USER'S MANUAL

Speedmaster LV
Micromaster LV

Contents

1.2 Conventions

Throughout this manual, certain conventions will be followed in typefaces in order to make instructions clear.

e.g. In the line
Type `EPROM` `<ENTER>`

`EPROM`: this font is used throughout the manual to indicate something that should be typed into the computer

`<ENTER>` : Any items encased in <> indicate the name of a key to press, i.e. in this case the ENTER or RETURN key.

Menu selections are indicated by words separated by forward slashes, e.g. `FILE/LOAD` indicates select the menu option `FILE`, followed by the submenu option `LOAD`.

Messages displayed by the programmer software are indicated in italics, for example;

`DEVICE PROGRAMMED AND VERIFIED`

1.3 Contents of the package

When you open the package you should find :

- Device programmer
  (Speedmaster LV or Micromaster LV)
- A floppy disk containing the software for the programmer.
- User manual
- A power supply (UK, Euro, USA or Japanese)
- A D25 male - male parallel cable
- Free software may be included in the package at the discretion of ICE Technology. These packages may assist you in product development. Please see the READ.ME files on the disk(s) for information on how to use these packages and whom to contact for further information.

The programmer can be operated from batteries. These are not included as standard. Use 8 alkaline AA cells (standard or rechargeable).

1.4 Low Voltage Features

The Speedmaster LV and Micromaster LV support both 3.3V and 5V devices. They are capable of programming and verifying devices down to 3.3 volts. When using low voltage devices the programming algorithm in the system will automatically be set to the correct voltages as specified by the manufacturer.

Even if the manufacturer's algorithm specifies verification at 5 volts you may wish to verify a device at 3.3V to ensure that it will work in your low voltage system, With the capabilities of the LV programmers it is possible to do this. See section 6.6 for full details.

2. Installing the Software

2.1 Making a backup

Before running the software it is advisable to make a back-up copy as soon as you receive the disks. To make a back-up copy use the DOS command `DISKCOPY`. To find out how to use this command please consult your DOS manual.

2.2 Running the Install Programme

An install programme is provided on the floppy disk. This software copies all of the necessary files onto your hard disk.

We suggest that you use the following directories to hold the programmer software

`C:\MMLV` for Micromaster LV,

or

`C:\SMLV` for Speedmaster LV

Where C: is the name of your hard disk

So, if you are installing Micromaster LV software onto the C: drive, to run the installation programme place the floppy disk in the drive and log onto this drive by typing **A:** or **B:** as necessary. Now type:

```
INSTALL   C:\MMLV   <CR>
```

After installation you should add this directory to your current PATH statement in your AUTOEXEC.BAT file. For example change

```
PATH=  C:\DOS;C:\BAT;C:\UTILS
```

to

```
PATH=  C:\DOS;C:\BAT;C:\UTILS;C:\MMLV
```

Then reset the PC.

Alternatively batch files can be used that change to the working directory and invoke the software by specifying the full path name of the software.

e.g.

```
EPROM.BAT  (IN C:\BAT)
cd  C:\MYWORK
C:\MMLV\EPROM.EXE
cd:  C:\
```


3. Connecting the Programmer

3.1 Computer Requirements

The programmer will operate with any IBM compatible PC with a standard IBM CENTRONICS interface (parallel printer port). This means that it is very easy to move it between PC's. Users who prefer to have an internal card or who do not wish to use their existing parallel port can fit an additional parallel card (these are available from ICE Technology). Alternatively, a manual switch box may be used.

The software is compatible with MSDOS V3.0 or later, and with Microsoft Windows version 3.1 or later.

3.2 Connection to the PC

STEP 1 - Connect one end of the parallel cable to the programmer. The other end of the cable should connect to the PC. The cable will only fit the centronics port. Tighten the holding screws thus ensuring that the cable will not cause programming problems.

NB The programmer does not have a power on switch. It will power up when a signal is received from the computer, and automatically power down when not in use.

STEP 2 - Connect the power supply provided to the programmer and plug into any mains socket. Alternatively, fit 8 alkaline AA batteries.

USE ONLY THE POWER SUPPLY ADAPTER PROVIDED.
Replacements and adapters for different countries (UK/European/US/Japan) can be supplied by ICE Technology Ltd or their distributors.

STEP 3 - Switch on the PC.

STEP 4 - Run the SELFTEST programme.

3.3 Running the SELFTEST Programme

Your programmer has been supplied with selftesting software which can check that the programmer has been installed correctly and can be used for fault diagnosis.
Before using the programmer for the first time, whenever moving it to a new PC or if a fault is suspected, run the **SELFTEST** programme. This has two purposes :

- It confirms which parallel port is being used by the programmer.
- It checks most of the programmer hardware.

WARNING :

ALWAYS ENSURE THAT THERE IS NO CHIP IN THE PROGRAMMER BEFORE RUNNING SELFTEST AS LEAVING A DEVICE IN THE SOCKET WOULD DAMAGE THE CHIP AND GIVE FALSE RESULTS.

To run the programme connect the programmer to the PC and either mains or battery power, remove any chips and type

```
SELFTEST <CR>
```

at the DOS prompt.
The SELFTEST will report the programmer's ROM version number and run a comms test. It will then run through a series of hardware checks and will report any faults found. The programme will report on which port the programmer is connected to. This information can then be used to set up the parallel port in the EPROM.EXE or PAL.EXE software.

If an error is reported it is possible to obtain a printout by using

```
SELFTEST >PRN
```

or

```
SELFTEST >ERROR.LST
```

followed by

```
PRINT ERROR.LST
```

See Appendix C for Technical Support information

3.4 Operating from Batteries

The Speedmaster LV and Micromaster LV can be operated from mains or batteries.To operate from batteries use 8 alkaline AA type cells.

The battery cover is held in place with screws. When replacing the cover, take care to use only the screws provided.

WARNING :

**USE ONLY THE SCREWS PROVIDED TO FASTEN THE BATTERY COVER.
DO NOT OVERTIGHTEN THE SCREWS.**

NB Certain Bipolar devices require high currents in very short bursts and therefore some batteries may not always be able to handle these devices.

3.4.1 Recharging the Batteries

The programmer can be used as a battery charger. In order to start recharging the batteries the programmer needs to be connected to a PC and connected to the mains using the mains adapter provided.

To charge the batteries, the command is: BCHARGER

usage;

```
BCHARGER [port] CHARGEMODE [REPORTMODE]
```

where:

`port` = parallel port to use (default = 1)

`CHARGEMODE` = `TRICKLE/SLOW/FAST` (default=slow)

`REPORTMODE` = `START/REPORT` (default=both)

`START` starts the charging and does immediate return

`REPORT` reports current stage of charging

When the batteries are charging the 'BUSY' LED on the programmer will flash red and yellow. Once charging is complete this LED will switch off.

Once recharging has commenced the programmer can be disconnected from the PC. Charging will stop once the batteries have been fully charged. If the PC is switched off battery charging will be interrupted for a few seconds. Recharging will resume automatically.

DO NOT ATTEMPT TO CHARGE NON-RECHARGEABLE BATTERIES

3.5 Programming devices other than dual in line and parts with over 40 pins

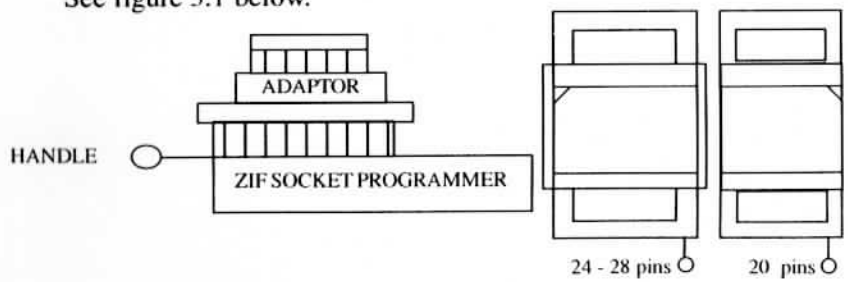
Devices in other packaging than dual in line, and devices over 40 pins require socket adapters. A range of adapters is available from ICE Technology for most common parts. If you are unsure as to which adapter to use please contact ICE Technology or your distributor for advice.

Adapters from other sources which are simple pin conversions can usually be used successfully. However, for devices over 40 pins pin mappings are not usually standard, and so it is necessary to use an adapter specifically designed for the programmer in use. Where devices require non-standard adapters to be supported these are available from ICE Technology.

3.5.1 Adapter usage

Insert the adaptor into the ZIF socket of the programmer with the bottom end of the adaptor in the bottom pins of the socket. The text on the adaptor boards should be the right way up.Lock the Zif socket handle in place.For PLCC adaptors, the notched corner of the adaptor socket should be in the top left hand corner. Devices are placed in the socket in the 'live bug' position, ie with legs downwards for all sockets 28 pins and above. 20 pin sockets are dead bug loading.Push to lock the device in place, then push again to release.

See figure 3.1 below.



4. Getting Started

We have provided some quickstart tutorials to help you start using the programmer as quickly as possible. If you have not used a programmer before, these may help you to begin. However, these should not be taken as a comprehensive guide to the programmer's capabilities. The following tutorials are provided:

TUTORIAL 1. Programming an EPROM or Microcontroller from a master device.

TUTORIAL 2. Programming an EPROM or Microcontroller from a file on disk.

TUTORIAL 3. Programming a PAL or GAL from a file on disk.

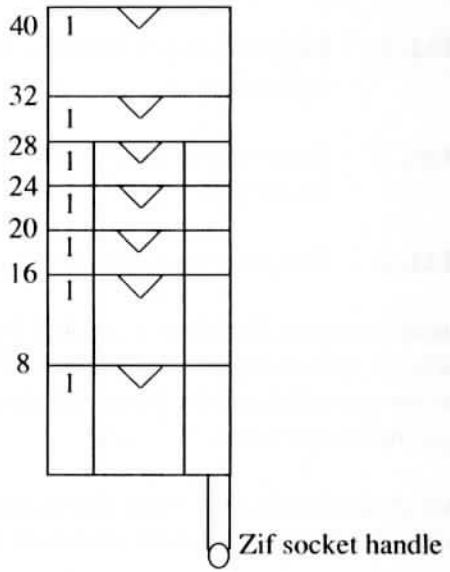
These tutorials are provided to get you using the programmer quickly for simple tasks. However, we recommend that you read the reference sections of the manual if you have any difficulty and before attempting any more complex operations.

The quickstart tutorials use the menu driven software. For information on how to use the batch (DOS command) software, please see section 9.

4.1 Before You Start

If you are not sure how to select menu items please see section 5.3 for more information .

Figure 4.1 below illustrates the orientation of chips in the programmer: The chip is located at the bottom of the ZIF socket with the bottom pins next to the ZIF socket handle.



If you are using a socket adapter, please see the instructions with the adapter for how to place the chip in the socket.

4.2 Quickstart tutorials

TUTORIAL 1:

Programming an EPROM or Microcontroller from a master device.

1. Type `EPROM <RETURN>`
2. Select the part you are using as a master, using `PART/MANUFACTURER`. Select the manufacturer and press `<CR>`. You will then be given a list of different types of devices. Select the type and press `<RETURN>`. You will then be prompted for the part name. Choose the correct part name from the menu you are given and press `<RETURN>` to accept. The details of the device you have chosen will now be displayed at the bottom of the screen.
3. Ensure that the buffer size is large enough to hold the data. You can change the buffer size using `BUFFER/SIZE`. See section 6.2.1 for more information.
4. Insert the master device in the programmer.
5. Select `ACTION/READ` to read the contents of the chip into the buffer. Press `<F10>` to accept the default addresses.
6. Take the Master out of the programmer.
7. Select the part you are copying to if it is different (see 2.)
8. Place the copy device in the programmer.
9. Select `ACTION/AUTOPROGRAMME`. Press `<F10>` to accept the default addresses.
10. The message: *`Device programmed and verified`* will be displayed.

TUTORIAL 2:

Programming an EPROM or Microcontroller from a file on disk

1. Type EPROM <RETURN>
2. Select the part you are using as a master, using PART/MANUFACTURER. Select the manufacturer and press <RETURN>. You will then be given a list of different types of devices. Select the type and press <RETURN>. You will then be prompted for the part name. Choose the correct part name from the menu you are given and press <RETURN> to accept. The details of the device you have chosen will now be displayed at the bottom of the screen.
3. Ensure that the buffer size is large enough to hold the data. You can change the buffer size using BUFFER/SIZE. See section 6.2.1 for more information.
4. Select FILE/LOAD
5. Specify the path and filename where your data is located.
6. Select the file format used (HEX auto recognition will detect any of the HEX formats available)
7. Press <F10> to accept the default settings and load the data. A checksum will be displayed on the screen.
8. Select the part you are copying to if it is different (see 2.)
9. Place the device to be programmed in the programmer.
10. Select ACTION/AUTOPROGRAMME. Press <F10> to accept the default addresses.
11. The message Device programmed and verified will be displayed.

TUTORIAL 3:

Programming a PAL or GAL

1. Type PAL <RETURN>.
2. Select the part you are going to programme using PART/MANUFACTURER. Select the manufacturer and press <RETURN> Choose the device part name from the list on the screen by highlighting the correct part and pressing <RETURN>
3. Select FILE/LOAD
4. Specify the path and filename where your JEDEC file is located.
5. Press <F10> to accept the defaults. A checksum will be displayed once the file has loaded.
6. Place the device to be programmed in the ZIF socket.
7. Select ACTION/AUTOPROGRAMME. The device will now be erased (where relevant), programmed, verified and any test vectors present in the JEDEC file will be applied.

5. Using the Programmer Software

5.1 Basic Software Configuration

The operating software contains the following programs.

EPROM.EXE : Used to programme EPROM's, EEPROM's, PROMS or Microcontrollers.

PAL.EXE : Used to programme GALs, PALs, PEELs, EPLDs etc.

CHIPTEST.EXE: Used to identify and test TTL and CMOS logic devices.

These are menu driven programs which contain all the features required to programme and test devices.
The actual programming information for individual devices is held in library files, which allows easy updating of the software to accommodate new devices.
Batch software is also included which allows the programmer to be operated from DOS batch files. Details of the commands available are shown in section 9.
For some programmable logic devices, separate programmes are supplied to handle non JEDEC standard files.
On first installing the software and programmer to a computer, remember to alter the parallel port selection within EPROM and PAL by selecting OTHER/PARALLEL PORT NO.

5.2 Screen Layout

Each programme displays information as follows :

Menu headings are displayed across the top
Device programming details are displayed in the bottom window.
Buffer contents are displayed in the main window.

5.3 General Menu Selection

The software has been designed to make the programmer as simple as possible to use. All functions are contained within the menu system. The menus are obtained by pressing the <ESCAPE> key. The user can then move around the menus by pressing the cursor keys. To select an option press <RETURN> .

Alternative **shorthand keystrokes** are available by pressing the first letter of the menu name followed by the first letter of the option. Some options can be selected by pressing **ALT+letter**. These are indicated by a triangle next to a letter on the menu display. For example, Autoprogramme can be selected by pressing **ALT+A**. A full list of shorthand keystrokes is given in Appendix A.

5.4 Dialogue Boxes

The menu system makes extensive use of dialogue boxes. Within these boxes the following prompt types may appear :

Prompt

: requires you to type in some information

> will display a series of options when you press <RETURN> .

Inside the dialogue boxes use the cursor keys to move between options; use <RETURN> to select the contents of the dialogue box. Use <F10> to accept all chosen options within the window. If you wish to abort from the window, press <ESCAPE> .

5.5 Changing the Settings

Before starting, you may wish to change any of the following. They are found in the OTHER menu. This will have to be done for both EPROM and PAL :

OTHER/PARALLEL PORT NO
Selects parallel port to be used. (Delivered default is Port No 1) This may or may not be the same as LPT1 depending on the BIOS. See instructions on **SELFTEST** (section 3.3) to find out which port number to choose. Once the parallel port has been chosen it will be remembered when next using the programmer.

OTHER/READ COLOUR INFO
To choose whether the display is MONO or COLOUR

OTHER/SNOW PRECAUTIONS
Usually set to OFF but set it to ON if you have problems with snow on your monitor. (affects early PC's with CGA)

OTHER/HI-RES MODE
Select ON for 40/50 line mode. Effective only with EGA and VGA monitors.

5.6 Using a mouse

All three application programmes can be mouse driven. However your mouse driver software must previously have been installed. This is normally done automatically in your CONFIG.SYS file (by loading a device driver such as MOUSE.SYS) or in your AUTOEXEC.BAT file (by invoking a mouse driver programme such as MOUSE.COM). If you are unsure how to do this please refer to your system manual or the manual supplied with your mouse. **EPROM.EXE, PAL.EXE** or **CHIPTEST.EXE** automatically enable the mouse if one is installed (see later if you wish to disable it).

5.6.1 Left Hand Mouse Button (LMB)

The LMB is used to select items in a list, such as a menu selection or a device name from the device list. Simply move the mouse to the required item and press the LMB. If the item is a menu heading, such as **FILE, ACTION** etc. then this will open up that particular menu. If the item is a sub-menu or part of a list this will highlight the chosen item. To select the highlighted item DOUBLE CLICK on the LMB. i.e. QUICKLY press and release the LMB twice. DOUBLE CLICKING performs the same action as PRESSING <CR>

5.6.2 Right Hand Mouse Button (RMB)

Pressing the RMB is exactly the same as pressing `<ESC>`.

5.6.3 Middle Mouse Button (MMB)

Pressing the MMB is exactly the same as pressing `<F10>`. When this key is a possible input the window being displayed will have F10 in the bottom border. Pointing at the text F10 and pressing the LMB will also produce the same result (For two button mouse users.)

5.6.4 Disabling the Mouse

To disable the use of the mouse invoke each of the programmes with the / NM option. e.g.

```
EPROM /NM
```

5.7 Edit and View modes and Scroll Bars

In order to observe data in the buffer without risking modification a **VIEW** mode is available in the **BUFFER** menu. This will only allow scrolling, paging etc. Mouse users can enter this mode directly by pointing at the buffer window and pressing the LMB. In order to make moving around the buffer much easier SCROLL BARS have been added to the window borders.

5.8 Running the Software under Windows

All three programmes can be run in a DOS window under WINDOWS 3.1. Example .PIF files and .ICO files are included. To create an ICON for the Speedmaster `EPROM.EXE` do the following:

Select "`NEW PROGRAMME ITEM`" in the Programme Managers `FILES/NEW` menu.
Change Description to `EPROM`
Change Command Line to `C:\SMLV\EPROM.PIF`
Change Working Directory to `C:\SMLV`
Select `CHANGE ICON` and change the file in the CHANGE ICON prompt to `C:\SMLV\EPROM.ICO` or `C:\SMLV\EPROM1.ICO`
Select `OK`.

Repeat for **PAL** and **CHIPTEST**. Use Optional Parameters if required. The mouse functions should work in a DOS window in 386 enhanced mode in WINDOWS 3.1 but if you have trouble see section 10 of the WINDOWS readme file.

5.9 Using the Programmer Software over a Network

It is possible to run the programmer software from a network hard drive with the programmer itself connected to the local computer.

Some networks do not pass the full execution path to the .EXE file. The EPROM and PAL programmes therefore have difficulty in locating the library and device files. To circumvent this problem use the following environment variables to set the full network path of the executable;

For Speedmaster LV

```
SET SMLV = [PATHNAME]
```

For Micromaster LV

```
SET MMLV = [PATHNAME]
```

where `PATHNAME` is the full network path where EPROM.EXE and PAL.EXE are located

For example, if the programmer software is held in

Z:\MMLV

Then the statement

```
set MMLV = z:\MMLV
```


Should be added to the AUTOEXEC.BAT file of the computer where the programmer is connected.

5.10 Using EMS Software

The programmer is supplied with EMS software which will enable expanded memory to be accessed if it is available. This requires an expanded memory manager compatible with EMS 3.2 standard. It allows up to 8 Mb of memory and will run on any PC with an expanded memory board or a 286 or 386 with an expanded memory manager. The software will not give the EMS option if a driver such as EMM386.EXE is not installed correctly.

If the EMS function is working correctly, the line

SPEED(MICRO)MASTER LV E/EPROM/uC SOFTWARE VX.XX - EMS

will appear on the screen.
If this line does not appear and an EMS driver is present, check the CONFIG.SYS file. For DOS 5 and greater this should contain the line

```
device = emm386.exe
```

Also ensure that the NOEMS option has not been included on this line.

5.11 Saving software settings

When exiting the programmer software, settings such as parallel port, the part chosen, etc. are automatically saved to avoid having to repeatedly set up the chosen options.

6. EPROM Programming

This section covers the Memory type parts which are programmed using **EPROM.EXE**, including Microcontrollers.

To use this programme, at the DOS prompt, type

```
EPROM <CR>
```

6.1 Part Selection

In order to tell the programmer software which programming algorithm to use, it is necessary first to select the part which is being used.

6.1.1 To select a part

To do this, enter **EPROM.EXE** and select

```
PART/MANUFACTURER
```

A list of supported manufacturers will be displayed on the screen. Use the up and down cursor keys to find the manufacturer of the part you are using. You can also press the initial letter of the manufacturer you need, e.g. pressing `M` will take the cursor down to `MICROCHIP` .When the cursor is highlighting the required manufacturer, press `<CR>` .

You will then be given a choice of device types to select. Highlight the type required or press the first letter, and press `<CR>` .

The software then displays a list of all the supported parts of the

manufacturer and type selected. Highlight the required part name and press <CR> again.

The details of the selected part will then be displayed in the bottom part of the screen.

6.1.2 Changing the part selected

Once a device has been selected, to select another part of the same manufacturer and type, press

PART/PARTNAME.

To select another part type within the same manufacturer, select

PART/TYPE.

6.1.3 Which part?

Generally it is necessary to select the part name which is as close as possible to that stamped on the chip you are using. For example, a PIC16C54 microcontroller in windowed versions might be marked 16C54, whereas an OTP version would be marked 16C54-XT. The software uses the correct nomenclature according to the manufacturer's datasheet. Speed and package indicators are only required when they affect programming algorithms.

6.1.4 Automatic part selection

Within **EPROM.EXE** it is possible automatically to select a part using ACTION/QUERY EPROM. This will also identify unknown EPROM's, so long as they contain an electronic signature of a known device. See section 6.4.3 for further details.

6.1.5 Low Voltage Parts

Low voltage parts are selected exactly as other parts.

6.2 Loading Files

Files can be loaded from disk into the data buffer and saved to disk after editing using the LOAD and SAVE options in the FILE menu. See section 6.2.8 for more details on saving files.

6.2.1 Setting the Buffer Size

It is necessary to ensure that the buffer has been set to a size large enough to accept the contents of the file to be loaded. Do this using

BUFFER/SIZE
Enter the required size in Kbytes. The maximum buffer size available is determined by the available memory of the PC. The programmer software uses approximately 150K plus the chosen buffer size. The EMS software will use expanded memory for the buffer before using conventional DOS memory. If the buffer cannot be increased sufficiently see section 5.10 for information on installing EMS.

6.2.2 Loading A File

To load a file, select

FILE/LOAD

When loading, you will be asked for a filename, file type, file window low address, file window high address, file increment and buffer address. The filename, complete with path must be typed in. Alternatively, wildcard file loading can be used (see 6.2.6).

6.2.3 File Format

The file type can be selected by pressing return to see the options available. Choose from:

```
RAW/BINARY
ASCII-SPACE-HEX
ASCII - OCTAL
HEX - AUTO RECOGNITION
HEX - MOTOROLA S1, S2 OR S3 RECORDS
HEX - INTEL MCS86
HEX - TECTRONIX
HEX - EXTENDED TECTRONIX
HEX - TI TAG
(MICROMASTER 1000 / LV ONLY)
```

Choosing "HEX - AUTO RECOGNITION" is generally the easiest way of loading a HEX formatted file. However some linkers put headers at the start of the HEX file and can confuse the loader when using this option. Hence the other options.

ASCII-SPACE-HEX embodies the formats of ASCII space HEX, ASCII ' HEX etc. The software assumes that the file consists of HEX digits separated by non HEX digits. Checksum information is not looked for and no separators are needed.

e.g.. If a file consisted of A BCD EF this would appear as

Address	0	1	2	3
Data	0A	BC	0D	EF

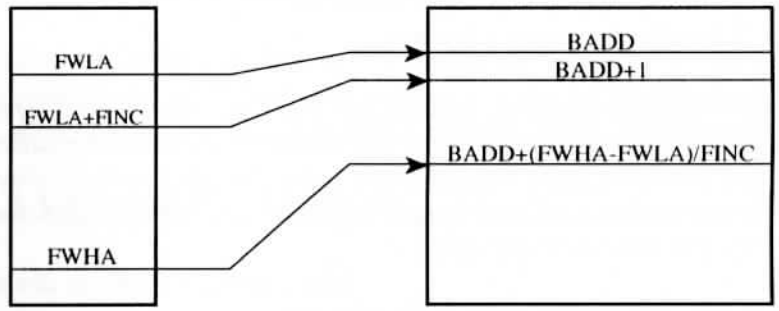
NB When using the PICALC assembler to produce files for Microchip PICs, use the option -F INHX8M to save the file as an 8 bit IntelHex file, then load as HEX-Auto Recognition. WARNING Any line within your code which sets the format to anything else will NOT be overridden by entering the INHX command at the command line.

6.2.4 Handling large files and devices

In order to facilitate the programming of EPROM's and the loading of files whose size is greater than the maximum buffer size the FILE/LOAD command (ALT-L) has the following parameters:

```
FILE WINDOW LOW ADDRESS (FWLA)
FILE WINDOW HIGH ADDRESS (FWHA)
FILE INCREMENT (FINC)
BUFFER START ADDRESS (BADD)
```

A byte is ONLY loaded into the buffer if its FILE ADDRESS is inside the window limits defined by the above (inclusive) range. Data at FWLA is loaded into the buffer at BUFFER START ADDRESS. The FILE INCREMENT works within this window. See Figure 6.1 below.



If you have file loading problems please check the parameters above. Normally FWLA=0 and FWHA=FFFFFFFF (i.e. the whole file is to be loaded). However these parameters are stored each time the programme is quit. If in a previous session FWLA was changed to 10000 (HEX) then files with information in the range 0-FFFF will appear empty.

6.2.5 File Checksum

When a file is loaded into the buffer a checksum is calculated on the contents of the file and displayed on the screen.

This is displayed as three figures:
a) The sum of unsigned bytes with the carry being added to the MSB of the checksum.
b) The complement of a)
c) The arithmetic negative of a)

6.2.6 Wild card Loading

Wild cards may be used in the file name to perform drive/directory searches. Wild cards consist of * and ?.

* matches to any number of characters
? matches to any one character

A list of matching file names is displayed alphabetically (DIRs first) and the user may select using the cursor keys or the first letter of the name. The first 100 lines (max. 74 chars per line) of any file can be displayed after LOAD has been selected by pressing ALT+X to view in HEX mode or ALT+I to view in ASCII mode.

6.2.7 Viewing the contents of a file

It is possible to view the contents of a file without having to load it into the buffer. This enables you to check the contents before loading. In order to view a file, select

FILE/VIEW

The first 100 lines (max. 74 characters per line) can be viewed by selecting VIEW FILE. Use ALT+X to view in HEX mode or ALT+I to view in

ASCII mode. <ENTER> records the last filename chosen. Filenames or wild cards can be entered to select the file to be viewed.

6.2.8 Saving a file to disk

Files edited in the buffer can be saved to disk as Binary or Motorola S2 records using;

FILE/SAVE

Frequently used files can be saved in binary mode to save disk space and reduce loading time.

6.3. Using the Editing Facilities

As well as loading from disk or reading a pre-programmed device, data to be programmed can be entered directly into the buffer from the keyboard. In addition, files can be amended using the BUFFER menu options.

6.3.1 Entering EDIT mode

To edit the buffer contents, select

BUFFER/EDIT

This allows you to type in HEX or OCTAL code or ASCII characters. Once EDIT is selected, the cursor will move to the HEX display in the buffer window. Use the TAB key to switch between HEX and ASCII. In HEX mode, only keys 0-9 and A-F are allowed. In OCTAL mode only keys 0 to 7 are allowed. The cursor keys can be used to move around the buffer.

6.3.2 Viewing without making changes

If you wish to view the buffer contents beyond the first screen without danger of making unintentional changes to the data, use

BUFFER/VIEW

This allows you to view the contents of the buffer as HEX or OCTAL code or ASCII characters. Once VIEW is selected, the cursor will move to the HEX display in the buffer window. Use the TAB key to switch between HEX and ASCII.

6.3.3 Quickly Moving Around the Buffer.

The <HOME> and <END> keys allow you to move more quickly. Press once to move to the beginning (end) of the line, twice to move to the beginning (end) of the page or three times to move to the beginning (end) of the file.

When you have finished editing or viewing the buffer, press <ESCAPE> to return to the main menu.

6.3.4 Quick buffer editing commands

BUFFER/FILL
Allows you to fill a defined area of the buffer with information in byte, word or long word format. Enter the buffer start and end addresses and the information to be inserted. Use less than 3 characters for a byte, 3 or 4 for a word, more than 4 for a long word. Press F10 to accept.

BUFFER/COPY
Copies a defined block of buffer memory to a new location.

BUFFER/SEARCH
Searches for data in HEX, OCTAL or ASCII format. Use the Up/Down

arrow in the dialogue box to move between HEX, OCTAL and ASCII data entry.

BUFFER/GOTO
Goes to a chosen address in the buffer.

BUFFER/SWAP BYTES
Swaps odd and even bytes within a defined range. Useful for 16 bit wide EPROM's.

6.3.5 Buffer checksum

The command

BUFFER/CHECKSUM
Performs a checksum on the contents of a user defined range in the buffer. This is displayed as three figures:

a) The sum of unsigned bytes in the buffer with the carry being added to the MSB of the checksum.

b) The complement of a)

c) The arithmetic negative of a)

NB The buffer could be larger than the size of the file which was loaded in. In order to obtain a correct checksum, specify the start and end address of the file. Checksums for the MICROCHIP PICS are displayed as defined by the data sheet.

6.3.6 Buffer Display Modes

The contents of the buffer can be displayed in 8 bit HEX, 8 bit OCTAL, 12 bit (16 bit storage, LSB = low address) or 16 bit mode (LSB = low address). To change the display, select

OTHER/DISPLAY MODE

and select the display mode required.

6.3.7 Printing the Contents of the Buffer

A printout of all or part of the buffer contents can be obtained by selecting

BUFFER/PRINT

Specify the start and end addresses of the block to be printed, and the name of the output device (e.g. LPT1)

6.4 Reading and Checking Devices

This section explains how to read the contents of a programmed chip, verify the contents against known data, check the device signature and identify a device from it's signature, and perform bit tests and blank checks on devices.

6.4.1 Reading a Device

To read the contents of a device, select the correct part name from the PART menu as described in section 6.1 and place the device to be read in the programmer socket. If you do not know the part name, you can use ACTION/QUERY EPROM to identify the device in many cases.

Please note that some devices CANNOT be read, either because the chip type itself is designed not to be read for security purposes, or because the on-chip security features have been enabled when programming.

Before reading EPROM's, ensure that the buffer size is set large enough to contain the data (see section 6.2.1 on setting the buffer size).

N.B. A 16 bit device with a maximum device address of 128K will require 256K bytes of buffer space.

To read the contents or part of the contents of a device, select

ACTION/READ <RETURN>
The programmer software then quickly accesses the library file for the programming algorithm required.
The screen which then appears will prompt you for information on where to get the data from, and where to place it in the buffer.

DEVICE START ADDRESS: Enter the first address from which you wish to start reading data from the device. Default = 0 for EPROM's and EEPROM's.

DEVICE END ADDRESS: Enter the last address from which you wish to read data in the device. defaults to the highest address available for the chip type selected.

BUFFER START BYTE ADDRESS: Enter the buffer address where you wish to begin reading the data into. Default = 0 for EPROM's and EEPROM's

BUFFER INCREMENT: Allows you to split data across a number of devices by loading in every other byte or every 4th or 8th byte. Default = every byte.

Press <F10> to accept the options displayed and the device will then be read. The contents of the device will then be visible in the buffer, and a checksum will be displayed.

Once the contents of a device have been read, they can be edited using the BUFFER/EDIT option, and then saved to disk, or programmed into another device. See section 6.3 for more detail on editing data.

6.4.2 Verifying the contents of a chip against a file.

To verify a chip's contents against known data, first select the part type you are using.
Ensure that the buffer size is large enough to handle the file.
Load in the file against which you need to verify the data.
Place the device to be verified in the programmer and select

`ACTION/VERIFY.`

If the data matches, the message `VERIFY OK` will be displayed. If there is a mismatch, the message will indicate at what address it has failed to verify the data.

6.4.3 Verifying the device type and identifying unknown devices.

Many more up to date parts contain ID codes which enable you to electronically identify the part. It is possible both to verify a known device type against this ID code, and to identify unknown parts. The ID code is also used by your programmer to check that the correct device type has been selected before proceeding with any actions that may damage a chip. However, not all devices contain this ID code, and the checks for electronic signature may themselves damage a chip without a signature, or one of a different number of pins from that expected. This means that these checks cannot be foolproof and need to be used with some caution.

To verify the signature of a device, choose

`ACTION/VERIFY SIGNATURE`

If the database contains a manufacturer's code and part code they can be verified against that of the device.

To identify an unknown device, select

`ACTION/QUERY EPROM`

This reads the electronic signature of the device in the socket. This value is then checked in the database to find out which device is in the programmer. If the device can be identified, it is selected as if this had been done using the `PART` menu option, and it's details are displayed at the bottom of the screen.

6.4.4 Checking if a device is blank

In order to programme a device, it must either be fully blank, or be capable of containing the information to be programmed into it in addition to the information it already contains. These checks are automatically carried out when you use `ACTION/AUTOPROGRAMME`, but it is possible to do them manually.

In order to test whether a chip can hold additional data, load the data to be programmed into the buffer (see section 6.2), and use

`ACTION/BIT TEST`

This checks to see if a non-blank device can be programmed with the current data in the buffer.
e.g. F0 may be programmed to 00 but not to 0F for EPROM's in which 1 is an unprogrammed state.

In order to test whether a device is empty, select

`ACTION/BLANK CHECK`

This option checks if device is blank (all 1's or all 0's depending on the device). NB it is possible for a device to appear blank even though it is not fully erased. If problems are encountered in programming a device, try erasing it for a longer period and reprogramming.

6.5 Programming and Erasing Devices

6.5.1 Programming devices

Two different options are provided for programming devices.

ACTION/AUTOPROGRAMME performs a bit test, chip erase (for electrically erasable devices) and programme and verify, whereas

ACTION/PROGRAMME
simply programmes and verifies the device.

In normal circumstances when you do not wish to retain any data which may already be in the device, first select the part required, ensure that the data is correct in the buffer, and then choose

ACTION/AUTOPROGRAMME

You will be prompted for information on programming the device. This information will differ depending on the type of device and the on-chip functions supported.

For basic EPROM's, the following options are given;

BUFFER START BYTE ADDRESS: Enter the address within the data buffer from where the data is to be taken. Defaults to 0 for EPROM's and EEPROM's.

BUFFER END BYTE ADDRESS: Enter the last address from which data is to be taken to programme into the device. Defaults to the number of bytes being programmed plus the buffer start address - i.e., it is dependant on the buffer increment and the size of the device selected.

BUFFER INCREMENT: Allows you to programme every byte, every second byte, every 4th byte or every 8th byte. Default is every byte.

DEVICE START LOCATION: Enter the address in the device at which you wish the data to start. Defaults to 0 for EPROM's and EEPROM's.

To accept the address selections or defaults, press <F10>

The programmer will first check the device selection and position, then perform a bit test, and then proceed with programming.
While a device is being programmed, a message will be displayed in the centre of the screen showing the address which has been reached.
Following programming, an automatic verify will be conducted and if all is well the message DEVICE PROGRAMMED AND VERIFIED will be displayed.

6.5.2 Overprogramming devices

If you have a device which already contains some data, and you wish to add data to it, use

ACTION/PROGRAMME

The same options are provided as for AUTOPROGRAMME.
PROGRAMME programmes device with the contents of the buffer. Programme should only be used when programming part of the device while retaining data in other address locations. **In normal circumstances, use** AUTOPROGRAMME.

6.5.3 Single key programming

If you need to programme more than one device in exactly the same way, simply press <F10> to repeat the process. You are given the option to change any of the address settings before proceeding again by pressing <F10>

6.5.4 Erasing devices

For Flash EPROM's, use

ACTION/CHIP ERASE

This electrically erases the data held on the chip.

To erase EEPROM'S, fill the buffer with FF's and programme.

6.6. Adding User Defined Parts

If you need to programme a device that is not on the list, please contact ICE Technology Ltd to find out if it is available on the latest software. For EPROM/EEPROM devices that have not yet been implemented, there is the facility to add user-defined parts.

The user database can also be used to keep frequently used devices from different manufacturers in one place, rather than having to select these parts via the menu system every time. To do this, select the device required, then select PART/ADD TO USER DATABASE. The device name could be changed to a project name if required using PART/MODIFY PARAMETERS before adding to the user database.

N.B. THIS FEATURE IS FOR THE CONVENIENCE OF ADVANCED USERS ONLY. ICE TECHNOLOGY LTD CANNOT BE HELD RESPONSIBLE FOR INCORRECT PROGRAMMING USING THE USER DATABASE.

The information needed to add your own parts will be found in the relevant data sheet. Please contact the device manufacturer for up to date data.

Please Note

- User defined programming information may not work for Microcontrollers since not all of these have standard programming algorithms.

- In order to protect the manufacturer recommended algorithms, it is not possible for the user to change the manufacturers' database.

The procedure for adding user defined parts is as follows:

1) - Select a device which is similar to that to be added from the parts menu. This will minimise the number of changes which it is necessary to make, and will ensure that the relevant options are displayed. Have ready the device data sheet.

2) - Select PART/MODIFY PARAMETERS. This will cause the user database to be called up, thus protecting the manufacturer database from unintentional changes.

Where a list of options is displayed (following a '>') this list is for the latest part type selected e.g. EPROM's, EEPROM's etc.

Always modify parameters in a top to bottom order.

PART NAME:	can be up to 16 characters
PART CODE:	Electronic signature on chip.
MANUFACTURER'S CODE:	Electronic signature on chip.
VCCP:	Vcc during programming. Type in voltage.
VPP:	Programming voltage. Type in voltage.
VCC(DURING VERIFY)	Press <CR> and select from menu.
ORGANISATION:	Press <CR> and select from menu.
LAYOUT:	Press <CR> and select from list.
PULSE TYPE:	Press <CR> and select from list.

PULSE WIDTH(µS): Type in. Typical values are 100 for the latest EPROM's, 1000 for older 2764 to 27010s. 50000 for 2716, 2732 and some microcontrollers.

MAX NUM OF TRIES: Type in. Typical values are 1, 25.

OVER PROGRAMME: Type in over-programme pulse width.

OVER PROG PULSE TYPE: Press <CR> & select variable or fixed. Generally N/A or variable.

SECONDARY MEMORY (MICROCONTROLLERS ONLY):
This is used to define the organisation of secondary EEPROM on board microcontroller devices.

3) - Once device parameters have been modified for the user database, select PART/ADD TO USER DATABASE to add them to the database on disk.

6.7 Microcontroller Programming

Programmable Microcontrollers have specific features which make their programming different to that of EPROM's. The Micromaster LV is specifically designed to handle a wide range of Microcontrollers, and can support most of the optional features which are available on them.

6.7.1 Address Locations

The programmer software handles address locations as they appear on the device. For example, default buffer and device start addresses are at the addresses where the data is held in the device.

6.7.2 Security Features

For Microcontrollers the security features are handled automatically through

PROGRAMME and AUTOPROGRAMME or manually within the SECURITY menu as follows:

SECURITY/STATUS
Gives the status of the device security features. The information displayed is representative of that on the device data sheet.

- P represents a security bit which is programmed
- U represents an unprogrammed security bit

NB Security status is performed automatically before a Blank Check or Auto-Programme. However some devices cannot indicate security status when the security is programmed. In these cases the device may look blank. Thus the device may fail programming attempts until it is erased. If any problems arise with devices that have on chip security please erase the device first and retry.

SECURITY/VERIFY ENCRYPTION
If encryption is in effect/relevant this will verify the encrypted buffer data with the contents of the device.

SECURITY/PROGRAMME
Gives a table of options for programming lock bit(s) and an encryption table if relevant. It is automatically chosen by ACTION/PROGRAMME and ACTION/AUTOPROGRAMME if relevant to the device.

6.7.3 Device- Specific Features

Where device specific features are supported, the options available will be displayed in the relevant window. For example, when programming a windowed PIC device options are displayed for the oscillator type setting, watchdog timer, serial number etc.. Please refer to the manufacturer's datasheet for more information on these options.

7. Programmable Logic

7.1 Part Selection

In order to tell the programmer software which programming algorithm to use, it is necessary first to select the part which is being used.

7.1.1 To select a part

To do this, enter `PAL.EXE` and select

`PART/MANUFACTURER`

A list of supported manufacturers will be displayed on the screen. Use the up and down cursor keys to find the manufacturer of the part you are using. You can also press the initial letter of the manufacturer you need, e.g. pressing `N` will take the cursor down to `NATIONAL SEMICONDUCTOR`. When the cursor is highlighting the required manufacturer, press `<ENTER>`.

The software then displays a list of all the supported parts of the manufacturer and type selected. Highlight the required part name and press `<ENTER>` again.

The details of the selected part will then be displayed in the bottom part of the screen.

NB Some devices which are supported on the programmer will not appear on the menu. This is due to the fact that they do not use JEDEC standard files and therefore are programmed using a separate piece of software. At present this affects Altera and Xilinx EPLDs. See section 7.8 and 7.9 for details of how to use this software.

7.1.2 Changing the part selected

Once a device has been selected, to select another part of the same manufacturer and type, press

```
PART/PARTNAME.
```

When a new part is selected which has a different fuse map from the current part, the fuse map displayed on the screen will alter.

7.1.3 Which part?

Generally it is necessary to select the part name which is as close as possible to that stamped on the chip you are using. However, manufacturers are specific in how they require programmer manufacturers to display part names, and sometimes a number of parts with different suffixes will come under the same part name in the software. The programmer will check the device signature and use the correct algorithm for the part you are programming. For example, AMD devices with a suffix /4 and /5 are both supported under the same part name.
When using an adapter to programme a non-DIL package, select the part as normal. Any pin differences are taken account of by the adapter. Use ICE Technology adapters wherever possible, especially for devices where the non-DIL version is not a straightforward pin for pin conversion of the DIL version.

7.1.4 Low Voltage Parts

Low voltage parts are selected exactly as other parts.

7.2 File Loading

7.2.1 File Formats

Standard JEDEC files can be loaded into the buffer. These can be produced by packages such as PALASM™, OPAL™, APEEL™ etc. Contact device manufacturers or distributors for details of programmable logic design packages. Some packages are given free with ICE Technology programmers from time to time.
In addition to JEDEC files, Altera POF format files and HEX files for Xilinx devices are accepted by the separate programmes provided for these devices.

7.2.2 Loading a file

To load a file into the buffer, first ensure that the correct part has been chosen, then select

```
FILE/LOAD
```

You will then be prompted for the filename to load in.
The software will expect the file to be in the correct format for the part selected. If there is a mismatch a warning will be displayed.
Once the file is loaded, the screen will show the total number of 1's in the file, followed by the checksum.
The JEDEC file loaded will include any test vectors that are present. These will be applied automatically when the part is programmed.

7.2.3 Wild card Loading

Wild cards may be used in the file name to perform drive/directory searches. Wild cards consist of * and ?.

* * matches to any number of characters
* ? matches to any one character

A list of matching file names is displayed alphabetically (DIRs first) and the user may select using the cursor keys or the first letter of the name. The first 100 lines (max. 74 chars per line) of any file can be displayed after LOAD has been selected by pressing ALT+X to view in HEX mode or ALT+I to view in ASCII mode.

7.2.4 Viewing the contents of a file

It is possible to view the contents of a file without having to load it into the buffer. This enables you to check the contents before loading. In order to view a file, select

FILE/VIEW

The first 100 lines (max. 74 characters per line) can be viewed by selecting VIEW FILE. Use ALT+X to view in HEX mode or ALT+I to view in ASCII mode. <ENTER> records the last filename chosen. Filenames or wild cards can be entered to select the file to be viewed.

7.2.5 Saving a file to disk

Files edited in the buffer can be saved to disk using;

FILE/SAVE

All current test vectors will be saved along with the fusemap.
If a file has been converted from a PAL to a GAL using the -conv part, the next time it is loaded it will be in the correct format for the GAL part.

7.3 Reading and Checking Devices

This section explains how to read the contents of a programmed chip, verify the contents against known data, check the device signature and identify a device from it's signature, and perform bit tests and blank checks on devices.

7.3.1 Reading a Device

To read the contents of a device, select the correct part name from the PART menu as described in section 7.1 and place the device to be read in the programmer socket.

Please note that some devices CANNOT be read, either because the chip type itself is designed not to be read for security purposes, or because the on-chip security features have been enabled when programming.

To read the contents or part of the contents of a device, select

ACTION/READ <RETURN>

The programmer software then quickly accesses the library file for the programming algorithm required.

The contents of the device will then be visible in the buffer, and a checksum will be displayed.

Once the contents of a device have been read, they can be edited using the BUFFER/EDIT option, and then saved to disk, or programmed into another device. See section 7.5 for more detail on editing data.

7.3.2 Verifying the contents of a chip against a file.

To verify a chip's contents against known data, first select the part type you are using.
Load in the file against which you need to verify the data.

Place the device to be verified in the programmer and select

ACTION/VERIFY

If the data matches, the message VERIFY OK will be displayed. If there is a mismatch, the message will indicate at what address it has failed to verify the data.

7.3.3 Checking if a device is blank

In order to test whether a device is empty, select

ACTION/BLANK CHECK

This option checks if device is blank (all 1's or all 0's depending on the device). NB it is possible for a device to appear blank even though it is not fully erased. If problems are encountered in programming a device, try erasing it for a longer period and reprogramming.

7.4 Programming and Erasing Devices

7.4.1 Programming devices

Two different options are provided for programming devices.

ACTION/AUTOPROGRAMME

performs a chip erase (for electrically erasable devices), bit test, programme and verify, whereas

ACTION/PROGRAMME
simply programmes and verifies the device.
In normal circumstances when you do not wish to retain any data which

may already be in the device, first select the part required, ensure that the data is correct in the buffer, and then choose

ACTION/AUTOPROGRAMME

You will be prompted for information on programming the device. This information will differ depending on the type of device and the on-chip functions supported.

The programmer will first check the device selection and position, then perform a bit test, and then proceed with programming.
While a device is being programmed, a message will be displayed in the centre of the screen showing the address which has been reached.
Following programming, an automatic verify will be conducted and if all is well the message

DEVICE PROGRAMMED AND VERIFIED

will be displayed.

7.4.2 Overprogramming devices

If you have a device which already contains some data, and you wish to add data to it, use

ACTION/PROGRAMME

The same options are provided as for AUTOPROGRAMME.
PROGRAMME programmes device with contents of buffer. Programme should only be used when programming part of the device while retaining data in other address locations. **In normal circumstances, use** AUTOPROGRAMME.

7.4.3 Single key programming

If you need to programme more than one device in exactly the same way, simply press <F10> to repeat the process. You are given the option to change any of the address settings before proceeding again by pressing <F10>

7.4.4 Erasing devices

For GALs and other electrically erasable devices, use

ACTION/CHIP ERASE

This electrically erases the data held on the chip.

7.5.5 Device Security Features

The SECURITY menu contains features which allow the security status of programmable logic devices to be checked and programmed.

To check whether or not a device is secure, select

SECURITY / STATUS

To programme the security on a device, select

SECURITY / PROGRAMME

To set the security on by default during programming, select

SECURITY / AUTOSECURE

7.5 Buffer Editing

Data to be programmed into programmable logic devices is usually created using a development package such as OPAL™, ABEL™ PALASM™, etc. This takes your logic equations and translates them into a JEDEC file which contains the AND/OR array which will be programmed into the device.
In general it will not be necessary to edit this AND/OR array, or fusemap. However, the programmer software gives the option to be able to do this, for example when entering simple designs without a logic compiler, or for educational purposes.

7.5.1 Editing the AND/OR Array

When a device is selected, the buffer is automatically adjusted to the correct size and layout for the chosen part.

The AND array and OR array (if applicable) are highlighted separately on the display. Product lines are represented horizontally and OR lines vertically.

Relevant keys:
1 - enters a 1 in the fuse map
0 - enters a 0 in the fuse map

In AND array:
ALT 1 : changes the whole horizontal product line to 1
ALT 0 : changes the whole horizontal product line to 0

In OR array:
ALT 1: changes the whole vertical OR line to 1
ALT 0: changes the whole vertical OR line to 0

BUFFER/BLANK FUSE MAP
Sets all fuses to 0 and deletes the test vectors.

BUFFER/SET FUSE MAP
Sets fuses to 1 and deletes the test vectors.

7.5.2 Moving around the buffer

BUFFER/GOTO
Moves the cursor to the selected fuse number.

7.5.3 User's Electronic Signature (UES)

Selecting
BUFFER/EDIT UES
Allows the user to enter the Users Electronic Signature (where relevant). This can be entered as either ASCII or HEX code and the number of bytes depends on the device in use. The UES can be used for stock control, labelling, revision numbers etc.

7.5.4 Viewing the Fusemap

To view the contents of the fusemap without making any changes, select
BUFFER/VIEW

When viewing is complete press <ESCAPE> to return

7.5.5 Buffer checksum

The command

BUFFER/CHECKSUM
Performs a checksum on the contents of a user defined range in the buffer. This is displayed as three figures:

a) The sum of unsigned bytes in the buffer with the carry being added to the MSB of the checksum.

b) The complement of a)

c) The arithmetic negative of a)

7.6 Converting PAL Files for GAL Devices

It is possible to use the programmer to convert old files which have been set up for non-eraseable PAL parts to work in GAL devices. To do this, select a -conv part from the menu, load in the file, and select

BUFFER/CONVERT FUSE MAP

This will convert the fuse map from the -CONV part to the generic part. This is performed automatically if -CONV part is to be programmed. e.g. If an existing design in a PAL16L8 is to be converted into a National Semiconductor GAL16V8 then do the following:

1) ALT+M (manufacturer)
2) Select NATIONAL SEMICONDUCTOR
3) Select GAL16V8-CONV from parts list
4) Select XPAL16L8 from conversion list
5) Load in the file using ALT+L

Then select CONVERT FUSE MAP in the BUFFER menu to perform the conversion. Alternatively choose ALT+A to convert and programme in one operation.

Once a file has been converted it can be saved and used again as a file for the part to which it has been converted, without the need for any further action.

7.7 Test Vector Editing and Operation

Test vectors can be edited or created using the `BUFFER/EDIT TEST VECTOR` command. This will open a window with pin numbers and test vector numbers.
Relevant keys are:

`0`	Forces a logic zero onto pin
`1`	Forces a logic one onto pin
`L`	Expect a logic zero on pin
`H`	Expect a logic one on pin
`C`	(Clock). Apply all other signals with C=0 and then apply C=1 followed by C=0
`X`	don't care
`Z`	Expect high impedance
`N`	Power pin (Vcc or GND). [Sometimes used as don't care but the test vector loader will issue a warning and convert all non power pins to X.]
`ALT+T`	apply the test vectors.

Standard JEDEC Test Vectors can be loaded into the test vector buffer and applied to the chip through the `VECTOR TEST (ALT+T)` command. They are applied automatically when using Autoprogramme. Errors are highlighted.
Test vectors are applied to all 40 pins at the same time. Skew will be determined by the chip's capacitance. This small skew may cause problems on fast pals <7ns if asynchronous circuitry is implemented within the PAL.

7.8 Using .POF files for Altera MAX devices

Altera MAX devices do not use JEDEC format files and are therefore not implemented in PAL.EXE. Separate software enables the programmer to

handle the .POF files which are used on these devices. If you do not have this software, please request it from ICE Technology or download it from the bulletin board.

The software contains drivers for each of the devices supported. To programme a device, type in the part name and the file to be programmed;

eg. EPM5064 file.pof -pN

where N is parallel port number (default is lpt1)

POF file editing is not implemented as the POF format data refers to ELECTRICAL ADDRESSES and DATA. Without recourse to the chip architecture data the information in the POF file is meaningless to the end user.

7.9 Programming Xilinx EPLDs

Xilinx EPLDs use HEX files rather than JEDEC files, and are not implemented in PAL.EXE. Separate software enables the programmer to handle the HEX files that are created by the Xilinx compiler. Each EPLD has a different .EXE file which operates in the following way. The example shown is for the XI7336.

All Xilinx software uses the following syntax.

```
XI7336 [FILENAME] [options]
```

FILENAME is the INTEL HEX file to be used. Only INTEL HEX files generated by the XEPLD development system can be used.

The following options are available :

-READ This can be used to read the Xilinx device in the socket and save the file to disk as an INTEL HEX. This is the default option and can be used in the following ways :

```
XI7336  readfile.hex
```

or `XI7336 readfile.hex -READ`

- BLANK Useful for checking to see if a device is blank. No filename is required. The syntax is therefore :

```
XI7336 -BLANK
```

- VERIFY A verify will be automatically executed after programming but this option may be used to verify the device against an INTEL HEX file. The syntax is :

```
XI7336 vfyfile.hex -VERIFY
```

- PROG This is used to programme and verify a device from the INTEL HEX file specified. If the security has been set in the file then this will automatically be programmed. If not then a separate command with -SEC may be executed as shown below.

```
XI7336 prgfile.hex -PROG
```

- DISPSIG This allows the user signature to be displayed. This option does not depend on whether the security has been programmed and can be displayed in either case.

```
XI7336 -DISPSIG
```


- DISPSEC Allows you to read and display the security status of the device in the socket.

```
XI7336 -DISPSEC
```

- SEC This option allows you to programme the security bits on the Xilinx device. Once this has been done only the signature can be read from the device.

```
XI7336  -SEC
```

-Pn If the programmer is not on Parallel Port 1 then the port number can be set using this option. The default is 1. For example to read from port 2 you shold type :

```
XI7336 testfile.hex -READ -P2
```

8. Using CHIPTEST.EXE

CHIPTEST.EXE is a utility which allows a whole range of ICs to be tested. These include 74 series and 4000 series logic devices, RAMS etc. To run the programme type :

```
CHIPTEST [P] [/NM] <CR>
```

where

`P` is the parallel port number (default = 1)

and

`/NM` disables the mouse

Functional tests can now be made on the more common 7400 and 4000 series devices as well as on static and dynamic rams. A search facility has also been included.

8.1 Testing a Device

To test a device :

1. Select the logic family or device to be tested
2. For logic devices select the device to be tested
3. Place the device in the ZIF socket and select `TEST DEVICE`.
4. The test vectors for the device will now be applied. If an error occurs then the failed vector will be displayed along with was actually found when the device was tested. If no errors are found then a message will tell you that the test vectors were OK.

8.2 Identifying a Device

If you are unsure of a device then insert the device into ZIF socket. Select one of the logic families from the main menu (It doesn't matter which family is selected, both the 74 and 4000 vectors will be applied). Select `FIND A MATCH`. The search will last for about 5 seconds. If all the vectors for a particular part in the library math that of the device being tested then the device number is shown at the bottom of the screen. In some cases there may be more than one match. The software displays all of the devices whose tested vectors match that of the device under test.

8.3 Adding Devices to the User Library

Users can create test vectors to test less common parts. This is done in the same way as defined for `PAL.EXE` (see section 7.7). Ground and Vcc pins can be defined with the G and V descriptors within one test vector. Ground Pins are currently limited (in software) to PIN 20 of the ZIF socket. This will be extended at a later date.

To add a new device to chiptest, select

`EDIT TEST VECTORS`

and enter the test vectors required.

Then select

`ADD TO USER LIBRARY`

8.4 Chiptest Restrictions

The chip test software works by applying test vectors to the device to be tested. If the chip passes the tests the device is assumed good, otherwise it is assumed bad. The search mode works by applying test vectors for all devices in the data base and recording all devices that are good.

Herein lies the problem. The chip is only tested at a frequency determined by the rate at which test vectors can be applied to the device. With the SPEEDMASTER /MICROMASTER hardware this is typically 500 - 3000 test vectors per second (depending on PC). Although this is much faster than most portable chip testers, it is generally much slower than the target hardware. The chip may pass the test vectors but may fail in the target system. Similarly, output loading may be completely different between system and tester.

N.B. If you suspect a chip is causing problems in your target system replace it. The only test we guarantee is one with a negative result.

Testing of rams and LSI devices is slightly different. This is done by the microprocessor inside the tester interfacing to the chip directly. RAM tests are effectively performed on a 1MHz bus. These results should therefore be more reliable. However access times are not measured, hence please note the above warning.

8.5 CHIPTEST Example

The CHIPTEST software allows simple logic gates to be tested quickly and easily.

As an example suppose a batch of 74LS245 devices are suspect. To test the devices run the chiptest software by typing :

```
CHIPTEST N
```

where N is the parallel port where the programmer is connected (default is 1 if this parameter is missed out). The main screen now appears listing the types of devices that can be tested. In our case we select the 74 series family which is the first selection in the menu. This can be selected by moving the cursor to the correct line and pressing return or by double clicking on this selection with the mouse.

A second menu should now appear on the right hand side of the screen. Choose the SELECT DEVICE option. You will now be given the list of devices supported in the 74 series. Move to 74245 by pressing the cursor keys or by using the mouse to move up and down the scroll bar by the side of the window. Once this device has been selected then information you are ready to test the suspect devices. Place the device in the ZIF socket as shown below.

Select TEST DEVICE from the left hand menu. The test vectors will now be applied. If the device passes all of the test vectors then a message will be displayed on the screen informing you that the test vectors were OK. If the device fails a particular vector test then the failed vector is displayed on the screen. Press a key to return to the left menu. You could select FIND A MATCH to apply all of the test vectors to the device to see if the device matches with any in the library. The matches are listed at the bottom of the screen. To quit from the left hand menu press <ESC> or select the MAIN MENU option. This takes you back to the right hand menu where you can test a device from another family or choose QUIT to return to the operating system.

9. Batch Software

Software for batch processing on the Speedmaster LV and Micromaster LV is provided as standard.

9.1 EPROM Batch Software Commands

```
EPBCHK
usage:      EPBCHK [options]
options:
-ds ADDR device start address in HEX (default=MIN ADDRESS)
-de ADDR device end address in HEX (default=MAX ADDRESS)
-eq      error quiet mode, i.e. error returned only via ERRORLEVEL
-e       displays error message on next line of screen (default)
-e r,c   displays error message at screen row=R, col=C requires
         ANSI.SYS
-h       gives this help screen


EPBITTST
usage:      EPBITTST file [options]
where:      file    is filename for bit test
options:
-ds ADDR device start address in HEX (default=MIN ADDRESS)
-de ADDR device end address in HEX (default=MAX ADDRESS)
-fl ADDR file window low address (default=0)
-ft B|H  file type is Binary|HEX (default=HEX)
-fi 1|2|4file increment, 1=every byte/word, 2=every other byte/
         word etc.
-eq      error quiet mode, i.e. error returned only via
         ERRORLEVEL
-e       displays error message on next line of screen (default)
```

-e R,C displays error message at screen row=R, col=C requires ANSI.SYS
-b NUMK buffer size in Kbytes. Min=64, def=device size (to system max.)
-r NUM number of times to repeat. Each repeat is prompted at R,C (the R,C parameters of the -e command must be specified)
-h gives this help screen

EPCONFIG
usage: EPCONFIG -m manfile.typ -n partname [options]
where: manfile is the manufacturer file and
typ = epr|eep|ucs
(see EPROMPAR directory for a complete list)
partname is an exact partname
(use EPROM.EXE ALT-N for a complete list)
options:
-eq error quiet mode, i.e. error returned only via ERRORLEVEL
-e displays error message on next line of screen (default)
-e R,C displays error message at screen row=R, col=C (requires ANSI.SYS)
-p 1|2|3 parallel port (default = 1)
-h gives this help screen

EPCHKSUM
usage: EPCHKSUM [options]
options:
-ds ADDR device start address in HEX (default=MIN ADDRESS)
-de ADDR device end address in HEX (default=MAX ADDRESS)
-eq error quiet mode, i.e. error returned only via ERRORLEVEL
-e displays error message on next line of screen (default)
-e R,C displays error message at screen row=R, col=C (requires ANSI.SYS)
-a displays current device address on next line of screen (default)

-a R,C displays current device address at screen row=R, col=C (requires ANSI.SYS)
-h gives this help screen

EPPROG
usage: EPPROG file [options]
where: file is filename containing data to be: programmed
options:
-ds ADDR device start address in HEX (default=MIN ADDRESS)
-de ADDR device end address in HEX (default=MAX ADDRESS)
-fl ADDR file window low address (default=0)
-ft B|H file type is Binary|HEX (default=HEX)
-fi 1|2|4 file increment, 1=every byte/word, 2=every other byte/word etc.
-eq error quiet mode, i.e. error returned only via ERRORLEVEL
-e displays error message on next line of screen (default)
-e R,C displays error message at screen row=R, col=C (requires ANSI.SYS)
-a displays current device address on next line of screen (default)
-a R,C displays current device address at screen row=R, col=C (requires ANSI.SYS)
-b NUMK buffer size in Kbytes. Min=64, def=device size (to system max.)
-r NUM number of times to repeat. Each repeat is prompted at R,C. The R,C parameters of the -e command must be specified
-i ignore electronic signature error and continue to programme
-h gives this help screen

EPREAD
usage: EPREAD file [options]
where: file is filename for output
Options:
-ds ADDR device start address in HEX (default=MIN ADDRESS)
-de ADDR device end address in HEX (default=MAX ADDRESS)

```
-eq         error quiet mode, i.e. error returned only via ERRORLEVEL
-e          displays error message on next line of screen (default)
-e R,C      displays error message at screen row=R, col=C
            ( requires ANSI.SYS )
-a          displays current device address on next line of screen (default)
-a R,C      displays current device address at screen row=R, col=C
            (requires ANSI.SYS )
 -h         gives this help screen
```

```
EPVERIFY
usage:      EPVERIFY file [options]
where:      file is filename to verify against
options:
-ds ADDR    device start address in HEX (default=MIN ADDRESS)
-de ADDR    device end address in HEX (default=MAX ADDRESS)
-fl ADDR    file window low address (default=0)
-ft B|H     file type is Binary|HEX (default=HEX)
-fi 1|2|4   file increment, 1=every byte/word, 2=every other byte/word
            etc.
-eq         error quiet mode, i.e. error returned only via ERRORLEVEL
-e          displays error message on next line of screen (default)
-e R,C      displays error message at screen row=R, col=C
            (requires ANSI.SYS )
-b NUMK     buffer size in Kbytes. Min=64, def=device size (to system
            max.)
-r NUM      number of times to repeat. Each repeat is prompted at R,C
            (the R,C parameters of the -e command must be specified )
-h          gives this help screen
```

N.B. the environment variable SMLV must point to the SMLVdirectory

9.2 PAL Batch Software Commands

```
PABCHK
usage:      PABCHK [options]
options:
-eq         error quiet mode, i.e. error returned only via ERRORLEVEL
-e          displays error message on next line of screen (default)
-e R,C      displays error message at screen row=R, col=C
            ( requires ANSI.SYS )
-h          gives this help screen
```

```
PACHKSUM
usage:      PACHKSUM [options]
options:
-eq         error quiet mode, i.e. error returned only via ERRORLEVEL
-e          displays error message on next line of screen (default)
-e R,C      displays error message at screen row=R, col=C
            ( requires ANSI.SYS )
-h          gives this help screen
```

```
PACONFIG
usage:      PACONFIG -m manufacturer -n partname [options]
where:      manufacturer is one of the following:-
                Altera      AMD
                AMI         ATMEL
                Cypress     Gould
                iCT         Intel
                Lattice     MMI
                National    Philips
                SGS         TI
partname is the partname (no conversions or xpal parts ) defined in
            PAL.EXE.( Only the name up to the first '/' is relevant.)
options:
-eq         error quiet mode, i.e. error returned only via ERRORLEVEL
-e          displays error message on next line of screen (default)
```

```
-e R,C      displays error message at screen row=R, col=C
            ( requires ANSI.SYS)
-p 1|2|3    Parallel port number ( default = 1)
-h          gives this help screen

PAERASE
usage:      PAERASE   [options]
options:
-eq         error quiet mode, i.e. error returned  only via ERRORLEVEL
-e          displays error message on next line of screen (default)
-e R,C      displays error message at screen row=R, col=C
            (requires ANSI.SYS )
-h          gives this help screen

PAPROG
usage :     PAPROG file [options]
where:      file  is filename containing data to be programmed
options:
-eq         error quiet mode, i.e. error returned only via ERRORLEVEL
-e          displays error message on next line of screen (default)
-e R,C      displays error message at screen row=R, col=C
            (requires ANSI.SYS )
 -h         gives this help screen


PAREAD
usage:      PAREAD file [options]
where:      file  is filename for output
options:
-eq         error quiet mode, i.e. error returned only via ERRORLEVEL
-e          displays error message on next line of screen (default)
-e R,C      displays error message at screen row=R, col=C
            (requires ANSI.SYS )
-h          gives this help screen
```



```
PASECURE
usage:      PASECURE [options]
options:
-eq         error quiet mode, i.e. error returned only via ERRORLEVEL
-e          displays error message on next line of screen (default)
-e R,C      displays error message at screen row=R, col=C
            (requires ANSI.SYS )
-h          gives this help screen

PAVECTST
usage:      PAVECTST file [options]
where:      file  is the jedec filename to use for the vector test
Options:
-eq         error quiet mode, i.e. error returned only via ERRORLEVEL
-e          displays error message on next line of screen (default)
-e R,C      displays error message at screen row=R, col=C
            (requires ANSI.SYS )
-h          gives this help screen


PAVERIFY
usage:      PAVERIFY file [options]
where:      file  is the jedec filename to verify against
options:
-eq         error quiet mode, i.e. error returned only via ERRORLEVEL

-e          displays error message on next line of screen (default)
-e R,C      displays error message at screen row=R, col=C
            ( requires ANSI.SYS )
-h          gives this help screen
```

N.B. the environment variable SMLV must point to the SMLV directory

9.3 Batch Software Utilities

TEXT

Usage: `text message row col [fg bg]`

`message` is the text to display (possibly enclosed in double quotes)

`row, col` is the position on the screen

`fg` is the foreground colour of the text

`bg` is the background colour of the text

BOX

Usage: `box col [ trow lcol brow rcol ]`

`col` is the colour of the box

`trow,lcol` is the position of the top left corner of the box on the screen (0,0 to max. of 22,77)

`brow,rcol` is the position of the bottom right corner of the box on the screen (trow+2,lcol+2 to max. of 24,79)

Default box is the whole screen

ANSIESC

Usage : `ansiesc message [ message... ]`

where: `message` is ordinary text with the special escape code sequences :

`$e` = esc code (0x1b)

`$$` = '$'

`$f` = formfeed char (for use with redirection to printer)

`$s` = space

`$n` = carriage return and newline

`$d` =current directory

`$xHH` = char HH (HEX)

For example:

```
ansiesc $e[0m "Current Directory is "
        $e[1;5;7;33;45m$d$e[0m
```

See DOS Manual for a list of ANSI ESC CODES (including key redefinitions)

BEEP

Usage : `beep ifreq1 duration1 [ ifreq2 duration2 ... ]`

Makes a beep. Frequency is inversely proportional to `ifreq`

e.g.

```
beep 400 8 500 8 400 8 500 8 400 8 500 8
```

WINDOW

Usage: `Window trow lcol brow rcol type[fgcol][bgcol][title]`

`trow,lcol` is the position of the top left corner of window on screen (0,0 to max. of 22,77)

`brow,rcol` is the position of the bottom right corner of window on screen (trow+2, lcol+2 to max. of 24,79)

`type` is the type of border (0 = double, 1 = single, 2 = blank, 3= horizontal double)

`fgcol` = foreground colour

`gbcol` = background colour

`title` is the optional title for the window (If `title` is specified then `fgcol` and `bkcol` must also be specified.)

WAITKEY

Usage : `Waitkey keylist`

`keylist` is a list of chars to wait for. All others are ignored

Possible keys are : A-Z, 0-9, F1 - F10, UP, DOWN, LEFT, RIGHT, SPACE, CR, PAGEDN, PAGEUP, HOME, END, ESC or DEFAULT

Returns when a key in `keylist` is pressed and sets the ERRORLEVEL equal to the position of the key within the `keylist` (leftmost argument = 1, etc.)

If DEFAULT is specified any key not in the keylist returns ERRORLEVEL = 0

E.g.

```
waitkey CR space F10 a b c DEFAULT
IF ERRORLEVEL 6   goto Keyc
IF ERRORLEVEL 5   goto Keyb
IF ERRORLEVEL 4   goto Keya
IF ERRORLEVEL 3   goto F10KEY
IF ERRORLEVEL 2   goto SPACE
IF ERRORLEVEL 1   goto CR
:default
```

N.B. Since, in DOS, ERRORLEVEL n statement implies ERRORLEVEL >=n then a reverse order is important here.

WARN

Usage :

```
Warn message [fgcol][bgcol][bfgcol][bbgcol][type]
```

fgcol = foreground colour

bgcol = background colour

bfgcol, bbgcol refers to border surround

type is the type of border (0=double, 1 = single, 2=blank, 3=horizontal double) default is white on black, type 1 border

9.4 Batch Software Errorlevels

MESSAGE	ERROR LEVEL RETURNED
No Error	0
Bad Command Line Parameters	1
No Environment Variable found	2
Manufacturers File Not Found or Invalid	3
Bad Partname Specified	4
Bad Device Driver Specified	5
Programmer is not a SMLV	6
Programmer is not a MMLV	7
Bad Library File	8
Bad comms with Micromaster/Speedmaster	9
Checksum Error	10
Memory Error	11
Bad file Read	12
Bad Device Specified	13
File was not found or could not be opened	14
Bad definitions in Command Line Parameters	15
Bad address range has been specified	16
Device is not Blank	17
Verify Error	18
Failure during BITTEST	19
Device or Parameter is not Programmable	20
Bad Electronic Signature	21
Device is not Secure	22
Bad JEDEC File	23
Device not Erased	24
No Test Vector Support	25
Bad Vector	26
See Error Log File	128

Any other values are internal errors and you should contact ICE Technology before continuing.

9.5 Batch Software Examples

EXAMPLE 1

This batch file shows the simplest way of using the batch programmes. No error checking is done and all information is echoed to the screen.

```
REM     setting up the environment variable to
        point to the Micromaster software
REM     This only needs to be done once and may
        be done in the AUTOEXEC.BAT file.
SET     MMLV=C:\MMLV
REM     Configuring the programmer to programme
        Signetics 27C64A's
EPCONFIG - MSIGNETIC.EPR -N27C64A
REM     Checking to see if the data can be
        programmed into the EPROM
EPBITTST TESTFILE.HEX
REM     Programming and verifying the device
EPPROG  TESTFILE.HEX
```

EXAMPLE 2

This batch file has basic error checking throughout. It reports a message at the end informing the user of the success or failure of the attempt to programme. For a complete list of errorlevels see section 9.4.

REMEMBER

In DOS when errorlevels are being tested a test of errorlevel = 1 will jump to 'fail' if the errorlevel is greater than or equal to 1.

```
@ECHO OFF
REM     Setting up the environment variable to
        point to the Micromaster software
REM     This only needs to be done once and may
        be done in the AUTOEXEC.BAT file.
SET     MMLV=C:\MMLV
```



```
REM     Configuring the programmer to programme
        Signetics 27C64A's
EPCONFIG -MSIGNETIC.EPR -N27C64A
If errorlevel = 1 goto fail
REM     Checking to see if the data can be
        programmed into the EPROM
EPBITTST TESTFILE.HEX
If errorlevel = 1 goto fail
REM     Programming and verifying the device
EPPROG TESTFILE.HEX
If errorlevel = 1 goto fail
echo Device programmed OK
goto end
:fail
echo Programming error
:end
```

Appendix A: Shorthand Keystrokes

For quick operation of ICE Technology Programmers the following shorthand keystrokes are available :-

	EPROM	PAL
Load	Alt+L	Alt+L
Save	Alt+S	Alt+S
View	Alt+I or Alt+X	Alt+I or Alt+X
Edit Buffer	Alt+E	Alt+E
Checksum	Alt+C	Alt+C
Fill Buffer	Alt+F	
Goto Fuse		Alt+G
Goto Address	Alt+G	
Edit UES		Alt+U
Buffer Size	Alt+Z	
Read	Alt+R	Alt+R
Verify	Alt+V	Alt+V
Blank Check	Alt+B	Alt+B
OverProgramme	Alt+P	Alt+P
Autoprogramme	Alt+A	Alt+A
Query EPROM	Alt+Q	
Vector Test		Alt+T
Manufacturer	Alt+M	Alt+M
Type		Alt+T
Partname	Alt+N	Alt+N
DOS Command	Alt+D	Alt+D
Hi-Res Mode	Alt+H	Alt+H
Display Mode	Alt+Y	

Appendix B : Menu Options

This section gives a brief overview of each of the menu options available in EPROM.EXE and PAL.EXE and tells you in which section of the manual you will find more detailed information.

EPROM.EXE

Menu Option	Section
FILE MENU	
Load Load a binary or HEX formatted from disk and store it in the buffer.	6.2
Save Save an area of the buffer to a binary or formatted file on disk.	6.2.8
View File View the contents of a file in ASCII or hexadecimal format.	6.2.7
Quit Exit the software and save the settings.	5.11

BUFFER MENU

Edit 6.3.1
Edit the contents of the data buffer.

View 6.3.2
View the contents of the data buffer without making any changes.

Fill 6.3.4
Fill an area of the buffer with a specified value.

Copy 6.3.4
Copy an area of the buffer to another area.

Search 6.3.3
Search an area of the buffer for a specified value or string.

Goto 6.3.3
Goto a user specified area of the buffer.

Swap Bytes 6.3.4
Swaps odd and even bytes. Useful for data into more than 1 EPROM.

Buffer Size 6.2.1
Changes to size of the buffer.

Print 6.3.7
Prints out an area of the buffer

Checksum 6.3.5
Calculates a checksum over a given area.

ACTION MENU

Read 6.4.1
Read the device in the ZIF socket and store the data in the buffer.

Verify 6.4.2
Verify the data in the device against the data in the buffer.

Verify Signature 6.4.3
Checks the electronic signature of the device.

Bit Test 6.4.4
Checks to see if a non blank device can be programmed with the data

Blank Check 6.4.4
Checks to see if a device is blank.

Programme 6.5.2
Programmes data in the buffer into the device.(inc. Verify Sig & Verify)

Erase 6.5.4
Erases Electronically erasable devices.

Auto Programme 6.5.1
Same as Programme but with bit test included.

Query EPROM 6.4.3
Find out what device is in the socket.

SECURITY MENU

Status 6.7.2
Gives details of the status of security bits

Verify Encryption 6.7.2
Verifies buffer data with the encrypted data in the device.

Programme 6.7.2
Manually programmes the security bits.

PART MENU

Manufacturer 6.1.1
Selects the manufacturer of the device to be programmed.

Type 6.1
Selects the type of device to be programmed.

Partname 6.1
Selects the partname of the device to be programmed.

Modify Parameters 6.6
Allows users to add their own devices.

Add to User Database 6.6
Stores new devices in the user library.

OTHER MENU

Parallel Port 5.5
Tells the software which parallel port the programmer is connected.

Screen Colour 5.5
Selects either mono or colour displays.

Snow Precautions 5.5
Overcomes snow problems on old CGA displays.

Hi-Res Mode 5.5
Changes number of lines on EGA and VGA displays.

Display Mode 5.5
Switch between 8, 12, 16 bit display.

DOS Command
Enables a DOS command to be run from the software.

PAL.EXE

Menu Option	Section
FILE MENU	
Load Load a binary or HEX formatted file from disk and store it in the buffer.	7.2
Save Save an area of the buffer to a binary or formatted file on disk.	7.2.5
View File View the contents of a file in ASCII or hexadecimal format.	7.2.4
Quit Exit the software and save the settings.	5.11

BUFFER MENU

Edit 7.5.1
Edit the contents of the data buffer.

View 7.5.4
View the contents of the data buffer without making changes.

Goto Fuse 7.5.2
Goto a defined fuse in the fuse map.

Blank Fuse Map 7.5.1
Sets all the fuses in the buffer to '0'.

Set Fuse Map 7.5.1
Sets all the fuses in the buffer to '1'.

Convert Fuse Map 7.6
Used to convert PALs to GALs.

Edit UES 7.5.3
Edits the Users Electronic Signature.

Edit Test vectors 7.7
Edit the test vectors for this file.

Checksum 7.5.5
Calculate checksum on the data in the fuse map..

ACTION MENU

Read 7.3.1
Read the device in the ZIF socket and store the data in the buffer.

Verify 7.3.2
Verify the data in the device against the data in the buffer.

Blank Check 7.3.3
Checks to see if a device is blank.

Programme 7.4.2
Programmes data in the buffer into the device.

Erase 7.4.4
Erases Electronically erasable devices.

Vector Test 7.7
Applies Test Vectors.

Auto Programme 7.4.1
Same as Programme but with bit test included.

SECURITY MENU

Status 7.4.5
Gives details of the status of security bits

Programme 7.4.5
Programmes the security bits

Auto Secure 7.4.5
Automatically secures devices when programming

PART MENU

Manufacturer 7.1
Selects the manufacturer of the device to be programmed.

Partname 7.1
Selects the partname of the device to be programmed.

OTHER MENU

Parallel Port 5.5
Tells the software which parallel port the programmer is connected.

Screen Colour 5.5
Selects either mono or colour displays.

Snow Precautions 5.5
Overcomes snow problems on old CGA displays.

Hi-Res Mode 5.5
Changes number of lines on EGA and VGA displays.

DOS Command
Enables a DOS command to be run from the software.

APPENDIX C: Technical Support

If you require technical support for any reason, please ensure that you have the following information to hand:

1. The serial number of your programmer.

2. The firmware version : To find out which firmware version you are using run SELFTEST. The firmware version number is displayed at the end of the test.

3. The software version and library version for the part you are programming: To find out which version of the library you are using connect the programmer and run EPROM. Select the manufacturer and part name of the device being programmed. The library version number is then displayed on the third line of the screen.

4. The results of the SELFTEST programme: To run the programme, ensure that you are in the directory containing the programmer software (e.g. C:\MMLV) and type :

```
SELFTEST
```

at the DOS prompt. The programme will then report on which port the programmer is connected to. This information can then be used to set up the parallel port in the EPROM.EXE or PAL.EXE software. If an error is reported then obtain a printout by using

`SELFTEST >PRN` or
`SELFTEST >ERROR.LST` followed by `PRINT ERROR.LST`

5. The exact part name of the device you are programming.

6. An exact note of any error messages you are getting.

It will assist our engineers if you are next to the programmer when calling.

TECHNICAL SUPPORT PHONE NUMBER:

+44 (0) 1226 767404

FAX NUMBER:

+44 (0) 1226 370434

BULLETIN BOARD:

+44 (0) 1226 761181

Guarantee Conditions

1. All programmers and adaptors are guaranteed for a period of twelve months from the date of purchase.
2. This guarantee covers both parts and labour.
3. In the event of an item being found to be faulty, ICE Technology Ltd guarantees to repair or replace the item at it's own discretion.
4. Repairs will be carried out on the premises of ICE Technology Ltd or it's agents or distributors at the discretion of ICE Technology Ltd.
5. No item will be accepted by ICE Technology Ltd for repair or refund without an RMA number having been issued by ICE Technology Ltd.
6. All costs of return delivery to ICE Technology Ltd are the responsibility of the purchaser.
7. Return delivery costs of guaranteed items back to the customer will be met by ICE Technology Ltd where these are within the UK. The method of delivery will be wholly at the discretion of ICE Technology Ltd.
8. Return delivery outside the UK will be the responsibility of the purchaser and must be paid for before delivery will be made.
9. While every endeavour will be made to repair or replace any item as quickly as possible, no guarantees can be made on the time taken and ICE Technology Ltd cannot be held responsible for any loss or inconvenience caused.
10. ICE Technology Ltd cannot be held responsible for any loss or damage, consequential or otherwise, incurred while using it's equipment.
11. While ICE Technology Ltd makes every endeavour to ensure that programming algorithms are correct, the said algorithms are not guaranteed by the manufacturers and therefore ICE Technology Ltd cannot make any guarantees regarding the algorithms implemented on it's programmers. Users are advised to ensure that devices work correctly in their systems before programming large quantities.
12. All items except for custom made items or software are covered by a 14 day money back guarantee if not satisfied. Before returning any goods for refund, an RMA number must be obtained. ICE Technology Ltd reserves the right to make a handling charge .

USER'S MANUAL

ICE TECHNOLOGY LTD
PENISTONE COURT, STATION BUILDINGS, PENISTONE, SOUTH YORKSHIRE UK S30 6HG

tel: +44 (0)1226 767404 fax: +44 (0)1226 370434
bbs: +44 (0)1226 761181