SMS2 USER MANUAL

- For the standard environment -

This manual deals with the use of the computing environment called **SMS2** which is supplied in two forms: A standard form and a system form. The standard form includes an adaptable general-purpose user interface whilst the system form does not. Secure computer systems can be built using the system form whereas the standard form is open.

The manual is an introduction and guide to the standard interface, it is assumed that the reader is familiar with computers that have a graphic user interface.

Written on a 1 Mbyte Atari 1040STF equipped with an Atari SM125 mono monitor and printed on a Canon BJ- lOex Bubble jet printer. The software used was the text processor, Texts", and SMS2.

Contents

Introduction

SMS2 is an object-oriented multitasking system that is network-ready. You do not have to do anything to make this happen, it is an inherent capability. SMS2 is built into a special type of memory called PER OM (Programmable Eraseable Read Only Memory). It provides you with the best of two worlds; the convenience of instant availablity with the power of being able to tailor the system to meet your needs.

SMS2 provides three mechanisms to allow you to control the **SMS2** user environment: 1)

A "graphic user interface" or "windowing system".

2) A "Hotkey" system.

3) **A** set of system access application programs.

SMS2 is an integrated memory resident environment supporting high resolution display modes and "windowing" software as standard. The Atari standard high resolution black and white display capability is a minimum for experiencing this multitasking environment.

There are three main distinctions between the SMS2 user environment and the environment provided by GEM.

The first is fundamental, in that the SMS2 user environment works in conjunction with an operating system which is designed to be multitasking: that is, it can support more than one application program executing at any time. This is very different from the GEM view of the world where, originally, it was possible to have only one application.

The second distinction is more abstract. The image that most Atari ST computer users have of the GEM environment is that of the 'Desktop'. However the Desktop was designed to interface to a single user singletasking operating system and it has great difficulty presenting a view of multiple applications executing concurrently.

The third distinction relates to the secondary effects of multitasking. Once you have a system capable of supporting a high level of concurrency the concept of sharing software resources follows naturally from the concept of sharing hardware resources. One step further and you arrive at the point of sharing resources across a network. This capability is inherent in **SMS2** and the user interface provides access and control of this power.

Because of the fundamental difference in operating system philosophy between the GEM environment and SMS2, there are a lot of detailed differences in the user interface. As the original GEM environment only allowed one application at a time, application programs were written to allow the user to have more than one window open to an application. Thus the **GEM** word-processor can be used to edit two letters each in its own window. The user switches between the windows and thus between the letters. The application 'owns' the menu bar at the top.

The two main problems with this approach are, firstly, that there seems no good reason to suppose that a user is more likely to want to look at two letters, than, for example, a letter and a spreadsheet, and secondly, all the work of handling the window needs to be done by the application program, which means that writing applications is much more difficult (and expensive) in these environments than it is for SMS2.

The approach used by SMS2 is much simpler. If you want to write two letters, you use two copies of a word-processor, if you want to write a letter and look at a spreadsheet at the same time, you use a word-processor and a spreadsheet program. You could, of course, use two copies of an integrated program that does both. For example much of the general control of the SMS2 user environment is provided by a group of application programs. There is no concept of a "standard desktop" in SMS2, the user decides which application programs are going to provide their desired computing environment.

SMS2 looks after the programs and the windows on the screen. Rather than an application program having to support one or more windows in the main part of the display as well as the menu bar at the top of the screen, each copy of an application program (Job in SMS2 jargon) is contained within one 'Primary' Window. The Primary Window may occupy the whole screen burying all other application programs.

Much of the work you are likely to do with your computer will entail using the keyboard. For this reason, SMS2 does not rely on the use of a mouse although it is a lot quicker to use a mouse for certain operations. SMS2 provides keyboard control for all user interaction. The pointer may be controlled with the cursor keys and single keystroke selection extends right through the whole of the menu-handling part of SMS2 and into the 'HOTKEY System' where a single keystroke can be used to start an application program, to 'pick' a application so that you can work with it, and to do all manner of customised operations.

Because the scope of single keystroke operations that can be done using the HOTKEY System is almost unlimited, we cannot provide you with any more than a guide to setting up your system. It is up to you to choose the operations you wish to do, and build them into your startup file. You can, of course, add to the operations, or remove them, at any time while you are using the computer.

Chapter 1

Hardware

SMS2 is designed to work with Atari Corporation's range of ST computers. STs are supplied with an operating system and user environment already built in called, respectively, TOS and GEM. On installation SMS2 replaces this existing system and the ST will no longer behave as originally intended.

SMS2 will work on most STs but there are exceptions. In particular machines that have had their processors, memory or graphics capability altered can be unsuitable. At present SMS2 does not work on TTs, Falcons and Books. SMS2 does not make use of the STs sound or the inbuilt television display facilities. SMS2 supports two graphics modes:- The standard Atari high resolution B&W mode, which requires an ST compatible B&W monitor connected to the ST and a colour graphics card, which plugs into the Mega STe VME slot and connects to a variety of multi-sync monitors. This colour graphics card is called the QVME card and is available from Furst Ltd.

There are many models of ST that **SMS2** supports, their various characteristics are listed below:

520ST, STF, STFM.
A keyboard machine with mouse equipped with 520 Kbytes of RAM. The F stands for inbuilt floppy disc drive and the M stands for video monitor. The video monitor port is not supported by SMS2. Its purpose is to allow the ST to be connected to a TV for games playing. All other hardware resources are supported. It is possible to connect an SCSI hard disc to this machine via an adaptor called the Link. Although SMS2 will work perfectly in this machine, 520 Kbytes of RAM is rather limiting. It is possible to extend the RAM but it is relatively difficult and expensive.

1040ST, STF, STFM.
As the 520 but equipped with lM bytes of RAM.
This machine might be considered as a base machine for running SMS2. They can be bought very cheaply second hand. This manual was written on a 1040STF.

Mega ST 1, 2, 4.
A "separates" machine. It was made in three RAM sizes as implied by the number that follows the name. eg Mega STl has IM bytes of RAM. the keyboard is separated from the main body of the computer. All the resources of this machine are supported by SMS2. An SCSI hard disc can be connected to this machine in the same way as the keyboard machines.

Stacy.
A portable machine with inbuilt hard disc. It was equipped with up to 4M bytes of RAM. It is fully supported by **SMS2** and is the only way to achieve portability with **SMS2.** It was a well made machine although curiously Atari never made a battery pack for it. They are difficult to find.

520STe, 1040STe.
The e stands for extended. These are keyboard machines with enhanced colour and sound capabilities. The memory of these machines is easily expanded by adding SIM memory modules .. The extended sound and graphics capability is not supported by SMS2. These are ideal low cost machines because their memory can be expanded to 4M bytes. Adding hard discs is as for standard STs.

Mega STe.
Top end "separates" ST supported by SMS2. Has a VME port, up to 4M bytes of RAM and has space for an internal SCSI hard disc. The QVME card slots into the VME port to provide high resolution graphics for the SMS2 user.

TT, Book and Falcon.
Not currently supported by SMS2.

SMS2 supports hard discs but connecting them to an Atari is not a simple task. Suitable hard disc connection kits and SCSI hard discs are available from Furst Ltd.

Installing SMS2

It is assumed that you have read the relevant sections of the Atari manual for your machine and chapter 2. Once you think you have a suitable selection of hardware and it is all connected together correctly you are ready to install **SMS2.**
Remember, there are two versions of SMS2; a black and white display version that only works with an Atari b&w monitor and a colour display version that works with the QVME graphics card and a wide variety of display monitors. Once you have decided which version is suitable for your needs the installation procedure is the same for both:

1) Locate the cartridge slot on your machine and, with the power off, carefully plug in the appropriate **SMS2** cartridge with the label face up.

2) Switch on the computer system.

SMS2 will automatically install itself into your computer's memory. It will be obvious if all is well because you will be confronted by a set of little windows called buttons, a CLI, a system monitor window, a system communication window and the screen pointer which will move in sympathy with your mouse movements. The system is now fully functional.

If this does not occur there should be a logical explanation:-

* Is your monitor switched on? Check the brightness and contrast settings.

* Is your monitor suitable and is it properly connected?

* Are you using a suitable Atari system? See chapter 2

* Is everything connected properly?

* When you tum the system on, is the green Atari light illuminated? If it isn't then you have some problem with power to the system or you have a faulty computer. Check the mains fuses.

* Is the yellow light on the **SMS2** cartridge illuminated? If it isn't then you have a faulty computer or a faulty cartridge.

§ If every thing seems OK but nothing is happening then it is possible that you have a faulty cartridge or a faulty or unsuitable Atari system.

If you press the ESC key on startup, ie when turning the computer on or when pressing the reset button, SMS2 will be ignored and the ST will become a GEM system.

NB We have found that there appear to be certain GEM/fOS programs which can corrupt the SMS2 PEROMS while the cartridge is plugged into the ROM port. Although there have only been a few instances of this, we would suggest removing the SMS2 cartridge from the port while working with GEM/fOS. In due course we hope to be able to provide a software protection scheme to overcome this and your software will be updated as and when this is available.

If you press the Return key on startup, SMS2 will ignore the inbuilt user environment and present you with the CLI. (see chapter 3). To proceed you will have to be able to provide your own environment as defined by a setup file containing a list of CLI instructions. This file is read by SMS2 when you enter the following instruction in the CLI:-

* [name of the setup file]

eg: * flpl_user (As found on the system utility disc)

If you enter the name of a setup file which is unavailable, SMS2 will report an error in the CLI as follows:-

[incorrect file name] - not found
Retry **Y** or N?

Note, there are two things consider:-

1) Some Atari keyboards require you to press the ESC key or Return key a few times in quick succession to achieve the above effect.

2) Do not cover the system communication window whilst extending the system. eg processing a setup file.

Chapter 3

Using SMS2

SMS2 provides a very high level of functionality at the most fundamental level. It is also an extendible system. The combination of SMS2's functionality, efficiency and reliability allows application programs to provide much of the user access. The PEROM contains a number of such application programs implemented as executable objects (Things' in SMS2 jargon). On startup these Things are loaded into the ST's memory and become available through the Hotkey system, the Buttons (the little windows at the top of the Screen) or the Command Line Interface.

There are certain conventions and key-presses which have been generally established which are useful to know as they provide the user with a consistent interface when learning new applications.

HIT - a 'hit' is the action of pressing either the Space Bar on the keyboard or the left mouse button. The resultant action depends upon the application but generally selects a particular item. So, for example, in the 'Files' menu one or more files may be selected (and de-selected) for a subsequent action by one or more 'hits'. It may also be combined with some other useful action as in the files menu when the name of the file is put into the 'stuffer buffer'. The 'stuffer buffer' is a piece of text which can be put into any other application simply by pressing ALT and Space. Previous stuffer buffer entries can also be recovered by subsequent ALT-Shift-Space key-presses.

DO - a 'do' is the action of pressing either the Enter key (Carriage Return) or the right mouse button. The resultant action also depends upon the application but the effect tends to be rather more drastic as some form of operation is then normally performed.

Button Pick - if you hold down the ALT key and press'.' (the Full Stop) then the Button Frame will be 'picked' to the top of your pile of jobs for easy access. Pressing both mouse buttons simultaneously will also have the same effect.

Button Sleep - any job may be 'buttonised' on the button frame even if they do not provide the facility as described below. By selecting ALT and Z the current job will be buttonised and can be selected again with a 'do'. Even buttons can be buttonised!

Disc Format - SMS2 has its own disc format which cannot be read from GEM and, similarly, GEM format discs cannot currently be read from SMS2. Any discs to be used with SMS2 therefore have to be formatted either from the 'Files' menu or the CLI, prior to use. There is a maximum filename length of 36 characters which includes any sub-directories.

Pointer - usually the pointer will appear as a small arrow pointing up and to the left. This indicates that the job which it is currently over will accept input via the mouse. However, the pointer does have some other manifestations. If it appears as a **'K'** within a box then the job underneath is currently waiting for keyboard input. Select the job with the space bar or a click on the mouse to pick the job. It can appear as a 'No Entry' symbol. This means that the job it is over is not able to accept input from the mouse or the keyboard at the present time. It can also appear in the form of a padlock which indicates that the job below has its window buried. The job can be picked by the usual methods. The only exception to this is the 'Background' job. The grey background is, in fact, a Public Domain program which displays the colour you see and, if picked, buries itself again. It is included for aesthetic reasons only and serves no other function. Of course, the pointer can also appear in any number of forms dependant upon the application it is over.

It is recommended to application writers that the following key combinations generate the suggested action within applications but the actual useage has to be left to the application writer to determine as to whether these are desirable or practical combinations.

CTRL-Fl - this should generate a 'sleep' event such that the application is tidied away onto the button frame. A commonly used icon which may be used for this purpose looks like three Z's decreasing in size.

CTRL-F2 - this should generate a 'wake' event such that any dynamic information which a job might be currently using is updated. The Files menu, for example, can be updated if you change the media in the floppy disc by using this key combination. The equivalent icon, if one has been defined, for the 'wake' event looks rather like a lightning bolt.

CTRL-F3 - this generates a 'resize' of the job's window if the job supports it. Normally the on-screen pointer will become a 'resize' icon which looks like a large box containing a smaller box in its lower righthand comer. The further to the lower right-hand side of the screen this is moved then the smaller the job's window becomes. The further to the upper left-hand side then the larger the job's window becomes. The action is terminated by either a 'hit' or a 'do'. It may not be appropriate for certain jobs to support this feature.

CTRL-F4 - this is the 'move' feature which allows jobs to be moved around the available screen dependant upon both the job's size and the display size. The icon for this looks like two boxes with their bottom-right and top-left comers overlapping. On selection, the pointer becomes a 'move' icon and you can reposition the jobs window accordingly. The action is terminated by either a 'hit' or a 'do'.

The Help' Button - with the standard Startup file you will see a button called 'Help'. This button is in fact a dedicated 'Files' menu which is configured to view files in a sub-directory called 'flpl_Help_'. This subdirectory is already set up on the **SMS2** system utilities disc provided and contains some useful information about the applications included etc. If you 'do' on this button you will get a list of text files which can be viewed by simply do'ing on the required file. In the standard 'Files' menu you can view any file by do'ing twice on that particular file. However, if you 'do' on a sub-directory (denoted by a'>' prior to the filename) then you will automatically move down into that sub-directory.

System Utilities - there are a number of system utilities already loaded into the PEROM cartridge. Details of the function of these utilities will be found on the System Utilities Disc (SUD) and can be viewed via the Help button. Some of the utilities have also been included on the SUD as they are user configurable and it may be useful to you to be able to do this. A program is supplied called 'Config' which enables you to do this. Many programs which run on SMS2 have user-configurable information which is held within configuration blocks in the program's code. This common format means that there is a consistency in the configuration of software from different sources.

Public Domain Software - there are a number of Public Domain (PD) programs already available for SMS2. A few of these may have been included with your copy of SMS2. These programs are not sold as part of the SMS2 system. They are merely distributed as PD software. One or more of the existing Atari Public Domain Libraries have expressed interest in distributing SMS2 PD software and details of these arrangements will be provided in due course.

Commercial Software - there are a number of commercially available programs which are already available for SMS2. These can be obtained either from Furst Ltd or from third-party Distributors and/or Software Houses in the UK or in mainland Europe.

Chapter4

The CLI

The command line interpreter is a program that provides a method for you to interact with the computer by typing simple commands and then by pressing the **return** key telling the computer to act on those commands. This action is termed executing a command. The CLI is the rectangular window that appears just below the button frame it is selected in the usual way. Once selected you can type into its window or you can tum it into a button by pressing **the Hotkey [Alt Z]**

Hotkeys	ALT (Alternate key) & other key, combinations assigned a certain action,
Buttons	Small (optionally movable) icons that when activated do something,

system parameters, resident extensions, I/0 control, the filing system, batch files and the network can all be controlled from the CLI

The principal constructions that can be manipulated with the CLI are:-

Files	Computer code capable of being stored in the filing system.
Programs	Computer code capable of being executed (ie used by a job).
Jobs	Principal units of execution.
Things	Special units of memory resident software controlled by the Thing system.

The Thing system is a means of defining and controlling areas of shared memory.

The CLI uses standard extension Things. The parameters required for each command are built into the extensions.

Command Line Interpreter Syntax

The command line interpreter has a very simple syntax. The case of key characters is ignored. A command consists of a procedure (like a verb) or a procedure and a parameter separated by any number of spaces and/or commas. The parameter defines the limits of the command.

Eg. For the procedure View with the parameter flpl_x

View Flpl_x: view,flpl_x : View, FLPl_x : **VIEW,,** ,,flpl_x - Are accepted

Viewflpl_x : View.flpl_x - Are not accepted
Error reported - Not implemented

A parameter consists of items and/or keys. Items should be separated from other items by any number of spaces and/or commas. A separator is not required after a key (because it can only be one character) or after a closing brace (which is recognisably the end of an item).

Items may be names or values or arbitrary strings. If an item contains a standard separator or separators (ie. spaces or commas) or backslashes then it should be enclosed in braces (curly brackets = { }). The CLI cannot interpret braces within braces. This limitation does not usually create problems.

Eg. View file called "Small letter"

View { Small letter} - **Will** look for and display a file called Small letter on the "data default" drive.
View Small letter - **Will** look for a file called Small
View { { Small letter}} - **Will** look for a file called { Small letter not { Small letter}, as closing braces are ignored by the interpreter.

Keys are a single character preceded by a backslash (\). Keys are use to introduce items of particular significance. eg:

\P this is used to pass a string to a program.

\J this is used to give a job a particular name. \

W this is used to specify a wake name.

A hotkey can be defined in two ways. If you define a lower case hotkey, then the hotkey action can usually be invoked by pressing ALT and the appropriate letter, regardless of whether the Shift key is pressed or Capslock is set. If, however, you define an upper case hotkey, then this action will only be invoked by ALT and the upper case character.

A filename is more than the name of a file it includes the name of the device where the file is to be found.

Eg.

flpl_demo	file whose name is "demo" on floppy disc number one
ram2_test.1	file called "test.I" on Ram disc number 1
winl_example 2	file called "example 2" on Winchester (Hard) disc number 1

Examples:

Procedure with no parameters.

Hot_go

Procedures with parameters	Type of parameter
View flpl_x	name.
Stuff computers	string.
Stuff { dear sirs}	string in braces
Ex flpl_x \p xxx \j yyy	name followed by two keys

If you try and execute an erroneous command the CLI will reject it and report an error. To simplify correction of a rejected command you can recall the last command by pressing the return key whilst holding down the ALT key. The standard system errors are reported by the CLI.

The command line interpreter is extendible to incorporate any procedure. This is a list of those implemented so far:

Inbuilt commands

QUIT	(quit)
* filename	process a batch file

Maintenance commands

Procedure	Parameters
Ex	program \P string \J Job name
Dir	full device name
Copy	source filename destination filename
Copy_o	source filename destination filename
Copyh	source filename destination filename
Copy_n	source filename destination filename
View	filename
Rext	filename
Program	full directory name
Data	full directory name
Dup	
Ddown	sub-directory sub-
Dnext	directory full
Dest	device name
Srtc	year month day hour minute [second]
Arte	seconds
Max_free	kilobytes limit
CLSO	
Baud	Baud rate
Wait	time in 1/50 sees
Kbd_table	O=English I=German
Accel_set	M=Mega STE T=TT
Accel __ on	
Accel_off	
Disp jnverse	0 or 1
Format	full device name

Button control commands

Procedure	Parameters
BT_sleep	program title x y button-origin colourway
BT_wake	program title x y button-origin colourway
BT_exec	program title x y button-origin colourway
BT_Hotkey	Hotkey title x y button-origin colourway

Hotkey control commands

Procedure	Parameters
Stuff	string
Hot_do	Hotkey character or name
Hot_go	
Hot_stop	
Hot_list	
Hot_res	Hotkey name \P string \J job name
Hot_resw	Hotkey name \P string \J job name \ **W** wake name
Hot_trn	Hotkey name \P string \J job name
Hot_trnw	Hotkey name \P string \J job name **\W** wake name
Hot_ldex	Hotkey name \P string \J job name
Hot_ldwk	Hotkey name \P string \J job name \ **W** wake name
Hot_exec	Hotkey name \P string \J job name
Hot_wake	Hotkey name **\P** string \J job name \ **W** wake name
Hot_pick	Hotkey name
Hot_char	string string ..... .
Hot_cmd	string string ..... .
Hot_on	Hotkey
Hot_off	Hotkey
Hot_set	Hotkey Hotkey or name
Hot_remv	Hotkey or name

Input/output control commands

Procedure	Parameters
Par_use	**PAR/SER**
Par_buff	size in bytes
Par_clear	
Par_abort	
Par_pulse	width in microseconds
Ser_use	**PAR/SER**
Ser_flow	**I/H/X**
Ser_buff	size in bytes
Ser_room	size in bytes
Ser_clear	
Ser_abort	
Prt_use	**PAR/ SER**
Prt_buff	size **in** bytes
Prt_clear	
Prt_abort	
Win_use	optional three character string
Win_drive	WIN number SCSI-target **SCSI** unit
Win_start	WIN number
Win_stop	WIN number

Description of Maintenance Commands

Ex - Execute program with parameters

The Ex command executes a program from an Executable Thing or a File. Optionally, a parameter string may be specified, and if you have some reason to do it, a job name may be specified.

Ex program \P string \J job name

The program name is the name of an executable file (using the program default directory) or executable Thing. The parameter string is passed to the program at the start of execution, and if a job name is given, the job will be set up with this name.

ExQD	execute QD
Ex QD \P fred	execute QD with filename fred
Ex QD \J {Text Editor}	execute QD with a job name
Ex Files	execute Files menu
Ex Files \P { \o \s}	\O and \S are keys for Files (passed in parameter string), they are ignored by CLI as they are within the braces. (see page 5.6)

Dir - Display a device directory

Dir will display the files contained on a device. If no device name is supplied then the Data Default device will be assumed. The medium name (if any) will be displayed followed by the amount of free space left out of the total for that medium. Sub-directories will be prefixed by '>' and executable files will be prefixed by a'E'.

When a page has been filled, Dir issues a "More?" request, ESC or N terminates the listing. Y or RETURN continues.

Dir flp1_	do a directory offlp1_
Dir flpl_test_	do a directory of a sub-directory on flp1_
Dir	do a directory of the current data default drive

Copy, Copy_o, Copy_h, Copy_n- File manipulation

The Copy commands copy files.

Copy source dest	copy file
Copy_o source dest	copy file, overwriting destination
Copy _h source dest	copy file, including header
Copy _n source dest	copy file, not including header

Copy flp1_fred ram l_fred

For the Copy, Copy _h and Copy _n commands, if the destination file name exists, then there will be a confirmation request before the file is overwritten. The Copy and Copy _o commands copy the file header if the file type is set. In all cases, the data default directory is used.

View - View file contents

View allows the contents of a file to be viewed a page at a time. The View command uses the data default directory. When a page of has been filled, View issues a 'More?' request. ESC or N terminates the viewing.

View filename	view a file called 'filename'

Rext - Load a resident Extension file

Rext loads and calls a resident extension file. The resident extension should set DO and the condition codes before returning to the CLI with an RTS. The Rext command uses the data default directory if a device name is not supplied.

Rext filename	load 'filename' as a resident extension

Program, Data, Dup, Ddown, Dnext - Default Directories

These commands control the default directories.

Program sets the default directory used by the commands which execute programs or set up Hotkeys to execute programs.

Data, Dup, Ddown and Dnext all control the data default directory. Data sets directory, while the others move up and down through a directory tree.

Program directory name	set program default
Data directory name	set data default
Dup	move up a directory level
Ddown sub-directory	move down a directory level
Dnext sub-directory	move up and back down
Data winl_ work_y90	set data default to winl_ work_y90
Dup	move up to winl_work
Ddowny89	move down to winl_work_y89
Dnext y88	move to winl_ work_y88

Dest - Set Destination Default
Dest sets the destination default device, usually this is a printer, although it could be any IO device.

Dest device name	set default destination
DestPRT	PRT is our destination

Srtc, Arte - Real Time clock
Srtc sets the real time clock to within one second or one minute. The year should be specified in full (eg 1990). All the parameters are numeric. Arte adjusts the real time clock by one second increments.

Srtc	year, month, day, hour, minute [second]
Srtc 1990 10 8 9 5	9:05 on 8 October 1990
Srtc 1990,10,8,9,5,27	9:05:27 on 8 October 1990
Arte seconds	positive values advance the clock
Arte -3600	end of summer time

Max_Free - Limit Memory Usage

This procedure is provided solely to inhibit programs which attempt to grab the whole memory. It limits the value returned by the system calls which give information on the available memory to programs.

Max_free limit	limit the free memory
Max_free 512000	pretend that there is no more than 500 kilobytes free memory

CLSO - Clear Window 0
This is a procedure to clean up the appearance of the system window.

Baud - Set baud rate

This procedure requires one parameter - the Baud rate. The rates will depend on the hardware. For the Atari ST the rates accepted are:

19200 9600 4800 2400 1200 600 300

Baud baud rate	set baud rate ...
Baud 2400	to 2400 bps

Wait - Wait for a period of time

This procedure requires one parameter - the time required to wait in units of 1/50 second.
The maximum specified period is 32,767 which is equivalent to approximately 665 seconds or 10.9 minutes. If you specify a negative value then the CLI will wait forever. Only a negative value of -1 should be used in this case and although we cannot think of any useful reason to suspend the CLI forever, you should be aware of this possibility.

Wait period	wait for a period of time
Wait 300	wait for 6 seconds
Wait 1500	wait for 30 seconds
Wait -1	wait forever

Kbd_table - Set the keyboard table layout

Kbd_table 0	defines an English keyboard
Kbd_table 1	defines a German keyboard

Accel_set - Sets up speed enhancements

Accel_set M	defines Mega STE accelerator and turns it on
Accel_set T	defines TT caches and turns them on

Accel_on - tum on caches if turned off

Accel_on	tum on caches

Accel_off - turn off caches if turned on

Accel_off	tum off caches

Disp_inverse - inverts the display image

Disp_inverse 1	inverts the display image
Disp_inverse 0	displays the normal image

Format - format a medium ready to accept data.
This procedure can be used to prepare, say, a floppy disc in a form such that data can be saved to it. As with other systems, formatting destroys any existing data on that medium and so **must be used with extreme caution!** There are three main 'media' to which formatting applies:-

Floppy Disc - place a blank disc into the disc drive and, from the CLI, type:-

Format flpl_name	where 'name' will become the title of that floppy disc

The format process will then begin and when the drive stops the disc is ready for use

Hard Disc - this is a MUCH more dangerous process and so certain safeguards are built in to try to stop you doing this by mistake. The same syntax applies as for floppy discs but there are some intermediate stages before the actual formatting begins. **PLEASE READ AND UNDERSTAND THIS SECTION BEFORE DOING ANYTHING TO YOUR HARD DISC.**

If you already have a hard disc and it is already partitioned under TOS then you may want to use one or more of those partitions for SMS2. In fact, the only requirement for getting a SMS2 drive is that you have to have one or more successive partitions. If you have **more** than four partitions then the first SMS2 partition should be one of the first three. Otherwise, the SMS2 partition can begin at any one of the first four partitions. For example:

If your hard disc has the target number 0, has three partitions and you want to change partition D into an SMS2 partition, then enter;

Format winl_harddisc (or whatever name you want to call the SMS2 hard disc

You will then be asked for the first partition to be formatted. Start counting from 0, as C=0, D= 1 etc and then enter;

1

for partition D. You are then asked for the number of partitions to be added into the SMS2 drive. If you enter more than one then the resulting **SMS2** partition will be the size of all the partitions which you are including; they are simply added together. So, for the purposes of this example you would then enter;

1

for one partition.

Two random characters are now displayed. THIS IS YOUR LAST CHANCE TO STOP THE FORMATTING PROCEDURE. If you type in the characters as displayed the selected partitions will be formatted as a SMS2 disc and all previous data will be lost. SO PLEASE MAKE SURE THAT YOU KNOW WHAT YOU ARE ABOUT TO DO. You can now use the SMS2 hard disc partition in the same way as you would use a floppy disc or ram disc.

SMS2 does not require any form of partitioning as the access speed does not go down dramatically with the number of directories and files stored on a hard disc. So, if you want to devote the whole of your hard disc for the use of SMS2, then you would format as for a floppy disc but using **WIN** as the device name rather than FLP. Before actually formatting the hard disc you will prompted to type in the two character code which appears on-screen. IF YOU TYPE IN THIS CODE IT IS TOO LATE TO STOP THE FORMATTING PROCESS SO MAKE SURE IT IS WHAT YOU WANT TO DO - ANY DATA ON THE HARD DISC WILL BE LOST FOREVER.

Ram Disc - these are a special case where formatting has an added effect. Rather than a medium name being given to the ram disc you 'name' it with the amount of memory you want to set aside for that ram disc. There are 8 ram discs (raml_ to ram8_) built in to SMS2 and they can be accessed in the same way as any other device. However, they disappear, along with their data, when the power is turned off.

Unless told otherwise, the ram discs behave 'dynamically'. This means that they will expand and contract depending on the amount of data stored in them and on how much memory is available on your machine. This is the most flexible way to use ram discs but in certain circumstances it can be inefficient in the use of memory and can sometimes lead to fragmentation of the memory. This is not as bad as it sounds, it just means that sometimes the memory gets divided up into smaller chunks rather than one large chunk.

If you use a ram disc for a specific purpose then it can sometimes cut down on this effect if you format that ram disc first with the maximum amount of storage space which is likely to be required. This then converts it into a 'static' ram disc and it stays at that fixed size until you either re-format to another size (losing all data contained in the ram disc) or you convert it back to a dynamic ram disc (also losing all data).

Eg:

Format flpl_test	formats the floppy disc in flpl_ with the name 'test'
Format flpl_	formats the floppy disc in flpl_ but with no name
Format winl_HD	formats the hard disc giving it the name 'HD'
Format ram2_100	formats ram2_ to 100*512 byte sectors which is equivalent to 50 Kbytes
Format ram8_0	formats ram8_ and makes it act as a dynamic ram disc

Description of button control commands

BT_sleep, Bt_wake, BT_exec, BT_hotkey - Make buttons

These four commands set up buttons.

```
BT _sleep program title x y button-origin colourway
BT_wake program title x y button-origin colourway
BT_exec program title x y button-origin colourway
BT_hotkey program title x y button-origin colourway
```

In all cases the title, origin and colourway are optional.

BT _sleep can only be used with Executable Things. BT _sleep will execute the Thing, starting off in the sleeping state. The Job may be woken using a Wake Hotkey, by pointing to the button and pressing the right hand mouse button, or using the Pick or Wake menus.

The program is the name of the executable Thing.

BT _sleep exec	Setup exec Thing as a button
BT_sleep exec	Programs Button will be called Programs

The title is the name that will appear in the button. If this is omitted, the Thing's normal name or symbol will be used. If you wish to omit it, but include some of the other parameters, you should use a null string {}.
The x,y button origin is a pair of numbers giving the position of the top left-hand comer of the button (inside the border). If they are omitted or zero, the button will be put into the Button Frame. If they are negative, the button will be placed at the current pointer position.

BT _sleep exec { } 1000 1000 Set up button at the bottom right comer.

The colourway is a number between 0 and 3. If omitted, the default will be used. 0 is white/green. 1 is black/red, 2 is white/red and 3 is black/green.

BT_sleep exec {} 0 0 2 Set up a white/red button.

BT_wake buttons look superficially similar. But, in this case, a Job is created which will Wake the executable Thing. There are two main differences in effect. A BT_sleep button will often be quicker to start up as you do not have to create a new Job each time it is used. But when a BT_sleep Job is woken, it leaves the Button frame until put back to sleep so it can be more difficult to find a BT _sleep job.

BT _exec buttons look slightly different. As well as the program name or title, it has a wake symbol. If you do on the program name, a new copy of the program will be started, even if there is one already. If you DO on the Wake item, you will Wake the program.

BT _hotkey will set up a button appropriate to the Hotkey type, If it is an exec Hotkey (set up by Hot_res, Hot_tm, Hot_ldex or Hot_thing) then the button will be the same as a BT_exec button. Ifit is a wake Hotkey (set_up by Hot_resw etc.) the button will look like BT_wake button. You cannot put Hot_char Hotkeys into a button:they would only send the Hot_char string to themselves, which is not very useful.

Description of hotkey control commmands

A Hotkey is the combination of ALT (Alternate key held) & some other predefined key pressed briefly (jabbed). The Hotkey parameter is the character of the key that is to be jabbed

ie the Hotkey [ALT d] is set with the Hotkey parameter d

Hot_do - Do a hotkey directly

Hotkeyed actions are usually started off by the HOTKEY system when it detects a Hotkey typed on the keyboard. Any program can invoke a hotkey. The CLI command for this is Hot_do.

Hot_do Hotkey or Hotkey-name

In common with other commands which reference a Hotkey, the Hotkey "Name" can be used in place of the Hotkey itself. Thus if [ALT C] is the Hotkey to pick the clock, and ALT Fis the Hotkey to execute Files then;

Hot_do C is the equivalent to Hot_do Clock &
Hot_do Fis the equivalent to Hot_do Files

Hot_go, Hot_stop

Hot_go starts the Hotkey system, Hot_stop stops it.

Hot-list - List the hotkeys

Hot_list lists all the Hotkeys currently set up.

When a page has been filled, Hot_list issues a "More?" request, ESC or N terminates the list. Y or RETURN continues.

Hot_res, Hot_tm, Hot_ldex, Hot_exec - Set up Hotkey to execute things or programs.

Hot_res loads a program, turns it into an executable Thing and then sets up a Hotkey to execute it. The Thing will be resident and will remain even if the Hotkey is removed. (loads code into resident procedure area)

Hot_tm loads a program, turns it into an executable Thing and then sets up a Hotkey to execute it. The Thing will be transient and will be removed if the Hotkey is removed. (loads code into transient area)

Hot_ldex sets up a Hotkey to load and execute a program. The program is not loaded until it is required.

Hot _exec sets up a Hotkey to execute an executable Thing. The executable Thing need not exist when the Hotkey is set up.

Hot_res	Hotkey name **\P** string **\J** job name
Hot_tm	Hotkey name **\P** string **\J** job name
Hot_ldex	Hotkey name **\P** string **\J** job name
Hot_exec	Hotkey name \P string **\J** job name

The name of the Thing set up by Hot_res and Hot_tm has is the job name (if given) or else the program name (if the program has one) or failing that, the file name.

If parameters are specified, these are passed to the program when it is executed. The Job name can be specified to distinguish programs set up with different parameters.

Hot_res a flpl_alarm	Sets up a resident alarm
Hot_exec ffiles \p {\o s}	Executes the thing called files with options

Hot_resw, Hot_tmw, Hot_ldwk, Hot_ wake - Set up Hotkey to Wake jobs or programs

Hot_resw	Hotkey name \P string \J job name \ W wake name
Hot_trnw	Hotkey name \P string \J job name \W wake name
Hot_ldwk	Hotkey name \P string \J job name \ W wake name
Hot_wake	Hotkey name \P string \J job name \W wake name

The Wake commands are similar to their exec counterparts, but as there are some programs which change their job names while executing, there is a provision to specify a Wake name.

Each of of these commands sets up a Hotkey to wake a program. If there is a job with a matching name already executing, then that job is picked and sent a wake event. Otherwise a new copy of the program is executed. Using wake Hotkeys rather than execute Hotkeys can help prevent your computer becoming clogged with unused jobs.

Hot_wake v Files \P {\ov} \W View [ALT v] wakes view default data files.

Hot_pick - Set up a Hotkey to pick a job.

The Hot_pick command sets up a Hotkey to pick a job. Picking a job brings its windows to the top of the heap, moves the pointer into its window area and connects the keyboard queue to the job.

When the Hotkey system tries to find a job to pick, it will accept an abbreviation of a program name, provided that the first character after the abbreviation is not a letter. If there is more than one job executing with the same program name , then each job will be picked in tum.

Hot_char, Hot_cmd - Set up a Hotkey to put character strings into the keyboard queue.

These two commands are used to set up character strings which are put into the keyboard queue when the appropriate Hotkey is pressed. If more than one string is specified, a newline character will be put between each string.

Hot_cmd has an additional function. Before stuffing the strings, it wakes the CLI and after the last string, it adds a new line.

Hot_char h hello	Stuff hello on [ALT h]

Hot-cmd x { copy raml_x ram2_x} copy x from raml_ to ram2_ on [ALT x}

Hot_on, Hot_off, Hot_set - Changing the Hotkeys

Individual Hotkeys can be turned on and off and the Hotkey used for a particular operation can be changed using the Hot_on, Hot_off, Hot_set commands.

Hot_remv - Remove a Hotkey

More permanent removal of a Hotkey is available using Hot_remv. This not only turns the Hotkey off, but removes the definition as well. If the Hotkey was set up using Hot_tm or Hot_tmw, the executable Thing, the program code and any jobs using it are also removed.

Hot_remv will need to be used before re-defining a Hotkey if the type of the Hotkey is to be changed.

NB Be careful not to replicate key definitions as this can lead to that hotkey being unavailable until a software reset.

Description of input/output control commands

SER and PAR devices

The serial and parallel ports are accessed through devices with the following names and parameters:-

SERnpftce	Serial Port receive and transmit
SRXnpftce	Serial Port receive only
STXnpftce	Serial Port transmit only
PARntce	Parallel Port transmit only Printer
PRT	Port (either SER or PAR)

The optional parameters are as follows:

n - port num her eg 1 or 2 (ignored)

p - parity	O (7 bit+ odd parity) E (7 bit+ even parity) M (7 bit+ mark= l) S (7 bit + space=O)
default is none	
f - flow control	H (Hardware CTS/DTR) I (Ignore flow control) **X (XON/XOFF)**
default is **H**	
t - translate	D (direct output) A (auto-CR)
c -<CR>	C (**<CR>** is end of line) **R** (no effect)
e - end of file	F (<FF> at end of file) Z (CTRL Z at end of file)
default is none	

Usually the only options that will be required are the 'F' for form feed and the 'C' option for daisywheel printers. If you are only going to use the SER port for output, it is better to use the STX name as this will enable the serial input port.

Par_use, Ser_use and Prt_use

The Ser_use and Par_use commands can be used to make the SER driver or the PAR driver accept the name of the other port. This is probably as confusing for you as it is for the system.

Par_use SER	PAR emulates SER (STX)
Par_use PAR	PAR does not emulate SER
Par_use	PAR does not emulate SER
Ser_use PAR	SER emulates PAR
Ser_use SER	SER does not emulate PAR
Ser_use	SER does not emulate PAR
Par_use ser	PAR driver emulates SER.
Copy fred ser	Copy a file to the PAR port!
Copy anne par	Copy another file to the PAR port.
Ser_use par	SER driver emulates PAR.
Copy fred, par	Copy a file to the SER port! now we are completely mixed up, the PAR port thinks it is called SER and the SER port thinks it is called PAR.
Ser_use Par_use	back to normal.

This apparent stupidity has one justification. If you are using a program which has built in the assumption that a printer is attached to a particular port, and your printer is actually attached to the other one, you can make one port pretend to be the other. This might be easier than re-writing the program.

Prt_use is another matter entirely. The PRT (printer) port does not really exist. Programs may be written to send their printed output to PRT, all that is required is that, at some stage, the user has set the PRT device to access either the PAR port or the SER (STX) port.

Prt_use STX npftce	Attach PRT device to STX with options
Prt_use SER npftce	Attach PRT device to STX with options
Prt_use PAR ntcf	Attach **PRT** device to **PAR** with options

For example, there might be a daisywheel printer attached to the serial port, and a fast dot matrix printer attached to the parallel port. The following commands could be put on **HOTKEYs** (using Hot_cmd) to switch between the two.

Prt_use par	Select PAR port for printing.
Prt_use sere	Select SER port with <CR> as end of line.

The PRT device attaches to the real SER and PAR ports, it is not fooled by Par_use and Ser_use. The device name given with the Prt_use command must be a valid SER or PAR device name.

Par_buff, Ser_buff, Ser_room and Prt_buff

Normally, the output to the serial and parallel ports is dynamically buffered. It is saved in temporary buffers which are thrown away as the output passes through the port. This has only one problem. Although it makes the output port appear as fast as a RAM disk, it is possible to completely fill your computer's memory with the pending output. If there is a risk of this happening, you can preset the output buffer size.

Par_buff value	Set buffer (dynamic if value O or absent)
Ser_buff value	Set output buffer size (as Par_buff)

Ser_buff out in	Set output and input buffer sizes
Ser_roomi value	Set minimum room in the input buffer Set
Prt_buff value	buffer (dynamic if value 0 or absent) Set
Par_buff 10000	PAR buffer size to 10,000 bytes.
Ser_buff 200	Set SER output buffer size to 200 bytes.
Prt_buff 500	Set output buffer size to 500 bytes for either the SER or PAR port, whichever is attached to PRT.
Par_buff	Restore PAR buffer to dynamic.
Par_buff0	Restore PAR buffer to dynamic.

Note that the output buffer size is the size of the output buffer for each channel that is opened to the port. You may have many output (PAR or STX) channels open at once.

Ser_room specifies the amount of room that is left in the input buffer when the flow control mechanism is activated to prevent loss of received data. This will often need to be rather more than one as many "intelligent" communications devices keep on sending data for quite some time after they have been told to stop. The default room is 32 bytes, this will normally be enough for all but transatlantic over-run.

Ser_buff can also be used to specify the size of the input buffer. If this is done, the sizes of both the input and output buffers must be specified. The output buffer size may be specified as zero, in which case the output buffer will be dynamically allocated. If the input buffer size is specified to be smaller than or the same as the current Ser_ROOM, then it will be ignored. The actually usable buffer space is the difference between the buffer size and the spare room.

Ser_buff 500 0	input buffer 500 bytes, output dynamic.
Ser_room 200	allow 200 bytes over-run for a really stupid intelligent device.

Ser_flow

The Ser_flow command is used to preset the flow control for the serial port. Although this can be specified in the device name, it is really much more to do with what the port is physically connected to, rather than how you wish to use it. This command can preset hardware, **XON/XOFF** or no flow control.

Ser_flow **H**	Preset Hardware flow control
Ser_flow I	Preset Ignore flow control Preset
Ser_flow X	**XON/XOFF** flow control Preset
Ser_flow h	Hardware flow control Preset
Ser_flow i	Ignore flow control Preset
Ser_flow x	**XON/XOFF** flow control

Par_clear, Ser_clear and Prt_clear
Par_abort, Ser_abort and Prt_abort

Because of the power of the dynamic and multiple buffering facilities, it is possible that you may fill your computer's memory with junk. Usually this will be tidied up when it has been sent. The CLEAR and ABORT commands will throw away all the buffered output, for all channels which have been closed, without waiting for it to be sent. Any channel which is still open to the port should be unaffected. In addition the ABORT commands send an "ABORTED" message to the port.

Par_clear	Clear all closed **PAR** buffers
Ser_clear	Clear all closed SER buffers
Prt_clear	Clear all closed **PRT** buffers
Par_abort	Par_clear then send "aborted" message

Ser_abort	Ser_clear then send "aborted" message
Prt_abort	Prt_clear then send "aborted" message

Par_PULSE
The Par_PULSE command can be used to set the parallel printer port STROBE pulse length to any value between 5 and 250 microseconds. This can be used to improve the reliability of the parallel printer port when using long cables (10 microseconds should enable printer cables up to about 5 metres to be used) or to drive Taxan or Canon printers on short cables (150 microseconds appears to be adequate for 2 metres of cable).

Par_PULSE value	Set length of strobe pulse
Par_PULSE 10	now we can drive Epson printer on a long cable
Par_PULSE 150	... or Canon on short cable.

Values of greater than 250 are limited to 250 microseconds, values less than 5 are limited to 5 microseconds. Long pulse lengths will degrade the system performance while printing.

Passing strings to programs with the CLI \P key.

It is possible to pass information to a program prior to executing or waking it. The string passed may be interpreted in any way by the program.

In particular the system access programs accept the following parameters sent as a string within braces. The form of the parameter string is a Key followed by a string of characters. A key is a backslash followed by a letter. There may be spaces between the key and the value. The keys may be upper or lowercase.

Standard parameters.

\z xpos ypos   Start off the program asleep

This key sets up the program as a sleeping button. If a button position is given a movable button is created. Xpos ypos are the starting position of the button in pixel co-ordinates from the top left comer. If the position is not given, the button will be put into the Button frame.

Hot_exec a channels \p{\z) Creates a Hotkey [ALT a] that makes a button called channels in the button frame.

\b value   Button Colourway (value Oto 3)

This key specifies the button colour for a program set up as a sleeping button. If you use \b, you do not need \z unless you wish to specify the button position.

Hot_exec a channels \p{\bl} Creates a Hotkey [ALT a] that makes a button called channels in the button frame.

\n characters Button name

This key specifies the name that will appear in the button. If you use \n. you do not need \z unless you wish to specify the button position.

\c value value Program colourways (values Oto 3)

This key sets the colour of the main border and window colours.

Hot_exec a channels \p{\z200 200 \bl \nChan \cl 2}
Creates a Hotkey [ALT a] that makes a moveable button called Chan with specified colour. The program colours are also specified.

The system access program called Files accepts the following parameters:-

\m command Sets the primary function of the program

The command should be the selection key for the particular command. This is usually the first letter of the command. You may give the full command name.

c or copy - Copy files
m or move - Move files etc.

Hot_exec a files \p{\mc) Creates a Hotkey, [ALT a], that executes a Files program for copying files.

\o options Set the program options (v=view, t=tree, s=statistics, z=sleep)

The option letters should follow the key. If the z option is given, then when you press ESC, the program will go to sleep, otherwise the program will remove itself. If you give any options, then you must give all options you require as this overrides all defaults.

\s +/- order Sort order

The sort order should be given as+ or - and a single letter (n for name, t for time etc). The+ sign is optional.

\dname Directory

This specifies the initial directory. If the name starts with an underscore, it is added to the end of the data default directory.

Hot_exec a files \p{\z200 200 \cl 2 \ov \s-t} Creates a Hotkey [ALT a] that creates a moveable Button for a files program to view files in reverse time.

Chapter 5

The System Editor

The system editor is a very simple line editor that provides a way of modifying PROM and startup files. It is accessed by pressing the Hotkey "d" ie. Holding down the Alternate key whilst tapping the 'd' key. It can also be invoked by typing EX ED in the CLI.

The PROM device is a user accessible, lKb area of PEROM memory for storing a sequential list of instructions that is read by SMS2 on startup. This list of instructions sets up the user environment. If the last or only instruction in the PROM is: * Filename SMS2 will read a further list of setup instructions from that file.

If the editor is accessed and the F2 function selected you will be asked to enter the name of a file. Type the word "prom" and press Return. You will now be able to edit the contents of the PROM device and subsequently save the result back to prom via the F3 function. On startup SMS2 will set up the user environment according to your new instructions.
The setup instructions are the same as those accepted by the CLI.

ED always works in INSERT mode. If a line is filled with 79 characters then nothing can be inserted. The following keystrokes are supported:

TAB	moves cursor in multiples of 8
ENTER	inserts a new line
LEFT	left
RIGHT	right
UP	up
DOWN	down
ALT LEFT	to start of line
ALT RIGHT	to end of line
ALT SHIFT UP ALT	to top of text
SHIFT DOWN ALT	to bottom of text
UP	scroll line up
ALT DOWN	scroll line down
SHIFT UP	scroll page up
SHIFT DOWN	scroll page down
CTRLLEFT	delete left
CTRLRIGHT	delete right
ALT CTRL LEFT	delete from cursor to start of line
ALTCTRLRIGHT	delete from cursor to end of line
CTRL UP	delete current line, make the rest move up a line the
CTRLDOWN	same
ALTCTRLDOWN	insert a line, let cursor stay in the new line (unlike
	ENTER)

Chapter 6

The Network

The Standard SMS2 network (MIDinet) is a Local Area Network (LAN), ie. short distance network. It uses the two MIDI (Musical Instrument Digital Interface) connectors located on either the left hand side or rear of the computer (see chapter 2).
MIDinet is organised as a ring, so the MIDI OUT of one machine has to be connected to the MIDI IN of the next machine. All cables have to form a complete circle and every machine in the circle has to be switched on and working.

It is suggested that standard MIDI connection cables are purchased for this task. For those who wish to make up their own, the **MIDI** connector pinouts are shown below. Only pins 4 and 5 need be connected.

```
1 - THRU Transmit data                              1  -  Not  connected
2 - Shield ground                                   2  -  Not  connected
3 - THRU Loop return 4                              3 - Not connected
- OUT Transmit data 5                               4 - IN Receive data
- OUT Loop return                                   5 - IN Loop return
```

As the name implies the MIDI system was never designed to support networking. It has been used by SMS2 because it is the only spare 1/0 hardware that is common to all Atari computers. If you want to use the MIDI ports for their original purpose you will not be able to network as well. It must be understood that, despite the shortcomings of MIDI, you can very easily perform powerful operations on MIDinet, it allows the average user to explore the world of networking. The penalty of using the MIDI hardware is that the receiving computer stops during a network communication and that it is slow.

Networking capability is built into SMS2. It is a powerful integrated technology that allows every machine to access the resources of as much of the network as you see fit. There is very little you have to do to use MIDinet. By reference to a user, their computer is called local and all the others are called remote. The letter N is used to identify the network.

First; you have to give each computer in the network an exclusive number. This is achieved by entering the following command in the CLI of each machine connected to the network:-

MNETn	Where n is a unique number (an integer between 1 & 50) for the computer. A computer on the network can then be referenced by its network code; N followed by its network number, eg; N4

Second; you have to decide which machines are going to share their resources with the network. This is achieved by entering the following command in the CLI of each machine that you have decided will share its resources.

MSHARE	Only a computer designated as a "sharer" can be accessed by the other computers on the network. There is nothing special about a sharer other than its resources are accessible across the network. Any number of the computers on the network can be set to sharing mode.

The MIDinet includes some special facilities for file protection. Files beginning with (or in directories beginning with) *Hor *h will be treated as local only, ie. they cannot be read over the network. Any attempt to use these files will return the error: 'not found'. Files beginning with *R will be classed as Read Only, and can be read, but not written to, as if the device is write protected. Files beginning *D will return the error: 'not implemented' to prevent access by direct sector techniques. These facilities allow you to keep sensitive files on a particular computer without anyone on the network being able to access them. However, this facility can cause problems for software that does not expect access to be denied.

Manipulating files on a remote computer is the same as manipulating files on a local machine, but with [the computer's network number]_ put in front of the filename. eg:

nl_flpl_x	A file called x on floppy disc number 1 on the computer called n l.

Everything you can do with a file on a stand-alone machine you can do with files across a network of machines. You can read, write, modify, copy, move, delete, backup, update, execute, print, and view files across the network. You can use any application program to help you in this task. As you can imagine, the possibilities are endless. A few examples should be helpful.

Accessing devices across the network is just as easy. All you have to do is place the net number in front of a device number separated by and that device will be treated as if it was local, albeit with a performance governed by the network hardware. eg:

nl_par	The parallel port on nl.	
n2_ram2	A ram disc on n2. Serial	
n3_ser	port on n3.	
n4_con	A console on n4.	etc
	....	

One other facility extends this capability even further. If you execute a file on a remote machine and prefix the filename with an asterisk (*) then the remote machine will interpret this as an attempt to locally run a batch file of that name.

ex nl_ *raml_test	Will run a batch file called raml_test on that remote machine

As you could have written that batch file previously then this opens up the possibility of a multi-processor capability.

It is inadvisable to try and access a file or device with a netno. that is yourself. The system will take some time to work out that this does not make sense. This action is about as useful as trying to use a telephone to phone yourself. ie if you are using computer number nl do not try to manipulate a file or access a device called nl_something. When using an application network response will seem higher if you know what you are looking for. For example searching for a file on a large and fairly full hard disc is not a rewarding experience. If you only have two machines networked together then the above does not apply. In this case, it is no problem to have identical network numbers.

It is possible to abort network communication at any time, by pressing and holding the Shift key and then pressing the TAB key. This is useful because you may be aware of a network transmission error before the system.

The power of the MIDinet does not lie in some mysterious network code. It is a reflection of a number of key technologies built into **SMS2.**

Appendix

System Error messages:-

Incomplete
invalid Job ID
insufficient memory
value out of range
buffer full
invalid channel ID
not found
already exists
is in use
end of file
drive is full
invalid name
transmission error
format failed
invalid parameter
medium check failed
error in expression
arithmetic overflow
not implemented
read only
invalid syntax
read or write failed

System monitor error detection.

The system monitor can detect major damage to the system tables or structures. The system will wail (as in sound) and the system monitor display will go white if this happens. There is not much you can do about this except try and remove the job that caused the damage and try and save sensitive data before resetting the system.

Hotkey error detection.

The hotkey system will burp (again, as in sound) at you if the system runs out of memory or it is unable to find the object it is looking for.

Floppy disc error detection.

The floppy disc I/O system will ring at you if it cannot write a specific cached sector. This error occurs if the disc is damaged or you remove a disc whilst it is being written to. It is likely that data will have been lost if this happens.