

EPOC O/S SYSTEM SERVICES

Version 2.30

(C) Copyright Psion PLC 1990-98

All rights reserved. This manual and the programs referred to herein are copyrighted works of Psion PLC, London, England. Reproduction in whole or in part, including utilization in machines capable of reproduction or retrieval, without express written permission of Psion PLC, is prohibited. Reverse engineering is also prohibited.

The information in this document is subject to change without notice.

Psion and the Psion logo are registered trademarks, and Psion, Psion MC, Psion HC, Psion Series 3, Psion Series 3s, Psion Series 3a, Psion Series 3c, Psion Siena and Psion *Workabout* are trademarks of Psion PLC.

Intel 8086 and 80286 are registered trademarks of Intel Corporation. IBM, IBM XT and IBM AT are registered trademarks of International Business Machines Corp. Microsoft and MS-DOS are registered trademarks of Microsoft Corporation. Psion PLC acknowledges that some other names referred to are registered trademarks.

CONTENTS

1 Introduction.....	1-1
System services	1-1
Single service interrupts	1-1
Multi service interrupts	1-1
Calling conventions	1-1
Documentation conventions	1-2
Include file epocdefs.inc	1-3

2 Segmented Memory Management.....	2-1
Memory segment names.....	2-1
Paragraphs	2-1
Permanent segments.....	2-1
Directly accessing segments	2-1
Size of available segmented memory.....	2-2
Creating a memory segment.....	2-3
Deleting a memory segment	2-4
Opening a memory segment	2-4
Closing a memory segment	2-4
Closing a locked or device segment	2-5
Locking a memory segment.....	2-5
Unlocking a memory segment	2-5
Size of a memory segment.....	2-5
Adjusting the size of a memory segment.....	2-6
Finding all segments	2-6
Copying to a memory segment	2-6
Copying from a memory segment.....	2-7
Size of RAM disk.....	2-7

3 Heap Memory Management.....	3-1
Dynamics of heap memory	3-1
Allocating heap memory	3-1
Re-allocating heap memory.....	3-2
Adjusting the size of heap memory.....	3-2
Freeing heap memory.....	3-3
Size of a heap cell	3-3
Setting the heap granularity	3-3
Size of available heap memory	3-3

4 Semaphore Management	4-1
Creating a semaphore.....	4-1
Deleting a semaphore.....	4-1
Waiting on a semaphore.....	4-1
Signalling once	4-2
Signalling more than once.....	4-2
Signalling once without re-schedule	4-2

5 Message Management..... 5-1

Inter process communication	5-1
Order of message reception	5-1
The message system and the I/O system	5-2
Initializing the message system	5-2
Asynchronous message reception	5-2
Synchronous message reception.....	5-3
Cancelling queued message receive	5-3
Sending messages	5-3
Sending and getting a reply asynchronously	5-4
Sending and waiting for a reply.....	5-4
Freeing a message	5-4
Requesting a signal from the Supervisor.....	5-5
Cancelling requested signal from the Supervisor	5-5
Cancelling requested signal from the Supervisor by type	5-6

6 Dynamic Library, Category and Object Management..... 6-1

Library names	6-1
Loading a dynamic library.....	6-1
Unloading a dynamic library	6-2
Linking a dynamic library	6-2
Getting a dynamic library handle	6-2
Getting a DYL handle by number	6-3
Creating an object by number	6-3
Creating an object by handle	6-3
Destroying an object.....	6-4
Sending a message	6-4
Sending a superclass message.....	6-4
Sending a direct message	6-5
Enter a sent message	6-5
Open a multi library file	6-5
Loading a multiple dynamic library.....	6-6
Reclass an object by number	6-6
Reclass an object by handle	6-7
Copying data from a category.....	6-7
Enter a control region.....	6-7
Leaving a control region.....	6-7
Returning from a method.	6-8

7 Device Management..... 7-1

Device names.....	7-1
Device drivers	7-1
Opening a physical device driver.....	7-1
Getting the PDD entry point.....	7-2
Installing a device driver	7-2
Holding all device drivers.....	7-2
Resuming all device drivers.....	7-3
Loading a logical device driver.....	7-3
Loading a physical device driver	7-3
Deleting a device driver.....	7-3
Removing a device driver	7-4
Querying the number of units	7-4
Finding all devices	7-4
Calling a device vector	7-5

8 Input Output Management	8-1
Devices and files	8-1
Sound file format	8-1
Asynchronous I/O	8-2
Asynchronous I/O without error reporting	8-2
Synchronous I/O	8-3
Chain to root device	8-3
Chain to superclass device	8-4
Wait for I/O completion	8-4
Wait for specific request to complete	8-4
Polling for completion	8-5
Signalling completion	8-5
Signalling completion by process ID	8-5
Signalling completion with no reschedule	8-5
Adding a handler	8-6
Removing a handler	8-6
Enabling a handler	8-6
Requesting a reset	8-7
Cancelling a requested reset	8-7
Opening a device	8-7
Closing a device	8-8
Reading from a device	8-8
Writing to a device	8-8
Seeking on a device	8-8
Mouse and keyboard	8-9
Adding an application handler	8-9
Removing an application handler	8-10
Enabling an application handler	8-10
Getting the shift states	8-10
Wait for I/O completion no handlers	8-10
Requesting a signal from the Supervisor	8-11
Cancelling requested signal from the Supervisor	8-11
Request signal on next half second	8-11
Query the completion of IoNextHalfSecond	8-12
Playing back a sound file synchronously	8-12
Playing back a sound file asynchronously	8-12
Cancelling playing back a sound file	8-13
Recording sound to file synchronously	8-13
Recording sound to file asynchronously	8-14
Cancelling recording sound to a file	8-14
Input Output Management update	8-15
Asynchronous partial sound file replay	8-15
 9 File Management	 9-1
The file server	9-1
Connecting to the file server	9-1
Execute an image file	9-1
Parse a file name	9-2
Get current path	9-2
Set current path	9-3
Test path available	9-3
Deleting a file or directory	9-3
Renaming a file or directory	9-4
Getting file or directory status	9-4
Setting file or directory status	9-4
Getting device status	9-5
Getting file system status	9-5
Making a new directory	9-6
Opening a unique file name	9-6
Attaching a file system	9-6
Detaching a file system	9-7

Get current path by ID.....	9-7
Change directory.....	9-7
Set initial path.....	9-8
Set file date.....	9-8
Local file system changed.....	9-8
Reading media information of a local device.....	9-9
Reading a local device directly.....	9-9

10 Process Management..... 10-1

The process ID and names.....	10-1
Process scheduling.....	10-1
Process control.....	10-2
Process ID and process table address.....	10-2
Terminate and Kill.....	10-2
Getting the current process ID.....	10-2
Getting a process ID by name.....	10-3
Getting a process priority.....	10-3
Setting a process priority.....	10-3
Getting the owning process.....	10-3
Creating a process.....	10-4
Creating a task.....	10-4
Resuming a process.....	10-5
Suspending a process.....	10-5
Killing a process.....	10-6
Registering termination.....	10-6
Terminating a process.....	10-6
Panicking a process.....	10-7
Getting a process name by ID.....	10-7
Renaming a process.....	10-7
Finding all processes.....	10-7
Watching all exits.....	10-8
Panicking the current process.....	10-8
Copying data from a process by ID.....	10-8
Copying strings from a process by ID.....	10-9
Copying data to a process by ID.....	10-9

11 Date and Time Management..... 11-1

Absolute and relative times.....	11-1
Waiting to a given time.....	11-1
Sleeping for tenths of a second.....	11-2
Sleeping for system clock ticks.....	11-2
Getting the system time.....	11-2
Setting the system time.....	11-2
Converting system time to day seconds.....	11-3
Converting day seconds to system time.....	11-3
Converting day seconds to date.....	11-3
Converting date to day seconds.....	11-3
Number of days in a month.....	11-4
Week day number.....	11-4
Name of day.....	11-4
Name of month.....	11-4
Week number.....	11-5
Abbreviated name of day.....	11-5
Abbreviated name of month.....	11-5

12 Conversion Management 12-1

Unsigned integer to buffer	12-1
Unsigned long integer to buffer	12-1
Integer to buffer	12-1
Long integer to buffer.....	12-2
Convert arguments to buffer	12-2
String to unsigned integer	12-2
String to unsigned long integer	12-2
String to integer	12-3
String to long integer	12-3
Floating point number to buffer.....	12-4
String to float.....	12-5

13 Long Integer Management 13-1

Comparing two long integers	13-1
Long integer multiplication.....	13-1
Long integer division	13-1
Compare two unsigned long integers.....	13-2
Unsigned long integer multiplication	13-2
Unsigned long integer division.....	13-2
Unsigned long integer random number.....	13-3

14 Floating Point Number Handling 14-1

Comparing two Floats	14-1
Multiplying two floats	14-1
Dividing floats	14-1
Adding two Floats	14-2
Subtracting Floats	14-2
Negating a Floats	14-2
Convert Float to a signed long.....	14-2
Convert Float to unsigned long.....	14-2
Convert Float to a signed integer.....	14-3
Convert Float to unsigned integer.....	14-3
Convert signed long to Float	14-3
Convert signed integer to Float.....	14-3
Convert unsigned integer to Float.....	14-3

15 Floating Point Function Interface..... 15-1

Arcsine of a float.....	15-1
Arctangent of a float	15-1
Cosine of a float	15-1
Exponentiation of a float	15-2
Zero fractional part of a Float.....	15-2
Natural logarithm of a float.....	15-2
Logarithm of a float	15-2
Modulo of a float.....	15-3
Power of two Floats.....	15-3
Float random number	15-3
Sine of a Float.....	15-3
Square root of a float.....	15-4
Tangent of a float.....	15-4

16 Character Management..... 16-1

Character is a digit.....	16-1
Character is a hexadecimal digit.....	16-1
Character is printable.....	16-1
Character is alphabetic.....	16-1
Character is alphabetic or digit.....	16-2
Character is upper case.....	16-2
Character is lower case.....	16-2
Character is space.....	16-2
Character is punctuation.....	16-2
Character is graphic.....	16-3
Character is control.....	16-3
Characters to upper case.....	16-3
Characters to lower case.....	16-3
Characters to folded characters.....	16-3

17 Buffer Management..... 17-1

Copying one buffer to another.....	17-1
Swapping the contents of two buffers.....	17-1
Comparing one buffer with another.....	17-1
Comparing one buffer with another folded.....	17-2
Locating a character in a buffer.....	17-2
Locating a character in a buffer folded.....	17-2
Finding a sub-buffer in a buffer.....	17-3
Finding a sub-buffer in a buffer folded.....	17-3
Matching a wild card buffer.....	17-3
Matching a wild card buffer folded.....	17-4
Justifying a buffer.....	17-4

18 String management..... 18-1

Copying one string to another.....	18-1
Copying one string to another folded.....	18-1
Converting a string to folded.....	18-1
Capitalising a string.....	18-1
Comparing one string with another.....	18-1
Comparing one string with another folded.....	18-2
Matching a wild card string.....	18-2
Matching a wild card string folded.....	18-2
Locating a character in a string.....	18-3
Locating a character in a string folded.....	18-3
Locating a character in reverse.....	18-3
Locating a character in reverse folded.....	18-3
Finding a substring in a string.....	18-4
Finding a substring in a string folded.....	18-4
Length of a string.....	18-4
Validating a system name.....	18-5

19 General management..... 19-1

Version numbers.....	19-1
Getting the operating system version number.....	19-1
Getting the ROM version number.....	19-1
Getting the system LCD type.....	19-2
Getting the system cold start reason.....	19-2
Getting the operating system data segment.....	19-2

Getting the country data	19-2
Setting the country data.....	19-2
Getting the O/S data.....	19-3
Getting error text	19-3
Dummy service	19-3
Generic file name parser	19-3
Setting deferred mode	19-4
Notify by text	19-4
Notify be error number	19-5
Hooking the notify interface	19-5
Unhooking the notify interface	19-6
Getting system ram size	19-6
Getting the command line.....	19-6
Getting the sound flags.....	19-7
Setting the sound flags	19-7
Making sound with the piezo	19-7
Marking a process as active.....	19-7
Marking a process as non-active.....	19-8
Getting operating system text	19-8
Getting notify state.....	19-8
Setting notify state	19-9
Getting the auto switch off time.....	19-9
Setting the auto switch off time	19-9
Capturing an interrupt.....	19-9
Releasing an interrupt	19-10
Getting the language code	19-10
Getting suffix text	19-11
Getting the am and pm text	19-11
Getting the battery type	19-11
Setting the battery type.....	19-11
Generating a CRC.....	19-12
Interrupt by number	19-12
Getting environment variable.....	19-12
Setting environment variable.....	19-12
Deleting environment variable	19-13
Finding environment variable	19-13
Getting string environment variable	19-14
Setting string environment variable.....	19-14
Deleting string environment variable	19-14
Finding string environment variable.....	19-14
Hooking the alarm interface	19-15
Unhooking the alarm interface	19-15
Getting the pid of the alarm server	19-15
Resetting the auto switch off timer	19-15
Controlling on events.....	19-16
Get state for auto-switch-off if mains present.....	19-16
Enable/disable auto switch-off if mains present	19-16
20 Database File Management.....	20-1
File structure.....	20-1
Buffering.....	20-2
Index Table.....	20-2
End of file record	20-2
Number of records.....	20-3
Opening a database file	20-3
Closing a database file.....	20-4
Flushing a database file.....	20-4
Trashing a database file.....	20-4
Copying down a DBF record	20-5
Compressing a database file	20-5
Copying a database file	20-5
Getting the size of a DBF	20-6

Reading a DBF extended header.....	20-7
Writing a DBF extended header	20-7
Reading a DBF descriptive record	20-7
Writing a DBF descriptive record.....	20-8
Getting the DBF version number.....	20-8
Reading an absolute DBF record	20-8
Reading and sensing an absolute DBF record	20-9
Read the next DBF record	20-9
Read the previous DBF record.....	20-9
Read the first DBF record.....	20-10
Read the last DBF record.....	20-10
Append a DBF record.....	20-10
Erasing a DBF record.....	20-11
Updating a DBF record	20-11
Finding a DBF record.....	20-12
Sensing the current DBF record number.....	20-13
Counting the number of DBF records	20-13
Finding a DBF record by field	20-13

21 Hardware Management.....21-1

Switching on the combo	21-1
Switching off the combo.....	21-1
Switching on the SSDs.....	21-1
Switching off the SSDs.....	21-1
Setting bits Asic2 register 1.....	21-2
Clearing bits Asic2 register 1	21-2
Reading Asic2 register 1	21-2
Writing Asic2 register 1	21-2
Setting bits Asic2 register 2	21-2
Clearing bits Asic2 register 2	21-3
Reading Asic2 register 2	21-3
Writing Asic2 register 2	21-3
Setting bits Asic2 register 3.....	21-3
Clearing bits Asic2 register 3	21-3
Reading Asic2 register 3	21-4
Writing Asic2 register 3	21-4
Selecting a serial channel	21-4
Sending a serial null frame.....	21-4
Switching off.....	21-4
Exiting to DOS	21-5
Capturing the combo subsystem	21-5
Freeing the combo subsystem	21-5
Getting a channel.....	21-5
Freeing a channel.....	21-6
Getting the power supply type	21-6
Getting supplies status.....	21-6
Getting supplies warnings	21-6
Changing the LCD contrast.....	21-6
Getting current LCD contrast	21-7
Setting backlight control	21-7
Getting backlight control.....	21-7
Operating the backlight.....	21-7
Scanning state of all keys	21-8
Switching on the combo in input mode.....	21-10
Getting additional power supply data.....	21-10
Hardware Management update	21-10
Reset the battery status	21-10
Enable/disable reset on recharge.....	21-10
Return battery information	21-11
Relog the SSDs.....	21-11
Set the IR power level.....	21-11
Sense the current tick count.....	21-11

Sense the expansion port state	21-12
Enable Honda connector power	21-12
Disable Honda connector power	21-12

Appendix A Interrupt and Function numbers A-1

Introduction	A-1
Alphabetical list of functions.....	A-2
Numerical list of functions	A-14
Additional Interrupt and Function numbers.....	A-27
Alphabetical list of extra functions	A-27
Numerical list of extra functions.....	A-27

Appendix B - Environment Variables B-1

PLIB	B-1
EM\$.....	B-1
Window server	B-1
\$WS_FL.....	B-1
\$WS_FNTS.....	B-2
\$WS_IF.....	B-2
\$WS_SD	B-2
\$WS_SF, \$WS_SF2 and \$WS_SF4	B-3
HWIM	B-3
M\$V	B-3
D\$X	B-3
L\$X.....	B-3
Printing.....	B-4
P\$D.....	B-4
P\$F	B-4
P\$S	B-4
P\$M.....	B-5
P\$P	B-5
P\$PP	B-5
P\$SP	B-5
P\$Z	B-5
P\$IP	B-5
P\$PX.....	B-5
Calculator application	B-6
C\$CALC.....	B-6
M\$0M0 to M\$9M9.....	B-6
Tips application	B-6
TW\$S.....	B-6
World application	B-7
W\$C	B-7
W\$R.....	B-7
Spell/Thesaurus.....	B-7
SP\$DRV	B-7
SP\$OPT	B-7
WP\$SPEL	B-7
WP\$THES	B-7
3Fax application.....	B-8
F\$X.....	B-8
F\$XM	B-8
F\$XP.....	B-8
Email applications	B-8
MAIL\$ST	B-8
Workabout	B-9
S\$SVER.....	B-9
C\$P@	B-9
C\$PA to C\$PZ	B-9
C\$P£	B-9
C\$P\$.....	B-9

CHAPTER 1

INTRODUCTION

This manual describes access at assembly language level to the services provided by the EPOC operating system. Access to these services is also supplied by the C function calls described in the *PLIB Reference manual*.

System services

All access to the services provided by the Epoc/Os is through the 80C86 software interrupt function INT XX. There are two flavours of interrupt services as follows:

- Single service.
- Multi service.

The single service interrupts are provided for commonly used services and those services which need to be executed with the least overhead.

Single service interrupts

These interrupts are invoked as follows:

```
INT XXH
```

where XX is the interrupt number in hexadecimal.

Multi service interrupts

These interrupts are invoked as follows:

```
MOV AH, ZZH  
INT XXH
```

where XX is the interrupt number in hexadecimal and the ZZ is the function number of the service required, also in hexadecimal.

It is worth remembering that all multi function interrupts will require the use of the AH register.

Calling conventions

Regardless of whether a service is through a single service interrupt or a multi service interrupt the same calling conventions are obeyed by all services.

- All registers except AX are preserved, unless they contain return values. AX is always assumed to be a scratch register by the services.
- If a service can return an error then it will signify an error condition by setting the carry flag.

- If a service returns with the carry flag set then the error will be in the AL register, and any other registers which would normally have contained the return results will be indeterminate unless otherwise stated.
- The error value returned in the AL register will be negative.
- Under normal circumstances processes will execute with the DS,ES and SS registers all pointing to the same segment. In this case the information regarding segment, register pairs in the documentation is irrelevant. However if this is not the case then the documented segment, register pairs must be obeyed.
- The services provide access to resources through handles, which are 16 bit integer values. All valid handles are guaranteed to be positive, non-zero, and even in value.
- Handles are passed in the BX register wherever possible and return values are always in the AX register.
- The state of the direction flag is preserved by services and can be in any state prior to calling the service. The interrupt flag is preserved by all services and no service enables interrupts if they were not already enabled before the call.
- If a service is called with an argument which is programmatically incorrect then the calling process will be immediately terminated. Many services do not return an error. In these cases it is foolhardy to rely on the setting of the carry flag, as it is indeterminate.
- Where a 32 bit value is required in a register pair the register pair will be shown as XX:YY. In this case XX is the most significant word and YY the least significant word.

Documentation conventions

In the following chapters, the documentation conventions are as follows:

- A shaded bar marks the start of the description of a service.
- Within the shaded bar and on the left is the name of the service. Single service interrupts are indicated by being preceded by a bullet point.
- Within the shaded bar and on the left will be seen the I symbol. This is to indicate that the service is available in Version 3 and upwards. These services are also supported on the HC but may do nothing.
- Within the shaded bar and on the right is a short description of the service.
- After the shaded bar follows all the registers which are input to the service. If there are no input registers this will be indicated by the word None.
- After the input registers description will come the section on return values. If the service does not return a value this will be indicated by RETURN: None. If it can return a value but not an error condition then RETURN: will be followed by the output registers and a description of their contents. If it can return a value and an error then RETURN: Carry clear will be followed by the output registers and a description of their contents, and RETURN: Carry set followed by the error returns.
- After the return values will come a list of the panics which could be generated by the service. If there are no panics then PANIC: None. will be shown on one line, otherwise the panic list will be enumerated.
- After the PANIC section follows a description of the function of the service.
- Any constants, structures or macros which are in include files are shown in the mono typeface.
- Where a 32 bit value is required in a register pair the register pair will be shown as XX:YY. In this case XX is the most significant word and YY the least significant word.

Include file epocdefs.inc

This include file is provided to ease the task of coding in the JPI assembler. It contains the interrupt numbers for all the services available in the operating system and equates for the offsets for many of the commonly used structures.

For multi service interrupts the name of the service to place in the AH register is as recorded in this documentation prefixed by Nm, and the name of the interrupt is XXXXManager, whereby XXXX is the multi service group. Thus to call the `ProcKill` service (where Proc is XXXX):

```
mov ah, NmProcKill
int ProcManager
```

The single service interrupts are just called by the same name as in the documentation. Thus to call the `StringCopy` service:

```
int StringCopy
```


CHAPTER 2

SEGMENTED MEMORY MANAGEMENT

Memory segment names

Segment names are zero terminated strings of up to eight characters followed by an optional period and three further characters. Examples of valid names are as follows:

- NOTES
- NUMBERS.DAT
- DATASEG.01

Paragraphs

The size of memory segments is expressed in paragraphs.

A paragraph contains sixteen (16) bytes. Paragraphs are also used in setting the value of the 80C86 segment registers CS,DS,ES and SS.

The maximum amount of memory which can be addressed by the 80C86 is 10000H paragraphs, i.e. 1 Mbyte. Thus the highest addressed memory in the 80C86 is at paragraph FFFFH.

Permanent segments

Associated with every segment is an access count which allows EPOC to determine the number of processes which have the segment open. As long as a segment's access count is non zero the segment cannot be deleted from memory.

Creating or opening a segment will automatically increase the access count, while closing will decrease the access count. In order to avoid a call to the SegDelete service, the operating system will automatically delete a segment when, after a call to SegClose, the access count is zero. Thus create followed by close will result in the segment being discarded after the call to close.

In order to generate a permanent segment in memory it must first be created and, while it is still open, i.e. before the call to close, the SegLock service must be called. This simply increments the access count so that after the close the access count will not be zero and hence the segment will not be discarded.

To remove a permanent segment, the segment must be opened and then a call to SegUnLock must be made. This will decrement the access count so that when the close is requested the access count will fall to zero and the segment will be discarded.

Directly accessing segments

Although the SegCopyTo and SegCopyFrom services are provided to allow access to memory segments it is often necessary to manipulate the data in the segment directly. The following code fragment can be used to gain access to a segment's base address.

```
GenDataSegment    ; Get the o/s data space in ES
MOV BX, SegHandle ; Get the segment handle
MOV ES, ES:[BX]   ; Get the base of the segment
    . . . .      ; Access the segment
CLI               ; Disable interrupts
PUSH DS           ; Save a copy of DS
POP ES            ; Recover ES
STI               ; Re-enable interrupts
```

Once the segment register is loaded then the operating system will keep it pointing to the right place even if segments are moved around. If the segment is bigger than 64K then the base of the segment can be added to in the following manner assuming ES has the segment base loaded.

```
CLI               ; Disable interrupts
MOV AX, ES        ; Get the value from ES
ADD AX, somevalue ; Adjust the base value
MOV ES, AX        ; Put the new base back in ES
STI               ; Re-enable interrupts
```

Of course DS can be used as well as ES.

In both of the above examples, interrupts must be disabled while the contents of the segment registers are being changed, as a pre-emptive context switch may occur and the segment that ES was pointing to could be moved.

SegFreeMemory Size of available segmented memory

None

RETURN:

AX

Available segmented memory in paragraphs.

PANIC: None

Returns the amount, in paragraphs, of currently unused addressable segmented memory.

Since the memory in machines containing more than 512 kilobytes of RAM is bank-switched, this function will never return a value larger than 32768 (corresponding to 512 kilobytes of RAM).

The value returned should be treated with some caution, as the amount of available memory, in a multi-tasking environment, is a dynamic function of the memory requests of all the currently running processes.

SegCreate Create a memory segment

AL	CreateSegmentLow - allocate in low memory. CreateSegmentHigh - allocate in high memory. CreateSegmentDevice - allocate in device memory. CreateSegmentLocked - allocate in low memory but do not log ownership of the segment.
ES:BX	Pointer to the memory segment name.
CX	Initial size of the memory segment in paragraphs.
RETURN: Carry clear	
AX	Memory segment handle.
RETURN: Carry set	
NoMemoryErr	Not enough memory to satisfy the request.
NoSegmentsErr	No memory segment handles are available.
ExistsErr	A memory segment of the requested name already exists.
NameErr	The requested name is invalid.
PANIC:	
PanicSeg1	Requested size was negative.
PanicSeg2	AL was not one of CreateSegmentLow, CreateSegmentHigh, CreateSegmentDevice or CreateSegmentLocked.

Create a memory segment with the name and size requested. The memory segment created is not initialised in any way and will contain random data.

The create service returns a handle to the created memory segment in the AX register. The returned handle allows access to the contents of the segment using the SegCopyTo and SegCopyFrom services.

The memory segment is automatically opened after being created, and should be closed when no longer required with SegClose. If the process exits or is panicked, then the memory segment will be automatically closed and if the resulting access count is zero, then it will be deleted by the Supervisor.

The initial size in CX must be positive, (i.e. in the range 0000h to 7fffh). No single memory segment may be greater than 7fffh paragraphs in size. AL determines the method by which the service will attempt to allocate the memory segment. The methods are as follows:

- CreateSegmentHigh will result in the memory segment being created above all other currently allocated memory segments. All other segments will not be moved in response to this request.
- CreateSegmentLow is provided for future expansion and currently has the same effect as CreateSegmentHigh.
- CreateSegmentDevice will result in the memory segment being created between all other device segments and the normal segments created using the above two parameters. This will result in all devices being held while memory is moved and then resumed. All normal segments will be moved up in memory to make room. This service is called by the File Server when loading dynamic device drivers and should not be used by normal applications.
- CreateSegmentLocked is the same as CreateSegmentHigh with the exception that no process owns the created segment. Like CreateSegmentDevice this parameter is used by various kernel services in the operating system and should not be used by normal applications.

SegDelete **Delete a memory segment**

ES:BX Pointer to the memory segment name.

RETURN: Carry clear

 Success

RETURN: Carry set

 NotExistsErr The requested memory segment does not exist.

 InUseErr The requested memory segment is still in use.

 NameErr The memory segment name is invalid.

PANIC: None

Delete the memory segment identified by the name pointed to by ES:BX.

If the specified memory segment is open to any other process, an InUseErr will be returned.

SegOpen **Open a memory segment**

ES:BX Pointer to the memory segment name.

RETURN: Carry clear

 AX Memory segment handle.

RETURN: Carry set

 NotExistsErr The requested memory segment does not exist.

 AlreadyOpenErr The requested memory segment is already open to this process.

 NameErr The memory segment name is invalid.

PANIC: None

Open the memory segment identified by the name pointed to by ES:BX.

The open service returns a handle to the opened memory segment in the AX register. The returned handle allows access to the contents of the segment using the SegCopyTo and SegCopyFrom services.

The Supervisor keeps track of which memory segments are opened by a process, and if a request is made to open a memory segment which is already open, the error AlreadyOpenErr will be returned. If the process terminates before closing the segment, the Supervisor will close the segment on behalf of the process.

There is no limit to the number of memory segments which may be opened by a process at any one time.

SegClose **Close a memory segment**

BX The memory segment handle to be closed.

RETURN: Carry clear

 Success

RETURN: Carry set

 NotOpenErr The memory segment is not open to this process.

PANIC:

 PanicSeg3 BX is not a valid memory segment handle.

Closes an open memory segment by its handle. The handle must be one returned from the SegCreate or SegOpen services. If the memory segment was not previously opened by the process, then the service will return NotOpenErr.

SegCloseLockedOrDevice Close a locked or device segment

BX The locked or device segment handle to be closed.

RETURN: None

PANIC:

PanicSeg3 BX is not a valid memory segment handle.

Closes a locked or device memory segment by its handle.

This service is used by the operating system to manage segments which are not owned by any process, i.e. a permanent segment left locked or a device segment.

The difference between this service and the SegClose service is that for SegClose the segment must have previously been opened with SegCreate or SegOpen, whereas for this service the segment does not need to be open. Thus, closing a segment which has not been opened will decrement the access count, making it zero which will then delete the segment. i.e. this service is short hand for calling SegOpen, SegUnLock, SegClose.

SegLock Lock a memory segment

BX The memory segment handle to be locked.

RETURN: None

PANIC:

PanicSeg3 BX is not a valid memory segment handle.

Locks an open memory segment by its handle.

The handle must be one returned from the SegCreate or SegOpen services. There is no limit to the number of times this service may be called, but it must be balanced by an equal number of calls to the SegUnLock service.

SegUnLock Unlock a memory segment

BX The memory segment handle to be unlocked.

RETURN: None

PANIC:

PanicSeg3 BX is not a valid memory segment handle.

Unlocks an open memory segment by its handle.

The handle must be one returned from the SegCreate or SegOpen services. There is no harm in unlocking a segment which is already unlocked although, if this is done inadvertently, the segment could be deleted by another process.

SegSize Size of a memory segment

BX The memory segment handle.

RETURN:

AX Memory segment size in paragraphs.

PANIC:

PanicSeg3 BX was not a valid memory segment handle.

Returns the size of an open memory segment.

The size returned is the size of the memory segment in paragraphs. The handle must be one returned from the SegCreate or SegOpen services.

SegAdjustSize **Adjust the size of a memory segment**

BX The memory segment handle.
CX The new memory segment size in paragraphs.

RETURN: Carry clear

 Success

RETURN: Carry set

 NoMemoryErr Not enough memory to satisfy the request.

PANIC:

 PanicSeg1 Requested size was negative.

 PanicSeg3 BX was not a valid memory segment handle.

Adjust the size of an open memory segment.

The handle must be one returned from the SegCreate or SegOpen services. CX must be positive (i.e. in the range 0000h to 7fffh) and represents the new size of the memory segment in paragraphs. Setting CX to zero will discard all the memory allocated to the memory segment but will not delete the segment itself.

SegFind **Find all segments**

BX The find handle.
ES:DI Pointer to a wild card match string.
DS:SI Pointer to the buffer to receive the name of the found segment.

RETURN: Carry clear

 AX The find handle for the next find.

RETURN: Carry set

 NotExistsErr No more segments found.

PANIC:

 PanicSeg3 BX was not a valid find handle.

Finds all the segments running which match the wild card string pointed to by DI.

The first time this service is called, BX should be set to zero; the first segment will be found. After a find, this service returns the find handle in the AX register. The find handle must be supplied on the next call to this service to find the next segment running.

No memory is used by this service and it can be abandoned at any time without taking any further action. The wild card string must always be supplied in DI and should be the same between calls to this service. The buffer pointed to by SI should be MaxNameESize+2 in size.

SegCopyTo **Copy to a memory segment**

BX The memory segment handle.
CX The number of bytes to copy.
DS:SI The source of the data, in the current process, to be copied.
DX:DI The target offset in the memory segment.

RETURN: None

PANIC:

 PanicSeg3 BX was not a valid memory segment handle.

 PanicSeg4 DX:DI + CX exceeded the memory segment size.

Copy data from the current process to an open memory segment.

The handle must be one returned from the SegCreate or SegOpen services. CX bytes are copied from DS:SI to the memory segment at offset DX:DI from the base of the memory segment. DX:DI is the offset in the memory segment for the target of the copy, as a 32 bit integer, with DI as the least significant word and DX as the most significant word. If DX:DI + CX is greater than the size of the memory segment, then the process will be panicked.

While the copy is being effected no other process can access the memory segment.

SegCopyFrom **Copy from a memory segment**

BX	The memory segment handle.
CX	The number of bytes to copy.
DS:SI	The target of the data, in the current process, to receive the copied data.
DX:DI	The source offset in the memory segment.

RETURN: None

PANIC:

PanicSeg3	BX was not a valid memory segment handle.
PanicSeg4	DX:DI + CX exceeded the memory segment size.

Copy data to the current process from an open memory segment.

The handle must be one returned from the SegCreate or SegOpen services. CX bytes are copied from offset DX:DI in the memory segment to DS:SI in the current process. DX:DI is the offset in the memory segment, for the source of the copy, as a 32 bit integer, with DI as the least significant word and DX as the most significant word. If DX:DI + CX is greater than the size of the memory segment, then the process will be panicked.

While the copy is being effected no other process can access the memory segment.

SegRamDiskUsed **Size of RAM disk**

None

RETURN:

AX	Size of the RAM disk in paragraphs.
----	-------------------------------------

PANIC: None

Returns the size, in paragraphs, of the RAM disk (LOC::M:).

On machines containing more than 512 kilobytes of RAM, the RAM disk will be created in an upper bank. In this case, unless the RAM disk overflows into the addressable RAM, SegRamDiskUsed will generally return zero.

The value returned should be treated with some caution as the size of the RAM disk in a multi-tasking environment is a dynamic function of the requests of all the currently running processes.

CHAPTER 3

HEAP MEMORY MANAGEMENT

Dynamics of heap memory

When a process is created using the ProcCreate service, the process is allocated an initial heap size. It is also possible, if required, not to have a heap at all by specifying an initial heap size of zero. A process is guaranteed its initial heap on start up and from then on it will have to contend with all the other processes in the system for memory.

Whenever the heap allocator runs out of memory, it will try and increase the size of the process' data segment in order to satisfy the memory request. The data segment is grown in fixed sizes dependent on a parameter held for each individual process. This value is initialised on process creation with the value HeapGrowByDefault. This value can be altered at any time by calling the HeapSetGranularity service.

Whenever the system runs out of segment memory it will try and compress the heap space of every running process by determining whether there is a free cell at the end of the heap. If there is, then the data segment will be shrunk, but only within the bounds of the initial heap size that the process was created with. Thus the initial heap size will be maintained throughout the life of the process.

The maximum size of the heap is a function of the size of the stack, the size of the initialised and uninitialised data areas and the largest size that the data segment can be grown. The maximum size for a data segment is 0ffeh paragraphs. As an example consider a process that has 1K of stack, 4K of initialised data and 5K of uninitialised data, then the maximum possible size of the heap would be, 0ffe0h-0400h-1000h-1400h which is 0de70h bytes.

HeapAllocateCell

Allocate heap memory

CX	Size of the cell to be allocated in bytes.
RETURN: Carry clear	
AX	Base of cell.
RETURN: Carry set	
NoMemoryErr	Not enough memory for the request.
PANIC:	
PanicHeap2	Heap is not initialized.

Allocates a cell in the process' heap memory.

The cell will be at least the size requested in CX and can possibly be bigger. The cell will always start on an even memory address and will have an even length. This service can result in the process' data segment growing in size.

HeapReAllocateCell **Re-allocate heap memory**

BX The base of the cell to be re-allocated or zero.

CX The new size of the re-allocated cell in bytes.

RETURN: Carry clear

AX Base of the re-allocated cell.

RETURN: Carry set

NoMemoryErr Not enough memory for the request.

PANIC:

PanicHeap2 Heap is not initialised.

PanicHeap4 The base of the cell is not in the heap memory area.

Re-allocates a cell in the process' heap memory.

A previously allocated cell can be changed in size by calling this service. The cell can be made larger or smaller. If the value in the BX register is zero then this service performs in exactly the same way as the HeapAllocateCell service.

If a cell is being extended then the service will attempt to do so by using any free memory existing immediately after the cell. If there is no free memory then the cell will be moved elsewhere and extended. Thus the value returned in AX will often not be the same as that passed in the BX register. i.e. do not rely on old copies of the cell base after a call to the HeapReAllocateCell service. The cell will be at least the size requested in CX and can possibly be bigger. The cell will always start on an even memory address and will have an even length.

Any changes to the cell always occur at the end of the cell. Thus if the cell is extended then it will be extended at the end and the data currently in the cell will be untouched. If the cell is shrunk then the shrink will be at the end of the cell and the data at the end of the cell will be lost. This service can result in the process' data segment growing in size.

HeapAdjustCellSize **Adjust the size of heap memory**

BX Base of cell to be adjusted.

CX Adjustment to the size of the cell in bytes.

DX Offset in the cell to make the adjustment.

RETURN: Carry clear

AX Base of cell after adjusting.

RETURN: Carry set

NoMemoryErr Not enough memory to satisfy the request.

PANIC:

PanicHeap2 Heap is not initialised.

PanicHeap3 Offset of adjust is greater than cell size.

PanicHeap4 The base of the cell is not in the heap memory area.

Adjust the size of the cell at BX, at an offset DX in the cell, by CX bytes.

The cell will be shrunk if CX is negative and grown if CX is positive. As this service can call the HeapReAllocateCell service, the cell may be moved if it is being extended. Thus the value returned in AX need not be the same as that passed in BX. Unlike the HeapReAllocateCell service BX may not be zero, i.e. the cell must already be allocated. This service can result in the process' data segment growing in size.

HeapFreeCell Free heap memory

BX Base of cell.

RETURN: None

PANIC:

PanicHeap2 Heap is not allocated.

PanicHeap4 The base of the cell is not in the heap memory area.

Free the memory cell whose base is in the BX register. The value passed in BX must be as returned from HeapAllocateCell, HeapReAllocateCell or HeapAdjustCellSize.

HeapCellSize Size of a heap cell

BX Pointer to the base of cell.

RETURN:

AX Cell size in bytes.

PANIC:

PanicHeap2 Heap is not allocated.

PanicHeap4 The base of the cell is not in the heap memory area.

Returns the size of a cell allocated in heap memory. This size is guaranteed to be even.

HeapSetGranularity Set the heap granularity

BX The new granularity in paragraphs.

RETURN: None

PANIC:

PanicHeap2 Heap is not allocated.

PanicHeap3 Attempt to set the granularity bigger than MaxHeapGrowBy.

Set the heap granularity parameter to the value in BX.

The maximum value for the heap granularity is MaxHeapGrowBy. Whenever the heap allocator runs out of memory, it will try and increase the size of the process' data segment in order to satisfy the memory request. The data segment is grown in fixed sizes dependent on a parameter held for each individual process. This value is initialised on process creation with the value HeapGrowByDefault.

HeapFreeMemory Size of available heap memory

None

RETURN:

AX Number of bytes potentially available in the heap.

BX Address of the base of the heap memory.

PANIC:

PanicHeap2 Heap is not allocated.

Returns the size of potentially available heap memory and the address of the base of the heap memory.

The size returned in AX consists of the amount of free memory in the heap that is currently available plus the amount by which the heap could be extended.

The value returned should be treated with some caution as the amount of available memory in a multi-tasking environment is a dynamic function of the memory requests of all the currently running processes.

CHAPTER 4

SEMAPHORE MANAGEMENT

SemCreate **Create a semaphore**

BX Initial semaphore count.

RETURN: Carry clear

AX Semaphore handle.

RETURN: Carry set

NoSemaphoreErr No semaphores are available.

PANIC:

PanicSem3 Requested initial count was negative.

Creates a semaphore, owned by the calling process, with an initial count as specified in BX.

A process should delete any owned semaphores before exiting but if it should exit abnormally or be panicked then the Supervisor will automatically delete it on behalf of the process.

SemDelete **Delete a semaphore**

BX The semaphore handle.

RETURN: None

PANIC:

PanicSem1 Invalid semaphore handle.

PanicSem2 Semaphore not allocated.

Deletes a semaphore identified by the handle passed in BX. The handle in BX should be the one returned from the SegCreate service. Any processes waiting on the semaphore are automatically signalled.

SemWait **Wait on a semaphore**

BX The semaphore handle.

RETURN: None

PANIC:

PanicSem1 Invalid semaphore handle.

PanicSem2 Semaphore not allocated.

Waits for a semaphore to be signalled.

If the semaphore has a count greater than 0 then Semwait will return immediately. If the semaphore has a count of 0 or less then the process will wait on the semaphore queue until the semaphore is signalled. More than one process can be waiting on a semaphore at a time, in which case the process that is released from the queue is on a first wait, first release basis.

If the semaphore that is being waited on has been deleted because the owning process has explicitly deleted it, exited or been panicked, then the waiting processes will be automatically released. There is no way of determining that this has happened other than co-operation between the processes sharing the semaphore.

SemSignalOnce **Signal once**

BX The semaphore handle.

RETURN: None**PANIC:**

PanicSem1 Invalid semaphore handle.

PanicSem2 Semaphore not allocated.

Signals a semaphore once.

If the signalled semaphore has a count less than zero then the first process waiting on the semaphore will be released and a re-schedule will take place. If the count is zero or positive then the count will just be incremented.

SemSignalMany **Signal more than once**

BX The semaphore handle.

CX The number of signals.

RETURN: None**PANIC:**

PanicSem1 Invalid semaphore handle.

PanicSem2 Semaphore not allocated.

PanicSem4 CX not greater than or equal to 1.

Signals a semaphore by the count in CX, which should never be negative.

If the signalled semaphore has a count less than zero then up to CX waiting processes will be released and a re-schedule will take place. If the count is zero or positive then the count will just be incremented by the value in CX.

SemSignalOnceNoReSched **Signal once without re-schedule**

BX The semaphore handle.

RETURN: None**PANIC:**

PanicSem1 Invalid semaphore handle.

PanicSem2 Semaphore not allocated.

Signals a semaphore once with no re-schedule.

If the signalled semaphore has a count less than zero then the first waiting process will be released. However, unlike SemSignalOnce, a re-schedule will not take place. This guarantees that the call to this service will return to the caller immediately. If the count is zero or positive then the count will just be incremented.

If this service is used instead of SemSignalOnce or SemSignalMany then the next re-schedule will only occur on the expiration of the next time slice. This may unnecessarily delay the signalled process. In order to overcome this, a re-schedule can always be forced by calling the TimSleepForTicks service with a value of 0.

CHAPTER 5

MESSAGE MANAGEMENT

Inter process communication

The messaging subsystem is a very important part of the operating system as it allows the high speed passing of information between processes.

Any process can send a message to another process if the target process is prepared to receive messages. However, a process can only receive messages if it has previously initialised the message system by calling the `MessInit` service.

A message consists of the header as described in the structure `MessEnt` followed by a buffer whose length is specified by the process receiving the message, when it calls the `MessInit` service. The sending process has no control over how much data is sent to the receiving process.

When a process sends a message it just specifies a message type, which is a 16 bit integer, and the address of a buffer. The receiving process will get the message and the following information

- The message type as specified by the sender.
- The process id of the sender.
- The first X bytes of data from the buffer where X is the number specified to the `MessInit` service.

The actual information content of message is defined by the receiver of the message and should be limited to as small a number as possible. For example the file server and the supervisor are both communicated with by messages where the number of bytes of information is 8.

It will often be necessary for processes to send more than the amount of information allowed for by the receiving process in the message buffer. In this case all that is necessary is to send the address of the buffer and the length of the data in the message.

The receiving process can then call the `ProcCopyFromById` and `ProcCopyToById` services to either copy data to or from the sending process.

Order of message reception

When a process initialises the message system, a queue is created to hold received messages. The number of entries which can be held in the queue is specified to the `MessInit` service.

Messages arriving are queued in arrival order and are normally removed in arrival order. However messages sent by a process whose priority is greater than 0x80 will jump to the beginning of the queue.

When a process sends a message to another process, two outcomes are possible:

- There will be room in the target process' queue, in which case the message will be delivered and the process will not wait.
- There will be no room in the target process' queue, in which case the process will wait until there is room in the queue. As soon as there is room and the process gains a time slice, the message will be delivered as previously.

- Asynchronous and Synchronous capability.
- Completion status words containing `PendingErr` while the request is still outstanding.
- Synchronisation through the `IoSignal`, `IoWaitForSignal`, `IoWaitForStatus` services.

MessInit	Initialise the message system
-----------------	--------------------------------------

MessReceiveAsynchronous	Asynchronous message reception
<pre> 1 void MessReceiveAsynchronous(2 void* pContext, 3 void* pMsg, 4 void* pMsgData, 5 void* pMsgData2, 6 void* pMsgData3, 7 void* pMsgData4, 8 void* pMsgData5, 9 void* pMsgData6, 10 void* pMsgData7, 11 void* pMsgData8, 12 void* pMsgData9, 13 void* pMsgData10, 14 void* pMsgData11, 15 void* pMsgData12, 16 void* pMsgData13, 17 void* pMsgData14, 18 void* pMsgData15, 19 void* pMsgData16, 20 void* pMsgData17, 21 void* pMsgData18, 22 void* pMsgData19, 23 void* pMsgData20, 24 void* pMsgData21, 25 void* pMsgData22, 26 void* pMsgData23, 27 void* pMsgData24, 28 void* pMsgData25, 29 void* pMsgData26, 30 void* pMsgData27, 31 void* pMsgData28, 32 void* pMsgData29, 33 void* pMsgData30, 34 void* pMsgData31, 35 void* pMsgData32, 36 void* pMsgData33, 37 void* pMsgData34, 38 void* pMsgData35, 39 void* pMsgData36, 40 void* pMsgData37, 41 void* pMsgData38, 42 void* pMsgData39, 43 void* pMsgData40, 44 void* pMsgData41, 45 void* pMsgData42, 46 void* pMsgData43, 47 void* pMsgData44, 48 void* pMsgData45, 49 void* pMsgData46, 50 void* pMsgData47, 51 void* pMsgData48, 52 void* pMsgData49, 53 void* pMsgData50, 54 void* pMsgData51, 55 void* pMsgData52, 56 void* pMsgData53, 57 void* pMsgData54, 58 void* pMsgData55, 59 void* pMsgData56, 60 void* pMsgData57, 61 void* pMsgData58, 62 void* pMsgData59, 63 void* pMsgData60, 64 void* pMsgData61, 65 void* pMsgData62, 66 void* pMsgData63, 67 void* pMsgData64, 68 void* pMsgData65, 69 void* pMsgData66, 70 void* pMsgData67, 71 void* pMsgData68, 72 void* pMsgData69, 73 void* pMsgData70, 74 void* pMsgData71, 75 void* pMsgData72, 76 void* pMsgData73, 77 void* pMsgData74, 78 void* pMsgData75, 79 void* pMsgData76, 80 void* pMsgData77, 81 void* pMsgData78, 82 void* pMsgData79, 83 void* pMsgData80, 84 void* pMsgData81, 85 void* pMsgData82, 86 void* pMsgData83, 87 void* pMsgData84, 88 void* pMsgData85, 89 void* pMsgData86, 90 void* pMsgData87, 91 void* pMsgData88, 92 void* pMsgData89, 93 void* pMsgData90, 94 void* pMsgData91, 95 void* pMsgData92, 96 void* pMsgData93, 97 void* pMsgData94, 98 void* pMsgData95, 99 void* pMsgData96, 100 void* pMsgData97, 101 void* pMsgData98, 102 void* pMsgData99, 103 void* pMsgData100, 104 void* pMsgData101, 105 void* pMsgData102, 106 void* pMsgData103, 107 void* pMsgData104, 108 void* pMsgData105, 109 void* pMsgData106, 110 void* pMsgData107, 111 void* pMsgData108, 112 void* pMsgData109, 113 void* pMsgData110, 114 void* pMsgData111, 115 void* pMsgData112, 116 void* pMsgData113, 117 void* pMsgData114, 118 void* pMsgData115, 119 void* pMsgData116, 120 void* pMsgData117, 121 void* pMsgData118, 122 void* pMsgData119, 123 void* pMsgData120, 124 void* pMsgData121, 125 void* pMsgData122, 126 void* pMsgData123, 127 void* pMsgData124, 128 void* pMsgData125, 129 void* pMsgData126, 130 void* pMsgData127, 131 void* pMsgData128, 132 void* pMsgData129, 133 void* pMsgData130, 134 void* pMsgData131, 135 void* pMsgData132, 136 void* pMsgData133, 137 void* pMsgData134, 138 void* pMsgData135, 139 void* pMsgData136, 140 void* pMsgData137, 141 void* pMsgData138, 142 void* pMsgData139, 143 void* pMsgData140, 144 void* pMsgData141, 145 void* pMsgData142, 146 void* pMsgData143, 147 void* pMsgData144, 148 void* pMsgData145, 149 void* pMsgData146, 150 void* pMsgData147, 151 void* pMsgData148, 152 void* pMsgData149, 153 void* pMsgData150, 154 void* pMsgData151, 155 void* pMsgData152, 156 void* pMsgData153, 157 void* pMsgData154, 158 void* pMsgData155, 159 void* pMsgData156, 160 void* pMsgData157, 161 void* pMsgData158, 162 void* pMsgData159, 163 void* pMsgData160, 164 void* pMsgData161, 165 void* pMsgData162, 166 void* pMsgData163, 167 void* pMsgData164, 168 void* pMsgData165, 169 void* pMsgData166, 170 void* pMsgData167, 171 void* pMsgData168, 172 void* pMsgData169, 173 void* pMsgData170, 174 void* pMsgData171, 175 void* pMsgData172, 176 void* pMsgData173, 177 void* pMsgData174, 178 void* pMsgData175, 179 void* pMsgData176, 180 void* pMsgData177, 181 void* pMsgData178, 182 void* pMsgData179, 183 void* pMsgData180, 184 void* pMsgData181, 185 void* pMsgData182, 186 void* pMsgData183, 187 void* pMsgData184, 188 void* pMsgData185, 189 void* pMsgData186, 190 void* pMsgData187, 191 void* pMsgData188, 192 void* pMsgData189, 193 void* pMsgData190, 194 void* pMsgData191, 195 void* pMsgData192, 196 void* pMsgData193, 197 void* pMsgData194, 198 void* pMsgData195, 199 void* pMsgData196, 200 void* pMsgData197, 201 void* pMsgData198, 202 void* pMsgData199, 203 void* pMsgData200, 204 void* pMsgData201, 205 void* pMsgData202, 206 void* pMsgData203, 207 void* pMsgData204, 208 void* pMsgData205, 209 void* pMsgData206, 210 void* pMsgData207, 211 void* pMsgData208, 212 void* pMsgData209, 213 void* pMsgData210, 214 void* pMsgData211, 215 void* pMsgData212, 216 void* pMsgData213, 217 void* pMsgData214, 218 void* pMsgData215, 219 void* pMsgData216, 220 void* pMsgData217, 221 void* pMsgData218, 222 void* pMsgData219, 223 void* pMsgData220, 224 void* pMsgData221, 225 void* pMsgData222, 226 void* pMsgData223, 227 void* pMsgData224, 228 void* pMsgData225, 229 void* pMsgData226, 230 void* pMsgData227, 231 void* pMsgData228, 232 void* pMsgData229, 233 void* pMsgData230, 234 void* pMsgData231, 235 void* pMsgData232, 236 void* pMsgData233, 237 void* pMsgData234, 238 void* pMsgData235, 239 void* pMsgData236, 240 void* pMsgData237, 241 void* pMsgData238, 242 void* pMsgData239, 243 void* pMsgData240, 244 void* pMsgData241, 245 void* pMsgData242, 246 void* pMsgData243, 247 void* pMsgData244, 248 void* pMsgData245, 249 void* pMsgData246, 250 void* pMsgData247, 251 void* pMsgData248, 252 void* pMsgData249, 253 void* pMsgData250, 254 void* pMsgData251, 255 void* pMsgData252, 256 void* pMsgData253, 257 void* pMsgData254, 258 void* pMsgData255, 259 void* pMsgData256, 260 void* pMsgData257, 261 void* pMsgData258, 262 void* pMsgData259, 263 void* pMsgData260, 264 void* pMsgData261, 265 void* pMsgData262, 266 void* pMsgData263, 267 void* pMsgData264, 268 void* pMsgData265, 269 void* pMsgData266, 270 void* pMsgData267, 271 void* pMsgData268, 272 void* pMsgData269, 273 void* pMsgData270, 274 void* pMsgData271, 275 void* pMsgData272, 276 void* pMsgData273, 277 void* pMsgData274, 278 void* pMsgData275, 279 void* pMsgData276, 280 void* pMsgData277, 281 void* pMsgData278, 282 void* pMsgData279, 283 void* pMsgData280, 284 void* pMsgData281, 285 void* pMsgData282, 286 void* pMsgData283, 287 void* pMsgData284, 288 void* pMsgData285, 289 void* pMsgData286, 290 void* pMsgData287, 291 void* pMsgData288, 292 void* pMsgData289, 293 void* pMsgData290, 294 void* pMsgData291, 295 void* pMsgData292, 296 void* pMsgData293, 297 void* pMsgData294, 298 void* pMsgData295, 299 void* pMsgData296, 300 void* pMsgData297, 301 void* pMsgData298, 302 void* pMsgData299, 303 void* pMsgData300, 304 void* pMsgData301, 305 void* pMsgData302, 30</pre>	

5 - 2

When the message has been received then the word pointed to by BX will contain the address of the message. After the process has finished with the data in the message, the message should be returned to the queue using the `MessFree` service. If messages are not freed then eventually the message queue will be empty and any process sending a message will wait indefinitely.

MessReceiveWithWait **Synchronous message reception**

DS:BX Pointer to a word to receive the address of the message received.

RETURN: None

PANIC:

PanicMess2 Messages are not initialised.

PanicIoPending Already waiting for a message.

Waits for a message to be received.

When the message has been received then the word pointed to by DS:BX will contain the address of the message. After the process has finished with the data in the message, then the message should be returned to the queue using the `MessFree` service. If messages are not freed then eventually the message queue will be empty and any process sending a message will wait indefinitely.

MessReceiveCancel **Cancel queued message receive**

None

RETURN: None

PANIC:

PanicMess2 Messages are not initialised.

If a request to receive a message has been queued with the `MessReceiveAsynchronous` service then the request may be cancelled with the `MessReceiveCancel` service.

It is not considered an error to call this service if no request is outstanding. After this service has been called, a signal will be delivered and the status word will be changed to `CancelErr`. Note that this service can not be called to cancel a request pending from the `MessSendReceiveAsynchronous` service.

MessSend **Send messages**

BX The id of the process to receive the message.

CX The type of message.

SS:SI Pointer to the message buffer.

RETURN: Carry clear

Success.

RETURN: Carry set

NoReceiverErr The target process does not exist.

PANIC: None

Sends a message to the process whose id is specified in the BX register.

If the target process either does not exist or has not initialised messages, the service will return the `NoReceiverErr`. The message, when it arrives, will have the field `MessType` in the `MessEnt` structure set to the value passed in the CX register. The rest of the message will be copied from the buffer pointed to by SS:SI.

There is no need to specify the length of the data to be copied as it is not a function of the sending process but a function of the receiving process. If the target process' message queue is full, the sending process will be suspended until space becomes available in the queue.

MessSendReceiveAsynchronous **Send and get a reply async**

BX	The process id to receive the message.
CX	The type of message.
DI	Pointer to the status word.
SS:SI	Pointer to the message data.

RETURN: Carry clear
Success.

RETURN: Carry set
NoReceiverErr The target process does not exist.

PANIC: None

Sends a message to a process and returns without waiting for the reply from that process.

The target process is as specified by the process id in BX, the message type is in CX and the message data is pointed to by SS:SI. These parameters have the same meaning as for `MessSend`. The pointer to the status word is specified by DI.

When the target process eventually frees the message then the reply will be returned in the status word pointed to by DI and a signal will be sent to the sending process.

MessSendReceiveWithWait **Send and wait for a reply**

BX	The process id to receive the message.
CX	The type of message.
SS:SI	Pointer to the message data.

RETURN: Carry clear

AX The received data.

RETURN: Carry set

NoReceiverErr The target process does not exist.

Returned error Depends on the value returned by the target.

PANIC: None

Sends a message to a process and then waits for a reply from that process.

The target process is as specified by the process id in BX, the message type is in CX and the message data is pointed to by SS:SI. These parameters have the same meaning as for `MessSend`. The target process returns the reply when it frees the message it received with the `MessFree` service.

MessFree **Free a message**

DS:BX	Pointer to message to be freed.
CX	The value to be returned to the sender of the message.

RETURN: None

PANIC:

PanicMess2 Messages are not initialised.

Returns a received message to the message queue.

If messages are not returned to the message queue then in due course there will be no messages left in the queue and all sending processes will be suspended and only the receiver will be left running (wondering where everyone else has gone).

If a message was sent with just a `MessSend` then the return value in CX is ignored.

If a message was sent with `MessSendReceiveWithWait` or `MessSendReceiveAsynchronous` then the value in CX is the returned reply.

MessSignal Request a signal from the Supervisor

BX The process id which will trigger the signal when it terminates.
 CX The message type to be sent when the process terminates.

RETURN: Carry clear
 Success

RETURN: Carry set
 NotExistsErr The requested process does not exist.

PANIC:
 PanicMess2 Messages are not initialised.

Notifies the supervisor to send a message of the type specified in CX to the process calling this service when the process specified in BX exits or is panicked.

This service is very similar to IoRequestReset in that it is a mechanism which allows processes and servers, in particular, to perform valuable housekeeping when a process connected to a server terminates.

For example, when a process connects to the file server with the FilConnect service, the file server calls this service to register an interest in the process making the connection. When the process terminates, the file server will receive a message from the supervisor that the process has terminated and the server can then free all the resources associated with that process.

When the process specified in BX terminates, a message will be sent from the supervisor with the type requested in CX. The first word in the message buffer is the id of the terminating process and the second word contains the exit code. The exit code is split into two bytes; the least significant byte has the actual reason for the termination and the most significant byte has the type of exit. This can be one of:

- KillExit - The process was killed or terminated; by convention, if the least significant byte (i.e. the reason for the termination) was 0 then it was a non-error exit.
- PanicExit - The process was panicked.
- TaskPanicExit - A task in the process was panicked which caused the process itself to be panicked.

Having registered an interest in a process with the supervisor, this interest can be cancelled by calling the MessSignalCancel service. When the process terminates, a signal will not be sent by the supervisor.

The process must have messages initialised in order to request this service.

MessSignalCancel Cancel requested signal from Supervisor

BX The process id which will trigger the signal, when it terminates.

RETURN: Carry clear
 Success.

RETURN: Carry set
 NotExistsErr The process has no signal request logged with the supervisor.

PANIC:
 PanicMess2 Messages are not initialised.

Cancels a previously requested signal from the supervisor as set up by an earlier call to MessSignal.

BX contains the process id as passed to an earlier call to MessSignal.

The process must have messages initialised in order to request this service.

This service is unsuitable for applications that have requested more than one signal for termination of the same process. In this situation, the application should use MessSignalCancelx; this allows the message type to be identified.

?MessSignalCancelX Cancel signal from Supervisor by type

BX The process id which will trigger the signal when it terminates.

CX The message type to be sent when the process terminates.

RETURN: Carry clear

 Success.

RETURN: Carry set

 NotExistsErr The process has no signal request logged with the supervisor.

PANIC:

 PanicMess2 Messages are not initialised.

Cancels a previously requested signal from the supervisor.

The parameters to this function are exactly same as those to the `MessSignal` service which set up the signal in the first place.

The process must have messages initialised in order to request this service.

This service must be used in preference to `MessSignalCancel` in cases where two or more `MessSignal` requests are made with the same value of BX.

CHAPTER 6

DYNAMIC LIBRARY, CATEGORY AND OBJECT MANAGEMENT

Library names

All dynamic libraries have two names by which they may be referenced.

- File name.
- Internal name.

The file name of a dynamic library is just used to load the library into memory so that it can be used by other processes. The default extension for dynamic libraries is DYLIB. The file name should be used to load the library using the `LibLoad` or `LibLoadFile` services.

Once loaded, a dynamic library publishes a name by which the library is known to the system. This name just consists of the name and extension portion of the file name.

An image can also be a dynamic library as well as an image in which case if the image name is `NNNN.IMG` or `NNNN.APP` the internal name published will be the name of the code segment, i.e. `NNNN.$SC`.

LibLoad	Load a dynamic library
ES:BX	Pointer to the dynamic library file name.
CL	Zero - Don't link the library automatically. Non Zero - Link the library after loading.
RETURN: Carry clear	
AX	The dynamic library handle.
RETURN: Carry set	
NameErr	Invalid file name.
NotExistsErr	Dynamic library file does not exist.
AlreadyOpenErr	Dynamic library already loaded.
ImageErr	File is not a valid dynamic library file.
PANIC:	
PanicObj6	All external references in the library could not be resolved.
PanicFs1	Process not connected to the File Server.

Loads a dynamic library into memory for access by the other Lib services.

ES:BX points to a zero terminated file name which can be a full path name. If the dynamic library is already in memory then it will be shared and the library will not be re-loaded. This service can also invoke the LibLink service, depending on the value in CL. If CL is non-zero then the library will be linked after loading. Clearly if the library is already linked then this will be skipped.

As long as the process keeps the dynamic library open, it will stay in memory. It is desirable that a process should unload the library, using the LibUnLoad service as soon as it has finished using it. This allows memory to be freed for use by other processes.

If the process should panic or be killed before it can close the library, the supervisor will automatically perform the unload. Note that the process must be connected to the file server before this service can be called.

LibUnLoad **Unload a dynamic library**

 BX Handle for the dynamic library.

RETURN: Carry clear

 Success.

RETURN: Carry set

 NotOpenErr Process does not have the library opened.

PANIC: None

Unloads a dynamic library from memory.

A loaded dynamic library can be shared by many processes; each time a process loads it, the library access count is incremented. Unloading the library decreases the access count. If the access count is 0 after unloading the dynamic library, it will be discarded from memory.

As dynamic libraries are in memory, it is important that programs unload a library as soon as they have finished using it. If a program is terminated for any reason, then the supervisor will automatically call this service for any library still loaded.

LibLink **Link a dynamic library**

 BX Handle to the dynamic library.

RETURN: None

PANIC:

 PanicObj6 All external references in the library could not be resolved.

Links a dynamic library which is already in memory.

If the library has already been linked because another process has already loaded and linked it then this service just returns.

All external categories which are required to satisfy external references in the library to be linked must already be in memory or on the ROM disk before this service is called, otherwise PanicObj6 will be generated.

Programs, i.e. .IMG files which contain a category must request this service to link themselves before any classes can be accessed. In order to specify the SELF category, the handle BX must be 0. Normally the handle will have been returned by LibLoad or LibFind.

LibFind **Get a dynamic library handle**

 ES:BX Pointer to the dynamic library internal name.

RETURN: Carry clear

 AX The dynamic library handle.

RETURN: Carry set

 NotExistsErr The dynamic library is not loaded into memory.

PANIC: None

Given the internal name of a dynamic library, this service will return a handle to that library. The library must exist on the ROM disk or have been loaded by the program.

Having found a handle to a library, it can be used to create objects using the classes contained in that library by invoking the `LibCreateByHandle` service.

LibHandle **Get a DYL handle by number**

BX The number of the required external category.

RETURN:

AX The category handle.

PANIC:

PanicObj2 Number does not specify a valid external category.

PanicObj5 Attempt to get a handle before the SELF category has been linked.

When a category is generated, it may contain references to external categories. After calling the `LibLink` service, these external references are resolved. In order to get a handle to these external categories, this service may be called by using the number provided by the category compiler. This number will be defined as `CAT_ ...`. If this service is passed a value of 0 in **BX**, it will return 0 as the handle for the self category is also 0.

This service is used by the `LibCreate` service.

LibCreate **Create an object by category number**

BX The number of the category.

CX The number of the class within the category.

RETURN: Carry clear

AX The object handle.

RETURN: Carry set

NoMemoryErr The object could not be created.

PANIC:

PanicObj2 Number does not specify a valid external category.

PanicObj3 Number does not specify a valid class.

PanicObj5 Attempt to get a handle before the SELF category has been linked.

Two components are always required to specify a class from which an object is to be created:

- The category handle.
- The class number.

This service converts the category number passed in **BX** to a handle by calling the `LibHandle` service and then calling the `LibCreateByHandle` service.

The category number can be 0, meaning the SELF category or the number of an external category as created by the category compiler.

The class number in **CX** is as created by the category compiler. The handle to the object created is returned in **AX**, if carry is clear. The only error which can occur is an out of memory condition. The object's data will be initialised to zero by this service.

LibCreateByHandle **Create an object by category handle**

BX The handle of the category.

CX The number of the class within the category.

RETURN: Carry clear

AX The object handle.

RETURN: Carry set

NoMemoryErr The object could not be created.

PANIC:

PanicObj2	Number does not specify a valid external category.
PanicObj3	Number does not specify a valid class.
PanicObj5	Attempt to get a handle before the SELF category has been linked.

The category handle can be 0 to mean the SELF category, otherwise it must be a handle returned by LibFind, LibLoad or LibHandle.

The class number in CX is as created by the category compiler. The handle to the object created is returned in AX, if carry is clear. The only error which can occur is an out of memory condition. The object's data will be initialised to zero by this service.

LibDestroy**Destroy an object**

BX	The handle of the object to be destroyed.
----	---

RETURN: None

PANIC: None

This service will de-allocate all the memory associated with an object including any compound objects.

This service is normally called by the root class in its DESTROY method. As this is the ultimate super class, all objects are inherently able to destroy themselves.

• LibSend**Send a message**

BX	The handle of the object to receive the message.
CL	The message number to be sent.
DX	Argument 1
SI	Argument 2
DI	Argument 3

RETURN:

AX	Whatever the method returns.
----	------------------------------

PANIC:

PanicObj0	No method prepared to handle the message number in CX.
-----------	--

This service will send a message to an object.

Any arguments to the method eventually invoked should be allocated to the registers DX,SI and DI in that order. The search for a method starts in the class which was used to create the object.

• LibSendSuper**Send a superclass message**

BX	The handle of the object to receive the message.
CL	The message number to be sent.
DX	Argument 1
SI	Argument 2
DI	Argument 3

RETURN:

AX	Whatever the method returns.
----	------------------------------

PANIC:

PanicObj0	No method prepared to handle the message number in CX.
-----------	--

This service will send a message to an object. Unlike LibSend, however, the search for a method starts in the object's immediate superclass. This is the way in which inheritance works.

• LibSendExact Send a direct message

DS:[DatEClassHandle]	The category handle.
DS:[DatEClassPtr]	The class number.
BX	The handle of the object to receive the message.
CL	The message number to be sent.
DX	Argument 1
SI	Argument 2
DI	Argument 3

RETURN:

AX	Whatever the object returns.
----	------------------------------

PANIC:

PanicObj0	No method prepared to handle the message number in CX.
-----------	--

This service will send a message to an object. Unlike LibSend, however, the search for the method starts at the class specified in DS:[DatEClassHandle],DS:[DatEClassPtr].

• LibEnterSend Send message under protection of Enter

BX	The handle of the object to receive the message.
CL	The message number to be sent.
DX	Argument 1
SI	Argument 2
DI	Argument 3

RETURN:

AX	Whatever the method returns or is passed to LibLeave.
----	---

PANIC:

PanicObj0	No method selected by CX.
-----------	---------------------------

This service will send a message to an object.

The first two arguments are mandatory; the other arguments depend on the method being selected.

The search for a method starts in the class which was used to create the object. Unlike LibSend the method called is entered as if it had been called with LibEnter. Thus any LibLeave calls from the method itself or any of the routines which it may call can be made to return to the caller of this service.

LibOpen Open a multi library file

ES:BX	Pointer to the file name.
RETURN: Carry clear	
AX	The open multi library file handle.
RETURN: Carry Set	
ImageErr	Not a multi library file.
Error	Any error that FilOpen can give.

PANIC:

PanicFs1	Process not connected to the File Server.
----------	---

This service opens a file which contains multiple libraries.

Most commonly this is an image file to which the multiple libraries have been added using the *OsMake* tool.

The returned handle in AX can be used with LibLoadFile to load libraries. The handle is a normal file handle as if the file had been opened with FilOpen and, as such, when access to the file is no longer required it can be closed with FilClose.

LibLoadFile **Load a multiple dynamic library**

BX	Multiple library file handle.
CH	The index of the library to be loaded.
CL	Zero - Don't link the library automatically. Non Zero - Link the library after loading.

RETURN: Carry clear

Success.

RETURN: Carry set

AlreadyOpenErr Dynamic library already loaded.

ImageErr File is not a valid dynamic library file.

PANIC:

PanicObj6 All external references in the library could not be resolved.

PanicFs1 Process not connected to the File Server.

Loads a dynamic library from an already opened multiple library file into memory for access by the other library services.

The handle in BX must have been provided by the LibOpen service. CH selects the library to be loaded from the file in the order that libraries were added to the image originally. Thus 0 will load the first library from the file, 1 the second and so on.

If the dynamic library is already in memory then it will be shared and the library will not be re-loaded. This service can also invoke the LibLink service, depending on the value in CL. If CL is non zero then the library will be linked after loading. Clearly if the library is already linked then this will be skipped.

As long as the process keeps the dynamic library open, it will stay in memory. It is desirable that a process should unload the library, using LibUnLoad, as soon as it has finished with it so that the memory can be used by other processes. If the process should panic or be killed before it can close the library, the supervisor will automatically perform the unload.

Note that the process must be connected to the file server before this service can be called.

LibReClass **Reclass an object by number**

BX	The number of the category.
CX	The number of the class within the category.
DI	Pointer to the object to be reclassified.

RETURN: None

PANIC:

PanicObj2 Number does not specify a valid external category.

PanicObj3 Number does not specify a valid class.

PanicObj5 Attempt to get a handle before the SELF category has been linked.

There are always two components to specifying a class in order to reclass an object.

- The category handle.
- The class number.

This service converts the category number passed in BX to a handle by calling the LibHandle service and then calling the LibReClassByHandle service. The category number can be 0 meaning the SELF category or the number of an external category as created by the category compiler. The class number in CX is as created by the category compiler.

LibReClassByHandle**Reclass an object by handle**

BX	The handle of the category.
CX	The number of the class within the category.
DI	Pointer to the object to be reclassified.

RETURN: None

PANIC:

PanicObj2	Handle does not specify a valid external category.
PanicObj3	Number does not specify a valid class.

This service changes the class to which an object belongs to that specified by BX and CX.

It is up to the user to ensure that the new class that the object will belong to will behave appropriately. BX must be the handle on an already loaded category. The class number in CX is as created by the category compiler.

LibCopy**Copy data from a category**

BX	The category number.
CX	Number of bytes to copy.
DI	Pointer to a buffer to receive the data.
SI	Offset of the source in the category segment.

RETURN: None

PANIC:

PanicObj2	Number does not specify a valid external category.
-----------	--

This service will copy data from a category segment into the data segment of the current process. The category is specified by the number in BX, which will be converted to a handle with LibHandle. The offset in the category is specified by the value in SI.

• LibEnter**Enter a control region**

AX	The routine to call.
BX	Argument 1.
CX	Argument 2.
DX	Argument 3.
SI	Argument 4.
DI	Argument 5.

RETURN:

AX	Whatever the routine called or is passed to the LibLeave service.
----	---

PANIC: None

This service will call a routine in a transparent fashion, while marking this call as entering a control region. A subsequent call to the LibLeave service will immediately exit from this service. Control regions can be nested.

• LibLeave**Leave a control region**

AX	The value to return.
----	----------------------

RETURN: None

PANIC:

PanicEnter1	No corresponding call to LibEnter.
-------------	------------------------------------

This service can be called to exit from a control region.

The value in AX will be returned in AX to the original caller of the LibEnter service which set up the control region.

All images must have a call to LibLeave at offset 0x0C in the code segment of the image.

• LibExitSend	Return from a method
----------------------	-----------------------------

AX

The value to return.

RETURN: None**PANIC:** None

This service can be called to exit from a method.

Normally this routine is not called directly and is used by the C libraries to effect the exit from `p_send()` etc. All images must have a call to the `LibSendExit` service at offset `0x0C` in the code segment of the image.

CHAPTER 7

DEVICE MANAGEMENT

Device names

Device names are zero terminated strings made up in one of the following ways:

- 3 characters followed by a colon
- 3 characters, period and 3 characters followed by a colon

Examples of valid names are as follows:

- TTY:
- TTY.AS5:
- FIL:

Device drivers

There are two kinds of device drivers supported by Epoc/Os.

- LDD - logical device driver.
- PDD - physical device driver.

LDDs provide a consistent interface through the I/O services to application programs. Thus there is only one RS232 LDD available which will handle all the RS232 devices in the system.

PDDs provide the interface by which LDDs access the hardware. Thus there can be a Psion custom RS232 PDD and a standard 16450 UART PDD in the system controlling very different hardware devices. However because all access to the hardware is through the LDD and then to the PDD both devices will behave in exactly the same way as far as application programs are concerned.

LDDs are opened using the IoOpen service, while PDDs are opened using the DevOpenPDD service. An application should never try and access the hardware by opening a PDD directly.

LDDs all have three character names, i.e. TTY, PAR, while PDDs have a seven character name, i.e. TTY.AS5, TTY.UAR, where the first part of the name represents the owning LDD.

DevOpenPDD

Open a physical device driver

ES:BX	Pointer to the PDD name.
RETURN: Carry clear	
AX	Device handle

RETURN: Carry set

NotExistsErr	The specified PDD does not exist.
Error	Error returned by the device driver.

PANIC:

PanicLib1	The named device driver was not a PDD.
-----------	--

Open the physical device driver identified by the name pointed to by ES:BX.

The device must already be installed either as a built in device driver or as a dynamic device driver. The DevLoadPDD service should be used to load a PDD device driver into memory if it is not already loaded. The open service returns a device handle which can be used by the DevGetPDDAddress service to get the far address of the PDD strategy entry point.

The functions that can be performed by a PDD are documented on a per device driver basis.

DevGetPDDAddress **Get the PDD entry point**

BX	The PDD device handle.
----	------------------------

RETURN:

BX:AX	The PDD entry point
-------	---------------------

PANIC:

PanicDev1	The device handle is not for a PDD or is invalid.
-----------	---

This service returns the address of the PDD's entry point.

The PDD's device handle is passed in BX. The entry point is returned in the BX:AX registers, where BX is the segment address and AX is the offset. This allows a far call to be made to the PDD. Opening the PDD will return the device handle for the PDD. After a resume, the entry point of the PDD should be re-loaded as the code may have moved.

DevInstall **Install a device driver**

BX	LHDSeg.
CX	LHDPtr.
DX	LDDSignature - LDD device type. PDDSignature - PDD device type.

RETURN: Carry clear

Success.

RETURN: Carry set

DeviceErr	Not a valid device driver.
-----------	----------------------------

PANIC: None

This function is reserved for use by the operating system and should never be used by applications. It is called automatically by the DevLoadLDD and DevLoadPDD services. After an LDD has been loaded its install vector will be executed. This is not the case for a PDD.

DevHold **Hold all device drivers**

CL	DevHoldNormal - Device memory is being moved. DevHoldPowerDown - The machine is switching off. DevHoldPowerFail - The machine is switching off after a power fail condition has been detected.
----	--

RETURN: None**PANIC:** None

This function will call all the LDDs in the system in turn to suspend all interrupt driven devices which are currently active. They can subsequently be re-activated by calling the DevResume service. It is strongly recommended that this service is not called and that its use is restricted to the operating system. It is called by the DevLoadLDD and DevLoadPDD services.

DevResume**Resume all device drivers**

None

RETURN: None**PANIC:** None

This function will call all the LDDs in the system in turn to resume all interrupt driven devices which were previously de-activated by DevHold. It is strongly recommended that this service is not called and that its use is restricted to the operating system. It is called by the DevLoadLDD and DevLoadPDD services.

DevLoadLDD**Load a logical device driver**

ES:BX

Pointer to the name of a file containing the logical device driver.

RETURN: Carry clear

Success

RETURN: Carry set

NotExistsErr

Logical device driver file does not exist.

ImageErr

Logical device driver file does not have the correct format for a LDD.

PANIC:

PanicFs1

The process is not connected to the file server.

Load the logical device driver from the file identified by the name pointed to by ES:BX.

If no extension is given for the file name then a file extension of .LDD will be assumed. If a full path name is not given then the current directory will be searched for the LDD file. After the LDD is loaded it may then be accessed using the I/O services in the normal way.

DevLoadPDD**Load a physical device driver**

ES:BX

Pointer to the name of a file containing the physical device driver.

RETURN: Carry clear

Success

RETURN: Carry set

NotExistsErr

Physical device driver file does not exist.

ImageErr

Physical device driver file does not have the correct format for a PDD.

PANIC:

PanicFs1

The process is not connected to the file server.

Load the physical device driver from the file identified by the name pointed to by ES:BX.

If no extension is given for the file name then a file extension of .PDD will be assumed. If a full path name is not given then the current directory will be searched for the PDD file. After the PDD is loaded it may then be accessed using the I/O services in the normal way.

DevDelete**Delete a device driver**

ES:BX

Pointer to the name of the device driver to be deleted.

DX

LDDSignature - LDD device type.

PDDSignature - PDD device type.

RETURN: Carry clear

Success

RETURN: Carry set

NotExistsErr

The device driver is not currently loaded.

NotSupportedErr

The device driver is a ROM device driver and cannot be deleted.

InUseErr

The device driver is currently open and cannot be deleted.

Error number

Error returned by the device driver.

PANIC:

PanicFs1 The process is not connected to the file server.

Delete the requested device driver of the name pointed to by ES:BX.

The name here should be the same name which would be passed to the open services. Only device drivers in RAM can be deleted by this service. DX specifies whether an LDD or a PDD is to be deleted. In both cases the remove vector will be executed before the device is deleted from memory. Calling this service will result in the DevHold and DevResume services being called.

DevRemove	Remove a device driver
------------------	-------------------------------

ES:BX Pointer to the name of the device driver to be removed.

DX LDDSignature - LDD device type.
 PDDSignature - PDD device type.

RETURN: Carry clear

 Success.

RETURN: Carry set

NotExistsErr The device driver is not currently loaded.

NotSupportedErr The device driver is a ROM device driver and cannot be removed.

InUseErr The device driver is currently open and cannot be removed.

Error Error returned by the device driver.

PANIC: None

Remove the requested device driver of the name pointed to by ES:BX.

The name here should be the same name which would be passed to the open services. Only device drivers in RAM can be removed by this service. DX specifies whether an LDD or a PDD device is to be removed. In both cases the remove vector will be executed before the device is deleted from memory. Calling this service will result in the DevHold and DevResume services being called.

This service is used by the File Server to call the remove vector of a device driver before deleting it from memory. This service is restricted and should not be used by normal programs. Use the DevDelete service instead.

DevQueryUnits	Query the number of units
----------------------	----------------------------------

ES:BX Pointer to the name of the device to be queried.

RETURN: Carry clear

AX The number of units supported by the device driver.

RETURN: Carry set

NotExistsErr Device driver not found.

PANIC: None

Queries the number of units supported by a device driver.

This service only applies to LDDs. If the device driver is found, the number of units supported is returned in the AX register. A value of -1 implies an unlimited number of units. This is what is returned by the FIL: device driver as it can open a large number of files.

DevFind	Find all devices
----------------	-------------------------

BX The find handle.

DX LDDSignature - LDD device type.
 PDDSignature - PDD device type.

ES:DI Pointer to a wild card match string.

DS:SI Pointer to the buffer to receive the name of the found device.

RETURN: Carry clear

AX The find handle for the next find.

RETURN: Carry set

NotExistsErr No more devices found.

PANIC:

PanicDev1 Invalid device find handle.

Finds all the devices installed of the type specified by DX which match the wild card string pointed to by ES:DI.

DX is either PDDSignature to find PDD devices or LDDSignature to find LDD devices.

The device name written to DS:SI is a zero terminated string and MaxNameESize+2 bytes should be allowed for in the buffer. The first time this service is called, BX should be set to zero; the first device will be found. After a find, this service returns the find handle in the AX register. This find handle must be supplied on subsequent calls to this service to find the next device installed.

No memory is used by this service and it can be abandoned at any time without taking any further action. The wild card string must always be supplied in ES:DI, and should be the same between calls to this service.

DevVector

Call a device vector

CL The vector number to call.
CH Moved to AH before the device vector is called.
DI Device handle.
BX Argument 1.
DX Argument 2.
SI Argument 3.

RETURN: Carry clear

AX The result from the device driver.

RETURN: Carry set

AL The error from the device driver.

PANIC:

PanicLib2 The device did not support the vector in CL.

Given a handle to a device in DI, then the devices vector number in CL will be called. The value in CH is transferred to AH and the values in BX,DX,SI are passed to the device driver untouched.

CHAPTER 8

INPUT OUTPUT MANAGEMENT

Devices and files

As far as the I/O services are concerned, devices and files are exactly the same. The documentation just refers to devices, but wherever device is read, it may be interchanged for files.

Thus the `IoOpen` service is used to open devices such as the RS232 device driver and files such as `A:LETTERS.DOC`.

.WVE Sound file format

Series 3a sound files contain a 32-byte header and a byte stream of digital sound. Such files normally have a `.wve` extension. During playback the byte stream is sampled and played at 8000 bytes per second (12-bit sound is converted to 8-bit using A-Law encoding).

In C, the file header is represented by the following struct (defined in *epoc.h*)

```
#define SignatureSize 16
#define ALawSignature "ALawSoundFile**"

typedef struct
{
    TEXT Signature[SignatureSize];
    UWORD Version;
    ULONG Samples;
    UWORD SilenceInTicks;
    UWORD Repeats;
    UWORD Spare[3];
} SndFile;
```

This header is written and read by the Series 3a sound services described below. The meanings of the items in the `SndFile` struct are:

Signature

The 16-byte (including the zero terminator) string `"ALawSoundFile**"`.

Version

The Series 3a sound file version number as a 4-digit hexadecimal number of the form `XYZZ`, where `X` is the major release number, `YY` is the minor release number and `Z` is normally the hexadecimal digit `F`. This is the same version format as used by `GenVersion`, for example.

Samples

The number of bytes following the header. This must always be the size of the file less the 32 bytes for the header. Dividing this by 8000 gives the duration of the sound in seconds.

SilenceInTicks

The number of system ticks of silence appended to each repeat on playback (in practice, you get at least 2 ticks between repeats).

Repeats

The number of times to repeat the sound on playback (0 and 1 are the same).

Spare

Reserved for future use.

A system tick is a 1/32th of a second, equivalent to 250 samples.

IoAsynchronous**Asynchronous I/O**

AL	I/O function number.
BX	I/O handle.
DS: CX	Pointer to argument 1.
DS: DX	Pointer to argument 2.
DS: DI	Pointer to status word.

RETURN: Carry clear

AX	Result from the device driver.
----	--------------------------------

RETURN: Carry set

Error	Depends on the device driver.
-------	-------------------------------

PANIC:

PanicIo1	Invalid I/O channel.
PanicLib1	Invalid library handle.
PanicLib2	Invalid library function number.

Request I/O services from device drivers.

The I/O is asynchronous, i.e. the call will return immediately even if the I/O request has not completed. The value in AL can be one of the constants which start with the prefix IoFunc. What functions are supported by a channel depends on the device driver originally opened.

CX and DX are pointers to two parameters the meaning of which depends on the function being requested and the device driver that is open on the channel. DI is a pointer to a word which will contain the I/O completion status when the I/O completes. While the request is still being serviced this value will be PendingErr.

If this service returns without an error then the device driver will always signal completion with one of the IoSignal, IoSignalByPid or IoSignalByPidNoResched services. Thus, when waiting for the I/O request to complete, one of the IoWaitForSignal or IoWaitForStatus services should be requested, instead of polling the status word.

IoAsynchronousNoError**Asynchronous I/O - no error reporting**

AL	I/O function number.
BX	I/O handle.
DS: CX	Pointer to argument 1.
DS: DX	Pointer to argument 2.
DS: DI	Pointer to status word.

RETURN: None

PANIC:

PanicIo1	Invalid I/O channel.
PanicLib1	Invalid library handle.
PanicLib2	Invalid library function number.

This is exactly the same service as `IoAsynchronous` except that errors in starting the I/O request, instead of being reported by setting the carry flag, are reported by setting the status word and signalling.

This service is provided as a convenience routine since many applications find it easier to handle starting errors as completion errors. In many cases there is no difference in the meaning for a given error and, therefore, no difference in action to be performed.

IoWithWait

Synchronous I/O

AL	I/O function number.
BX	I/O handle.
DS: CX	Pointer to argument 1.
DS: DX	Pointer to argument 2.

RETURN: Carry clear

AX	Result from the device driver.
----	--------------------------------

RETURN: Carry set

Error	Depends on the device driver.
-------	-------------------------------

PANIC:

PanicIo1	Invalid I/O channel.
PanicLib1	Invalid library handle.
PanicLib2	Invalid library function number.

Request I/O services from device drivers synchronously.

This service will return when the I/O request has completed. The value in AL can be one of the constants which start with the prefix `IoFunc`.

The functions supported by a channel depends on the device driver originally opened. CX and DX are pointers to two parameters whose meaning depends on the function being requested and the device driver that is open on the channel.

IoRoot

Chain to root device

DS: BX	I/O handle.
DS: SI	Pointer to the request packet.

RETURN: Carry clear

AX	Result from device driver.
----	----------------------------

RETURN: Carry set

<code>IoInvalidErr</code>	I/O function requested is not valid for this device.
---------------------------	--

PANIC:

PanicIo2	Application requested the device to panic.
----------	--

If a root device driver does not support a particular I/O function then it should call this service (attached device drivers should call the `IoSuper` function).

DS:SI points to a structure of the type `RqEnt`. This service provides code which will respond to the following function requests:

- `IoFuncPanic`
- `IoFuncClose`
- `IoFuncCancel`
- `IoFuncAttach`
- `IoFuncDetach`

If the function number requested is not one of the above, then this service will return the `IoInvalidErr`. This service should only be used by root device drivers.

IoSuper Chain to superclass device

DS:BX I/O handle.
DS:SI Pointer to the request packet.

RETURN: Carry clear

AX Result from device driver.

RETURN: Carry set

`IoInvalidErr` I/O function requested is not valid for this device.

Error Number Depends on the device driver.

PANIC:

`PanicIo1` I/O channel invalid.

Requests I/O service from the device driver attached next on the I/O channel.

This service allows an attached device driver to pass on I/O requests that the attached driver does not understand to the superclass device driver.

DS:SI points to a structure of the type `RqEnt`. Note that the parameters are identical to those for the `IoRoot` service as the last device in the chain (root device driver) will eventually pass on any unknown requests to the `IoRoot` service. This service should only be used by attached device drivers.

IoWaitForSignal Wait for I/O completion

None

RETURN: None

PANIC:

`PanicIo1` Channel invalid.

`PanicIo3` Handler invalid.

When an application wants to wait for any of the outstanding asynchronous I/O requests to complete, it should call this service.

After this service returns, one of the status words associated with the outstanding I/O requests is guaranteed to have changed from `PendingErr` to either a return result (i.e. 0 or positive) or another error. Even though this routine has no parameters it can still panic; after the signal has been received, this service will invoke any attached I/O handlers which have the ability to cause the process to be panicked.

If an application wants to poll the status words, it cannot just look at the values in the status words. It must first give any handlers a chance to complete the I/O request. As a call to this service can suspend the process until one of the I/O requests completes, it is better to call the `IoYield` service which is guaranteed to return after allowing the handlers to execute.

IoWaitForStatus Wait for specific request to complete

DS:DI Pointer to status word of I/O request.

RETURN: None

PANIC:

`PanicIo1` Channel invalid.

`PanicIo3` Handler invalid.

Waits for a specific I/O request to be completed.

As long as the status word pointed to by DI has the value `PendingErr`, this service will continue waiting. As soon as the value changes then this service will return.

IoYield**Poll for completion**

None

RETURN: None**PANIC:**

PanicIo1 Channel invalid.

PanicIo3 Handler invalid.

Update the status words of any I/O requests which may have completed.

If an application wants to poll the status words, it cannot just look at the values in the status words. It must first give any handlers a chance to complete the I/O request. This service provides the mechanism to do this without suspending the process.

IoSignal**Signal completion**

None

RETURN: None**PANIC:** None

Signals the completion of an outstanding asynchronous I/O request to the current process.

All calls to a device driver through the `IoAsynchronous` service, which are started successfully are balanced by a call to a signal service when they complete. Equally the application that requested the I/O must balance this signal with a call to the `IoWaitForSignal` service.

IoSignalByPid**Signal completion by process ID**

BX The process ID to be signalled.

RETURN: Carry clear

Success.

RETURN: Carry set

NoExistsErr The process does not exist.

PANIC: None

Signals the completion of an outstanding asynchronous I/O request to the process whose ID is in the BX register.

All calls to a device driver through the `IoAsynchronous` service which are started successfully are balanced by a call to a signal service when they complete. Equally the application that requested the I/O must balance this signal with a call to the `IoWaitForSignal` service.

This service should not be used by device drivers in the interrupt part of their code because the interrupt service routine could be running under any process and a possible re-schedule that could occur would cause a stack build up.

IoSignalByPidNoReSched**Signal completion, no reschedule**

BX The process ID to be signalled.

RETURN: Carry clear

Success.

RETURN: Carry set

NoExistsErr The process does not exist.

PANIC: None

Signals the completion of an outstanding asynchronous I/O request to the process whose ID is in the BX register.

If this service is used instead of `IoSignalByPid` then the next re-schedule will only occur on the next time slice expiry. This may unnecessarily delay the signalled process. In order to overcome this, a re-schedule can always be forced by calling the `TimSleepForTicks` service with a value of 0.

All calls to a device driver through the `IoAsynchronous` service which are started successfully are balanced by a call to a signal service when they complete. Equally the application that requested the I/O must balance this signal with a call to the `IoWaitForSignal` service.

This service should only be used by device drivers in the interrupt part of their code because the interrupt service routine could be running under any process and a re-schedule is to be avoided.

IoAddHandler **Add a handler**

AL	The vector number of the handler.
BX	Identifying data to be passed to the handler.
DS, ES	Must be pointing at the user process data space

RETURN: Carry clear

AX	Handler handle.
----	-----------------

RETURN: Carry set

<code>IoAllocErr</code>	Not enough memory to add the handler.
-------------------------	---------------------------------------

PANIC: None

Adds a wait handler to the list of I/O handlers for that process.

Whenever the `IoWaitForSignal` service receives a signal, it will call all the active handlers for the process. The handler will be passed the channel handle passed in the BX register. When the handler is added, it will initially be disabled and in order to activate it, the device driver should call `IoEnableHandler`. Only device drivers should use this service.

IoRemoveHandler **Remove a handler**

BX	The handler handle.
DS, ES	Must be pointing at the user process data space

RETURN: None

PANIC:

<code>PanicIo3</code>	Invalid handler handle.
-----------------------	-------------------------

Removes a previously added wait handler from the process' list of I/O handlers.

The handle in BX must be a handle returned from the `IoAddHandler` service. Only device drivers should use this service.

IoEnableHandler **Enable/Disable a handler**

BX	The handler handle.
CL	0 - To disable the handler. 1 - To enable the handler.
DS, ES	Must be pointing at the user process data space

RETURN: None

PANIC:

<code>PanicIo3</code>	Invalid handler handle.
-----------------------	-------------------------

When a particular handler is first added to a list of handlers for a particular process, it is initially disabled. Thus, any signals which arrive will not cause the handler to be called. In order to enable the handler, this service must be called with CL set to 1. The handler indicates whether it wishes to remain enabled or be disabled on exit (see the section on device handlers).

If asynchronous requests are cancelled, the handler may no longer be required to be called. The handler can be disabled by setting CL to 0.

Devices should only keep their handlers enabled when they have outstanding requests to deal with; system performance would be significantly degraded if all handlers were permanently enabled.

Only device drivers should use this service.

IoRequestReset**Request a reset**

BX	Device handle.
CX	Identifying data.

RETURN: None**PANIC:** None

Request the Supervisor to log a reset request for a device driver.

When a channel is opened by a process, it is always possible that the process may be panicked or killed before it has a chance to close the channel. If this were to happen without the device driver knowing about it, the device would remain in use forever and the resource would be lost to the system.

This service provides a mechanism for device drivers to be notified when the process which has opened the device has been terminated. It can then cleanup as necessary and mark itself no longer in use.

The value passed in BX should be the device handle. This is passed in register DX by the device manager to the open and strategy vectors of a device driver. The value in CX is device dependent information to allow multiple unit device drivers to determine which unit should be reset. If the device is just a single unit device then it should place 0 in CX.

This service should only be called by device drivers.

IoRequestResetCancel**Cancel a requested reset**

BX	Device handle.
CX	Identifying data.

RETURN: None**PANIC:** None

Request the Supervisor to cancel a reset request for a device driver.

When a device is closed and the device driver has a reset request logged with the Supervisor, the outstanding request must be cancelled as it is no longer valid.

The value in CX should always be exactly the same as the value passed to the IoRequestReset service, even if the device driver is a single unit device.

This service should only be called by device drivers.

IoOpen**Open a device**

ES:BX	Pointer to the device name.
CX	Open mode.
DX	The I/O handle of an already opened device, if an attached driver is being opened.

RETURN: Carry clear

AX	I/O handle.
----	-------------

RETURN: Carry set

Error	Depends on the device being opened.
-------	-------------------------------------

PANIC: None

Open a device driver for I/O.

The name of the device is a zero terminated string. The open mode parameter in CX is device dependent and the relevant device driver documentation should be consulted.

The open mode parameters for the filing system device driver are the constants starting with Mode. DX need only be set if opening a device driver which is an attached driver, in which case DX must be the handle of the already opened I/O channel to which the driver must be attached.

IoClose **Close a device**

BX I/O handle.

RETURN: Carry clear

AX Result from device driver.

RETURN: Carry set

Error Depends on the device being opened.

PANIC: None

Close an open I/O channel.

A zero value can be passed in BX and the service will just do nothing. This is useful as it allows zero to indicate a closed channel, especially in clean up situations.

IoRead **Read from a device**

BX I/O handle.

DS: CX Pointer to buffer to receive the read data.

DX Number of bytes to read.

RETURN: Carry clear

AX Amount actually read.

RETURN: Carry set

Error Depends on the device being read.

PANIC: None

Read data from the device.

The data read is written to the buffer pointed to by CX. Up to DX bytes will be read. The actual amount of data read is also returned in the DX register. If no error occurs then AX will have the same value as DX. If an error does occur then AL will have the error number and DX will contain the number of bytes read before the error occurred.

IoWrite **Write to a device**

BX I/O handle.

DS: CX Pointer to buffer to be written.

DX Number of bytes to write.

RETURN: Carry clear

AX Result from device driver.

RETURN: Carry set

Error Depends on the device being written.

PANIC: None

Write data to the device.

DX bytes of data is written from the buffer pointed to by CX.

IoSeek **Seek on a device**

BX I/O handle.

CX SeekFromStart
SeekFromEnd
SeekFromCurrent
SeekRecordSense
SeekRecordSet
SeekRewind

DX Pointer to a double word containing the new position.

RETURN: Carry clear

AX Result from device driver.

RETURN: Carry set

Error Depends on the device being "seeked".

PANIC: None

Perform a seek on a device.

The type of seek to perform is as specified in the CX register and the new position is pointed to by the DX register. In most cases the seek position is a double word.

IoKeyAndMouseWithWait

Mouse and keyboard

DS:BX Pointer to a KeyEnt structure.

RETURN: None

PANIC:

PanicIo4 Another process is already waiting for an event.

Get the next keyboard or mouse event into the KeyEnt structure pointed to by BX.

The system maintains a queue of up to 16 keyboard and mouse events in a buffer. If the buffer is not empty when this service is requested then it will return immediately with the event data copied to the structure at BX. If there is no event then the requesting task will be suspended until an event occurs.

Four events are possible and more than one can be returned per call. The events are as follows:

- Mouse valid. The mouse has been activated. (i.e. touched).
- Mouse moved. The mouse has moved.
- Mouse button state. The mouse button has changed state.
- Key valid. A key has been pressed.

Which event has occurred can be determined by consulting the KeyState field in the returned structure. This service can only be requested by a task and not a process. There is usually only one task in the system, usually of the window server, which calls this service and then distributes the events to all other processes.

IoAddApplicationHandler

Add an application handler

BX Identifying data to be passed to the handler.

CX Address of the application handler.

DX Address of the application handler dispatcher.

DS, ES Must be pointing at the user process data space

RETURN: Carry clear

AX Handler handle.

RETURN: Carry set

IoAllocErr Not enough memory to add the handler.

PANIC: None.

Adds an application wait handler to the list of I/O handlers for that process.

Whenever the IoWaitForSignal service receives a signal, it will call all the active handlers for the process.

The channel handle in register BX will be passed to the application handler dispatcher in register AX while the handler's address will be passed in register CX.

The dispatcher should execute a CALL BX to run the handler and then execute a RET FAR.

When the handler is added, it will initially be disabled; in order to activate it, IoEnableApplicationHandler should be called.

IoRemoveApplicationHandler Remove an application handler

BX The handler handle.
DS, ES Must be pointing at the user process data space

RETURN: None

PANIC:

PanicIo3 Invalid handler handle.

Removes a previously added handler from the process' list of I/O handlers. The handle in BX must be a handle returned from the IoAddApplicationHandler service.

IoEnableApplicationHandler Enable/Disable application handler

BX The handler handle.
CL 0 - Disable the handler.
 1 - Enable the handler.
DS, ES Must be pointing to the user process data space

RETURN: None

PANIC:

PanicIo3 Invalid handler handle.

When a particular handler is first added to the list of handlers for the process, it is initially disabled. Thus any signals which arrive will not cause the handler to be called. In order to enable the handler, this service must be called with CL set to 1.

The handler indicates whether it wishes to remain enabled or be disabled on exit (see the section on device handlers). If asynchronous requests are cancelled, the handler may no longer be required to be called; the handler can be disabled by setting CL to 0.

Handlers should only be enabled when they have outstanding requests to deal with as system performance would be significantly degraded if all handlers were permanently enabled.

IoShiftStates Get the shift states

None

RETURN:

AX The current shift states.

PANIC: None

This service will return the current shift states.

It is valuable to enquire on the status of the CAPS lock and the NUM lock states when the system first powers up.

IoWaitForSignalNoHandler Wait for I/O completion no handlers

None

RETURN: None

PANIC:

PanicIo1 Channel invalid.

PanicIo3 Handler invalid.

This service is identical to IoWaitForSignal except that no handlers are allowed to run. This is valuable for tasks which are waiting for a signal as the normal service would run the handlers belonging to the parent process of the task, a bad mistake!

Recall that a task is a subsidiary process that shares the same data segment as the parent process; because a task can never have a handler, there is never any problem in calling this service.

IoSignalKillAsynchronous Request signal from Supervisor

BX The process ID, which will trigger the signal, when it terminates.
DI The status word to be cleared on completion.

RETURN: Carry clear
 Success

RETURN: Carry set
 NotExistsErr The requested process does not exist.

PANIC: None

Notifies the supervisor to send a signal to the process calling this service when the process specified in BX exits or is panicked.

This service is very similar to `MessSignal` in that it is a mechanism which allows processes to perform valuable housekeeping when another process terminates.

When the process specified in BX terminates, a signal will be sent from the supervisor and the status word will contain the reason code for termination.

Having registered an interest in a process with the supervisor, this interest may be cancelled by calling the `IoSignalKillCancel` service. This stops the supervisor from sending a signal on termination of that process.

IoSignalKillCancel Cancel requested signal from Supervisor

BX The process ID which will trigger the signal, when it terminates.

RETURN: Carry clear
 Success.

RETURN: Carry set
 NotExistsErr The process has no signal request logged with the supervisor.

PANIC: None

Cancels a previously requested signal from the supervisor. The status word will be updated to `CancelErr` and a signal will be sent from the supervisor.

IoNextHalfSecond Request signal on next half second

None.

RETURN:
 None.

PANIC:
 None.

Requests a signal to be sent when the next exact half second expires.

Unlike other asynchronous requests there is no status word associated with this call as it is held internally by the operating system.

Completion can be polled for in the normal way by calling `IoNextHalfSecondStatus` which will return either `PendingErr`, `FailErr` or a positive non zero integer. `PendingErr` will be returned if the request has not yet completed. `FailErr` is returned if the system time has changed or the machine has been switched off. A positive non-zero integer represents the number of half seconds since the last request was completed.

This service is NOT to be used by any applications; it is reserved for use by the window server to keep itself synchronised with the system clock with the minimum system overhead.

• IoNextHalfSecondStatusQuery completion of IoNextHalfSecond

None .

RETURN:

AX The completion status.

PANIC:

None .

This function will return either `PendingErr`, `FailErr` or a positive non-zero integer. `PendingErr` will be returned if the request has not yet completed. `FailErr` is returned if the system time has changed or the machine has been switched off. A positive non-zero integer represents the number of half seconds since the last request was completed.

This service is NOT to be used by any applications; it is reserved for use by the window server to keep itself synchronised with the system clock with the minimum system overhead.

③ IoPlaySoundW Play back sound file synchronously

BX Pointer to the sound file name.
CX The duration in ticks or zero if supplied in the file.
DX The sound volume to be used.

RETURN:

`FailErr` Sound is disabled.
Other errors File system errors in general.

PANIC: None

Play back a sound file synchronously.

The string at BX should be either the file specification of the sound file which is simply parsed with a `.wve` extension or it should begin with a `*` character followed by just the name component of the sound file.

If the string starts with a `*` character, the extension `.wve` is assumed and the service automatically hunts `ROM::` and the `\wve` directories of `M:`, `A:` and `B:` (in that order). Note that the Series 3a `ROM::` sound files have names `sys$al01.wve`, `sys$al02.wve`, etc.

CX specifies the duration in system ticks that the sound file will play. If the absolute value is shorter than the natural duration of the sound file then playback will be truncated; if greater than the natural duration then playback will be padded with trailing silence.

Note that the natural duration includes any trailing silence and the number of repeats that are specified in the file header.

DX specifies the playback volume between 0 and 5 inclusive, with 0 the loudest. On the Series 3a there are only 4 actual volume levels: (0,1), 2, 3, (4,5).

The format of the sound file header is described at the beginning of this chapter.

This service fails with `FailErr` if sound is disabled.

③ IoPlaySoundA Play back sound file asynchronously

BX Pointer to the sound file name.
CX The duration in ticks or zero if supplied in the file.
DX The sound volume to be used.
DI Pointer to the status word.

RETURN:

None .

PANIC: None

Play back a sound file asynchronously.

The string at BX should be either the file specification of the sound file which is simply parsed with a .wve extension or it should begin with a * character followed by just the name component of the sound file. If the string starts with a * character, the extension .wve is assumed and the service automatically hunts ROM:: and the \wve directories of M:, A: and B: (in that order). Note that the Series 3a ROM:: sound files have names sys\$al01.wve, sys\$al02.wve, etc.

CX specifies the duration in system ticks that the sound file will play. If the absolute value is shorter than the natural duration of the sound file then playback will be truncated; if greater than the natural duration then playback will be padded with trailing silence.

Note that the natural duration includes any trailing silence and the number of repeats that are specified in the file header.

DX specifies the playback volume between 0 and 5 inclusive, with 0 the loudest. On the Series 3a there are only 4 actual volume levels: (0,1), 2, 3, (4,5).

During playback the status word at DI contains PendingErr. On completion of playback, the status word contains the completion status which will be zero if playback completed successfully or CancelErr if cancelled using IoPlaySoundCancel or a negative error number.

The format of the sound file header is described at the beginning of this chapter.

This service fails with error FailErr if sound is disabled.

③ IoPlaySoundCancel

Cancel play back of sound file

None

RETURN:

None.

PANIC: None

Cancel playing back a sound that was initiated using IoPlaySoundA.

After a call to IoPlaySoundCancel, the completion status of IoPlaySoundA will be CancelErr.

③ IoRecordSoundW

Record sound to a file synchronously

BX Pointer to the sound file name.

CX Sound file length in 2048-byte units.

RETURN:

FailErr Sound is disabled.

VolumeErr Cannot record to a Flash SSD.

other errors File system errors in general.

PANIC: None

Record a sound to file synchronously.

The file may not be on a Flash SSD. The register BX points to a string containing the name of the sound file. By default, the extension is .wve. Sound will be recorded to this file which will be replaced if it already exists.

CX specifies the number of bytes to be recorded in 2048-byte units, excluding the 32 byte header. A value of 4 (8K) in CX corresponds to approximately one second. Before recording starts, a file of length 32+CX*2048 bytes is created so this amount of space must exist on the disk.

The format of the sound file header is described at the beginning of this chapter.

This service fails with FailErr if sound is disabled.

③ IoRecordSoundA Record sound to a file asynchronously

BX	Pointer to the sound file name.
CX	Sound file length in 2048-byte units.
DI	Pointer to the status word.

RETURN: None.

PANIC: None

Record a sound to file asynchronously.

The file may not be on a Flash SSD. The register BX points to a string containing the name of the sound file. By default, the extension is .wve. Sound will be recorded to this file which will be replaced if it already exists.

CX specifies the maximum number of bytes to be recorded in 2048-byte units, excluding the 32 byte header. A value of 4 (8K) in CX corresponds to approximately one second. Before recording starts, a file of length $32 + CX * 2048$ bytes is created so this amount of space must exist on the disk.

During recording the status word at DI contains PendingErr. On completion, the status word contains the completion status which will be zero if recording completed successfully, or CancelErr if cancelled using IoRecordSoundCancel, or a negative error number.

The format of the sound file header is described at the beginning of this chapter.

This service fails with FailErr if sound is disabled and volumeErr on attempting to record to a Flash SSD.

③ IoRecordSoundCancel Cancel recording sound to a file

None

RETURN: None.

PANIC: None

Cancel recording sound to a file that was initiated using IoRecordSoundA. The sound file is truncated to the actual length that was recorded before cancellation.

After a call to IoRecordSoundCancel, the completion status of IoRecordSoundA will be CancelErr.

Input Output Management update

The majority of the additional EPOC I/O management system services described in this section were introduced for the Series 3c and Siena.

With the exception of the HC, all the services are, in principle, available on any machine that contains EPOC version 3.90F or later. On an HC with a suitable version of EPOC, all the functions described in this section should generate an `E_GEN_NSUP` error.

Some services require the presence of hardware that is not built into all machines in the SIBO range. If the relevant hardware is not present on a particular machine, calling the service will either have no effect or return an error of `E_GEN_NSUP`. The descriptions of such services contain a list of the machines on which they are intended to be used.

IoPlaySoundAO **Asynchronous partial sound file replay**

BX	Pointer to full file specification
CX	Required duration, in ticks
DX	Playback volume
DI	Pointer to a status word
SI	Offset, in ticks

RETURN:

DI	Pointer to status word
----	------------------------

PANIC: None

This service is only available on Series 3c machines.

Play back a selected section of a sound file asynchronously.

The string at BX should be either the file specification of a sound file, which is simply parsed with a `.wve` file extension, or it should begin with a `*` character, followed by just the name component of the sound file.

If the string starts with a `*` character, the extension `.wve` is assumed and the service automatically hunts `ROM::` and the `\wve` directories of `M:`, `A:` and `B:` (in that order). Note that the Series 3a and Series 3c ROM sound files have names `sys$al01.wve`, `sys$al02.wve`, etc.

DX specifies the playback volume, which should be in the range 0 to 5 inclusive, with 0 being the loudest. There are only four actual sound levels: (0,1) 2, 3, (4,5).

SI specifies the offset, in ticks, from the start of the sound to the point at which replay will start.

CX specifies the duration in system ticks for which the sound will play. If this duration is less than the remaining natural duration of the sound from the specified starting point, playback will be truncated to the specified duration. If the specified duration exceeds the remaining natural duration of the sound then playback will be padded with trailing silence.

Note that the natural duration includes any trailing silence and the number of repeats that are specified in the file header

During playback the status word at DI contains `PendingErr`. On completion of playback, the status word contains the completion status, which will be zero if playback completed successfully, `CancelErr` if playback was cancelled by the use of `IoPlaySoundCancel`, or a negative error number.

This service fails with `FailErr` if sound is disabled.

CHAPTER 9

FILE MANAGEMENT

The file server

The file server is a separate process which runs whenever an application process needs to access any of the filing systems.

The server program provides a mechanism to serialise access to the resources of the filing systems in the same way as the window server serialises access to the screen, keyboard and mouse.

By default a process does not have any access to the filing system. If the process requires access, then it must tell the file server program of its presence by using the `FilConnect` service. Once established, the connection cannot be broken throughout the life of the process.

Many of the services such as `DevopenPDD` make requests to the file server and, as such, require that the process has first made connection with the server. If the connection has not been made then the process will be panicked.

FilConnect	Connect to the file server
------------	----------------------------

AL	Zero - Synchronous request. Non-zero - Asynchronous request.
DX	If AL is non-zero then DX points to the status word.

RETURN: None

PANIC:

PanicFile1	Already connected to file server.
------------	-----------------------------------

Connects the process to the file server.

Any process requiring access to the filing system must connect to the server by using this service before calling any of the other services in the File manager or in the I/O manager.

It is only necessary to connect to the server once; the connection can be established at any time.

FileExecute	Execute an image file
-------------	-----------------------

AL	Zero - Synchronous request. Non-zero - Asynchronous request.
DS:DX	If AL is non-zero then DX points to the status word.
ES:BX	Pointer to the path name of the image file to be executed.
DS:CX	Pointer to the command line argument.
DS:DI	Pointer to a word to receive the handle of the executed process.

RETURN: Carry clear

Success

RETURN: Carry set

NameErr	Invalid file name.
DeviceErr	Invalid device specification.
ImageErr	Invalid image file format.
NotExistErr	Image file does not exist.
NoProcessErr	No process slots available.
Others	Various other I/O errors.

PANIC: None

Creates a process in memory from the contents of the image file specified by the BX register.

The process created will have the same name as the name of the image file. Before the new process will run, it must be resumed using the ProcResume service, passing it the handle as returned in the word pointed to by the register DI. If a process of the same name as that of the file name is already running in memory then the new process will share the code segment of the running process and only the data segment of the new process will be loaded from the image file.

If CX is zero then no command line will be passed to the executed process. If CX is non-zero then it should point to a byte counted command buffer. The address will be passed to the executed routine. The buffer is byte counted so that binary data can be passed in the command line. The maximum length of the command line is MaxCommandBuffer bytes. The executed process will be given a command line allocated in its heap which consists of the full path used to find the image file followed by the data pointed to by CX.

FilParse

Parse a file name

AL	Zero - Synchronous request. Non-zero - Asynchronous request.
DX	If AL is non-zero then DX points to the status word.
BX	Input file name string.
CX	Related file specification string.
DI	Pointer to a buffer to receive the parsed file name.
SI	Pointer to a FullParseEnt buffer to receive the parse information block.

RETURN: Carry clear

Success

RETURN: Carry set

NameErr	Invalid file name.
DeviceErr	Invalid device specification.

PANIC: None

The parse service provides an equivalent service to that in PLIB.

This service requires that the process be connected to the file server. The current directory and device will provide the defaults if a complete related file specification is not provided.

FilPathGet

Get current path

AL	Zero - Synchronous request. Non-Zero - Asynchronous request.
DX	If AL is non-zero then DX points to the status word.
BX	Pointer to buffer to receive the path.

RETURN: Carry clear

Success

RETURN: Carry set

AL The error number.

PANIC: None

This service gets the current path of the process. This path is the default for all file services if no path is specified.

FilPathSet **Set current path**

AL Zero - Synchronous request.
Non-zero - Asynchronous request.
DX If AL is non-zero then DX points to the status word.
BX Pointer to the new path name.

RETURN: Carry clear

Success

RETURN: Carry set

AL The error number.

PANIC: None

This service sets the current path of the process. The path must be valid at the time of setting.

FilPathTest **Test path available**

AL Zero - Synchronous request.
Non-zero - Asynchronous request.
DX If AL is non-zero then DX points to the status word.
BX Pointer to the path name to be tested.

RETURN: Carry clear

Success

RETURN: Carry set

AL The error number.

PANIC: None

This service tests that the path referenced in register BX is available; in other words it checks that it exists.

FilDelete **Delete a file or directory**

AL Zero - Synchronous request.
Non-zero - Asynchronous request.
DX If AL is non-zero then DX points to the status word.
BX Pointer to the name of the file or directory to be deleted.

RETURN: Carry clear

Success

RETURN: Carry set

NameErr Invalid file name.
ExistsErr Directory is not empty.
LockedErr File or directory protected from delete.
NotExistsErr File or directory does not exist.
DirErr Invalid path specification.

PANIC: None

Delete the file whose name is pointed to by BX.

The name can specify a file or a directory. If a directory is specified then it must be empty before it can be deleted.

FilRename **Rename a file or directory**

AL	Zero - Synchronous request. Non-zero - Asynchronous request.
DX	If AL is non-zero then DX points to the status word.
BX	Pointer to the old name of the file or directory to be renamed.
CX	Pointer to the new name of the file or directory to be renamed.

RETURN: Carry clear

Success

RETURN: Carry set

NameErr	Invalid file name.
ExistsErr	New file or directory already exists.
NotExistsErr	Old file or directory does not exist.
DirErr	Invalid path specification.
DeviceErr	Attempt to rename across devices.

PANIC: None

Rename the file whose name is pointed to by BX to the name pointed to by CX. The name can specify a file or a directory.

FilStatusGet **Get file or directory status**

AL	Zero - Synchronous request. Non-zero - Asynchronous request.
DX	If AL is non-zero then DX points to the status word.
BX	Pointer to the name of the file or directory.
CX	Pointer to a FileStatusEnt structure.

RETURN: Carry clear

Success

RETURN: Carry set

NameErr	Invalid file name.
NotExistsErr	File or directory does not exist.
DirErr	Invalid path specification.

PANIC: None

Get the status of the file whose name is pointed to by BX. The name can specify a file or a directory.

The status information is written to a FileStatusEnt structure pointed to by CX. This structure is defined as:

```
FileStatusEnt  struc
FileVersionNo  dw    ?
FileAtt        dw    ?
FileSize       dd    ?
FileModDate    dd    ?
FileSpare      db    4    dup (?)
FileStatusEnt  ends
```

and is equivalent to the PLIB P_INFO struct.

FilStatusSet **Set file or directory status**

AL	Zero - Synchronous request. Non-zero - Asynchronous request.
DX	If AL is non-zero then DX points to the status word.
BX	Pointer to the name of the file or directory.
CX	Mask of status bits to set.
DI	Value of status bits.

RETURN: Carry clear

Success

RETURN: Carry set

NameErr	Invalid file name.
NotExistsErr	File or directory does not exist.
DirErr	Invalid path specification.

PANIC: None

Sets the status of the file whose name is pointed to by BX according to the bits set in CX and DI.

If a bit is set in the CX register the appropriate bit in DI will be used. The name can specify a file or a directory. The bits correspond to the FileAtt flags.

FilStatusDevice

Get device status

AL	Zero - Synchronous request. Non-zero - Asynchronous request.
DX	If AL is non-zero then DX points to the status word.
BX	Pointer to the name of the device.
CX	Pointer to a DeviceStatusEnt structure.

RETURN: Carry clear

Success

RETURN: Carry set

DeviceErr	Invalid device name.
-----------	----------------------

PANIC: None

Get the status of the device whose name is pointed to by BX and write this information into the DeviceStatusEnt structure pointed to by CX. This structure is defined as:

```
DeviceStatusEnt    struc
DeviceVersionNo    dw      ?
DeviceMediaType     dw      ?
DeviceIsRemovable   dw      ?
DeviceStatusSize    dd      ?
DeviceStatusFree     dd      ?
DeviceStatusName     db      MaxVolumeName dup (?)
DeviceBatteryState   dw      ?
DeviceSpare          db      16 dup (?)
DeviceStatusEnt     ends
```

where MaxVolumeName has the value 32. This structure is equivalent to the PLIB P_DINFO struct.

FilStatusSystem

Get file system status

AL	Zero - Synchronous request. Non-zero - Asynchronous request.
DX	If AL is non-zero then DX points to the status word.
BX	Pointer to the name of the file system.
CX	Pointer to a NodeStatusEnt structure.

RETURN: Carry clear

Success

RETURN: Carry set

AL	Error value.
----	--------------

PANIC: None

Write the status of the file system whose name is pointed to by BX into the NodeStatusEnt structure pointed to by CX. This structure is defined as:

```
NodeStatusEnt    struc
NodeVersionNo    dw      ?
NodeType          dw      ?
NodeSupportsFormat dw    ?
NodeStatusSpare  db      26 dup (?)
NodeStatusEnt    ends
```

and is equivalent to the PLIB P_NINFO struct.

FilMakeDirectory **Make a new directory**

AL	Zero - Synchronous request. Non-zero - Asynchronous request.
DX	If AL is non-zero then DX points to the status word.
BX	Pointer to new directory name.

RETURN: Carry clear

Success

RETURN: Carry set

ExistsErr	Directory already exists.
DeviceErr	Invalid device name.
NameErr	Invalid path name.

PANIC: None

Makes a new directory using the name pointed to by BX.

If the path to the directory does not exist then the full path will be made.

FilOpenUnique **Open a unique file name**

AL	Zero - Synchronous request. Non-zero - Asynchronous request.
DX	If AL is non-zero then DX points to the status word.
ES:BX	Pointer to related file specification.
CX	Open mode.

RETURN: Carry clear

AX I/O handle.

RETURN: Carry set

FailErr	Failed to get a unique file name.
Errors	Dependent on the filing system.

PANIC: None

Open a unique file name.

The related file name pointed to by BX is used only for its path in order to generate the unique file name. If a unique file name is successfully opened then the name of the file is written back to the buffer pointed to by BX. The open mode parameter in CX should only specify the format and access fields as the file will always be opened with ModeCreate and ModeUpdate.

FilSystemAttach **Attach a file system**

AL	Zero - Synchronous request. Non-zero - Asynchronous request.
DX	If AL is non-zero then DX points to the status word.
BX	Pointer to a file system PDD.

RETURN: Carry clear

Success

RETURN: Carry set

AL

Error value.

PANIC: None

Attaches a file system PDD to the file server.

FilSystemDetach

Detach a file system

AL

Zero - Synchronous request.

Non-zero - Asynchronous request.

DX

If AL is non-zero then DX points to the status word.

BX

Pointer to a file system name.

RETURN: Carry clear

Success

RETURN: Carry set

AL

Error value.

PANIC: None

Detaches a file system from the file server. The built in file systems cannot be detached.

FilPathGetById

Get current path by ID

AL

Zero - Synchronous request.

Non-zero - Asynchronous request.

DX

If AL is non-zero then DX points to the status word.

BX

Process ID whose path is required.

CX

Pointer to buffer to receive the path.

RETURN: Carry clear

Success

RETURN: Carry set

AL

The error number.

PANIC: None

This service gets the current path of the process whose ID is specified in BX. The path is copied into the buffer pointed to by CX.

FilChangeDirectory

Change directory

AL

Zero - Synchronous request.

Non-zero - Asynchronous request.

DX

If AL is non-zero then DX points to the status word.

BX

Pointer to file or directory name string.

CX

Pointer to buffer to receive the new file/dir name string.

DI

Mode of changing directory.

SI

Pointer to sub directory name if DI is FileChangeDirSubdir.

RETURN: Carry clear

Success

RETURN: Carry set

AL

The error number.

PANIC: None

This service allows a path name to be manipulated. No assumptions about file names are made. The root, parent or sub directory may be requested.

FilSetInitialPath **Set initial path**

AL	Zero - Synchronous request. Non-zero - Asynchronous request.
DX	If AL is non-zero then DX points to the status word.
BX	Pointer to the new initial path name.

RETURN: Carry clear

Success

RETURN: Carry set

AL	The error number.
----	-------------------

PANIC: None

This service sets the initial path given to a process when it connects to the file server. This service has no effect on processes already connected.

FilSetFileDate **Set file date**

AL	Zero - Synchronous request. Non-zero - Asynchronous request.
DX	If AL is non-zero then DX points to the status word.
BX	Pointer to the file path name.
DI: CX	New file date in seconds.

RETURN: Carry clear

Success

RETURN: Carry set

AL	The error number.
----	-------------------

PANIC: None

Sets the time and date for a file. CX is the least significant word and DI is the most significant word of the date and time in seconds.

FilLocChanged **Local file system changed**

BX	The bit mask for the changed channels
----	---------------------------------------

RETURN:

AX	The bit mask for any changed channels
----	---------------------------------------

PANIC: None

This service reports on whether the local file system has changed since the last time it was called.

Because independent software routines may require this service, up to 15 channels are provided. Each channel is a single bit in the register BX passed to this service and the corresponding bit in the result in register AX. The bits available are 0-14, the top bit being unavailable to ensure that a negative error can still be returned.

Bits 8-14 are reserved for system components and should not be used by any application. An application is free to choose any other bit in the range 0-7. Normally bit 0 would be used.

BX is loaded with 1 and the function called. If 0 is returned then the file system has not changed. If 1 is returned then it has changed. Note that the very first time this service is called, it is guaranteed to return changed. If BX is loaded with 2 and the file system has changed then AX will return 2.

Change as far as the file system is concerned is anything which would cause directory information on the file system to change but not alterations to an actual file. This service allows programs displaying file lists to update them automatically if any changes occur (such as changing the SSD or deleting a file).

③FilLocDevice Read media information of a local device

AL	Zero - synchronous request. Non-zero - Asynchronous request.
DX	If AL is non-zero then DX points to the status word.
BL	The local device.
CX	Pointer to a word to take the media type.

RETURN: Carry clear

Success.

RETURN: Carry set

NotReadyErr	The device is not available.
DeviceErr	The device BL is invalid.
NotSupportedErr	No PDD exists which can
UnknownErr	handle the device.

PANIC: None.

Read media information of a **local** device (that is, a device on *LOC::*) specified by BL. The value of BL must be one of the ASCII characters 'M' or 'I' or lie in the range 'A' to 'H' inclusive where 'I' (Internal) is an alias for 'M'.

Apart from two additional flags, the value written to the word at CX is the same as the value written to the DeviceMediaType field of the DeviceStatusEnt structure using the FilStatusDevice system service. These two extra flags are BatteryValid and BatteryGood defined in *osloc.inc*. If BatteryValid is not set then the device does not support battery measurement. If BatteryValid **is** set, then BatteryGood will be set if the voltage is good or clear if it is too low.

Note that the media does not have to be mountable for this service to work.

③FilLocReadPdd Read a local device directly

AL	Zero - synchronous request. Non-zero - Asynchronous request.
DX	If AL is non-zero then DX points to the status word.
BL	The local device.
CX	Pointer to a double word SSD position.
SI	Number of bytes to read.
DI	Pointer to buffer to take the data read.

RETURN: Carry clear

Success.

RETURN: Carry set

OsErr	The medium is not mounted.
CorruptMediaErr	The specified position is greater than the size of the SSD.

PANIC: None.

Read data directly from a given offset on a **local** device (that is a device on *LOC::*) into the buffer provided.

BL specifies the device and must be one of the ASCII characters 'M' or 'I' or lie in the range 'A' to 'H' inclusive where 'I' (Internal) is an alias for 'M'. CX points to a double word (4-byte) device offset. SI is the number of bytes to be read into the buffer at DI. The data is read very efficiently as it is performed by a direct access to the physical device driver (PDD).

The medium must have been mounted prior to using this service. To ensure that the medium is mounted, make any normal device access (e.g. call service FilStatusDevice).

CHAPTER 10

PROCESS MANAGEMENT

The process ID and names

In this chapter the following terms are all used to mean the same thing:

- process handle.
- process ID.
- pid.

All processes in the system are known by their ID. Many of the services return the process ID but often only the name of the process is known.

It is necessary to determine the ID of a process before services requiring an ID can be used. This can be accomplished by calling the `ProcIdByName` service.

There is an additional complication due to the fact that multiple instances of the same program can be running at any one time. The operating system overcomes this by giving each process a different number as part of its name depending on which process control block that it occupies. Processes have names of up to eight characters followed by a period, a \$ and a two digit number.

If a program called sort is executed twice, the process names might be as follows:

- SORT.\$07
- SORT.\$11

Since a program cannot know in advance the number part of a process name, the `ProcIdByName` service allows a wild card match string. Thus to find the process ID of the first sort process the string should be "SORT.*". Of course, only the first process would be found. If it is required to find all the sort processes the `ProcFind` service can be called repeatedly to enumerate all the matching processes.

Process scheduling

Epoc/Os is a time slice, multi-tasking operating system which schedules processes to run on the basis of their priorities.

A fixed time slice is given after every re-schedule which will be used by the process until it is all used up or the process gives up the CPU (e.g. when waiting for an I/O request to complete).

At any one time, only the highest priority process available to run will execute. All other lower priority processes are effectively blocked until the higher priority process gives up the CPU.

If two or more processes share the highest priority, each process will be executed on a round robin basis, for a time slice each.

If a higher priority process becomes available to run at any time, the lower priority process will be pre-empted and it will lose its time slice.

Priorities are unsigned byte values in the range 1 to 255. The operating system reserves the values in the range 1 to `CPBMinPriority-1` and `CPBMaxPriority+1` to 255. The supervisor runs at priority 248 while the file server runs at priority 240.

It is worth noting that interrupts run at the same priority as the process that they interrupt.

Process control

Often it is necessary for a process to be concerned about the presence of other processes in the system. For example, servers need to tidy up resources after processes which have connected to the server terminate before releasing the resources they own.

The services `MessSignal`, `IoSignalKillAsynchronous`, `MessSignalCancel` and `IoSignalKillCancel` are specially designed to help in tracking a process in the system.

If a process has logged on an interest in a process then it will receive a message or an I/O signal, when the process terminates. The message or status word will contain the reason code for the process terminating.

Process ID and process table address

The process ID consists of two components in a 16 bit value.

The first component occupying the 4 most significant bits is a value in the range 0 - 7 which is allocated by the operating system each time a process is created, and serves to uniquely identify the process.

The second component occupies the remaining 12 bits of the ID and is the offset in the operating system's data segment of the process control table. There is a lot of valuable information available in the process control table which can be retrieved using the `GenGetOsData` service. There is a mask declared `PidMask` which will mask out the address portion from the ID, suitable for use as the address to retrieve the process control data.

Terminate and Kill

There are two services which can be called to stop a process from running, `ProcTerminate` and `ProcKill`.

In certain types of applications it may be necessary for some cleanup code to be executed before the application actually exits. In these cases the process can register with the `ProcOnTerminate` service that it should be sent a message in response to the `ProcTerminate` request. It is then up to the application to detect the message, do the necessary cleanup code and then kill itself by calling `ProcKill`.

If a process is terminated with `ProcTerminate` and it has not registered a `ProcOnTerminate` then the operating system just calls `ProcKill`.

`ProcKill` will always destroy the application regardless of whether the application has registered a message with the `ProcOnTerminate` service or not.

`ProcTerminate` is the recommended method of stopping a process as it allows any process which does have cleanup code to execute the cleanup code on exit. If you are paranoid then a time out can be set up after the `ProcTerminate` service and if the process has not terminated then the `ProcKill` service can be called to definitely remove the process.

Procid

Get the current process ID

None

RETURN:

AX

Current process handle.

PANIC: None

Returns the ID of the process which calls this service.

ProcIdByName Get a process ID by name

ES:BX The process name.

RETURN: Carry clear

AX Process handle.

RETURN: Carry set

NotExistsErr The process does not exist.

PANIC: None

Returns the ID of a process corresponding to the name pointed to by BX. The name is a zero terminated string which may contain wild cards.

ProcGetPriority Get a process priority

BX The process handle.

RETURN: Carry clear

AL Process priority.

RETURN: Carry set

NotExistErr The process does not exist.

PANIC:

PanicProc1 Invalid process ID.

Returns the priority of the process specified in BX. The priority is returned in register AL.

ProcSetPriority Set a process priority

BX The process handle.

AL The new priority.

RETURN: Carry clear

Success.

RETURN: Carry set

RangeErr Priority was invalid.

NotExistErr The process does not exist.

FailErr Tried to set the priority of the null, supervisor or file server process.

PANIC:

PanicProc1 Invalid process ID.

Set the priority of the process specified in BX to the new priority value specified in register AL. The priority in AL must lie between the limits `CPBMinPriority` and `CPBMaxPriority` inclusive. Calling this service will cause a re-schedule. It is not possible to change the priority of the null, supervisor or file server processes.

ProcGetOwner Get the owning process

BX The process handle.

RETURN: Carry clear

AX The owning process PID.

RETURN: Carry set

NotExistErr The process does not exist.

PANIC:

PanicProc1 Invalid process ID.

This service returns the PID of an owning process. The ID of the process whose owner is being sought, is passed in register BX

The window server uses this service when a foreground process dies to determine which process should be brought to foreground next. For example if the system launches a program and it exits, the system will return to foreground. If the program, however, runs an OPL program which exits, the program will return to foreground.

Note that the owning process need not be running. You can check this by calling something harmless, such as `ProcGetPriority` on the returned PID in `AX`.

ProcCreate **Create a process**

`ES:BX` Pointer to the create process block.

RETURN: Carry clear

`AX` Process handle.

RETURN: Carry set

`NameErr` Not a valid process name.

`ExistsErr` A different process of the same name is already running.

`NoProcessErr` No more process slots.

`ArgumentErr` The size of data+stack+heap was greater than 0xFFE paragraphs.

PANIC: None

This is the basic create process service.

The information required to create a process is in the control block pointed to by `BX`. The control block must have the structure `CPBlock`. The create service leaves the process suspended and `ProcResume` should be called to start the process executing.

This service should not be called directly by an application. Instead, the higher level `FileExecute` service should be called; `FileExecute` will itself call this service at some point.

Because code segments can be shared and the key to shared code segments is the name of the process running, it is important to ensure that two images which are different but which have the same name are not run simultaneously. To this end a checksum of the code segment is remembered and if a request is made to start a new process of the same name as one already running their checksums are compared. If they are the same then the new process will share the already loaded code segment. If they are not then the `ExistsErr` will be returned. This error can also be returned from the `FileExecute` service.

ProcCreateTask **Create a task**

`DS:BX` Pointer to the task name.

`CL` Task priority.

`SS:DI` Pointer to top of stack in current process' data segment.

`CS:SI` Pointer to task code in the current process' code segment.

RETURN: Carry clear

`AX` Process handle.

RETURN: Carry set

`NameErr` Not a valid process name.

`NoProcessErr` No more process slots.

PANIC:

`PanicProc3` Task tried to create a task.

A task is a process just like any other process with the special property that it shares its data segment with an owning process. Since it shares the data space of the owning process, it can clearly access all the data belonging to the parent which makes communication with the parent very easy. Because it is still a separate entity in its own right, it needs its own stack in the data space of the parent (which is passed in `DI`). Clearly the parent and the task must co-operate closely when accessing the data in the data segment or unpredictable results will occur.

Imagine a spreadsheet program which must often go away for long periods of time to re-calculate all the formulae when a value is changed. While the re-calculation is being executed by the process it cannot be looking for user input, so the program appears to go dead for a while. The traditional way to overcome this problem is to poll the keyboard every now and then to see if there is some input requiring processing and if there is, to abandon the re-calculation to process the new input. Using tasks, it is much easier just to have a task to do the job of re-calculation. When the parent process needs the data re-calculating, it just resumes the task, which will immediately start the re-calculation and then wait either for the task to finish or for more input to arrive. When the task has completed its work it can simply suspend itself until needed again. By having the task at a lower priority than the parent process, the re-calculation will always be halted allowing the parent process to deal with fresh input.

A process can create as many tasks as it has stack space available and there are process slots free in the system. Remember that a task is still a process. If a task is panicked then the parent process will be panicked as well. If the parent process terminates, then the supervisor will arrange for all the tasks belonging to it to be terminated as well. Task are initially given the priority as passed in CL, but the priority may be changed at any time with the `ProcSetPriority` service. A task cannot have any heap space since the parent process owns and controls the heap. Thus the parent must provide any space that the task requires.

No services which use the heap allocator can be called. Thus a task cannot open an I/O channel; the parent must open the channel on behalf of the task.

A good way to co-ordinate a task and its parent is to use a semaphore to lock the shared data. The semaphore is created with a value of one. Before either parent or task accesses the data, the semaphore is waited on with `SemWait`. When the data is no longer required the semaphore is signalled with `SemSignalOnce`.

ProcResume

Resume a process

BX	The process handle.
----	---------------------

RETURN: Carry clear

Success.

RETURN: Carry set

ArgumentErr

The process is not currently suspended.

NotExistErr

The process does not exist.

PANIC:

PanicProc1

Invalid process ID.

Resume a suspended process specified in BX.

ProcSuspend

Suspend a process

BX	The process handle.
----	---------------------

RETURN: Carry clear

Success.

RETURN: Carry set

NotExistsErr

The process does not exist.

FailErr

Tried to suspend the null, supervisor or file server process.

PANIC:

PanicProc1

Invalid process ID.

Suspend the process specified in BX.

Processes which are on the ready queue or currently running (i.e. the calling process), will be suspended immediately.

Attempting to suspend a process which is currently waiting on the semaphore or delta queues will cause that process to be marked as requiring suspension. When it is eventually transferred to the ready queue, it will be suspended immediately.

Conversely, attempting to resume a process which is currently waiting on the semaphore or delta queues and is marked as requiring suspension, will simply be unmarked.

It is not possible to suspend the null, supervisor or file server process.

A suspended process is resumed by calling the `ProcResume` service.

ProcKill **Kill a process**

AL	The kill reason.
BX	The process handle.

RETURN: Carry clear

Success

RETURN: Carry set

NotExistsErr	The process does not exist.
--------------	-----------------------------

FailErr	Tried to kill the null, supervisor or file server process.
---------	--

PANIC:

PanicProc1	Invalid process ID.
------------	---------------------

Kill the process specified in BX.

Regardless of what state the process is in, it will be killed and the kill reason will be remembered as the reason for death. It is recommended that the `ProcTerminate` service is called in preference to this service as the terminate service will allow any processes which have cleanup code to execute this code, whereas this service will just kill the process immediately.

ProcOnTerminate **Register termination**

BX	The message to be sent when being terminated.
----	---

RETURN: None

PANIC:

PanicMess2	Messaging not initialised.
------------	----------------------------

Register a message to be sent when termination of the process is requested with `ProcTerminate`.

Messaging must previously have been initialised with the `MessInit` service. Having registered an "on termination" message, it can be cancelled later by registering the message 0. A process which registers an "on termination" message is duty bound to respond to the message by eventually committing suicide by calling the `ProcKill` service.

ProcTerminate **Terminate a process**

AL	The terminate reason.
BX	The process handle.

RETURN: Carry clear

Success.

RETURN: Carry set

NotExistsErr	The process does not exist.
--------------	-----------------------------

FailErr	Tried to terminate the null, supervisor or file server process.
---------	---

PANIC:

PanicProc1	Invalid process ID.
------------	---------------------

Terminate the process specified in BX.

If the process has registered an "on terminate" message then it will be sent this message. If it has no "on terminate" message registered then it will be killed. This is the recommended way to terminate processes.

ProcPanicById**Panic a process**

AL The panic reason.
 BX The process handle.

RETURN: Carry clear
 Success

RETURN: Carry set
 NotExistsErr The process does not exist.
 FailErr Tried to panic the null, supervisor or file server process.

PANIC:
 PanicProc1 Invalid process ID.

Panic the process specified in BX.

Regardless of what state the process is in, it will be killed and the panic reason will be remembered.

ProcNameById**Get a process name by ID**

BX The process handle.
 ES:DI Pointer to the buffer to receive the name.

RETURN: Carry clear
 Success.

RETURN: Carry set
 NotExistsErr The process does not exist.

PANIC:
 PanicProc1 Invalid process ID.

Returns the name of the process specified by the ID in BX. The buffer should be large enough for MaxNameESize bytes.

ProcRename**Rename a process**

BX The ID of the process to be renamed.
 ES:DI Pointer to the new name for the process.

RETURN: Carry clear
 Success

RETURN: Carry set
 NotExistsErr The process does not exist.
 NameErr The new name was invalid.
 FailErr Tried to rename the null, supervisor or file server process.

PANIC: None

Rename the process whose ID is contained in BX to the new name pointed to by ES:DI. The new name can be between 1 and 8 characters long and must be zero terminated. No check is made to ensure that the new name is unique.

ProcFind**Find all processes**

BX The find handle.
 ES:DI Pointer to a wild card match string.
 DS:SI Pointer to the buffer to receive the name of the found process.

RETURN: Carry clear

AX

The find handle for the next find.

RETURN: Carry set

NotExistsErr

No more processes found.

PANIC:

PanicProc1

Invalid process ID.

Finds all the processes which match the wild card string pointed to by DI.

The first time this service is called, BX should be set to zero; the first process will be found.

After a find, this service returns the find handle in the AX register. The find handle must be supplied on the next call to this service to find the next process. No memory is used by this service and it can be abandoned at any time without taking any further action. The wild card string must always be supplied in DI and should remain the same between calls to this service. The buffer to receive the name must be at least MaxNameESize+2 bytes long.

ProcWatchAllExits

Watch all exits

BX

The message number.

RETURN: Carry clear

Success

RETURN: Carry set

FailErr

Watch all exits already activated.

PANIC: None

A process can arrange to be sent the message specified in BX whenever a process exits. Only one process at a time can request this service and it is usually reserved for the SHELL so that it can monitor the exits of all processes. Messaging must be initialised before this service can be invoked. Requesting a message number 0 cancels this service. The body of the message, when it arrives, contains two words; the first word is the ID of the exiting process and the second is the reason for the exit.

• ProcPanic

Panic the current process

AL

The panic reason.

RETURN: None

PANIC: None

Panic the current process for the reason specified in AL. This is suicide of some kind.

• ProcCopyFromByld

Copy data from a process by ID

BX

The process handle.

CX

The number of bytes to be copied.

ES:DI

Pointer to the destination buffer in the current process.

SI

The offset in the process, specified in BX, data segment from which the data is to be copied.

RETURN: Carry clear

Success

RETURN: Carry set

ArgumentErr

Process did not exist or SI+CX exceeded the segment size.

PANIC: None

Copies CX bytes of data from the buffer pointed to by SI within the data segment of the process specified by BX to the buffer pointed to by DI. If CX is less than or equal to 64 then the data will be copied with interrupts disabled.

• ProcIndStringCopyFromByld Copy strings from process by ID

BX	The process handle.
CX	The maximum number of bytes to be copied.
ES:DI	Pointer to the destination buffer in the current process.
SI	The address of a pointer to a string in the processes (specified in BX) data segment from which the string is to be copied.

RETURN: Carry clear

Success

RETURN: Carry set

ArgumentErr

Process did not exist or SI+CX exceeded the segment size.

PANIC: None

Copies up to CX bytes of text to the buffer pointed to by DI; the source of the text string is in the data segment of the process specified by BX and SI points to the pointer of the text string. If CX is less than or equal to 64 then the data will be copied with interrupts disabled.

• ProcCopyToByld Copy data to a process by ID

BX	The process handle.
CX	The number of bytes to be copied.
DI	The offset in the process, specified in BX, data segment to which the data is to be copied.
DS:SI	Pointer to the source buffer in the current process from which the data is to be copied.

RETURN: Carry clear

Success

RETURN: Carry set

ArgumentErr

Process did not exist or DI+CX exceeded the segment size.

PANIC: None

Copies CX bytes of data from the buffer pointed to by SI to the buffer pointed to by DI in the data segment of the process specified by BX. If CX is less than or equal to 64 then the data will be copied with interrupts disabled.

CHAPTER 11

DATE AND TIME MANAGEMENT

Absolute and relative times

The operating system has the concept of relative and absolute times and all services work in either relative or absolute time.

Absolute time services always work in seconds where the seconds are in the same format as the system time, i.e. they specify a real time in the future. If the operating system enters standby mode with an absolute time event still pending then it will ensure that it will wake up in time to service the event. Thus, if a process gets the system time, adds $24 \times 60 \times 60$ to it and then waits absolutely until that time then the operating system will ensure that it is running 24 hours later. Absolute times are unaffected by changing the system time so that if the system time is advanced by 1 day then absolute times are not also advanced by 1 day.

Relative time services work in tenths of a second or system ticks. Unlike absolute times, a relative time means x units from now. Again, changing the system time will have no effect upon relative times. If a relative time is set for 5 seconds and the system time is advanced by 1 hour, the relative time will still wait for 5 seconds. Again, unlike absolute times, if the operating system enters standby mode, the time in standby is not taken into account. If there is a relative time outstanding and the operating system enters standby mode for 1 hour then the relative time will be 1 hour plus 5 seconds. This also means that the operating system will not exit from standby mode to service relative time events.

TimWaitAbsolute

Wait to a given time

CX:DX The time to wait till in seconds.

RETURN: Carry clear

 Success

RETURN: Carry set

 ArgumentErr The value in CX:DX was in the past.

PANIC: None

Sleep the process calling this service until the system time given in CX:DX, where CX:DX is a 32 bit integer.

DX is the least significant word and CX is the most significant word. This is an absolute time service.

TimSleepForTenths **Sleep for tenths of a second**

CX:DX The time to sleep in tenths of a second.

RETURN: Carry clear

 Success

RETURN: Carry set

 ArgumentErr The value in CX:DX was negative.

 OverflowErr The value in CX:DX overflowed when converted to ticks.

PANIC: None

Sleep the process calling this service for CX:DX tenths of a second, where CX:DX is a 32 bit integer.

DX is the least significant word and CX is the most significant word. If sleep is requested for N tenths, then the sleep is guaranteed to be between N and N+1 tenths.

This is a relative time service.

TimSleepForTicks **Sleep for system clock ticks**

CX:DX The time to sleep in system clock ticks.

RETURN: Carry clear

 Success

RETURN: Carry set

 ArgumentErr The value in CX:DX was negative.

PANIC: None

Sleep the process calling this service for CX:DX system clock ticks where CX:DX is a 32 bit integer.

DX is the least significant word and CX is the most significant word. If sleep is requested for N tenths, then the sleep is guaranteed to be between N and N+1 ticks.

On IBM compatible PCs, a clock tick occurs 18.2 times a second and on the SIBO hardware occurs 32 times a second.

A request to sleep for zero ticks is valid and will just force a re-schedule; the process calling this service loses the remainder of its time slice if there are other processes at the same priority.

This is a relative time service.

TimGetSystemTime **Get the system time**

 None

RETURN:

 AX:BX System time in seconds.

PANIC: None

Returns the system time in seconds.

The system time is stored as a 32 bit unsigned integer; the number of seconds since 1st. January 1970 at 00:00:00 (i.e. UNIX time). BX is the least significant word and AX is the most significant word.

TimSetSystemTime **Set the system time**

CX:DX The new system time.

RETURN: None

PANIC: None

Sets the system time to the new value passed in CX:DX.

The system time is stored as a 32 bit unsigned integer; the number of seconds since 1st. January 1970 at 00:00:00 (i.e. UNIX time). DX is the least significant word and CX is the most significant word.

TimSystemTimeToDaySeconds Convert sys' time to day secs

CX:DX The time to be converted in seconds.
DS:DI Pointer to a DyScEnt structure.

RETURN: None

PANIC: None

Converts the system time to the number of days since 1970 and the number of remaining seconds.

DS:DI points to a DyScEnt structure which contains a long integer number of days and a long integer number of seconds. DX is the least significant word and CX is the most significant word of the time to be converted.

TimDaySecondsToSystemTime Convert day secs to system time

DS:SI Pointer to a DyScEnt structure.

RETURN: Carry clear

AX:BX System time.

RETURN: Carry set

ArgumentErr DyScSeconds greater than 24*60*60

overflowErr Exceeded system time.

PANIC: None

Convert a DyScEnt structure pointed to by DS:SI to system time.

DX is the least significant word and AX is the most significant word of the system time.

TimDaySecondsToDate Convert day seconds to date

DS:SI Pointer to a DyScEnt structure.

DS:DI Pointer to a DateEnt structure.

RETURN: Carry clear

Success

RETURN: Carry set

ArgumentErr DyScSeconds greater than 24*60*60

overflowErr Years out of range.

PANIC: None

Converts the day seconds in the DyScEnt structure pointed to by DS:SI to date in the DateEnt structure pointed to by DS:DI.

TimDateToDaySeconds Convert date to day seconds

DS:SI Pointer to a DateEnt structure.

DS:DI Pointer to a DyScEnt structure.

RETURN: Carry clear

Success

RETURN: Carry set

ArgumentErr Date is invalid.

PANIC: None

Converts the date in the DateEnt structure pointed to by DS:SI to day seconds in the DyScEnt structure pointed to by DS:DI.

TimDaysInMonth **Number of days in a month**

CH The month number, 0 to 11 (0 equals January).

CL The year number (0 equals 1900).

RETURN: Carry clear

AX Number of days in the month.

RETURN: Carry set

ArgumentErr Invalid month.

PANIC: None

Returns the number of days in a month corresponding to the month number passed in CH and the year number passed in CL.

The value in AX will be in the range 28 to 31. If this service returns 29 for any year in CL and 1 in CH then the year is a leap year (i.e. February with 29 days).

TimDayOfWeek **Week day number**

CX:DX The number of days since 1900.

RETURN:

AX The week day number, 0 = Monday.

PANIC: None

Returns the weekday number corresponding to the number of days since 1900 in CX:DX.

DX is the least significant word and CX is the most significant word. The value in AX will be in the range 0 to 6, with 0 being Monday and 6 being Sunday.

TimNameOfDay **Name of day**

AL The day of week number, 0 to 6.

DS:BX Pointer to a buffer to receive the name of the day.

RETURN: Carry clear

Success

RETURN: Carry set

ArgumentErr The day of week number is not in the range 0 to 6.

PANIC: None

Returns the name of the day corresponding to the day of week number in AL.

The day name is returned as a zero terminated string in the buffer pointed to by DS:BX. The maximum size of the string is 32 bytes. This is a language dependent service.

TimNameOfMonth **Name of month**

AL The month number, 0 to 11.

DS:BX Pointer to a buffer to receive the name of the month.

RETURN: Carry clear

Success

RETURN: Carry set

ArgumentErr The month number is not in the range 0 to 11.

PANIC: None

Returns the name of the month corresponding to the month number in AL.

The month name is returned as a zero terminated string in the buffer pointed to by DS:BX. The maximum size of the string is 32 bytes. This is a language dependent service.

TimWeekNumber	Week number
---------------	-------------

CX:DX	The number of days since Jan 1 1900.
-------	--------------------------------------

RETURN: Carry clear

AX	The week number in the range 1-53.
----	------------------------------------

RETURN: Carry set

ArgumentErr	The number of days exceeds the year 2165.
-------------	---

PANIC: None

Returns the week number corresponding to the number of days since Jan 1 1900.

This service takes the value in CDataStartOfWeek in the country dependent data into account.

• TimNameOfDayAbb	Abbreviated name of day
-------------------	-------------------------

AL	The day of week number, 0 to 6.
----	---------------------------------

BX	Pointer to a buffer to receive the name of the day.
----	---

RETURN: Carry clear

Success

RETURN: Carry set

ArgumentErr	The day of week number is not in the range 0 to 6.
-------------	--

PANIC: None

Returns the abbreviated name of the day corresponding to the day of week number in AL.

The day name is returned as a zero terminated string in the buffer pointed to by BX. This is a language dependent service.

All month name abbreviations for a particular language have the same length and could be one, two or three characters. The abbreviation will not exceed three characters in any language.

• TimNameOfMonthAbb	Abbreviated name of month
---------------------	---------------------------

AL	The month number, 0 to 11.
----	----------------------------

BX	Pointer to a buffer to receive the name of the month.
----	---

RETURN: Carry clear

Success

RETURN: Carry set

ArgumentErr	The month number is not in the range 0 to 11.
-------------	---

PANIC: None

Returns the abbreviation for the name of the month corresponding to the month number in AL.

The month name is returned as a zero terminated string in the buffer pointed to by BX. This is a language dependent service.

All month name abbreviations for a particular language have the same length and could be one, two or three characters. The abbreviation will not exceed three characters in any language.

CHAPTER 12

CONVERSION MANAGEMENT

ConvUnsignedIntToBuffer Unsigned integer to buffer

BX	The number to be converted.
CX	The conversion radix.
ES:DI	Pointer to buffer to receive the converted number.

RETURN:

AX	The number of characters written to DI.
----	---

PANIC: None

Converts an unsigned integer in BX to its corresponding digits in the buffer pointed to by DI using the radix specified in CX, i.e. 2 for binary, 8 for octal, 10 for decimal and 16 for hexadecimal.

The number of characters written to the buffer at DI is returned in AX.

ConvUnsignedLongIntToBuffer Unsigned long integer to buffer

DX:BX	The number to be converted.
CX	The conversion radix.
ES:DI	Pointer to buffer to receive the converted number.

RETURN:

AX	The number of characters written to ES:DI.
----	--

PANIC: None

Converts an unsigned long integer in DX:BX to its corresponding digits in the buffer pointed to by DI using the radix specified in CX, i.e. 2 for binary, 8 for octal, 10 for decimal and 16 for hexadecimal.

BX is the least significant word and DX is the most significant word. The number of characters written to the buffer at DI is returned in AX.

ConvIntToBuffer Integer to buffer

BX	The number to be converted.
ES:DI	Pointer to buffer to receive the converted number.

RETURN:

AX	The number of characters written to DI.
----	---

PANIC: None

Converts an integer in BX to its corresponding digits in the buffer pointed to by DI.

If BX is negative then the first character in the buffer will be a minus sign '-'. The conversion is performed using a radix of 10. The number of characters written to the buffer at DI is returned in AX.

ConvLongIntToBuffer **Long integer to buffer**

DX:BX The number to be converted.
ES:DI Pointer to buffer to receive the converted number.

RETURN:

AX The number of characters written to DI.

PANIC: None

Converts a long integer in DX:BX to its corresponding digits in the buffer pointed to by DI.

If BX is negative then the first character in the buffer will be a minus sign '-'. The conversion is performed using a radix of 10. BX is the least significant word and DX is the most significant word. The number of characters written to the buffer at DI is returned in AX.

ConvArgumentsToBuffer **Convert arguments to buffer**

SS:BX Pointer to the argument list.
ES:DI Pointer to buffer to receive the converted arguments.
DS:SI Pointer to format control string.

RETURN:

AX The number of characters written to ES:DI.

PANIC: None

Converts the values in the list pointed to by BX into a buffer at DI.

The conversion is controlled by the format string pointed to by SI. This service is used by `p_atob()` in PLIB. Note that the arguments in the list pointed to BX must be as if they were being passed on the stack to a C function. i.e. only words and double words are allowed.

ConvStringToUnsignedInt **String to unsigned integer**

CX Radix for conversion.
DS:SI Pointer to zero terminated string to be converted.

RETURN: Carry clear

AX Resultant unsigned integer.
SI Pointer to the first unused character in the string.

RETURN: Carry set

FailErr No valid digits in string.
OverflowErr Number too large for an unsigned integer.

PANIC: None

Convert a zero terminated string to an unsigned integer using the radix specified in CX, i.e. 2 for binary, 8 for octal, 10 for decimal and 16 for hexadecimal. The conversion will stop at a character that is not valid for the conversion radix.

If the string consists only of valid digits and the number does not overflow then SI will be returned pointing to the terminating 0 otherwise SI will be pointing at the invalid digit.

ConvStringToUnsignedLongInt **String to unsigned long integer**

CX Radix for conversion.
DS:SI Pointer to zero terminated string to be converted.

RETURN: Carry clear

AX:BX Resultant unsigned long integer.
SI Pointer to the first unused character in the string.

RETURN: Carry set

FailErr No valid digits in string.

OverflowErr Number too large for an unsigned long integer.

PANIC: None

Convert a zero terminated string to an unsigned long integer using the radix specified in CX, i.e. 2 for binary, 8 for octal, 10 for decimal and 16 for hexadecimal. The conversion will stop at a character that is not valid for the conversion radix.

If the string consists only of valid digits and the number does not overflow then SI will be returned pointing to the terminating 0 otherwise SI will be pointing at the invalid digit. The result is returned in BX:AX where BX is the least significant word and AX is the most significant word.

ConvStringToInt

String to integer

DS:SI Pointer to zero terminated string to be converted.

RETURN: Carry clear

AX Resultant integer.

SI Pointer to the first unused character in the string.

RETURN: Carry set

FailErr No valid digits in string.

OverflowErr Number too large for an integer.

PANIC: None

Convert a zero terminated string to an integer using a radix of 10.

The string can begin with a minus character '-', in which case the integer will be negative or a plus character '+', in which case the integer will be positive. The maximum positive value is 32767 and the maximum negative value is -32768. The conversion will stop at a character that is invalid for a conversion radix of 10.

If the string consists only of valid digits and the number does not overflow then SI will be returned pointing to the terminating 0 otherwise SI will point to the invalid digit.

ConvStringToLongInt

String to long integer

DS:SI Pointer to zero terminated string to be converted.

RETURN: Carry clear

AX:BX Resultant long integer.

SI Pointer to the first unused character in the string.

RETURN: Carry set

FailErr No valid digits in string.

OverflowErr Number too large for a long integer.

PANIC: None

Convert a zero terminated string to a long integer using a radix of 10.

The string can begin with a minus character '-', in which case the long integer will be negative or a plus character '+', in which case the long integer will be positive. The maximum positive value is 4294967295 and the maximum negative value is -4294967295. The conversion will stop at a character that is invalid for a conversion radix of 10.

If the string consists only of valid digits and the number does not overflow then SI will be returned pointing to the terminating 0 otherwise SI will point to the invalid digit. The result is returned in BX:AX where BX is the least significant word and AX is the most significant word.

ConvFloatToBuffer **Floating point number to buffer**

SI	Pointer to the float to be converted.
DX	Pointer to the format structure.
DI	Pointer to buffer to receive the converted float.

RETURN: Carry clear

AX	Length of converted string.
----	-----------------------------

RETURN: Carry set

ArgumentErr	Invalid float, or illegal [DX].DtobType.
FailErr	Representation exceeds [DX].Dtobwidth characters.
OverflowErr	Float >= 1E+100.
UnderflowErr	Float < 1E-99.

PANIC: None

Converts a double floating point number in [SI] to a printable ASCII zero terminated string pointed to by DI using the supplied format specification. The format specification is contained in the DtobEnt structure as defined in *epocdefs.inc* as:

```
Dtob struc ; dtob format string structure
DtobType   db      ? ; conversion type
Dtobwidth  db      ? ; width of representation in characters
DtobNdec   db      ? ; number of decimal places
DtobPoint  db      ? ; decimal point character
DtobTriad  db      ? ; triad separator character
DtobTrilen db      ? ; threshold for triad character use
DtobEnt ends
```

Numbers may be represented in various formats by setting DtobType as follows:-

- DtobTypeFixed, fixed point format.
- DtobTypeExponent, exponent format.
- DtobTypeGeneral, general format.

Integer format is obtained as a special case of fixed point format where the number of decimal places required is zero. The parameter [DX].Dtobwidth specifies the maximum number of characters allowed to represent the number and there should be [DX].Dtobwidth+1 bytes (the +1 is for the zero terminator) reserved at DI. If the output exceeds this limit, FailErr is returned. Although numbers are normally displayed right-aligned, ConvFloatToBuffer makes no attempt to align the result in the buffer. Alignment is quite different for monospaced and proportionally spaced character fonts and is best handled by post-processing the output from ConvFloatToBuffer.

[DX].Dtobwidth should be in the range 1 to 255 inclusive. The parameter [DX].DtobNdec specifies the number of decimal digits following the decimal point when [DX].DtobType is DtobTypeFixed or DtobTypeExponent. [DX].DtobNdec must be in the range 0 to FloatSignificantDigits (15 for IEEE floating point format) inclusive. The parameter [DX].DtobPoint specifies the decimal point character which separates the integer portion from the fractional portion and would normally be either a '.' or a ','.

The parameter [DX].DtobTriad specifies the triad separator character which delimits groups of 3 digits in the integer part of the fixed point representation and would normally be either ',' or '.' or ' '. The insertion of triad separation characters is disabled if [DX].DtobTrilen is 0 and otherwise enabled when the integer portion of the number contains greater than [DX].DtobTrilen digits. Normally one would use [DX].DtobTrilen=1 to enable triad separation and [DX].DtobTrilen is 4 to conform to French conventions for triad separator insertion.

In all cases, negative numbers are represented by the insertion of a leading '-' sign (positive numbers do not have a leading '+' sign). This can easily be post-processed to obtain a bracketed representation of negative numbers, if desired. There can never be more than FloatSignificantDigits(15 FOR IEEE floating point format). Where there are less than FloatSignificantDigits, the number is rounded to the number of significant digits displayed.

The limitation to floats with magnitude between 1E-99 and 1E+100 results from the use of lookup tables for speedily generating the results. Floating point numbers of magnitude smaller than 1E-99 can easily be converted to 0 before calling this service. The detailed formatting details as a function of [DX].DtobType is as follows:

- **DtobTypeFixed** - The number is represented with [DX].DtobNdec decimal places where [DX].DtobNdec may be zero to represent an integer (in which case no decimal point character is displayed). If the ASCII form exceeds [DX].DtobWidth (usually due to the number being large and having too many digits before the decimal point), **FailErr** is returned. A zero is displayed in the form "0.000" where there are [DX].DtobNdec zeros following the decimal point or as just "0" if [DX].DtobNdec is zero.
- **DtobTypeExponent** - The number is represented in exponent notation with one non-zero digit before the decimal point and [DX].DtobNdec digits beyond the decimal point followed by 'E', a sign ('+' or '-') and the exponent as two digits (with leading zero if necessary). If [DX].DtobNdec is zero, the number is rounded to one digit of precision and no decimal point is displayed. A zero is displayed in the form "0.000E+00" where there are [DX].DtobNdec zeros following the decimal point or as "0E+00" if [DX].DtobNdec is zero. Triad separation is not available and triad separation parameters are ignored.
- **DtobTypeGeneral** - converts either as fixed format (with no triad separator) or exponent format, making best use of [DX].DtobWidth. Here, "making best use" is defined as showing the greater number of significant digits and preferring fixed format when the number of significant digits shown is the same. The number of decimal places is chosen as a function of [DX].DtobWidth and the value of [DX].DtobNdec is ignored. A zero is displayed as just "0". Triad separation is not available and triad separation parameters are ignored.

ConvStringToFloat

String to float

SI	address of pointer to text to convert
DX	Decimal point character
DI	Pointer to double destination

RETURN: Carry clear

Success

RETURN: Carry set

FailErr	Failed to recognise a number.
OverflowErr	Number too large.
UnderflowErr	Number less than 1E-99, 0 written to [DI].

PANIC: None

Scans the string for a number and writes the value as a double float to [DI]. If underflow occurs, zero is written to [DI].

The supplied ASCII string should take the form:

```
[+|-]<<int>>.<fract>[E|e][+|-]<<exp>>
```

Where:

- The leading '+' sign may be omitted for positive numbers. <<int>> and <<fract>> are optional but at least one should be present.
- Leading zeros in <int> are legal but have no effect.
- Trailing zeros in <fract> are legal but have no effect.
- There is no reasonable limit to the number of significant digits but digits which are beyond the precision of the floating point representation will not be reflected in the mantissa of the number which is produced.
- The exponent field which starts with and 'E' or 'e' is optional.
- The leading '+' sign in the exponent field may be omitted for positive exponents.
- The resulting number should be in the range approximately 1e-99 to approximately 1e+99.

CHAPTER 13

LONG INTEGER MANAGEMENT

• LongIntCompare Compare two long integers

AX:BX The left operand long integer.
CX:DX The right operand long integer.

RETURN:

Flags < 0 If AX:BX < CX:DX
Flags = 0 If AX:BX = CX:DX
Flags >> 0 If AX:BX > CX:DX

PANIC: None

Compares two long integers for equality. The flags are set in the same way as for a normal compare for integers, i.e. CMP AX:BX, CX:DX.

Note that the flags are set so that only the signed tests can be performed, i.e. JLE,JL,JE,JNE,JG,JGE and not the unsigned tests JB,JBE,JA,JAЕ.

All registers are preserved by this service.

• LongIntMultiply Long integer multiplication

AX:BX The left operand long integer.
CX:DX The right operand long integer.

RETURN: Carry clear

AX:BX Product.

RETURN: Carry set

OverflowErr AX:BX * CX:DX is bigger than 32 bits.

PANIC: None

Multiply two long integers together. If the resultant product overflows then an error will be returned by setting the carry flag.

If no error occurs, the flags will not be set for the result; carry will be clear.

• LongIntDivide Long integer division

AX:BX The left operand long integer (dividend).
CX:DX The right operand long integer (divisor).

RETURN: Carry clear

AX:BX The quotient.

CX:DX The remainder.

RETURN: Carry set

DivideByZeroErr CX:DX is zero.

PANIC: None

Divide one long integer by another.

If CX:DX (i.e. the divisor), is zero then an error will be returned and the carry flag will be set.

If no error occurs, the flags will not be set for the result; carry will be clear.

The remainder will have the same sign as the dividend.

• LongUnsignedIntCompare Compare 2 unsigned long integers

AX:BX	The left operand unsigned long integer.
CX:DX	The right operand unsigned long integer.

RETURN:

Flags << 0	If AX:BX < CX:DX
Flags = 0	If AX:BX = CX:DX
Flags > 0	If AX:BX > CX:DX

PANIC: None

Compares two unsigned long integers for equality.

The flags are set in the same way as for a normal compare for integers, i.e. CMP AX:BX, CX:DX.

Note that the flags are set so that only the unsigned tests can be performed, i.e. JBE,JB,JE,JNE,JA,JAЕ and not the signed tests JL,JLE,JG,JGE.

All registers are preserved by this service.

• LongUnsignedIntMultiply Unsigned long integer multiplication

AX:BX	The left operand unsigned long integer.
CX:DX	The right operand unsigned long integer.

RETURN: Carry clear

AX:BX	Product.
-------	----------

RETURN: Carry set

OverflowErr	AX:BX * CX:DX is bigger than 32 bits.
-------------	---------------------------------------

PANIC: None

Multiply two unsigned long integers together.

If the resultant product overflows, an error will be returned by setting the carry flag.

If no error occurs, the flags will not be set for the result; carry will be clear.

• LongUnsignedIntDivide Unsigned long integer division

AX:BX	The left operand unsigned long integer (dividend).
CX:DX	The right operand unsigned long integer (divisor).

RETURN: Carry clear

AX:BX	The quotient.
CX:DX	The remainder.

RETURN: Carry set

DivideByZeroErr	CX:DX is zero.
-----------------	----------------

PANIC: None

Divide one unsigned long integer by another.

If CX:DX (i.e. the divisor), is zero then an error will be returned and the carry flag will be set.

If no error occurs, the flags will not be set for the result; carry will be clear.

LongUnsignedIntRandom Unsigned long integer random number

DS:BX Pointer to the unsigned long integer seed.

RETURN:

AX:BX The unsigned long integer random number.

PANIC: None

Return an unsigned long integer random number given the seed.

The 4 bytes pointed to by BX are used to generate the next random number which is returned in AX:BX as well as being written back to the seed. The seed can start with any number required from which the same sequence of random numbers will be generated.

CHAPTER 14

FLOATING POINT NUMBER HANDLING

• FloatCompare Compare two Floats

DI Pointer to left hand floating point operand.
SI Pointer to right hand floating point operand.

RETURN:

Z flag set if [DI]=[SI].
S flag set if [DI]<<[SI].

PANIC: None

Compares the two floating point operands pointed to by SI and DI, setting the Z and the S flags as for [DI]-[SI]. The Z flag is set if the operands are equal and the S flag is set if [DI] is less than [SI].

• FloatMultiply Multiply two floats

DI Pointer to left hand (and destination) floating point operand.
SI Pointer to right hand floating point operand.

RETURN: Carry clear
 [DI]

RETURN: Carry set
 overFlowError Product exponent overflowed.

PANIC: None

Multiplies the two floating point operands pointed to by SI and DI. Returns the product in [DI].

• FloatDivide Divide floats

DI Pointer to left hand (and destination) floating point operand.
SI Pointer to right hand floating point operand.

RETURN: Carry clear
 [DI]

RETURN: Carry set
 overFlowError Quotient exponent overflowed.

PANIC: None

Divides the operand at DI by the operand at SI and returns the quotient in [DI].

• FloatAdd **Add two Floats**

DI Pointer to left hand (and destination) floating point operand.

SI Pointer to right hand floating point operand.

RETURN: Carry clear

[DI]

RETURN: Carry set

OverflowError Sum exponent overflowed.

PANIC: None

Adds the two floating point operands pointed to by SI and DI. Returns the sum in [DI].

• FloatSubtract **Subtract Floats**

DI Pointer to left hand (and destination) floating point operand.

SI Pointer to right hand floating point operand.

RETURN: Carry clear

[DI]

RETURN: Carry set

OverflowError Sum exponent overflowed.

PANIC: None

Subtracts the operand at SI from the operand at DI and returns the difference in [DI].

• FloatNegate **Negate a Floats**

DI Pointer to floating point operand.

RETURN:

[DI]

PANIC: None

Negates the float at DI.

• FloatToLong **Convert Float to a signed long**

SI Pointer to float operand to be converted.

RETURN: Carry clear

AX:BX Long integer result.

RETURN: Carry set

ArgumentErr Float not in range $[-2^{31}, 2^{31}-1]$.

PANIC: None

Converts the floating point operand pointed to by SI to a 32 bit signed long integer in AX:BX with the most significant word in AX.

• FloatToUnsignedLong **Convert Float to unsigned long**

SI Pointer to float operand to be converted.

RETURN: Carry clear

AX:BX Unsigned long integer result.

RETURN: Carry set

ArgumentErr Float not in range $[-2^{32}+1, 2^{32}-1]$.

PANIC: None

Converts the floating point operand pointed to by SI to a 32 bit unsigned integer in AX:BX with the most significant word in AX. Note that the sign of the float is ignored.

• FloatToInt Convert Float to a signed integer

SI Pointer to float operand to be converted.

RETURN: Carry clear

AX unsigned integer result.

RETURN: Carry set

ArgumentErr Float not in range [-32768,32767].

PANIC: None

Converts the floating point operand pointed to by SI to a 16 bit signed integer in AX.

• FloatToUnsignedInt Convert Float to unsigned integer

SI Pointer to float operand to be converted.

RETURN: Carry clear

AX Integer result.

RETURN: Carry set

ArgumentErr Float not in range [-65535,65535]

PANIC: None

Converts the floating point operand pointed to by SI to a 16 bit unsigned integer in AX. The sign of the float is ignored.

• LongToFloat Convert signed long to Float

AX:BX Signed long to be converted.

DI Pointer to destination float.

RETURN:

[DI]

PANIC: None

Converts the signed 32 bit integer in AX:BX to a float at DI.

• IntToFloat Convert signed integer to Float

AX Signed integer to be converted.

DI Pointer to destination float.

RETURN:

[DI]

PANIC: None

Converts the signed 16 bit integer in AX to a float at DI.

• UnsignedIntToFloat Convert unsigned integer to Float

AX Unsigned integer to be converted.

DI Pointer to destination float.

RETURN:

[DI]

PANIC: None

Converts the unsigned 16 bit integer in AX to a float at DI.

CHAPTER 15

FLOATING POINT FUNCTION INTERFACE

FloatASin Arcsine of a float

DI	Pointer to destination floating point operand.
SI	Pointer to floating point function argument.
RETURN: Carry clear	
[DI]	
RETURN: Carry Set	
ArgumentErr	Invalid float.
PANIC: None	

Calculates the Arcsine in radians of a double argument in [SI] returning the result in [DI]. [SI] is preserved unless SI equals DI.

FloatATan Arctangent of a float

DI	Pointer to destination floating point operand.
SI	Pointer to floating point function argument.
RETURN: Carry clear	
[DI]	
RETURN: Carry Set	
ArgumentErr	Invalid float.
PANIC: None	

Calculates the Arctangent in radians of a double argument in [SI] returning the result in [DI]. [SI] is preserved unless SI equals DI.

FloatCos Cosine of a float

DI	Pointer to destination floating point operand.
SI	Pointer to floating point function argument.
RETURN: Carry clear	
[DI]	
RETURN: Carry Set	
ArgumentErr	Invalid float, or argument not in range.
PANIC: None	

Calculates the Cosine in radians of a double argument in [SI] returning the result in [DI]. [SI] is preserved unless SI equals DI.

FloatExp **Exponentiation of a float**

DI Pointer to destination floating point operand.

SI Pointer to floating point function argument.

RETURN: Carry clear

[DI]

RETURN: Carry Set

ArgumentErr Invalid float.

PANIC: None

Calculates the exponentiation of a double argument in [SI] returning the result in [DI]. [SI] is preserved unless SI equals DI.

FloatInt **Zero fractional part of a Float**

DI Pointer to destination floating point operand.

SI Pointer to floating point function argument.

RETURN: Carry clear

[DI]

RETURN: Carry Set

ArgumentErr Invalid float.

PANIC: None

Removes the fractional part of the float in [SI] and returns the result as a float in [DI]. [SI] is preserved unless SI equals DI.

FloatLn **Natural logarithm of a float**

DI Pointer to destination floating point operand.

SI Pointer to floating point function argument.

RETURN: Carry clear

[DI]

RETURN: Carry Set

ArgumentErr Invalid float, or argument not greater than zero.

PANIC: None

Calculates the Natural Logarithm of a double argument in [SI] returning the result in [DI]. [SI] is preserved unless SI equals DI.

FloatLog **Logarithm of a float**

DI Pointer to destination floating point operand.

SI Pointer to floating point function argument.

RETURN: Carry clear

[DI]

RETURN: Carry Set

ArgumentErr Invalid float, or argument not greater than zero.

PANIC: None

Calculates the Logarithm of a double argument in [SI] returning the result in [DI]. [SI] is preserved unless SI equals DI.

FloatMod Modulo of a float

DI Pointer to destination floating point operand.
 SI Pointer to floating point function argument.
 DX Pointer to floating point modulo value.

RETURN: Carry clear
 [DI]

RETURN: Carry Set
 ArgumentErr Invalid float.
 OverflowErr Integer overflow.

PANIC: None

Calculates [SI] modulo [DX] returning the result in [DI]. The calculation is $[DI] = [SI] - \text{FloatInt}([SI]/[DX]) * [DX]$. [SI] and [DX] is preserved unless SI or DX equals DI.

FloatPow Power of two Floats

DI Pointer to destination floating point operand.
 SI Pointer to floating point base.
 DX Pointer to floating point power.

RETURN: Carry clear
 [DI]

RETURN: Carry Set
 ArgumentErr Invalid float, 0^0 , or $[SI] < 0$ with [DI] not integral.
 OverflowErr Integer overflow.

PANIC: None

Calculates [SI] raised to the power of [DI] returning the result in [DI]. [SI] and [DX] are preserved unless SI or DI equals DI.

FloatRand Float random number

DI Pointer to destination floating point number.
 SI Pointer to unsigned long integer seed.

RETURN: Carry clear
 [DI]

RETURN: Carry Set
 None

PANIC: None

Generates a pseudo random number using the unsigned long integer seed in [SI] and returns the result in [DI]. [SI] is preserved unless SI equals DI.

FloatSin Sine of a Float

DI Pointer to destination floating point operand.
 SI Pointer to floating point function argument.

RETURN:

RETURN: Carry clear
 [DI]

RETURN: Carry Set
 ArgumentErr Invalid float, or argument not in range.

PANIC: None

Calculates the Sine in radians of a double argument in [SI] returning the result in [DI]. [SI] is preserved unless SI equals DI.

FloatSqrt **Square root of a float**

DI Pointer to destination floating point operand.

SI Pointer to floating point function argument.

RETURN: Carry clear

[DI]

RETURN: Carry Set

ArgumentErr Invalid float or argument less than 0.

PANIC: None

Calculates the square root of a double argument in [SI] returning the result in [DI]. [SI] is preserved unless SI equals DI.

FloatTangent **Tangent of a float**

DI Pointer to destination floating point operand.

SI Pointer to floating point function argument.

RETURN: Carry clear

[DI]

RETURN: Carry SetOverflowErr Input argument equals $\pi/2$.**PANIC:** None

Calculates the Tangent in radians of a double argument in [SI] returning the result in [DI]. [SI] is preserved unless SI equals DI.

CHAPTER 16

CHARACTER MANAGEMENT

• CharIsDigit Character is a digit

AL The character to be tested.

RETURN:

 Z flag = 0 If character is a digit.

 Z flag = 1 If character is not a digit.

PANIC: None

Returns the flags set depending on whether the character in AL is a digit. This is a language dependent service.

• CharIsHexDigit Character is a hexadecimal digit

AL The character to be tested.

RETURN:

 Z flag = 0 If character is a hexadecimal digit.

 Z flag = 1 If character is not a hexadecimal digit.

PANIC: None

Returns the flags set depending on whether the character in AL is a hexadecimal digit. This is a language dependent service.

• CharIsPrintable Character is printable

AL The character to be tested.

RETURN:

 Z flag = 0 If character is printable.

 Z flag = 1 If character is not printable.

PANIC: None

Returns the flags set depending on whether the character in AL is printable. This is a language dependent service.

• CharIsAlphabetic Character is alphabetic

AL The character to be tested.

RETURN:

 Z flag = 0 If character is alphabetic.

 Z flag = 1 If character is not alphabetic.

PANIC: None

Returns the flags set depending on whether the character in AL is alphabetic. This is a language dependent service.

• CharIsAlphaNumeric **Character is alphabetic or digit**

AL The character to be tested.

RETURN:

Z flag = 0 If character is alphanumeric.
Z flag = 1 If character is not alphanumeric.

PANIC: None

Returns the flags set depending on whether the character in AL is alphanumeric. This is a language dependent service.

• CharIsUpperCase **Character is upper case**

AL The character to be tested.

RETURN:

Z flag = 0 If character is upper case.
Z flag = 1 If character is not upper case.

PANIC: None

Returns the flags set depending on whether the character in AL is upper case. This is a language dependent service.

• CharIsLowerCase **Character is lower case**

AL The character to be tested.

RETURN:

Z flag = 0 If character is lower case.
Z flag = 1 If character is not lower case.

PANIC: None

Returns the flags set depending on whether the character in AL is lower case. This is a language dependent service.

• CharIsSpace **Character is space**

AL The character to be tested.

RETURN:

Z flag = 0 If character is space.
Z flag = 1 If character is not space.

PANIC: None

Returns the flags set depending on whether the character in AL is a space. This is a language dependent service.

• CharIsPunctuation **Character is punctuation**

AL The character to be tested.

RETURN:

Z flag = 0 If character is punctuation.
Z flag = 1 If character is not punctuation.

PANIC: None

Returns the flags set depending on whether the character in AL is punctuation. This is a language dependent service.

• CharIsGraphic Character is graphic

AL The character to be tested.

RETURN:

Z flag = 0 If character is graphic.
Z flag = 1 If character is not graphic.

PANIC: None

Returns the flags set depending on whether the character in AL is graphic. This is a language dependent service.

• CharIsControl Character is control

AL The character to be tested.

RETURN:

Z flag = 0 If character is control.
Z flag = 1 If character is not control.

PANIC: None

Returns the flags set depending on whether the character in AL is control. This is a language dependent service.

• CharToUpperChar Characters to upper case

AL A character to be converted.

AH A character to be converted.

RETURN:

AL Converted to upper case.
AH Converted to upper case.

PANIC: None

Converts the characters in AH and AL to upper case. This is a language dependent service.

• CharToLowerChar Characters to lower case

AL A character to be converted.

AH A character to be converted.

RETURN:

AL Converted to lower case.
AH Converted to lower case.

PANIC: None

Converts the characters in AH and AL to lower case. This is a language dependent service.

• CharToFoldedChar Characters to folded characters

AL A character to be folded.

AH A character to be folded.

RETURN:

AL Folded.
AH Folded.

PANIC: None

Folds the characters in AH and AL. This is a language dependent service.

CHAPTER 17

BUFFER MANAGEMENT

• BufferCopy Copy one buffer to another

DS:SI	Pointer to source buffer.
ES:DI	Pointer to target buffer.
CX	Number of bytes to copy.

RETURN: None

PANIC: None

Copies CX bytes of data from the source buffer to the target buffer.

The service is optimised to perform the copy in words. If the target buffer pointer is greater than the source buffer pointer, the copy will be done backwards so as to avoid the possibility of corrupting the source buffer during the copy. In most cases it is better to use the REP MOVSW 80C86 instruction.

• BufferSwap Swap the contents of two buffers

DS:SI	Pointer to one of the buffers.
ES:DI	Pointer to the other buffer.
CX	The number of bytes to swap.

RETURN: None

PANIC: None

Swap the contents of the two buffers pointed to by SI and DI. CX bytes of data will be swapped. This service is optimised to use words.

• BufferCompare Compare one buffer with another

DS:SI	Pointer to left operand buffer.
ES:DI	Pointer to right operand buffer.
CX	Number of bytes in the left operand.
BX	Number of bytes in the right operand.

RETURN:

Flags < 0	If [SI] < [DI]
Flags = 0	If [SI] = [DI]
Flags > 0	If [SI] > [DI]

PANIC: None

Compares two buffers for equality. The flags are set in the same way as for a normal compare for integers, i.e. CMP [DI], [SI] but only sets the flags such that the unsigned comparisons can be used. The signed comparisons such as JAE etc. will lead to unpredictable results. This service makes the comparison dependent on the case.

• BufferCompareFolded Compare a buffer with another folded

DS:SI	Pointer to left operand buffer.
ES:DI	Pointer to right operand buffer.
CX	Number of bytes in the left operand.
BX	Number of bytes in the right operand.

RETURN:

Flags < 0	If [SI] < [DI]
Flags = 0	If [SI] = [DI]
Flags > 0	If [SI] > [DI]

PANIC: None

Compares two buffers for equality. The flags are set in the same way as for a normal compare for integers, i.e. CMP [DI], [SI] but only sets the flags such that the unsigned comparisons can be used. The signed comparisons such as JAE etc. will lead to unpredictable results.

This service is case independent and is language dependent.

• BufferLocate Locate a character in a buffer

AH	The character to be located.
DS:SI	Pointer to the buffer to be searched.
CX	The number of bytes in the buffer to be searched.

RETURN: Carry clear

AX	Index of the character in the string.
----	---------------------------------------

RETURN: Carry set

AL undefined	Character is not in the string.
--------------	---------------------------------

PANIC: None

Locates the character in AH in the buffer pointed to by SI. CX bytes of the buffer will be searched. If the character is located then the index into the buffer is returned in AX. The index of the first character in the buffer is 0.

This service is case dependent.

• BufferLocateFolded Locate a character in a buffer folded

AH	The character to be located.
DS:SI	Pointer to the buffer to be searched.
CX	The number of bytes in the buffer to be searched.

RETURN: Carry clear

AX	Index of the character in the string.
----	---------------------------------------

RETURN: Carry set

AL undefined	Character is not in the string.
--------------	---------------------------------

PANIC: None

Locates the character in AH in the buffer pointed to by SI. CX bytes of the buffer will be searched. If the character is located then the index into the buffer is returned in AX. The index of the first character in the buffer is 0.

This service is case independent.

• BufferSubBuffer Find a sub-buffer in a buffer

DS:SI	Pointer to the buffer to be searched.
ES:DI	Pointer to the buffer to be located.
CX	The number of bytes in buffer being searched.
BX	The number of bytes in sub-buffer.

RETURN: Carry clear

AX	Offset of the sub-buffer within the buffer.
----	---

RETURN: Carry set

AL undefined	Sub-buffer not found.
--------------	-----------------------

PANIC: None

Locates a buffer as a sub-buffer within another buffer.

If the buffer pointed to by DI and of length BX is a sub-buffer of the buffer pointed to by SI and of length CX, then the service returns the index of the first occurrence of the sub-buffer within the buffer pointed to by SI and with the carry flag clear. The index of the first character in the buffer is 0. If the buffer is not a sub-buffer, then the service returns the carry flag set.

This service is case dependent.

• BufferSubBufferFolded Find a sub-buffer in a buffer folded

DS:SI	Pointer to the buffer to be searched.
ES:DI	Pointer to the buffer to be located.
CX	The number of bytes in buffer being searched.
BX	The number of bytes in sub-buffer.

RETURN: Carry clear

AX	Offset of the sub-buffer in the buffer.
----	---

RETURN: Carry set

AL undefined.	Sub-buffer not found.
---------------	-----------------------

PANIC: None

Locates a buffer as a sub-buffer within another buffer.

If the buffer pointed to by DI and of length BX is a sub-buffer of the buffer pointed to by SI and of length CX, then the service returns the index of the first occurrence of the sub-buffer within the buffer pointed to by SI, and with the carry flag clear. The index of the first character in the buffer is 0. If the buffer is not a sub-buffer, then the service returns the carry flag set.

This service is case independent.

• BufferMatch Match a wild card buffer

DS:SI	Pointer to the buffer to be searched.
CX	Length of the buffer to be searched.
ES:DI	Pointer to the wild card match buffer.
DX	Length of the match buffer.

RETURN: Carry clear

Match

RETURN: Carry set

AL undefined	No match.
--------------	-----------

PANIC: None

Search a buffer for a match with the supplied wild card buffer.

If the wild card buffer matches then the service will return with carry clear. If the wild card buffer does not match then the service will return with carry set. The `MatchAny` character will match any set of characters. The `MatchSingle` character will match any single character.

This service is case dependent.

• BufferMatchFolded Match a wild card buffer folded

DS:SI	Pointer to the buffer to be searched.
CX	Length of the buffer to be searched.
ES:DI	Pointer to the wild card match buffer.
DX	Length of the match buffer.

RETURN: Carry clear

Match

RETURN: Carry set

AL undefined No match.

PANIC: None

Search a buffer for a match with the supplied wild card buffer.

If the wild card buffer matches then the service will return with carry clear. If the wild card buffer does not match then the service will return with carry set. The `MatchAny` character will match any set of characters. The `MatchSingle` character will match any single character.

This service is case independent.

• BufferJustify Justify a buffer

DS:SI	Pointer to the buffer to be justified.
CX	Length of the buffer to be justified.
ES:DI	Pointer to the target buffer.
BX	Length of the target buffer.
DL	JustifyLeft or JustifyCentre or JustifyRight.
DH	The fill character.

RETURN:

AX Points to the character after the last byte copied to the target buffer.

PANIC: None

Justifies a source buffer DS:SI of length CX into a target buffer ES:DI of length BX using the justification method passed in DL and the fill character passed in DH. If BX is negative, then BX bytes are just copied to the target buffer. If DL is not one of the 3 options then the left justified method will be used by default.

CHAPTER 18

STRING MANAGEMENT

• StringCopy Copy one string to another

DS:SI Pointer to source string.

ES:DI Pointer to target string.

RETURN: None

PANIC: None

Copies the source string to the target string.

• StringCopyFolded Copy one string to another folded

DS:SI Pointer to source string.

ES:DI Pointer to target string.

RETURN: None

PANIC: None

Copies the source string to the target string. Characters are folded as they are copied.

• StringConvertToFolded Convert a string to folded

DS:SI Pointer to string to be folded.

RETURN: None

PANIC: None

Converts the string pointed to by SI to folded.

• StringCapitalise Capitalise a string

DS:SI Pointer to string to be capitalised.

RETURN: None

PANIC: None

Converts the string pointed to by SI so that the first letter is uppercase and the remaining characters are lowercase.

• StringCompare Compare one string with another

DS:SI Pointer to left operand string.

ES:DI Pointer to right operand string.

RETURN:

Flags < 0 If [SI] < [DI]

Flags = 0 If [SI] = [DI]

Flags > 0 If [SI] > [DI]

PANIC: None

Compares two strings for equality.

The flags are set in the same way as for a normal compare for integers, i.e. CMP [DI], [SI], but only sets the flags such that the unsigned comparisons can be used. The signed comparisons such as JAE etc. will lead to unpredictable results.

This service is case dependent.

• StringCompareFolded Compare a string with another folded

DS:SI Pointer to left operand string.
ES:DI Pointer to right operand string.

RETURN:

Flags < 0 If [SI] < [DI]
Flags = 0 If [SI] = [DI]
Flags > 0 If [SI] > [DI]

PANIC: None

Compares two strings for equality.

The flags are set in the same way as for a normal compare for integers, i.e. CMP [DI], [SI], but only sets the flags such that the unsigned comparisons can be used. The signed comparisons such as JAE etc. will lead to unpredictable results.

This service is case independent and language dependent.

• StringMatch Match a wild card string

DS:SI Pointer to the string to be searched.
ES:DI Pointer to the wild card match string.

RETURN: Carry clear

Match

RETURN: Carry set

AL undefined No match.

PANIC: None

Search a string for a match with the supplied wild card string.

If the wild card string matches then the service will return with carry clear. If the wild card string does not match then the service will return with carry set. The MatchAny character will match any set of characters. The MatchSingle character will match any single character.

This service is case dependent.

• StringMatchFolded Match a wild card string folded

DS:SI Pointer to the string to be searched.
ES:DI Pointer to the wild card match string.

RETURN: Carry clear

Match

RETURN: Carry set

AL undefined. No match.

PANIC: None

Search a string for a match with the supplied wild card string.

If the wild card string matches then the service will return with carry clear. If the wild card string does not match then the service will return with carry set. The MatchAny character will match any set of characters. The MatchSingle character will match any single character.

This service is case independent.

• StringLocate Locate a character in a string

AH The character to be located.
 DS:SI Pointer to the string to be searched.

RETURN: Carry clear

AX Index of the character in the string.

RETURN: Carry set

AL undefined Character is not in the string.

PANIC: None

Locates the character in AH within the string pointed to by SI.

If the character is located then the index into the string is returned in AX. The index of the first character in the string is 0.

This service is case dependent.

• StringLocateFolded Locate a character in a string folded

AH The character to be located.
 DS:SI Pointer to the string to be searched.

RETURN: Carry clear

AX Index of the character in the string.

RETURN: Carry set

AL undefined Character is not in the string.

PANIC: None

Locates the character in AH within the string pointed to by SI.

If the character is located then the index into the string is returned in AX. The index of the first character in the string is 0.

This service is case independent.

• StringLocateInReverse Locate a character in reverse

AH The character to be located.
 DS:SI Pointer to the string to be searched.

RETURN: Carry clear

AX Index of the character in the string.

RETURN: Carry set

AL undefined Character is not in the string.

PANIC: None

Locates the character in AH within the string pointed to by SI in reverse order.

If the character is located then the index into the string is returned in AX. The index of the first character in the string is 0.

This service is case dependent.

• StringLocateInReverseFolded Locate char' in reverse folded

AH The character to be located.
 DS:SI Pointer to the string to be searched.

RETURN: Carry clear

AX Index of the character in the string.

RETURN: Carry set

AL undefined Character is not in the string.

PANIC: None

Locates the character in AH within the string pointed to by SI in reverse order.

If the character is located then the index into the string is returned in AX. The index of the first character in the string is 0.

This service is case independent.

• StringSubString Find a substring in a string

DS:SI Pointer to the string to be searched.

ES:DI Pointer to the string to be located.

RETURN: Carry clear

AX Offset of the substring in the string.

RETURN: Carry set

AL undefined Substring not found.

PANIC: None

Locates a string as a substring within another string.

If the string pointed to by DI is a substring of the string pointed to by SI, then the service returns the index of the first occurrence of the substring; the carry flag is clear. The index of the first character in the string is 0.

If the string is not a substring, then the service returns with the carry flag set.

This service is case dependent.

• StringSubStringFolded Find a substring in a string folded

DS:SI Pointer to the string to be searched.

ES:DI Pointer to the string to be located.

RETURN: Carry clear

AX Offset of the substring in the string.

RETURN: Carry set

AL undefined Substring not found.

PANIC: None

Locates a string as a substring within another string.

If the string pointed to by DI is a substring of the string pointed to by SI, then the service returns the index of the first occurrence of the substring; the carry flag is clear. The index of the first character in the string is 0.

If the string is not a substring, then the service returns with the carry flag set.

This service is case independent and language dependent.

• StringLength Length of a string

ES:DI Pointer to the string whose length is to be found.

RETURN:

AX The length of string.

PANIC: None

Returns the length of the string pointed to by DI excluding the terminating zero.

• StringValidateName **Validate a system name**

AL	Maximum number of characters in the name allowed in the name.
AH	Non 0 - An extension is valid. 0 - An extension is invalid.
ES:DI	Pointer to the string to be validated.

RETURN: Carry clear
String is valid.

RETURN: Carry set
NameErr String is not valid.

PANIC: None

Validates that the string at DI points to a valid system name.

An extension is only allowed if AH is non zero. AL specifies the number of characters that can be in the name before the period.

The BNF for a valid name is as follows:

```
ANY := ALPHA | DIGIT | $ | _  
NAME := ALPHA [ [ANY]*7 [. [ANY]*3 ] ]
```


CHAPTER 19

GENERAL MANAGEMENT

Version numbers

A version number is a 16 bit integer. If the 16 bit integer is converted to 4 hex digits, i.e. XYYZ, then the version is:

X.YYZ

where X is the major release number, YY is the minor release number and Z is the version type. The version type may have three values.

- A - Alpha release.
- B - Beta release.
- F - Final release.

Thus a typical version number 0x100f would be 1.00F.

GenVersion	Get the operating system version number
------------	---

None

RETURN:

AX

The operating system version number.

PANIC: None

Returns the operating system version number as a 16 bit integer.

GenRomVersion	Get the ROM version number
---------------	----------------------------

None

RETURN:

AX

The ROM version number.

PANIC: None

Returns the ROM version number.

The operating system is never released on its own and is always supplied with a number of files in its built in ROM disk. The tool which builds the operating system together with the ROM disk allows a version number to be specified. This service can be used to retrieve the version number so specified.

GenLcdType **Get the system LCD type**

None

RETURN:

AL The LCD type.

PANIC: NoneReturns the system LCD type. The various types are defined in *epocdefs.inc*.**GenStartReason** **Get the system cold start reason**

None

RETURN:

AL The cold start reason.

PANIC: None

Returns the system cold start reason. The operating system can perform a cold start for five reasons:

- Initialise; system RAM is invalid.
- Power fail; the system was forced to power down, but RAM is still valid.
- Reset; the user requested a reset, but RAM is still valid.
- Kernel fault; a serious fault occurred while executing in the operating system kernel.
- New O/S restart; a new operating system has been programmed into the flash memory and a restart has occurred.

The shell on first starting up should request the cold start reason and if it is not an initialise it should inform the user of the reason for the cold start.

• GenDataSegment **Get the operating system data segment**

None

RETURN:

ES The operating system data segment.

PANIC: None

Returns the operating system's data segment address. This service is reserved for use by system utilities.

GenGetCountryData **Get the country data**

BX Pointer to a CDataEnt structure.

RETURN: None**PANIC:** None

Returns the country dependent data currently installed. The operating system on start-up copies the country dependent data from the configuration file into RAM so that the GenSetCountryData can be called to update the information.

GenSetCountryData **Set the country data**

ES:BX Pointer to a CDataEnt structure.

RETURN: None**PANIC:** None

Sets the country dependent data. The source of the data is the CDataEnt structure pointed to by ES:BX.

GenGetOsData**Get the O/S data**

DI	Pointer to a buffer to receive the data.
SI	Offset in the operating system data space.
CX	The number of bytes to be copied.

RETURN: None**PANIC:** None

Copies data from the operating system's data space to the buffer provided.

All operating system handles are the actual address in the operating system data space of the appropriate control entry. For example, the handle returned by the `FileExecute` service is the address in the operating system data space of the `E_PROC` process control entry. By fetching this data, a program can determine many things about the state of a process.

GenGetErrorText**Get error text**

AL	The error number.
BX	Pointer to a buffer to receive the error text.

RETURN: None**PANIC:** None

Returns the text associated with an error number.

The buffer pointed to by BX must be `MaxErrorTextSize` in length. If AL is not negative or it contains an error unknown to the operating system, then "Unknown error [-xx]" will be returned where xx is the unknown error number.

This is a language dependent service.

• Dummy**Dummy service**

None

RETURN: None**PANIC:** None

This service provides a means for generating a call to a known location in the operating system. It is used mainly for debugging the operating system. The service itself does nothing at all.

GenParse**Generic file name parser**

BX	Pointer to a <code>GenParseEnt</code> structure.
----	--

RETURN: Carry clear

Success

RETURN: Carry set

NameErr	Invalid name.
---------	---------------

PANIC: None

This service provides a generic parse service which can be used to parse file names.

This service should not be confused with the `FileParse` service. `FileParse` calls the file server which in turn calls a file system to parse the file name and this will always be successful, assuming that the file name obeys the naming rules for the target file system.

This service can only parse generic MSDOS like file names. The generic parser considers a file name to consist of up to five components:

SystemName Drive Path Name Extension

A SystemName consists of a FileSystemName followed by two ':'.s.

A Drive consists of a DriveName followed by DriveSeparator.

A Path consists of a PathSeparator followed by zero or more DirectoryName PathSeparator pairs.

An Extension consists of an ExtensionSeparator followed by an ExtensionName.

The GenParseEnt structure allows a single character to be specified for each of the separators and the maximum size for each of the four components. Note that the maximum size of a component includes any separators. Note also that the SystemName separator of two ':'s cannot be specified.

For MSDOS filing systems, the values which should be loaded into the structure are as follows:

```
GenParseDeviceSeparator= ':'
GenParsePathSeparator=  '\\'
GenParseExtSeparator    =  '.'
GenParseMaxDeviceSize=  2
GenParseMaxPathSize     =  64
GenParseMaxNameSize     =  8
GenParseMaxExtSize      =  4
```

The remaining fields of the structure specify pointers to three input names, a pointer to the output buffer and a pointer to a FullParseEnt structure.

Parsing is effected as follows. The three input strings are prioritised in the order GenParseSourceNamePtr, GenParseRelatedNamePtr and GenParseDefaultsPtr. Each of these input strings is parsed into its four components (some of which may be missing). The resulting string is built by taking components from the first string. Any missing components are filled in from the second string (where they exist). Any components still missing are filled in from the third string. Thus, if the "source" string does not have a drive, but the "related" string does, then the resulting name will use the drive as specified in the "related" string.

When the resulting string has been built, the size of each component is put into the FullParseEnt structure.

Finally, the name and extension components are examined for the wild card characters '?' and '*' and, if found, the ParsewildName and ParsewildExt flags are set appropriately in the FullParseEnt.FullParseFlags field. If either ParsewildName or ParsewildExt is set then ParsewildAny is also set.

GenDeferredMode Set deferred mode

AL	0	- Increment deferred mode.
	Non 0	- Decrement deferred mode

RETURN: None

PANIC: None

This service only applies to the MC version of the operating system and can be used to defer some of the work which is performed in the 32Hz. tick interrupt.

Calling the service with AL equal to zero will defer keyboard and mouse polling, parallel I/O polling and the piezo sound system. This has the additional benefit of stopping the serial channel from being temporarily diverted by the tick interrupt service routines. Normal operation can be resumed by calling this service with AL set to a non zero value.

If the tick interrupt overhead must be reduced further then the anti-nesting flag can be incremented. This is a byte at address 0438h in the operating system data space. While this flag is set, only the time is kept up to date but be warned, pre-emptive multi tasking is disabled as are all timer services.

GenNotify Notify by text

BX	Pointer to the first message.
CX	Pointer to the second message or zero.
DX	Pointer to the first option or zero.
DI	Pointer to the second option or zero.
SI	Pointer to the third option or zero.

RETURN: Carry clear

AL

- 0 - First option chosen by the user.
- 1 - Second option chosen by the user
- 2 - Third option chosen by the user.

RETURN: Carry set

FailErr

No notify process running.

PANIC: None

This service will send a message to the notification process and await the result from the notifier, returning the result in AL. If a notifier is not currently running then FailErr will be returned.

BX and CX specify two zero terminated text messages which will be displayed by the notifier process. Each string can be up to MaxNotifyTextSize in length including the zero terminator. CX can be optionally zero, in which case the second message line will be blank.

DX, DI and SI specify up to three options which the user may select. The selected option is returned in AL and will be 0 if the DX option is chosen, 1 if the DI option is chosen and 2 if the SI option is chosen. By convention, if all the options are specified as 0 then this is the same as having DX point to an option of "CONTINUE". Each option string can be up to MaxOptionTextSize in length including the zero terminator. Finally if DI is 0 then SI should also be 0.

The presentation of the notifier depends on the process which has hooked the notify interface.

GenNotifyError

Notify by error number

AL

The error number to be notified.

BX

Pointer to the first message.

DX

Pointer to the first option or zero.

DI

Pointer to the second option or zero.

SI

Pointer to the third option or zero.

RETURN: Carry clear

AL

- 0 - First option chosen by the user.
- 1 - Second option chosen by the user.
- 2 - Third option chosen by the user.

RETURN: Carry set

FailErr

No notify process running.

PANIC: None

This service first calls GenGetErrorText using AL as the parameter and then calls the GenNotify service with CX pointing to the resultant error text, all other registers being the same.

Only error numbers catered for by the configuration file should be notified using this service. It is reasonable to expect that all errors returned by the operating system can be notified with this service.

GenNotifyHook

Hook the notify interface

BX

The message number.

RETURN: Carry clear

Success

RETURN: Carry set

FailErr

Notify interface already hooked.

PANIC: None

This service allows a process to get a message in response to calls by all other processes to the GenNotify and GenNotifyError services.

The message will be delivered with the message number specified in BX and the message buffer will contain 5 words. The 5 words will consist of the 5 parameters to the GenNotify service in the order BX, CX, DX, DI and SI. Note that as 5 words need to be delivered, a process which hooks the notify interface should initialise messaging using the MessInit service with a size of at least 10 in BL.

The text string pointed to by the 5 parameters can be fetched from the requesting process using the `ProcCopyFromById` service. The result should be returned in CX when the `MessFree` service is called to give the result of the notification.

A process which has hooked the notify interface should not call either the `GenNotify` or the `GenNotifyError` services as it would then try and send itself a message, resulting in a lock up situation. It is probably wise for the process to disable file server notifies for itself by calling the `GenSetNotifyState` to off, as it might be waiting for a file request to complete when the file server sends a notify message, resulting in lock up.

If the process which has hooked the notify interface either exits or is panicked, then the supervisor will automatically free the interface so that another process can hook it.

GenNotifyUnHook **Unhook the notify interface**

None

RETURN: None

PANIC:

PanicGen1 Process does not have the interface hooked.

This service will release the notify interface provided the process calling this service already has the interface hooked. If not, then the process will be panicked.

GenGetRamSizeInParas **Get addressable system RAM size**

None

RETURN:

AX Size of system ram in paragraphs.

PANIC: None

Returns the size, in paragraphs, of the currently addressable system RAM. On machines containing more than 512 kilobytes of RAM, this is not the same as the total amount of RAM that is fitted in the machine.

GenGetCommandLine **Get the command line**

None

RETURN:

AX Pointer to the command line or 0.

PANIC: None

Returns a pointer to the command line.

The command line is a memory cell in the heap and can be freed if required with `HeapFreeCell`. Processes can also be started with no command line in which case this service will return 0.

The address of the command line is also stored in the global variable `DS:[DataCommandPtr]`. If the command line is freed then, for consistency, this global variable should be set to 0.

The structure of the command line is a zero terminated string which contains the full path name used to start the process. This can be used to find other files associated with the process being run or to open the image file in order to access either added files or added DYLS.

After the 0 of the zero terminated string is a leading byte string containing any arguments for the process. The string is leading byte counted so that binary arguments can be passed to programs. If the `FileExecute` service is called with CX equal to 0 then no command line is passed. This should only be used to execute programs with no heap, since they obviously have nowhere to store the command line. It is preferable to have CX pointing to a string containing just the zero terminator.

GenGetSoundFlags**Get the sound flags**

None

RETURN:

AX

The sound flags.

PANIC: None

This service returns the current setting in the sound flags. The bits in the sound flags are as follows:

- SoundKeyboardEnable - If set, will enable keyboard clicks.
- SoundBuzzerEnable - If set, will enable the piezo sound system.
- SoundDeviceEnable - If set, will enable the SND: device driver.
- SoundLoud - If set, will make the piezo sound louder.
- SoundDisable - If set, will disable all sound in the system.

GenSetSoundFlags**Set the sound flags**

BX

The new sound flags.

RETURN: None**PANIC:** None

This service sets the sound flags to the value in BX. The bits in the sound flags are as follows:

- SoundKeyboardEnable - If set, will enable keyboard clicks.
- SoundBuzzerEnable - If set, will enable the piezo sound system.
- SoundDeviceEnable - If set, will enable the SND: device driver.
- SoundLoud - If set, will make the piezo sound louder.
- SoundDisable - If set, will disable all sound in the system.

GenSound**Make sound with the piezo**

BX

The duration of the sound in ticks.

CX

The pitch of the sound.

RETURN: None**PANIC:** None

This service will make a sound through the piezo for the duration specified in BX ticks and at the pitch specified in CX. The pitch can be calculated as $(512/CX)$ KHz. The piezo uses very little power and is an easy way of generating sound, although it is quite soft. If greater sound complexity, or a louder sound is required then the SND: device driver can be used.

Access to this service is controlled by a semaphore which has been pre-counted with 1. When service is requested, the semaphore is waited on and when completed the sound request is run. The service then returns to the process requesting the service. When the duration elapses, the semaphore is signalled, allowing the next service request to be processed. The effect of the above, assuming the piezo is not already in use, is that the first call to this service will complete immediately allowing the application to go about its business, but subsequent calls will wait until the current request is completed. If multiple processes make requests on this service, they are run on a first come first served basis.

GenMarkActive**Mark a process as active**

None

RETURN: None**PANIC:** None

This service informs the operating system that the process invoking this service is to be considered active.

The operating system has the ability to auto switch off if no activity takes place within a certain length of time. Activity is considered to be a context switch to a process which has been marked active, i.e. whenever the process executes, the timer controlling the auto switch off will be reset.

By default, all processes, when first created, are marked as active so that this service does not need to be called unless the `GenMarkNonActive` service has been called.

GenMarkNonActive **Mark a process as non-active**

None

RETURN: None

PANIC: None

This service will inform the operating system that the process invoking this service is not to be considered active.

The operating system has the ability to auto switch off if no activity takes place within a certain length of time. Activity is considered to be a context switch to a process which has been marked active. Hence marking a process as non-active will ensure that whenever the process executes, the timer controlling the auto switch off will not be reset.

By default all processes, when first created, are marked as active so that this service must be called if the process is not to be considered as active. All servers must mark themselves as non-active, since they only execute when required by clients and the status of the client will determine activity or not. Thus if a client of the file server is non-active and requests some file activity, it will not be considered as activity because the file server is also marked as non-active. However if the client is active then by virtue of making the request to the file server, the auto switch off timer will be reset.

If, for example, a program was left running displaying the time every second, by default, the machine would never switch off as every second the process would execute resetting the auto switch off timer, possibly not a desirable state of affairs. By marking the process as non-active then the activity of the process would not reset the timer and the machine would be able to switch off.

GenGetText **Get operating system text**

AL The number of the text message to be retrieved.

ES:BX Pointer to the buffer to receive the text.

RETURN: Carry clear

Success

RETURN: Carry set

AL undefined Failed to find the message.

PANIC: None

This service will scan the operating system's built in configuration file for the text message associated with the number in AL. This service is similar to `GenGetErrorText` when AL is negative but can also be passed positive numbers. The text messages available depend entirely on the configuration file built into the ROM with the operating system.

GenGetNotifyState **Get notify state**

None

RETURN:

AL The notify state.

PANIC: None

This service will get the current notify state for the process.

The file server, when it detects a problem which the user could possibly correct, will call the notifier process to inform the user of the error and any action which must be performed (e.g. replacing an SSD which had been accidentally removed before all files open on it were closed).

If the state is 0 then the file server will not call the notifier and will return the error directly. If the state is 1 then the notifier will be called.

Some applications are intended to work in an unattended fashion so that a request for user attention would be to no avail. In this case, the state should be set to 0 so that the process itself can take any action required. By default, processes have the state set to 1.

GenSetNotifyState**Set notify state**

AL The notify state.

RETURN: None

PANIC: None

This service will set the current notify state for the process.

The file server, when it detects a problem which the user could possibly correct, will call the notifier process to inform the user of the error and any action which must be performed (e.g. replacing an SSD which had been accidentally removed before all files open on it were closed).

If the state is 0 then the file server will not call the notifier and will return the error directly. If the state is 1 then the notifier will be called.

Some applications are intended to work in an unattended fashion so that a request for user attention would be to no avail. In this case, the state should be set to 0 so that the process itself can take any action required. By default, processes have the state set to 1.

GenGetAutoSwitchOffValue**Get the auto switch off time**

None

RETURN:

AX The auto switch off time in seconds.

PANIC: None

This service can be used to get the current auto switch off time.

The time, in seconds, is returned in AX. It represents the amount of time which must expire with no activity before the machine will auto switch off. If the value is -1 then auto switch off is disabled. By default, the auto switch off is set to 300 seconds.

GenSetAutoSwitchOffValue**Set the auto switch off time**

BX The auto switch off time in seconds.

RETURN: None

PANIC: None

This service can be used to set the auto switch off time.

The time, in seconds, is passed in BX. It represents the amount of time which must expire with no activity before the machine will auto switch off. If the value is -1 then auto switch off is disabled. By default, the auto switch off is set to 300 seconds.

GenSetRevector**Capture an interrupt**

AL The vector number.

CX:BX The segment and offset on the interrupt routine.

RETURN: None

PANIC: None

The operating system hooks all interrupt vectors to itself and in the case of hardware interrupt vectors and the special interrupt vectors, provides a shell interrupt service routine which will do all the right things to satisfy the operating system rules.

This service allows a new interrupt service to be installed and should only be called by device drivers. As the new interrupt service is being called by the operating service through a shell the following rules apply to the interrupt service routine.

- All registers may be destroyed except BP,SP and SS.
- The routine should return with a far ret and not an iret.
- If a re-schedule is required, then it should return with carry set; if not then carry should be clear.

GenGetSuffixes**Get suffix text**

ES:BX

Pointer to buffer to receive the suffix text.

RETURN: None**PANIC:** None

This service will copy the language dependent suffixes from the configuration file into the buffer pointed to by ES:BX.

The suffixes are fixed length zero terminated strings with a maximum length of three bytes including the zero terminator.

Suffixes follow numbers for the day of the month (for example, the *st* in 1st. September 1990). Hence there are 31 suffixes so that ES:BX must point to a buffer of at least 31*3 bytes.

This is a language dependent service.

GenGetAmPmText**Get the AM and PM text**

AL

Zero - Get AM text

Non zero - Get PM text.

ES:BX

Pointer to buffer to receive the AM and PM text.

RETURN: None**PANIC:** None

This service will copy the language dependent "AM" and "PM" text from the configuration file into the buffer pointed to by ES:BX.

The two text strings are fixed length zero terminated strings with a maximum length of three bytes including the zero terminator. The "am" text is first, followed by the "pm" text. There are 2 strings of 3 bytes each so that ES:BX must point to a buffer of at least 6 bytes.

This is a language dependent service.

GenGetBatteryType**Get the battery type**

None

RETURN:

AL

The battery type.

PANIC: None

This service gets the current battery type.

By default, Epoc sets the battery type to `BatteryUnknown`. The battery types are declared in the header file *epocdefs.inc*.

On machines whose hardware does not support the detection of the battery type, a meaningful result depends on a prior call having been made to `GenSetBatteryType`.

GenSetBatteryType**Set the battery type**

AL

The battery type.

RETURN: None**PANIC:** None

This service sets the current battery type. By default, Epoc sets the battery type to `BatteryUnknown`. The battery types are declared in the header file *epocdefs.inc*.

Epoc needs to know about the various battery types because the levels at which low battery warning messages are issued depends on the type.

If the battery type is `BatteryUnknown` then Epoc gives the same warning levels as for `BatteryAlkaline`.

A call to this function is not required on machines, such as the Workabout, whose hardware supports detection of the battery type.

GenCrc **Generate a CRC**

CX The number of bytes in the buffer.
DX The current CRC.
DS:SI Pointer to the buffer to be CRC checked.

RETURN:

AX The updated CRC check.

PANIC: None

This service will generate a CRC polynomial checksum ($X^{\text{power } 16} + X^{\text{power } 12} + X^{\text{power } 5} + 1$, as recommended by CCITT) from the buffer pointed to by DS:SI containing CX bytes. If the checksum is being started then the value in DX should be passed as 0.

• GenIntByNumber **Interrupt by number**

AL The interrupt number.
DS:SI Pointer to the input register values.
DS:DI Pointer to the output register values.

RETURN:

AX The flags register after the call.

PANIC:

Depends on the
interrupt called.

This service can be used to call any software interrupt and is provided to make calling the operating system easier from high level languages.

The register values are stored sequentially as 6 words and represent the values for AX, BX, CX, DX, SI and DI. BP is never needed by the operating system and so is not required. DS:SI and DS:DI can point to the same memory location. The value returned is the flags register because although the carry flag is the most important, some of the operating system routines also set the arithmetic flags. The carry flag is in bit 0 of the returned result in AX.

GenEnvBufferGet **Get environment variable**

ES:DI Pointer to the environment variable name.
DL Length of environment variable name.
ES:SI Pointer to the buffer to receive the variable's value.

RETURN: Carry clear

AX The length of the data returned in ES:SI.

RETURN: Carry set

NotExistsErr No environment variable of the specified name exists.

PANIC: None

This service will locate an environment variable. The name of the variable is pointed to by ES:DI and has a length of DL bytes.

The name may include wild cards, in which case the first matching name will be found. The value of the environment variable is copied to ES:SI and the length of this data is returned in AX.

The maximum size of an environment variable's data is 255 bytes.

GenEnvBufferSet **Set environment variable**

ES:DI Pointer to the environment variable name.
DL Length of environment variable name.
ES:SI Pointer to the buffer containing the variable's data.
CL The length of the data as ES:SI.

RETURN: Carry clear

Success

RETURN: Carry set

NoMemoryErr

No space available to store environment variable.

FailErr

Environment variable name contained wild cards.

PANIC:

PanicEnv0

DL exceeded MaxEnvNameSize

This service will either add or replace an environment variable. The name of the variable is pointed to by ES:DI and has a length of DL bytes.

The name may not include wild cards nor exceed MaxEnvNameSize. The value of the environment variable is pointed to by ES:SI and the length of the data to be copied is in CL. Since the data is a buffer of length CL there is no restriction on what data may be placed in the buffer.

The maximum size of an environment variable's data is 255 bytes.

GenEnvBufferDelete

Delete environment variable

ES:DI

Pointer to the environment variable name.

DL

Length of environment variable name.

RETURN: Carry clear

Success

RETURN: Carry set

NotExistsErr

No environment variable of the specified name exists.

PANIC: None

This service will delete an environment variable. The name of the variable is pointed to by ES:DI and has a length of DL bytes.

The name may include wild cards in which case the first matching name is deleted.

GenEnvBufferFind

Find environment variable

BX

The find handle.

ES:DI

Pointer to the environment variable name.

DL

Length of environment variable name.

ES:SI

Pointer to the buffer to receive the variable's data.

RETURN: Carry clear

AX

The next find handle.

RETURN: Carry set

EofErr

No more matching environment variables.

PANIC: None

This service will find all occurrences of environment variables which match the supplied wild card name. The wild card name is pointed to by ES:DI and has a length of DL.

A wild card of "*" will locate all environment variables. When this routine is first called, BX must contain zero; on subsequent calls, it must contain the value returned in AX. The wild card match string must remain the same on successive calls. EofErr is returned when there are no more matching names.

After a successful call, the buffer pointed to by ES:SI contains two leading byte strings. The first string contains the name of the environment variable while the second string contains its value. The maximum size of an environment variable's data is 255 bytes.

GenEnvStringGet **Get string environment variable**

ES:DI Pointer to the environment variable name string.
ES:SI Pointer to the buffer to receive the variable's value.

RETURN: Carry clear
Success

RETURN: Carry set
NotExistsErr No environment variable of the specified name exists.

PANIC: None

This service will locate an environment variable. The name of the variable is pointed to by ES:DI. The name may include wild cards, in which case the first matching name will be found.

The value of the environment variable will be copied to ES:SI and will be zero terminated. Environment variables should not include the byte 0 as this would terminate the string prematurely. The maximum length of the string is 256 bytes including the terminating zero.

GenEnvStringSet **Set string environment variable**

ES:DI Pointer to the environment variable name.
ES:SI Pointer to the string containing the variable's data.

RETURN: Carry clear
Success

RETURN: Carry set
NoMemoryErr No space available to store environment variable.
FailErr Environment variable name contained wild cards.

PANIC:
PanicEnv0 The name string exceeded MaxEnvNameSize

This service will either add or replace an environment variable. The name of the variable is pointed to by ES:DI. The name may not include wild cards nor exceed MaxEnvNameSize.

The value of the environment variable is pointed to by the string at ES:SI. The maximum size of the string should be limited to 255 bytes not including the zero terminator. Should the string be longer it will truncated module 256.

GenEnvStringDelete **Delete string environment variable**

ES:DI Pointer to the environment variable name.

RETURN: Carry clear
Success

RETURN: Carry set
NotExistsErr No environment variable of the specified name exists.

PANIC: None

This service will delete an environment variable. The name of the variable is pointed to by ES:DI. The name may include wild cards in which case the first matching name is deleted.

GenEnvStringFind **Find string environment variable**

BX The find handle.
ES:DI Pointer to the environment variable name.
ES:SI Pointer to the buffer to receive the variable's name.
ES:CX Pointer to the buffer to receive the variable's data.

RETURN: Carry clear
AX The next find handle.

RETURN: Carry set

EofErr

No more matching environment variables.

PANIC: None

This service will find all occurrences of environment variables which match the supplied wild card name. The wild card name is pointed to by ES:DI. A wild card of "*" will locate all the environment variables.

When this routine is first called, BX must contain zero; on subsequent calls it must contain the value returned in AX. The wild card match string must remain the same on successive calls. EofErr is returned when there are no more matching names.

After a successful call, the buffer pointed to by ES:SI contains the name of the environment variable which was found, as a zero terminated string. The maximum size of the name of an environment variable is MaxEnvNameSize.

The buffer pointed to by ES:CX contains the value of the environment variable which was found, as a zero terminated string. The maximum size of the data is 256 bytes, including the zero terminator.

GenAlarmHook

Hook the alarm interface

BX

The message number.

RETURN: Carry clear

Success

RETURN: Carry set

FailErr

Alarm interface already hooked.

PANIC: None

This service allows a process to capture the alarm interface built into the operating system. Having hooked the alarm interface the ALM: device driver can be used to request alarms from the alarm server.

The availability of an alarm server and what it does, varies from machine to machine and the appropriate documentation for the specific machine should be consulted.

GenAlarmUnHook

Unhook the alarm interface

None

RETURN: None

PANIC:

PanicGen1

Process does not have the interface hooked.

This service will release the Alarm interface if the process calling this service already has the interface hooked. If it does not, the process will be panicked.

GenAlarmId

Get the pid of the alarm server

None

RETURN:

AX

The alarm server pid.

PANIC: None

This service will return the ID of the alarm server. If the alarm interface is not currently hooked then this service will return zero in AX.

GenTickle

Reset the auto switch off timer

None

RETURN: None

PANIC: None

This service will reset the auto switch off timer to the value as specified to the last GenSetAutoSwitchOffValue. This routine is useful for processes which have called GenMarkNonActive and require to "tickle" the auto switch off from time to time.

GenSetOnEvents**Control on events**

AL The find handle.

RETURN: None

PANIC: None

This service controls whether the system will report on-events(for example, reporting to the window server when the machine switches on). If AL is non-zero then on-events will be reported. If AL is zero, they will not.

By default on-events are reported on Series 3 and Series 3a operating systems and not reported on other versions.

③ GenGetAutoMains**Get state for auto-sw-off if mains present**

RETURN:

AX The current auto-switch-off state for when mains is present.

PANIC: None

Returns a non-zero value in AX if auto-switch-off is disabled when mains is present, and zero if enabled.

③ GenSetAutoMains**Disable/enable auto-sw-off if mains present**

AL Flag specifying whether to enable or disable.

RETURN: None

PANIC: None

Enable or disable auto-switch-off if mains is present. If AL is non-zero, auto-switch-off is disabled, otherwise it is enabled.

By default auto-switch-off is enabled.

Even if enabled, the machine will not switch off when mains is absent if auto-switch-off has been stopped by calling GenSetAutoSwitchOffValue with value -1.

CHAPTER 20

DATABASE FILE MANAGEMENT

File structure

Database files (DBFs) start with a 22 byte header which contains the following information:

Offset in header	Information
0 - 15	Zero terminated file signature.
16, 17	Version of DBF software used to produce the file.
18, 19	Offset from the start of the file to the first record.
20, 21	Minimum version of DBF software required.

Note that all 16 bytes of the file signature are used for verification, not just the zero terminated string. Therefore it is recommended that all file signatures are padded with zeros to fill the 16 bytes.

See the `DbfVersion` service for the format of the version numbers.

The offset within the file of the first record is to allow additional information to be added to the header (called the `Extended Header`). Note that the first record will always be a type 2 record - see below.

The data consists of records, each with a 2 byte header stored as a word. The high nibble of the most significant byte (i.e. the 2nd byte in the record) gives the record's type. This can take the following values:

0	A deleted record.
1	A main record.
2	The <code>Field Information Record</code> .
3	The <code>Descriptive Record</code> .
4 - 7	Reserved for record types which won't be merged.
8 - 13	Reserved for record types which will be merged.
14	Reserved for voice entries.
15	Used internally (not to be used by applications).

Note that types 8 - 14 will be copied and merged by the `DbfCopyFile` service when `DbfRecordTypeAll` is specified whereas record types 3 - 7 will only be copied if the target file is **created**, i.e. not if merging two files.

The remaining 12 bits of the header word give the size of the record. However the maximum size of a record is 4094 bytes, so that there is room in a 4096 byte buffer for the longest record including its header.

The `Field Information Record` is used to store the field structure used by the other records. There will be exactly 1 `Field Information Record` per file and it will always be the first record in the file (any other type 2 records will be ignored). Each byte in the record indicates the type of the corresponding field in the records which follow, so the length of this record is the number of fields. The possible values for each byte are:

0	Word
1	Long

2	Double
3	String
4 - 255	Reserved

The maximum length of this record is 32 bytes representing 32 fields.

The 22 byte header and the `Field Information Record` must be passed to the `DbfOpen` service when a file is created or replaced and will be returned by `DbfOpen` when an existing file is opened.

Buffering

When a database file is opened, the address of a buffer must be provided which should be at least as large as the maximum record size to be used. Thus, a buffer of 4096 bytes is guaranteed to open all database files. This buffer will be used to read this many bytes worth of records from the File System at a time so as to reduce the calls to the File System and vastly increase the speed of operation of most of the DBF services.

If the buffer provided is smaller than 4096 and there are records which are longer than the buffer, an error will be given when the file is opened.

Note that the read services simply return the offset of the record within the buffer. If the buffer needs to be used by the application, e.g. for editing a record, the `DbfCopyDown` service should be called to copy the current record to the start of the buffer and to signal that the buffer is invalid. The `DbfTrash` service simply marks the buffer as invalid. These two services will therefore cause the entire buffer to be read in the next time a record is read, inevitably resulting in a loss of performance.

All DBF services may overwrite the buffer containing the current record apart from the following:

- `DbfFlush`
- `DbfVersion`
- `DbfAppend`
- `DbfSense`
- `DbfCount`

which are guaranteed not to alter the buffer.

Index Table

In addition to buffering, a sparse index table consisting of a 4 byte address for every 16 records will be constructed when the file is opened to increase the speed of random access to the file. As records are added to the file, the table will also be appended. When a record is deleted, each pointer in the table after the deleted record will be moved to the next record, so that they always point to every 16th record. Note that the index table will reside in a different segment so as not to use up the application's space.

End of file record

When any of the record services attempt to read past the end of the file, `EofErr` will be returned and the current record number will be the number of the last record plus 1. Also, attempting to read before the first record in the file, either with the `DbfBackRead` service or the `DbfFindRead` service searching backwards will result in `EofErr` and the current record number will be zero (i.e. the first record if there is one). However the offset in the buffer of the first/last record will not be returned, when `EofErr` is returned.

The "current record" is always given by the record number returned by the `DbfSense` service but if any service gives `EofErr`, the current record will be the last record number plus 1 - this is called the end of file record (unless the error is caused by going before the first record). When this is the case, any services which work on the current record, e.g. `DbfEraseRead`, `DbfUpdate`, `DbfFindRead` searching forwards will return `EofErr`. Similarly if there are no records in the file, `DbfSense` record will return 0 but the above services will return `EofErr`.

Number of records

The maximum number of records which can be present is 65534 and they are numbered from 0 to 65533. An error will be given by the DbfAppend service if an attempt is made to write more than 65534 records.

DbfOpen

Open a database file

CL	Type of record.
SI	Pointer to the main buffer.
DI	State.
DX	Length of the main buffer.
BX	Points to a DbfOpenEnt structure containing the remaining parameters.
RETURN: Carry clear	
DI	State.
RETURN: Carry set	
RecordErr	There are records longer than buffer supplied or buffer length is invalid.
InvalidFileErr	The file is not a valid DBF file.
PANIC:	
None	

This service works in the same way as the standard file open service with the following features:

- The service can be called in a loop with DI equal to DbfStateStart the first time and then passed as it is returned until it becomes DbfStateStart again or alternatively if DI is passed as DbfStateDisabled, the service will not return until it has finished.

Also DI can be passed as DbfStateOpenNoIndex to disable the building of the index. This means it will be faster to open, but only the following services can be used on a file opened this way: DbfClose, DbfFlush, DbfTrash, DbfCopyDown, DbfCopyFile, DbfAbsRead, DbfAbsReadSense, DbfNextRead, DbfBackRead, DbfFirstRead, DbfSense. Note that reading records non-sequentially will be much slower than when the file is opened with index building. Calls to any other services will produce unpredictable results.

- The DbfOpenMode field of the DbfOpenEnt structure need not specify the file format. The DBF file format will be assumed. If a file is created or replaced, ModeUpdate must be specified since the header is written to the file.
- The DbfOpenHeader field of the DbfOpenEnt structure is a pointer to a 56 byte buffer. The header consists of the 22 byte header described above followed immediately by the Field Information Record as a type 2 record. The maximum length of the Field Information Record is 32, plus its 2 byte header = 34. Therefore the buffer must be 22 + 34 = 56 bytes. Even if an Extended Header is required, no gap should be left in the header when creating a file and no gap will be returned when opening an existing file. The file itself, however will contain a gap for the Extended header, the length of which can be calculated from the 'start of data' field in the 22 byte header.

When opening an existing file, the header buffer must contain the file signature which will be verified against the signature in the file and InvalidFileErr will be returned if it is not identical. Note that all 16 bytes of the signature are always checked, not just the zero terminated string. The remainder of the header buffer will be filled in. Note that the type 2 record is not verified to be the same as the header and so need not be supplied.

When creating or replacing a file, the header buffer must contain all 56 bytes to be written to the file as a header.

In both the above cases, the minimum version number at offset 20 in the header will be checked and `InvalidFileErr` returned if this DBF software cannot handle it. See the `DbfVersion` service for the format of the version numbers. Note that only the major version number is checked (i.e. the most significant 4 bits only of the version number word. Also, the type 2 record is checked to be valid and `InvalidFileErr` returned if it is not.

- The main buffer at SI is used to read DX bytes worth of records at a time from the File System.
- CL specifies the type (0 - 14) of records to be accessed (it will usually be 1).
- A sparse index table consisting of a 4 byte address for every 16 records will be constructed when the file is opened to increase the speed of random access to the file. This will reside in a separate segment so as not to use any of the application's space.

Note that after opening the file, the current record number will be 0 (as returned by `DbfSense`) so that a call to `DbfNextRead` would read record number 1 in the file and `DbfEraseRead` would erase record 0. `DbfFirstRead` should be called to read record 0.

The length of buffer supplied must be in the range 512 to 16384. Any length outside this range will result in `RecordErr` when the file is opened. The maximum length of a record is 4094 bytes, so there is always room in a 4096 byte buffer for all records (including the 2 byte header).

DbfClose **Close a database file**

BX The DBF handle to be closed.

RETURN: Carry clear

 Success

RETURN: Carry set

 AL Error number.

PANIC:

 PanicDbf1 BX is not a valid DBF handle.

Closes a database file.

The handle must be one returned from the `DbfOpen` service.

DbfFlush **Flush a database file**

BX The DBF handle.

RETURN: Carry clear

 Success

RETURN: Carry set

 AL Error number.

PANIC:

 PanicDbf1 BX is not a valid DBF handle.

Flushes all buffers.

DbfTrash **Trash a database file**

BX The DBF handle.

RETURN: None

PANIC:

 PanicDbf1 BX is not a valid DBF handle.

Signals that the main database file buffer is no longer valid, so that it can be used by an application. See also the `DbfCopyDown` service.

DbfCopyDown**Copy down a DBF record**

BX The DBF handle.
 SI The offset into the main buffer of the record to copy down.

RETURN:

AX The length of the record copied down.

PANIC:

PanicDbf1 BX is not a valid DBF handle.
 PanicDbf2 SI is not a valid offset.

Copies a record at the given offset in the main buffer down to the start of the buffer and sets a flag to signal that the buffer is no longer valid (i.e. there is no need to call `DbfTrash`). The length of the record is read from the buffer and is returned by the service.

DbfCompress**Compress a database file**

BX The DBF handle.
 DI State.

RETURN: Carry clear

DI State.

RETURN: Carry set

AL Error number

PANIC:

PanicDbf1 BX is not a valid DBF handle.

Recovers space used by deleted records provided the file is stored on a compressible media. If the media is not compressible, this service will do nothing and will return carry clear.

After calling this service, the current record will be the end of file record (unless the media was not compressible - in which case the current record is unchanged).

DI can be passed as `DbfStateStart` or `DbfStateDisabled`. If it is passed as `DbfStateStart`, the service must be called repeatedly until state becomes `DbfStateStart` again. If it is passed as `DbfStateDisabled`, the service will not exit until it is finished.

DbfCopyFile**Copy a database file**

BX The DBF handle.
 CL Record type to copy.
 CH The direction of the copy (**to** or **from** the target file).
 DX Open mode for target file.
 DI State.
 SI Pointer to the target file name.

RETURN: Carry clear

DI State.

RETURN: Carry set

AL Error number.

PANIC:

PanicDbf1 BX is not a valid DBF handle.

Copies records except deleted records from or to the open source file. This service will append records to the end of an existing file if DX is `ModeOpen` or `ModeAppend`. Note that `ModeAppend` will perform exactly the same as `ModeOpen`. Copying with `ModeUnique` will copy the records to a unique file and return the name in the buffer at SI, in exactly the same way as `DbfOpen`. Note that the source file can be opened with index building disabled, i.e. with DI as `DbfStateOpenNoIndex` with no loss in performance of the copy.

CL is used to specify the type of record to be copied. If DbfRecordTypeAll is specified, all record types will be copied. Note that when merging files (i.e. DX is ModeOpen or ModeAppend) with DbfRecordTypeAll, only record types 1 and 8 - 14 will be copied across. Record types 2 - 7 will not be copied. Record type 2 is the Field Information Record and record type 3 is the Descriptive Record and types 4 - 7 are reserved for future use. Record types 2 - 7 will be copied if DX is ModeCreate, ModeReplace or ModeUnique.

CH must be passed as DbfCopyFromHandle to copy from the open file to the named file or DbfCopyToHandle to copy the other way. Note that in the latter case DX must obviously be passed as ModeOpen. Note that it is always slower using DbfCopyToHandle because of the need to update the index table for the open file.

The following procedure is used to implement the copy:

- The target file is opened in the mode specified. If copying to an existing file, the signatures of the two files are verified and the FIR's are checked to be compatible. If copying to a new file, the header (including the FIR) and any Extended Header are copied from the source file to the target file.
- All records of the specified type are copied from the source file to the target file.
- The target file is closed.
- If any error occurs during the above procedure and the target file has been created by DbfCopyFile, the target file will be deleted, if possible.

DI can be passed as DbfStateStart, DbfStateDisabled or DbfStateCopyAbort:

- If it is passed as DbfStateDisabled the service will not exit

until it is finished.

- If it is passed as DbfStateStart the service must be called repeatedly until state becomes DbfStateStart again. This parameter can be used to plot the progress of the copy, for example by drawing a bar graph. DI will be incremented for every 'buffer size' number of bytes that are copied (approximately). Hence the scale of the graph can be calculated by dividing the size of the file by the buffer size allocated. The graph plotting procedure must take account of the error in the number of times required to call the service. A fudge factor of 2 should be added to calculate how many times the service will be needed to be called.
- DI can be passed as DbfStateCopyAbort to abort the copy which was started with DbfStateStart.

Warning: using DbfCopyFile to merge files can result in a file with more than 65534 records of a particular type in it. When this file is opened with the DbfOpen service, only the first 65534 records will be accessible. No error is given from the copy or the open.

DbfFileSize **Get the size of a DBF**

BX The DBF handle.

RETURN: Carry clear

DI:DX The file size in bytes.

RETURN: Carry set

AL Error number.

PANIC:

PanicDbf1 BX is not a valid DBF handle.

Gets the size of an open database file.

DbfExtHeaderRead Read a DBF extended header

AL	0 to start 1 to continue.
BX	The DBF handle.
CX	The number of bytes to read.
SI	The address of the buffer to receive the data.
RETURN:	Carry clear
AX	The number of bytes actually read.
RETURN:	Carry set
EofErr	The end of the Extended Header has been reached.
PANIC:	
PanicDbf1	BX is not a valid DBF handle.

Reads CX bytes from the Extended Header of a database file into the buffer supplied.

AL must be passed as 0 the first time and 1 to continue.

EofErr is returned when the end of the Extended Header is reached, otherwise the actual number of bytes read is returned.

DbfExtHeaderWrite Write a DBF extended header

AL	0 to start 1 to continue.
BX	The DBF handle.
CX	The number of bytes to write.
SI	The address of the buffer containing the data to write.
RETURN:	Carry clear
AX	The number of bytes actually written.
RETURN:	Carry set
EofErr	A write past the end of the Extended Header was attempted.
PANIC:	
PanicDbf1	BX is not a valid DBF handle.

Writes the buffer supplied into the Extended Header of the file.

AL must be passed as 0 the first time and 1 to continue.

EofErr is returned when the end of the space allocated for the Extended Header is reached, otherwise the actual number of bytes written is returned.

DbfDescRecordRead Read a DBF descriptive record

BX	The DBF handle.
RETURN:	Carry clear
AX	The length of the Descriptive Record.
RETURN:	Carry set
EofErr	There is no Descriptive Record in the file.
PANIC:	
PanicDbf1	BX is not a valid DBF handle.

Reads the Descriptive Record into the main buffer at offset zero. If there is no Descriptive Record, EofErr will be returned.

DbfDescRecordWrite **Write a DBF descriptive record**

BX The DBF handle.
CX The length of the Descriptive Record to be written.

RETURN: Carry clear
 Success

RETURN: Carry set
 AL Error number.

PANIC:
 PanicDbf1 BX is not a valid DBF handle.

Writes out a Descriptive Record. The record to be written must be stored at the start of the main buffer as a DbfRecord structure; there must be 2 bytes before the data starts where the record header will be constructed. See DbfAppend.

Any existing Descriptive Record will be erased, in other words, there can be a maximum of one Descriptive Record per file.

If CX is passed as zero, any existing Descriptive Record will be erased and no new one will be written out. If there is no Descriptive Record, no error is given.

DbfVersion **Get the DBF version number**

None

RETURN:
 AX The DBF version number.

PANIC: None

Gets the version number of the DBF software. This will be in the form:

XYF

where

X is the major version number (4 bits)
YY is the minor version number (8 bits)
F is the release type, either A,B or F for Alpha, Beta or Final respectively (4 bits).

For example, if 110FH is returned, the DBF software version is 1.10F.

Note that only the major version number is used to determine whether or not the DBF file system can handle a particular file.

DbfAbsRead **Read an absolute DBF record**

BX The DBF handle.
CX The absolute record number to be read.
RETURN: Carry clear
 AX The length of the record read.
 SI The offset in the main buffer of the record read.

RETURN: Carry set
 EofErr The requested record number is greater than the number of records in the file.

PANIC:
 PanicDbf1 BX is not a valid DBF handle.

This service will seek to the given record and read it. Records are read into the buffer (supplied when the file was opened) and the record's offset within the buffer is returned in SI.

If the record number requested corresponds to a record beyond the last one, the error EofErr will be returned, the current record will be the end of file record and SI will be invalid.

DbfAbsReadSense Read and sense an absolute DBF record

BX The DBF handle.
 CX The absolute record number to be read.
RETURN: Carry clear
 AX The length of the record read.
 SI The offset in the main buffer of the record read.
 DI:DX File position of start of record.
RETURN: Carry set
 EofErr The requested record number is greater than the number of records in the file.
PANIC:
 PanicDbf1 BX is not a valid DBF handle.
 Same as the DbfAbsRead service but also returns the file position of the start of the record.

DbfNextRead Read the next DBF record

BX The DBF handle.
RETURN: Carry clear
 AX The length of the record read.
 SI The offset in the main buffer of the record read.
RETURN: Carry set
 EofErr The current record was already the last record in the file.
PANIC:
 PanicDbf1 BX is not a valid DBF handle.
 Seeks to the next record and reads it. Records are read into the buffer (supplied when the file was opened) and the record's offset within the buffer is returned in SI.
 If the current record is already the last record in the file or there are no records of the current type in the file, EofErr will be returned, the current record will be the end of file record and SI will be invalid.

DbfBackRead Read the previous DBF record

BX The DBF handle.
RETURN: Carry clear
 AX The length of the record read.
 SI The offset in the main buffer of the record read.
RETURN: Carry set
 EofErr The current record was already the first record in the file.
PANIC:
 PanicDbf1 BX is not a valid DBF handle.
 Seeks to the previous record and reads it. Records are read into the buffer provided when the file was opened and the offset into this buffer of the record required is returned in SI.
 If, on entry to the call, the current record is already the first record in the file or there are no records of the current type in the file, EofErr will be returned, the current record will be 0 and SI will be invalid.

DbfFirstRead **Read the first DBF record**

BX The DBF handle.

RETURN: Carry clear

AX The length of the record read.

SI The offset in the main buffer of the record read.

RETURN: Carry set

EofErr There are no records in the file.

PANIC:

PanicDbf1 BX is not a valid DBF handle.

Seeks to the first record and reads it. Records are read into the buffer (supplied when the file was opened) and the record's offset within the buffer is returned in SI.

If there are no records in the file of the current type, EofErr will be returned, and SI will be invalid.

DbfLastRead **Read the last DBF record**

BX The DBF handle.

RETURN: Carry clear

AX The length of the record read.

SI The offset in the main buffer of the record read.

RETURN: Carry set

EofErr There are no records in the file.

PANIC:

PanicDbf1 BX is not a valid DBF handle.

Seeks to the last record and reads it. Records are read into the buffer (supplied when the file was opened) and the record's offset within the buffer is returned in SI.

If there are no records in the file of the current type, EofErr will be returned, and SI will be invalid.

DbfAppend **Append a DBF record**

BX The DBF handle.

CX The length of the record to be appended.

RETURN: Carry clear

Success

RETURN: Carry set

OverflowErr There are already 65534 records in the file.

RecordErr The total length of the record (including the 2 byte header) is greater than the length of the main buffer.

PANIC:

PanicDbf1 BX is not a valid DBF handle.

This service appends a record of the current type to the end of the file and makes this the current record.

The record to be written must be placed at the start of the main buffer as a DbfRecord structure which is defined as:

```
typedef struct
{
    UWORD header; /* Used for record header word */
    UBYTE data[2]; /* Data to be written... */
} DbfRecord;
```

The header word will be used to construct the header for the record so that it can be written in one.

CX is the length of the data only.

DbfEraseRead**Erase a DBF record**

BX The DBF handle.

DI State.

RETURN: Carry clear

AX The length of the record read.

SI The offset in the main buffer of the record read.

DI State.

RETURN: Carry set

EofErr The current record is the end of file record.

PANIC:

PanicDbf1 BX is not a valid DBF handle.

Erases the current record and reads the next one.

A record is erased by overwriting its (4 bit) type to 0. The space used by the record can only be recovered by calling DbfCompress. The file *must* be stored on a compressible medium.

If there are no records of the current type in the file or if the current record number is the last record plus 1, EofErr will be returned. If the current record is the last record in the file, it will be erased and EofErr will be returned.

Note that DbfEraseRead may return EofErr in 2 different circumstances:

- If the current record is already the end of file record (or there are no records).
- If the current record is the last record.

In the first case, the service does nothing. In the second case, the last record is erased and the current record becomes the end of file record. It is up to the application to deduce which of these cases has occurred by checking whether the current record is the end of file record before calling the service (using DbfSense and DbfCount) or by noting the decrease in the total number of records from DbfCount.

DI can be passed as DbfStateStart or DbfStateDisabled. If it is passed as DbfStateStart, the service must be called repeatedly until state becomes DbfStateStart again. If it is passed as DbfStateDisabled, the service will not exit until it is finished.

DbfUpdate**Update a DBF record**

BX The DBF handle.

CX The length of the record to be written.

DI State.

RETURN: Carry clear

DI State.

RETURN: Carry set

EofErr The current record is the end of file record.

PANIC:

PanicDbf1 BX is not a valid DBF handle.

Erases the current record and appends the new one to the end of the file making this the new current record. The new record to be appended will be taken from the beginning of the main buffer.

The main buffer must begin with a word where the record header will be built, followed by the record itself.

CX is the length of the data only and does not include the word at the start.

Note that the current record will only be erased after the supplied record has been successfully appended.

If there are no records of the current type in the file or if the current record number is the last record plus 1, EofErr will be returned. If the current record is the last record in the file, it will be erased and EofErr will be returned.

DI can be passed as `DbfStateStart` or `DbfStateDisabled`. If it is passed as `DbfStateStart`, the service must be called repeatedly until state becomes `DbfStateStart` again. If it is passed as `DbfStateDisabled`, the service will not exit until it is finished.

DbfFindRead Find a DBF record

AL	The number of fields to search.
BX	The DBF handle.
CX	The length of the buffer to match.
DL	The maximum field length to match with.
DH	The type and direction of the search.
DI	State.
SI	Pointer to the match buffer.

RETURN: Carry clear

AX	The length of the record read.
DI	State.
SI	The offset in the main buffer of the record read.

RETURN: Carry set

EofErr	No matching record was found.
--------	-------------------------------

PANIC:

PanicDbf1	BX is not a valid DBF handle.
PanicDbf2	Parameters are invalid.

Matches the given wild-card string with the string components of each record starting at the current record. If the current record is the end of file record, `EofErr` will be returned, unless `DH` specifies `DbfFindBackwards`.

If a match is found, the record containing the match is made the current record and is read into the buffer; the offset will be returned in `SI`.

`AL` specifies how many string fields are to be searched (the number of string fields in the `Field Information Record` is now irrelevant). A value of `DbfFindAllStrings` must be used to specify continue matching string fields until the end of the record is reached. The `Field Information Record` is used to specify the 'types' of the first 32 fields. After that, all fields are assumed to be strings until the end of the record.

`DL` specifies the maximum length of a string field to be used in the match, i.e. longer strings are truncated for matching purposes. 255 specifies no truncation.

`DH` is split into 2 halves.

The most significant nibble of `DH` specifies the type of match and must be one of the following:

<code>DbfFindCaseIndependent</code>	Case independent match.
<code>DbfFindCaseDependent</code>	Case dependent match.

The least significant nibble of `DH` specifies the search direction and must be one of the following:

<code>DbfFindForwards</code>	Search forwards from current record to next match.
<code>DbfFindBackwards</code>	Search backwards from current record to previous match.
<code>DbfFindFirst</code>	Search to first match in file.
<code>DbfFindLast</code>	Search to last match in file.

If a match is not found, `EofErr` will be returned and `SI` will be invalid. The current record will then be the first record if the search was backwards or the last record number plus one (the end of file record) if the search was forwards.

The `Field Information Record` contains the record structure which is used to find the string components of the record. This is always passed to `DbfOpen` as part of the header when a file is created and is returned when an existing file is opened.

DI can be passed as `DbfStateStart` or `DbfStateDisabled`. If it is passed as `DbfStateStart`, the service must be called repeatedly until state becomes `DbfStateStart` again. If it is passed as `DbfStateDisabled`, the service will not exit until it is finished.

DbfSense **Sense the current DBF record number**

BX The DBF handle.

RETURN:

AX The current record number.

PANIC: None

Sense the current record number. This will be the last record number plus 1 if an `EofErr` has just been given (unless there are no records, in which case it will be 0).

DbfCount **Count the number of DBF records**

BX The DBF handle.

RETURN:

AX The number of records of the current type.

PANIC: None

Returns the number of records in the file of the current type. It will not alter the current record number.

③DbfFindReadField **Find a DBF record by field**

AL The number of fields to search.

BX The DBF handle.

CX The length of the buffer to match.

DL The maximum field length to match with.

DH The type and direction of the search.

DI State.

SI Pointer to the match buffer.

`DateClassPtr` The starting field from which to search (0 for first field)

RETURN: Carry clear

AX The length of the record read.

DI State.

SI The offset in the main buffer of the record read.

RETURN: Carry set

`EofErr` No matching record was found.

PANIC:

`PanicDbf1` BX is not a valid DBF handle.

`PanicDbf2` Parameters are invalid.

Matches the given wild-card string with the string components of the specified fields in each record starting at the current record. If the current record is the end of file record, `EofErr` will be returned unless DH specifies `DbfFindBackwards`.

This service is the same as `DbfFindRead` except that `DateClassPtr` specifies the starting field from which the search starts (with 0 specifying the first field). For example, to search only the third, fourth and fifth text fields, set `DateClassPtr` to 2 and AL to 3.

CHAPTER 21

HARDWARE MANAGEMENT

HwComboOn

Switch on the combo

None

RETURN: None

PANIC: None

This service will switch on the combo hardware subsystem (CHS) if not already switched on.

Although the CHS has been enabled, it can be accessed either by an external expansion device or by Asic1. If it is desired to access the CHS using Asic1 then the SLDTX bit in the Asic1 Control register needs to be enabled as well. Before turning the CHS on, it is important to see if it is available for use by calling the HwGetCombo service.

This service is equivalent to HwComboOnInput for all variants except Asic9 variants (Series 3a). On Asic9 variants HwComboOn puts the codec into output mode, while HwComboOnInput puts it into input mode.

HwComboOff

Switch off the combo

None

RETURN: None

PANIC: None

This service will switch off the combo hardware subsystem (CHS) if not already switched off.

HwPacksOn

Switch on the SSDs

None

RETURN: None

PANIC: None

This service will switch on the SSD subsystem (SSDS) if not already switched on.

This service is provided for the built in SSD drivers and should not be called by any other drivers or applications.

HwPacksOff

Switch off the SSDs

None

RETURN: None

PANIC: None

This service will switch off the SSD subsystem (SSDS) if not already switched off.

This service is provided for the built in SSD drivers and should not be called by any other drivers or applications.

HwSetA2Control1Bits**Set bits Asic2 register 1**

AL Mask of bits to be set.

RETURN: None

PANIC: None

This service can be used to set bits in Asic2 control register 1.

This service is necessary because the operating system cannot read directly from Asic2 and must keep its own up-to-date copy of the register contents.

Each bit which is set in the mask will cause the corresponding bit in the control register to be set.

HwClearA2Control1Bits**Clear bits Asic2 register 1**

AL Mask of bits to be cleared.

RETURN: None

PANIC: None

This service can be used to clear bits in Asic2 control register 1.

This service is necessary because the operating system cannot read directly from Asic2 and must keep its own up-to-date copy of the register contents.

Each bit which is set in the mask will cause the corresponding bit in the control register to be cleared.

HwReadA2Control1**Read Asic2 register 1**

None

RETURN:

AL The value currently in control register 1.

PANIC: None

This service can be used to read Asic2 control register 1.

This service is necessary because the operating system cannot read directly from Asic2 and must keep its own up-to-date copy of the register contents.

In effect, it returns the value in its up-to-date copy.

HwWriteA2Control1**Write Asic2 register 1**

AL The new value to be written to control register 1.

RETURN: None

PANIC: None

This service can be used to write to Asic2 control register 1.

This service is necessary because the operating system cannot read directly from Asic2 and must keep its own up-to-date copy of the register contents.

In effect, it updates the value in its up-to-date copy and then writes to the Asic2 register.

HwSetA2Control2Bits**Set bits Asic2 register 2**

AL Mask of bits to be set.

RETURN: None

PANIC: None

This service can be used to set bits in Asic2 control register 2.

This service is necessary because the operating system cannot read directly from Asic2 and must keep its own up-to-date copy of the register contents.

Each bit set in the mask will cause the corresponding bit in the control register to be set.

HwClearA2Control2Bits**Clear bits Asic2 register 2**

AL Mask of bits to be cleared.

RETURN: None

PANIC: None

This service can be used to clear bits in Asic2 control register 2.

This service is necessary because the operating system cannot read directly from Asic2 and must keep its own up-to-date copy of the register contents.

Each bit set in the mask will cause the corresponding bit in the control register to be cleared.

HwReadA2Control2**Read Asic2 register 2**

None

RETURN:

AL The value currently in control register 2.

PANIC: None

This service can be used to read Asic2 control register 2.

This service is necessary because the operating system cannot read directly from Asic2 and must keep its own up-to-date copy of the register contents.

In effect, it returns the value in its up-to-date copy.

HwWriteA2Control2**Write Asic2 register 2**

AL The new value to be written to control register 2.

RETURN: None

PANIC: None

This service can be used to write to Asic2 control register 2.

This service is necessary because the operating system cannot read directly from Asic2 and must keep its own up-to-date copy of the register contents.

In effect, it updates the value in its up-to-date copy and then writes to the Asic2 register.

HwSetA2Control3Bits**Set bits Asic2 register 3**

AL Mask of bits to be set.

RETURN: None

PANIC: None

This service can be used to set bits in Asic2 control register 3.

This service is necessary because the operating system cannot read directly from Asic2 and must keep its own up-to-date copy of the register contents.

Each bit set in the mask will cause the corresponding bit in the control register to be set.

HwClearA2Control3Bits**Clear bits Asic2 register 3**

AL Mask of bits to be cleared.

RETURN: None

PANIC: None

This service can be used to clear bits in Asic2 control register 3.

This service is necessary because the operating system cannot read directly from Asic2 and must keep its own up-to-date copy of the register contents.

Each bit set in the mask will cause the corresponding bit in the control register to be cleared.

HwReadA2Control3**Read Asic2 register 3**

None

RETURN:

AL

The value currently in control register 3.

PANIC: None

This service can be used to read Asic2 control register 3.

This service is necessary because the operating system cannot read directly from Asic2 and must keep its own up-to-date copy of the register contents.

In effect, it returns the value in its up-to-date copy.

HwWriteA2Control3**Write Asic2 register 3**

AL

The new value to be written to control register 3.

RETURN: None**PANIC:** None

This service can be used to write to Asic2 control register 3.

This service is necessary because the operating system cannot read directly from Asic2 and must keep its own up-to-date copy of the register contents.

In effect, it updates the value in its up-to-date copy and then writes to the Asic2 register.

HwSelectChannel**Select a serial channel**

AL

The new channel to be selected.

RETURN:

AL

The channel that was previously selected.

PANIC: None

This service will select which channel the serial controller in Asic2 will be connected to for subsequent serial data transfers.

This service is necessary because the operating system cannot read directly from Asic2 and must keep its own up-to-date copy of the register contents.

Interrupt service routines which use the serial controller must re-select the channel that was previously selected. This can be achieved by saving the value returned in AL when this service is called, as it is the currently selected channel.

HwNullFrame**Send a serial null frame**

None

RETURN: None**PANIC:** None

This service is useful for sending a null frame to a serial channel. This is important in order to guarantee that the controller and the slave device attached to the channel are synchronised.

HwSwitchOff**Switch off**

CX

The number of quarter seconds to switch off for.

RETURN: None**PANIC:** None

This service may be called to switch off the machine. In fact, the machine is never truly switched off and can wake up again in order to service an event in the future. The value in CX determines how many quarters of a second must pass before the machine will wake up again. If the value in CX is less than or equal to 8 then this service will do nothing.

If an absolute timer is pending or a process is sleeping until an absolute time then the value in CX will be adjusted to make sure that the machine wakes up in time to service the outstanding timer or to wake up the process.

If the value in CX is 0xFFFF then the machine will just switch off until an outstanding absolute time event is ready to expire or until the user switches on the machine.

The IBM PC version of EPOC does not support this service; instead, the `HwExit` service can be called which will return to DOS.

HwExit

Exit to DOS

None

RETURN: None

PANIC: None

This service is only available on the IBM PC version of Epoc/Os and will exit from the operating system and return to DOS.

HwGetCombo

Capture the combo subsystem

None

RETURN: Carry clear

Success

RETURN: Carry set

InUseErr

The combo subsystem is already captured.

PANIC: None

This service acts as a gate to the combo subsystem so that two device drivers do not both try to access the combo subsystem at the same time.

After capturing the combo subsystem, it must be released by calling the `HwFreeCombo` service when no longer required

HwFreeCombo

Free the combo subsystem

None

RETURN: None

PANIC: None

This service will free the combo subsystem after it has been captured with the `HwGetCombo` service.

HwGetChannel

Get a channel

AL

The mask of the channels being captured.

RETURN: Carry clear

Success.

RETURN: Carry set

InUseErr

The channel is already captured.

PANIC: None

This service provides a gate to control access to the hardware interrupt service routines.

It is also a handy way of ensuring that two device drivers do not start talking to the same expansion port at the same time, by getting the channel which is associated with that expansion port.

The strategy is to request the channel before trying to talk to the hardware. If the channel is allocated successfully, then all is well and the driver can then talk to the expansion port. Whenever a driver has captured a channel in this way it must free the channel when it is no longer required by calling the `HwFreeChannel` service.

HwFreeChannel**Free a channel**

AL The mask of the channels being freed.

RETURN: None

PANIC: None

This service will free a channel after it has been captured with the HwGetChannel service.

HwGetPsuType**Get the power supply type**

None

RETURN:

AL The power supply type.

PANIC: None

There are two power supply variants in the MC range of computers which use the EPOC operating system. Consequently there are two version of the operating system due to the different power supply handling code. Apart from this service, EPOC hides the differences between the two power supplies. The REPRO software which will load a new operating system into the FLASH memory uses this service to know which version of EPOC to load.

HwGetSupplyStatus**Get supplies status**

SS:BX Pointer to a SupplyEnt structure.

RETURN: None

PANIC: None

This service may be used to get the current status of the various supplies.

The value returned for the main battery and lithium batteries are in millivolts. The MainsPresent field can be:

< 0	mains status cannot be determined at the current time (if the SSD doors are open)
0	mains is <i>not</i> present
1	mains is present

HwSupplyWarnings**Get supplies warnings**

SS:BX Pointer to a SupplyWarningsEnt structure.

RETURN: None

PANIC: None

This service may be used to ask the operating system what the maximum value of the main and lithium battery reading can be and what an appropriate warning level would be. The values in the structures are in the same units as for the HwGetSupplyStatus, i.e. millivolts.

This service will return different values depending on the battery type set with the GenSetBatteryType service. If no battery type is set then the values for an alkaline battery will be returned.

HwLcdContrastDelta**Change the LCD contrast**

AL +ve to step contrast up.
 -ve to step contrast down.

RETURN: None

PANIC: None

This service can be used to step the LCD contrast up or down depending on whether AL is positive or negative.

HwReadLcdContrast**Get current LCD contrast**

None

RETURN:

AL

The current contrast value.

PANIC: None

This service can be used to get the current contrast setting.

HwSetBackLight**Set backlight control**

BX

The new backlight control value.

RETURN: None**PANIC:** None

This service can be used to set the backlight control.

The value in BX contains two values. The bottom 15 bits are a time-out in ticks (1/32nd of a second) to switch off the backlight. If this value is zero then the backlight is not switched off automatically.

The top bit (i.e. the sign bit), is used to enable/disable the operating system from toggling the backlight state on reception of the backlight key. Setting the bit will disable the operating system.

HwGetBackLight**Get backlight control**

None

RETURN:

AX

The backlight control value.

PANIC: None

This service can be used to get the current backlight control value.

HwBackLight**Operate the backlight**

AL

0 - Switch off the backlight.

1 - Switch on the backlight.

2 - Toggle the backlight.

3 - Return the backlight state.

RETURN: Carry clear

AL

The previous or current backlight state:

0 - backlight is/was off.

1 - backlight is/was on.

RETURN: Carry set

NotSupportedErr

Machine does not support a backlight.

PANIC: None

This service can be used to perform the following functions:

- Switch the backlight on and off.
- Toggle the backlight state.
- Query the current backlight state.

A backlit version of the machine can be determined by checking for the NotSupportErr being returned with AL = 3 to query the backlight state.

③ HwGetScanCodes**Scan the state of all keys**

BX

Pointer to 10 word array to take the scan codes.

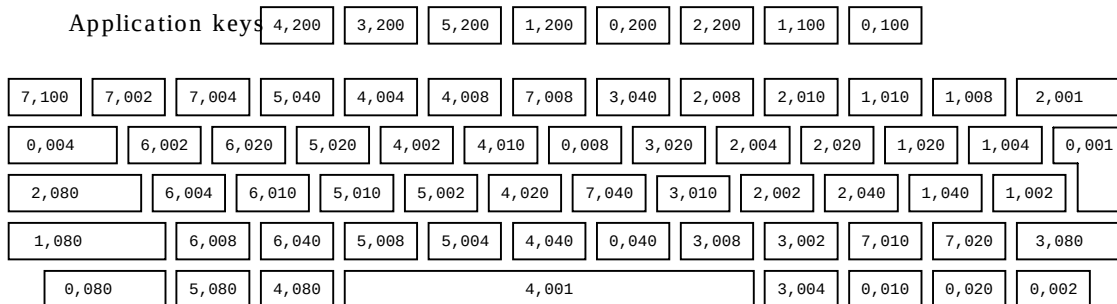
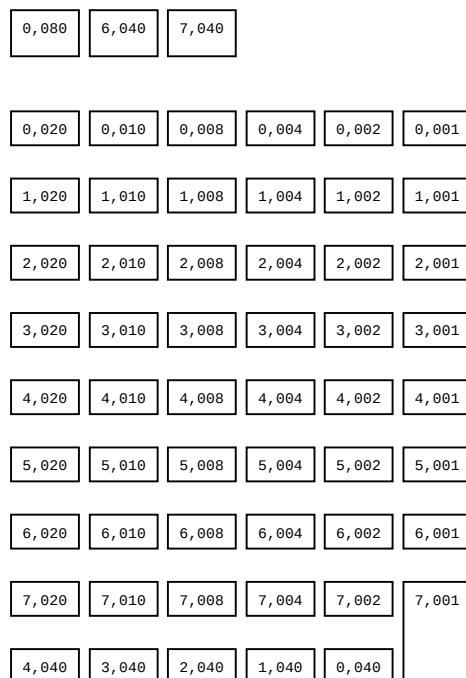
RETURN: Nothing.**PANIC:** None

Writes values to the array at BX corresponding to the state of each key on the keyboard and to each application button. A unique bit is set for each key being pressed when this service is called. If the key is up then no bit is set.

On the Series 3a, eleven bits are valid in each of the first eight words and the *Workabout* uses nine bits in each of the first eight words. On HC machines, eight bits in ten words are valid. This service is not available on the MC400, MC200 and Series 3.

The set of scan codes is different for each machine's keyboard layout, but is fully determined by the position of the key on each type of machine.

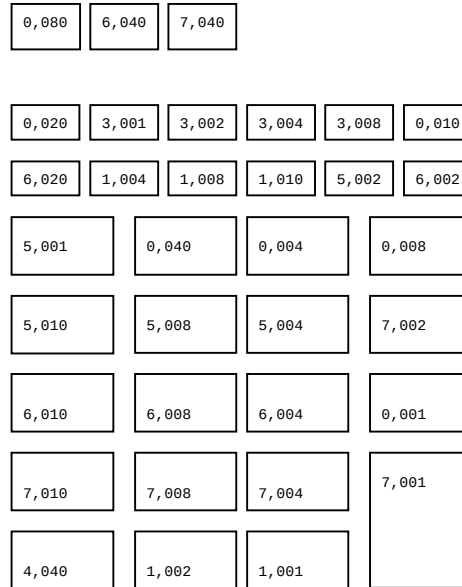
The following diagrams specify the scan code associated with each key on the different machines which support this service. Each box represents a key. The first number in each box gives the element of the array at BX used for that key (first element 0), and the second number gives the **hexadecimal** mask which, when ANDed with that array element, gives a non-zero result if that key is down. For example, on a Series 3a if the Control key is being pressed, element 2 of the array ANDed with hex 80 is non-zero.

Series 3a keyboard**HC alphabetic keyboard**

Note that the scan code (0,080) given for the On/Off key is that for Off. The scan codes for On (8,080 - not shown in the above diagram) and Off (0,080) are not normally received by application code.

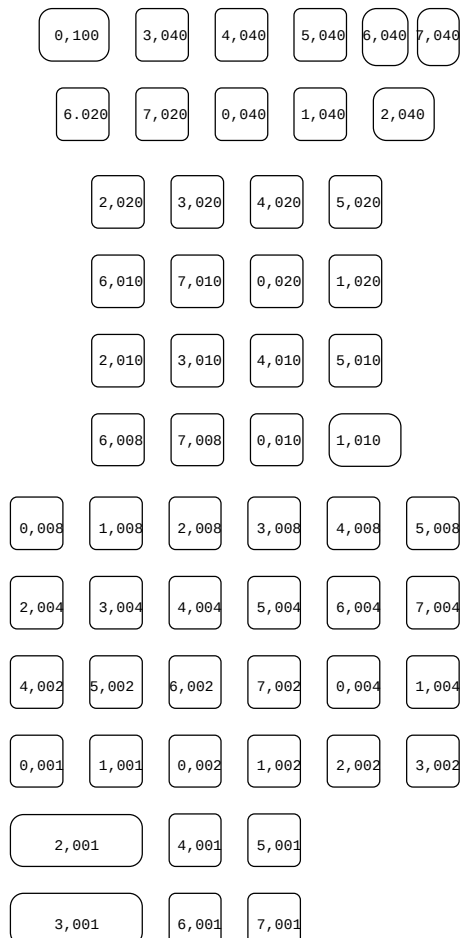
The Off scan code is received if the application captures the Off key (capture of this key by the HC Command Shell must first be disabled - see the *Command Shell* chapter of the *HC Programming Guide*)

HC numeric keyboard



Note that the scan code given for the On/Off key (0,080) is that for Off. The scan codes for On (8,080 - not shown in the above diagram) and Off (0,080) are not normally received by application code. The Off scan code is received if the application captures the Off key (capture of this key by the HC Command Shell must first be disabled - see the *Command Shell* chapter of the *HC Programming Guide*).

Workabout keyboard



Note that the scan code given for the On/Esc key (0, 100) is the Escape scan code. The scan codes for On (0, 080 - not shown in the above diagram) and Off (6, 020) are not normally received by application code. The Off scan code is received if the application captures the Off key.

③HwComboOnInput **Switch on the combo in input mode**

None

RETURN: None

PANIC: None

This service is equivalent to HwComboOn for all variants except Asic9 variants (Series 3a). On Asic9 variants HwComboOn turns on the codec and puts it into output mode. HwComboOnInput also turns on the codec but puts it into input mode.

③HwSupplyInfo **Get additional power supply data**

BX Pointer to SupplyInfoEnt structure

RETURN: Nothing.

PANIC: None.

Write information concerning the various power supplies to the SupplyInfoEnt structure at BX. This information can be used to monitor battery and mains usage. Only Asic9 variants (Series 3a) return meaningful data.

The SupplyInfoEnt structure is defined in *epocsibo.inc*.

Hardware Management update

The majority of the additional EPOC hardware management system services described in this section were introduced for the Series 3c and Siena.

With the exception of the HC, all the services are, in principle, available on any machine that contains EPOC version 3.90F or later. On an HC with a suitable version of EPOC, all the functions described in this section should generate an E_GEN_NSUP error.

Some services require the presence of hardware that is not built into all machines in the SIBO range. If the relevant hardware is not present on a particular machine, calling the service will either have no effect or return an error of E_GEN_NSUP. The descriptions of such services contain a list of the machines on which they are intended to be used.

HwResetBatteryStatus **Reset the battery status**

None

RETURN: None

PANIC: None

Reset the battery status information. This service has the same effect as replacing the main batteries.

HwEnableAutoBatReset **Enable/disable battery info reset**

BX Enable/disable/query the status

RETURN:

AL The current auto battery reset status, if queried, otherwise none.

PANIC: None

This service is primarily intended for use on the Workabout.

Enable or disable an auto reset of the battery information when the battery is recharged in-place.

The value of BX should be 1 to enable, 0 to disable, or -1 to query the auto reset status.

If querying the status, a value of 1 or 0 is returned in AX, respectively meaning that auto reset is enabled or disabled.

HwGetBatData**Return battery information**

None

RETURN:

AX

Pointer to battery information.

PANIC: None

Return the address of the SupplyInfoEnt battery information structure in the OS data segment.

The structure is defined as:

```

SupplyInfoEnt struc
SuMainBatLevel      db      ?
SuMainBatStatus     db      ?
SuBackupBatLevel    db      ?
SuDcLevel           db      ?
SuWarningFlags      dw      ?
SuInsertionDate     dd      ?
SuTicksInUseBattery dd      ?
SuTicksInUseDc      dd      ?
SuMilliampTicks     dd      ?
SupplyInfoEnt ends

```

This structure is equivalent to the PLIB E_SUPPLY_INFO struct.

HwReLogPacks**Relog the SSDs**

None

RETURN: Carry clear

Success

RETURN: Carry set

AL

Error number

PANIC: None

Relog the packs. This service has the same effect as opening and then closing the pack doors on a Series 3a.

This service is supplied for internal use and is not intended to be called by application code.

HwSetIRPowerLevel**Set the IR power level**

BX

Required power level

RETURN:

AX

The previous IR power level

PANIC: None

This service is only available on Series 3c and Siena machines.

Set the power level used to drive the IR device to be high or low.

BX should be passed as 1 to set the high power level, or 0 to set the low power level.

Return a value (0 for low and 1 for high) representing the IR power level as it was before the service was called.

HwReturnTickCount**Sense the current tick count**

None

RETURN:

AX

Tick count.

PANIC: None

Return, in AX, a value that is incremented on every tick (32 times per second).

HwReturnExpansionPortState Sense the expansion port state

None

RETURN:

AX Expansion port state.
BX At present, always zero.

PANIC: None

Return the type and current state of the expansion port:

The value of AL is non-zero if the pack doors are open. Additionally, on Series 3c machines, it is non zero for a short period after something is plugged into, or removed from, the Honda connector.

AH contains one of the following values in its lower three bits:

0x00 Expansion port is Series 3/ Series 3a 6-pin
0x01 Expansion port is Workabout LIF
0x02 Expansion port is Siena Honda
0x03 Expansion port is Series 3c Honda
0x04 Expansion port is HC

In addition, the following value may be Ored into AH:

0x80 The machine contains the Condor chip

HwExpansionOn Enable power to Honda connector

None

RETURN: None**PANIC:** None*This service is only available on Series 3c machines.*

Enable the supply of power to a peripheral device connected to the machine via the Honda connector.

This service is provided for the built-in SSD drivers and should not be called by any other drivers or applications.

HwExpansionOff Disable power to Honda connector

None

RETURN: None**PANIC:** None*This service is only available on Series 3c machines.*

Disable the supply of power to a peripheral device connected to the machine via the Honda connector.

This service is provided for the built-in SSD drivers and should not be called by any other drivers or applications.

APPENDIX A

INTERRUPT AND FUNCTION NUMBERS

Introduction

Epoc system services are invoked using the INT nn 8086 instruction. There are two types of system services.

- Single - which just do one function.
- Multi - which do more than one function.

The multi service functions also require the AH register to be loaded with a value which selects the actual function to be performed.

The following section lists the actual numbers associated with the system services and their function numbers. In the listings, names starting with Nm are function numbers and should be placed in AH. All names starting with Nm are made up of Nm followed by a number of name components. The first component after Nm is the name of the interrupt to invoke. For example:

NmFilOpen, where Fil is the first name component uses FilManager.

NmHeapFreeCell, where Heap is the first name component uses HeapManager. NmDbfClose, where Dbf is the first name component uses DbfManager.

```
MOV AH, NmFilOpen
INT FilManager
```

All names not starting with Nm are the names of the single and multi level interrupts.

In the documentation, functions are referred to by the name with the leading Nm missing. Thus SegOpen can be called as follows:

```
MOV AH, NmSegOpen
INT SegManager
```

Single service interrupts are just referred to by their names. Thus StringLength is called as follows:

```
INT StringLength
```

As usual there are a few exceptions to this rule:

- NmLongUnsignedIntRandom is under INT GenManager. NmIoOpen is under INT DevManager.

Alphabetical list of functions

CONVMANAGER	EQU	008AH
NMCONVARGUMENTSTOBUFFER	EQU	0004H
NMCONVFLOATTOBUFFER	EQU	0009H
NMCONVINTTOBUFFER	EQU	0002H
NMCONVLONGINTTOBUFFER	EQU	0003H
NMCONVSTRINGTOFLOAT	EQU	000AH
NMCONVSTRINGTOINT	EQU	0007H
NMCONVSTRINGTOLONGINT	EQU	0008H
NMCONVSTRINGTOUNSIGNEDINT	EQU	0005H
NMCONVSTRINGTOUNSIGNEDLONGINT	EQU	0006H
NMCONVUNSIGNEDINTTOBUFFER	EQU	0000H
NMCONVUNSIGNEDLONGINTTOBUFFER	EQU	0001H
DBFMANAGER	EQU	00D8H
NMDBFABSREAD	EQU	000CH
NMDBFABSREADSENSE	EQU	000BH
NMDBFAPPEND	EQU	0011H
NMDBFBACKREAD	EQU	000EH
NMDBFCLOSE	EQU	0001H
NMDBFCOMPRESS	EQU	0005H
NMDBFCOPYDOWN	EQU	0004H
NMDBFCOPYFILE	EQU	0006H
NMDBFCOUNT	EQU	0016H
NMDBFDESCRECORDREAD	EQU	0017H
NMDBFDESCRECORDWRITE	EQU	0018H
NMDBFERASEREA	EQU	0012H
NMDBFEXTHEADERREAD	EQU	0008H
NMDBFEXTHEADERWRITE	EQU	0009H
NMDBFFILESIZE	EQU	0007H
NMDBFFINDREAD	EQU	0014H
NMDBFFINDREADFIELD	EQU	0019H
NMDBFFIRSTREAD	EQU	000FH
NMDBFFLUSH	EQU	0002H
NMDBFLASTREAD	EQU	0010H

NMDBFNEXTREAD	EQU	000DH
NMDBFOPEN	EQU	0000H
NMDBFSENSE	EQU	0015H
NMDBFTRASH	EQU	0003H
NMDBFUPDATE	EQU	0013H
NMDBFVERSION	EQU	000AH
DEVMANAGER	EQU	0085H
NMDEVDELETE	EQU	0008H
NMDEVFIND	EQU	000AH
NMDEVGETPDDADDRESS	EQU	0002H
NMDEVHOLD	EQU	0004H
NMDEVINSTALL	EQU	0003H
NMDEVLOADLDD	EQU	0006H
NMDEVLOADPDD	EQU	0007H
NMDEVOPENPDD	EQU	0001H
NMDEVQUERYUNITS	EQU	0009H
NMDEVREMOVE	EQU	000BH
NMDEVRESUME	EQU	0005H
NMDEVVECTOR	EQU	000CH
NMIOOPEN	EQU	0000H
FILMANAGER	EQU	0087H
NMFILCHANGEDIRECTORY	EQU	0011H
NMFILCONNECT	EQU	0000H
NMFILDELETE	EQU	0006H
NMFILEEXECUTE	EQU	0001H
NMFILLOCCHANGED	EQU	0014H
NMFILLOCDEVICE	EQU	0015H
NMFILLOCREADPDD	EQU	0016H
NMFILMAKEDIRECTORY	EQU	000CH
NMFIOPENUNIQUE	EQU	000DH
NMFILPARSE	EQU	0002H
NMFILPATHGET	EQU	0003H
NMFILPATHGETBYID	EQU	0010H

NMFILPATHSET	EQU	0004H
NMFILPATHTEST	EQU	0005H
NMFILRENAME	EQU	0007H
NMFILSETFILEDATE	EQU	0013H
NMFILSETINITIALPATH	EQU	0012H
NMFILSTATUSDEVICE	EQU	000AH
NMFILSTATUSGET	EQU	0008H
NMFILSTATUSSET	EQU	0009H
NMFILSTATUSSYSTEM	EQU	000BH
NMFISSYSTEMATTACH	EQU	000EH
NMFISSYSTEMDETACH	EQU	000FH
FLOATMANAGER	EQU	008CH
NMFLOATACOS	EQU	0004H
NMFLOATASIN	EQU	0003H
NMFLOATATAN	EQU	0005H
NMFLOATCOS	EQU	0001H
NMFLOATEXP	EQU	0006H
NMFLOATINT	EQU	000DH
NMFLOATLN	EQU	0007H
NMFLOATLOG	EQU	0008H
NMFLOATMOD	EQU	000CH
NMFLOATPOW	EQU	000AH
NMFLOATRAND	EQU	000BH
NMFLOATSIN	EQU	0000H
NMFLOATSQRT	EQU	0009H
NMFLOATTAN	EQU	0002H
GENMANAGER	EQU	008BH
NMGENALARMHOOK	EQU	002BH
NMGENALARMID	EQU	002DH
NMGENALARMUNHOOK	EQU	002CH
NMGENCRC	EQU	0029H
NMGENDEFERREDMODE	EQU	0008H
NMGENENVBUFFERDELETE	EQU	0023H

NMGENENVBUFFERFIND	EQU	0024H
NMGENENVBUFFERGET	EQU	0021H
NMGENENVBUFFERSET	EQU	0022H
NMGENENVSTRINGDELETE	EQU	0027H
NMGENENVSTRINGFIND	EQU	0028H
NMGENENVSTRINGGET	EQU	0025H
NMGENENVSTRINGSET	EQU	0026H
NMGENGETAMPTEXT	EQU	001DH
NMGENGETAUTOMAINS	EQU	0038H
NMGENGETAUTOSWITCHOFFVALUE	EQU	0017H
NMGENGETBATTERYTYPE	EQU	001FH
NMGENGETCOMMANDLINE	EQU	000EH
NMGENGETCOUNTRYDATA	EQU	0005H
NMGENGETERRORTEXT	EQU	0006H
NMGENGETLANGUAGECODE	EQU	001BH
NMGENGETNOTIFYSTATE	EQU	0015H
NMGENGETOSDATA	EQU	0007H
NMGENGETRAMSIZEINPARAS	EQU	000DH
NMGENGETSOUNDFLAGS	EQU	000FH
NMGENGETSUFFIXES	EQU	001CH
NMGENGETTEXT	EQU	0014H
NMGENLCDTYPE	EQU	0001H
NMGENMARKACTIVE	EQU	0012H
NMGENMARKNONACTIVE	EQU	0013H
NMGENMASKDECRYPT	EQU	0036H
NMGENMASKENCRYPT	EQU	0035H
NMGENMASKINIT	EQU	0034H
NMGENNOTIFY	EQU	0009H
NMGENNOTIFYERROR	EQU	000AH
NMGENNOTIFYHOOK	EQU	000BH
NMGENNOTIFYUNHOOK	EQU	000CH
NMGENPARSE	EQU	0003H
NMGENPASSWORDCONTROL	EQU	0030H
NMGENPASSWORDQUERY	EQU	0031H
NMGENPASSWORDSET	EQU	002EH
NMGENPASSWORDTEST	EQU	002FH
NMGENRESETREVECTOR	EQU	001AH

NMGENROMVERSION	EQU	002AH
NMGENSETAUTOMAINS	EQU	0039H
NMGENSETAUTOSWITCHOFFVALUE	EQU	0018H
NMGENSETBATTERYTYPE	EQU	0020H
NMGENSETCONFIG	EQU	0033H
NMGENSETCOUNTRYDATA	EQU	001EH
NMGENSETNOTIFYSTATE	EQU	0016H
NMGENSETONEVENTS	EQU	0037H
NMGENSETREVECTOR	EQU	0019H
NMGENSETSOUNDFLAGS	EQU	0010H
NMGENSOUND	EQU	0011H
NMGENSTARTREASON	EQU	0002H
NMGENTICKLE	EQU	0032H
NMGENVERSION	EQU	0000H
NMLONGUNSIGNEDINTRANDOM	EQU	0004H
HEAPMANAGER	EQU	0081H
NMHEAPADJUSTCELLSIZE	EQU	0002H
NMHEAPALLOCATECELL	EQU	0000H
NMHEAPCELLSIZE	EQU	0004H
NMHEAPFREECELL	EQU	0003H
NMHEAPFREEMEMORY	EQU	0006H
NMHEAPREALLOCATECELL	EQU	0001H
NMHEAPSETGRANULARITY	EQU	0005H
HWMANAGER	EQU	008EH
NMHWBACKLIGHT	EQU	0020H
NMHWCLEARA2CONTROL1BITS	EQU	0005H
NMHWCLEARA2CONTROL2BITS	EQU	0009H
NMHWCLEARA2CONTROL3BITS	EQU	000DH
NMHWCOMBOFF	EQU	0001H
NMHWCOMBOON	EQU	0000H
NMHWCOMBOONINPUT	EQU	0021H
NMHWEXIT	EQU	0016H
NMHWFORCESUPPLYREADING	EQU	001DH

NMHWFREECHANNEL	EQU	001AH
NMHWFREECOMBO	EQU	0018H
NMHWGETBACKLIGHT	EQU	001EH
NMHWGETCHANNEL	EQU	0019H
NMHWGETCOMBO	EQU	0017H
NMHWGETPSUTYPE	EQU	001BH
NMHWGETSCANCODES	EQU	0028H
NMHWGETSUPPLYSTATUS	EQU	0011H
NMHWLCDCONTRASTDELTA	EQU	0012H
NMHWNULLFRAME	EQU	0015H
NMHWPACKSOFF	EQU	0003H
NMHWPACKSON	EQU	0002H
NMHWREADA2CONTROL1	EQU	0006H
NMHWREADA2CONTROL2	EQU	000AH
NMHWREADA2CONTROL3	EQU	000EH
NMHWREADLCDCONTRAST	EQU	0013H
NMHWSELECTCHANNEL	EQU	0010H
NMHWSETA2CONTROL1BITS	EQU	0004H
NMHWSETA2CONTROL2BITS	EQU	0008H
NMHWSETA2CONTROL3BITS	EQU	000CH
NMHWSETBACKLIGHT	EQU	001FH
NMHSUPPLYINFO	EQU	0022H
NMHSUPPLYWARNINGS	EQU	001CH
NMHSWITCHOFF	EQU	0014H
NMHWWRITEA2CONTROL1	EQU	0007H
NMHWWRITEA2CONTROL2	EQU	000BH
NMHWWRITEA2CONTROL3	EQU	000FH
IOMANAGER	EQU	0086H
NMIOADDAPPLICATIONHANDLER	EQU	0015H
NMIOADDHANDLER	EQU	000BH
NMIOASYNCHRONOUS	EQU	0000H
NMIOASYNCHRONOUSNOERROR	EQU	0001H
NMIOCLOSE	EQU	0010H
NMIOENABLEAPPLICATIONHANDLER	EQU	0017H
NMIOENABLEHANDLER	EQU	000DH

NMIOKEYANDMOUSEASYNCHRONOUS	EQU	001CH
NMIOKEYANDMOUSEWITHWAIT	EQU	0014H
NMIONEXTHALFSECOND	EQU	001DH
NMIOPLAYSOUNDA	EQU	001EH
NMIOPLAYSOUNDCANCEL	EQU	0020H
NMIOPLAYSOUNDW	EQU	001FH
NMIOREAD	EQU	0011H
NMIORECORDSOUNDA	EQU	0021H
NMIORECORDSOUNDCANCEL	EQU	0023H
NMIORECORDSOUNDW	EQU	0022H
NMIOREMOVEAPPLICATIONHANDLER	EQU	0016H
NMIOREMOVEHANDLER	EQU	000CH
NMIOREQUESTRESET	EQU	000EH
NMIOREQUESTRESETCANCEL	EQU	000FH
NMIOROOT	EQU	0003H
NMIOSEEK	EQU	0013H
NMIOSHIFTSTATES	EQU	0018H
NMIOSIGNAL	EQU	0008H
NMIOSIGNALBYPID	EQU	0009H
NMIOSIGNALBYPIDNORESCHED	EQU	000AH
NMIOSIGNALKILLASYNCHRONOUS	EQU	001AH
NMIOSIGNALKILLCANCEL	EQU	001BH
NMIOSUPER	EQU	0004H
NMIOWAITFORSIGNAL	EQU	0005H
NMIOWAITFORSIGNALNOHANDLER	EQU	0019H
NMIOWAITFORSTATUS	EQU	0006H
NMIOWITHWAIT	EQU	0002H
NMIOWRITE	EQU	0012H
NMIOYIELD	EQU	0007H
IOSERMANAGER	EQU	00DEH
NMIOSERADDHANDLER	EQU	0001H
NMIOSERATTACHONOPENCHAN	EQU	000DH
NMIOSERCANCELALLSIGNALUSER	EQU	001CH
NMIOSERCANCELIOREQUEST	EQU	001BH
NMIOSERCHECKREADSI	EQU	0011H

NMIOUSERCHECKWRITESI	EQU	0010H
NMIOUSERCLOSETIMERHANDLER	EQU	0008H
NMIOUSERDETACHFREE	EQU	0009H
NMIOUSERFREE	EQU	0007H
NMIOUSERHANDLERSAVEERROR	EQU	0004H
NMIOUSERONOPENCHAN	EQU	000FH
NMIOUSEROPEN	EQU	0000H
NMIOUSEROPENHANDLER	EQU	0005H
NMIOUSEROPENTIMERHANDLER	EQU	0006H
NMIOUSERQUEUEUEREAD	EQU	0017H
NMIOUSERQUEUEUESUPER	EQU	0019H
NMIOUSERQUEUEUETIMER	EQU	001AH
NMIOUSERQUEUEUWRITE	EQU	0018H
NMIOUSERREMOVEHANDLER	EQU	0002H
NMIOUSERSENSEONOPENCHAN	EQU	000EH
NMIOUSERSETHANDLER	EQU	0003H
NMIOUSER SIGNALCOMPLETE	EQU	001EH
NMIOUSER SIGNALCOMPLETEOK	EQU	001DH
NMIOUSER SIGNALUSER	EQU	0016H
NMIOUSER SIGNALUSERREAD	EQU	0015H
NMIOUSER SIGNALUSERREADOK	EQU	0014H
NMIOUSER SIGNALUSERWRITE	EQU	0013H
NMIOUSER SIGNALUSERWRITEOK	EQU	0012H
NMIOUSER SYNCWRITE	EQU	001FH
NMIOUSER TIMERCANCEL	EQU	000BH
NMIOUSER TIMERCLOSE	EQU	000CH
NMIOUSER TIMEROPEN	EQU	000AH
LIBMANAGER	EQU	0084H
NMLIBCOPY	EQU	0008H
NMLIBCREATE	EQU	0005H
NMLIBCREATEBYHANDLE	EQU	0006H
NMLIBDESTROY	EQU	0007H
NMLIBFIND	EQU	0003H
NMLIBHANDLE	EQU	0004H
NMLIBLINK	EQU	0002H

NMLIBLOAD	EQU	0000H
NMLIBLOADFILE	EQU	000AH
NMLIBOPEN	EQU	0009H
NMLIBRECLASS	EQU	000BH
NMLIBRECLASSBYHANDLE	EQU	000CH
NMLIBUNLOAD	EQU	0001H
MESSMANAGER	EQU	0083H
NMMESSFREE	EQU	0007H
NMMESSINIT	EQU	0000H
NMMESSRECEIVEASYNCHRONOUS	EQU	0001H
NMMESSRECEIVECANCEL	EQU	0003H
NMMESSRECEIVEWITHWAIT	EQU	0002H
NMMESSSEND	EQU	0004H
NMMESSSENDRECEIVEASYNCHRONOUS	EQU	0005H
NMMESSSENDRECEIVEWITHWAIT	EQU	0006H
NMMESSSIGNAL	EQU	0008H
NMMESSSIGNALCANCEL	EQU	0009H
NMMESSSIGNALCANCELX	EQU	000AH
PROCMANAGER	EQU	0088H
NMPROCCREATE	EQU	0004H
NMPROCcreatETASK	EQU	0005H
NMPROCFind	EQU	000BH
NMPROCGETOWNER	EQU	0010H
NMPROCGETPRIORITY	EQU	0002H
NMPROCID	EQU	0000H
NMPROCIDBYNAME	EQU	0001H
NMPROCKILL	EQU	0008H
NMPROCNAMEBYID	EQU	000AH
NMPROCINTERMINATE	EQU	000EH
NMPROCPANICBYID	EQU	0009H
NMPROCRename	EQU	000CH
NMPROCRESUME	EQU	0006H
NMPROCSETPRIORITY	EQU	0003H

NMPROCSUSPEND	EQU	0007H
NMPROCTERMINATE	EQU	000DH
NMPROCWATCHALLEXITS	EQU	000FH
SEGMANAGER	EQU	0080H
NMSEGADJUSTSIZE	EQU	0006H
NMSEGCLOSE	EQU	0004H
NMSEGCLOSELOCKEDORDEVICE	EQU	000DH
NMSEGCOPYFROM	EQU	0009H
NMSEGCOPYTO	EQU	0008H
NMSEGCREATE	EQU	0001H
NMSEGDELETE	EQU	0002H
NMSEGFIND	EQU	0007H
NMSEGFREEMEMORY	EQU	0000H
NMSEGLOCK	EQU	000AH
NMSEGOPEN	EQU	0003H
NMSEGRAMDISKUSED	EQU	000CH
NMSEGSIZE	EQU	0005H
NMSEGUNLOCK	EQU	000BH
SEMMANAGER	EQU	0082H
NMSEMCREATE	EQU	0000H
NMSEMDELETE	EQU	0001H
NMSEMSIGNALMANY	EQU	0004H
NMSEMSIGNALONCE	EQU	0003H
NMSEMSIGNALONCENORESCHED	EQU	0005H
NMSEMWAIT	EQU	0002H
TIMMANAGER	EQU	0089H
NMTIMDATETODAYSECONDS	EQU	0007H
NMTIMDAYOFWEEK	EQU	0009H
NMTIMDAYSECONDSTODATE	EQU	0006H
NMTIMDAYSECONDSTOSYSTEMTIME	EQU	0005H
NMTIMDAYSINMONTH	EQU	0008H

NMTIMGETSYSTEMTIME	EQU	0002H
NMTIMNAMEOFDAY	EQU	000AH
NMTIMNAMEOFDAYABB	EQU	000EH
NMTIMNAMEOFMONTH	EQU	000BH
NMTIMNAMEOFMONTHABB	EQU	000FH
NMTIMSETSYSTEMTIME	EQU	0003H
NMTIMSLEEPFORTENTHS	EQU	0000H
NMTIMSLEEPFORTICKS	EQU	0001H
NMTIMSYSTEMTIMETODAYSECONDS	EQU	0004H
NMTIMWAITABSOLUTE	EQU	000CH
NMTIMWEEKNUMBER	EQU	000DH
BUFFERCOMPARE	EQU	00A3H
BUFFERCOMPAREFOLDED	EQU	00A4H
BUFFERCOPY	EQU	00A1H
BUFFERJUSTIFY	EQU	00ABH
BUFFERLOCATE	EQU	00A7H
BUFFERLOCATEFOLDED	EQU	00A8H
BUFFERMATCH	EQU	00A5H
BUFFERMATCHFOLDED	EQU	00A6H
BUFFERSUBBUFFER	EQU	00A9H
BUFFERSUBBUFFERFOLDED	EQU	00AAH
BUFFERSWAP	EQU	00A2H
CHARISALPHABETIC	EQU	0096H
CHARISALPHANUMERIC	EQU	0097H
CHARISCONTROL	EQU	009DH
CHARISDIGIT	EQU	0093H
CHARISGRAPHIC	EQU	009CH
CHARISHEXDIGIT	EQU	0094H
CHARISLOWERCASE	EQU	0099H
CHARISPRINTABLE	EQU	0095H
CHARISPUNCTUATION	EQU	009BH
CHARISSPACE	EQU	009AH
CHARISUPPERCASE	EQU	0098H
CHARTOFOLDEDCHAR	EQU	00A0H
CHARTOLOWERCHAR	EQU	009FH
CHARTOUPPERCHAR	EQU	009EH

DUMMY	EQU	00D4H
FLOATADD	EQU	00C1H
FLOATCOMPARE	EQU	00C5H
FLOATDIVIDE	EQU	00C4H
FLOATMULTIPLY	EQU	00C3H
FLOATNEGATE	EQU	00C6H
FLOATSUBTRACT	EQU	00C2H
FLOATTOINT	EQU	00C7H
FLOATTOLONG	EQU	00C9H
FLOATTOUNSIGNEDINT	EQU	00C8H
FLOATTOUNSIGNEDLONG	EQU	00CAH
GENDATASEGMENT	EQU	008FH
GENINTBYNUMBER	EQU	00D5H
INTTOFLOAT	EQU	00CBH
IOKEYANDMOUSESTATUS	EQU	00DAH
IONEXTHALFSECONDSTATUS	EQU	00DDH
LIBENTER	EQU	00D2H
LIBENTERSEND	EQU	00D9H
LIBLEAVE	EQU	00D3H
LIBSEND	EQU	00CFH
LIBSENDEXACT	EQU	00D1H
LIBSENDEXIT	EQU	00D7H
LIBSENDSUPER	EQU	00D0H
LONGINTCOMPARE	EQU	00BBH
LONGINTDIVIDE	EQU	00BDH
LONGINTMULTIPLY	EQU	00BCH
LONGTOFLOAT	EQU	00CDH
LONGUNSIGNEDINTCOMPARE	EQU	00BEH
LONGUNSIGNEDINTDIVIDE	EQU	00C0H
LONGUNSIGNEDINTMULTIPLY	EQU	00BFH
PROCCOPYFROMBYID	EQU	0091H
PROCCOPYTOBYID	EQU	0092H
PROCINDSTRINGCOPYFROMBYID	EQU	00DCH
PROCPANIC	EQU	0090H
STRINGCAPITALISE	EQU	00DBH
STRINGCOMPARE	EQU	00AFH
STRINGCOMPAREFOLDED	EQU	00B0H

STRINGCONVERTTOFOLDED	EQU	00AEH
STRINGCOPY	EQU	00ACH
STRINGCOPYFOLDED	EQU	00ADH
STRINGLENGTH	EQU	00B9H
STRINGLOCATE	EQU	00B3H
STRINGLOCATEFOLDED	EQU	00B4H
STRINGLOCATEINREVERSE	EQU	00B5H
STRINGLOCATEINREVERSEFOLDED	EQU	00B6H
STRINGMATCH	EQU	00B1H
STRINGMATCHFOLDED	EQU	00B2H
STRINGSUBSTRING EQU	00B7H	
STRINGSUBSTRINGFOLDED	EQU	00B8H
STRINGVALIDATENAME	EQU	00BAH
UNSIGNEDINTTOFLOAT	EQU	00CCH
UNSIGNEDLONGTOFLOAT	EQU	00CEH
WSERVFUNCTONS EQU	00D6H	
WSERVOPCODES	EQU	008DH

Numerical list of functions

SEGMANAGER	EQU	0080H
NMSEGFREEMEMORY	EQU	0000H
NMSEGCREATE	EQU	0001H
NMSEGDELETE	EQU	0002H
NMSEGOPEN	EQU	0003H
NMSEGCLOSE	EQU	0004H
NMSEGSIZE	EQU	0005H
NMSEGADJUSTSIZE	EQU	0006H
NMSEGFIND	EQU	0007H
NMSEGCOPYTO	EQU	0008H
NMSEGCOPYFROM	EQU	0009H
NMSEGLOCK	EQU	000AH
NMSEGUNLOCK	EQU	000BH
NMSEGRAMDISKUSED	EQU	000CH
NMSEGCLOSELOCKEDORDEVICE	EQU	000DH

HEAPMANAGER	EQU	0081H
NMHEAPALLOCATECELL	EQU	0000H
NMHEAPREALLOCATECELL	EQU	0001H
NMHEAPADJUSTCELLSIZE	EQU	0002H
NMHEAPFREECELL	EQU	0003H
NMHEAPCELLSIZE	EQU	0004H
NMHEAPSETGRANULARITY	EQU	0005H
NMHEAPFREEMEMORY	EQU	0006H
SEMMANAGER	EQU	0082H
NMSEMCREATE	EQU	0000H
NMSEMDELETE	EQU	0001H
NMSEMWAIT	EQU	0002H
NMSEMSIGNALONCE	EQU	0003H
NMSEMSIGNALMANY	EQU	0004H
NMSEMSIGNALONCENORESCHED	EQU	0005H
MESSMANAGER	EQU	0083H
NMMESSINIT	EQU	0000H
NMMESSRECEIVEASYNCHRONOUS	EQU	0001H
NMMESSRECEIVEWITHWAIT	EQU	0002H
NMMESSRECEIVECANCEL	EQU	0003H
NMMESSSEND	EQU	0004H
NMMESSSENDRECEIVEASYNCHRONOUS	EQU	0005H
NMMESSSENDRECEIVEWITHWAIT	EQU	0006H
NMMESSFREE	EQU	0007H
NMMESSSIGNAL	EQU	0008H
NMMESSSIGNALCANCEL	EQU	0009H
NMMESSSIGNALCANCELX	EQU	000AH
LIBMANAGER	EQU	0084H
NMLIBLOAD	EQU	0000H
NMLIBUNLOAD	EQU	0001H

NMLIBLINK	EQU	0002H
NMLIBFIND	EQU	0003H
NMLIBHANDLE	EQU	0004H
NMLIBCREATE	EQU	0005H
NMLIBCREATEBYHANDLE	EQU	0006H
NMLIBDESTROY	EQU	0007H
NMLIBCOPY	EQU	0008H
NMLIBOPEN	EQU	0009H
NMLIBLOADFILE	EQU	000AH
NMLIBRECLASS	EQU	000BH
NMLIBRECLASSBYHANDLE	EQU	000CH

DEVMANAGER	EQU	0085H
------------	-----	-------

NMIOOPEN	EQU	0000H
NMDEVOPENPDD	EQU	0001H
NMDEVGETPDDADDRESS	EQU	0002H
NMDEVINSTALL	EQU	0003H
NMDEVHOLD	EQU	0004H
NMDEVRESUME	EQU	0005H
NMDEVLOADLDD	EQU	0006H
NMDEVLOADPDD	EQU	0007H
NMDEVDELETE	EQU	0008H
NMDEVQUERYUNITS	EQU	0009H
NMDEVFIND	EQU	000AH
NMDEVREMOVE	EQU	000BH
NMDEVVECTOR	EQU	000CH

IOMANAGER	EQU	0086H
-----------	-----	-------

NMIOASYNCHRONOUS	EQU	0000H
NMIOASYNCHRONOUSNOERROR	EQU	0001H
NMIOWITHWAIT	EQU	0002H
NMIOROOT	EQU	0003H
NMIOSUPER	EQU	0004H
NMIOWAITFORSIGNAL	EQU	0005H
NMIOWAITFORSTATUS	EQU	0006H

NMIOYIELD	EQU	0007H
NMIOSIGNAL	EQU	0008H
NMIOSIGNALBYPID	EQU	0009H
NMIOSIGNALBYPIDNORESCHED	EQU	000AH
NMIOADDDHANDLER	EQU	000BH
NMIOREMOVEHANDLER	EQU	000CH
NMIOENABLEHANDLER	EQU	000DH
NMIOREQUESTRESET	EQU	000EH
NMIOREQUESTRESETCANCEL	EQU	000FH
NMIOCLOSE	EQU	0010H
NMIOREAD	EQU	0011H
NMIOWRITE	EQU	0012H
NMIOSEEK	EQU	0013H
NMIOKEYANDMOUSEWITHWAIT	EQU	0014H
NMIOADDAPPLICATIONHANDLER	EQU	0015H
NMIOREMOVEAPPLICATIONHANDLER	EQU	0016H
NMIOENABLEAPPLICATIONHANDLER	EQU	0017H
NMIOSHIFTSTATES	EQU	0018H
NMIOWAITFORSIGNALNOHANDLER	EQU	0019H
NMIOSIGNALKILLASYNCHRONOUS	EQU	001AH
NMIOSIGNALKILLCANCEL	EQU	001BH
NMIOKEYANDMOUSEASYNCHRONOUS	EQU	001CH
NMIONEXTHALFSECOND	EQU	001DH
NMIOPLAYSOUNDA	EQU	001EH
NMIOPLAYSOUNDW	EQU	001FH
NMIOPLAYSOUNDCANCEL	EQU	0020H
NMIORECORDSOUNDA	EQU	0021H
NMIORECORDSOUNDW	EQU	0022H
NMIORECORDSOUNDCANCEL	EQU	0023H
 FILMANAGER	 EQU	 0087H
 NMFILCONNECT	 EQU	 0000H
NMFILEXECUTE	EQU	0001H
NMFILPARSE	EQU	0002H
NMFILPATHGET	EQU	0003H
NMFILPATHSET	EQU	0004H

NMFILPATHTEST	EQU	0005H
NMFILDELETE	EQU	0006H
NMFILRENAME	EQU	0007H
NMFILSTATUSGET	EQU	0008H
NMFILSTATUSSET	EQU	0009H
NMFILSTATUSDEVICE	EQU	000AH
NMFILSTATUSSYSTEM	EQU	000BH
NMFILMAKEDIRECTORY	EQU	000CH
NMFIOPENUNIQUE	EQU	000DH
NMFI SYSTEMATTACH	EQU	000EH
NMFI SYSTEMDETACH	EQU	000FH
NMFILPATHGETBYID	EQU	0010H
NMFILCHANGEDIRECTORY	EQU	0011H
NMFILSETINITIALPATH	EQU	0012H
NMFILSETFILEDATE	EQU	0013H
NMFILLOCCHANGED	EQU	0014H
NMFILLOCDEVICE	EQU	0015H
NMFILLOCREADPDD	EQU	0016H
PROCMANAGER	EQU	0088H
NMPROCID	EQU	0000H
NMPROCIDBYNAME	EQU	0001H
NMPROCGETPRIORITY	EQU	0002H
NMPROCSETPRIORITY	EQU	0003H
NMPROCCREATE	EQU	0004H
NMPROCCREATETASK	EQU	0005H
NMPROCRESUME	EQU	0006H
NMPROCSUSPEND	EQU	0007H
NMPROCKILL	EQU	0008H
NMPROCPANICBYID	EQU	0009H
NMPROCNAMEBYID	EQU	000AH
NMPROCFIND	EQU	000BH
NMPROCRENAME	EQU	000CH
NMPROCTERMINATE	EQU	000DH
NMPROCONTERMINATE	EQU	000EH
NMPROCWATCHALLEXITS	EQU	000FH
NMPROCGETOWNER	EQU	0010H

TIMMANAGER	EQU	0089H
NMTIMSLEEPFORTENTHS	EQU	0000H
NMTIMSLEEPFORTICKS	EQU	0001H
NMTIMGETSYSTEMTIME	EQU	0002H
NMTIMSETSYSTEMTIME	EQU	0003H
NMTIMSYSTEMTIMETODAYSECONDS	EQU	0004H
NMTIMDAYSECONDSTOSYSTEMTIME	EQU	0005H
NMTIMDAYSECONDSTODATE	EQU	0006H
NMTIMDATETODAYSECONDS	EQU	0007H
NMTIMDAYSINMONTH	EQU	0008H
NMTIMDAYOFWEEK	EQU	0009H
NMTIMNAMEOFDAY	EQU	000AH
NMTIMNAMEOFMONTH	EQU	000BH
NMTIMWAITABSOLUTE	EQU	000CH
NMTIMWEEKNUMBER	EQU	000DH
NMTIMNAMEOFDAYABB	EQU	000EH
NMTIMNAMEOFMONTHABB	EQU	000FH
CONVMANAGER	EQU	008AH
NMCONVUNSIGNEDINTTOBUFFER	EQU	0000H
NMCONVUNSIGNEDLONGINTTOBUFFER	EQU	0001H
NMCONVINTTOBUFFER	EQU	0002H
NMCONVLONGINTTOBUFFER	EQU	0003H
NMCONVARGUMENTSTOBUFFER	EQU	0004H
NMCONVSTRINGTOUNSIGNEDINT	EQU	0005H
NMCONVSTRINGTOUNSIGNEDLONGINT	EQU	0006H
NMCONVSTRINGTOINT	EQU	0007H
NMCONVSTRINGTOLONGINT	EQU	0008H
NMCONVFLOATTOBUFFER	EQU	0009H
NMCONVSTRINGTOFLOAT	EQU	000AH
GENMANAGER	EQU	008BH
NMGENVERSION	EQU	0000H
NMGENLCDTYPE	EQU	0001H

NMGENSTARTREASON	EQU	0002H
NMGENPARSE	EQU	0003H
NMLONGUNSIGNEDINTRANDOM	EQU	0004H
NMGENGETCOUNTRYDATA	EQU	0005H
NMGENGETERRORTEXT	EQU	0006H
NMGENGETOSDATA	EQU	0007H
NMGENDEFERREDMODE	EQU	0008H
NMGENNOTIFY	EQU	0009H
NMGENNOTIFYERROR	EQU	000AH
NMGENNOTIFYHOOK	EQU	000BH
NMGENNOTIFYUNHOOK	EQU	000CH
NMGENGETRAMSIZEINPARAS	EQU	000DH
NMGENGETCOMMANDLINE	EQU	000EH
NMGENGETSOUNDFLAGS	EQU	000FH
NMGENSETSOUNDFLAGS	EQU	0010H
NMGENSOUND	EQU	0011H
NMGENMARKACTIVE	EQU	0012H
NMGENMARKNONACTIVE	EQU	0013H
NMGENGETTEXT	EQU	0014H
NMGENGETNOTIFYSTATE	EQU	0015H
NMGENSETNOTIFYSTATE	EQU	0016H
NMGENGETAUTOSWITCHOFFVALUE	EQU	0017H
NMGENSETAUTOSWITCHOFFVALUE	EQU	0018H
NMGENSETREVECTOR	EQU	0019H
NMGENRESETREVECTOR	EQU	001AH
NMGENGETLANGUAGECODE	EQU	001BH
NMGENGETSUFFIXES	EQU	001CH
NMGENGETAMPMTTEXT	EQU	001DH
NMGENSETCOUNTRYDATA	EQU	001EH
NMGENGETBATTERYTYPE	EQU	001FH
NMGENSETBATTERYTYPE	EQU	0020H
NMGENENVBUFFERGET	EQU	0021H
NMGENENVBUFFERSET	EQU	0022H
NMGENENVBUFFERDELETE	EQU	0023H
NMGENENVBUFFERFIND	EQU	0024H
NMGENENVSTRINGGET	EQU	0025H
NMGENENVSTRINGSET	EQU	0026H

NMGENENVSTRINGDELETE	EQU	0027H
NMGENENVSTRINGFIND	EQU	0028H
NMGENCRC	EQU	0029H
NMGENROMVERSION	EQU	002AH
NMGENALARMHOOK	EQU	002BH
NMGENALARMUNHOOK	EQU	002CH
NMGENALARMID	EQU	002DH
NMGENPASSWORDSET	EQU	002EH
NMGENPASSWORDTEST	EQU	002FH
NMGENPASSWORDCONTROL	EQU	0030H
NMGENPASSWORDQUERY	EQU	0031H
NMGENTICKLE	EQU	0032H
NMGENSETCONFIG	EQU	0033H
NMGENMASKINIT	EQU	0034H
NMGENMASKENCRYPT	EQU	0035H
NMGENMASKDECRYPT	EQU	0036H
NMGENSETONEVENTS	EQU	0037H
NMGENGETAUTOMAINS	EQU	0038H
NMGENSETAUTOMAINS	EQU	0039H
FLOATMANAGER	EQU	008CH
NMFLOATSIN	EQU	0000H
NMFLOATCOS	EQU	0001H
NMFLOATTAN	EQU	0002H
NMFLOATASIN	EQU	0003H
NMFLOATACOS	EQU	0004H
NMFLOATATAN	EQU	0005H
NMFLOATEXP	EQU	0006H
NMFLOATLN	EQU	0007H
NMFLOATLOG	EQU	0008H
NMFLOATSQRT	EQU	0009H
NMFLOATPOW	EQU	000AH
NMFLOATRAND	EQU	000BH
NMFLOATMOD	EQU	000CH
NMFLOATINT	EQU	000DH

WSERVOPCODES	EQU	008DH
HWMANAGER	EQU	008EH
NMHWCOMB00N	EQU	0000H
NMHWCOMB00FF	EQU	0001H
NMHWPACKSON	EQU	0002H
NMHWPACKSOFF	EQU	0003H
NMHWSETA2CONTROL1BITS	EQU	0004H
NMHWCLEARA2CONTROL1BITS	EQU	0005H
NMHWREADA2CONTROL1	EQU	0006H
NMHWWRITEA2CONTROL1	EQU	0007H
NMHWSETA2CONTROL2BITS	EQU	0008H
NMHWCLEARA2CONTROL2BITS	EQU	0009H
NMHWREADA2CONTROL2	EQU	000AH
NMHWWRITEA2CONTROL2	EQU	000BH
NMHWSETA2CONTROL3BITS	EQU	000CH
NMHWCLEARA2CONTROL3BITS	EQU	000DH
NMHWREADA2CONTROL3	EQU	000EH
NMHWWRITEA2CONTROL3	EQU	000FH
NMHWSELECTCHANNEL	EQU	0010H
NMHWGETSUPPLYSTATUS	EQU	0011H
NMHWLCDCONTRASTDELTA	EQU	0012H
NMHWREADLCDCONTRAST	EQU	0013H
NMHWSWITCHOFF	EQU	0014H
NMHWNULLFRAME	EQU	0015H
NMHWEXIT	EQU	0016H
NMHWGETCOMBO	EQU	0017H
NMHWFREECOMBO	EQU	0018H
NMHWGETCHANNEL	EQU	0019H
NMHWFREECHANNEL	EQU	001AH
NMHWGETPSUTYPE	EQU	001BH
NMHWSUPPLYWARNINGS	EQU	001CH
NMHWFORCESUPPLYREADING	EQU	001DH
NMHWGETBACKLIGHT	EQU	001EH
NMHWSETBACKLIGHT	EQU	001FH
NMHWBACKLIGHT	EQU	0020H

NMHWCOMBOONINPUT	EQU	0021H
NMHSUPPLYINFO	EQU	0022H
NMHWGETSCANCODES	EQU	0028H
GENDATASEGMENT	EQU	008FH
PROCPANIC	EQU	0090H
PROCCOPYFROMBYID	EQU	0091H
PROCCOPYTOBYID	EQU	0092H
CHARISDIGIT	EQU	0093H
CHARISHEXDIGIT	EQU	0094H
CHARISPRINTABLE	EQU	0095H
CHARISALPHABETIC	EQU	0096H
CHARISALPHANUMERIC	EQU	0097H
CHARISUPPERCASE	EQU	0098H
CHARISLOWERCASE	EQU	0099H
CHARISSPACE	EQU	009AH
CHARISPUNCTUATION	EQU	009BH
CHARISGRAPHIC	EQU	009CH
CHARISCONTROL	EQU	009DH
CHARTOUPPERCHAR	EQU	009EH
CHARTOLOWERCHAR	EQU	009FH
CHARTOFOLDEDCHAR	EQU	00A0H
BUFFERCOPY	EQU	00A1H
BUFFERSWAP	EQU	00A2H
BUFFERCOMPARE	EQU	00A3H
BUFFERCOMPAREFOLDED	EQU	00A4H
BUFFERMATCH	EQU	00A5H
BUFFERMATCHFOLDED	EQU	00A6H
BUFFERLOCATE	EQU	00A7H
BUFFERLOCATEFOLDED	EQU	00A8H
BUFFERSUBBUFFER	EQU	00A9H
BUFFERSUBBUFFERFOLDED	EQU	00AAH
BUFFERJUSTIFY	EQU	00ABH
STRINGCOPY	EQU	00ACH
STRINGCOPYFOLDED	EQU	00ADH
STRINGCONVERTTOFOLDED	EQU	00AEH
STRINGCOMPARE	EQU	00AFH

STRINGCOMPAREFOLDED	EQU	00B0H
STRINGMATCH	EQU	00B1H
STRINGMATCHFOLDED	EQU	00B2H
STRINGLOCATE	EQU	00B3H
STRINGLOCATEFOLDED	EQU	00B4H
STRINGLOCATEINREVERSE	EQU	00B5H
STRINGLOCATEINREVERSEFOLDED	EQU	00B6H
STRINGSUBSTRING	EQU	00B7H
STRINGSUBSTRINGFOLDED	EQU	00B8H
STRINGLENGTH	EQU	00B9H
STRINGVALIDATENAME	EQU	00BAH
LONGINTCOMPARE	EQU	00BBH
LONGINTMULTIPLY	EQU	00BCH
LONGINTDIVIDE	EQU	00BDH
LONGUNSIGNEDINTCOMPARE	EQU	00BEH
LONGUNSIGNEDINTMULTIPLY	EQU	00BFH
LONGUNSIGNEDINTDIVIDE	EQU	00C0H
FLOATADD	EQU	00C1H
FLOATSUBTRACT	EQU	00C2H
FLOATMULTIPLY	EQU	00C3H
FLOATDIVIDE	EQU	00C4H
FLOATCOMPARE	EQU	00C5H
FLOATNEGATE	EQU	00C6H
FLOATTOINT	EQU	00C7H
FLOATTOUNSIGNEDINT	EQU	00C8H
FLOATTOLONG	EQU	00C9H
FLOATTOUNSIGNEDLONG	EQU	00CAH
INTTOFLOAT	EQU	00CBH
UNSIGNEDINTTOFLOAT	EQU	00CCH
LONGTOFLOAT	EQU	00CDH
UNSIGNEDLONGTOFLOAT	EQU	00CEH
LIBSEND	EQU	00CFH
LIBSENDSUPER	EQU	00D0H
LIBSENDEXACT	EQU	00D1H
LIBENTER	EQU	00D2H
LIBLEAVE	EQU	00D3H
DUMMY	EQU	00D4H

GENINTBYNUMBER	EQU	00D5H
WSERVFUNCTIONS	EQU	00D6H
LIBSENDEXIT	EQU	00D7H
DBFMANAGER	EQU	00D8H
NMDBFOPEN	EQU	0000H
NMDBFCLOSE	EQU	0001H
NMDBFFLUSH	EQU	0002H
NMDBFTRASH	EQU	0003H
NMDBFCOPYDOWN	EQU	0004H
NMDBFCOMPRESS	EQU	0005H
NMDBFCOPYFILE	EQU	0006H
NMDBFFILESIZE	EQU	0007H
NMDBFEXTHEADERREAD	EQU	0008H
NMDBFEXTHEADERWRITE	EQU	0009H
NMDBFVERSION	EQU	000AH
NMDBFABSREADSENSE	EQU	000BH
NMDBFABSREAD	EQU	000CH
NMDBFNEXTREAD	EQU	000DH
NMDBFBACKREAD	EQU	000EH
NMDBFFIRSTREAD	EQU	000FH
NMDBFLASTREAD	EQU	0010H
NMDBFAPPEND	EQU	0011H
NMDBFERASEREAD	EQU	0012H
NMDBFUPDATE	EQU	0013H
NMDBFFINDREAD	EQU	0014H
NMDBFSENSE	EQU	0015H
NMDBFCOUNT	EQU	0016H
NMDBFDESCRECORDREAD	EQU	0017H
NMDBFDESCRECORDWRITE	EQU	0018H
NMDBFFINDREADFIELD	EQU	0019H
LIBENTERSEND	EQU	00D9H
IOKEYANDMOUSESTATUS	EQU	00DAH
STRINGCAPITALISE	EQU	00DBH
PROCINDSTRINGCOPYFROMBYID	EQU	00DCH

IONEXTHALFSECONDSTATUS	EQU	00DDH
IOSERMANAGER	EQU	00DEH
NMIOSEROPEN	EQU	0000H
NMIOSERADDDHANDLER	EQU	0001H
NMIOSERREMOVEHANDLER	EQU	0002H
NMIOSERSETHANDLER	EQU	0003H
NMIOSERHANDLERSAVEERROR	EQU	0004H
NMIOSEROPENHANDLER	EQU	0005H
NMIOSEROPENTIMERHANDLER	EQU	0006H
NMIOSERFREE	EQU	0007H
NMIOSERCLOSETIMERHANDLER	EQU	0008H
NMIOSERDETACHFREE	EQU	0009H
NMIOSERTIMEROPEN	EQU	000AH
NMIOSERTIMERCANCEL	EQU	000BH
NMIOSERTIMERCLOSE	EQU	000CH
NMIOSERATTACHONOPENCHAN	EQU	000DH
NMIOSESENSEONOPENCHAN	EQU	000EH
NMIOSERONOPENCHAN	EQU	000FH
NMIOSERCHECKWRITESI	EQU	0010H
NMIOSERCHECKREADSI	EQU	0011H
NMIOSERSIGNALUSERWRITEOK	EQU	0012H
NMIOSERSIGNALUSERWRITE	EQU	0013H
NMIOSERSIGNALUSERREADOK	EQU	0014H
NMIOSERSIGNALUSERREAD	EQU	0015H
NMIOSERSIGNALUSER	EQU	0016H
NMIOSERQUEUEUREAD	EQU	0017H
NMIOSERQUEUEUWRITE	EQU	0018H
NMIOSERQUEUEUSUPER	EQU	0019H
NMIOSERQUEUEUTIMER	EQU	001AH
NMIOSERCANCELIOREQUEST	EQU	001BH
NMIOSERCANCELALLSIGNALUSER	EQU	001CH
NMIOSERSIGNALCOMPLETEOK	EQU	001DH
NMIOSERSIGNALCOMPLETE	EQU	001EH
NMIOSERSYNCSWRITE	EQU	001FH

Additional Interrupt and Function numbers

The majority of the additional EPOC system services functions described in this section were introduced for the Series 3c and Siena.

With the exception of the HC, all the services are, in principle, available on any machine that contains EPOC version 3.90F or later. On an HC with a suitable version of EPOC, all the functions described in this section should generate an E_GEN_NSUP error.

Some services require the presence of hardware that is not built into all machines in the SIBO range. If the relevant hardware is not present on a particular machine, calling the service will either have no effect or return an error of E_GEN_NSUP.

Alphabetical list of extra functions

HWMANAGER	EQU	008EH
NMHWENABLEAUTOBATRESET	EQU	002bH
NMHWEXPANSIONOFF	EQU	0033H
NMHWEXPANSIONON	EQU	0032H
NMHWGETBATDATA	EQU	002cH
NMHWRELOGPACKS	EQU	002eH
NMHWRESETBATTERYSTATUS	EQU	002aH
NMHWRETURNEXPANSIONPORTSTATE	EQU	0031H
NMHWRETURNTICKCOUNT	EQU	0030H
NMHWSETIRPOWERLEVEL	EQU	002fH
IOMANAGER	EQU	0086H
NMIOPLAYSOUNDAO	EQU	0024H

Numerical list of extra functions

IOMANAGER	EQU	0086H
NMIOPLAYSOUNDAO	EQU	0024H
HWMANAGER	EQU	008EH
NMHWRESETBATTERYSTATUS	EQU	002aH
NMHWENABLEAUTOBATRESET	EQU	002bH
NMHWGETBATDATA	EQU	002cH
NMHWRELOGPACKS	EQU	002eH
NMHWSETIRPOWERLEVEL	EQU	002fH
NMHWRETURNTICKCOUNT	EQU	0030H
NMHWRETURNEXPANSIONPORTSTATE	EQU	0031H
NMHWEXPANSIONON	EQU	0032H
NMHWEXPANSIONOFF	EQU	0033H

APPENDIX B

ENVIRONMENT VARIABLES

This document is a beta version and is subject to change.

This chapter documents all environment variables that, at the time of writing, are created or read by Psion's software running on SIBO machines. Note that the names of all such environment variables contain the '\$' character.

Environment variables survive a soft reset but are cleared on a hard reset. Some environment variables will be restored to their default values, by being loaded from a ROM initialisation file, on a hard reset. The set of environment variables that are restored in this way depends on both the machine type and the machine's localisation (language).

PLIB

EM\$

This environment variable is used by the CLIB and PLIB startup modules. It contains a string that specifies a search path for the 8087 emulator, `sys$8087.ldr`.

Window server

\$WS_FL

On an HC with version 3.5 of the window server, and in all machines that use version 4 or later, the initial value of the internal parameter that is set by `wSystem` is loaded from the `$WS_FL` environment variable when the window server starts.

After setting `$WS_FL` to the desired `WORD` value, you must reset the HC by pressing the recessed reset button to make the new value effective.

The `wSystem` flags parameter is made up by ORing a number of bit fields of the form `WSERV_FLAG_XXX`.

After a hard reset on an HC with version 3.5 of the window server, the `$WS_FL` environment variable does not exist (which is equivalent to it being zero).

The following example program sets the `$WS_FL` environment variable:

```
#include <plib.h>
#include <wlib.h>

GLDEF_C INT main(VOID)
{
    WORD flags;

    flags=WSERV_FLAG_NO_NOTIFIER_REBOOT|WSERV_FLAG_HOOK_NOTIFIER
        |WSERV_FLAG_LOW_BATTERY_WARNINGS|WSERV_FLAG_HUNG_UP_SW;
    return(p_setenvron("$WS_FL", 6, &flags, 2));
}
```

After running this program and resetting the HC, the window server will:

- provide the notifier service
- report low battery voltages
- present a hung-up status window if an application hangs
- report a process that terminates with a panic or with a negative reason number

\$WS_FNTS

This environment variable contains a series of words, each of which contains the index of a font used by the window server. The fonts are as follows:

- System font
- Notifier/Alert font
- Status Window font
- Symbols font used for the status window diamond symbol
- Medium 2 digital clock font
- Medium 2 date font
- Notifier/alert button font
- Small status window clock font

\$WS_IF

On the HC, the font used for output that is not graphics context directed is determined by the `$WS_IF` ("Internal Font") environment variable.

This should contain a word binary value of 0 for `WS_FONT_BASE`, 1 for `WS_FONT_BASE+1`, and so on. If you change the value of `$WS_IF`, you must reset the machine by pressing the recessed reset button to effect the change.

The "factory" setting of `$WS_IF` is 4 (which selects the S3 font).

\$WS_SD

Pressing SHIFT-CTRL-PSION-S on an MC, S3, S3a, S3c, Siena or a *Workabout* saves the current screen to a file called *screen.pic* in the current path of the window server. Any existing file of the same name is replaced.

In practice, the current path of the window server on a SIBO machine is always `LOC::M:\` (it is defined when the window server process is started - well before you have any chance of influencing it).

However, if an environment variable with the name `$WS_SD` exists, the window server uses its value to open the file to be created. For example, running the following program:

```
#include <p_std.h>

GLDEF_C INT main(VOID)
{
    p_setenv("$WS_SD", "B:\\SCREEN.PIC");
    return(0);
}
```

subsequently causes the screen dump to be written to the root directory of the local *B:* drive.

If the save fails for any reason (such as disk full), the file is not produced and no notification of the failure is given.

You can use this behaviour to disable the SHIFT-CTRL-PSION-S screen dump key by setting up `$WS_SD` to contain an illegal file specification. For example, just inserting the following line of code:

```
p_setenv("$WS_SD", "");
```

disables the screen dump key.

\$WS_SF, \$WS_SF2 and \$WS_SF4

On the HC, the Siena and the Series 3, Series 3a and Series 3c, the system font is determined by the \$WS_SF environment variable which should contain a WORD binary value of 0 for WS_FONT_BASE, a WORD binary value of 1 for WS_FONT_BASE+1, and so on. If you change the value of \$WS_SF, you must reset the machine by pressing the recessed reset button to effect the change.

On the MC, the system font is determined in the same way, except that two alternative environment variables are used; \$WS_SF2 and \$WS_SF4. If the screen has fewer than 300 lines (as on the MC200), \$WS_SF2 is used. Otherwise (as on the MC400), \$WS_SF4 is used.

The following program illustrates how the environment variable may be changed.

```
#include <p_std.h>

GLDEF_C INT main(VOID)
{
    WORD flags;

    flags=1; /* choose WS_FONT_BASE+1 */
    return(p_setenv("WS_SF",6,&flags,2));
}
```

Changing the system font may have an adverse effect on existing applications.

HWIM

M\$V

Evaluator format preferences, stored as an HWIM EXTENDED_MEM_VALUES structure. This environment variable is read and written by the ws_eval_env method of the WSERV class. The default values are:

evalDegrees	DEGREES_MODE
calcDegrees	DEGREES_MODE
memVal.evalFormat	P_DTOB_FIXED
memVal.evalDPlaces	EVAL_DEFAULT_PLACES
memVal.calcFormat	P_DTOB_GENERAL
memVal.calcDPlaces	CALC_DEFAULT_PLACES
memVal.values[0] to memVal.values[9]	0.0

D\$X

Telephone dialling preferences, stored as a DIAL_ENVAR structure. The environment variable is read and written by ws_dial_env method of the WSERV class. The normal default values are:

toneLengthTicks	8
delayLengthTicks	8
pauseLengthTicks	48
dialOutCode[]	"9,"

These values may vary in non-English machines.

L\$X

This environment variable is read by the Series 3c only, to provide a possible extra option for the 'Use' choice list of the System Screen's 'Communications' dialog.

If it exists, the environment variable should contain three leading byte counted items which are, in order:

- text for the extra option, which will be appended to the choice list
- the full file specification of the file to `p_execc` if the new option is selected
- any additional command line data

For example, to add an 'IRcom' option that, on selection, executes the file `loc::m:\sys$irc.img`, passing it the command line "-P1", the environment variable could be set (using an HC-style Command Processor).by:

```
set L$X=\05IRCom\13LOC::M:\SYS$IRC.IMG\03-P1
```

Additional command line options can be appended to any specified in the environment variable by use of the 'Extra parameters' line in the System screen's 'Communications' dialog.

Printing

P\$D

The type of the port used for printing, held as a zero terminated character containing a single ASCII digit. The possible port types and their representations are:

PRINTER_PORT_PARALLEL	'0'
PRINTER_PORT_SERIAL	'1'
PRINTER_PORT_FILE	'2'
PRINTER_PORT_FAX	'3'

The default value represents `PRINTER_PORT_PARALLEL`.

These environment variables are set/created by the `PRINTER pr_set_port_type` method, and got by the `PRINTER pr_port_data` method, (see the *FORM Reference* manual).

P\$F

The name of the print file, that is, the file to which printing is to be directed, held as a zero terminated character string. The default print file name is *p.lis*.

This environment variable is set/created by the `PRINTER pr_store_file` method, and got by the `PRINTER pr_port_data` method, (see the *FORM Reference* manual).

P\$S

The characteristics of the serial port when it is used for printing, held as a `P_SRCHAR` structure. The default values are:

tbaud	P_BAUD_9600
rbaud	P_BAUD_9600
frame	P_DATA_8
parity	0
hand	P_OBEY_XOFF P_OBEY_DSR P_IGN_CTS
xoff	0x13
xon	0x11
flags	0
tmask	0

This environment variable is set/created by the `PRINTER pr_store_srchar` method, and got by the `PRINTER pr_port_data` method, (see the *FORM Reference* manual).

P\$M

The specification of the current printer model, held as a zero terminated character string. The string contains an ASCII digit, followed by the name of a printer driver (*.wdr*) file, where the digit specifies the index number, starting from zero, of the particular model within the printer driver file. The default value is "OBJ.WDR" (the file *bj.wdr* is present in the ROM of all relevant machines and contains only one model -that for the BJ-10e printer).

This environment variable is set/created by the `PRINTER pr_set_model` method, and got by the `PRINTER pr_sense_model` method, (see the *FORM Reference* manual).

P\$P

This environment variable contains two bytes of data that specify the display preferences for print preview. The first byte is an ASCII digit specifying the number of pages to display. This must be in the range '1' to '4' inclusive. The second byte is also an ASCII digit, which may be '1', indicating that margins are to be visible during print preview, or '0'.

This environment variable is set/created by the `PRVIEW wn_init` method (see the *XADD Reference* manual).

P\$PP

The port used for parallel printing, specified as a single ASCII character, for example, 'B'. This environment variable should only be set on machines that have more than one port, such as the *Workabout*. It is read if it exists, but is not created, by the `pr_sense_port` method of the `PRINTER` class, in the FORM library.

P\$SP

The port used for serial printing, specified as a single ASCII character, for example, 'A'. This environment variable should only be set on machines that have more than one port, such as the *Workabout*. It is read if it exists, but is not created, by the `pr_sense_port` method of the `PRINTER` class, in the FORM library.

P\$Z

The paper size, stored as a single ASCII digit. It is normally '0' (A4) or '4' (Letter).

This environment variable is only used on the Siena and the Series 3c. It is not supported on Siena machines with a version number of 4.20 and below.

P\$IIP

A single ASCII character, specifying the port letter for the IR printing device. This environment variable is used on the Siena and the Series 3c only.

P\$PX

This environment variable contains the device type and the serial characteristics for 'Parallel' printing. It is used only on the Siena and the Series 3c, which communicate with the Parallel cable via a serial interface.

The environment variable contains a one byte device type (0 is parallel) followed by a `P_SRCHAR` struct, as defined in *p_serial.h*.

Calculator application

C\$CALC

This environment variable is used on the Siena and Series 3c machines only, to store Calculator display preferences. It contains the following structure:

```
typedef struct
{
    INT    bitmapId;
    INT    currentView;          /* store current Calc View */
    INT    statusWinSize;       /* store status window size */
    INT    nDec;                /* -1=off, or 0..4 fixed dec places */
    INT    zoom;                /* zoom setting for Advanced view */
    DOUBLE memory;
}CR_CALC_ENV_INFO;
```

whose members have the following meanings:

BitmapId	for internal use only
CurrentView	0=Desk view, 1=Advanced view
StatusWinSize	one of the Window server flags: W_STATUS_WINDOW_OFF, W_STATUS_WINDOW_SMALL, or (Series 3c only) W_STATUS_WINDOW_BIG
NDec	used by the Desk view: values can be 0 to 4 inclusive, to specify the fixed number of decimal places to display, or -1 to display a variable number of decimal places
Zoom	used by the Advanced view to contain the ID of the font used in the current zoom state. For the Siena, the allowed range of values is FONT_ID_SWISS_8 to FONT_ID_SWISS_8+3 inclusive, and for the Series 3c the range is FONT_ID_SWISS_8 to FONT_ID_SWISS_8+4 inclusive
Memory	the current contents of the Desk view's memory.

M\$0M0 to M\$9M9

These environment variables contain the current values of the ten (Advanced view) calculator memories.

Note that the names of these environment variables are dependent on the names of the memories, as seen from within the Calculator application. If, for example, memory M2 is renamed to "Memory2", the environment variable M\$2M2 will be replaced by an environment variable with the name M\$2MEMORY2.

The name of each of these environment variables will never exceed eleven characters.

Tips application

TW\$\$

Contains permanent data for the Tips application. The data consists of a single byte containing two flags:

0x02	if set, the display of tips is enabled
0x04	if set, tips are displayed once per day, otherwise they are displayed whenever the machine is turned on

World application

W\$C

Contains the display preferences for the World application, as three WORDS:

clock type	either WS_CLOCK_FORCE_ANALOG or WS_CLOCK_FORCE_DIGITAL
map colour	either TRUE for a grey map or FALSE for a black map
distance units	one of WR_UNITS_MILES (0), WR_UNITS_KILOMETERS (1) or WR_UNITS_NAUTICAL (2)

W\$R

This environment variable stores permanent data for the World database services. The content has three elements:

- a signature for the world database file, including the file version,
- data specifying the home city,
- data specifying the default country, that is, the country to which telephone numbers are assumed to belong if a particular country is not specified.

Spell/Thesaurus

SP\$DRV

This environment variable identifies the drive that contains the Spellchecker's global dictionary, as set from the Spell application's Install menu option. It contains a single ASCII character that may be 'A', 'B' or 'M'.

SP\$OPT

This environment variable stores the preferences settings from the Spell application as a series of flags, stored in a single UWORD. The contents affect the spellchecker and thesaurus (although not necessarily used by both)

The content is an ored combination of the following set of values, selected by the user from the Spell application's Preferences menu option:

0x0100	if set, ignore words all in upper case
0x0200	if set, ignore words containing punctuation
0x0400	if set, ignore repeated words
0x0800	if set, ignore the case of repeated words
0x1000	if set, show the definitions window

WP\$SPEL

This is used by all applications that may wish to access the Spellchecker. The content is a single byte with a value of zero, but has no significance; the mere existence of the environment variable indicates that the Spellchecker is currently installed.

WP\$THES

This is used by all applications that may wish to access the Thesaurus. The content is a single byte with a value of zero, but has no significance; the mere existence of the environment variable indicates that the Thesaurus is currently installed.

3Fax application

F\$X

This environment variable contains two bytes of preferences.

The first byte contains one of the ASCII characters 'M', 'A' or 'B', representing the drive that is currently used to store the application's intermediate files.

The second byte contains a combination of the following flags:

0x01	if set, a new fax job is created on selection of 'Print to fax'. Otherwise, the document is simply processed to produce an intermediate file, for later sending
0x02	if set, intermediate files are automatically deleted after they have been sent

F\$XM

Contains the current 3Fax modem parameters.

F\$XP

Stores power usage data for the 3Fax device. Contains the time on batteries and the time on mains.

Email applications

MAIL\$ST

This environment variable is used, with some differences in content, by both the Corporate and the Internet PsiMail applications. It is created by an email application whenever a mail session completes, to contain data passed from the message transfer agent (MTA) to the mail client. It is not a permanent store of data, as it is deleted and recreated every time the MTA starts.

A MAIL\$ST environment variable created by the Corporate mail application will not disrupt the Internet mail application, should it be run on the same machine, and vice versa.

The content for the Corporate application consists of a sequence of five UWORDS, in the following order:

- a count of the messages that were sent
- a count of the messages that were received
- a count of the messages that were not sent
- a count of the messages that are marked as read
- a flag which, if set to TRUE, indicates that some messages were not received

The content for the Internet application consists of a sequence of eight UWORDS, in the following order:

- a count of the messages that were sent
- a count of the messages that were received
- a count of the messages that were not sent
- a count of the messages that are marked as read
- a count of messages that were deleted from the mail server
- the return code from the MTA
- the return code from the sending process (normally 0)
- the return code from the receiving process (normally 0)

As can be seen from the above lists, the first four items are common to both variants of MAIL\$ST.

Workabout

The following environment variables are used only on Workabout machines. See also P\$PP and P\$SP, described in the *Printing* section of this chapter.

S\$SVER

Contains a text string representing the Workabout Startup Shell version number, for example, “1.00F”.

C\$P@

This environment variable is set when exiting from the Workabout command processor. It contains a single ASCII character representing the current drive, with a default value of ‘M’.

C\$PA to C\$PZ

The environment variable C\$PA may be set when exiting from the Workabout command processor, to contain a text string representing the current path on drive A. It is not set if the drive A path is to the root directory.

Similar environment variables may be set for all other possible drives - C\$PB to C\$PZ inclusive.

C\$P£

This environment variable contains the parameters used by Link when accessed from the Workabout System Screen and/or Command Processor.

C\$P\$

This environment variable is set following selection of the keyboard from the Command Processor or the System Screen. It contains a single byte whose binary value is either 0 (Standard keyboard selected) or 1 (Special keyboard selected).

INDEX

- `$WS_FL`
 - environment variable, B-1
- `$WS_FNTS`
 - environment variable, B-2
- `$WS_IF`
 - environment variable, B-2
- `$WS_SD`
 - environment variable, B-2
- `$WS_SF`
 - environment variable, B-3
- `$WS_SF2`
 - environment variable, B-3
- `$WS_SF4`
 - environment variable, B-3
- `.wve` files
 - sound file format, 8-1
- active
 - marking a process, 19-7
 - unmarking a process, 19-8
- add
 - two floats, 14-2
- adjust
 - a heap memory cell size, 3-2
 - size of a memory segment, 2-6
- alarm
 - getting the server pid, 19-15
 - hooking the interface, 19-15
 - unhooking the interface, 19-15
- allocate
 - a heap memory cell, 3-1
 - re-allocating a heap memory cell, 3-2
- am
 - getting the amtext, 19-11
- append
 - a DBFrecord, 20-10
- arcsine
 - float function, 15-1
- arctangent
 - float function, 15-1
- asynchronous
 - I/O, 8-2
 - I/O without error reporting, 8-2
 - message reception, 5-2
- attach
 - a file system, 9-6
- auto-switch-off
 - disable/enable if mains present, 19-16
 - get state if mains present, 19-16
 - processes and, 19-8
 - processes and, 19-7
 - resetting, 19-15
 - setting value, 19-9
- Auto-switch-off
 - Getting value, 19-9
- battery
 - enable/disable reset, 21-10
 - getting type, 19-11
 - reset the status, 21-10
 - return pointer to information, 21-11
 - setting type, 19-11
- buffer
 - comparing, 17-1
 - comparing folded, 17-2
 - copying, 17-1
 - justifying, 17-4
 - locating, 17-2
 - locating folded, 17-2
 - subbuffer, 17-3
 - sub-buffer folded, 17-3
 - swapping, 17-1
 - wild card match, 17-3
 - wild card match folded, 17-4
- BufferCompare
 - compare buffers service, 17-1
- BufferCompareFolded
 - compare buffers folded service, 17-2
- BufferCopy
 - copy buffer service, 17-1
- BufferJustify
 - justify a buffer service, 17-4
- BufferLocate
 - locate a character in buffer service, 17-2
- BufferLocateFolded
 - locate a character in buffer folded service, 17-2
- BufferMatch
 - match a wildcard buffer service, 17-3
- BufferMatchFolded
 - match a wildcard buffer folded service, 17-4
- BufferSubBuffer
 - find a sub-buffer in a buffer service, 17-3
- BufferSubBufferFolded
 - find a sub-buffer in a buffer folded service, 17-3
- BufferSwap
 - swap buffers service, 17-1
- C\$CALC
 - environment variable, B-6
- C\$P\$
 - environment variable, B-9
- C\$P@
 - environment variable, B-9
- C\$P£
 - environment variable, B-9
- C\$PA to C\$PZ
 - environment variables, B-9
- cancel
 - message receive, 5-3
 - playing back sound file, 8-13
 - recording sound to file, 8-14
 - signal from the supervisor, 5-5
 - signal from the supervisor by type, 5-6
 - signal from the supervisor I/O, 8-11
- capitalising
 - a string, 18-1

- category
 - copying data from, 6-7
- change
 - size of a memory segment, 2-6
- character
 - is a digit, 16-1
 - is a hexadecimal digit, 16-1
 - is alphabetic, 16-1
 - is alphabetic or digit, 16-2
 - is graphic, 16-3
 - is lowercase, 16-2
 - is printable, 16-1
 - is punctuation, 16-2
 - is space, 16-2
 - is uppercase, 16-2
 - to fold, 16-3
 - to uppercase, 16-3
- Character
 - is control, 16-3
 - to lowercase, 16-3
- CharIsAlpha
 - character is alphabetic service, 16-1
- CharIsAlphaNumeric
 - character is alphabetic or digit service, 16-2
- CharIsControl
 - character is control service, 16-3
- CharIsDigit
 - character is a digit service, 16-1
- CharIsGraphic
 - character is graphic service, 16-3
- CharIsHexDigit
 - character is a hexadecimal digit service, 16-1
- CharIsLowerCase
 - character is lowercase service, 16-2
- CharIsPrintable
 - character is printable service, 16-1
- CharIsPunctuation
 - character is punctuation service, 16-2
- CharIsSpace
 - character is space service, 16-2
- CharIsUpperCase
 - character is uppercase service, 16-2
- CharToFoldedChar
 - character to fold service, 16-3
- CharToLowerCase
 - character to lower service, 16-3
- CharToUpperCase
 - character to upper service, 16-3
- close
 - a database file, 20-4
 - afile, 8-8
 - a locked or device segment, 2-5
 - a memory segment, 2-4
 - an I/O device, 8-8
- coldstart
 - getting the reason for, 19-2
- command line
 - getting, 19-6
- compare
 - two buffers, 17-1
 - two buffers case independent, 17-2
 - two floats, 14-1
 - two long integers, 13-1
 - two strings, 18-1
 - two strings case independent, 18-2
 - two unsigned long integers, 13-2
- compress
 - a database file, 20-5
- connect
 - to the file server, 9-1
- ConvArgumentsToBuffer
 - convert arguments to buffer service, 12-2
- conversion
 - arguments to buffer, 12-2
 - floating point number to buffer, 12-4
 - integer to buffer, 12-1
 - long integer to buffer, 12-2
 - string to floating point number, 12-5
 - string to integer, 12-3
 - string to long integer, 12-3
 - string to unsigned integer, 12-2
 - string to unsigned long integer, 12-2
 - unsigned integer to buffer, 12-1
 - unsigned long integer to buffer, 12-1
- convert
 - a string to folded, 18-1
 - float to signed integer, 14-3
 - float to signed long, 14-2
 - float to unsigned integer, 14-3
 - float to unsigned long, 14-2
 - signed integer to float, 14-3
 - signed long to float, 14-3
 - unsigned integer to float, 14-3
- ConvFloatToBuffer
 - convert floating point number to buffer service, 12-4
- ConvIntToBuffer
 - convert integer to buffer service, 12-1
- ConvLongIntToBuffer
 - convert long integer to buffer service, 12-2
- ConvStringToFloat
 - convert string to floating point number, 12-5
- ConvStringToInt
 - convert string to integer, 12-3
- ConvStringToLongInt
 - convert string to long integer, 12-3
- ConvStringToUnsignedInt
 - convert string to unsigned integer, 12-2
- ConvStringToUnsignedLongInt
 - convert string to unsigned long integer, 12-2
- ConvUnsignedIntToBuffer
 - convert unsigned integer to buffer service, 12-1
- ConvUnsignedLongIntToBuffer
 - convert unsigned long integer to buffer service, 12-1
- copy
 - a buffer, 17-1
 - a database file, 20-5
 - a string, 18-1
 - a string folded, 18-1
 - copying down a DBF record, 20-5
 - data from a process, 10-8
 - data to a process, 10-9
 - from a category, 6-7

- from a memory segment, 2-7
 - strings from a process, 10-9
 - to a memory segment, 2-6
- cosine
 - float function, 15-1
- count
 - the number of DBF records, 20-13
- country data
 - getting, 19-2
 - setting, 19-2
- CRC
 - generating, 19-12
- create
 - a memory segment, 2-3
 - an object by handle, 6-3
 - an object by number, 6-3
 - a process, 10-4
 - a semaphore, 4-1
 - a task, 10-4
- D\$X
 - environment variable, B-3
- data segment
 - of the operating system, 19-2
- Database file
 - appending a record, 20-10
 - closing, 20-4
 - compressing, 20-5
 - copying, 20-5
 - copying down a record, 20-5
 - counting the number of records, 20-13
 - Deleted records, 20-5
 - end of file, 20-2
 - erasing a record, 20-11
 - file buffering, 20-2
 - file structure, 20-1
 - finding a record, 20-12
 - finding a record by field, 20-13
 - flushing, 20-4
 - getting the size, 20-6
 - Getting the version number, 20-8
 - index table, 20-2
 - number of records, 20-3
 - opening, 20-3
 - reading an absolute record, 20-8
 - reading and sensing an absolute record, 20-9
 - reading the descriptive record, 20-7
 - reading the extended header, 20-7
 - reading the first record, 20-10
 - reading the last record, 20-10
 - reading the next record, 20-9
 - reading the previous record, 20-9
 - sensing the record number, 20-13
 - trashing the buffer, 20-4
 - updating a record, 20-11
 - writing the descriptive record, 20-8
 - Writing the extended header, 20-7
- date
 - abbreviated name of day, 11-5
 - abbreviated name of month, 11-5
 - convert date to day seconds, 11-3
 - convert day seconds to date, 11-3
 - convert day seconds to system time, 11-3
 - convert the system time to day seconds, 11-3
 - getting am and pm text, 19-11
 - getting suffixes, 19-11
 - getting the systemdate, 11-2
 - name of day, 11-4
 - name of month, 11-4
 - number of days in a month, 11-4
 - set file, 9-8
 - setting the system date, 11-2
 - weekday number, 11-4
- day
 - abbreviated name of, 11-5
 - name of, 11-4
- days
 - number of, 11-4
- DbfAbsRead
 - reading an absolute DBF record service, 20-8
- DbfAbsReadSense
 - reading and sensing an absolute DBF record service, 20-9
- DbfAppend
 - append a DBF record service, 20-10
- DbfBackRead
 - read the previous DBF record service, 20-9
- DbfClose
 - closing a database file service, 20-4
- DbfCompress
 - compressing a database file service, 20-5
- DbfCopyDown
 - copying down a DBF record service, 20-5
- DbfCopyFile
 - copying a database file service, 20-5
- DbfCount
 - count the number of DBF records service, 20-13
- DbfDescRecordRead
 - reading a DBF descriptive record service, 20-7
- DbfDescRecordWrite
 - writing a DBF descriptive record service, 20-8
- DbfEraseRead
 - erasing a DBF record service, 20-11
- DbfExtHeaderRead
 - reading a DBF extended header service, 20-7
- DbfExtHeaderWrite
 - writing a DBF extended header service, 20-7
- DbfFileSize
 - getting the size of a database file service, 20-6
- DbfFindRead
 - finding a DBF record service, 20-12
- DbfFindReadField
 - finding a DBF recordbyfield service, 20-13
- DbfFirstRead
 - read the first DBF record service, 20-10
- DbfFlush
 - flushing a database file service, 20-4
- DbfLastRead
 - read the last DBF record service, 20-10

- DbfNextRead
 - read the next DBF record service, 20-9
- DbfOpen
 - opening a database file service, 20-3
- DbfSense
 - sense the current DBF record number service, 20-13
- DbfTrash
 - trashing the DBF buffer service, 20-4
- DbfUpdate
 - updating a DBF record service, 20-11
- DbfVersion
 - getting the DBF version number service, 20-8
- deferred mode
 - setting, 19-4
- delete
 - a device driver, 7-3
 - a file or directory, 9-3
 - a memory segment, 2-4
 - a semaphore, 4-1
- destroy
 - an object, 6-4
- detach
 - a file system, 9-7
- DevDelete
 - delete a device driver service, 7-3
- DevFind
 - find all devices service, 7-4
- DevGetPDDAddress
 - get PDD entry point service, 7-2
- DevHold
 - hold all device drivers service, 7-2
- devices
 - calling a vector, 7-5
 - delete a device driver, 7-3
 - drivers, 7-1
 - find all devices, 7-4
 - getting the PDD entry point, 7-2
 - hold all device drivers, 7-2
 - install a device driver, 7-2
 - load a logical device driver, 7-3
 - load a physical device driver, 7-3
 - names, 7-1
 - open a physical device driver, 7-1
 - query the number of units, 7-4
 - remove a device driver, 7-4
 - resume all device drivers, 7-3
- devices and files
 - and I/O, 8-1
- DevInstall
 - install a device driver service, 7-2
- DevLoadLDD
 - load a logical device driver service, 7-3
- DevLoadPDD
 - load a physical device driver service, 7-3
- DevOpenPDD
 - open PDD service, 7-1
- DevQueryUnits
 - query the number of units service, 7-4
- DevRemove
 - remove a device driver service, 7-4
- DevResume
 - resume all device drivers service, 7-3
- DevVector
 - call a device vector, 7-5
- directory
 - changing, 9-7
 - deleting, 9-3
 - getting status, 9-4
 - making, 9-6
 - renaming, 9-4
 - setting status, 9-4
- display type
 - getting, 19-2
- divide
 - floats, 14-1
 - two long integers, 13-1
 - two unsigned long integers, 13-2
- dummy
 - call, 19-3
 - service, 19-3
- DYL
 - getting a handle, 6-3
- dynamic library
 - finding, 6-2
 - getting a handle, 6-3
 - linking, 6-2
 - loading, 6-1
 - loading multiple, 6-6
 - names, 6-1
 - unloading, 6-2
- EM\$
 - environment variable, B-1
- endoffile
 - database file, 20-2
- enter
 - a control region, 6-7
 - leaving from a control region, 6-7
- environment variable
 - \$WS_FL, B-1
 - \$WS_FNTS, B-2
 - \$WS_IF, B-2
 - \$WS_SD, B-2
 - \$WS_SF, B-3
 - \$WS_SF2, B-3
 - \$WS_SF4, B-3
 - C\$CALC, B-6
 - C\$P\$, B-9
 - C\$P@, B-9
 - C\$P£, B-9
 - C\$PA to C\$PZ, B-9
 - contents and names of all, B-1
 - D\$X, B-3
 - deleting buffer, 19-13
 - deleting string, 19-14
 - EM\$, B-1
 - F\$X, B-8
 - F\$XM, B-8
 - F\$XP, B-8
 - finding buffer, 19-13
 - finding string, 19-14
 - getting buffer, 19-12
 - getting string, 19-14
 - L\$X, B-3
 - M\$0M0, B-6
 - M\$1M1, B-6
 - M\$2M2, B-6

- M\$3M3, B-6
- M\$4M4, B-6
- M\$5M5, B-6
- M\$6M6, B-6
- M\$7M7, B-6
- M\$8M8, B-6
- M\$9M9, B-6
- M\$V, B-3
- MAIL\$ST, B-8
- names and contents of all, B-1
- P\$D, B-4
- P\$F, B-4
- P\$IP, B-5
- P\$M, B-5
- P\$P, B-5
- P\$PP, B-5
- P\$PX, B-5
- P\$S, B-4
- P\$SP, B-5
- P\$Z, B-5
- SS\$VER, B-9
- setting buffer, 19-12
- setting string, 19-14
- SP\$DRV, B-7
- SP\$OPT, B-7
- TW\$S, B-6
- W\$C, B-7
- W\$R, B-7
- WP\$SPEL, B-7
- WP\$THES, B-7
- Environment variable
 - Finding all, 19-13
- epocsibo.inc*
 - Include file, 21-10
- erase
 - aDBFrecord, 20-11
- error
 - notificationof, 19-5
- errors
 - gettingthetext, 19-3
- execute
 - animagefile, 9-1
- exits
 - watchingall, 10-8
- expansion port
 - sense state of, 21-12
- exponentiation
 - floatfunction, 15-2
- F\$X
 - environment variable, B-8
- F\$XM
 - environment variable, B-8
- F\$XP
 - environment variable, B-8
- FilChangeDirectory
 - change directory service, 9-7
- FilConnect
 - file server connect service, 9-1
- FilDelete
 - delete file or directory service, 9-3
- filebuffering
 - database file, 20-2
- filemanagement
 - attaching a file system, 9-6
 - change directory, 9-7
 - connect to the file server, 9-1
 - deleting, 9-3
 - detaching a file system, 9-7
 - execute a program file, 9-1
 - get current path, 9-2
 - get current path by ID, 9-7
 - getting device status, 9-5
 - getting status, 9-4
 - getting system status, 9-5
 - local file system changed, 9-8
 - making a new directory, 9-6
 - parse a filename, 9-2
 - read a local device directly, 9-9
 - read media information of a local device, 9-9
 - renaming, 9-4
 - set current path, 9-3
 - set file date, 9-8
 - set initial path, 9-8
 - setting status, 9-4
 - test path available, 9-3
- filename
 - generic parse, 19-3
- fileserver
 - process, 9-1
- filestructure
 - database file, 20-1
- FilExecute
 - execute image file service, 9-1
- FilLocChanged
 - report if the local file system has changed, 9-8
- FilLocDevice
 - read media information of a local device, 9-9
- FilLocReadPdd
 - read a local device directly, 9-9
- FilMakeDirectory
 - make a new directory service, 9-6
- FilOpenUnique
 - I/O open a unique filename service, 9-6
- FilParse
 - parse filename service, 9-2
- FilPathGet
 - get current path service, 9-2
- FilPathGetById
 - get current path by ID service, 9-7
- FilPathSet
 - set current path service, 9-3
- FilPathTest
 - test path available service, 9-3
- FilRename
 - rename a file or directory service, 9-4
- FilSetFileDate
 - set file date service, 9-8
- FilSetInitialPath
 - set initial path service, 9-8
- FilStatusDevice
 - get device status service, 9-5
- FilStatusGet
 - get file or directory status service, 9-4
- FilStatusSet
 - setfile or directory status service, 9-4

- FilStatusSystem
 - get file system status, 9-5
- FilSystemAttach
 - attach a file system service, 9-6
- FilSystemDetach
 - detach a file system service, 9-7
- find
 - a DBF record, 20-12
 - a DBF record by field, 20-13
 - a dynamic library, 6-2
 - all Devices, 7-4
 - all processes, 10-7
 - all segments, 2-6
- float
 - adding, 14-2
 - arc sine function, 15-1
 - arc tangent function, 15-1
 - comparing, 14-1
 - conversion to a buffer, 12-4
 - converting signed integer to float, 14-3
 - converting signed long to float, 14-3
 - converting to signed integer, 14-3
 - converting to signed long, 14-2
 - converting to unsigned integer, 14-3
 - converting to unsigned long, 14-2
 - converting unsigned integer to float, 14-3
 - cosine function, 15-1
 - dividing, 14-1
 - exponentiation function, 15-2
 - logarithm function, 15-2
 - modulo function, 15-3
 - multiplying, 14-1
 - natural logarithm function, 15-2
 - negating, 14-2
 - power function, 15-3
 - random number function, 15-3
 - sine function, 15-3
 - square root function, 15-4
 - subtracting, 14-2
 - tangent function, 15-4
 - to integer, 15-2
- FloatAdd
 - add floats service, 14-2
- FloatASin
 - arc sine service, 15-1
- FloatATan
 - arc tangent service, 15-1
- FloatCompare
 - compare floats service, 14-1
- FloatCos
 - cosine service, 15-1
- FloatDivide
 - divide floats service, 14-1
- FloatExp
 - exponentiation service, 15-2
- FloatInt
 - integer service, 15-2
- FloatLn
 - natural logarithm service, 15-2
- FloatLog
 - logarithm service, 15-2
- FloatMod
 - modulo service, 15-3
- FloatMultiply
 - multiply floats service, 14-1
- FloatNegate
 - negate floats service, 14-2
- FloatPow
 - power service, 15-3
- FloatRand
 - random number service, 15-3
- FloatSin
 - sine service, 15-3
- FloatSqrt
 - square root service, 15-4
- FloatSubtract
 - subtract floats service, 14-2
- FloatTangent
 - tangent service, 15-4
- FloatToInt
 - convert float to signed integer service, 14-3
- FloatToLong
 - convert float to long service, 14-2
- FloatToUnsignedInt
 - convert float to unsigned integer service, 14-3
- FloatToUnsignedLong
 - convert float to long service, 14-2
- flush
 - a database file, 20-4
- free
 - a heap memory cell, 3-3
 - a message, 5-4
- GenAlarmHook
 - hook the alarm interface, 19-15
- GenAlarmId
 - get the pid of the alarm server, 19-15
- GenCrc
 - generate a CRC check, 19-12
- GenDataSegment
 - operating system data segment, 19-2
- GenDeferredMode
 - set deferred mode, 19-4
- GenEnvBufferDelete
 - delete environment variable, 19-13
- GenEnvBufferFind
 - find environment variable, 19-13
- GenEnvBufferGet
 - get environment variable, 19-12
- GenEnvBufferSet
 - set environment variable, 19-12
- GenEnvStringDelete
 - delete environment variable, 19-14
- GenEnvStringFind
 - find environment variable, 19-14
- GenEnvStringGet
 - get environment variable, 19-14
- GenEnvStringSet
 - set environment variable, 19-14
- GenGetAmPmText
 - get the am and pm text, 19-11
- GenGetAutoMains
 - get state for auto-switch-off if mains present, 19-16
- GenGetAutoSwitchOffValue
 - get the auto switch off value, 19-9

- GenGetBatteryType
 - get the battery type, 19-11
- GenGetCommandLine
 - get the command line, 19-6
- GenGetCountryData
 - get country dependent data, 19-2
- GenGetErrorText
 - get error text, 19-3
- GenGetLanguageCode
 - get the language code, 19-10
- GenGetNotifyState
 - get notify state, 19-8
- GenGetOsData
 - get O/S data, 19-3
- GenGetRamSizeInParas
 - get address able system RAM size, 19-6
- GenGetSoundFlags
 - get the sound flags, 19-7
- GenGetSuffixes
 - get suffix text, 19-11
- GenGetText
 - get operating system text, 19-8
- GenIntByNumber
 - interrupt by number, 19-12
- GenLcdType
 - LCD type, 19-2
- GenMarkActive
 - mark process as active, 19-7
- GenMarkNonActive
 - mark process as non-active, 19-8
- GenNotify
 - notification service, 19-4
- GenNotifyError
 - notification of error, 19-5
- GenNotifyHook
 - hook the notifier interface, 19-5
- GenParse
 - generic parse, 19-3
- GenResetRevector
 - release an interrupt, 19-10
- GenRomVersion
 - get the ROM version, 19-1
- GenSetAutoMains
 - disable/enableauto-switch-off if mains present, 19-16
- GenSetAutoSwitchOffValue
 - set the auto switchoff time, 19-9
- GenSetBatteryType
 - set the battery type, 19-11
- GenSetCountryData
 - set country dependent data, 19-2
- GenSetNotifyState
 - set notify state, 19-9
- GenSetOnEvents
 - enable/disable on events, 19-16
- GenSetRevector
 - capture an interrupt, 19-9
- GenSetSoundFlags
 - set the sound flags, 19-7
- GenSound
 - make a sound with the piezo, 19-7
- GenStartReason
 - getting the system cold start reason, 19-2
- GenTickle
 - reset the autoswitch off timer, 19-15
- GenUnAlarmHook
 - unhook the alarm interface, 19-15
- GenUnNotifyHook
 - unhook the notifier interface, 19-6
- GenVersion
 - operating system version number, 19-1
- granularity
 - of the heap memory, 3-3
- halfseconds
 - query completion, 8-12
 - signal on next half second, 8-11
- handle
 - of a DYL, 6-3
 - of a dynamic library, 6-3
- handler
 - adding, 8-6
 - adding an application, 8-9
 - enabling, 8-6
 - enabling an application, 8-10
 - removing, 8-6
 - removing an application, 8-10
- hardware
 - capturing the combo subsystem, 21-5
 - changing the LCD contrast, 21-6
 - clearing bits in Asic2 register1, 21-2
 - clearing bits in Asic2 register2, 21-3
 - clearing bits in Asic2 register3, 21-3
 - enable/disable reset, 21-10
 - exiting toDOS, 21-5
 - expansion port sense state of, 21-12
 - freeing a channel, 21-6
 - freeing the combo subsystem, 21-5
 - get additional power supply data, 21-10
 - getting achannel, 21-5
 - getting backlight control, 21-7
 - getting current LCD contrast, 21-7
 - getting supplies status, 21-6
 - getting supplies warnings, 21-6
 - getting the power supply type, 21-6
 - Honda connector power disable, 21-12
 - Honda connector power enable, 21-12
 - infrared power level set, 21-11
 - operating the backlight, 21-7
 - reading Asic2 register2, 21-3
 - reading Asic2 register3, 21-4
 - reset the battery status, 21-10
 - return battery information, 21-11
 - scan state of all keys, 21-8
 - select serial channel, 21-4
 - sending a serial null frame, 21-4
 - setting backlight control, 21-7
 - setting bits in Asic2 register1, 21-2
 - setting bits in Asic2 register2, 21-2
 - setting bits in Asic2 register3, 21-3
 - SSDs relog, 21-11
 - switching off, 21-4
 - switching off the combo, 21-1
 - switching off the SSDs, 21-1
 - switching on the combo, 21-1
 - switching on the combo in input mode, 21-10
 - switching on the SSDs, 21-1

- tick count sense current, 21-11
- writing Asic2 register1, 21-2
- writing Asic2 register2, 21-3
- writing Asic2 register3, 21-4
- Hardware
 - Reading Asic2 register1, 21-2
- HeapAdjustCellSize
 - adjust heap cellsize service, 3-2
- HeapAllocateCell
 - allocate heap cell service, 3-1
- HeapCellSize
 - size of heap memory cell, 3-3
- HeapFreeCell
 - free a heap cell service, 3-3
- HeapFreeMemory
 - size of available heap memory, 3-3
- heap memory
 - dynamics, 3-1
- HeapReAllocateCell
 - re-allocate heap cell service, 3-2
- HeapSetGranularity
 - set heap grow by parameter service, 3-3
- Honda connector
 - power disable, 21-12
 - power enable, 21-12
- HwBackLight
 - operating the backlight, 21-7
- HwClearA2Control1Bits
 - clearing bits in Asic2 register 1, 21-2
- HwClearA2Control2Bits
 - clearing bits in Asic2 register 2, 21-3
- HwClearA2Control3Bits
 - clearing bits in Asic2 register 3, 21-3
- HwComboOff
 - switch off the combo, 21-1
- HwComboOn
 - switch on the combo, 21-1
- HwComboOnInput
 - switch on the combo in input mode, 21-10
- HwEnableAutoBatReset
 - enable/disable battery reset on recharge, 21-10
- HwExit
 - exit the program, 21-5
- HwExpansionOff
 - Honda connector power disable, 21-12
- HwExpansionOn
 - Honda connector power enable, 21-12
- HwFreeChannel
 - free a channel, 21-6
- HwFreeCombo
 - free the combo, 21-5
- HwGetBackLight
 - get backlight control, 21-7
- HwGetBatData
 - return battery information, 21-11
- HwGetChannel
 - get a channel, 21-5
- HwGetCombo
 - capture the combo, 21-5
- HwGetPsuType
 - get power supply type, 21-6
- HwGetScanCodes
 - scan the state of all keys, 21-8
- HwGetSupplyStatus
 - get supplies status, 21-6
- HwLcdContrastDelta
 - change the LCD contrast, 21-6
- HwNullFrame
 - send a serial null frame, 21-4
- HwPacksOff
 - switch off the SSDs, 21-1
- HwPacksOn
 - switch on the SSDs, 21-1
- HwReadA2Control1
 - read Asic2 register 1, 21-2
- HwReadA2Control2
 - read Asic2 register 2, 21-3
- HwReadA2Control3
 - read Asic2 register 3, 21-4
- HwReadLcdContrast
 - read current contrast, 21-7
- HwReLogPacks
 - relog the SSDs, 21-11
- HwResetBatteryStatus
 - reset the battery status, 21-10
- HwReturnExpansionPortState
 - expansion port sense state of, 21-12
- HwReturnTickCount
 - tick count - sense current, 21-11
- HwSelectChannel
 - select serial channel, 21-4
- HwSetA2Control1Bits
 - setting bits in Asic2 register 1, 21-2
- HwSetA2Control2Bits
 - setting bits in Asic2 register 2, 21-2
- HwSetA2Control3Bits
 - setting bits in Asic2 register 3, 21-3
- HwSetBackLight
 - set backlight control, 21-7
- HwSetIRPowerLevel
 - Set the infrared power level, 21-11
- HwSupplyInfo
 - get additional power supply data, 21-10
- HwSupplyWarnings
 - get supplies warnings, 21-6
- HwSwitchOff
 - switch off service, 21-4
- HwWriteA2Control1
 - write Asic2 register 1, 21-2
- HwWriteA2Control2
 - write Asic2 register 2, 21-3
- HwWriteA2Control3
 - write Asic2 register 3, 21-4
- I/O
 - adding a handler, 8-6
 - adding an application handler, 8-9
 - a synchronous, 8-2
 - a synchronous without error reporting, 8-2
 - cancel playing back a sound file, 8-13
 - cancel recording sound to a file, 8-14
 - cancel requested reset, 8-7
 - cancel request for a signal from the supervisor, 8-11
 - chain to root device, 8-3
 - chain to super class device, 8-4
 - closing a device, 8-8
 - enabling a handler, 8-6

- enabling an application handler, 8-10
- getting the shift states, 8-10
- keyboard and mouse, 8-9
- opening a device, 8-7
- opening a unique filename, 9-6
- play back a sound file asynchronously, 8-12
- play back a sound file synchronously, 8-12
- polling for completion, 8-5
- query the completion of `IoNextHalfSecond`, 8-12
- reading from a device, 8-8
- record sound to a file synchronously, 8-13
- record sound to file asynchronously, 8-14
- removing a handler, 8-6
- removing an application handler, 8-10
- request a signal the supervisor, 8-11
- requesting reset, 8-7
- request signal on next half second, 8-11
- seeking on a device, 8-8
- signalling completion, 8-5
- signalling completion by pid with no re-schedule, 8-5
- signalling completion by process ID, 8-5
- synchronous, 8-3
- wait for completion, 8-4
- wait for completion no handlers, 8-10
- wait for specific completion, 8-4
- writing to a device, 8-8
- I/O system
 - messaging, 5-2
- image
 - opening to access multiple libraries, 6-5
- include file
 - `epocdefs.inc`, 1-3
- index table
 - database file, 20-2
- infrared
 - set power level, 21-11
- initialize
 - the message system, 5-2
- install
 - a device driver, 7-2
- int
 - by number, 19-12
- integer
 - comparing longs, 13-1
 - comparing unsigned longs, 13-2
 - conversion to a buffer, 12-1
 - divide longs, 13-1
 - divide unsigned longs, 13-2
 - multiply longs, 13-1
 - multiply unsigned longs, 13-2
 - of a float, 15-2
 - unsigned conversion to a buffer, 12-1
 - unsigned long random number, 13-3
- inter process communications
 - messaging, 5-1
- interrupt
 - calling conventions, 1-1
 - capturing, 19-9
 - multi service, 1-1
 - releasing, 19-10
 - single service, 1-1
- interrupts
 - alphabetic listing, A-2
 - alphabetic listing - extra functions, A-27
 - function numbers, A-1
 - function numbers - extra functions, A-27
 - numerical listing, A-14
 - numerical listing - extra functions, A-27
 - using single or multi, A-1
- `IoAddHandler`
 - I/O add handler service, 8-6
- `IoApplicationAddHandler`
 - I/O add handler service, 8-9
- `IoAsynchronous`
 - I/O asynchronous service, 8-2
- `IoAsynchronousNoError`
 - I/O asynchronous without error reporting service, 8-2
- `IoClose`
 - I/O close a device service, 8-8
- `IoEnableApplicationHandler`
 - I/O enable/disable application handler service, 8-10
- `IoEnableHandler`
 - I/O enable/disable handler, 8-6
- `IoKeyAndMouseWithWait`
 - get keyboard and mouse events service, 8-9
- `IoNextHalfSecond`
 - request completion on the next half second, 8-11
- `IoNextHalfSecondStatus`
 - query the completion of Io Next Half Second, 8-12
- `IoOpen`
 - I/O open a device service, 8-7
- `IoPlaySoundA`
 - play back a sound file asynchronously, 8-12
- `IoPlaySoundCancel`
 - cancel playing back a sound file, 8-13
- `IoPlaySoundW`
 - play back a sound file synchronously, 8-12
- `IoRead`
 - I/O read from a device service, 8-8
- `IoRecordSoundA`
 - record sound to a file asynchronously, 8-14
- `IoRecordSoundCancel`
 - cancel recording sound to a file, 8-14
- `IoRecordSoundW`
 - record sound to a file synchronously, 8-13
- `IoRemoveApplicationHandler`
 - I/O remove application handler service, 8-10
- `IoRequestReset`
 - I/O request reset service, 8-7
- `IoRequestResetCancel`
 - I/O cancel requested reset service, 8-7
- `IoRoot`
 - I/O chain to root device service, 8-3
- `IoSeek`
 - I/O seek to a new position, 8-8
- `IoShiftStates`
 - I/O get shift states service, 8-10
- `IoSignal`
 - I/O signal completion service, 8-5

- IoSignalByPid
 - I/O signal completion by process ID service, 8-5
- IoSignalByPidNoReSched
 - I/O signal completion by pid with no reschedule service, 8-5
- IoSignalKillAsynchronous
 - request signal from supervisor service, 8-11
- IoSignalKillCancel
 - cancel signal kill from supervisor service, 8-11
- IoSuper
 - I/O chain to super class device service, 8-4
- IoWaitForSignal
 - I/O wait for completion service, 8-4
- IoWaitForSignalNoHandler
 - I/O wait for completion with no handlers service, 8-10
- IoWaitForStatus
 - I/O wait for specific request to complete service, 8-4
- IoWithWait
 - I/O with wait service, 8-3
- IoWrite
 - I/O write to a device service, 8-8
- IoYield
 - I/O update status words service, 8-5
- justify
 - a buffer, 17-4
- keyboard
 - reading, 8-9
 - scanning state of all keys, 21-8
- Keyboard
 - scan codes HC alphabetic, 21-8
 - scan codes HC numeric, 21-9
 - scan codes Series 3a, 21-8
 - scan codes Workabout, 21-9
- kill
 - aprocess, 10-6
- L\$X
 - environment variable, B-3
- languagecode
 - getting, 19-10
- LCD
 - getting the type, 19-2
- length
 - of a string, 18-4
- LibCopy
 - copy data from a categories segment, 6-7
- LibCreate
 - creating an object by number service, 6-3
- LibCreateByHandle
 - creating an object by handle service, 6-3
- LibDestroy
 - destroying an object service, 6-4
- LibEnter
 - enter a control region, 6-7
- LibEnterSend
 - send message with an enclosing Lib Enter, 6-5
- LibExactSend
 - send message to a known class, 6-5
- LibFind
 - dynamic library find service, 6-2
- LibHandle
 - dynamic library get handle service, 6-3
- LibLeave
 - exit from a control region, 6-7
- LibLink
 - dynamic library link service, 6-2
- LibLoad
 - dynamic library load service, 6-1
- LibLoadFile
 - dynamic library load from multiple library file, 6-6
- LibOpen
 - open an image file containing multiple libraries, 6-5
- library
 - opening in an image, 6-5
- library names
 - dynamic, 6-1
- LibReClass()
 - reclassing an object by number, 6-6
- LibReClassByHandle
 - reclassing an object by handle, 6-7
- LibSend
 - send message to an object service, 6-4
- LibSendExit
 - exit from a method, 6-8
- LibSuperSend
 - send message to the objects superclass service, 6-4
- LibUnLoad
 - dynamic library unload service, 6-2
- link
 - a dynamic library, 6-2
- load
 - a dynamic library, 6-1
 - a logical device driver, 7-3
 - a multiple dynamic library, 6-6
 - a physical device driver, 7-3
- locate
 - a character in a buffer, 17-2
 - a character in a buffer folded, 17-2
 - a character in a string, 18-3
 - a character in a string folded, 18-3
 - a character in a string in reverse, 18-3
 - a character in a string in reverse folded, 18-3
- lock
 - a memory segment, 2-5
- logarithm
 - float function, 15-2
- LongIntCompare
 - compare long integers service, 13-1
- LongIntDivide
 - long integer divide service, 13-1
- longinteger
 - conversion to a buffer, 12-2
 - unsigned conversion to a buffer, 12-1
- LongIntMultiply
 - long integer multiply service, 13-1
- LongToFloat
 - convert signed long to float service, 14-3
- LongUnsignedIntCompare
 - compare unsigned long integers service, 13-2

- LongUnsignedIntDivide
 - unsigned long integer divide service, 13-2
- LongUnsignedIntMultiply
 - long unsigned integer multiply service, 13-2
- LongUnsignedIntRandom
 - unsigned long integer random number service, 13-3
- M\$0M0
 - environment variable, B-6
- M\$1M1
 - environment variable, B-6
- M\$2M2
 - environment variable, B-6
- M\$3M3
 - environment variable, B-6
- M\$4M4
 - environment variable, B-6
- M\$5M5
 - environment variable, B-6
- M\$6M6
 - environment variable, B-6
- M\$7M7
 - environment variable, B-6
- M\$8M8
 - environment variable, B-6
- M\$9M9
 - environment variable, B-6
- M\$V
 - environment variable, B-3
- MAIL\$ST
 - environment variable, B-8
- mark
 - resetting the auto switch off timer, 19-15
- match
 - a wildcard buffer, 17-3
 - a wildcard buffer folded, 17-4
 - a wildcard string, 18-2
 - a wildcard string folded, 18-2
- media
 - read a local device directly, 9-9
 - read information of a local device, 9-9
- memory
 - adjust heap memory size, 3-2
 - adjust the size of a memory segment, 2-6
 - allocate heap memory, 3-1
 - close a memory segment, 2-4
 - copy from a memory segment, 2-7
 - copy to a memory segment, 2-6
 - create a memory segment, 2-3
 - delete a memory segment, 2-4
 - find all segments, 2-6
 - free heap memory, 3-3
 - heap memory dynamics, 3-1
 - lock a memory segment, 2-5
 - open a memory segment, 2-4
 - paragraphs size of, 2-1
 - re-allocate heap memory, 3-2
 - segment directly accessing, 2-1
 - segment locking, 2-1
 - segment names, 2-1
 - setting the heap granularity, 3-3
 - size of available heap memory, 3-3
 - size of available segmented memory, 2-2
 - size of addressable system ram, 19-6
 - size of a heap cell, 3-3
 - size of a memory segment, 2-5
 - size of RAM disk, 2-7
 - unlock a memory segment, 2-5
- message
 - enter send, 6-5
 - sending to a known class, 6-5
 - sending to an object, 6-4
 - sending to an objects superclass, 6-4
- message reception
 - order of, 5-1
- message system
 - I/O system, 5-2
- messages
 - asynchronous reception, 5-2
 - cancelling receive, 5-3
 - cancel request for a signal from the supervisor, 5-5
 - cancel request for a signal from the supervisor by type, 5-6
 - freeing, 5-4
 - initializing, 5-2
 - request a signal the supervisor, 5-5
 - sending, 5-3
 - sending and getting a reply asynchronously, 5-4
 - sending and waiting for a reply, 5-4
 - synchronous reception, 5-3
- messaging
 - inter process communication, 5-1
- MessFree
 - free message service, 5-4
- MessInit
 - initialize messages service, 5-2
- MessReceiveAsynchronous
 - receive message asynchronously, 5-2
- MessReceiveCancel
 - cancel queued message receive service, 5-3
- MessReceiveWithWait
 - synchronous message reception, 5-3
- MessSend
 - send message service, 5-3
- MessSendReceiveAsynchronous
 - send message and get reply asynchronously service, 5-4
- MessSendReceiveWithWait
 - send message and wait for reply service, 5-4
- MessSignal
 - request signal from supervisor service, 5-5
- MessSignalCancelX
 - cancel requested signal from Supervisor by type service, 5-6
- method
 - returning from a method, 6-8, 7-1
- modulo
 - float function, 15-3
- month
 - abbreviated name of, 11-5
 - name of, 11-4
 - number of days, 11-4
- mouse
 - reading, 8-9

- multiply
 - two floats, 14-1
 - two long integers, 13-1
 - two unsigned long integers, 13-2
- name
 - validation, 18-5
- names
 - device, 7-1
 - memory segments, 2-1
 - of processes by ID, 10-7
- naturallogarithm
 - float function, 15-2
- negate
 - floats, 14-2
- notify
 - by error number, 19-5
 - by text messages, 19-4
 - getting state, 19-8
 - hooking the interface, 19-5
 - setting state, 19-9
 - unhooking the interface, 19-6
- number
 - getting suffixes text, 19-11
 - of week, 11-5
- number of records
 - database file, 20-3
- object
 - creating by handle, 6-3
 - creating by number, 6-3
 - destroying, 6-4
 - enter a sent message, 6-5
 - reclassing by handle, 6-7
 - reclassing by number, 6-6
 - sending a message, 6-4
 - sending a message to a known class, 6-5
 - sending a super class message, 6-4
- on events
 - receiving, 19-16
- open
 - a database file, 20-3
 - afile, 8-7
 - a memory segment, 2-4
 - a multi library file, 6-5
 - an I/O device, 8-7
 - a physical device driver, 7-1
 - a unique filename, 9-6
- operating system
 - data segment getting, 19-2
- operating system
 - getting the data, 19-3
- operating system text
 - getting, 19-8
- owner
 - getting, 10-3
- P\$D
 - environment variable, B-4
- P\$F
 - environment variable, B-4
- P\$IP
 - environment variable, B-5
- P\$M
 - environment variable, B-5
- P\$P
 - environment variable, B-5
- P\$PP
 - environment variable, B-5
- P\$PX
 - environment variable, B-5
- P\$S
 - environment variable, B-4
- P\$SP
 - environment variable, B-5
- P\$Z
 - environment variable, B-5
- panic
 - a process, 10-7
 - the current process, 10-8
- paragraphs
 - size of memory segments, 2-1
- parse
 - a filename, 9-2
 - generic filename, 19-3
- path
 - get current, 9-2
 - get current by ID, 9-7
 - set current, 9-3
 - set initial, 9-8
 - test available, 9-3
- PDD
 - physical device driver, 7-1
- piezo
 - sound, 19-7
- pm
 - getting the pm text, 19-11
- polling
 - I/O status words, 8-5
- power
 - float function, 15-3
- power supply
 - getting additional data, 21-10
- priority
 - getting, 10-3
 - setting, 10-3
- ProcCopyFromById
 - copy data from a process service, 10-8
- ProcCopyToById
 - copy data to a process service, 10-9
- ProcCreate
 - create process service, 10-4
- ProcCreateTask
 - create task service, 10-4
- processes
 - controlling, 10-2
 - copying data from by ID, 10-8
 - copying data to by ID, 10-9
 - copying strings from by ID, 10-9
 - creating, 10-4
 - find all, 10-7
 - get an ID by name, 10-3
 - get name by ID, 10-7
 - get owner, 10-3
 - get priority, 10-3
 - get the current process ID, 10-2
 - ID and process table, 10-2
 - IDs and names, 10-1
 - killing, 10-6
 - panicking, 10-7
 - panicking current, 10-8

- renaming, 10-7
- resuming, 10-5
- scheduling, 10-1
- setpriority, 10-3
- suspending, 10-5
- terminate and kill, 10-2
- terminating, 10-6
- termination registration, 10-6
- watching all exits, 10-8
- ProcFind
 - find all processes service, 10-7
- ProcGetOwner
 - get the PID of the owning process service, 10-3
- ProcGetPriority
 - get process priority service, 10-3
- ProcId
 - get current process ID service, 10-2
- ProcIdByName
 - get process ID by name service, 10-3
- ProcIndStringCopyFromById
 - copy a string from a process service, 10-9
- ProcKill
 - kill process service, 10-6
- ProcNameById
 - name of a process by ID service, 10-7
- ProcOnTerminate
 - register termination service, 10-6
- ProcPanic
 - panic current process service, 10-8
- ProcPanicById
 - panic process service, 10-7
- ProcRename
 - rename a process service, 10-7
- ProcResume
 - resume process service, 10-5
- ProcSetPriority
 - set process priority service, 10-3
- ProcSuspend
 - suspend process service, 10-5
- ProcTerminate
 - terminate process service, 10-6
- ProcWatchAllExits
 - monitor exits service, 10-8
- query
 - the number of units, 7-4
- RAM disk
 - return size of, 2-7
- random
 - float function, 15-3
 - unsigned long integer, 13-3
- read
 - a DBF descriptive record, 20-7
 - a DBF extended header, 20-7
 - an absolute DBF record, 20-8
 - from a file, 8-8
 - from an I/O device, 8-8
 - the first DBF record, 20-10
 - the last DBF record, 20-10
 - the next DBF record, 20-9
 - the previous DBF record, 20-9
- reclass
 - an object by handle, 6-7
 - an object by number, 6-6
- remove
 - a device driver, 7-4
- rename
 - a file or directory, 9-4
 - a process, 10-7
- reset
 - I/O cancel request, 8-7
 - I/O Request, 8-7
- reset system
 - getting the reason for, 19-2
- resume
 - a process, 10-5
- re-vectors
 - capturing, 19-9
 - releasing, 19-10
- \$\$SVER
 - environment variable, B-9
- Scan codes
 - HC alphabetic, 21-8
 - HC numeric, 21-9
 - Series 3a, 21-8
 - Workabout, 21-9
- seek
 - a file to a new position, 8-8
- SegAdjustSize
 - adjust the size of a memory segment, 2-6
- SegClose
 - close memory segment service, 2-4
- SegCloseLockedOrDevice
 - close a locked or device segment, 2-5
- SegCopyFrom
 - copy from memory segment service, 2-7
- SegCopyTo
 - copyto memory segment service, 2-6
- SegCreate
 - create memory segment service, 2-3
- SegDelete
 - delete memory segment service, 2-4
- SegFind
 - find all segments service, 2-6
- SegFreeMemory
 - size of available segmented memory, 2-2
- SegLock
 - lock memory segment service, 2-5
- segment
 - change size of a memory segment, 2-6
 - close a memory segment, 2-4
 - close locked or device, 2-5
 - copy from a memory segment, 2-7
 - copy to a memory segment, 2-6
 - create a memory segment, 2-3
 - delete a memory segment, 2-4
 - directly accessing, 2-1
 - find all segments, 2-6
 - lock a memory segment, 2-5
 - locking, 2-1
 - names, 2-1
 - size of available segmented memory, 2-2
 - size of a memory segment, 2-5
 - unlock a memory segment, 2-5
- SegOpen
 - open memory segment service, 2-4
- SegRamDiskUsed
 - size of RAM disk service, 2-7

- SegSize
 - size of memory segment service, 2-5
- SegUnLock
 - unlock memory segment service, 2-5
- semaphores
 - creating, 4-1
 - deleting, 4-1
 - signalling once without re-schedule, 4-2
 - signalling more than once, 4-2
 - signalling once, 4-2
 - waiting, 4-1
- SemCreate
 - create semaphore service, 4-1
- SemDelete
 - delete semaphore, 4-1
- SemSignal
 - signal once service, 4-2
- SemSignalMany
 - signal many service, 4-2
- SemSignalOnceNoResched
 - signal once with no re-schedule, 4-2
- SemWait
 - wait on semaphore service, 4-1
- send
 - a message, 5-3
 - and get reply asynchronously, 5-4
 - and wait for reply, 5-4
- sense
 - an absolute DBF record, 20-9
 - the current DBF record number, 20-13
- shiftstates
 - Getting, 8-10
- signal
 - a semaphore once without re-schedule, 4-2
 - a semaphore more than once, 4-2
 - a semaphore once, 4-2
 - from the supervisor, 5-5
 - from the supervisor I/O , 8-11
 - I/O completion, 8-5
 - I/O completion by pid with no re-schedule, 8-5
 - I/O completion by process ID, 8-5
- SignedIntToFloat
 - convert signed integer to float service, 14-3
- sine
 - float function, 15-3
- size
 - a database file, 20-6
 - of addressable systemRAM, 19-6
 - of a memory segment, 2-5
 - of a string, 18-4
 - of systemram, 19-6
- sleep
 - a process in system clock ticks, 11-2
 - a process in tenths of a second, 11-2
 - a process till a given time, 11-1
- sound
 - cancel playing back file, 8-13
 - cancel recording to file, 8-14
 - getting the flags, 19-7
 - play back file (partial) asynchronously, 8-15
 - play back file asynchronously, 8-12
 - play back file synchronously, 8-12
 - record to file asynchronously, 8-14
 - record to file synchronously, 8-13
 - setting the flags, 19-7
 - using the piezo, 19-7
- Sound
 - Pitch calculating, 19-7
- Sound file names
 - Series 3a ROM, 8-12
- sound files
 - format .wve files, 8-1
- SP\$DRV
 - environment variable, B-7
- SP\$OPT
 - environment variable, B-7
- square root
 - float function, 15-4
- SSD
 - relog, 21-11
- status
 - of a device, 9-5
 - of a file or directory, 9-4
 - of a file system, 9-5
- string
 - capitalising, 18-1
 - comparing, 18-1
 - comparing folded, 18-2
 - conversion to float, 12-5
 - conversion to folded, 18-1
 - conversion to integer, 12-3
 - conversion to long integer, 12-3
 - conversion to unsigned integer, 12-2
 - conversion to unsigned long integer, 12-2
 - copying, 18-1
 - copying folded, 18-1
 - length, 18-4
 - locating, 18-3
 - locating folded, 18-3
 - locating in reverse, 18-3
 - locating in reverse folded, 18-3
 - substring, 18-4
 - substring folded, 18-4
 - validate, 18-5
 - wildcard match, 18-2
 - wildcard match folded, 18-2
- StringCapitalise
 - convert a string to have the first letter uppercase and the rest lowercase service, 18-1
- StringCompare
 - comparestrings service, 18-1
- StringCompareFolded
 - compare strings folded service, 18-2
- StringConvertToFolded
 - convert string to folded service, 18-1
- StringCopy
 - copy string service, 18-1
- StringCopyFolded
 - copy string folded service, 18-1
- StringLength
 - length of string service, 18-4
- StringLocate
 - locate a character in string service, 18-3
- StringLocateFolded
 - locate a character in string folded service, 18-3

-
- StringLocateInReverse
 - locate a character in a string in reverse service, 18-3
 - StringLocateInReverseFolded
 - locate a character in string in reverse folded service, 18-3
 - StringMatch
 - match a wild card string service, 18-2
 - StringMatchFolded
 - match a wild card string folded service, 18-2
 - StringSubString
 - find a substring in a string service, 18-4
 - StringSubStringFolded
 - find a substring in a string folded service, 18-4
 - StringValidateName
 - validate a system name, 18-5
 - structures
 - SupplyInfoEnt, 21-11
 - sub-buffer
 - in a buffer, 17-3
 - in a buffer folded, 17-3
 - substring
 - in a string, 18-4
 - in a string folded, 18-4
 - subtract
 - floats, 14-2
 - suffix
 - getting text, 19-11
 - SupplyInfoEnt
 - Data structure, 21-10
 - structure, 21-11
 - suspend
 - a process, 10-5
 - swap
 - two buffers, 17-1
 - switching off
 - disable/enable if mains present, 19-16
 - get state if mains present, 19-16
 - switching on
 - reporting, 19-16
 - synchronous
 - I/O , 8-3
 - message reception, 5-3
 - tangent
 - float function, 15-4
 - tasks
 - creating, 10-4
 - terminate
 - a process, 10-6
 - termination
 - registration, 10-6
 - text
 - getting operating system, 19-8
 - tick count
 - sense current, 21-11
 - tickle
 - resetting the autoswitch off timer, 19-15
 - TimDateToDaySeconds
 - convert date to day seconds service, 11-3
 - TimDayOfWeek
 - day of week service, 11-4
 - TimDaySecondsToDate
 - convert day seconds to date service, 11-3
 - TimDaySecondsToSystemTime
 - convert day seconds to system time service, 11-3
 - TimDaysInMonth
 - days in month service, 11-4
 - time
 - convert date to day seconds, 11-3
 - convert day seconds to date, 11-3
 - convert day seconds to system time, 11-3
 - convert the system time to day seconds, 11-3
 - getting the system time, 11-2
 - setting the system time, 11-2
 - sleeping for system clock ticks, 11-2
 - sleeping for tenths of a second, 11-2
 - waiting till a given time, 11-1
 - times
 - absolute and relative, 11-1
 - TimGetSystemTime
 - get the system time service, 11-2
 - TimNameOfDay
 - name of day service, 11-4
 - TimNameOfDayAbb
 - abbreviated name of day service, 11-5
 - TimNameOfMonth
 - name of month service, 11-4
 - TimNameOfMonthAbb
 - abbreviated name of month service, 11-5
 - TimSetSystemTime
 - set the system time service, 11-2
 - TimSleepForTenths
 - sleep for tenths of a second service, 11-2
 - TimSleepForTicks
 - sleep for system clock ticks service, 11-2
 - TimSystemTimeToDaySeconds
 - convert the system time to day seconds service, 11-3
 - TimWaitAbsolute
 - wait till a given time service, 11-1
 - TimWeekNumber
 - week number service, 11-5
 - trash
 - a DBF buffer, 20-4
 - TW\$\$
 - environment variable, B-6
 - unload
 - a dynamic library, 6-2
 - unlock
 - a memory segment, 2-5
 - UnsignedIntToFloat
 - convert unsigned integer to float service, 14-3
 - update
 - a DBF record, 20-11
 - validate
 - a string, 18-5
 - vectors
 - calling, 7-5
 - version
 - of the ROM, 19-1
 - operating system, 19-1
 - the DBF version number, 20-8
-

W\$C

environment variable, B-7

W\$R

environment variable, B-7

wait

an I/O completion, 8-4

an I/O completion no handlers, 8-10

a process till a given time, 11-1

a specific I/O completion, 8-4

on a semaphore, 4-1

watch

all exits, 10-8

week

number, 11-5

wildcard

buffer match, 17-3

buffer match folded, 17-4

string match, 18-2

string match folded, 18-2

WP\$SPEL

environment variable, B-7

WP\$THES

environment variable, B-7

write

a DBF descriptive record, 20-8

a DBF extended header, 20-7

to a file, 8-8

to an I/O device, 8-8

WVE sound files

format, 8-1