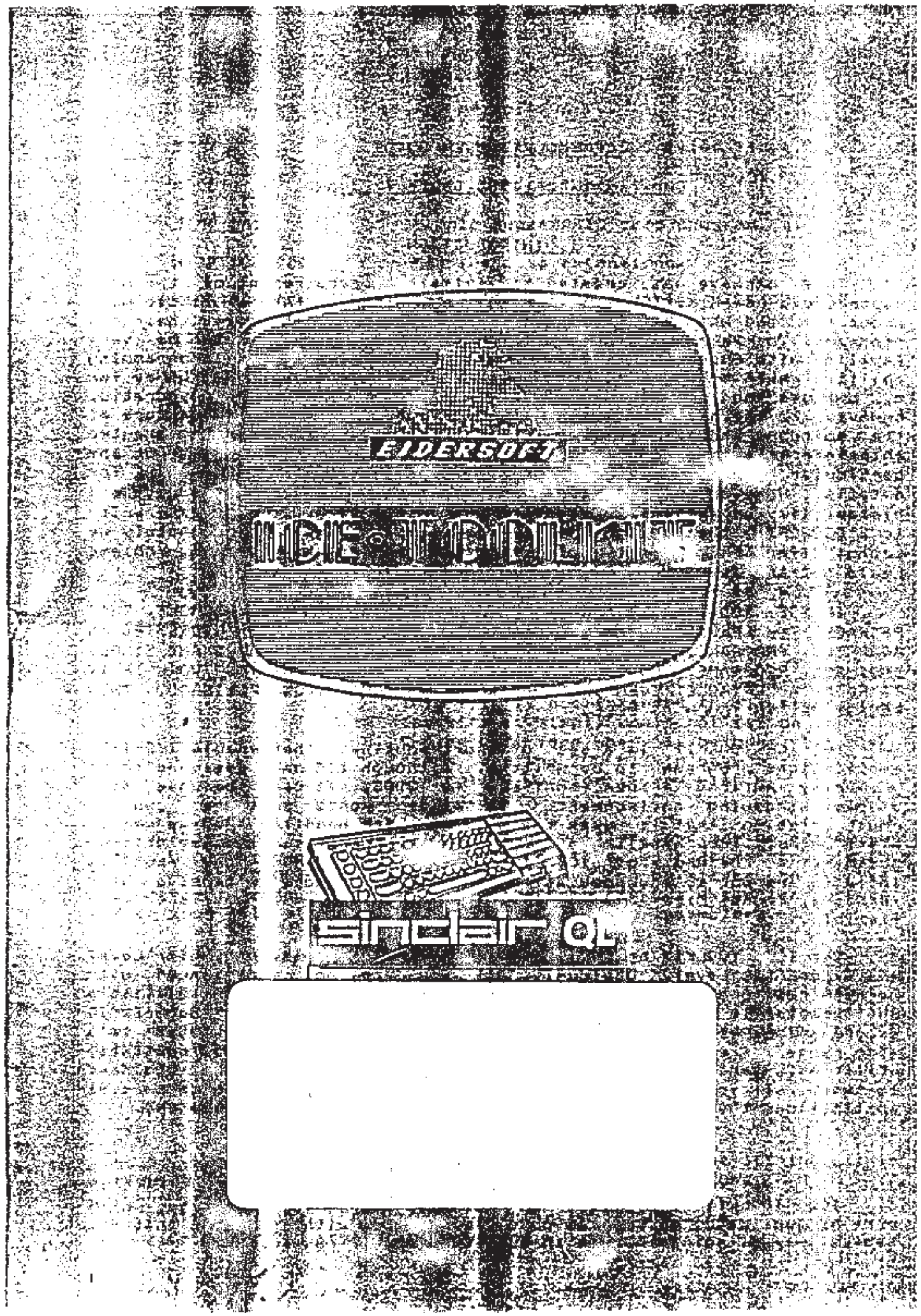
EIDERSOFT
ICE TOOLKIT
sinclair QL

ICE TOOLKIT USER GUIDE

1. INTRODUCTION

This package was created to fulfill a requirement of the more experienced users of ICE and the QL who wish to know a little more about it, and who wish to use some of its facilities in their own programs. To make full use of the ICE TOOLKIT you should be very familiar with SuperBASIC and a smattering of assembler programming ability would be an advantage. These routines were developed by me for my own use and so in some places they are a little idiosyncratic, however to the best of my knowledge they do work. Due to the nature of the software and documentation, neither EIDERSOFT nor myself guarantee this information or software to be error-free or that it is fit for any particular purpose (please see the disclaimer on the ICE USER GUIDE for details). However on a lighter note, please enjoy using the bits and pieces. If you have any suggestions about ICE TOOLKIT then please write to EIDERSOFT, the same goes for reporting any bugs - unfortunately we cannot offer help on the application of this package as such problems could get too involved.

IF YOU PRODUCE ANY USEFUL ICE SOFTWARE WITH OR WITHOUT THIS PACKAGE THEN PLEASE CONTACT EIDERSOFT. WE ARE ESPECIALLY INTERESTED IN GRAPHICS PROGRAMS, LANGUAGE LIBRARIES (BCPL, C etc.), GAMES AND SIMPLE BUSINESS PROGRAMS. YOU WILL OF COURSE BE REWARDED FOR YOUR WORK IF IT IS FOUND ACCEPTABLE. EVEN IF YOUR WORK IS NOT 100% POLISHED, THEN PLEASE GET IN TOUCH AS WELL.

The ICE TOOLKIT is a set of short routines that enable ICE-like applications programs to be written. Although ICE was designed to be primarily an icon driven front-end for QDOS, it can be used as a simple windowing environment. In other words software can be constructed so that it uses the ICE ROM for handling pop-up menus, alert boxes, icon handling and some time in the future, mouse driving software. As the whole of ICE fits into a 16k ROM, the routines included are not so advanced as those for the more expensive WIMP (Windows, Icons, Mice and Pull-down menus) systems, but are very flexible.

The ICE TOOLKIT has been written in two levels: Assembler Level and SuperBASIC Level. Obviously, if you are not experienced with machine code programming then the Assembler Level will be of little use, however it is surprising how ICE makes even short SuperBASIC programs look very impressive indeed. ICE support can be added to a wide range of existing programs, such as file copiers, BOOT routines, utilities, games and especially graphics programs and business software. If you are interested in using ICE TOOLKIT in commercial applications then please refer to the LICENSING section at the end.

2. THE TOOLKIT

ICE TOOLKIT is supplied on a single microdrive cartridge, which must be duplicated as soon as possible. It goes without saying that the easiest way of doing this is to use ICE's BACKUP routine.

The programs included are as follows...

BOOT	- 'Reminder' to read _DOC files first.
MBOOT	- Boot up Mice Toolkit.
DEMO_BAS	- Simple SuperBASIC demo program
MICETOOL_CDE	- Machine code extensions.
*MICETOOL_ASM	- Assembler source code for MICETOOL_CDE
ICONEDIT_BAS	- A simple icon editing program
*HEADER_ASM	- Assembler source code header file
*MOUSE_ASM	- e.g assembler to read mouse.
MICE_DEMO	- Simple Basic program to demonstrate readin and using the mouse.
+COLOUR	- Desktop colour editor (EXECable version)
MANUAL_DOC	- Quill document of these notes
MANUAL2_DOC	- Notes on additions made for the MOUSE
LOGO_BAS	- Basic program to produce ICE logo.
ARBOOT_BAS	- A simple routine for booting up Archive. This file is not intended to illustrate the toolkit, it simply would not fit on the CHOice cartridge where it belongs. By all means use the TOOLKIT to make a icon controlled version if you wish.

* files are standard assembler source files, and may be edited and viewed using any of the standard text editors, such as Metacomco's ED. They may be IMPORTED into QUILL where they can be altered or printed. The assembly code was produced by the Computer One assembler package, and so some alteration may be required for other assemblers.

+ files are EXECable (ICE EX files)

To have a glimpse of the available routines in action, insert a copy of the cartridge and either LRUN the BOOT program (not forgetting the device name), or preferably double-click the program from the ICE desktop. After booting, then LRUN/Double-click DEMO_BAS.

3. USING THE ICE TOOLKIT

By examining the demo program, you will notice that ICE TOOLKIT is activated just like any other set of SuperBASIC extensions:

```
1 a=RESPR(1024) : REM Reserve 1K in the RESPR area
3 CALL a : REM initialise routines
```

The supported routines are as follows:

addr = ALHEAP(bytes)	- Allocate Common Heap Space
REHEAP addr	- Release Common Heap Space
WSAVE console no., width, height, x, y	- Save screen area
WLOAD console no.	- Close console and restore
event_code = MOUSE(x,y)	- Create mouse pointer
item_number = MENU(#chan, map_addr)	- Activate menu & return item number.
zone_number = ZONE(map_addr)	- Search for mouse location in zone map
addr = ICE_BASE	- Returns start of ICE variables
ICON shape_addr,x,y	- Plot icon at x,y location

Each extension will now be discussed in detail.

addr = ALHEAP(bytes)

This function will allocate the specified number of bytes in the Common Heap. The address of the allocated block is returned and must be stored for later use. Compare to the SuperBASIC RESPR keyword.

REHEAP addr

This procedure releases Common Heap space. The addr value must be one of the addresses returned by a previous ALHEAP call. If addr is not a valid heap address, the system is likely to crash. The ability to free the heap space after use is the main difference between ALHEAP and RESPR.

WSAVE console no., width, height, x, y

WSAVE is the core routine for creating a pop-up window of any kind. The routine will save the specified screen area in the Common Heap. Note that the window image needs quite a lot of memory in the heap - a full screen would of course need 32k. An 'out of memory' error will occur if there isn't enough heap space available.

ICE supports up to 4 screen images at any one time. ICE screen images are numbered from 0-3. Note that ICE does not fully support stacked windows - if more than one window is opened so that the windows overlap, then they must be loaded in the reverse order to properly restore the screen.

The width, height, x, y values are identical to the similar values needed by the SuperBASIC WINDOW procedure to describe the window's size and position.

WLOAD console no.

This command is used to restore a previously defined screen area. ICE keeps track of the size and location of each image, and so only the image number needs to be given. After an image has been loaded back, the heap space used is returned to the operating system. If the specified image is not available (if it has already been WLOADed for example) then ICE will do nothing.

event_code = MOUSE(x,y)

This function allows ICE programs access to the 'mouse' pointer. The pointer is defined as a diagonal arrow, and current versions of ICE do not allow this to be altered. The pointer is non-destructive, and behaves in a similar way to a 'sprite' on other machines. When called this function will suspend SuperBASIC until the space-bar

```
310 GO TO 230
500 CLS #0:AT #0,1,6:INPUT #0,"SAVE as (B)ASIC, (A)ssembler or (L)BYTES
    :"!fm$
510 IF fm$(1)=="b"
520 CLS #0:AT #0,1,6:INPUT #0,"Starting line-number ="!lno
530 END IF
1000 CLS #0:AT #0,1,6:INPUT #0,"Filename :"!f$
1010 AT #0,3,0:INPUT #0,"(I)con or icon (M)ask :"!t$:IF t$(1)=="i" THEN
     f$=f$&"_icn":ELSE f$=f$&"_msk"
1020 OPEN_NEW #4,f$
1025 FOR m=131072+40*128+16 TO 131072+55*128+16 STEP 128
1026 IF fm$(1)=="a":PRINT #4,FILL$(" ",16);"DC.B    ";:
1027 IF fm$(1)=="b":PRINT #4,lno;"
```

Dease the efficiency of directing the pointer, and to keep ICE compatible with suitable mouse drivers, the ICE pointer drivers bypass the normal QDOS keyboard routines. The only bad effect of this procedure is that when the user presses the cursor keys or uses a joystick the keyboard characters are stored by QDOS in the current keyboard queue where they will be stored until a program reads from the keyboard in the usual manner. There are two solutions to this problem. One is to disable the QDOS keyboard routines before calling up the ICE pointer, and then enable them again afterwards. The seconds solution is similar, but instead of switching off the keyboard interrupts, the keyboard buffers are 'hidden' from the operating system. The two methods are as described below.

Method 1:

```
10 v=PEEK_L(163900) : REM Keep the old value!
20 POKE_L 163900,0 : REM Disable polled interrupts
30 n=MOUSE(50,50) : REM Activate mouse pointer
40 POKE_L 163900,v : REM Enable interrupts again
```

Method 2 (may disagree with some multi-tasking jobs):

```
10 v=PEEK_L(163900) : REM Keep the old value!
20 POKE_L 163916,0 : REM Hide the keyboard buffers
30 n=MOUSE(50,50) : REM Activate pointer
40 POKE_L 163916,v : REM Here are the buffers again!
```

Another method for getting around this problem is demonstrated in the DEMO_BAS program.

item_number = MENU(#chan, map_addr)

This command will activate a menu - a menu being defined as a list of items that may be selected. To use a menu a suitable console must first be opened using the OPEN or WOPEN command. The menu itself consists of a list of screen areas, known as a zone map. This table must be POKEd into previously allocated heap/RESPR areas, and is now defined.

A zone map allows ICE to know where each of the selections is. This map is used to decide which (if any) of the selections was being pointed to when the user 'clicked' - it is defined thus:

WORD - Origin of window (x-coord)

WORD - Origin of window (y-coord)

Then for each item in the menu...

BYTE - Width of item (0-255 pixels)

BYTE - Height of item (0-255 pixels)

WORD - X-coord of item relative to origin

WORD - Y-coord of item relative to origin

and lastly...

WORD - 65535 (-1) as an end of list marker
When the MENU command is used, SuperBASIC is suspended and the mouse pointer is activated. The value returned by the function is the number of the item in the table (0,...n) that was clicked. Clicking on screen areas that do not contain selection items will cause a return of -1.

zone_number = ZONE(map_addr)

If you do not like the way that ICE handles menus, you may write your own handlers. This routine facilitates this by performing a very fast zone search. The map_addr value must point to a zone table as defined for the MENU command. The function will return the zone number of the item currently being pointed to by the mouse pointer. The mouse sprite does not have to be currently on screen for this routine to work. If the pointer is at a location not included in the zone map, then a value of -1 is returned else the zone number is returned (N.B. 0 represents the first zone in the list).

The current x,y values may be obtained by using the values stored in the POINTER_X and POINTER_Y ICE variables.

Program example:

```
10 map_addr=RESPR(512)
20 REM Poke in the zone table bytes
30 POKE_W map_addr,10:POKE_W map_addr+2,20
40 POKE map_addr+1,16:POKE map_addr+5,10....etc.

100 n=ZONE(map_addr)
105 REM PEEK_W POINTER_X and POINTER_Y values
110 x=PEEK_W(ICE_BASE+150):y=PEEK_W(ICE_BASE+152)
120 PRINT "You clicked item number ";n+1
130 PRINT "The pointer was last at location ";x;",";y
```

ICON icon_map_addr,x,y

This command quite simply will draw a 24x16 pixel icon at the specifie coordinates. ICE icons are black and white only, and the x coordinate are taken to the nearest multiple of eight - RELATIVE TO THE TOP LEF HAND CORNER OF THE SCREEN. The latter limitation was imposed t increase drawing speed, b forcing the icon to the nearest 8 pixe boundary no bit shifting is required to plot the shape. If that sound complicated don't worry - all you need to know is that the x value i only accurate to the nearest 8 pixels.

The icon_map consists of 2 tables each of 48 bytes, one containing the actual shape of the icon, the other a mask. The mask is required t tell ICE which parts of the icon are transparent, in other words each pixel in the icon can be black, white or transparent. These tables must be poked into consecutive memory locations. A very simple utility has been provided to ease the design of icons. The program is describe later, and is able either to poke the data directly into memory or t create a text file with suitable data statements in it (SuperBASIC DATA statements or assembler DC.B lines). Have a look at the demo program to see how it pokes the icon data into memory, and to see the layout of the data statements.

4. Assembler Level Support

The first thing to know about ICE at this level is the memory map that it has on a standard QL. ICE uses 512 bytes of the Common Heap as a system variable area. This area is used for temporary workspace and for keeping track of configuration details e.g. desktop background colour, double-click speed. These variable must be available to any program that requires, no matter whether it is a SuperBASIC procedure or an independent job. To accomplish this, ICE stores the address of this space in the QDOS system variable area. This pointer has the label "ICE_DADR" and is found at $280E4 (164068 decimal), it is of course a 32-bit value. Try typing PRINT PEEK_L(164068) in SuperBASIC to find out the address. ICE also uses the area $280E8-$280EB to hold an identity code. The code is represented by the ASCII characters "ICYI", and is used by ICE to see whether it has already been booted, and this code is the standard way to indicate the presence/absence of an ICE ROM.

Five ICE routines have been vectored in current versions of ICE (v1.4 - v1.11), more may be vectored in the future, however you are advised to stick to the 5 routines described here. The vectors are stored as 16 bit offsets from the base address ($C000) and the vector table starts at $00E8.

Routine	Vector Address
OPEN_CONSOLE	$C02E
CLOSE_CONSOLE	$C030
GO_MOUSE	$C032
MENU	$C034
ZONE_SEARCH	$C036

THE ROUTINES ARE CALLED LIKE THIS:

```
N.B. -->  MOVE.L    ICE_DADR,A6   ; Get ICE system var. addr.
          LEA       $C000,A1      ; Get ROM base addr.
          MOVE.W    GO_MOUSE,A2   ; Get vector
          JSR       0(A1,A2.W)    ; DO IT!
```

*** N.B. REGISTER A6 MUST BE SET TO THE ICE VARIABLE ADDRESS BEFORE CALLING ANY ICE ROUTINE I.E. MOVE.L ICE_DADR,A6. ***

I will conclude this section by briefly describing the effect of each routine, please experiment and examine the assembler source code examples on the Microdrive cartridge.

Routine: OPEN_CONSOLE

Entry: D3 = Console number (0-3), A1 = Address of parameter block

Exit: D0 = QDOS error code, A0 = channel ID

D1,D2 smashed

OPEN_CONSOLE will open a QDOS channel using the CONsole device driver. The routine used to open the channel is UT_CON (QDOS vector $C6) - see the QL Technical Guide for a definition of the parameter block. The main difference between ICE consoles and QDOS consoles is that ICE keeps the overlapped screen area in the heap. You should ensure that you do not confuse the use of the console number, and you may care to turn back to the definition of the WSAVE command to remind yourself. ICE ONLY STORES THE ADDRESS OF THE PARAMETER BLOCK - WHEN THE CONSOLE IS CLOSED THIS PARAMETER BLOCK MUST BE UNCHANGED AND AT THE ORIGINAL MEMORY ADDRESS.

Routine: CLOSE_CONSOLE

Entry: D3 = console number to close

D0-D3/A0-A2 smashed

QDOS channel associated with the specified console is closed, and the overlapped screen area is restored. All heap space used by the routine is released.

Routine: GO_MOUSE

Exit: D2 = 1 or 2 depending on number of clicks

D0,D1,D3,A1,A2 smashed

The mouse arrow is enabled and may be moved by means of the curso keys/joystick (or a mouse in future ICE versions). This routine is als responsible for handling menus. The initial position of the arrow ma be defined by placing the appropriate values in the ICE syste variables POINTER_X and POINTER_Y. After using GO_MOUSE, there wil probably be a large number of unwanted characters in the keyboar buffer. These characters can be removed by repeatedly readin characters from a channel until a zero byte is returned. Alternativel you can zero the QDOS queue pointer (SV_KEYQ) before calling GO_MOUS and then restore the original address afterwards - see the descriptio of the MOUSE SuperBASIC extension.

To use the 'animated menus', three extra ICE variables must b initialised:

MENU_ID (4 bytes) : QDOS channel ID for the menu
MENU_MAP (4 bytes) : Address of menu map
MENU_ENTRY (2 bytes) : Latest menu item number (must be set to -1 before calling GO_MOUSE)

MENU_MAP MUST BE SET TO ZERO IF THE MENU FACILITIES ARE NOT BEING USED

The structure of the menu map was defined earlier for the MENU procedure.

On exit from GO_MOUSE, ICE system variable MENU_ENTRY will contain the selected menu item (0..n) or -1 if no item was selected.

Routine: MENU

Entry: A1 = address of menu definition, A0 = Channel ID

D0-D3,A1 smashed

The purpose of MENU is quite simply to display a menu. This is really just a convenient way of positioning, colouring and printing short text messages for use in menus.

The structure of a menu definition is as follows:

Word - number of items in list

Then for each item...

Byte - Cursor horizontal location (0-255 pixels) - REL. TO WINDOW ORIGIN

Byte - Cursor vertical location (0-255 pixels) - REL. TO WINDOW ORIGIN

Byte - Ink colour (0-255)

Byte - Strip colour (0-255)

Bytes - Selection text followed by zero byte

Routine: ZONE_SEARCH

Entry: A1 = address of menu map

Exit: D1 = item number (0-n) or -1 if not found

A1,A2,D0 smashed

This routine will search a menu map to see whether the current pointer coordinates are within the scope of a defined area. See the ZONE procedure description for more details.

5. ICE MEMORY MAP

LABEL	DEC.	HEX.	NOTES
ROM_ADDR	49152	$C000	BASE ADDRESS OF ROM CARTRIDGE
ICE_DADR	164068	$280E4	32-BIT POINTER TO ICE VARIABLES
ICE_ID	164072	$280E8	'ICY1'

The following addresses hold 16-bit vectors to ICE routines all vectors relative to $C000)

OPEN_CONSOLE	49198	$C02E
CLOSE_CONSOLE	49200	$C030
GO_MOUSE	49202	$C032
MENU	49204	$C034
ZONE_SEARCH	49206	$C036

MENU	49204	$C034
ZONE_SEARCH	49206	$C036

The following addresses are relative to the address held in ICE_DADR, and are found in the common heap area:

WINDOW0_ID	0	$00	QDOS ID FOR CONSOLE #0
WINDOW0_HP	4	$04	HEAP ADDRESS FOR CONSOLE #0
WINDOW0_PA	8	$08	PARAM POINTER FOR CONSOLE #0
WINDOW1_ID	12	$0C	QDOS ID FOR CONSOLE #1
WINDOW1_HP	16	$10	HEAP ADDRESS FOR CONSOLE #1
WINDOW1_PA	20	$14	PARAM POINTER FOR CONSOLE #1
WINDOW2_ID	24	$18	.
WINDOW2_HP	28	$1C	.
WINDOW2_PA	32	$20	ETC.
WINDOW3_ID	36	$24	.
WINDOW3_HP	40	$28	.
WINDOW3_PA	44	$2C	.
ICE_BASE	48	$30	START OF ICE JOB CODE
TEMP_BUFFER	52	$34	TEMPORARY BUFFER
FOLDER_PTR	116	$74	POINTER TO LATEST FOLDER ITEM
POINTER_X	150	$96	POINTER X-COORD
POINTER_Y	152	$98	POINTER Y-COORD
PARAMETERS	206	$CE	PARAMETER BUFFER
FOLDER_ADDR	220	$DC	ADDR OF FILE FOLDER IN HEAP
CLICK_DELAY	224	$E0	DOUBLE-CLICK DELAY
MENU_ID	280	$118	CHANNEL ID FOR CURRENT MENU
MENU_MAP	284	$11C	ADDRESS OF CURRENT MENU MAP (0 IF NON
MENU_ENTRY	288	$120	LATEST MENU ITEM NUMBER (-1 IF NONE)
FBUFFER	330	$14A	FILE HEADER BUFFER
LRUN_ADDR	394	$18A	ADDR OF SUPERBASIC LRUN CMD
MDRIVE_NAME	430	$1AE	MDV DEVICE NAME -'MDV'
DISK_NAME	434	$1B2	DISK DEVICE NAME - 'FLP'
USER_NAME	438	$1B6	USER DEVICE NAME - 'RAM'
KEYDELAY	442	$1BA	DELAY TILL KEYBD REPEAT
KEYRATE	443	$1BB	KEYBD REPEAT RATE
MOUSERATE	444	$1BC	MOUSE POINTER SPEED
CLICKRATE	445	$1BD	DOUBLE CLICK SPEED
DESK_COLOUR	446	$1BE	COLOUR OF DESKTOP (4=GREEN)
LED_COLOUR	447	$1BF	COLOUR OF CALC DISPLAY
BAUD	448	$1C0	BAUD RATE (9600 DEFAULT)
PP_TIME	450	$1C2	PPINTER TIMEOUT
PRINTER_NAME	452	$1C4	PRINTER DEV NAME - 'SER1'

N.B. All device names are in standard format (length word + string)

6. THE ICON EDITOR

Included on the microdrive cartridge is a crude, but functional editor. The program will allow icons to be created, viewed and output in a useful form. Just double-click ICON_EDITOR from ICE DESK to use it.

The editing section responds to the following commands:

UP/DOWN/LEFT/RIGHT- Move cursor

CTRL-UP/DOWN/LEFT/RIGHT- Move cursor and invert point

ENTER- Save icon data

ESC- Quit editor

F1- Restart editor

The program is used to define two files: the icon file and the mask file. As a general rule the mask image should look like a silhouette of the main icon for the type of icons used in ICE DESKTOP. A black point in the mask image represents a non-transparent point in the icon image.

Have a look at the demo program to see how the output data files may be used within SuperBASIC. Each file may be incorporated into a program by using the SuperBASIC MERGE command. Assembler text editors should have facilities to read text files and insert them at the current cursor position (e.g. the IF command in Metacomco's ED).

The MOUSE (Mice) now being supplied by Eidersoft will operate the cursor in place of the cursor keys without any modification to the example program provided with this toolkit.

The mouse alters a byte of memory within the same address space as the ICE rom, address 65535 (0000FFFF)hex.

```
BIT 0 & 1    change if the mouse moves up and down.
BIT 2 & 3      "     "   "    "    "    left and right.
Bit 4        Right button.
Bit 5        Centre button.
Bit 6        Left button.
```