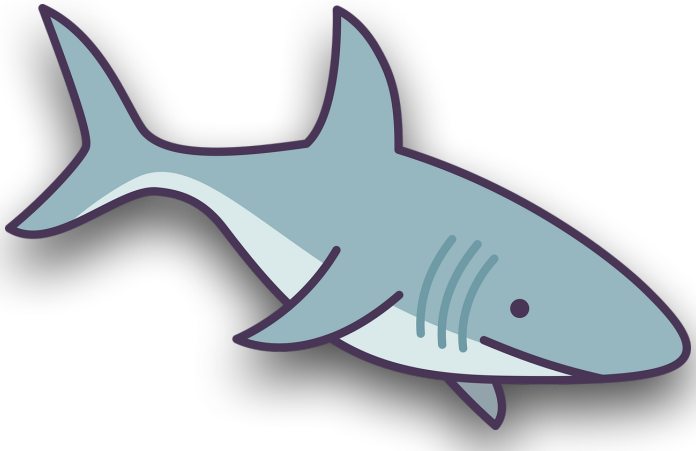
BW-DOS 1.5

06-11-2024

Manual
and
History

Holger Janz & Jiří Bernášek



Revised
09-04-2025

Content

Introduction

BW-DOS is a disk operating system for 8-bit ATARI computers with at least 16kB RAM, and at least one disk drive. It is designed to be compatible with SpartaDOS and to use as little memory as possible. In addition, it does not contain some small bugs known from SpartaDOS, especially concerning the XF 551 disk drive. BW-DOS may be used separately or as a "junior version" for SpartaDOS users. BW-DOS may be freely used and distributed under following conditions:

- Only the whole and unchanged master disk may be distributed,
- It may not be sold with, exceptions are PD services at cost price,
- When used as a part of another software package, the whole package may be marked with a "(C)", sold etc.; in this case it's possible to exclude files from BW-DOS, but all distributed files still must be unchanged, and an information where the whole BW-DOS is available must be included.

BW-DOS uses a SpartaDOS compatible disk format. It supports SpartaDOS File System version 2.0 with an extension that it supports more than 126 entries per directory (see chapter Disk Format). That means it is possible to use BW-DOS disks under SpartaDOS (X), and SpartaDOS (X) disks under BW-DOS. It is compatible with SpartaDOS including the boot process. Most of CIO commands and other features are also compatible, so a lot of programs written for SpartaDOS will work under BW-DOS without any problems. Disks with capacity up to 16 megabytes are possible using drives with double density. The valid size of a file is any size below 8 megabytes.

BW-DOS is a command driven system - it is more flexible than a common menu driven systems like Atari DOS. An advanced menu program is also available. Up to 5 files may be open at the same time, and five disk drives are supported (1, 2, 3, 4 and 8).

BW-DOS does not use the RAM under OS-ROM (XL/XE machines only), so it is compatible with software which uses this memory space, such as the popular Turbo Basic, and others. The MEMLO value is $1D65 for BW-DOS version 1.5 CP. The value is low enough to support all programs that works with Atari DOS 2/2.5. BW-DOS version 1.5 with internal commands has a MEMLO of $1EFE. This value should support most programs. One exception is Inter-Lisp/65 that needs a maximum MEMLO of $1D80. MEMLO value may increase by resident commands.

BW-DOS supports MEMLO up to $2800 e.g., for installing resident commands or drivers. If MEMLO is above $2800 some commands will not work anymore because some of them start at $2800.

Because of limited memory space, BW-DOS doesn't support some of the functions known from SpartaDOS as internal commands. They are supported by external and resident commands. Atari DOS 2 disks are only supported by the menu program and special commands.

The write lock status of disks known from SpartaDOS is not supported by BW-DOS. If write lock is set with SpartaDOS then this is not protecting from changing or formatting these disks with BW-DOS.

BW-DOS comes as a complete package. That means it contains a common set of known DOS commands, an editor to edit text and hexadecimal files, and an assembler to develop new commands or programs in 6502 assembly language. Furthermore, the source for the BW-DOS kernel and its commands are provided so one can assemble the whole system on itself.

The following rules apply for examples and explanations in this manual:

- <KEY> - This tells you that you need to press specified key.
- ... - (three dots) means "and so on".
- UPPER CASE LETTERS, and other characters such as numbers etc. must be written exactly as shown in the manual.
- lower case letters - It must be substituted by corresponding parameter, for example name of the file you want to work with.

If you find any bug in BW-DOS or its documentation then please send me an email. Certainly you can also send complimentary and critical comments to Holger Janz (holgerjanz@abbuc.social). The original source of version 1.3 was provided by Jiří. The current version of BW-DOS and all previous versions including sources and documentation can be found on GitHub at: https://github.com/HolgerJanz/BW-DOS.

History

This chapter is a making-of BW-DOS for the interested reader. The original author explains how BW-DOS was developed in the 90’ and it describes how BW-DOS was rediscovered and further developed almost 30 years later.

The Beginning

by Jiří Bernášek (JB)

The story begins back in 1989, in Czechoslovakia ruled by communists. It was both heaven and hell for free software. For the authorities, a computer was barely more than a pocket calculator, so the users were expected to type-in the necessary formulas themselves, and share the experience with others. All software was in fact free (despite any copyright messages), copying was the preferred way, encouraged even in official clubs. Tons of programs circulating between friends with no fear. On the downside, there was no official software import, no market, insufficient documentation, and no respect to the integrity of original code. With very limited hardware import (often just for those with personal contacts to the west), it was also the realm of tape recorders, the cheapest data-storage solution. To top it all, a local tape-speeder called "Turbo 2000" (1987) gained great popularity. It was a port of Sinclair ZX Spectrum tape routines, establishing a new platform incompatible with Atari system API's, resulting in the need to crack & patch all the available software. When I wanted to have TurboBasic, as an example, I was provided with a dozen versions overnight - all patched and expanded with various drivers of local hardware, but none compatible with a DOS (did anyone even know, what a DOS is - in the realm of tape?) This was my first lesson, how free software was useful, but still some level of quality (and access to original version) was needed.

Then the miracle happened (thanks to family contacts), and the XF 551 disk drive joined my 800 XL. It was no fun to make the U.S. drive work with European computer (which is also where my contact with ABBUC started), and another challenge was software. The box contained DOS XE, with it's horrible user interface, high memory footprint, and failure to read the omnipresent DOS2 disks, so I was in need of another DOS. The heaven of free software provided quite a few options, but I was disappointed: DOS 2.5 changed my 360kB double speed drive into basic speed 130kB, MYDOS failed miserably at first disk change (just like others), BiboDOS worked, but proved unreliable with the XF. It's a specific drive, oh well. In the end SpartaDOS won, thanks to flawless operation with full XF capacity (which is inherent to its filesystem, even if just by coincidence), DOS2 backwards compatibility, and efficient command line UI (thanks to Czech translation of the manual, available at local Atari club). But still, the memory footprint was high, and no XF 551 high speed transfer. This is where the ultimate goal first came to my mind - or was it so unreasonable to want just reliable operation with full speed and capacity, efficient user interface, and full compatibility with Atari software?

As years passed, I slowly shifted from early games and demos to more serious software. I also learned a few lessons about the copyright law - and also how the commercial model doesn't really work for small platforms like Atari: With copying illegal and purchase unavailable, the potential of new software is wasted. I saw several attempts to make profit on Atari software - all more or less unsuccessful. My belief in free software became permanent.

I eventually patched my copy of SpartaDOS 3.2 for XF 551 high speed (and CompyShop compatible Ramdisk), created a game loader (Micro-SpartaDOS), various small command line utilities, a hardware-based XF 551 speeder, a Directory Master, a packer for executable files, a new menu for ABBUC magazine. With all these projects, I was gaining insight into the Atari system. But unlike the relatively well documented OS-ROM, SpartaDOS 3.2 became a mysterious black box for me. A handful of bugs came to my attention, one of them rather nasty (a root directory meltdown through usage of third-party utilities with frequent disk changes in a single drive), so I started to feel uneasy about SpartaDOS.

I evaluated the SpartaDOS X cartridge (which I had since 1992), but didn't like it's approach. The MS-DOS standard (of the time...) didn't go well with the Atari system, creating a new, hugely incompatible platform - I've seen this approach already with Turbo 2000, and it was no good. The system was high-profile both in terms of performance and memory footprint (despite the extra memory in the cart), and usage of new features created a dependency on the cartridge, so that widespread usage was very unlikely. Evolution of wide user-base, beyond the dated DOS 2 and its disk format, needed quite the opposite: A reliable, moderately modern system with maximum availability and compatibility - which means free, disk-based DOS with just basic memory footprint. I was deeply unsure, whether I would be able to create such a thing or not, but I decided to give it a try (or at least to examine the SpartaDOS insides and perhaps patch bugs).

And so came the memorable evening of January 17, 1994, when the disassembler of ATMAS II was driving a Seikosha printer to utter madness. When the last of 72 pages of SpartaDOS disassembly came out, shortly before midnight, the printer was hot like hell, covered in pillows from my bed to muffle all the noise. Several evenings with a pencil over the pages followed. I learned how the basic structure is not really difficult, but also how Sparta was optimized for high performance and user experience. With its support for two different filesystems, three methods of file reads/writes, two methods for VTOC searches and two for POINT operations, rich set of internal CP commands, textual error messages, high speed SIO and keyboard buffer in core, or extensive 2kB buffering space, the memory footprint was bound to be always high.

It was time to try and write some alternative, so a month of coding followed. My Atari witnessed unusual combinations, like BW-DOS command line running over SpartaDOS core as an app, or a system only able to open a file and then hang up. Fortunately, the compatible API and disk format allowed for relatively easy connections between both the systems in this phase of core testing. Some structures were taken from SpartaDOS, some re-designed and minimized, some dropped, some improvements included, a few bits even came from SpartaDOS X (most notably directory sizes up to 32kB). The only code identical with Sparta was the binary-to-decimal conversion of numbers in directory listings, because I just couldn't make it any shorter. Since I always hated how SpartaDOS external commands had to stay in the root directory to ensure their accessibility across the disk, the first BW-DOS prototype got a PATH command modeled after MS-DOS (and SpartaDOS X), accepting multiple pathnames and drive numbers to search for commands. Date/time format got adjusted to European style (DD.MM.YY and 24-hour clock), and I was rather surprised to see that same order of fields already existing in the SpartaDOS insides. Many comparative tests against SpartaDOS 3.2d were conducted to ensure compatibility of all implemented calls, revealing also more problems in SpartaDOS - the final list, including later updates, collected 19 bugs and insufficiencies of SpartaDOS 3.2 (most of them avoided in BW-DOS, where similar code exists). Yet another challenge was the ATMAS II assembler and it's relatively small capacity for source. It was my favorite assembler since the old tape-driven days, when file-to-file compilation wasn't an option, and integrated IDE was very desirable, but BW-DOS went quite over it's size. Code comments were the first to go, then descriptive label names, then the source was split in half, then into three parts with rather tedious manual process of linking. Documentation was always just on paper, in a pair of hand-written notebooks. By the end of February, I moved the thing from $1C00 down to $700 for the first independent boot.

It worked just fine, but MEMLO somewhere around $2400 was not acceptable, so I rushed to reduce the code. Several commands went to external (most notably COPY, SAVE, or clock support), a few features got dropped altogether (LOCK/UNLOCK, discontinued in SpartaDOS X as well), my precious PATH command ended it's short life, to be replaced with the simple ">DOS>" pattern. When there was nothing more to remove, I went for tricky code optimizations. On the opposite side, I had to re-introduce one of the performance-related hacks known from both Sparta and DOS 2.5, because the completely stripped-down BW-DOS prototype (with just basic byte-oriented CIO operations) was too slow to process file reads in sync with floppy disk rotation. To reach the desired MEMLO, I finally even introduced an "Universal Copy" call, to interpret sequences of short pseudo-ops for various memory transfers and related subroutine calls. After weeks of effort, it finally got below the magical value of $2000, with enough margin to install a few little resident drivers.

Now, it was getting really tiresome. My projects were always exciting at the start and boring at the end, and I always had (and still have) the tendency to underestimate their size, but a two months'

intensive work without getting anywhere near the finish - that was something new and unpleasant. On the other side, there was a working DOS core already, with promising features, and I was pretty sure that it just can't stay unfinished.

And so the third month was dedicated to all the small utilities, commands and drivers. These were not so much limited in size, so I aimed for further improvements: Batch files got basic level of interactivity (IF/ELSE), resident drivers got the ability to change settings and uninstall, code debugging tools got extended (MEMEDIT/HEXEDIT/DISASS), and more. Some of my previously existing little tools got updated, and joined the package as well (Micro SpartaDOS, Directory Master, BLOAD, CUT, XBAT; some of them date back to 1991). The formatting tool didn't go beyond XF 551 densities simply because I haven't seen any other drive (let alone a HDD), and the XF specifics scared me enough to avoid any risk in this area. For advanced configurations, I anticipated BW-DOS to become a "junior" alternative for SpartaDOS users, so formatting large disks with Sparta was an acceptable compromise.

I was getting exhausted, but there were still the two most scary items left, with no possibility to postpone them any further: BACKUP (actually an improved rewrite of my old disk archiving tool "ACOPY"), and MENU, with all it's DOS 2 import/export code. These two became the biggest beasts of it all, and took another two months to finish.

By the end of May, the code was more or less complete, but the project still had to continue. I had to write some kind of a manual (for which the Czech clone of original SpartaDOS manual was a great help). Now a funny thing happened: The news about FTe taking over SpartaDOS, and releasing it as Shareware, reached me shortly before finishing my project, effectively taking away one of the main motivation points. Fortunately, the other points were still valid. I was absolutely sure about releasing BW-DOS as free software, but my previous experience also dictated some quality standard - I really wanted to avoid a repetition of the "dozen hacked TurboBasics" scenario. That's why BW-DOS got the Freeware - don't modify label. Now that I can't provide updates anymore, I released it to Public Domain.

So, after five months of work, BW-DOS 1.00 was finished. I decided to release it with style, so I printed the whole manual with a nice Epson printer (recently provided by ABBUC in return for the new magazine menu), completely exhausting the ribbon cartridge in the process, paid some copies in a copy center, bought nice covers, and created ten original packages in professional style. While the post carried them to my friends and contacts, I was already on my way to summer holidays, thinking that the project is finally over. How naive I was!

I was delighted to receive positive feedback, but there were further requests as well. Wolfgang wanted a non-free version for ABBUC, so I created the version 1.10 (just a rebranded copy of the free 1.00 version) for the 1994 Jahresausgabe disk, which came with a nice printed manual in German. What a pity, that the newly released SpartaDOS made it to the magazine a few months earlier, but I was still happy to see it.

When the ABBUC version came out by the end of 1994, I was already working on further improvements, in large part due to valuable input from Erhard Pütz. His suggestions and feedback greatly helped the Micro SpartaDOS game loader back in 1993, and now a stream of classic paper-based mail connected Prague with Germany for good 18 months - actually, half the improvements are more or less based on Erhard's input. The process was far from ideal, though. These old letters reveal a lot of in-depth discussion, but also a sad picture of many brilliant ideas getting dismissed due to "lack of time". Oh well, back in the 90's I certainly wasn't ready to maintain such a project. What I expected to be just another of these one-month "release and forget" games, turned into endless nightmare of changes and updates, well over 2 years long. Atari was my entire world back in the day, but I was exhausted and missing the thrill of mixed small projects, I had a new and more demanding day job, also the amount of mail went up with BW-DOS published, and so my tempo dropped. I had to draw a line somewhere, so I pitifully backed out of all the difficult suggestions, but managed to finish the rest at last. It was Erhard's feedback, and my belief in the importance of the project, what kept me going so long. (Thanks, Erhard!

Beside of numerous small fixes and improvements, a lot of attention was on further hardware drivers (big Ramdisks, real time clocks), often written without real hardware, and tested remotely via floppy disks sent by post. A lot of talk was around I/O redirection (batch files etc.), where SpartaDOS 3.2 established a powerful concept of system-wide redirection through changed "E:" device driver, but didn't go far in terms of compatibility with further screen drivers. While SpartaDOS X dropped the concept entirely as a part of its shift away from standard CIO workflow, we tried to enhance the compatibility (mostly for Erhard's XEP-80, even if I disliked the device). Another busy topic was about the pointers for file/directory sector allocation, where the lightweight BW-DOS configuration (with few buffers and no internal speeders) revealed various problems with the allocation strategy inherited from SpartaDOS. We did our best to improve it, even if the result wasn't ideal. A discussion about a PATH enhancement (command-searching in further locations) went over and over, boiling down to lack of space for extensive code in core, and lack of usable API for external solution, so just the DOSDRIVE compromise was added in the end. We added the MOVE command, and discussed a lot of further options like large hardware drivers in banked memory, textual help and unified command line syntax in external commands, lifting the line length limit for TYPE, various features from SpartaDOS X menu utility, and even large hard disk partitions with long sectors. Most of these further ideas were either too problematic, or went beyond my time-budget, and so remained unrealized. With the slow speed of pre-internet communication (well over a month per exchange, with some work on the project in between), our collaboration never went beyond consultations, so I had to do it all myself. As time went by, I released two update packs, BW-DOS version 1.20 (quickly replaced with 1.21 due to incompatibility with Q-MEG OS), and the final release 1.30.

Throughout this phase, I also had a contact with JRC, a Czech distributor of gaming stuff, back in the day still supporting the 8-bit machines. As a result, BW-DOS got support for their plug-in Ramdisk module (RAMBOX), and a new release with nicely printed Czech manual eventually hit the shelves in their shop. Since the company was founded by the same person (Jiří Richter), who created the Turbo 2000 tape speeder, it's no wonder that I also got talked into making a Turbo tape driver for BW-DOS, to take it to a new level. They already had a DOS with a Turbo tape driver (TT-DOS, with 1kB blocks format), but it was just a DOS2 clone with all its limits. I did my best to integrate the thing with BW-DOS, so it supported a lot - random file-access through audible rewind/forward indications, tape booter and game loader... I released the thing as B-TAPE, but my feelings were rather mixed, so I kept it strictly separate from the main BW-DOS package. Turbo was never really recognized outside Czech republic, and local users were not used to DOS workflow (let alone it's size and booting time with tape) - the local scene was already dominated by cartridge-based version of the old primitive loader instead. The boom of turbo tape was already long gone, and I doubt whether B-TAPE was ever seriously used.

As the support for 8-bit computers and 5.25" floppies was rapidly fading, I went to the JRC shop in Prague city center one day in 1996, to buy another XF-551 drive and a stack of floppy disks. The shop assistant saw what I was buying, and actively asked me, whether I already have BW-DOS (which was on the shelf too). What a situation to experience!

After that, I took a break, and rushed to catch up with the evolving style of game/demo scene (with little luck, after the 3 years absence). Initially I had a plan to resume BW-DOS improvements later on, but I never really dared to touch the beast again (despite a variety of suggestions landing in my mailbox until the very end of my Atari activity in 2002). By the end of 1998, I was already aware of my inability to proceed with BW-DOS, and I was considering a possibility to give sources to another trustworthy person (Stefan Birmanns). I didn't really want to block further progress, but I was also still worried about quality, afraid of the possibility of multiple forks undermining the professional image of my DOS - simply not accustomed to the Open Source concept in general. With all the sluggish paper-mail in the way, the discussion faded out with no result.

Meanwhile, a HDD was connected to my Atari, so I finally saw my BW-DOS going into the realm of megabytes, and was happy to see it working well. As the last addition came the BTC in 1999 (a configurable text encoding converter, replacing my old BASIC-based utility from 1992), more or less fulfilling one of the postponed Erhard's wishes. It's not really a part of the BW-DOS package, but prove very useful for management of Atari archives of text files, and I use it to this day (with Atari emulator).

About the name BW-DOS: There was a friend in the same class, Miroslav Werner, trying to create some stuff on his Sinclair ZX Spectrum computer, just like I did on my Atari. We decided to produce our stuff as versions for both the platforms, and so we invented the name for our little group - Bernášek and Werner Software, BeWeSoft. This is how BW-DOS got its name .

30 Years Later

by Holger Janz (HJX)

It started again in the beginning of 2019. I was standing in the storage room in the basement of our house looking for my old Atari ST. At this time my life started to be easy again. My children have grown up and my professional job was serious but in clam water. So I had time again for a hobby and I decided to revive my Atari ST.

The Atari ST was my working horse during my whole studies at university. My fellow students have always been wondering how I did it. They used to have Linux computers with 386 CPUs and I just had a Atari Mega ST with Minix.

So as I have been browsing all the boxes in our basement I discovered a box with big letters: ATARI 800 XL. There I remembered that my first computer around 1987 was an Atari 800 XL. At this time the subject computer science was pretty new at school but I was fascinated from the beginning. I was born and grown up in Rostock a city on the Baltic in East Germany. Computers had been very expensive at this time. In West Germany the wave of affordable home computer was already rolling. I had the same experience in East Germany like Jiří in Czechoslovakia. I was happy to be able to finance a computer for myself: my own Atari 800 XL. At the beginning I could not afford any storage device. So my first mass storage device had been paper and pencil. Later on I could afford a XF 551 disk drive. At this time there was a quite big community around the Atari 800 XL in East Germany even it was hard to get information and software. My XF 551 came with DOS 2.5. I really was wondering what Atari did: deliver a disk drive that is capability of storing 360k but deliver software that was only able to use 130k. I was happy to get the hands on a copy of BiboDOS from CompyShop that supports 360k disks. My favorite assembler had been ATMAS2 by Hofacker and Finzel. In 1990 I switched to the new generation of Atari computers, the Atari ST. Unfortunately, because of this decision I did not notice the development of BW-DOS described by Jiří and had no idea of SpartaDOS at this time.

With this memory again in my mind I decided to get my Atari 800 XL out of the basement and to revive it instead of my Atari ST. I grabbed the box and set up everything on my desk. I was really surprised, everything was still working without any issues after almost 30 years. Even all my old disks are still readable. So I was starting to play (work) again with my old Atari 800 XL using an old 810 disk drive that was my second drive in the 80`s. Unfortunately, in the late 90's a friend of mine managed to persuade me to sell my XF 551 to him. Now I started to use Atari DOS 2.5 with ATMAS2 at this new beginning. Later on I discovered there is still a big and great Atari 8bit community. Even the ABBUC is still alive. So I renewed my ABBUC membership and bought some modern equipment like a SIO2SD device, The!Cart multi cart, and IDE plus 2.

I have been used to command line interfaces for disk operating systems. So my next question was: Does a DOS with command line interface and modern features like batch processing and input/output redirection exist for the Atari 8bit?

Searching the internet for disk operating system with command line interface, the top results are SpartaDOS X. It is quite easy to use SpartaDOS X with The!Cart multi cart. I was fascinated again what was possible with Atari 8bit computers. This disk operation system does have it all what I have dreamed of. Why couldn't this have existed in the beginning of the 80's?

There was one point that bothered me: SpartaDOS X works only from cartridge. That is an advantage because it is easy and fast and it is a disadvantage, you need a cartridge. So the next search was for DOS like SpartaDOS X but booting from disk without cartridge. Furthermore, I was looking for a DOS that does not touch extended memory or RAM under OS ROM. Certainly, the first hit was SpartaDOS. I liked version 1.2 because it does not use the RAM under OS ROM but has a very high MEMLO. That makes it not usable for a lot of applications. SpartaDOS 3.2/3 on the other hand had a pretty low MEMLO but uses the RAM under the OS ROM and I ran in all the issues Jiří described above. After further searches I discovered BW-DOS.

BW-DOS had everything I was looking for. For me it is like the little sibling of SpartaDOS X. There are some things I wanted to change. I was looking for the source code but one could only find

very few information about BW-DOS. I already got to know Erhard Pütz and he told me some parts of the story of Jiří and that he is not active anymore and nobody has any contact information. There is a post address in the original documentation but I thought it is not valid anymore after 30 years. Later Jiří wrote me that it is indeed still valid and I just had to try. The next lesson I learned: Always try even if you have some doubts.

What to do? I decided to disassemble BW-DOS to get sources. I did it before with my now favorite assembler, Fast Assembler for SpartaDOS X but this is a different story. Certainly I was very curious how BW-DOS works and how Jiří managed it to put all this functionality between $700 and $1EE4. So in the name of science I started to disassemble the DOS file.

I was pleasantly surprised how well structured the DOS binary was. Only some parts where hard to be analyzed. These parts turned out to be the magic tool for the small size of the DOS binary what leads to a small size in memory and a low MEMLO. I call it the Move-Call-Engine. The 6502 CPU has no instructions to move words (two bytes). For this reason the 6502 needs two load (3 bytes each) and two store (3 bytes each) instruction. That means to move an address and a length (16bit) that is often used to set IOCBs, one need code with the size of 24 bytes (moving two words). This Move-Call-Engine is a subroutine that takes the parameter from behind the JSR instruction. The decisive point is that most of the time BW-DOS move only data between $700-$1FFF. All addresses involved are less than $2000 that means the 3 most significant bits of an address are 0. These bits are used to code instructions and length. The first address was used to code also the instruction like jump to subroutine and move data. The second address was used to code the length of data to be moved (3 bits = 0-7 bytes). With this coding a move of an address and length only requires the size of 8 bytes instead of the 24 bytes. Certainly, the call of Move-Call-Engine and the subroutine itself needs an amount of bytes for its code, too, but if it is called often enough then there is a break even. The time I had figured that out, the work was done.

Now I was able to extend and adapt BW-DOS. One wish I always had was an absolute COMTAB. It was always cumbersome to program all features relative to DOSVEC.

Next important feature was to introduce a concept for manual pages. Atari 8bit users are used to menu driven interfaces and it is not easy to switch to a command line interface. Manual pages are a big help here.

Another small point was to make it useable with 16k machines (at least the basic commands) so you can use it with unmodified 400 or 600XL.

Error handling is often a not very well regarded topic. A lot of programs and tools abort at error and write a standard error message like „Error 170“ before exit. A conversion from hexadecimal to decimal is needed for this message. This conversion was part of every program and tool as well as the message text. On the other hand exactly this text and conversion is already part of BW-DOS because the command processor returns the same message e.g., if a command was not found. To remove this redundancy I introduce the vector BW_FAIL that behaves like a jump to DOSVEC but expects the error code in the accumulator and writes the error message including finishing a running batch execution. Because of the already absolute COMTAB it is now quite easy to develop new commands for BW-DOS. BW_FAIL behaves pretty much like the symbol U_FAIL in SpartaDOS X.

In 2023 there was another great occurrence, I got mail from Jiří. Some friends had told him that his work BW-DOS is still alive and vital. He also had given his source code to the community so that everybody could see and use what is possible with an Atari 8bit. The first thing for me was to verify whether my disassembly was correct and I was happy to see, it was. I was already starting to disassemble all the tools but now I could stop it. Nevertheless I am sill working on providing Fast Assembler source for every tool and make most of them working with only 16k.

With all this new extensions to BW-DOS it was easier to port Fast Assembler from SpartaDOS X to BW-DOS. My goal was to provide a system that was able to assemble itself so it was possible to develop the system further on the system itself. Now I had BW-DOS and Fast Assembler and both could be assembled with Fast Assembler on BW-DOS. Now there was one thing missing: the Atari OS. There is the GitHub repository FastAssemblerAtariXLOS with sources of OS rev 2 for

Fast Assembler. Now its possible to assemble the Atari OS, BW-DOS, and Fast Assembler on BW-DOS using Fast Assembler. The result is a self containing system.

MEMLO is always a point. I was always unhappy that Inter-Lisp/65 was not running with BW-DOS because it requires a MEMLO below $1D80. How to keep MEMLO as low as possible? With version 1.5 there is a command processor without any internal command that provides a MEMLO below $1D80. This is a magic mark because DOS 2.5 with 4 drives and 3 possible open files has a MEMLO of $1D7C so applications usually start at a minimum of $1D80.

Jiří already wrote about the topic of PATH and DOSDRIVE to search for executables for easy access of external commands. It has become even more important now because all commands are external executables. The resident command DOSDRIVE from BW-DOS 1.3 works pretty well but needs a lot of memory. I integrated it directly in the CIO function for loading and executing executable files so it just needed 34 extra bytes. Because I could spare some bytes in other places I could manage to raise MEMLO only by 3 bytes in total. So with version 1.5, DOSDRIVE is no longer a resident program anymore but just a small program to set the DOS drive for the internal DOSDRIVE functionality.

Another project was to provide native BW-DOS tools for FujiNet. These tools can be obtained from GitHub repository FujiNetToolsSpartaDOS. This project was also an excellent collaboration with Erhard. These tools also work with SpartaDOS X and SpartaDOS 3.

A further project is to provide command line tools to access Atari DOS 2/2.5 disks. This is ongoing and the command ADIR is just the beginning.

At this point I would like to say thank you to the whole community around the world. Without your help and your feedback this would not be possible and always remember this is still a hobby.

Structure

BW-DOS is divided into three parts:

1. The resident part (the DOS itself) is loaded during the boot process and stays in memory to provide basic system functions like every other DOS. The resident part is loaded with a file with the extension DOS (XBW15CP.DOS or XBW15.DOS) and it contains two parts: the Command Processor (CP) and the File Management System (FMS). The CP is the program that takes the user input on the command line, interprets the commands with parameters, and executes them. The FMS is used by all programs (including the CP) to get access to files on the disk e.g., read and write files, get directory etc.

2. The non-resident part is divided into several small programs, called commands, to provide more functions e.g., copy files, format disk etc. These programs are loaded from disk every time they are used. These commands are stored in the DOS directory of the DOS drive. For more information see chapter Command Processor.

3. The Menu program provides easy access to file operations and functions in a menu driven way like Atari DOS. It displays the content of disks and directories in a browsable window. All operations and functions can be accessed using the menu or the key board with short cuts. The Menu program can be started from the command processor with the command MENU. For more information see chapter Menu and description of command MENU.

There are two kinds of commands for the command processor:

1. The non-resident commands (or common commands) are executed by programs loaded from disk. They are read from the DOS directory on the DOS drive which must be accessible at any time. External commands may use the user memory (memory between MEMLO and MEMTOP, see command MEM) so programs loaded before the execution of an external command must be reloaded before the next execution. In the chapter Command Processor every command is explained and it is noted if this command uses the user memory. Most other programs (like programming languages, editors etc.) may be loaded in the same way like these commands - simply by typing name of the program. If the program has another extension than COM, the complete name of the program including it extension must be used. If no path is specified with the command then the command processor searches first at the DOS drive. If the command is not found on the DOS drive then the command is searched at the current drive. The default DOS drive is 1. It can be changed with the command DOSDRIVE. For a more detailed explanation see chapter Command Processor.

2. The second kind of BW-DOS commands are resident commands. They are external commands with extension SYS. The important difference is: If you start a resident command then a part of the command stays in memory - like the resident part of the DOS - and it will provide additional functions to the system e.g., driver for a Ramdisk. Note that this will increase MEMLO and with this decrease the user memory. Most of the resident commands of BW-DOS can also be removed from memory by restarting them with parameter OFF e.g., RAMDISK.SYS OFF. You can only remove the last installed resident command. For example, if you have installed CLOCK.SYS and then RAMDISK.SYS, and you want to remove CLOCK.SYS later then you must remove RAMDISK.SYS first and afterward you can remove CLOCK.SYS. Resident commands without remove ability, can only be remove by rebooting the system without installation of them again. Residents programs are searched by the command processor in the same way like non-resident commands.

Commands may use parameters (for example filenames). You need to type parameters after the command name on the same line separated by space. A command and its parameters may be up to 64 characters long. One parameter can only be 28 characters long. If one parameter is longer than 28 characters then the remaining characters are discarded and they are not passed to the command.

Every BW-DOS command may be aborted by pressing the <ESC> or <Break> key.

WARNING: Never change a disk if there are files is open on it e.g., during execution of a batch file, during output redirection to a file on the disk, during editing a file etc. This will be result in data loss of these files. Furthermore, do not press <RESET> or turn off the computer if a file is open for write or update.

Installation

This chapter explains how to install BW-DOS on floppy disk or hard disk.

BW-DOS is delivered on four disks:

1. BW-DOS 1.5 - Bootable medium density disk with the complete BW-DOS 1.5 including all commands and tools described in chapter Commands

2. BW-DOS 1.5 Sources Kernel - Non-bootable double density and double side disk with sources of BW-DOS kernel and menu.

3. BW-DOS 1.5 Sources Commands - Non-bootable double density and double side disk with sources of BW-DOS commands.

4. BW-DOS 1.5 Single Density (no man) - Bootable single density disk with the complete BW-DOS 1.5 including all commands and tools described in chapter Commands but without manual pages.

All sources are compatible with Fast Assembler for BW-DOS and SpartaDOS X. For more information see https://github.com/HolgerJanz/FastAssembler/. The sources can also be compiled using the MADS cross assembler (see https://mads.atari8.info/) because this Assembler is compatible to Fast Assembler.

For a preliminary configuration you can start the batch file CONFIG.BAT on the BW-DOS 1.5 disk with the following command:

```
-CONFIG
```

This batch file will ask you to install a Ramdisk and clock. Chose the clock that fits your hardware setup. Questions are answered with <Y> or <N>. The batch file can be canceled at any time with key <ESC>.

If you want to read a short description for a command, e.g. FORMAT just type:

```
MAN FORMAT
```

There are four steps to install BW-DOS on a new disk:

1. Format and initialize disk
2. Copy BW-DOS
3. Make disk bootable
4. Adapt Start-Up file

Format and Initialize Disk

The command FORMAT is used to format and initialize floppy disks. Start the command FORMAT and it asks for parameter.

```
FORMAT
```

To initialize a hard disk partition use the command FORMAT with the option Create File System at the Density question. The command FDISK can be used to create and mount APT (Atari Partition Table) compatible partitions, e.g. Side1/2/3 or IDE plus 2. Make sure that the new partition is assigned to drive 3. Detailed documentation for FDISK and APT can be found here: https://atari8.co.uk/apt/documentation/

Another way to prepare a hard disk for BW-DOS is to use SpartaDOS X (see SpartaDOS X User Guide) because the file system is compatible. It is also possible to use the same partition of a hard disk for booting BW-DOS or SpartaDOS X because the names of the used configuration files are different. BW-DOS uses file STARTUP.BAT and SpartaDOS X uses files CONFIG.SYS and AUTOEXEC.BAT.

Copy BW-DOS

Directories are created and files are copied after the new disk is formatted and initialized. For the following examples assume that the BW-DOS disk is in drive 1 and the destination disk is the current drive. To make the new disk the actual drive use command D2: for drive 2 or D3: for drive 3 etc.

All commands of BW-DOS require ca. 110kB of disk space. If your destination does not provide so much free disk space, you can just copy the commands you need. There is program file (extension COM or SYS) in directory DOS and the manual file with extension MAN in directory MAN for every command. You can also skip all manual files but then command MAN will not work anymore.

Copy all BW-DOS commands and manuals:

```
COPY /R D1:>
```

The option /R of the command COPY is used for recursive copy so the subdirectories DOS and MAN including all files are also copied.

Make Disk Bootable

The file to load by the boot process is `DOS>XBW15CP.DOS`. This is the executable file to be started at boot phase. After all files have been copied to the new disk, the command BOOT can be used to make this disk bootable.

Use the following command to set the bootable DOS file:

```
BOOT >DOS>XBW15CP.DOS
```

If you want to boot the DOS version with internal command then use the following command to set the bootable DOS file:

```
BOOT >DOS>XBW15.DOS
```

Adapt Start-Up File

The last step is to adapt the Start-Up file STARTUP.BAT for your needs.

Use the command ED to change the start-up file:

```
ED STARTUP.BAT
```

If you want to switch off the key click and install a driver for the clock of the IDE+2 interface then STARTUP.BAT file could like this:

```
POKE 02DB FF
IDE2PTD.SYS
```

If you want to switch off the key click and SIO noise, install the software clock, and set date and time then the STARTUP.BAT file could look like this:

```
POKE 02DB FF 0041 00
CLOCK.SYS ON
DATE
TIME
```

The statements DATE and TIME ask for new date and time and set the installed clock.

If you like to work with the Menu (see chapter Menu) then you can automatically start it if the command MENU is the last command in your STARTUP.BAT file.

Boot from Cartridge

It is possible to boot BW-DOS 1.5 from a cartridge that supports executables e.g., The!Cart, AVG, Uno etc. Two executables are provided for this:

```
XBW15CP.DOS
XBW15.DOS
```

For the difference of this two DOS files see chapter Command Processor.
After starting BW-DOS 1.5 from cartridge the STARTUP.BAT file is read from drive 1.

Disks and Directories

Every disk under BW-DOS has an 8 character name, the volume name, and a main directory. There can be an almost unlimited number of directories. Directories can be nested.

A directory within another directory is called subdirectory. The directory containing other directories is called parent directory of these directories.

A directory can contain up to 1424 files or subdirectories. It is recommended to keep this number of files and subdirectories less than 100 per directory because the access to long directories is slow. Only SpartaDOS X (version 4.x) is also able to use BW-DOS directories in full - other SpartaDOS 3 compatible DOS versions only work with the first 126 files of a directory, the remaining files are ignored.

The working directory is the directory, where files are searched if no path was specified, and it is the starting point for relative paths. For each disk drive there is a current working directory. They are independent of other drives. The current directory is set to the main directory of a drive if the disk was change and at warm and cold start of the system.

You can change the current drive number to n by typing Dn:<RETURN>. The current drive number is used if no drive was specified for a command or file. Note that this only works in the command processor. If the CIO functions are used e.g., from BASIC, then the drive must always be specified. If only D: is specified for a file or command then the drive 1 is always be used.

A path is used to specify to subdirectory a file or other subdirectory is located. A Path is a list of subdirectories, starting from current directory to the target directory. Subdirectory names and i the filename at the end of the path are separated by the > character. There are two kinds of paths: absolute paths and relative paths. Absolute paths start with a single ">" character. They always start in the main directory of a drive. Relative paths does not start with >. They always start at the current working directory. In addition, at the beginning of relative paths there can be any number of < characters - each one indicates to go up to the next parent directory.

Some path examples:

The file TEXT1.TXT is located in directory TEXTS, which is a subdirectory of DATA in the main directory.

```
>DATA>TEXTS>TEXT1.TXT
```

The file SONG.TXT is in subdirectory MUSIC, which is a subdirectory of the current working directory.

```
MUSIC>SONG.TXT
```

The file GAME.BAS is in subdirectory BASIC, which is ia subdirectory of the parent directory of the current working directory.

```
<BASIC>GAME.BAS
```

Subdirectory DOS in the main directory is a special directory. Every file, that cannot be found in the current working directory, is also searched in the DOS directory but only if no path or drive was specified or the file.

It is recommended to place external commands, the DOS itself, and other frequently used read-only files in the DOS directory to provide simple access.

Besides of this, it is a very good idea to create separate directories for different file types, such as Basic programs, texts etc. If all files are stored in the main directory then the main directory becomes very large and access to the files will slow down.

Batch Files and Hard Copy

Batch files are input redirection. Commands are read from a file by the command processor instead of the command line. Hard copy is output redirection, the screen output is also send to a different device e.g., printer or file.

Batch files are text files that contains commands for the command processor, one command per line in the text file. The standard extension for batch files is BAT. A batch file is started by typing the - character in front of the name e.g., -MY_BATCH must be typed to start the batch file MY_BATCH.BAT. If a batch file is started, the command processor reads the commands line by line from the batch file and executes the commands or programs.

You can use batch files in two ways:

1. If several long running programs should be executed in a sequence then the list of this programs must be stored into a batch file. If the batch file ist started then the list of programs is automatically executed.

2. Simple tasks can be executed using a batch file. For example, to install a Ramdisk and copy a set of external commands to it or to install a clock driver and set the current date and time with just one command.

A batch file named STARTUP.BAT in main directory or in the directory DOS of the boot drive is executed every time the system is booted. The batch file STARTUP.BAT can be used to place commands for system configuration that are executed every time the system is booted. BW-DOS does not support the CONFIG.SYS file known from SpartaDOS X.

For example, if the system should be setup in the following way: install Ramdisk at drive 8, turn off the internal Basic, change current directory to subdirectory ATMAS in main directory, and start the ATMAS II macro assembler from main directory then the STARTUP.BAT file could look like this:

```
RAMDISK.SYS 8
BASIC OFF
CD >ATMAS>
>ATMAS
```

If any error occurs at command execution then the command processor stops the processing of the batch file to prevent further problems. The current batch file is aborted if another batch file is started from it.

Batch files can contain comment lines. Such a line must start with the ; character. These lines are not executed but just printed to the screen.

In contrast to SpartaDOS, BW-DOS has no XDIV command. That is because the HATABS table is in its original state if no batch file or hard copy function is active. This means programs can be directly started that otherwise needs the XDIV command in SpartaDOS.

Batch files and the Hard copy function work as follows: A small routine is inserted between CIO and the handler of "E:" device by changing the HATABS ($31A) table This routine checks all editor input/output, if is it going through IOCB 0. In that case, another handler will be used to get data from a batch file, or to send data to the hard copy device.

The hard copy device can be set with the PRINT command with the destination device or file as parameter. If the PRINT command is called without parameter then the output redirection is stopped.

Batch files are accessed through IOCB 5 by the command processor and the hard copy function uses IOCB 4. These IOCBs are modified to look like a closed IOCB, so a CLOSE command done by many programs, for example Basic, may not cause any issues.

With redirection it is possible to start a Basic program and save the output to a file. The next section describes a small example.

The Basic program ADD.BAS requests to numbers X and Y for input and outputs the sum of X and Y.

The batch file ADD.BAT executes the following steps:

1. redirects the standard output to file ADD.TRC
2. starts the cartridge (make sure that Atari Basic is active)
3. loads the Basic program ADD.BAS
4. runs the Basic program with numbers 11 and 31, and 17 and 25
5. returns to DOS
6. stops the the redirection of the standard output.

The Basic program that controls the Basic program is saved as ADD.BAS:

```
10 PRINT "X=";:INPUT X
20 PRINT "Y=";:INPUT Y
30 PRINT "X+Y=";X+Y
```

Next, the batch file is saved as BASIC.BAT in the same directory like ADD.BAS:

```
PRINT D:ADD.TRC
CAR
LOAD "D:ADD.BAS"
RUN
11
31
RUN
17
25
DOS
PRINT
```

Now the batch file ADD.BAT can be started with:

```
-ADD
```

The result of the execution of the batch file ADD.BAT is a text file with the name ADD.TRC with the following content:

```
D1:CAR

READY
LOAD "D:ADD.BAS"

READY
RUN
X=?11
Y=?31
X+Y=42

READY
RUN
X=?17
Y=?25
X+Y=42

READY
DOS

D1:PRINT
```

It is possible the redirect the input and output of every program that uses the standard input and output via IOCB 0 on device E: of the operation system. For more information about Basic and the usage of IOCBs see chapter Functions in Basic.

Date and Time

Every file and directory has the date and time in BW-DOS. It is the date and time when a file or directory was created or changed. Date and time are read from the BW-DOS clock every time a file or directory is changed or created.

If no clock driver was installed then the date and time set by BW-DOS is constant. This means, that every file or directory gets the same date and time until date and/or time are manually changed e.g., with the commands DATE, TIME, or APETIME.

An appropriate clock driver must be installed if BW-DOS should use an installed real time clock. Usually resident commands are delivered by the provider of a real time clock for this purpose.

CLOCK.SYS is a software clock for BW-DOS. Date and time must be manually set at every startup of the system with the DATE and TIME command. The software clock is based on the VBI interrupt. It is possible that some programs will stop it. It works on both PAL and NTSC machines.

BW-DOS provides drivers for some real time clocks:

IDE plus 2	`IDEP2TD.SYS`
RTime 8	`RTIME8.SYS`

There is the command APETIME that reads time and date via SIO commands from all APE (Atari Peripheral Emulator) compatible devices e.g., SIO2USB and FujiNet. This command can also be used to set the date and time for the CLOCK.SYS driver. The command sequence to install the software clock and set date and time using APETIME could look like this:

```
CLOCK.SYS ON
APETIME
```

The command GETTD can be used to output current date and time.

All clock driver designed for SpartaDOS 3 also should work with BW-DOS e.g., clock driver for the Ultimate 1MB and the Side2/3 cartridge can be obtained from the APT Hard Disk Toolkit by Jonathan Halliday (https://atari8.co.uk/apt/toolkit/).

Menu

The menu command is a comfortable tool to work with many files e.g., to copy files between different disks with only one disk drive, or to copy files from an Atari DOS 2 disk to a BW-DOS disk or vice versa.

Every command in the menu - except the ones for subdirectories - works on BW-DOS and Atari DOS 2 compatible disks. The Atari DOS filesystem format is only supported by the menu program. It is not available by any other command of BW-DOS. The following Atari DOS 2 compatible formats are supported: single density (DOS 2.0, DOS 2.5, BiboDOS, and others), enhanced (medium) density (DOS 2.5), and double density (BiboDOS, and others). Note that Atari DOS 2 alike file systems have different restrictions for file names e.g., no _ (underscore) characters are allowed and the first character must be a letter. These restrictions must be fulfilled if files are copied from BW-DOS disks to Atari DOS disks.

The menu displays the contents of the current drive on the screen. It shows the same information like the command DIR. There is no date and time displayed for Atari DOS disks and the length of files is not exact. Above the directory there is the current drive number and path displayed. A little arrow on the top or bottom edge of the directory indicates that the directory continues above or below the screen. The menu program can work with a maximum of 255 files per directory.

The cursor is on the left hand side. It can be moved with the <UP> and <DOWN> keys. Files can be selected with the key <SPACE>. Selected files are shown in inverse. If you select a command, it is executed for every selected file. If no file is selected then the command is executed for the file pointed by the cursor.

At the bottom of your screen there is a command menu with another cursor. The command cursor can be moved with the keys <LEFT> and <RIGHT>. Press <RETURN> to execute the selected command. There are hotkeys for every command to quickly select a command. The letter of the hotkey for a command is displayed inverse to give you a hint.

There is an input line between directory listing and the command menu. Here the menu asks for additional parameter. You can use the <LEFT> and <RIGHT> key to navigate in the line. Press <RETURN> to confirm the input. The keys <DELETE>, <CONTROL>+<left>, <CONTROL>+<right>, <CONTROL>+<DELETE>, <CONTROL>+<INSERT>, and <RETURN> can be used to edit the line. In addition <CONTROL>+<CLEAR> can be used to clear the complete line.

The menu supports the following command:

`ChangeDrv <V>`

> This command is used to change the current drive number. The working directory of selected drive is displayed. Use this function if you want to work with another drive. In case the disk in the drive was changed the main directory is displayed. The current path of the working directory is saved for every drive and is used as a default path e.g., to copy files to the corresponding drive.

`CheckDisk <K>`

> This command displays information about a disk drive. It is the same information like command CHKDSK in the command processor.

`ChangeDir <H>`

> This command changes the current path of a drive to the selected subdirectory. The cursor of the directory listing must point to a subdirectory.

`ParentDir <I>`

This command changes the current path to the parent directory. The cursor is set to the subdirectory in the parent directory listing from where the command was executed.

`MakeDir <M>`

This command create new subdirectories.

`RemoveDir <R>`

This command deletes a subdirectory. It is only possible if the subdirectory is empty.

`Copy <C>`

This command copies the selected files. It asks for destination drive and path. The destination path can be up to 231 characters long. The input line scrolls if necessary.

If the same drive number is selected for source and destination then the command ask whether the same disk should be used or the disk should be changed for the destination. If the answer is No (destination should not be the same disk) then the menu program asks for changing source and destination disks. This makes it possible to copy from one disk to another with only one disk drive. If source and destination disk are the same, use different paths.

If a file with the same name already exists in the target directory then an option menu appears. It is possible to overwrite the destination file, to skip the file, or to abort the copy command. This feature can be turned off with the Setup command.

`Delete <D>`

This command deletes the selected files. Protected files cannot be deleted.

`Rename <N>`

This command renames the selected files. Multiple files can be renamed with one command. See description the REN command.

`Protect <P>, Unprotect <U>`

These commands protect or unprotect files or directories. A protected file cannot be changed or deleted.

`TypeFile <T>`

This command copies a file to the screen that is pointed by cursor. It is an easy way to read small text files.

`NewFile <F>`

This command creates a new text file. The input from the editor E: is copied to the file after <CONTROL>+<3> is pressed. This command provides the same functionality like `COPY E: file` in the command processor.

`AtariIni <A>`

This command formats a disk in a Atari DOS compatible format. It asks for drive number and density. After the last confirmation the disk is formatted. BW-DOS commands cannot handle disks in Atari DOS format, only the menu can read and write such disks.

Warning: Formatting a disk deletes all files even if the files or directories are protected.

Exit `<X>`

This command exits the menu and returns to the command processor.

Setup <S>

This command sets up some properties of the menu program:

1. Specifies the order in which files are sorted, e.g. by extension and name, by name and extension, by length, by date and time, or the same order as on the disk (default). The last option is useful if the directory is already sorted and it saves some seconds if the directory is very large.
2. Turns on (default) or off the bell if an error or other important messages occur.
3. Turns on (default) or off the menu selection before overwriting a file, see command Copy.

All the selections can also be set from the command processor using parameters, see description the command MENU in the chapter Commands.

`Select <L>, Deselect <E>`

These commands select or deselect files specified by a filter using the wildcard characters * and ?. You can select an arbitrary set of files using a sequence of Select and Deselect commands.

Command Processors

The command processor is the program that reads the user input from the command line and executes the corresponding commands and programs. The command processor is automatically load at boot time with DOS itself.

Commands are programs that come with BW-DOS. Depending on the DOS version and the command itself, a command is either built into DOS or comes as an external program. External commands are stored in the DOS directory of the DOS drive. The DOS drive is usually the boot drive 1. The DOS drive can be changed with the command DOSDRIVE e.g., if a Ramdisk is used and all commands are copied to it.

Example of a command (all commands are executed by pressing <RETURN>):

```
DIR D1:
```

This command line will list all files of drive 1 in the current directory.

When a command is specified with drive or path then the command is only searched at the specified drive or path. If no drive or path is specified with the command then the command is searched in the following directories in that order:

1. current directory of DOS drive
2. DOS directory of DOS drive
3. current directory of current drive
4. DOS directory of current drive

If the DOS drive is switched off or the current drive is also the DOS drive then the steps 1 and 2 are skipped.

A command line starts with the name of the command or program and can be followed by a list of parameters separated by spaces. The maximum length of a command line is 64 characters. The maximum length of one parameter is 28 characters.

The differences of the two command processors XBW15CP.DOS and XBW15.DOS are described in the next two chapters. It is possible to switch between these two command processors just by calling XBW15CP.DOS and XBW15.DOS from the command line. If a new command processor is called then DOS is reinitialized and the start-up batch file STARTUP.BAT is processed. Both command processors require at least 16k of main memory and a MEMLO below $2200 if they are started from the command line or cartridge.

XBW15CP.DOS - Command Processor

The advantage of this command processor is the low MEMLO. MEMLO contains the address of the first byte that can be used by a user programs and that is not used in any way by DOS and the command processor. If the command processor XBW15CP.DOS is booted without any other resident programs, then the MEMLO is $1D65. Even if a resident command like the clock driver IDEP2TD.SYS installed in read-only mode, MEMLO is still below $1D80.

A low MEMLO is important if you want to start some programs which were designed for Atari DOS. One example is Inter-Lisp/65. This program requires the memory starting from $1D80. This is just above the MEMLO that is provided by Atari DOS 2 and 2.5 configured for four drives and three open files at a time.

The current MEMLO can be determined with the command MEM. If you want to check what memory a certain program is loaded to then you can use the command OFFLOAD. The command OFFLOAD shows all memory segments of a program. If one segment starts below current MEMLO then you cannot start this program because it properly is going to crash.

This a list of different DOS versions and there startup MEMLO:

BW-DOS 15 CP	$1D65
BW-DOS 15	$1EFE
BW-DOS 14	$1ECF
BW-DOS 13	$1EE4
Atari DOS 2	$1D7C
Atari DOS 2.5	$1D7C
SpartaDOS 3.2g	$17A2
SpartaDOS 3.2f	$1819
SpartaDOS 3.3a	$1900
MyDOS 4.53/4	$1EE8
MyDOS 4.55 Beta 4	$1F9F

The disadvantage of the command processor XBW15CP.DOS is that it contains no internal commands. This means that all commands have to be loaded from disk and must always be accessible on the DOS drive or the current drive (see command DOSDRIVE).

XBW15.DOS - Command Processor with Internal Commands

The advantage of the command processor XBW15.DOS is that the basic commands are integrated. These commands are called internal commands. These internal commands do not have to be loaded from disk every time they are executed.

The following internal commands are integrated in the command processor XBW15.DOS:

BASIC	Switch Internal Basic
CAR	Execute Cartridge
CD	Change Directory
DEL	Delete Files
DIR	List Directory
DIRS	Short List Directory
LOAD	Load Executable File
MD	Make Directory
PRINT	Redirect Console Output
PROT	Protect File
RD	Remove Directory
REN	Rename File
RUN	Run Address
TYPE	Display File
UNPROT	Unprotect File

The disadvantage is that MEMLO is higher ($1EFE) than the command processor XBW15CP.DOS provides. Some programs that run under Atari DOS cannot be executed with this command processor and will most likely crash. The current MEMLO can be determined using the command MEM and the command OFFLOAD can be use to check the addresses a program is loaded to. If a program is loaded below the address contained in MEMLO then this program will crash.

Furthermore the internal versions of commands sometimes have fewer options and less parameter checks e.g., see command DIR, only the external version supports parameter lists.

Commands

Every command is explained in detail in this chapter. Every explanation starts with the structure of the command. The following notation is used:

- `[parameter]` - This means that a parameter can be used or skipped. It is called an optional parameter.
- `(ON|OFF)` - Either ON or OFF must be provided but not both.
- `name` - Name of a file or directory. Names can be up to 8 characters long, and may contain upper case letters, numbers, and the "_" character. Any invalid character terminates the specification of a file.
- Wildcards are supported: a "*" means that the rest of name may contain any character, and "?" means exactly one arbitrary character. Wildcards cannot be used to create a new file or directory. Commands that only work with one file select the first matching file if wildcards are used e.g., the command TYPE only shows one file, if TYPE `*.TXT` is specified then the first file in the current directory that matches *.TXT is shown.
- `ext` - The extension part of a filename. It may be up to three characters long. Usually it identifies type of a file. Standard extensions are for example: COM for executable programs, BAT for batch files, BAS for a Basic programs, TXT for text files, DOC for a documentation files (usually also text file), PIC for a picture. It is recommended to use no extensions for directories.
- `Dn:` - The specification of a disk drive, where n is the number of the drive. If no drive number is specified then the default drive number 1 is used e.g., just D: refers to drive 1.
- `device:` - The specification of a device that doesn't have to be a disk drive. For example "E:" for screen editor, "C:" for tape recorder, "P:" for printer etc.
- `filename` - The complete name of a file - name including an optional extension. File names containing wildcards are called patterns e.g., `DIR *.TXT` filters all files with extension TXT.
- `path` - The specification of a directory. There are two kinds of paths - Absolute paths (always start with >): ">[name][>name]...", and relative paths: "[<][<]...[name][>name]...". See chapter Files and Directories for more details.
- `file` - The full specification of a file including an optional path. The complete syntax is: `"[Dn:][path>]filename"`.
- `pattern` - The specification of a pattern (filter) for file names that can contain wildcards * and ?.
- All numbers are entered as hexadecimal numbers without leading $ character. File positions are up to 6 hexadecimals digits long. Memory addresses and sector numbers are up to 4 hexadecimals digits long.
- .. - This means the parameter before .. can be repeated several times.

Some of the commands described in this chapter are marked as Low Memory. This means that the commands are loaded into the memory area from $480 to $6FF. These commands do not use the main memory the memory area from MEMLO to MEMHI (these area is usually used by application programs). This allows you to work with data or other programs in the main memory without changing the main memory.

Commands that need more than 16k are marked with Requires 48k. Commands that need the RAM under the OS ROM are marked with Requires 64k.

Some of the commands are marked as Internal. These are commands that are integrated in the command processor XBW15.DOS in contrast to command processor XBW15CP.DOS that does not contain any internal commands. If there are differences between the internal version and the executable command then these are described in the command description (see chapters below). The external commands can also be started from XBW15.DOS using a path specification e.g.:

```
DOS>DIR D8:
```

or full qualified path:

```
D1:>DOS>DIR D8:
```

Where as the following command starts the internal version if command processor XBW15.DOS is used:

```
DIR D8:
```

The following sections describe every command in detail.

ADIR - List Atari DOS Directory

```
ADIR [Dn:[sub_1:..:sub_n:]pattern]
```

This program lists the directory of Atari DOS, MyDOS, and compatible disks. BW-DOS itself does not support other files systems than the SpartaDOS file system. This command does not check for a valid Atari or MyDOS file system to list directories also for non-compliant Atari/MyDOS file systems e.g., copy protection. If there is no valid Atari DOS file system then the command lists garbage.

Besides the drive a path can also be specified for which the directory should be listed. The path must be specified in MyDOS notation. That means the subdirectory separator is : instead of > in SpartaDOS. The pattern may contain the wildcards * and ? with the same meaning as in SpartaDOS.

Every directory entry is listed with name (maximum of 8 characters), extension (maximum 3 characters), and size of the file in numbers of sectors (maximum 65535). Subdirectories are always displayed with the size of 8 sectors. Two directory entries are listed per line. Furthermore there are the following mark characters at the beginning of a directory entry:

: marks subdirectories

* marks protected files and subdirectories

The last line of a directory listing shows the number of free and total sectors of the disk (maximum 65535) and the size of a sector in bytes. Disks with sector size of 128 bytes (single density) and 256 bytes (double density) are supported by ADIR.

Example output:

```
 :TOOLS        00008  :ASM          00008
 :ATM          00008  :BAS          00008
 :ATW          00008 *:GAMES        00008
  TEXT    TXT 00042 * READ    ME   00001
62404 free, 65491 total, 256bps
```

If there are any errors at accessing the disk then same error numbers are reported as for the SpartaDOS file system.

APETIME - Get Time and Date form APE

```
APETIME
```

Low Memory

This command reads the SIO clock of an APE compatible SIO device and sets date and time of the BW-DOS clock. This command does not install a resident driver. It works with all SIO clocks compatible with the APE protocol like SIO2PC and FujiNet. It can be used together with the resident command CLOCK.SYS. First install CLOCK.SYS and then use APETIME to set BW-DOS clock. It is possible to install any drive for a hardware clock e.g. IDE2PTD.SYS and use APETIME to set the hardware clock e.g. hardware clock of IDE plus 2.

APETIME can be used without a clock driver but it must be called explicitly if the date and time of BW-DOS should be updated.

The command prints the version number and the current time and date that have been read. If there is no compatible APE clock connected to the SIO then usually the error 138 (Device does not respond) is returned.

Note: If you are using a SIO2PC device then RespeQt R3 and newer support APETIME whereas AspeQt versions 0.8.x and higher do no support APETIME any longer. It is compatible with the APE time implementation of the IDE plus 2 interface.

BASIC - Switch Internal Basic

```
BASIC ON|OFF
```

Low Memory, Internal

This internal command switches the internal Basic (XL/XE only) on or off. With Basic off you have more free memory, several programs works in this mode only. This command works only if no other cartridge is installed. User memory is always erased by this program e.g., current Basic program is lost.

If there is no internal Basic installed e.g., on an ATARI 800, or another cartridge is installed then no error is returned.

Whether the internal Basic is active or another cartridge is installed can be check with the command MEM and the value of RAMSIZ. If the value is below $C000 on machine with at least 64k then the internal Basic is active or a cartridge is installed. The value of RAMSIZ is $A000 for internal Basic and 8k cartridge or value $8000 for 16k cartridges.

BLOAD - Load Binary File

```
BLOAD file address [length]
```

Low Memory

This external command loads data files (non DOS executable files) into memory. If no length is specified, then the whole file is loaded. The content of the file is loaded at the given address.

Address and length are specified in hexadecimal numbers without preceding $.

The file is loaded as a raw data block to the specified address and control is handed back to the command processor. There are no checks whether the file fits into memory or OS or DOS memory areas are overwritten.

BOOT - Specify Boot File

```
BOOT file
```

With this command you can specify a file to be loaded when the system is booted up (when you turn the computer on). Usually, it is a DOS file but other files may also be booted - see the conditions below.

The common use of this command is installing the DOS on a disk that was previously formatted with no DOS, or with an older version. To install a DOS on a disk, you should copy the X*.DOS file in the DOS directory and then use the BOOT command to install it for booting.

The boot process is done by a little loader program that is stored in sectors 1 to 3 on every BW-DOS (or SpartaDOS 2.x and later) disks. The file which is booted must have the DOS loadable format and it must contain a run address.

If there are not at least two bytes between the end of file and the end of physical data sector, that is in Basic:

```
IF 128*INT((LENGTH+127)/128)-LENGTH<2
```

then two zero bytes must be added as an end of file mark.

The BOOT command simply writes the number of the first sector map of the file into a specific location of sector 1 of the disk. See chapter Disk Format for more information.

The booted file may not be loaded at the memory area from $2E00 to $317F, where loader itself resides. Any executable file that does not load to this memory area can be loaded (booted) and specified with the command BOOT. The file may reside anywhere on the disk - even in a subdirectories.

Note that the loader is little different from the one found on SpartaDOS disks. If you are booting from a SpartaDOS disk, then segments in the file - excepting the first one - must not begin with the $FFFF header. Besides of this, the SpartaDOS loader is not fully compatible with the XF 551 disk drive.

If you delete the file specified to be booted, an attempt to boot such a disk gives unpredictable results, mostly the computer will just crash.

CAR - Execute Cartridge

```
CAR
```

Low Memory, Internal

This command hands over control to the active internal Basic (see command BASIC) or an installed cartridge. If the internal Basic is not active and no other cartridge is installed then the error 156 (Cartridge error) is returned by the command CAR.

If no external commands have been called since the cartridge was left, the user memory and the current Basic program are not lost. If you use the command processor XBW15CP.DOS then leaving and reentering Basic always destroys user memory and the current Basic program.

BW-DOS does not support any files to save and restore the user memory for cartridges.

CD - Change Directory

```
CD [Dn:]path ..
```

Low Memory, Internal

This command sets the directory specified by path as current working directory of the current drive or the drive specified with Dn:.

Note: The parameter buffer has only the size of 28 characters including drive specification and end of line. This means that only 24 character are left for the path without a drive specification. If the path is longer then it is cut and the command CD returns with error 150 (Directory not found). This limitation can be bypassed using multiple parameters, see examples below. This does not work with the internal version of the command CD.

The operator > can be used multiple times in a path. Both operators < and > can be used in combination in a path.

Given the following directory structure and the current directory is `MAIN`:

```
MAIN
  SUBDIR1
    SUBDIR2
      SUBDIR3
        SUBDIR4
    SUBDIR2A
      SUBDIR3A
```

The following command sets current directory to `MAIN>SUBDIR1>SUBDIR2`:

```
CD SUBDIR1>SUBDIR2
```

then the following command sets current directory to `MAIN>SUBDIR1>SUBDIR2A>SUBDIR3A`:

```
CD <SUBDIR2A>SUBDIR3A
```

then the following commands set the current directory to `MAIN>SUBDIR1`:

```
CD <<
```

The following command always sets the current directory to the `MAIN` directory independent of the current directory:

```
CD >
```

The internal version of CD does not evaluate multiple parameters. Only the first parameter is evaluated by the internal version of CD.

Multiple parameters can be used to set the working directory for different drives with one execution or to set the path for one drive in multiple steps to circumvent the limitation of the short parameter buffer.

The command below returns with error 150 (Directory not found) because the path is too long for one parameter:

```
CD SUBDIR1>SUBDIR2>SUBDIR3>SUBDIR4
```

It is possible to overcome this limitation by using multiple parameters:

```
CD SUBDIR1>SUBDIR2 SUBDIR3>SUBDIR4
```

It is even possible to use one parameter per subdirectory:

```
CD SUBDIR1 SUBDIR2 SUBDIR3 SUBDIR4
```

The two statements above are a equivalent for multiple executions of CD. This is the only solution for the internal version of the command CD:

```
CD SUBDIR1>SUBDIR2
CD SUBDIR3>SUBDIR4
```

or if only one parameter per subdirectory is used then the equivalent is:

```
CD SUBDIR1
CD SUBDIR2
CD SUBDIR3
CD SUBDIR4
```

There is also a limitation of 64 characters per command line. If this limit is reached then the error 137 (Truncated record) is returned by the command processor.

CHKDSK - Check Disk

```
CHKDSK [Dn:] ..
```

This external command displays information about disks. If no drive was specified then the information of the current drive are displayed.

The information contains the volume name, two internal hexadecimals 8bit numbers, the version number of the SpartaDOS file system, the size of sectors, the total disk capacity, and the free capacity in bytes.

The two hexadecimal 8bit numbers following the volume name are used for detection of disk changes in case there are disks with the same volume name.

The output looks like this:

```
      Volume: BWDOS15  53 FF
     Version: 2.0
Sector size: 128 bytes
   Capacity:    92160 bytes
 Free space:    67200 bytes
```

Note: The write lock status of disks known from SpartaDOS is not supported by BW-DOS. If a write lock is set with SpartaDOS then this is not protecting from changing or formatting these disks with BW-DOS.

A list of parameters can be used to get disk information for more than one device.

CHVOL - Change Volume Name

```
CHVOL [Dn:]name
```

Low Memory

This external command changes the volume name of disk n. If no drive was specified then name of the current drive is changed.

A maximum of eight characters is allowed for volume names. The volume name may contain any ATASCII characters including inverse characters.

This command returns error 148 (Invalid disk format) if the disk has no SpartaDOS format.

CLS - Clear Screen

```
CLS
```

Low Memory

This command clears the screen by sending the Clear Screen control character (125,$7D) to standard output.

This command can be used in batch files to improve screen output or in the command processor instead of the key combination <SHIFT>-<(Clear).

COLD - Cold Restart

COLD

This external command executes a cold start. The cold start is executed by jumping to vector COLDSV ($E477).

The main difference between the command COLD and the command RUN E477 (see command RUN) or a cold start via hot key (like IDE+2 and Ultimate 1MB provide) is that the hard copy and the batch processing are consistently stopped. This ensures that the file system is not corrupted and no data is lost by the cold start.

COMP - Compare Files

```
COMP [/REQ] files1 files2
```

This command compares files.

If two files are different then the offset of the first byte that differs is printed. This offset can be used with the command DUMP for further analysis.

If the two files have different lengths and the smaller file is equal to the beginning part of the larger file then the offset of the first byte after the beginning part is printed. If a file from files1 is missing in files2 then the message „file does not exist“ is printed instead an offset.

Files are recursively compared if the option /R is set. Additional files in the second specified directory (files2) are ignored.

If option /E is set then the program terminates with error 255 at the first two different files. This option can be used to terminate batch execution if a difference occurs.

With option /Q (quite) only messages about different files are shown.

Examples:

```
COMP /R D1:> D2:>
```

Compares all files of disk 1 with files of disk 2.

```
COMP /E MAKE>*.OBJ BIN>*.COM
```

Compares all OBJ files in directory MAKE with COM files in directory BIN. The command terminates at the first two different files.

COPY - Copy Data

```
COPY [/RQ] [dev1:][path1>][patt1] [dev2:][path2>][patt2[/A]]
```

This command copies data from device dev1 to device dev2. If a device is not specified then the current disk drive is used.

It copies file(s) specified by the first parameter to file(s) specified by the second parameter

You can use any device instead of Dn: e.g.:

```
COPY K: P:.
```

This copies the keyboard input directly to the printer so there will be no screen output during typing. This copy command must be terminated with <CONROL>+<3>.

You can use wildcards in both patterns to copy a set of files and to rename them at destination (see REN command). If no file name is specified for destination then all files in the source directory are copied (first parameter), or the files at destination get their original names (second parameter). To copy a file to itself destroys the file!

Examples:

```
COPY >DOS> D8:
```

Copies the whole directory DOS of the current disk drive to disk drive 8.

```
COPY E: NEW.TXT
```

Create a new text file NEW.TXT in the current directory. Press <CONTROL>+<3> to finish the copy process.

```
COPY NEW.TXT P:
```

Send file NEW.TXT to the printer.

It is also possible to rename files at copying using patterns.

```
COPY *.TXT D2>MY_DOC>*.DOC
```

Copies all files from current directory of the current drive with extension TXT to directory MY_DOC of drive 2 and changes the extension to DOC.

The option /R recursively copies all files, that match the pattern pattern1, also from all subdirectories. Subdirectories are automatically created if they do not already exist on the destination drive. This option is only allowed if the source and the destination device are disk drives. With this option it is possible to copy all files of a disk drive to another disk drive e.g.:

```
COPY /R D1: D2:
```

The example above copies all files from disk drive D1: to disk drive D2: including the subdirectory hierarchy.

Option /Q (quite) suppresses messages for every file copied.

CUT - Cut File

```
CUT file1 file2 position [length]
```

This command saves a specified part of file1 in file2. If no length is specified, it saves the maximum length - number of bytes between position and the end of file1.

Position and length must be hexadecimals numbers without proceeding $.

It is possible - for example - to divide a DOS loadable files into single segments using OFFLOAD and CUT. The offset and length of the different segments of a file can be determined with the command OFFLOAD and a sequence of command CUT to divide this file.

```
CUT file1 file2 6
```

The example above cuts off the binary file header (6bytes, executable identifier $FFFF, and two addresses) from DOS loadable file1 and saves it in file2.

DATE - Set Date

```
DATE
```

This external command displays the current date from the BW-DOS clock, and allows you to enter a new value. Press <RETURN> without any input to leave it unchanged and exit the command.

The date format is always day, month, and year with two digits separated by dashes e.g., 31-12-24 (DD-MM-YY).

If a clock driver is used then it depends on that driver whether the change of date is supported. Some driver have an option to omit the handler to change the date to save memory and keep MEMLO lower. Usually no error is returned (just the date is not changed) if the driver does not support change of date. Refer to the description of the respective clock driver.

The command DATE can also be used without clock driver. It will just set the date of BW-DOS but no running clock will advance the date.

DEL - Delete Files

```
DEL [/RQ] [Dn:][path>][pattern] ..
```

Internal

This internal command deletes files. If wildcards are used then several files can be deleted with one command. Supported wildcards are * for arbitrary list of character and ? for exactly on arbitrary character.

Protected files are invisible for this command (see commands PROT and UNPROT). Error 170 (File not found) is returned if a protected file should be deleted.

The option /R recursively deletes files including subdirectories. This option is not supported by the internal version. Protected subdirectories are ignored.

The internal version also does not support a parameter list. Only the first parameter is evaluated by the internal version.

A parameter list can be used to delete files defined by several patterns or from several drives or paths with only one command.

```
DEL *.BAT *.SYS
```

The command above deletes all files with the extensions BAT and SYS from the current directory.

```
DEL D2:>*.OBJ D3:>*.OBJ
```

The command above deletes all files with extension OBJ from the main directory of drive 2 and drive 3.

```
DEL /R TMP>
```

The command above deletes all files and subdirectories of subdirectory TMP that are not protected.

```
DEL *.COM /R MAKE>*.OBJ
```

The command above combines non-recursive and recursive delete. First all files with the extensions COM are non-recursively deleted from the current directory and subsequently all files with extension OBJ are recursively deleted.

Option /Q (quite) suppresses messages for every directory deleted.

DIR - List Directory

```
DIR [Dn:][path>][pattern] ..
```

Low Memory, Internal

This command list the content of the specified directory. If no drive or path are specified then the current drive or path are used. If no filename is specified then every files is listed (equals *.*). The listing starts with the volume name of the disk, and the name of the listed directory. It shows for each file: filename, length, and date/time. Directories are indicated by <DIR> instead of length. At the end of listing, number of free sectors on the disk are displayed.

Example of a listing:

```
Volume: TESTS
Dir:    MAIN

DOS          <DIR>  11-06-94 16:01
TEST1    BAS   1231 11-06-94 16:32
   573 FREE SECTORS
```

The internal version of this command does not support a parameter list. Only the first parameter is evaluated by the internal version.

A parameter list can be used to list files with different patterns or from different paths or drives.

```
DIR *.BAT *.SYS
```

The command above lists all files with extensions BAT and SYS from the current directory. The result could look like this:

```
Volume: BW-DOS
Dir:    MAIN

AUTOEXEC BAT     43 24-08-22 20:05
CONFIG   BAT    372 30-12-23 15:05
STARTUP  BAT     14 15-01-24 08:05
 58026 free sectors

Volume: BW-DOS
Dir:    MAIN

CONFIG   SYS    311 05-06-23 20:10
 58026 free sectors
```

The directory header is slightly different to the header of SpartaDOS. The volume and directory name are shifted three characters to the left to support and display directory names with extensions. The size of the header is still 42 characters so programs analyzing the directory list will work without any adaption. Only programs that analyze the header have to be adapted for BW-DOS 1.5.

DIRC - List Directory Compact

```
DIRC [Dn:][path>][pattern] ..
```

Low Memory

This command like DIR lists the content of a directory but in a very compact format.

All directory entries are 12 characters long (one character for an indicator and 11 for the name with extension) and they are separated by spaces. Lines are automatically adapted to the screen width. There are three columns on a 40 character per line screen and six columns on a screen with the width of 80 characters.

There are three indicator characters:

```
* protected files
: subdirectories
# protected subdirectories
```

An example output could look like this:

```
:BIN          :DEMOS        :DOS
:FN           :MAE          :MAN
:PRG          :TMP          #TOOLS
 AUTOEXECBAT   BASIC    SAV   CAR      SAV
 CONFIG   BAT  CONFIG   OLD   CONFIG   SD3
*CONFIG   SYS  CONFIG   U1M   STARTUP  BAT
 STARTUP  OLD  STARTUP  TXT   SYSDEF   816
 XEDIT    CFG
```

BIN, DEMOS, etc are subdirectories. TOOLS is a protected subdirectory and CONFIG.SYS is a protected file.

The internal version does not support lists of parameters only the first parameter is evaluated.

A parameter list can be used to list files of different name patterns with one command e.g, if you want to list all files with extension BAT and SYS of the current directory then the command looks like this:

```
DIRC *.BAT *.SYS
```

The listing of the directory from the example above looks like this:

```
 AUTOEXECBAT   CONFIG   BAT   STARTUP  BAT

*CONFIG   SYS
```

All results are listed in the order the name patterns have been specified. The command is aborted if one pattern is invalid e.g., with an invalid path.

DIRL - List Directory Long

```
DIRS [Dn:][path>][filename] ..
```

Low Memory

This command lists the content of a directory like DIR but in addition protected files are marked and the change time includes seconds.

It shows for each file: name, size in byte, and change date/time. Directories are indicated by <dir> instead of size in bytes. The * character in front of a file or directory name indicates that the file or directory is protected.

The listing looks like this:

```
 DEMOS         <dir>   30-09-23 19:08:59
*DOS           <dir>   19-11-22 13:16:08
 GAMES         <dir>   06-10-24 16:07:44
*MAN           <dir>   08-05-11 16:28:06
 TOOLS         <dir>   13-01-24 08:05:45
*STARTUP  BAT       35 12-02-25 18:27:50
 STARTUP  OLD      132 10-09-22 17:03:00
```

In difference to the command DIR the header with volume and directory name and the free sector count at the end are not displayed. Furthermore the size of files is displayed using 8 digit so the size of large files is correctly displayed e.g., files size of 1MB or even 10MB.

DIRS - List Directory Short

```
DIRS [Dn:][path>][filename] ..
```

Low Memory, Internal

This command lists the content of a directory like DIR but the listing is in Atari DOS 2 style.

Directories are indicated by the character : in front of the name. Note that length of files is in sectors. One sector contains 128 bytes for single or medium density, or 256 bytes for double density.

The * character in front of a filename indicates that the file is protected. All numbers in the listing (file length and number of free sectors) are only 3 digits long. Any number greater than 999 is shown as 999.

The listing looks like this:

```
*:DOS          000
* TEST1   BAS 010
573 FREE SECTORS
```

Note: This format is slightly different to the original SpartaDOS format of DIRS where directories were indicated by inverse DIR. This is the same format like the command DIRS of SpartaDOS X uses. This format shows the extensions of directories that have been omitted in the original SpartaDOS format of DIRS.

The internal version does not support lists of parameters only the first parameter is evaluated.

DISASS - Disassembler

```
DISASS file [position]
```

This command lists contents of specified file as an assembler listing. With a position specified, the listing starts from this position in file. Press <CONTROL>+<1> to stop the listing and continue, or press <ESC> to abort the command.

One can use output redirections (see command PRINT) to save the assembler listing to a file.

If you want to display a program code from memory, first you need to save the memory area to a file. For example, to display a program code from memory area $8000-$9000, you need to type the following commands:

```
SAVE D1:TEMP.DAT 8000 9000
DISASS D1:TEMP.DAT 6
```

The assembler listing of D1:TEMP.DAT can be saved to file D1:TEMP.ASM using output redirection with the following command sequence:

```
PRINT D1:TEMP.ASM
DISASS D1:TEMP.DAT 6
PRINT
```

The command PRINT without parameter just stops output redirection.

DOSDRIVE - Change DOS Drive

```
DOSDRIVE (n|OFF)
```

Low Memory

This command changes the DOS drive. The DOS drive is the drive BW-DOS searches first for commands before the current drive is searched, if no drive or path was provided. The default DOS drive is 1. If no parameter is passed then the commands prints the current DOS drive.

If commands should be searched on a different drive than drive 1 e.g., on the Ramdisk on drive 8 (see resident command RAMDISK.SYS) ,then command DOSDRIVE can be used to set the DOS drive to the Ramdisk e.g., drive 8:

```
DOSDRIVE 8
```

If DOSDRIVE is called with parameter OFF or drive number 0 then DOS drive is switched off and commands are only searched on the current drive.

DUMP - Dump File

```
DUMP file [position [length]] [/A]
```

This command displays the content of specified file as hexadecimal and ATASCII listing.

With the option /A non printable characters $00 to $1F and greater $7C are changed to printable ones. Invers characters are converted to their non-inverse counterparts. All other characters are substituted with the character ‚.‘ (dot). This can be used for printing the listing (see the command PRINT).

If a position is specified then the listing will start from this position. If the length parameter is specified then only the specified number of bytes are listed.

Press <CONTROL>+<1> to stop the listing and to continue, or <ESC> to abort the command.

ED - Text Editor

```
ED [file]
```

This command is an editor for text files. If no file is specified a new file is created and the name can be specified at save or on exit. The maximum text buffer size on computers with 16kB main memory is ca. 1700 characters. The text buffer size is ca. 34k characters on computers with 48kB main memory if no cartridge or internal Basic are active.

This command can be used to create and change batch files e.g., for STARTUP.BAT file.

The editor shows the mode and current file name at the top line of the screen and information about the text buffer on the bottom line. If an error occurs e.g., during loading or saving a file, the error number returned by DOS is displayed at the top line.

The top line shows the current mode e.g., edit, load, or save. At start up the version number is shown. Next to the mode the name of the current file is shown. A star (*) in front of the name indicates that there are are unsaved changes in the text buffer.

The bottom line shows a cap lock indicator (arrow up for caps locked and arrow down for capes unlocked), free size of the text buffer in characters, the current line, the current column, and the ATASCII number of the character under the cursor. All numbers are decimals.

At the left and right margin there is an indicator (a dot) for the tagged line. For tagged line see commands below.

The editor is line oriented. The maximum length of a line is 128 characters. If a file should be loaded with lines longer than 128 characters then the error 137 (Truncated lines) is returned and the file is not loaded. If the file is larger than the text buffer then error 137 is also returned but the file is loaded up to the size of the text buffer.

All edit commands known from the standard Atari editor can be used to edit the text buffer including the function keys F1 to F4 of the ATARI 1200XL. All additional commands can be executed by pressing a Control key combination. <ESC> can be use to get the character of a command key combination.

Load and Save:

<CTRL><X>

> Exits the editor. If there are unsaved changes then the editor ask for saving before exit. If you do want to want save changes then you can abort the save dialog by pressing <ESC> and the editor exits.

<CTRL><L>

> Loads a file to the text buffer, the name of the file can be specified at the top line, <ESC> aborts this operation without loading a file.

<CRTL><S>

> Saves the text buffer to a file, the name of the file can be specified at the top line, <ESC> aborts this operation without saving the file.

<CTRL><SHIFT><E>

> Erases the text buffer.

Moving around:

<CTRL><A>

Sets the cursor to the begin of the current line.

<CTRL><Z>

Sets the cursor to the end of the current line.

<CTRL><B>

Sets the cursor to the begin of the text.

<CTRL><E>

Sets the cursor to the end of a text.

<SHIFT><[> or <SHIFT><Up>

Scrolls page up

<SHIFT><]> or <SHIFT><Down>

Scrolls page down

Tagged Line:

<CTRL><T>

Tag the current line, the line is indicated with a dot at the left margin

<CTRL><G>

Goto tagged line

<CTRL><C>

Copy current line before tagged line, text blocks can be copied with a sequence of this command

Display:

<CTRL><V>

Toggles visibility of end of line characters.

<CTRL><U>

Move low margin up, shrink window

<CTRL><D>

Move low margin down, enlarge window

This command is like to the SpartaDOS X command ED an adaptation of the JBW Editor by Janusz B. Wiśniewski (see http://atariki.krap.pl/index.php/JBW_Edit).

FORMAT - Format Disk

```
FORMAT [Dn:]
```

This command formats disks and installs BW-DOS. The drive number specified as a parameter is used for searching DOS files, not for the formatting operation itself. The command asks for all necessary input and formats the disk. The Ramdisks cannot be formatted with this command (see RAMDISK.SYS).

Warning: Formatting erases all data on the disk - even if the disk or files are protected. The only protection against formatting is a write protect tab on the disk.

The first question is which DOS version you wish to install on the formatted disk. There is a menu with every DOS files found. Files of the pattern X*.DOS in the DOS directory and in the main directory are listed. Select one or press <N> for none. It is possible to install any version of BW-DOS or any disk based version of SpartaDOS 2 and later.

Note: It is not possible to format and copy DOS with only 16k main memory. The error „DOS too large“ occurs because there is not enough memory to keep the DOS file in memory. In this case first format without installing DOS and then manually copy and install DOS using the commands COPY and BOOT.

The command ask for the following parameters: drive number, density, volume name, and if XF 551 high speed format should be used.

Note: You can only select a density, which is supported by the drive e.g., Atari 810 supports only single density. See manual of your drive.

Besides the four standard formats single, medium, double, and double-side there are three more options.

> **Reset File System**: Clears disk. The disk must already be formatted and initialized with SpartaDOS file system. This option is the fastest way to clear an already used disk.
>
> **Create File System**: Initializes SpartaDOS file system. Prerequisite is that the disk is already physically formatted and the drive must support the configuration block (Percom block). This option is used to initialize hard disks.
>
> **Custom Configuration**: Formats and initializes the disk with SpartaDOS file system using freely selectable parameters. The drive must support the configuration block (Percom block). This option is used for special formats supported by drives like ATR8000, HDI, XF 551 with 3,5“ modification, or 1050 Mini.

After you have answered all questions insert the disk you want to format and press <RETURN> to start formatting.

If the disk was already formatted in the same density already and you want to clear it only, press <B>. In this case the FORMAT program skips the physical formatting and only an empty main directory is built.

After formatting and if you have selected a DOS, directory DOS is created in the main directory and the DOS file is copied and installed for booting.

The last question is if you want to format another disk.

Note: To physically format a hard disk drives a special programsfrom the hard disk provider must be used. See manual of your hard disk.

GETTD - Get Time and Date

```
GETTD
```

Low Memory

This command prints the current date and time from the BW-DOS clock. It can be used to include time stamps in the output of long running batch files.

Certainly, you can also used it just to see what time it is.

HEXEDIT - Hexadecimal Editor

```
HEXEDIT file [position]
```

This command allows you to edit a file in hexadecimal mode. If a position specified, the editing starts from this position.

This command can be used to edit the PORTB value file for RAMDISK.SYS, see explanation of command MEM and RAMDISK.SYS.

If the file does not exist then the command creats it.

After starting the command it displays position in the file, the old value, and you can enter a new value (hexadecimal without $), or just press <RETURN> to keep it unchanged. Press <ESC> to exit the command. The file is saved before exiting.

Note: The only consistent way to exit HEXEDIT is the <ESC>. Because the edited file is open for write all the time you are working with HEXEDIT, the file can be corrupted by just pressing <RESET> or turning off the computer.

It is possible to set the position greater than the length of file, in this case no old value will be displayed. Once any value is entered, the file is enlarged to the new length with that value.

If you set the position far after the end of the file then the part between the old and new position is not physically stored (non-allocated sectors). A physical sector is allocated at first access to this area. Files with non-allocated sectors cannot be read or copied in the normal way because allocating a sector is not allowed in a read mode.

IF/ELSE/ENDIF - Conditional Batch Execution

IF/ELSE/ENDIF

These commands are used in batch files only. It will ask the user which block of commands should be executed.

The syntax is:

```
IF
the question
commands...
ELSE
commands...
ENDIF
```

It displays the line with question and waits for the answer <Y> or <N>. If the answer is <Y>, then all the commands between IF and ELSE are executed. If the answer is <N> then all commands between ELSE and ENDIF are executed. The <ESC> key stops batch processing. If there are no commands for the answer <N> then the ELSE part can be omitted.

It is possible to define much more complicated structures by starting other batch files inside an IF or ELSE block but there is no way to return to the starting batch file.

LOAD - Load Executable File

```
LOAD file ..
```

Low Memory, Internal

This command loads a DOS loadable file (program) but the program is not started after loading. All INIT and RUN address set by the program are ignored.

If no drive or path is specified with the file then the file is first searched at the DOS drive (see command DOSDRIVE) and if not found there then searched at the current drive.

There are no checks whether the to be loaded program overrides already loaded programs, the memory of BW-DOS, or the OS. General it is not possible to load programs to the memory area from $580 to $6FF because this is the area where the load program itself reside. The command OFFLOAD can be used to check what memory areas are used by a program. The command MEM can be used to check MEMLO (the lowest memory available for programs).

The internal version does not support a list of parameters only the first parameter is evaluated.

MAN - Display Manual

```
MAN name
```

This command provides an environment to display manual pages (small documentation) for every command or program.

The parameter is the name of the command or program without the extension. The command MAN searches in the directory >MAN> on the DOS drive (see command DOSDRIVE) for an ATASCII document with the extension MAN e.g., MAN DIR searches with current DOS drive 1 for the file D1:>MAN>DIR.MAN. If the file is found then it is displayed page by page. Any key can be used to continue to the next page. The command can be aborted any time with <ESC> or >BREAK>.

BW-DOS is delivered with one manual page for every command. The manual pages are omitted on the single density BW-DOS disk because there is not enough disk space. Furthermore there are three special manual pages:

HELP	displays information how to use MAN
COMMANDS	displays a list of all delivered commands
ERRORS	list of all BW-DOS error codes with a short description

For every other program a documentation file with extension MAN can be copied to the MAN directory and then it can be displayed with the command MAN.

If the DOS drive is switched off (see command DOSDRIVE) then the command MAN searches only on the current drive for the manual page.

Manual pages are ATASCII documents and can be edited with the command ED.

MD - Make Directory

```
MD [Dn:][path>]name ..
```

Low Memory, Internal

This command makes a new directory specified by name. There is no logical limit how deep directories can be nested if there is enough disk space. If no drive or path is specified then the current drive or path is used.

Note: You cannot make a directory with the same name as a file in the same directory.

The internal version does not support a lists of parameters. Only the first parameter is evaluated.

MDUMP - Memory Dump

```
MDUMP address [length] [/A]
```

Low Memory

This command displays the content of the memory starting from address with optional length specified as hexadecimal number. If no length is provided the output is endless and restarts after $FFFF with $0000. The command can be aborted at any time with <ESC>,

With the option /A non printable characters ($00 to $1F and above $7C) are converted to printable ones. Invers characters are converted to their non-inverse counterparts. All other characters are substituted with the character ‚.‘ (dot). This is useful if the output is redirected to a printer or file (see the PRINT command).

MEM - Memory Information

```
MEM [file]
```

This command displays information about memory and OS ROM. First the accessible main memory is shown e.g., on 800XL with Basic off is about 62 kB (64 kB minus 2 kB hardware register $d000 to $d7ff). Free memory shows the memory between MEMLO and MEMTOP. The memory info areas $6a, and $02e4 to $02e8 containing RAMTOP, RAMSIZ, MEMTOP, and MEMLO.

RAMSIZ: RAM size, this is the number of pages that the top of RAM represents (one page equals 256 bytes). This value represents the physical memory size for the OS and should never be changed. To reserve high memory see RAMTOP.

RAMTOP: RAM size, defined by power-up, given in the total number of available pages (one page equals 256 bytes). The highest address available for application programs is one byte less than this value. RAMTOP can be lowered to reserve memory.

MEMLO: Pointer to the bottom of free memory. It is updated by DOS or other resident programs. This is the address of the first byte of RAM available for application programs.

MEMTOP: Pointer to the top of free memory. This address is the highest free location of RAM for application programs and data. The value is updated on power-up, when <RESET> is pressed, and when the graphic mode is changed. The display list starts at the next byte above MEMTOP.

The extended memory information area contains the kind of extension (PORTB, or Axlon compatible), size in kilo byte and bank count with optional indicator `+Main` if main memory is also banked e.g., for Rambo XL 256.

OS ROM information areas $C000-$C001 and $FFEE-$FFF9 contain:

`$C000-1`	`CHK1`	Checksum 1, two bytes (LSB/MSB)
`$FFEE`	`DAT`	Revision date D1 and D2 (four-bit BCD)
`$FFEF`	`DAT`	Revision date M1 and M2
`$FFF0`	`DAT`	Revision date Y1 and Y2
`$FFF1`	`XLO`	Option byte; should read 1 for the 1200XL, other XL/XE reads 2
`$FFF2-6`	`PN`	Part number in the form AANNNNNN
`$FFF7`	`REV`	Revision number
`$FFF8-9`	`CHK2`	Checksum 2, two bytes (LSB/MSB)

Example output (depending on your machine):

```
RAM Info:
  Main: 62 kB Free: 40060 bytes
  RAMSIZ=C000 RAMTOP=C000
  MEMLO =1FA4 MEMTOP=BC1F
  PORTB:  192 kB Banks: 12+Main
  AXLON:   48 kB Banks:  3
ROM Info:
  REV=0B DAT=231282 XLO=01
  PN=4141000001 CHK1=9462 CHK2=7740
```

If the computer has extended memory of type PORTB and a `file` name is specified then the valid PORTB values for the extended memory are written to this file. This file can be used with RAMDISK.SYS. It contains exact the values that RAMDISK.SYS would use. The file can be edited

before using it with RAMDISK.SYS to exclude banks with the commands CUT and HEXEDIT to remove banks that collide with other software.

MEMCLEAR - Clear User Memory

```
MEMCLEAR [vv]
```

Low Memory

This command clears the memory from MEMLO to MEMTOP and the user zero page area from $80 to $FF with the hexadecimal value vv. The default value is $00. This command can be used for testing and analyzing tasks.

Example:

```
MEMCLEAR FF
```

The example above clears the user memory from MEMLO to MEMTOP with $FF.

See command MEM for explanation of MEMLO and MEMTOP.

MEMEDIT - Edit Memory

```
MEMEDIT address
```

Low Memory

This command allows you to edit the content of memory. Use this only if you exactly know what you are doing. It is possible to corrupt the system or even to crash it with this command.

If started the command displays the position and the old value. It is possible to enter a new value (in hexadecimal digits without leading $), or just press <RETURN> to keep it unchanged.

Press <ESC> to quit this command.

MENU - BW-DOS Menu

```
MENU [Dn:][/S(E|N|S|D|0)][/B(Y|N)][/Q(Y|N)]
```

Requires 32k

This command is a handy tool to work with sets of files, to copy between two disks with only one disk drive, and to copy files from Atari DOS 2 disks to BW-DOS disks and vice versa (see chapter Menu).

Dn: specifies the current drive for the command menu. The other options are the same like used in the Setup command (see chapter Menu):

/B	bell for alerts
/Q	question before overwriting files
/S	sorting of directories

Setup options B and Q must be followed by Y for yes (on) or N for no (off). Setup option S must be followed by one of the following characters to define the sorting criteria:

E	extension and name
N	name and extension
S	size
D	date and time
0	no sort

MOVE - Move a File

```
MOVE [/Q] [Dn:][path1>]pattern [Dn:][path2]
```

This command allows you to quickly move files and subdirectories to another directory on the same disk.

The difference between to COPY is that MOVE is much faster. It also does not need any free sectors on the disk because it only moves the directory entries of files and subdirectories to another directory without copying the content of file.

This command only works for disk drives. If other devices than disk drives are specified then the error 168 (Command not implemented) is returned.

The MOVE command can only move files and directories on the same drive. If the source and destination drives are different then the error 160 (Drive number error) is returned.

Option /Q (quite) suppresses messages for every file or directory.

Note: If a directory should be moved to a subdirectory of itself then error 150 (File already exists) is returned.

OFFLOAD - List Segments of Executables

```
OFFLOAD file [offset] [/L|Q]
```

This command displays the segments of a DOS loadable file.

The start address, end address, and the file position of every segment (after the header) are displayed.

With option /L the file is also loaded to memory. With option /Q the file is also loaded to memory but the command ask for each segment. If an offset is specified then it is added to the loading addresses of each segment.

This command works like the commands marked as low memory only if the file is not loaded (without option /L or /Q) because the program usually is loaded to user memory.

PAUSE - Wait for Key in Batch

```
PAUSE
```

This command is used in batch files only. It waits for the <RETURN> key.

The <ESC> key stops the batch processing. The PAUSE command can be used to ask for a disk change during the execution of batch files. Do not change the disk the batch file is processed from.

PERCOM - Display Percom Block from Drive

```
PERCOM [Dn:] ..
```

This command reads the Percom block from supporting drives.

If no drive is specified the current drive is read. If the drive does not support Percom blocks then error 139 (Device NAK) is returned. See chapter Disk Format.

The Percom standard was first used and introduced by Percom Corporation, the first manufacture besides Atari producing disk drives for Atari 8bit computers. The Percom block is a configuration block within the memory of the disk control microprocessor describing disk parameters. It has the size of 12 bytes with the following structure:

Offset	*Length*	*Short Description*
0	1	Number of Tracks
1	1	Step Rate (values have no universal meaning)
2-3	2	Sectors per Track (byte 2 is high byte; byte 3 is low byte)
4	1	Number of Sides or Heads (0=one head, 1=two heads)
5	1	Flags e.g., Bit 2 for Density (0=FM/Single, 1=MFM/Double)
6-7	2	Bytes per Sector (byte 6 is high byte; byte 7 is low byte)
8	1	Drive type
9-11	3	Miscellaneous

The Percom block is supported by the Atari XF 551 and also by the Atari 1050 disk drive with upgrades to support double density. The Percom block is standard for hard disks.

The output for a disk with single density in drive 1 looks like this:

```
        Drive: 1
 Num of sides: 1
  Tracks/side: 40
Sectors/track: 26
 Bytes/sector: 128
Stepping rate: 1
  Flag FM/MFM: 04
   Drive type: 01
   Misc bytes: C0 00 00
```

In difference to the Percom block, describing the current configuration of the drive, the command CHDSK shows information contained on the disk.

This command only displays the current Percom block of a drive. It is not possible to change the configuration by writing the Percom block to change the values with this command.

POKE - Change Memory Bytes

```
POKE aaaa vv[vv] ..
```

Low Memory

This command writes a byte vv or a word vvvv to the memory address aaaa. All parameters are hexadecimals. It can be used to change system variables.

Examples:

Write the value $00 to address $2C7 COLOR2 (sets background color to black).

```
POKE 02C6 00
```

Write the value $50 to address $02e7 and value $20 to address $02e7+1 (sets MEMLO to address $2050).

```
POKE 02E7 2050
```

A list of address and value pairs can be passed to change the value of several addresses e.g., the following command does the same like to two examples above:

```
POKE 02C6 00 02E7 2050
```

Furthermore, the command POKE can be used to fake the version number of SpartaDOS and BW-DOS. The following command set the version number to SpartaDOS 3.3:

```
POKE 0701 33
```

To fake RealDOS version 1.0 the following command can be used:

```
POKE 0700 52 0701 10
```

To set the version variables back to SpartaDOS 3.2 (standard for BW-DOS) the following command can be used:

```
POKE 0700 53 0701 32
```

PRINT - Redirect Console Output

```
PRINT [device:|file]
```

Low Memory, Internal

This command is used for output redirection. It starts the hard copy function to a device or file. Without parameter an active output redirection is stopped.

The hard copy function allows you to copy all screen output from almost every program to a specified device or file. It does not work with programs that does not use CIO calls for screen output.

Originally this command was introduced to redirect output from programs to a printer but you can redirect the output to any file or device.

PROT - Protect Files

```
PROT file ..
```

Low Memory, Internal

This command protects files. Protected files cannot be changed or deleted.

Wildcards * and ? you can be used to define a set of files to be protected. This command can also protect directories. If a directory is protected it cannot be remove but files in that directory can still be changed and deleted.

The internal version does not support a lists of parameters. Only the first parameter is evaluated.

Note: The write lock status of disks known from SpartaDOS is not supported by BW-DOS. If write lock is set with SpartaDOS then this is not protecting from changing or formatting these disks with BW-DOS.

PWD - Print Working Directory

```
PWD [Dn:] ..
```

This command lists the current path of a drive. If no drive is specified then the current drive is used. The current directory can be changed with the command CD. This command supports a list of drives. If more than one drive is specified then the current path of every drive is listed, one per line. Additional paths specified at the drive parameters are ignored.

Example:

Command to print work directory of drive 1 and 2.

```
PWD D1: D2:
```

The output could look like below.

```
D1:>
D2:>WORK>BASIC>
```

RD - Remove Directory

```
RD [Dn:][path>]name ..
```

Low Memory, Internal

This command deletes directories.

You can only delete empty directories. If there are any files or directories in it, you need to remove them first. Never try to delete the current working directory because this can corrupt the file system.

The internal version does not support a list of parameters only the first parameter is evaluated.

REN - Rename File

```
REN file|directory newname
```

Low Memory, Internal

This command allows you to rename a files specified by parameter `file` or `directory` to new name `newname`. You can use wildcards for `file` and `filename` to rename a set of files.

For example, if you have the files TEST and BEST in the current directory then the following command renames them to TESTS and BESTS:

```
REN ?EST ????S
```

The wildcards * and ? must be used very carefully. If you change two file names in a directory to the same name, you cannot work with the second one anymore.

If two files in the same directory have the same name there is only one option. First the command MOVE must be used to move one of the files to another directory, next rename the remaining file with command REN, and at the end use the command MOVE to move the first file back to its original directory.

Only the external version of REN can also rename directories. It is not possible to use wild cards for renaming subdirectories.

```
REN OLD_DIR NEW_DIR
```

You cannot rename directories with the internal version of this command. In this case you can create a new directory with the command MD, move all files from old to new directory using command MOVE, and remove the old directory with command RD. The following command sequence „renames“ directory OLD to directory NEW:

```
MD NEW_SUB
MOVE OLD_SUB>*.* NEW_SUB
RD OLD_SUB
```

The internal version of REN performs no parameter check. Every character that is not a valid character for a file name or the path separator > separates file from new file name. Additional parameters are ignored by the internal version. With the internal version the following examples trenames the file WORK to TEST.BAS (< is not a valid character for file names) instead of renaming WORK>TEST.BAS to TEST2.BAS.

```
REN WORK<TEST.BAS TEST2.BAS
```

A further limitation for the internal version is that the maximum length of file plus the length of the new file name is 28, the length of the parameter buffer (see BW_COMFNAM in chapter Page 7 and COMTAB). The limitation for the external version is 28 bytes for each the file and the new file name.

RUN - Run Address

```
RUN [A:|[A:]address [ParameterList]]
```

Low Memory, Internal

This command starts a program at the given address. Without parameter it restarts the last loaded program, or the last program started with this command. If a start address is specified then an optional parameter list can be specified, too.

This command is for advanced users and programmers. RUN used with a wrong address can results in a system crash and may even cause data losses.

Examples:

```
RUN
```

Restarts the last loaded program (first check the manual of the program whether this is supported)

```
RUN E471
```

or

```
RUN A:E471
```

Jumps to the Self Test program (XL/XE only), or to the menu of the Q-MEG operating system if installed.

The qualifier A: (address) can be used to restart a program with a different parameter list but same start address. If A: is specified without an address then the last start address is used.

```
FA PROG1.ASM PROG1.COM
RUN A: PROG2.ASM PROG2.COM
```

The command sequence above starts the assembly of PROG1.ASM using the assembler FA.COM (Fast Assembler 1.8, see https://github.com/HolgerJanz/FastAssembler/) and restarts the assembler with PROG2.ASM. This sequence is the same like the sequence below but there the assembler FA.COM is loaded twice.

```
FA PROG1.ASM PROG1.COM
FA PROG2.ASM PROG2.COM
```

It is also possible to specify the right address for every restart with different parameter lists.

```
FA PROG1.ASM PROG1.COM
RUN 2800 PROG2.ASM PROG2.COM
```

or

```
FA PROG1.ASM PROG1.COM
RUN A:2800 PROG2.ASM PROG2.COM
```

In the example above the right start address for Fast Assembler is $2800.

The external version resides in the memory area from $580 to $6FF so every specified address below $700 is ignored by the external command to prevent system crashes.

SAVE - Save Memory Segment

```
SAVE file[/A] start end [/A]
```

Low Memory

This command saves the memory area specified by start and end addresses as a DOS loadable file. With the /A parameter at the end of the command it adds another segment to an existing file.

The difference between the parameter /A as a part of file and the parameter at the end of the command is that the command with the separate /A at the end writes the segment header without the $FFFF marker.

SECOPY - Copy Sectors

```
SECOPY Dn:[filename] Dm:[filename]
```

This command copies all sectors of a disk to file or vice versa. It works only with 128 Bytes sector disks (single density). The source or destination disk is not analyzed or formatted. Sectors are copied from or to disk until either a read or a write error occurs.

Examples:

```
SECOPY D1: D8:
```

Copies all sectors from drive 1 to drive 8.

```
SECOPY D8: D4:>BACKUP>RAMDISK.DSK
```

Copies all sectors from drive 8 to file D4:>BACKUP>RAMDISK.DSK.

```
SECOPY D4:>BACKUP>RAMDISK.DSK D8:
```

Copies all sectors from file D4:>BACKUP>RAMDISK.DSK to sectors of drive 8.

One use case is to back up a Ramdisk so that it can be restored the next time the system is started.

SORTDIR - Sort Directories

```
SORTDIR [/RQ] :|path ..
```

This command sorts the directory entries first by subdirectory or file and second alphabetically by name and extension. The colon (:) can be used to specify the current directory of the current drive.

With option `/R` subdirectories are also sorted. Protected subdirectories are excluded from sorting.

Option `/Q` (quite) suppresses messages for every directory.

Examples:

```
SORTDIR D1:>TMP
```

Sorts directory `TMP` in main directory of disk 1.

```
SORTDIR D2:
```

Sorts the current directory of disk 2.

```
SORTDIR /R D2:>
```

Sorts main directory of disk 2 including all subdirectory. With this command all directories on disk 2 are sorted.

```
SORTDIR :
```

Sorts the current directory of the current drive.

TIME - Set Time

```
TIME
```

This command displays the current time from the BW-DOS clock and allows you to enter a new value. Press <RETURN> to abort without changing the time.

The time is displayed and must be supplied in 24h format and with the following layout HH:MM:SS.

TYPE - Display File

```
TYPE file ..
```

Low memory, Internal

This command shows the content of files, for example a batch files like STARTUP.BAT.
<CTRL>+<1> can be used to pause and to continue the output.

The internal version does not support a list of parameters. Only the first parameter is evaluated by the internal version.

UNPROT - Unprotect File

```
UNPROT file ..
```

Low memory, Internal

This command unprotects files (see PROT for protecting files and directories). This command can also unprotect directories.

The internal version does not support a lists of parameters. Only the first parameter is evaluated by the internal version.

VERIFY - Switch Write Verify

```
VERIFY [ON|OFF]
```

Low Memory

This command switches on or off the option write with verify.

The system default is off. Set to on, every sector is read after writing and data are checked. In case there are differences the error 145 (Verify error) is returned.

Without parameter the command only displays the current verify state.

If the option verify is on then the write performance is decreasing.

Hint 1: Some drive types redirect the write with verify command internally to the standard write command so neither a verify nor an extended write time will occur.

It can be useful if the used disk is knowingly unreliable.

Hint 2: Knowingly unreliable media should be rather replaced instead of continuously being used even with write verify.

VERS - Print DOS Version

```
VERS
```

Low Memory

Displays the current DOS version. It supports BW-DOS, SpartaDOS 3 and higher, and RealDOS.

In case of BW-DOS the currently set SpartaDOS version is also displayed.

Resident Commands

Residents commands are commands that do not immediately execute a certain function but rather install new functions or extensions to the system. These extensions are not always needed or there are different versions so a certain driver is only installed on demand by calling the resident command. Usually the program code is installed at MEMLO and MEMLO is raised to point after the newly installed code.

Well known resident commands are extensions to provide access to a specific hardware clock or install a ramdisk e.g., CLOCK.SYS and RAMDISK.SYS.

Some resident commands can be deinstalled. Usually the option OFF is used for deinstalling. There are some limits for deinstalling resident commands. All resident commands are installed in the order they are called like a chain. Only the last installed program can be deinstalled. If several drivers have to be deinstalled then they must be deinstalled in the reverse order they have been installed.

There are also limits to the memory all installed resident programs may use. MEMLO must be checked using the command MEM. If MEMLO is raised above $2000 then some programs will not work anymore. In general: the higher the MEMLO the higher the chance that some program might not work. External BW-DOS commands do not use memory below $2800. The command OFFLOAD can be used to check what memory areas are needed by a certain program. If MEMLO is raised above $4000 further issues occur if extended memory is used because at this address the mapped window for the extended memory starts.

CLOCK.SYS - Software Clock

```
CLOCK.SYS ON|OFF
```

This resident command is a software clock for BW-DOS.

If you have installed this driver and entered the current date/time (see commands DATE/TIME or APETIME) then all files and directories get the current date and time if they are created or changed.

This clock is based on a counter that is set during the VBI interrupt, so it is possible that a few programs may stop or pause it. Usually serial input and output pauses this counter so this software clock will be behind the real actual time more and more. A hardware clock is required to avoid this.

This software clock works on both PAL and NTSC machines.

The parameter ON installs the clock and OFF deinstalls the driver, if it was the last driver installed.

IDEP2TD.SYS - Clock for IDE+2

```
IDEP2TD.SYS [/R]
```

This command installs a driver for a hardware real-time-clock (RTC) of the IDE plus 2 interface.

The IDE plus 2 device gives the correct date and time information to the system without the need to reenter the current date and time at each reboot. It works with all versions of the PBI (Parallel Bus Interface) device IDE plus 2 by JZ & KMK.

This driver has no uninstall option.

If the driver is successfully installed the version of the driver and the current date and time are displayed.

The driver can be installed more efficiently if it is the first resident program installed. In this case a shorter version of the drive is installed.

The option /R installs a read-only version of the driver that requires less memory and keeps MEMLO lower. With this option it is not possible to set the hardware clock. All changes to date and time are ignored without any error message.

The last character in the status line indicates how the driver was installed:

S	short version, read and write driver
M	read and write driver
s	short version, read driver (requires the fewest memory)
m	read driver

For more information about the IDE plus 2 interface see http://drac030.krap.pl/en-kmkjz-pliki.php.

RAMDISK.SYS - PORTB and Axlon Ramdisk

```
RAMDISK.SYS (n[E][F][A] [file])|OFF[A]
```

This resident command sets up an emulated disk drive using extended RAM up to 1MB.

PORTB (volume name is set to RAMDISKB) and Axlon (volume name is set to RAMDISKA) compatible extensions are supported. This enables you to use the extended memory like a disk drive - except reformatting it and booting from it. All files on a ramdisk are lost if you turn off the computer (unless battery buffered RAM is used like Freezer 2011, 800 XL Advanced Remake). The ramdisk is destroyed and the system may crash if you use a program, that uses extended memory of the ramdisk for other purposes.

The parameter n is the drive number for the ramdisk drive.

With option F the ramdisk is formatted. Without option F it is only formatted if the previously existing format is invalid. Without of option F you can reinstall the driver - even after a re-boot without turning the computer off - without destroying the contents of your ramdisk.

With option E the 130XE compatible banks (only for machines with more than 128kB memory) are not used for the ramdisk so the ramdisk does not collide with other 130XE software. This option is ignored for Axlon extensions or if a configuration file is used.

First it is checked for PORTB extensions. If there are no PORTB banks then Axlon banks are check. Afterwards the bank count and kind are displayed.

The option A enforced the use of Axlon extended memory. This is useful if the computer is equipped with both PORTB and Axlon compatible extended memory.

It is possible to install two ramdisks if the computer has PORTB and Axlon memory. The driver must be started two times and one time with the option A e.g.:

```
RAMDISK.SYS 8
RAMDISK.SYS 4A
```

The example above installs the ramdisk using PORTB extended RAM on drive 8 and the ramdisk using Axlon extended RAM on drive 4.

If the ramdisk collides with your software and you know that there are some memory banks left then you can use the configuration file to resolve this issue. This file is a binary file with the size between 1 and 64 bytes. It contains the values for the control register of PORTB ($D301) or Axlon ($CFFF) to switch banks in the area of $4000-$7FFF. The driver checks the values before activating the ramdisk.

The command HEXEDIT can be used to create and edit a binary configuration file. The command DUMP can be used to display a configuration file. The command MEM can be used to create a suitable file with valid PORTB values. This file can be used as a starting for creating an own configuration file.

A PORTB configuration file, containing the value for PORTB values for all banks of an Atari 130XE (4 banks), looks like this (hexadecimal display using command DUMP):

```
EF EB E7 E3
```

An Axlon configuration file, contain values for a Ramdisk of 64k with the first 4 banks, looks like this (hexadecimal display using command DUMP):

```
01 02 03 04
```

To reformat an existing Ramdisk restart the command with the same configuration and option F. If you want to change the drive number then restart the command only with a different drive number. In this case the Ramdisk is preserved. If other parameter are changed the Ramdisk is reinitialized.

Note: The resident part of the Ramdisk driver uses parts of the extended RAM so any collision in the use of the extended RAM may cause a crash of the system at SIO operations.

Option OFF removes the driver if it was the last driver installed. If the computer is equipped with PORTB and Axlon extended memory then option OFFA deinstalls the Axlon driver. If both PORTB and Axlon Ramdisks are installed then the right order must be adhered, the last installed driver must be removed first e.g. for the example above the right order to deinstall both is:

```
RAMDISK.SYS OFFA
RAMDISK.SYS OFF
```

RTIME8.SYS - Clock for R-Time 8

```
RTIME8.SYS [/R]
```

This command installs a driver for a hardware real-time-clock of the R-Time 8 cartridge designed by ICD or compatible ones.

Such a device gives the correct date and time to the system without of a need to reenter the it after each reboot.

With option /R a read-only version is installed. It allows only reading from the clock, setting a new date or time is ignored without error in this mode. This option saves memory and keeps MEMLO low.

Note: This driver can also be used with emulators. Atari800MacX supports the R-Time-8 hardware clock be default. In Altirra, this hardware clock can be configured.

XFSIO.SYS - Fast SIO for Atrai XF 551

```
XFSIO.SYS n..|OFF
```

This resident command provides fast communication for the XF 551 disk drive.

The drive number(s) of the XF 551 drive(s) must be passed to install the driver.

The communication is 2.5 times faster if the disk is formatted in the fast mode, see command FORMAT.

The keyboard is disabled during fast SIO operations to prevent problems.

It is also possible to use other high-speed SIO driver e.g., HISIO driver developed by Matthias Reichl (see https://github.com/HiassofT/highspeed-sio). The disadvantage of this driver is that they are usually use the RAM under the OS ROM.

Functions in Basic

This chapter shows how to use the operations provided by BW-DOS in your own programs written in Basic. These operations can also be used in other programming languages.

IOCB

The IOCBs (Input Output Control Blocks) are the central data structures for the input and output via the Atari operating system. They work like channels. There are 8 IOCBs numbered from #0 to #7. Each IOCB can be used for one specific device for input and output depending on the device e.g., disk drive for input and output but keyboard only for input and printer only for output.

Some of the IOCBs are used by the system for certain tasks. The following table shows the IOCBs and there usage by the system:

#0	used by operating system for standard input and output via E:
#1	unused
#2	unused
#3	unused
#4	used by BW-DOS for hard copy
#5	used by BW-DOS for batch processing
#6	used by Basic for graphics e.g., GRAPHICS 8 opens this channel
#7	used by Basic for load, save, and printing e.g. statements LOAD, SAVE, LIST

Only IOCBs #1-#3 should be used by applications to ensure that there are no collisions with system functions e.g., that hard copy and batch processing works with the applications. If Basic is not in use then the IOCBs #6 and #7 are also available.

Open

```
OPEN #iocb,aux1,aux2,"Dn:[path>]filename"
```

The command OPEN (3) opens a specified IOCB for access to a specified file on a disk.

Example to open the file D1:TEST.TXT for read:

```
OPEN #1,4,0,“D1:>TEST.TXT“
```

If there is no drive number specified e.g, just D:, then the default drive number is 1.

If the file should be opened at the current drive then the current drive number must be determined using BW_CURDRV (see chapter Page 7 and COMTAB). An example in Basic could look like this:

```
DIM F$(20)
F$="Dx:TESTFILE.TST"
F$(2,2)=CHR$(PEEK(1959))
```

Here the x in the file specification F$ is substituted with the current drive number. Now F$ can be used to open the file TESTFILE.TST at the current drive.

If the file should be opened at the DOS drive then the DOS drive number must be determined using BW_DOSRV (see chapter Page 7 and COMTAB) and x must be substituted. An example could look like this:

```
DIM F$(20)
F$="Dx:TESTFILE.TST"
F$(2,2)=CHR$(48+PEEK(1962))
```

Here x of the file specification F$ is substituted with the DOS drive number. Because BW_DOSDRV contains the number of the DOS drive (not the ASCII presentation), it must be converted to ATASCII by adding 48. Furthermore it should be check whether DOS drive is switched of e.g., BW-DOSDRV is 0 (see command DOSDRIVE).

The valid value of aux1:

4 - Reading from the file.

6 - Reading the directory. If the aux2 value is greater than 127 then the listing is like the command DIR. If aux2 is less than 128 then the listing is like the command DIRS. You may not open more than one directory listing at the same time.

8 - Writing to the file. If the file already exists, it is replaced with the new data. If option /A is added to the filename then it works like value 9 for aux1.

9 - Append, the new data written to the file is appended to the existing file. If the file does not exist then it is created first.

12 - Update. Data can be read from and written to the file.

If 16 is added to the aux1 value then the open operation gives direct access to the directory specified by path, the filename is ignored. If 32 is added to aux1 then the open operation gives direct access to the subdirectory specified by filename. See the chapter Disk Format for more information on the internal directory structure. These two options do not work with the aux1 mode 6. If a directory is opened for direct access in mode 8, 9, or 12, and the directory does not exist then it is created first. The position in directory data is set to 23 byte (after the first directory entry) in every mode except in mode 9. If the first block should be accessed then the POINT operation must be used.

Warning: Operations on directories must be handled with care. It is possible to destroy the whole directory structure of a disk.

Input and Output

```
GET #iocb,variable
PUT #iocb,variable
INPUT #iocb,variable
PRINT #iocb,variable
```

These commands send data to the specified IOCB or receive data from the IOCB.

Note: It is much faster to send data as one block. This can be done by using CIO commands $07 and $0B in assembler language or by the commands BGET and BPUT in Turbo Basic.

Close

```
CLOSE #iocb
```

The command CLOSE (12) closes the IOCB. Every IOCB must be closed after use.

Point and Note

```
      X=POS:Y=0:POINT #iocb,X,Y
```

or

```
      Y=16*iocb:X=INT(POS/65536):POKE 846+Y,X
      POS=POS-65536*X X=INT(POS/256):POKE 845+Y,X
      POKE 844+Y,POS-256*X
      XIO 37,#iocb,PEEK(842+Y),PEEK(843+Y),"Dn:"
```

The examples above set the position in a file, the POINT function. The position is an offset in bytes in a file. If the file is opened in mode 8, 9, or 12, then you can also set the position behind the end of file.

The first method works only for positions less than 32768 while the second method can be used for any position up to 8 megabytes. The IOCB must be opened first to set a position.

```
      Y=16*iocb
      XIO 39,#iocb,PEEK(842+Y),PEEK(843+Y),“Dn:"
      LENGTH=PEEK(844+Y)+256*PEEK(845+Y)+65536*PEEK(846+Y)
```

The example above provides the size of a file for the opened IOCB.

```
      NOTE #iocb,X,Y:POS=X+65536*Y
```

The command above gets the current position in a file (operation NOTE) for the opened IOCB. This command works similar to XIO command 37 (set position), the XIO command code is 38.

SpartaDOS extends the concept by defining the content of the bookmark value as byte offset from the beginning of a file (full 24-bit size). This value may be handled in the same way as with DOS2 (NOTE-store-POINT bookmarking), but may also be easily built from scratch without previously visiting the file position and executing a NOTE. This enhancement opens a great possibility to go straight to any file position even beyond existing file length, as long as the file is open in a write-allowed mode, in this case non-written parts of the file are set by default to zero and do not need to be physically allocated to disk sectors until first access. SpartaDOS defined the 6502 byte-order (low-to-high) for the 24-bit value.

Atari Basic provides the commands NOTE and POINT. These commands assume the sector/byte format of the bookmark value. It is divided into two arguments: X (16-bit sector number) and Y (8-bit byte offset). SpartaDOS definition maps to these in such a way, that small file positions (under 32768) may be directly passed through the X argument, while the Y argument (upper byte of the position) is set to zero. Higher values may be retrieved through NOTE as well, but because BASIC extracts the upper byte into a separate argument Y, both the parts need to be joined by adding 65536*Y. This does not work with POINT because BASIC only allows 15-bit value for X (sector number), so any greater value needs to be passed directly to CIO using the XIO command.

Special Operations

```
XIO 32,#iocb,0,0,"Dn:[path>]filename filename"
```

This operation provides the same functionality as the command REN. The space character between old and new filename may be replaced by any character, which is not allowed in the filename e.g., character "," . The IOCB must be closed before this operation.

Following operations provide the same functionality like the BW-DOS commands - as listed below. The IOCB used for such an operation must be closed first.

```
XIO 33,#iocb,0,0,"Dn:[path>]filename"    DEL command
XIO 35,#iocb,0,0,"Dn:[path>]filename"    PROT command
XIO 36,#iocb,0,0,"Dn:[path>]filename"    UNPROT command
XIO 40,#iocb,4,0,"Dn:[path>]filename"    load and run a program
XIO 40,#iocb,4,128,“Dn:[path>]filename“  LOAD command
XIO 42,#iocb,0,0,“Dn:[path>]name“        MD command
XIO 43,#iocb,0,0,“Dn:[path>]name“        RD command
XIO 44,#iocb,0,0,"Dn:path"               CD command
```

The last operation XIO 47 is not easily used from Basic. The settings for CIO in Assembler language are:

```
ICCMD   = 47
ICBAL/H = address of the string "Dn:"
ICBLL/H = address of the output buffer
```

It provides the information similar to the command CHKDSK. The result in output buffer is always 17 bytes long:

```
0   : Version of disk format
1   : Size of sectors. (0 means 256.)
2,3 : Number of every sectors
4,5 : Number of free sectors
6-13: Volume name
14  : Sequential number
15. : Random number
16. : Always 0 (reserved for compatibility with SpartaDOS).
```

Disk Format

BW-DOS uses the same disk format like SpartaDOS.

Every disk is divided in consecutive numbers of sectors. A sector consist of a consecutive number of bytes. Sector sizes of 128 and 256 bytes are supported. There are four different kinds of sectors:

1. Boot sectors
2. Bitmap sectors
3. Map sectors
4. Data sectors

Boot sectors are the sectors 1, 2, and 3 on every disks. These sectors are always 128 bytes long and contain a small program (boot loader) that is started if the computer is turned on - this program loads DOS or any other file specified by the BOOT command. The boot loader is different between BW-DOS and SpartaDOS but the functionality is almost identical.

Sector 1 contains important information about the disk (the numbers are offsets in the sector 1):

0: Unused, set to 0 by command FORMAT

1: Number of boot sectors, 3 for single and double density

2: Load address for boot sectors, $3000 for SpartaDOS 3 and BW-DOS

4: Init address for DOS, $7E0 (see BW_DINT)

6: Jump instruction (three bytes) to execute boot sectors, after boot sectors are loaded the system jumps to this address, contains instruction JMP $3080,

7: Is always $80. This byte is used to identify a SpartaDOS compatible disk. If the value is different, error 148 is returned at access to such a disk.

9: Sector number of first map sector of the main directory (2 bytes)

11: Number of all sectors on the disk (2 bytes)

13: Number of free sectors (2 bytes)

15: Number of bitmap sectors

16: Number of the first bitmap sector (2 bytes)

18: The sector number that is used to start a search for a free sector. Usually it is the last allocated sector. It is used to speed up write operations because not all bitmap sectors must be searched for a free sector. In addition, a few sectors are leaved unused on first tracks of the disk to provide space for expansion of the directories. This space is only used for files if there are no other free sectors on the disk. This state is indicated by a value equal to maximum sector -1. If a file or directory is deleted then it is set set to the last sector that was freed by the delete operation. This ensures the disk dos not become defragmented and speeds up the search for the next free sector.

20: Starting sector for allocating directories - see the previous explanation. If a file or directory is deleted then, it is set to 2.

22: The volume name of the disk (8 bytes) - together with the sequential and random number - is used to identify a disk. It is used to invalidate buffers if a disk was changed.

30: Number of tracks, if the bit 7 is set then the drive supports double sides.

31: Size of sectors (exception the boot sectors). Valid values are 128 or 0 for 256 bytes per sector.

32: Version of disk format. BW-DOS and SpartaDOS versions 2.x and later use the format marked as $20 in this byte. Older disks with value $11 - created by SpartaDOS 1.x - must not used with BW-DOS. SpartaDOS X disks use $21 and can be used with BW-DOS.

33-37: Reserved for different SpartaDOS versions.

38: Sequential number of the disk. This number is incremented every time the disk is modified.

39: Random number of the disk. This number is created when the disk formatted.

40: Number of the first map sector of the file specified for booting. (2 bytes)

42-47: Reserved for different SpartaDOS versions.

Bitmap sectors contain the information which sectors are free and which are used. Each byte holds the information for eight sectors. The highest bit 7 is for the first, and the lowest bit 0 for the last of the eight sectors. If the bit set to 1 it means that the sector is free. The first byte contains the information for sectors 0-7, the second byte for 8- 15 etc.

If more than one sector is necessary for the bitmap of a disk then the rest of the bitmap is stored in the consecutive sectors (for example in sectors 4, 5, and 6).

Map sectors contain information about which sectors are used for a file. It contains the numbers of sectors (2 bytes for each). The first number in every map sector (two bytes!) points to the next map sector of the file and zero indicates the last map sector. The second number points to the previous map sector of the file and zero indicates the first map sector. The remaining part of the map sector contains the numbers of data sectors used for the file.

Directories are in just special files. The directories contains information about every file and every subdirectory, 23 bytes each. The numbers are offsets in the 23 bytes long entry block.

0: The status bytes. Single bits are used:
- 0 -The file is protected.
- 3 - Block in directory is used.
- 4 - The file is deleted (block in directory is free).
- 5 - Flag for a subdirectory.

The status equal to zero indicates the end of directory.

1: Number of the first map sector of the file (2 bytes).
3: Length of the file (3 bytes - not for a subdirectory).
6: Name and extension (11 bytes).
17: Date and time in the same format like DATER and TIMER (see chapter Page 7 and COMTAB).

The first entry is special. It contains the information about the directory itself. If you want to read or write this entry then you must use the POINT operation - every direct OPEN to a directory sets the file position to 23.

The information in the first entry contains:

1: Number of the first map sector of the parent directory (2 bytes).
3: Length of the directory in bytes (3 bytes).
6: Name of the directory (8 bytes).

Page 7 and COMTAB

In memory page 7 there are special information and the table COMTAB e.g., the version of the DOS. They are used by machine code programs to use special functions of BW-DOS. The minimum BW-DOS version number that are required to use this parameter is in parentheses e.g., (1.4) for minimum of BW-DOS version 1.4.

Page 7

$700: BW_SPARTA
There is always the character "S" for compatibility with SpartaDOS. Many programs determine by this address, if they can use advanced SpartaDOS (BW-DOS) functions.

$701: BW_SVERS
There is always the number $32 for compatibility with SpartaDOS 3.2.

$702: BW_SSVERS
There is always the number $00 to provide compatibility with SpartaDOS 3.2.

$703: BW_DOS
Two bytes. There are the letters "BW" if any version of BW-DOS is installed.

$705: BW_VERS
BW-DOS version. $10 is for version 1.0, $15 is for version 1.5 etc. This can be used to check for a specific version of BW-DOS because some functions are only available starting from certain versions.

$706: BW_GETTD
Address of the GETTD routine. This routine reads the date and time from the BW-DOS clock, and stores them into BW_DATER and BW_TIMER in the COMTAB.

$708: BW_SETTD
Address of the SETTD routine. This routine reads the date and time from BW_DATER and BW_TIMER in the COMTAB, and stores it into the BW-DOS clock.

$70A: BW_CONVDC
Address of CONVDC routine. This routine will convert a 24-bit binary number from BW_DECIN in the COMTAB to the 8 characters long ASCII text in BW_DECOUT.

$70C: BW_BBSIO
Contains a jump to the SIO routine address in BW_LSIO in the COMTAB. This is provided for compatibility with BiboDOS. Any changes done to this address may not change functions of BW-DOS.

$70F: BW_FAIL (1.4)
Address for error handling and jump to DOSVEC. This routine never returns. The error code must be passed in register A. Error handling includes termination of a batch processing and the output of the error number to the screen. It can be used for error handling of CIO or SIO calls e.g.:

```
            jsr BW_SIO
            bpl no_error
            tya
            jmp (BW_FAIL)
no_error    ...
```

This address is only available in BW-DOS as from version 1.4. Check BW_DOS and BW_VERS for $14 before using it.

COMTAB

The COMTAB table allows machine code programs to get parameters from the command processor, and to use many other features of BW-DOS. The starting address of this table is stored in the DOSVEC ($0A). The minimum BW-DOS version number that are required to use this parameter is in parentheses e.g., (1.4) for minimum of BW-DOS version 1.4. Numbers following COMTAB in the following description are relative positions in this table e.g., COMTAB-10 means subtract 10 from the address in DOSVEC and COMTAB+8 means add 8 to the address in DOSVEC.

With BW-DOS version 1.4 and 1.5 the COMTAB can also be absolutely addressed in page 7. Check BW_DOS for ‚BW' and BW_VERS for greater equal $14 or $15 before using absolute addresses to COMTAB. This is not compatible with SpartaDOS.

$711: BW_DWARM (1.4) COMTAB-21:
The value of WARMST ($08) is stored here before starting a cartridge. It provides the information whether the memory was changed by an external command $00, or not $FF.

$713: BW_DECOUT (1.4) COMTAB-19:
This is the output buffer for the BW_CONVDC routine. It contains up to 8 characters long decimal representation of the number from BW_DECIN. Spaces are added at the begin. This address is only compatible with SpartaDOS X.

$71B: BW_SIO (1.4) COMTAB-11:
Jumps to BW_LSIO (see BW_LSIO).

$71C: BW_LSIO (1.4) COMTAB-10:
Contains the address of the BW-DOS SIO routine. This routine is used by BW-DOS for every disk operation. It has the same input and output like the standard SIO routine in the OS ROM ($E459). At startup this address points to $E459. It is changed by commands like RAMDISK.SYS.

$71E: BW_ECHOFLG (1.4) COMTAB-8:
This is the index into HATABS for the hard copy function. It contains the value $FF if the hard copy function is not active. It is possible to check, if the hard copy function is active by checking this byte.

$71F: BW_BATFLG (1.4) COMTAB-7:
It is the similar to BW_ECHOFLG. This flag is set for batch file processing.

$720: BW_DECIN (1.4) COMTAB-6:
Input for the BW_CONVDC routine (3 bytes). This address is only compatible with SpartaDOS X.

$724: BW_WRTCMD (1.4) COMTAB-2:
Contains the SIO command used for write operations. It is "P" for writing without verify, or "W" for writing with verify. Any other value may cause issues. It is recommended to use this SIO command for all write operations in every program.

$726: BW_COMTAB (1.4) COMTAB+0:
There is an instruction to start the command processor. You can use the instruction JMP ($0A) to exit a program. It starts the command processor, and is compatible with other DOS versions e.g., in Atari DOS it jumps to the DUP menu.

$729: BW_CRNAME (1.4) COMTAB+3:
Contains an instruction to jump to the CRNAME routine. This routine retrieves the next parameter from command line in BW_LBUF, adds the prefix Dn: if no drive was specified, and stores the result in BW_COMFNAM. Sets Z=0 if a parameter was fetched, and Z=1 if there are no more parameters. A sequence of calls to this routine must be used to retrieve

all parameters.

$72C: BW_DIVIO (1.4) COMTAB+6:
Contains the address of the DIVIO routine. This routine is used to start a batch file or the hard copy function. Before starting this routine, the complete filename, "device:" or "Dn:[path>]filename" must be stored in BW_COMFNAM, and the Y register must be set to 0 for the hard copy function, or 1 for a batch file procession. This routine returns any errors in the Y register and sets the N flag in the same way like the CIO and SIO routine. This is different to SpartaDOS 3.x.

$72E: BW_XDIVIO (1.4) COMTAB+8:
Contains the address of the XDIVIO routine. This routine is used to stop a batch file processing or the hard copy function. Before starting this routine, the Y register must be set to 0 for the hard copy function, or to 1 for batch file processing.

$730: BW_BUFOFF (1.4) COMTAB+10:
Contains the offset of the next character in BW_LBUF that is used by the next call to the BW_CRNAME routine.

$731: BW_ZORIG (1.4) COMTAB+11:
Contains the origin address of BW-DOS ($700).

$733: BW_DATER / $736: BW_TIMER (1.4) COMTAB+13:
These are the main variables for date and time (3+3 bytes). The format is: day, month, year, hour, minutes, seconds. This variables are used by the BW_GETTD and BW_SETTD routines and they are read to set date and time for a file.

$739: BW_ODATER / $73C: BW_OTIMER (1.4) COMTAB+19:
These variables have the same format like BW_DATER and BW_TIMER. They are used at file copy to keep the date and time of a copied file (see BW_TDOVER).

$73F: BW_TDOVER (1.4) COMATB+25:
This flag determines if a modified file gets the current date and time from the BW-DOS clock ($00), or the date and time set in BW_ODATER and BW_OTIMER ($FF). It is used at file copy to preserve the date and time of a copied file. This flag must explicitly be reset after files are copied.

$740: BW_TRUN (1.4) COMATB+26:
Contains the RUN address of the last program loaded by the command processor or the XIO 40 command.

$744: BW_SMEMLO (1.4) COMTAB+30:
Contains the value of MEMLO ($2E7) at startup of BW-DOS. If the current value of MEMLO is different to the value of BW_SMEMLO then there are resident programs are installed between BW_SMEMLO and MEMLO.

$746: BW_INCMND (1.4) COMTAB+32:
This flag determines, if the command processor or a cartridge is started at reset (warm start) e.g., pressing the <RESET> key. If the value is $FF then the command processor is started after reset.

$747: BW_COMFNAM (1.4) COMTAB+33:
This is a buffer for the complete device and filename specification. It is used for output by the BW_CRNAME routine and used as input by the BW_DIVIO routine. It is 28 bytes long. If parameters are read from the command line with the CRNAME routine then this buffer always start with Dn:. If other parameters than file names are read then the parameter starts at BW_COMFNAM+3.

$763: BW_RUNLOC (1.4) COMTAB+61:

Contains the address that is used by the command RUN without parameter. This address is changed if a program is loaded, or by the program RUN with an address parameter.

$765: BW_LBUF (1.4) COMTAB+63:
This is the buffer for the command line. The whole command line is stored here before a command is executed. This buffer is 64 bytes long.

$7A6: BW_CURDRV (1.4) COMTAB+128:
This address contains the drive specification of the current drive of the command processor in the format Dn:<EOL>. BW_CURDRV+1 contains the current drive number n in format ATASCII e.g., $31 for drive 1. It is used by BW_CRNAME as a prefix to store a parameter in BW_COMFNAM if the drive was not specified with the parameter.

$7AA: BW_DOSDRV (1.5) COMTAB+132:
This byte contains the number of the DOS drive e.g., 1-4 or 8. If set to 0 then no DOS drive is active and commands are only searched at the current drive. If set to an invalid drive number e.g. 5 then error 160 ($A0) is returned if a command cannot be found on the current drive. The value can be set with the command DOSDRIVE. It is used by the command processor and certain commands like MAN to search specific files on the DOS drive.

$7AB: BW_EXTBAT (1.5) COMTAB+133:
The following 5 bytes contain the standard extension for batch files used by BW-DOS, including the dot and terminated with end of line e.g, .BAT<EOL>.

$7B0: BW_EXTCOM (1.5) COMTAB+138:
The following 5 bytes contain the standard extension for executable files used by BW-DOS, including the dot and terminated with end of line e.g., .COM<EOL>.

$7B5: BW_STDPAT (1.5) COMTAB+143:
The following 4 bytes contain the standard file pattern for all files used by BW-DOS including the terminating end of line e.g., *.*<EOL>.

$7CB: BW_DFMSDH (1.5) COMTAB+165:
Entry point to a 16 bytes disk drive handler. The address of this handler is stored in HATABS by the DOS initialization routine.

$7E0: BW_DINT (1.5) COMTAB+186:
Contains an instruction to jump to the initialization routine of BW-DOS e.g., all working directories are reset.

COMTAB Access

If you want to program compatible to SpartaDOS then you must access `COMTAB` variables by relative addressing using the following pattern. In this example the command line is parsed for one parameter with CRNAME and the SIO vector is used. Furthermore XDIVIO is used in case of an error to stop batch processing. An example could look like this:

```
…
* set crname
            lda DOSVEC
            clc
            adc #$03 ; CRNAME = COMTAB+3
            sta crname+1
            lda DOSVEC+1
            adc #$00
            sta crname+2
* set sio
            lda DOSVEC
            sec
            sbc #11 ; SIO = COMTAB-11
            sta lsio+1
            lda DOSVEC+1
            sbc #$00
            sta lsio+2
…
            jsr crname
…
            jsr sio
            tya
            bmi error
…
* call CRNAME
crname      jmp $ffff
* call SIO
sio         jmp $ffff
* error handling
error       ; print error message in any way
…
* stop batch
            ldy #$08 ; XDIVIO = COMTAB+8
            lda (DOSVEC),Y
            sta xdivio+1
            iny
            lda (DOSVEC),Y
            sta xdivio+2
            ldy #$01
xdivio      jsr $ffff
* back to DOS
            jmp (DOSVEC)
```

The same example using absolute addresses to COMTAB:

```
…
BW_CRNAME   equ $729
BW_SIO      equ $71b
BW_FAIL     equ $70f
…
            jsr BW_CRNAME
…
            jsr BW_SIO
            bpl sio_ok
            tya
            jmp (BW_FAIL)
sio_ok
…
```

The jump to the BW_FAIL routine prints the standard error message to the screen (error code in register A), aborts a running batch process, and at the end it jumps to the command processor.

Note that this example is not compatible with SpartaDOS. It only runs with BW-DOS version 1.4 and greater. For this reason, you should check BW_DOS for ‚BW' and BW_VERS for being greater equal $14 or $15 depending with what version the variable is available.

Errors

This chapter gives a list of all error codes (decimal and hexadecimal numbers), and their descriptions. All error code less than 128 are used by other programs (for example Basic), so these error codes are not returned by BW-DOS.

Not all error codes have their origin in BW-DOS. There are errors caused by CIO, SIO, and other device drivers e.g., for screen, tape recorder etc.

The error numbers from 224($E0) to 255($FF) are reserved for applications. They are not and will never be used by BW-DOS.

The first two codes are no errors. BW-DOS returns these codes if everything is OK. These codes are only returned if CIO routines are used.

001($01) OK - This code is returned by all device handlers if no error occurs.

003($03) Last Byte read - This code is returned by BW-DOS instead of the code 1 (OK), in if the last byte of the file was read. Next read operation will return the error code 136 (End of file). This code is not returned in the directory listing mode (aux1=6). This code is not associated with the error code 3 returned by Atari Basic (Value Error - see Atari Basic Reference Manual) or other programs.

128($80) Break - The user pressed the <BREAK> key during an input or output operation. The operation was canceled.

129($81) IOCB already open - A given IOCB can only be open once at a time. If you try to open an IOCB that is already open then this error is returned. Always close an IOCB before using it. Close operations on a closed IOCB does not return an error.

130($82) Non existent device - The specified device does not exist. Please check if the corresponding handler is installed. If you are accessing a disk file, try to add Dn: before the filename. This is always necessary if files are accessed from other programs.

131($83) IOCB write only - This error occurs if you try to read from an IOCB that was only open for write.

132($84) Invalid CIO command - There are commands that are specific to certain devices, such as the disk drive and screen handler. This errors occurs if either an operation is incorrect, or a specific operation is not supported by the specific device.

133($85) IOCB not open - This error occurs if a command should be executed that requires an open operation beforehand but there was no previous open operation.

134($86) Invalid IOCB number - If an IOCB number other than 0-7 is specified with a command then this error is returned.

135($87) IOCB read only - This error occurs if you try to write to an IOCB that was only open for read.

136($88) End of file - This is not a real error. This code indicates that there are no more bytes left to read from the file.

137($89) Truncated record - The line in a file is longer than buffer specified in the IOCB. This error can occur if reading an incorrect type of file.

138($8A) Device does not respond - The device is not connected correctly, it is turned off, or it does not exist. It is also returned if the device number is wrong e.g., disk drive number for a not connected drive, and if the device took more time to process the command than specified.

139($8B) Device NAK - The device does not support the command. This can occur if a disk drive is configured that does not support the configuration command e.g., read the Percom block from an unmodified Atari 810 or 1050, or format a disk with medium density on a Atari 810 drive.

140($8C) Serial framing error - This error can be caused by a tape recorder, damaged device, bad or broken connection to the device, and non-matching data rates. (START and or STOP bit mismatch).

141($8D) Out of screen - The position of cursor is should be set out of screen.

142($8E) SIO overrun - These errors may be caused by tape recorder, or by damaged input/output device, computer, or cables. The data register is not cleared in time by reading the data and it is overwritten by incoming new serial data.

143($8F) SIO checksum error - These errors may be caused by tape recorder, or by damaged input/output device, computer, or cables. It means that the calculated checksum does not match the received checksum value. The data should be considered corrupt.

144($90) Device done error - The device cannot execute the command. The most common reasons are: writing to a write-protected disk, accessing a non formatted disk, there is no disk in the drive, or a bad sector on the disk.

145($91) Verify error or illegal screen mode - There are two reasons, either there was a difference at write and verify in verify mode or the selected screen mode is not supported e.g., the original 400/800 machines have no graphics modes from 12 to 15. Open the screen in one of these modes on a 400/800 results in this error.

146($92) Function not implemented - This error is returned if an unsupported operation is send to a device e.g., reading data from the printer or writing data to the keyboard.

147($93) Insufficient RAM - The selected command or operation needs more memory than available.

148($94) Invalid disk format - The disk is not a SpartaDOS compatible disk. You can only use disks created with the FORMAT command in BW-DOS or SpartaDOS 2.0 and later.

150($96) Directory not found - A specified directory in the path does not exist.

151($97) File or directory already exists - A new file or subdirectory should be created with the same name as an already existing file or subdirectory in the same directory.

152($98) Not a binary file - The file is not a DOS loadable program file.

156($9C) Parameter or Cartridge error - The CAR command does not detect a cartridge, or a wrong parameter was passed to a command, or an expected parameter is missing.

160($A0) Drive number error - The specified number for a disk drive is not supported. BW-DOS only supports drive numbers 1, 2, 3, 4, and 8.

161($A1) Too many open files - There are too many open files. BW-DOS can open up to 5 files at the same time, but only one of them can be open for directory listing (aux1=6).

162($A2) Disk full - There are no more sectors left on the disk to write a file. The file was only partly written and must be manually deleted.

163($A3) Invalid wild card in file name - A wild card * or ? was used for a name of a newly created file or directory.

164($A4) File is protected - The file cannot be changed because it is protected. See commands PROT and UNPROT.

165($A5) Illegal file name - Please check the file specification. File name contains illegal characters. Only upper letters, numbers, and the underscore are supported for file name.

166($A6) Illegal file position - The file position given to the POINT operation is greater than 8 megabytes, or it is greater than the length of the file if it is open in read-only mode (aux1=4). This error can also occur if you try to copy a file that contains non-allocated sectors. While working with Atari DOS 2 compatible disks using the MENU program, this error indicates that the file number in a sector is incorrect. This error may also be caused by a non-compatible disk format, or by a corrupt disk structure.

167($A7) Directory is not empty - The directory specified for the command RD (remove directory) is not empty, so it cannot be deleted.

168($A8) Command not implemented - BW-DOS does not support the command code. Please check the command code for the CIO or the XIO command. This error can also occur if a special functions of SpartaDOS are used that are not supported by BW-DOS.

169($A9) Directory full - There is no more space for a new directory entry. BW-DOS directories can contain up to 1424 files. Atari DOS 2 compatible disks (supported by the MENU program only) may contain up to 64 files.

170($AA) File not found - The specified file or directory does not exist. This error can also occur if a protected file or subdirectory should be deleted, or a subdirectory should be renamed.

182($B6) Path too long - The path of a file is too long to be handled by the command. The length of a path can by limited by a command e.g., recursive copy can only handle paths including filename und drive specification with a maximum length of 128 characters. Furthermore the path depth can be limited e.g., recursive copy only supports path depth to a maximum of 20.

History of Changes

Changes from Version 1.3 to 1.4

Command Processor

- Needs only 16kB (not yet for all external commands)
- Starts always command processor at boot time. It does not automatically start a cartridge. Use command CAR to run cartridge.
- Sets left margin always to 0

Commands

- Command CREDIR is renamed MD
- Command CWD is renamed CD
- Command DELDIR is renamed RD
- Command ERASE is renamed DEL
- Command PROTECT is renamed PROT
- Command RENAME is renamed REN
- Command UNPROTECT is renamed UNPROT
- RAMDISK.SYS now supports Rambo 256k with main banking and Axlon extensions
- Additional commands

Internal

- Absolute address COMTAB (see chapter Page 7 and COMTAB)
- Change of sector allocation to improve performance at common overwrite cases

Changes from Version 1.4 to 1.5

Command Processor

- Integrated DOS drive (specific drive BW-DOS always looks first for commands)
- Command processor without any internal commands for maximizing user memory (low MEMLO)
- Reintroduce DOS vectors BW_DFMSDH and BW_DINT for compatibility with Atari DOS 2 e.g., the debugger BUG65 by OSS now works with BW-DOS 1.5 (compatibility was lost with BW-DOS 1.4)

Commands

- New external commands for internal commands with more features and better error handling: BASIC, CAR, CD, DEL, DIR, DIRS, LOAD, MD, PRINT, PROT, REN, RD, RUN, TYPE, and UNPROT
- New MAN command with new manual pages in separate directory >MAN> on DOS drive
- New commands CLS, PWD, DIRC, DIRL, SORTDIR, and VERS
- Support of parameter lists e.g.; `DIR *.BAT *.SYS`
- More robust installation and deinstallation of the resident commands CLOCK.SYS, RAMDISK.SYS, RTIME8.SYS , and XFSIO.SYS because BW_SMEMLO is used for installation check instead of fix addresses
- Adapt the function names of MENU to the new command names and display the hotkey of a function inverse
- New command ED, text editor based on JBW ED
- All commands except MENU running with only 16k (see command descriptions)
- COMP/COPY/DEL/MOVE/SORTDIR have new option /R for recursive compare/copy/delete/move/sort directory and option /Q for quite operation
- MOVE now moves multiple files matching the pattern
- FORMAT is now also able to initialize hard disks and non-standard disk drives
- RUN now can take parameters for reruns
- REN now can rename subdirectories

Internal

- Change of directory listing OPEN with AUX1(6) to support directories with extensions, see commands DIR and DIRS