THE SERVANT

Table of Contents

Introduction

Read Me / License Terms

This is NOT free software!!!! THE SERVANT is SHAREWARE. If you intend to use it on a regular basis, either as a RAM resident program or programmed into an EPROM, you are required to pay a reasonable user license fee (see below). Using the shareware route, it is possible to offer you a program which would otherwise not be available. The feasibility of a venture like this depends on the honesty of the users.

You will be entitled to technical support through correspondence. You will also encourage further development of THE SERVANT; talented C128 programmers are definitely on the endangered species list. Want more 64/128 software? The limited support from 64/128 users is the main cause of the continuing exodus of talented programmers to the ST, Amiga, MAC and MSDOS platforms. Send shareware contributions! Your name and address will not be sold or otherwise disclosed.

DISTRIBUTION

Anyone is allowed, and invited, to copy and spread the four .sfx files which contain THE SERVANT on a non-profit basis. That is, you are NOT allowed to charge money for THE SERVANT except to cover the price of the media. However, a small profit is allowed for public domain/shareware libraries and BBS systems. From version 4.82 and onwards, only the four .sfx files released by the author may be copied and distributed, and all four of them MUST always be included. The distribution of any of the material contained therein in any other way is not permitted. Re-compressing of the material is not allowed except for strictly personal use.

MODIFICATIONS

THE SERVANT can be examined and/or modified at will. However, modified versions are NOT allowed to be distributed to others. Any reference to the author's name, address or license terms is NOT allowed to be removed from THE SERVANT's code or documentation.

MAKING EPROMS

Anyone is allowed to produce SERVANT EPROMs on a non-profit basis for personal use, friends or computer club members. Please note however, that each and everyone who mount a SERVANT EPROM into his or her computer are REQUIRED to pay a user license fee. If you have no means of making EPROMS, the author will do it for you for a small fee.

FEES FOR LICENSES and EPROM PROGRAMMING:

Currency:	**Norwegian**	**US**	**British**
User license:	NOK 65.00	USD 10.00	GBP 5.20
EPROM programming EA*:	NOK 32.00	USD 5.00	GBP 2.60
Shipping:**	NOK 26.00	USD 4.00	GBP 2.10

PLEASE NOTE:

* The EPROM programming fee DOES NOT include the EPROM itself. The EPROM must be supplied by yourself along with your payment.

** The shipping charge applies worldwide. Covers a maximum of 5 EPROMS.

Please send cash (money bills). The bank now charge about $8 (US) for cashing in each and every foreign currency check. That money is not much use for the author, or the user. Payment of the above amounts in any of the indicated currencies will be acceptable regardless of exchange rate fluctuations. No credit cards accepted, sorry. You will NOT be ripped off!

If you upgrade from one version to another, no additional license payment is required, provided that you previously have paid your license fee.

The author's name and address:

Alf H. Jonassen
Fjellvegen 16A
N-4300 Sandnes
Norway

DISCLAIMER:

Neither the author nor any other person, firm or other party can be made liable to any damage or loss whatsoever, either expressed, implied or indirect, from the installation, use or other relations with THE SERVANT or its documentation. Any handling, use or otherwise is entirely at your own risk. No warranty is provided regarding THE SERVANTs function or fitness for any particular purpose.

Introduction to THE SERVANT

A ROM-based utility package designed

Exclusively for the Commodore 128

Hi! And welcome to THE SERVANT. The guys who designed the Commodore 128 did a great job. Both you and I have enjoyed the fruits of their efforts. In the design of the C128 they included a feature which has only rarely been put to use. If you open your trusty computer there seems to be something missing. Among all the black, rectangular chips there's a vacant socket. What kind of chip might fit? An extra processor? Nah! Extra memory? Well kind of, the socket is meant to contain a ROM chip of up to 32 kilobytes. A creative programmer is free to use this space in whatever way he or she (I would like to see female C128 programmers!) sees fit. A lot of goodies can be put into that space.

This is where THE SERVANT comes in. THE SERVANT offers a plethora of nifty utilities and convenience features designed to make your computing life easier and more fun. This goes even if you never use the 128-mode at all. THE SERVANT is designed with maximum convenience, compatibility and security in mind, making it unusual or even unique in many ways. But they all say that don't they?

Well, see for yourself and enjoy!

The author welcomes all kinds of reactions from the users. Complaints, suggestions, technical or other questions, comments, corrections, reports on errors and bugs, nitpicks, boast or whatever. Everyone, please write!

THE SERVANT and this manual has been produced using the following: C128, Master 41, 1570, 1571, 1581 disk drives, 1750 Ram Expansion Unit, Promenade C1 EPROM programmer, Brother HR15 printer with Super G interface, Midnight Assembly System machine language assembler, The Write Stuff word processor.

The author wishes to thank all people using and supporting the 128 (and then some) who, mostly without knowing, have made this possible. One person deserves a special mention though. I wish to thank Matthew Montchalin for his wonderful Midnight Assembly System, the assembler used to create THE SERVANT. Without it, THE SERVANT would never have seen the light of day.

This work is dedicated to all the good people around the world who are working for peace, understanding, non-violence and the savior of our wonderful blue planet.

And last but not least; please bear with me as you read this manual. I am not native to the English language. I was born, raised and still live in Norway, the land of the midnight sun. Norway is known for its mountains and fiords and for being a peaceful corner of the world. Thank you.

IMPORTANT NOTE:

You'll probably spend your first hours using THE SERVANT in computer RAM. THE SERVANT is somewhat hampered when working in that configuration. There will be less buffer space available for the Disk Copier and File Copier. Furthermore, some programs won't launch properly when using THE SERVANT in RAM. Especially, using RAMDOS will produce severe problems in some cases. Also, you can't load programs of more than 100 blocks in size, as bigger programs will overwrite THE SERVANT and crash it. You may also discover other quirks when using THE SERVANT from computer RAM. However, all these problems will disappear when THE SERVANT is allowed to live on a chip. THE SERVANT working as a RAM-resident program is only for your review, customizing and then for making a copy that can be programmed into an EPROM. THE SERVANT is NOT intended to be run from computer RAM on a regular basis! I should also mention that the copyright message you see when reviewing THE SERVANT in RAM will disappear when it is working from an EPROM.

Overview of THE SERVANT's Features

SERVANT features, main menu:

100% compatibility with all software and hardware you might throw at it. RAMDOS is fully supported where appropriate. Does not use any memory. Supports 40 or 80 column mode, fast (2 MHz) operation in 80 columns. Simple device number selection. Device numbers other than 8 (9 through 12) can be accessed by holding down SHIFT, C=, CONTROL or ALT along with the command key.

- Your computer won't try to boot a disk when powered up or reset. If you DO want to boot a disk, just smack the "0" key which is big and easily accessible on the numeric keypad.
- Run the first program on a disk as a BASIC program.
- Load the first program, switch to 64-mode, and run it. Can handle files of up to 249 blocks.
- Run C64 programs as if you entered 'LOAD "*",x,1' in 64-mode where 'x' is the device number.
- Directory. RUN, DLOAD, BLOAD or BOOT a 128-mode program, or run a 64-mode program by pointing at the desired file. View the contents of SEQ, PRG, USR, REL and even DEL files. Display as ASCII or screen codes. Quick and convenient 1581 partition selection. Scratch selected file.
- Recover a NEW-ed BASIC program. Even works after a reset when a graphics screen was involved.
- Go to 64-mode. You can go back to 128-mode and then re-enter 64-mode with ML programs, BASIC programs and variables as you left them. Even utilities will still be working!
- Enhanced DOS commands. Validate will now protect the boot area (if boot sector exists), even if it consists of several sectors. Device number change/swap command. Recall last command.
- Swap 40/80 column screens.
- Convert memory to BASIC DATA statements. Create stand-alone program or merge DATA statements into any BASIC program. Select initial line number, increment, line length and data type (2 types of decimal and 3 types of hexadecimal). Can read all banks and memory ranges.
- Integrated file manager for the QUICK BROWN BOX. Unlike the QBB's own file manager it won't interfere with JiffyDOS, programs, utilities or anything else. Fast (2 MHz) and convenient one-key loading from the box. Download files from disk by using the directory. You can freely mix 64 & 128-mode programs within the same box.

SERVANT features, disk tools:

Can use all device numbers 4 through 30. Automatically detects the hardware you are using, and adjusts accordingly. Uses burst mode whenever possible. Recognizes RAMDOS as any other drive. Full 1581 partition support. The copiers automatically detect and take advantage of 64K VDC memory. 1700, 1764 and 1750 memory expansion units supported as an option. Will automatically detect the size of the REU (up to 1Mb supported) when selected.

- Directory. View normal directory on source or target disk, or view an extended directory* which displays deleted files as well.

- Disk report. Statistical overview of the number of files of each type, the blocks they use, boot blocks and blocks free on disk.
- Disk copier*. Copy whole disk or only the tracks which are used (marked as used in the BAM). Single or dual drive copy. Automatically detects if the disk is single or double sided. With 64K VDC memory (but without a REU), it will copy a whole single sided disk in one pass.
- File copier. Single or dual drive copy. Adjustable sector interleave on target disk for maximum reload speeds. No limits on the use of 1581 subdirectories. It can copy files freely between subdirectories, even from one subdirectory to another on the same disk. Automatically detects the space remaining on the target disk, and reports if the space is inadequate. Options to make multiple copies of files, scratch copied files from the source disk, and automatically replace of duplicate files on the target disk.
- Scratch files. Bulks scratch the files you select.
- Unscratch files*. Recover scratched files. Will report, as far as possible, which files are recoverable or not. Can recover all file types, even CBM files.
- Header/format disk. On the 1571 you can select single or double sided format, and you can convert a single sided disk to double sided. Change disk name & ID. Create partitions and subdirectories on the 1581. Graphic representation of the disk which displays the free space usable as subdirectory areas.
- Edit directory*. Re-arrange, sort, rename, and lock/unlock files. Change program load address, change file type & compress directory.
- Print directory*. Output directory to printer including all normally hidden information. This includes start track/sector, REL file side sector and record length. Optionally, print start and end addresses for PRG files and SEQ/USR file lengths.
- Auto boot tool. Create boot sector on a disk for a variety of purposes. Write a BASIC command line to be executed upon booting. Limited only by BASIC and size of the boot sector; all direct mode commands can be used. Make boot sector to run 64-mode programs, even programs requiring a SYS command or LOAD "*",8,1 to execute. Analyze boot sector. Kill boot sector, transfer boot sector from one disk to another. Convert boot sector to an executable program or vice versa. Boot sectors will support all device numbers.

* The marked features won't work with RAMDOS, the old CBM high capacity drives such as the 8250, 4040 and 8250 or hard drives. Whether it will work with hard drives using "emulation partitions" have not been tested. 1581, 1571 and all drives using 1541 compatible disks will work perfectly if they can be hooked up to the 128.

If you prefer, the Servant ROM can be put into a cartridge for the expansion port, or even inside the 17xx series REU's.

Before we begin

Manuals are generally boring reading. Apart from the installation instructions (appendix A), there's really no need to read these docs at once if you're a fairly confident 128 user. Rather, I suggest you return to the docs when you're stuck or want a thorough description

of a specific feature. THE SERVANT is designed with focus on being user-friendly, so just play around and have fun!

Hardware requirements

Minimum hardware:

- Commodore 128 or 128D
- Any Commodore compatible disk drive.
- 40 or 80 column (or both) monochrome or color monitor.

Hardware supported but not required:

- Additional disk drive(s) (devices 4 through 30 supported except on main menu which supports devices 8-12). Full 1581 support.
- Hard disks will work with some features.
- Quick Brown Box - utilizing THE SERVANT's built-in file manager.
- 1700, 1764 and 1750 memory expansion units. Upgraded REU's of up to 8Mb are supported. RAMDOS is supported where appropriate.
- 64K VDC RAM is automatically detected and utilized.

Special note to owners of the 1764 RAM expansion unit: Despite what your 1764 manual tells you, the device will work perfectly with your 128 and THE SERVANT can utilize it to maximum advantage.

Installation: please refer to appendix A. There you will find separate instructions for each type of the 128.

This manual is written on the assumption that you have installed THE SERVANT internally in your computer. THE SERVANT will work without modification however, if mounted into a cartridge or Ram Expansion Unit. This would, bearing in mind that the servant then would use the BANK 13 configuration rather than BANK 12, not cause THE SERVANT to act differently than described in this manual. Please note however, that the Quick Brown Box features will not work if THE SERVANT is mounted externally. This is because the QBB and THE SERVANT chip then will use the same address space, cancelling each other out.

SPECIAL NOTE to 128D users: Whenever I refer to the 1571 drive, the same will always apply to your computer's internal disk drive.

IMPORTANT NOTE:

You'll probably spend your first hours using THE SERVANT in computer RAM. THE SERVANT is somewhat hampered when working in that configuration. There will be less buffer space available for the Disk Copier and File Copier. Furthermore, some programs won't launch properly when using THE SERVANT in RAM. Especially, using RAMDOS will produce severe problems in some cases. Also, you can't load programs of more than 100 blocks in size, as bigger programs will overwrite THE SERVANT and crash it. You may also discover other quirks when using THE SERVANT from computer RAM. However, all these problems will disappear when THE SERVANT is allowed to live on a chip. THE SERVANT working as a RAM-resident program is only for your review, customizing and

then for making a copy that can be programmed into an EPROM. THE SERVANT is NOT intended to be run from computer RAM on a regular basis!

The Main Menu

The main menu is what you in the future are going to see when you power up or reset your 128. Upon power up, THE SERVANT will change the function key definitions and set the screen colors. If you don't like the new F-keys or colors, please refer to appendix B on how you can have them changed if you so desire. Changing the F-keys is the key to THE SERVANT's high level of compatibility. Except for colors & F-keys, THE SERVANT doesn't change or interfere with your computer.

You will notice that your drive won't go click-whizz-click-click anymore, knocking the heads all over the place, trying to auto boot a disk. Furthermore, the computer won't reboot when leaving an application. This feature is included to save wear and tear on your drive as well as your nerves.

THE SERVANT supports both 40 and 80 column mode. In 80 column mode, the computer will always operate in fast (2 MHz) mode, and will still be in fast mode whenever you exit THE SERVANT. To insure compatibility, all 64-mode features, "boot disk" and "Call Exrom" will put the computer into slow (1 MHz) mode regardless of screen mode.

No functions on the main menu interfere with the computer's memory; using the features here won't interfere with any programs, variables or other things already in memory. The exceptions from this rule are obvious: If you load a BASIC program for example, it will of course replace any BASIC program which happened to be in the memory already.

Bypassing the menu:

There are only two ways to bypass THE SERVANT main menu when you turn on or reset your computer. You can hold down the STOP key which will drop you into the 128's ML monitor. Alternatively, you can hold down the C= key on the bottom left of your keyboard, which will immediately make the computer go into C64-mode. As you can see, both work exactly as if THE SERVANT was not present.

If you mount a 128-mode ROM cartridge into the Expansion port, it will probably, depending on its design, bypass THE SERVANT. If THE SERVANT itself is mounted externally, it will bypass any internal ROM you might have. However, using the "Call Exrom" command on the main menu will enable you to access the internal ROM.

IMPORTANT! Escape keys: THE SERVANT respond to the ESC key as an all-purpose panic button. If you have chosen a function, but later find that it was not such a good idea, press the ESC key. The ESC key will let you exit from everything everywhere! The only exception is when a disk operation is in progress (reading or writing to disk). Then, the STOP key will in most cases be able to abort. Please note that the RUN/STOP key may have to be held down firmly for several seconds before THE SERVANT responds.

IMPORTANT! The shift keys: When using all disk-related selections on the main menu, the state of the shift keys is significant. When none of the shift keys are held down, THE SERVANT will access device 8. If you hold down SHIFT, device 9 will be accessed, device 10 can be accessed by C=, device 11 by CONTROL, and finally device 12 by ALT. If you

try to access a device which is not present, you will hear a "ping", and the menu will reappear. If you're getting pings when you think you shouldn't, check that the SHIFT LOCK is up, that you're using the proper shift key, and that the drive you try to access is properly connected and turned on. You will also get pings if the disk is missing, the drive door isn't closed, or if other general disk errors occurs.

BOOT DISK (0)

As you have noticed, after installing THE SERVANT your computer won't try to auto boot. But if you do want to boot a disk, just hit the "0" key which is big and easily accessible on the numeric keypad. If you are using a 1581, the root directory is selected, because the drive can't access the boot sector if a subdirectory is selected. Please note that the shift keys will determine which drive will be used. Unfortunately I have found very few pieces of software that will boot properly from a device other than 8. Many software manuals instruct you to simply turn on your system to make the software run. Some instruct you to press the reset button. With THE SERVANT installed, there will be an extra step: Pressing the zero key. Boot Disk doesn't work with RAMDOS. Also, you can't use this feature with a drive which has its device number changed through software (using the command "u0>"+chr$(n) for example), because the 128's boot routine will reset the drive and thereby also restore its hardware-set device number.

RUN DISK (1)

Will run the first file on a disk as a BASIC program, similar to using RUN "0:*" in BASIC.

RUN 64 (2)

This is an interesting feature. It will load the first program off a disk, switch to 64-mode, and run it as a BASIC program. Thereby, you can use the speed of your 1571 or 1581 to load the program before actually switching to 64-mode. This is especially advantageous with bigger programs. In fact, this feature can perfectly load programs of up to 63231 bytes in size, the equivalent of 249 disk blocks. This is a great boon if you own one of those C64 freeze cartridges. If you're using a 1571 disk drive, it will remain in 1571 mode, so you can easily enjoy its full capacity in 64-mode as well.

SPECIAL NOTE to JiffyDOS users: When JiffyDOS is enabled, leaving your 1571 drive in 1571 mode while the computer is working in 64-mode will make it as slow as a stock drive. Fortunately, a JiffyDOS equipped drive can still use both sides of the disk, even when it is switched to 1541 mode. It is therefore feasible to switch your 1571 to 1541 mode after the computer has switched to 64-mode. This can easily be accomplished by resetting the drive (s). If you have a 128D, you will find a small reset button on the side of your computer. Please refer to the computer's manual to find the exact location. If you have an old flat 128, you may want to buy or make a one yourself. JiffyDOS users might prefer the next menu selection (Load "*",64) since your disk drive will be equally fast in both 128 and 64-mode. It will in most cases eliminate the need for a disk reset button.

LOAD "*",64 (3)

This is somewhat similar to RUN 64. This will switch to 64-mode and automatically enter the command 'load "*",X,1', then 'RUN', where X is the device number selected by the shift keys. This feature is practical when you want to boot commercial C64 software (games for example). If the drive you're using is a 1571, it will be switched to 1541 mode to aid compatibility. Please note that you can specify devices 8-12 by the shift keys, but most software that loads in more than one part won't allow other devices than 8. Due to a limitation in the 64's operating system, this feature can't handle files bigger than 201 blocks. Don't work with RAMDOS.

DIRECTORY (4)

This will read the directory off a disk and display it on the screen. The display will halt when a full page has been displayed. To halt the listing before the end of the page has been reached, press the RUN/STOP or NO SCROLL key. If you prefer the directory to scroll continuously, hold down the LINE FEED key. To halt the listing, release the LINE FEED key. You can continue by pressing the space bar, or exit by pressing the ESC key. Pressing the HOME key at any point will re-display the directory. When the directory is halted, a flashing pointer will appear. Using the cursor keys you can move the pointer up and down to select a file. Using the 80 column screen will give you the advantage of being able to see the various command keys. You have the following options:

KEY	FUNCTION
RETURN	Will load the file into memory and run it as a BASIC program, similar to RUN "filename"
F1	Will do the equivalent of DLOAD "filename"
F3	Will do the equivalent of BLOAD "filename",b0 (will use bank 0). Unlike the others, this command will return to THE SERVANT's main menu after the file has been loaded.
F5	Will do the equivalent of BOOT "filename", using bank 0 while loading. Please note however, if the start address of the program is lower than 16384 ($4000) the computer will be configured to bank 15 when the program is executed. PLEASE NOTE: Using BOOT "filename" in BASIC will in some cases get you into trouble. It won't work with RAMDOS, and some older 128's may malfunction when using a slow serial disk drive (1541 for example). You won't encounter these problems with THE SERVANT.
F7	Will load the program, switch to 64-mode, and run it. 1571 disk drives will NOT be switched to 1541 mode. You can load programs of up to 249 blocks in size.
S	Will scratch the selected file, but you will be asked to confirm before the action is performed.
T	Will "type" the file as ASCII, meaning that it will display the file contents, whatever it is, on the screen. Control codes will be displayed as their reversed counterparts; no file will "upset" the screen. SEQ, PRG, USR, REL and even DEL files can be displayed with equal ease. Unfortunately, most GEOS files have a structure which can't be displayed. Please note that you won't be limited to text files. You may equally well list a program, allowing

	you to read text which is embedded in the code. This may sometimes reveal things about the program which would otherwise be difficult to disclose. Press space to halt, space again to continue, HOME to re-type the file from the beginning, ESC to exit.
SHIFT/T	Will display the file as SCREEN CODES. Normal and reversed characters will be displayed as such. This feature is especially advantageous with word processor files which are often stored in screen code format. Such word processors normally use the back arrow to serve as carriage return. The screen code lister will therefore begin a new line whenever a back arrow code is encountered. If you find that you should have selected another "type" mode, simply press T again along with the appropriate shift key (if any) and the file will be re-displayed using that mode.
RETURN	Can also be used to select 1581 subdirectories. When the cursor points at a "CBM" file, the RETURN key will select that subdirectory.
/	Will select the root directory if you're using a 1581 drive.

DISK TOOLS (5)

This will enter the disk tools menu. The state of the shift keys will determine which drive the disk tools will adopt as the source drive. If no disk drive is present, you will be "pinged", and won't be allowed to enter the disk tools section. For a complete explanation, refer to the DISK TOOLS section of these docs.

MONITOR (6)

This will simply enter the 128's ML monitor. To get back to THE SERVANT you may enter 'G C8000' for a full restart (will initialize THE SERVANT's colors, and function key definitions), or 'G C8009' if you prefer to retain the colors and function keys currently in use. (If THE SERVANT is mounted externally, the respective commands are 'G D8000' and 'G D8009'.)

DATAMAKER (7)

The purpose of this utility is to read the contents of a specified memory range and turn it into DATA statements. You can freely merge DATA statements into any BASIC program. Or you can create a stand-alone program containing the DATA statements only. All parameters are specified in decimal, but you can use hexadecimal numbers by preceding the number with a dollar sign (Even when specifying line numbers!). The procedure is as follows:

Parameter	Function
BANK	The bank number (0-15) from which to read the desired data.
Start address	The first memory address of the data.
End address	The last memory address of the data.
First line	The line number of the first line.
Increment	Line number increment.
Line length	The number of data elements per line. If you specify a greater number of data elements per line than would actually fit, the data maker will

	always pack as many data elements as possible on each line and place the rest on the next line (which in turn will be as long as possible).
Type (1-5)	This will determine how the numbers in the data statements will be represented. There are five types of data statements available for maximum flexibility: 1 = decimal with gaps (251, 74, 86, etc.), 2 = decimal without gaps (251,74,86,), 3 = hex with gaps (fb, 4a, 56,), 4 = hex without gaps (fb,4a,56,), 5 = hex without gaps, nor commas (fb4a56). The latter is possible because hex values are always two characters long, so they can be separated by using MID$. Please use this function and see the BASIC program it creates. This variant is the most compact, but at the same time the slowest. Type 2 is the fastest to run. Please note that types 3, 4 and 5 contain hexadecimal numbers, and is thereby not usable if the resulting program is to be run in 64-mode.

A little tip: Sometimes it would be practical to know the actual address each data element was taken from. This can be accomplished by using line numbers in a cunning way:

- Decide how many data elements you wish to put on each line.
- Multiply the number by two, and subtract the resulting figure from the start address.
- Specify the resulting figure as initial line number.
- Then use the same value for line number increment as the number of data elements you decided to put on each line.

Now, each line number will show the actual address of the first data element on that line.

UN-NEW (8)

This will bring back a BASIC program after a NEW command has been issued or even if the computer has been reset. Of course, it's important that the BASIC memory hasn't been used by another BASIC program or otherwise been overwritten. A prompt reading "Graphics screen? (Y/N)" will appear, and normally you should answer NO.

However, if you reset the computer while a graphics screen is allocated (using GRAPHIC 1, 2, 3 or 4) there will be a problem. Normally, a BASIC program starts at memory location 7169 ($1c01), but when a graphics screen is opened, the program is moved to start at 16385 ($4001). When the computer is reset, the program stays where it is while the pointer which tells the where the program starts is moved back. To cover that eventuality you will be asked if you want to re-allocate the graphics screen before attempting to recover the program. If in doubt you should try recovering your program without allocating the graphics screen and see if it lists properly. If the program won't list, can be listed only in part or turns to garbage, then you can try to specify graphics screen (answer YES).

GO 64 (9)

As its name implies, this will switch the computer to 64-mode, similar to GO64 in BASIC. You will be asked to confirm before the action is actually performed. However, THE SERVANT uses a slightly different approach when switching the computer to 64-mode. If

you're not technical minded, you can just skip the rest of this explanation. You will probably not be able to detect the difference.

If you're a programmer, this feature may open some interesting possibilities. As you know, the 128 has two distinct segments of RAM memory of 64K each called BANKs. BASIC uses one bank (bank 0) to store system variables, vectors, programs and high resolution screens among other things, and the other (bank 1) to store BASIC variables. Normally, when you enter GO64 in BASIC or reset the computer while holding down the C= key, the computer will set up 64-mode to work in bank 0, and bank 1 will be inaccessible entirely as long as you stay in 64-mode. THE SERVANT however, sets up 64-mode in bank 1. This is also true with all other features in THE SERVANT that switches the computer to 64-mode.

So what? Well, using bank 1 does have a quite significant advantage: You can run 64-mode ML or BASIC programs as usual. Then, you can reset the computer, and examine the entire memory from $0400 to $feff with the monitor without corruption. Screen memory, RAM under ROM & I/O will be intact. Then you can use GO 64 from the menu to return to 64-mode, again without corruption (screen memory will be corrupted of course, because the screen is cleared in the process).

I have even more in my sleeve. Hold down SHIFT when you select GO 64, and a BASIC program won't even be NEW-ed, even variables will be intact! Even utilities will still be working! Yep, see for yourself. The only exceptions are some utilities which depend on pre-set conditions in the I/O area (sprites for example). You can run programs in 128-mode while preserving 64-mode memory, provided they don't corrupt bank 1 memory. Running 128 BASIC programs is also possible if you change the variable area. POKE48,160:CLR will protect the 64-mode BASIC area, leaving 24K for variables in 128-mode. If you wish to protect the 49152 ($C000) area as well, use POKE 48,224:CLR, leaving nearly 8K of variable space. The secret behind this spectacular feature is that the lowest 1K of the 64 memory which contains all important pointers and vectors is safe under the 1K common area which is completely inaccessible in 128-mode without the use of some ML trickery. Please note that the computer will crash (if using SHIFT/9) unless you have already been in 64-mode at least once since the computer was powered up.

To be able to exploit this feature you must be careful on how you enter 64-mode: You should always enter 64-mode by using one of THE SERVANT's features. THE SERVANT will always drop you into bank 1. On the other hand, using GO64 from BASIC, resetting the computer while holding down the C= key or boot software which auto boots into 64-mode (as many games do), the computer will use bank 0 as per normal practice.

RESET PREFS (+)

This will set THE SERVANT's default function key definitions and colors.

Pressing SHIFT/+ will reset all colors to the C128 default values. I have seen some programs that set the 40 column background and border colors to black, and not setting the text color, assuming it is set to the default light green when the program is run. THE SERVANT's default 40 column text color is black, so you won't be able to see anything. To circumvent this sloppy programming, the SHIFT/+ feature has been added. A more

satisfactory solution, if the troublesome program (or its loader) is written in a BASIC, is to insert the statement COLOR 5,14 at the beginning of the program.

SWAP 40/80 (-)

If you need to switch to the alternate screen mode, this feature will enable you to do so. The current screen will go blank, and the menu will be set up in the other screen mode. Now, throw your monitor's 40/80 columns selector.

CALL EXROM (.)

A rather obscure feature this one, but it will let you enable an external ROM cartridge of your own design from the SERVANT menu. Due to the 128's design, most commercial produced cartridges will suppress THE SERVANT entirely. To use this feature, you'll probably have to design your own cartridge. The QUICK BROWN BOX is ideal for this purpose. It contains battery backed RAM, but acts much like a ROM cartridge because it retains its contents without power applied. You'll have to know ML programming to do it in any case but a ML course goes beyond the scope of this manual. If you have the QBB, your manual will describe briefly how to make a cartridge work. Also, function ROM cartridges are briefly described in the 128 Programmers Reference Manual.

Call Exrom will search for the presence of a function ROM in all banks, cycling from 0 to 15. However, it won't check the bank configurations where THE SERVANT itself is currently visible, so it won't inadvertently call itself. There is a quirk here: If THE SERVANT is working in computer RAM, it cannot access cartridge software in other RAM banks. Checking all banks could make it easier for you to develop cartridge software. Use RAM bank 0 or 1 during development and then modify it to work in either external or internal ROM later on (or make it adjust automatically according to which bank it found itself in).

PLEASE NOTE: If THE SERVANT is mounted externally, this command will access the internal ROM if present. Mounted externally THE SERVANT will suppress any internal ROM during startup. Also please note that the computer always will be set to 1 MHz mode when accessing the other ROM. This will ensure compatibility with the QBB which can't operate in 2 MHz mode.

A tip: If you set the ROM I.D. (byte 6, either $8006 or $c006) in the QBB or ROM to zero, it will be dormant until it is called from THE SERVANT main menu. That byte would normally hold a value of 2 or greater. You could use this trick with THE SERVANT too if desired.

DOS-COMMAND (@)

This will let you send a CBM DOS command over the disk drive command channel. You will see a '@' character at the bottom of the screen along with the device number you will be using. The useful commands are as follows:

Command	Function

C0:newfile=oldfile	Copy a file on the same disk.
I0	Initialize disk.
N0:diskname,ID	Format (header, new) disk, use with caution.
R0:newfile=oldfile	Rename a file.
S0:filename	Scratch (delete) a file.
UJ	Reset disk drive.
U0>M0	Force drive into 1541 mode (1571 only)
U0>M1	Put drive into 1571 mode (1571 only)
U0>H0	Select head 0 (1571 in 1541 mode only)
U0>H1	Select head 1 (1571 in 1541 mode only)
U0>V1	Write verify OFF (1571 with new ROM & 1581)
U0>V0	Write verify ON (1571 with new ROM & 1581)
V0	Validate disk. PLEASE NOTE: This command has been enhanced. Unlike the standard validate command, it will protect boot sector(s), if present, from being overwritten later. Please note that the boot-protect won't work with drive 1 in a dual drive unit.
/0:partition	Change disk partition (1581 only).
/	Change to root directory (1581 only).

Please refer to your disk drive manual for a more thorough description of the above commands.

Extra DOS commands

#number	Change/swap device number. Typing '#' followed by a number between 4 and 29 will change the device address of the current disk drive to that number. If the device number specified is used by another disk drive, the two drives will swap device numbers. DO NOT use this command if there's currently a disk drive active on the serial bus with device number of 30. Device number 30 is used temporarily as the device numbers are swapped. Also, you can NOT swap devices with printers or other non-disk units. It will work fine with RAMDOS though.

After a DOS command has been performed, the drive status will be displayed. You can press RETURN without entering a command if you only wish to view the drive status. Please refer to your disk drive manual for interpretation of the status messages.

You may recall the last DOS command by pressing SHIFT/RETURN while the cursor is flashing on the DOS command line. Please note that this feature is only effective as long as you stay within THE SERVANT. If you exit, and then re-enter THE SERVANT, any previous commands is erased.

FORMAT QBB (^)

THE SERVANT has a proprietary system for storing programs into a device called the QUICK BROWN BOX. To do this you must format your QBB. There is a separate section in these docs which deals with this subject.

BACK TO BASIC (RETURN key)

Of course, since THE SERVANT always pops up whenever you power up or reset your trusty ol' 128, there should be a way to return to the familiar "READY" mode. Pressing the RETURN key will allow you to do so.

Please note that the state of the shift keys may be significant even at this point. If you're using a DOS utility, DOS wedge or such like, the shift keys will in some cases determine the device number your utility will access. Please note, JiffyDOS won't obey the shift keys, having its own special device number change command. To get back to THE SERVANT, hold down SHIFT, and press the RUN/STOP key.

SPECIAL NOTE about the SHIFT-RUN/STOP combination: Whenever you enter the servant, this key will always be set to 'BANK12:SYS32777'. You can't really change the definition of that key without having it reset every time you enter THE SERVANT. You may change this permanently by altering THE SERVANT itself. Please refer to Appendix B.

You can have the function key definitions and screen colors (and some other things) changed to suit your own preferences. Please refer to Appendix B for further details.

It may very well happen that THE SERVANTs function keys might be disabled or messed up for some reason or another. If the SHIFT-RUN/STOP combination (or another f-key of your choice) won't let you enter THE SERVANT, you can enter 'BANK12:SYS32777', and THE SERVANT will pop up. Also, you may want to press the '+' key once you're on THE SERVANT menu to re-enable the function keys. You may find it easier just pressing the reset button however.

USING THE QUICK BROWN BOX TOGETHER WITH THE SERVANT

The QBB's main objective is to store programs of your own choice. The programs can then be reloaded just as if they were loaded from disk. The advantage is that they can be reloaded in a flash, with just a couple of keystrokes. Furthermore, the programs are immediately accessible whenever you turn on your computer. Using a lithium battery, the QBB will retain its contents even when the computer is turned off.

The Quick Brown Box original file manager causes some compatibility problems with some applications however. A different kind of file manager was included in THE SERVANT in order to solve those problems. You can still use the QBB's proprietary file manager if you wish, or even both at the same time. THE SERVANTs file manager also has the ability to live peacefully together with ordinary cartridge software inside the QBB.

Setting up the THE SERVANT file manager

Before you can use your QBB to store programs, you must make it ready to do so. First, the QBB's 64/128 switch must be set to 128. Then, select "Format QBB" from THE SERVANT main menu using the up-arrow.

You will then be asked how many banks you would like to reserve for THE SERVANTs file manager. You should NOT reserve more banks than your QBB contains though. The 16K QBB has one bank, the 32K has two, and the 64K model has four banks. If you have the

128K or 256K models, THE SERVANT can still use only 64K (four banks) of your QBB's space. In that case, you may wish to use the QBB and THE SERVANT file managers both at the same time. Now, select the number of banks that suits you.

You can change the number of banks reserved for THE SERVANTs file manager at a later time if you wish, without disturbing the contents of your QBB. If you have formatted the QBB (using THE SERVANT) already, and the QBB contains programs, you will be asked if you would like to remove the programs from the QBB at the same time. If you answer NO, THE SERVANT checks to see if the programs currently in the QBB will fit into the number of banks you specify. Specifying a lower number of banks than needed to hold the current programs won't be allowed. If you answer YES to removal of the programs, all programs will be wiped out, and you are free to specify the number of banks you wish (within the limits of your QBB). This is the fastest way to remove all QBB files in one go if you wish to use the QBB for other programs.

You may also set the number of banks to zero. This will effectively turn off the file manager and your QBB is free to be used for other purposes.

Using THE SERVANT's QBB file manager

When you have formatted your QBB using THE SERVANT, you will see the number of bytes free for storage along with a menu at the bottom of the screen. You can now store up to 26 programs into the QBB. Their names will appear below THE SERVANT main menu, and can be reloaded by the letter keys A - Z. The menu entries are:

Key	Function
F1, Store BASIC	Will store the BASIC program currently in the 128's memory into the QBB. You will be asked to give it a name of up to 16 characters. Press RETURN, and your program will appear in the directory below THE SERVANTs main menu. It will then run automatically when you press the letter key next to the program name. If the program is too large to fit into the remaining QBB space, the message "QBB is full" will appear. The process will be aborted.
F3, Copy from disk	This command resembles the main menu directory command. The directory will be read from the disk drive indicated by the shift keys. When the directory is halted, a flashing pointer will appear. The command keys will now determine how the selected programs are to be reloaded from the QBB. You have the following options: **RETURN:** Will make the program load into memory and run as a BASIC program, similar to RUN "filename" If the file has been created using "Save QBB" (see below), the current QBB contents will be wiped out and replaced with the contents of the file. **F1**: Will do the equivalent of DLOAD "filename". **F3:** Will do the equivalent of BLOAD "filename",b0 (will use bank 0). Unlike the others, this command will return to THE

	SERVANT's main menu after the file has been loaded. **F5**: Will do the equivalent of BOOT "filename", using bank 0 while loading. Please note however, if the start address of the program is lower than 16384 ($4000) the computer will be configured to bank 15 when the program is executed. **F7**: Will load the program, switch to 64-mode, and run it. Please note that, except for BASIC programs, it is not possible to store a program into the QBB directly from memory. You must save it to a disk file first, and then download it from disk to the QBB. Also please note that upon reload from the QBB, the state of the shift keys is significant if the program in question is sensitive to which disk drive it was loaded from. If the file you try to download is too large to fit into the remaining QBB space, the message "QBB is full" will appear. The process will be aborted.
F5, Save QBB	You may save the whole QBB contents (the parts currently allocated by THE SERVANT file manager) to disk as a single file. The QBB contents can then be reloaded using the "copy from disk" command (F3, see above).
F7, Scratch	To remove a program for the QBB, press F7 and then the letter next to the program you wish to delete. Please note that if you delete a file near the beginning of the directory, the deletion may take some time. Please be patient.

SPECIAL NOTE about saving/deleting in the QBB: Due to the C128 electronics design, writing to the QBB will corrupt system RAM memory in the same address range. THE SERVANT minimizes this problem by selecting RAM bank 1 during reads/writes to the QBB. Expect to have bank 1 memory from 32768 ($8000) to 49151 ($bfff) corrupted when storing or deleting programs in the QBB.

THE SERVANTs banking scheme using the Quick Brown Box

Unlike the QBB native file manager THE SERVANT will always start using the highest numbered bank in the QBB and work downwards. This goes for the whole range of QBB models, 16K to 256K. Reserving two banks of a 64K QBB for THE SERVANT, will reserve banks 2 & 3 leaving banks 0 & 1 free for other uses.

Reserving all but one bank in your QBB will let you use bank 0 for cartridge software while using the rest of the box for program storage.

Using QBB and THE SERVANT file managers at the same time

The reversed banking scheme of THE SERVANT makes it possible to use both managers at the same time. The native QBB file managers can reserve one or more banks at the end of the QBB for other uses. Then, let THE SERVANT use those banks for program storage. Or indeed, you may elect not to use THE SERVANT file manager at all. You could even use the 64-mode QBB manager either alone or together with THE SERVANT file manager in 128-mode.

There is one problem with this approach however if you want to use the 128-mode QBB manager. Using the native QBB file manager will disable the servant. To use both, there must be a way to set the SHIFT-RUN/STOP combination to 'BANK12:SYS32777'. That's all that's needed to use THE SERVANT. There are several ways this can be accomplished. Below you will find a description of the method preferred by the author. The method won't corrupt any memory, and you can load your favorite function keys automatically upon power-up. Here's how:

1) Save the QBB contents to disk as described in the beginning of your QBB manual.
2) Run the "128 manager" program that came with your QBB.
3) If you wish to start with a blank box, choose to "Initialize the QBB for loading". Refer to your QBB manual for a description of this process.
4) Exit the file manager and define your favorite function key definitions using the KEY command.
5) Type 'BANK12:SYS32777' to enter THE SERVANT, and then press RETURN to get back to BASIC. The SHIFT-RUN/STOP combination will now have been set to 'BANK12:SYS32777' by THE SERVANT.
6) Type 'BSAVE "f-keys",p4096top4351', then press RETURN. This will save your function key definitions to disk.
7) Run the 128 manager again. Select "load new program". Enter the name of the file you used for your function key definitions; "f-keys". Follow the prompts, enter ID keys.
8) Enter a 6 letter name, using a < symbol as the first letter. The < character will cause the QBB not to execute the file; the f-keys are not an executable file. Leave the RAM bank and SYS banks as they are.
9) Now select "auto-start program" from the manager menu. Enter the two letter identifier you just specified
10) Finished. When you reset your computer, your F-keys along with THE SERVANT start key combo will be activated. Press SHIFT-RUN/STOP to enter the servant.

Getting rid of the QBB auto start

You may wish to remove the QBB control over your computer entirely, leaving all control to THE SERVANT. Type the following line in BASIC:

slow:bank13:poke56832,16:poke32774,0

Hit RETURN, and then press the reset button.

THE SERVANT will now be present at power-up, but if you wish, the QBB can be invoked by pressing the point (.) key, "Call Exrom".

To re-enable the QBB auto start

```
slow:bank13:poke56832,16:poke32774,255
```

Hit RETURN, and then press the reset button.

Booting GEOS 128 from the QBB

The Maverick V5 includes a feature which converts the GEOS core system into a single bootable file. Make your GEOS boot file following the instructions supplied with Maverick.

PLEASE NOTE: The Maverick GEOS tool will at one point instruct you to press the reset button. With THE SERVANT installed you'll have to do another step: Press the "0" (null) key.

Load the newly created GEOS boot file into the QBB using THE SERVANT's F3 command. Press F5 to make the file boot, and specify bank 0. Please note that due to size of the GEOS boot file, the QBB must be set up with at least 3 banks, meaning that your QBB must be of at least 64K in size.

Upon reload from the QBB, You can make GEOS access drive 9 (for loading of the Desktop and Configure) by holding down SHIFT. It should be noted that GEOS won't work properly when booted from other drives than 8 or 9. Furthermore, the GEOS boot file can't use a 1581 if it was created for another drive type or vice versa. 1541 versus 1571 won't matter though.

The Disk Tools

The Disk Tools are a collection of utilities for all those necessary things which the C128 operating system can't do. I should note that these utilities don't attempt to speed up your disk drives except for burst mode which is used wherever possible. You may find THE SERVANT to be slow. Security and compatibility is given priority over speed. On the basis of the above, I would like to give JiffyDOS my heartfelt recommendations. It will greatly enhance your system's performance, both within and outside THE SERVANT. If you want more speed you could try Maverick (recommended in any case, since it can copy nearly all copy protected software) or FasTrac128. For suppliers, please refer to Appendix F.

PLEASE NOTE: Unlike the main menu selections, nearly all Disk Tools selections use the 128's memory. Most routines overwrite bank 1 memory (where BASIC variables are normally stored). The Disk Copier and File Copier however, exploit all memory resources; bank0, bank1 and the proprietary memory used by the 80 column chip.

The Disk Tools menu

When the Disk Tools menu is entered, THE SERVANT will analyze the disk drives connected to your computer. THE SERVANT will check all device numbers 8 through 12 for active disk drives. When it comes to device 12, it wraps around and starts at device 8 again. It will adopt the first disk drive found as the main drive. Then, the next drive found becomes the copy drive. If only one drive is connected, that drive will become both the main and copy drive. When the Disk Tools section is entered from THE SERVANT main menu, the state of the shift keys determines which device number THE SERVANT will analyze first.

The main drive will be the source drive for the disk and File Copiers, and all other operations will be performed on this drive. The copy drive will be the target drive for the copiers. The exception is the "interleave set" command which will always affect the copy drive.

Drive type check

THE SERVANT will automatically figure out the type of drive connected. The supported drive types are as follows:

Type	Details
Not active	THE SERVANT can't find a disk drive with the device number specified. In that case, check that your drive is properly connected and turned on. Also check that you have indeed selected the right device number.
Unknown	THE SERVANT has detected a drive which doesn't match any of the Commodore disk drives. THE SERVANT will assume the drive to be 1541 compatible. That is, single sided, 35 tracks, directory on track

	18. If your drive doesn't fit this description, only the scratch, File Copy and Auto boot Tool utilities will work with your drive.
1541	Commodore 1541 drive or compatible. The Commodore 1570 will be identified as a 1541, but THE SERVANT will use burst mode when communicating with the 1570.
1571	Commodore 1571 or compatible in double sided mode.
1571 side 0	Commodore 1571 set to 1541 mode. For maximum performance, you should switch the drive to 1571 mode.
1571 side 1	Commodore 1571 in 1541 mode using the back side of the disk. THE SERVANT will fully support this mode of operation.
1581	Commodore 1581 disk drive. THE SERVANT will identify if your drive will access the root directory or a partition. If a partition is selected, the start track of the partition will be displayed as well.
17xx RAMDOS	Commodore RAMDOS. THE SERVANT can use RAMDOS as any other disk drive within the capabilities of RAMDOS itself.

If the copy drive is identified as a 1541 or 1571 (in all modes) then the drive's sector interleave setting will be displayed. Please refer to the Interleave set command for further details. If your disk drive is equipped with JiffyDOS, the interleave will be zero unless it has been set to another value previously. Please refer to your JiffyDOS manual.

THE SERVANT will always check if the drive(s) you are using is compatible with the utility you select, and adjusts itself automatically to your setup. Don't worry about compatibility, THE SERVANT will handle it for you (unless your drive is identified as "unknown", see above).

SPECIAL NOTE to 1581 users: All operations will be performed within the currently selected partition unless indicated otherwise.

IMPORTANT! Escape keys: THE SERVANT respond to the ESC key as an all-purpose panic button. If you have chosen a function, but later find that it was not such a good idea, press the ESC key. The ESC key will let you exit from everything everywhere! The only exception is when a disk operation is in progress (reading or writing to disk). Then, the STOP key will in most cases be able to abort the process. Please note that the RUN/STOP key may have to be held down firmly for several seconds before THE SERVANT responds.

IMPORTANT! Disk errors: Most errors will be reported using the standard disk error messages. Please refer to your disk drive manual for correct interpretation of the error messages. If you see the message I/O error, some error has occurred when communicating with the drive, probably a "device not present".

Disk Tools Quick Keys

Key	Function
SPACE **Swap main/copy**	Your main drive will become the copy drive and vice versa.
F1/F2	Displays a directory from the specified drive.

Directory (main/copy)	
F3/F4 **Extended dir (main/copy)**	Displays ALL files in the directory including any deleted files. In addition, each file's start track (T) and sector (S) is displayed. This feature won't support RAMDOS. For both directory types, you can scroll up and down using the cursor keys, use + and - to move one page up and down respectively, and HOME to go to the top of the directory. To select a 1581 subdirectory, place the cursor on the subdirectory name and press RETURN. Press "/" to select the root directory. You can press F1 through F4 directly to access the other directory type and/or the other drive without returning to the Disk Tools menu.
F5/F6 **Disk report (main/copy)**	This feature will produce a statistical overview of the disk in the accessed drive. If you're using a 1581, only the currently selected directory will be taken into account. The disk name and ID will be printed first. Then a list of how many files there are of each type, along the sum of the disk blocks each file type uses. Also the total number of files and the blocks they occupy is displayed. Finally, the number of boot sectors (if any), blocks free and total blocks on disk is displayed. Please note improperly closed files (splat files) are NOT included in the file count. Also please note that DEL files in the list DOES NOT mean scratched files. Only files that appear as DEL files in a normal directory will be counted. A DEL file, if properly closed, is a completely legitimate file type which can be read as any SEQ file. Regarding boot sectors, you'll see "0" or "1" most often, but multi-sector boot areas can also be encountered. There are two cases where you'll see "(N/A)" where the number of sectors should be: This will happen if RAMDOS or a 1581 subdirectory is accessed. RAMDOS can't have boot sectors, and in the case of the 1581, the boot area of track 1 is accessible only if the root directory is selected. The "Grand total" is the sum of the blocks used by files, boot blocks and free blocks left on the disk. Normally this should add up to the total capacity of your disk. There are however, a number of reasons where this might not be the case. For example, the blocks that are counted for each file in the directory may in some cases not reflect the actual size of the file. It is possible to alter the number of blocks in the directory without changing the length of the file (or vice versa). Disk blocks may also be marked as used without being part of files (this is what Commodore disk manuals refer to as "random files"). Bear this in mind if the grand total doesn't make sense. Actually, the grand total is an excellent indicator that something might be wrong with the disk. Errors in the BAM occur more often than you

	think. Validating the disk might then be a good idea.
F7/F8 **DOS command (main/copy)**	Enter a DOS command on the specified drive. You will see a '@' character at the bottom of the screen along with the device number you will be using. Identical to the main menu DOS commands. Refer to the main menu section for further details.
DEL **Ram Expansion Unit on/off**	THE SERVANT can use any Commodore Ram Expansion Unit as data storage when using the Disk Copier and File Copier. When turned ON, THE SERVANT will analyze and display the size of your REU. Please note however, THE SERVANT won't identify any REU to be bigger than 1Mb, even if your REU does have more memory onboard. Since duplicating 1581 disks is the most memory hungry job you're ever going to do within THE SERVANT, this limitation won't hamper you. Analyzing the REU's size won't change its contents, but you should NOT turn it on if RAMDOS is active. The REU contents will be disturbed and your system will crash as soon as THE SERVANT begins to store data in the REU.

CONFIGURE DRIVES

To manually set which drive is to be the main & copy drives, press the **C** key. Enter the device number for the main drive, then the device number for the copy drive. THE SERVANT will analyze your new configuration and the drive type(s) will be displayed at the top of the screen.

INTERLEAVE SET

You may want to adjust the sector interleave on your copy drive. Press **I** and then enter the sector interleave you require. The drive in question will retain that interleave setting until you change the setting to some other value or reset the drive. You won't be allowed to set the interleave to a greater value than 16, as it may make your drive malfunction with potential loss of data as a result. In addition, the interleave should not be set to zero unless your drive is equipped with JiffyDOS. The interleave set feature will only be available if THE SERVANT identifies your drive as 1541 or 1571 (in all modes).

What is sector interleave anyway? When data is saved to disk it is organized in tracks and sectors. The tracks are like concentric rings on the disk, and each track is further divided into subsections much like a pie. These subsections are called sectors. To read a sector the disk drive must make sure the read/write head is on the right track, and then wait for the sector to come around to the head's position where it can be read.

This is where interleave comes in. If more than one sector on a track is to be read, which is most often the case, it is not desirable to have the sectors placed in succession. The drive needs a split second to process the data just read and send it to the computer through the serial bus. Then the next sector will have passed the read/write head and the drive will have to wait for nearly a whole revolution. When reading a large amount of data, a program for example, this waiting becomes quite noticeable. It is better to leave

some space between each sector in a sequence so that the next sector is just ahead of the read/write head just when the drive needs it. Since the data has to be read in the same order it was written, the interleave between the sectors will have to be established as the data is written to the disk.

The 1541 normally uses a interleave spacing of 10 (reading every tenth sector) while the 1571 uses an interleave of 6. If your drive(s) are equipped with JiffyDOS, it uses a custom interleave scheme when the interleave is set to zero. Please refer to your JiffyDOS manual for details. The 1581 don't need sector interleave since it got enough internal memory to hold a whole track of data at the same time.

But why tinker with the sector interleave? The faster the drive, the fewer sectors has to be skipped to maintain maximum reload speed. So if the data is to be read using another drive than the one used for saving, or if it is to be reloaded using some kind of fast load, adjusting the interleave will often improve loading speed. Furthermore, if the files you are going to save or copy are not programs but data for some program, the program's access speed will often be affected by the interleave setting. Experimentation is the only way to get ultimate performance but here are some settings which can act as a rule of thumb:

Reload drive	Stock	Using fast loader
1541	10	6
1571	6	4-6

This table goes for program (PRG) files. For other file types the ideal interleave can only be found by experimentation. The above values may serve as a starting point, and you can try to increase or decrease these values.

THE FILE COPIER

A file copier will copy the contents of a disk in a file by file manner. You may select which files to copy or not to copy, and files already present on the copy disk won't be disturbed. Copying files to a new disk may also straighten them out, making them reload faster. Furthermore it is possible to adjust the sector interleave (using the Interleave Set command) which may further increase the speed. The file copier will work with all disk drives that can be hooked up to the 128, including RAMDOS. The file copier can copy files without restrictions between dissimilar drive types.

The file copier can copy program (PRG) files, sequential (SEQ) files and user (USR) files. It cannot cope with relative (REL) files. Please note that the file copier cannot handle GEOS files (which are marked in the directory as USR) because of their non-standard structure.

The file copier will automatically detect if your computer is equipped with 64K VDC (80 column) RAM, and utilize it to reduce the number of passes. It will also take advantage of a Ram Expansion Unit of any size if turned on at the Disk Tools menu.

SPECIAL NOTE to 1571 users: The original 1571 disk drive and the drive inside the first 128D model (the so-called 128D portable), but not the ones inside the newer 128D, has

some quirks which may affect the performance of the file copier. Please refer to Appendix D for further details.

SPECIAL NOTE to 1581 users: The file copier will fully support subdirectories. If you're using two disk drives (regardless if one or both are 1581s) the 1581(s) will access the subdirectory (or subdirectories) currently selected. However, if the main and copy disk drive is one and the same, you must enter the directory paths individually. The cursor will appear, and you may enter the subdirectory name. To use the root directory, press RETURN at a blank line (do NOT enter "/" in order to select the root directory) Use cursor left/right, INST, DEL, CLR and HOME to edit if necessary. You can access several levels of subdirectories by separating the directory names using commas (directory1, directory2, directory3). You must enter the whole directory path from the root on (but DON'T specify the root itself, this is done automatically). If the drive can't find one of the subdirectories you have entered, you will be informed about the error and prompted to correct the directory specification. To copy files from one subdirectory to another on the same disk, leave the same disk in the drive when you are prompted for the target disk.

When the file copier is selected, you will be prompted for the source disk (or both source and target disk if you're using two drives) and an options menu will appear. The options are as follows:

Option	Meaning
Auto replace	The file copier will automatically replace any files on the copy disk with the same names as the ones being copied to it.
Multi output	Will let you make several copies of the files. After every pass you will be asked if you want to make another copy of the files. Insert a new disk and press Y for YES. If you don't want any more copies, press N for NO, and another read pass will begin if there's more files to be copied.
Delete source	This option will let you scratch the copied files from the source disk. Using this feature, the files are effectively MOVED to the copy disk. The scratch operation will be done using an extra pass after finishing the actual copying. Only the files which were successfully copied will be deleted.

Press the **space bar** to continue. The directory of the source disk will now be read into the computer's memory. If there are no files on the disk that the file copier can copy, the message "No valid files" will appear, and the file copier will be aborted.

PLEASE NOTE: The disk copier can only copy SEQ (SEQuential), PRG (PRoGram) and USR (USeR) files (but NOT GEOS files), and only those types will appear on the disk directory. There may well be other file types on the disk, but they won't be displayed on the disk copier's directory.

Move the flashing cursor up and down using the cursor keys, press + and - keys to move one page up and down respectively, and **HOME** to go to the top of the directory. Pressing the HOME key will place the cursor on the first directory entry.

Select the files you want to copy using **SPACE** (cursor doesn't move), **RETURN** (cursor moves down) or **SHIFT/RETURN** (cursor moves up). Pressing any of those keys on a file which has already been selected will undo the selection of that file. All files can be selected at once by pressing '**A**', or all files can be de-selected by pressing **CLR**.

To start copying, press **F1**. The file copier will start loading the files into memory. If you haven't selected any files, the message "No valid files" will appear, and the file copier is aborted.

Errors may occur during the reading. If a file can't be read, the file copier will inform you about the problem and ask you if you want to retry reading the file. You may want to remove the disk and then re-insert it as this may align the disk more correctly and the disk drive might just be able to read the file. If you don't want to retry the file, that file will be skipped. If a file is too large to fit into the computer's memory altogether, the message "File too large" will appear, and the file will have to be skipped. By the way, the file copier can handle files as large as the available memory, roughly 480 blocks on a old flat 128, 713 blocks on a 128D (or 128 with 64K VDC memory), or 2776 if you add a 512K REU.

When the computer memory is filled up (or there are no more files to be read) the computer will prompt you for the target disk. If you're using two drives however, the copy process will continue without delay.

Again, errors when saving files to the target disk may occur. The file copier will monitor how much space there's left on the disk, and inform you if the space is insufficient. If a file can't be written properly, the file copier will inform you about the problem and ask you if you want to retry writing the file. You may then elect to skip the file, try again, or insert a new target disk to continue. If the file exists, you will be asked if you want to replace the file. The existing file is erased, and the new file (the one being copied from the source disk) will then be written.

PLEASE NOTE: Despite THE SERVANTs safeguard system, there are two reasons why you might encounter the error message "disk full". Some disk drives has a quirk which throws up this error if you try to save a new file when the number of blocks free are 3 or less. The error message will also appear if you exceed the maximum number of files allowed on the disk. Unfortunately, odd things sometimes happen to the disk in these cases, especially in the latter case. To be safe it is probably best to validate the disk if you encounter the "disk full" error message with the file copier. So far, I haven't been able to circumvent those problems.

The file copier will make every effort to avoid error conditions on the copy disk. It will even validate the disk if found to be necessary to recover safely from an error.

If there are more files to be copied than would fit in the computer's available memory, you will be prompted for the source disk (unless you're using two drives) and another pass will be performed for as many times over as necessary to copy all the files.

SCRATCH FILES

This utility lets you select several files to delete which is then scratched all in one go. The scratch utility will work with all disk drives that can be hooked up to the 128, including RAMDOS.

Move the cursor up and down using the cursor keys, the + and - keys to move one page up and down respectively, and HOME to go to the top of the directory. Select the files you want to delete using SPACE (cursor doesn't move), RETURN (cursor moves down) or SHIFT/RETURN (cursor moves up). Pressing any of those keys on a file which has already been selected will undo the selection of that file. All files can be selected at once pressing 'A', or all files can be de-selected pressing CLR.

To start scratching, press F1. As an extra safeguard, you will be asked to confirm your action. If you haven't selected any files, the message "No valid files" will appear, and the scratch utility will be aborted.

UNSCRATCH FILES

This is a utility to bring back files which has previously been deleted. When a file is deleted, it is not really removed from the disk. The DOS (Disk Operating System) makes two changes which render the file as deleted: One, the file type flag is erased (set to 0), making the file unrecognizable to the DOS. Two, the space which the file used on the disk is made available to use as storage for other files. You should therefore restore scratched files before saving something new on the disk. It will work only with 1541, 1571, 1581 and compatible disk drives.

SPECIAL NOTE to GEOS users: You can unscratch GEOS files using this utility. However, you MUST validate the disk from the GEOS desktop before further saving anything to the disk. Furthermore, as an ounce of prevention if you suspect that the disk has been written to since the files were deleted, copy all the non-deleted files to another disk BEFORE attempting to restore lost GEOS files.

You will be asked to insert the target disk, the disk with the files you want to unscratch. The directory is then read into the computer's memory. If there are no deleted file entries in the disk's directory, the message "No valid files" appears, and the unscratch utility is aborted.

Move the cursor up and down using the cursor keys, the + and - keys to move one page up and down respectively, and HOME to go to the top of the directory.

All files will initially be marked as '*DEL'. To restore a file you must select a file type. Your choices are listed at the top of the screen. It is of the utmost importance that you select the right file type, as using the wrong file type will in most cases prohibit the correct function and use of the file. To undo the selection of a file, press D to make it a deleted file.

Some files may be marked as "Blocked". A blocked file is a file which is marked in the directory as deleted, but the block where the first part of the file should be is marked as used in the BAM. This would mean that the block is now occupied by a valid file. A

blocked file is thereby unrecoverable since there is no way knowing where the rest of the file might be on the disk.

A file might be unrightfully "blocked" due to an error in the BAM. If you experience that the unscratch utility identifies that the file is blocked although you think it should be recoverable, validate the disk and attempt unscratching the file again.

PLEASE NOTE: A file name might appear more than once in the directory. You see two or more identical names, and you don't know which is the right one. A deleted file may also have the same name as a valid file. The solution: Restore both. Then rename the first one, the RENAME command will only affect the first file that match the command parameters. Repeat if more than two identical files. Try using the newly restored files to see which one(s) you would like to keep. Scratch the other ones and, IMPORTANT!, validate the disk to make sure the BAM is OK.

Since the disk space previously occupied by scratched files is left free for other files to overwrite, unscratching does imply some potential problems. To be absolutely sure that a file can be restored, it must be unscratched before something new is written to the disk. All might not be lost however, even if the disk has been written to. Each subsequent write to the disk reduce the odds that previously scratched files can be restored. If subsequent writes has occurred, or if you're in doubt, please read the following:

1) A scratched file might be partly overwritten. When you don't find the file contents you expect, the file is garbage, partly garbage, partly missing or has parts that repeat, you are probably dealing with a partly overwritten file. If a new file saved to the disk happens to claim even a single block of the scratched file, that block and the rest of the deleted file is lost. This is why: The two first bytes of each block in the file always points to the next block in the file (unless it's the last block). Now, if one block is overwritten, the pointers for the next block will obviously point towards the rest of the new file. At some point therefore, the two files will seem to have been spliced into one. The solution: If you want to try to recover as much as possible of the file in question, copy it. Either to another disk or by using the COPY command to make a copy under a new name. Scratch the original file and, IMPORTANT!, validate the disk to make sure the BAM is OK.
2) A file might be "looping". This is a phenomenon which is very rare, and equally difficult to recover from. It is related to the "partly overwritten file" problem. What happens is that, during the course of events, the pointers in one of the blocks are now, due to overwriting, pointing back to an earlier block in the same "file". The "file" will then repeat itself into infinity. We can't really call it a file, it's just a complete jumble. The unscratch utility will never be able to finish because it will never stop validating the file. The same goes for scratching, loading, reading or any other operation on the file; it just repeats forever! The Solution: Reset your system. The best solution is to clear the file's identifier in the directory using a sector editor, then validate the disk. If you don't have the means or knowledge to do that, copy all the other files to another disk using a FILE COPIER. If there is a boot sector on the original disk, do a "short new" (formatting the disk without specifying ID characters) on the original disk. Validate the new-ed disk to make

sure that the boot sector(s), if present, are re-protected, and then file copy the files back.

3) The directory entry of the original file might have been overwritten. The place in the directory used by the scratched file has later been used by another file. Unless you know how to use a sector editor and are prepared to spend some time, the file is unrecoverable.

You may restore as many files as you wish. To start restoring the files, press F1. To be absolutely sure, you will be asked to confirm your action. Please note that unscratching may be a lengthy process as the disk has to be validated, reclaiming the space for the restored files. If you haven't selected any files, the message "No valid files" will appear, and the unscratch utility is aborted.

IMPORTANT NOTE: If a disk error message appears at this point, one of the restored files is probably malfunctioning. In any case, the disk hasn't been properly validated, and you should go about with great caution. Using the file copier, copy as many disks as possible to another disk. THE SERVANT file copier will identify the troublesome files. Then, you might want to try recovering those files. You need to read as much as you can from the original file and save it in a new file. Then, scratch the troublesome files and, IMPORTANT, validate the disk.

EDIT DIRECTORY

The objective of this utility is to re-arrange the directory. You also have the option of changing the load address of program files.

You'll be prompted for a disk, and a directory will appear on the screen. This utility will display all files, even the deleted ones. Move the cursor up and down using the cursor keys, the + and - keys to move one page up and down respectively, and HOME to go to the top of the directory. Now you're ready to edit the directory.

PLEASE NOTE: Except for the load address change command, no changes will be permanent until you press **F1** to rewrite the directory. If you mess things up, press **ESC** to back out.

To move a file manually, move the cursor to the file, press **RETURN** or **SPACE**. Using the **cursor keys**, move the file to the desired position and then press **RETURN** or **SPACE** again. Now you're ready to move another file or do something else.

Pressing **S** will sort the files alphabetically. All deleted files will be moved to the end of the directory. If you want to use the sort feature, use it before moving files manually. The sort feature will alphabetize the directory regardless of which order the files was originally in.

C will compress the directory. All deleted files will be removed from the directory. If your disk contains many deleted files, compressing the directory will increase the overall performance. Furthermore, all new files saved to the disk will be tacked onto the end of the directory. With deleted files in the directory, new files will be saved at the first available directory position, which might be right in the middle. This might be

undesirable. PLEASE NOTE: If you rewrite a compressed directory, all the removed files will be lost forever. They can NOT be revived by the unscratch utility.

L will lock or unlock a file. If it was unlocked it will be locked and vice versa. Locking a file will protect it from being scratched. However, it will NOT be protected against formatting of the disk. A locked file appears with a < -sign behind the file type. Deleted files cannot be locked.

T will change the file type of a file. Repeated presses will rotate the file types around. Keep pressing T until the desired file type identifier is shown. You can change the file type to DEL, but this will NOT erase the file from the disk. A DEL file (without a preceding asterisk) is a completely legitimate file type which can be read as any SEQ file. A DEL file cannot be copied using a file copier because the DOS won't allow the creation of a DEL file. Any attempt of doing so will result in a SEQ file being created. You cannot change the file type of a deleted file (DEL file with asterisk). To do that, you must unscratch the file first.

You can change the load address of a program (PRG) file by pressing **A**. All programs have two bytes at the beginning of it which tells from where in the computer's memory it was saved. This address is important in many cases. The BLOAD command will load the file to the address specified in the file, if you don't specify otherwise by using the P option. The familiar LOAD "file",8,1 from the 64-mode loads a file to the address in the file; normally the address from where it was originally saved. THE SERVANT will read the load address of the file and display it in decimal and hexadecimal notation. A cursor appears, and you may enter a new address for the file. On default, a decimal address is required, but you may enter a hexadecimal value if you put a dollar sign in front of it. If you don't wish to change the address, just press the ESC key. If you do enter a new address, it will be displayed in decimal and hexadecimal, and you'll be asked if it's OK to proceed. If you answer Y for yes, the new address will be written back to the disk. Otherwise, you'll get another chance to re-enter the address or exit.

N will let you change the name of a file. A window appears with the current name being displayed and the cursor flashing waiting for you to enter a new name. If you enter a name of an existing file you will be warned about this. You may want to re-enter the file name or ignore the warning and willfully use a duplicate file name. This is not good practice though. If a file name appears more than once in the directory, only the first one can be OPENed or LOADed. On the other hand, scratching the file will scratch all files with identical names.

Press **F1** to save the modified directory back to disk if desired. You'll be asked if it's OK to do it. If not, you'll return to the directory display.

PRINT DIRECTORY

Making a hardcopy of the directory is often practical, and using this utility will allow you to do so. A menu with five options will appear on the screen. THE SERVANT offers a great variety of ways to print the directory for your convenience.

As you select any of the printing options, the directory will be loaded from the disk and the printing will commence. The printer must be a Commodore or compatible printer connected to the serial port as device 4, or any other printer connected to the serial port via a suitable interface. If a printer is not active on the serial bus, you'll hear a bell tone and the menu is redisplayed. After the printer output is finished, you'll be returned to the print menu, and you can immediately select from the menu again to print the directory from another disk.

Option	Meaning
Two columns:	Displays the directory in two columns. In addition to the information normally contained in a directory listing, the start Track (T) and Sector (S) is displayed for each file. All file types will be displayed. Positions in the directory without files of any kind are listed as "- Empty slot -". This will normally occur at the end of the directory.
2 col load addr:	Similar to the above, but instead of the track and sector display, the load address of program files will be listed in hexadecimal format.
Wide output:	This option displays the same information as by the "Two columns" option. In addition, all information in the directory which is normally hidden from the user is displayed. This includes: USR files: For GEOS files, the following information is provided: "IT" and "IS" is the track and sector of the Info/Icon block. This is the disk block which contains the file's icon, start, end and call address, author name, permanent file name and comment field. "ST" identifies the file structure; 0 for sequential and 1 for VLIR (Variable Length Indexed Record). "FT" is the GEOS file type: 0 = non-GEOS USR file, 1 = BASIC program (normally, GEOS BASIC files are PRG files, but GEOS will recognize USR files as well), 2 = Machine language (ditto as above), 3 = data file, 4 = system file, 5 = desk accessory, 6 = application, 7 = application data, 8 = font, 9 = printer driver, 10 = C64 input driver, 11 = disk driver, 12 = system boot file, 13 = temporary, 14 = auto-exec 15 = 128 input driver. REL files: "ST" and "SS" is the track and sector of the first side sector block respectively. "RL" is record length. CBM files: "ET" and "ES" is the End track and sector of the partition area.
Wide w/load addr:	This includes all the information in the options above; track and sector, extra info for USR, REL and CBM files, and load address (in decimal & hexadecimal notation) for PRG files.
Full file check:	In addition to the above, this option displays the length, in bytes, of all file types except REL and CBM. If the length is more than 65536 bytes, the length is output as a mathematical formula which must be resolved using a calculator. Just key in the formula shown on your calculator, and the correct length, in bytes, will appear in the calculator display. For program files the start and end address of each file is also shown.

DISK COPIER

The objective of the disk copier is to make an exact duplicate of a disk. This is especially useful if the original disk contain nonstandard files or data which is not part of files.

IMPORTANT NOTE: All or part of what might be contained on the target disk will be destroyed. If a single side copy is performed, only the front side of the copy disk will be overwritten. The disk copier is dependent of the disk layout. It cannot use a 1581 in conjunction with another drive type. Also, it won't work with RAMDOS.

SPECIAL NOTE to 1581 users: The disk copier copies the WHOLE disk, not only the currently selected partition. The disk copier does automatically select the root directory on the 1581s involved.

The disk copier automatically detects if your computer is equipped with 64K VDC (80 column) RAM, and utilizes it to reduce the number of passes. It can also take advantage of a Ram Expansion Unit of any size if Ram Expansion is enabled at the Disk Tools menu.

You are asked if you want a BAM controlled copy. The disk contains a map, called BAM (Block Allocation Map), which keeps track of where on the disk files are stored. A BAM controlled copy will copy only those tracks which are used according to the BAM. Unless the disk is fairly full, this will reduce the time and number of passes needed to copy the disk. A non-BAM controlled copy will copy the entire disk no matter what. Use this option if you suspect that the disk may hold important data not contained in files or otherwise not marked in the BAM as used.

You will be prompted to insert the source disk (or both disks if you're using a dual drive copy) and the copying process will start. Unless you are performing a copy using 1571 drive(s), the copier will analyze the disk and report if it's single or double sided. The copier can't cope with an error in the BAM, and will report the error and abort if such error occurs.

You may see the message "(Double sided source disk)". This most often means that the source disk is identified as double sided, but your hardware is unable to cope with double sided disks. Data contained on the back side of the disk won't be transferred to the target disk. If you are using 1571 drives only, check the Disk Tools menu to see if your drive(s) are really set to double sided mode (identified as 1571) and not single sided mode (identified as 1571 side 0 or 1571 side 1).

The "Double sided source disk" message may also occur when the disk is a "fake" double sided disk. In the BAM there's a flag that identifies if the disk is double sided. If this flag is set, indicating a double sided disk, while the disk for some reason is formatted on the front side only, the disk is a "fake" double sided disk. This is the case with some commercial software. Before the 1571 was introduced some software used the BAM location, now used for a dual sided flag, for its own purpose. Early GEOS disks fall into this category. This condition will not affect the copying (except that only only one side is copied) or the use of the copy disk. However, you may want to convert the copy disk to dual sided using the Header/format utility.

The disk copier will read as much data as possible into the computer's memory. If the copier can't read a sector due to an error on the disk, it will retry using slow mode (if your drive is using burst mode). The slow mode is slightly better when it comes to reading weak sectors. If that won't do the trick, the copier will report the error, and then continue. The bad sector won't be transferred correctly. A bad sector may indicate three things:

1) Important data may not be stored in the damaged sector. The copy disk will work O.K.
2) The damaged sector holds important data. You may want to use the copy for a repair attempt using a sector editor if you're confident with the use of such a program.
3) The bad sector serves as copy protection. The copy disk won't work. A lot of commercial software has some sort of protection against illegal copying (and legal backup copying for that matter) utilizing bad disk sectors. In this particular case you may (A, use a so-called "parameter" on the copy disk to make it work, (B, try using a "nibbler" duplicating the errors as well as the data, or (C, give up. You may consider purchasing a utility called "The Maverick" (see Appendix F) which includes both nibblers and parameters. No 64 or 128 user should be without that package in any case.

You may try to removing and then re-inserting the source disk and attempt to copy the disk again. Or you may try to use another disk drive. This will sometimes fix the problem.

When the computer memory is filled up (or there's nothing more to read) the computer will prompt you for the target disk. If you're using two drives however, the copy process will continue without delay.

The target disk is first formatted, and the data is then written. If an error occurs during formatting or writing to disk, the error is reported, and the copy is aborted. I can see no reason why you should want to use a bad disk. Discard the disk and try another one.

You may sneer upon the fact that the copy disk is formatted every time a disk is copied, whether or not it is already formatted. However, it is my experience that formatting the target disk will sometimes produce a more reliable copy. I prefer reliability over speed, but if you're of the impatient type, try Maverick or FasTrac (see appendix F), or some other disk duplicating program.

If there is more to be copied, you will be prompted for the source disk (unless you're using two drives) and another pass will be performed for as many times over as necessary to copy the disk.

HEADER/FORMAT UTILITY

It may seem superfluous to include a disk formatting utility since formatting can be done using the DOS commands. This utility however will provide a painless way to format disks in a variety of ways. You can also change the disk name and ID characters without losing data. You are presented a menu which offers the various options available for the

disk drive you are using. This utility won't work with RAMDOS, if you need to reformat your RAM disk, use the DOS command option.

Please note that you should insert the disk you want to use before making a selection, as some selections require to read data from the disk to work. The various options are as follows:

Change header name (all drives):

Change the disk name without disturbing the disk contents. The current disk name is displayed. The header ID is also displayed, and the "real" ID is displayed in parenthesis (unless you are accessing a 1581 drive which doesn't use a "real" ID). You may now enter a new name and/or ID characters. A comma separates the name from the ID. You have three alternatives:

1. Change name only: Enter the new name and no comma.
2. Change ID only: Enter a comma, then the new ID characters.
3. Change name & ID: Enter new name, a comma, and then ID characters.

The computer will check if the name is legal (a disk name must be between 1 and 16 character long, and the ID must be exactly two characters). If something is wrong, you are prompted to re-enter the name and/or ID.

The new name & ID combination is displayed, and you are asked if it is OK to proceed. Check that the name & ID really is OK, and press the Y key to write the new header onto the disk.

PLEASE NOTE about the difference between header ID and real ID: When a disk is formatted two identification characters (called ID) is written to each and every disk block. The disk drive uses the ID to distinguish between the various disks. The ID is also placed into the disk header so that you can check the ID characters used. You can change the header ID using this utility, but the actual ID used seen by the disk drive, the ID that is written to each disk block, does not change. It is recommended that each of your disks is assigned its own unique ID, so having another ID on your directory than your drive actually use will only complicate matters. It is therefore good practice to leave the header ID alone. THE SERVANT will let you change the header ID if you want to disregard good practice or set the ID back to the proper value. The 1581 doesn't need the ID, so you may change the ID on 1581 disks at will.

Format single sided (1541 & 1571):

Formats the front side of a disk. On the 1571, this results in a single sided 170K disk which is compatible with 1541 disk drives. You are asked to enter a disk name and ID. You must enter both, and THE SERVANT will check if the name is legal (a disk name must be between 1 and 16 character long, and the ID must be exactly two characters). You're asked if it is OK to proceed. Check that the name & ID really is OK (and that you are about to format the right disk!), and press the Y key to initiate the format process.

Format whole disk (1571 & 1581):

Formats both sides of a disk. On the 1581, the whole disk is formatted, not just the current selected partition. You're asked to enter a disk name and ID. You must enter both, and

THE SERVANT will check if the name is legal (a disk name must be between 1 and 16 character long, and the ID must be exactly two characters). You're asked if it is OK to proceed. Check that the name & ID really is OK (and that you are about to format the right disk!), and press the Y key to initiate the format process.

Format back side (1571):

The 1571 can use the back side of a disk as a totally separate single sided disk, without flipping the disk over. The front side of the disk should be formatted as a single sided disk (or not be formatted at all). This has little but novelty value, as the back side formatted this way can't be read by a 1541 disk drive by flipping the disk over. To create a double sided disk readable on a 1541, format the disk single sided, flip it over, and format the back side likewise using the "format single sided" option discussed above.

You're asked to enter a disk name and ID. You must enter both, and THE SERVANT will check if the name is legal (a disk name must be between 1 and 16 character long, and the ID must be exactly two characters). You're asked if it is OK to proceed. Check that the name & ID really is OK (and that you are about to format the right disk!), and press the Y key to initiate the format process.

I can imagine two reasons for using this format: 1) The data can be hidden from prying eyes. 2) If you need to store many small files on the disk you may reach the maximum 144 files. Formatting the back side will let you store another 144 files on the disk. To access the back side in your own programs you must use these two commands: 'u0>m0' then 'u0>h1'. To get back, use 'u0>m1'. In BASIC this would look like:

To access the back side:	To get back:

`10 open 1,8,15`
`20 print#1,"u0>m1"`
`30 print#1,"u0>h1"`
`40 close 1`
	`10 open 1,8,15`
	`20 print#1,"u0>m1"`
	`30 close 1`

Alternative format:	
`10 open 1,8,15,"u0>m1":close1`	

PLEASE NOTE: When the back side of the disk has been formatted, the disk drive is reset to its normal condition. To access the back side, you must select the back side manually.

Convert to double sided (1571):

Most software is supplied on single sided disks to accommodate users having access to 1541 disk drives only. You may want to utilize the software using the increased storage space available on a double sided disk. The best way is always to file copy all the files to a separate double sided disk. This is not always possible however. The disk might contain data which is not transferable by a file copier. In such situations this utility may come in handy. It will convert a single sided disk to double sided without disturbing the data on the front side.

WARNING! This operation should never be performed on an original disk unless you are absolutely sure about what you do. Use the Disk Copier to make a backup, and try to convert the backup, NOT the original. Make sure that nothing important is stored on the back side of the disk. Flip the disk over and try to read the disk directory before attempting the conversion.

You're asked if it is OK to proceed. Check that you are indeed trying to convert the right disk.

Some problems may arise when trying to convert a disk:

1) The disk is already double sided. You're informed about this, and the conversion is aborted.
2) The disk is not a standard formatted 1541 disk. THE SERVANT will inform you about this and you'll once again be asked if it's OK to proceed. This condition may be caused by a number of things. THE SERVANT detects this by checking the dual sided flag (a byte in the header which tells if the disk is single or double sided) and checking the area where the BAM for the back side of the disk is to be stored (a part of the back side BAM is stored near the end of the front side header block). Both the double sided flag and the back side BAM area should all contain zeros. Any nonzero value is detected by THE SERVANT, and great caution must be taken. Some software actually store data in these areas, the BAM area might contain data which is necessary for the software to boot. Overwriting such information may render the software unusable. On the other hand, overwriting the BAM area might be quite harmless. For example, some hackers exchange secret messages using this area. There are also a number of reasons why plain garbage might be present in the BAM area.
3) The software might not be usable, even if no vital data is overwritten. The software may have been written exclusively for use on single sided disk for some reason or another. The software might refuse to load, hang up, act strangely, or work perfectly but being unable to exploit the extra disk space. I cannot give any further advice on this, other that you go back and use your original disk. Please note however, that the conversion might not be the reason why the software won't work. If the original disk was copy protected, it won't be able to run from the copy disk in any case.

Fix double sided flag (1571):

This utility will flip the double sided flag on a disk. If the flag was cleared (rendering the disk single sided) it is set, and if it is set it is cleared. You're asked if it is OK to proceed. Check that you are indeed trying to convert the right disk.

You may want to use this utility if you inadvertently have validated a double sided disk while the disk drive was in single sided mode. A bug in the old 1571 ROM will render the disk single sided, while important data might be stored on the back side of the disk. This condition is easily detected by that the drive light blinks and upon reading the drive status you get the message "66, illegal track or sector, TT, SS" when you try to read, load or scratch some files or validate the disk.

On the other hand, you may want to convert a double sided disk to single sided to prevent data to be written to the back side. The flag can later be toggled again to re-enable the back side of the disk.

WARNING: Toggling the double sided flag on a disk which was not originally formatted as double sided MAY put the disk in an odd condition: It can be read as usual, but any attempt to write to the disk (including trying to change the double sided flag back or trying to use the convert to double sided utility) will throw up a "disk id mismatch" error. Disk copying the disk won't fix the problem either. The only solution is to transfer what you can to another disk using the file copier. In conclusion: Leave the flag alone unless you are sure the disk was originally formatted as a double sided disk!

Create subdirectory (1581):

Creating subdirectories on a 1581 is usually a rather cumbersome process. THE SERVANT changes that by offering a visual overview of the disk and the areas which are free to use as subdirectory areas. It doesn't matter if something is already stored on the disk, THE SERVANT won't let you disturb disk contents.

When the Make Subdirectory feature is selected, a map of the disk is displayed on the screen. Tracks 1 through 40 are displayed on the top row, and tracks 41 through 80 are displayed on the bottom row. Each position represents one track. A plus sign marks the tracks which are currently in use. Track 40 will always be occupied, since it contains the root directory.

A flashing cursor will appear at track 1, and you're ready to create a subdirectory. Use the cursor keys to move the cursor around. Below the track map, the following information is displayed:

Start track:	Current track under cursor – the start track if you elect to set the start of your subdirectory area at the current cursor position.
Last track:	The highest possible track a subdirectory can use if it were to start at the current cursor position.
Blocks free:	The maximum number of blocks free possible if a subdirectory start is set at the current cursor position.

Please move the cursor around and see how these values change. A subdirectory area must consist of at least three tracks. The last track and blocks free is displayed as 'xx' if a directory can't start at the current cursor position.

To set the start of a subdirectory, press RETURN. If a subdirectory area can't start at the current cursor position, a bell tone will sound and nothing will happen. Otherwise, you'll be allowed to mark a subdirectory area. Move the cursor to the right to mark the tracks you wish to include in your subdirectory area. The selected tracks are marked with a reversed asterisk as you move along. Please note the change in the track information change meaning:

Start track:	The start track of the current subdirectory area. This won't change as

	long as you are defining an area.
Last track:	Current cursor position, The highest track a subdirectory would use if it were to end at the current cursor position.
Blocks free:	The number of blocks free in the subdirectory if the end of the subdirectory is set at the current cursor position.

Please move the cursor around and see how these values change. A subdirectory area must consist of at least three tracks, and the last track and blocks free is displayed as 'xx' if a directory can't end at the current cursor position. You'll have to move the cursor further to the right.

If you change your mind and wish to change the start track, move the cursor to the left, past the currently selected start track. The selection will then effectively be terminated.

When you have marked your subdirectory area, press RETURN a second time. Now, you'll be asked to enter a file name and ID for your directory. The name must be in the range of 1 to 16 characters, and the ID must be exactly two characters. If the name entered doesn't meet these criteria, you'll be prompted to enter the name again. You'll be asked if it's OK to proceed. If you answer Y for yes, the subdirectory is created. A CBM file is created using the name (but not the ID), and the subdirectory is formatted using the name & ID.

When the subdirectory is formatted, it is selected as the current subdirectory. Please note that you are not limited to creating subdirectories under the root (main) directory. You may create many levels of sub-sub directories until you run out of disk space. Then, the track display will mark all tracks as used except the free tracks inside the currently selected subdirectory area. This is due to the inherent limits in the 1581 partition scheme.

Protect disk area (1581):

This selection will let you create a partition to protect a disk area from being overwritten. First, you are asked to give the partition a name, then you must enter the start track, start sector and number of sectors. You're then asked if it's OK to proceed. Please check that all the entered parameters are correct. This utility does not check for errors, except for those which are trapped by DOS.

Change subdirectory (1581):

To change the current directory, press '/' and enter the subdirectory name. To access the root directory, press RETURN without entering anything (do NOT enter '/'). The start track of the currently selected directory is displayed at the bottom of the screen.

THE AUTOBOOT TOOL

The boot sector is a feature which is supposed to launch software off the disk as soon as the computer is powered up or reset. However, using this handy feature for your own purposes has been very difficult. The auto boot tool consists of several modules which makes creation and manipulation of boot sectors quick and easy. Please note that using THE SERVANT, a boot sector may be created and executed on a disk drive of any device number.

SPECIAL NOTE to 1581 users: Using the Auto boot Tool, the root directory will be automatically selected. A boot sector cannot exist under a subdirectory.

ANALYZE BOOT SECTOR

This feature will decode the boot sector and display its results on the screen. First, it indicates if a boot sector is present, if the boot sector is occupied by a file, or if a "killed" boot sector is present. If a boot sector exists, you'll be warned if it is not protected from overwriting (not protected by BAM). In that case use the Protect Sector(s) feature to protect it if desired. If a boot sector or killed boot sector exists, the following information is provided:

1) **Additional boot sectors info**. The C128 boot system includes an option to read additional blocks off the disk as part of the boot area. These extra sectors starts at track 1, sector 1, and continue onward from there. A total of 255 extra boot sectors can be specified, making the boot area use several whole tracks. The C128 boot feature makes it possible to place these extra blocks anywhere in the computer's memory. How many extra blocks to be loaded, the address and memory bank where they are placed are displayed.
2) **Load PRG file**: A file name can be specified in the boot sector, and if present, a PRG file of that name is loaded into bank 0 RAM. The load address is determined by the load address of the program itself. The file name (if any) will be displayed along with the load address. The load address must be sought in the file itself, and if the file can't be read, a DOS error message is displayed instead of the load address.
3) **ML code start**: The boot sector must contain some ML code to determine what the boot sector is supposed to do. It may be a simple JMP instruction (similar to BASIC SYS), or an elaborate ML program. The start address of the ML code is not fixed; it is determined by the length of the boot message (see below) and file name (above). The start address is displayed as an offset from the start of the boot buffer in the computer's memory. The boot buffer is at address 2816 ($0b00) in bank 0. To calculate the real start address of the ML code, add this number to the offset value.
4) **Boot message**: An optional message can be displayed when a boot is performed (BOOT xxxx ...). Often, the name of the program to be booted is displayed, but the message may contain just about anything. The message may contain control codes which are (by the boot sector analyzer) displayed as RVS characters.
5) **Command line**: Unlike the other information on the analyzer screen, this is not an inherent feature of C128 boot sectors. Rather, by using a small piece of ML code, it is possible to execute a BASIC command line. THE SERVANTs Auto boot Tool contains a provision for creating a command line of your own fancy. Since this is the kind of boot sectors you are probably most often going to create yourself, the BASIC command line (if found) is included here. Incidentally, the same method is used by the "auto boot maker" included on the Test/Demo disk supplied with Commodore disk drives.

Hit any key to return to the Auto boot Tool menu.

DUMP SECTOR

Displays the contents of track 1 sector 0 as ASCII characters, regardless if it contains a boot sector or not. Control characters are displayed as RVS codes. A boot sector has the letters "cbm" as the three first characters. The Dump feature might be useful for revealing things about a boot sector that would otherwise be difficult to disclose. Also, if the boot sector is blocked by a program, you might be able to identify the program which causes the interference.

Hit any key to return to the Auto boot Tool menu.

MAKE BOOT SECTOR:

The objective here is to create a direct mode BASIC program line to be executed upon booting. All immediate mode BASIC commands can be used, and the possibilities are limited by imagination and the size of the boot sector.

Upon selection of Create from the Auto boot Tool menu, you're warned if a boot sector is already present on the disk, or if the boot sector is taken by a program or other data. If you wish to preserve an original boot sector, please use the Boot To File feature first.

Entering the boot message

Optionally, you can enter a message to be displayed upon boot (BOOTING xxxx ...). There are a total of 239 characters available for the boot message and command line. Every character of the boot message will steal space from the command line. Just press RETURN without entering anything if you don't want a boot message.

Please note that you are not limited to plain text; control characters may also be used. Please refer to Appendix F for an overview of the control characters and how they are entered. Also please note that your computer in most cases will be in uppercase/graphics mode (alias "cursor up" mode) when the boot message is displayed. Text entered as lowercase will appear as uppercase, and uppercase characters will appear as graphic symbols. You may circumvent this by entering a RVS n as the first character. This will put the computer into lowercase mode (also called "cursor down" mode) when the boot sector is executed.

PLEASE NOTE: NEVER use a RVS @ in your boot sectors. A RVS @ will yield an ASCII code of zero which will effectively terminate the string or command line. Malfunction will be inevitable.

Entering the command line

This is just like entering a command line in the BASIC immediate mode. Several commands can be used, separated by colons. Loops can be constructed using FOR-NEXT or DO-LOOP/EXIT. Decisions can me made using IF/THEN/ELSE. Please note that you are not restricted to the 160 character limit allowed by the BASIC screen editor. 239 characters are available, minus the length of the boot message (if any). You may use abbreviated BASIC commands if necessary. Of course, the BASIC commands you are going to use most often is BOOT "filename" BLOAD "filename", SYS and RUN "filename".

It is important to use good sense when constructing a command line. Remember that you are limited to BASIC commands legal in immediate mode. This means that INPUT,

INPUT#, GET and GET# are out. READ/DATA, GOTO and GOSUB won't work because the command line doesn't contain line numbers.

Also be careful when using IF-THEN. The command line after the THEN statement will be executed ONLY if the criteria set in the IF statement is true. Let's say that you want to set fast mode if the computer is booted with the 80 column screen active and then run a program. You might try this:

```
if rgr (0)=5 then fast:run "program"
```

This will seem to work perfectly in 80 columns, but the program won't be run in 40 columns, why? The commands after THEN will be executed only if the conditions in the IF statements is true. The line below will cure the problem:

```
if rgr (0)=5 then fast:run "program":else run "program"
```

You must tell the computer that you want to run the program whether or not the 80 column screen is active. Again, please note that any commands following an ELSE statement will be executed ONLY of the conditions in the IF statement is FALSE. This means that if several commands is to be executed in either case after an IF-THEN command, they must all be included twice; once after the THEN command, and once after the ELSE command. Of course, you may consider writing a normal BASIC program which is run by a simple run "filename" command in the boot sector.

Control codes may be entered as part of the command line, but only inside quotes. Control characters outside quotes will invoke a SYNTAX ERROR. You may notice that typing a quote character won't invoke "quote mode". Rather, if you want to include cursor control, insert, delete, home, clear or return codes, hold down ALT and press the appropriate key. Also please read Appendix E.

When the command line is entered, press RETURN to write your boot sector to disk. Before doing so however, the computer will ask you if it's OK to proceed.

Tricks and tips

You may find it very useful to be able to boot a program from any disk drive. PEEKing address 186 will enable you to do so. When the boot sector is executed, this location will hold the device number used. For example running a BASIC program from any disk drive is done like this:

```
run "program",u(peek(186))
```

Please note the use of parentheses. The PEEK command must be enclosed in parentheses for the U specifier to work.

All software might not be able to run off any other disk drive than device 8. To circumvent this sloppy programming practice, you may simply change or swap device numbers. Please refer to your disk drive manual for the command sequence needed. Swapping device numbers, say between device numbers 8 and 9 is done by temporarily setting device 9 to, say, 30, changing device 8 to 9, and then changing the disk drive now using device number 30 to 8.

EDIT COMMAND LINE

You may want to edit a command line boot sector for debugging purposes or other reasons. The boot sector will be read off the disk and a check is made to establish if a valid boot sector really exists and that it contains a command line. If everything is found to be OK, editing the command line is just like creating it for the first time. Please refer to the section about creation of a boot sector above. The boot message cannot be edited.

RUN/64 BOOT SECTOR

This selection will create a boot sector which will load a 64-mode program into computer memory, switch to 64-mode, and run it. If you have a 1571 or 1581 disk drive, the 128 burst load will be active, and will load your program many times faster than in 64-mode. The program to be loaded must be a BASIC program or a ML program which is started with a RUN command.

The boot sector will work with drives of any device number. However, the program you load might not be able to do this.

Please note that the boot sector can't safely handle programs bigger than 153 blocks. However, it is worth trying bigger programs, as they may very well work, especially programs of up to 201 blocks. The boot sector can't handle large program as well as the Main Menu C64 options because of memory constraints. Some RAM under the 64-mode ROMs and I/O is corrupted during setup of the 64-mode.

You will be warned if a boot sector is already present on the disk, or if the boot sector is taken by a program or other data.

Besides entering the file name to be loaded, you are asked if you want to reset the 1571 drive. Resetting the 1571 will switch it to 1541 mode which may be necessary to retain compatibility with the software you are using. If you want to enjoy the double sided mode on the 1571 in 64-mode, select NOT to reset the drive. If you have JiffyDOS installed in your system, you should reset the drive in any case, since JiffyDOS will be active only if the drive is operating in 1541 mode, while still being able to use double sided disks. If you elect to reset the 1571 and the disk is later booted from another drive type, the error light will flash. However, this is without harmful side-effects.

You will be asked if it is OK to proceed, and the boot sector will then be written to disk.

LOAD "",x,1/64 BOOT

This selection will create a boot sector that switches the computer to 64-mode, and then performs a load command similar to LOAD "program",8,1. You are not required to use device 8 however, the boot sector will work with all device numbers. The software you are using might not be able to work with such variety of device numbers though. The boot sector can handle programs of up to 201 blocks.

You will be warned if a boot sector is already present on the disk, or if the boot sector is taken by a program or other data.

You will be asked to enter a file name to be loaded by the boot sector code. Optionally, you may also enter a command line of up to 12 characters to be executed after the program has been loaded. Typically, you would use RUN or SYS to start the program.

Please note that you don't have to add a return character to the command, the boot sector will execute the command line automatically. If the program is auto-running, meaning that it runs by itself without the need for a RUN or SYS call, press RETURN without entering anything.

You'll also be asked if you want to reset the 1571 drive. Resetting the 1571 will switch it to 1541 mode which may be necessary to retain compatibility with the software you are using. If you want to enjoy the double sided mode on the 1571 in 64-mode, select NOT to reset the drive. If you have JiffyDOS installed in your system, you should reset the drive in any case, since JiffyDOS will be active only if the drive is operating in 1541 mode, while still being able to use double sided disks. If you elect to reset the 1571 and the disk is later booted from another drive type, it won't produce any side-effects.

You will be asked if it is OK to proceed, and the boot sector will then be written to disk.

BOOT TO FILE

This feature reads the boot sector off a disk and saves it as a PRG file. It might be useful to use the PRG file as a backup copy of the boot sector, especially if you want to replace the original boot sector. Please note that only track 0, sector 1 will be saved. Multisector boot areas can't be saved by this utility. If in doubt, please use the Analyze Boot Sector option to see if additional boot sectors are present.

The file created by this utility will be fully executable. That is, rather than booting the disk, you may now boot the PRG file:

```
bank 15:boot "bootfile"
```

Sometimes, this approach won't work. Please analyze the boot sector you are going to convert and see if a PRG file is loaded by the boot sector. To make the bootfile work is such cases, use the following command syntax:

```
bload "prgfile",b0:bank 15:boot"bootfile"
```

The above commands may easily be used as part of the command line in a custom boot sector created by the Create utility. For example, you may want to send commands to a printer interface or change disk device numbers before booting your software. Please note that a bootfile will overwrite the boot sector in the computer's memory when executed. It is therefore not possible for a custom boot sector to load or boot a bootfile and then return for further BASIC commands. This condition will invoke a SYNTAX ERROR, or could even crash the system. The execution of the bootfile should be executed LAST in a custom boot sector command line.

The transfer process begins by reading the boot sector off the disk in the drive. You'll then be prompted for a file name, then the target disk, a disk where to store the resulting file. The target disk must be put into the same drive that was used to read the boot sector.

FILE TO BOOT

To do the reverse of the above, use this utility. A PRG file is read from disk and its contents is placed in the boot sector. The combination of the boot transfer utilities may

be used for archival purposes or for transferring boot sectors between dissimilar drive types. The utility can't handle multisector boot areas, files larger than 256 bytes is truncated.

This utility may equally well be used to enable a boot sector entirely of your own design, where you yourself have written the ML code. A PRG file to become a boot sector must have the following structure:

1) 7 zero bytes. Actually, the three first bytes should contain the ASCII characters "cbm", but these are automatically provided by the transfer utility. You may substitute the three first bytes with a JMP instruction to the start of the ML code for debugging purposes. Incidentally, this is what the Transfer Boot To File utility does.
2) Optional boot message (BOOT xxxx ....). All CBM ASCII characters can be used and the length of the string is limited only by the size of the boot sector.
3) One zero byte (regardless if you include a boot message or not).
4) Optional PRG file to load. If you provide a file name at this point it will be loaded into bank 0 memory (similar to BLOAD "file",b0), but it will NOT be called. Any call address must be provided in the ML code (see below).
5) One zero byte (regardless if you provide a file name or not).
6) ML code. This is normal 6502 ML code.

The exact start of the ML code will be determined by the length of the boot message and/or file name, if any. In its simplest form, a boot sector may contain 9 zero bytes (no boot message or file name) and the ML code must then start at the 10th byte. Any use of boot message and/or file name will force the start of the ML code upwards. As the computer's boot buffer starts at 2816 ($0b00), the actual entry point of the ML code can be calculated using the following formula:

2816+9+length of boot message+length of file name

The transfer process is initiated by a prompt for a file name and the file will be read off the disk. You will then be asked for the target disk, the disk where to put the boot sector. You'll be warned if the target disk already has a valid boot sector or if the boot sector is occupied by a file. Then, the boot sector is saved to disk.

COPY BOOT SECTOR

You may easily copy a boot sector from one disk to another. Please note that only track 1, sector 0 will be transferred. Multisector boot areas cannot be transferred by this routine.

The original boot sector is read off the disk and you'll be prompted for the target disk, the disk where to save the boot sector. You'll be warned if the target disk already has a valid boot sector or if the boot sector is occupied by a file. Then, the boot sector is saved to disk.

KILL AUTOBOOT

You may want to remove the boot feature from a disk, and this selection will enable you to do so. Actually, this utility isn't very radical, the "cbm" boot sector identifier is changed

to "kbm", and the boot sector is marked as free in the BAM. It may be a good idea to transfer the boot sector to a file before killing it.

You'll be asked if it is OK to commence the action to be performed. Then, the boot sector will be disabled. You'll be notified if a boot sector doesn't exist on the disk, and no changes will then be made.

UN-KILL AUTOBOOT

If you have killed a boot sector, either deliberately or by accident, it may be re-enabled with this feature. Since the Kill feature marks the boot sector as free in the BAM, the boot sector might easily be overwritten by a file saved to disk. It is therefore feasible to un-kill a boot sector as soon as possible.

The disk is checked to see if it contains a recoverable boot sector, and you're asked if it is OK to proceed. The boot sector is then resurrected.

PROTECT SECTOR(S)

For a boot sector to be safe, it must be protected by the Block Availability Map (BAM). Although the DOS commands validate will protect the boot sector, the BASIC COLLECT command and other ways of validating the disk will de-protect it again. To make things right, the Auto boot Tool protects command will re-protect the boot sector(s) without validating the whole disk. Please note that multisector boot areas WILL be protected by this utility.

APPENDICES

Appendix A: INSTALLATION INSTRUCTIONS

PROCURING THE CHIP

Before actually procuring the chip, it is important that you modify THE SERVANT to suit your own preferences. Please refer to appendix B.

Once your servant has been modified, it's time to make a chip. THE SERVANT is designed to be programmed into an EPROM (Erasable Programmable Read Only Memory) chip. To do that, you'll need an EPROM programmer. Any 64/128 programmer will do.

You may even attempt transferring THE SERVANT code to MS-DOS using Big Blue Reader and use a PC equipped with an EPROM burner card. In that case, please note that the PRG file produced by THE SERVANT (using CONTROL/+) has two bytes in the beginning which holds the start address of the file, which is not an actual part of THE SERVANT code. You must get rid of those two bytes in one way or another. You may load THE SERVANT into a word processor or text editor and use it to delete the two first bytes. Alternatively, the EPROM burner software might have the ability to skip a given number of bytes in the beginning of the object file.

The chip itself must be a 27256. Program it using the EPROM programmer as described in your EPROM programmer manual. The 27256 come in two versions; One for 12.5 Volts programming voltage, and one for 21 Volts. Often, the programming voltage is printed on the chip itself. If in doubt, try programming using 12.5 Volts first. Then, if it won't work, try 21 Volts. If you don't have an EPROM programmer yourself, some member of a local user group might have one. If you wish to buy an EPROM programmer, please find the address of Jason Ranheim in appendix F.

If you can't find an EPROM programmer, just send a 27256 EPROM to the author. There will be a turnaround fee; please refer to the license terms printed in the file named "read me". The EPROM will be returned with THE SERVANT programmed into it. Please read appendix B on customizing THE SERVANT and describe as closely as you can how you want your SERVANT to be configured. I will read and follow your instructions! You may, at any time, return your SERVANT EPROM if you wish to alter THE SERVANTs configuration. The same turnaround fee as above will apply.

INSTALLATION

THE SERVANT can be installed in the internal ROM socket in any 128 or 128D, in a ROM cartridge, or even inside the Commodore 17xx series Ram Expansion Units. I will include instructions for each of the above.

Opening your computer tends to make you both nervous and eager at the same time, and you may tend to become absent minded. Therefore, read the whole installation guide and troubleshooting section, BEFORE opening your computer. What it all boils down to is that if you feel confident, you are more likely to avoid blunders.

COMMON TO ALL C128 MODELS:

Turn off your entire system. Unplug all cables (even keyboard cable on 128D), and take your computer to the kitchen. Computer chips are sensitive to static charges (but not as sensitive as often claimed), and the kitchen is a room where static normally don't occur. No carpets and a fairly high humidity. Don't wear clothes made of synthetic materials. Wear wool or cotton. You need a large clean working space, at least twice as wide as your computer. You'll probably be surprised how much space your computer takes up in disassembled condition. Then put on your surgeon's gloves, scalpel in hand, and dive into it. It is of paramount importance to your success that you take your time and use good common sense. When unplugging connectors, note (using pencil and paper) where they belong and their orientation, just to be on the safe side. Marking the connectors might also have the side effect that it increases your confidence of you're a bit nervous in the outset.

Tools required:

- A medium sized Phillips screwdriver
- A pair of flat-nosed pliers (for the flat 128)

You may also consider:

- Pencil and paper
- A small flat screwdriver or chip pulling tool
- A pair of tweezers (to pick up lost screws)
- Contact cleaning spray (ozone friendly!)
- A chip pin straightening tool
- A chip insertion tool
- An ashtray, egg carton or similar to hold the screws.

Opening the "flat model" Commodore 128

1) Turn the computer up-side down, and remove the six screws.

   Turn the computer back the right way. There are some small plastic hooks on each side near the top of the keyboard which still holds the two halves together. Pull the two halves apart in the REAR LEFT CORNER just enough so you can get your fingers in-between. Then, using brute force, pull the two halves apart.

   On the left, there is a small connector with three wires attached to it that goes to the power LED. Unplug it.

2) Now, turn the keyboard over towards the right. You will notice that it hangs in the keyboard wires. There's no need to unplug those.
3) Remove the seven screws around the edges of the metal shield. The screw at the front right has a grounding strap from the keyboard which you must detach temporarily. Some computers also have a screw roughly in the middle of the shielding (near the RF connector). The shield is also held down by a number of metal tags around the edges. You'll have to bend them aside. You may even break them off as they serve no useful purpose. You may also have to bend aside the tags surrounding the plastic stalk in the middle.

4) In some 128s the shielding is even soldered in place at one point. If that is the case, bend the shielding up and aside enough so that you can access the rear left portion of the circuit board.
5) Now, please refer to the chip insertion section below.

Opening the 128D portable (Plastic case with carrying handle)

1) Turn the computer upside down and remove the four screws.
2) Grasp the computer firmly with both hands (so it won't fall apart uncontrollably) and turn it over.
3) Remove the cover by lifting it slightly at the back and then pulling it back slightly.
4) The lid is attached to the power lead to the main circuit board, so you'll have to turn the lid over and lay it down to the left of the computer.
5) Pull out the disk drive lever
6) Remove the three screws just behind the front cover at the top.
7) Lift the front panel up and away from the base.
8) Remove the disk drive circuit board, removing the five connectors and the two fixing screws.
9) There is a wire from the circuit board to the main shielding which will have to be removed.
10) Slide the circuit board to the left and lift it away.

    Remove the power supply. Undo the four screws and remove the connector in the far right corner. Remove the wire attached to the disk drive unit. Lift the power supply away.

11) Now, read the installation instructions below.

Opening the "new" 128D (Metal case with no carrying handle)

1) Look at the rear and find the three largest Phillips screws that are fastened to the top lip on the case on the back. If you are lost, do step 2 first, and then return to this step.
2) Turn the computer upside down. There are two screws to undo on the underside. One very close to the cassette port and one on the other side (near the power light).
3) Grasp the computer firmly with both hands so that it won't fall apart uncontrollably, and then turn it the right way up.
4) Now the hardest part. Grab the top of the case and slide it back from the rest of the 128D till it stops (just short of an inch movement or about 2 cm). Pulling from the top lip where the three first three screws were seems to work best. Now lift the cover straight up and off of the 128D. Notice how the cover slides off of the four tabs which fit into the four slits on the top cover (two on each side).
5) Now, read the installation instructions below.

INSTALLING THE CHIP

If you have a flat 128 or 128D portable, you will soon notice the empty socket on the rear left. In the "new" 128D however, the socket is located 2.5 inches from the FRONT and 2.5 inches from the left. In either case, the socket is marked U36. That is the internal ROM socket.

Before insertion, you will probably have to straighten the pins of the chip. On a fresh new chip, the two rows of pins are normally pointing slightly apart. You will need to straighten them so they are completely vertical. You may hold the chip in both ends, and lay one row of pins flat down on the table. Then roll the chip slightly so that the pins become straight. Repeat the process with the other row of pins. DO NOT bend the pins so that they point inwards. Alternatively, you may use a pin straightening tool if you have it.

Now, insert the chip. The chip has a notch at one end which must point towards the front of the computer. All the other chips in the vicinity are oriented the same way. Check that each and every pin goes into a hole in the socket. Push it down.

A common problem when the chip is first inserted is that one or more of the pins don't make proper contact with the socket. This is due to oxidization of the hitherto unused socket contacts. Malfunction will result. To remedy the problem, lift the EPROM slightly in both ends and then push it down again. Repeat a couple of times.

COMPUTER RE-ASSEMBLY & TESTING

Follow the disassembly instructions for your computer in reversed order. Connect the power cord and monitor (and keyboard if applicable). Don't connect other peripherals yet. Turn on your monitor and let it warm up for about thirty seconds. Then turn on your computer. THE SERVANT main menu should appear on the screen. If THE SERVANT does appear as expected, power down and connect your peripherals. Enjoy!

Help! It doesn't work

Don't despair (yet). Take a deep breath and calmly follow the test procedure below that fits your problem in a step by step fashion.

No screen display:

1) Check the power lights on your computer and monitor. Do they indeed shine?
2) Check that the computer monitor are using the same screen mode and is properly hooked up to each other.
3) Try removing THE SERVANT chip.
4) If the system is still dead, the computer itself must have become faulty. Hard luck! Shake the computer and listen for a rattling sound. There might be a metal object (maybe a screw) which makes a short circuit somewhere. Also, you may check the few wires on the circuit board to see if they have come loose at the solder points. Also check that everything is re-assembled properly. This goes especially for the 128D portable.

The C128 power-on message appears, the system then hangs up:

1) Press the reset button. If that doesn't help, turn off the computer, wait for approximately 15 seconds, and then turn back on.
2) If you have connected the serial cable, power down and unplug it. If the computer works now, try to locate the peripheral which causes the interference. This should not happen with any Commodore brand peripherals; printer interfaces are probably the most likely to cause problems although every effort has been put

forth to avoid it. I should note that if you are using a 1581 disk drive, it will hold back THE SERVANT for a moment. Please write the author and explain your problem if you find that a certain peripheral is causing problems.

If none of these suggestions will cure the problem, read the section below.

The computer powers up just like it did before the ROM installation, THE SERVANT menu appears, the system then hangs up or goes haywire, THE SERVANT works, but hangs up or goes haywire after a while:

There is some problem with the chip. It may be faulty, which is not very likely, or one of more of the pins are not making proper contact with the socket.

1) Inspect the chip where it sits. Is it indeed oriented with the notch towards the front of the computer? If not, turn it around and test. Surprisingly, ROM chips often survive being inserted the wrong way around.
2) Remove the chip; check that no pins are bent out of shape which may account for the malfunctioning. Carefully straighten any bent pin(s), and then re-insert the chip.
3) If all pins were OK, then clean the socket contacts using a contact cleaning spray or fluid. Re-insert the chip and test.
4) Try bending the two rows of pins on the chip slightly closer together.
5) If nothing works, send the chip to the author for testing. Please describe the nature and symptoms of the problem and everything you have done to fix it. If possible, please state your SERVANTs version number.

THE SERVANT works OK, except that it crashes when using the Datamaker and some Disk Tools sections:

1) The chip might have been programmed using the file "the servant.bin". This file lacks a part which is essential to THE SERVANT's operation when working in ROM. THE SERVANT must be programmed using the file "the servant.mod" as created by pressing CONTROL/+ on the main menu.
2) The file used to program the EPROM was saved to disk using BSAVE, ML monitor S command or by other means. While this method MAY produce an error-free file, it is very unreliable. Same cause, effects and cure as 1) above.

MOUNTING THE SERVANT INTO A CARTRIDGE

THE SERVANT will work without modification in a cartridge. However, this solution is not recommended unless you have a very good reason to choose it. Mounting THE SERVANT in a cartridge will tie up your expansion port, and as soon as you remove the cartridge your computer becomes plain stock again. THE SERVANT in a cartridge will not work switched on at the same time in conjunction with other cartridges (except 17xx REU's) using an expansion board. If you can, mount THE SERVANT in the internal ROM socket!

The cartridge you wish to use must be able to hold a 32K ROM chip. Some cartridges can be reconfigured using jumper wires or switches to hold different kinds of chips. The cartridge manufacturer should supply the cartridge with instructions on how to set up the cartridge. PLEASE NOTE: There is a significant difference between 64 and 128-mode

cartridges. Make sure that you get a 128 type. Some units have jumpers or switches to configure the cartridge for 64 or 128. Using a 64-mode cartridge will make the computer always power up in 64-mode. However, a 64 only cartridge can be modified for use on a 128. Here's how: Hold the cartridge the right way up with the connector towards you. On the top side, locate contacts number 8 and 9 counting from the left (counting 1-2-3 not 0-1-2 and remember to count missing contacts). Cut the leads connected to those two contacts, and cut the two contacts apart if they are joined together. The cartridge should now work in 128-mode. For troubleshooting, the same will apply as for the internal ROM installation. However, the cartridge won't work if it's improperly configured.

MOUNTING THE SERVANT INSIDE THE 17xx SERIES RAM EXPANSION UNITS

This modification is only for the confident soldering iron slingers among you. This explanation will therefore be very brief. Please note that the DMA chip inside the REU is extremely sensitive to static charges. When opening the REU, you will see the outline of a chip printed on an otherwise empty part of the circuit board. Just outside the outline there are two rows of holes. Mount a LOW PROFILE 28 pin IC socket into the holes and solder it in place. Mount THE SERVANT chip into the socket with the notch the same way as shown on the outline printed on the circuit board (notch to the left, towards the big REC chip). You'll also need to alter a jumper or two on the REU's PCB. There are two versions of the Commodore REU, and the altering of jumpers is different. The two REU types are distinguished by the big REC chip close to the edge connector:

A) REU with SQUARE REC chip: Change over J2 (it actually looks like I2). The jumper looks like three tiny metal squares. Normally the middle one and the one furthest away from the edge connector is interconnected by a tiny strip of metal. You must cut the existing connection and connect the middle square to the one closest to the edge connector using a drop of solder.

B) REU with RECTANGULAR REC chip: You'll need to connect two jumpers: One is marked J5, connect the two metal squares. The other is marked J2 and consists of three metal squares. Connect the middle one to the left one; the one marked 27256.

Re-assemble the REU and test. PLEASE NOTE: If you mount THE SERVANT into the REU you will no longer be able to use your REU and other cartridges switched on at the same time using an expansion board.

IMPORTANT NOTE about cartridge & REU installation: Due to that THE SERVANT chip (when mounted externally) and the QBB cartridge use the same memory space, the QBB features won't work in this configuration. Any attempt to use the QBB and THE SERVANT switched on at the same time using a cartridge expander board will probably make THE SERVANT malfunction as well.

Appendix B: CUSTOMIZING THE SERVANT

Since THE SERVANT pops up whenever you turn on or reset your computer, it would be quite annoying if you were stuck with THE SERVANTs original appearance.

Modifying THE SERVANT is not hard. First, load THE SERVANT into computer RAM memory. To verify that THE SERVANT really is present and working in the computer's

memory, invoke THE SERVANTs main menu. You should see the author's name and address below the main menu, and the message "Hit CONTROL/+ to save EPROM burner file". DO NOT save THE SERVANT yet. First, modify your copy according to your preferences. To do that, hit RETURN to return to BASIC. Now, you're ready to make modifications.

FUNCTION KEYS

Whenever you power up or reset your computer, THE SERVANT will install its own function keys definitions. The default definitions are as follows:

```
F1 = dload "                  (Load a BASIC program)

F2 = dsave "@                 (Save a BASIC program)

F3 = cA                       (Directory on disk drive 8)

F4 = cAu9                     (Directory on disk drive 9)

F5 = clear screen, set colors, set fast mode if using 80 columns.

F6 = bank15:sysdec("1300")    (Call a ML routine at 4864/$1300)

F7 = lI                       (List a BASIC program)

F8 = moN                      (Go to the ML monitor.)
```

Type KEY, and then RETURN. A list of the current function keys will appear on the screen. Please note how control characters are utilized. The control characters will make the function keys work flawlessly, even if they are pressed in the middle of a screen full of text (try that using the C128's default definitions!).

Please note about F5 above: Even if you wish to keep the overall definition of the key, you may wish to modify it to suit your color preferences. The definition of the colors in the function key string does NOT automatically change with the default color preferences you set for THE SERVANT itself (see below).

You can change or modify any of the definitions using the BASIC "KEY" command. If you want, you can change the function of the HELP key and the SHIFT/RUN-STOP combination as well. Here's how:

To change the RUN key:

```
bank 15:sys 24812 ,,8,,,"definition"
```

To change the HELP key:

```
bank 15:sys 24812 ,,9,,,"definition"
```

Replace the word "definition" with the actual definition you want to use. Please note the use of commas; you'll need to use them exactly as shown for this trick to work.

You won't need to include a function key definition to invoke THE SERVANT itself. However, THE SERVANT will automatically claim one function key definition (of your own choice) for this purpose. See the section on THE SERVANT call key below. Please note that the total number of characters in the function key strings is limited to 246. THE SERVANT call key use 16 characters, so a total of 230 characters are available for your use.

When you think your custom function keys are OK, test them to verify that they really work according to your expectations. The final step is to copy the definitions into THE SERVANT itself. Do it using the following BASIC line:

```
bank0:for x=0to255:poke32795+x,peek(4096+x):next
```

DISABLING THE SERVANTs FUNCTION KEY INSTALLATION

If you don't want THE SERVANT to install function keys upon power up or reset, use the command:

```
bank0:poke33058,0
```

This will disable THE SERVANT function keys installation. To re-enable the function key definition, poke a nonzero value into location 33058. Please note that you cannot disable THE SERVANTs automatic call key definition (see below).

SETTING THE SERVANT CALL KEY

One key must be reserved to call THE SERVANT. THE SERVANT automatically sets this definition. It's up to you however, to decide which function key THE SERVANT is going to use. Use the command:

```
bank0:poke33057,function key number.
```

The function key numbers are 1 through 8 for the function keys 1 through 8; 9 is the SHIFT/RUN-STOP combination, and 10 is the HELP key. Take your pick. The default is 9, using the SHIFT/RUN-STOP combination.

40 COLUMN SCREEN COLORS

```
Text color:       bank0:poke33051,color code

Background color: bank0:poke33052,color code

Border color:     bank0:poke33053,color code
```

The color codes are as follows:

```
0 = black   4 = purple    8 = orange      12 = medium gray

1 = white   5 = green     9 = brown       13 = light green

2 = red     6 = blue     10 = light red   14 = light blue

3 = cyan    7 = yellow   11 = dark gray   15 = light gray
```

80 COLUMN SCREEN COLORS

```
Text color:       bank0:poke33054,color code
```

```
Background color: bank0:poke33055,color code
```

Please note that these color codes are different from the codes for the 40 column screen. The color codes are as follows:

```
0 = black      4 = dark green   8 = dark red     12 = brown

1 = dark gray  5 = light green  9 = light red    13 = yellow

2 = dark blue  6 = dark cyan   10 = dark purple  14 = light gray

3 = light blue 7 = light cyan  11 = light purple 15 = white
```

SETTING UPPERCASE ON EXIT

If you have played around with THE SERVANT a bit, you'll notice that the letters are lowercase when you exit.

To set uppercase upon exit:

```
bank0:poke33056,142
```

To set lowercase upon exit:

```
bank0:poke33056,14
```

DISABLING FAST (2 MHz) MODE

THE SERVANT will use fast mode whenever the 80 column screen is active. If you need to, you can prohibit THE SERVANT from using fast mode:

```
bank0:poke32774,81
```

Setting this byte to any other value (avoid 0 or 1; it will make THE SERVANT malfunction) will enable fast mode as usual. Setting this location to 0 will disable THE SERVANTs auto start. Still, THE SERVANT can be called by bank12:sys32777 and will work flawlessly.

SAVING THE MODIFIED VERSION OF THE SERVANT

When you have made all your modifications, type `bank0:sys32768` and press RETURN to cold start THE SERVANT. Check that the colors are as you expect. Exit THE SERVANT again and check that the function key definitions work as expected. Press THE SERVANT call key to see if THE SERVANT main menu pops up as expected. Try all of this in both 40 and 80 column mode.

If everything seems OK, save THE SERVANT to disk by holding down CONTROL and pressing the "+" key. Please make sure that the disk has at least 130 blocks free. THE SERVANT will be saved to disk drive 8 using the file name "the servant.mod". If a file of that name already exist on the disk (a previously saved SERVANT), that file will be deleted. Please note that you should NOT use BSAVE from BASIC, "S" from the machine language monitor, or any other save function. If THE SERVANT is saved in any of those ways, it may be corrupted. Use CONTROL/+!

The file just saved can be programmed directly into an EPROM.

IMPORTANT: You MUST test your modified THE SERVANT running in computer RAM before you have it programmed into the EPROM. A wrong POKE during your customizing may make THE SERVANT malfunction. It is better to reveal errors at this point rather than when THE SERVANT is physically installed into your computer. An error discovered at after the installation will make it much harder to figure out where the error really is.

Appendix C: THE SERVANT MEMORY USAGE

This is an overview of how THE SERVANT uses the C128 memory. This may serve as a help to avoid conflicts with other applications or to understand why conflicts might occur.

When you don't actually see THE SERVANT on screen, your computer is completely "stock", except for colors & function keys. THE SERVANT does not hook into your computer's operating system in any way whatsoever, unlike cartridges for example. This is the key to THE SERVANT high level of compatibility. The trade off is the lack of some features like a fast loader, DOS wedge and additional BASIC commands.

ZERO PAGE:

The zero page is used heavily, but on a non-interference basis. That is, THE SERVANT uses locations which are normally changed by BASIC under normal circumstances. Utilities or "wedges" should not suffer under THE SERVANTs use of the zero page. One exception however, is the Datamaker. Since this utility utilizes the BASIC screen editor, it has to use some memory locations not used by BASIC. Most notably this applies to locations 251 through 255 ($fb - $ff).

STACK:

THE SERVANT assumes the use of the upper half of the stack. This means that addresses 384 to 507 ($0180 - $01fb) is used. Memory conflicts are avoided because THE SERVANT resets the stack pointer whenever it is invoked.

INPUT BUFFER:

The input buffer at 512 to 672 ($0200 - $02a0) is used heavily for various storage. Since this area is used by BASIC as input buffer, and thereby frequently corrupted, no conflicts should occur. Again the memory is used temporarily on a non-interference basis.

VECTORS:

THE SERVANT changes no pointers or vectors except one. The KEYCHK vector (828/$033C) is changed temporarily to disable the function keys. But it is gracefully set back to its original value before The Servant is exited.

BASIC RUNTIME STACK:

BASIC reserves a 512 byte area in the range 2048 to 2559 ($0800 - $09ff) as a stack to process commands like FOR-NEXT, DO-LOOP, GOSUB etc. In the Disk Tools section, THE SERVANT uses this area as two buffers. The first 256 bytes is used as a general purpose working buffer, while the upper 512 bytes is used to hold the directory path for 1581 disk

drives. The directory path is used when using the file copier with one and the same 1581 disk drive. Since this area is normally used by BASIC, no conflicts should occur.

APPLICATION PROGRAM AREA AND BASIC PROGRAM AREA (Bank 0):

The Disk Copier and File Copier use the entire bank 0 from 4864 ($1300) and up as part of its main data buffer. Otherwise THE SERVANT doesn't touch this area. The exceptions from this rule are obvious: If you load a BASIC program for example, it will of course replace any BASIC program which happened to be in the memory already. IMPORTANT NOTE: When THE SERVANT is working in bank 0 RAM, bank 0 memory will also be used by the Disk Tools for storing directories (see below).

BASIC VARIABLE STORAGE AREA (bank 1):

All Disk Tools sections which uses a directory, use bank 1 memory from 16384 ($4000) and up to 32768 ($8000) according to the directory size. The File Copier and Disk Copier use the entire bank 1 as buffer space. It is not possible to protect memory from being overwritten by THE SERVANT.

BANK 0/1 COMMON MEMORY:

The 128 hardware has a provision for reserving a memory area common to bank 0 and 1 at the top, bottom or in both ends of the memory range. Under normal operation, the 128 is set up using a 1k common memory area at the bottom of memory. The Disk Tools section temporarily extends the bottom common area to 16k to retain compatibility with RAMDOS. The common area is set back to 1k upon exit.

VDC MEMORY:

If your 128 is equipped with 64K memory for the 80 column chip, the entire memory space is used by the Disk Copier and File copier. Only the area used for screen display, attribute memory, and the lowercase character definitions are unused to retain the screen display even if the 80 column screen is active. This way, THE SERVANT can squeeze 56K out of the VDC memory. Because the uppercase/graphics characters are overwritten, the ROM character definitions are copied back to the 80 column memory upon exit.

RAM EXPANSION UNIT:

The Disk Copier and File Copier can make use of any size REU. Please note however, THE SERVANT won't identify any REU to be bigger than 1Mb, even if your REU does have more memory onboard. Since duplicating 1581 disks is the most memory hungry job you're ever going to do within THE SERVANT, this limitation won't hamper you. Also, using the REU is purely optional; there's no need to corrupt the REU's contents if you don't want to.

Appendix D: HARDWARE PROBLEMS

Commodore computer systems have always had bugs and quirks of varying severity. Some of the quirks may affect the operation when THE SERVANT is in charge of operations. Several known bugs have been circumvented in the design of THE SERVANT,

but then there are some that can't be patched over. Below you'll find a list of the bugs and quirks known by the author that you may encounter.

ALL DISK DRIVES:

When you attempt to exceed the number of files permitted, odd things sometimes happens. The problem is worst on the 1571 (and possibly the 1541). Space is allocated for the new file while there is no room for the file name in the directory. The bug will especially affect the file copier. If you see the error message "disk full", you should validate the disk to put things right again. It should be noted that you won't encounter the disk full error if the file copier runs out of disk space (not exceeding the maximum number of files). A safeguard system prevents that. So far, I have not found a way to circumvent the problems that occur when you try to save too many files. So for the time being, validate the disk after a disk full error.

1571 DISK DRIVE:

The bugs apply to the 128D portable (128D with cooling fan & carrying handle), and the stand-alone 1571. However, upgrade ROM's has been released by Commodore to fix all problems. The Commodore part number of this ROM is 310654-05. You can get yourself the fix ROM by sending a 27256 EPROM along with the same turnaround fee and shipping charge that applies for SERVANT EPROMs. An even better alternative is JiffyDOS, which further enhances the performance of your system (Please refer to appendix F).

Problems that may occur using "old-ROM" 1571's with THE SERVANT:

1) Random "device not present" errors. THE SERVANT may give up and make a bell tone on the Main Menu or aborting by displaying "I/O error" in the Disk Tools section.
2) Sequential file corruption. When a sequential file use blocks on both sides of the disk, it will sometimes get corrupted. Affects the File Copier.
3) The drive can't burst load "locked" files. Affects several SERVANT features.
4) Several bugs affect the speed of operations. The disk drive goes click-click-click for a very long time trying to recognize some disks, and the drive is very slow at writing data to the back side of the disk.

A public domain program exist named "1571.diag" which will check your drive to establish which type of ROM it contains.

1581 DISK DRIVE:

Three faults are known to the author:

1) At rare occasions, a disk may become severely corrupted. This is due to the WD1770-00 disk controller chip used in the original 1581's. This controller was later replaced by a WD1772-00 chip which solved that problem. It is not easy to replace the chip if you have the "wrong" one, because it is soldered in place. If you change chips, you may also short jumper J1 with a 47 ohm resistor. Although this is not essential, it will allow the drive to use the 6 ms step rate.

2) When creating a sub-sub directory (one subdirectory inside another subdirectory), the parent directory would sometimes be erased when the sub-sub directory was supposed to be formatted. In the design of THE SERVANT, an attempt has been made to patch up the problem, but I haven't been able to verify if it really fixes the problem. Please report if you run into trouble using THE SERVANT's partition creator making sub-sub directories.
3) Some 1581s has a poor solder joint on the motherboard. This joint is on pin 10 on device U10. It causes occasional "device not present" errors. THE SERVANT may give up and make a bell tone on the Main Menu or aborting by displaying "I/O error" in the Disk Tools section.

A program exists in the public domain called "1581.diag". It will identify the controller chip and the jumper J1, but not the poor solder joint.

1541 FORMATTED DISKS:

Some disks formatted by old 1541s or compatible disk drives seem to malfunction when read using a 1571 in 1571 mode. The disk seems to spin forever. This also happens with some older C64 commercial programs. The drive will recognize the disk if it's left alone for a while though, and the delay may repeat every time the disk is read. The delay can be eliminated by switching your 1571 to 1541 mode.

From V4.84 and up, Call Exrom will work with the ROM chip called KeyDos. This didn't work before due to the inferior method used by KD to sense in which memory bank it is located. Not a TS bug, but a workaround has been employed which will ensure compatibility. However, the SYS address provided by KD at power-up to invoke it will not work. This is not a big problem, because you can use Call Exrom while holding down ALT, and KD will install gracefully.

The RAMlink (and probably RAMdrive) from Creative Micro Designs don't work well together with THE SERVANT nor any other function ROM chip. This is a problem inherent to the RAMlink hardware. However, THE SERVANT will work smoothly if RAMlink is switched off.

Appendix E: COMMODORE RVS CODES

Commodore ASCII has a number of control codes besides all the so-called printable characters. Printable characters are letters, numeric digits, mathematic operators, punctuation and graphic characters. The other doesn't print, but they control the screen output in various ways, for example by changing the letter color. In THE SERVANT we need to represent these control characters in a way that won't change the screen output. Printed normally, the control characters would have fouled up the screen display. THE SERVANT therefore, use reversed letters to represent the control characters. Many of them will probably be familiar, as a normal programming technique is to include RVS characters in strings. The C128s "quote mode" is a means of entering RVS characters in strings. For example, pressing the HOME key while in quote mode will yield a RVS 's'. Below you will find a list of all RVS code letters, along with their ASCII code value and meaning.

RVS codes can also be entered using those THE SERVANT utilities that require text input. By holding down ALT, you can enter the "lowercase" RVS codes. By holding down ALT and SHIFT simultaneously, you can enter the "uppercase" RVS codes. Please use good sense though when entering RVS codes; using them in a wrong way could mess things up considerably. The input routine provides alternative ways of entering some RVS codes for your convenience; the input routine doesn't have the familiar quote mode used by BASIC. The alternative entries are listed in the column "alternative entry" below.

COMMODORE RVS CODE TABLE, Lowercase RVS codes:

DEC	HEX	RVS code	Function	Alternative entry
0	00	@	Null character	
1	01	a	-	
2	02	b	Underline ON (80 col.)	
3	03	c	RUN/STOP key	ALT/RUN/STOP
4	04	d	-	
5	05	e	White	CONTROL/2
6	06	f	-	
7	07	g	Bell tone	
8	08	h	SHIFT/C= OFF (64 mode)	
9	09	i	Tab (SHIFT/C= ON, 64 mode)	ALT/TAB
10	0A	j	Line feed	ALT/LINE FEED
11	0B	k	SHIFT/C= OFF (128 mode)	
12	0C	l	SHIFT/C= ON (128 mode)	
13	0D	m	RETURN	ALT/RETURN
14	0E	n	Switch to lowercase	
15	0F	o	Flash ON (80 col.)	
16	10	p	-	
17	11	q	Cursor down	ALT/CRSR DOWN
18	12	r	Reverse ON	CONTROL/9
19	13	s	Home	ALT/HOME
20	14	t	Delete	ALT/DEL
21	15	u	-	
22	16	v	-	
23	17	w	-	
24	18	x	-	
25	19	y	-	
26	1A	z	-	
27	1B	[	ESCAPE*	ALT/ESC
28	1C	\	Red	CONTROL/3
29	1D	]	Cursor right	ALT/CRSR RIGHT
30	1E	^	Green	CONTROL/6
31	1F	_	Blue	CONTROL/7

* The ASCII character immediately following the code will determine the actual function. Please refer to the table "Escape Codes" for a rundown of the Escape Sequence Codes.

COMMODORE RVS CODE TABLE, Uppercase RVS codes:

DEC	HEX	RVS code	Function	Alternative entry
128	80	À		
129	81	A	Orange/dark purple	CBM/1
130	82	B	Underline off	
131	83	C	SHIFT-RUN/STOP	ALT/SH/RUN/STOP
132	84	D	HELP	ALT/HELP
133	85	E	F1	ALT/f1
134	86	F	F3	ALT/f3
135	87	G	F5	ALT/f5
136	88	H	F7	ALT/f7
137	89	I	F2	ALT/SHIFT/f2
138	8A	J	F4	ALT/SHIFT/f4
139	8B	K	F6	ALT/SHIFT/f6
140	8C	L	F8	ALT/SHIFT/f8
141	8D	M	SHIFT/RETURN	ALT/SHIFT/RETURN
142	8E	N	Switch to uppercase	
143	8f	O	Flash off	
144	90	P	Black	CONTROL/1
145	91	Q	Cursor up	ALT/CRSR UP
146	92	R	Reverse off	
147	93	S	Clear screen	ALT/SHIFT/CLR
148	94	T	Insert	ALT/SHIFT/INST
149	95	U	Brown/dark yellow	CBM/2
150	96	V	Light red	CBM/3
151	97	W	Dark gray/dark cyan	CBM/4
152	98	X	Medium gray	CBM/5
153	99	Y	Light green	CBM/6
154	9A	Z	Light blue	CBM/7
155	9B	Û	Light gray	CBM/8
156	9C	Ü	Purple	CONTROL/5
157	9D	Ý	Cursor left	ALT/CRSR LEFT
158	9E	¦	Yellow	CONTROL/8
159	9F	ß	Cyan	CONTROL/4

ESCAPE CODES

Whenever you see or use a RVS [, the following ASCII character determine the actual function of the Escape Code Sequence. Please note that the letters in this table should NOT be entered as a RVS code, but rather as a normal lowercase letter. The 128 supports the following codes:

DEC	HEX	Letter	Function
64	40	@	Erase to end of screen window
65	41	a	Auto-insert ON
66	42	b	Set window bottom right corner
67	43	c	Auto-insert OFF
68	44	d	Delete logical line
69	45	e	Cursor blink OFF
70	46	f	Cursor blink ON
71	47	g	Bell enable
72	48	h	Bell disable

73	49	i	Insert blank line
74	4A	j	Go to start of line
75	4B	k	Go to end of line
76	4C	l	Screen scroll ON
77	4D	m	Screen scroll OFF
78	4E	n	Reverse OFF (80 col. only)
79	4F	o	Quote/insert mode OFF (also ESC-ESC)
80	50	p	Erase from start of line to cursor
81	51	q	Erase from cursor to end of line
82	52	r	Reverse ON (80 col. only)
83	53	s	Block cursor (80 col. only)
84	54	t	Set top left corner of window
85	55	u	Underline cursor (80 col. only)
86	56	v	Scroll up
87	57	w	Scroll down
88	58	x	Swap 40/80 display
89	59	y	Set default tabs, 8 characters spacing
90	5A	z	Clear all tabs

Appendix G: THE SERVANT DEVELOPMENT

This appendix contains an overview of the release dates of each release. The changes made for each release is also described.

Version 1:

My first experiments. The Main Menu found its form, and the directory was developed.

Version 2:

The "Load "*",64 feature was added to be able to load commercial software. The greatest enhancement though, was to include my assembler, the Midnight Assembly System, into the chip.

Version 3:

The QBB file manager was added. The size of the code increased so that there was no longer room for the Midnight Assembly System.

Version 4:

THE SERVANT got its name. The Disk Tools section was added. The idea of selling or otherwise distributing THE SERVANT formed. Some prospect users showed an interest and received an information sheet. The progression of the 4.x versions further enhanced many features. Version 4.5 got the ability to live in bank configurations 0, 12 and 13 without the need for making separate versions.

Release #1, Version 4.6, January 31 1992:

This is the first public release of THE SERVANT.

Release #2, Version 4.7, March 26 1992:

Only minor changes were made comparing to Version 4.6 in addition to bug fixes:

- The fact that the Directory Printer was part of the Disk Editor was found to be pretty awkward. The Directory Printer was separated, and got its own entry on the Disk Tools menu.
- Rename command was added to the Directory Editor.

Release #3, Version 4.8, May 10 1992:

Disk report feature added. Actually, this version was released in order to correct a number of bugs that had surfaced in the previous versions.

Also, an error in the docs has been corrected. There was an omission in the Ram Expansion Unit installation pertaining to the alteration of a jumper inside the REU. Without this alteration, the REU-installed SERVANT will go haywire upon power-up or not work at all.

Release #4, Version 4.82, Jul 1 1992

Still swatting bugs. A severe bug introduced in V4.8 had to be fixed. If using a single 1571 or 1581 disk drive and you copy enough stuff to trigger a second pass, the system hung up when the prompt for the target disk appeared.

A minor enhancement has been added to the DOS commands. When at the DOS command line, you can revive the last command by pressing SHIFT/RETURN.

Release #5, Version 4.85, Oct 5 1992

The main reason for this version is difficulties with co-existing with another ROM chip called KeyDos. Its specific design made the Call Exrom command crash the system.

Some small enhancements have been made: The Scratch and File Copier modules now have a counter for the number of tagged files and the sum of blocks they use. The Disk Tools directory counts all files and also displays blocks free on disk. Extended Dir and Edit Directory displays empty directory slots, and will also display non-standard file types. In the Disk Tools section, you can swiftly move to the bottom of all directories by tapping the British pound key. A few minor bugs have also been corrected. JiffyDOS users will appreciate that THE SERVANT won't disturb JD's default device number.

The installation instructions have been updated to cover two different versions of the Commodore Ram Expansion Unit.