DES v1. 0

(Desktop Environment System)

PROGRAMMER'S GUIDE

1: INTRODUCTION

This guide explains how to write programs which use the features of DES, the graphical user interface on ROM and disc from CampurSoft.

These programs are known as DES Applications; they are run from the DES desktop, using the FILE|RUN command.

DES Applications have to be written in machine code, using an assembler. However, much of your work is already done for you, as all the input and output routines that you could possibly need are already provided.

The DES routines provide such features as Text output, graphics, Alert Boxes, buttons, icons, pointer control, text entry, disc routines and list box routines.

It is assumed that you have a fair knowledge of CPC assembly language, as the things discussed in these instructions aren't for absolute beginners.

If you are writing DES Applications, you should only use the routines outlined in this guide. These routines are the key to upwards compatability, as they will remain the same in any future updates of the DES ROMs. There is no other guaranteed method of accessing the DES routines.

Memory usage

In order to provide easy access to the DES routines, DES installs a new jumpblock at &9200 in RAM. The memory above this address is all reserved, as DES uses it as a workspace.

DES also uses main memory below &3000 as a buffer area, so you can't store any code there. This area is used as follows:

```
&0040-&0fff    - Screen store for menus and alert boxes
&1000-&2fff    - Disc buffer: 8k is needed for the maximum disc
                 directory size, which is ROMDOS D20, with 256 files.
                 Each entry in the directory requires 32 bytes,
                 and 32*256 = 8k.
```

This leaves main memory from &3000 to &91ff for your program - that's around 25k. The second 64k of the 6128 machines (and upgraded 464s and 664s) is not used, so you can use this for storage as well - a program can thus access around 89k of memory, with use of bank switching.

The disc version of DES uses RAM banks 4 and 5 - the first two 16k blocks of the extra RAM - so you cannot use these if you want your program to run with both the disc and ROM versions of DES.

If you really need some more RAM, you could use the second extra bank - bank 5 - but your program cannot have an “exit” option, as the DES program manager is stored here. The disc version of DES stores its system routines in bank 4, so you must never use this bank, unless your program is to be ROM only.

DES provides two routines to deal with bank switching - one to check whether or not the computer has 128k, and another to switch the banks.

Click Zones

This is the method which DES uses to detect whereabouts on the screen the pointer was when it was last “clicked”. The programmer uses DES routines to define rectangular areas of the screen, each one of which is given a unique number. Up to 30 of these zones can be defined.

Each time you add a click zone, the number will be increased by 1; so, your program sets up three buttons, and no click zones have been set up already, the first button will be zone number 1, the second will be zone 2, and the third will be zone 3.

2: CREATING APPLICATIONS

Writing a DES Application is just like writing any other assembly program. However. a few simple guides have to be followed.

(i) Firstly, you'll have to save your object code to a file on disc, as the application has to be run from within DES. An assembler such as Arnor's Maxam makes this very easy, with the WRITE directive.

(ii) At the start of the program, you need to include a header which tells DES that the program is an application. The first bytes of the program have to be set up as follows:

```
db “DES APPLICATION:”
dw <start address>
```

Substitute the load address of the code for <start address>. For example, if your code starts at &4000, then the above will become

```
WRITE “DEMO”

ORG &4000

db “DES APPLICATION:”
dw &4000

.... rest of code ....
```

(iii) When DES runs your application, it provides you with a means of returning directly to the DES desktop. This is actually quite simple - when your program starts, the HL register will contain an address and the C register will contain a rom number. If you store these values, then you can use them as far address to recall DES. One method of doing this, which is recommended, is as follows:

```
WRITE “DEMO”

ORG &4000

db “DES APPLICATION:”
dw &4000

push bc                         ; Store rom number
push hl                         ; Store address
call main_program               ; Call main code of application
pop hl                          ; Retrieve address
pop bc                          ; Retrieve rom number
jp &001b                        ; Jump to address HL in rom number C

.mainprogram
.... rest of code....
```

For the disc version of DES, the rom select number which is passed onto the program will always be &00.

3: SUMMARY OF DES JUMPBLOCK ROUTINES

As previously stated, the DES jumpblock lives at &9200 in main memory. Each jumpblock entry has a unique name, which indicates the function of the routine. It is advised that you stick to using the names provided in these instructions, to avoid confusion.

Output/graphical routines:

&9200	PrintChr	Displays a single character
&9203	PrintString	Displays a string of characters
&9206	Space	Displays a space
&9209	BackSpace	Performs a back-space
&920c	TinyFont	Set text to “Tiny” font
&920f	DwdFont	Set text style to double-width
&9212	DimFont	Set text style to dimmed
&9215	InvFont	Set text style to inverse mode
&9218	PrintA	Output value in A in decimal
&921b	PrintHL	Output value in HL in decimal
&921e	PrintHexA	Output value in A in hex
&9221	PrintHexHL	Output value in HL in hex
&9224	Locate	Set DES cursor position
&9227	CursorOn	Displays a cursor at current position
&922a	CursorOff	Removes cursor from screen
&922d	ClearBox	Clears an areas of the screen
&9230	FillBox	Fills an area of the screen with a value
&9233	InvertBox	Inverts colours in an area of the screen
&9236	FileGard	Displays a “file card” window
&9239	NormBox	Displays a single-lined box
&923c	MenuBox	Displays a box with a shadow behind it
&923f	Window	Displays a window
&9242	AlertBox	Displays a box with a thick border
&9245	MenuBar	Displays a menu bar
&9248	AlertResponse	Gets user response from an alert box
&924b	Button	Displays a pushbutton
&924e	RadioButtons	Displays a column of radio buttons
&9251	RadioZones	Sets up click zones for radio buttons
&9254	CheckBoxes	Displays a column of check boxes
&9257	CheckZones	Sets up click zones for check boxes
&925a	DrawIcon	Displays an icon
&925d	DrawArrow	Displays an up or down arrow
&9260	DrawGraphic	Displays a graphic block

Input routines:

&9263	Mouse	Waits for user to select using pointer
&9266	MousePos	Returns pointer coordinates
&9269	Wait	Waits until all mouse select keys are released
&926c	FileName	Inputs and validates a filename
&926f	InputNumber	Inputs or edits a number
&9272	InputString	Inputs a string
&9275	EditString	Edits an existing string
&9278	StringZone	Sets input window for string editing
&927b	StringCase	Sets automatic upper case conversion
&927e	PullMenu	Display & select from a pull-down menu
&9281	ClearInput	Same as CLEAR INPUT in BASIC 1.1
&9284	ReadChr	Reads a single character with flashing cursor

Disc routines:

&9287	FormatDisc	Formats a disc to Data/System/ROMDOS
&928a	CheckOn	Enables advanced disc error checking
&928d	CheckOff	Disables advanced disc error checking
&9290	CheckAbandon	Aborts advanced disc error checking
&9293	ReadSector	Reads a sector from disc
&9296	WriteSector	Writes a sector to disc
&9299	ScanFormat	Scans disc & works out format
&929c	LastFormat	Returns values for format of last disc scanned
&929f	GetDrive	Returns current disc drive
&92a2	GetUser	Returns current user number
&92a5	SetDrive	Sets current disc drive
&92a8	SetUser	Sets user number
&92ab	GrindOff	Stops drive grinding during read error
&92ae	GrindOn	Restores drive settings after GrindOff
&92bl	ScanDir	Scans filenames from disc
&92b4	NameAddr	Returns address of a name scanned by “ScanDir”
&92b7	PrintName	Prints a filename scanned by “ScanDir”
&92ba	CentreName	Same as “PrintName”, except name is centred

Miscellaneous routines:

&92bd	PrinterOnline	Checks if printer is on-line
&92c0	C1rZones	Clears all click zones from memory
&92c3	RamZones	Sets up click zones from a list in memory
&92c6	SetZone	Sets up a single click zone
&92c9	ZoneAddr	Returns address of “number of zones” byte
&92cc	ZoneKeys	Set shortcut keys for click zone(s)
&92cf	ClearKeys	Clears all shorteut keys
&92d2	Ping	Performs the error sound
&92d5	ScrAddr	Returns screen address for a coordinate
&92d8	NextLine	Get address of next screen line down
&92db	StoreBox	Stores an area of the screen in RAM
&92de	RestoreBox	Redraws a stored area of the screen
&92el	ListPrev	Moves list box pointer up by 1 position
&92e4	ListNext	Moves list box pointer down by 1 position
&92e7	ListItem	Returns current item in list box
&92ea	ListSelect	Selects list box item under pointer
&92ed	ListUp	Scroll list box up by 1 position
&92f0	ListDown	Scroll list box down by 1 position
&92f3	ListInit	Initialise a list box
&92f6	ListDisplay	Displays visible part of list box
&92f9	ScrollBar	Displays a scroll bar
&92fc	RomCommand	Calls the “Execute ROM commands” function
&92ff	GetPercent	Returns a percentage for a value
&9302	SecPercent	Returns percentage for sector position
&9305	RetNum	Returns string equivalent of a number
&9308	ReadDirSecs	Reads in directory sectors from disc
&930b	WriteDirSecs	Writes directory sectors to disc
&930e	CampLogo	Displays CampurSoft logo

&9311	RAMtest	Checks if computer has 128k RAM
&9314	RAMbank	Switches in extra RAM banks
&9317	PokeName	Copies a filename into memory location
&931a	SelectDrive	Prompts for disc drive A or B
&931d	TwoButWin	Displays an Ok/Cancel window
&9320	SetMess	Enables/disables AMSDOS error messages
&9323	SelectFormat	Selects a disc drive and format
&9326	HoldESCbox	Displays a box with “Hold ESC to abort” in it
&9329	hESCpercent	Displays “x% completed…” in a HoldESCbox
&932c	GetBlanker	Returns address of screen blanker status byte
&932f	CameraVec	Called by pointer and flashing cursor routines
&9332	DisplayPointer	New pointer display routine
&9335	ClearPointer	New pointer removal routine

4: DETAILED DESCRIPTION OF DES JUMPLOCK ROUTINES

&9200 PrintChr

Displays a single character on the screen, and increments the DES screen cursor position.

Entry: A = character to display
Exit: Flags corrupt

Note that as well as the standard characters from CHR$(32) to CHR$(126), there are also some extra characters, as well as a set of control codes.

Extra characters:

128:	CHR$(143)	139:	<= symbol
129:	Pound sign	140:	>= symbol
130:	Copyright symbol	141:	approx. symbol
131:	Up arrow	142:	equivalence symbol
132:	Down arrow	143:	<< symbol
133:	Left arrow	144:	>> symbol
134:	Right arrow	145:	Tick
135:	CHR$(244)	146:	Left quote
136:	CHR$(245)	147:	Right quote
137:	CHR$(246)	148:	Bullet
138:	CHR$(247)	149:	Small bullet

(Where CHR$(n) is shown, the character is the same as the built-in CPC character of that number.)

Control codes:
 1: Shortcut key “underline” character: doesn't advance cursor
 4: Dimmed font on
 5: Dimmed font off
 6: Inverse font on
 7: “Bong” warning noise
 8: Backspace, doesn't change display
 9: Forward space, doesn't change display
10: Inverse font off
ll: Double width on
12: Clears screen
14: Double width off
15: Tiny font on
16: Tiny font off
17: Displays “A” or “B”, according to current disc drive

&9203 PrintString

Displays a string of characters and control codes on the screen, using the *PrintChr* routine.

```
Entry:  HL = address of string. String is terminated by CHR$(0).
Exit:   HL = address of byte after CHR$(0).
        AF, BC corrupt
```

As well as the control codes for the *PrintChr* routine, the following codes are also supported by *PrintString:*

```
18,n,x: Displays CHR$(x) n times.
31,x,y: Sets cursor position to (x,y) - see Locate routine
```

&9206 Space

Displays a space character - CHR$(32)

Entry: no conditions
Exit: AF corrupt

&9209 BackSpace

Moves the DES cursor backwards by one character position.

Entry: no conditions
Exit: All registers preserved

&920c TinyFont

Set text to “Tiny” font.

Entry: A=&00 to disable “Tiny” font, non-zero to enable it
Exit: All registers preserved

&920f DwdFont

Set text style to double-width.

Entry: A=&00 to disable double-width font, non-zero to enable it
Exit: All registers preserved

&9212 DimFont

Set text style to dimmed.

Entry: A=&00 to disable dimmed font, non-zero to enable it
Exit: All registers preserved

&9215 InvFont

Set text style to inverse mode.

Entry: A=&00 to disable inverse video, non-zero to enable it
Exit: All registers preserved

&9218 PrintA

Outputs a byte in Decimal, with no leading zeros.

Entry: A = value to display
Exit: All registers preserved

&921b PrintHL

Outputs a word in decimal, with no leading zeros.

Entry: HL = value to display
Exit: All registers preserved

&921e PrintHexA

Outputs a byte in hexadecimal.

Entry: A = value to display
Exit: All registers preserved

&9221 PrintHexHL

Outputs a word in hexadecimal.

Entry: HL = value to display
Exit: All registers preserved

&9224 Locate

Sets the DES cursor position.

Entry: H = X-coordinate (0-79)
L = Y-coordinate (0-199)
Exit: All registers preserved

&9227 CursorOn

Displays a cursor block at the current DES cursor position.

Entry: no conditions
Exit: All registers preserved

&922a CursorOff

Removes the cursor block from the screen.

Entry: no conditions
Exit: All registers preserved

&922d ClearBox

Clears a rectangular area of the screen.

Entry: H = X-coordinate (0-79)
L = Y-coordinate (0-199)
D = Width of area (in characters)
E = Height of area (in lines)
Exit: All registers preserved

&9230 FillBox

Fills a rectangular area of the screen with a given byte.

Entry: H = X-coordinate (0-79)
L = Y-coordinate (0-199)
D = Width of area (in characters)
E = Height of area (in lines)
A = Byte to fill with
Exit: All registers preserved

&9233 InvertBox

Inverts colours in a rectangular area of the screen.

```
Entry:  H = X-coordinate (0-79)
        L = Y-coordinate (0-199)
        D = Width of area (in characters)
        E = Height of area (in lines)
Exit:   All registers preserved
```

&9236 FileCard

Displays a “file card” window.

```
Entry:  H = X-coordinate (0-79)
        L = Y-coordinate (0-199)
        D = Width of window (in characters)
        E = Height of window (in lines)
Exit:   DES cursor position is set to start of window status bar
        All registers preserved
```

&9239 NormBox

Displays a single-lined box.

```
Entry:  H = X-coordinate (0-79)
        L = Y-coordinate (0-199)
        D = Width of box (in characters)
        E = Height of box (in lines)
Exit:   All registers preserved
```

&923c MenuBox

Displays a box with a shadow behind it.

```
Entry:  H = X-coordinate (0-79)
        L = Y-coordinate (0-199)
        D = Width of box (in characters)
        E = Height of box (in lines)
Exit:   All registers preserved
```

&923f Window

Displays a window.

```
Entry:  H = X-coordinate (0-79)
        L = Y-coordinate (0-199)
        D = Width of box (in characters)
        E = Height of box (in lines)
Exit:   DES cursor position is set to start of window title bar
        All registers preserved
```

&9242 AlertBox

Displays a box with a thick border.

```
Entry:  H = X-coordinate (0-79)
        L = Y-coordinate (0-199)
        D = Width of box (in characters)
        E = Height of box (in lines)
Exit:   All registers preserved
```

&9245 MenuBar

Displays a menu bar at the top of the screen.

Entry: HL = Address of menu bar text, terminated by CHR$(0)
Exit: All registers preserved

&9248 AlertResponse

Returns user response from an alert box.

Entry: HL = Address of alert text, terminated by CHR$(0). All control
            codes from PrintChr routine still work, plus CHR$(13)
            takes a new line in the alert message.
        A = icon number and number of buttons:
               bits 0-6 contain icon number
               - set bit 7 to give a single button 110k11 response
               - clear bit 7 for a two-button response,
                 and set DE & BC = Data for left & right buttons
                      (Data consists of INKEY number,string,&00)
Exit: AF corrupt
        For two-buttoned alert box, left button returns carry set,
        right button returns no carry

&924b Button

Displays a pushbutton, and sets it up as the next click zone.

Entry: H = X-coordinate (0-79)
        L = Y-coordinate (0-199)
        B = Width of button (in characters)
        DE = address of button text, terminated by CHR$(0)
Exit: All registers preserved

&924e RadioButtons

Displays a column of radio buttons.

Entry: H = X-coordinate of column (0-79)
        L = Y-coordinate of top button in column (0-199)
        B = Number of buttons in column
        C = Distance between tops of two consecutive buttons
        A = Button to set
Exit: All registers preserved

&9251 RadioZones

Sets up click zones for a column of radio buttons.

Entry: H = X-coordinate of column (0-79)
        L = Y-coordinate of top button in column (0~199)
        B = Number of buttons in column
        C = Distance between tops of two consecutive buttons
Exit: All registers preserved

&9254 CheckBoxes

Displays a column of check boxes.

Entry: H = X-coordinate of column (0-79)
        L = Y-coordinate of top button in column (0-199)
        B = Number of buttons in column
        C = Distance between tops of two consecutive buttons
        DE = Status table - one entry for each check box,
        zero to clear, non-zero to set.

Exit: All registers preserved

&9257 CheckZones

Sets up click zones for a column of check boxes.

Entry: H = X-coordinate of column (0-79)
        L = Y-coordinate of top button in column (0-199)
        B = Number of buttons in column
        C = Distance between tops of two consecutive buttons

Exit: All registers preserved

&925a DrawIcon

Displays a DES icon on the screen.

Entry: D = X-coordinate (0-79)
        E = Y-coordinate (0-199)
        A = icon number
Exit: All registers preserved

Icon numbers:

1: Query   5: Basic   9: Folder
2: Warning   6: Binary   10: Waste bin
3: Printer   7: Document   ll: Open waste bin
4: ROM   8: Mouse   12: Disc

If the icon number is not one of the above, then the icon data pointed at address in HL will be displayed.
The icon data is standard MODE 2 sprite data, of width 4 bytes and height 16 bytes.

&925d DrawArrow

Displays an up or down arrow.

Entry: H = X-coordinate (0-79)
        L = Y-coordinate (0-199)
        A = &00 for up arrow, non-zero for down arrow
Exit: All registers preserved

&9260 DrawGraphic

Displays a graphic block on the screen.

Entry: D = X-coordinate (0-79)
        E = Y-coordinate (0-199)
        B = Width of graphic (in bytes)
        C = Height of graphic (in lines)
        HL Address of graphic data
Exit: All registers preserved

The graphic data is standard MODE 2 sprite data.

&9263 Mouse

Waits for the user to make a selection using the pointer.

```
Entry:  no conditions
Exit:   B = Character X-coordinate (0-79)
        C = Character Y-coordinate (0-199)
        DE = Graphics X-coordinate (0-638)
        HL = Graphics Y-coordinate (0-398)
        A = Click zone number (&00 if user hasn't clicked on a zone)
        IX, IY, flags corrupt
```

&9266 MousePos

Returns pointer coordinates.

```
Entry:  no conditions
Exit:   B = Character X-coordinate (0-79)
        C = Character Y-coordinate (0-199)
        DE = Graphics X-coordinate (0~638)
        HL = Graphics Y-coordinate (0-398)
        A = Click zone number (&00 if user hasn't clicked on a zone)
        flags corrupt
```

&9269 Wait

Waits until the user has released all “click” keys - i.e. SPACE, ENTER and RETURN.

```
Entry:  no conditions
Exit:   All registers preserved
```

&926c FileName

Inputs and validates a filename. The filename has to be a valid AMSDOS filename, otherwise an error message is displayed, and the name has to be changed.

```
Entry:  D = X-coordinate for input (0-79)
        E = Y-coordinate for input (0-199)
        HL = Address to store filename at
Exit:   Carry set if ESC pressed, no carry if RETURN pressed
        All other registers preserved
```

&926f InputNumber

Inputs or edits a number.

```
Entry:  H = X-coordinate for input (0-79)
        L = Y-coordinate for input (0-199)
        BC = Minimum permissable value
        DE = Maximum permissable value
        A = Maximum number of digits allowed
        No carry -> enter new nunber
        Carry set -> edits number in IX register
Exit:   HL = Number typed
        Carry set if ESC pressed, no carry if RETURN pressed
        All other registers preserved
```

&9272 InputString

Inputs a string.

```
Entry:  D = X-coordinate for input (0-79)
        E = Y-coordinate for input (0-199)
        HL = Address to store string at
        C = Maximum number of characters allowed
        No carry -> use built-in validation
        Carry set -> calls routine at address in IX for each keypress
Exit:   Carry set if ESC pressed, no carry if RETURN pressed
        C = Length of string
        B corrupt
        All other registers preserved
```

&9275 EditString

Edits an existing string.

Entry and exit conditions are the same as InputString (see above).

&9278 StringZone

Sets input window for string editting.

```
Entry:  D = X-coordinate of left of window
        E = X-coordinate of right of window
Exit:   All registers preserved
```

&927b StringCase

Sets automatic upper case conversion.

Entry: A = &00 - allow all input characters
        A = non-zero - convert characters into upper case
Exit: All registers preserved

&927e PullMenu

Display & select from a pull-down menu.

Entry: HL = Address of menu option data
        DE = Address of invalid option data
Exit: BC, DE, HL are the same as the *Mouse* routine
        No carry -> Selected from menu, A = Menu option number
        Carry set -> Selected outside menu, A = Click zone number
        If A = &FF, then the user has pressed ESC.
        IX, IY corrupt.

Menu option data:

Byte 0: X-Coordinate of left of menu (0-79)
Byte 1: Y-Coordinate of top of menu (0-199)
Byte 2: Width of menu (in characters)
Byte 3: Number of options in menu

... followed by a string for each menu option, each string being terminated by CHR$(0). If you include the CHR$(1) control code in an option string, the next character in the string will be taken as the shortcut key for that option.

If a string is defined as simply a dash (-) character, followed by CHR$(0), then a menu seperator bar will be displayed at that position.

Invalid option data:

This tells DES which menu options should be dimmed, i.e. unselectable. If all options are selectable, then this data is simply a zero byte.

If there are dimmed options, then this data will consist of the number of each dimmed option; the data is terminated by a zero byte. So, if options 1,3 and 4 are to be dimmed, point the DE register to a table containing the bytes 1,3,4,0.

&9281 ClearInput

Same as CLEAR INPUT in Basic 1.1

Entry: no conditions
Exit: All registers preserved

&9284 ReadChr

Reads a single character from the keyboard, with flashing cursor. The cursor will be displayed at the current DES screen cursor position. The character will be converted into upper case, if the *StringCase* routine has been set for this.

Entry: no conditions
Exit: A = Character typed
        HL, DE, BC corrupt

&9287 FormatDisc

Formats a disc to Data, System or ROMDOS formats. The ROMDOS formats which are supported are D1, D10, D2, D20.

```
Entry:  A = Format number:    1: Data       2: Vendor
                              3: D1         4: D10
                              5: D2         6: D20
        HL = Address of 2204-byte buffer for screen storage. If this
             value is set to zero, then the screen won't be stored.
Exit:   All registers corrupt
```

&928a CheckOn

Enables advanced disc error checking, replacing the AMSDOS error messages. Whilst this is enabled, any text output using the standard &BB5A printing routine will be checked to see if it is an error message. If a call to &BB06 is made, then DES recognises it as a “Retry, Ignore or Cancel?” error, and displays a “Retry/Cancel” alert box. As a result, you can't use the &BB06 or &BB5A calls when the error checking is enabled.

```
Entry:  No conditions
Exit:   AF corrupt
```

&928d CheckOff

Disables advanced disc error checking, set up by the *CheckOn* routine. If an error which is not of the “Retry, Ignore or Cancel?” type, such as “File does not exist”, then an “Ok” alert box is displayed, informing you of the error.

```
Entry:  No conditions
Exit:   All registers preserved
```

&9290 CheckAbandon

Aborts advanced disc error checking, without reporting any error messages which have been detected.

```
Entry:  No conditions
Exit:   All registers preserved
```

&9293 ReadSector

Reads a sector from disc. This is simply a convenient way of calling the AMSDOS routine.

```
Entry:  HL = Address of 512-byte buffer to read into
        E = Drive number (&00 for A, &Ol for B)
        D = Track number
        C = Sector number
Exit:   DE, BC, HL, IX, IY preserved
        Carry set -> sector read Ok, A = 0
        No carry -> sector not read, A = error code, HL corrupt
```

&9296 WriteSector

Writes a sector to disc. Like *ReadSector,* this is just an easy way of calling the AMSDOS routine.

```
Entry:  HL = Address of 512-byte buffer to write from
        E = Drive number (&00 for A, &Ol for B)
        D = Track number
        C = Sector number
Exit:   DE, BC, HL, IX, IY preserved
        Carry set -> sector read Ok, A = 0
        No carry -> sector not read, A = error code, HL corrupt
```

&9299 ScanFormat

Scans a disc and identifies the format. Note that this routine calls the CheckOn and CheckOff routines, so if you've already called *CheckOn, you'll* need to call *CheckOff* before calling *ScanFormat.*

```
Entry:  Carry set -> Report all errors
        No carry -> Don't report a Disc read error

Exit:   A = Format number:    1: Data        2: Vendor
                              3: D1          4: D10
                              5: D2          6: D20
                              7: PCW in A    8: PCW in B
        If A = 0, then the disc format wasn't recognised.
        If A = &FF, then a read error occurred, indicating that the disc was not formatted correctly,
        but no error message was displayed.
        All other registers preserved
```

&929c LastFormat

Returns details about the format of the last disc tested by the *ScanFormat* routine.

```
Entry:  no conditions
Exit:   A = number of tracks on disc
        B = number of sectors in directory
        C = first sector number
        D = track number of directory
        E = number of sectors per track
        HL = maximum K available on disc
        IX, IY preserved
```

&929f GetDrive

Returns current disc drive.

```
Entry:  No conditions
Exit:   E = &00 if current drive is A
        E = &Ol if current drive is B
        All other registers preserved
```

&92a2 GetUser

Returns current user number.

```
Entry:  no condtions
Exit:   A = Current user number
```

&92a5 SetDrive

Sets current disc drive.

Entry: E = &00 to set current drive to A
        E = &0l to set current drive to B
Exit: All registers preserved

&92a8 SetUser

Sets current user number. Although it is possible to set any user number, from 0255, AMSDOS and DES will only recognise users 0-15.

Entry: A = user number
Exit: All registers preserved

&92ab GrindOff

Stops disc drive grinding during read error, so error message is displayed more quickly.

Entry: no conditions
Exit: All registers preserved

&92ae GrindOn

Restores drive settings altered by *GrindOff.*

Entry: no conditions
Exit: All registers preserved

&92bl ScanDir

Scans filenames from disc in current drive. These filenames are stored in a buffer, and can be displayed or referenced later. Note that this routine calls the CheckOn and CheckOff routines, so if you've already called *CheckOn, you'll* need to call *CheckOff* before calling *ScanFormat.*

Entry: A = Scan type: set bit 0 to include hidden files
                       set bit 1 to reset all characters to < 128
        DE = Address of 3-character extension of files to scan
        If the extension begins with a “*”, all files are scanned.
        Note that the “?” wildcard character is NOT supported.
Exit: A = number of files scanned
        HL = number of K free on disc
        Carry set -> disc read Ok
        No carry -> error occurred whilst reading disc
        All other registers corrupt

&92b4 NameAddr

Returns address in memory of a name scanned by *ScanDir.*

Entry: A = filename number
Exit: HL = address of filename
        Flags corrupt, all other registers preserved

&92B7 PrintName

Displays a filename, scanned by *ScanDir,* at the current DES cursor position.

Entry: A = filename number
Exit: All registers preserved

&92ba CentreName

Displays a filename, scanned by *ScanDir,* at the current DES cursor position. The name will be centred within the 12-character filename length, so, for example, it can be displayed centrally under an icon.

Entry: A = filename number
Exit: All registers preserved

&92bd PrinterOnline

Checks if the printer is on-line. If it isn't then an alert message will be displayed, and the user can retry the test if they wish.

Entry: No conditions
Exit: Carry set -> Printer is on-line
        No carry -> Printer is not on-line
        All other registers preserved

&92c0 C1rZones

Clears all mouse click zones from memory.

Entry: no conditions
Exit: All registers preserved

&92c3 RamZones

Sets up click zones from a list in RAM. The list consists of xl,yl,x2,y2 for each click zone.

Entry: HL = Address of click zone data list
        B = number of zones to initialise
Exit: All registers preserved

&92c6 SetZone

Sets up a single click zone.

Entry: H = X-Coordinate of left of zone (0-79)
        L = Y-Coordinate of top of zone (0-199)
        D = X-Coordinate of right of zone (0-79)
        E = Y-Coordinate of bottom of zone (0-199)
Exit: HL, DE corrupt

All other registers preserved

&92C9 ZoneAddr

Returns the address where the “number of zones” counter is stored.

Entry: no conditions
Exit: HL = Address of “number of zones” counter
        All other registers preserved

&92ce ZoneKeys

Set up shortcut keys for a number of click zones.

Entry: HL = Address of key data list (list of: key number, zone ...)
        B = Number of keys to set up

Exit: All registers preserved

&92cf ClearKeys

Clears all shortcut keys set up using the *ZoneKeys* routine.

Entry: no conditions
Exit: All registers preserved

&92d2 Ping

Performs the error “bong” sound.

Entry: no conditions
Exit: All registers preserved

&92d5 ScrAddr

Returns the screen address for a coordinate.

Entry: H = X-Coordinate (0-79)
        L = Y-Coordinate (0-199)
Exit: HL = Screen address of coordinate
        All other registers preserved

&92d8 NextLine

Get address of the next screen line.

Entry: HL = Address of current screen line
Exit: HL = Screen address of next screen line
        All other registers preserved

&92db StoreBox

Stores an area of the screen in RAM. Note that the buffer for storing the screen area has to be at least (Width*Height)+4 bytes long.

Entry: H = X-Coordinate (0-79)
        L = Y-Coordinate (0-199)
        D = Width of area (in characters)
        E = Height of area (in lines)
        BC = Address of buffer to store screen in
Exit: All registers preserved

&92de RestoreBox

Redraws an area of the screen, which was stored using *StoreBox.*

Entry: BC = Address of screen storage buffer
Exit: All registers preserved

&92el ListPrev

Moves list box pointer up by 1 position.

Entry: IX = Address of 9-byte status buffer
Exit: All registers preserved

&92e4 ListNext

Moves list box pointer down by 1 position.

Entry: IX = Address of 9-byte status buffer
Exit: All registers preserved

&92e7 ListItem

Returns currently selected string in list box.

Entry: IX = Address of 9-byte status buffer
Exit: HL = Address of string
A = String number
All other registers preserved

&92ea ListSelect

Selects list box item currently under pointer.

Entry: IX = Address of 9-byte status buffer
Exit: All registers preserved

&92ed ListUp

Scrolls list box up by 1 position.

Entry: IX = Address of 9-byte status buffer
Exit: All registers preserved

&92f0 ListDown

Scrolls list box down by 1 position.

Entry: IX = Address of 9-byte status buffer
Exit: All registers preserved

&92f3 ListInit

Initialises and displays a list box.

Entry: IX = Address of 9-byte status buffer
A = Number of entries in list
B = Width of list box (in characters)
C = Number of items visible at once
H = X-Coordinate of list box (0-79)
L = Y-Coordinate of list box (0-199)
DE = list data
Exit: All registers preserved

&92f6 ListDisplay

Displays the currently visible part of a list box.

Entry: IX = Address of 9-byte status buffer
Exit: All registers preserved

&92f9 ScrollBar

Displays a scroll bar.

Entry: H = X-Coordinate (0-79)
L = Y-Coordinate (0-199)
B = Current item number
C = Total number of items
D = Number of items displayed at once
E = Height of scroll bar (in lines)
Exit: All registers preserved

&92fc RomCommand

Calls the “Execute ROM commands” function, as used by the ROM icon in the DES desktop. Note that this routine wipes most of the main memory, so any program in RAM will be lost; as a result, this is only really useful when being called from another ROM.

Entry: no conditions
Exit: All registers preserved

&92ff GetPercent

Returns a percentage, i.e. (current value / maximum value) * 100.

Entry: DE = Current value
        HL = Maximum value
Exit: A = percentage (0-100)
        All other registers preserved

&9302 SecPercent

Returns percentage for a position within a disc.

Entry: D = Track number
Exit: A = percentage (0-100)
        All other registers preserved

&9305 RetNum

Converts a number to a string.

Entry: HL = Number to convert
Exit: HL = Address of string produced by conversion
        All other registers preserved

&9308 ReadDirSecs

Reads in the directory sectors from disc in current drive. Note that this routine calls the CheckOn and CheckOff routines, so if you've already called *CheckOn, you'll* need to call *CheekOff* before calling *ReadDirSecs.*

Entry: HL = Address of an Sk (&2000 byte) buffer
Exit: All registers (except IX & IY) corrupt
        Carry set -> Directory read Ok
        No carry -> error whilst reading disc

&930b WriteDirSecs

Writes directory sectors to disc in current drive. Note that this routine calls the CheckOn and CheckOff routines, so if you've already called *CheckOn, you'll* need to call *CheckOff* before calling *Wri teDirSecs.*

Entry: HL = Address of directory buffer
Exit: All registers (except IX & IY) corrupt
        Carry set -> Directory written Ok
        No carry -> error whilst writing to disc

&930e CampLogo

Displays the CampurSoft logo on the screen.

Entry: D = X-Coordinate (0-79)
        E = Y-Coordinate (0-199)
Exit: All registers (except HL & BC) preserved

&9311 RAMtest

Checks if the computer has 128k RAM.

Entry: no conditions
Exit: Zero flag set -> 128k available
        Zero flag not set -> Extra 64k not available

&9314 RAMbank

Switches in RAM banks from the extra 64k to &4000 in main RAM.

Entry: A = 1, 2, 3 or 4 -> switches in extra RAM bank 1-4
        A = 0 -> switches back to “normal” RAM.

Exit: All registers preserved

&9317 PokeName

Copies a filename (read in by ScanDir) into a memory location. Could be used, for example, to display a filename in an alert box.

Entry: A = filename number
        DE = Address of 12-byte buffer to load the name into
Exit: All registers preserved

&931a SelectDrive

Asks user to select either drive A or drive B, using radio buttons.

Entry: HL = Address of window title string, terminated by CHR$(0)
        (This takes the same format as PrintString)
Exit: A = &00 -> drive A selected
        A = &0l -> drive B selected
        A = &FF -> “Cancel” selected
        Flags corrupt, all other registers preserved

&931d TwoButWin

Displays a window with “Ok” and “Cancel” buttons in it. “Ok” is set up as click zone number 1, and “Cancel” is click zone 2. The screen area which the window is displayed over is saved at address &F0 in RAM.

Entry: HL = Address of window text, terminated by CHR$(0)
        (This takes the same format as PrintString)
Exit: AF, BC, DE, HL corrupt

&9320 Setmess

Enable or disable AMSDOS error messages, reducing them to a simple “Bad command” error.

```
Entry:  A = &00 to enable error messages
        A = &FF to disable error messages
Exit:   A = previous state
        All other registers preserved
```

&9323 SelectFormat

Selects a disc drive and format, using a window, radio buttons and a list box- the same as DISC:FORMAT on the DES desktop.

```
Entry:  no conditions
Exit:   No carry -> User has selected “Cancel”
        Carry set -> User has selected “Format”:
        A = Format number:    1: Data        2: Vendor      3: D1
                              4: D10         5: D2          6:D20
        All registers (except IY) corrupt
```

&9326 HoldESCbox

Displays a box with a “Hold ESC to abort” message in it.

```
Entry:  no conditions
Exit:   All registers preserved
```

&9329 hESCpercent

Displays a percentage completed message in a *HoldESCbox.*

```
Entry:  A = Percentage to display
Exit:   All registers preserved
```

&932c GetBlanker

Returns the address of the screen blanker status byte. Setting the value at this address to zero will disable the screen blanker, nonzero will enable it.

```
Entry:  no conditions
Exit:   HL = address of screen blanker status byte
        All other registers preserved
```

&932f CameraVec

This vector is called by pointer and flashing cursor routines. You have to make sure that theregisters are all preserved, as they may contain values used by the routines.

&9332 DisplayPointer
&9335 ClearPointer

These two routines are vectors which can be patched to replace the DES on-screen pointer with one of your own choosing. This could be used, for example, for drawing rubber-band lines in an art program.

Initially, the two routines contain only RET instructions, and the standard DES pointer is used. However, if you patch them with a jump to your own routine, a different pointer can be displayed.

Simply poke the two addresses with JMP xxxx instructions, and your routine will be called instead. DES will enter your routine with DE=graphics X coordinate and HL=graphics Y coordinate of the pointer. Your routine can corrupt all registers.

DisplayPointer is called by DES to display the pointer, and *ClearPointer* is called to clear the pointer from the screen.

To return to the standard DES pointer, simply poke both addresses with RET (&c9) instructions.

5: USING LIST BOXES

In all of the DES list box routines, a 9-byte buffer is required for each list box. The contents of this buffer are as follows:

```
Byte 0:        Width of list box (in characters)
Byte 1:        Number of entries in list
Byte 2:        Number of entries displayed at once
Bytes 3,4:     Address of list box strings
Byte 5:        X-coordinate of list box (0~79)
Byte 6:        Y-coordinate of list box (0~199)
Byte 7:        Top visible item in list
Byte 8:        Currently selected item
```

When a list box is set up, using *ListInit,* three click zones are added. These are the list area, the top arrow in the scroll bar, and the bottom arrow in the scroll bar.

The control routine is something like:

```
.loop call wait call mouse

cp l:call z,ListSelect:jr z,loop
cp 2:call z,ListUp:jr z,loop
cp 3:call z,ListDown:jr z,loop
.... rest of code ....
```

To get the keyboard shortcut keys, you'll need to define two “impossible” keyboard equivalents- for example, assign key f8 to zone 100 and key f5 to zone 101. You'll never be able to select these zones using the pointer. Add the following to your code:

```
cp 100:call z,ListPrev:jr z,loop
cp 101:call z,ListNext:jr z,loop
.... rest of code ....
```

Note that it is possible to have more than one list box on the screen at once. In fact, you could probably have as many as nine, bearing in mind that 3 click zones are needed for each list box, and you'll need a number of click zones for buttons outside the list boxes. All you have to do is point IX to a different 9-byte buffer, when you're calling the routines. This technique was used in the *RomCommand* routine and in FILE:DES SETUP on the DES desktop.

If you experience any difficulty writing DES Application programs, you can contact DES's programmer at the following address:

Michael Beckett
11 Steeple Gardens
Antrim
Co. Antrim
BT41 1BW

Telephone 0849 463 786 Saturday & Sunday 7.30pm - 10.00pm ONLY.

Please note that CampurSoft is interested in marketing third party DES applications that pass our quality standards. For more information on this subject write to the address at the front of this manual.