



## **ELAN DIGITAL SYSTEMS LTD.**

**LITTLE PARK FARM ROAD,  
SEGENSWORTH WEST,  
FAREHAM,  
HANTS. PO15 5SJ.  
TEL: (44) (0)1489 579799  
FAX: (44) (0)1489 577516  
e-mail: [support@elan-digital-systems.co.uk](mailto:support@elan-digital-systems.co.uk)  
website: [www.pccard.co.uk](http://www.pccard.co.uk)**

## **MEMORY CARD EXPLORER**



## **USER & PROGRAMMER'S GUIDE**

**For important DLL licensing information  
please refer to section 2.0.**

### **REVISION HISTORY**

<b>ISSUE</b>	<b>PAGES</b>	<b>DATE</b>	<b>NOTES</b>
8	29	04.02.1999	Document now User & Programmer's Guide
9	30	29.03.1999	NT4 Additions
10	32	25.06.1999	4.4, 5.1, 5.2, 5.2.3, 5.2.11, 6.1
11	35	02.12.1999	4.1.2, 5.2.13, 5.2.15, 5.2.16, 6.5
12	36	14.12.1999	front page, 1.3, 2.0
13	36	02.03.2000	front page, 1.3.2, 2.0, 4.1.4, 5.2.1
14	36	15.03.2000	Section 1.3, 4.1.2, 5.2, 5.2.17
15	40	20.09.2000	Section 1.3, 5.1, 5.2, 5.2.11, 5.2.18, 6.5

# **CONTENTS**

<b><u>1.0 INTRODUCTION TO MEMORY CARD EXPLORER</u></b>	<b>4</b>
<u>1.1 GUI</u>	4
<u>1.2 DLL</u>	4
<u>1.3 PRODUCT VARIANTS</u>	5
<u>1.3.1 MCE (Full product)</u>	5
<u>1.3.2 MCE (for 95/98/Mm)</u>	5
<u>1.3.3 MCE Demo (Demonstration Version)</u>	5
<u>1.3.4 MCE DLL Engine</u>	5
<u>1.3.5 MCE Customisation Kit</u>	6
<b><u>2.0 LICENSING</u></b>	<b>7</b>
<b><u>3.0 WHAT MCE CAN DO</u></b>	<b>9</b>
<b><u>4.0 USER'S GUIDE</u></b>	<b>11</b>
<u>4.1 MENUS</u>	11
<u>4.1.1 File</u>	11
<u>4.1.2 Setup</u>	12
<u>4.1.3 Operations</u>	14
<u>4.1.4 Help</u>	16
<u>4.2 MCE MAIN PANEL</u>	17
<u>4.3 PROGRESS / MESSAGE REPORTING</u>	18
<u>4.4 COMMAND LINE PARAMETERS</u>	18
<b><u>5.0 PROGRAMMER'S GUIDE</u></b>	<b>20</b>
<u>5.1 LIST OF FILES REQUIRED</u>	21
<u>5.2 DLL API</u>	22
<u>5.2.1 MCE_LibIssue</u>	23
<u>5.2.2 MCE_EvaluateCard</u>	23
<u>5.2.3 MCE_OpWithFile</u>	24
<u>5.2.4 MCE_GetChkSum</u>	29
<u>5.2.5 MCE_GetCardDetail</u>	29
<u>5.2.6 MCE_GetCardEraseDetail</u>	30
<u>5.2.7 MCE_SetCardDetail</u>	30
<u>5.2.8 MCE_Inserted</u>	31
<u>5.2.9 MCE_NextSlot</u>	31
<u>5.2.10 MCE_InitCardSlot</u>	32

<a href="#"><u>5.2.11 MCE_CheckWindow</u></a> .....	32
<a href="#"><u>5.2.12 MCE_Restore</u></a> .....	34
<a href="#"><u>5.2.13 MCE_Customisations</u></a> .....	34
<a href="#"><u>5.2.14 MCE_Callbacks</u></a> .....	35
<a href="#"><u>5.2.15 MCE_SetAddresses</u></a> .....	35
<a href="#"><u>5.2.16 MCE_SetVCCControl</u></a> .....	36
<a href="#"><u>5.2.17 MCE_IOCCardTest</u></a> .....	36
<a href="#"><u>5.2.18 MCE_Status</u></a> .....	37
<b><a href="#"><u>6.0 INSTALLATION ISSUES</u></a></b> .....	<b>38</b>
<a href="#"><u>6.1 UPPER MEMORY BLOCK</u></a> .....	38
<a href="#"><u>6.2 WINDOWS95/98 “NEW HARDWARE” PROMPT</u></a> .....	38
<a href="#"><u>6.3 MULTI-TASKING</u></a> .....	39
<a href="#"><u>6.4 ATA</u></a> .....	39
<a href="#"><u>6.5 WINDOWS NT4</u></a> .....	40

## **Disclaimer**

This document has been carefully prepared and checked. No responsibility can be assumed for inaccuracies. Elan reserves the right to make changes without prior notice to any products herein to improve functionality, reliability or other design aspects. Elan does not assume any liability out of the use of any product described herein; neither does it convey any licence under its patent rights not the rights of others. Elan products are not authorised for use as components in life support services or systems. Elan should be informed of any such intended use to determine suitability of the products.

Source code supplied with Elan PC-Cards is provided “as-is” with no warranty, express or implied, as to its quality or fitness for a particular purpose. Elan assume no liability for any direct or indirect losses arising from use of the supplied code.

All trademarks are duly acknowledged.

## **1.0 INTRODUCTION TO MEMORY CARD EXPLORER**

Memory Card Explorer (MCE for short) is a 32-bit software package that is designed to provide access to Memory PCMCIA cards (“PC-Cards”) under Windows95/98 in a hardware independent way; that is MCE can work with any type of PCMCIA socket controller. Windows NT4 is also supported for “365” compatible socket controllers.

MCE can be thought of as two parts:

1. GUI “front end” (MCE.EXE)
2. DLL “engine” (MCELIB.DLL)

### **1.1 GUI**

The GUI is structured to make it easy to read/write data on a PC-Card to/from a disk file. The GUI also offers features that allow cards to be erased, blank-checked, compared etc. The GUI also allows the capability of the DLL engine to be demonstrated to potential OEM customers who may wish to use only the DLL engine and to create their own GUI or application.

### **1.2 DLL**

The DLL is where all the “clever” bits go on. It can be thought of as the “engine” of MCE. All accesses to the card and system hardware are performed from inside this DLL.

This guide will describe the functions of the DLL so that an OEM developer who wants to licence the DLL separately can become familiar with the API and call structures.

## **1.3 Product Variants**

### **1.3.1 MCE (Full product)**

This program offers the ability to read/write to PCMCIA cards. The variant of MCELIB.DLL provided with this product will not work with any GUI other than MCE.EXE.

Platforms: Windows 95, 98, Millennium, NT4 and 2000.

### **1.3.2 MCE (for 95/98/Mm)**

This program offers the same functionality as 1.3.1.  
Platforms: Windows 95, 98, Millennium.

### **1.3.3 MCE Demo (Demonstration Version)**

This program offers the ability to read/write to PCMCIA cards but the lifetime of the product is restricted to 30 days operation. The library provided (MCELIB.DLL) is also restricted to 30 days operation and can be used by developers GUI's during this evaluation period.

Platforms: Windows 95, 98, Millennium, NT4 and 2000.

### **1.3.4 MCE DLL Engine**

This is the library of functions that MCE uses to analyse cards and perform read/write functions. This product may be bought under licence and embedded into a user's application. This variation of MCELIB has no time restrictions imposed and it has been allowed to work with any user GUI.

Platforms: Windows 95, 98, Millennium, NT4 and 2000.

### **1.3.5 MCE Customisation Kit**

A version of MCE called the Customisation Kit may also be purchased by the user. This kit allows a software developer to make adjustments to the visual presentation of the GUI.

Control of the following is provided:

- \* Titles
- \* Progress bar / on off indication
- \* Desktop buttons size/shape/position
- \* Bitmap design in applications window area
- \* Toolbar icons
- \* Menu text content
- \* Dialog text/visual layout
- \* Help text

Provision for building full and demo variants is provided.

Please note that once the development phase is complete the developer will have to purchase a number of MCE DLL Engine licenses to provide with the customised GUI.

Platforms: Windows 95, 98, Millennium, NT4 and 2000.

User Requirements:

C++ programming skills, Borland Turbo C++ V5.02

A full Microsoft Visual Basic 6.0 GUI example is also provided.

## 2.0 LICENSING

MCE can be purchased in three ways:

- i) as a fully functional Windows package (i.e. the GUI & DLL)
- ii) the DLL component (special unprotected version) bought under license for embedding\*
- iii) customised by Elan to your exact requirement.

The cost of option ii) is substantially lower than option i) as we aim to make it affordable for you to use the DLL in your own products without making them non-competitive.

Elan also manufacture PC-Card Reader/Writers for both ISA and PCI buses. For volume requirements “bundle” prices are available for MCE/MCELIB when ordered with hardware.

Please contact Elan or your local representative for a pricing structure for the two schemes.

*See file MCELA.doc for licence agreement details.*

### **\* IMPORTANT NOTE**

Developers wishing to use the MCE library functions in their own application will need to purchase licenses from Elan. The licensed library file is supplied on the disk marked “*MCE DLL ENGINE*”. If you have not received this disk then you should check to ensure you have purchased the correct product from Elan.

The library supplied with the full MCE product is a protected version. It has not been sold to embed in your own application and we have prevented this possibility.

Purchasers of DLL licenses must also purchase the full MCE product, so care must be taken not to get the protected and unprotected versions of MCELIB.dll mixed up (the library will report with the appropriate error message if this has happened).

The full MCE product is provided so that any problems encountered on a development platform can quickly be isolated into one of two categories :

1. A set-up issue or problem with the library.
2. A problem with the developers use of the library.

The free demo version of MCE includes a 30 day trial version which has no restrictions other than this time limit. It can be linked to your own trial application for testing purposes.



### 3.0 WHAT MCE CAN DO

The following summary shows what functions MCE can perform:

- **Read the whole or part of the card's data to disk file**
- **Write data from a disk file to part or all of the card**
- **Erase all or part of the card's data**
- **Check card holds blank data (FFH) over the set address**
- **Compare card data to disk file data over the set address**
- **Checksum card data over set address (16 bit additive)**
- **Automatically detect card type and size**
- **Card Explore feature to view card data "real-time"**
- **File Explore (view) and edit**

To keep the DLL interface as simple as possible MCE always works with disk files where data is concerned. Operations may be performed on the whole card or part of it. The card addressing capability is restricted to block sizes of typically 64K or 128K bytes depending on card type.

The technologies that MCE can handle are listed below.

- **SRAM**
- **Flash Series I (Intel based or equivalent)**
- **Flash Series II / 2+ (Intel based or equivalent)**
- **Flash Multi-Level Cell (MLC)**
- **ATA Flash or Rotating**
- **AMD C,D & E Series (AMD based or equivalent)**

***This covers over 98% of all known PCMCIA Memory cards !***

Additionally, MCE can handle and detect cards that are organised as 8-bit only or 16-bit only. Normally cards follow the PCMCIA “standard” that defines both 8-and-16-bit access modes (needless to say MCE handles these too).

Finally, just in case the auto-detection process gets it wrong or you have a card that is unusual, there is a “manual” selection route that allows complete control of the card’s parameters.

## **4.0 USER'S GUIDE**

This section describes how to use MCE to read and write to PC-Cards and it describes the other facilities that are available.

### **4.1 Menus**

The following drop down menu groups are available from the menu bar, full details of each option is given in the mcehelp file:

#### **4.1.1 File**

##### **File Select**

Select an existing file or create a new file to use with MCE operations. This function has a button on the toolbar.

##### **Generate Test File**

Produce test data in the currently selected file. The length of the file will be determined by the size of the card in the current slot or that allowed by the currently set address range (see 4.1.2).

To view the test file's contents go to Operations/ Explore(file).

##### **Exit**

Quit the MCE program. This will not be allowed if an operation is in progress.

### 4.1.2 Setup

#### Memory Bank

Choose between *Common* and *Attribute* memory banks.

Common memory can usually be read in words (16 bit) or bytes (8 bit) although some cards are specifically 8 bit or 16 bit only. MCE automatically detects this during card evaluation.

Attribute memory is byte wide and found on even byte locations only.

Notes:

1. Common memory is the data storage area that most users will use to read/save data. Attribute memory is used primarily by the Operating System to obtain information about the card's operational characteristics, size etc. This information area is called the Card Information Structure or CIS. Some CIS information may be viewed from the *Card Information* button on the toolbar.
2. Not all linear memory cards have a CIS and not all have attribute memory. I/O Cards and ATA Cards usually have attribute memory with a CIS structure but rarely have common memory.

#### Addresses

These allow reading and writing operations to be performed on part of a card. This is determined by submitting card start and end addresses, and removing the check in box *Perform operations on the whole card*.

#### Card Detection

Select between automatic card detection and manual card selection.

The later should be used only if the automatic detection has failed.

Note: Users cannot manually select ATA cards.

#### Slot

Go to the next slot and evaluate any card found in it.

The skipping of certain slots can be set up by a command line parameter...see 4.4. MCE can also be forced to always start at a given slot number. This function has a button on the toolbar.

#### Vcc Power Levels

Control the Vcc voltage supplied to the card. Selections are Automatic, 3.3V and 5V. Note that providing 3.3V may not be possible for the read/write adapter in use. MCE will set the selection to the preferred option for the hardware. Warnings are given when the voltage to be applied does not match that indicated by the card under evaluation.

## **Other Options**

This dialog is a collection of various customisation features, one group of which provides various IO and ATA evaluation options.

### 4.1.3 Operations

All operations will operate on:

1. the card data between the start and end addresses as set. The default range is the whole card (see 4.1.2).
2. either Common or Attribute memory as selected (see 4.1.2).

#### **Read**

Data read from the card and written to the currently selected file.

#### **Write**

Data read from the currently selected file and written into the card (unused locations left erased).

#### **Compare**

Compare data from the currently selected file and the card data.

#### **Blank**

Check card is in the erased state (0xFF data).

#### **Erase**

Erase the card data leaving it in the erased state (0xFF data).

#### **Checksum**

Compute a 16-bit additive checksum of the card's data.

#### **Format Card**

Use this to prepare SRAM/ATA card with DOS format ready for system file use. (ATA-Cards up to 32M). Using formatted SRAM cards in Windows : Look-up *SRAM* in Windows Help for a guide to setting up removable drives.

## **ATA Initialise**

Use this to prepare ATA cards of any size ready for formatting in Windows by writing a partition table at the start of the drive (in Windows Explorer the initialised drive should appear with a drive letter assignment and can be formatted by right clicking the mouse on it and proceeding as directed).

## **Abort**

Abort any current card operation in progress. Abort has a hot button on the MCE front panel . Abort does NOT undo any already modified data.

## **Card Information**

This dialog shows card information taken from the CIS. Information shown includes card description & manufacturer details. Note that not all cards have CIS fields setup. Additionally information about the card's ID or, in the case of ATA cards, drive parameters are given. Card Information has a button on the toolbar.

## **Explore (Card Memory Viewer)**

Allows the user to browse the card's data in Common or Attribute memory banks in "real time".

For the attribute memory selection only the valid even bytes are shown. **Warning :** This feature is not restricted to the card size and will read "beyond" common or attribute physical boundaries.

This feature has a button on the MCE main panel called *Explore*.

## **Explore (Current File Data Viewer)**

This utility allows data files to be viewed and edited in hex. This feature has a button on the toolbar.

#### **4.1.4 Help**

##### **Help Reference**

You can view the help reference manually by locating MCEHELP.HLP and double clicking it from Windows Explorer.

The reference can also be called up from the help key for any MCE dialog and the help displayed will relate to that context .

Pressing the ? button on the toolbar will also bring up the help reference.

##### **Help About**

Get MCE's date/issue and support information. Pressing the Elan logo will also activate this dialog.

##### **Help About Library**

Get information about the issue of MCELIB.DLL in use.

For MCEDEMO this information also includes the number of trial days remaining.

##### **Status Help**

Any additional help that is available for a given status response.

A button for this is provided on the status line. From the menu the Index of status help messages found in the help file may be viewed.



## 4.2 MCE Main Panel

Buttons are available on the MCE main panel to perform the most commonly used operations.

Most of these are associated with performing operations on cards.

Additionally users may drag files from applications like Windows Explorer straight into areas of the MCE screen. This technique can be used to update the file selection or perform a PC-Card operation using that file.

Figure 4.2-1 shows the main panel.

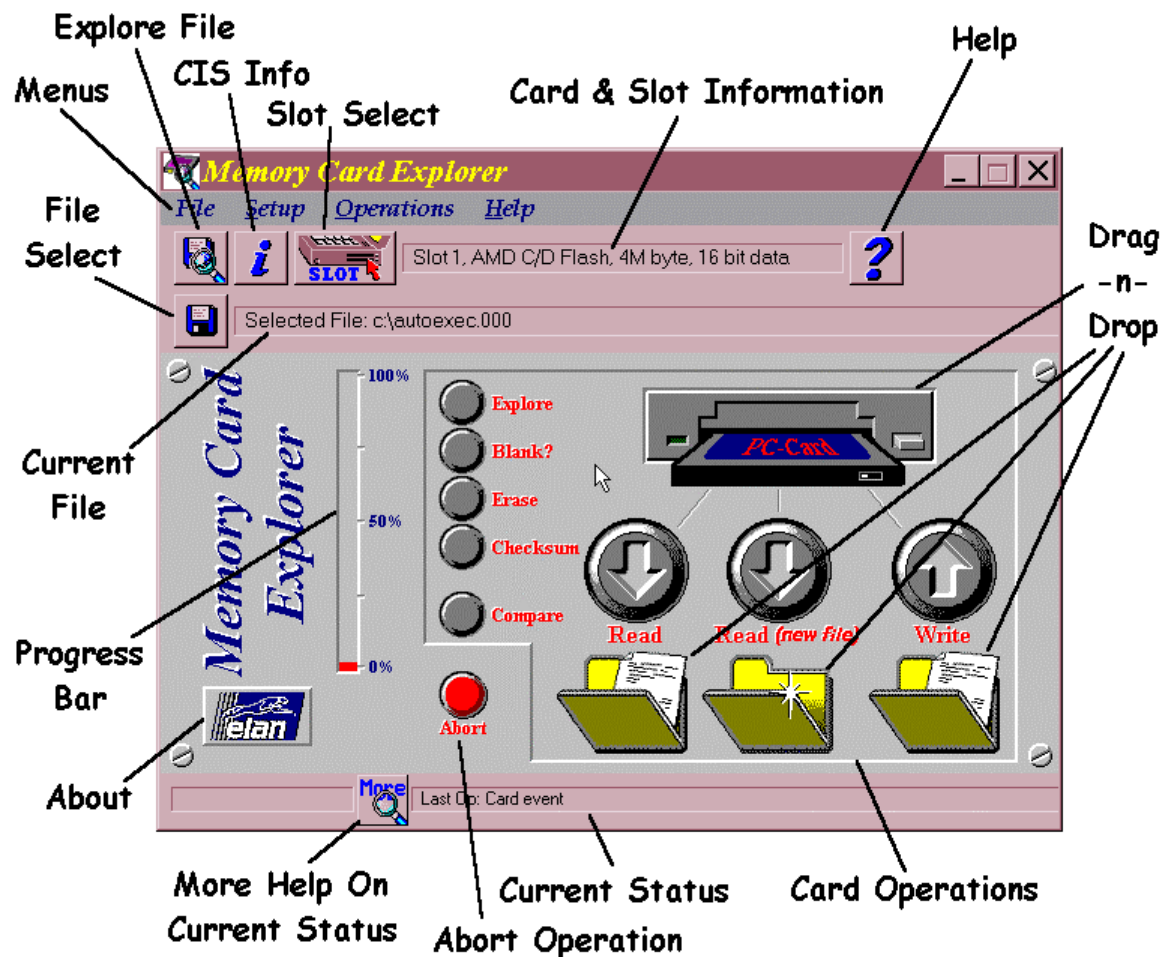


Figure 4.2-1 MCE Main Panel

### 4.3 Progress / Message reporting

The progress of a card operation and any result is reported on the status line. A progress bar also shows the % complete.

During an operation the picture of the card slot will illuminate its LED: red for card writes and green for card reads.

### 4.4 Command Line Parameters

The following command line parameters can be used to invoke special features of MCE.

#### **“iXXX”**

XXX=base I/O address in hex for ATA operations e.g. “x2E0”. To disable I/O use “x000”. This option is written to a configuration file and used for subsequent MCE sessions (so you actually only need to enter this command line once) . Please note that the address 2e0 is default setting and this can now be enabled from the menu selection *Options / Other Features*.

#### **“sN”**

N=MCE starting slot e.g. “s3” will start MCE on slot 3. N=1 to 8. Default is 1. For example, this is useful if you always have a modem card in slot 1 and so want to start at slot 2 for memory cards (slot 1 will still be selectable using the “Slot” button).

#### **“wXX”**

XX=base memory address for card operations e.g. “wD8” will use D8000h as a base address. XX can be C0 to EF for most PCs. Default is D0. The 4K region from address chosen must be “free”.

#### **“xBBBBBBBB”**

BBBBBBBBB=slot exclude binary code with slot 1 rightmost and slot 8 leftmost. This code will let you always exclude which slots the “Slot” button or MCE\_NextSlot() function will cycle round. A ‘1’ denotes an excluded position. e.g. “x00001011” will never cycle to slot positions 1, 2 or 4. Default is all ‘0’s (only those slots physically present can be excluded).

**“d”**

Put into debug mode. Files mceerror.txt & skterror.txt are saved in the root directory of the drive that mce is installed to. These files may provide Elan with usefull debug information should problems arise.

## 5.0 PROGRAMMER'S GUIDE

The API for the DLL is described in this section on a function by function basis. To help ensure that the guide does not go out of date, only the function name and the purpose of the routine will be described. **Please refer to *MCELIB.H* for the complete function prototype and parameter information.**

### NOTE

*It is assumed that you are familiar with Windows and the C++ language.*

## 5.1 List Of Files Required

* PCCGO32.DLL	32-bit DLL for Win95/98/Mm/NT4/2000
* PCCARDGO.VxD	32-bit Kernel driver for Win95/98/Mm
* PCCGOCLS.SYS	32-bit Kernel driver for WinNT4/2000
* MCELIB.DLL	Main MCE “engine” 32-bit DLL
MCELIB.H	Main MCE “engine” DLL header file

To use the MCE DLL, you need to “link” MCELIB.DLL to your project and use the “.H” file to provide function prototypes for the API. For Win95/98/Mm the VxD file must be located in your `\windows\system` folder and for NT4/2000 the SYS file must be in `\winnt\system32\drivers` (this should be done by the installation process). When you buy a licence to use the MCE DLL you get the right to use only the files listed above.

In NT/2000 applications the driver `pccgocls.sys` should be registered in the

Windows Registry. For an MCE installation the registration key may be found under the following path:

`\HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\pccgocls`

Use the information contained here to assist your own installation of this driver.

When MCELIB.DLL is invoked, it loads PCCGO32.DLL which in-turn loads the correct kernel driver(s) so there is no need for you to do any of this manually.

Linking the MCELIB.DLL file to your project is normally done by “implib” (Borland) or a similar utility (“lib” for Microsoft). Some compilers also allow you to place the DLL straight into your project file so that the import “.LIB” file gets created on-the-fly. Each compiler is slightly different so you will need to figure this out yourself for your particular installation.

*\* A disk called “MCE DLL Engine” contains the files marked \*, which subject to license agreement are the files that will need to be supplied as part of the developers software product to the end-user.*

## 5.2 DLL API

Functions	Use summary
-----------	-------------

### One-off Initialisation Functions:

MCE_CheckWindow	-95/98/: <b>optional initialisation</b> PC Memory window check and Default PC Window address override -NT4: <b>required initialisation</b> of PC window
MCE_Customisations	- <b>optional initialisation</b> features

### General Functions:

MCE_LibIssue	- <b>information</b>
MCE_NextSlot	-MCELIB's <b>slot selection</b> mechanism
MCE_Inserted	-card insertion check <b>slot operation</b>
MCE_Status	-card status check <b>card information</b>
MCE_InitCardSlot	- <b>slot initialisation</b>
MCE_IOCardTest	-evaluate card looking for IO card types
MCE_EvaluateCard	- <b>slot initialisation</b> + card auto evaluation
MCE_OpWithFile	-for all <b>card operations</b>
MCE_SetCardDetail	-manual <b>card initialisation</b>
MCE_GetChkSum	- <b>card information</b> from last operation
MCE_GetCardDetail	- <b>card information</b> from current card
MCE_GetCardEraseDetail	- <b>card information</b> (current card)
MCE_Callbacks	-card notification event reporting function
MCE_SetAddresses	-card operation <b>initialisation</b>
MCE_SetVCCControl	-card Vcc control ( <b>initialisation</b> )

### Exit Functions:

MCE_Restore	-tidy-up procedure Recommended prior to user program exit
-------------	--

### Notes:

- A “slot” is a PCMCIA socket. Slots are numbered from 0 for the API calls.
- A card’s “technology” refers to its type i.e. SRAM,Flash Series I/II/2+ etc.

### 5.2.1 MCE\_LibIssue

This function returns the issue of the MCELIB.DLL module. This may be useful for you to track our revisions. The string format can be found in the .H file.

For the demonstration version of the library a message giving the number of trial days remaining is appended to the end of the issue string.

### 5.2.2 MCE\_EvaluateCard

This function allows automatic detection of the type of card in a particular slot.

No pre-conditions exist with this function, it is self contained.

A series of read/write/read tests are performed on the card during this test to determine:

1. the technology (SRAM, Flash etc)
2. the card's size in bytes
3. the data width (8 or 16 bit)
4. the size of each logical device in the card
5. the size of an erasable block inside the logical devices.

Due to the reliance on write operations, the state of the battery in SRAM cards and the state of the WriteProtect switch on all cards will affect this test.

This function deposits its results inside the DLL for later use.

Use the MCE\_IOCardTest function to avoid upsetting IO card types prior to MCE\_EvaluateCard (windows 95/98 only). Proceed with MCE\_EvaluateCard only if the MCE\_IOCardTest returns the fact that the card is a memory type.

*See also: MCE\_GetCardDetail, MCE\_GetCardEraseDetail, MCE\_SetCardDetail.*

### 5.2.3 MCE\_OpWithFile

This function performs all the read/write operations on the card detected via MCE\_EvaluateCard (or manually selected via MCE\_SetCardDetail). A parameter selects common or attribute memory.

All data is exchanged between the card and a disk file.

If the operation involves reading to a disk file and the file already exists, it will get overwritten (no warnings) and if the file does not exist it will be created.

If the operation is a read from disk file and the card will be overwritten without warning. If the file does not exist an error code will be returned.

All disk file “user dialogues” relating to file creation, overwriting etc. must be handled by your own s/w.

The following diagram shows a typical flow for “auto” mode to process the whole card.

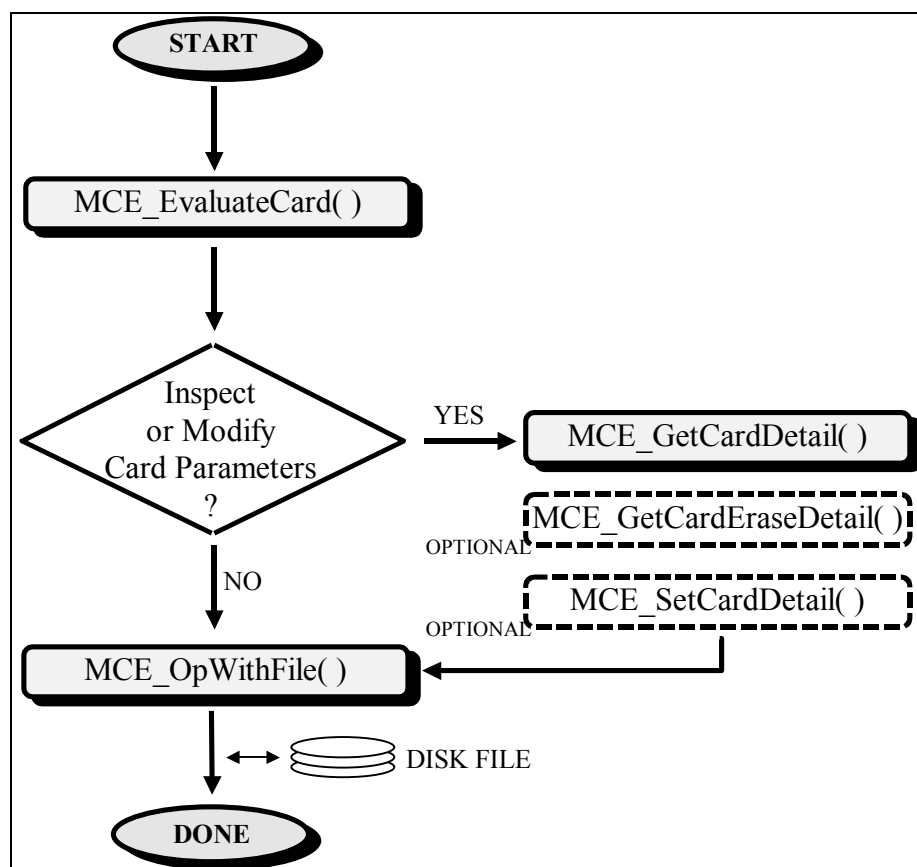


Figure 5.2.3-1 Typical Operation Sequence For Auto Card Evaluation



Operations on cards normally process the whole card. The use of `MCE_SetAddresses()` will allow part card programming (see sect. 5.2.15).

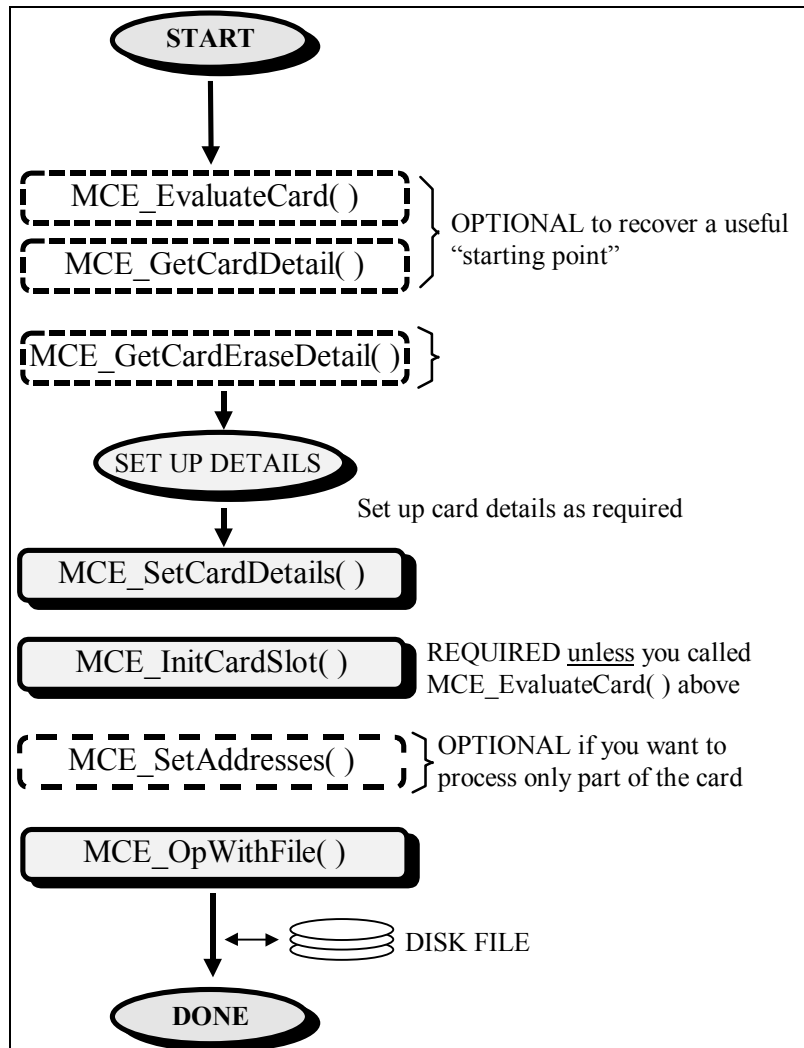


Figure 5.2.3-2 Typical Operation Sequence For Manual Card Configuration

- ***Note that a write operation will internally perform [erase-write] on the whole card or set address range automatically.***
- ***This function relies on either `MCE_EvaluateCard()` or `MCE_InitCardSlot()` to initialise the PCMCIA slot.***

During the operation, a progress “callback” is provided to enable you to inform the user what % is complete. For a write operation, there will be a single 0-100% notification for [erase-write] process. NB: do not rely on the % going to exactly 100. The string passed to the callback is formatted like this (example) “Reading... 01 %”. There is a space between the last digit and the % so that you can use `sscanf(s, "%*s%d", &pct)` to extract the numerical percentage.

There must be a pointer to a callback function passed in to the `MCE_OpWithFile( )` function and a `char*` to allow an abort mechanism<sup>1</sup>.

Because of the high processor effort required to program a PCMCIA card, the callback alone is not sufficient to display a % complete. The reason is that all parts of the program are part of the same “thread” (processor task). Because one part of the thread is “hogging” processor time there will never be an opportunity for the display part to update itself. The net effect is that you will never see any progress updates.

The solution to this problem is to launch a new thread which then calls `MCE_OpWithFile( )`. The new thread gets allocated its own virtual process and so will still allow other threads to get the opportunity to perform useful tasks (at the expense of the new thread that is busy operating on the PCMCIA card). Note that in order to affect this “sharing” of processor time, the DLL internally calls a “yield” function every so often to ensure that the Windows kernel can properly task-switch.

The following code extract shows how this is done (this code has been tested using Borland C++ 5.0...don’t forget to implib the DLL and then place the .LIB file in the project):

---

<sup>1</sup> . If you do not want to use callbacks you can pass in NULL for both pointers. Note that if you elect not to have callbacks then you cannot have the abort mechanism either (the two are related).

```

#include <windows.h>
#include <process.h>
#include <string.h>
#include <sdtio.h>
#include "mcelib.h"

char AbortNow = FALSE;
BYTE Slot = 0;
BYTE AttributeSpace = FALSE; //work with common memory only
char FileName[200] = "My Card Data.bin"; //data file name
int UpdateProgressFlag = FALSE;
int OperationFinished = FALSE; //use to track when operation is done
int Pct = 0; //progress %
//-----
int FAR PASCAL _exportMCE ProgressCallback(LPSTR PercentStr)
{
    sscanf(PercentStr,"%*s%d",&Pct); //throw away the "Reading..." part (%*s)

    UpdateProgressFlag = TRUE; //force main loop to paint progress bar

    if (WantToStopTheOperation)
        AbortNow = TRUE;

    return FALSE; //always return FALSE...no error
}
//-----
void ThreadCardRead(PVOID ptr)
{
    char s[200];
    int status;
    DWORD spare;
    strcpy(s,FileName);
    //could post a message to say "operation starting" here
    status=MCE_OpWithFile(Slot, //PCMCIA slot no.
        AttributeSpace, //select common/attribute
        COP_READ, //what to do, could be a global var
        s, //in:file out:status
        (WORD)NULL, //not used
        (FPSTRCB)ProgressCallback, //callback fn.
        (char far *)&AbortNow, //abort flag
        spare); //spare parameter
    //could post a message to say "operation ending" here
    OperationFinished = TRUE; //let WinMain finish now
    _endthread(); //terminate once operation completed
}

```

```

//-----
int PASCAL WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
    LPSTR lpCmdLine, int nCmdShow) //entry point
{
    char msg[100];
    WORD status;
    int result;
    CARDTYPE cardtype;
    DWORD cardsize,devsize,erasesize,chipid;
    BYTE datawidth;
    BYTE attribute;
    DWORD spare;

    //note : MCE_CheckWindow() is required here for NT apps
    status = MCE_EvaluateCard(Slot,msg,&result);
    status = MCE_GetCardDetail(attribute,
        &cardtype, //use to check evaluation result
        &cardsize,
        &datawidth,
        &devsize,
        &erasesize,
        &chipid,
        &spare); //spare parameter field
    status = _beginthread(ThreadCardRead,0,NULL); //NB: returns -1 if fail

    do //wait for new thread to finish its task
    {
    }
    while (!OperationFinished);}
//-----

```

Clearly the above example is quite simple but can be easily enhanced to cover all types of scenarios. Due to the thread launch syntax, all parameters that the thread “needs” must be globally available.

Note that the global “abort” flag is only inspected every time the callback is invoked so this is your chance to terminate the operation prematurely if so desired.

*See also: MCE\_EvaluateCard, MCE\_GetCardDetail, MCE\_GetCardEraseDetail, MCE\_SetCardDetail, MCE\_InitCardSlot.*

#### 5.2.4 MCE\_GetChkSum

This function recovers any checksum that was computed by a previous read operation (or one that included a read or compare operation) using the MCE\_OpWithFile( ) function. The checksum is held inside the DLL.

Any operation pre-clears the checksum to 0000h.  
The checksum is a 16-bit additive type.

#### 5.2.5 MCE\_GetCardDetail

This function recovers the card parameters found by a previous call to MCE\_EvaluateCard( ). The parameters are stored internally by the DLL.

The parameters are used by MCE\_OpWithFile( ) to enable it to read/write/erase the card in the correct manner.

Attribute memory sizing is also performed by this function if the attribute flag is set. A size of 0 indicates a failure to size. This can be due to a number of reasons the main ones being that attribute memory does not exist or is not of the writeable variety. For this case the user will need to manually select attribute size using MCE\_SetCardDetail.

*See also: MCE\_GetCardEraseDetail, MCE\_SetCardDetail.*

### 5.2.6 MCE\_GetCardEraseDetail

This function enquires what erase geometry is required for a given hypothetical card technology and data width.

The function is used to complete the set of essential parameters to be passed to MCE\_SetCardDetail( ) when you only know the card's technology and data width. Normally a 16-bit data width should be used unless you have known 8-bit only card (example Intel Value100 Series)

The function is used only with manual card configuration.

*See also: MCE\_GetCardDetail, MCE\_SetCardDetail.*

### 5.2.7 MCE\_SetCardDetail

This function allows the set of essential card parameters to be passed manually to the DLL ready for use by subsequent calls to MCE\_OpWithFile( ).

Note: You cannot manually select an ATA card type. Use MCE\_EvaluateCard() to auto identify it.

*See also: MCE\_GetCardDetail, MCE\_GetCardEraseDetail.*

### **5.2.8 MCE\_Inserted**

This function checks to see if there is a card in the requested slot.

It does not tell you what type of card it is.

The function is self contained and has no pre-conditions.

The function is quite fast and can be used to “poll” slots for card insertion and extraction events on-the-fly (perhaps every 500ms or so).

### **5.2.9 MCE\_NextSlot**

This function will report whether the “next” logical slot is available or not.

The function is useful to implement a “next slot” button in the GUI without needing to know how many slots are physically installed in the computer.

When the “next” slot does not exist, the function returns 0.

If the “next” slot does exist the function will return its number.

The function does not “do” anything to the hardware or to the internal state of the DLL.

If you wish not to look at certain slots then see function MCE\_Customisations.

### 5.2.10 MCE\_InitCardSlot

This function configures the requested slot ready for use by MCE\_OpWithFile( ) when MCE\_EvaluateCard( ) has NOT been used to detect the card type. See Figure 5.2.3-2.

MCE\_EvaluateCard( ) performs an internal slot initialisation (housekeeping) so MCE\_InitCardSlot( ) is not needed for automatic card evaluation; it is only needed for manual card configuration.

### 5.2.11 MCE\_CheckWindow

#### Windows 95/98

This function does a quick check on the upper memory window used to access the PCMCIA card.

It scans the whole region (4KBytes) for all FFH data and checks that it can't alter the data at key addresses.

The function can be used to warn the end user that the window may not be free for use and that he may need to alter his computer's configuration to free the window from some other application.

It cannot categorically decide if the window is free to use.

This function is also used as means to override the default P.C. memory window to be used. Pass in 0 for the default memory window (D0000H) or the value of the window address to be tested. Pass or Fail the window address given is the one that the library will try and use.

#### Windows NT4

For NT4 this function should be performed prior to any other. It serves three purposes:



1. It registers the PC Memory window to be used so that no other Windows utility can access it. If this function returns a failed response then there is potentially a resource conflict and an alternative window may need to be found. This can be done by repeating the process with an alternative memory window address to try. Failing this, PC resources may need to be re-assigned.

(Valid addresses = D0000-EC000H inclusive in 4000H steps e.g. D8000.)

From issue 3-08 the PC memory window can be set to any 16K of free PC addressable space. Users can use the traditional D0000H – EC000H range or try an address above the PC's DRAM memory.

e.g. for a PC fitted with 256 Mbytes of memory then set a value greater than 10000000H say 100D0000H.

2. The process of registration also sets-up the MCELIB library with an internal memory 'handle' it can use. This is required for all subsequent card access type operations. e.g. those found in functions MCE\_EvaluateC and MCE\_OpWithFile etc.

3. It finally proceeds to do the memory check described in the Windows 95/98 section expecting to find FFH data that can not be overwritten.

### 5.2.12 MCE\_Restore

This function does a “clean-up” after library operations are finished. Its purpose is to leave the PCMCIA card controller(s) in the same state as found on start up. For example, MCE\_InitCardSlot/MCE\_EvaluateCard functions disable card insertion events (Thus quietening Card Services). Performing MCE\_Restore once finished will restore this.

### 5.2.13 MCE\_Customisations

This function has been added as a mechanism to pass in custom features/data from the calling programme to the DLL.

Information passed:

- \* An I/O port start address (that is to be used for ATA card control). see section 6.4 ATA.
- \* A socket exclusion data mask that will allow the function MCE\_NextSlot() to skip the defined socket positions.
- \* Diagnostic mode enable.
- \* Vcc/Vpp power hold control enable.

If this function is not used as part of the initialisation sequence then the these “Customisations” are left in their default state:

- \* ATA auto-detection is not enabled.
- \* All Sockets examined by MCE\_NextSlot().
- \* Diagnostic mode OFF.
- \* Power hold control OFF.

Refer to the file mcelib.h for the most up to date and more detailed information.

#### **5.2.14 MCE\_Callbacks**

This function provides a mechanism for building card notification events into the users calling programme.

This function is called upon initialisation to register for card events. It should be called at the end of the users programme to De-register. During registration an additional parameter is passed which is the calling applications handle. Card notification events can then be posted to this “address”.

#### **5.2.15 MCE\_SetAddresses**

This function allows start and stop card address values to be set-up ready in the library prior to calling the MCE\_OpWithFile() function. These addresses are reset at the end of MCE\_OpWithFile(). When the values are reset operations are performed on the whole card.

The addresses permitted are determined arbitrarily by the library depending on the card inserted in the current slot. e.g. an SRAM card has 64K boundaries.

The function will supply the nearest start/end addresses it can set to the ones requested. A status indication is returned for the following outcomes:

- \* Address(es) changed
- \* Addresses OK
- \* No card (so no address evaluation)
- \* Card not identified (so no address evaluation)

Byte swapping of odd and even bytes may be set up in this function.

### 5.2.16 MCE\_SetVCCControl

This function allows control over the Vcc voltage level applied to the card. This is conditional on :

- \* the card read/write hardware being able to provide low voltage (3.3V) to the card.
  - \* the card read/write hardware having the ability to read voltage sense lines supplied from the card indicating the voltage it requires.
- These features are usually found together.

The function returns an error if the voltage requested cannot be provided. It returns a warning if the voltage requested does not match the voltage sense requirement of the card.

#### Notes

- \* Some cards do not provide correct voltage sense levels. To read or write successfully the voltage provided may need to be forced to the appropriate value.
- \* Setting this function does not directly apply Vcc. This is applied later by other functions.

### 5.2.17 MCE\_IOCCardTest

This function allows determination of the card type plugged into the specified socket. The function applies to Windows 95/98 operating systems and not Windows NT4.

The test analyses CIS information in card attribute memory to determine if the card is of type IO or Memory. It returns TRUE for IO and FALSE for Memory. Additionally it will set the CARDTYPE variable to one of the following: CT\_IO ,CT\_ATA, CT\_UNK. Where CT\_ATA is for a specific IO card type and CT\_IO is for any other IO card (e.g. modem or network). CT\_UNK will describe any memory card type (not as yet fully categorised).

This test has been added to prevent problems with MCE interrogating IO Cards inorder to identify the memory card type, and creating system problems in the process.

### **5.2.18 MCE\_Status**

This function provides card status information for a particular card slot. Status information includes Write Protect and Battery Status.

## **6.0 INSTALLATION ISSUES**

### **6.1 Upper Memory Block**

The MCE DLL uses, by default, an upper memory block located at D0000h to D3FFFh (D000:0000h to D000:3FFFh in segment:offset terms). Use MCE\_CheckWindow function to change this address or use the command line parameter detailed earlier.

This region **MUST** be free for exclusive use by MCE. Please refer to the on-line help installation notes for complete details.

Remember that when embedding MCE into your own application, your instructions and installation notes to your end customer must reflect this memory block requirement.

For NT4 see special note for the function MCE\_CheckWindow.

### **6.2 Windows95/98 “New Hardware” Prompt**

If you plug a new PCMCIA memory card into Windows95/98, it will either be recognised and use one of the “generic” built in drivers (MTDs), or it will be treated as “new hardware” and a wizard dialog will appear prompting for a driver disk etc. MCE does not require any of the built in drivers for it to work, so how you respond to this wizard is up to you. The easiest approach is to select “do not install driver” and this way Windows will not keep prompting you to load a driver disk every time you plug in the card.

If Windows already has a driver configured for the card you will normally get a “happy beep” (low-tone then high-tone) when you insert the card. This too is OK for MCE. You may find these beeps are very delayed when MCE is loaded...this is normal.

### **6.3 Multi-tasking**

If you use the thread techniques shown in Section 4, then MCE can run concurrently with other applications without any problems.

### **6.4 ATA**

MCELIB needs to be informed of the I/O address range set-up for it by the calling software. Use the MCE\_Customisations function to do this. The range required is 16 bytes long starting on 16 byte boundary. e.g. passing “2E0” would tell MCELIB to use 2E0-2EF for ATA access. The I/O setting obviously must not conflict with any other hardware in use on the system.

It is the calling software’s responsibility for finding a suitable I/O address and reserving its use on the Windows 95/98 OS.

MCE will always I/O map an ATA card. It cannot use the memory mapped mode that some newer drives support.

Should the ATA card in use contain a valid DOS FAT format then great care should be taken:

1. The user should not run other Windows programmes that will access the card during MCELIB operations.
2. If the card data has been altered then it must be ejected and reinserted before the operating system will re-evaluate the card.

## 6.5 Windows NT4

Windows NT4 only contains the most basic PCMCIA support; it is not hot swappable and only supports a limited set of cards. Also the amount of user configuration that is possible is quite low.

MCE uses its own Socket Services layer to allow memory cards and ATA cards to be read and written. The support is limited to “365” compatible controllers (also known as “PCIC” compatible). You will need to determine whether the controller you are working with is 365 compatible by referring to the computer’s (or adapter’s) documentation. As a guide, the following devices are known to be OK:

Chip Manufacturer
Intel
Vadem
Texas Instruments (TI)
Ricoh
Cirrus Logic
O2Micro
Toshiba

You can check the controller type by clicking on the PC Card icon in  
<Settings><Control Panel>



PC Card  
(PCMCIA)

On start up, MCE will attempt to determine the presence of one or more such controllers (up to four 365 devices can be supported).

It is recommended that software such as Award Cardware or SystemSoft Cardwizard be disabled while you are using MCE. While most operations will work correctly, there could be some interaction with MCE and such utilities that could lead to erratic behaviour.