SuperCharger

CONDOR

USER MANUAL

Version 1.4

Dear Customer,

The SuperCharger you have purchased brings PC and MS-Dos standards to the Atari ST range. It is the ultimate PC solution for Atari users, offering;

easy installation - no need to modify your ST-range system

excellent performance (Norton factor 4.4) from

an elegant hardware/software combination

full concurrent processing - run TOS and MS-Dos applications at the same time

You now have the power of Atari with the flexibility of an MS-Dos environment. The PC world is at your fingertips - enjoy it!

Condor-Beta

Your team at Condor/Beta

PS We can only provide you with new software releases if we know where you are so please return the enclosed Registration Card now.

contents: section I - Installation

1.0 Pre-installation procedure

First, copy protect all 3 disks, then take a copy of your original utility disk, as *the installation procedure makes changes to files on the Utility disk which cannot be* reversed.

Before you continue, please read parts 1.0 to 5.0 of this section of the manual and the section on fault diagnosis (section 9.0). If you're a real hot-shot then you can then try installing the SuperCharger. If you're like the rest of us, then read through this entire first section of the manual before installation and, if you're new to MS-Dos, read through section II - the brief MS-Dos guide.

Note: The file README.1ST (On the utility Disk) will give you the latest information about any software updates not yet covered in this manual.

Where there is any discrepancy between README.1ST and this manual, it is the Utility Disk which you should follow. For this reason, you should never mix files from different releases of SuperCharger software - always install a complete system and fully delete the old one.

In order to ensure maximum performance from your SuperCharger you first have to set up the system parameters before starting the SuperCharger.

Insert your Utility disk copy and start the program LANGUAGE.PRG. You will then be asked to select the working language for SuperCharger. Should you wish to abort, an opportunity will be given.

After processing is complete, you will be informed of a successful translation. Click OK. to exit the translation routine.

We now recommend you re-read the README.1ST file before you go ahead and install your SuperCharger.

2.0 Computer Setup

From your Utility disk, open the ABIO folder and start the INSTALL.PRG program. You will then see your system parameters displayed at the bottom of the screen and a menu bar at the top.

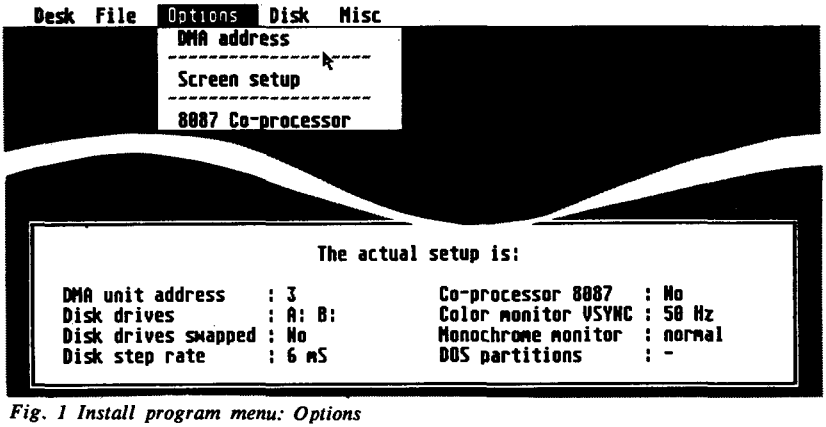


Fig. 1 Install program menu: Options

Options Menu

8087 Co-Processor

If you have a maths co-processor, select "OPTIONS" from the menu bar and then choose "8087 co-processor". This will tell the system that you have the co-processor. (The standard SuperCharger comes without - you can fit one later if you wish). Default is NO.

Fig. 2 8087 selection

Screen Set Up

Define the screen refresh rate and whether inverted video or not. This is important for colour mode Set to 50 Hz for UK, 60hz for USA.

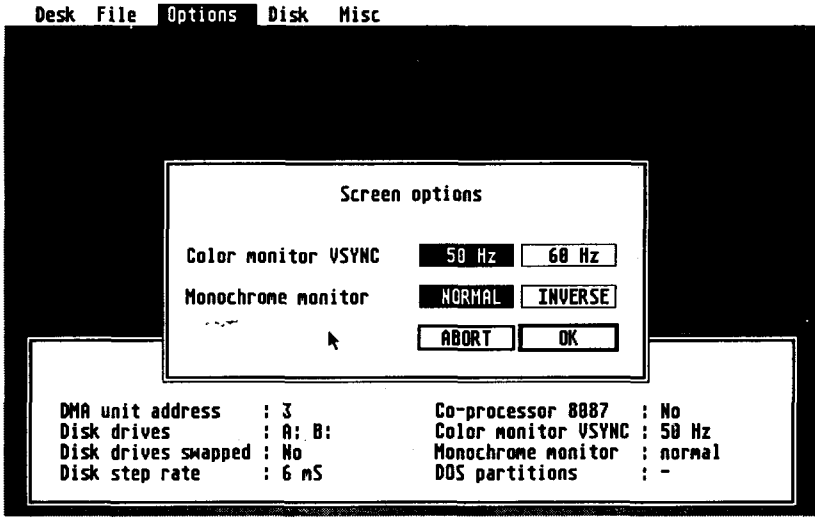


Fig. 3 Screen options

DMA Address

DMA address adjustment will only be required if you have to change the DMA address of the SuperCharger. This is not normal unless you have two SuperChargers running off the same ST or unless you have another device with a conflicting DMA address. If you change the address, you have to change the jumper in address block X5 on the SuperCharger (see Appendices A & B). The default is set to address 3.

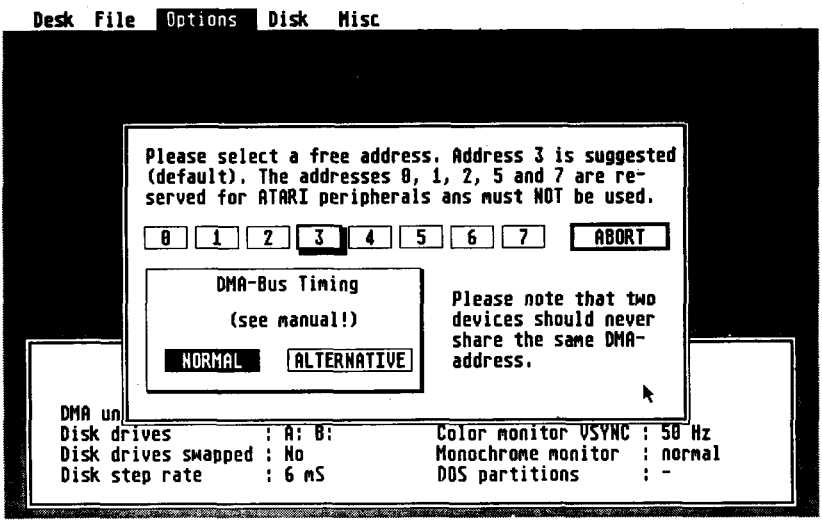


Fig. 4 DMA bus settings

There is also an option which enables you to adjust the DMA bus timing. Users of certain (very few) hard drives may need to set this in order to ensure compatability with their disk. This applies particularly to early models of the Vortex HD-Plus series. Start off with the default NORMAL. Use ALTERNATIVE only if you encounter hard disk compatibility problems.

Step Rate

You will need to alter "Steprate" only if you have connected an external 5.25" drive or if you have one of the older, slower disk drives. Always install the steprate of your slowest disk drive. Default = 3 ms.

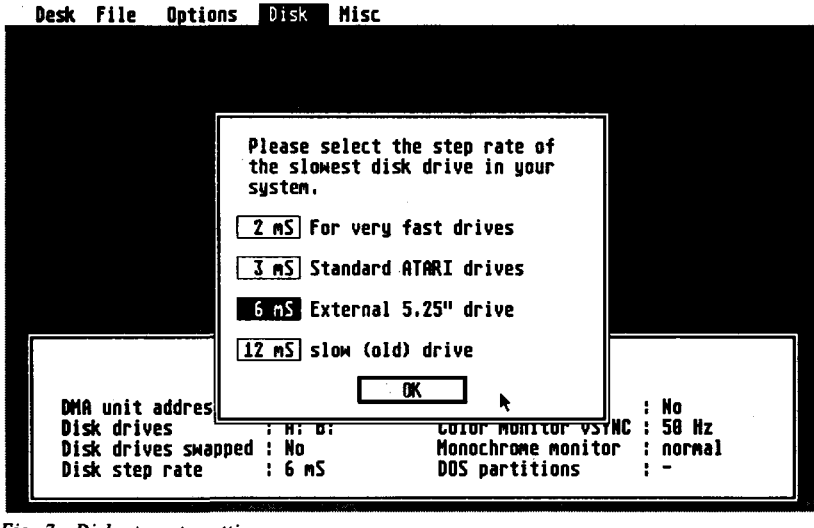


Fig. 7 Disk steprate setting

"Misc" menu

Fig. 8 The "misc" menu

Make Dos bootable

This function is used in hard-disk installation of MS-Dos. Ignore at this stage.

Install Launcher

The Launcher enables you to start MS-Dos programs directly from TOS, providing the SuperCharger desk accessory (LAUNCH.TTP) is loaded.

The installation screen shows the types of file launcher will start; (EXE, .COM, .BAT) and the search path for LAUNCH.TTP which should always be in the root directory on the boot device. Click OK. and INSTALL will modify your Desk top information.

Note, Installing LAUNCHER here sets up only the parameters - it does not copy the file for you. You may have to install LAUNCH.TTP into your desk top menu.

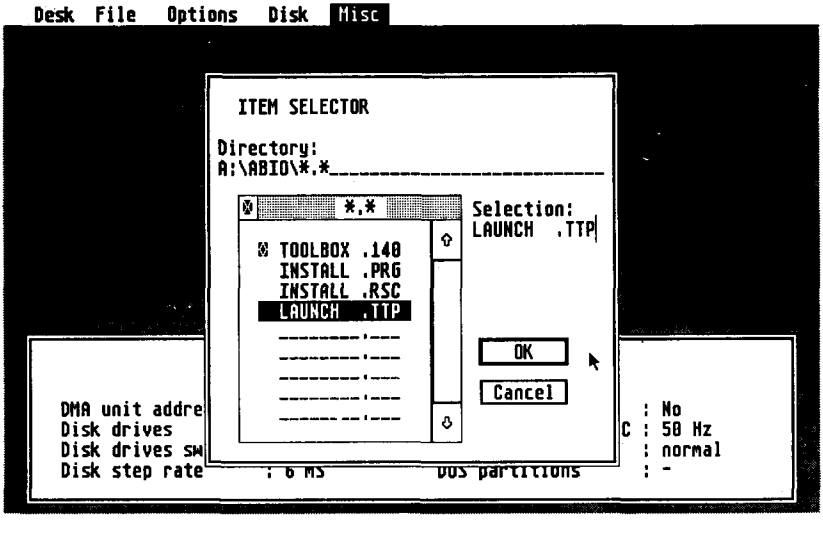


Fig. 9 Installing LAUNCH.TTP

You can now leave the INSTALL program (menu bar option "FILE"). Select "QUIT AND SAVE".

After completing INSTALL.PRG you must ensure that ABIO.ACC and ABIO.DAT are copied to your Atari's boot or hard disk. If you have installed the launcher in the "misc" menu above you should also copy LAUNCH.TTP from the ABIO folder on the utility disk to your Atari's boot or hard disk root directory.

Note: If you leave INSTALL without saving the file, the information relating to disk standards, screen characteristics, DMA addressing and co-processor is not saved.

After you have saved your settings "write protect" your utility disk copy.

3.0 Connecting up the SuperCharger

Turn off every component of your system

Eliminate any potential conflicts of DMA address. Your SuperCharger is preset to address 3, but can be changed. Refer to Appendix 3, this section. Connect the DMA cable to either of the SuperCharger's DMA ports (there is no distinction between DMA IN and DMA OUT on the SuperCharger).

Connect the other end of the cable to the port marked HARD DISK on the back of the Atari. If this port is occupied (for example by a hard disk), then see if you have another port marked HARD DISK OUT or DMA Out on the hard disk (SH-205 or MEGAFILE are examples) where you can connect the DMA cable. The system should now look like this:

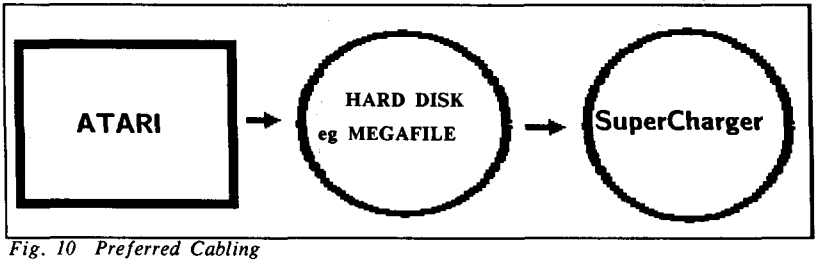


Fig. 10 Preferred Cabling

If your device doesn't have such a spare port then unplug your hard disk (or other peripheral) from the Atari and connect the SuperCharger as the first peripheral on the hard disk port. Now use the DMA Out port (ie the other port) on the SuperCharger for connecting your hard disk or other peripheral. See over:

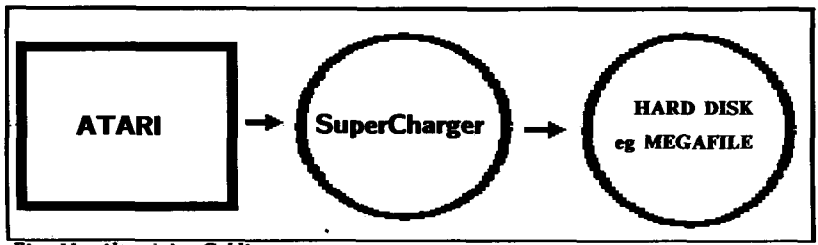


Fig. 11 Alternative Cabling

Now ensure the SuperCharger is turned Off. Plug the Power Supply cable into the back of the SuperCharger and into the mains supply.

Switch on the SuperCharger. The led marked Power should glow. The led marked Data may also glow at this stage. Now press the RESET button on the front of the SuperCharger and switch on your Atari after which the data light should go out. This procedure must be carried out every time you switch your system on. A connected SuperCharger should always be switched on first to ensure correct operation on the DMA-bus.

4.0 MS-Dos Installation

Before installing MS-Dos on the Hard disk, you must have first set up and initialised an appropriate partition using INSTALL.PRG on the Atari. (See 1.0 Installation).

The SuperCharger allows MS-Dos to be loaded from the floppy disk or the Hard disk, when you start the MS-Dos system. You may therefore use a second floppy disk or any hard disk partion you have available.

If you have a hard disk, the SuperCharger will boot from the installed boot partition as long as the selected disk A: is empty.

If a bootable MS-Dos disk is in the floppy A: drive SuperCharger automatically tries to start the system from the floppy.

If however the MS-Dos disk turns out not to be bootable, the SuperCharger will display a message and ask if it should attempt to reboot.

Hard Disk MS-Dos Installation

If you have a hard disk, you must now use TOS to create a partition on your hard disk from which to boot MS-Dos. This partition must be less than 16 Mb and over 2Mb in size.

Note: If your hard disk has already been partitioned you can format an existing partition for MS-Dos, but note you will lose all existing files in that partition.

Don't forget to back up your disk before initialising a partition.

This partition must never be entered via TOS or you will lose all your data. To ensure this does not happen, delete the Icon for the MS-Dos partition from the TOS desktop.

Having created the partition , return to INSTALL.PRG on your Utility disk (ABIO folder)and select "Misc" from the menu bar. Open "Make DOS Bootable". Select your relevant partition and follow the on screen instructions.

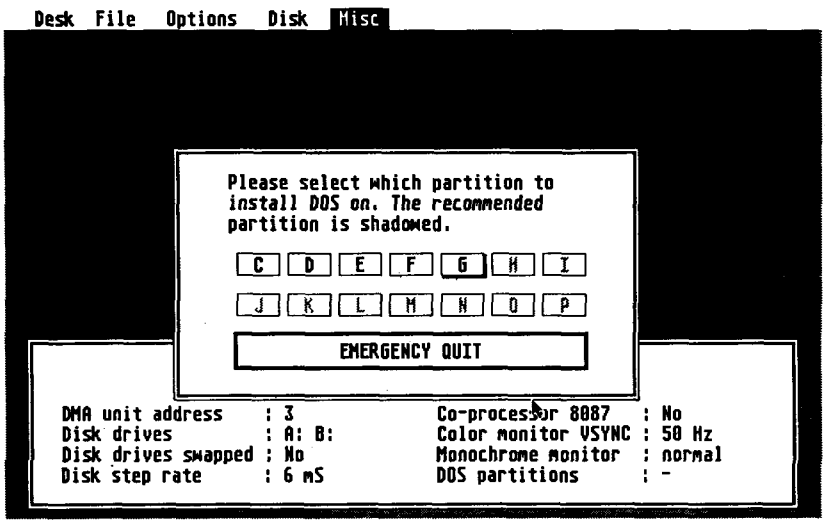


Fig. 12 Hard disk boot using INSTALL

"Now select "Quit" and "Save" to exit from INSTALL..

The new hard disk partition is now ready to accept MS-Dos system files. Read section II, "Introduction to MS-Dos" to learn how an MS-Dos system disk is created.

The partition may be reactivated for use under TOS using any Atari hard disk installation program though any data will, of course, be lost.

Start ABIO.TOS from your Utility disk.

After a short wait, you will see the initialisation message on the screen and the Data led will indicate that data is being transfered between the Atari and the SuperCharger. When requested, remove the Utility Disk and replace it with the MS-Dos Boot disk. Press "Enter".

Loading MS-Dos

After a moment, MS-Dos will load and the system prompt will appear. Re insert the Utility disk and start the MS-Dos installation program by typing SETUP.

The program will ask you which language you will be operating MS-Dos in. Choose UK for Great Britain or English for the USA, and the system will load the appropriate keyboard driver next time the SuperCharger is started.

The SETUP program then offers the ability to install MS-Dos on either the hard disk or to create a workdisk floppy, or you may exit the program. Select A to format your selected partition. SETUP will now request the SuperCharger boot disk.

Formatting the disk will take some time depending on the capacity and speed of the drive used. After formatting, MS-Dos asks for a "Volume Name" for the hard disk. Select a name of your choice or just [Enter] if you don't need it named.

SuperCharger will automatically load in the DOS files. You will then be asked for the "Betriebssystem" (Operating system) disk and the Utility disk. After following all the on-screen instructions, return to the menu and select [x] to exit. Remove the disk in drive A:

Installation details will only be effected when the SuperCharger has been rebooted. Now is a good time to do this. Remove the floppy from the drive and enter ALTERNATE+CONTROL+DELETE. The SuperCharger will reboot. There is no need to reboot the Atari.

Return to TOS by using the Hot key ALTERNATE+CONTROL+BACKSPACE. This will reboot the Atari if your SuperCharger was started by ABIO.TOS

You can now copy the SuperCharger utilities from the Utility disk to the hard disk. Set up a file on a TOS partition and copy the utility disk across. Having done that you can in future start MS-Dos from this partition by running ABIO.TOS.

Note: If you want to use the SuperCharger as a desk accessory, ABIO.DAT and ABIO.ACC have to be in the Root Directory of the TOS boot partition (or disk) together with LAUNCH.TTP if you have selected the launch facility in the installation routine.

Now re-boot your computer before continuing

At this stage you should use the MS-Dos command DISKCOPY to take copies of both of your SuperCharger MS-Dos disks (See Section II of your manual for details of DISKCOPY).

Floppy Disk MS-Dos installation

Because not all the necessary programs are on the Boot disk you will need to set up an MS-Dos working disk . For this you need an unformatted 3.5" 720kb diskette.

REMEMBER: Write protect all original disks.

From the Utility disk select ABIO.TOS.

Insert the Boot disk when requested and at the DOS prompt A: re-insert the Utility disk and type "SETUP" then follow the on-screen instructions.

Note: Select "U" for Great Britain or "E" for US. Then select "B" to create an MS-Dos Work disk.

Insert Boot disk when requested and press RETURN. Have your 720 Kb unformatted disk ready. Remove the Boot disk and insert the "BETRIEBSSYSTEM" (Operating system) disk when requested. Press RETURN.

Note: When copying: SOURCE = boot disk, TARGET = empty disk

Replace the "BETRIEBSSYSTEM" disk with the MS-Dos Boot disk. The Boot disk will then be copied. When the "TARGET" disk is requested, insert your unformatted 720 Kb disk.

Repeat this procedure until completion.

DOS 4.01 will then automatically allocate a disk serial number and ask if you wish to copy another disk. Answer "N" (No).

After copying, ensure you have a copy of the boot disk in your floppy, not the original.

SuperCharger will then configure your Work disk and request the "UTILITY" disk for the remaining files to be copied.

When requested insert your new "WORK" disk, and the "BETRIEBSSYSTEM" disk as requested when finally your new "WORK" disk will be complete.

Note: Your Work Disk contains all essential non-resident DOS commands. It does not contain all external DOS commands. Should you require any of these, then use a copy of your original disk.

If you decide against using the MS-Dos 4.01 command shell you will gain the space otherwise used by "DOSSHELL", enabling you to fit all DOS commands on one disk.

Exit setup [x]

Note: If you want to use the SuperCharger as a desk accessory, ABIO.DAT and ABIO.ACC have to be in the Root Directory of the TOS boot partition (or disk) together with LAUNCH.TTP if you have selected the launch facility in the installation routine.

Now re-boot your computer before continuing

At this stage you should use the MS-Dos command DISKCOPY to take copies of both of your SuperCharger MS-Dos disks (See Section II of your manual for details of DISKCOPY).

5.0 Up-dating From Previous Versions Of Supercharger

WARNING: IF YOU HAVE ALREADY INSTALLED A PREVIOS VERSION OF SUPERCHARGER SOFTWARE DO NOT USE "SETUP" ON THE NEW V1.4 UTILITY DISK. SETUP is only for complete installation and will re-format the MS-Dos partition, losing all your existing data.

Hard disk systems

First read the instructions on hard disk MS-Dos installation in part 4.0 of this section of the manual.

Remove ALL files installed from your earlier version of SuperCharger utilities, especially ABIO.TOS and ABIO.BAT.

Copy from the new utility disk ABIO.TOS, ABIO.DAT, and ABIO,ACC into your Boot partition. Also copy across the ABIO folder.

Open the ABIO folder and start the INSTALL.PRG (see chapter 4.0, this section) to tell SuperCharger about your system configuration.

Do **not** initialise MS-Dos Boot Partition.

Now quit and save.

From the Boot folder, click on ABIO.TOS to launch SuperCharger, Using the copy commands copy from the Utility disk the following files to replace those existing in the original Root directory:

ALERT.EXE

ATARIDR.SYS

ATARIDSK.COM

JUMPHOT.COM

MSMOUSE.COM (Only if not existing)

NOTE: If your disk does not contain these files, read chapter 4.0, this section.

Finally, using a text editor or EDLIN commands, modify the following in the CONFIG.SYS file if you wish the Floppy drive A: to read, write and format 720Kb disks as the default format.

Existing line	DEVICE=driver.sys/d=0/t:80
New line	DRIVEPARM=d:0/c/i/t:80

Now re-boot your machine to install the modifications in both the TOS and DOS sectors.

NOTE: If you do not wish to use SuperCharger as a desk accessory, remove file ABIO.ACC from your boot partition.

Floppy disk systems

Refer to the installation instructions regarding floppy disks (Chapter 4.0, this section) to create a new work disk.

Note that you must already have changed your CONFIG.SYS file to read and write 720kb diskettes as the default format in order to successfully run the MS-Dos DISKCOPY command. See this page, above, for details of the change to Config.sys.

6.0 Ramdisk and the SuperCharger

Ramdisk(s) under MS-Dos.

When you create a Ramdisk, you do not have to lose any of SuperCharger's memory as you can use the memory of your Atari. The first 512K of Atari ram will be used for both TOS and SuperCharger programs, leaving any remaining memory free for a Ramdisk.

In the "RAMDISK" folder on your utility disk there is a ramdisk program from Germany that you may wish to use. It is documented in the read me file called LUFTSCHLOSS.TXT in the same folder.

NOTE: Always keep 512k RAM free for TOS and SuperCharger software!

There are numerous Atari-specified programs which you may use as ramdisk. However, some won't work with MS-Dos 4.0, as in order to be acknowledged as a logical disk by DOS v.4, the program must accept the Dos 4.0 12-bit FAT format.

Ramdisk under TOS

In order to use the memory of the SuperCharger as a RAMdisk under TOS, the program SCTB.PRG is located within the ABIO/TOOLBOX140 folders. (This program may be placed in the "Auto" folder). Double click on the SC_RAM.TTP option.

The default, (without entering any parameters) is a RAMdisk with 1 Mb maximum storage, accessible as a desk Icon "I". A RAMdisk with maximum storage is to be found at the DMA Address 3 as Floppy 1, whenever you start a program without parameters.

You can use several SuperChargers at the same time by inserting the correct parameters. For further details see the Toolbox Section of this Manual (section III).

7.0 Entering SuperCharger mode

You may enter SuperCharger in one of two modes:

1) Using Desk Accessory

   or

2) Using ABIO.TOS

To use SuperCharger as a desk accessory, ensure ABIO.DAT and ABIO.ACC are in your TOS boot or hard disk.

Using Desktop

Select "Desk" and click on SuperCharger. SuperCharger will then install. You may return to TOS using the "Advanced Hot key" combination SHIFT+CONTROL+BACKSPACE.

This combination will leave any MS-Dos program running in the background while you load an Atari program.

You may now hot key via the "Desk" accessory and key combination to switch between both MS-Dos and TOS progarms - true multi-tasking.

Alternatively by using the hotkey combination ALTERNATE+CONTROL+BACKSPACE you may freeze the MS-Dos program and return to TOS.

NOTE: In order to use the Advanced Hotkey or the Launcher MS-Dos and TOS must be synchronised which means that the SuperCharger **must** be started under the desktop accessory **not** via ABIO.TOS when using these facilities.

ABIO TOS

Launch SuperCharger by clicking on to ABIO.TOS and SuperCharger will load MS-Dos.

You may still use the normal (freeze) hotkey combination (ALT+CONTROL+BACKSPACE) but you must close your TOS application before re-entering SuperCharger. You should also make a point of closing any non reset-proof ramdisk, as the Atari will reset when you come back by using Hotkey.

NOTE: When in TOS You risk losing data if you access any file which may be in use under MS-Dos.

By launching from ABIO.TOS, you will save 2-300K of your ATARI ram.

The Toolbox

The SuperCharger TOOLBOX contains functions which enable programmers to speed up and to enlarge programs under TOS by the parallel use of both processors in the ATARI and in the SuperCharger. The programmer even has the option of using several SuperChargers working in parallel and with co-processors if required.

For example, you can use the RAMdisk and see both sides of the operation - the ATARI side and the SuperCharger side. For details, please refer to Section III of this manual - TOOLBOX.

The Launcher

The Launcher allows you to start MS-Dos programs under TOS (endings .EXE, .COM and .BAT) with a mouse click. It is no longer necesssary to boot MS-Dos manually, providing SuperCharger has already been activated. MS-Dos automatically returns to TOS when the program is finished.

NOTE: If SuperCharger is not started with the Desk Accessory, there will be a warm start when returned to TOS, resulting in loss of all programs and data.

JUMPHOT.COM

In order to enable you to leave your MS-Dos program running, the JUMPHOT.COM program (on your Utility disk) allows the same function under MS-Dos as the ADVANCED HOTKEY - a jump to TOS while MS-Dos continues to run. Of course, the SuperCharger has to be started as an Accessory.

ALERT.EXE

In many instances, you may want to know the status of MS-Dos programs running in the background. This status is given by ALERT.EXE (utility disk) on your TOS screen. ALERT.EXE avoids switching from TOS to MS-Dos just to see where the program is. It is an .EXE program which can be called from within any process which flashes a message of your choice on the TOS screen.

If you create an MS-Dos batch file including the command ALERT followed by your message, then JUMPHOT i.e;

```
JUMPHOT
ALERT This is a report
```

when you return to TOS with the batch file running, your message will appear on screen advising you when MS-Dos has completed its operation. See also the ALERTEST.BAT MS-Dos batch file on your Utility disk.

If you booted the SuperCharger as Accessory and started ALERTEST under TOS via the launcher, the report automatically returns to TOS.

8.0 Other SuperCharger Features

The Atari Laserprinter

You may, of course, use your ATARI laserprinter under MS-Dos. You just install the laserprinter under TOS and then start your SuperCharger using ABIO.TOS or the Desktop Accessory.

To make use of print emulation from MS-Dos, you have to install the printer emulator under TOS before you start MS-Dos for the first time. To print graphics you will require at least 2Mb of RAM in your Atari.

When installing the ATARI Laserprinter with the SuperCharger, we advise that you install the SuperCharger as the last device on the DMA chain:

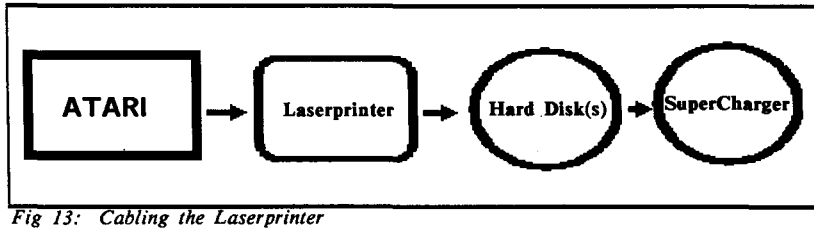


Fig 13: Cabling the Laserprinter

The sequence to follow when switching on the configuration is:

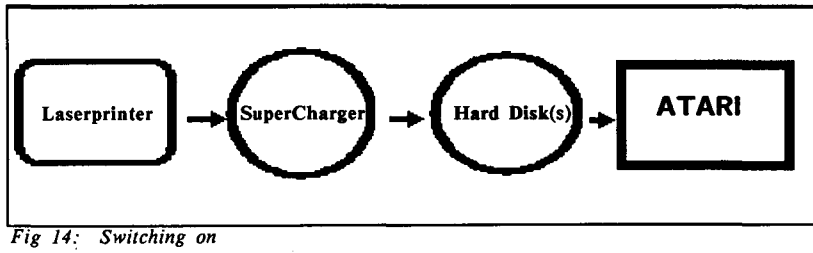


Fig 14: Switching on

NOTE: Some older laser printers have to be modified so that they can be addressed as device No. 6 if they don't work with the SuperCharger.

Additional Memory Under MS-Dos

The SuperCharger has 704kb available under MS-Dos; whereas the original PC has only 640kb. You therefore have more resident utility programs available in your memory. Some PD-software programs even have the ability to enlarge the memory to 896kb!

Note: please test your software to see if it runs with any such DOS-extender first.

Monochrome To Colour Screen

You have the ability to switch from monochrome/multisync monitor to colour monitor under TOS. SuperCharger will adapt to the new screen format when you return via the hotkey.

NOTE: A monochrome monitor can be switched to inverse printing by holding down ALTERNATE, then S, then A, at the same time.

Intel 8087 Co-processor

Like an original PC, the SuperCharger offers the opportunity to install a co-processor to speed up all mathematical programs such as CAD programs and programming languages.

When you install your co-processor, you will find an empty socket in the SuperCharger next to the V30 processor. Install an 8087-2 in the empty socket, making sure that the notch on the co-processor is lined up with the notch on the socket. The speed of the co-processor must be 8 Mhz.

When you have installed the co-processor, re-open the INSTALL.PRG. Select "Options" 8087 co-processor and select "YES". Quit and save.

The Atari hardware under MS-Dos

Basically, there are no restrictions in using the ATARI hardware under MS-Dos: use them as if they were PC-world equivalents.

The following table shows the equivalents of the input/output devices on the PC. On the PC these devices are named with an acronym such as COM1, LPT1, CON. See Section II of this manual for a more detailed description of MS-Dos.

Atari/PC Equivalents

Atari	PC
Keyboard S	Standard keyboard. 10 function keys above, 10 key cursorblock .
Mouse	Mouse systems compatible mouse (PC mouse). 2 keys used, connection on COM1.
Diskdrive 3.5	Double-sided floppy 3.5"/720Kb. ATARI-alike. Disks are directly exchangeable.
Colour Monitor	Full picture of 640 x 200 pixels CGA mode (RGB screen): 640 x 200 with 2 colours 320 x 200 with 4 colours
Monochrome Monitor	In CGA mode this is treated as above, but colours will be displayed as patterns. Uses an 8x16 character matrix and double-scan display in graphic mode to use full screensize. In Hercules mode the video display conforms to Hercules Graphics standards with 720 x 350 pixels.
Hard disks	You can run any partition via (C, D, E,etc) after partitioning/installation with the INSTALL programs supplied via "ATARTDR.SYS".
Parallel Port	LPT1 is installed on this port.
Serial Port	Installed as COM2.

9.0 Compatibility

Your SuperCharger combines hardware and software technology to produce the highest possible level of PC compatibility. Even PC programs which for maximum speed run directly on the PC hardware (instead of indirectly via the BIOS), will function on the equivalent ATARI hardware.

Given the differences in design concept between PC and ATARI hardware, there will always be some minor barriers ito complete compatibility.

Graphics

The maximum graphic resolution of ATARI computers in monochrome mode is only 640 x 400 pixels.

The use of HERCULES graphic mode with 720 x 350 pixels therefore means that you will always have an 80 pixel column invisible. In order to allow maximum compatibility, the SuperCharger therefore allows the picture to be adjusted to the left or right or centered with the "/" key on the numeric keypad.

The best match between PC and Atari graphics is found with CGA emulation. The Atari colour monitor con display the CGA's 640 x 200 pixels in two colours and the 320 x 200 mode in four colours directly. In text mode the Atari displays less colours, but has a larger character size of 8 x 16 pixels by making full use of the higher resolution.

But you may also have CGA mode on your monochrome monitor; you then have grey tones instead of colours.

The higher resolution of the monochrome monitor is used in text modes through an 8 x 16 matrix. Graphics are activated in the so-called double-scan-mode, meaning that you can use the whole monitor by doubling the number of lines.

The choice of the graphic mode is made with with the utility program SETVIDO.EXE under MS-Dos, together with one of the following two parameters:

SETVIDEO C	Switches to CGA Emulation
SETVIDEO H	Switches to Hercules Emulation

Diskette Drive

Though PC-compatible 3.5" diskettes may be read and written just as though the drive were connected to an IBM PC, the floppy controller of the ATARI-ST has been designed in such a way that it will not allow certain copy-protected PC software diskettes to be read.

Reading or booting these disks is, in most cases, not possible,

Software

In the PC world, diskettes can be either 5.25" or the 3.5" format. Take care when you purchase that you obtain PC software on the 3.5" medium.

Although now very rare, the software copy protection devices known as "Dongles" will cause problems when connected to the printerport. Software of this kind, in most cases, does not run.

5.25" floppy disk

An alternative solution is the purchase of a 360kb 5.25" floppy, as used in earlier PC's. You simply connect it to your ATARI, then you never need worry about media size when you purchase software. However, for the same price you only get one half the capacity of a double sided 3.5" floppy disk.

Note that the purchase of a 5.25" floppy drive will not unlock protected disks as the problem lies in the Atari controller, not the drive.

Printerport

The ATARI printerport is somewhat different in its connection to a standard PC printerport. "Paper Empty" for example is not utilised, therefore you don't get the usual signal.

See also the note on Dongles, under Software, above.

Serial Port

As with the printerport, the ATARI is one line short of full compatibility. There is no signal "Data Set Ready" (DSR, pin 6) which is needed by some communication programs for synchronisation.

If a communications program doesn't work with a hardware handshake, you can often switch to software handshake (3-lines mode)

A suitable cable for serial connection is detailed in Appendix E.

Users of LapLink II™ communication software will find that connection between a SuperCharger and a PC is possible, but only when the PC is used to control communication (ie is the "local" operator, in LapLink terminology).

Keyboard

In order to allow the highest possible compatibility, the board should be PC-compatible. Though the ATARI keyboard is quite similar, there are some very minor differences. Note, however, that all characters in the ASCII character set can be generated under MS-Dos by holding down the ALTERNATE key any using the **numeric keypad** to enter the ASCII 3-digit code for that character. For example:

Alternate - 036 produces $

Mouse

The ATARI mouse can be used for any MS-Dos program that supports the mouse-systems mouse and will work directly with most windows environments (GUIs)..

For those programs that require an external mouse driver such as Microsoft Word or Norton Commander an external mouse driver is also supplied. To install the mouse driver type MSMOUSE.COM. The mouse systems mouse has 3 buttons, but only the outer two are supported by the ATARI-mouse. This should not cause any problems as very few programs support a third mouse button.

9.0 Fault Diagnosis

As in any complex computer system, the new MS-Dos user may encounter a number of problems before becoming familiar with the system.

Some of the most common problems our users have encountered are as follows:

1. The monitor remains dark and the system hangs.

Turn off all components of the system. Disconnect the DMA cable between the ATARI and the SuperCharger.

Check the correct function of your ATARI system; including disks and floppies.

If the ATARI is functioning, then check the DMA cable for external damage or bad plug pins.

If these are OK, now check SuperCharger's power supply. The POWER LED shows the stystem is functioning.

2. System failure when MS-Dos started.

Read the chapter on installation. Turn off all components and start again. if this doesn't help, you may have electrical disturbance through another machine working too close to your ATARI and SuperCharger.

The most common failures of this sort are a result of errors in the system files AUTOEXEC.BAT and CONFIG.SYS in MS-Dos. If the system, even after a change of the two files, still does not work, then you need to start the system again with a copy of the original MS-Dos boot disk and rechange the sequence of commands within the two files.

3. A program doesn't boot or doesn't install properly.

First check for diskette damage and write protection (see chapter on compatility).

If this doesn't give you an answer, then you may have a conflict of two or more resident programs which are incompatible. This is not something you can change.

If the package doesn't allow the co-existence of certain resident help programs you have installed check the program manual. Faults of this kind may be the case with very simple resident programs such as mouse drivers or international keyboard drivers.

You can often cure these problems by changing the sequence in which the drivers appear in the CONFIG.SYS file of MS-Dos.

4. The hard disk doesn't start when SuperCharger is connected.

Check that the power led is glowing.

If this is not the case, all peripherals connected to the ATARI hard disk interface will not be accepted by the ATARI computer. You must ensure that SuperCharger is always the first peripheral of your system when you switch on (unless you have a laserprinter, which you should switch on immediately before the Atari).

If the DATA LED lamp is on, push RESET.

Check the correct sequence and addresses of the connected peripherals. The SuperCharger has to be the last in the row of connected peripherals with one exception, which is when you work with a laserprinter.

5. MS-Dos doesn't find your ATARI hard disk partition or disks.

Without ATARIDR.SYS (for hard disks) and ATARIDSK.COM (for diskettes), MS-Dos only finds the hard disk boot partition (C:) and only reads MS-Dos disks.

Therefore check to see if you can find "device=ataridr.sys" in MS Dos file CONFIG.SYS and if the operation program itself is on the Boot disk or Work disk. (The same operation applies to ATARIDSK.COM in the AUTOEXEC.BAT file).

6. Formatting problems with 720K (80track) disks.

The support of 720kb format of the ATARI floppy disks is given on a PC under MS-Dos through the input of the floppy parameters of DRIVPARM. See CONFIG.SYS in the MS-Dos Boot disk.

DRIVPARM = /D:0/C/I/T:80

If this remark is missing, MS-Dos assumes PC standards and only allows the formatting of 360kb disks.

If you have a second 720kb drive, you add to CONFIG.SYS the following line using any kind of text editor:

DRIVPARM =/D:1/C/I/T:80

See also Drive Swapping in chapter 2.0 of this section

7. Program files clicked under TOS won't start.

You can only start MS-Dos programs (.EXE, .COM, .BAT files) from TOS when LAUNCH.TTP is installed correctly. If the search path for LAUNCH doesn't fit with that of DESKTOP.INF (which was modified by INSTALL.PRG), then you fall back into TOS.

8. The monitor turns dark when using HOTKEY.

Do not use so-called screensavers, i.e those programs which turn off the monitor after a period of idle time. They will conflict with SuperCharger's HOTKEY command

9. MS-Dos Doesn't find all TOS partitions.

The ATARIDR.SYS only recognises TOS partitions between 4Mb and 32Mb when MS-Dos is started.

If your system has a partition of > 32Mb or has an MS-Dos incompatible data format, you'll get the information:

" ..... ATARI partitions have incompatible data formats ....."

and because MS-Dos does not allow a free sequence in its letters, all following partitions get moved to the next letter.

Example:

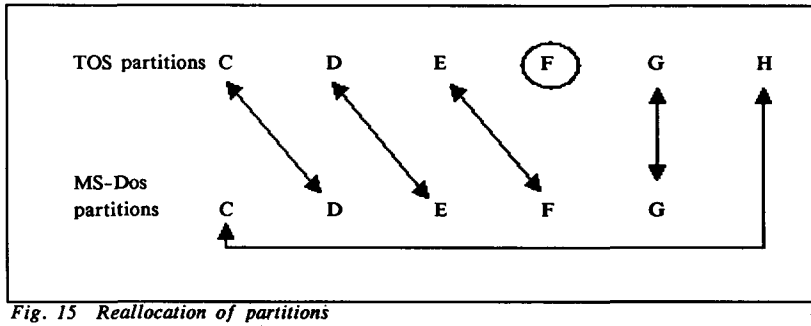


Fig. 15 Reallocation of partitions

Your system has 6 TOS partitions C - H, having C - E with 15 Mb, F with 2Mb, G with 30Mb and H reserved for the MS-Dos initialised Boot partition.

Under MS-Dos, the H (TOS) becomes C (MS-Dos) and all other partitions will follow the sequence they had been recognised, eliminating (TOS) F.

NOTE: This only names the MS-Dos partitions; there is no change in the partitioning undertaken under TOS. Partition F (example) can still be reached under TOS and all other partition names remain the same. The only change you will recognise is to be found in the hard disk used when you work under MS-Dos.

10. A PC program won't run.

Note that certain items of PC software that are on sale currently are not compatible with the latest version of MS-Dos (4.01) which is the version shipped with SuperCharger. Most of the incompatibilities arise from an inability of the software product concerned to handle the increased size of the FAT (File Allocation Table) that enables MS-Dos 4.01 to handle large hard disk partitions.

To overcome this problem, either ask the software author for a modified version or obtain a version of MS-Dos 3.3 under which the software will run.

11. Concurrent Processing Conflicts

To enable parallel processing of TOS and MS-Dos programs which involve disk accesses, the source and data files must both be directed to their respective exclusive partitions.

If your application won't work concurrently after using the advanced HOTKEY, make sure there are no hidden write accesses to TOS partitions. For example, during processes such as buffering or temporary backups which may be "hidden" from the user.

NOTE: Because the printer port can be accessed by both TOS and MS-Dos we recommend manual synchronising of printing by using the ALERT command. If MS-Dos and TOS printing processes are carried out at the same time TOS will send output to the printer every time MS-Dos takes a break, resulting in mixed printer output.

Appendix A:
Changing your device address

Up to eight devices can address the Atari DMA port at the same time, each of which must be addressed separately in order to avoid confusion. Your SuperCharger is factory-preset to address 3. If another device already occupies that address and cannot be changed, you can change the SuperCharger device address by modifying the jumper X5 inside your SuperCharger. See the diagram of the main board in Appendix B to help locate the position of the jumper.

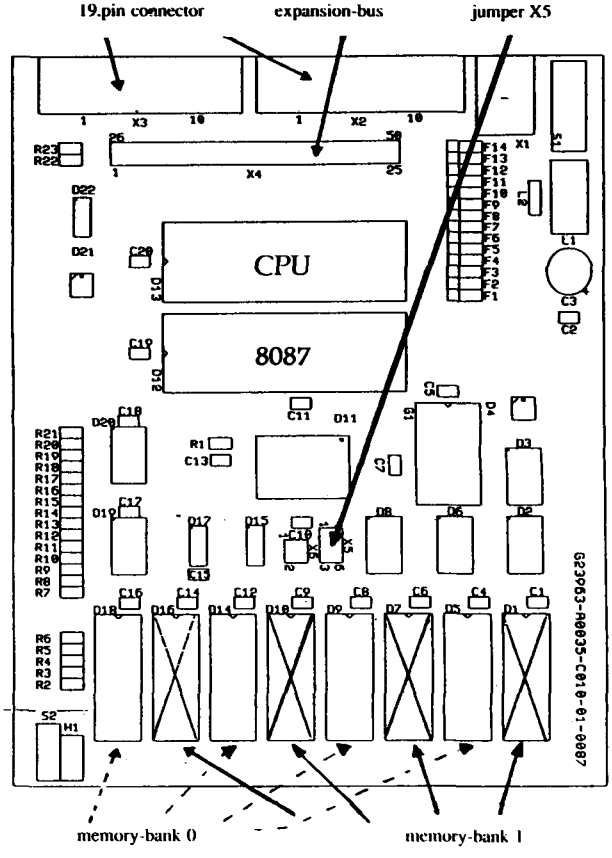
The jumper value is set according to the following table:

Jumper X5

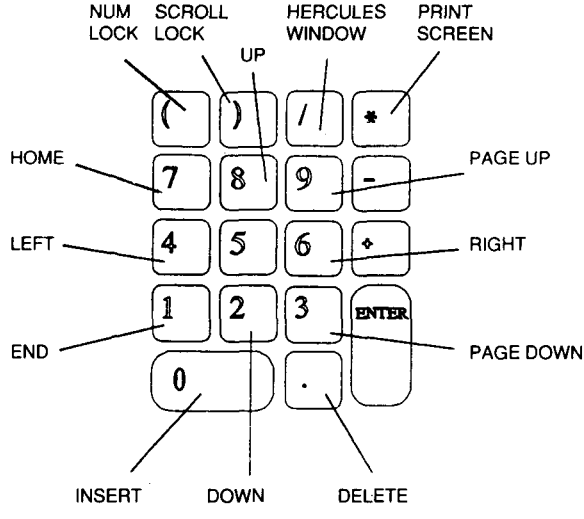
Jumper between:	1-4	2-5	3-6	=device address
	yes	yes	no	1
	yes	no	yes	2
	yes	no	no	3
	no	yes	yes	4
	no	yes	no	5
	no	no	yes	6
	no	no	no	7
	yes	yes	yes	0

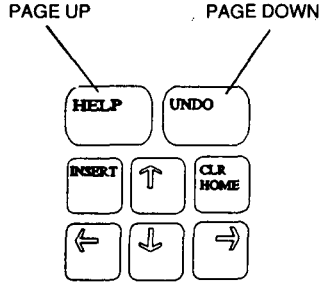
Note: Do not forget to run INSTALL.PRG again to let the software know of the change of hardware address!

Appendix B:
Main Board layout



Appendix C:
Cursorblock layout

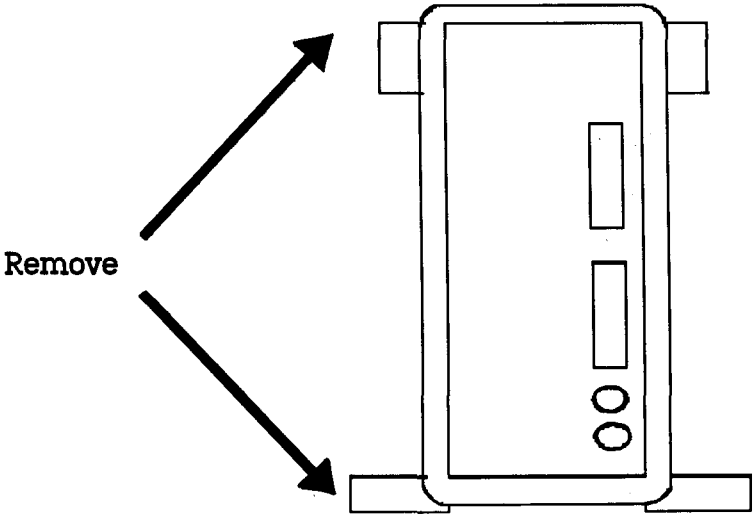




Appendix D:
Casing disassembly

To disassemble your SuperCharger, all you need is a screwdriver.

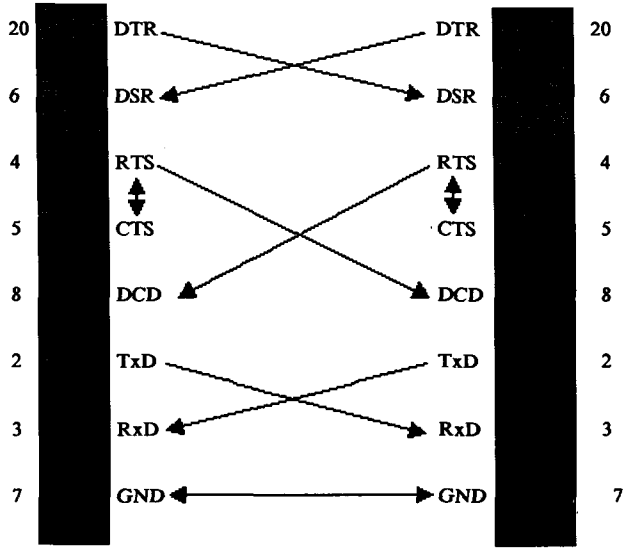
1. Remove all four of the black feet on the flat side opposite to the 19-pin connectors.
2. Remove the screws that were hidden by the black feet.



3. Carefully pull the two halves apart.

When reassembling, ensure that the two seal rings around the front and rear cover plate are properly in place or they will be damaged.

Appendix E:
Serial Connection Cable



Appendix F: Key Combinations

ALTERNATE+CONTROL+DELETE

Initiates a Warmstart (Reboot) of the SuperCharger

ALTERNATE+CONTROL+BACKSPACE

SuperCharger Hotkey. Returns control to the Atari operating system and suspends SuperCharger.

ALTERNATE +S+A

Switch to Alternate Colour Key. Switches between normal and inverse video when in monochrome mode.

ALTERNATE+ASCII

By holding down ALTERNATE and using the numeric keypad to enter the 3-digit ASCII character code you can generate any of the 256 ASCII characters.

SHIFT+CONTROL+BACKSPACE

Returns to TOS and the Atari whilst leaving your MS-Dos process running. Progress messages can be flashed on the TOS screenfrom MS-Dos using the ALERT utility.

NOTE that this requires synchronisation between TOS and MS-Dos, so the SuperCharger must have been started via the Desk Accessory, **not** via ABIO.TOS.

Note: **All hotkey combinations must be entered in the order shown and the keys held down concurrently.**

contents: section II - Intro to MS-Dos

Preface

Note: This section of your manual contains a brief introduction into MS Dos version 4 so that you can configure your system yourself and master the basic routines such as creating, editing and copying files etc. However, this is not a substitute for a closer study of manuals on MS-Dos.

MS-Dos is a command line orientated user interface; only since version 4 has there been additional graphical help for the user. It is up to the user to decide whether he wants to continue to work in the command line environment or make use of the help menu "SHELL". A drawback of "SHELL" is that it takes up more memory than the command line version thus decreasing the proportion of the Supercharger's 704Kb memory that is available to the user.

A small part of the DOS command set is resident, i.e. is available at any time. The larger part is however downloaded from the hard disk or floppy disk when needed; If you work without a hard disk you may find that you have to change disks quite often. All commands in files with labels .COM or .EXE on your system disk are non-resident commands, e.g. FORMAT, MODE, SYS etc.

Ports

The most important MS-Dos ports to the SuperCharger user are:

COM1-COM4 = possible serial interfaces.

When using the SuperCharger, COM1 is used by the ATARI mouse and COM2 is used as the serial port of the ATARI. Other COM are currently not occupied.

CON = console / keyboard for input and output.

LPT1-LPT3 = possible parallel interfaces.

When using the SuperCharger LPT1 is used for the ATARI printerport.

PRN = LPT1

Function keys

In the MS-Dos command line environment the main function keys (F1 to F6) can be used to edit and re-enter the previous command. The function keys are used as follows:

F1 Copies the next character from the last stored line to the new command line.

F2 Copies every character from the last stored line into the new command line upto the cursor.

F3 Copies every character from the last stored line into the new command line. If F1, F2, F4 or DEL have previously been used, only the remaining characters are copied.

F4 Copies all the given characters in the stored command line to the right of this character.

F5 Replaces the old with the new command line before ENTER; F1 to F4 work according to the new command line.

F6 Adds Ctrl.Z (= end of file) to command line.

Furthermore, you have the editing keys DEL, ESC AND INSERT

DEL Skips one character of the old input line.

ESC Abandons current command, but does not overwrite previous commands.

INS(ERT) Switches the Insert mode on and off.

Example:

You made a typing error in the following command:

COPY A:\ERROR.EXE C:\DOS\UTILITY\DATA\HELP\ERRROR.EXE

To avoid keying in the whole line, you press F1 until the cursor in the new command line is on the error (if you go too far, BACKSPACE will allow you to go back). After correcting the error, press F3 to copy the rest of the command line.

File names:

Under MS-Dos, file names have a maximum of 8 characters and an extension of no more than 3 characters. This extension can define the file type. All files of the type .BAT, .COM, and .EXE can be executed directly; hence, when the file name is called (without the extension!) the program is started. All other extensions are used for file type identification, e.g. .BIN for binary files and .LIB for library; however, they have no special significance to MS-Dos

Invalid names

MS-DOS prohibits the use of names which are similar to devices like:
aux, clock$, com1, com2, com3, com4, con, lpt1, lpt2, lpt3, lst, nul, and prn.

Warning: MS-Dos does not differentiate between upper and lower case.

Boot disk and hard disk:

When you have formatted your boot disk or hard disk appropriately (see also our example at the end of this section of the manual for commands), the system files IO.SYS and MSDOS.SYS will be on your disk. These are not normally accessible and will not be shown in the path directory. Together with the file COMMAND.COM these 2 files constitute the resident part of MS-DOS.

A system formatted disk/hard disk with these 3 files can be used for booting. However, you will only have the resident DOS commands available. Depending on space and your needs you can copy further non-resident programs onto the boot disk. Only if you are working with a hard disk will it make sense to copy all the files from the original system disks.

IO.SYS, MSDOS.SYS and COMMAND.COM together take around 110Kb of disk space.

Device drivers

MS-DOS runs on a whole range of different machines. In order to customize it for your own hardware configuration, there are device drivers. These fall into two groups; resident and nonresident. Resident device drivers are used for such things as national keyboards, special hard disk types, graphics tablets etc, which MS-DOS stores automatically when booting the system. A representative of this group (ending on .SYS) is ATARIDR.SYS on your utility disk. It ports the TOS partitions with the MS-DOS partitions on your hard disk.

Loadable device drivers are different from resident device drivers in that they can be loaded at any time when there is a need for them, e.g. for a mouse driver. If the memory space is required, the application can be taken out afterwards.

When the MS-Dos system is booted (started up) it will load certain device drivers automatically. They are determined by the contents of two files which you must create. The first file to be processed by MS-Dos is the CONFIG.SYS file which contains device drivers and certain specific commands. The second file to be processed automatically is the AUTOEXEC.BAT file which may also contain MS-Dos command line commands. Both CONFIG.SYS and AUTOEXEC.BAT must be located in the root directory of the disk you boot from.

CONFIG.SYS file

As the name indicates, this file contains the calls for .SYS driver programs which are installed when the system is started.
Apart from this, there are a number of special commands for controlling the system. The following are important:

BREAK

If the BREAK status is set, i.e. BREAK=ON is written into CONFIG.SYS, the user can abort the current process by keying in CTRL+C.

BUFFERS

This command sets the number of 512 byte blocks MS-DOS may occupy in the main memory to buffer disks/hard disk data. The line "BUFFERS=10" reserves space for 10 sectors, which corresponds to about 5KB. For applications which access disk all the time such as database programs, we recommend "BUFFERS=20". It is not useful to set the number higher unless you have enough space (SuperCharger with 1Mb) and the programs used actually demand a higher number of buffers.

COUNTRY

The command "COUNTRY=xx" determines the special characters and formats you want MS-DOS to use. 'xx' represents the code number and determines the nationality. 49 represents Germany, 44 the UK, 31 the Netherlands, and 32 Belgium. If the country is not set in CONFIG.SYS, DOS will use an ASCII character set and American Standards for the representation of time and date.

DEVICE

DEVICE = is the CONFIG.SYS command line to load external drivers. It is followed by the path/filename of the driver concerned. Multiple device drivers may be loaded and used concurrently, each preceded by DEVICE =.

WARNING! Both drivers COUNTRY.SYS and KEYBOARD.SYS are automatically loaded and configured according to the default of "COUNTRY=xx" you set. Do not try to load these device drivers manually! The most important driver for you is ATARIDR.SYS on the utility disk. ANSI.SYS which is available under MS-DOS should only be installed if it is required by a program, because it slows down the screen output on the SuperCharger.

FILES

FILES sets the number of files MS-DOS handles simultaneously. If FILES is set to 22 the number of opened files at any one time is limited to 22.

Sample CONFIG.SYS File

On the 1MB SuperCharger the CONFIG.SYS file should look as follows:

```
BREAK=ON
DEVICE=ATARIDR.SYS
COUNTRY=44
FILES=22
BUFFERS=30
```

If you are working with a 512k SuperCharger the number of BUFFERS needs to be reduced accordingly. 20 might make a good starting point.

AUTOEXEC.BAT

Under MS-DOS all files ending in .BAT are known as "batch" files.

Although it is only a text file, a .BAT file is controlled via the command processor. The file is processed sequentially and submits commands to MS-Dos just as if they were being input from the keyboard. Some special control commands are also available for use in batch files. Consult a detailed MS-Dos manual for these.

Both AUTOEXEC.BAT and CONFIG.SYS are system control files which MS-DOS seeks and executes automatically when the system is started up. AUTOEXEC.BAT contains a number of commands which configure your system before you start your work. This includes setting of time and date, loading of a keyboard driver and setting of the path or loading of a mouse driver.

Sample AUTOEXEC.BAT File

For a hard disk system AUTOEXEC.BAT would look as follows: (If you have a Mega-ST with internal real time clock, time and date will be superfluous).

```
keyb uk
date
time
path c:\;c:\dos;a:\;
prompt $p$_$g
ataridsk
dir
```

When you start the system will first load the UK keyboard driver and then ask you to enter date and time; afterwards it will set the path for the main directory of drives C: and A: as well as the sub directory \DOS and create a prompt which shows the current path. Loading ataridsk means to load a driver which enables you to work with ATARI disks under MS-DOS. As a final step the system will list the main directory.

Primary MS-Dos commands

After this general overview we will now describe the most important commands as they will appear to the user in the MS-Dos command line environment.

The following syntax is applicable:

{ } Commands in braces are optional

[ ] List of options; one of them has to be selected

| Separates the parameters in a command / of an option

* The asterisk represents a "wildcard" (joker).

A wildcard stands for a filename or part of it, e.g. *.BAT stands for all files of the file type .BAT.

Brackets and vertical line are not entered when you type in a command. They are only used for structuring the manual.

The following subset of the MS-Dos command set is described in this chapter for SuperCharger users:

CHDIR/CD
CLS
COPY
DATE
DIR
DISKCOPY
ERASE/DELETE/DEL
FORMAT
KEYB
MKDIR/MD
PATH
PROMPT
RENAME/REN
RMDIR/RD
SHELL
SYS
TIME
TYPE
XCOPY

chdir/cd

Syntax: CHDIR [path] or CD [path]

effect: changes the current path

Example: The current path is: \dos\utility\help. If you type in CD\DOS you get directly into the subdirectory "\dos", omitting "\dos\utility". If you enter CD.. the path is "\dos\utility". By repeating CD.. you then get back to \dos.

To get back onto the old path, key in CD\DOS\UTILITY\HELP and you are again in "\dos\utility\help".

Step by step you can type: cd utility and then cd help Note that the "\" is not typed here.

cls

Syntax: CLS

effect: CLS erases the screen but for the prompt and the cursor.

copy

Syntax COPY [drive:]path+name1[drive:]path+name2]

Effect: COPY copies a file with name1 from directory1 into directory2 under name2. Name2 is created in directory2; if name2 already exists in directory2 it is overwritten with name1 without request!

Example: COPY A:\DATA\WORK.EXE C:\DOS copies the file "work.exe" from drive a and path\data to drive c: and path \dos. COPY A:\DATA\WORK.EXE C:\DOS\LAZY.EXE copies work.exe again, but creates a new file called "lazy.exe" under c:\dos. If you use the joker "*" for name1 or any part, all copies of the same name will be copied.

If you type COPY A:*.EXE C:\DOS MS-Dos will copy all files from the main directory from drive A ending on .EXE into drive C, subdirectory \dos.

If you omit path and name 2 MS-Dos will copy the file in path and name 1 to your current directory under the same name.

date

Syntax: DATE [mm-dd-yy]

Effect: DATE shows the current system time; if month/day/year are typed in, the system time is reset. If CONFIG.SYS is correctly set to COUNTRY=44, the sequence is day/month/year. When starting a Mega-ST model, the date is automatically taken over from the real time clock.

del\erase

Syntax: DEL [drive:]path+name[/p]

Effect: DEL erases a file at the given pathname/drive If the optional parameter /p is typed in, DEL will prompt the user before deleting the file.

dir

Syntax: DIR [drive:][path+name][/p][/w]

Effect: DIR shows all files in the path/drive which you have called. /p puts in a break after each screen page. /w shows an overview of the files, not a list.

Example: DIR or DIR *.* shows all files in the current path. DIR *.EXE shows all files in the current path ending on .EXE. dir c:\dos shows all files in drive c:, path \dos.

diskcopy

Syntax: DISKCOPY [drive1:] [drive2:]

Effect: DISCOPY copies the contents of the disk in drive1 onto the disk in drive2 and formats it if necessary. If drive1=drive2, source and destination disk may have to be changed several times during the copying process.

format

Syntax FORMAT drive:[/s]

Effect: FORMAT formats disks and hard disks for the use of MS-DOS. By typing in /s a system disk is created.

Example: FORMAT A: creates a data disk. format c:/s formats the hard disk and copies the disks necessary to boot.

WARNING! After calling format the computer requests whether you really want to format. At this stage, verify that you have selected the correct destination drive. After formatting a drive, all previous data on that drive will be erased!

keyb

Syntax: KEYB [xx]

Effect: KEYB together with the national code xx loads a new keyboard driver (ASCII). The file KEYBOARD.SYS needs to be in the same path when called. The system will default to US keyboard if this is omitted.

Code: gr = German uk = United Kingdom nl = Netherlands

In case you are not able to find a certain character on your keyboard you can enter the ASCII-code by holding the ALT key pressed while typing the respective code number on the numeric key pad.

mkdir

Syntax: MKDIR [drive:]path or md [drive:]path

Effect: MKDIR or MD create a new sub directory.

Example:

MD DATA creates a new sub directory \data under the current path. md\data\junk creates under \data another sub directory called \junk.

prompt

Syntax: PROMPT [[text]$option]...]

Effect: PROMPT is used to customize the prompt from the MS-DOS command interpreter. The default only shows the current drive, e.g. "A".

The following $characters are valid options:

$q creates an =

$$ creates a $

$t gives you the time

$d gives you the date

$p shows you the current path

$v the MS-DOS version number

$n indicates the current drive

$g gives a > character

$b gives a |

$_ gives a new line

$h creates a backspace character

Example:

PROMPT P_$G Given the current drive C, this prompt creates the following display:

```
c:\dos
>
```

The next command line starts behind the flashing ">".

path

Syntax: PATH [drive:][path][;[drive:][path]...]

Effect: PATH tells MS-DOS in which paths it should look for program calls outside the current path. The maximum length of the seek path is 127 characters!

Example:

PATH C:\DOS;C:\DOS\DATA;D:\UTILITY;D:\PROGRAM tells MS-Dos to seek executable files (.exe, .com, .bat) in the following paths:

c:\dos c:\dos\data d:\utility d:\program

ren

Syntax: ren [drive:][path]file name1 file name2

Effect: REN and RENAME change file name1 to file name2 on the current drive / path.

Example: ren *.txt *.bak

changes all files from file type .TXT to file type .BAK. MS-DOS does not ask for confirmation, so be careful using * when selecting.

rmdir/rd

Syntax: RD [drive:]path or RMDIR [drive:]path

Effect: RD and RMDIR remove a sub directory from the path. This is only possible if the sub directory is empty and does not contain sub directories itself.

sys

Syntax: SYS [drive]

Effect: SYS transfers MS-DOS system files to the drive given. The destination drive must be formatted. SYS will transfer both IO.SYS and MSDOS.SYS, which are not listed in the directory! COMMAND.COM must be copied with COPY. Sys is mainly used to transfer new MS-DOS versions onto old boot disks / hard disks. Data on the disk/hard disk remains intact.

time

Syntax: TIME [hours:minutes[:seconds[.hundredth]]]

Effect: TIME sets the system clock - On the ATARI Mega-ST, the DOS system clock is automatically set according to the built-in real time clock when you boot.

Valid formats are:

Hours: 0 to 23
Minutes: 0 to 59
Seconds: 0 to 59
Hundredths: 0 to 99

type

Syntax: TYPE [drive:][path]file name

Effect: TYPE shows the contents of text files.

xcopy

Syntax: XCOPY [drive:][path+name][drive:][path+name][/e][/s][/v]

Effect: Unlike COPY, XCOPY can also copy subdirectories.

The parameters have the following meaning:

/s copy subdirectories, too /e also copy empty subdirectories
/v compare source and destination files

Example:

You are in drive C: \dos\utility. Thea disk in drive A contains a number of files in the main directory and 2 subdirectories \util1 and \util2. After entering

XCOPY A:*.*/S/E

all files from the main directory of drive A are copied to C:\dos\utility. Underneath, \util1 and \util2 will be created with all the files.

Sample MS-Dos hard disk Installation

The following example shows you how MS-DOS can be installed on a system with a hard disk without using SETUP.

On the Atari, run INSTALL.PRG (see Section I, Chapter 2.0) to prepare a boot partition; then insert your utility disk in drive A: and start ABIO.TOS. Now boot SuperCharger either with the MS-DOS boot disk that comes with the operating system, or any other version of the MS-Dos operating system that you may wish to install. Then format the partition for MS-Dos using the command:

```
FORMAT C:/S
```

MS-DOS asks you to confirm the format command; enter Y for Yes and it will format the hard disk and create the system files.

The hard disk is now bootable, but it is not yet complete. You require the control files CONFIG.SYS and AUTOEXEC.BAT as well as all non-resident DOS programs. To copy these you first change from the current drive A: to drive C: and create a subdirectory \DOS for the non-resident DOS programs:

```
C:
MD DOS
```

Now insert the utility disk:

```
COPY A:\DOS\AUTOEXEC.HDx  C:\AUTOEXEC.BAT
   (replace x with U for UK systems, or E for USA
COPY A:\DOS\CONFIG.HDD  C:\CONFIG.SYS)
COPY A:\DOS\ATARIDR.SYS
COPY A:\DOS\ATARIDSK.COM
COPY A:\DOS\SETVIDEO.EXE
COPY A:\DOS\MSMOUSE.COM
COPY A:\DOS\DOSSHELL.BAT  C:\DOS
   (omit this for MS-Dos versions < 4.00)
```

Now all nonresident MS-Dos files on the system disk are copied into the subdirectory.

```
COPY A:\DOS\*.* C:\DOS
```

This process must be repeated for each MS-DOS disk (There is a \dos directory on each of them). Press function key F3. The copy command will be repeated on the screen. Copying is started again by pressing Enter.

Now the installation of MS-DOS on the boot hard disk is complete.

contents: section III - Toolbox

1. What you need to know to use TOOLBOX

In conjunction with the SuperCharger this TOOLBOX offers a host of new opportunities to the TOS programmer. It will allow him to accelerate his programs by using both the Atari 68000 processor and the V30 in the SuperCharger (or several connected V30 processors) at the same time. TOOLBOX also enables you to implement innovative programming concepts such as parallel processing.

For reasons of data protection it is essential that you read through the entire manual before you use the TOOLBOX. The quick reference tables in the appendix do not supply sufficient information to guarantee that you handle the functions of the TOOLBOX safely. We would like to point out that the TOOLBOX has been conceived as a device for the programmer to improve his own programmes for the Supercharger. Nevertheless, we decline responsibility for possible faults and any consequences which you might encounter when using the TOOLBOX. The TOOLBOX is not a "safe" programming unit and inappropriate use of the TOOLBOX may lead to loss of data. It is the user's own responsibility to ensure the safe handling of the TOOLBOX.

2. General introduction: The SuperCharger TOOLBOX

The SuperCharger itself is capable of far more than simply emulating PCs and the TOOLBOX was developed with this idea in mind. Although the initial purpose for designing the SuperCharger was to develop a powerful emulator and this feature will be uppermost in the mind of any buyer we have come to think that emulation itself is really only a special case, where the Atari and the SuperCharger each run a program at the same time and where these programs can communicate with each other.

With the SuperCharger we have made available an all inclusive system with CPU, its own memory and hardware suitable for communication on the DMA bus. In order for the programmer to use this system for his own applications, he needs to load his V30-programs into the SuperCharger, to start them, to add data and to retrieve results. The TOOLBOX provides all these features as well as functions for controlling program execution (Timer). Additional interupts can be inserted for the maths coprocessor 8087.

3. Introduction to Communications between the SuperCharger and Atari

Before describing the functions of the TOOLBOX the following is a brief introduction to DMA-bus communication.

Communication between the Atari and the appliance connected to the DMA-bus functions as follows: The first step is the so-called addressing of the desired appliance by keying in the relevant number of the appliance, which is between 0 and 7. Thus the Atari activates the relevant appliance. At the same time the appliance receives a command: the Atari informs it of the required operations (e.g. read sectors, format hard disk, print page on laser printer etc). In case the command needs more data, all necessary communication follows directly after the addressing. During this time the DMA-bus is reserved for communication between the Atari and the selected appliance; this means in particular that no other DMA appliance may be addressed until the operation is complete. At the end of the operation the DMA-bus is deselected. It is only then that the DMA-bus can make the Atari communicate with another DMA-appliance.

4. Dangers: Inappropriate use of the DMA-bus

Bear in mind that you can only address one appliance via the DMA-bus at any one time and that communication on the DMA bus is always initiated by the Atari. The communicative functions of the TOOLBOX take care of the addressing and the deselection of the DMA-bus automatically before and after every communication.

This does not apply for all the functions mentioned in Chapter entitled "Enhanced functions of the toolbox for systems programmers". If these functions are being used inappropriately, e.g. if the DMA-bus is not deselected and the hard disk is addressed afterwards, you are likely to lose data on the hard disk. We have included these functions nevertheless to permit time critical programming and extremely fast communication between the SuperCharger and the Atari.

For more detailed explanation of the DMA-bus, please refer to the relevant publications in professional magazines and books.

5. The Levels of the TOOLBOX

When the TOOLBOX is loaded the SuperCharger can be in two states: After starting TOOLBOX no application program is yet active in the memory of the SuperCharger. This means that some functions have to be made available which run completely under the control of the Atari. At this point, the scheduler, a component of the TOOLBOX, makes the functions available to write and read data from the Atari into the memory of the SuperCharger. Scheduler also enables you to start a program from an address submitted by the Atari. The second state is the running performance of an application program. In this case all further functions are available to the program via the TOOLBOX which has already initialised Interrupt 60h.

On the 68000 such a distinction does not exist, as the TOOLBOX can be exclusively controlled by an application program. Nevertheless, the 68000 programmer has to be aware at all times of the state of the SuperCharger. If e.g. Scheduler is running in the SuperCharger, then the 68000 program must only use those functions which the Scheduler has made available to manipulate memory and run a V30 program. If, on the other side, a V30 application program has called a function of the TOOLBOX. then the 68000 application can only call the respective function of the TOOLBOX.

The following is a brief description of the normal running of the TOOLBOX from the Atari; you will find further details under the descriptions for the respective functions.

Execution through the AUTO-control when starting the computer or manually before starting the appplication program:

1 - Install TOOLBOX-Trap by starting SCTB.PRG on the Atari.

Execution of the application program:

2 - Load TOOLBOX into SuperCharger by calling SC_BOOT

3 - Load application program (V30) into the SuperCharger by calling SC-SM

4 - Execute application program (V30) by calling SC-EXEC

5 - general communication now controlled by the context of the program executed with commands SC_SD, SC_GD, SC_SB, SC_GB

6. The functions of the SuperCharger-TB for TOS-programs.

All functions of the TOOLBOX and TOS are called through a parameterised call of Trap #4. Trap #4 is being made available by the resident part of the program SCTB.PRG, which can be copied into the AUTO-control.

General comments on the allocation of registers

A note on our notation: The endings .L, .W , and .B refer to the width of the register which is being addressed. If, e.g. the register D1.W is in question, it is sufficient to load the value by entering a command such as MOVE.W#...,D2. If, however, D2.L is in question, the value must not be abbreviated.

All TOOLBOX functions are called by loading the number of the function into the D0.W register and executing the command "TRAP #4". The only exception to this rule is the function SC_CHECK, which is described later in more detail. The transfer of further parameters is described in in detail under the decriptions of the respective functions. Each TOOLBOX function can change the registers D0, D1, D2, D3, A0, and A1.

Each function which has to address the SuperCharger has the application DMA address of the SuperCharger in the register D1.W. The valid range is 0 to 7; the SuperCharger is being delivered with a standard setting of 3. By changing these parameters it is possible to connect several SuperChargers at once to the Atari.

The register D2 contains the length in bytes for memory transfer as a double word. However, the maximum length is limited to 64Kbytes. Longer blocks of data must be transfered in several chunks.

If the TOOLBOX function needs a V30 memory address, this will be transfered in register D3 as a double word. The V30 segments is in bits 31 to 16, the offset is in bits 15 to 0. Thus The V30 address 1230:0506 corresponds to the register value of D3.L = $12300506.

In case the Atari needs a memory address, this will be given through Register A0.

At the end of the TOOLBOX function there is the so-called result code, and this can be found in register D0.L. 0 indicates that the function has been successfully executed, negative values normally indicate faults. The following is an interpretation of the different values:

Result Codes

0 okay (function successfully executed)

-1 Resident TOOLBOX not loaded

-2 (reserved)

-3 SuperCharger not connected

-4 Timeout

-5 Fault in protocol

-6 ABIO Hotkey acknowledged

-7 (reserved)

-8 Illegal command code in register

6.1. Administrative functions of the TOOLBOX

Apart from the functions for direct communication with the SuperCharger, further functions are available for superior tasks on the side of the Atari. These are:

SC_CHECK Verify availability of TOOLBOX

Entry: -

Exit: D0.L = Result-Code
D1.L = number of version if D0 = 0.

Note: SC_CHECK is the only TOOLBOX function, which is not called by TRAP #4, but by JSR SC_CHECK. It can be found in the include file SCTB_INC.S and should be directly included into your program.

SC_CHECK is used to test the availability of the TOOLBOX. As SC_CHECK can therefore not be a part of the TOOLBOX, it is the only TOOLBOX function that is delivered with the source code. Together with a few general definitons (EQU

instructions) it can be found in file SCTB_INC.S. The result code of SC_CHECK is 0 (SC_OKAY) if the TOOLBOX is available; otherwise it is -1. If the TOOLBOX is available the number of the version will be returned in D1 in Ascii as a double word (eg. $312E3330 for version 1.30).

SC_STAT determine status of TOOLBOX

Entry: D0.W = $0002 D1.W = DMA address (0 to 7)

Exit: D0.L = result code

SC_STAT verifies whether a TOOLBOX or an ABIO Hotkey have already been activated in the SuperCharger. If a TOOLBOX is installed and ready to communicate, then SC_STAT shows 0 (okay). If an ABIO hotkey is active (i.e. you quit an MS-DOS application with the hotkey), then SC_STAT will show -6 (SC_ABIO). Any other message means that the SuperCharger is still in an undefined state and that the TOOLBOX on the V30 needs to be started before communication with SC_BOOT. SC_STAT does not work together with the comunicative functions described under "Extended functions of the TOOLBOX for systems programmers", as these functions do not have a protocol with ID verification before each message.

SC_BOOT install TOOLBOX on the side of the V30

Entry: D0.W = $0003

D1.W = DMA address (0 to 7)

Exit: D0.L = result code

SC_BOOT resets the SuperCharger (RESET) and the TOOLBOX is loaded on the V30. Afterwards, the scheduler is started. On the V30, the memory is partly erased. With this function you normally start the use of the SuperCharger through the TOOLBOX when the TOOLBOX has not yet been loaded in the SuperCharger.

6.2. Functions while the scheduler runs in the SuperCharger

The scheduler in the SuperCharger is able to accept the following commands:

SC_SM send data to V30-memory

Entry: D0.W = $0006
D1.W = DMA address (values 0 to 7)
D2.L = length (max. 64K)
D3.L = V30 segment (high-word) and offset (low word)
A0.L = address of the source data in the Atari memory

Exit: D0.L = result code

The scheduler sends a memory block from the Atari to the V30 memory. The registers A0 and D2 specify the memory area which is to be sent; D3 indicates at which address the data can be filed in the memory of the SuperCharger. The main task of this function is to transfer a V30-program into the memory of the SuperCharger and then have it executed through SC_EXEC. In connection with function SC_GM it is also possible to administer the entire memory of the SuperCharger from the Atari (excluding the areas reserved for the TOOLBOX) without having to write V30-assembler at all.

SC_GM get data from V30-memory

Entry: D0.W = $0007
D1.W = DMA-address (0 to 7)
D2.L = Length (max 64K)
D3.L = V30 segment (high-word) and Offset (low word)
A0.L = target address in the Atari memory

Exit: D0.L = result code

SC_GM requests a memory area from the SuperCharger and files it in the Atari memory which is indicated by A0. Functions SC_SM and SC_GM enable 'you to use the memory of the SuperCharger as a Ramdisk for TOS-programs. In our example "SC_RAM.TTP" we have not made use of this possibility for the simple reason that we wanted to present the full functional range of the TOOLBOX.

SC_EXEC Execute V30-program

Entry: D0.W = $0008
D1.W = DMA address (0 to 7)
D3.L = V30 CS-register (high-word) and IP (low-word)

Exit: D0.L = result code

SC_EXEC starts a V30-program once it is loaded in the SuperCharger. The Atari determines the start address from the D3.L register. The V30-register CS (code segment) is loaded with the values of register D3.L, bits 31 to 16. The V30-instruction-pointer is loaded with bits 15-0 of register D3.L. You can return to the scheduler by a software interrupt on the V30 (see further down).

6.3. Functions used while an application is running in the SuperCharger

The TOOLBOX enables a V30-application program, which is running in the SuperCharger to communicate with a program which is running simultaneously in the Atari. Note that in order to do so both programs need to call complimentary functions: the Atari may for example call "send byte", whereas the V30 calls "get byte".

SC_SD Send data to the SuperCharger

Entry: D0.W = $0004
D1.W = DMA address (0 to 7)
D2.L = length (max 64kb)
A0.L = source address in the Atari memory

Exit: D0.L = result code

When running on the SuperCharger, this command requires the equivalent function receivedata (int 60h, fkt# 4)

ATTENTION: If as a result of faulty protocol from the application program the SuperCharger expects more data than the Atari is sending, the consequence may be a DMA-bus conflict.

The function sends a data block of max. 64kb length to the Supercharger. The data transfer is especially fast if the user takes care that A0 sends an even address and that the length of the data block is a multiple of 512. However, this is not a prerequisite for the use of SC_SD.

SC_RD Receive data from the SuperCharger

Entry: D0.W = $0005
D1.W = DMA address (0 to 7)
D2.L = length (max 64K)
A0.L = target address in the Atary memory

Exit: D0.L = result code

When running on the SuperCharger, requires the equivalent function senddat (int 60h, fkt# 5)

ATTENTION: If as a result of faulty protocol from the application program the SuperCharger wants to send more data then the Atari expects, the consequence may be a DMA-bus conflict.

The function receives a data block of max 64 KB length from the SuperCharger. Data transfer is particularly fast if the user takes care that A0 sends an even address and that the length is a multiple of 512. However, this is not a prerequisite for the use of SC_RD.

SC_SB send Byte to the SuperCharger

Entry: D0.W = $0009

D1.W = DMA address (0 to 7)

D2.B = byte to be sent

Exit: D0.L = result code

When running on the SuperCharger, this command requires the equivalent function: getbyte (int 60h, fkt# 9)

This function sends a single byte to the SuperCharger. It can for example be used to communicate the number of a function to a V30 application program. This byte will not be filed in the memory of the SuperCharger but in the AL register of the V30-CPU.

SC_RB receive Byte from SuperCharger

Entry: D0.W = $000A

D1.W = DMA address (0 to 7)

Exit: D0.L = result code

D2.B = received byte

When running on the SuperCharger, this command requires the equivalent function sendbyte (int 60h, fkt# 10)

This function receives one byte from the SuperCharger. It can be used to for example tell the Atari whether a function has been executed correctly (transfer of a Boolean value).

7. The functions of the SuperCharger-TB for V30 programs

The scheduler makes available 3 functions which are exclusively controlled by the Atari. We only list them again for completeness; the details have been mentioned previously.

1. Loading a data block: The Atari transfers the number of data in bytes (max 64kb), the target segment and the target offset into the SuperCharger memory.

2. Selecting a data block: The Atari transfers the number of data in bytes (max 64kb), the source segment, and the source offset.

3. Executing a program The Atari transfers a segment value which is then loaded into the CS-register and an offset value which is loaded into the IP. The scheduler is thus left but can be reactivated by a function after finishing the program. Note that the memory addresses 0000:3000 and 0300:0000 are not physically different; if you change the start address, however, the 8086 assembler would create wrong offset values in the code.

During a running program, you can access the TOOLBOX functions by calling interrupt 60h. The function number will be transfered in register AH. All functions mentioned here are self-synchronising, which means that you can only escape from the interrupt after the task has been completed.

Function 0: Terminate

entry: ah:0

exit: no escape from interrupt; scheduler running.

This function is used to abort the program and start the scheduler so that for example a new program can be loaded or started.

Function 1: Killtb

entry: ah:1

exit: no escape from interrupt, idle loop running.

This function deletes the TOOLBOX; the DMA-bus is released and the SuperCharger can only be reactivated by SC_BOOT of the Atari or ABIO.

Function 2: Intserv

This function is intended only for systems programmers and is discussed separately in a special chapter.

Function 3:

This function starts the scheduler and is reserved for troubleshooting.

Function 4: Receivedata

entry: ah:4

es:di target address of the data block

cx number of bytes to be received

exit: no registers changed

A certain number of bytes are received by the Atari and written into the SuperCharger memory. This is controlled through the program.

Function 5: Senddata

entry: ah:5

ds:si source address

cx: number of bytes to be transfered

exit: no registers changed

A certain number of bytes are sent from the SuperCharger memory to the Atari; this is controlled by the program.

Functions 6 to 8:

These functions are reserved; scheduler started for troubleshooting.

Function 9: Getbyte

entry: ah: 9

exit: al: Data, otherwise no registers changed

One data byte is received in AL.

Function 10: Sendbyte

entry: ah: Oah

al: byte to be sent

exit: no registers changed.

One data byte sent in the AL.

8. Extended functions of the TOOLBOX for systems programmers

The extended functions are faster but not as safe as the general functions mentioned previously. This means that the programmer has to be even more informed about the status of the two machines at any given point in time; in return, he has access to a much faster way of communicating. In the following, the extended functions are described in three groups:

1. communications functions on the Atari
2. communications functions on the SuperCharger
3. interrupt services on the SuperCharger

Communications functions on the Atari

Owing to the interrupt structure of the Atari, not all interrupts are permitted during communication on the DMA-bus. The programmer should therefore take care that the interrupts are disabled before addressing and that they are only enabled again after the release of the DMA-bus. Otherwise, DMA-bus conflicts may occur. Typical symptoms for wrong interrupt handling are programs which function occasionally and then crash again for no apparent reason.

While the interrupts are switched off, functions of the Atari operating system (GEMDOS, BIOS, XBIOS) must not be used.

The following are command sequences to switch on and off interrupts on the Atari; supervisor mode 68000 required.

```
vblvec    equ       $00000070       * VBL vector
mfpenb    equ       $fffffa09       * MFP enable reg B
INT_OFFmove.l       vblvec,oldvbl   * save old vector
          move.l    #myvbl,vblvec   * install own VBL
          move.w    sr,savsr        * restore content of SR
          or.w      #$0700,sr       * disable interrupts
```

```
            bset    #7,mfpenb       * enable DMA communication on
            rts
INT_ON      bclr    #7,MFPENB       * DMA communication off
            move.l  oldvbl,vblvect  * restore VBL
            move.w  savsr,sr        * restore SR
            rts
myvbl       rte                     * VBL does nothing
oldvbl      ds.l    1               * old VBL vector
savsr       ds.w    1               * old SR content
```

After calling INT_OFF the SuperCharger can be addressed. Contrary to the functions explained in the previous chapters, the DMA bus is not automatically released.

In principle, the DMA addressing can only be executed by the 68000 whilst the DMA is released by the V30 processor. It is the programmer's responsibility to observe this and to release the DMA-bus at the end of the communication.

SC_LLCAD Low Level: address SuperCharger

Entry: D0.W = $000B

D1.W = DMA-address of the SuperCharger (area 0 to 7)

Exit: -

Attention! Before calling SC_LLCAD interrupts on the Atari must be disabled! The function addresses the SuperCharger on the DMA bus following which the DMA bus should be deselected as soon as possible. If the interrupts are disabled for too long, you may have problems with the Atari keyboard processor (Overruns).

SC_LLCSB Low Level: Send 1 byte to the SuperCharger

Entry: D0.W = $000C

D2.B = byte to be sent

Exit: -

This function sends one byte to the SuperCharger, which was addressed last. The Atari interrupts must be switched off.

SC_LLCGB Low Level: Receive 1 byte

Entry: D0.W = $000D

Exit: D2.B = byte received

This function receives one byte from the SuperCharger which was addressed last. The Atari interrupts must be switched off.

SC_LLCTO Low Level: Tooltransout

Entry: D0.W = $000E
D2.L = Size of the memory area to be sent ((max. 64K)
A0.L = start address from where on data is transfered

Exit: -

Sends a memory area to the SuperCharger which was addressed last. The Atari interrupts must be disabled.

SC_LLCTI Low level: Tooltransin

entry: Do.W = $000F
D2.L = Size of the memory area to be received (max. 64K)
A0.L = start address from where on data is stored

exit: -

Receives a data block from the SuperCharger which was addressed last. The Atari interrupts must be disabled.

Communication functions on the SuperCharger

These functions are not called via the interrupt 60h but with 'call segment:offset' and they end in retf. Following is a list of functions with entrance address and parameters where appropriate.

0000:1000 WAITSLAVE In this routine the SuperCharger waits to be addressed by the Atari. This routine usually introduces a communication and this is the easiest way to do so. Instead of the polling described here you can used the interrupt which we describe later. No parameters are necessary.

0000:1004 QUITSLAVE

The SuperCharger releases the DMA bus. This routine MUST be called before the DMA accesses another unit (i.e. harddisk, floppy disk, laser printer).

0000:1008 SBYTE

The byte in AL is sent to the Atari. No automatic release of the DMA bus!

0000:100C GBYTE

The received byte is returned in AL. No automatic release of the DMA bus!

0000:1010 TOOLMOUT

This functions sends CX bytes from DS:SI to the Atari without automatic addressing or release of the DMA bus.

0000:1014 TOOLMIN

This function receives CX bytes to ES:DI without automatic addressing and release of the DMA bus.

Interrupt Services from the SuperCharger

With the TOOLBOX the programmer has two timers, an 8087-NMI as well as an addressing interrupt at his disposition. When the TOOLBOX is first called, all the interrupts are masked. Those bits which are marked 0 must be 0 when called, or other interrupts may occur which are not documented and not initialised!

The release and the locking of the interrupt is executed by calling interrupt 60h, function # 2.

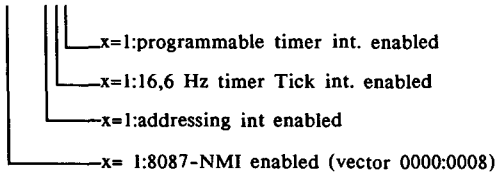
BEWARE!

If any one of these interrupts is enabled, addresses f000:0008 and f000:000a must be kept clear. If no interrupts are used, this is not the case.

Interrupt 60h, function 2:

entry: ah=02

al=x00xxx00



dl: (if the programmable timer is cleared)

0:16666 Hz

1: 8347 Hz

2: 4166 Hz

3: 2083 Hz

4: 1042 Hz

5: 521 Hz

6: 259 Hz

7: 130 Hz

exit: all registers unchanged.

The following table indicates the interrupts which need to be provided for and initialised by the applications program and which are accessed from the TOOLBOX.

NAME	vector#	Trigger
addressing int	int 61h	SC has been addressed
timer1	int 62h	16,6Hz-timer ticker
timer2	int 63h	timer is programmable step by step
NMI	int 2	8087

When starting the TOOLBOX all interrupts are initialised with IRET with the exception that the addressing interrupt releases the DMA-bus beforehand.

9. Example application: reset resistant TOS RAMdisk with the TOOLBOX

SC-RAM.S, a file on the utility disk, contains the source code for a reset resistant Ram disk, which makes the memory of the SuperCharger accessible under TOS. This file was programmed for the TOOLBOX and is extensively documented in the source code. The source code has been written for the DEVPAC assembler. A transfer to other assemblers should not cause any problems.

The V30 program to go with it is called RAMDISK.ASM. The code has been written by Borland for TASM 1.0, but it is likely to run on different assemblers, too. Note that the .286 directive was applied so that you can use V30 commands such as PUSHA etc. The .BIN file, which the Atari needs, is created with the help of the following commands:

```
TASM RAMDISK.ASM;
TLINK RAMDISK;
EXE2BIN RAMDISK.EXE, RAMDISK.BIN
```

The Ramdisk SC_RAM.TTP is also on the disk, ready to run. The Ramdisk has not been optimized for speed or compactness. Its main purpose is to make use of as many routines of the TOOLBOX as possible to make available a universal examplary application to the programmer. If a ramdisk only accessed the SuperCharger memory with functions SC_SM (Send memory) and SC_GM (Get memory), it would do without a V30 program code and it might also be faster. This, however, was not our aim.

As the individual steps for initialising and running SC_RAM are fully documented in the source code, we present only the general procedure here:

The installation of SC_RAM happens in two steps: Firstly, the communication to the SuperCharger is established. To avoid the loss of data after an Atari RESET, SC_STAT verifies, whether the TOOLBOX is already active. If this is not the case, the TOOLBOX is activated through SC_BOOT; then the V30 ramdisk program is loaded with SC_SM and started with SC_EXEC.

Only when these steps have been performed successfully, SC_RAM is installed in the Atari operating system. For this purpose the vectors BPB, MEDIACHANGE and RWABS are "redirected" to the routines. BPB always supplies a prefabricated bios parameter block; only the entry "free clusters" is modified according to the memory capacity of the SuperCharger.

MEDICHANGE always supplies the value "no change". If RWABS is being called, SC_RAM sends the relevant command code for each sector to the V30 program as well as the sector number. Then the data is being received or sent.

Command syntax of the ramdisk:

In the command line of the ramdisk you can indicate which SuperCharger is to be used as a ramdisk and you can name the drives. The exact syntax is explained in file SC_RAM.S. If no command line is indicated or SC_RAM.TTP has been renamed to SC_RAM.PRG, then unit address 3 will come up as drive I:. This corresponds for example with command line "3I".

10. Appendices

Appendix A: Quick Reference Tables for V30 TOOLBOX functions.

Function 0: Terminate

entry: ah:0

exit: no escape from interrupt, scheduler running.

Function 1: Killtb

entry: ah:1

exit: no escape from interrupt, idle-loop running

Function 4: Receivedata

entry: ah:4

exit: es:di target address of data block

cx: number of bytes to be received

exit: no registers changed

Function 5: Senddata

entry: ah:5

ds:si source address

cx: number of bytes to be transfered

exit: no registers changed

Function 9: Getbyte

entry: ah: 9

exit: al: date, otherwise, no registers changed

Function 10: Sendbyte

entry: ah: 0ah

al: byte to be sent

exit: no registers changed

Appendix B: Quick reference tables for MC68000 TOOLBOX functions.

In the D0 register each function supplies the result code

Function SC_STAT Get Toolbox Status

entry: D0.W = $0002

D1.W = Device

Function SC_BOOT Boot Toolbox

entry: D0.W = $0003

D1.W = Device

Function SC_SD Send data

entry: D0.W = $0004

D1.W = Device

D2.L = Size

A0.L = address

Function SC_GD Get data

entry: D0.W = $0005

D1.W = Device

D2.L = Size

A0.L = address

Function SC_SM Send memory (scheduler)

entry: D0.W = $0006

D1.W = Device

D2.L = Size

D3.L = Segment, offset

A0.L = address

Function SC_GM Get memory (scheduler)

entry: D0.W = $0007

D1.W = Device

D2.L = Size

D3.L = Segment, offset

A0.L = address

Function SC_EXEC Execute program (scheduler)

entry: D0.W = $0008

D1.W = Device

D3.L = segment, offset

Function SC_SB Send 1 byte

entry: D0.W = $0009

D1.W = Device

D2.B = Byte to send

Function SC_GB Get 1 Byte

entry: D0.W = $000A

D1.W = Device

exit: D0.L = result code

D2.B = received byte

Appendix C: Memory layout of the SuperCharger when TOOLBOX is running.

Certain early versions of SuperCharger were optionally supplied with 512kb of Ram and the information relating to this is given below for those users who have not yet upgraded to 1MB.

The memory layout of a 512K Ram SuperCharger is divided in three partitions. This has to do with the prerequisites of the PC emulation (video storage, BIOS area). We therefore list the memory partition for 512KB and 1MB separately.

SuperCharger with 512 KB RAM

0000:0000 - 0000:03FF	reserved for INT vector table
0000:0400 - 0000:2FFF	reserved for SuperCharger Toolbox
0000:3000 - 6000:7FFF	Free RAM area
B000:8000 - B000:FFFF	Free RAM area
F000:0000 - F000:FFFF	Free RAM area (see [1])

SuperCharger with 1MB RAM

0000:0000 - 0000:03FF	reserved for INT vector table
0000:0400 - 0000:2FFF	reserved for SuperCharger Toolbox
0000:3000- F000:FFFF	free RAM area (see [1])

N.B. [1] Where special interrupts are being enabled as discussed in the chapter "Extended functions of the TOOLBOX for systems programmers", the following also applies:

F000:0008 - F000:000B	reserved for SuperCharger Toolbox

Appendix D: SCTB.INC

```
*************************************************************
*                                                           *
*    This Source-Code must not be modified, extended or     *
*    abbreviated. The application program is to read this   *
*    file with INCLUDE.                                     *
*    Only then we can guarantee that your programs will     *
*    be compatible with future versions of the              *
*                                                           *
*                    SuperCharger TOOLBOX                   *
*                                                           *
*    If this is impossible for you for your application     *
*    please write to                                        *
*                                                           *
*         Beta Systems AG                                   *
*         Toolbox Support                                   *
*         Staufenstrasse 42                                 *
*         D - 6000 Frankfurt                                *
*         Germany                                           *
*                                                           *
*************************************************************

     FUNCTION CALLS

* are made with TRAP 4 with the function code in register
D0. The other registers must be loaded as detailed below.
For more information see documentation on the SuperCharger
TOOLBOX.

SC_STAT   equ  $0002 * GET STATUS          D1=DEVICE
SC_BOOT   equ  $0003 * BOOT TOOLBOX        D1=DEVICE
SC_SD     equ  $0004 * SEND DATA           D1=DEVICE,D2=SIZE,A0
                                           =ADDR
SC_GD     equ  $0005 * GET DATA            D1=DEVICE,
                                                A0=ADDR
SC_SM     equ  $0006 * SEND MEMORY         D1=DEVICE,D2=SIZE
                                           D3=SC_ADDR,A0=ADDR
SC_GM     equ  $0007 * GET MEMORY          D1=DEVICE,D2=SIZE,
                                           D3=SC_ADDR,A0=ADDR
SC_EXEC   equ  $0008 * EXECUTE PRG.        D1=DEVICE D3=SC_ADDR
SC_SB     equ  $0009 * SEND 1 BYTE         D1=DEVICE,D2=DATA
SC_GB     equ  $000A * GET 1 BYTE          D1=DEVICE (returns
                                              byte in D2)
SC_LLCAD  equ  $000B * LLC ADDRESS DMA     D1=DEVICE
SC_LLCSB  equ  $000C * LLC send byte       D1=DEVICE,D2=data
SC_LLCGB  equ  $000D * LLC get byte        D1=DEVICE (returns
                                              byte in D2)
SC_LLCTO  equ  $000E * LLC Tooltransout    D1=DEVICE, D2=SIZE
                                           A0=ADDR
SC_LLCTI  equ  $000F * LLC Tooltransin     D1=DEVICE, D2=SIZE
                                           A0=ADDR
```

SC_CHECK Check Toolbox

Function: Verifies whether resident part of the TOOLBOX has been loaded.

Input: - none -

Output: D0 = result code
D1 = version number of TOOLBOX in ASCII double word.

Note: This function is written in source code as it verifies the existence of the TOOLBOX and can therefore not be part of it.

```
trap4vec  equ  $90            * Trap 4 vector
xbraid    equ  $58425241      * XBRA identification
sctbid    equ  $53435442      * SCTB identifcation

          BRA       SC_AROUND

SC_CHECK  move.l    trap4vec,a0         * get trap 4 vector
          cmp.l     #xbraid,-12(a0)     * check XBRA ID
          bne       sc_chk01
          cmp.l     #sctbid,-8(a0)      * check SCTB ID
          bne       sc_chk01

          moveq     #0,d0               * result code
          move.l    -16(a0),d1          * version number
          rts

sc_chk01  moveq     #0,d1               * version number n/a
          moveq     #-1,d0              * result code -1

SC_AROUND nop
```

FUNCTION CODES

0	ok, no fault
-1	resident TOOLBOX not loaded
-2	reserved
-3	SuperCharger not connected
-4	timeout (SuperCharger program does not respond)
-5	protocol failure (e.g. data expected, memory block received)
-6	ABIO hotkey recognized (see SCCHECK)
-7	reserved
-8	illegal function call

```
sce_okay  equ  0    * ok
sce_na    equ  -1   * TOOLBOX not available (SCTB.PRG
                      not resident)
sce_nc    equ  -3   * SuperCharger not connected
sce_to    equ  -4   * timeout (no response from SuperCharger
                      program)
sce_pf    equ  -5   * protocol failure
sce_abio  equ  -6   * ABIO hotkey detected
sce_if    equ  -8   * illegal function call
```

Appendix E: SC_RAM.S

```
File SC_RAM.S version 1.30

makes the SuperCharger memory under TOS accessible as a
ramdisk. Uses the SuperCharger TOOLBOX for programming.

SC_RAM.TTP  3G
            !!
            !+-- preferred label for disk drive
            !    identification. If already used, the first
            !    free label is chosen. Default: I
            !
            +-----DMA device address. Default is "3"

If SC_RAM.TTP is renamed SC_RAM.PRG, defaults are being used
for initialisation, i.e. device address 3 and preferred
label "I".

* System variables

hdv_bpb    equ  $472          * Bios parameter block routine
hdv_ rw    equ  $476          * read/write routine
hdv_med    equ  $47e          * media change routine
drvbits    equ  $4c2          * bit vector active drives
hz200      equ  $4ba          * 200 Hz timer

* Inherent constants

err_drv    equ  $01           * "more than 32 drives"
err_aktiv  equ  $02           * "ABIO active" message
err_retry  equ  $03           * "retry, abort?" message
signup     equ  $04           * signon message
warmstart  equ  $05           * warmstart message

* Command codes for communication with the V30 program part

cmd_reset  equ  $0000         * reset ramdisk, delete
                                memory
cmd_size   equ  $0001         * get ramdisk size
cmd_read   equ  $0002         * read sector(s)
cmd_write  equ  $0003         * write sector(s)

start      jmp       install  * install ramdisk resident

           INCLUDE   SCTB_INC.S * TOOLBOX software interface
```

RAMDISK KERNEL: GET BIOS PARAMETER BLOCK

Transfers the Bios parameter block to the drive indicated. Verifies whether ramdisk has been accessed. If not goes to the former BPB vector. Otherwise ramdisk BPB is transfered.

```
          dc.b      'XBRA'          * XBRA label
          dc.b      'SCRD'          * program label "SCRD"
old_bpb   ds.l      1               * old BPB vector
my_bpb    move.w    drive_id,d0     * ramdisk label
          cmp.w     4(sp),d0        * ramdisk accessed?
          beq       our_bpb
          move.l    old_bpb,a0      * load old vector
          jmp       (a0)            * and jump to it

our_bpb   move.l    #bpb,d0         * transfers ramdisk BPB
                                      back
          rts                       * that's all!
```

RAMDISK KERNEL: MEDIA CHANGE

Verifies whether the medium was changed. As a change is not possible when the SuperCharger is running, the message will always be "no change"

```
          dc.b      'XBRA'          * XBRA label
          dc.b      'SCRD'          * SuperCharger ramdisk
odl_med   ds.l      1
old_med   move.w    drive_id,d0     * ramdisk label
          cmp.w     4(sp),d0        * ramdisk accessed?
          beq       our_med         * load old vector
          jmp       (a0)

our_med   moveq.l   #0,d0           * no media change
```

RAMDISK KERNEL: READ/WRITE

```
* Read and write sectors
          dc.b      'XBRA'          * XBRA label
          dc.b      'SCRD'          * SuperCharger ramdisk
old_rw    ds.l      1
my_rw     move.w    drive_id,d0     * ramdisk label
          cmp.w     14(sp),d0       * ramdisk accessed?
          beq       our_rw
          move.l    old_rw,a0       * load old vector
          jmp       (a0)
```

```
our_rw    move.w    12(sp),sector  * start sectors
          move.w    10(sp), number * number of sectors
          beq       ready          * ready when sectors 0
          move.l ,  6(sp),startadr * start address
          btst      #0,5(sp)       * read or write?
          beq       our_read
          bra       our_write

* Read sectors. Each sector is read individually.

our_read  moveq.b   #sc_sb,d0      * send task code as a
                                     byte
          move.w    dev_id,d1
          moveq.b   #cmd_read,d2   * send command code to
                                     V30 program
          trap      #4
          bmi       retry          * error handling

          moveq.b   #sc_sb,d0      * send single byte
          move.w    dev_id,d1
          move.b    sector+1.d2    * send sector LSB
          trap      #4
          bmi       retry

          moveq.b   #sc_sb,d0      * send single byte
          move.w    dev_id,d1
          move.b    sector,d2      * send sector MSB
          trap      #4
          bmi       retry

          moveq.b   #sc_gd,d0      * receive data
          move.w    dev_id,d1
          move.l    #$00000200,d2  * length 512 bytes
          move.l    startadr.a0    * data address
          trap      #4             * toolbox
          bmi       retry          * error handling

          add.l     #512,startadr  * count up start address
          addq.w    #1,sector
          subq.w    #1,number      * count down sector
                                     counter
          bne       our_read       * continue until all
                                     sectors ready.

ready     moveq     #0,d0          * no error
          rts                      * end OUR_READ
```

```
our_write moveq.b   #sc_sb,d0       * send task code as a
                                      byte
          move.w    dev_id,d1
          moveq.b   #cmd_write,d2   * send command code to
                                      V30 program
          trap      #4
          bmi       retry           * error handling

          moveq.b   #sc_sb,d0       * send single byte
          move.w    dev_id,d1
          move.b    sector,d2       * send sector MSB
          trap      #4
          bmi       retry

          moveq.b   #sc_sb,d0       * send data
          move.w    dev_id,d1
          move.l    #$00000200,d2   * length 512 bytes
          move.l    startadr,a0     * data address
          trap      #4              * toolbox
          bmi       retry           * error handling

          add.l     #512,startadr   * count up start address
          addq.w    #1,sector       * next sector
          subq.w    #1,number       * count down sector
                                      counter
          bne       our_write       * continue until all
                                      sectors ready
          bra.s     ready
```

* error handling. The error will be reported, then the text "R=RETRY, A=ABORT -->'. The relevant action will then be taken.

```
retry     bsr       message         * report error
          move.w    #err_retry,d0   *"retry, abort?"
          bsr       message

retry1    bsr       getkey          * wait for keying in
          cmp.b     #'R',d0-        * retry?
          beq       our_rw          * again!
          cmp.b     #'A',d0         * abort?
          bne       retry1          * if not wait for new
                                      key

          moveq.l   #-1,d0          * "general error"
          rts
```

* ═══════════════════════

```
*
* Install ramdisk in the Atari
*
*

install:  move.l    4(a7),a5        * basepage address to A5
          move.l    a5,basepage     * save basepage address

* Calculate memory required

          move.l    12(a5),d0       * text
          add.l     20(a5),d0       * data
          add.l     28(a5),d0       * bss
          add.l     #$100,d0        * basepage
          move.l    d0,mysize       * length of the reserved
                                      Atari memory

* Stackpointer to program end + $0400

          add.l     #$400,d0        * stack during
                                      installation
          move.l    d0,d1
          add.l     a5,d0           * end address in D0
          bclr.l    #0d0            * make address even
          move.l    d0,sp           * set inherent
                                      stackpointer

* Release remaining memory

          move.l    d1,-(sp)        * length of program +
                                      stack + basepage
          move.l    a5,-(sp)        * from basepage
          clr.w     (-(sp)
          move.w    #$4a,-(sp)      * SETBLOCK: reserve
                                      memory area
          trap #1
          add.l     #12,sp

In the following you must switch to Supervisor mode as
system variables are being accessed.

          clr.l     -(sp)           *
          move.w    #32,-(sp)       * GemDos: swich on
                                      SUPERVISOR
          trap      #1
          addq.l    #6,sp
          move.l    d0,ssave        * save user stackpointer
```

```
* Parsing of the command line

        move.l    basepage,a0     * basepage address to A0
        move.b    $80(a0),d0      * command line length to
                                    D0
        beq       init00          * use defaults if command
                                    line length zero

* First item: DMA address. Permitted range: "0" to "7".
  Default: "3"

        moveq     #0,d1
        move.b    $81(a0),d1      * get DMA device number
        sub.b     #'0',d1
        bcs       init00          * verify area 0..7
        cmp.b     #'7'-'0'+1,d1
        bcc       init00
        move.w    d1,dev_id       * save device ID
        cmp.b     #1,d0           * one parameter only?
        beq       init00          * Then we are ready

* Second item: Preferred drive label. Permitted range:
  "A" to      "P".

        move.b    $82(a0),d1      * get drive label
        and.w     #$5F,D1         * fold to upper case
        sub.b     #'A',d1
        bcs       init00          * verify range A..Z
        cmp.b     #'P'-'A'+1,d1
        bcc       init00
        move.w    d1,drive_id     * save default drive
                                    label

* Verify whether the resident part of the TOOLBOX has been
loaded. If not, error message comes up.

init00  bsr       sc_check        * verify whether toolbox
                                    existent
        bmi       abort           * if not abort
```

At this point SC_STAT verifies whether the TOOLBOX is already active in the SuperCharger. If an AHDI.TOS hotkey is recognized, an error message will be reported. If a TOOLBOX is recognized, it will be assumed that the ramdisk is already active (e.g. after an Atari Reset). In this case the SuperCharger memory will not be deleted so that the contents of the ramdisk are still available.

```
init01    move.w    #sc_stat,d0     * verify toolbox status
          move.w    dev_id,d1
          trap      #4
          bne       init01a         * if not toolbox,
                                      continue test
          move.w    #warmstart,d0   * report warmstart
                                      message
          bsr       message
          bra       init04          * no reset

init01    cmp.w     #sce_abio,d0    * abio hotkey active?
          bne       init03          * any other message leads
                                    * to coldstart

* An ABIO hotkey has been recognized. The user can chose
beween abort (ramdisk will not be installed and hotkey
remains intact) and reset (ramdisk overwrites ABIO hotkey).

          move.w    #err_active,d0 * "abio hotkey active"
          bsr       message

init02    bsr       getkey
          cmp.b     #'R',D0
          beq       init03          * [R]eset --> coldstart
          cmp.b     #'A',D0
          bne       init02          * if not [a]bort then
                                      loop

          move.w    #sce_abio,d0    * identifcation of abio
                                      error
          bra       abort

* The SuperCharger is reset, the ramdisk program is loaded
and started ( cold start; SuperCharger ram is erased).

init03    move.w    #sc_boot,d0     * boot SuperCharger
                                      toolbox
          move.w    #dev_id,d1
          trap      #4
          bmi       abort           * error handling if
                                      necessary

          move.w    #sc_sm,d0       * send memory
                                      (ramdisk program)
          move.w    dev_id,d1       * DMA device address
          move.l    #v30_size,d2    * size of V30 program
          move.l    #v30_addr,d3    * target address in
                                      SuperCharger
          lea       v30_start,a0    * source address in
                                      Atari memory
```

```
        trap    #4
        bmi     abort           * error handling

        move.w  #sc_exec,d0     * start ramdisk in
                                  SuperCharger
        move.w  dev_id,d1
        move.l  #v30_addr.d3    * start address of the
                                  V30 program
        trap    #4
        bmi     abort
```

* The SuperCharger ramdisk is now activated; it receives the command RESET which erases the memory.

```
        moveq.b sc_sb,d0        * send byte
        move.w  dev_id,d1
        moveq.b #cmd_reset,d2   * erase memory in
                                  SuperCharger
        trap    #4
        bmi     abort
```

* Now the size of the available ram memory is being asked for; the value is written into the Bios parameter block BPB

```
init04  moveq.b #sc_sb,d0       * send data byte
        move.w  dev_id,d1
        moveq.b #cmd_size,d2    * command "get size" to
                                  V30 program
        trap    #4
        bmi     abort           * if error, abort.
```

```
* Type in CMD_SIZE; ;the V30 program will send 1 byte back.
$01 means 512KB SuperCharger, $00 means 1MB SuperCharger.

          moveq.ab  #sc_gb,d0       * get data byte
          move.w    dev_id,d1
          trap      #4
          bmi       abort

* Note for calculation of free clusters:
*
* In the SuperCharger meory, $8000 bytes (32k) are reserved
for the TOOLBOX and the program. Therefore:

1 MB machine: $F8000 free memory / $200 sectorlength = 1984
sectors
512k-machine:  $78000 free memory / $200 sector length =
960 sectors

* 18 sectors are subtracted  for 2 FATs and the directory
(see BPB definition). The remaining number is divided by 2,
as one cluster consists out of 2 sectors. Hence there are
983 free data cluster for 1MB and 471 free data clussters
for 512 KB.

          cmp.b     #$01,d2         * if answer $01 go to
                                        SC512K
          beq       sc512k
          move.w    #983,bpb_size   * 983 data clusters if
                                      1 MB
          bra.s     sclmeg
sc512K    move.w    #471,bpb_size   * 471 data cluster if
                                      512k
sclmeg    nop

* The initialisation process of the SuperCharger is now
finished. In the process that follows the ramdisk is
installed in the Atari operating system.

* Seek appropriate drive label.

          move.l    drvbits,d0      * bit vector of the
                                      active drives
          move.w    drive_id,d4     * load preferred drive
                                      into D4
          bset      d4,d0           * set relevant bit
          beq       init06          * if bit set to 0, end.

          moveq.l   #0,d4           * otherwise from A: seek
                                      free label
          moveq.l   #31,d5          * abort after 32 tests
```

```
init05  bset    d4,d0           * test bit and set
        beq     init06          * if 0, free label found
        addq.w  #1,d4           * test next bit
        dbf     d5,init05

        moveq   #err_drv,d0     * error! No more free
                                  bits
        bra     abort           * error handling and exit

init06  move.w  d4,drive_id     * save drive label
        move.l  d0,drvbits      * save bit vector for TOS

        add.b   #'A',d4         * transfer drive label
                                  into letter
        move.b  d4,drive        * include in signup
                                  message

* Installation into vectors BPB, MEDIACH and RWABS

    move.l  hdv_bpb,old_bpb     * include in BPB vector
    move.l  #my_bpb,hdv_bpb

    movel   hdv_rw,old_rw       * include in RW vector
    move.l  #my_rw,hdv_rw

    move.l  hdv_med,old_med     * include in media change
    move.l  #my_med,hdv_med

* Report signup and wait for a second so that user can read
the selected drive label.

    move.w  #signup,d0          * signup message ID
    bsr     message
    bsr     delay               * wait for 1 second

Ready! Tidy up stack and switch off Supervisor.

    move.l  ssave,-(sp)         * restore user stack
    move.w  #32,-(sp)           * SUPER
    trap    #1
    addq.l  #6,sp

Make ramdisk program resident with PTERMRES so that it
cannot be overwritten.

    move.w  #0,-(sp)            * retransfer error code 0
                                  (OKAY)
    move.l  mysize,-(sp)        * size of reserved memory
    move.w  #49,-(sp)           * ptermres
    trap    #1                  * program is finished.
```

```
*-------------------------------------------------------
*    HELP ROUTINES
* -------------------------------------------------------
* MESSAGE: Reports the appropriate error message from
  result code D0
*
* If MESSAGE is called with a positive error number in D0, a
user defined error message is transfered from table USERERR.
In case of a negative field number the appropriate TOOLBOX
error message is transferred from table TOOLERR.

message   move.w    d0,-(sp)      * safe error number
          pea       errhead       * 'ERROR:' display prefix
          move.w    #9,-(sp)      * print string
          trap      #1
          addq      #6,sp

          lea       usererr,a0    * user error message for
                                    for D0>0
          move.w    (sp)+,d0      * retrieve error label
                                    from stack
          bpl       msg01         * if D0>0, go to MSG01

          lea       toolerr,a0    * toolbox message for
                                    d0<0
          neg.w     d0            * make number positive

msg01     subq.w    #1,d0         * decrement error number
          neg.w     d0            * correct message found
                                              when 0

msg02     tst.b     (a0)+         * seek next 0-byte
          bne       msg02
          bra       msg01         * on until message has
                                    been found

msg03     move.l    a0,-(sp)      * address of error text
          move.w    #9,-(sp)      * print line
          trap      #1
          addq.l    #6,sp
          rts

* If an error occurs while installing ramdisk ABORT is
called. The appropriate error message is reported; the
program will wait for a key to be pressed, restore the stack
and then finish. In this case the ramdisk is not installed
resident in the memory.

abort     bsr       message       * evaluate return code
          bsr       delay         * leave at least 1 sec
          bsr       getkey        * wait for key press

          move.l    ssave,-(sp)   * restore user stack
          move.w    #32,-(sp)     * SUPER
```

```
          trap      #1
          addq.l    #6,sp

          clr.w     -(sp)           * terminate process
          trap      #1              * so long and thanks
                                      for all the fish
* GETKEY  waits for a keypress. The result will be folded to
upper ;case by Anding with #$005F,d0.

getkey    move.w    #8,-(sp)        * input w/o echo
          trap      #1
          addq.l    #2,sp
          and.w     #$005F,d0       * convert lowercase to
                                      uppercase

*DELAY waits for 1 sec, while it accesses the tick counter
_hz200.

delay     move.l    hz200,d0        * 200 Hz counter
          add.l     #400,d0         * plus 2 seconds
delay01   cmp.l     hz200,d0
          bcc       delay01         * wait for time to pass

data                                * start of DATA SEGMENT

dev_id    dc.w      3               * device address,
                                      default 3
drive_id  dc.w  +   8               * drive label,
                                      default I: (Nr. 8)

bpb       dc.w      $200            * bytes per second
          dc.w      2               * sectors per cluster
          dc.w      1024            * bytes per cluster
          dc.w      7               * root dir sectors
          dc.w      5               * FAT length
          dc.w      6               * 2nd FAT start
          dc.w      18              * 1st data sector
bpb_size  dc.w      0               * free DATA cluster
                                      (excluding FAT, DIR)
          dc.w      0               * 12bit FAT
          dc.l      0               * reserved

v30_start incbin    ramdisk.bin     * V30 program code
v30_size  equ       *-v30_start     * length of program
v30_addr  equ       $03000000       * start address
                                      0300:0000

errhead   dc.b      13,10,10,'SC-RAMDISK: ',0
```

```
toolerr   dc.b    7,'resident toolbox SCIB.PRG not loaded!
                    0
          dc.b    7,'(reserved)',0
          dc.b    7,'SuperCharger not connected!',0
          dc.b    7,'TIMEOUT! SuperCharger does not
                    respond',0
          dc.b    7,'protocol error',0
          dc.b    7,'ABIO hotkey recognized (can be
                    activated with ABIO.TOS)'
          dc.b    7,'(reserved)',0
          dc.b    7,' wrong command code!',0
userr     dc.b    'all 32 drives are already active; this
                   is the limit!'
          dc.b    0               * message 2 follows
          dc.b    'ABIO hotkey still running.
                   [A] = abort ramdisk,'
          dc.b    '[R] = hotkey RESET ?'
          dc.b    0               * message 3 follows
          dc.b    'SELECT: R = RETRY, A = ABORT ---->'
          dc.b    0               * message 4 follows
          dc.b    'SuperCharger ramdisk as drive'
drive     dc.b    '?: installed.',0
          dc.b    'warm start, ramdisk is already in
                   SuperCharger'
          dc.b    'installed.',0

bss

startadr  ds.l    1         * start address in read/write
sector    ds.w    1         * sector number
number    ds.w    1         * number of sectors

basepage  ds.l    1         * basepage address
ddsave    fd/1    1         * user stack pointer (USP)
mysize    ds.l    1         * size of the reserved
                              Atari memory
```

Appendix F: Ramdisk program

```
page 62,132
.286c                    * enable v30 commands
;-------------------------------------------------------------;
TITLE RAMDISK
;
;
;version number          0.00
;
;example for the use of the SuperCharger TOOLBOX
;
;
;
;-------------------------------------------------------------;
;EQUATES
terminate       equ  0    ;program is terminated scheduler
running
killtb          equ  1    ;exit from toolbox (idle-loop)
receivedata     equ  4    ;data block cx butes is loaded into
es:di
senddat         equ  5    ;data block cx bytes is sent from
ds.si
getbyte         equ  9    ;receives 1 byte into al
sendbyte        equ  10   ;sends contents of al to ATARI
;    (All numbers not used are for internal use or are
reserved for expansion)
;-------------------------------------------------------------;
CODE SEGMENT para public 'code'
ASSUME CS:CODE
org  0

init:     cli                   ;initialise stack
          mov  ax, cs           ;toolbox makes available
          mov  ss, ax           ;about 0f0h words
          mov  sp, offset stacktop
main:     mov  ah, getbyte;     ;wait for task number
          int  60h              ;comes back when Atari reports
          xor  ah, ah           ;
          shl  al, 1            ;index for function call
          mov  si, ax           ;ready for indexed function
call
          call word ptr  cs: [task tab][si]
          jmp  main             ;and on we go

;-------------------------------------------------------------
```

```
reset     label near          ;erases memory from 0000:8000
with 00.
          mov  bx, 1000h      ;start segment is 1000h
          xor  ax, ax         ;memory value 0
          mov  di, ax         ;initialise offset 0
r00       mov  es, bx         ;segment to describe
          mov  cx, 8000h      ;set counter to 64k
          rep  stosw          ;writes 0 in memory
          add  bx, 1000h      ;next segment
          jnz  r00            ;loop for all segments
          mov  es, bx         ;segment now 0, from offset
8000h
          mov  cx, 4000h      ;32k
          mov  di,  8000h     ;offset
          rep  stosw          ;that's all
          retn
;-------------------------------------------------------
ramsize   label near          ;we select the easy option...
          push 0c000h         ;test segment
          pop  ds             ;to ds
          xor  si, si         ;offset 0
          mov  bx, word ptr ds:[si]     ;save old values
          move word ptr ds:[si], 1234h  ;test pattern
          xor  al, al         ;default response 1M
          cmp  word ptr ds:[si], 1234h ;listed now?
          je   einmeg         ;yes...
          mov  al, 1          ;answer for ATARI (512k)
einmeg:   mov  byte ptr cs:[megaflag], al
          mov  ah, sendbyte   ;toolbox function number
          int  60h            ;sends value in al
          retn

;-------------------------------------------------------
read      label near          ;reads from ramdisk
          call getaddress     ;supplies segment address in
bx
          mov  ds, bx         ;initialise toolbox call
          xor  si, si         ;start offset always 0
          mov  cx,512         ;always 1 sector
          mov  ah, senddat    ;toolbox functions# send data
block
          int  60h            ;toolbox call
          retn
;-------------------------------------------------------
write     label near          ;written in ramdisk

          call getaddress     ;supplies segment address in
bx
          mov  es, bx         ;initialise toolbox call
          xor  di, di         ;start offset always 0
          mov  ah, receivedata      ;function# receive data
                                     block
          int  60h                  ;toolbox call
          retn
```

```
;-------------------------------------------------
;getaddress: supplies address of the accessed sector
;in bx
;Note:
In the 1M SuperCharger the ramdisk data use memory area
0000:8000 to f000:ffff.
In the 0.5M SuperCharger the following RAM is available:
0000:8000 to 6000:7fff, b000:8000 to b000:ffff as well as
f000:0000 to f000:ffff.
In addition area 0000:0000 to 0000:2fff is used in both
versions for the TOOLBOX and the interrupt vector
table. The memory area which the TOOLBOX can
utilise starts at 0000:3000.  It is here that the ramdisk
program itself resides.

entry:    -
exit:     segment for accessed sector in bx
          ax not saved
;-------------------------------------------------
getaddress      label     near
          mov   ah, getbyte    ;get sector number lsb
          int   60h            ;wait for ATARI
          mov   bl, al         ;toolbox call
          mov   ah, getbyte    ;get sector number msb
          int   60h            ;toolbox call
          mov   bh, al         ;sector number now in bx
          shl   bx, 5          ;*32 makes segment for sector
          add   bh, 8h         ;add reserved memory area
          cmp   byte ptr cs:[megaflag], 0 ;special case
          jnz   checkexp       ;512 charger, test memory gaps
          retn
checkexp  cmp   bh, 68h        ;1st gap from 6800:0
          jae   check1         ;further tests necessary
          retn
check1:   add   bh, 50h        ;on to b800h
          cmp   bh, 0c0h       ;lower area without problems
          jae   check2         ;
          retn
check2:   or    bh, 0f0h       ;remaining only fxxx
          retn

;-------------------------------------------------
megaflag        db    0        ;if 0, 1M RAM, if 1:0.5M
even
;-------------------------------------------------
tasktab         dw    reset    ;erase memory
                dw    ramsize  ;whether SC 0.5 or 1M RAM
                dw    read     ;read from ramdisk
                dw    write    ;writes onto ramdisk
;-------------------------------------------------
stapel          dw    1feh dup (?)  ;
stacktop        dw    ?
y;-------------------------------------------------
CODE ENDS
END
```

In addition to SuperCharger, Condor supplies a wide range of hardware including the full range of Atari Computers, disk drives, multisynch monitors and other hardware.
Call us for details. Educational discounts are also available.

CONDOR TRADING LTD.
6 Bacchus House
Calleva Park
Aldermaston
Berks RG7 4QW
Tel.: (0734) 810066
Fax.: (0734) 819791