Introduction

This document describes Domesdos V0.07. No understandings are given or implied that this software will form any part of any computer system to be manufactured by or on behalf of Sinclair Research Limited.

The description of the system organisation, including the memory structure and system entry points is intended to provide the information required to write programs which operate in conjunction with Domesdos and which use the program control and input and output facilities of Domesdos.

Note that Domesdos is not a program, nor do applications programs or other software run 'under' Domesdos. Domesdos is an assembly of procedures which may be invoked by programs to perform operations commonly required. This assembly of procedures may be expanded or modified as required.

In addition to Domesdos there are a number of utility programs; these programs are not essential to the operation of the machine, and have no influence on the way in which application programs run on the machine. The most important of these utility programs is the Basic Command Interpreter. This is a resident command processor which will always have some data stored in the ZX83 memory. Special provisions are made in Domesdos for this permanent data store.

Changes From V0.06

There are three changes from Version 0.06. The first is the rationalisation of the error reporting mechanism, the second is the extensions to and reorganisation of the screen handling, and the last is the slight clarification of the microdrive file handling.

Changes From V0.05

The changes from Version 0.05 are mostly in the internal organisation. These reflect the change in emphasis from a small machine to a 128 Kbyte minimum configuration, a high probability that a 512 Kbyte expansion will be available at launch, a hard disk will be available soon after, and that the machine is a flagship of a new line.

For this reason the resource allocations have been made dependent on the available memory, the security of the system has been improved, while simultaneously increasing the system performance for simple machine loading and expanding the loading capability.

Both the job and the channel IDs are now long words, the most significant word is a tag which is allocated cyclically by the OS, the least significant word is a channel or job number.

Protection is incorporated to ensure that more than one job can access an IO channel without interference. This multiple accessing capability is a major step forward in the handling of program communication and IO.

Changes From V0.03

The principal change since version 0.03 is the change of register for the timeout parameter on IO calls. This is now in register D3 throughout. This should help to avoid some confusion which was in the previous version.

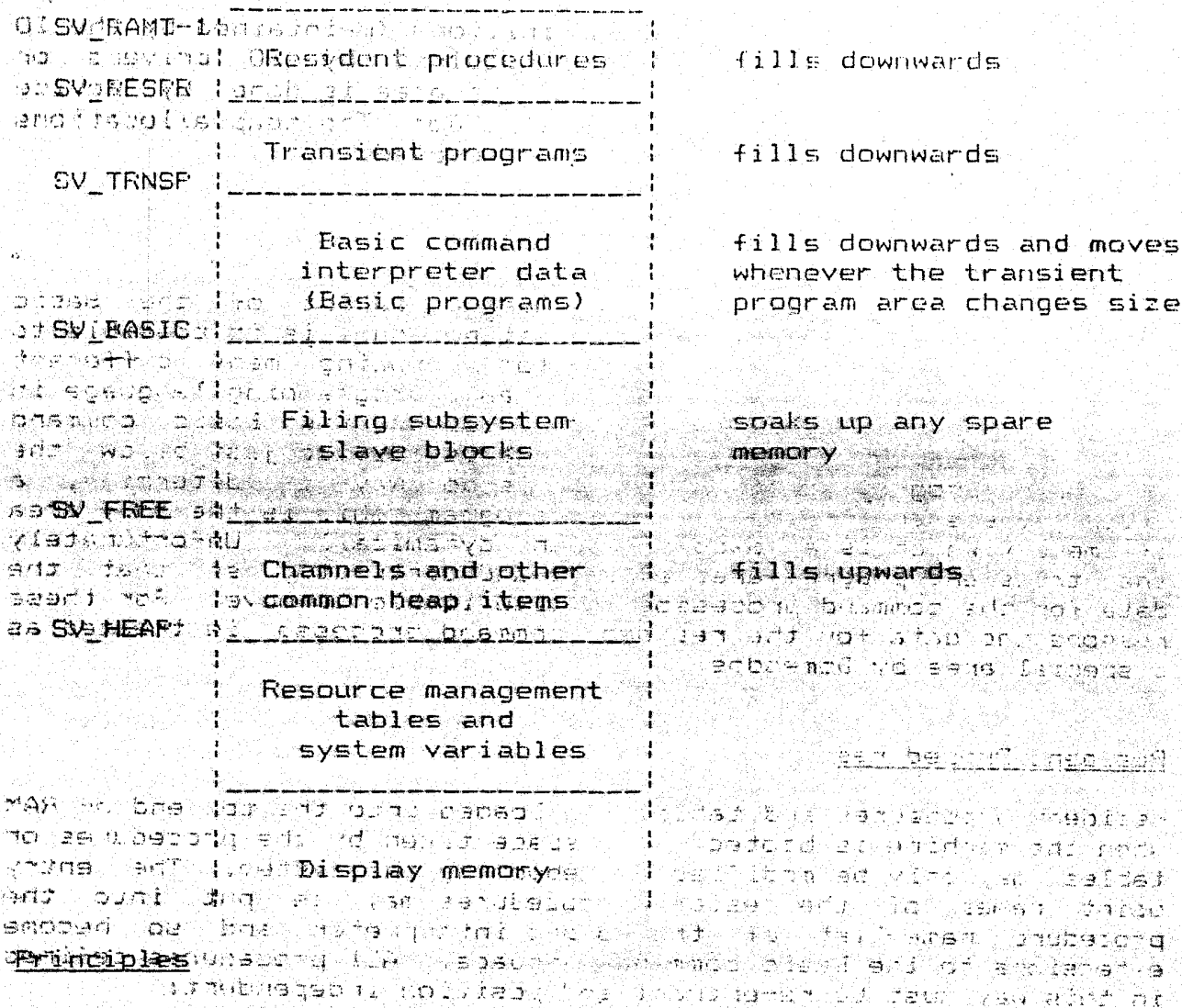
The address register, pointing to the name of a device to be opened has also been changed to reflect the usage in the utility routines for ASCII string to numeric conversion and v.v.

Memory Map

There are two aspects of the memory map used by the ZX83: the first is the physical memory map imposed by the ZX83 hardware; the second is the random access memory map which is managed by Domesdos.

Address	Area	Notes
FFFFF – FFFF3	Interrupt vectors	processor status 7 only
3FFFF – 30000	(RAM1)	128k machine only
2FFFF – 28000	RAM0 / SCR1	screen 1 or 32k bytes
27FFF – 20000	SCR0	screen 0 (default)
1FFFF – 1C000	I/O expansion	for external I/O devices
1BFFF – 18000	I/O hardware	
17FFF – 10000	I/O expansion	for external I/O devices
0FFFF – 0C000	ROM3	normally plug-in
0BFFF – 08000	ROM2	normally plug-in
07FFF – 00000	System ROM0	

The memory map within the RAM is determined by Domesdos.



Principles

There is no memory management hardware in the ZX83. This implies that all code must execute using physical memory addresses, and that, once a procedure is loaded into memory, it may not be moved within the physical memory. For this reason memory is usually allocated in fixed size areas which remain in a fixed location until deleted (c.f. IBM OS/MVT).

System Variables

The system variables are in a small block of RAM located immediately above the display memory. Since the display memory is of unknown size, the base address of the available RAM is calculated on entry to Domesdos.

System Tables

The system tables, and supervisor stack are resident just above the system variables.

System Heap

The heap area contains channel definitions (maintained by the IO sub system) and working storage required by IO drivers or programs. The space allocation in this area is done by device drivers (when invoked) or directly by a Job. The heap allocations of a Job are released when the Job is removed.

Basic Programs

The resident command processor uses a dialect of the Basic programming language. This means that not only is it possible to write complex command procedures for invoking many different operations, but it can also be used as a programming language in its own right. The data for the processor (the Basic command source and its associated variables) is located just below the transient program area. As there is no way to determine, a priori, the space taken by a Basic program, this is the only area of memory which is allowed to expand dynamically. Unfortunately the transient program area expands and contracts, so that the data for the command processor is also liable to move. For these reasons the data for the resident command processor is treated as a special area by Domesdos.

Resident Procedures

Resident procedures and tables are loaded into the top end of RAM when the machine is booted. The space taken by the procedures or tables may only be modified by rebooting the system. The entry point names of the resident procedures may be put into the procedure name-list of the Basic interpreter and so become extensions to the Basic command language. All procedures defined in this way must be re-entrant and position independent:

NO SELF MODIFYING CODE
NO LOCAL VARIABLES

Transient Programs

Transient programs are loaded into the area of RAM below the resident procedures. Each program must include areas for its own stack and working variables. These programs are not re-entrant. Programs which are not position independent must be loaded using a relocating loader written for specific programs. This loader could also resolve linkages into the resident procedures as well as providing facilities for overlaying programs. A general purpose loader is beyond the scope of Domesdos.

The transient program area may also be used for data areas. These may be created in the same way as Jobs, and they will have "Job" numbers, but care should be taken to ensure that they are not activated.

Filing System Slave Blocks

The filing system uses all the remaining memory for file slave blocks. The existence of these blocks is invisible to the normal file system accesses, as they merely duplicate data held on the microdrives. Accesses to data held in these blocks are much faster than accesses directly to microdrive.

A cold start will cause execution to commence at the bottom of the system ROM. This initialises the system variables and performs a RAM test and display demonstration/test.

The next stage is to check in turn the addresses 8000 and C000 for a characteristic word (4AFB); in each case, if this word is found, a CALL is made to the following address (8002 or C002).

Next the expansion slots are checked for device drivers, If these are found they are linked into the device driver chain using the resident procedure area. The format of the device driver ROMS is more fully described in the hardware expansion document.

If the code invoked by these calls returns control to the bootstrap ROM, then each of the microdrives will be searched in turn for a file called PROCS; if this is found, the file will be loaded into the resident procedures area.

Next, each of the microdrives will be searched in turn for a file called RUN; if this is found, the file will be loaded into the transient program area, and then the program will be invoked.

System Calls

In general, system calls are treated as atomic; while one job is in system mode no other job in the system can take over the processor. This provides for resource table protection, without need for complex procedures using semaphores. Some calls are only partially atomic, that is, when they have completed their primary function, the calling job may be "swapped out" before control returns. Such are all the IO calls (unless immediate return is specified), and the scheduler calls.

The standard system call mechanism is a trap to one of the system vectors (manager or I/O subsystem) with a parameter (byte) in D0 which determines the action to be taken. In the following tables, the value of D0 is given in HEX. D0 is also used to indicate the error return status. If, on return from a trap, D0 (long word) is non-zero, then an error has occurred during processing. Error returns from traps may be either negative (a system recognised error code represented in this document by a two letter mnemonic) or, in the case of traps which invoke additional device drivers, they may be a pointer to the error message. Registers D1 to D3 and A0 to A3 are not only treated as volatile, but may also be used to provide additional arguments for the traps. Using data registers to pass parameters, rather than using a parameter block in RAM pointed to by an address register is preferred from the point of view of simplicity and efficiency in the trap routines. It is, of course, a rather less flexible mechanism.

All system calls can potentially return an error "BP" (bad parameters).

The manager traps are used to control the allocation of machine resources. These are trap vector #1.

```
*****************************************************************
*                                                               *
*    TRAP #1    D0=0       MT.INF                               *
*                                                               *
*          System information                                   *
*                                                               *
*    Call parameters                 Return parameters          *
*                                                               *
*    D1                              D1.L current Job ID        *
*    D2                              D2.L ASCII version (n.nn)  *
*    D3                              D3   ???                   *
*    A0                              A0   pointer to system vars*
*    A1                              A1   ???                   *
*    A2                              A2   ???                   *
*    A3                              A3   ???                   *
*                                                               *
*****************************************************************
```

Jobs are created in the transient program area. A Job has a fixed allocation of memory which must include its stack and working storage. The Job save area is located above the top of the Job's own stack and occupies 72 bytes, programs must therefore allow sufficient room for this above the stack.

The command interpreter is itself a Job, but with the exceptional characteristic that its data area is expandable.

```
**************************************************************
*                                                            *
*    TRAP #1    D0=1        MT.CJOB                          *
*                                                            *
*         Creates a Job in transient program area            *
*                                                            *
*    Call parameters                 Return parameters       *
*                                                            *
*    D1.L length of Job (bytes)      D1.L Job ID             *
*    D2.L owner Job ID               D2   ???                *
*    D3                              D3   ???                *
*    A0                              A0   base of area allocated*
*    A1   top of stack WRT base      A1   ???                *
*    A2                              A2   ???                *
*    A3                              A3   ???                *
*                                                            *
*    Error returns:                                          *
*                                                            *
*         OM out of memory                                   *
*         NJ no room in Job table or D2 is not a Job         *
*                                                            *
**************************************************************
```

This trap allocates space in the transient program area, and sets up a Job entry in the scheduler tables. This does not invoke the Job and the space allocated is not initialised in any way. The program itself would normally be loaded, by another Job, into the space allocated, by a scatter file load, after this system call.

If the Job is to be independent, then D2 should be zero; if D2 is passed negative, then the current Job is the new Job's owner.

The Job area may also be used as subsidiary working space for existing Jobs. In this case, the register A1 should be passed as zero.

```
*****************************************************************
*                                                               *
*    TRAP #1    D0=2        MT.JINF                             *
*                                                               *
*         Information on a Job                                  *
*                                                               *
*    Call parameters                  Return parameters         *
*                                                               *
*    D1.L Job ID                      D1.L next Job in tree     *
*    D2.L Job at top of tree          D2.L owner Job            *
*    D3                               D3.L MSB -ve if suspended *
*                                          LSB priority         *
*    A0                               A0   base address of Job  *
*    A1                               A1   ???                  *
*    A2                               A2   ???                  *
*    A3                               A3   ???                  *
*                                                               *
*    Error returns:                                             *
*                                                               *
*         NJ Job does not exist                                 *
*                                                               *
*****************************************************************
```

This trap returns the status of a Job.

This trap may be used to check the status of a tree of Jobs. On each call D2 should be the ID of the Job at the top of the tree; to scan a complete tree the trap is made with D1 being the return value of the previous call. When the tree has been completely scanned D1 is returned equal to zero.

```
*****************************************************************
*                                                               *
*    TRAP #1    D0=4        MT.RJOB                             *
*                                                               *
*         Remove Job from transient program area                *
*                                                               *
*    Call parameters                  Return parameters         *
*                                                               *
*    D1.L Job ID                      D1   ???                  *
*    D2                               D2   ???                  *
*    D3                               D3   ???                  *
*    A0                               A0   ???                  *
*    A1                               A1   ???                  *
*    A2                               A2   ???                  *
*    A3                               A3   ???                  *
*                                                               *
*    Error returns:                                             *
*                                                               *
*         NJ Job does not exist                                 *
*         NC Job not inactive                                   *
*                                                               *
*****************************************************************
```

This trap removes a Job (and its subsidiaries) from the transient program area. Only inactive Jobs may be removed.

```
*****************************************************************
*                                                               *
*    TRAP #1    D0=5        MT.FRJOB                            *
*                                                               *
*          Force remove Job from transient program area         *
*                                                               *
*    Call parameters                 Return parameters          *
*                                                               *
*    D1.L Job ID                     D1    ???                  *
*    D2                              D2    ???                  *
*    D3                              D3    ???                  *
*    A0                              A0    ???                  *
*    A1                              A1    ???                  *
*    A2                              A2    ???                  *
*    A3                              A3    ???                  *
*                                                               *
*    Error returns:                                             *
*                                                               *
*          NJ Job does not exist                                *
*                                                               *
*****************************************************************
```

This inactivates a complete Job tree and deletes all Jobs in it. If D1 is negative then the Job is the current Job.

Neither of the traps to remove Jobs can remove Job 0.

```
*****************************************************************
*                                                               *
*    TRAP #1    D0=6        MT.FREE                             *
*                                                               *
*          Find largest contiguous free space                   *
*                                                               *
*    Call parameters                 Return parameters          *
*                                                               *
*    D1                              D1.L length of space found *
*    D2                              D2    ???                  *
*    D3                              D3    ???                  *
*    A0                              A0    ???                  *
*    A1                              A1    ???                  *
*    A2                              A2    ???                  *
*    A3                              A3    ???                  *
*                                                               *
*****************************************************************
```

Jobs have three well defined states: they are active (sharing CPU resources with other Jobs), suspended (e.g. waiting for IO or another Job) or inactive (occupying memory but not capable of using CPU resources).

In practice the only difference between an inactive Job and a Job which has been suspended indefinitely, is that the latter cannot be removed by a simple remove call (trap #1 D0=4).

The following four calls (D0=8 to D0=B) are not fully atomic as they invoke the scheduler.

A Job may be suspended for an indefinite period, or until a given time has elapsed. The timeout period is up to 32K*frame time.

```
**************************************************************************
*                                                                        *
*    TRAP #1   D0=8      MT.SUSJB                                        *
*                                                                        *
*         Suspends a Job                                                 *
*                                                                        *
*    Call parameters                 Return parameters                   *
*                                                                        *
*    D1.L Job ID                     D1   ???                            *
*    D2                              D2   ???                            *
*    D3.W timeout period             D3   ???                            *
*    A0                              A0   ???                            *
*    A1   address of flag byte       A1   ???                            *
*    A2                              A2   ???                            *
*    A3                              A3   ???                            *
*                                                                        *
*    Error returns:                                                      *
*                                                                        *
*         NJ not a valid Job ID                                          *
*                                                                        *
**************************************************************************
```

If the Job ID is negative, then the current Job is suspended. The flag byte is cleared when the Job is released. If there is no flag byte, then A1 should be 0. If the timout period is specified negative, then the suspension is indefinite. If the Job is already suspended, the suspension will be reset. All jobs are rescheduled.

```
*****************************************************************
*                                                               *
*    TRAP #1    D0=9        MT.RELJB                            *
*                                                               *
*         Releases a Job                                        *
*                                                               *
*    Call parameters                  Return parameters         *
*                                                               *
*    D1.L Job ID                      D1    ???                 *
*    D2                               D2    ???                 *
*    D3                               D3    ???                 *
*    A0                               A0    ???                 *
*    A1                               A1    ???                 *
*    A2                               A2    ???                 *
*    A3                               A3    ???                 *
*                                                               *
*    Error returns:                                             *
*                                                               *
*         NJ not a valid Job ID                                 *
*                                                               *
*****************************************************************
```

After this call all Jobs are rescheduled.

The activity of Jobs can be controlled by activation or by modification of the priority levels. A Job at priority level 0 is inactive, at any other priority level it is active.

```
*****************************************************************
*                                                               *
*    TRAP #1    D0=A        MT.ACTIV                            *
*                                                               *
*         Activates a Job                                       *
*                                                               *
*    Call parameters                  Return parameters         *
*                                                               *
*    D1.L Job ID                      D1    ???                 *
*    D2.B priority (0 to 127)         D2    ???                 *
*    D3                               D3    ???                 *
*    A0                               A0    ???                 *
*    A1                               A1    ???                 *
*    A2                               A2    ???                 *
*    A3                               A3    ???                 *
*                                                               *
*    Error returns:                                             *
*                                                               *
*         NJ Job does not exist or cannot be activated          *
*         NC Job already active                                 *
*                                                               *
*****************************************************************
```

This activates a Job in the transient area. Execution commences at the base address of the space allocated to the Job.

```
*****************************************************************
*                                                               *
*    TRAP #1    D0=B       MT.PRIOR                             *
*                                                               *
*         Changes Job priority                                  *
*                                                               *
*    Call parameters              Return parameters             *
*                                                               *
*    D1.L Job ID                  D1   ???                     *
*    D2.B priority (0 to 127)     D2   ???                     *
*    D3                           D3   ???                     *
*    A0                           A0   ???                     *
*    A1                           A1   ???                     *
*    A2                           A2   ???                     *
*    A3                           A3   ???                     *
*                                                               *
*    Error returns:                                             *
*                                                               *
*         NJ Job does not exist                                 *
*                                                               *
*****************************************************************
```

This call is used to change the priority of a Job. If D1 is negative it will change the priority of the current Job. Setting the priority to 0 will cause inactivation. This call re-enters the scheduler and so a job setting its own priority to zero will be immediately inactivated.

Atomic Actions

Actions by a Job which must be completed in one timeslice, so that no other Job can intervene, should be executed in supervisor mode. Two traps are provided to control this entry and the corresponding exit.

```
*****************************************************************
*                                                               *
*    TRAP #1    D0=C       MT.SUPVS                             *
*                                                               *
*         Enter supervisor mode                                 *
*                                                               *
*   -Call parameters                 Return parameters          *
*                                                               *
*    A7                              A7   supervisor stack pntr *
*                                                               *
*****************************************************************

*****************************************************************
*                                                               *
*    TRAP #1    D0=D       MT.SUPEX                             *
*                                                               *
*         Exit from supervisor mode                             *
*                                                               *
*    Call parameters                  Return parameters          *
*                                                               *
*    A7                              A7   user stack pointer    *
*                                                               *
*****************************************************************
```

Registers D0 to D7 and A0 to A6 are not changed by these calls. Only 64 bytes should be used on the supervisor stack. All space used on the supervisor stack must be released before exiting supervisor mode. In general there should be nothing on the supervisor stack when a manager trap (#1) is made.

While a job is in supervisor mode, it is recommended that no system calls, which are not fully atomic, (trap #1 D0=8 to D0=B and all trap #3 with timeout<>0) are made.

A pair of traps is available to clear out, or allocate the resident procedure area. These traps should only be invoked when the transient program area is empty.

```
*****************************************************************
*                                                               *
*     TRAP #1    D0=E        MT.ALRES                           *
*                                                               *
*          Allocate resident procedure area                     *
*                                                               *
*     Call parameters                 Return parameters         *
*                                                               *
*     D1.L number of bytes reqd.      D1   ???                  *
*     D2                              D2   ???                  *
*     D3                              D3   ???                  *
*     A0                              A0   base address of area *
*     A1                              A1   ???                  *
*     A2                              A2   ???                  *
*     A3                              A3   ???                  *
*                                                               *
*     Error returns:                                            *
*                                                               *
*          OM out of memory                                     *
*          NC unable to allocate (TRNSP area not empty)         *
*                                                               *
*****************************************************************

*****************************************************************
*                                                               *
*     TRAP #1    D0=F        MT.RERES                           *
*                                                               *
*          Release resident procedure area                      *
*                                                               *
*     Call parameters                 Return parameters         *
*                                                               *
*     D1                              D1   ???                  *
*     D2                              D2   ???                  *
*     D3                              D3   ???                  *
*     A0                              A0   ???                  *
*     A1                              A1   ???                  *
*     A2                              A2   ???                  *
*     A3                              A3   ???                  *
*                                                               *
*     Error returns:                                            *
*                                                               *
*          NC unable to release (TRNSP area not empty)          *
*                                                               *
*****************************************************************
```

```
*****************************************************************
*                                                               *
*    TRAP #1    D0=10      MT.DMODE                             *
*                                                               *
*         Sets or reads the display mode                        *
*                                                               *
*    Call parameters                 Return parameters          *
*                                                               *
*    D1.B key -1 read mode           D1.B display mode          *
*              0 mode is 4 colour                               *
*              8 mode is 8 colour                               *
*    D2.L zero                       D2   ???                   *
*    D3                              D3   ???                   *
*    A0                              A0   ???                   *
*    A1                              A1   ???                   *
*    A2                              A2   ???                   *
*    A3                              A3   ???                   *
*                                                               *
*****************************************************************

*****************************************************************
*                                                               *
*    TRAP #1    D0=11      MT.NSCRN                             *
*                                                               *
*         Sets or reads the screen number                       *
*                                                               *
*    Call parameters                 Return parameters          *
*                                                               *
*    D1.B key -1 read screen nr.     D1.B current screen number *
*              0 select screen 0, remove screen 1 (if exists)   *
*              1 (create and) select screen 1                   *
*              2 select screen 0, retain screen 1 (if exists)   *
*    D2                              D2.B number of screens     *
*    D3                              D3   ???                   *
*    A0                              A0   ???                   *
*    A1                              A1   ???                   *
*    A2                              A2   ???                   *
*    A3                              A3   ???                   *
*                                                               *
*    Error returns:                                             *
*                                                               *
*         OM no room for second screen                          *
*                                                               *
*****************************************************************
```

These calls are used to set the current display mode. They are treated as a manager traps as they affect all the displayed windows. There are serious risks involved in calling these traps when the machine is not completely idle.

```
*****************************************************************
*                                                               *
*     TRAP #1    D0=13      MT.RCLCK                            *
*                                                               *
*          Reads the clock                                      *
*                                                               *
*     Call parameters                 Return parameters         *
*                                                               *
*     D1                              D1.L time in seconds      *
*     D2                              D2   ???                  *
*     D3                              D3   ???                  *
*     A0                              A0   ???                  *
*     A1                              A1   ???                  *
*     A2                              A2   ???                  *
*     A3                              A3   ???                  *
*                                                               *
*****************************************************************

*****************************************************************
*                                                               *
*     TRAP #1    D0=14      MT.SCLCK                            *
*                                                               *
*          Sets the clock                                       *
*                                                               *
*     Call parameters                 Return parameters         *
*                                                               *
*     D1.L time in seconds            D1.L time in seconds      *
*     D2                              D2   ???                  *
*     D3                              D3   ???                  *
*     A0                              A0   ???                  *
*     A1                              A1   ???                  *
*     A2                              A2   ???                  *
*     A3                              A3   ???                  *
*                                                               *
*****************************************************************

*****************************************************************
*                                                               *
*     TRAP #1    D0=14      MT.ACLCK                            *
*                                                               *
*          Adjusts the clock                                    *
*                                                               *
*     Call parameters                 Return parameters         *
*                                                               *
*     D1.L adjustment in seconds      D1.L time in seconds      *
*     D2                              D2   ???                  *
*     D3                              D3   ???                  *
*     A0                              A0   ???                  *
*     A1                              A1   ???                  *
*     A2                              A2   ???                  *
*     A3                              A3   ???                  *
*                                                               *
*****************************************************************
```

As setting the clock takes a significant time, no adjustment is made if a call is made to adjust the clock and D1=0.

Time starts at 00:00 1 January 1970.

Memory Allocation for the Basic Command Interpreter

The basic command interpreter executes in user mode. There are two traps available to the interpreter to allocate and release memory.

TRAP #1 D0=16 MT.ALBAS

Allocate Basic program area

Call parameters		Return parameters	
D1.L	number of bytes required	D1.L	nr. bytes allocated
D2		D2	???
D3		D3	???
A0		A0	???
A1		A1	???
A2		A2	???
A3		A3	???
A6	base address	A6	new base address
A7	user stack pointer	A7	new stack pointer

Error returns:

DM out of memory

TRAP #1 D0=17 MT.REBAS

Release Basic program area

Call parameters		Return parameters	
D1.L	nr. of bytes to release	D1.L	nr. bytes released
D2		D2	???
D3		D3	???
A0		A0	???
A1		A1	???
A2		A2	???
A3		A3	???
A6	base address	A6	new base address
A7	user stack pointer	A7	new stack pointer

Common Heap Allocation

Space can be allocated in the common heap area by Jobs. The space can be owned by another Job and will be automatically released when the owner Job is removed.

```
*****************************************************************
*                                                               *
*    TRAP #1    D0=18      MT.ALCHP                             *
*                                                               *
*         Allocate common heap area                             *
*                                                               *
*    Call parameters                 Return parameters          *
*    ---                                                        *
*    D1.L number of byte required    D1.L nr. bytes allocated   *
*    D2.L owner Job ID               D2   ???                   *
*    D3                              D3   ???                   *
*    A0                              A0   base address of area  *
*    A1                              A1   ???                   *
*    A2                              A2   ???                   *
*    A3                              A3   ???                   *
*                                                               *
*    Error returns:                                             *
*                                                               *
*         OM out of memory                                      *
*         NJ Job does not exist                                 *
*                                                               *
*****************************************************************

*****************************************************************
*                                                               *
*    TRAP #1    D0=19      MT.RECHP                             *
*                                                               *
*         Release common heap area                              *
*                                                               *
*    Call parameters                 Return parameters          *
*                                                               *
*    D1                              D1   ???                   *
*    D2                              D2   ???                   *
*    D3                              D3   ???                   *
*    A0   base of area to be freed   A0   ???                   *
*    A1                              A1   ???                   *
*    A2                              A2   ???                   *
*    A3                              A3   ???                   *
*                                                               *
*****************************************************************
```

OS Extensions

The operating system may be extended by adding routines to service interrupts, and device drivers. These new routines are linked into lists maintained by Domesdos. As these routines and drivers are called before the corresponding system routines, they may be used to replace the system routines. For the interrupt linked lists Domesdos requires 8 bytes of RAM: 4 bytes for the link pointer (set by a manager trap), followed by a long word holding the entry address of the routine. The device and directory drivers require 16 bytes of RAM: the link pointer followed by the entry addresses for input/output, open and close routines. Note that the RAM used for these lists must be allocated by a Job before the link is made, and that the allocation should be in the resident procedure area, if possible, or else in the common heap. If the allocation is in the common heap, then the space should be owned by Job 0, otherwise if the Job owning the space is force removed from memory, before the entries are removed from the linked lists, the operating system will certainly crash.

There are five linked lists:

external interrupt servers,
50/60 Hz interrupt servers,
scheduler loop tasks,
device drivers and
directory device drivers.

For each driver there is a trap to link in a routine, and a trap to remove a routine from a list.

```
*****************************************************************
*                                                               *
*    TRAP #1    D0=1A      MT.LXINT                             *
*               D0=1C      MT.LPOLL                             *
*               D0=1E      MT.LSCHD                             *
*               D0=20      MT.LIOD                              *
*               D0=22      MT.LDD                               *
*                                                               *
*         Links an external interrupt service routine           *
*               a polling 50/60 Hz service routine              *
*               a scheduler loop task                           *
*               an IO device driver                             *
*               or a directory device driver                    *
*                   into the operating system                   *
*                                                               *
*    Call parameters                  Return parameters         *
*                                                               *
*    D1                               D1   ???                 *
*    D2                               D2   ???                 *
*    D3                               D3   ???                 *
*    A0   address of link             A0   preserved           *
*    A1                               A1   ???                 *
*    A2                               A2   ???                 *
*    A3                               A3   ???                 *
*                                                               *
*****************************************************************

*****************************************************************
*                                                               *
*    TRAP #1    D0=1B      MT.RXINT                             *
*               D0=1D      MT.RPOLL                             *
*               D0=1F      MT.RSCHD                             *
*               D0=21      MT.RIOD                              *
*               D0=23      MT.RDD                               *
*                                                               *
*         Removes an external interrupt service routine         *
*               a polling 50/60 Hz service routine              *
*               a scheduler loop task                           *
*               an IO device driver                             *
*               or a directory device driver                    *
*                   from the operating system                   *
*                                                               *
*    Call parameters                  Return parameters         *
*                                                               *
*    D1                               D1   ???                 *
*    D2                               D2   ???                 *
*    D3                               D3   ???                 *
*    A0   address of link             A0   preserved           *
*    A1                               A1   ???                 *
*    A2                               A2   ???                 *
*    A3                               A3   ???                 *
*                                                               *
*****************************************************************
```

I/O Allocation

The I/O subsystem may be divided into two distinct sections: the allocation of channels, devices and files, and the actual input or output calls. The I/O allocation calls are trap vector #2.

Device Names

All input or output is performed to a logical device or file. There is no direct mapping between logical devices or files and physical devices. Logical devices are named using the same convèntions as true file names, and so there are certain reserved filenames which may not be used. Within filenames no distiction is made between upper and lower case letters. All serial I/O is redirectable and it is not necessary for applications to know the type of device which is being driven. However, there are certain aspects of physical devices which may need to be specified when a channel is opened. These physical characteristics are appended to the logical device name. The I/O system itself does not act on these additional definitions, but passes the complete logical device name onto the appropriate device driver.

CON_wXhaxXy_k	console I/O, window area "w" by "h" pixels, top left hand corner at pixel position "x","y". Keyboard type-ahead buffer length "k" characters. The size and position are defined in terms of pixels on a 512x256 display map (position 256x128 is the centre of the screen in both display modes). The width and X position are both specified in multiples of 16 pixels. Default CON_512x244a0x0_128
SCR_wXhaxXy	screen output: window definition is as for CON. Default SCR_512x244a0x0
SERn_bp	serial I/O, port "n", baud rate "b" (e.g. 9600), "p" indicates parity: E,O,M,S for even, odd, mark or space parity, or if absent 8 bit no parity. Default SER1_9600_8.
NETnn	serial network link to node "nn".
PIPE_n	if "n" given it is an output pipe of length n bytes, otherwise it is an input pipe connected to the channel ID passed in D3
MDVn_name vol_name name	alternative forms of file name. MDV1 refers to microdrive "1". Neither the unit nor the volume name need be given, but giving them may speed up the open operation.

File system names

Requirements

1) File names should be compatible with ZX83 Basic name conventions.
This will allow the use of constructions such as

```
OPEN #1,mytext
```

where "mytext" is the name of the file, as well as

```
OPEN #chan,fname$
```

where the filename is in the string variable "fname$".

2) File names should allow the use of (actual or simulated) directory structures.

3) File names should allow the medium or drive to be

expicit (included in the name),
defaulted (system or previously specified default) or
undefined (all drives are searched).

4) File names should allow the automatic creation of related file names.

Approach

1) File names comprise letters, digits and underscores.

2) Each group of alphanumeric characters, separated by an underscore from the following group, is regarded as a directory.

3) A default string can be provided by a job for the operating system to append to the start of the filename given in any call to open a file. In addition the file system will recognise certain strings (e.g. MDV1) at the start of a filename as being a physical device name and will not append the default.

The option to search all drives for a file name (by specifying the medium name, for example) could be very expensive and will not be considered for initial software.

4) The usual mechanism for allowing automatic creation of a filename related to a given filename is to add to the end of the name an "extension". In our case we cannot distinguish a file_extension name (e.g. FRED_BAS) from a directory_file name (e.g. MYFILES_FRED), so that extensions will appear to be files within a directory.

An open call should supply both a filename and the extension expected by the application. The system will do the best it can.

Components of File Names

The general form of a file name is a series of alphanumeric strings connected by underscores.

Using "aaa" to denote a alphanumeric string (commencing with a letter), then

drive.name is	aaa	(names defined by the device drivers e.g. MDV1)
medium.name is	aaa	(name defined when the medium is formatted)
default is	drive.name{_aaa}	
or	medium.name{_aaa}	
file.name is	aaa{_aaa}	
full.name is	drive.name_file.name	
extension is	aaa	

NOTE: medium.name_file.name cannot be distinguished from file.name.

Opening Files

Files may be opened explicitly by an open operation, or implicitly by a copy, rename or delete operation.

Each open operation is characterised by the state of the file store before the file is opened (file does or does not exist), and the access rights after the file is opened (exclusive or shared).

Operation	state		access		type
	new	exists	exclusive	shared	
OPEN new	X		X		
OPEN overwrite	X	X	X		
OPEN exclusive		X	X		
OPEN share		X		X	
COPY source		X		X	share
COPY destination	X		X		new
RENAME source		X	X		exclusive
RENAME destination	X		X		new
DELETE		X	X		exclusive

The open file has three sources of names. The default name is defined on a per job basis, an extension may be supplied as an additional default (per open call), and either a full name or file name is supplied.

In the case of opening a new file, the name used by the open call is fully defined; in the case of opening an existing file, the file system tries to find a file name which corresponds to one out of a number of combinations of the supplied name and the defaults.

Order of Search for Files

full.name given	file.name given
For new and overwrite	
1) full.name	1) default_file.name
For exclusive and share	
1) full.name_extension 2) full.name	1) default_file.name_extension 2) default_file.name 3) file.name_extension 4) file.name

Examples of Opening Existing Files

Default	Filename	Ext.	Order of search
MDV1	FRED	BAS	MDV1_FRED_BAS MDV1_FRED FRED_BAS) only if FRED FRED) is a medium
MDV1	FRED_BAS	BAS	MDV1_FRED_BAS_BAS MDV1_FRED_BAS FRED_BAS_BAS) as FRED_BAS) above
MDV1	MDV2_FRED	null	MDV2_FRED
MDV1	MDV2_FRED	DATA	MDV2_FRED_DATA MDV2_FRED
QPAC_GAMES	INVADERS	EXEC	QPAC_GAMES_INVADERS_EXEC QPAC_GAMES_INVADERS INVADERS_EXEC) as INVADERS) above
QPAC_GAMES	MDV1	null	MDV1 (directory read only)

Channel Open and Close

The channel open calls use the file or device name to determine the type of device required. As each Job requires its own list of channels, the Job number must be given.

```
**************************************************************************
*                                                                        *
*    TRAP #2    D0=1        IO.OPEN                                      *
*                                                                        *
*         Open a channel                                                 *
*                                                                        *
*    Call parameters                  Return parameters                  *
*                                                                        *
*    D1                               D1    ???                         *
*    D2                               D2    ???                         *
*    D3.L code                        D3    ???                         *
*         0 old (exclusive) file or device                               *
*         1 old (shared) file                                            *
*         2 new (exclusive) file                                         *
*         3 new (overwrite) file                                         *
*    A0   address of channel name     A0    channel ID                  *
*    A1                               A1    ???                         *
*    A2                               A2    ???                         *
*    A3                               A3    ???                         *
*                                                                        *
*    Error returns:                                                      *
*                                                                        *
*         NO Not opened - too many channels open                         *
*         NJ Job does not exist                                          *
*         OM out of memory                                               *
*         NF file or device not found                                    *
*         EX file already exists                                         *
*         IU file or device in use                                       *
*         BN bad file or device name                                     *
*                                                                        *
**************************************************************************
```

If the job number is passed as a negative word (e.g. -1) then the channel will be associated with the current Job.

The file or device name should be a string of ASCII characters. This string is preceded by a character count (word), the pointer should point to this word (on a word boundry).

The error return "BN" indicates that the name of the device has been recognised but that the additional information is incorrect. (E.g. CON_512y240.)

The code should be zero for access to any non-shared device (in practice, anything but a file store). If the error code is non zero then no channel has been opened.

```
*****************************************************************
*                                                               *
*     TRAP #2    D0=2        IO.CLOSE                           *
*                                                               *
*          Closes a channel                                     *
*                                                               *
*     Call parameters                 Return parameters         *
*                                                               *
*     D1                              D1    ???                 *
*     D2                              D2    ???                 *
*     D3                              D3    ???                 *
*     A0    Channel ID                A0    ???                 *
*     A1                              A1    ???                 *
*     A2                              A2    ???                 *
*     A3                              A3    ???                 *
*                                                               *
*     Error returns:                                            *
*                                                               *
*          NO channel is not open                               *
*                                                               *
*****************************************************************
```

Serial I/O Calls

The serial I/O is fully redirectable. There are three types of return from the serial I/O system: wait for completion, return immediately, and wait until time-out or completion. These are all treated as one type of call distinguished only by the length of wait. This wait may be zero, defined, or indefinite. The call parameter is a 16 bit number of display frames (50 or 60 Hertz), which will allow a time-out period in excess of 15 minutes. A time-out period of -1 indicates an indefinite wait.

If a Job is waiting on IO and another Job requests IO on the same channel, then, if the timeout is not zero, the second Job is re-scheduled until the first Job is no longer waiting (complete or timed-out). The timeout period commences from the time the Job gets access to the channel, not the time it requests access (unfortunately).

If an output call returns incomplete, then it will remain incomplete, and any output not sent must be re-sent.

Serial I/O is in the form of bytes or characters. The single byte IO calls are used for data transfers where the Job transferring the data requires to control the actions performed by the driver; any bytes which may represent embedded control codes are processed by the Job not by the driver. The string transfers, however, are used for dumb transfers of bytes, if there are any embedded control codes recognised by the driver, these will be actioned, any unrecognised control codes will cause some device dependent recovery action.

File copies should normally use string transfers using as large an internal buffer as is consistant with efficient memory usage and the size of files being moved. This is much more efficient than using single byte moves! If the timeout period on the read string operation is zero, then the operation will fetch as many bytes at a time as are available.

Serial I/O calls use TRAP #3.
The channel I/D (long word) is always passed in A0 and it is not modified by the TRAP. The time-out period is always passed in D3 and is not modified by the TRAP. If a pointer to an array of bytes is passed in A1, then on return, A1 will point to the next byte.

```
*****************************************************************
*                                                               *
*     TRAP #3    D0=0        IO.PEND                            *
*                                                               *
*          Check for pending input                              *
*                                                               *
*     Call parameters                  Return parameters        *
*                                                               *
*     D1                               D1    ???                *
*     D2                               D2    ???                *
*     D3.W timeout                     D3.L preserved           *
*     A0   channel ID                  A0    preserved          *
*     A1                               A1    ???                *
*     A2                               A2    ???                *
*     A3                               A3    ???                *
*                                                               *
*     Error returns:                                            *
*                                                               *
*          NC not complete (no pending input)                   *
*          NO channel not open                                  *
*          EF end of file                                       *
*                                                               *
*****************************************************************
```

This trap is used to check for pending input on a channel. It does not read any data or modify the input channel in any way.

```
*****************************************************************
*                                                               *
*     TRAP #3    D0=1        IO.FBYTE                           *
*                                                               *
*          Fetch a byte                                         *
*                                                               *
*     Call parameters                  Return parameters        *
*                                                               *
*     D1                               D1.B byte fetched        *
*     D2                               D2    ???                *
*     D3.W timeout                     D3.L preserved           *
*     A0   channel ID                  A0    preserved          *
*     A1                               A1    ???                *
*     A2                               A2    ???                *
*     A3                               A3    ???                *
*                                                               *
*     Error returns:                                            *
*                                                               *
*          NC not complete                                      *
*          NO channel not open                                  *
*          EF end of file                                       *
*                                                               *
*****************************************************************
```

```
*****************************************************************
*                                                               *
*    TRAP #3    D0=2 or 3                                       *
*                                                               *
*    D0=2       IO.FLINE  fetch a line of characters terminated*
*                         by ASCII <LF> (hex A)                 *
*    D0=3       IO.FSTRG  fetch a string of bytes               *
*                                                               *
*    Call parameters              Return parameters             *
*                                                               *
*    D1                           D1.W nr. of bytes fetched     *
*    D2.W length of buffer        D2.W preserved                *
*    D3.W timeout                 D3.L preserved                *
*    A0   channel ID              A0   preserved                *
*    A1   base of buffer          A1   updated ptr to buffer    *
*    A2                           A2   ???                      *
*    A3                           A3   ???                      *
*                                                               *
*    Error returns:                                             *
*                                                               *
*         NC not complete                                       *
*         NO channel not open                                   *
*         EF end of file                                        *
*         BO buffer overflow (fetch line only)                  *
*                                                               *
*****************************************************************
```

The character count of a fetch a line trap includes the <LF> if found.

For the console I/O device, the trap to fetch a line has special significance. The characters read from the keyboard are echoed in the associated screen window. The cursor keys modify the line typed according to the standard cursor key rules.

```
                  < >  move cursor left or right by character
          shift < >  move cursor left or right by word
        control < >  delete character left or right
control/shift < >  delete word left or right
        control ^ v  delete to start or end of line
```

In addition the cursor in the appropriate window is enabled when ~~an~~ the fetch a line trap is made, and the cursor is suppressed when the line has been read.

When fetching more than one byte from an I/O channel the time-out period is the maximum time allowed from the issuing of the trap, rather than the time between the reception of consecutive bytes. It follows that the timout cannot be used with the fetch line trap to detect, for example, slow typing. As many bytes are fetched as is possible within the time limit.

```
*****************************************************************
*                                                               *
*    TRAP #3    D0=5        IO.SBYTE                            *
*                                                               *
*         Send a byte                                           *
*                                                               *
*    Call parameters                  Return parameters         *
*                                                               *
*    D1.B byte to be sent             D1    ???                 *
*    D2                               D2    ???                 *
*    D3.W timeout                     D3.L preserved            *
*    A0   channel ID                  A0    preserved           *
*    A1                               A1    ???                 *
*    A2                               A2    ???                 *
*    A3                               A3    ???                 *
*                                                               *
*    Error returns:                                             *
*                                                               *
*         NC not complete                                       *
*         ND channel not open                                   *
*         DF drive full                                         *
*         OR off window / paper etc                             *
*                                                               *
*****************************************************************

*****************************************************************
*                                                               *
*    TRAP #3    D0=7        IO.SSTRG                            *
*                                                               *
*         Send a string of bytes                                *
*                                                               *
*    Call parameters                  Return parameters         *
*                                                               *
*    D1                               D1.W nr. of bytes sent    *
*    D2.W nr of bytes to be sent      D2.W preserved            *
*    D3.W timeout                     D3.L preserved            *
*    A0   channel ID                  A0    preserved           *
*    A1   base of buffer              A1    updated ptr to buffer *
*    A2                               A2    ???                 *
*    A3                               A3    ???                 *
*                                                               *
*    Error returns:                                             *
*                                                               *
*         NC not complete                                       *
*         ND channel not open                                   *
*         DF drive full                                         *
*                                                               *
*****************************************************************
```

The error return OR will not occur when sending a string of bytes to a screen or a console; newlines are automatically inserted at the right hand margin, and the window is automatically scrolled up when there is a newline on the bottom line. In addition the byte value $A (ASCII <LF>) will cause a newline.

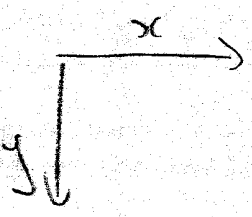
For some types of device, it is possible that a wait on output may not operate as expected. This will occur when the output device includes buffering and this internal buffering is invisible to the I/O subsystem. This applies to both single and multiple byte output. A return indicating that the the I/O transaction is complete indicates merely that all the output has been passed to the device driver.

TRAP #3 D0 = A SD_PXENQ.
D0 = B SD_CHENQ.

A0 channel id
A1 base of enquiry block preserved.
D3 trincat.

~~de~~ puts the window size X, Y
and cursor position X, Y into the

4 word enquiry block. The top left hand edge of the window is cursor position 0,0. The cursor position must always be less than the window size.



The screen output commands are not part of the basic redirectable I/O system, but since it would be possible to write screen emulating drivers, the screen manipulation calls are included under trap #3.

The screen output commands cover the operations of modifying the window, controlling the cursor, clearing part or all the window, scrolling and setting the colours.

Window control

```
**************************************************************
*                                                            *
*    TRAP #3    D0=C        SD.BORDR                         *
*                                                            *
*         Sets the border width and colour                   *
*                                                            *
*    Call parameters                Return parameters        *
*                                                            *
*    D1.B colour                    D1    ???                *
*    D2.W width                     D2    ???                *
*    D3.W timeout                   D3.L  preserved          *
*    A0   channel ID                A0    preserved          *
*    A1                             A1    ???                *
*    A2                             A2    ???                *
*    A3                             A3    ???                *
*                                                            *
*    Error returns:                                          *
*                                                            *
*         NC not complete                                    *
*         NO channel not open                                *
*                                                            *
**************************************************************
```

This call redefines the border of a window. By default this is of no width. The width of the border is doubled on the vertical edges. The border is inside the window limits.

```
*****************************************************************
*                                                               *
*    TRAP #3    D0=D        SD.WDEF                             *
*                                                               *
*         Redefines a window                                    *
*                                                               *
*    Call parameters                  Return parameters         *
*                                                               *
*    D1.B border colour               D1    ???                 *
*    D2.W border width                D2    ???                 *
*    D3.W timeout                     D3.L  preserved           *
*    A0   channel ID                  A0    preserved           *
*    A1   base of window block        A1    ???                 *
*    A2                               A2    ???                 *
*    A3                               A3    ???                 *
*                                                               *
*    Error returns:                                             *
*                                                               *
*         NC not complete                                       *
*         NO channel not open                                   *
*         OR window does not fit on page                        *
*                                                               *
*****************************************************************
```

This call redefines the shape or position of a window, the contents are not moved or modified, but the cursor is repositioned at the top left hand corner of the new window. The window block is 4 words long and is the width, height, X origin and Y origin.

Cursor control

```
*****************************************************************
*                                                               *
*    TRAP #3    D0=E        SD.CURE                             *
*                                                               *
*         Enable the cursor                                     *
*                                                               *
*    Call parameters                  Return parameters         *
*                                                               *
*    D1                               D1    ???                 *
*    D2                               D2    ???                 *
*    D3.W timeout                     D3.L  preserved           *
*    A0   channel ID                  A0    preserved           *
*    A1                               A1    ???                 *
*    A2                               A2    ???                 *
*    A3                               A3    ???                 *
*                                                               *
*    Error returns:                                             *
*                                                               *
*         NC not complete                                       *
*         NO channel not open                                   *
*                                                               *
*****************************************************************
```

The cursor is automatically enabled when a read line trap is issued to a window.

```
*****************************************************************
*                                                               *
*     TRAP #3    D0=F        SD.CURS                            *
*                                                               *
*          Supprcss the cursor                                  *
*                                                               *
*     Call parameters                  Return parameters        *
*                                                               *
*     D1                               D1    ???                *
*     D2                               D2    ???                *
*     D3.W timeout                     D3.L preserved           *
*     A0   channel ID                  A0    preserved          *
*     A1                               A1    ???                *
*     A2                               A2    ???                *
*     A3                               A3    ???                *
*                                                               *
*     Error returns:                                            *
*                                                               *
*          NC not complete                                      *
*          ND channel not open                                  *
*                                                               *
*****************************************************************
```

The calls to suppress or enable the cursor do not return an error if the cursor is already suppressed or enabled (respectively), as they merely ensure that the cursor is in the desired state.

```
*****************************************************************
*                                                               *
*     TRAP #3    D0=10 to 16                                    *
*                                                               *
*          Cursor positioning by character intervals            *
*                                                               *
*     D0=10       SD.POS     absolute position                  *
*     D0=11       SD.TAB     tabulate                          *
*     D0=12       SD.NL      newline                           *
*     D0=13       SD.PCOL    previous column                   *
*     D0=14       SD.NCOL    next column                       *
*     D0=15       SD.PROW    previous row                      *
*     D0=16       SD.NROW    next row                          *
*                                                               *
*     Call parameters                    Return parameters      *
*                                                               *
*     D1.W column number  (D0=10,11)     D1   ???               *
*     D2.W row number     (D0=10)        D2   ???               *
*     D3.W timeout                       D3.L preserved         *
*     A0   channel ID                    A0   preserved         *
*     A1                                 A1   ???               *
*     A2                                 A2   ???               *
*     A3                                 A3   ???               *
*                                                               *
*     Error returns:                                            *
*                                                               *
*          NC not complete                                      *
*          NO channel not open                                  *
*          OR position would be out of window                   *
*                                                               *
*****************************************************************
```

In the case of an error return, the cursor position is not changed. The cursor position is the top left hand corner of the next character rectangle referred to the top left hand corner of the window.

```
*****************************************************************
*                                                               *
*     TRAP #3    D0=17       SD.PIXP                            *
*                                                               *
*          Position cursor using pixel coordinates              *
*                                                               *
*     Call parameters                 Return parameters         *
*                                                               *
*     D1.W X coordinate               D1    ???                 *
*     D2.W Y coordinate               D2    ???                 *
*     D3.W timeout                    D3.L  preserved           *
*     A0   channel ID                 A0    preserved           *
*     A1                              A1    ???                 *
*    -A2                              A2    ???                 *
*     A3                              A3    ???                 *
*                                                               *
*     Error returns:                                            *
*                                                               *
*          NC not complete                                      *
*          NO channel not open                                  *
*          OR off window                                        *
*                                                               *
*****************************************************************
```

The cursor position is the top left hand corner of the next character rectangle referred to the top left hand corner of the window.

Part or all of a window may be scrolled; for partial scrolling the cursor is used as a reference. These traps cause pixels to be transferred from one row to another. Vacated rows of pixels are filled with paper colour. A positive scroll distance implies that the pixels in the window will be moved in a positive direction - down. The space left behind will be filled with paper colour.

```
**************************************************************
*                                                            *
*    TRAP #3   D0=18 to 1A                                   *
*                                                            *
*         Scrolls part or all of a window                    *
*    --                                                      *
*    D0=18      SD.SCROL  scroll all of window               *
*    D0=19      SD.SCRTP  scroll top of window               *
*    D0=1A      SD.SCRBT  scroll bottom of window            *
*                                                            *
*    Call parameters              Return parameters          *
*                                                            *
*    D1.W distance to scroll      D1   ???                   *
*    D2                           D2   ???                   *
*    D3.W timeout                 D3.L preserved             *
*    A0   channel ID              A0   preserved             *
*    A1                           A1   ???                   *
*    A2                           A2   ???                   *
*    A3                           A3   ???                   *
*                                                            *
*    Error returns:                                          *
*                                                            *
*         NC not complete                                    *
*         ND channel not open                                *
*                                                            *
**************************************************************
```

The dividing line between the top of the window and the bottom of the window is the top of the cursor line. Thus the cursor line is in the bottom of the window.

The whole of a window, or the whole of the cursor line, or the right hand end of the cursor line may be panned by any number of pixels to the right or to the left. A positive distance implies that the pixels will move to the right. The space left behind will be filled with paper colour.

```
**************************************************************************
*                                                                        *
*    TRAP #3   D0=1B, 1E and 1F                                          *
*                                                                        *
*         Pans part or all of a window                                   *
*                                                                        *
*    D0=1B      SD.PAN     pan all of window                            *
*    D0=1E      SD.PANLN   pan cursor line                              *
*    D0=1F      SD.PANRT   pan right hand end of cursor line            *
*                                                                        *
*    Call parameters                 Return parameters                   *
*                                                                        *
*    D1.W distance to pan            D1   ???                            *
*    D2                              D2   ???                            *
*    D3.W timeout                    D3.L preserved                      *
*    A0   channel ID                 A0   preserved                      *
*    A1                              A1   ???                            *
*    A2                              A2   ???                            *
*    A3                              A3   ???                            *
*                                                                        *
*    Error returns:                                                      *
*                                                                        *
*         NC not complete                                                *
*         ND channel not open                                            *
*                                                                        *
**************************************************************************
```

The cursor line is the whole height of the current character fount (either 10 or 20 rows). The right hand end includes the character at the current cursor position.

Clear Window

The clear window traps can clear all or part of a window. To clear a part of a window the cursor is used as a reference. The clear operation consists of overwriting all the pixels in the designated area with paper colour.

```
**************************************************************
*                                                            *
*    TRAP #3   D0=20 to 24                                   *
*                                                            *
*         Clears part or all of a window                     *
*                                                            *
*    D0=20      SD.CLEAR  clear all of window                *
*    D0=21      SD.CLRTP  clear top of window                *
*    D0=22      SD.CLRBT  clear bottom of window             *
*    D0=23      SD.CLRLN  clear cursor line                  *
*    D0=24      SD.CLRRT  clear right hand end of cursor line*
*                                                            *
*    Call parameters              Return parameters          *
*                                                            *
*    D1                           D1   ???                   *
*    D2                           D2   ???                   *
*    D3.W timeout                 D3.L preserved             *
*    A0   channel ID              A0   preserved             *
*    A1                           A1   ???                   *
*    A2                           A2   ???                   *
*    A3                           A3   ???                   *
*                                                            *
*    Error returns:                                          *
*                                                            *
*         NC not complete                                    *
*         NO channel not open                                *
*                                                            *
**************************************************************
```

The dividing line between the top of the window and the bottom of the window is the top of the cursor line. Thus the cursor line is in the bottom of the window.

The cursor line is the whole height of the current character fount (either 10 or 20 rows). The right hand end includes the character at the current cursor position.

Setting the Character Fount

The character fount is a 5x9 array of pixels in a 6x10 rectangle. There is a default fount built into the OS, but alternative founts may be selected.

```
*****************************************************************
*                                                               *
*    TRAP #3    D0=25      SD.FOUNT                             *
*                                                               *
*          Sets or resets the fount                             *
*                                                               *
*    Call parameters                 Return parameters          *
*                                                               *
*    D1                              D1    ???                  *
*    D2                              D2    ???                  *
*    D3.W timeout                    D3.L  preserved            *
*    A0   channel ID                 A0    preserved            *
*    A1   base of fount              A1    ???                  *
*    A2                              A2    ???                  *
*    A3                              A3    ???                  *
*                                                               *
*    Error returns:                                             *
*                                                               *
*          NC not complete                                      *
*          ND channel not open                                  *
*                                                               *
*****************************************************************
```

If the fount address is given as zero the default fount will be used.

The structure of the fount assumes that up to a certain value characters are invalid (default hex 1F), from the next value (default hex 20) all characters are valid up to a known maximum (default hex 7F). Thus the structure is as follows:

```
00          lowest valid code-1 (byte)
01          highest valid code (byte)
02 to 0A    9 bytes of pixels for all invalid characters
0B to 03    9 bytes of pixels for the first valid character
04          etc.
```

Each byte of pixels has the pixels in the least significant 5 bits of the byte. The top row of any character is implicitly blank.

Colours and Attributes

3 has a choice of eight colours, with flash from ink to
he colours are coded as follows:

	256 mode	512 mode
0	black	black
1	blue	black
2	red	red
3	magenta	red
4	green	green
5	cyan	green
6	yellow	white
7	white	white

dition any two colours may be put in a 2 by 2 pixel square
standard screen driver to produce stippled areas.

ase colour is in bits 2 to 0 of the colour byte, bits 5 to 3
ne byte contain the XOR of the base colour and the second
r in the pattern (if zero the colour is therefore solid).
7 and 6 are used to indicate the pattern:

00sssbbb	one dot of (sss XOR bbb) per four pixels
01sssbbb	horizontal stripes of (sss XOR bbb)
10sssbbb	vertical stripes of (sss XOR bbb)
11sssbbb	checkerboard.

: recoloured without changing the information in
vs the same sort of effects as resetting the
an attribute based screen, but it is very much

```
**************************************************
                                                  *
 D0=26      SD.RECOL                              *
                                                  *
olour a window                                    *
                                                  *
ameters                  Return parameters        *
                                                  *
                         D1    ???                *
                         D2    ???                *
meout                    D3.L  preserved          *
annel ID                 A0    preserved          *
ointer to colour list    A1    ???                *
                         A2    ???                *
                         A3    ???                *
                                                  *
returns:                                          *
                                                  *
NC not complete                                   *
NO channel not open                               *
                                                  *
**************************************************
```

our list is 8 bytes long and should contain the new
required for each of the 8 colours in the window. Each of
colours must be in the range 0 to 7. For 4 colour mode,
es 0, 2, 4 and 6 need to be filled in.

Colours

driver uses three colours. There is the background a window: referred to as **paper** colour; this is the ich is used by the scroll, pan and clear operations. the colour which is used by the character generator to highlighting background for individual characters or referred to as **strip** colour. And there is the colour used ng characters and drawing graphics: referred to as **ink**

```
**************************************************************
                                                             *
 #3   D0=27 to 29                                            *
                                                             *
 Set screen colours                                          *
                                                             *
 27     SD.SETPA  set paper colour                           *
=28     SD.SETST  set strip colour                           *
=29     SD.SETIN  set ink colour                             *
                                                             *
11 parameters                 Return parameters              *
                                                             *
.B colour                     D1    ???                      *
2                             D2    ???                      *
3.W timeout                   D3.L  preserved                *
0   channel ID                A0    preserved                *
1                             A1    ???                      *
2                             A2    ???                      *
3                             A3    ???                      *
                                                             *
Error returns:                                               *
                                                             *
   NC not complete                                           *
   ND channel not open                                       *
                                                             *
**************************************************************
```

TRAP #3 D0=2A and 2B

Set flash and underscore

D0=2A SD.SETFL set flash
D0=2B SD.SETUL set underscore

Call parameters	Return parameters
D1.B 0 attribute off else attribute on	D1 ???
D2	D2 ???
D3.W timeout	D3.L preserved
A0 channel ID	A0 preserved
A1	A1 ???
A2	A2 ???
A3	A3 ???

Error returns:

NC not complete
NO channel not open

TRAP #3 D0=2C SD.SETMD

Sets the character writing or plotting mode

Call parameters	Return parameters
D1.W mode	D1 ???
-1 ink is exclusive ored into the background	
0 character background is strip colour	
1 character background is transparent	
0 or 1 plotting is in ink colour	
D2	D2 ???
D3.W timeout	D3.L preserved
A0 channel ID	A0 preserved
A1	A1 ???
A2	A2 ???
A3	A3 ???

Error returns:

NC not complete
NO channel not open

generator supports two widths and two heights of
3 colour mode, only the double width characters may
ddition the spacing between characters is entirely
for simplicity of use only two additional spacings
directly: these are 8 pixel (in single width) and
double width).

```
**********************************************************
                                                         *
   D0=2D     SD.SETSZ                                    *
                                                         *
 t character size and spacing                            *
                                                         *
 arameters                 Return parameters             *
                                                         *
 haracter width/spacing  D1   ???                        *
 )    single width, 6 pixel spacing                      *
 1    single width, 8 pixel spacing                      *
 2    double width, 12 pixel spacing                     *
 3    double width, 16 pixel spacing                     *
 character height/spacing D2   ???                       *
 0    single height, 10 pixel spacing                    *
 1    double height, 20 pixel spacing                    *
 timeout                  D3.L preserved                 *
 channel ID               A0   preserved                 *
                          A1   ???                       *
                          A2   ???                       *
                          A3   ???                       *
                                                         *
ror returns:                                             *
                                                         *
 NC not complete                                         *
 NO channel not open                                     *
                                                         *
**********************************************************
```

with D1=0 or 1 in 8 colour mode will appear as if a call
n made with D1 equal to 2 or 3.

Block Filling

There are two traps which modify a rectangular block of a window. In one case the colour requested is written directly into the screen, in the other the colour is exclusive ored with the screen. Stipple colours may be used.

```
*****************************************************************
*                                                               *
*    TRAP #3    D0=2E or 2F                                     *
*                                                               *
*         Fill rectangular block in window                      *
*    ---                                                        *
*    D0=2E      SD.FILL    write colour into block              *
*    D0=2F      SD.OVER    XOR colour into block                *
*                                                               *
*    Call parameters                 Return parameters          *
*                                                               *
*    D1.B colour                     D1    ???                  *
*    D2                              D2    ???                  *
*    D3.W timeout                    D3.L  preserved            *
*    A0   channel ID                 A0    preserved            *
*    A1   base of block definition   A1    ???                  *
*    A2                              A2    ???                  *
*    A3                              A3    ???                  *
*                                                               *
*    Error returns:                                             *
*                                                               *
*         NC not complete                                       *
*         ND channel not open                                   *
*         OR block falls outside window                         *
*                                                               *
*****************************************************************
```

The block definition is in the same form as the window definition. It is 4 words long: width, height, X origin and Y origin. The origin is referred to the window origin.

This is a fast way of drawing horizontal or vertical lines.

ile subsystem has some extra traps to deal with two main onal requirements: record structured or direct access , and entire file load and save. To a certain extent both of functions are redirectable; and so they are included under #3.

only trap level support for structured files is the file ter positioning trap. This trap may be used to position the pointer to any byte in a file; this, together with the ity to read an arbitrary length string of bytes, means that ner level software may impose any file structure required.

addition some protection for data base files is provided by trap to flush file buffers.

```
**************************************************************************
*                                                                        *
*    TRAP #3   D0=40      FS.CHECK                                       *
*                                                                        *
*         Check all pending operations on a file                         *
*                                                                        *
*    Call parameters                  Return parameters                  *
*                                                                        *
*    D1                               D1    ???                         *
*    D2                               D2    ???                         *
*    D3.W timeout                     D3.L  preserved                   *
*    A0   channel ID                  A0    preserved                   *
*    A1                               A1    ???                         *
*    A2                               A2    ???                         *
*    A3                               A3    ???                         *
*                                                                        *
*    Error returns:                                                      *
*                                                                        *
*         NC not complete                                                *
*         NO channel not open                                            *
*                                                                        *
**************************************************************************
```

All attempts to read or write to a file, or to move the file pointer will cause implicit buffering operations, of the required physical blocks of the file, into the slave block area. These buffering operations are carried out in the background, and will continue even if the calls causing the operations return not complete.

This trap is used to check whether all of the pending operations have completed.

```
*****************************************************************
*                                                               *
*     TRAP #3    D0=41       FS.FLUSH                           *
*                                                               *
*          Flush buffers for this file                          *
*                                                               *
*     Call parameters                 Return parameters         *
*                                                               *
*     D1                              D1    ???                 *
*     D2                              D2    ???                 *
*     D3.W timeout                    D3.L  preserved           *
*     A0   channel ID                 A0    preserved           *
*     A1                              A1    ???                 *
*     A2                              A2    ???                 *
*     A3                              A3    ???                 *
*                                                               *
*     Error returns:                                            *
*                                                               *
*          NC not complete                                      *
*          ND channel not open                                  *
*                                                               *
*****************************************************************
```

When a write operation to a file is complete, the data written may still be in the slave blocks rather than on the file. This call may be used to check that a file is in a known (complete?) state.

```
**********************************************
*                                            *
*  TRAP #3   D0=42      FS.POSAB             *
*                                            *
*  Position file pointer absolute            *
*                                            *
*  Call parameters          Return parameters *
*                                            *
*                           D1    ???         *
*                           D2    ???         *
*  W timeout                D3.L  preserved   *
*    channel ID             A0    preserved   *
*    file position          A1    new file position *
*                           A2    ???         *
*                           A3    ???         *
*                                            *
*  Error returns:                            *
*                                            *
*     NC not complete                        *
*     NO channel not open                    *
*     EF end of file                         *
*                                            *
**********************************************

**********************************************
*                                            *
*  TRAP #3   D0=43      FS.POSRE              *
*                                            *
*     Position file pointer relative         *
*                                            *
*  Call parameters          Return parameters *
*                                            *
*  D1.L offset to file pointer  D1    ???     *
*  D2                           D2    ???     *
*  D3.W timeout                 D3.L  preserved *
*  A0   channel ID              A0    preserved *
*  A1                           A1    new file position *
*  A2                           A2    ???     *
*  A3                           A3    ???     *
*                                            *
*  Error returns:                            *
*                                            *
*     NC not complete                        *
*     NO channel not open                    *
*     EF end of file                         *
*                                            *
**********************************************
```

a file positioning trap returns an off file limits error then e pointer is set to the nearest limit (0 or end of file). The lative file positioning may, of course, be used to read the rrent file position.

Each file has a header containing information about the file. For Microdrive files this header is 64 bytes long and has the following format.

```
00   file type (word)
02   file length (long word)
06   last update (time)
0A   last backup (time)
0E   last reference (time)
12   8 bytes of type dependent information
1A   length of file name
1C   up to 36 characters of file name
```

```
*************************************************************************
*                                                                       *
*    TRAP #3   D0=47     FS.HEADR                                       *
*                                                                       *
*         Read file header                                              *
*                                                                       *
*    Call parameters                 Return parameters                  *
*                                                                       *
*    D1                              D1.W length of header read         *
*    D2.W buffer length              D2.W preserved                     *
*    D3.W timeout                    D3.L preserved                     *
*    A0   channel ID                 A0   preserved                     *
*    A1   base of read buffer        A1   top of read buffer            *
*    A2                              A2   ???                           *
*    A3                              A3   ???                           *
*                                                                       *
*    Error returns:                                                     *
*                                                                       *
*         NC not complete                                               *
*         NO channel not open                                           *
*         BO buffer overflow                                            *
*                                                                       *
*************************************************************************
```

The read header call is provided so that a Job can allocate the space for a load call as well as determining the characteristics of a file.

The information in the file header is duplicated in the directory so that the header is available as soon as a file is opened, and before any blocks of a file have been slaved into memory.

The file pointer is such that position zero is the first byte after the header. Thus block boundries on microdrive files are at positions 512*n-64.

e transferred and out of memory in their entirety with
ad and save. If the transient program area is used for
rap #1 must have been invoked to reserve the space
file load trap is invoked.

```
*************************************************************
*                                                           *
*  #3   D0=48      FS.LOAD                                  *
*                                                           *
*  Load file into memory                                    *
*                                                           *
*    1 parameters                Return parameters          *
*                                                           *
*                                D1    ???                  *
*    2                           D2    ???                  *
*    3                           D3.L  preserved            *
*    0   channel ID              A0    preserved            *
*   A1   base address for load   A1    top address after load*
*   A2                          A2    ???                  *
*   A3                          A3    ???                  *
*                                                           *
*  Error returns:                                           *
*                                                           *
*       NO channel not open                                 *
*                                                           *
*************************************************************

*************************************************************
*                                                           *
*  TRAP #3   D0=49      FS.SAVE                             *
*                                                           *
*       Save file from memory                               *
*                                                           *
*  Call parameters             Return parameters            *
*                                                           *
*  D1.L length of file         D1    ???                    *
*  D2                          D2    ???                    *
*  D3                          D3.L  preserved              *
*  A0   channel ID             A0    preserved              *
*  A1   base address of file   A1    top address of file    *
*  A2                          A2    ???                    *
*  A3                          A3    ???                    *
*                                                           *
*  Error returns:                                           *
*                                                           *
*       NO channel not open                                 *
*       DF drive full                                       *
*                                                           *
*************************************************************
```

drivers comprise many parts. For some devices the
al device driver which operates off either the scheduler
(50/60 Hz) or an interrupt. This type of driver will
transfer data into or out of internal queues or buffers.

t of the device driver forms the access layer which
the facilities required by the IOSS calls. The access
of all the device drivers must have an entry point for
opening, an entry point for channel closing and an entry
or input or output.

vice driver routines should finish with RTS (not RTE).

ysical device drivers are supplied for all the input/output
es of the ZX83 hardware, and the design of any additional
e drivers will depend very heavily on the type of hardware
type of device, no general rules have been formulated for the
gn of these.

n adding new device drivers (e.g. for a printer) the usual
hnique will be to use a standard physical driver (e.g. RS232),
h a new access layer to provide the special characteristics
quired by the new device.

evice Driver Definition Blocks

As device drivers may be added into the operating system at run time (or more usually on booting the machine), all the entry points for the various operations are maintained in linked lists by the manager. This type of list must be maintained in RAM and there is a standard form which is used for plug in device drivers.

There are three lists for the physical layer of device drivers these are for the routines invoked by:

external interrupts,
50/60 Hz interrupt,
scheduler loop.

The scheduler loop invocation is similar to the 50/60 Hz interrupt, but may occur more frequently. In particular, while the machine is idle (waiting for IO) there is a very tight scheduler loop.

A block of memory is linked into a phsical device driver list by reserving two long words at the start. The first long word is used by the manager to form the linked list, the second long word should hold the entry address of the physical device driver code. The rest of the block may be used by the device driver for maintaining flags, pointers, buffers etc.

A block of memory to be linked into the access layer of the IO sub-system must have four long words reserved at the start.

The first long word is used by the manager to form the linked list, the next three long words of an access layer driver are the entry addresses for input/output, opening and closing a channel.

that a device driver in ROM may have its own data area,
ating system passes a device driver several pieces of
formation. For the physical device drivers these are:

D3	number of 50/60 Hz interrupts since last 50/60 Hz service (50/60 Hz service and scheduler loop only)
A3	pointer to the base of the device driver def block
A6	pointer to system variables
A7	supervisor stack - routines may use up to 64 bytes

r registers may be treated as volatile.

ss layer calls the three address registers A3, A6 and A7
ie same significance, but the other registers are rather
it.

se address of the definition block refers to the standard
driver definition as set up for ROM device drivers at
It is entirely possible to have a different organisation
omitting some of the entry addresses and link pointers),
that case A3 will not have as convenient a value.

(A3)	link
4(A3)	address of external interrupt routine
8(A3)	link
$C(A3)	address of 50/60 Hz interrupt routine
$10(A3)	link
$14(A3)	address of scheduler loop routine
$18(A3)	link
$1C(A3)	address of input/output routine
$20(A3)	address of channel open routine
$24(A3)	address of channel close routine
$28(A3)	physical device driver working space

n a channel is made with the poi
a key in D3. A3 points to the assumed ba
definition block, A6 points to the system variable
s the supervisor stack pointer.

D2, A1, A2 and A3 may be treated as volatile.

s are implicitly bi-directional; hence it is the
ty of the device driver to trap illegal operations.

device operation has three error returns; ERR.NF
that the name was not recognised; ERR.BP indicates that
e name has been recognised, but the some of the
information is incorrect in value or format; while
dicates that all was correct, but the attempt to set up
nel definition block or buffer failed.

ations of a successful channel open are:

Decode name.
Allocate channel definition block and buffers in
common heap.
Initialise channel definition block.
Return address of channel definition block in A0.

el Close

close channel operation just requires to ensure that
ything is tidy and then release the space taken for the
nnel definition block and buffer. It is passed the address of
channel definition block in A0. A3 points to the assumed base
the device driver definition block, A6 points to the system
-iables area and A7 is the supervisor stack pointer.

egisters D1, D2, D3, A1, A2 and A3 may be treated as volatile.

Input/Output

The input/output operation is called once when an I/O trap is made, and then, if a wait until complete is required, on every scheduler loop, until the operation is complete or the operation times out.

The input/output operation is called with A0 pointing to the channel definition block, D0 defining the operation (this is a byte key, but the IOSS clears the top three bytes) , and with additional information in D1, D2 and A1. In the case of an incomplete operation, the values in these three registers should be set on exit so that on reentry the operation can continue; to assist in this D3 is zero on the first entry but is set to -1 on the the second and subsequent attempts at the operation while D1, D2 and A1 are unaltered. On all calls D0 defines the operation. A3 points to the assumed base of the device driver definition block, A6 points to the system variables area and A7 is the supervisor stack pointer.

The register D3 may be treated as scratch within the device driver, registers D4 to D7, A4 and A5 should be saved and restored if they are to be used.

Decoding the Name

To assist in decoding the device names there is a IOSS utility IO_NAME. This checks the device name and evaluates the additional or optional parameters.

The full device name is formed using four components:

Name	ASCII characters (normally letters and the case is ignored)
Separator	ASCII character (if a letter the case is ignored)
Number	decimal number in range 0 to $2^{16}-1$
Code	one of a list of ASCII characters

The utility is passed a pointer to the actual device name (in A0) and a pointer to a block of memory sufficient to store the parameter values (in A3). If successful, it fills in the parameter block with either given or default parameter values.

The utility has three returns:

return	D0 (+ SR)	
standard	ERR.NF	name not recognised
standard +2	ERR.BN	name recognised, bad parameter
standard +4	0	OK

The description of the device name starts 6 bytes after the call.

Device Name Description

The description is of the following form:

Number of characters in name, characters of name.
Number of parameters
For each parameter one of
Space+separator, default value (numeric values)
Negative number, default value (number, no separator)
Number of codes, list of ASCII codes.

(All items are defined as words, all letters must be upper case)

For each numeric parameter value in the description, the utility will return either a given value, or the default. For each code list in the description the utility will return the position of the code in the list, or zero.

Examples

The CON description is:

```
DC.W 3,'CON'              console
DC.W 5                    five parameters
DC.W ' _',512,' X',244    window size
DC.W ' @',0,' X'0         window position
DC.W ' _',128             keyboard queue length
```

Device name	Parameters
CON	512,244,0,0,128
CON_256	256,244,0,0,128
con__60	512,244,0,0,60
con@0x12	512,244,0,12,128
con_256x64@64x128_20	256,64,64,128,20

The SER description is

```
DC.W 3,'SER'              RS232 serial device
DC.W 3                    three parameters
DC.W -1,1                 port number (default 1)
DC.W ' _',9600            transmission speed
DC.W 4,'EOMS'             parity even, odd, mark, space
                          (default is eight bit)
```

Device name	Parameters
SER	1,9600,0
sere	1,9600,1
ser1_m	1,9600,3
ser2_1200	2,1200,0

The physical layer drivers must not allocate or release machine resources.

The access layer drivers must be re-entrant: all working variables must be stored in the channel definition block between calls to a device driver. The channel definition blocks must have 6 long words (24 bytes) reserved at the start for use by the IOSS; all the rest of the space is usable by the device driver.

To allocate space the routine MM_ALCHP is called with the required allocation (in bytes) in D1. D0 (and the condition codes) is returned negative if the alloction fails. MM_ALCHP may allocate a slightly larger space than requested: the actual length is returned in D1. The base address of the allocated block is returned in A0.

To return a table to the free space area, MM_RECHP should be called with the base address of the block in A0.

Five routines are provided for handling queues: IO_QSET sets up the queue pointers, IO_QIN puts a byte in a queue, IO_QOUT takes one out, IO_QEOF puts an EOF flag in and IO_QTEST checks if the queue has anything in.

A queue is defined by a block of 4 long words at the start. Only the most significant bit of the first long word is used by the queue routines to flag end of file. The entire word is cleared by IO_QSET and the rest of the first long word may be used by the device drivers for linking queues etc.

The usable length of a queue is one byte less than the actual length of the queue. This means that the minimum space taken by a queue and its header is 18 bytes.

For all routines the pointer to the queue header is passed in A2 and the data (queue length for IO_QSET, byte in or out for IO_QIN or IO_QOUT) is in D1. A3 is modified in an arbitrary way, and D0 is used as an error flag (ERR.NC) if the queue is full (IO_QIN and IO_QTEST) or empty (IO_QOUT). IO_QOUT and IO_QTEST can also return ERR.EF if end of file is reached.

Simple Serial IO

For simple serial IO there is a direct queue handling routine. When the channel definition block is set up for simple serial IO then the 7th and 8th long words should be set to point to the queues for input and output respectively. If either input or output is prohibited, then the corresponding pointer should be zero.

IO_SERQ should be called with the standard IOSS values in D0, D1, D2, A0 and A1. It treats actions 0 to 7 inclusive: for undefined actions, it returns error ERR.BP

For serial IO where the operations for byte input and output are not so simple, the routine IO_SERIO may be called. The call instruction should be followed by three long words, these being the entry addresses for

testing for pending input,
fetch byte,
send byte.

he machine the operating system checks for plug in
The format of these drivers is a long flag word
lowed by the driver header.

```
Flag (long)     4AFB0001
pointer to description (word)
number of drivers in this ROM (word)
first driver entry address list
00   length of driver definition block (long)
04   pointer to external interrupt routine (or 0)
06   pointer to 50/60 Hz interrupt routine (or 0)
08   pointer to scheduler loop routine (or 0)
0A   pointer to input/output routine
0C   pointer to channel open routine
0E   pointer to channel close routine
10   pointer to initialisation routine
second driver entry list
etc..
```

inters are relative to the base of the ROM.

he machine will set up a driver definition block in RAM
of the drivers; then call the initialisation routines
A3, A6 and A7 set to the standard values for device

ription should be in the form of a character count (word)
by the ASCII characters of the device description(s). It
mmended that the number of characters should be limited to

mber of routines in the ROM which may be useful to code; also there are certainly routines which are even required by) device drivers. These routines oups depending on the requirements on the calling

llocation

ap allocation routines are usually accessed by trap certain circumstances, for example within the parts of the rs invoked by trap #2, the routines are called by in the supplied operating system. The requirements g code are that:

e execution is in supervisor mode and
e code has not been invoked by interrupt other than the 50/60 Hz polling interrupt.

Trap Routines

ines are used to invoke certain system traps where the to be set up for the trap conform to a predefined ey must be called in user mode.

ility Routines

al utility routines may be called from any code and make l demands.

lity Routines

c utility routines may be called from any code, but with itation that all addresses passed to the routines should ive to A6.

Table

ry point adresses of these routines are held in a vector To access a routine the following code may be used:

```
VE.W    aa.aaaa,An      aa.aaaa is the entry vector
R       (An)            An is an address register
```

urn from any utility routine which sets a status code in status register is set according to the value in D0.

ines are provided for common heap management. One s space, the other frees the space. The space requested clude room for the heap entry header. For simple heap this is 16 bytes long, for IOSS channels this is 24 bytes

ress of the heap area is the base of the area allocated, base of the area which may be used (contrast with trap #1 rd 19).

ctor C0 MM.ALCHP

Allocate common heap area

ll parameters		Return parameters	
1.L	space required	D1.L	space allocated
2		D2	???
3		D3	???
0		A0	base of area allocated
1		A1	???
2		A2	???
3		A3	???

Error returns:

OM out of memory

Vector C2 MM.RECHP

Releases common heap space

Call parameters		Return parameters	
D1		D1	???
D2		D2	???
D3		D3	???
A0	base af area to release	A0	???
A1		A1	???
A2		A2	???
A3		A3	???

simplified trap routines are used to reduce the overheads of ost common IO operations.

first three set up console or screen windows using a eter list which follows the call statement. In the first the window is opened using a name which has been supplied; lock of parameters defines the border, the paper and strip ur and the ink colour. The window is set up and cleared for

second two also define the window using an additional block four words.

```
*****************************************************************
                                                                *
  Vector C4 UT.WINDW  set up a window using a supplied name*
         C6 UT.CON    set up console window                 *
         C8 UT.SCR    set up screen window                  *
                                                            *
  Call parameters                Return parameters          *
                                                            *
  D1                             D1   ???                   *
  D2                             D2   ???                   *
  D3                             D3   ???                   *
  A0   ptr to name (WINDW only)  A0   channel ID            *
  A1   ptr to parameter block    A1   ???                   *
  A2                             A2   ???                   *
  A3                             A3   ???                   *
                                                            *
  Error returns:                                            *
                                                            *
       OM out of memory                                     *
       NO out of channels                                   *
       OR window is off-screen                              *
                                                            *
*****************************************************************
```

The parameter block is as follows:

```
00   border colour (byte)
01   border width  (byte)
02   paper/strip colour (byte)
03   ink colour (byte)
04   width (word)          )
06   height (word)         )  not required for UT.WINDW
08   X origin (word)       )
0A   Y origin (word)       )
```

ır routines exist for writing simple messages to a channel. Two them are a basic error message handlers which write a standard device driver supplied error message to either the system annel 0, or else to a defined channel. The other two write rts of more complex messages to a defined channel. (To select ıe system channel try SUB.L A0,A0)

```
**********************************************************************
*                                                                    *
*    Vector CA UT.ERR0   write error message to channel 0            *
*           CC UT.ERR    write error message to given channel        *
*                                                                    *
*    Call parameters                  Return parameters              *
*                                                                    *
*    D1.L error code                  D1   ???                       *
*    D2                               D2   ???                       *
*    D3                               D3   -1                        *
*    A0   channel ID (UT.ERR only)    A0   channel ID                *
*    A1                               A1   ???                       *
*    A2                               A2   ???                       *
*    A3                               A3   ???                       *
*                                                                    *
**********************************************************************


**********************************************************************
*                                                                    *
*    Vector CE UT.MINT                                               *
*                                                                    *
*         Converts an integer to ASCII adds a space and sends        *
*              it to the defined channel                             *
*                                                                    *
*    Call parameters                  Return parameters              *
*                                                                    *
*    D1.W integer value               D1   ???                       *
*    D2                               D2   ???                       *
*    D3                               D3   -1                        *
*    A0   channel ID                  A0   preserved                 *
*    A1                               A1   ???                       *
*    A2                               A2   ???                       *
*    A3                               A3   ???                       *
*                                                                    *
*    Error returns:                                                  *
*                                                                    *
*         All the usual IO                                           *
*                                                                    *
**********************************************************************
```

```
*****************************************************************
*                                                               *
*    Vector D0 UT.MTEXT                                         *
*                                                               *
*         Sends a message to a channel                          *
*                                                               *
*    Call parameters              Return parameters             *
*                                                               *
*    D1                           D1   ???                      *
*    D2                           D2   ???                      *
*    D3                           D3   -1                       *
*    A0   channel ID              A0   preserved                *
*    A1   base of message         A1   ???                      *
*    A2                           A2   ???                      *
*    A3                           A3   ???                      *
*                                                               *
*    Error returns:                                             *
*                                                               *
*         All the usual IO                                      *
*                                                               *
*****************************************************************
```

The message is in the form of a text string: number of characters (word) followed by the characters in ASCII. If a new line is required at the end of the message, this should be included in the message.

es are provided for handling linked lists.

utines are passed the base address of the item to be
r unlinked, and a pointer which points to either the
o the first item in the list, or to an item in the list.

item is linked in, it will be linked in at the start of
, or, if the pointer was to an item in the list, after
em.

item is removed, the pointer may point to the pointer to
st item in the list, or to any item in the list before the
be removed.

starting a new list, the pointer to the first item in the
must be zero.

item in the list must have 4 bytes reserved at the start for
ink pointer.

Vector			Return parameters	
	D2	UT.LINK link an item into a list		
	D4	UT.UNLNK unlink an item from a list		
Call parameters				
D1			D1	preserved
D2			D2	preserved
D3			D3	preserved
A0	base of item (un)linked		A0	preserved
A1	pointer to previous item		A1	updated
A2			A2	preserved
A3			A3	preserved

are provided for user heap manage...
ser heap is in multiples of 8 bytes. Free space is
using two long words per space. The first is the
space. The second is the relative pointer to the
ce. The use of relative pointers ensures that user
ble. Provided the user code can remember the length
n the heap, all of an area allocated may be used by
allocation of area the first long word holds the
area, and so, if desired, this may be retained by

e requires to keep one pointer to the free space in
is is a long word, and is a relative pointer to the
the heap. When the heap has no free space, either
does not exist, or because it is full, this pointer

set up by linking an area of ram into a non existant
p is expanded by linking an area of ram, contiguous
rent top of the heap, into the heap.

```
**********************************************************
                                                         *
 D8 MM.ALLOC                                             *
                                                         *
Allocates an area in a heap                              *
                                                         *
parameters                Return parameters              *
                                                         *
length required           D1.L length allocated          *
                          D2   ???                       *
                          D3   ???                       *
ptr to ptr to free space  A0   base of area allocated    *
                          A1   ???                       *
                          A2   ???                       *
                          A3   ???                       *
                                                         *
or returns:                                              *
                                                         *
OM no free space large enough                            *
                                                         *
**********************************************************

**********************************************************
                                                         *
tor DA MM.LNKFR                                          *
                                                         *
 Links a free space (back) into a heap                   *
                                                         *
l parameters              Return parameters              *
                                                         *
L length to link in       D1   ???                       *
                          D2   ???                       *
                          D3   ???                       *
base of new space         A0   ???                       *
ptr to ptr to free space  A1   ???                       *
                          A2   ???                       *
                          A3   ???                       *
                                                         *
**********************************************************
```

Basic Utility Routines

The string comparison routine used by the directory system, and the Basic interpreter, uses an extended interpretation of the value of a string and has four modes of operation.

Order

Since comparisons may be used to sort strings into order as well as checking for equality or equivalence, the order must be well defined. A form of dictionary order is attempted - this will require to be modified for foreign character sets.

Space is the first character.

Punctuation is in ASCII order (except "." which is the last).

All punctuation is defined to be before all letters or digits (e.g. **A.** before **AA.**).

Optionally, embedded numbers may be taken in numerical order (e.g. **Case5A** before **Case10A**, and also **Case5.10** before **Case5.5**).

All digits or numbers are defined to be before all letters (e.g. **bat1** before **bath1**).

An upper case letter comes before the corresponding lower case letter but after the previous lower case letter (e.g. **Bath** is before **bath** but after **axe**).

Optionally, an upper case letter is treated as equivalent to a lower case letter.

Space
!"#$%&'()*+,-/:;<=>?@[\]^_'{|}~0.
Digits or numbers
AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz

Comparisons

The relationship of one string to another may be

equal	all characters or numbers are the same or equivalent
lesser	the first part of the first string, which is different from the corresponding character in the second string, is before it in the defined order
greater	the first part of the first string, which is different from the corresponding character in the second string, is after it in the defined order.

Comparisons may be made directly on a character by character basis (type 0), or made ignoring the case of the letters (type 1), or made using the value of any embedded numbers (type 2), or both ignoring the case of letters and using the value of embedded numbers (type 3).

File and variable name comparisons use type 1.

Basic <, <=, =, >=, > and <> operators use type 2.

Basic == (equivalence) operator uses type 3.

```
*****************************************************************
*                                                               *
*    Vector DE UT.CSTR                                          *
*                                                               *
*         Compares two strings                                  *
*                                                               *
*    Call parameters                 Return parameters          *
*                                                               *
*    D0.B comparison type            D0.L -1, 0 or +1 ,2,3      *
*    D1                              D1   preserved             *
*    D2                              D2   preserved             *
*    D3                              D3   preserved             *
*    A0   base of string 0 wrt A6    A0   preserved             *
*    A1   base of string 1 wrt A6    A1   preserved             *
*    A2                              A2   preserved             *
*    A3                              A3   preserved             *
*    A6   base address register      A6   preserved             *
*                                                               *
*****************************************************************
```

D0 (and the status register) is set negative if the string at (A6,A0) is less than the string at (A6,A1) etc..