A.R.T's

EPP-1 E(E) PROM PROGRAMMER

manual

EPP-1 E(E)PROM

PROGRAMMER

APPLIED READER TECHNOLOGY BV

HEAD OFFICE:
KANAALDIJK-NOORD 25
5613 DH, EINDHOVEN
THE NETHERLANDS
TELEPHONE: 040-433671/433691
TELEFAX : 040-433653

Contents

1.0. General Product Description.

The ART "EPP-1" EPROM programmer programs all current EPROMs in the 2716 - 27513 range, numerous 25XX EPROMs and the X2864 EEPROM. (The 27513 must be programmed "page by page".) Either "Intel" or "FPC" HEX files may be up- or downloaded.

The unit may be operated by any host system equipped with a RS232 interface. No special communications protocol is required (any standard terminal emulation program, such as 'Procomm', should be adequate). The EPP-1's command line interpreter enables the user to enter simple 1-character commands, or even multiple commands. Execution begins after a <CR> (carriage return) entry. Example, to display the Offset Address, "O<CR>" keys would be pressed. (The only exceptions to the need for <CR> are: 1) when using the <ESC> key, or 2) when downloading files to the EPP-1.

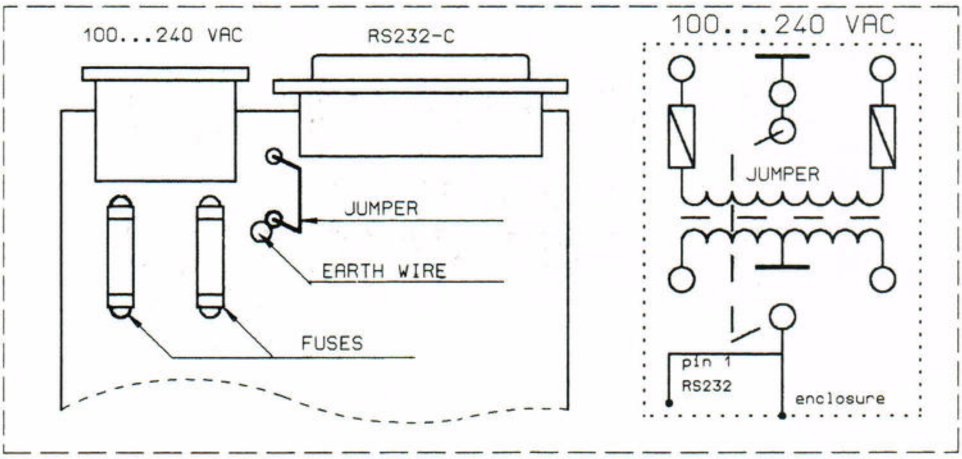
The EPP-1 uses a 1200 baud rate. This increases programming time by only 20% or so when programming higher speed EPROMs, but offers the advantage that these higher speed EPROMs can be programmed by user systems not equipped with RTS/CTS flow control (although flow control is always desirable).

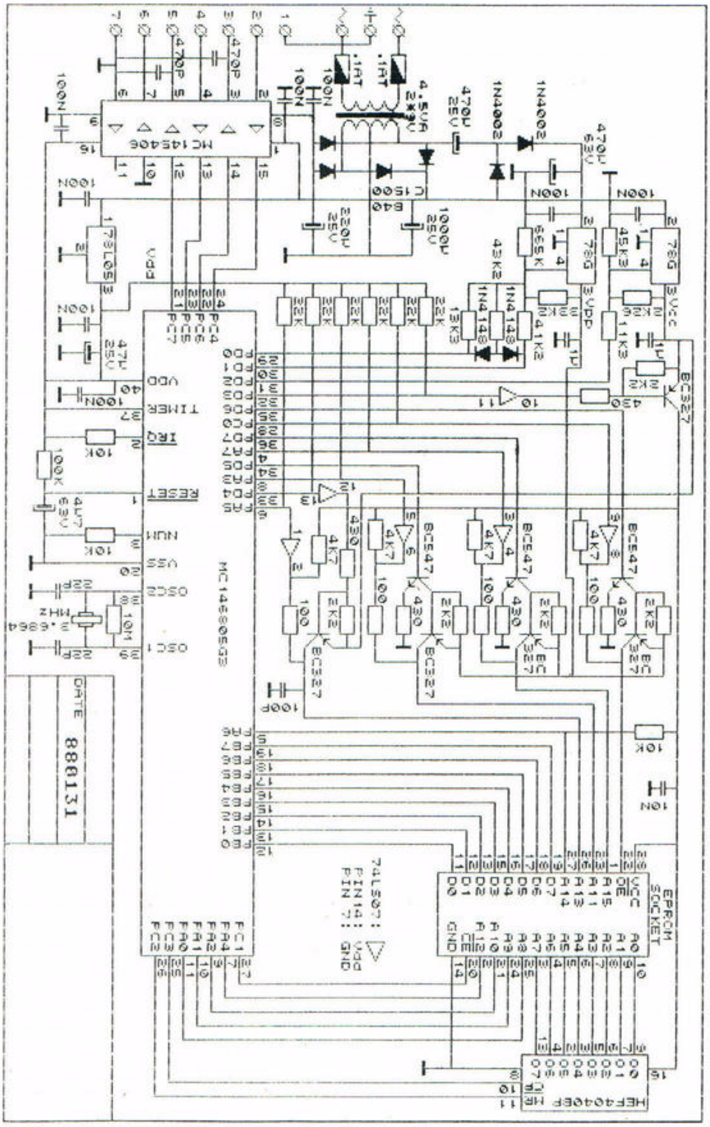
1.1. Technical summary.

Operating voltage	: 220 VAC (Europe) & 110 VAC (N. America)
Power consumption	: 4.5 VA Power
Connector	: "Euro" connector
Fuses	: 2 x 225ma, slow blow
Interface	: RS232
Connector	: DB25 (female)
Baudrate	: 1200
Data bits	: 8 (bit 7 = 0)
Parity	: None
Start/Stop bits	: 1
Enclosure	: Anodized aluminum
Dimensions	: 171 x 110 x 63 [mm]
Weight	: 800 gr.

==
*Additionally, if your RS232 cable is shielded and you wish to ground the shield (connected to **pin 1 of the RS232 connector**) to the **EPP-1 case**, 1) Open the enclosure and 2) solder the jumper wire shown in Fig. 1.1.1. below.*
==

Figure. 1.1.1. PCB Earth Contact.





MC146805G3
MC14S406
HEF4040BP
EPROM SOCKET
74LS07: PIN 14: Vdd PIN 7: GND
78LO5
78G
BC327
BC547
3.6864 MHz
DATE 881131

2.0. Command Description.

The Epp-1 utilizes 14 commands. A command in the process of execution may be canceled by pressing <ESC>. The command syntax has been developed to quickly enable the user to operate the EPP-1 without having to continually refer back to the instructions.

Summary of Commands

Table 2.0.1.

<ESC>	Cancel present command
vP	Define EPROM start address
P	Display EPROM start address
vL	Define last EPROM address
L	Display last EPROM address
vO	Define offset address
O	Display offset addres
T*	Test
R*	Read
W*	Write
V*	Verify
G	Get result code
wS	Select EPROM type
S	Display selected EPROM type
*	Error code will appear (if error results) when using these commands
v	Address in HEX
w	EPROM select data word

<ESC> = Cancel present command execution

Execution of a command is cancelled, and the EPP-1 returns to initial state. Values set by the P, L, O and S command remain uneffected.

P = EPROM Start address (HEX)

"P<CR>" outputs the currently selected start address to the user interface. Default is "0000". "Error" message is output if no EPROM type has been selected.

vP = Define Start address (HEX)

"vP<CR>" sets the EPROM start address. The value must be lower than the defined "highest address" or an "error" message will output to the user interface when trying to execute the W/R/T/V commands. An "error" is also generated if no EPROM type has been selected.

Example: If "200P<CR>" is entered, then address zero will be relocated to address "200" in the EPROM. Therefore, the EPP-1 will not access any physical address from "0000" to "0200". (See also "L" command.)

L = Last EPROM address (HEX)

"L<CR>" outputs the currently selected high address to the user interface. Default setting is the selected EPROM's highest address. "error" message is output if no EPROM type has been selected.

vL = Define Last EPROM address (HEX) to be accessed by the EPP-1

Allows user to select last(highest) address from total address range (not lower than the "start address" or an "error" will output whenever a W, R, T or V command is entered.) An error code is also output if no EPROM has been selected. Example: entering "2FFFL" would set the last(highest) accessable address to "2FFF". No commands will access addresses higher than 2FFF.

O = Offset address (HEX)

The resident offset value is output to the user interface. The default value is "0000".

vO = Define Offset address (HEX)
User defines an offset address (which is subtracted from the downloaded file addressing or added to the uploaded file addressing) in order to relocate the file to the correct EPROM address. The default value is "0000". This value is also output to the user interface if no EPROM has been selected.

T = Test
"T<CR>" verifies that the EPROM is "empty" between the defined "vP" and "vL"(first and last) address values. If the range is not empty, an "error" message will be output to the user interface.

R = Read
"R<CR>" outputs the contents of the selected EPROM in Intel format to the user interface.

W = Write
"W<CR>" is entered and the EPP-1 waits for the user interface to download an Intel format file which will be relocated according to the P, L and O command settings. An "error code" is output to the user interface if an error occured or if no EPROM type has been selected.

V = Verify
"V<CR>" is entered and the EPP-1 waits for the user interface to download its Intel format file (to the EPP-1).

This file will be checked against the code already in the EPROM. If the two match, no error code is generated. Otherwise the EPP-1 will output an "error code" indicating the nature of the problem. An error code is generated if no EPROM type has been selected.

G = Get Result Code
Output the result code (as a 16-bit word) to user interface. If both bytes are zero, no error(s) occurred. Table 2.0.1. defines the result codes (see also examples given under the "Tutorial" section, Paragraph 3.2(f).

Table 2.0.1. Result Code Word

Bit 0	Bit can't be programmed.	E(E)PROM is damaged or wrong selection code.
Bit 1	Illegal bit error.	Bit is programmed when it should not have been programmed. (EPROM wasn't totally erased or is damaged.)
Bit 2	Hex digit error.	A down-loaded digit is outside HEX range values 0..9 or A..F. Check downloaded file.
Bit 3	Address range error.	Address is not valid (out of range!) Check range.
Bit 4	Command error.	Illegal command. Trying to read, write etc. while no valid selection code was entered.
Bit 5	HEX check error.	Sumcheck of a downloaded record is incorrect.
Bit 6	Abort flag.	Sets after use of <ESC> command.
Bit 7	Not empty flag.	E(E)PROM not empty within defined range.
Bit 8	FPC Format error. (See appendix -B-)	An error has occured during the conversion to FPC format and/or downloading.
Bit 9	Input overflow error	Input buffer could not capture all incoming data. Use of flow control.
Bit 10-15	Not used.	

S = Display EPROM type
"S<CR>" outputs the selected EPROM type to the user interface. The default value is "0000".

wS = Select EPROM type
"wS<CR>" selects a specific EPROM type. If the code "w" is not valid, an error will be produced. "w" consists of 2 bytes (one word). Figure 2.0.2. shows the structure of the selection code word, while Table 2.0.2. defines the meanings of the bits in the selection code word.

If only the "R" (Read) command is to be utilized, only bits 0-2 need to be selected. The remaining bit values "don't care".

Note: To understand the calculation of the selection code (necessary if trying to determine whether an E(E)PROM not listed, can be programmed by the EPP-1), see Fig. 2.0.1. (The specific parameter values are found on the EPROM's technical data sheet.)

Figure 2.0.1. Interactive Programming

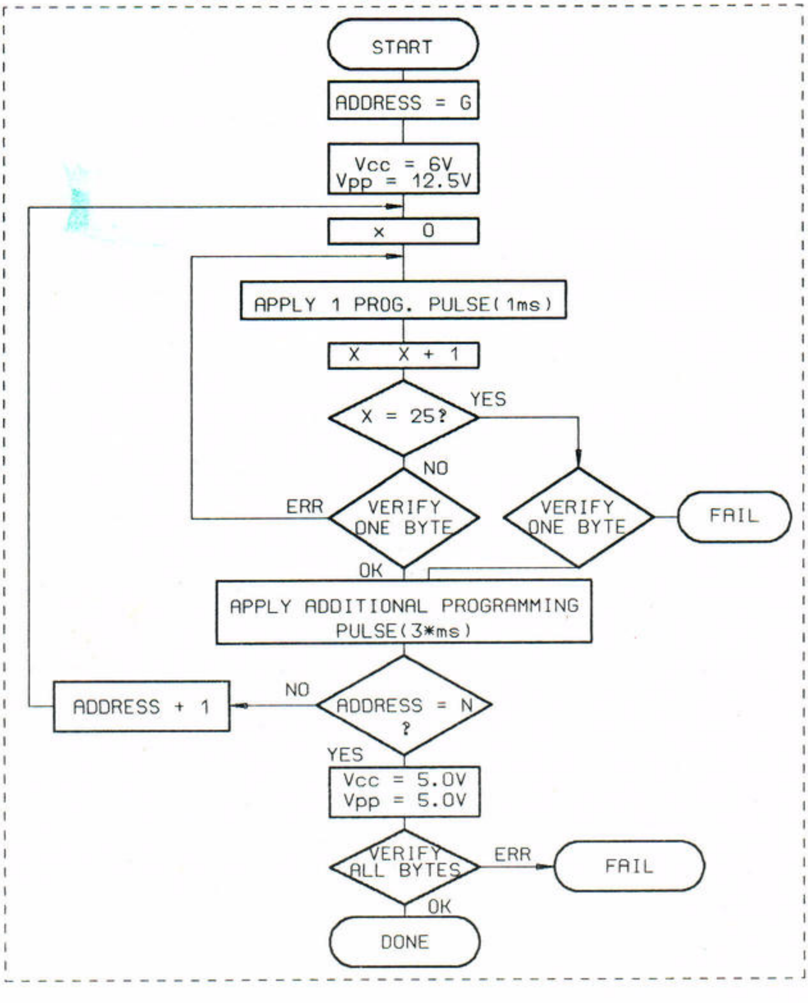


Table 2.0.2. "EPROM Type Selection Parameters"

Bit 0-2	Length/Pinning		Bit 8-11	Program time	
	000	invalid		0000	2.5 ms
	001	2716		0001	5.0 ms
	010	2732		0010	7.5 ms
	011	2764		0011	10.0 ms
	100	27128		0100	12.5 ms
	101	27256		0101	15.0 ms
	110	27512 6		0110	17.5 ms
	111	invalid		0111	20.0 ms
				1000	22.5 ms
				1001	25.0 ms
Bit 3	Reserved			1010	30.0 ms
	0	OK		1011	35.0 ms
	1	invalid		1100	40.0 ms
				1101	45.0 ms
				1110	50.0 ms
Bit 4-5	Program voltage			1111	55.0 ms
	00	12.5			
	01	21	Bit 12-14	Program factor	
	10	25		000	"Dumb"
	11	invalid		001	1x
				010	2x
				011	3x
				100	4x
Bit 6	Vcc program			101	5x
	0	5 VDC		110	6x
	1	6 VDC		111	7x
Bit 7	Reserved		Bit 15	FF Skip	
	0	OK		0	NoFF Skip
	1	invalid		1	FF Skip

Figure 2.0.2. on the next page gives the structure of the selection code word. Using this and table 2.0.2. and the data sheet of an E(E)PROM you can establish the correct selection code if a certain type is not listed in appendix -A-.

Example: For the Fujitsu 2764 EPROM the correct selection is put together as follows:

Type selection	:	2764	Program time	:	15 ms
Program voltage	:	21 VDC	Program factor	:	4
Vcc program	:	6 VDC	FF skip	:	yes

The following selection code is now established(check with figure 2.0.2):

0100 0101 0101 0011

$1^3+1^2+0^1+0^0$	$0^3+1^2+0^1+1^0$	$0^3+1^2+0^1+1^0$	$0^3+0^2+1^1+1^0$
C	5	5	3

Selection code Fujitsu 2764 = C553

"Dump" programming implies that the programming algorithm is not used, which valid for some older types of eproms which cannot be 'pulse-programmed' as the algorithm does.

"FF skip" can be used while programming empty eprom's. If FFh is to be programmed the EPP-1 will skip programming this because an empty eprom already contains FFh. For EEproms "FF skip" should not be used.

Figure 2.0.2.Selection code word.

Bit 0-2 Length/pinning

	Bit 2	Bit 1	Bit 0
invalid	0	0	0
2716	0	0	1
2732	0	1	0
2764	0	1	1
27128	1	0	0
27256	1	0	1
27512	1	1	0
invalid	1	1	1

Bit 3

ok	0
invalid	1

Bit 4-5 Program voltage

	Bit 5	Bit 4
12.5 VDC	0	0
21 VDC	0	1
25 VDC	1	0
invalid	1	1

Bit 6 Vcc program

5 VDC	0
6 VDC	1

Bit 7

ok	0
invalid	1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	**Bit nr.**

Bit 8-11 Program time

Bit 11	Bit 10	Bit 9	Bit 8	
0	0	0	0	2.5 ms
0	0	0	1	5.0 ms
0	0	1	0	7.5 ms
0	0	1	1	10.0 ms
0	1	0	0	12.5 ms
0	1	0	1	15.0 ms
0	1	1	0	17.5 ms
0	1	1	1	20.0 ms
1	0	0	0	22.5 ms
1	0	0	1	25.0 ms
1	0	1	0	30.0 ms
1	0	1	1	35.0 ms
1	1	0	0	40.0 ms
1	1	0	1	45.0 ms
1	1	1	0	50.0 ms
1	1	1	1	55.0 ms

Bit 12-14 Program factor

Bit 14	Bit 13	Bit 12	
0	0	0	0 X
0	0	1	1 X
0	1	0	2 X
0	1	1	3 X
1	0	0	4 X
1	0	1	5 X
1	1	0	6 X
1	1	1	7 X

Bit 15 FF skip

0	No FF skip
1	FF skip

3.0 TUTORIAL - sample session

==

IMPORTANT - To prevent serious damage to the EPP-1 and the user interface, insure that all conductors in the RS232 cable are correctly "pinned" according to Figure 3.1 below!

==

3.1. SETUP

3.1a Connect the EPP-1 to the mains power source (be sure a 3-conductor grounded cable is used).

3.1b Run your computer's terminal emulation program, setting the communication port (frequently "COM1" on the IBM PC's) as follows:

Baudrate	:	1200 baud
Start/Stop bits	:	1 bit
Data bits	:	8 bits
Flow control	:	RTS/CTS (if available)

3.1c Connect the EPP-1 to the host system's RS232 interface. If in doubt, check with Fig. 3.1 before proceeding, making your own "flat cable" if necessary.

Figure 3.1

```
                   Pin 1 = Shield   ---   Shield = Pin 1
                   Pin 2 = Tx       ->-   Rx     = Pin 2
                   Pin 3 = Rx       -<-   Tx     = Pin 3
HOST               Pin 4 = RTS      ->-   CTS    = Pin 4          EPP-1
====               Pin 5 = CTS      -<-   RTS    = Pin 5          =====
                   Pin 6 = input    ---   high   = Pin 6
                   Pin 7 = ground   ---   ground = Pin 7
```

The monitor should display:

```
ARTepp, ver 870808
*
```

3.2. OPERATION

A Fujitsu 2764 will be used as the basis for this tutorial. The data will be copied to another Fujitsu 2764, but it could just as easily be an Intel 27512 or any other device which would possess adequate "empty" capacity.

3.2a. Use Table 3.1. to determine the correct selection code for the Master EPROM, which is shown as "C553".

Note: if your EPROM is not listed in the table (or if the code shown next to the device in the table), you can calculate the code by referring to Table 2.0.2. You should do this now for the Fujitsu 2764 and compare the result with the value given in appendix -A-.

3.2b Enter "S<CR>" to see if an EPROM code has already been selected. If no value is currently selected, the result will be:

```
0000
*
```

....or you can directly....

Enter "*C553S" which will overwrite any previously entered value. Enter "S<CR>" to verify your entry. "C553" should appear on the screen.

SUGGESTION: Check your Intel file range to be certain that it will "fit" into the eprom.

3.2c Enter "*PLO" to check the Start, Last and Offset values (use this to verify the values you may have already entered or to see what previously entered values are contained).

```
0000
1FFF
0000
*
```

This (at left) will be the result for our sample EPROM. In this case we agree that the memory range for the Fujitsu 2764 is 0000h to 1FFFh. Other types will produce different values.

3.2c(1) If a range or offset address need to be changed, enter the correct HEX address value followed by "P", "L" or "O" and <CR>.

EXAMPLE: 1000P<CR> sets the start address to 1000h.

3.2d Upload the data from the master EPROM to the host computer (use your terminal emulation program to define the parameters (directory, filename, etc.) in order to store all data coming from the RS232 interface).

Once you have defined the parameters, enter R<CR> which accomplishes the uploading from the EPP-1.

- The EPP-1 will read the contents of the EPROM, convert it to Intel formatted lines and upload them to the user interface.
- This upload is done between the defined Start- and Last addresses (See P, L and O commands).

3.2d(1) After the uploading is complete, the EPP-1 should cause the dot prompt to appear, which means that the file has been successfully uploaded. If *NOT*, an error message will appear:

*	File was successfully uploaded from the EPROM.
error	File was not succesfully uploaded. Use Result code (par. 3.2f) to determine the programming broblem. Correct problem and try again.

3.2e Assuming that there was no error indication after uploading the file, we could now Verify that the programmed EPROM and uploaded file match:

Enter "V<CR>", return to your terminal emulation program and download the file. The EPP-1 will check the file against the code in the EPROM. If they match, no error indication will appear. If they do not match, "error" will be output, after which the G<CR> command can be entered to indentify the specific problem.

3.2f Confirm the target EPROM "PLO" addresses to insure that the program will fit within the range shown for the target and that the offset address is also correct.

T<CR> Check whether the target EPROM is empty within the selected range.

3.2g W<CR> Assume EPROM tested "empty". Now issue the Write command, which tells the EPP-1 to expect the downloading of an Intel or FPC formatted file.

You must now go to your emulation program and download the file.

3.2g(1) After the downloading is completed, the EPP-1 should cause the dot prompt to appear, which means that the EPROM has been successfully programmed. If *NOT*, an error message will appear:

*	File was successfully downloaded to the EPROM.
error	File was not successfully downloaded. Use Result code (Par. 3.2f) to determine the programming problem.

If "error" then:
G<CR> Enter G<CR> to display the Result code. Convert the hex code to binary and examine the "set"£ bits to identify the error(s).

Assume we have entered G<CR> and now see the result of "29":

29 29(Hex) = $2^5 + 2^3 + 2^0$ = 0010 1001. Table 2.0.1. tells us that we have:

Bit 0 = Not programmed error.
Bit 3 = Address range error.
Bit 5 = Hex check error.

3.2h Assuming that there was no error indication after downloading the file, we could now *Verify* that the programmed EPROM and downloaded file match:

Enter *V<CR>, return to your terminal emulation program and download the file again. The EPP-1 will check the file against the code in the EPROM. If they match, no error indication will appear. If they do not match, "error" will be output, after which the "G<CR>" command can be entered to identify the specific problem.

4.0. Other practical features and additional examples.

4.1. Using the Start-, Last- and Offset- Commands:

Table 4.1.1. Intel Format

```
:2000800013415254657070 2C207665722038373038303820627920 4A18
```

```
:20008000134152546570702C2076657220383730383038206279204A18
:2000A0007073652E054572726F72070F1F3F7FFF09121B242D363F4844
:2000C000AD0BAD693D4B27023A4B1F09803A322A0BA600B7323D3426AC
:2000E00034A602B735813A352BF72601813A34270DB634A109260181C2
:20010000A47FA11B26021E39B63097B0312A02AB20A11F26051E021453
:20012000B633E7105CA32025015FBF30813D372B03260B81A603B7369E
:2001400026 16A603B7363A37270EB6374A2707363825031B02811A0253
:2001600020A10824021F024D27EFBE31E6105CA32025015FBF31A47F51
:20018000B7383A3781B630B73181B6391F39153981AE036F605A2AFB50
```

4.1.a. Start Address Command

If we define the *start address* as 0100h (entered as 100P)..(0100P, 1FFFL, 00800) then: *100h would be added to the EPROM start address. After programming, the first line in the EPROM (see table 4.1.1.) would start at address 180h as shown below:*

Figure. 4.1.1. Start Address Command

```
hardware     0000h :          :    not accessable
               .   :   2764   :    not accessable
             0100h :          :    0000h..relocated start address
               .   :          :      .
               .   :          :    0080h Intel code
               .   :          :      .
               .   :          :      .
             1FFFh :          :    1EFFh
```

4.1.b. Last Address Command

If we define the *Last Address* to be 1000h (1000L), it becomes the highest address in the EPROM. Using combinations of the Offset-, Start- and Last-address commands provides complete flexibility in relocating code or to select specific address regions within the same EPROM.

Figure. 4.1.2. Last Address Command

```
hardware     0000h :          :    0000h
               .   :   2764   :      .
               .   :          :      .
               .   :          :      .
               .   :          :   1000h
               .   :          :    not accessable
               .   :          :    not accessable
             1FFFh :          :    not accessable
```

4.1.c. Offset Command

Looking at the first line of data (Table 4.1.3.) we see that the code starts at address 80h. The address in Intel format is is given by digits 3 through 6, which is shown as 0080h. Using the 2764's default values (0000P, 1FFFL, 00000), this address would be the actual address in the EPROM.

If we define the offset address to be 0080h (may be entered at 800)...(0000P, 1FFFL, 00800), then

> *80H will be substracted from the Intel address and the file will be relocated to start at address 0000h in the EPROM.*

This has a function in hardware environments where an EPROM start address is not mapped at address 0000H.

Figure 4.1.1. Offset Command

```
hardware     0000h :           :    0080h Intel code
                 . :   2764    :      .
                 . :           :      .
                 . :           :      .
                 . :           :      .
                 . :           :      .
                 . :           :      .
             1FFFh :           :    2080h
```

5.0 Trouble shooting.

If you can Read but not Write the E(E)PROM correctly then:

1. Check the Selection Code with data sheet of the E(E)PROM used. Correct any Selection Code and/or try another E(E)PROM.

2. Check wether the Start-, End-, and Offset addresses are within the range of the device.

3. Insure that the file being downloaded does not contain unauthorized characters according to the Intel, FPC or other file format standards.

4. Check the host to EPP-1 connection cable.

5. Does your computer utilize correct hardware flow control? It may not respond quickly enough to the CTS input and may transmit one or more characters too many before stopping transmission. Try using another communication port or slow down transmission speed via software on your computer. This means slowing down the rate at which characters are send and not reducing the baudrate (1200).

If you can Write but not Read the E(E)PROM correctly:

1. Check your host to EPP-1 cable.

2. Check for correct flow control.

If there is no communication at all then:

1. Check your host to EPP-1 cable.

2. Check wether communication port is functioning.

3. Check fuses inside the EPP-1 (Disconnect power first).

4. Test the EPP-1 on another computer. If it functions, contact your computer dealer. If not, contact your EPP-1 dealer.

APPENDIX -A-
Programmable E(E)PROM's.

Other E(E)proms may be programmable -see table 2.0.2. and figure 2.0.2.

	TYPE	Selection code
AMD:	2716	8E21
	2732	8E22
	2732A	8E12
	2764	C553
	2764A	
	27128	C554
	27128A	
	27256	B945
	27512	
EUROTECHNIQUE:		
	ET2716	8E21
	ETC2716	8E21
	ET2732	8E22
	ET2764	B533
FUJITSU:	2716	8E21
	2732	8E22
	2732A	8E12
	2764	C553
	27128	C554
	27256	B945
	27C32A	8E12
	27C64	C553
	27C128	
	27C256	
	27C512	B946
HITACHI:	27128A	B944
	27256	B945
	27512	B946
	27C64G	C553
	27C256	B945
	462716	8E21
	462732	8E22
	482732AG	8E12
	462732P	8E22
	482764	C553
	4827128	C554
	4827128P	C554
INTEL:	2716	8E21
	2732	8E22
	2732A	8E12
	2764	C553
	2764A	
	27128	C554
	27128A	
	27256	B945
	27512	B964
	27C64	
	27C256	
MITSUBITSHI:		
	2716	8E21
	2732	8E22
	2732A	8E12
	2764	C553
	27128	C554
	27C128	C554
	27256	
MOTOROLA:	MCM2716	8E21
NATIONAL SEMICONDUCTOR:		
	2716	8E21
	2732	8E22
	27C16	8E21
	27C16H	
	27C32	8E22
	27C32B	
	27C32H	
	27C64	
	27C128	
	27C256	
	27C512	
	27CP128B	
NEC:	2732	8E22
	2732A	8E12
	2764C/D	C553
	27128C/D	C554
	27256D	
	27C64C/D	C553
	27256A	B945
	27C64	C553
	27C256C/D	
	27C256AD/A	B945
	27C512	B946
OKI:	2716	8E21
	2732	8E22
	2732A	8E12
	2764	C553
	27128	C554
	27256	C945
	27C256	C945
	27256A	C945
RICOH/PANATECH:		
	27C64	
SEEQ:	2764	C553
	27128	
	27C256	B945

SGS-ATES SEMICONDUCTOR CORP.:		
	2716	8E21
	2732A	8E12
	2764	
SIGNETICS:		
	27C64	
TEXAS INSTRUMENTS:		
	2732	
	2732A	8E12
	2764	C553
	27128	C554
	27C128	B944
	27C256	B945
TOSHIBA:	2732A	8E12
	2732D	8E22
	2764	C553
	27128	C554
	27256	9755
	27512	
VLSI TECHNOLOGY INC.(VTI):		
	27C64	
	27C256	B945
WAFERSCALE INTEGRATION (WSI):		
	WS27C64	
	WS27C128	
XICOR:	X2864A	0303
	X2864AL	0303

Not responsible for errors made in printed selection codes. Verify code prior to programming.

APPENDIX -B-

Terminal emulation programs

Any terminal emulation program using the RS232 communication port of the host system can be used to operate the EPP-1. The emulation program should also be able to download files from a disk memory to the RS232 port (sending Intel/FPC code files), and capture incoming data (upload) and store it on a disk medium.

Using a standard terminal emulation program, keep in mind the following:

Upload — After preparing the upload on the host system you might give the Read command to actually upload the contents of an EPROM. After finishing there will be a file on disk containing this contents. But the "R" command will be at the <u>beginning</u> of this file while the "*" (coming from the EPP-1) will be at the <u>end</u>. Use your editor to remove them. The EPP-1 will not recognize the "*" as a command, which will result in an error.

Download — First prepare the EPP-1 for receiving a Intel/FPC file from your host system. After giving the write command download the file via the terminal emulation program.

APPENDIX -C-

Intel Format

This code format uses only ASCII representation of data. There are 8- and 16-bit Intel formats. The EPP-1 applies the 8-bit version. With this format it's still possible to program memories for use in sixteen bit environments. The format is as follows:

:LLAAAA00DDDDDDDDSS

:	- Start sentile
LL	- Number of data bytes
AA	- Address of first byte in the string
DD	- Data byte. (example: 01001010B = 4A)
SS	- Sumcheck on entire string. The total sum of bytes including the sumcheck, excluding the start sentile is 00h.

Example of sumcheck: :02000000A20448

02h + 00h + 00h + 00h + A2h + 04h + 48h = 00h

So the sumcheck of this string is : 48h

APPENDIX -D-

"FPC" - "Four Packed Code" Format

This code format significantly decreases the time needed to send the file from the user interface to the EPP-1. The FPC format can be used by simply writing a conversion program based upon the following information.

Each string consists of 1+Lx4 blocks of five digits each, preceeded by a "$". Characters following the string are allowed, except for the "$". A digit represents a number from 0 through 84 (where "0" is "%", and "84" is "z". The character "*" is omitted. Each block of 5 digits represents 4 bytes of information. One block (the "header") is always present. It contains the sumcheck, length and code.

SLCC - where each letter represents a byte:
- S - sumcheck (causes the sum of all bytes in the string mod 256 to be "0".
- L - length of bytes following the header.
- C - code (information about what the data in the string represents).

(If all 4 bytes are 0, this is the last string transmitted.)

SLCCAAAADDDDDDD.... (code 0)- if the code is 0, the string has an Intel-like hex format.

L - length, number of bytes + 4 address bytes
A - absolute address
D - data bytes

The number of bytes need not be a multiple of 4, but if not, the block of 4 bytes (5 digits) has to be completed. Any remaining bytes are "meaningless". The data bytes do not have to be present. If only the address is present, the address pointer of the receiver is replaced by the new value.

SLCCDDDDDDD........ (code 1)- code 1 has no address...only data.
SLCCRRRRDDDDDDD.... (code 2)- code 2 indicates a relative address. The relative 32 bit value supplied here is added to the current address pointer.

L - length (=4)
R - relative address increment
D - data bytes