing in a 2716 EPROM.

Procedure: Knowing the inter-device connections of the hardware, code can be written for the character generator to produce the required characters. Enter the code into the EP4000 RAM, dial 2716 on the selector and plug the machine into the vacant socket where the character generator will eventually sit. Keying 'DMA' will now let the external hardware access the EP4000's RAM, clocking data as required. The video output can now be viewed on a monitor. If any mistakes are seen - i.e. a character not properly formed, 'DMA' can be keyed again to isolate the EP4000 from the video circuit, so that a modification can be made to the data at those addresses which affect this character. Now switch back to emulate by depressing the 'DMA' to observe the effect of the modification on the character formation. The procedure can be repeated as many times as required until all the characters are properly formed. Finally the code is transferred from the EP4000's RAM to a blank 2716 (see EPROM programming section), for use by the video circuit.
The programmed EPROM can now be placed into the video circuit, enabling the circuit to work correctly by itself.

Example 2: A commercial video game has to be modified to take account of a new jackpot. The program dealing with the jackpot pay-out is located inside the machine and resides in a 2732 EPROM.

Procedure:

1/ Dial 2732 on the EP4000 device selector.
2/ Remove the 2732 from the games machine, and plug the EP4000 into the empty socket.
3/ Put the existing 2732 into the programming socket and copy it's contents into the EP4000's RAM.
4/ Locate the program area dealing with the jackpot, and modify it as required.
5/ Emulate the EPROM by depressing the 'DMA' key, and let the games machine run the program.
6/ Modify the program if required by depressing the 'DMA' key again and making any further changes. Confirm correct operation by further emulation, depressing the 'DMA' key.
7/ Program a new 2732 with the corrected program, and substitute it for the EP4000.

The example illustrates the means by which existing programs can easily be modified or upgraded. If an emulator were not available the process would be very time consuming, since repeated EPROM programming and erasing would be required, with a consequent lack of confidence by the program writer in his ability to get, even the simplest modifications done correctly.
The EP4000 eliminates these problems, because small changes to the total modification can be made quickly, and run in real time, before moving onto the next small change etc., until the whole modification has been made.

Example 3: The problem is to design a controller for an industrial production machine. The hardware has been built and the control program is to reside in a 2708 EPROM.

Procedure: Write the program in one of 3 ways -
1/ Write the program on a development system in a high level language (if the controller has an interpreter). Create an object code file and download to the EP4000.
2/ Write the program in assembly language on a development system. Assemble the object code and download to the EP4000.
3/ Write the program directly in object code, and key into the EP4000. (This may be most appropriate, since the program is only 1K bytes long.)

Once the program has been entered to the EP4000, it is now ready to emulate the EPROM. Communication between the EP4000 and the development system or program writer takes place until all the bugs have been eliminated. If program writing method (3) is adopted, it is usual to write small program segments, and subroutines, running and modifying each part until they are working, and then transferring them to EPROM at each step, until the EPROM contains the full working control program.

7.02 EMULATION PROCEDURE

The emulation plug at the rear of the machine gives direct access to the EP4000 RAM via line driver/buffers. These devices are not protected in any way in order to achieve the best possible access time of the emulator. For this reason, use a recommended simulator cable, either a SSC (Standard simulator cable), a buffered simulator cable, or a MESA4 multi-EPROM simulator adaptor.
Also in order to avoid damage to the buffers, particularly the data buffers, ensure the external hardware has been thoroughly debugged and that no short circuits exist on the buses.

Adopt the following procedure:

1/ Plug the simulator cable into the EP4000 26 way plug (cable down) - Ensure proper connection - the arrow on the cable socket corresponds to pin 1 on the DIL plug and should be under the socket when plugged into the EP4000, so that the cable feeds downwards from the plug.

2/ Plug the DIL plug into the host system in an EPROM socket. Pin 1 is indicated by an arrow or '1'.

3/ Dial the required EPROM type on the device selector - the EP4000 will configure the address and chip select lines so that it'looks'like the slected device.

4/ Remember - the external system has access to the EP4000 RAM only after DMA has been keyed - prior to this the host's EPROM socket is effectively empty. Hold the external microprocessor reset until it is allowed access to the EP4000. The RAM can be isolated at any time from the host system by keying 'DMA' a second time.

7.03 EMULATION CHARACTERISTICS

The table below compares the EP4000 emulation characteristics with those of a typical EPROM -- a 2716.

NOTE: 1/ No power is supplied by the EP4000 from the +5v rail when emulating single rail devices, nor the +5v, -5v or +12v rails when emulating 3 rail devices.
2/ No current is taken from the host +5v rail at any time.
3/ When emulating 3 rail devices, 20mA is taken from the -5v and +12v lines from the host system. These lines are used as references by the EP4000.
4/ The EPROM emulation facility is READ only - do not attempt to 'PROGRAM' the EP4000 !

COMPARISON BETWEEN 2716 AND EP4000 CHARACTERISTICS

READ OPERATION

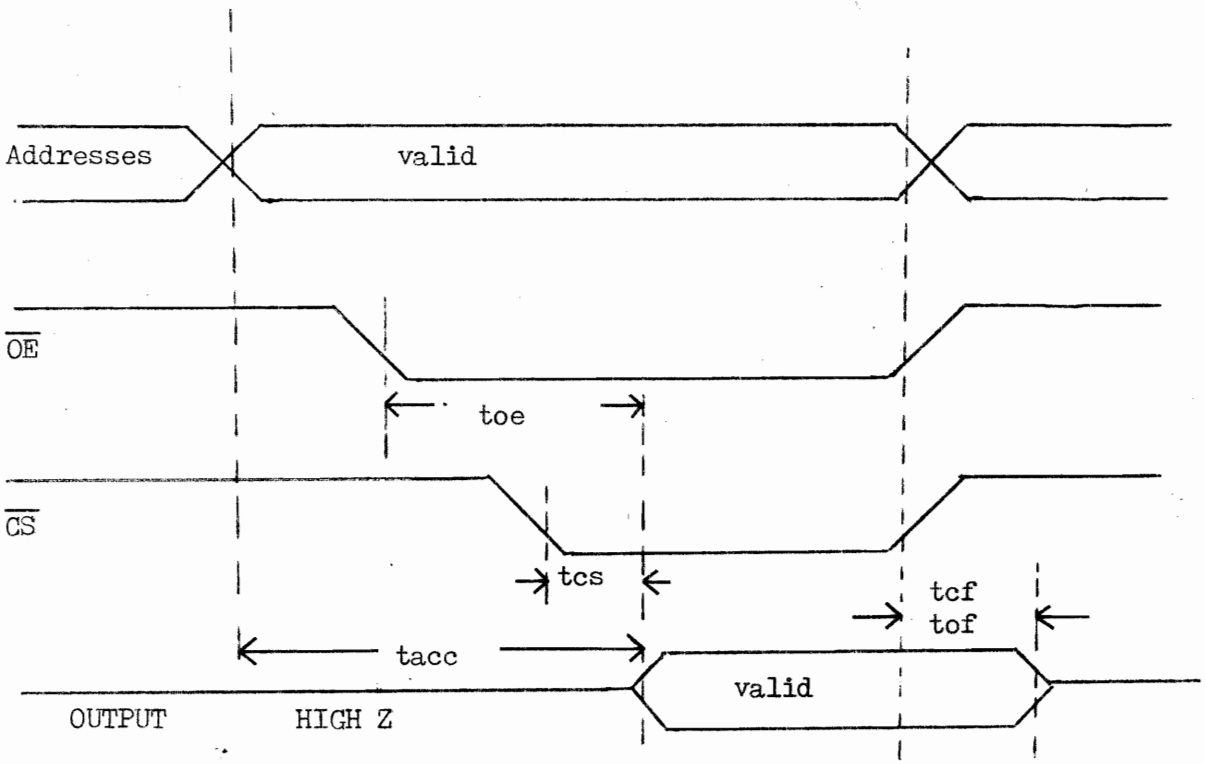
DC & Operating Characteristics

Parameter	Symbol	2716 Limits Min	Typ	Max	EP4000 limits Min	Typ	Max	Unit	Conditions
Input load current	ILI			10			LSTTL	uA	
Output leakage current	ILO			±10			±20	uA	Vout= 5.25/0.45v
Vpp current	Ipp			5			0.02	mA	Vpp = 5.85v
Vcc current	Icc			100			0	mA	$\overline{OE}$ = $\overline{CS}$ = VIL
Input low voltage	VIL	-0.1		0.8			0.8	v	
Input high voltage	VIH	2.2		Vcc+1	2			v	
Output low voltage	VOL			0.45			0.4	v	IOL = 2.1mA
Output high voltage	VOH	2.4			2.4	3.4		v	IOH = -400uA

AC CHARACTERISTICS

Address to output delay	tacc		250	450			300	ns	$\overline{OE}$ = $\overline{CS}$ = VIL
$\overline{OE}$ to output delay (Pin 18)	toe		280	450			70	ns	$\overline{CS}$ = VIL
$\overline{CS}$ to output delay (Pin 20)	tcs			120			70	ns	$\overline{OE}$ = VIL
$\overline{OE}$ to output float	tof	0		100	0		30	ns	$\overline{CS}$ = VIL
Chip deselect to float	tcf	0		100	0		30	ns	$\overline{OE}$ = VIL
Address to output hold	tah	0			0		20	ns	$\overline{OE}$ = $\overline{CS}$ = VIL

READ TIMING WAVEFORMS 2716 / EP4000



SECTION 8

EPROM PROGRAMMING AND ERASING

8.00 EPROM ERASING

All the EPROMs handled by the EP4000 are erased by exposure to short-wave ultra violet light - 253.7nm. There is no way to erase a part of the bit pattern, and after eraser all bits are set to logic 1.
The recommended integrated dose is 15 W.sec/cm^2.
There is no absolute rule for erasing time and the erasing equipment should be calibrated as follows:

1/ Program a device so that all bits are set low.
2/ Erase the device for 1 minute, then 'blank check'.If not blank, continue to erase for a further minute, repeating until blank.
3/ Over erase the device by a factor of two. i.e. if the device appears erased after 5 minutes, then continue to erase for a further 10 minutes for a total of 15 minutes.
4/ Repeat the calibration procedure every 4 weeks because the ultra-violet source will age, and it's light intensity will reduce with time.

Before erasing devices, always remove gummed labels and clean the window with acetone, alcohol or methylated spirit. The gum from labels and greasy finger prints are opaque to UV light and affect the erasing characteristics of the device.

Note: 1/ The same device type from various manufacturers seem to require different doses for erasing. We recommend the use of UV140 or UV141 erasers, and an erase time of 30 minutes.

2/ To prevent unintentional erasure by sunlight, or fluorescent lighting, cover the window with opaque tape or a label. Direct sunlight can erase devices in 1 week, and fluorescent lighting within 3 years.

3/ Improperly erased devices - i.e. devices which appear to be erased, but have not received the required integrated dose, when programmed, will have long access times, and will be prone to 'bit setting' with consequent unreliable operation.

8.01 EPROM PROGRAMMING GENERAL

Initially, and after each erasure, all bits in the device are set to logic 1. Information is introduced by selectively programming 0's into the required bit locations. Programmed 0's can only be changed to 1's by erasure.
The EP4000 is capable of programming two types of EPROM -
1/ The '3 rail' devices, 2704, 2708, TMS 2716.
The programming requirement for these devices is that the program enable line be taken to +12v, then after data and address are set-up, a 25v pulse is applied to the programming pin. The procedure is applied to all addresses on the chip, and then repeated 100 times.

2/ The 'single rail' devices 2508, 2758, 2516, 2716, 2532, 2732. These devices require the program enable line at +25v, then a 50ms TTL level pulse applied to the programming pin, after data and address have been set up. These devices require only one pulse per address so any address can be programmed in sequence, or at random.

The differences between devices within the 2 families affect only the high order address line, chip slect and program enable. The differences between devices are shown below:

Pin	Mode	2704	2708	2716(3)	2508/2758	2516/2716	2532	2732
21	READ	-5v	-5v	-5v	+5v	+5v	+5v	A11
21	PROG	-5v	-5v	-5v	+25v	+25v	+25v	A11
20	READ	$\overline{CS}$	$\overline{CS}$	A10	$\overline{OE}$	$\overline{OE}$	$\overline{CS}$	$\overline{OE}$
20	PROG	+12v	+12v	A10	+5v	+5v	⊔ 50ms	+25v
19	READ	+12v	+12v	+12v	0v	A10	A10	A10
19	PROG	+12v	+12v	+12v	0v	A10	A10	A10
18	READ	0v	0v	$\overline{CS}$	$\overline{CS}$	$\overline{CS}$	A11	$\overline{CS}$
18	PROG	⊓ 1ms 25v	⊓ 1ms 25v	⊓ 1ms 25v	⊓ 50ms	⊓ 50ms	A11	⊔ 50ms

Note (1) Be careful of the difference between the 2 types of 2716 One is single rail, the other three rail, and these depend upon the manufacturers (as a general rule, the 3 rail types are prefixed by TMS - i.e. TMS 2716 is a three rail type).

(2) The 2732 and 2532 (4k) devices are not pin compatible in either the read or program mode.

8.02 PROGRAMMING WITH THE EP4000

There are 2 ways of transferring data from the EP4000 RAM to the programming socket - i.e. programming EPROMs.

1/ Standard program.
This uses the PROG key to initiate the required programming cycle as indicated by the device selector. In this mode, the emulation RAM area contents for the particular device are transferred to the ZIF.

Data is transferred in accordance with the table below:

device	RAM address	TO	EPROM address
2704	3400 - 35FF		6000 - 61FF
2708	3400 - 37FF		6000 - 63FF
2716(3)	3000 - 37FF		6000 - 67FF
2598/2758	3000 - 33FF		6000 - 63FF
2516/2716	3000 - 37FF		6000 - 67FF
2532	3000 - 3FFF		6000 - 6FFF
2732	3000 - 3FFF		6000 - 6FFF
EXT	Depends upon external programming adaptor.		

Prior to programming the device, the EP4000 will first check if it is erased by performing a blank check. If the device is not blank, this will quickly be shown by FAIL in the LED display. If the device passes the check, then the programming cycle is started. When programming is complete, the RAM data and the EPROM data is compared and the result is shown in the LED display as PASS or FAIL.

Example 1: Programming a 2716 with RAM data at 3000 - 37FF. Dial 2716 on the selector and key RESET to power down the ZIF. Insert a blank device and key PROG. The ZIF will power up, and after the initial blank check, programming will commence. After the 100 second programming time, the EP4000 will verify the device, and show PASS if the device is correct, or FAIL if there are any errors.

Example 2: Transferring the data in 2 x 2716 into a single 2732.
Procedure: 1/ Dial 2716 on the selector and place the 2716 whose data will reside in the high part of the 2732 into the ZIF.
2/ Key STOR to copy the device into the RAM at 3000 - 37FF. A correct data copy is confirmed by PASS in the LED display.
3/ Move the cursor to address 3000 by keying MEM 3000.
4/ Copy the data from 3000 - 37FF to 3800 - 3FFF by keying RAM TO MEM3800. (this is a special key sequence which works because an exact copy is being made in each half of the RAM).
5/ Now replace the 2716 with the device whose data is to sit in the lower 2k of the 2732, and key STOR. This transfers the EPROM contents to 3000 - 37FF and verifies a correct transfer.
6/ The data has now been assembled into the RAM and can now be transferred to a 2732 device.
7/ Remove the 2716, dial 2732 on the selector and key RESET. Insert a blank 2732 and key PROG to start the programming sequence.

2/ Block program.
The second method of programming is called 'block programming' and this allows selected areas of the device to be programmed.
This means that any block or memory area can be transferred to any address in a blank or partially programmed device.

Example: Transfer page 00 data to a 2716 device. Target starting address is 6321.
Key PAGE 00 TO MEM6321. This defines the data on page 00 as the object data, and transfers this via the programming cycle to the EPROM starting at address 6321.
No blank check is performed on the device, but at the end of the cycle, the block data is verified for correct transfer, and indicated by PASS or FAIL if different.

APPENDIX A ACCESSORIES FOR THE EP4000

1/ EPROM ERASERS

Model UV141. Erases up to 14 devices simultaneously. Slide tray loading of devices. Electronic timer 5 - 50 minutes.
Auto shut off safety feature. EX-STOCK

Model UV140, similar to UV141 but without timer EX-STOCK

2/ SIMULATOR CABLES

SSC - Standard simulator cable 18 inch length Use for direct connection between EP4000 emulation plug and 24 pin EPROM socket EX-STOCK

BSC - Buffered simulator cable Extended cable length of 4 feet, buffered at user end. Direct connection to EP4000 EX-STOCK

MESA 4 - Multi EPROM Simulator adaptor. Direct connection to the EP4000 to allow it to emulate up to 4 x 2708/2508 or 2 x 2716 EX-STOCK

3/ PROGRAMMING ADAPTORS

SA - 2732A 24 pin module plugs straight into the EP4000 programming socket to program the high speed HMOS 2732A EPROM from INTEL EX-STOCK

SA - 2764 plugs into the EP4000 ZIF socket to program the INTEL/ FUJITSU/MITSUBISHI 2764 EPROM in 2 x 4k blocks EX-STOCK

SA - 2564 programming adaptor for the 2564 EPROM from TEXAS Instruments. Plugs into the EP4000 ZIF socket. EX-STOCK

4/ BIPOLAR PROM PROGRAMMING MODULES

BP4 (TEXAS) - Plugs into the EP4000 expansion plug to allow programming of TEXAS Instruments series 14 & 18 Bipolar PROMs :
74S188 (TBP18SA030), 74S288 (TBP18S030), 74S287 (TBP14S10), 74S387 (TBP14SA10), 74S470 (TBPSA22), 74S471 (TBP18S22), 74S472 (TBP18S42), 74S473 (TBP18SA42), 74S474 (TBP18S46), 74S475 (TBP18SA46) EX-STOCK

BP4 (SIGNETICS) - The next module in the BP4 range covers Bipolar PROMs from Signetics Available JUNE 83

5/ VIDEO MONITOR

CM12 - 12inch green screen monitor, 20MHz bandwidth.
External controls - horizontal width, linearity, frequency and phase: Focus, vertical height and linearity and black level.

6/ GR1 PRINTER INTERFACE CABLE

for connection between the EP4000 50 way expansion connector and a Centronics type printer

APPENDIX B EP4000 MEMORY MAP ADDRESSING

The microprocessor used to control the function of the EP4000 is the INS8060. Commonly called the SC/MP. Its addressing space of 64k is divided into fields of 4k, and the pointer reference instructions will only operate within a 4k field. This has proved useful in the EP4000 because it means that if the cursor is in a field, it cannot escape unless an address or page is assigned with the MEM or PAGE keys., This means that once in a field, continuous cursor movement will eventually bring the cursor back to the starting point.

Example: Call address 3000 using the MEM key. Move the cursor back one space: the address shown is 3FFF which is the end of the 4k field.

EP4000 MEMORY MAP

All four high order address lines have been decoded from the data bus, and are taken to the expansion connector for use by external programming modules and peripherals. Only 32k of memory is decoded within the machine, which leaves a further 32k for use by external modules.
Memory decoding is arranged as follows:

ADDRESS	DEVICE
0000 - 07FF	Monitor 1 in 2716
0800 - 0FFF	redundant
1000 - 17FF	Monitor 2 in 2716
1800 - 1FFF	redundant
2000 - 27FF	Monitor 3 in 2716
2800 - 2FFF	redundant
3000 - 3FFF	Main RAM area.
4000 - 4FFF	PAD1
5000 - 53FF	redundant
5400 - 57FF	Video RAM
5800 - 5BFF	Device configurator
5C00 - 5FFF	Control RAM and parallel port
6000 - 6FFF	ZIF socket
7000 - 73FF	PAD2
7400 - 7FFF	redundant

The ZIF socket addresses used by the monitor depend upon the device selected, although all addresses are accessible. These are as follows:

DEVICE	ADDRESS
2704	6000 - 61FF
2708	6000 - 63FF
2716(3)	6000 - 67FF
2508/2758	6000 - 63FF
2516/2716	6000 - 67FF
2532	6000 - 6FFF
2732	6000 - 6FFF

BP 4 - TEXAS Bipolar PROM Programming Module for the EP4000

General

This is the first in a series of modules designed to program BIPOLAR PROMs. Since all manufacturers' PROMs program differently, each module will program most devices from one manufacturer. The TEXAS-BP4 will program all 11 devices listed below (under TEXAS), but will also read equivalent devices from other manufacturers, so allowing copies to be made using the TEXAS Instruments equivalent device.

Operation

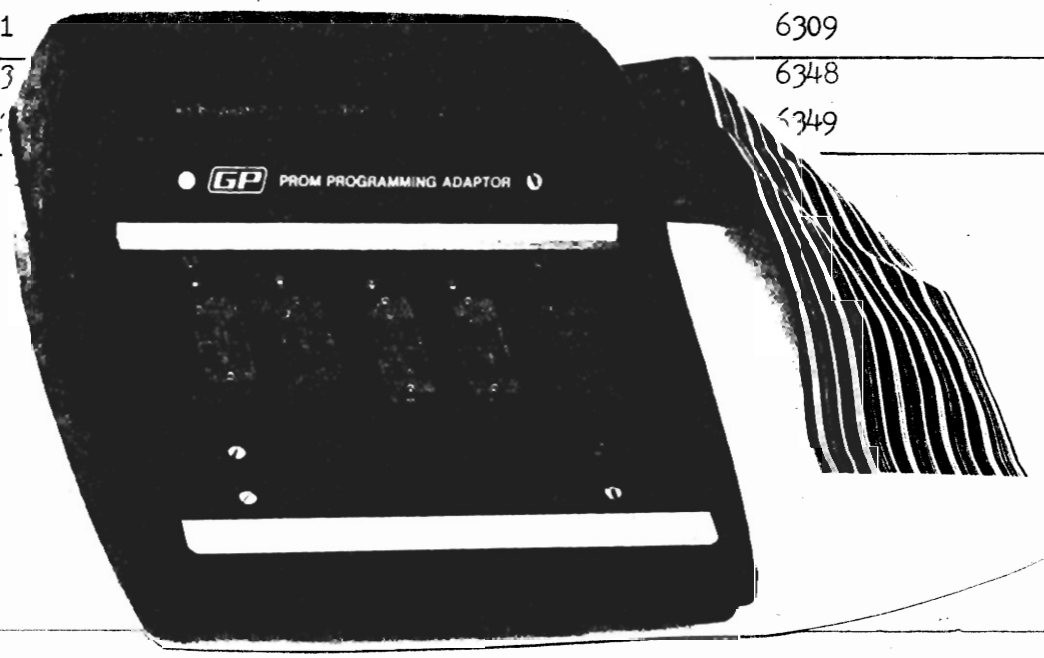
The module plugs straight into the EP4000 expansion connector at the rear of the machine. It has four panel mounted zero insertion force sockets to accomodate the different sizes of PROMs (16 to 24 pins).
All functions are controlled by the EP4000 and it's keypad.

Functions

The module allows:
1/ Device type selection - to set up the required ZIF socket for use.
2/ 'STORE'- to copy the selected device to the EP4000 RAM.
3/ 'VERIFY'- to compare stored RAM data with the PROM data.
4/ 'PROGRAM'- to program then verify the device with RAM data.
5/ 'BLANK CHECK'- to check the devices fuses are intact.
6/ All EP4000 functions can be used in the usual way.

BIPOLAR PROM Cross Reference

Size	TEXAS	NATIONAL	FAIRCHILD	HARRIS	INTEL	INTERSIL	SIGNETICS	M.M.I.	AMD
32x8	74188A	DM8577		HM1-7602		IM5600	8223	6330	AM27S08
	74S188			HM1-8256			82S23		
	74S288	DM8578		HM1-7603		IM5610	82S123	6331	AM27S09
256x4	74S387	DM74S387	93417	HM1-7610	3601-1	IM5603	82S126	6300	AM27S10
	74S287	DM74S287	93427	HM1-7611	3621	IM5623	82S129	6301	AM27S11
512x4		DM74S570	93436	HM1-7620	3602	IM5604	82S130	6305	
		DM74S571	93446	HM1-7621	3622	IM5624	82S131	6306	
512x8	74S474	DM87S295	93438	HM1-7640	3604	IM5605	82S140	6340	
	74S475	DM87S296	93448	HM1-7641	3624	IM5625	82S141	6341	
256x8	74S470	DM74S470						6308	
	74S471	DM74S471						6309	
512x8	74S473	DM74S473						6348	
	74S472	DM74S47						6349	



2732A, 2764, 2564 PROGRAMMING ADAPTOR SERIES FOR THE EP4000 EPROM PROGRAMMER

General

These three adaptors are designed to enable the EP4000 to program the 2732A, 2764 and 2564 EPROMs. They each plug straignt into the 24 pin EP4000 programming socket and provide the necessary pin-out and programming requirement for their respective EPROM.
Normal operation of the EP4000 is not affected in any way.

SA2732A

This carries a 24 pin zero insertion force socket (ZIF) to program the high speed HMOS 2732A from INTEL. This EPROM programs in the same way as a standard 2732 but requires a programming voltage (Vpp) of 21v +/- 0.5v.
The EP4000 device selector should be set to 2732.

SA2764

This adaptor box carries a 28 pin ZIF for the INTEL 2764 and its equivalent (Fujitsu, Mitsubishi etc). Since the EP4000 has 4k x 8 of RAM, the 2764 (8k x 8) has to be programmed in two halves. A DIL switch is also mounted on the adaptor and this is used for 4k block selection - i.e. A12 logic high or low.
Set the EP4000 device selector to 2732. NB. This adaptor is not suitable for MOSTEK MK2764 since this device is different in both program and read modes.

SA2564

This is similar to the SA2764 but is designed for the 2564 EPROM from TEXAS Instruments. Set the EP4000 device selector to 2532.

Manufactured by:
GP Industrial Electronics Ltd.,
Unit E, Huxley Close,
Newnham Industrial Estate,
PLYMOUTH PL7 4JN.

Tel: Plymouth (0752) 332961 (2 lines)
Telex: 42513 SHARET

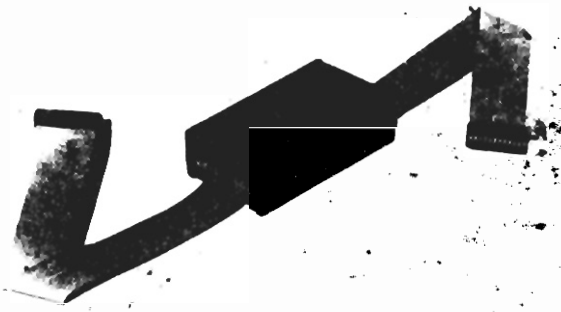
RECOMMENDED SIMULATOR CABLES FOR THE EP4000 Ref: SC42

SSC - Standard simulator cable

This is an 18inch flat cable terminated at one end with a 24 pin DIL plug for connection to the host system EPROM socket. The other end is terminated in a 26 pin socket for direct connection to the EP4000 simulator plug. This cable is suitable for use in systems where the EPROM buses are well buffered.

BSC - Buffered simulator cable

This comprises an extended length cable of 4 feet, a pod containing the buffers and device configuator, and a short cable (6 inch) terminated in a 24 pin DIL plug for connection to the host system. It has been designed to provide a more convenient length of cable, reduce the loading on the host system buses, and protect the EP4000 without significantly increasing the emulation access time.



MESA 4 - Multi EPROM Simulator adaptor

The MESA 4 has been designed to increase the flexibility of the EP4000 by allowing it to emulate more than one device at a time. Two 2716's, or up to four 2708/2508 devices can be emulated. The MESA 4 is not suitable for use with 16 bit processors where 2 EPROMs are accessed simultaneously, since it is designed to emulate multiple devices sharing the same address and data buses. Supplied complete with all the cables needed to connect up to 4 host system sockets, and cable to connect the MESA 4 pod to the EP4000.