Elan Digital Systems

MODEL 6000J/E

USER'S GUIDE

JUNE 1995

ISSUE: D

MODEL 6000 UNIVERSAL PROGRAMMER - JUNIOR AND ELITE

CONTENTS

Section 1 PREFACE

1.1 Introduction

Welcome to the new enhanced version of the Elan Model 6000 Universal Programming System. This product is suitable for programming memory, logic and microcontroller devices. You will discover new powerful operating features, wider device coverage, faster support services and exceptional product quality.

You may install the system to display commands in your own language (1994). The system is designed for easy use and simple expansion to support complex devices of the future. It may be used in conjunction with any suitable programming compiler software.

This User's Manual provides both operational instructions and reference information. The list of supported programmable devices is shown under the device selection menu in the operating software and also in the most recent issue of the Elan Programmable Device Support List.

Full product specifications may be found in Appendix D.

We would ask you first, before powering up, to read carefully the following Section 2 (INSTALLATION) and Section 3 (GETTING STARTED).

1.2 Product Preview

The Model 6000 is configured to meet your particular requirements.

The complete programming system supplied to you consists of:

* Base unit (with internal plug-in pin driver cards)
* Top connector Pod (PD)
* Plug-in Programmable Device interface (PDi)

PC centronics and mains power cables
3 1/2" distribution disk with control and algorithm software
User's Guide
(* these items come already assembled as one unit)

Options are available to further characterise, expand or enhance your system. Please refer to the sub-section for System Upgrades and contact your nearest Elan representative.

There are 2 Base systems: 6000 Junior (6000J) and 6000 Elite (6000E).

The Junior model can be converted into the Elite model.
The Model number describes the configuration of your system:
example - Model 6000E/PDi84-11

6000J - starter system for 24 to 48 pin drivers

6000E - full system for up to 128 or more pin drivers
(Standard system suitable up to 88 pin drivers)

PDi48 - Pod to access up to 48 pin drivers

PDi84 - Pod to access up to 88 pin drivers

-xx - indicates the number of pin driver cards fitted
(8 full analogue/digital pin drivers per card)
thus -11 indicates 88 pin drivers are fitted.

Normally all Model 6000J/PDi48 versions are supplied with PDi48 adapters for up to 48 pin DIL/DIP device packages, and all Model 6000E/PDi84 versions are supplied with PDi84 universal PLCC adapters for 24 to 84 pin PLCC device packages. Other PDis are available for LCC, SO, Flatpack, FPGA and other types of packages. However all PDis may be used with both the Junior and Elite system.

All configurations are expandable to the maximum available.

1.3 Feature Highlights

The Model 6000 has been designed to produce the highest possible programming integrity, without compromise. All pins, to the maximum configuration, have full analogue and digital pin drivers. Fast, high quality miniature relays are used to allow every pin to be switched between ground and full rated current and voltage supplies. The fastest signal switching, rise and fall times, can be implemented directly without customised system adjustments.

Using advanced programming techniques the time to implement newly introduced parts has been reduced dramatically. Software quality programs and testing processes provide "first time success" results. Device coverage is therefore not only improved but may be tailored to meet customers' specific requirements.

The Model 6000 is the only known programming system capable of being expanded in the field from the simplest model to the maximum size configuration able to handle the most complex and largest parts.

The user interface is presented in easy to follow command structures using pull-down menus and windows options. File editing, data manipulation and programming controls are powerful and simple.

System start-up from power on is fast and easy. Programming speeds and very rapid file transfers can be increased with an optional plug-in High Speed Interface half card for the IBM ISA bus.

Section 2 INSTALLATION

2.1 System Requirements

The 6000 programmer requires an IBM PC compatible with 640K base memory, a hard disk with a minimum of 1.5MByte spare space. If you are using large density memories such a 8Mbit (1Mbyte) devices, you will need twice the size of the memory in spare disk space. For example a 8Mbit (1 MByte) device will require an additional 2Mbyte of hard disk space for data buffers.

The PC also requires a floppy disk drive and a parallel (centronics) printer port.

If you are unclear how much memory and disk space is available on your system, you can run the DOS "CHKDSK" command.

The 6000 can be used on an XT compatible programmer and monochrome display. To get the best from the 6000 a 486 or Pentium based computer with a mouse and colour display is recommended.

The 6000 requires MSDOS 3.3 operating system or greater. It can be used under Windows and OS/2 in a DOS window, but MUST be run with exclusive execution during programming.

The 6000 is not powered from the PC, but has it own integral mains supply to maintain sufficient and fully regulated power supply.

2.2 Hardware Installation

Connect the provided mains cable to connector at rear of 6000, and power the unit on.
The red indicator on the top of the 6000 marked "Power" should illuminate.

Connect the centronics cable to the rear of the 6000, taking care to push home the wire retention clips. The connect the other end to the printer port of the PC.

Caution: Ensure that connection is made to a printer port, not a serial interface or other function. Check your PC manual if unsure.

Caution: The 6000 has a cooling fan expelling at the rear of the machine. Do not obstruct this air flow.

2.3 Software Installation

Place the distribution disk into drive A: of the PC. Type <a:> and then <INSTALL>. The 6000 will then install the control package with guidance and options where necessary.

The 6000 install package will modify your AUTOEXEC.BAT file if you assent during the installation progress. If your AUTOEXEC.BAT file has not been modified, you will need to edit it for correct 6000 operation.

Add the line SET EL6HOME = c:\el6000\lib.

If your EL6000 files are not in the el6000 directory, substitute the directory you have chosen. Similarly if your EL6000 files are not in drive c:, substitute your drive letter.

You will also need to add EL6000 to your path if you wish to call the EL6000 program while not in the El6000 directory. For example:
"PATH =c:\DOS;c:\WINDOWS;c:\el6000\"

2.4 System Upgrades

The Model 6000 system may be re-configured or expanded by yourself as required.

Various hardware options are available, such as:

PIN8 -	Pin driver cards, each providing all analogue and digital controls for 8 pins, to increase pin driver capacity.
POD84	Pod assembly to increase number of accessible pins from 48 to 84 (other larger Pods are also available
PDi84	With 84 pin universal PLCC socket for device sizes from 20 to 84 pins (other PDis are available for different device packages as required).
CON11	Connector expansion kit to convert 6000J to 6000E system.
6AC	High Speed Accelerator Card Interface using half size PC plug-in card into IBM compatible ISA Bus (1994).

NOTE:
Pods (e.g. Pdi84 are offered with different quantities of ground relays the number of which is shown on the pod label after the dash (-). Thus a Pdi84 pod with 48 ground relays is shown as "Pdi84-48" (relays).

It is necesssary for the same number of relays to be available as there are pins on the largest device being programmed. Pod upgrades may therefore be required.

Whereas some large logic devices may be programmed using pin-swap adapters, this creates limitations in vector testing and operational integrity. It is recommended that a full complement of pin drivers and supporting hardware be used for as many pins as appear on the maximum device size to be programmed with an appropriate PDi.

Software updates are made available on a regular basis. These may be accessed by any registered customer via the Elan BBS. Please make sure you return the Registration Card directly to Elan's UK office to become eligible for a BBS registration number. Cards returned more than 1 month after delivery of the Model 6000 will not qualify for additional privileges.

A proprietary Device Support Assurance Policy (DSAP) is optionally available to offer personalised guarantees for any customer seeking special device support services.

Please refer to your local Elan representative for full details and pricing on all options.

Section 3 GETTING STARTED

Install the 6000 software and hardware as described in section 2.
Reboot your system.

Type "EL6000" to invoke the control program.

The 6000 will display an information window with some software issue information. Press <ENTER> or click the mouse on <OK>. The 6000 will then go through its various self tests and hardware initialisation. If a message is displayed "6000 not found, retry?", the 6000 programming hardware could not be found. Select <Yes> to try again or <NO> to run in demo mode.

The first time the 6000 software is operated, a device manufacturer screen will appear. Select the manufacturer required using the cursor keys and then the device type required in the following device window.

The 6000 is now ready for operation.

Section 4 USING THE 6000

4.1 Mouse Control

The main control for the 6000 is through the menu bar at the top of the screen. Click on the required menu heading to access the menu headings. Once a menu is opened, a single click on a menu item will operate that command.

On the bottom bar are more general commands. Click on any of these as required.

Information Window

In top left hand of the screen is the Information Window. This contains a history of programmer response messages. If you click on this window it becomes active. To get a full screen viewing of the window, click on the upwards arrow on the top of the window. To return to a normal screen, click on the dual upwards/downwards arrow on the top of the window. You can also page up and down the Information history by clicking on the up and down arrow on the gridded bar on the extreme right of the window.

Device Status Window

This window provides status on the currently selected device and also programmer controls by a series of "radio buttons". Active buttons will be in a highlight. To use a programmer command, simply click on the button of the function required.

The status information displayed is contingent on the type of device selected. A short cut to change a parameter is to click on the item required in the status window. For example, if you wish to change the device, simply click anywhere on the device line.

For memory and microcontroller devices the following information is displayed:

Manufacturer:
Device:

Device Start:
Device End :
Buffer Start:

Wordsize :
Blocks :
Current set :

Message Windows

A message such as a warning will be displayed in a pop-up message window. Often there may be two or three buttons requesting user action. Click on the button required.

Dialog Boxes General

Dialog boxes are used for entering data such as setup parameters.

Parameters that are toggled on or off are displayed next to square brackets [ ]. Clicking in the square brackets will toggle the field on or off. An 'X' displayed in the square brackets denotes the option active.

Parameters that are one of a choice are displayed next to normal brackets ( ). Click on the required option. A dot (.) displayed denotes which option is active.

Parameters that are user definable are displayed along side the current value. Click on the required parameter and type in the new value. Note that in default any key presses will add to those present, so you may have to use <DEL> or backspace to erase existing characters. Alternatively you can press <INS> to enable overtype mode. Overtype mode is indicated by a blocked cursor rather than a line cursor.

File Selection Dialog Boxes

This dialog boxes are used when a file name is entered by the user for example an edit.

The box at the top of the window will display the current file name or mask, usually on entry "*.*". Below this will be displayed all the files in the current directory conforming to that file mask.

To change the file mask, click the mouse on the mask and type in the new name. Click on the OPEN button, and a new list of files will be displayed. To select a file click on the file name to highlight it, then click on the OPEN button.

You will notice that on the bottom of the window the size and date stamp of the file currently selected is displayed.

If you click on the down arrow button to the right of the file mask field a history of previously selected files and masks will be displayed. To activate one of these, double click on the required item. Then click on the OPEN button.

You can also change directories by clicking on the directory entries in the file list. If the list of files is too big to be displayed in the window, click on the left and right arrows on the etched bar on the bottom of the files window.

4.2 KeyBoard Only Control

The main control for the 6000 is by the menu bar on the top of the screen. To access a menu you can press <ALT> and the highlighted letter of the menu title. For example to access the utilities menu press <ALT U>. The up/down cursor keys are then used to move the highlight bar up and down the menu until it is over the required command. Press <RETURN> to execute that command. To move from menu to menu, use the left and right cursor keys.

Information Window

In top left hand of the screen is the Information Window. This contains a history of programmer response messages. With keyboard only operation, only the information currently in the window can be seen, the user cannot scroll back through the history.

Device Status Window

This window provides status on the currently selected device and also programmer controls by a series of "radio buttons". This buttons are not used for keyboard only control.

Message Windows

A message such as a warning will be displayed in a pop-up message window. Often there may be two or three buttons requesting user action. When there is a button for confirmation, for example "OK" after a warning, simply press <RETURN> after viewing the message. Where there is a button for abort or cancel, press <ESC>. Where there are three or more buttons, use the TAB key to move from button to button until the required one is active. An active button is in highlight. The press <RETURN> to activate the command.

Dialog Boxes General

Dialog boxes are used for entering data such as setup parameters.

Parameters that are toggled on or off are displayed next to square brackets [ ]. Use the up/down cursor keys to select the required parameter of a group, and use <SPACE> to toggle the option. An 'X' displayed in the square brackets denotes the option active.

Parameters that are one of a choice are displayed next to normal brackets ( ). Use the up/down cursor keys to select the required option within a group. A dot (.) displayed denotes which option is active.

In more complex setups there may be two or more groups of setable parameters. Use the <TAB> key to move from group to group. Note also the <TAB> key will also move to any radio buttons present also. The active group will be displayed in highlight.

Parameters that are user definable are displayed along side the current value. Use the <TAB> key to select and highlight the required parameter and type in the new value. Note that in default any key presses will add to those present, so you may have to use <DEL> or backspace to erase existing characters. Alternatively you can press <INS> to enable overtype mode. Overtype mode is indicated by a blocked cursor rather than a line cursor.

File Selection Dialog Boxes

This dialog boxes are used when a file name is entered by the user for example an edit.

The box at the top of the window will display the current file name or mask, usually on entry "*.*". Below this will be displayed all the files in the current directory conforming to that file mask.

To change the file mask, type in the new name. Press <RETURN> to select the file. To select a file in the group of files below conforming to the file mask, press <TAB> and then use the cursor keys to highlight a file. Press <RETURN> to select the file.

You will notice that on the bottom of the window the size and date stamp of the file currently selected is displayed.

With the filemask field highlighted, press the down cursor key to reveal a history of previous selected filenames and masks. Use the cursor keys to highlight the required name, and press <RETURN> to select.

You can also change directories by selecting one of the directory entries in the file list. If the list of files is too big to be displayed in the window, use the <PgUp> and <PgDn> keys.

4.3 Accommodating Different IC Packages

The 6000 Universal programmer is designed to support all programmable devices no matter how they are physically packaged. Most devices are supplied in DIP, but increasingly devices are being supplied in PLCC, TSOP and a host of other configurations.

There are two ways to support non-DIP devices on the 6000. The first most conventional way is via an adapter that converts the package socket connections from a DIP format and plugs into a DIP ZIF socket. For devices with 44 pins or less, these adapters are readily available from your local Elan representative. For devices greater than 44 pins, often the adapter will be custom to that device and may not be available. Also for PLD's test vectors cannot be supported as the 6000 cannot get access to all the pins. If you wish to program a number of surface mount devices, in this way you may have to pay for a whole range of adapters.

To overcome the problems with such adapters, Elan has developed its unique PDi approach. A Universal PDi is a single board interface with mounted ZIF socket which plugs into the top of the 6000 Pod. Every pin of the PDi is connected to a universal pin driver. Therefore any device in that package style can be supported on that PDi. The PDi is a robust solution, suitable for production environments. Each PDi has its own identification code that is read by the 6000 on initialisation. The 6000 software can then determine which devices are available to the user with the current hardware configuration.

As an alternative to the Universal PDi (reference PDi Selector Chart) Elan can offer Dedicated PDi which allows high pin count devices to be programmed with fewer pin drivers. Each dedicated PDi is specific to a single device or group of devices. Check with Elan.

If you have a package style not currently supported on you hardware, contact your local Elan representative for information on PDi's and adapters.

4.4 What's a Buffer?

A buffer is a temporary store of all data used in programming operations. The buffer is the master source of all data for programming operations. Any data from files is always loaded into the buffer, where it can be modified if required before use to program a device. Any data saved to file is always read from the buffer.

When using memory or microcontroller type devices, all data transformation is done when transferring a file into the buffer.

Remember its what is in the buffer, not the input file is what is programmed into the device. Use the buffer for What You See Is What You Get (WYSISYG)

Section 5 FILES MENU

5.1 Load File

The Load File command loads data from a user specified file to the buffer. On executing this command there are two types of dialog boxes that appear. If the current device is a PLD, a PLD file load dialog box is used, otherwise the standard memory dialog box is displayed.

Loading a File for a PLD

The PLD file menu will only request a file name. The search for the file name is always made in the current default directory, **unless a path is appended to the file name**. For operation details on the file selection window, see File Selection Dialog Boxes , Section 4.1 The file format is always selected automatically to be JEDEC, unless the device is an Altera MAX type device, when POF format is used. The 6000 buffer is always cleared, when a new PLD file is loaded.

Loading a File For a Memory of Microcontroller Device

File Format

If you have a binary file, select the binary option. For other file formats select <HEX>. The 6000 will intelligently recognise the file format you are using.

Byte Order

For most files, use the "Normal" default setting. This will load the first byte encountered in the file into the first location of the buffer.

The Reverse setting only has relevance for 16bit devices. If using 16bit devices under a 16bit Wordsize setting with the Reverse option set, then the 1st byte of the file is loaded into the second location in the buffer, and the 2nd byte in the file is loaded into the first byte of the buffer.

Example: 16 bit device.

(ADD: = ADDRESS IN BUFFER)

ADD: 0	LO
ADD: 1	HI
ADD: 2	LO
ADD: 3	HI
ADD: 4	LO
ADD: 5	HI
ADD: 6	LO
ADD: 7	HI

With byte order set to Normal.

DEVICE

ADD: 0	LO	HI
ADD: 1	LO	HI
ADD: 2	LO	HI
ADD: 3	LO	HI

Set to Reverse

DEVICE

HI	LO
HI	LO
HI	LO
HI	LO

Example: 32 bit device into two 16 bit devices.

Your file will be organised thus:

ADD: 0	Byte1
ADD: 1	Byte2
ADD: 2	Byte3
ADD: 3	Byte4
ADD: 4	Byte1
ADD: 5	Byte2
ADD: 6	Byte3
ADD: 7	Byte4
ADD: 8	Byte1
ADD: 9	Byte2
ADD: A	Byte3
ADD: B	Byte4
ADD: C	Byte1
ADD: D	Byte2
ADD: E	Byte3
ADD: F	Byte4

With Byte Order Set to Normal.

DEVICE 1

ADD: 0	Byte1	Byte2
ADD: 1	Byte1	Byte2
ADD: 2	Byte1	Byte2
ADD: 3	Byte1	Byte2

DEVICE 2

Byte3	Byte4
Byte3	Byte4
Byte3	Byte4
Byte3	Byte4

With Byte Order Set to Reverse

DEVICE 1

ADD: 0	Byte4	Byte3
ADD: 1	Byte4	Byte3
ADD: 2	Byte4	Byte3
ADD: 3	Byte4	Byte3

DEVICE 2

Byte2	Byte1
Byte2	Byte1
Byte2	Byte1
Byte2	Byte1

Pre Clear Buffer

If this option is set to "Yes", the buffer will be cleared of existing data and set to the blank state of the device currentlyselected. If set to "NO", any data previously in the buffer will be retained as long as it is not overwritten by the file transfer data.

Unless you specifically have a reason to use the "No" option, don't. Your input file may have gaps in it that will be left in an unknown state.

File Start, File End and Buffer Offset

The buffer offset will be added to the address of the input file. For most applications, leave this in the default state of 0.

For example: Your input file is binary, and therefore has its first data byte at file address 0h. You may wish to put the data at address 2000, so set Buffer Offset = 2000H. The 6000 will then on file transfer into the buffer add 2000H to all addresses in the file.

The File Start and File End fields specify what portion of the file you wish to load into the buffer. In most cases use the default setting, set to the currently selected device size.

For Example: You have an input file in Motorola format that has data that starts at 2000H. You wish the data to be placed at address 0H in the buffer. Set the File Start Address Field to 2000.

Filename

This is used to enter the filename containing the data to be loaded into the buffer. The current default directory will be searched for this file, unless prefixed by a different path.

If using the mouse on the arrow to the right of the filename field, a history of previously selected filenames will be displayed.
Double-click on any of these if required.

5.2 Save Buffer

Save Buffer command will write the contents of the buffer to a user-specified file in a user-specified format. Two different dialog boxes are used, one for PLD's, the other for memories and microcontroller products. A progress window will be displayed in the top right hand portion of the screen.

Saving A Buffer for a PLD

The PLD file menu will only request a file name. The search for the file name is always made in the current default directory, unless a path is appended to the file name. For operation details on the file selection window, see File Selection Dialog Boxes , Section 4.1 The file format is always selected automatically to be JEDEC, unless the device is an Altera MAX type device when POF format is used.

The JEDEC file produced will store the device currently selected and a date stamp in the comment field. It will also include a fuse checksum and transmission checksum for security. Any test vectors in the buffer will also be appended to the JEDEC file.

Saving A Buffer For a Memory Or Microcontroller Device

File Format

This specifies what data file format will be used to write to the file.

Buffer Start, Buffer End and File Offset

The file offset will be added to the address of the buffer address.
For most applications, leave this in the default state of 0.

For example: You may wish to put the data in the buffer starting at address 0 in the file at address 2000. Set File Offset = 2000H. The 6000 will then on file transfer add 2000H to all addresses before writing to the file. Note this field will be ignored for Binary file format transfers.

The Buffer Start and Buffer End fields specify what portion of the buffer you wish to load into the file. In most cases use the default setting, set to the currently selected device size.

For example, with File Offset =0, Buffer Start = 2000h, only data from the buffer at address greater than 2000h will be loaded into the file, starting at address 0.

Byte Order

For most files, use the "Normal" default setting. This will load the first byte encountered in the buffer into the first location of the file.

If using the reverse setting with a 16bit Wordsize, then the 1st byte of the buffer is loaded into the second location in the file, and the 2nd byte in the buffer is loaded into the second byte of the file.

Example: 16 Bit WordSize

Buffer

ADD: 0	LO	HI
ADD: 1	LO	HI
ADD: 2	LO	HI
ADD: 3	LO	HI

With byte order set to Normal.

File Data

ADD: 0	LO
ADD: 1	HI
ADD: 2	LO
ADD: 3	HI
ADD: 4	LO
ADD: 5	HI
ADD: 6	LO
ADD: 7	HI

With byte order set to Reverse.

File Data

ADD: 0	HI
ADD: 1	LO
ADD: 2	HI
ADD: 3	LO
ADD: 4	HI
ADD: 5	LO
ADD: 6	HI
ADD: 7	LO

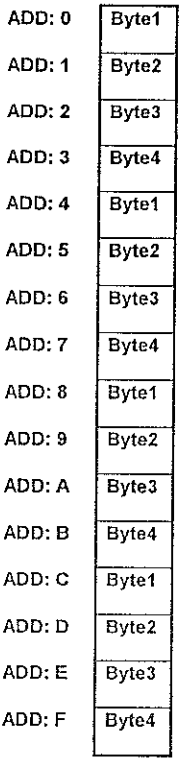
Example: 32 bit Wordsize

Buffer

ADD: 0	Byte1	Byte2	Byte3	Byte4
ADD: 1	Byte1	Byte2	Byte3	Byte4
ADD: 2	Byte1	Byte2	Byte3	Byte4
ADD: 3	Byte1	Byte2	Byte3	Byte4

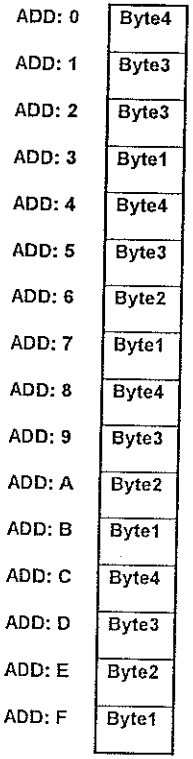
For Byte Order Set To Normal

File



For Byte Order Set To Reverse

File



5.3 Edit File

For the viewing and editing of programming data files prior to loading these into the Model 6000 Buffer, it is recommended to use the view and editing commands outside EL6000 software with proprietary file management software i.e. XTREE GOLD.

5.4 Edit Test Vectors

This function is available when using file management software in the PC as noted under Edit File.

5.5 Edit Buffer

General Operation

This command calls up the hex editor for the 6000 buffer. The buffer is used as the master data to program devices from. Also all devices are read into the buffer. Edit Buffer can only be used for memory and microcontroller devices.

The edit buffer screen has three regions. The left hand region is the address of the data. The middle region is the device data displayed in hexadecimal format. The right hand region is the ASCII equivalent of the hexadecimal data.

The editor is automatically configured to the current wordsize. Note that the address are also set to this wordsize. So a 16 bit address is half the equivalent 8 bit address.

On entering the editor, it is automatically active and any valid data typed in will be entered into the buffer with PLD logic devices the Jedec fuse map is shown.

To move about the edit screen you can use the cursor keys. To move quickly use the PgUp and PgDn keys.

To exit the editor type <F2> or click on the close box in the top left hand corner. Your changes will be automatically written to the buffer.

With PLD logic devices the JEDEC fuse map is shown.

Edit Menu

To invoke the edit menu press <ALT+F to B > to open Edit in the top left corner of the screen.

GoTo

This will move the cursor to a specified address.

Section 6 OPERATIONS

6.1 Program

This will program data from the buffer into the device in the ZIF socket. A progress window will be displayed in the top left hand corner of the screen. The program command will take account of several settings or options the user may have selected.

Select Device:	The algorithm for the currently selected device will be used.
Insertion Checks:	The options selected for tests before programming will be used. See Section 7.6.
Device Start:	Programming will start at the address specified in this field. See Section 7.2.
Device End:	Programming will terminate at the address specified in this field. See Section 7.2
Buffer Start:	Data for programming will be used starting from this address. See Section 7.2
Wordsize:	See Section 7.3
Number of blocks:	See Section 7.3
Current Device:	If using set programming this will be the particular device in that set. See Section 7.3
Checksums:	If checksums are enabled, after programming a checksum of the type specified will be displayed in the status window. If there are several devices in the set, all the set device checksums will be displayed See Section 7.4.
Programming Auto Options:	Depending on these settings, other operations may be carried out before and after programming. See Section 7.7.

If programming is successful, a success box will be displayed. Any failures will also be displayed, indicating what address or location the programming has failed on.The Information Window displays a log of operations and results.

While the programming operation is in progress, the Busy LED (green) will be illuminated on the top of the 6000. Do not remove the device while the Busy LED is on. If there is an error the Error LED (yellow) will illuminate.

6.2 Read

This will read data from the device in the ZIF socket into the buffer A progress window will be displayed in the top left hand corner of the screen. The read command will take account of several settings or options the user may have selected.

Select Device:	The algorithm for the currently selected device will be used
Insertion Checks:	The options selected for tests before reading will be used. See Section 7.6.
Device Start:	Reading will start at the address specified in this field. See Section 7.2.
Device End:	Reading will terminate at the address specified in this field. See Section 7.2
Buffer Start:	Data from the device will be written to the buffer from this address. See Section 7.2.
Wordsize:	See Section 7.3.
Number of blocks:	See Section 7.3
Current Device:	If using set programming this will be the particular device in that set. See Section 7.3.
Checksums:	If checksums are enabled, after reading a checksum of the type specified will be displayed in the status window. If there are several devices in the set, all the set device checksums will be displayed. See Section 7.4
Programming Auto Options:	Depending on these settings, other operations may be carried out after reading. If the Verify Level is not set to OFF, the data will be read first into the buffer and then verified against the data again read from the device.

If reading is successful, a success box will be displayed. Any failures will also be displayed, indicating what address or location the read has failed on. The Information Window displays a log of operations and results.

While the reading operation is in progress, the Busy LED (green) will be illuminated on the top of the 6000. Do not remove the device while the Busy LED is on. If there is an error the Error LED (yellow) will illuminate.

6.3 Verify

This will verify data from the buffer against data read from the device in the ZIF socket. A progress window will be displayed in the top right hand corner of the screen. The verify command will take account of several settings or options the user may have selected.

Select Device:	The algorithm for the currently selected device will be used.
Insertion Checks:	The options selected for tests before verifying will be used. See Section 7.6
Device Start:	Verification will start at the address specified in this field. See Section 7.2.
Device End:	Verification will terminate at the address specified in this field. See Section 7.2.
Buffer Start:	Data from the device will be verified with data in the buffer from this address. See Section 7.2
Wordsize:	See Section 7.3.
Number of blocks:	See Section 7.3
Current Device:	If using set programming this will be the particular device in that set. See Section 7.3
Checksums:	If checksums are enabled, after verification a checksum of the type specified will be displayed in the status window. If there are several devices in the set, all the set device checksums will be displayed. See Section 7.4.
Programming Auto Options:	Depending on these settings, other operations may be carried out after reading. If the Verify Level is not set to OFF, the data will be read first into the buffer and then verified against the data again read from the device.

If reading is successful, a success box will be displayed. Any failures will also be displayed, indicating what address or location the verification has failed on. The Information Window displays a log of operations and results.

While the verification operation is in progress, the Busy LED (green) will be illuminated on the top of the 6000. Do not remove the device while the Busy LED is on. If there is an error the Error LED (yellow) will illuminate.

6.4 Blank Check

This operation checks that the device currently inserted in the ZIF socket is the blank or erased state. New, virgin devices are normally in the blank state.

A progress window will be displayed during the operation. The Blank Check command will take account of several settings the user may have selected.

Select Device:	The algorithm for the currently selected device will be used.
Insertion Checks:	The options selected for tests before blank checking will be used. See Section 7.6.
Device Start:	Blank check will start at the address specified in this field. See Section 7.2.
Device End:	Blank Check will terminate at the address specified in this field. See Section 7.2.

If blank check is successful, a success box will be displayed. Any failures will also be displayed, indicating what address or location the blank check has failed on. The Information Window displays a log of operations and results.

While the blank operation is in progress, the Busy LED (green) will be illuminated on the top of the 6000. Do not remove the device while the Busy LED is on. If there is an error the Error LED (yellow) will illuminate.

6.5 Bit Test

This is used to check if a non-blank device or part of device can be over-programmed. This test is not valid for PLD devices and some microcontrollers.

Programmable devices have a default state of '0' or '1'. This is either set on erase if the part is erasable, or by the technology if the part is one-time -programmable (OTP).

For example the blank state for a conventional EPROM is '1'. It is possible to program a '0' over a '1', but not vice versa. Bit check will test that no individual bits in the buffer attempt to program a '0' to a '1'.

The Bit Check command will take account of several settings the user may have selected.

Select Device:	The algorithm for the currently selected device will be used.
Insertion Checks:	The options selected for tests before bit checking will be used. See Section 7.6
Device Start:	Bit Check will start at the address specified in this field. See Section 7.2.
Device End:	Bit Check will terminate at the address specified in this field. See Section 7.2

If Bit Check is successful, a success box will be displayed. Any failures will also be displayed, indicating what address or location the blank check has failed on. The Information Window displays a log of operations and results.

While the bit checking is in progress, the Busy LED (green) will be illuminated on the top of the 6000. Do not remove the device while the Busy LED is on. If there is an error the Error LED (yellow) will illuminate.

6.5 Erase

The Erase command is used for applicable technology devices to return them to the blank state. This command is not valid for One-Time-Programmable (OTP), EPROM, bipolar, anti-fuse technologies.

The Erase command will normally erase the whole of the device selected. It will use the Insertion Check setup (see Section 7.6) before beginning erase cycle.

If Erase is successful, a success box will be displayed. Any failures will also be displayed, indicating what address or location the blank check has failed on. The Information Window displays a log of operations and results.

While the Erase operation is in progress, the Busy LED (green) will be illuminated on the top of the 6000. Do not remove the device while the Busy LED is on. If there is an error the Error LED (yellow) will illuminate.

6.6. Test Vectors

These are applicable only for PLD's. They are normally downloaded with a JEDEC file. Test vectors are a user-specified method of functionally testing the device, usually after programming. The normal Verify command ensures that the correct fuses have been programmed in the device. Test vectors exercise the device and can be used to check the device functions as intended.

Test Vectors are either created by the Logic compiler used for JEDEC creation, or by hand in a text editor. The Edit File option can be used for this. See Section 10 for details of the format requirements. Remember that the JEDEC format used to specify to the programmer how to apply test vectors over-rides all. The programmer will do exactly as specified. For example you could drive an output with an output.

Test Vectors will use the Insertion Check setup. (see Section 7.6).

If test vectors fail, a message will be displayed indicating what wasexpected on a particular pin and what was actually read back. The option is provided to continue with test vectors or abort.

6.7 Secure

Secure command is used to blow a special fuse(s) on the device to prevent fuse data from being read. Secure is a feature available on most PLD's and some microcontrollers. Once this fuse is blown, no data can be read from the device and no verification can occur. For PLD's test vectors can be used to functionally verify the device.

Secure will use the Insertion Check setup. (see Section 7.6).

SECTION 7 SETUP MENU

7.1 Select Device

This command is used to select a new device for some programming option.

On invoking this command, a manufacturer screen is displayed. Position the highlight over the required manufacturer and either press <ENTER> or click on the <OK> button.

A device selection window will now be displayed. Only devices valid for the particular hardware configuration will be displayed. For example if a PLCC type PDi is fitted, only PLCC type devices will be displayed. Devices that are supported by the 6000, but not on that hardware configuration are displayed in background and cannot be accessed.

Position the highlight bar over the required device and press <ENTER> or click on the <OK> button.

The <SHOW> button will display information about the currently selected device such as size,pins and blank state.

You can scroll through the adjacent pages of manufacturers device windows by using the <NEXT> and <PREV> buttons.

Press the <CANCEL> button if you wish to abort the operation.

Note: If a device and device package is selected which is not compatible with the last Pdi installed a warning is given.

When a new device is selected, the buffer is automatically cleared.

OPTIONAL AUTO DEVICE SELECT

Where memory device devices (e.g. EPROM) have an internal ID code then with "Auto Device Select " option the device family and device number can be identified and after configuration the device algorithm will be automatically selected.

Assuming that the device package matches the installed PDi then this function is activated by:-

- Insertion of the device in the PDi
- Select on top tool bar "Setup" menu
- Select in this "Setup" menu "Auto Device Select
- Read as OK the warning
- The ID of insertion device is now read
- Confirm or reject the read ID.

7.2 Set Addresses

This command is only active for microcontroller and PLD devices.

The Device Start field specifies where a programming command will begin access to the device. For example if Device Start is set to 2000h, a Read command will start to read the device from address 2000h, ignoring all data upto that address.

The Device End field specifies where a programming command ends access to a device. For example, if Device End is set to 2000h, a read command will not read any data after that address.

The Buffer Start field specifies where a programming command will begin access to the buffer. For example, if Buffer Start is set to 2000h, a read command will only place data in the buffer from address 2000h and above.

By pressing the <DEFAULT> button, the values for Device Start, Device End and Buffer Start are restored to the default values.

7.3 Set Options

Set programming is used to program several memory devices, a "set", from one data buffer. There are two variables in set programming, the Wordsize and the number of blocks.

Wordsize

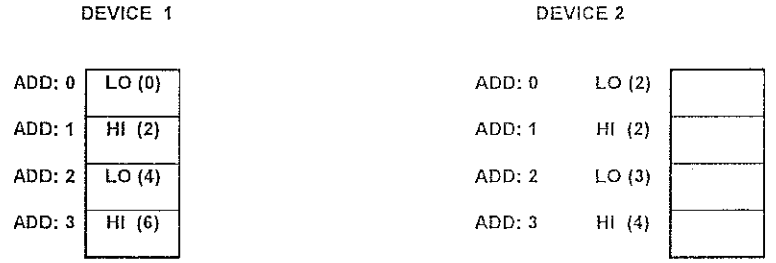
Wordsize is used to define the data width of the target set of EPROMS.The default setting is that of the device selected. For example a 27C1024 will have a Wordsize set by default to 16. However you can set the Wordsize to beyond that of the device currently selected.

You may be have a 16 bit microprocessor development system that generates a single EPROM file, although there are two 8 bit EPROMS on your target board. The first device is used for the low byte, the second for the high byte.

Your file will be organised thus:

ADD: 0	LO
ADD: 1	HI
ADD: 2	LO
ADD: 3	HI
ADD: 4	LO
ADD: 5	HI
ADD: 6	LO
ADD: 7	HI

But you wish the devices to be organised thus:



In this application, the Wordsize would be set to 16, and two devices are programmed from the same data buffer. Note that the 6000 does all data transformation of file load, so the 16 bit Wordsize, must be set first.

The 6000 supports 4, 8, 12, 16, 24, 32 and 64 bit Wordsizes and up to eight devices in a set. So the following combinations are legal:

Wordsize	Combinations
8	1 x 4bit
8	2 x 4bit
8	1 x 8bit
12	3 x 4bit
16	4 x 4bit
16	2 x 8bit
16	1 x 16bit
24	6 x 4bit
24	3 x 8bit
32	8 x 4bit
32	4 x 8bit
32	3 x 16bit
64	8 x 8bit
64	4 x 16bit

* **Note1:** Microcontroller set programming is not supported.

* **Note2:** There are no 12bit, 24bit, 32bit memory devices at the time of writing.

Another Example. For a 32 bit system, the hardware may be implemented with two 16 bit devices. The data transformation required will be:

Your file will be organised thus:

ADD: 0	Byte1
ADD: 1	Byte2
ADD: 2	Byte3
ADD: 3	Byte4
ADD: 4	Byte1
ADD: 5	Byte2
ADD: 6	Byte3
ADD: 7	Byte4
ADD: 8	Byte1
ADD: 9	Byte2
ADD: A	Byte3
ADD: B	Byte4
ADD: C	Byte1
ADD: D	Byte2
ADD: E	Byte3
ADD: F	Byte4

If the Wordsize is set to 32 bit, and the currently selected device is 16 bit and the Byte Order set to Normal, the devices will be organised thus:

DEVICE 1

ADD: 0	Byte1	Byte2
ADD: 1	Byte1	Byte2
ADD: 2	Byte1	Byte2
ADD: 3	Byte1	Byte2

DEVICE 2

Byte3	Byte4
Byte3	Byte4
Byte3	Byte4
Byte3	Byte4

If the Wordsize is set to 32 bit, and the currently selected device is 16 bit and the Byte Order set to Reverse, the devices will be organised thus:

DEVICE 1

ADD: 0	Byte4	Byte3
ADD: 1	Byte4	Byte3
ADD: 2	Byte4	Byte3
ADD: 3	Byte4	Byte3

DEVICE 2

Byte2	Byte1
Byte2	Byte1
Byte2	Byte1
Byte2	Byte1

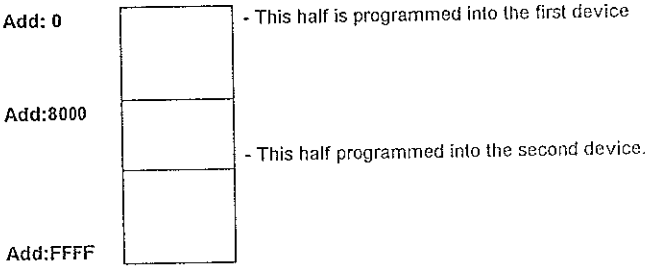
Number Of Blocks

Setting the number of blocks causes the 6000 to partition the buffer into a number of contiguous blocks. The default setting for the 6000 is a single block. (i.e number of blocks =1)

The best way to illustrate the block concept is by example. A user may have an 8 bit microprocessor system that generates an EPROM file of 512KBit. The use may wish to implement the hardware with two 256Kbit devices. In order to instruct the 6000 to divide the data in two, the number of blocks must be set to 2.

The file would be partitioned as follows:

8 bit Wordsize



It is possible to partition the data in up to 8 blocks. When the number of blocks is set, the 6000 creates a buffer equal to the number of blocks multiplied by the device size.

Using Wordsize and Number Of Blocks

For more advanced applications you may wish to use a combination of Wordsize and Number Of Blocks other than the default.

The best way to visualise the target set of EPROMS is as a matrix, with the Wordsize as the columns and the Number of Blocks as the rows.

For example with a 32 bit Wordsize and Number of Blocks = 2, with an 8 bit device selected there will be 8 devices. If the file is 2x long, then 4 devices will be programmed with data of address <x, and 4 device with data >x

ADD: 0	Byte1
ADD: 1	Byte2
ADD: 2	Byte3
ADD: 3	Byte4
ADD: 4	Byte1
ADD: 5	Byte2
ADD: 6	Byte3
ADD: 7	Byte4
ADD: 8	Byte1
ADD: 9	Byte2
ADD: A	Byte3
ADD: B	Byte4
ADD: C	Byte1
ADD: D	Byte2
ADD: E	Byte3
ADD: F	Byte4

Device Arrangement

Device1	Device2	Device3	Device4	
Byte 1	Byte 2	Byte 3	Byte 4	Data from first half of file.

Device5	Device6	Device7	Device8	
Byte 1	Byte 2	Byte 3	Byte 4	Data from first half of file.

Current Device

This is used to indicate and select the current device within a set.

Auto Set Increment

When this option is set active, after a programming operation, the next device in the set will automatically be selected. If this option is not enabled, then the user must explicitly set the Current Device before any set programming operation.

The Buffer And Set Programming

All data transformation is done when the file is loaded into the buffer. Therefore all Set Options must be in their correct state during file transfer and not set afterwards. Note that the input file is almost always treated as 8bit data, and no Wordsize formatting is embedded in that file. Checksums for each device in the set are displayed in the Device Status Window.

7.4 Checksums

This option is only applicable for memory and microcontroller devices.A range of different algorithms can be used. For most applications the default ADDITIVE selection is adequate. The CRC algorithms are more vigorous, but slower.

After a programming operation, the checksum is displayed in the Device Status Window.

For details of the algorithms, contact your local Etan agent.

7.5 JEDEC Test Vector Selections

Default X State

Some test vectors contain an X, don't care state. For some test vector implementations the manner the programmer interprets this state and applies it effects the test vector outcome.

Some JEDEC files will contain a value for this state. Otherwise it can be set to three possible states:

1: - Set to high via 470R resistor

0: - Set to low via 470R resistor

F: - Line floats at high impedance.

The default state is '0'.

See Section 10.7 for further explanations

Clocks C, K

The JEDEC for PLD programmer download formats has undergone several revisions. The latest version 3B, introduced the 'D' and 'U' fast transition fields. These field specify that an input is driven low and high respectively with a fast transition. A fast transition is used particularly on the clock pins for registered designs, where a slow rise or fall time on the clock can cause erroneous operation.

However, fast transition implies that a low impedance source is used, rather than the default 470R source. A resistive source has an advantage if the PLD is driving at the same time as the programmer.

With a resistive source, the PLD overcomes the resistive drive. With a low impedance drive, the programmer will win, and may damage the device.

Many test vectors don't use the 'D' and 'U' settings and use the 'C'and 'K' fields instead for lock pins. low transitions from resistive impedance drives can cause double-clocking errors.

In default the 6000 will use low impedance drives for 'C' and 'K' fields. You can use the Fast Clock State to override this to havetrue JEDEC 3B compliance.

7.6 Insertion Checks

Insertion checks are performed on the part before programming operations commence. There are several levels of detection.

Insertion Check

This field in enabled will perform a basic insertion check to see if a device is inserted in the ZIF socket.

Pin Continuity Test

This field if enabled will check that every pin of the device has mad connection. It is particularly useful for surface mounted devices that may easily be damaged by handling.

Reverse Device Check

This field if enable will check if the part is inserted with reverse orientation.

Electrical ID Check

Some parts, notably EPROMS contain an internal ID, which can be accessed by the programmer. If this field is inserted, the programmer will check this ID matches the ID of the part currently selected. Some PLD's have ID's, but they are always read regardless of the setting of this option.

Auto Device Select

Where memory devices (e.g. EPROM) have an internal ID code, then with this option the device family and device number can be identified, and after configuration the device algorithm will be automatically selected. This function is only used for EPROM devices.

Some Notes On Insertion Checks

For maximum safety and programming integrity, have all the insertion checks enabled. Then if a device is inserted in backwards it will not be destroyed.

If you are using a surface mount adapter in the DIP ZIF socket of a PDi 48 and the device has more pins than the DIP connection, them pin continuity tests are not valid.

Some PLD's and microcontrollers have technologies that preclude reliable reverse device and pin continuity tests. These fields have no effect for these devices.

Some microcontrollers are programmed as EPROMS in vendor-supplied adapters. Some of these adapters will fail reverse device test, and this option should be turned off.

7.7 Programming Auto Settings

These are optional operations that are invoked when the program command is executed.

Auto Blank Check

If this option is enabled, a device will be checked to see if it is in the blank state before programming occurs. If the device is not of applicable technology for a blank test, for example EEPROM, this option will be ignored.

Auto Secure

If this option is enabled, a device will be secured after the programming operation. If the device does not support a secure operation, the option will be ignored.

Auto Vectors

If this option is enabled, test vectors will be executed after the programming operation. This will only be applicable for PLD devices.

7.8 Save Setup

This operation will save the current settings of all the 6000 parameters to a user defined file. The default file is in the EL6000 directory and called EL6000.CFG. On startup, the 6000 will look for this file to restore the last used options.

If required, users may save the current settings to another file name, to be restored at a later time using the Restore command.

A configuration file contains all of relevant system set up information for a given device, details such as:

Device Family and Part number

Package Style

Technology type (PLD,EPROM etc)

Set Addresses (not PLD)

Set Options (EPROM only)

File Format Data

Checksum selection (not PLD)

Programming Auto Options

Insertion tests

Jedec Test Vector Selections (PLD only)

HARDWARE and SOFTWARE issues

Creation

Configuration files are created by the software on normal programme termination and whenever the user selects the SET:Save menu option. The file created on programme termination is named EL6000.cfg. The configuration file produced using the menu selection can be named as the user wishes. We would however recommend the use of the .cfg filename extension.

Restoration

Upon programme initialisation the EL6000.cfg file is sought in order to restore the last used system setup. If the file is not found the user will have to select the device, all other settings will be in the default state.

By accessing the SET:Restore menu selection the user may recover previously saved system configuraions.

Safety

Configuration files are evaluated before being accepted as valid.

Such considerations are outlined below:

Configuration file not corrupted.

Configuration file not out of date (incompatible software/hardware issues... changed between save restore).

Current hardware status (that any hardware change should not affect device selection, for example the PDi board changed for a PLCC device when the device to be restored is a DIL type).

7.9 Restore Setup

This operation will restore the settings of a previous setup from a named configuration file. The default configuration file is EL6000.CFG, which is stored in the EL6000 home directory.

Section 8. UTILITIES MENU

8.1 Hardware Reset

This will cause the initialisation of the 6000 programming hardware.Use this command if the machine has lost power or any hardware configurations such as the PDi have changed.

8.2 Hardware Status

Information about the particular hardware configuration will be displayed.

8.3 Edit Configuration

This command is not normally user accessible and is used for Elan personnel.

Section 9 MISCELLANEOUS MENU

9.1 Edit Encryption Data

Not user-accessible for issue HT1.00

9.2 Edit Security Bits

This command is only accessible for the microcontrollers with special bit settings, such as the Intel/Philips 87C51 series and the Microchip PIC devices. Refer to the manufacturer's data manual for the function of each bit.

9.3 Edit UES

The UES is a user programmable string of 8 characters that can be programmed into PLD's of the GAL type. When a device is read, this command will display the UES of the device read.

9.4 Batch File Facility

GUIDING NOTES.

This batch file facility provides the instructions for programming memory devices with single key stroke directly from an embedded batch file sequence in a text file.

The batch file structure can be written, in line with the fundamental pattern for with programming memory devices.

The range of options, in addition to the basic pattern, are most useful and will be further expanded.

Before a structure is indicated some general points should be understood.

1. The batch file is generated via a text editor in non doc mode.

2. The file name can be up to 8 char and is ended via . tbf (--------.tbf)

3. The written batch files can be loaded in the el6000 directory ready for use.

4. The basic pattern, as always, is to select the current wanted device, followed by the loading of the data and concluded in general with the programming operation.

5. When the batch file is written, and saved and loaded in to the el6000 directory, the batch file can be selected and run via the RUN BATCH, in the top bar of the Model 6000 main screen.

Example:

If the user has selected the intel 28F256 plcc device and set all the normal conditions with this device this can be saved as

IN28F256.CFG full device selection data.
Thus when this file is called the device will be selected. (RESTORE)

In this note the batch file content is shown in caps and the comments are in lower case.

Thus

file IN28F010.TBF

file content .
REMARK2:TO PROGRAM INTEL 28F010 PLCC PROM file comment
REMARK1:TO PROGRAM INTEL28F010 PLCC screen comment
RESTORE SETUP:"IN28F010.CFG"

This is one method to select devices with all the peripheral settings of the system, (useful system to have).
This covers settings in options, insertion, check sum type, program auto, marginal verify, and device setting with package and file name as will be seen.

REMARK1:"NOTICE WITH PLCC DEVICE INDENT CORNER"
PAUSE:"INSERT DEVICE INTEL 28F010PLCC"
CHANGE DIRECTORY:"C:\EL6000"
PAUSE:"CHANGED DIRECTORY"
the pause command is added for evaluation purposes because the user may want to see what is happening, but this could be left out.

PAUSE:"AM LOADING FILE"
LOAD FILE:"AT29F256.EHX",HEX,0,7FFF,0,BYTE_NOM,PRECLEAR_YES

Notice the file as string and data range in terms of buffer range and offset, byte order and preclear on. (to clean buffer)

PAUSE:"DOING INSERTION CHECK"
INSERTION CHECK:INSERTION_ON,ELECID_ON
PAUSE:"DO CHECK SUM"
CHECKSUMS:ADDITIVE there are other checksums possible.
PAUSE:"ERASE FLASH"

With normal Eprom devices the erase command cannot be used.

ERASE: It is a flash device.
PAUSE:"BITTEST"
BIT TEST:
PAUSE:"BLANK"
BLANK CHECK:
PAUSE:" DO PROGRAM FROM BUFFER"
PROGRAM:

With this pattern many pause command are added, as this will help to follow what is taking place.

Have added additional verify but this can be normally form a part of auto setting

VERIFY:
CHECKSUM:

This is the general pattern and can be repeated for other devices.

Other method of device selection:

PAUSE:"ENSURE PDi IS MATCHING NEXT DEVICE PACKAGE"
SELECT DEVICE:INTEL (PROM),"27C256"
same notation as on screen.

SET ADDRESSES:00000000,0000FFFF,00000000
This last notation consist of start address data, end address data, buffer offset, and is used when deviating from the default settings.

SET OPTIONS:8,2,1,AUTO_ON
8 bit wide,2 blocks,first device,auto_on

PROGRAMMING AUTO SETTINGS: (1), (2), (3), (4), (5).

	(1)	(2)	(3)	(4)	(5)
Eprom	BLANK_ON	BIT_OFF	ERASE_OFF	MARG_5	
Micro	BLANK_ON	BIT_OFF	SECURE_ON	MARG_5	VERIFY_3V/5V
Pid	BLANK_ON	SECURE_ON	TESTVECTORS_ON		

We do not select the device package but warn the user to match PDi required.

Useful method for repeat programming operation is the repeat notation.

REPEAT:"LABELX",10
do what is required
REPEATEND:"LABELX"

We can load prom files, as shown and logic will be a 22v10.jed or pof form as required in the same manner as shown above. Notice that many commands do not need addition parametric data but other will for the purposes of selection that must be made.

BIT TEST:
BLANK CHECK:
ERASE:
PROGRAM:
READ:
SECURITY:
TEST VECTORS:
VERIFY:

Logic command

JEDECTESTVECTORS SELECTIONS:
DEFAULTX_LO/HI/FLOAT,FAST_OFF/ON,PULLUP_OFF/ON all subject to the jedec file content

Eprom serial numbers.

The next new feature is the placing of serial numbers in the loaded memory map, providing place is left clear for this, and programming these in ascending order from a start number or values is possible as shown.

The placing of a two digit serial number at an specified address and in incremented for the next device can be seen in the following code:

```
RESTORE SETUP:"IN28F010.CFG"
CHANGE DIRECTORY:"C:\EL6000"
LOAD FILE: "AT28F020.EXH",HEX,0,7FFF,0,BYTE_NORM,PRECLEAR_YES
PAUSE:"LOAD TARGET DEVICE"
INSERTION CHECK:INSERTION_ON,ELECID_ON
CHECKSUMS:ADDITIVE
```

This so far, is the setting up of the programmer and loading in of data .

```
SET SERIAL NO:          will ask for start value.
REPEAT:"LABELX",3
WRITE SERIAL NO:1,000000A1   write this value at A0 address.
WRITE SERIAL NO:2,000000A0
PAUSE:"INSERT THE NEXT TARGET DEVICE"
PROGRAM:
INCREMENT SERIAL NO:
REPEATEND:"LABELX"
```

This example will permit the user to deposit up to four characters in to the memory area at defined address A0

Section 10 TEST VECTORS

10.1 Test Vector Basics

Test vectors are used to functionally test a PLD device after programming. The test vectors are appended at the end of the JEDEC file. The format of the test vectors is laid down in a JEDEC standard. It is not a complex standard, and users can edit their own vectors for particular test sequences.

A test vector consists of a test vector number, a series of letters and numbers for each pin and a terminating character "*".

E.g.

V0001 01111110000000001000N110HHHHHHHHHH0110100N*

Each pin of the device must be represented by a letter or number. The 6000 will support up to 999999999 test vectors, the main limitation being disk space to store the vector sequence.

10.2 Test Vector Commands

'0' - Drive pin low. On the 6000, this is via a 470R resistor.

'1' - Drive pin high. On the 6000, this is via a 470R resistor.

'L' - Test pin for low.

'H' - Test pin for high.

'N' - Use for power pins, untested outputs and no-connects.

'C' - Drive input low, high and then low.

'K' - Drive input high, low and then high.

'D' - Drive input low, fast transition.

'U' - Drive input high, fast transition.

'X' - Output not tested, input default level.

'Z' - Test input or output for high impedance.

'F' - Float pin.

10.3 Resistive Drive

For the '0' and '1' commands the 6000 applies a logic low or logic high via a 470R resistor. This protects the device if inadvertently the 6000 is trying to drive an output.

10.4 Clocked Designs

If the '0' and '1' command are used in a sequence in order to clock a registered device, beware. As the drive is through a 470R resistor, the rise and fall times are slower and in the order of 100ns to 200ns. This may give problems on some faster parts are cause double clocking.

It is recommended that the 'D' and 'U' commands are used for dedicated clocks on registered designs. These are low impedance drives that achieve rise and fall times of less the 20ns.

The 'C' and 'K' commands will produce a positive and negative going pulse respectively. The default drive for these pins is low impedance drive. This can be changed to high impedance drive in the Setup Menu using the JEDEC Test Vector Selections command (see section 7.5).

10.5 Vector Passes

The 6000 will make three passes on each vector. It sequentially processes the test vector commands starting at the lowest pin number unless the 'P' sequence command has been encountered. (see below). On the third and last pass it will test any pins designated as test pins by the test vector commands.

10.6 Float and High Impedance

The 6000 has advanced pin drivers that will allow pins to be presented with a high impedance above 1M. If you are using CMOS devices, take care as floating inputs can cause erroneous operation.

10.7 'X' State

This is defined in the JEDEC standard as the "don't care" state. Unfortunately some designs do care and the 'X' state must be carefully handled. The 6000 in default will apply a low level via a 470R resistor.

The default 'X' state can be changed in two ways. It can be set in the Setup Menu using the JEDEC Test Vectors Selections to '1' (drive high), '0' (drive low) or 'F' (float). Alternatively it can be set in the JEDEC file on download using the 'X' command.

E.G. "X1*" will define any 'X's' in the test vectors to be set high.

The 'X' command is usually defined before fuse data.

10.8 'P' Sequence

The test vectors are executed sequentially starting from the lowest pin Many logic compilers will generate test vectors for DIP PLD pinouts only. Alternatively test vectors may have been developed by hand in development for a DIP package and production uses PLCC. You can use the Test Vectors designed for a DIP package on alternative pinouts of the same device by using the "P" sequence field in the JEDEC file. Use the edit file option and add a P sequence field.

For example a 40 pin dip test vector may appear in the JEDEC as follows:

C53B7*
V0001 011111100000000001000N110HHHHHHHHHH0110100N*
V0002 011111100000000001000N100HHHHHHHHHH0111000N*
□0000

The 6000 will apply the vector in an incrementing order of pins, that is pin1, pin2 etc to pin 40. Adding the P sequence will reorder this.

C53B7*
P 2 32 4 5 6 7 1 10 12 14 16 18 20 22 24 26 28 30 3 34 36 38
8 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 40 39*
V0001 011111100000000001000N110HHHHHHHHHH0110100N*
V0002 011111100000000001000N100HHHHHHHHHH0111000N*
□0000

This vector will be processed by the 6000 starting at pin 2, pin 3 and through to pin 39.

Note if you edit the JEDEC file, the transmission checksum may be made invalid. You can suppress the error warning by editing the transmission checksum to 0000. The transmission checksum is prefixed by "□" symbol.

Some PLCC pinouts have more pins than the DIP package, either no connects or power pins. You can overcome this by adding the "N" symbol at that location.

The above example for a EP900 may be used on the 44 pin PLCC version. But the 44 pin PLCC version has no connects on 17 and 39, VCC's on 1 and 44 and GND's on 22 and 23. It has no ground on 20 or VCC on 40.

The first step is to remove the "N" field on 20 and 40, then add "N" fields on 1,17,22,23,39 and 44.

V0001 N011111100000000N1000NN110HHHHHHHHHH011N0100N*
V0002 N011111110000000N1000NN100HHHHHHHHHH011N1000N*
□0000

10.9 Level Testing

The 6000 will test pins for normal TTL in standard mode. That is less than 0.8V for logic low and greater than 2.4V for logic high.

10.10 Editing Test Vectors

Test vectors are normally appended to the JEDEC file after the fuse data and after the fuse checksum, but before the transmission checksum.

The fuse checksum is of the form "Cxxxx*" where xxxx are hexadecimal characters. The transmission checksum is of the form "<ETX>xxxx". Where <ETX> normally appears as a "heart" character and xxxx is hexadecimal data.

Note that is you edit test vectors, the transmission checksum will be incorrect. The 6000 will issue a warning when loading a JEDEC file with an incorrect transmission checksum. One way out of this is to set the transmission checksum to "0000". The 6000 will then ignore the transmission checksum. The other way is to load the JEDEC into the 6000 buffer and then to save it with the Save Buffer command. The 6000 will add the correct transmission checksum.

Section 11 PROGRAMMING TIPS

11.1 UV Light

Some EPROMS are sensitive to sunlight while being programmed. As a precaution us a special opaque label over the window while programming.

11.2 Static

CMOS device are static sensitive. The 6000 is statically connected to safety earth. Ensure that you use static precautions when handling static devices.

11.3 Power Interruption

Although the 6000 is fully protected, it is not recommended that you remove devices during any programming operations. Similarly do not reboot your PC or remove the centronics cable during a programming cycle. The 6000 is equipped with special hardware that detects a break in the communication links to the PC and safely powers down the device in the socket. However some devices may still be damaged.

11.4 Dirt

The most common cause of programming failures is dirt in the ZIF sockets. Try not to let these get too dirty.

11.5 Programming Algorithms

Every programmable device has its own unique programming algorithm designed to ensure a high yield of programming and long term data reliability. Therefore you cannot program a device under any setting that you think will do. All 27128's don't program under Intel 27128 setting. Please ensure that you select the correct device type for the device in your ZIF socket. If you are not sure, contact your local Elan representative. He will either advice you of a suitable setting or be able to supply you with upgrade information to support you device. Semiconductor manufacturers often change programming algorithms. To ensure you are using the correct algorithm, ensure you have the latest 6000 software update.

11.6 Surface Mount Packages

Surface mount packages are delicate, even PLCC types. Handle them with care otherwise you will get pin connection problems in the ZIF socket.

11.7 Test Vectors For Non-DIP Packages

Many logic compilers will generate test vectors for DIP PLD pinouts only. Alternatively test vectors may have been developed by hand in development for a DIP package and production uses PLCC. You can use the Test Vectors designed for a DIP package on alternative pinouts of the same device by using the "P" sequence field in the JEDEC file. Use the edit file option and add a P sequence field.

For example a 40 pin dip test vector may appear in the JEDEC as follows:

```
C53B7*
V0001 011111100000000001000N110HHHHHHHHHH0110100N*
V0002 011111110000000001000N100HHHHHHHHHH0111000N*
□0000
```

The 6000 will apply the vector in an incrementing order of pins, that is pin1, pin2 etc to pin 40. Adding the P sequence will reorder this.

```
C53B7*
P 2 32 4 5 6 7 1 10 12 14 16 18 20 22 24 26 28 30 3 34 36 38
8 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 40 39*
V0001 011111100000000001000N110HHHHHHHHHH0110100N*
V0002 011111110000000001000N100HHHHHHHHHH0111000N*
□0000
```

This vector will be processed by the 6000 starting at pin 2, pin 3 and through to pin 39.

Note if you edit the JEDEC file, the transmission checksum may be made invalid. You can suppress the error warning by editing the transmission checksum to 0000. The transmission checksum is prefixed by "□" symbol.

Some PLCC pinouts have more pins than the DIP package, either no connects or power pins. You can overcome this by adding the "N" symbol at that location.

The above example for a EP900 may be used on the 44 pin PLCC version.
But the 44 pin PLCC version has no connects on 17 and 39, VCC's on 1
and 44 and GND's on 22 and 23. It has no ground on 20 or VCC on 40.

The first step is to remove the "N" field on 20 and 40, then add "N"
fields on 1,17,22,23,39 and44.

```
V0001 N011111100000000N1000NN110HHHHHHHHHH011N0100N*
V0002 N011111110000000N1000NN100HHHHHHHHHH011N1000N*
□0000
```

11.8 Using Microchip PICS

The Microchip PIC series of microcontrollers have four configuration bits, that determine the setting of some hardware features of the device. Some of the OTP parts already have these set.

The function of the four config bits is shown below.

1XXX Memory UnProtected.
0XXX Memory Protected.
X1XX Watchdog Timer Enabled.
X0XX Watchdog Timer Disabled.
XX11 RC Oscillator.
XX10 HS-High Speed Crystal.
XX01 XT-Standard Crystal.
XX00 LF-Low Frequency Crystal

When a device is read, the setting of the config bits is displayed in the device status window.

The config bits can be changed by selecting the Edit Security Bits command under the Utilities menu.

Section 12 TROUBLESHOOTING

- Can't find the device required to be programmed under the selection screen.

Every programmable device has its own unique programming algorithm designed to ensure a high yield of programming and long term data reliability. Therefore you cannot program a device under any setting that you think will do. For example, All 27128's don't program under Intel 27128 setting. Please ensure that you select the correct device type for the device in your ZIF socket. If you are not sure, contact your local Elan representative. He will either advice you of a suitable setting or be able to supply you with upgrade information to support you device.

- The device required is shown on the device selectionscreen, but is not in highlight and cannot be accessed.

The 6000 intelligently interprets the hardware configuration of the particular programmer fitted. It will find the number of pin driver cards fitted and hence calculate the number of pin drivers available. It will also read the identification code of the PDi fitted and hence be aware of what package styles are supported. Therefore it only displays in highlight those devices that can be supported on the current hardware configuration. If you change the PDi, or upgrade the hardware in any way ensure that you execute the Hardware Reset command under the UTILITIES menu.

- The mains power on the 6000 programmer was interrupted and now the programmer doesn't operate properly.

The 6000 hardware resets itself on power-up. Execute the Hardware Reset command under the UTILITIES menu.

- I get intermittent programming failure when running under Windows.

The 6000 uses exact timing loops to program devices. Any interference with these can give erroneous results on some devices. When running under Windows use Exclusive operation (set in installed PIF file).

-Structured test vectors pass simulation, but sometimes fail during 6000 test vector test?

There are a number of variables which are not always considered by simulators, but have an impact during functional test. These include design considerations, device characteristics, and programmer test hardware/algorithm attributes.

Designs containing non-registered feedback may behave differently during software simulation than during structured test if the simulator does not properly handle this condition. Designs which contain race conditions or RS flip-flops will often simulate differently from the way the part actually works during functional test. This can happen because the simulator applies the inputs simultaneously and assumes equal propagation time through the device, while the programmer may apply the inputs in a particular sequence or a byte at a time. Therefore, propagation delays become a real factor. This can be compensated for by using several vectors to apply the inputs in the sequence that the design would encounter in the target circuit.

"Don't care" conditions are another reason simulation and test may disagree. A don't care state can be assigned or assumed by a simulator to be always a one or always a zero. This may differ from what is actually applied to a device during functional test and thereby possibly affect the output. (See Section 10.7).

The devices themselves may also function differently than the simulator assumes. Some manufacturer's registered devices preset on power up where as other brands do not. And, some devices require that the first vector receive a double clock. Again, simulators may assume a power up state which will not always be encountered during functional test, or may fail to take into account any programmable polarity influence on the power up output state. It is always a good practice to write vectors which will first preset the device to a known state by activating a dedicated preset pin or preset product term. Also, vectors that were developed for one manufacturer's device which has registered preload capability may fail on a second source device which does not have this feature. All logic devices are not created equal.

- My EPROM based devices occasionally fail blank check

Suspect your eraser. Erasing EPROMs, Microcontrollers, and EPLDs with Ultra Violet (UV) light is a simple process, if you take some important facts into consideration.

UV erasable devices use a floating gate transistor as the programmable element. The charge or lack of charge in the floating gate determines the OFF/ON state of the bit or cell. Normally this charge remains in the floating gate indefinitely.

Short wave UV radiation at the proper wavelength and intensity raises the energy level of the electrons in the floating gate, allowing the charge to leak away. Sun light and fluorescent light contain enough short wave radiation to cause partial erasure over time. So, after programming, the device window should always be covered with an opaque label.

The UV lamp must have a wavelength of 2537 angstroms and the device must receive a minimum integrated dose of 15W-sec/cm2 (intensity x exposure time) for proper erasure. For example, a lamp producing12,000uW/cm2 (0.012W/cm2) would typically erase a device in about 21minutes.

Be sure that the device window is clean, free of any label adhesive residue or fingerprints, which can filter the UV light.

As a UV bulb ages, the intensity diminishes, so you'll need to increase the exposure time or replace the bulb. Refer to the lamp manufacturer's specifications to determine the minimum exposure time needed as the age of the bulb increases, or when to replace the bulb. Some UV lamps have an hour meter for this purpose, and automatic shut-off timers to increase the bulb's usable lifetime.

A device may check blank after only a short exposure time, but expose it for the full minimum time to insure that the gate charge levels are sufficiently below threshold. Insufficient UV exposure can cause reduced access time or unstable bits resulting in programming/verify errors or intermittent operation in the target system.

Standard precautions should include anti-static protection for the devices by using conductive foam or open top, anti static tubes. Also, avoid exposure of the eyes or skin to the UV light.

- Programming my EPROMS is intermittent

Some EPROMS are UV sensitive during programming. Ensure an opaque label is stuck across the window.

Section 13 SERVICE AND SUPPORT INFORMATION

For hardware servicing, calibration, software updates and technical support, in the first instance contact your local distributor.

Otherwise, or if in the UK, USA or Canada contact:

ELAN DIGITAL SYSTEMS LTD, FAREHAM ENGLAND.

TEL: (0) 1489 579799
FAX: (0) 1489 577516

ELAN SYSTEMS/ASCEND, LIVERMORE, CA. USA

TEL: 510 606 2000
FAX: 510 606 2006

If you are within warranty period or have a maintenance agreement, software updates may be obtained from the Elan BBS system.

BBS (0) 1489 578979

Glossary

ZIF	-	Zero Insertion Force
PDi	-	Programmable Device Interface. The hardware inteface to carrying the device ZIF socket the plugs into the top of the 6000 Pod.
Hexadecimal	-	Abbreviated to Hex. A number of base 16 (0 to F)
Format	-	A format is a method of encoding data in a specified manner. The format is either to allow the data to be user readable and in ASCII format that can be handled by communications packages, or to facilitate error detection.
JEDEC	-	This is data format for most PLD's to be accepted and produced for device programmers.
POF	-	POF is an alternative to the JEDEC format as used by Altera for MAX devices.
ROM	-	Read Only Memory. A memory technology with a data pattern set at the factory of manufacture.
PROM	-	Programmable Read Only Memory. A memory technology that can be programmed but cannot be erased.
EPROM	-	Erasable Programmable Read Only Memory. A memory technology that can be programmed, but only erased with the application of intense UV light.
FLASH	-	A memory technology that can be programmed and erased electrically. Often only the whole chip or zones of the chip can be erased.
EEPROM	-	Electrically Erasable Programmable Read Only Memory.This memory technology can be programmed and erased electrically. Erasure can normally be carried out on an individual byte by byte basis.
PLD	-	A Programmable Logic Device. Generic name to encompass a myriad of logic devices using programmable technology.
GAL	-	A particular PLD of EEPROM technology, first marketed by Lattice

APPENDIX A OTHER ELAN PRODUCTS

A.1 EF-PER SERIES Device Programmers

(Extra Fast - Programming Enhancement Routine)

Model 3000	-	PC Remote Universal Programmer
Model 4000	-	EPROM Copiers (Standard and Turbo)
Model 5000	-	Stand-Alone/PC Universal Programmer (Standard and Turbo)
MOdel 5-J08	-	Memory Card Copiers (Turbo)
EXFILE	-	PC Remote Software

A.2 Programming Adapters

Axxx	-	Various items to suit specific device types and packages.

A.3 J-SERIES PCMCIA Card Products

J101/2	-	PCMCIA Card R/W units (Internal or External) (all memory cards including OTP, I/O, ATA/IDE disk)
J103/4	-	PCMCIA Card R/W units (Internal or External) (all memory cards excluding OTP, plus I/O etc.)
J105/6	-	PCMCIA Card R/W units (Internal single and dual)
JC_xxx	-	PC Software packages for R/W units

A.4 FT-SERIES PCMCIA Card Verifier/Testers

FT200/1	-	Functional Verifier/Tester of all PCMCIA Cards

Please ask for data sheets on any product of interest.

APPENDIX B ERROR MESSAGES

B.1 JEDEC Input Transfer

"File not JEDEC format file"

The user has selected a PLD and is using the Load File command. The 6000 is expecting a file of the JEDEC format. Check the filename spelling and that the file has not been corrupted.

"Too many warnings"

When loading a JEDEC file into the buffer, the 6000 will display warnings if it encounters possible problems. If there are too many warnings the 6000 will abort the file transfer.

"Not ASCII character"

The JEDEC format specifies that characters should be printable ASCII characters. The 6000 has encounter a character that is not ASCII.Check the file is JEDEC format and has not been corrupted.

"Unexpected end of file"

The 6000 has encountered an end of file marker before the expected end of the JEDEC file. Normally the JEDEC file is terminated by a transmission checksum. The 6000 did not find this.

"Bad Number of Fuses Field"

The 6000 found a non-valid "QF" field.

"Bad vector number character"

The 6000 found a bad non-numeric character in the "QV" (number of vectors) field.

"Bad Fuse Number"

The number following the "L" field designating a fuse number was found to be invalid.

"Vector has more pins than device"

While processing the vector it was found that more pins were specified than are present on the currently selected device.

"End of file or file error"

A general error message indicating some kind of access problem with the input JEDEC file.

"Bad or Missing Q Field"

There is a problem with the "QF", "QP" or "QV" fields. Check JEDEC syntax.

"P field has too many pins"

The P sequence field has more pins specified than there are on the currently selected device.

"P field has too few pins "

The P sequence field has less pins specified than there are on the currently selected device.

"Unrecognised field"

The field processed in the JEDEC is not recognised to conform to the JEDEC 3B standard. Check syntax of JEDEC file.

"Unsupported Field"

The JEDEC file contains a field directive that is not supported on this version of software.

"Fuse checksum incorrect"

The fuse checksum field value does not agree with the checksum calculated by the 6000 on the fuse data in the JEDEC file.

"Fuse number too big. field ignored"

The fuse number processed is bigger than the total number of fuses in the device. The fuse data following this number has been ignored.

"Vector number in QV has too many digits"

Too many vectors are specified in the "QV" field.

"Bad pin number in QP field",

The pin number in the "QP" field is not numeric(0 to 9).

"Pin number in QP field too big",

The pin number in the "QP" field is too large.

"Bad default fuse field "

The value in the "F" field is not valid. Valid values are:

"F1*" or "F0*" "

"Fuse number greater than QF field"

The fuse number processed is bigger than the total number of fuses in the device. The fuse data following this number has been ignored.

"Bad Fuse checksum digit"

The digits in the "C" fuse checksum field are not hexadecimal.

"Too many chars in FuseChecksum"

There are too many digits in the fusechecksum field, there should be 4.

"Vector number greater than QV field"

The "QV" field specifies to the 6000 how many vectors are present in the file. The 6000 has encountered more vectors actually present than specified.

"Bad default X field"

The value in the default "X" field for the don't care state in test vectors is not valid. Valid values are:

"X1*", "X0" and "XF"

"Bad autosecure field"

The autosecure field "G" is not valid.

"Bad transmission checksum"

The 6000 checks all the data in the JEDEC file to ensure that it corresponds to the JEDEC transmission checksum.

"Missing QF (No of Fuses) Field"

There is no number of fuses field. The 6000 checks this to see if this corresponds to the device currently selected.

"Missing Fuse Checksum Field"

The "C" field is missing. The 6000 has no method of checking the validity of the fuse data.

"Test vectors missing- QV field"

The "QV" field specified more test vectors than the 6000 actually encountered.

"Bad Pin Sequence number"

The number encountered in the "P" field was not numeric (0 to 9)

"P Field pin number too big",

The number encountered in the "P" field was bigger than the number of pins in the currently selected device.

"QP field and device selected mismatch"

The "QP" field specifies how many pins the device has. The 6000 has detected that this does not correspond to the number of pins the currently selected device has.

"Qf field mismatch with current device"

The "QF" field specifies how many fuses the target device has in the JEDEC file. The 6000 has detected that this does not correspond to the currently selected device.

B.2 Test Vectors

"No Vectors Present"

No vectors have been loaded into the buffer.

"File Error or unexpected end of file"

There is an error accessing the buffer.

"Test Vectors Aborted"

The user aborted the test vectors before completion.

"Test Vectors Failed"

The expected test data during the test vector operation did not correspond to the actual data read back.

"Test Vectors Complete With Errors"

Test vectors have been completed, but errors were encounter.

"Error in Vector XX, Field not supported"

The vector XX contains a test vector command not supported by this 6000 release.

"Error in Vector XX, Vector shorter than number of pins"

The vector XX has less commands than number of pins.

"Error in Vector XX, Bad character Y"

The 6000 has encountered a non-valid test vector command 'Y' in test vector XX.

"Error in Vector XX, Vector too long for number of pins"

The vector XX has more test vector command characters than the number of pins of the currently selected device.

B.3 JEDEC Output Transfer

"Cannot Open Input File"

There was an access problem with the buffer.

"Cannot open Output File"

There was an access problem for the target output file.

"Access Problem with output file"

There was an access problem with target output file.

"Access problem with input file"

There was an access problem with target input file.

B.4 Device Selection Errors

"Insufficient Pin Drivers for Selected Device"

The current 6000 programming hardware is not fitted with sufficient pin drivers to program the selected part. In the device selection screen, the devices that can be programmed with the current hardware configuration are displayed in bold. Other devices are in a fainter colour.

"Wrong Carrier Fitted for Selected Device";

The current 6000 programming hardware is not fitted with the incorrect PDi carrier to program the selected part package. In the device selection screen, the devices that can be programmed with the current hardware configuration are displayed in bold. Other devices are in a fainter colour.

B.5 Startup Initialisation Errors

"No Parallel Port Installed. Retry ?"

The 6000 could not find any parallel ports to attempt communications with the programming hardware.

"EL6000 Not Found. Retry ?"

The 6000 found a parallel port, but could not establish communications with any valid 6000 programming hardware. Check cable connections and that the red power LED is illuminated on the 6000 hardware.

"Invalid Serial Number! Expected format is: eg. HW00001"

The internal serial number on the 6000 is invalid. Contact your local Elan agent.

"Invalid MotherBoard Type"

Invalid or faulty 6000 hardware was detected. Contact your local agent.

"Minimum 3 Pin8 Cards required"

One or two Pin8 cards have been detected, a minimum of three are required

"Pin8 Card Insertion Error(s)"

The Pin8 card shave been inserted in the incorrect slots on the motherboard. Each slot is marked showing which pins on the ZIF socket are driven by the Pin8 card. Pin8 cards must be inserted so as to serve contiguous pin numbers.

"Relay Module Not Fitted"

The POD48, POD84 or POD128 has not been fitted to the 6000 hardware.

"ZIF Carrier Not Fitted"

The PDi carrier is incorrectly inserted or not present. Power-down and insert it.

"Invalid ZIF Carrier"

The PDi carrier detected is invalid. Check insertion or contact local Elan agent.

"Fatal Error!"

A fatal error has been detected on 6000 hardware initialisation. The 6000 cannot be used, contact your local Elan agent.

"Supply Rail Calibration Fail!"

The 6000 has failed its internal calibration. Please contact your local Elan agent.

"Relay stuck on"

The 6000 initialisation has found a hardware fault in the POD48, POD84 or POD128. Contact your local Elan agent.

"Ground Fail"

The 6000 initialisation has found a hardware fault in the POD48, POD84 or POD128. Contact your local Elan agent.

"TTL Driver Fail"

The 6000 initialisation has found a hardware fault. Contact your local Elan agent.

"Super Voltage Stuck On"

The 6000 initialisation has found a hardware fault. Contact your local Elan agent.

"Threshold detect fail"

The 6000 initialisation has found a hardware fault. Contact your local Elan agent.

"Mapping Failed"

The 6000 initialisation has found a hardware fault. Contact your local Elan agent.

"More Pin Drivers than Grounds"

The 6000 is fitted with more pin drivers than relay grounds on the POD48, POD84 and POD128. Therefore the limit on pin count of devices is determined by the POD48 (48pins), POD84 (84pins) and POD128 (128 pins). This is not a fatal error.

"Please remove device"

The 6000 initialisation has detected a device. It cannot continue test until this device is removed.

"Device Present or Hardware Error(s)"

The 6000 initialisation has detected a device. It cannot continue test until this device is removed. If you still get this message despite removing the device, please contact your local Elan agent.

"Capacitor Failed"

The 6000 has detected a decoupling capacitor failure. This is not fatal but may give poor programming yields on some devices.

"EL6000 Time Out Error"

The 6000 communications to the hardware have failed. Power the unit off then on, reboot the PC and try again.

B.6 General Errors

"Device Insertion Error(s)"

The device is not inserted or inserted incorrectly.

"Illegal Signature" or "Illegal Device ID"

The 6000 detected a device with an internal ID that differed from that expected for the selected device.

"Reversed Device" or "Device Inserted Backwards"

The device has been inserted incorrectly or is faulty. This error may appear on some third-party adapters for microcontrollers. Turn the reverse device setting off under the Settings Menu.

"Continuity Error"

The 6000 has detected a bad connection between the pins and ZIF socket. Check that the contacts on the device are true and clean. Otherwise try switch cleaner on the ZIF socket or contact your local Elan agent.

"Illegal Algo revision." or "Illegal device code!"

The 6000 has detected a device with internal information designating which programming algorithm to use. The code read back is not recognised.You may require a software update, contact your local Elan agent.

"Illegal device revision ID."

The 6000 has detected a device with internal information designating which programming algorithm to use. The code read back is not recognised.You may require a software update, contact your local Elan agent.

"MES failed to program."

Part of the programming sequence has failed on a GAL type PLD.

"Failed to secure device." or "Security failed."

The secure sequence failed. The device has not been secured.

"Revision %x not yet supported."

The 6000 has detected a device with internal information designating which programming algorithm to use. The code read back is not recognised.You may require a software update, contact your local Elan agent.

"Turbo bit failed programming."

A programming failure occurred on a PLD device.

"Select 25/30/35/40 speed!"

A programming operation has been attempted on an Altera "Classic" device of the newer faster type. Please select the slower speed variants.

"Could not erase device."

There was a failure in the erase cycle.

"Not enough memory for operation"

The operation requested needs more memory. There is insufficient memory left in the system. Exit and optimise your system.

"Insufficient disk space"

There is not enough space left on the hard drive you are using. Exit and do some housekeeping.

"Invalid Config file"

This message occurs either on initialisation or restoring a previous configuration. Either the file had been corrupted or it is not a valid 6000 configuration file.

"Could not open help file."

The help file EL6000.HLP could not be found.

"Could not restore device"

The configuration file restored contains a device selection that is not valid for the current configuration.

"Issue discrepancy between restored/current Hardware"

The configuration file restores was saved by a 6000 of a different hardware setup.

"Invalid Config file ...aborting Operation."

The configuration file selected or the default EL6000.CFG has been corrupted.If the EL6000.CFG has been corrupted, delete it.

"Too many files"

This message is displayed if on a file selection window has a mask that will show a large number of files. Too many files means that all the files conforming to this mask cannot be displayed. If you cannot view any files at all it probably means you are very close to running out of memory.

APPENDIX C TEXT EDITOR REFERENCE

Cursor Movement Commands

Character Left	<-	or Ctrl+S
Character Right	->	or Ctrl+D
Word Left	Ctrl+<-	or Ctrl+A
Word Right	Ctrl+->	or Ctrl+F
Line Up	Up arrow	or Ctrl+E
Line Down	Down arrow	or Ctrl+X
Scroll up one line	Ctrl+W	
Scroll down one line	Ctrl+Z	
Page up	PgUp	or Ctrl+R
Page down	PgDn	or Ctrl+C
Beginning of line	Home	or Ctrl+QS
End of line	End	or Ctrl+QD
Top of window	Ctrl+QE	
Bottom of window	Ctrl+QX	
Top of file	Ctrl+QR	
Bottom of file	Ctrl+QC	
Move to previous position	Ctrl+QP	

Insert And Delete Commands

Delete Character	Del	
Delete Character left	Backspace	or Shift+Tab
Delete Line	Ctrl+Y	
Delete to end of line	Ctrl+QY	
Delete word	Ctrl+T	
Insert line	Ctrl+N	
Insert mode on/off	Ins	

Block Commands

Move to beginning of block	Ctrl+QB
Move to end of block	Ctrl+QK
Set beginning of block	Ctrl+KB
Set end of block	Ctrl+KK
Exit to menu bar	Ctrl+KD
Hide/Show block	Ctrl+KH
Mark line	Ctrl+KL
Print selected block	Ctrl+KP
Mark word	Ctrl+KT
Copy block	Ctrl+KC
Move block	Ctrl+KV
Delete block	Ctrl+Del
Indent block	Ctrl+KL
Read block from disk	Ctrl+KR
Unindent block	Ctrl+KU
Write block to disk	Ctrl+KW

Marking Blocks

Left one character	Shift+ <-
Right one character	Shift+ ->
End of line	Shift + End
Beginning of line	Shift + Home
Same column on next line	Shift+ down arrow
Smae column on previous line	Shift+ up arrow
One page down	Shift + PgDn
One page up	Shift + PgUp
Left one word	Shift+Ctrl+left arrow
Right one word	Shift+Ctrl+ right arrow
End of file	Shift+Ctrl+End
Beginning of file	Shift+Ctrl+Home

Other Editing Commands

Autoindent mode on/off	Ctrl+OI
Cursor through tabs on/off	Ctrl+OR
Find place marker	Ctrl+Qn*
Help	F1
Help Index	Shift+F1
Insert Control character	Ctrl+P**
Exit 6000 control program	ALT+X
Optimal fill mode on/off	Ctrl+OF
Pair matching	Ctrl+Q[
	Ctrl+Q]
Save file	Ctrl+KS
Search	Ctrl+QF
Search and replace	Ctrl+QA
Set marker	Ctrl+Kn*
Tabs mode on/off	Ctrl+OT
Topic search help	Ctrl+F1
Undo	Alt+Backspace
Redo	Alt+Shift+Backspce
Unindent mode on/off	Ctrl+OU
Over type toggle	Ins

* Represent number 0 to 9

** Enter control character by first pressing **Ctrl+P**, then pressing the desired control character