MQP ELECTRONICS

PROM PROGRAMMER SERIES II

USER MANUAL

MODELS 160 170 180 200

MQP Electronics Ltd
Unit 2
Park Road Centre
MALMESBURY
Wiltshire
SN16 0BX
UK

Tel. +44 (0)1666 825666
Fax +44 (0)1666 825141
e-mail sales@mqp.com
Website www.mqp.com

Issue 7 10/03

1. INTRODUCTION

This manual describes the Series II range of EPROM Programmers; Models 160, 170, 180 and 200.

The Programmers are designed to program all common EPROM types. Models 170 and 200 are equipped with a 32 pin ZIF socket as standard, and will program parts up to 1Mbit and beyond.

The other models have 28 pin sockets, but they can also be used to program 32 pin parts with the addition of a Socket Adapter.

This table summarises the capabilities of the models.

MODEL	160	170	180	200
STD SOCKET	28 PINS	32 PINS	28 PINS	32 PINS
24 PIN EPROMS	Y	Y	Y	Y
28 PIN EPROMS	Y	Y	Y	Y
32 PIN EPROMS	*	Y	*	Y
40 PIN EPROMS	*	*	*	*
EEPROMS			Y	Y
8 PIN SERIAL EEPROMS			*	*
8748 FAMILY			*	*
8751 FAMILY			*	*
EMULATORS			Y	Y

Y = YES * = WITH SUITABLE ADAPTER

All these Programmers are designed to operate with any host computer or terminal having an RS232C port. However best performance is obtained in conduction with the specially designed PROMDRIVER software.

The Programmers are microprocessor controlled, automatically selecting correct voltage, programming timings, and sequencing for the PROM type selected. Manufacturer approved fast programming algorithms are automatically employed where appropriate.

The firmware of the Programmers is regularly upgraded to keep pace with developments in the field and the upgrades are made available to existing owners at a nominal charge.

The internal RAM size was chosen to provide sufficient storage not to significantly slow down the transfer protocol. This provides future proofing as there is no limit to the size of PROM which can be programmed with an appropriate socket adapter and internal firmware.

Communication with the host computer is via a serial RS232C link. The data rate may be any standard rate between 300 b/s and 19200 b/s and the Programmers automatically adapt to the rate of the host computer.

A unique feature of the range is that two independent communication protocols are built in: 'Host Computer Protocol' and 'Terminal Protocol':

Host protocol

Host Machine Protocol is designed to permit reliable communication with a host computer, taking into account the range of capabilities of serial ports on different computer types. It allows the use of the MQP Electronics host computer software package 'PROMDRIVER' with its many advanced features.

PROMDRIVER is a versatile support program for MQP Electronics EPROM Programmers. It is designed to support MQP's Series II Programmers, Models 160, 170, 180 and 200.

PROMDRIVER runs on any MS-DOS/PC-DOS computer and provides advanced programming facilities to the user with an extremely helpful menu driven presentation. Its capabilities include displaying and printing of PROMs and files, PROM security and other special functions, byte splitting, data polarity control, prom set blowing, manual programming and the ability to

read and write a variety of industry standard object file formats.

Terminal mode protocol

Terminal Mode Protocol is designed to be used with a VDU or computer running a terminal emulation program connected to the Programmer. Messages which it sends are intended to be displayed directly on the terminal screen. This permits the Programmers to be used with any computer having a serial interface and a suitable 'terminal emulation' or 'communications' program. The use of the Programmer with a terminal emulator is covered in appendices 1,2,3 and 4.

Programmers of this versatility are the subject of continual technical review and MQP Electronics reserves the right to change details of the specification for the purpose of product improvement.

2. DEFINITIONS

HEXADECIMAL VALUE REPRESENTATION.

Hexadecimal values are represented in this manual by following them with H. e.g. 0DH represents the hexadecimal number 0D i.e. the decimal number 13.

ASCII

American Standard Code for Information Interchange. The method of encoding characters (letters, numbers, punctuation, and controls) into 7 bits. Specific examples used in the Programmer description are:

<cr> represents the ASCII Carriage Return character (value 0DH)
<lf> represents the ASCII Line Feed character (value 0AH)

<BREAK>

BREAK is defined in the Programmer by a non-quiescent (positive) voltage on the TXD input lasting at least 67 ms. As it is not always possible to generate the BREAK character and maintain software portability, the ASCII '<196>' character is allowed as an alternative. In this case the character must be sent at the speed being used by the Programmer at the time.

<ENTER>

Represents a press of the ENTER, RETURN, CR or CARRIAGE RETURN key on the user terminal keyboard.

EPROM

Erasable Programmable Read Only Memory. Usually means erasable by means of UV light.

PROM

Programmable Read Only Memory.

RS232C

American modem interface standard, very similar to a combination of the CCITT standards V24 and V28. The interface is widely used (or misused) as a form of interface for connection between various items of computer equipment. As RS232C was never intended for this use, it is a constant cause of headaches for computer users. However you should encounter no problems if you use a lead supplied by MQP Electronics with the Programmer.

TERMINAL EMULATOR

(also known as Communications Program)

A program which is run on a computer (usually with a serial interface) which makes the computer appear to be a VDU or terminal (perhaps of a particular type). Usually permits the capture of received data to a file, and the ability to transmit data from a (text) file. Examples are the public domain programs MODEM7 and SIMTERM, and commercial programs like Communicator (from Computer Concepts).

ZIF

Zero Insertion Force. Refers to the type of socket used for the PROMs to be programmed.

3. SYSTEM REQUIREMENTS

In order to run PROMDRIVER successfully, the following system requirements must be met.

MSDOS 2.0 or later.

Windows (Parallel version only).

RAM requirement. Program space of 512k

Serial asynchronous RS232C port: 300, 600, 1200, 2400, 4800, 9600 or 19200 baud.

Parallel Port.

4. INSTALLATION

4.1. Hardware Installation

The only hardware installation required is to fit a mains plug and connect the cable.

Connect the programmer to your PC using the lead provided, either to the serial or to the parallel port depending on which programmer option you have chosen.

Hint: If you are not sure if you have the parallel option board installed, switch on your programmer and watch the red socket active LED. One flash on powering up indicates serial communications and two indicates parallel.

If you are not using PROMDRIVER you will need to ensure that the host system serial communication characteristics are correctly set. The last of these is usually a software function, involving running a short Serial Port Parameter Setting program. If hardware adjustments are required then refer to the parameter details in appendix 4.

4.1.1. Mains Power Connection

IMPORTANT SAFETY INFORMATION.

WARNING: THIS APPARATUS MUST BE EARTHED.

It is essential that the Programmer is correctly earthed through the use of a correctly connected 3 pin mains plug.

IMPORTANT: The wires in the mains lead are coloured in accordance with the following code:

Green-and-yellow	: Earth
Blue	: Neutral
Brown	: Live

As the colours of the wires in the mains lead may not correspond with the coloured markings identifying the terminals in your plug proceed as follows:

The wire which is coloured green-and-yellow must be connected to the terminal in the plug which is marked by the letter E, or by the safety earth symbol, or coloured green or green-and-yellow.

The wire which is coloured blue must be connected to the terminal which is marked with the letter N or coloured black.

The wire which is coloured brown must be connected to the terminal which is marked with the letter L or coloured red.

If using a 13A mains plug it must be fitted with a 3A fuse.

Check on the rear panel that you have the correct unit for the local mains voltage.

4.1.2. RS232C Connection

The Programmer is equipped with a female 25 way D type connector. A male to female RS232C cable is required; cables for both the PC/XT, and the AT are available. If any other type of lead is required please contact MQP Electronics.

Appendix 2 contains a detailed description of the RS232C interface.

4.2. PROMDRIVER Installation

The software can quickly be installed by following the instructions on the MQP Electronics website or on the disc supplied with your programmer.

The first time PD is run it will call a simple installation program for you to set up the communications method with your Programmer and printer.

And that's all there is to it. You can now run PD regardless of which directory you are in; your settings will be saved in each directory you use.

The serial port speed will govern the rate at which you can communicate with your PROM Programmer. The maximum rate possible will depend on your computer. By default, PROMDRIVER uses 9600 Baud, which should work on all close compatibles.

If your clock rate is greater than 4.77MHz you should be able to select 19200 Baud. To change your communications settings just type PD/I instead of PD

5. WARNINGS

There are some certain ways to destroy EPROMs, and possibly even to damage your Programmer. Please read the warnings in this section.

NEVER switch the Programmer on or off with a PROM in the socket.

NEVER change the PROM type with a PROM in the socket

NEVER insert a PROM of a type other than that selected.

NEVER insert a PROM or socket adapter upside down or otherwise misaligned.

NEVER remove (or insert) a PROM while the Programmer Active light is on.

BEWARE of misleading PROM type numbers. There have been cases of different types being given the same number. There are even cases of one manufacturer producing, over the years, devices with the same type number but incompatible methods of programming. If in doubt check the data sheet for your device against one from the PROM TYPE LIST in appendix 6. The algorithms used are briefly described below that list.

6. PROMDRIVER OPERATION - MENU MODE

The user interface provides pop-up windows for all functions.

Context sensitive HELP is available by pressing F1 and all options available are indicated on screen

7. PROMDRIVER OPERATION - COMMAND LINE MODE

A complete and accurate detailed description of the batch mode commands is given in the help utility accessed using F1 from the main menu mode screen of PROMDRIVER or by running PDHELP

PROMDRIVER may be invoked by a single command line with complete instructions for programming a PROM or PROM set. This mode is available to allow the use of PROMDRIVER by non-technical personnel, or as part of an MS-DOS batch file program. It allows a single MS-DOS command line to program a PROM or PROM set either as a once off event or repetitively.

Suppose an end user frequently manufactures a small production run of a product. He wants to be able to blow a number of identical EPROMs from a file on the disk called PRODUNIT.HEX. The file is in Intel Hex format, and is always to be programmed into a 27128A. The user wants to be able to type the word 'BLOW' at the MS-DOS prompt, and be instructed exactly what to do to program his devices. To achieve this you should create for him an MS-DOS batch file called BLOW.BAT. In its simplest form the batch file would contain a single line of text as follows:

PD T=31 O=3 P=PRODUNIT.HEX M

The command line comprises 'PD' followed by 4 instructions. Notice that the instructions are all separated by space characters and that there are no space characters within an instruction. The instructions will be carried out in the order you specify them. They are shown in upper case for clarity but may equally well be in lower case.

The command

PD

will invoke PROMDRIVER which will switch to batch mode because you have included instructions in the command line.

T=31

sets the device type to 27128A. 31 is the PROMDRIVER type number for this device type.

O=3

sets the Object file format to INTEL HEX.

3 is the select file format number in the PROMDRIVER menus.

P=PRODUNIT.HEX

causes the device to be programmed from the file named. It is mandatory to have specified the device type before this instruction. The user will see information concerning the progress of programming, together with the opportunity to retry in the event of failure.

M

causes a message 'Again (Y/N) ?' to be displayed at the end of the command line. If the user responds with a 'Y', the whole sequence of instructions in the command line is repeated. The 'M' instruction can appear anywhere on the command line.

Note that because we have not specified otherwise, the following defaults are assumed:

File Address = 000000
Polarity = TRUE
Byte Select = ALL (NORMAL)

A similar command line could have programmed two sets of PROMs, one containing the ODD bytes, the other the EVEN bytes of the file. If the target device had been a micro-processor with a code locking function, the device could be locked as part of the same command line.

See the examples below. A useful feature is the option to request a parameter from the user at the time of programming. The normal parameter is replaced with a question mark to achieve this.

To assist in the testing of new batch commands, an instruction is available to display the effects of each batch instruction. /D should be included in the command line and removed when testing is complete. Using the example above you can run the debug mode with the modified command:

PD /D T=31 O=3 P=PRODUNIT.HEX M

7.1.Syntax

* Arguments will be performed in the order present.

* Arguments are separated by one or more spaces.

* Arguments may not have embedded spaces.

* Any parameter can be replaced by ? or ?? in which case that parameter will be requested from the user. ? requests the parameter only the first time if the M instruction is selected.

* All parameters except filename, file address, and security parameter are decimal numbers equal to the normal responses to menu mode questions.

7.2.Defaults

Parameters not defined are assumed to be their usual default as follows:

OBJECT FILE FORMAT	BINARY OR INTEL .HEX
FILE ADDRESS	000000
POLARITY	NORMAL
BYTE SELECT	ALL (NORMAL)

7.3.Instructions

A comprehensive list of instructions available in batch mode may be found in the on-line HELP.

8. PROBLEMS

Whether you are using PROMDRIVER or a Terminal Emulator, the problem you are most likely to encounter is an apparent total lack of communication. The following comments may help.

The serial lead must be made up as described or be one supplied by MQP Electronics. Attempts to simplify the lead wiring will not succeed.

In Terminal Mode remember that DTR (pin 20) of the Programmer RS232 connector must be ON (greater than +6V relative to pin 7) for it to operate. If your computer cannot or does not turn it on then connect pin 20 to pin 8 of the Programmer RS232 connector.

If you have had a communication failure and think you have found the reason, switch the Programmer off and on again and restart PROMDRIVER from scratch.

After the Programmer has established its data rate you can only CHANGE data rate if the Programmer receives a '-' character at its current rate, or a<BREAK> character. In other circumstances you must switch the Programmer off and on again.

The following are the more common problems you may encounter:

"Unless I use a baud rate of 2400 or less I keep getting 'Communication errors'."

You may not have installed PDINIT.DAT correctly. For PROMDRIVER to be aware of the location of PDINIT.BAT the environment must contain PDDIR=<directory name> This is achieved by typing (for example) SET PDDIR=\PD. This command is normally included in the AUTOEXEC.BAT file executed by the system at start-up.

"The various menus are displayed on the screen but they include some odd characters (<[2J)."

This will not occur if PROMDRIVER is installed in the standard way. If you have modified PDINIT.DAT to use communication modes 0, 1 or 2; or if you have not put the 'SET PDDIR=' command into your AUTOEXEC.BAT then PROMDRIVER assumes you have installed ANSI.SYS. The odd characters are cursor positioning strings. ANSI.SYS is a standard MS-DOS/PC-DOS system file.

"When in terminal emulator mode the Herald is displayed but no prompt appears. Instead I keep seeing 'NS' messages."

The Programmer is using Host Mode Protocol. Pins 23 and 8 should be linked, alternatively type M<cr>.

"The Programmer just locks up after I issue a program, read or verify command."

First check the serial lead as explained above.

Although the host protocol and PROMDRIVER have been designed to make best use of the host computer serial port, it is possible that the host may not be capable of handling the full 19200 bits/second rate without losing characters.

For example an IBM PC compatible using the MSDOS Auxiliary driver is limited to around 2400 bits/sec in this application (although correct installation as explained in the appropriate section normally permits operation at 9600 or 19200 bits/sec).

If you have problems on a particular computer with data loss as evidenced by 'COMMUNICATION ERROR' messages or lock-ups, then try running the serial link at lower rate.

Unfortunately, even using interrupt control and hardware handshaking as PROMDRIVER does in modes 3 to 12, there is nothing that can be done if data from the Programmer arrives at a rate that the interrupt routine cannot keep up with. This usually happens when other interrupt driven software steals time away from PROMDRIVER. This other software may be resident software loaded by AUTOEXEC.BAT or CONFIG.SYS, or left resident after use like PRINT.EXE.

If unsure, try booting up with empty AUTOEXEC.BAT and CONFIG.SYS files. If this cures the problem, try putting back the contents of those files a bit at a time.

Finally a tip for the situation where the Programmer locks up with the ACTIVE light on and an expensive device in the socket (as a result of using too high a communications rate):

- DO NOT remove device with light on.
- Reboot your PC (CTRL/ALT/DEL).
- Type PD to start PROMDRIVER.
- PROMDRIVER will reset the Programmer safely.
- Now, with the ACTIVE light off, remove your PROM.

9.FILE FORMATS

Object files up to 16 Mbytes in size are accommodated by PROMDRIVER.

Files are permitted to be in any of the following formats. (Note that some versions of PROMDRIVER may omit certain file formats.)

9.1.Binary Image

This format is so called because each 0 or 1 required in the prom is represented by a 0 or 1 in the file. The individual bytes of the file represent the individual bytes in the prom in a one to one relationship. No address information is included in such a file. The file may be shorter than a prom in which case PROMDRIVER will fill the rest of the PROM with erased byte value. It may be the same size as the PROM; or it may be larger than the PROM. In the latter case, bytes in excess of those required to fill a PROM are assumed to be intended for the next PROM in a set.

When programming or verifying, the first byte in the file is assumed to correspond to address 000000. This does not prevent the PROM being mapped at some different address in subsequent use, but merely provides a convenient way to refer to the contents of the file. The File Address setting function may be used to program or verify a PROM from any convenient point within the file. For example, setting the File Address to 002000, and the PROM type to 2764 would allow programming to occur starting with the (2000H)th byte in the Binary Image file, and thus relate to the second PROM in the set.

When reading a PROM to a file the File Address offset has no effect on a Binary Image file. The number of bytes in the file will always equal the number of bytes in the PROM.

For normal use with PROMDRIVER the Binary Image format is recommended; as it involves less processing the system response is much faster.

9.2.FreeformHexadecimal Format

This is a printable file format in which each byte is represented by two ASCII hexadecimal characters, which are separated from adjacent bytes by spaces or new lines. Like binary files no address information is included in the file.

PROMDRIVER generates such files with a space character after each byte, and a <cr> <lf> pair after each 16 bytes, e.g.:

```
03 36 5F 3D C4 DD 02 B3 65 7D 91 04 D3 7A 56 8A
7C 45 F3 6D 74 28 D4 6E CD C3 C9 56 D6 94 3D 56
```

When reading in this format PROMDRIVER will regard sequences of valid hexadecimal characters as bytes (ignoring overflow) and any other characters (including new lines) as separators, thus:

```
1,2,3,C,D

01,02
03,0C,0D

101:102:
103:F0C:30D
```

are all equivalent.

9.3.Freeform Decimal Format

This format is the same as Freeform Hexadecimal except that up to 3 decimal characters are used to represent each byte.

9.4.Intel Hexadecimal

(Intel 8080 Hex ASCII and Intel 8086 Hex ASCII)

This format combines the Intel 8080 Hex ASCII and the Intel 8086 Hex ASCII formats. The latter is a superset of the former. PROMDRIVER reads both formats and generates the 8080 subset wherever possible.

The Intel Hexadecimal format is a way of representing an object file in a format which may be directly displayed and read, as all characters within the file are printable ASCII characters, or <cr> or <lf>characters.

Binary byte values are represented by a hexadecimal representation of the byte coded in ASCII. For example the eight bit value 01011100 is 5C in hexadecimal. To code this in ASCII the 2 bytes 00110101 (ASCII for 5) and 01000011 (ASCII for C) are required.

A hexadecimal file is made up of records. There are 4 possible record types:

Extended address Record

RECD MARK	REC LEN	ZEROES	REC TYP	USBA	CHK SUM		
':'	'02'	'0000'	'02'	'HHHH'	'CC'	<CR>	<lF>

Data Record

RECD MARK	REC LEN	LOAD ADDR	REC TYP	DATA	CHK SUM		
':'	'HH'	'HHHH'	'00'	'HH,HH'	'CC'	<CR>	<lF>

Start address Record

RECD MARK	REC LEN	ZEROES	REC TYP	CS	IP SEGM	CHK SUM		
':'	'04'	'0000'	'03'	'HHHH'	'HHHH'	'CC'	<CR>	<lF>

End of file Record

RECD MARK	REC LEN	ZEROES	REC TYP	CHK SUM		
':'	'00'	'0000'	'01'	'FF'	<CR>	<lF>

where H represents a hexadecimal character (0 1 2 3 4 5 6 7 8 9 A B C D E F) coded in ASCII,

CC is a single byte (2 character) checksum which is the two's complement of the sum (modulo 256) of all the bytes (not characters) after the record mark but before the checksum in a record, and USBA is the Upper Segment Base Address; which is multiplied by 16 and added to the following record Load Addresses to produce the actual load addresses. USBA is assumed to be 0000 until an Extended Address Record is encountered.

PROMDRIVER will read files containing any of these record types.

When creating object files, it will never generate a Start Address Record, and will only generate an Extended Address Record if it is necessary. Therefore if a PROM address plus File Address offset exceed 00FFFFH then an Extended Address Record will be generated.

Hexadecimal files contain address information. When PROMDRIVER creates a Hexadecimal file it commences generating addresses from the value of the File Address.

Similarly when programming or verifying a PROM from a file, the Prom Offset must be set to the address within the file which corresponds to the first byte to be programmed into the PROM.

Intel hexadecimal files are limited to 1 Mbyte in size.

To allow compatibility with Digital Research Hex files a Record with a record length of 00 is regarded as an End of File Record.

9.5.Motorola Hexadecimal

(Exorciser and Exormax formats)

This format combines the Exorciser format and the Exormax format. The latter is a superset of the former. PROMDRIVER reads both formats and generates the Exorciser subset wherever possible.

Sign on record

START CHARS	BYTE COUNT	LOAD ADDR	DATA	CHK SUM		
'S0'	'HH'	'HHHH'	'HH..HH'	'CC'	<CR>	<IF>

Data record (2 byte address).

START CHARS	BYTE COUNT	LOAD ADDR	DATA	CHK SUM		
'S1'	'HH'	'HHHH'	'HH..HH'	'CC'	<CR>	<IF>

Data record (3 byte address).

START CHARS	BYTE COUNT	LOAD ADD	DATA	CHK SUM		
'S2'	'HH'	'HHHHHH'	'HH..HH'	'CC'	<CR>	<IF>

End of File Record.

START CHARS	BYTE COUNT	LOAD ADDR	CHK SUM		
'S8'	'03'	'0000'	'FC'	<CR>	<IF>
'S9'	'03'	'0000'	'FC'	<CR>	<IF>

where H represents a hexadecimal character (0 1 2 3 4 5 6 7 8 9 A B C D E F) coded in ASCII,

CC is a single byte (2 character) checksum which is the one's complement of the sum (modulo 256) of all the bytes (not characters) after the start characters but before the checksum in a record.

The BYTE COUNT is the total number of bytes following in the record, including the checksum.

PROMDRIVER will read files containing any of these record types.

When creating object files, it will never generate an S0 or S8 record, and will only generate an S2 record if necessary. Therefore if a PROM address plus File Address offset exceed 00FFFFH then an S2 record will be generated.

Hexadecimal files contain address information. When PROMDRIVER creates a Hexadecimal file it commences generating addresses from the value of the File Address.

Similarly when programming or verifying a PROM from a file, the Prom Offset must be set to the address within the file which corresponds to the first byte to be programmed into the PROM.

Motorola hexadecimal files are limited to 16 Mbyte in size.

9.6.Intel 8080 Absolute Object

Both 8080 and 8086 Absolute Object formats are formed from a series of contiguous records each having the same basic format.

RECORD TYPE	RECORD LENGTH	CONTENTS RECORD	CHECK SUM
1 BYTE	2 BYTE	(R(RECORD LENGTH - BYTES	1 BYTE

All bytes are represented directly in the file by their binary values. All multiple byte values are ordered least significant byte first. The CHECKSUM is the 2's complement, modulo 256, of all the other bytes in the record.

In 8080 Absolute Object format PROMDRIVER is only concerned with the following record types. (These descriptions are not rigorous and only describe the records to the extent that they are used by PROMDRIVER.)

MODULE HEADER RECORD.

RECORD TYPE	RECORD LENGTH	MODULE NAME LENGTH	MODULE NAME	RSVD	CHECK SUM
02	LLLL	LL	N...N	00 00	CC

CONTENT RECORD

RECORD TYPE	RECORD LENGTH	SEG ID	OFFSET	DATA BYTES	CHECK SUM
06	LLLL	00	AAAA	DD... DD	CC

MODULE END RECORD

RECORD TYPE	RECORD LENGTH	MOD TYPE	SEG ID	OFFSET	CHECK SUM
04	05 00	00	00	0000	F7

END OF FILE RECORD

RECORD TYPE	RECORD LENGTH	CHECK SUM
0E	01 00	F1

where:

LLLL is 2 byte length.

AAAA is 2 byte load address for the data bytes.

DD is a data byte. The number of data bytes is only limited by the record length.

CC is the checksum byte.

For an absolute object record the Segment ID must be 0.

LL is a one byte name length.

N...N is a variable length ASCII string (the module name).

RSVD is a reserved field of 2 zero bytes.

PROMDRIVER responds to CONTENT records, and requires that the file ends with an END OF FILE record.

All other record types are ignored, except for illegal types which cause a file format error to be reported.

When creating object files PROMDRIVER generates:

a Module Header record
a variable number of Content records
a Module End record
an End of File record.

The MODULE NAME generated is

MQP@ELECTRONICS

Intel 8080 Absolute Object files are limited to 64 Kbytes in size.

9.7.Intel 8086 Absolute Object

This format is made up of records having the same basic form as for 8080 Absolute Object.

RECORD TYPE	RECORD LENGTH	RECORD CONTENTS	CHECK SUM
1 byte	2 byte	(record length -1) bytes	1 byte

The following record types are relevant to PROMDRIVER operation:

THEADR RECORD

RECORD TYPE	RECORD LENGTH	MODULE NAME LENGTH	MODULE NAME	CHECK SUM
80	LLLL	NL	N…N	CC

LHEADR RECORD

RECORD TYPE	RECORD LENGTH	MODULE NAME LENGTH	MODUL NAME	CHECI SUM
82	LLLL	NL	N...N	CC

MODEND RECORD

RECORD TYPE	RECORD LENGTH	MODULE TYPI	
8A	02 00	00	74

PEDATA RECORD

RECORD TYPE	RECORD LENGTH	FRAME NO	OFFSET	DATA BYTES	CHECK SUM
84	LLLL	AAAA	0A	DD...DD	CC

PIDATA RECORD

RECORD TYPE	RECORD LENGTH	FRAME NO	OFFSET	ITERATED DATA BLOCK	CHECK SUM
86	LLLL	AAAA	0A	DD....DD	CC

ITERATED DATA BLOCK.

REPEAT COUNT	BLOCK COUNT	CONTENT

REPEAT COUNT is the number of times the CONTENT field is to be repeated.

If BLOCK COUNT is non zero then the CONTENT field is that number of

If BLOCK COUNT is zero then the CONTENT field contains a DATA COUNT byte followed by that number of data bytes.

Maximum iteration depth is 17.

Load address is obtained from (FRAME NO * 16)+ OFFSET.

OFFSET must be less than or equal to 0FH.

PROMDRIVER responds to PEDATA and PIDATA records, and requires that the file ends with a MODEND record. All other record types are ignored, except for illegal types which cause a file format error to be reported.

When creating object files PROMDRIVER generates:

an LHEADR record
a THEADR record
a variable number of PEDATA records
a MODEND record

The MODULE NAME generated is

MQP_ELECTRONICS

Intel 8086 Absolute Object files are limited to 1 Mbyte in size.

9.8.Intel 286 Absolute Object

This file format comprises the following sequence:

a FILE HEADER (1 byte)
a BOOTLOADABLE MODULE HEADER (95 bytes)
a variable number of ABSTXT fields (variable length)

a CHECKSUM (1 byte)

FILE HEADER.

One byte with the value A2H

BOOTLOADABLE MODULE HEADER

TOTAL SPACE	RSVD	ABSTXT LOCATIOI	RSVD	LAST LOCATI	RSVD
4 BYTE:	71 ZERO BYTES	4 BYTES	4 ZERO BYTES	4 BYTE:	8 ZERO BYTES

TOTAL SPACE is a 4 byte value indicating the minimum number of bytes in main memory needed to load the module.

ABSTXT LOCATION is a 4 byte value representing the offset in bytes of the ABSTXT section from the start of the file.

LAST LOCATION is a 4 byte value representing the offset in bytes of the last byte in the file from the start of the file.

ABSTXT field

REAL ADDRESS	LENGTH	TEXT
3 BYTES	2 BYTES	LENGTH BYTES

CHECKSUM is a single byte which is the complement mod 256 of all the previous bytes in the file.

Intel 286 Absolute Object files are limited to 16 Mbytes in size.

10.HOST COMPUTER PROTOCOL

The Series II PROM Programmer communicates with its host computer via a serial RS232C link. The Host Computer Protocol was developed to provide a secure data transfer which at the same time may be implemented on host computers without the limitations of the serial port on some machines causing problems.

This section will present a brief summary of the Protocol. A comprehensive specification can be purchased for a nominal fee from MQP Electronics.

The Protocol has two phases:

Automatic Baud Rate Selection phase

and Normal Command Mode.

After switch-on, the Programmer is in the Baud Rate Selection phase, until Baud rate selection has been satisfactorily accomplished, after which it is in the Normal Command mode. The Baud Rate Selection mode may be re-entered by transmitting a '-' character (ASCII character 2DH) or a <BREAK> condition to the Programmer.

Baud Rate Selection.

The Rate Selection Sequence comprises:

A <BREAK> condition or a '-' character followed, after 0.25 seconds, by 4 ASCII <cr> characters, preferably sent at intervals of approximately 0.25 seconds.

A wait (lasting about 1.25 seconds) for a sign-on message from the Programmer which provides version information and terminates with the characters <cr><lf> (ASCII 0DH 0AH). The message has a maximum permitted length of 81 characters including the <cr><lf> characters.

Normal Command Mode.

The following list summarises the commands accepted by the Programmer.

Blankcheck

B<cr>

Checksum

C<cr>

Request Model Description.

D<cr>

Echo control

E01<cr> Echo ON
E00<cr> Echo OFF

Fast Transfer Commands FQ<cr>

FB <no of bytes><<addr><cr>
FC<no of bytes><addr>< cr>
FP<no of data bytes><start addr>
<data byte >data byte<checksum><cr>
FR<no of bytes><addr><cr>

PROM In socket check

I<cr>

Enter TERMINAL (Monitor) mode

M<cr>

No verify control

N01<cr> Set No verify Mode
N00<cr> Set Verify Mode

Program

P<no of data bytes><start addr> <data byte>...<data byte><checksum> <cr>

Read

R<addr><cr>

Security programming

S<no of bytes><sec type> [<param>...<param>]<checksum><cr>

Type set

T<type no><cr>

Request Version No.

V<cr>

Enter test mode

Z

Enter Baud Rate Selection Phase

- or <BREAK>

Possible Responses from the Programmer:

Y<cr><lf>	Command performed no error
Y[<param>...<param>]<cr><lf>	Command performed -result
NB<addr><cr><lf>	PROM is not blank
NT<cr><lf>	Type error.
NS<cr><lf>	Syntax error
NI<cr><lf>	PROM in socket error
NP<addr><cr><lf>	Failure to program

Commands are sent to the Programmer. Responses are received from the Programmer as a result, but are never sent by the Programmer until the complete Command has been received (except during fast transfer commands which allow special concessions to achieve high transfer rates). The host computer also follows this practice, waiting for a complete response before transmitting any characters of the next command. The responses from the Programmer may take up to 40 seconds from receipt of Command, so the Driver must not assume a communication failure if there is not an immediate response.

11.CALIBRATION AND SERVICING

The Series II PROM Programmer is guaranteed for one year against any defects of manufacture. Please note that we cannot be responsible for any unit which has been opened by the user or has in the opinion of our service engineers been misused.

The Series II PROM Programmer contains a small number of adjustable components which are calibrated during manufacture. These components control the special voltages used for Programming various PROM types. Under normal circumstances it is unlikely that any further attention need be paid to these adjustments.

If recalibration appears necessary, as evidenced perhaps by a difficulty in achieving reliable programming, or if your company has a requirement for a calibration certificate on its items of equipment then the Programmer may be returned to MQP Electronics for servicing or calibration. Please contact us first for information about the charge for this.

Make allowance for the fact that EPROM devices are not indefinitely reprogrammable, and can also very easily be permanently damaged by mishandling. Please consider this before assuming a fault in the Programmer.

A good way to destroy an EPROM and also to damage the Programmer is to define the wrong PROM type before programming. Such damage is readily detectable by our engineers, and would have to be regarded as being caused by mishandling.

MQP Electronics cannot be responsible for damage to PROM devices or consequential loss, whether or not caused by the Programmer.

Appendix 1.Terminal Mode

This appendix is included for reference purposes only, for those customers who already use Terminal mode.

MQP no longer recommends the use of Terminal Mode, as its limitations prevent its being generally useful.

There is no customer help service for questions relating to the use of Terminal Mode.

1. System Requirements

Terminal Mode is intended for situations where an MS-DOS computer is not available. The user will need to provide a computer running a serial communications package.

To use the Programmer in Terminal Mode with all functions the requirement is:

Host computer with Serial asynchronous RS232C port: 300, 600, 1200, 2400, 4800, 9600 or 19200 baud.

Terminal Emulation Program with terminal file capture and file transmit capability, using a suitable form of handshaking.

Console screen at least 80 columns by 24 rows.

2. Terminal Mode Operation

In this mode all messages sent from the programmer are intended to be displayed directly on the screen of the user terminal.

2.1.Starting up

If using a terminal emulation program, start running this from the normal operating system level of your computer.

2.2.Entering a Command

Each command consists of a 1 or 2 letter command mnemonic possibly followed by 1 or 2 hexadecimal parameters: e.g.

2764 > D 0 FF

is an instruction to display the contents of the PROM in the socket between the addresses 0000h and 00FFh. Spaces must separate the parameters from each other and from the command mnemonic letters.

A command is not actioned until <ENTER> is pressed. Until it is the command may be modified using the <backspace> or <delete> keys, or cancelled by typing $.

2.3.Command List

A complete list of commands is available at any time by entering the HELP command H <ENTER> which will display the following menu:

D <addr> <addr>	:Display
DI <addr> <addr>	:Display (Intel)
DM <addr> <addr>	:Display (Motorola)
DH <addr> <addr>	:Display (plain Hex)
P <addr>	:Program (manually)
PI [file offset [prom offset]]	:Program (Intel)
PM [file offset [prom offset]	: Program (Motorola
PH <addr> <addr>	: Program (plain Hex)
PB <addr> <addr>	: Program (Binary)
VI [file offset [prom offset]	: Verify (Intel)
VM [file offset [prom offset]]	: Verify (Motorola)
VH <addr> <addr>	: Verify (plain Hex)
VB <addr> <addr>	: Verify (Binary)

B: Blankcheck	C: Checksum	E: file Echo
H: Help	R: firmware Rev	S: Status
T: Set type	X:.Exchange Bytes	$: abort command
- or <BREAK>	:reset	[]=optional

2.4.Command Descriptions

B

Blankcheck

Checks whether the PROM in the socket is fully erased and reports result.

C

Checksum

Calculates PROM checksum by adding together the values of all the bytes in the PROM.

D <addr1> <addr2><R>

Display PROM contents between addresses entered, e.g.

2764 > D 0 13
0000 C3 04 07 FF 41 42 43 44 45
39 38 37 36 AB CD EF
0010 98 87 76 54
2764 >

The display may be paused by pressing the CTRL and the S keys of the user terminal simultaneously and may be restarted by pressing CTRL and Q simultaneously. It may be aborted by pressing $.

DI <addr1> <addr2>

DM <addr1> <addr2>

DH <addr1> <addr2>

Display PROM contents between addresses entered in:

Intel Hexadecimal Format

Motorola Hexadecimal Format

unformatted Hexadecimal

Each of these 3 commands is intended to be used to create a file using the file capture facility of a terminal emulator program. The procedure is:

a.) Enter the display command except for the final <ENTER>.

b.) Instruct the terminal emulator program to start saving to a given file.

c.) Complete the command with the <ENTER> key.

d.) The PROM will be displayed in the appropriate format. When complete the 'Socket Active' lamp will be extinguished. No prompt is yet displayed, to avoid the prompt being added to the saved file.

e.) Terminate the file capture by the terminal emulator.

f.) Press the <ENTER> key. The prompt will now be displayed.

E

File Echo toggle

While programming or verifying from a file the Programmer may be set to echo each character sent to it by the terminal emulation program, which may use the echoed character as a means of data flow control. If flow control is implemented by RS232 control line or by XON/XOFF character protocol it may be required that the Programmer not echo during object file reception.

Each time E<ENTER> is typed, the state of the file echo function is toggled and the new state is reported, e.g.

8755 >

File echo ON

8755 >

H

Help

Displays the help menu shown above.

P <addr>

manual Program

Permits manual programming of sequential bytes of the PROM.

Each address is displayed along with the current contents of that location, followed by a prompt inviting a value to be programmed at the location.

Enter a hexadecimal value to program a byte, or <ENTER> to skip to the next byte, or $ to terminate the command. The message 'failed' will be displayed next to any unsuccessfully programmed byte.

Remember that bits which are already programmed will remain programmed.

e.g.

8751 > P 345

0345 FF-04

0346 FF-87

0347 00-01 failed

0348 01-00

0349 FF-$

8751 >

PI <filoff><promoff>

PM <filoff><romoff>

PH <addr1> <addr2>

PB <addr1> <addr2>

Program from Intel Hexadecimal format

Program from Motorola Hexadecimal format

Program from unformatted Hexadecimal

Program from Binary

<filoff> is File Address corresponding to start of PROM.

<promoff> is desired offset in prom.

a.) Enter command including <ENTER>.

b.) Instruct terminal emulation program to transmit the file containing the object file in the appropriate format.

c.) The file will be transmitted and programming will take place. The programmer will control the data flow as it programs the bytes of the PROM.

d.) On completion the 'Socket active' lamp will be extinguished (may also be caused by incorrect format), the file echo (if not deselected) will show that the transfer has ceased, and the terminal emulator program may report that the transfer has finished.

e.) Type $ to terminate the command. A report on the success of the operation will be displayed.

Note. Binary file programming will not be echoed and will terminate when, and only when the number of bytes specified in the command has been received by the Programmer. To use the PB or VB commands the terminal emulation program must employ either RS232 control line flow control or XON/XOFF flow control.

R

firmware Revision no.

Displays the version number of the Programmer firmware, e.g.

8748A > R

Rev V3.60

8748A >

S

Status report

Reports the result of the most recent attempt to program or verify a PROM, e.g.

27513> S

Failed at 87D4

27513 >

SD <value>

Scroll Delay

Sets delay after <cr> <lf> combination sent by Programmer, in units of about .008 milliseconds from 1 to ffff (hexadecimal). Default setting is 1000 or about 33 ms. Consider varying this if your Terminal Emulator loses

characters from the Programmer as the result of the time taken to perform a software scroll.

T

Type set

Displays PROM type menu and requests type required, e.g.

```
NO TYPE>T
01 : 2508/10MS02 : 2508/50MS    03 : 2516/10MS  04 : 2516/50MS
05 : 2532/10MS06 : 2532/50MS    07 : 2564/10MS  08 : 2564/50MS
09 : 2758       10.:2716:       11 : 2732       12 :2732A/10MS
13 : 2732A/50M14 : 2764/50MS    15 : 2764       16 : 2764A
17 : 27128      18 : 27128A     19 : 27256      20 : 27256/21V
21 : 27512      22 : 27513      23 : 87C64      24 : 87C256
25 : 8755       26 : 8755A      27 : 8355 *     28 : 8748
29 : 8749       30 : 8750       31 : 8748H      32 : 8749H
33 : 8750H      34 : 8741       35 : 8742       36 : 8041 *
37 : 8042 *     38 : 8048 *     39 : 8049 *     40 : 8050 *
41 : 8751       42 : 8752/21V   43 : 8744       44 : 8051 *
45 : 8052 *     46 : 8044 *     47 : 87C51      48 : 63705V
49 : 75P54/6    50 : EMULATC    51 : 2816A      52 : 2817A
 53 : 2864A
Type 18
27128A>
```

May also be used with type number to save displaying the menu, e.g.

NO TYPE > T 10

2716 >

VI <filoff><promoff>

VM <filoff><romoff>

VH <addr1> <addr2>

VB <addr1> <addr2>

Verify from Intel Hexadecimal format

Verify from Motorola Hexadecimal format

Verify from unformatted Hexadecimal

Verify from Binary

Procedure is as for PI, PM, PH, PB. The file will be compared against the PROM and the report will display the first address which did not match, if any.

X

EXCHANGE bytes command

This command reverses the default setting for the byte sequence when programming 16 bit wide PROMs.

Default settings are:

BINARY	MOST SIG BYTE	HIGHER ADDRESS
HEXADECIMAL	MOST SIG BYTE	HIGHER ADDRESS
INTEL	MOST SIG BYTE	HIGHER ADDRESS
MOTOROLA	MOST SIG BYTE	LOWER ADDRESS

e.g.
27210 X
Bytes REVERSED
27210

$

use to abort any command

Note. Program Binary and Verify Binary can only be aborted by providing the pre-specified number of bytes, or sending a <BREAK> or by cycling the mains switch. Remember that turning off the Programmer is likely to damage the PROM in the socket.

- **or <BREAK>**

Reset Programmer.

Resets the Programmer software except that the PROM type remains set (to prevent accidental damage caused by changing type).

may be keyed at any time (including characters sent as part of an object file) except while programmer is expecting a binary file.

<BREAK> will be accepted at ANY time by the Programmer

Appendix 2.RS232C Lead

The Programmer is equipped with a female 25 way D type connector. Only pins 2, 3, 5, 6, 7, 8, 20 and 23 are connected. The pinout is designed so that connection to an IBM PC, XT or compatible may be achieved using an off-the-shelf male to female RS232C cable which connects at least pins 2, 3, 5, 6, 7, 8 and 20.

PC Connection Chart.

IBM PC 25 PIN MALE			IBM AT 9 PIN MALE			MOP SERIES II 25 PIN FEMALE		
1	GROUND							
2	TXD	O/P	3	TXD	O/P	2	TXD	I/P
3	RXD	I/P	2	RXD	I/P	3	RXD	O/P
4	RTS	0/P	7	TRS	O/P			
5	CTS	I/P*	8	CTS	I/P*	5	CTS	O/P*
6	DSR	I/P	6	DSR	I/P			
7	GROUND		5	GROUND		7	GROUND	
8	DCD	I/P	1	DCD	I/P	8	DCD	O/P
20	DTR	O/P*	4	DTR	O/P*	20	DTR	I/P*
22	(RI)	I/P	9	(RI)	I/P*			
						23	(DSRS))	I/P**

* flow control

** DSRS is not connected to anything if PROMDRIVER is to be used. It must be connected to pin 8 if TERMINAL mode is required.

If using the programmer with other host computers a lead may be made up quite simply using the following table of pin connection requirements. The use of pins and connector by the Programmer is that of a DCE (Data Communications Equipment) or Modem.

WARNING: When used with PROMDRIVER under MS-DOS all connections are mandatory. The optional requirement referred to in this section refer to Terminal Mode operation only .

PROGRAMMER PIN DESCRIPTIONS FOR USE WITH COMPUTERS OTHER THAN PC COMPATIBLES.

Pir	Direction	Name	Function	Description
2	INPUT	TXD	Data to Programmer	Mandatory
3	OUTPUT	RXD	Data from Programmer	Mandatory
5	OUTPUT	CTS	Data Flow Control by Programmer	Optional output. Programmer turns this signal OFF if it is not prepared to accept a character at that time
6	OUTPUT	DSR	Programmer on	Optional output. In programmer firmware versions up to v.3.00 this output is ON whenever the programmer is powered.
7	-	GND	Signal Ground	Mandatory
8	OUTPUT	DCD	Programmer on	Optional output. This output is always ON when the programmer is powered.
20	INPUT	DTR	Data Flow Control by host	If not used connect to pin 8 (IMPORTANT). If DTR is taken to OFF by the host data transmission on pin 3 will cease after the character currently being transmitted, and will resume when DTR is taken to ON.
23	INPUT	DSRS	Select TERMINAL mode (i.e. direct connection of a terminal to the Programmer)	Connect to pin 8 if TERMINAL MODE required. OFF - HOST COMPUTER Protocol N/C - HOST COMPUTER Protocol ON – TERMINAL Protocol.

RS232 specifies that ON is a voltage greater than +6 and OFF is a voltage more negative than -6.

Appendix 3.Data Flow Control

Four types of data flow control (or lack of it) are used by the Series II Programmer.

RS232 control line.

The Programmer will stop transmitting if the RS232 line DTR is turned OFF, and will continue if it is turned ON. In Terminal emulation it will turn CTS to OFF if it wants the Terminal to stop transmitting.

XON / XOFF.

The Programmer will stop transmitting if it receives an XOFF character (Control/S). It will resume if it receives an XON (Control/Q). In Terminal emulator it will send XOFF if turning CTS off did not stop the flow, and will send XON when ready for further characters.

Character echo.

This is only appropriate in Terminal emulation when transmitting a file to the Programmer for programming or verifying a PROM. The method by which some terminal emulation programs expect to have their file transmission controlled is by character echo. That is, the emulator will send each successive character of the file only after the last one has been echoed by the Programmer. (This is the method used for example by the public domain program MODEM7.) If this method is used the setting of File Echo ON must be used. If the emulator allows file transmit flow control by RS232 line or by XON/XOFF then it may be necessary to turn File Echo OFF (default setting is OFF).

None.

If no flow control is implemented then successful communication depends on the devices in communication being able to process all input and output communication without character loss.

Host computer protocol uses no flow control in serial port modes 0, 1 and 2. In modes 3 to 12 it uses hardware handshaking, using DTR and CTS.

If using a Terminal Emulator, it is essential to provide a form of positive flow control when transmitting files to the Programmer for programming or verifying a PROM using commands PI, PM, PH, PB, VI, VM, VH or VB. Flow control by character echo cannot be used by the PB or VB commands.

In many Emulators flow control is confined to file transfer. The normal terminal emulation may not be able to keep up with the rate of reception from the Programmer at the higher rates. If you cannot modify the Emulator then you will have to run at a slower rate. A good test of the Emulator is whether the Programmer sign-on message is displayed correctly. For example a version of Modem7 that we tested running on a 4MHz Z80 would only display characters received at 4800 bits/sec or less.

Speed

Although the host protocol and PROMDRIVER have been designed to make best use of the host computer serial port, it is possible that the host may not be capable of handling the full 19200 bits/second rate without losing characters. For example an IBM PC compatible using the MSDOS Auxiliary driver is limited to around 2400 bits/sec in this application (although correct installation as explained in the appropriate section normally permits operation at 9600 or 19200 bits/sec).

If you have problems on a particular computer with data loss as evidenced by 'COMMUNICATION ERROR' messages or lock-ups then try running the serial link at lower rate.

Unfortunately, even using interrupt control and hardware handshaking as PROMDRIVER does in modes 3 to 12,

there is nothing that can be done if data from the Programmer arrives at a rate that the interrupt routine cannot keep up with. This usually happens when other interrupt driven software steals time away from PROMDRIVER. This other software may be resident software loaded by left resident after use like PRINT.EXE

If unsure, try booting up with empty AUTOEXEC.BAT and CONFIG.SYS files. If this cures the problem, try putting back the contents of those files a bit at a time.

Appendix 4.Serial Interface Parameters

If using Terminal Mode the serial port of the host computer should be set up to communicate as follows:

Mode:	Asynchronous.
Speed	300, 600, 1200, 2400, 4800, 9600 or 19200 b/s. The higher the rate the faster the data transfers will take place.
Bits	8 data + no parity bit + 1 stop bit. or 7 data + 1 parity bit (any pol.) + 1 stop bit. (second option does not permit binary files in Terminal Mode)
Receive Parity:	Ignore.

Appendix 5. Security Programming etc.

This is a list of the security programming and other special programming features for each series of devices.

The functions marked with an [S...] number in square brackets are those available in batch mode. This means that, for example, S1 is the instruction to use in batch mode to lock an 8751 device (also see chapter 7).

8751 FAMILY

[Special function s1]: Lock PROM (y/n) ?

87C51 FAMILY

1 - program lock bit 1 [s1]

2 - program lock bit 2 [s2]

3 - program encryption table

Select option:_

Item 3 refers to the 32 byte encryption table that can be programmed in these devices

93CS06/26/46/56/66 EEPROMS

SERIAL EEPROM SPECIAL FUNCTIONS
As throughout PROMDRIVER, addresses relate to 8 bit byt The EEPROM stores 16 bit words. The byte address is obtai by multiplying the required word address by two
1 - protect register read 2 - protect register write 3 - protect register clear 4 - protect register disable (irrevocable)

Select option:_

Z86E11/21 (SGS-T)

1 -	eprom lock	(prota bit)	[s1]
2 -	external memory lock	(protb bit)	[s2]

Select option:_

Z86E21 (ZILOG)

1 - eprom lock	[s1]
2 - ram lock	[s2]
3 - 4k rom lock select	[s3]

Select option:_

PIC 16C5X FAMILY

PIC 16C5x SPECIAL FUNCTIONS		
PROGRAMMABLE SETTINGS		
Oscillator Type	:RC	*p
Watchdog Timer Disabled	:NO	*p
Memory Codelocked	:NO	*p
Programmed ID	:FFFF	*p
n.b. Only options followed by a *p may still be programmed. Oscillator Type of OTP devices must not be reprogrammed.		
FOR INFORMATION ONLY		
Calculated Checksum	:7FFF	

1	re-read settings shown above	[s1]
2	disable watchdog timer	[s2]
3	activate code protect	
4	program checksum or id	
5	program oscillator configuration	[s5]

Select option:_

If item 4 is selected the following sub-menu appears:

1 - program checksum	[s3]
2 - program user defined id	[s4]

Select option:_