SMSQ/E for Atari ST and TT

Introduction

From the point of view of the hardware dependent features, SMSQ/E as implemented on the Atari ST and TT series computers is very similar to the latest E level drivers for QDOS. There are two main changes. Firstly, monochrome monitors are supported. Secondly, the incorporation of the DV3 disk driver subsystem, which replaces the V2 disk driver used by the E level drivers means that Atari GEMDOS partitions of hard disks and GEMDOS (as well as IBM) format floppy disks may be read and written.

Machine Type

The two standard functions to determine the machine type are, of course, supported.

MACHINE

The MACHINE function returns the machine type.

0 ST / STM / STF / STFM.
1 ditto, with blitter.
2 Mega ST (without blitter) or ST etc. with real-time clock.
3 Mega ST (with blitter) or ST etc. with RTC and blitter.
8 Mega STE (without blitter!).
9 Mega STE.
24 TT 030

PROCESSOR

The PROCESSOR function returns the 680x0 family member - 0 or 30 for the Atari ST and TT series. The PROCESSOR function is provided in addition to the MACHINE function as it is possible to fit an MC68030 accelerator card in the lesser ST machines.

```
IF MACHINE < 24 AND PROCESSOR = 30: PRINT "68030 accelerator fitted"
```

Memory Protection

One feature of the ST series of computers is its memory access control. This causes a system error (access fault) if a program attempts to access memory which does not exist or which can only be accessed in supervisor mode (the vector area, the TOS system variables and the IO hardware).

Early versions of SMSQ on the ST series of computers detected legitimate accesses to the QL vector area but trapped all other memory access faults. This provided a certain measure of protection against the worst excesses of QL software. While this policy provided compatibility with well written, fault free, QL software, not much of the other 99% would work at all. A new policy has, therefore, been introduced.

1. All legitimate read operations from the QL vector area are allowed.
2. All other read operations from protected areas read 0.
3. All write operations to protected areas are ignored.

This policy can be applied to all Jobs or just to Job 0. If an access fault is trapped. The job goes into a state of hibernation with the fault program counter on

the stack and all other registers preserved. The Job may, therefore, be examined by a debugger.

PROT_MEM

The PROT_MEM (*level*) procedure sets the level of the memory protection. All legitimate accesses to the vector area are always allowed. Other access faults may be trapped or ignored depending on the level. The default level is 3 which will trap common faults in C programs, but allows certain famous system extensions to be LRESPRed. Cautious users should change this to level 7. Devil-may-care users should change it to level 0.

There are five levels: 0, 1, 2, 3 and 7.

- Level 0 does not trap any memory access faults.
- Level 1 traps write access faults in all jobs except Job 0. Read operations from a protected area read 0.
- Level 2 traps read access faults in all jobs except Job 0. Write operations to a protected area are ignored.
- Level 3 traps both read and write access faults in all Jobs except Job 0.
- Level 7 traps access faults in all Jobs.

```
PROT_MEM 0          Ignore all access faults - almost like the QL
PROT_MEM 1          Ignore all but write access faults from Jobs other than Job 0
PROT_MEM 7          Trap all access faults
```

POKES POKES_W POKES_L

POKES (*address, value*) POKES_W (*address, value*) and POKES_L (*address, value*) are the "supervisor mode" equivalents of POKE, POKE_W and POKE_L. By operating in supervisor mode they enable data to be written to the ST series IO hardware. Do not be surprised if your computer self-destructs when you use them.

PEEKS PEEKS_W PEEKS_L

PEEKS (*address*) PEEKS_W (*address*) and PEEKS_L (*address*) are the "supervisor mode" equivalents of PEEK, PEEK_W and PEEK_L. By operating in supervisor mode they enable data to be written to the ST series IO hardware. Do not be surprised if your computer self-destructs when you use them.

Atari ST and TT Displays

DISP_TYPE

The DISP_TYPE function is used to find the type of display adapter. For the Atari ST and TT computers, there are four values that may be returned.

0 Original ST QL emulator (this value is returned on QL based hardware).

1 Extended mode 4 emulator (standard and extended display sizes).

2 QVME mode 4 emulator.

4 Monochrome display.

```
ncol = 4                        Assume 4 colour display
if DISP_TYPE = 4: ncol = 2      If it is monochome, there are two colours only - grey and grey
```

Monochrome Display

If you have an ST series computer and there is a monochome monitor plugged in when you boot, SMSQ/E will automatically load the monochrome (ST high resolution 640x400) display driver. If you have a TT computer, and SMSQ/E does not find a QVME card, then SMSQ/E will set the TT to ST high resolution and load the monochrome display driver.

In either case, if you have not bought the monochrome display driver, you will not get a picture!

DISP_INVERSE

The DISP_INVERSE (*0 or 1*) command is used to invert the monochome display from the normal (black background) to inverse (white background) state.

DISP_INVERSE 1	*Invert black and white*
DISP_INVERSE 0	*Restore normal*

Colour Displays

SMSQ/E supports the old QL emulator card, in its original form and modified for MODE 8, the extended QL mode 4 card and the QVME card.

The extended QL mode 4 card may be switched from normal to extended display, and the display size of the QVME card may be changed at will.

DISP_SIZE

DISP_SIZE (*xpixels, ylines*) is used to set the display size. The nearest feasable size will be selected by the driver. It is best not to change the display size when the pointer sprite is visible, or you may get some spurious blobs left on the display. There should be few other problems changing from a smaller size to a larger size. You should, however, avoid changing from a larger size to a smaller if there are any windows outside the smaller screen. Note that, for the QVME card, the width and height are set in increments of 32 pixels and 8 lines respectively, while for the extended QL mode 4 card, any width of 512 or less will select the standard resolution mode while any width greater than 512 pixels will select the extended mode.

DISP_SIZE 800,600	*change to 800x600 (QVME) or 768x280 (extended mode 4)*
DISP_SIZE 1	*change to 512x256 (extended mode 4) (ignored by QVME)*

DISP_RATE

DISP_RATE (*frame rate, line rate*) is used to specify the frame and line scan rates for the QVME card only: it is ignored for all the other display cards. It would be usual to specify only the frame rate: the line rate is equal to the frame rate multiplied by the total number of lines.

DISP_RATE 75	*set the frame rate to 75 Hz*
DISP_RATE 70,48000	*set the frame rate to 70 Hz and line rate to 48 kHz (686 lines)*

DISP_BLANK

DISP_BLANK (*x blank, y blank*) sets the size of the blank area to the sides of and above and below the image for the QVME card only: it is ignored for all the other display cards. If the blank is too small, you will loose some of your image, if it is too large, the image will be too small.

DISP_BLANK 128,64	*set horizontal blank to 128 pixels and vertical to 64 lines*

As the display size is altered, the blank is automatically adjusted to maintain the proportion of blank. The DISP_BLANK command will not usually be required.

If preferred, the parameters for the DISP_RATE and DISP_BLANK commands may be tacked on to the DISP_SIZE command.

DISP_SIZE 640,480,60	*set standard VGA*
DISP_SIZE 800,480,80	*set 80 Hz refresh rate, squashed VGA.*
DISP_SIZE 800,600,70,,128,60	*set all of size, frame scan rate and x and y blank*

Note that if you specify both frame and line rates, as well as the number of blank lines, the line rate is over-specified: it will be determined by the frame rate and the total number of lines (visible + blank) and the line rate will be ignored.

DISP_SIZE Experimenter

The QVME card does not have an infinite choice of pixel rates. Some combinations of size and display rates may not be acceptable to your monitor. A small experimenter program can be used to change the size and frame rate in small intervals.

This program starts off with the standard VGA settings and adjusts the width when you press the W key, the height when you press the H key and the frame rate when you press the F key. Because you cannot see the display when you have struck an unsatisfactory combination, you can save a satisfactory setting whith the S key and restore it later with the R key.

If, for example, you are increasing the width (SHIFT W) and the display dissolves, DONT PANIC. Pressing the key a few more times may shift you past a bad patch. Alternatively, adjusting the frame rate up or down may improve matters. It is for you to find out what your monitor will accept.

The true hackers can add code to this program to adjust the blank as well.

```
100 REMark - This is an experimenter for the QVME display size and frame rate.
110 REMark
120 REMark - w reduces the width          - SHIFT W increases the width.
130 REMark - h reduces the height         - SHIFT H increases the height.
140 REMark - f reduces the frame rate     - SHIFT F increases the frame rate.
150 REMark - s saves the current settings
160 REMark - r restores the saved settings
170 REMark
180 REMark - ESC finishes
190 REMark
200 REMark - Set initial values for standard VGA
210 sw=640: sh=480: sf=60
220 a%=CODE('r')
230 :
240 REPeat
250   SELect ON a%
260   =27: STOP
270   =CODE('w'):  dw=dw-32
280   =CODE('W'): dw=dw+32
290   =CODE('h'):  dh=dh-16
300   =CODE('H'):  dh=dh+16
310   =CODE('f'):   df=df-1
320   =CODE('F'):  df=df+1
330   =CODE('s'):  sw=dw: sh=dh: sf=df
340   =CODE('r'):   dw=sw: dh=sh: df=sf
350   END SELect
360   DISP_SIZE dw,dh,df:         REMark - Set width, height and frame rate.
370   PRINT #1,dw,dh,df
380   BGET #1,a%
390 END REPeat
```

This program demonstrated that a perfectly legible 1600x496 (266 column) display was obtainable using a standard monochrome VGA monitor (cost about Dm200).

Serial (RS232) Ports on the Atari ST and TT Series

The number of serial ports depends on the model.

SER1	MODEM 1	ST/STE	Mega STE	TT
SER2	MODEM 2		Mega STE	TT
SER3	SERIAL 1			TT
SER4	SERIAL 2		Mega STE	TT

The ports themselves are connected to three different serial controllers of two different types. The communications speeds are, therefore, a bit special.

SER1
: The rates available on this port are sub-multiples of 19,200. All the standard rates from 300 to 19,200 are available except 7,200.

SER2
: The rates available on this port are sub-multiples of 250,000. 19,200 is *very* close to 250,000/13. All the standard rates from 300 to 19,200, including 7200 (within 1%) are supported. In addition it supports 1x and 2x MIDI speeds.

  If the rate is specified as 0, the rate used is 153,600 (19,200x8)

SER3
: The rates available on this port are sub-multiples of 19,200. All the standard rates from 300 to 19,200 are available except 7,200.

  Hardware handshaking is not available on this port.

SER4
: The rates available on this port are sub-multiples of 114,750. All standard rates from 300 to 38,400 are supported (within 0.4%) as well as 57,600 (19,200x3)

  If the rate is specified as 0, the rate used is 230,000.

Atari ST Printer Port

The Atari ST (and TT) printer port (the SMSQ/E PAR device) is notionally "centronics compatible". unfortunately a combination of very substandard drive capability on the part of the ST computers, excessive drive requirements of some printers (notably Canon) and long cables can significantly reduce the reliability of the printer connection. The problem can be reduced by extending the length of the strobe pulse.

PAR_PULSE

PAR_PULSE (*pulse length*) sets the notional pulse length in microseconds. The time will depend on the processor and the clock speed.

```
PAR_PULSE 50        drive a Canon printer from a standard ST
PAR_PULSE 500       . . . or from a Hypercache 030
```

Atari ST and TT Hard Disks

ACSI and SCSI Drives

Hard disks for the Atari ST and TT series computers come in two varieties: ACSI and SCSI. Although **most** drives attached to the ACSI bus will be full

standard SCSI devices, the SMSQ/E drivers assume that any drive connected to the ACSI bus does not necessarily conform to the SCSI CCS specifications so, normally, no attempt is made to do anything other than read or write sectors or read the error status on these devices. This means that, for example, the drivers cannot detect whether an ACSI disk drive has a removable cartridge.

On the other hand, the SMSQ/E drivers assume that all drives connected to the SCSI bus (TT only) conform to the minimum CCS specifications for hard disk operations. Any disk drive which responds "OK" to a request to lock the door is considered to have a removable cartridge.

If a file is open on a removable cartridge, the door is locked. It will be unlocked automatically later.

WIN Drive Numbers and Name

ACSI and SCSI drives are identified by a whole series of numbers: the "target" number, the "unit" number and the "partition" number. The target number is the identification number of the disk drive controller. For internal drives, this is 0. For external drives, this is the number (0 to 7) that you set on the little switches on the back of the box. The unit number selects one of a number of drives controlled by a single controller. It is possible, but rare in the Atari world, for a controller to have up to 8 units. In general, there is only one unit per controller, and 99% of Atari hard disk utility software assumes that you can only have one unit per controller, so the unit number is usually 0. Finally, the partition number defines a section of the disk reserved for a particular purpose (e.g. GEM partitions, QDOS partitions etc.).

GEMDOS numbers its target, unit and partitions from 2 (=C) as it finds them. This is a superficially attractive scheme which collapses completely if you have removable media with different numbers of partitions or if the medium is not in the disk drive when you boot the computer.

SMSQ/E adopts a more cumbersome approach which is, however, much more precise. Unless you configure SMSQ/E to boot from a target and partition other than 0,0, the initialisation code will attempt to find a file called "BOOT" on any partition on target 0. (For the TT, SMSQ/E will try SCSI 0 first and then try ACSI 0). WIN1 will be set to this partition. Thereafter, you must define your own WIN drives for any other target, unit and partition you wish to access.

WIN_DRIVE

WIN_DRIVE (*drive, target, unit, partition*) is used to select a particular target, unit and partition combination to be accessed using a particular WIN drive.

If an SCSI drive is to be accessed, 8 should be added to the target number. The unit number may be omitted or both the unit and partition numbers may be omitted.

WIN_DRIVE 2,1,0,2	*WIN2 is ACSI target 1, unit 0, partition 2*
WIN_DRIVE 3,9	*WIN3 is SCSI target 1, unit 0, partition 0*
WIN_DRIVE 4,3,1	*WIN4 is ACSI target 3, unit 0, partition 1*

Issuing a WIN_DRIVE command for a particular drive will cause the drive map to be re-read the next time the disk is accessed. It can, therefore, be used to force the drivers to recognise a disk change.

WIN_DRIVE$

WIN_DRIVE$ is a function which returns a string giving the target, unit and partition used by a particular WIN drive.

```
WIN_DRIVE 2,1,0,2        WIN2 is ACSI target 1, unit 0, partition 2
WIN_DRIVE 3,9            WIN3 is SCSI target 1, unit 0, partition 0
PRINT WIN_DRIVE$(2)          Prints 1,0,2
PRINT WIN_DRIVE$(3)          Prints 9,0,0
PRINT WIN_DRIVE$(4)          Prints nothing if WIN4 has not been set
```

WIN_USE

WIN_USE may be used to set the name of the WIN device. The name should be 3 characters long and in upper or lower case.

```
WIN_USE MDV              The WIN device is renamed MDV
WIN_USE win              The WIN device is restored to WIN
WIN_USE                  The WIN device is restored to WIN
```

Handling ACSI Adapter Timing Faults

Certain ACSI adapters exhibit a timing fault. If commands are issued too quickly one after the other, the adapter fails. The SMSQ/E ACSI driver can be slugged to bring its interval between commands down to GEMDOS levels.

WIN_SLUG

The WIN_SLUG (*value*) command sets the minimum time that must elapse between operations on the ACSI bus (in units of 80 µs). ICD recommend 1 ms for their adapters. As an interval of 2.5 ms between operations has proved adequate for most adapters, this is the default. As the typical access times for ACSI hard disks are of the order of 20 ms to 30 ms, this does not represent a large overhead.

```
WIN_SLUG 12              Wait at least 12*80 µs between ACSI operations
WIN_SLUG 30              Wait at least 30*80 µs between ACSI operations (default)
```

FORMAT WIN

As SMSQ/E is "hosted" on the Atari ST and TT computers, it only takes control of and formats partitions on the hard disk which you have previously marked as being reserved for QDOS compatible disk drivers. We know that **you** would not destroy all your GEM desktop publishing files by formatting a QDOS disk on top of them, but someone else might do it.

Before formatting a QDOS compatible partition, therefore, you will need to use your favourite GEM utility to make a suitable partition available, marking it as "QWA" (GEMDOS partitions are identified by the letters "GEM" or "BGM").

Before formatting a WIN drive with SMSQ/E, it is necessary to define the ACSI or SCSI target number (and the unit number if it is not 0) and partition.

```
WIN_DRIVE 2,1            Set WIN2 to ACSI target 1, unit 0, partition 0
FORMAT win2_Fred         and FORMAT it

WIN_DRIVE 1,8,2          Set WIN1 to internal TT drive, partition 2
FORMAT win1_BOOT         and FORMAT it
```

WIN Control Commands

The rest of the commands specific to the Atari ST and TT WIN device control or set the characteristics of a specific WIN drive.

WIN_WP

WIN_WP (*drive, 0 or 1*) is used to software write protect a WIN drive.

WIN_WP 1,1	*Set the "write protect" flag for the drive accessed by WIN1*
WIN_WP 1,0	*Clear the "write protect" flag for the drive accessed by WIN1*

WIN_START
WIN_STOP

The WIN_START (*drive*) and WIN_STOP (*drive*) commands may be used to start and stop a drive. If you issue one of these commands for an ACSI drive, the drivers may assume that the drive will accept other SCSI control commands.

WIN_STOP 2	*Stop the drive accessed by WIN2*
WIN_START 2	*Start the drive accessed by WIN2*

WIN_REMV

WIN_REMV (*drive, 0 or V*) is used to notify that the target accessed by the WIN drive has a removable medium. WIN_REMV is ignored for SCSI drives where this is detected automatically. No parameter is required to mark the drive as being a standard removable device. A "V" marks the drive as a VORTEX naughty drive. A "0" cancels the removable medium flag.

WIN_REMV 2	*Set the "removable" flag for the drive accessed by WIN2*
WIN_REMV 2,0	*Clear the "removable" flag for the drive accessed by WIN2*
WIN_REMV 3,V	*Set the VORTEX flag for the drive accessed by WIN3*

Atari ST and TT Floppy Disks

Most of the models in the ST and TT range are equipped with a single DD 3.5" floppy disk drive. Some of the later models are equipped with an HD drive. The SMSQ/E FLP driver can read or write QL5A, QL5B, TOS and MSDOS format diskettes. It can format QL5A (DD) and QL5B (HD) format diskettes.

Floppy Disk Driver Name

The default name of the floppy disk driver is FLP. The internal drive is FLP1. The external drive (if any) is FLP2.

FLP_USE

FLP_USE may be used to set the name of the FLP device. The name should be 3 characters long and in upper or lower case.

FLP_USE mdv	*The FLP device is renamed MDV*
FLP_USE FLP	*The FLP device is restored to FLP*
FLP_USE	*The FLP device is restored to FLP*

FORMAT FLP

The SMSQ/E FLP driver will usually format a diskette to the highest density it can. The density may, however, be set using the FLP_DENSITY command or by adding a special code to the end of the medium name in the format command.

FLP_DENSITY

The SMSQ/E format routines will usually attempt to format a disk to the highest density possible for a medium. The FLP_DENSITY (*code*) is used to specify a particular recoding density during format.

The density codes are "S" for single sided (double density), "D" for double density and "H" for high density.

FLP_DENSITY S	*Set the default format to single sided*
FLP_DENSITY H	*Set the default format to high density*
FLP_DENSITY	*Reset to automatic density selection*

The same code letters may be added (after a *) to the end of the medium name to force a particular density format. (For compatibility with older drivers, if the code letter is omitted after the *, single sided format is assumed.

FORMAT 'FLP1_Disk23'	*Format at highest density or as specified by FLP_DENSITY*
FORMAT 'FLP1_Disk24*'	*Format single sided*
FORMAT 'FLP1_Disk25*S'	*Format single sided*
FORMAT 'FLP1_Disk25*D'	*Format double sided, double density*

FLP_TRACK

The FLP_TRACK (*number of tracks*) is used to limit the number of tracks formatted.

FLP_TRACK 23	*Only format 23 tracks*

FLP Control Commands

FLP_SEC

FLP_SEC (*level*) was used to set the security level. The security of the data stored on the diskettes can be seriously compromised if you change diskettes while there are files open. The security level affects the amount of time the FLP driver spends maintaining the data on the diskette up to date with the internal copies of the data in memory. In principle, a lower level is more efficient, but more risky. With the increasing use of hard disks, the security level of the FLP has been fixed at level 2: the most secure. FLP_SEC is ignored.

FLP_START

The FLP_START (*ticks*) command specifies the number of ticks (1/50th of a second) that the FLP driver waits after starting the drive before writing to it. This allows the diskette to get up to speed before the write operation. The default value is 24, which is a wait of about 0.5 s. There should not be any reason to use this command.

FLP_STEP

In the days when QLers used scrap 5¼" disk drives with 30 ms step rates, FLP_STEP allowed the disk drive step rate to be set. In the 5 years that the Atari drivers for QDOS were distributed, no-one ever complained that FLP_STEP was completely ignored. It still is ignored in the SMSQ/E drivers.