SECTION 5

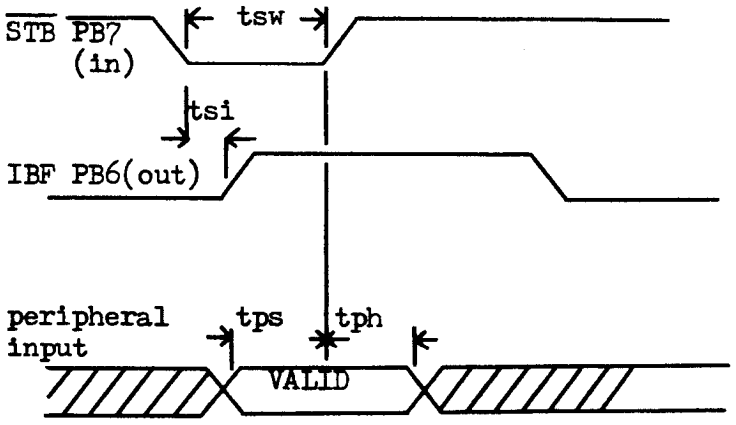
PARALLEL PORT

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.



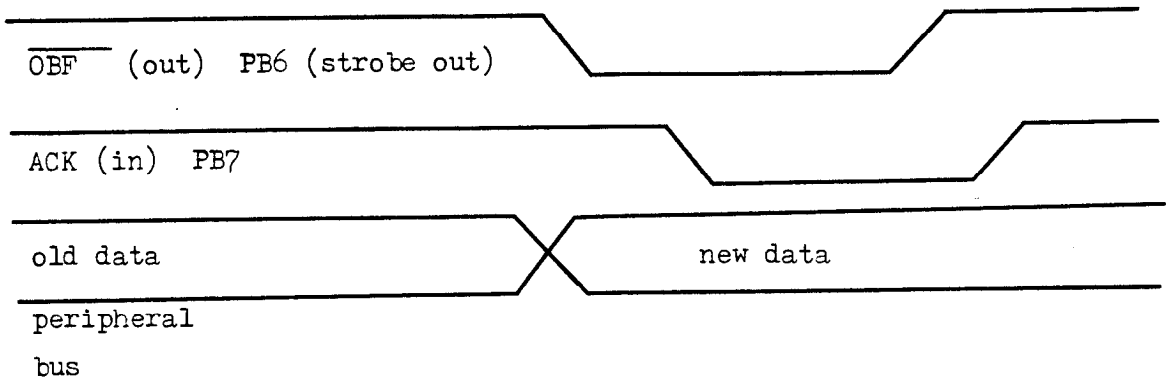
tsw	$\overline{\text{STB}}$ pulse width	300ns typical
tsi	$\overline{\text{STB}}$ to IBF delay	200ns typical
tps	data setup	150ns typical
tph	data hold	50ns typical

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 EP8000 IBF goes low when the EP8000 has read the data and informs the peripheral that the next transaction can take place.

5.02 STROBED OUTPUT.

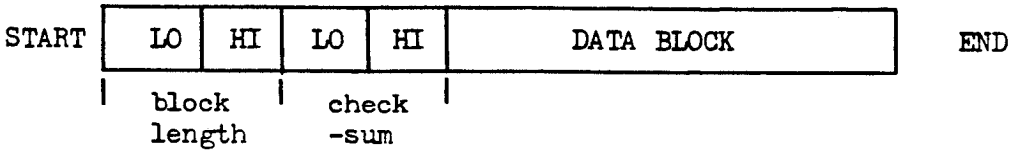
The port allows output of data to an asynchronous peripheral from the EP8000 The CPU writes data into the output latch which creates a handshake signal to strobe data into the peripheral's port.
The other signal involved in the transaction informs the EP8000 that data transfer is complete and that the 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 EP8000 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 EP8000 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 EP8000 and some external system, rather than transfer from the EP8000 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 EP8000 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 EP8000 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 EP8000 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 EP8000 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.

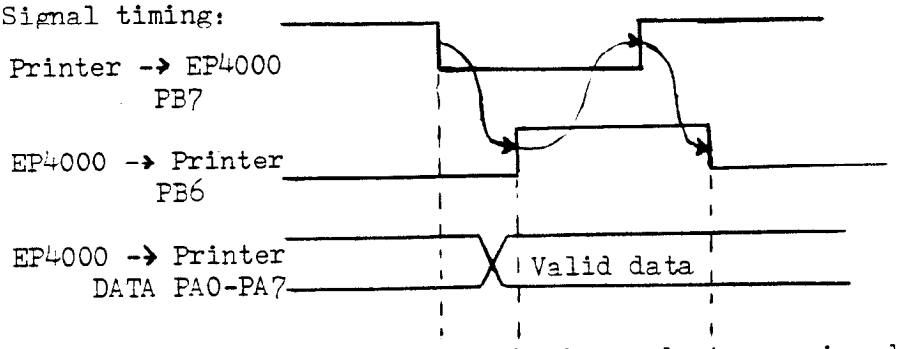
5.07 The Printer Interface

By using the GP1 interface the EP8000 may be interfaced to a Centronics type printer. The data is transmitted as groups of sixteen bytes, preceded by the address of the first byte in the group.

Printer Interface Specification

1/ The routine has been written to run printers with a parallel interface. See section 2.08 of the manual for connector data. Handshake signals are compatible with centronics type printers, using the GR1 interface.

2/ Signal timing:



PB7 is set low by the printer when it is ready to receive data.

The EP8000 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 6

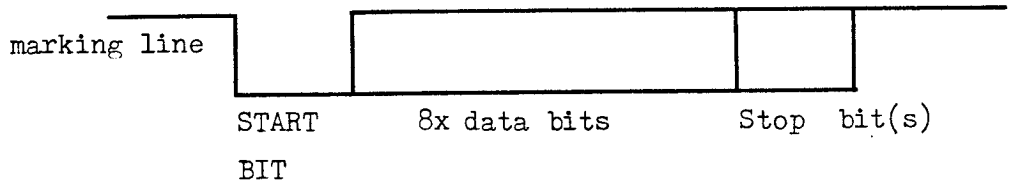
Serial Data Transfers

6.00 General

Serial data transfers made by the EP8000 are made in asynchronous format - i.e. no common clock is shared by the EP8000 and the communicating device. There are several formats of data transfer catered for by the EP8000's internal program. These formats are described separately. Setting up for data transfers is made using the keypad to enter an input or output word to define various parameters. 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, but no parity. Transmissions received by the EP8000 should follow the same format.



6.02 GP Binary Format

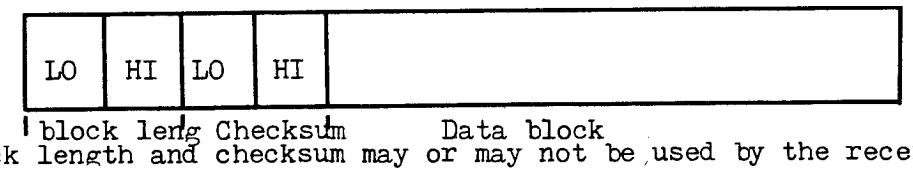
This format is designed to transmit and receive binary data i.e. object code of 8 bits. The machine will receive ASCII files, but it will assume that this is object code and no conversion of ASCII data to object code will take place.

6.03 Block Length & Checksum - - in GP Binary

Prior to sending a data block the EP8000 will calculate the number of bytes in the block and store the 2 byte result as a block length. At the same time, a a sum is made of all the bytes in the data block, and this is stored as a two byte checksum.

These four 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 EP8000 should be arranged to append these bytes to the data block.

The block length is used by the EP8000 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 comparison is shown by PASS or FAIL in the status display, and this also indicates the end of the data transfer.

6.04

Serial Data Transfer Examples using GP Binary

The TO key is used by the EP8000 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 EP8000 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 3108 and transmit data from the external system. The EP8000 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 EP8000 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.

6.05 OFFSET ADDRESSES in the Motorola/Intel Serial Formats

Definition of Offset Addresses

In order to load data into the RAM area of the EP8000 when using INTEL/ Motorola formats, it is necessary to specify an offset address. That is, an address which the machine will subtract from the block start address before adding the base address of the RAM. The offset address is specified in the following way : To do a serial transfer at 2400 baud in INTEL ASCII hex format :

TYPE FN SIN TO FN RAM

The machine will reply with FL

TYPE 4 1 1 8 —— 8 bits

2400 baud RS232 INTEL ASCII

The machine will reply AD

TYPE The offset address

Example: To store a program ORG'ED at 7000H in the start the RAM.

In reply to AD type 7000

If it was wished to store the program at 3800H for example, the offset would be given as 6800H.

To decide where to store the data the EP8000 carries out the following operation;

START ADDRESS OF BLOCK		7000H
- O.S. ADDRESS		6800H
	=	0800H
+ RAM START ADDRESS		3000H
ACTUAL LOAD ADDRESS		3800H

NOTE: If the load address does not lie between 3000H and 4FFFH, the data will not be loaded and no error message will be produced.

6.06 INTEL ASCII Hex Format

The format is based upon the MOSTEK standard output definition. The EP8000 recognises two types of record. These are the data record (type 00) and the end of file record type 01.

6.07 Data Record Format (type 00)

Byte	Contents
1	Colon (:) delimiter
2-3	Number of binary bytes in the record. Maximum 32 binary bytes (64 ASCII bytes)
4-5	MSB of start address
6-7	LSB of start address
8-9	ASCII zeros (Record type)
10	Data bytes
Last two bytes	Checksum of all bytes, except the delimiter, carriage returns, line feed. (The checksum is the two's complement of the binary sum of all the bytes in the record).

6.08 End of File Record (type 01)

Byte	Contents
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.

6.09 INTEL ASCII O/P

When the EP8000 outputs data in INTEL HEX format, the data records are 16 binary bytes long. The first start address is entered by the user in response to the prompt Ad on the LED display. All subsequent start addresses are assumed to be sequential to this and calculated as such. The transfer address is the address first entered by the user.

Example: Transfer contents of RAM to external machine at 2400 baud with 1 stop bit. Reset the EP8000, set up the receiving machine to expect data. Then key RAM TO SOUT. The LED display will prompt with FL. This indicates that it requires set up data. The user then keys C099. The EP8000 will then prompt again, this time with Ad indicating that it requires a start address (4 digits). Once this is done, the display will blank, indiacting that the machine is transferring data.

6.10 Notes

1/ The data transfer is ended when the end of file record is received.

2/ If the checksum calculation fails at any time during a transfer, the EP8000 will abort and show FAIL in the LED display. When all data has been properly received, including the end of file record, the EP8000 will display PASS.

3/ The maximum speed of transfer using the ASCII-Hex format is 2400 baud.

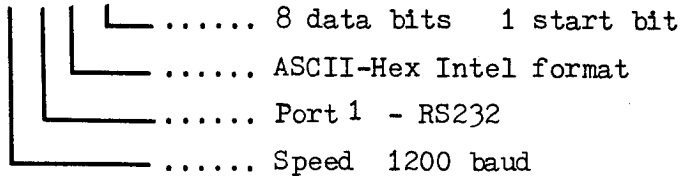
6.11 Example of SOUT using Intel Ascii O/P

Data in a development system is to be sent to the EP8000 for programming into a 2716 EPROM. Speed is 1200 baud, using the Intel 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 EP8000 requests 'FL' in the status display for the serial input word.

Key: 3 1 1 8



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

6.12 Motorola Exorciser Format

The EP8000 handles two types of record. The data record (type S1) and the end of file record (type S9).

6.13 Data Record Format (type S1)

Bytes	Contents
1-2	S1 Start character
3-4	Byte Count - The number of data bytes plus 3 (1 for checksum and 2 for address) in hexadecimal
5-6	MSB of address of 1st data byte in record.
7-8	LSB of address
9	Data bytes
last two bytes	Checksum. One's complement of binary summation of preceding bytes in record (including byte count and data bytes) in hexadecimal notation.

6.14 End of File Record (type S9)

Bytes	Control
1-2	S9 start characters
3-4	Byte count = 03 in end of file record
5-6	MSB of address
7-8	LSB of address
9-10	Checksum

6.15 Motorola Exorciser O/P

Each output record is 16 bytes long. The start address of the 1st record is entered by the cursor. All following addresses being calculated by the EP8000. The transfer address is the address entered by the user.

Example: Transfer block of data from the EP8000 to an external machine at 600 baud with one stop bit.

Define the block then key BLOCK TO SOUT, the machine prompts with FL. The user then enters A0A9. The machine then requests a start address. Once the start address is entered the machine starts the serial transfer.

6.16 Motorola Exorciser I/P

Each input block should be a maximum of 32 Hex bytes long.
The checksums are compared at the end of each record. Should any checksum fail the machine will abort and display FAIL in the LED display. When all data, including the end of file record has been received ,the EP8000 will display PASS.
Example: Download data from an external system to the EP8000 at 2400 baud via the RS232 port.
Key SIN TO RAM. The EP8000 requests 'FL' in the status display, requesting the serial input word.
Key 4128
Data will now be transferred to the EP8000, which will show PASS/FAIL dependent upon whether the transfer was successful.

6.17 PORTS

Two 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 2 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.

1/ RS232

This port is usually used in noisy environments, or where the communicating device is remote from the EP8000, 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 EP8000 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.

6.18 INPUT AND OUTPUT WORDS (Speed selection)

Once the EP8000 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	0 GP Binary	1 1
1	300	1 RS232	1 INTEL Hex	2 2
2	600	2 TTL	2 Motorola	3 3
3	1200		Exorciser	4 4
4	2400			5 5
5	4800			6 6
6	6400			7 7
7	9600			8 8

Digit 1	Baud rate	Digit 2	Digit 3	Digit 4 Data bits & Stop bits
8	110	Enter 0	8 GP Binary	9 8 1
9	300	(not used)	9 Intel Hex	A 8 2
A	600		A Motorola	B 8 3
B	1200		Exorciser	C 8 4
C	2400			D 8 5
D	4800			E 8 6
E	6400			F 8 7
F	9600			

The Cassette Interface

7.00 General

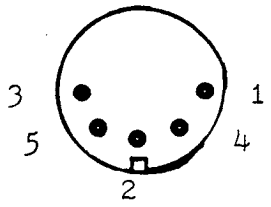
The interface provides a simple reliable method of storing object code from the EP8000. The data is phase encoded and the system virtually eliminates problems due to 'drop out', speed fluctuation and AVC systems.
The data is transferred at approximately 1200 baud with 2 stop and 2 start bits in each record.

7.01 Cassette recorder

Any reasonable quality cassette recorder should work. Those without a tone control are more simple to set up, as there is one less variable.
For machines with a tone control, this should usually be set to 'high'.
To ensure reliability it is essential to use high quality tape and to ensure that the record /playback head(s) are free from dirt.

7.02 Interconnection

Connection to the cassette recorder is made from the RS232/cassette DIN socket at the rear right of the EP8000 (viewed from the rear). Use a 3 pin or 5 pin 180 deg DIN plug for connection with a 2 wire screened cable.



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.

7.03 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 8k each may be stored on cassette (i.e approximately 2 megabyte of data).
During the course of a load from cassette, the EP8000 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.

7.04 Data Format

Prior to transmission, a 16 bit checksum is calculated on the data block, together with a 2 byte block length. These bytes, together with the filename are saved as a leader block on the tape.

File format:

OF bytes (256 bytes)	File (2 bytes)	Block listing (2 bytes)	Checksum (2 bytes)	Data bytes (Up to 8k Bytes)

Preceding the file number is 256 OF (hex) bytes. The tone produced is used to defeat the AVC of the recorder, before valid information is transmitted. These bytes also help to ensure that the transmission is synchronised. 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 the end of the transmission. When transmission is complete the EP8000 will perform a checksum on the received data and compare this with the received checksum. The result will be shown on the LED display as either PASS or FAIL.

7.05 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 upon your recorder). When done, the EP8000 will be in C1 mode. Transmission of a screen page will take about 7 seconds.

7.06 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 EP8000 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 EP8000 will consistently load data reliably with no error rate.

SECTION 8

EPROM EMULATION

8.00 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:

8.01 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 EP8000 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 EP8000 RAM, select 2716 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 EP8000'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 EP8000 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 EP8000'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.

8.02 Emulation Procedure

The emulation plug at the rear of the machine gives direct access to the EP8000 RAM via line driver/buffers. These devices are not protected in the same 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 MESA8 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 EP8000 26 way plug (cable down) - Ensure proper connection.

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

3/ Select the required EPROM type with the 'DEV' key- the EP8000 will configure the address and chip select lines so that it 'looks' like the selected device.

4/ Remember - the external system has access to the EP8000 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 to access the EP8000. The RAM can be isolated at any time from the host system by keying 'DMA' a second time.

8.03 Emulation Characteristics

The table below compares the EP8000 emulation characteristics with those of a typical EPROM - a 2716.

NOTE: 1/ No power is supplied by the EP8000 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 three rail devices, 20mA is taken from the -5v and +12v lines from the host system. These lines are used as references by the EP8000.

4/ The EPROM emulation facility is READ only - do not attempt to 'PROGRAM' the EP8000 !!

COMPARISON BETWEEN 2716 AND EP8000 CHARACTERISTICS

READ OPERATION

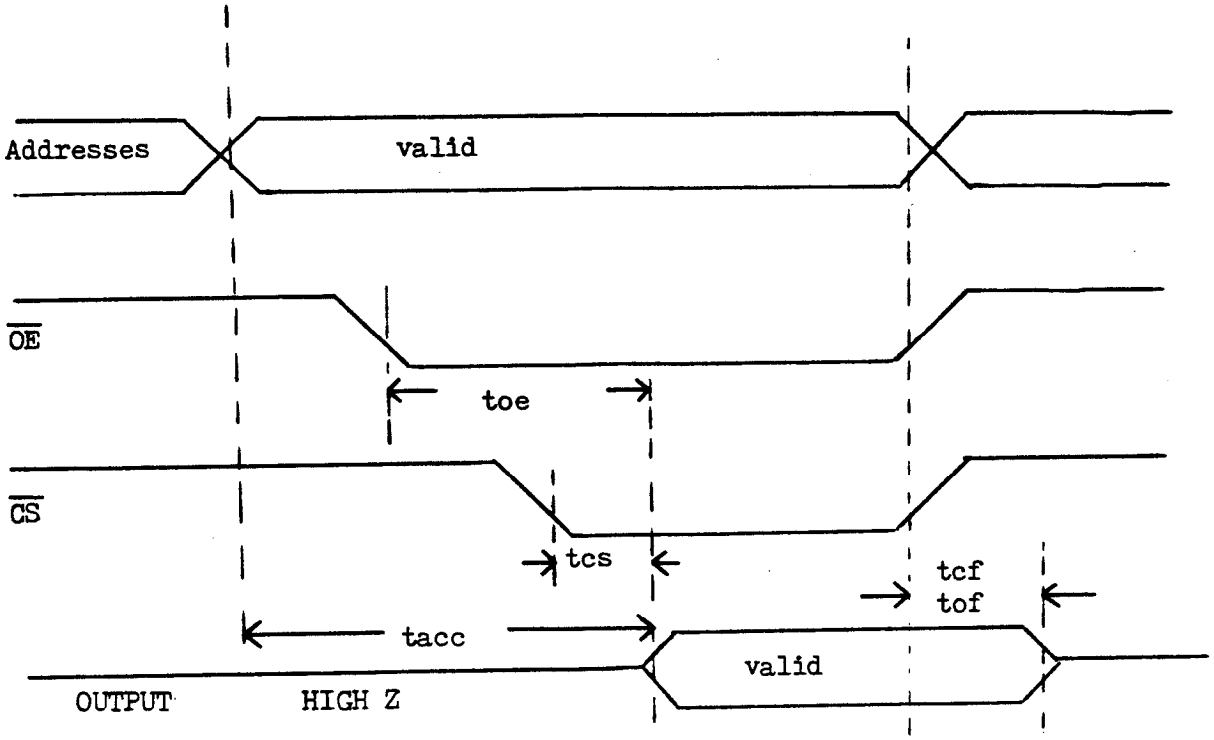
DC & Operating Characteristics

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 / EP8000



Section 9

EPROM Programming & Erasing

9.00 EPROM Erasing

All the EPROMs handled by the EP8000 are erased by exposure to shortwave ultra violet light - 253.7nm except for the 48016 which is electrically erased by the EP8000. There is no way to erase a part of a bit pattern, and after erasure all bits are set to logic 1.

The recommended integral dose is 15W.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 ultraviolet 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 EPROM erasers, and an erase time of 20 minutes.

2/ To prevent unintentional erasure by sunlight, or flourescent 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.

9.01 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 EP8000 is capable of programming four types of EPROM.-

1/ The '3 rail' devices, 2704, 2708, TMS2716.
The programming requirement for these devices is that the program enable line be taken to +12v, then after data and address are set-up, 2 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 2508,2758,2716,2532,2732,2564,2764,2732A.
These devices require the program enable line at +25v or +21v 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.

3/ The 3rd type of device supported by the EP8000 is the motorola family 68 series. These are similar to the 3 rail devices in that they require a pulsed Vpp line and multiple passes through all addresses.

4/ The fourth device type is the 48016 E^2PROM from Hitachi.
Prior to programming, the EP8000 will electrically erase and blank check the device. No erase is performed during block programming (i.e. when programming a few bytes into a non-blank device).

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. TMS2716 is a three rail type).
(2) The 2732 and 2532 (4k) devices are not pin compatible in either the read or program mode.
(3) The 2564 and 2764 are not compatible in either the read or program mode.

9.02 Programming with the EP8000

There are 2 ways of transferring data from the EP8000 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 on the next page:

Device	RAM address	ZIF address
2704	3000-31FF	6000-61FF
2708	3000-33FF	6000-63FF
2716(3)	3000-37FF	6000-67FF
2508	3000-33FF	6000-63FF
2758A	3000-33FF	6000-63FF
2758B	3000-33FF	6400-67FF
2516	3000-37FF	6000-67FF
2716	3000-37FF	6000-67FF
48016	3000-37FF	6000-67FF
2532	3000-3FFF	6000-6FFF
2732	3000-3FFF	6000-6FFF
2732A	3000-3FFF	6000-6FFF
68732-0	3000-3FFF	6000-6FFF
68732-1	3000-3FFF	7000-7FFF
68766	3000-4FFF	6000-7FFF
68764	3000-4FFF	6000-7FFF
2564	3000-4FFF	6000-7FFF
2764	3000-4FFF	6000-7FFF

Prior to programming the device, the EP8000 will first check if it is programmable by performing an illegal bit check. If the device is not programmable, this will quickly be shown by faults 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. Select 2716 with the 'DEV' key and key RESET to power down the ZIF. Insert a blank device and key PROG. The ZIF will power up, and after the initial illegal bit test, programming will commence.
After the 100 second programming time, the EP8000 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/ Select 2716 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.
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, select 2732 with the 'DEV' key 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 MEM 6321. 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.
An illegal bit check is performed on the device only on those addresses that require programming. Once programmed, the block data is verified for correct transfer as indicated by PASS or FAIL.

APPENDIX A EP8000 Memory Map Addressing

The microprocessor used to control the function of the EP8000 is the INS8060, commonly called the SC/MP.

EP8000 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
0000-0FFF	Monitor in 2732
1000-1FFF	Monitor in 2732
2000-2FFF	Monitor in 2732
3000-4FFF	Main RAM
5000-57FF	Video RAM
5800-5BFF	Ports & Configurator
5C00-5FFF	Scratchpad RAM (PAD)
6000-7FFF	ZIF Socket

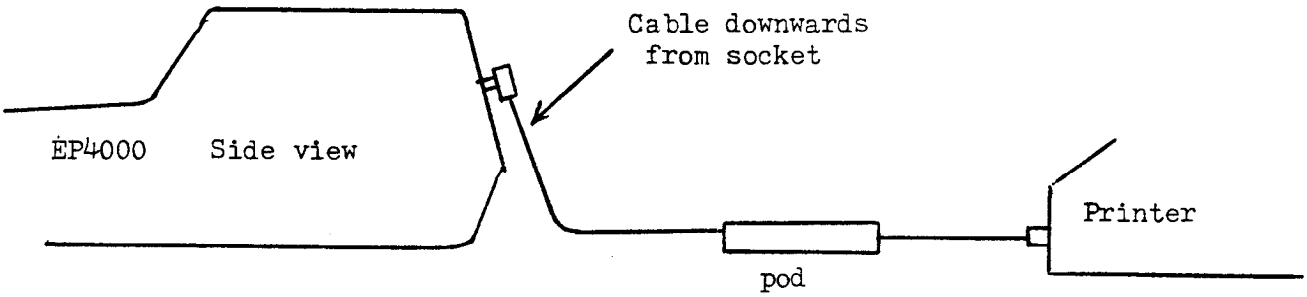
The ZIF socket addresses used by the monitor depend upon the device selected, although all addresses are accessible. These are as follows:

Device	Address
2704	6000-61FF
2708	6000-63FF
2716(3)	6000-67FF
2508/2758A	6000-63FF
2758B	6400-67FF
2516/2716/48016	6000-67FF
2532	6000-6FFF
2732	6000-6FFF
68732-0	6000-6FFF
68732-1	7000-7FFF
68766/68764	6000-7FFF
2564/2764	6000-7FFF

GR1 PRINTER INTERFACE FOR EP4000

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

Connecting the interface to the EP4000



Use 'Print' Key

Example Fn Print Fn RAM, prints entire RAM contents

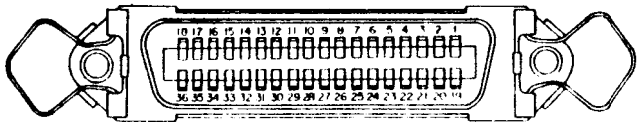
Fn Print Block, prints highlighted block

NB: If 'Print' key gives a flashing 'NO' message, your EP4000 monitor EPROM set requires updating -- Please contact the sales desk or your local distributor.

[6] INTERFACE

1. Input Connector

Use a connector, AMP CHAMP 36 BAIL LOCK TYPE, to input data into the Printer. Pin configuration and its signals of the receptacle in left rear of the Printer are described below.



PIN	SIGNAL	PIN	SIGNAL
1	$\overline{\text{STROBE}}$	19	TWISTED PAIR GND (PAIR WITH 1 PIN)
2	DATA 1	20	TWISTED PAIR GND (PAIR WITH 2 PIN)
3	DATA 2	21	TWISTED PAIR GND (PAIR WITH 3 PIN)
4	DATA 3	22	TWISTED PAIR GND (PAIR WITH 4 PIN)
5	DATA 4	23	TWISTED PAIR GND (PAIR WITH 5 PIN)
6	DATA 5	24	TWISTED PAIR GND (PAIR WITH 6 PIN)
7	DATA 6	25	TWISTED PAIR GND (PAIR WITH 7 PIN)
8	DATA 7	26	TWISTED PAIR GND (PAIR WITH 8 PIN)
9	DATA 8	27	TWISTED PAIR GND (PAIR WITH 9 PIN)
10	$\overline{\text{ACK}}$	28	TWISTED PAIR GND (PAIR WITH10PIN)
11	BUSY	29	TWISTED PAIR GND (PAIR WITH11PIN)
12	LOW	30	GND
13	NC	31	$\overline{\text{INITIAL}}$ (PAIR WITH14PIN)
14	GND	32	$\overline{\text{ERROR}}$ (PAIR WITH15PIN)
15	GND	33	GND
16	GND	34	$\overline{\text{CLK}}$ (PAIR WITH33PIN)
17	CHASSIS GND	35	TEST (PAIR WITH16PIN)
18	+5V 80mA Max.	36	+5V

NOTES: 1. LOW of pin 12 is a "LOW" level output of TTL.
2. The combined output of pins 18 and 36 is 80 mA maximum.
3. NC stands for no connection.

2. Input/output Signals

(1) Input signals to the Printer

* DATA 1, DATA 2, DATA 3, DATA 4, DATA 5, DATA 6, DATA 7, DATA 8 — 8-bit data signals. Signal "HIGH" represents Logic '1'.
* $\overline{\text{STROBE}}$ The strobe signal is used to read in 8 bits of data. Data is read in when the signal goes 'LOW'.
* $\overline{\text{INITIAL}}$ This signal is used to set the Printer to an initial state and is normally "HIGH". Bringing the line "LOW" and returning it "HIGH" starts the clearing action which sets the Printer to an initial state.
* TEST This signal is used for the self-printing test which is executed by bringing the line "LOW"

(2) Output signals from the Printer

* BUSY This signal indicates the BUSY status of the Printer. When "HIGH" the Printer can not accept data.
* $\overline{\text{ACK}}$ This signal is output at the end of the BUSY signal without fail and is used to indicate that the Printer is awaiting data.

NOTE: The BUSY and $\overline{\text{ACK}}$ signals are always output when the Printer accepts data input.

* $\overline{\text{ERROR}}$ A printer error condition causes this signal to go "LOW". When this happens all the control circuits internal to the Printer halt. There are two ways to correct this situation.
  - Turn the Printer off – and then back on two seconds later.
  - Input the $\overline{\text{INITIAL}}$ signal

  An ERROR occurs if the dot timing goes bad.
* $\overline{\text{CLK}}$ This is a 400 KHz clock signal. It can be used with an interface if needed.

BSC-8 Buffered Simulator Cable for EP8000 EPROM Programmer

1: Introduction

The BSC-8 has been designed to provide a convenient length of simulator cable without impacting emulation access time & to reduce bus loading. The unit is powered from the host system (typical supply current is 80mA).

The unit comprises 4 sections:

1: A long cable for connection to the EP8000 simulator plug.
2: A short 24 pin cable for emulating 24 pin EPROMs.
3: A short 28 pin cable for emulating 28 pin EPROMs.
4: A pod containing the buffers and personality selector.

2: Connection

Select either the 24 pin or the 28 pin short cable and plug it into the Buffer Pod (the case easily splits apart to facilitate this).
Plug the long cable into the EP8000 Simulator plug - ensure proper connection - the cable should feed downwards from the socket.
Plug the dual in line plug into the host system EPROM socket.
Pin 1 is indicated by a '1' or an arrow. Ensure correct orientation otherwise the buffers will be damaged.

3: Personality Selector

This comprises 2 DIL switches inside the buffer pod. The single pole switch determines 24 or 28 pin emulation. The 8 pole switch is used to select which EPROM pins are used as chip select. The switches should be set up in accordance with the table below.

EPROM Type	8 pole switch position	1 pole switch position
2704	7	0
2708	7	0
2716(3)	8	0
2508	7	0
2758A	7	0
2758B	7	0
2716	7	0 Switch button moved
48016	7	0
2532	6	0 away from radial
2732	7	0 capacitor
68732-0	6	0
68732-1	6	0
68766	6	0
68764	6	0
2564	5	1 Switch button moved
2764	4	1 towards radial capacitor

SA27128

GP Industrial Electronics Ltd.,
Unit E, Huxley Close,
Newnham Ind Estate, PLYMOUTH

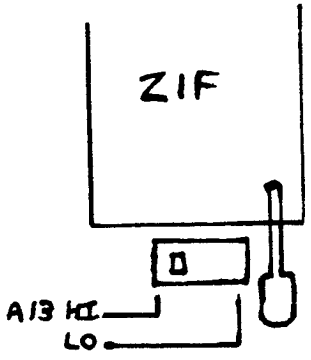
SA- 27128 Programming Adaptor for the INTEL 27128

General: The SA27128 has been designed for quick connection to the EP8000 to allow it to program and read the INTEL (and others) 27128 EPROM. The module configures the high order address lines, output enable and chip select as required for read and program modes.
Since the EP8000 has 8k x 8 RAM, the 27128 (16k x 8) must be programmed in 2 halves, and a DIL switch is provided for 8k block selection (i.e. A13 logic high or low).

Connection and Use:
1) Press reset on the EP8000 to power down the ZIF socket.
2) Select 2764 using the device select key.
3) Plug the adaptor into the EP8000 ZIF socket and close the lever. The adaptor box should rest in the ZIF, and on the edge of the EP8000 case with the adaptor's ZIF lever furthest from the ZIF LED. 'Pin 1' is in the same relative position as on the EP8000 - i.e. top left hand corner of the adaptor ZIF furthest from the ZIF lever (red dot indicates).
4) Select the required 8k block with the DIL switch - '1' indicates A13 high, '0' indicates A13 logic low.
5) Plug in the 27128 device and use the EP8000 in the usual way.

DO NOT attempt to read or program any device other than 27128's in the adaptor - always select A-2764 on the EP8000.

NOTE: In common with all devices repeatedly inserted into the TEXTOOL ZIF sockets, device leads eventually become pitted and dirty at the point of contact. This will cause erratic device read and program operation. Always ensure the adaptor ZIF connector pins are clean - This also applies to 'Master' EPROMs which undergo repeated insertion into the ZIF socket.
Also clean the ZIF socket itself from time to time with a stiff bristle brush only.



PROGRAMMING AN EPROM USING THE EP8000.

1. Switch on mains supply. The display will show C1 3000 FF to indicate mode C1, and that the cursor is at address 3000, and that the data at that address is FF.
2. Set the device selection to the required device type. By pressing function (FN) and then the DEV key, the device select mode is entered. The current device type is shown in the LED display and is highlighted in the video display. The device type may be changed by using the cursor key to scroll through the list of devices.
3. Press the reset key RST to power down the zero insertion force (ZIF) socket. Insert the device to be copied in the ZIF socket and close the leaver flush with the ZIF socket and the panel - check that the device is the correct way round, not misplaced, and properly seated.
4. Copy the contents of the EPROM to RAM by using the following keys. Press function (FN) PROM, TO, (FN) RAM, this stores the contents of the EPROM to the RAM starting at address 3000(H).
5. Verify the EPROM. This is done to check that the RAM contains a true record of the contents of the EPROM by using the following keys, Function (FN) VFY. PASS will be shown in the LED display if the data is identical, else FAIL will be shown. If a FAIL occurs the differences will be highlighted, but only if the page displayed is a RAM page corresponding to the EPROM. Each page can be checked by keying function (FN) PAGE and using the cursur keys ↑↓ to allow the screen to be moved through the memory, page at a time.
6. Press function (FN) CHCK. This will do a checksum by adding all the bytes in the EPROM defined by the device sellection. The checksum is shown in the address section of the LED display and the HEX NUMBER should be noted to check the copy EPROM. The values of a 'blank check' checksum of an EPROM are shown in the manual. eg. a 2716 blank checksum is F800.
7. Press reset RST to power down the ZIF socket and then remove the master EPROM.
8. Fit blank EPROM to ZIF socket and close leaver flush with ZIF socket and the panel - check the device is correctly fitted.
9. Program EPROM by pressing the following keys, function (FN) PROG. This will transfer the contents of the RAM to the EPROM. Once programming is complete the machine verifies the data, showing PASS or FAIL and highlighting any discrepancy bytes. If the device is not programmable the display will show fault B and the reset must be keyed to continue.
10. A checksum should be carried out and a comparison of the HEX NUMBER made with the checksum number noted on the master EPROM.

FUNCTION (FN) STORE (F) KEY

(C)

2716 BLANK CHECKSUM F800