flp-level-2

Manual

Computer Technik

JÜRGEN FALKENBERG
Thanweg 36
D-7539 Ersingen
☎ 07231/81058
fax 07231/81058

Manufactured and distributed under license of Jochen Hassler.
The manual was written in co-operation with Jochen Merz.

2nd Edition 2/92

INTRODUCTION

The QL user now has access to the much more powerful level-2 driver for data storage devices, which have been written by Tony Tebby for the GoldCard and Atari emulator, and have been adapted for the TrumpCard and SuperQBoard by Jochen Hassler. The new driver has been constructed in a better way, with a common logical part for the file management, which communicates by an uniform interface with the second part, which drives the physical medium.
A much more quickly handling especially for loading operations and the long missing subdirectories now are available. The SuperToolkit II commands DLIST, DUP, DDOWN (see TK2 manual) now may be used for a comfortable application of the new subdirectories. Additionally new are the extended file attributes backup- and update-date and version number. The file management itself has not been changed, so the user will become familiar with the level-2 drivers immediately.

Some of the controller-specific commands have been changed with the level-2 drivers. Due to the much improved read-/write-timing some configuration settings may need to be changed.

FLP_START ***time*** adjusts the delay for the spindle motor in units of 20ms, e.g.
FLP_START 30 will select 600 ms = 0.6 sec delay. Preadjusted is 30 (= 600 ms). This value is a good compromise for the most floppies. Modern 3.5" drives with good power suppy may allow down to 15 (= 300ms), older units and/or bad power supplies may need up to 40...50.
FLP_STEP ***sr1,sr2*** will adjust the steprate for the drives, one value for each, e.g.
FLP_STEP 3,6 3ms for flp1_ and 6 ms for flp2_. Valid steprates are 2, 3, 6 and 12 ms for the WD1772 controller and 20, 30, 6 and 12 ms for the WD1770. Preadjusted are 6ms, many (modern) drives will work also with 3ms. A faulty set steprate will lead to read-/write errors, so the steprate suggested by the manufacturer should be selected always.
FLP_SEC is no more available, the flp-level-2 always selects the highest security level.
FLP_TRACK ***num*** allows selection of the number of tracks, e.g.
FLP_TRACK 40 will define 40 tracks. The preset default is 80 tracks. Never format a disc with more than 80 tracks! The QL disk standard QL5A only supports disk with up to 80 tracks.

The error messages 'read/write failed' and 'files still open' are no more available. Now the usual QDOS messages are used. On a faulty write operation the QL now will give a loud noise.

SUBDIRECTORIES

Organising your files in directories is a very important discipline. Rather than having a disk with your correspondence, and another with your accounts, and another with your notes, these and other categories of files should be stored in their own directories. In QDOS terms, this merely means making the names longer by adding a bit at the front. Put 'letter_' at the beginning of all the files which are letters, 'acct_' at the beginning of all the files which are accounts etc.

```
letter_jones_doc
letter_smith_doc
letter_...          Quill letter documents
acct_jul88_aba
acct_...            Abacus accounts
```

Toolkit II users will already be familiar with this, but the version 2 directory drivers take this one stage further by allowing 'hard' directories to be created. These have some advantages. When you look at the directory of the disk, any files that you have put in a 'hard' directory will be invisible you will just see the directory itself. If you create two directories 'letter' and 'acct' on a floppy disk, and all your files are in either of these, then DIR flp1_ will just give you 'letter' and 'acct', while DIR flp1_letter will give you a list of all the 'letter' files. Directories can, of course, be created within directories thus making 'tree' structures. SuperToolkit II Directory Listing DIR and STAT have both been extended to give the true disk size and free space in 512 byte sectors, even for drives exceeding 16 Megabytes. DIR, WDIR and WSTAT all mark a directory in the listing by adding the '->' symbol to the end of the name. The commands are otherwise identical to the old SuperToolkit II versions.

There is just one command to create a new (sub-)directory. This is MAKE_DIR. It has the directory filename as parameter. As it can return a variety of errors, there is also a function to do the same operation: FMAKE_DIR and will return the error code or 0:

MAKE_DIR *filename* or **ferr=FMAKE_DIR (*filename*)**

Normal Error Codes are:
-7 not found, medium or drive is not available
-8 already exists, there is a directory or file of the same name
-9 in use, there is a directory or file of the same name
-15 bad parameter, the device does not support sub-directories

If there are any files which, by virtue of their names, would belong in the directory being made, then these files will be transferred to the new directory, even if they are open. Using directories helps to speed access to individual files and makes system maintenance easier. Directories can be RENAMEd, and files can be RENAMEd from one directory to another. A directory which contains no files or directories may be DELETEd.

FILE-ATTRIBUTES

Whenever a file, which has been modified, is closed, the file is marked with the date (and time) when it was closed. This is called the update date. It is calculated in seconds from the beginning of 1961. The Version II Directory Drivers maintain a file version number which is incremented by one when a modified file is closed. There is also a facility to set a 'backup' date in a file header to record the most recent backup copy of the file.
There are two new functions to complement the SuperToolkit II function FUPDT. All three return a floating point value. They can be used to find the update date, the backup date and the version number of a file which has already been opened. In this case the channel number should be specified (or it will default to #3). Otherwise, they can be used to open a file, find the date or version and close the file. In this case, the filename should be specified. The filename can be given as a string or a name in the usual fashion and will default to use the data default directory.

value=FUPDT (#channel) returns the update date of the file open to *channel*
value=FUPDT (*filename*) returns the update date of the file *filename*
value=FBKDT (#channel) returns the backup date of the file open to *channel*
value=FBKDT (*filename*) returns the backup date of the file *filename*
value=FVERS (#channel) returns the version number of the file open to *channel*
value=FVERS (*filename*) returns the version number of the file *filename*

Three procedures are provided to set the update date, the backup date and the version number of a file. Like the three functions for reading the dates and version number, they can be used with either a channel number (default #3) or a filename. The filename can be a name or a string and uses the data default dictionary.

SET_FUPDT #*channel* set update date to now
SET_FUPDT \filename set update date to now
SET_FUPDT #channel,*date* set update date to the given date (and time)
SET_FUPDT \filename,*date* set update date to the given date (and time)
SET_FBKDT #channel set backup date to now
SET_FBKDT \filename set backup date to now
SET_FBKDT #channel,*date* set backup date to the given date (and time)
SET_FBKDT \filename,*date* set backup date to the given date (and time)
SET_FVERS #channel preserve version number
SET_FVERS \filename preserve version number
SET_FVERS #channel,*number* set version number to the given value
SET_FVERS \filename,*number* set version number to the given value

Giving a date or version number with a value 0 will have the same effect as omitting it. Giving a date or number with a value -1 will have no effect on the file. If the update date has been set, then it will not be reset when the file is closed. If the version number has been set, then it will not be incremented when the file is closed.

RAM-DISK

The term "Ram disk" is a misnomer. It is used to denote a memory range that looks and behaves like a directory device. Since this device does not use mechanical elements with high mass it is the most quickly data storage device. The installation in the RAM means, of course, that any space taken by a Ram disk is not available to any programs executing. Furthermore, any data stored in a Ram disk will be lost when machine is turned off or reset! Much more powerful is the MOS-Drive Ram disk which is available for the special static RAM of our QL-ROM-Card, which does not loose it's data contents nor occupies memory of the QL standard RAM.

A Ram disk supports all of the normal disk operations:

COPY flp1_x,ram1_x copies file x from floppy 1 to Ram disk 1
DIR ram1_ will give a directory listing of RAM disk 1
SAVE ram1_myprog save the current SuperBASIC program as "myprog" in RAM disk 1

A **dynamic** Ram disk is created automatically the first time it is used. This type of Ram disk takes memory as required, and releases any memory as files are deleted or truncated. A **static** Ram disk is created by formatting it: the size, in sectors, is given in place of the usual medium name. This pre-allocates all the space that will be available in the Ram disk.
FORMAT ram2_*sect* removes the old Ram disk number 2 (if installed), and sets up a new Ram disk of *sect* sectors size. The Ram disk number should be between 1 and eight, inclusive, while the number of sectors (512 bytes) is only limited by the memory available. Using a static Ram Disk not only reduces the danger of heap fragmentation, but as provides higher access speeds during file creation. However, since it always occupies the maximum space you ever wish to use, it is much less flexible. A Ram disk may be removed by giving either a null name or zero sectors:
FORMAT ram1_ or FORMAT ram1_0

The Ram driver also includes a SuperBASIC procedure to change the name of the Ram disk device.
RAM_USE *flp* or **RAM_USE *'flp'*** These commands set the name of the Ram disk driver to "*flp*", so that all subsequent open calls for floppy disks will use the Ram disks instead. Any three letters may be used as a new device name, in particular
RAM_USE *ram* or **RAM_USE** will reset the driver to its normal state.

THE ATR-DEVICE DRIVER

ATRdev is a device driver for direct reading, writing and formatting disks in IBM- or ATARI format in the QL disk drives with the standard QL commands for file handling. The driver is included in the level-2 replacement eproms for the Sandy SuperQBoard and Miracle TrumpCard. ATR-level-2 now also may access and create IBM subdirectories and format bootable IBM system disks.

Initialisation of the driver and new commands

After the QL has booted only the new command *ATR_DEV* is available from the ATR. The driver will not be installed automatically to save memory.

ATR_DEV
installs the ATR driver (with up to four new drives atr1_ ... atr4_) and adds three additional new commands to Basic. If ATR has been installed already, *ATR_DEV* will reset the settings of the ATR floppy driver to their basical values (of *FLP_START* and *FLP_STEP*). Furtheron the driver will delete all disk informations of the actual disk so that they will be read in completely new at the next access like with a new medium.

ATR_USE [*nam*]
renames the ATR device name like used from *FLP_USE* or *RAM_USE*.

ATR_CONV *x*
sets the automatic character translation for file transfers from "flp" (QL) to "atr" (IBM/ATARI) and vice versa to on (*x*=1) or off (*x=0*). This is useful for the transfer of text files since special ASCII codes and linefeed differ between QL and IBM/ATARI. Of course the translation may not be used for code files and executable programs. The translation tables are included in the file "ATRconv_dat" of the enclosed disk and have to be loaded (ATR installed) before with *LRESPR flp1_ATRconv_dat*.
With *x*=2 the use of a second self-defined translation table will be selected. It may be created with "ATRconv_asm" of the enclosed disk.
If the translation tables are not available a "not implemented" error message will be given, if for *x*=2 no second table has been loaded a "bad parameter" error will be printed.

ATR_TRACK *num*
like with *FLP_TRACK* this command defines the number of tracks for formatting disks, 80 tracks is the predefined default.

Formatting an IBM/ATARI disk

ATR may be used to create an IBM- or ATARI-disk with the usual *FORMAT* command:

FORMAT atr1_[*name*] [+/#] [*]

The optional parameter *name* enters an IBM/ATARI Volume-ID, + or left out first flag formats an IBM disk, # is the switch for an ATARI-disk. The flag * will select single sided formatting, e.g. *FORMAT atr1_test#*.

The file "ATRconv_dat" also includes an original IBM boot sector of MS-DOS 3.3. If this file has been loaded an IBM disk will be formatted as a system disk automatically. You are free to include boot sectors of your own by modifying "ATRconv_asm".

In principle the disk will be formatted like described below:

- 512 bytes per sector, each set to $00.
- 9 sectors per track.
- up to 80 tracks (giving 720 or 1440 sectors).
- double density (like QL floppies).
- two file allocation tables (FAT) with 3 (IBM) respectively 5 sectors/FAT (ATARI).
- main directory for 112 files (7 sectors).
- single or doublesided.
- with or without Volume-ID (medium bame).
- IBM MS-DOS 3.3 system disk or IBM/ATARI data disk.

Subdirectories

Like with the new level-2 QL floppy driver *MAKE_DIR* (or Sysdef of QPAC2) creates a subdirectory on IBM/ATARI disks. The names have to be selected according to the IBM standard. They will be entered with the full pathname or by using the *DATA_USE*, *DDOWN* and *DUP* commands. Please note the following system-dependant differences between QL and IBM subdirectories:

- an IBM subdirectory already has to be created prior to using it. This restriction does not exist for QL subdirectories.
- the file length of an IBM subdirectory is preset to 2048 bytes on its creation. This is necessary since MS-DOS does not apply directory lengths. If new entries (exceeding 32) are added further 2048 bytes will be appended. This is exactly the number of file headers filling an IBM cluster.
- an new subdirectory immediately includes two files named "_." and "_.." which are necessary for the IBM directory management but of no meaning for the QL. They have to be available always and so can not be deleted, copied, renamed or overwritten; on viewing into them they seem to be empty.
- like with the QL only empty subdrectories (except of the "_." and "_.." file) may be deleted.
- the depth of directory levels is not restricted. Subdirectories may include further subdirectories and so on. In the practical use however the complete file and pathname may not exceed 36 characters (64 on MS-DOS). This should be sufficiant for at least three subdirectory levels, enough even for exotic IBM disks.

Transfer of IBM file names and file attributes

File transfer IBM -> QL

Since the QL filing system is more flexible than the IBM standard it is always possible to use the original IBM filename. The "." seperator between filename and extension will be replaced by "_". In a subdirectory level the full file and pathname will be shown always like used from the level-2 driver. On the QL side there will not be distinguished between pathname, filename, extension and wildcards. However ATR makes an internal note of the original pathname length and the filename extension, that the original filename automatically will be restored when writing the file back to an IBM disk. This information will not be lost for files copied to other level-2 drivers (win, flp, ram).

The original IBM file attributes are stored within the QL file and for information of the user additionally written as the QL file version number. For the restore of the file attributes to an IBM disk the internal saved data will be taken, so a version number changed with the *SET_FVERS* command will not make any damage. Files transfered from an IBM/ATARI disk always will get the QL file type 0, so they may not be started as a job by accident.

File transfer QL -> IBM

If the QL file originally comes from an IBM disk the complete original information will be restored. The header of a "real" QL file will be modified in an IBM header according to the following rules:

1. The pathname will be removed.
2. If the QL filename ends with a "_" followed by up to 3 characters these will be used for the MS-DOS filename extension.
3. Up to eight characters from the remaining QL name will be used for the IBM name.
4. Finally the complete name is changed to upper case letters.
5. The IBM file gets the following file attributes: archive bit set, normal unrestricted access (%00100000 = $20 = 32).

This name transfer is always possible and no "invalid name" error message will happen if more files with complex QL names are copied with *WCOPY*.
If you are not satisfied with the IBM names they may be renamed with *RENAME*, *WREN* or "Move" of QPAC2.

Additional information for the ATRdev

You will note that ATR needs some time to show the directory of a newly inserted disk at the first time. This comes from the MS-DOS disk management where most disk and file informations needed by the filing system are not written directly onto disk and have to be read explicitely. There is no counter for the number of free sectors and ATR (and any MS-DOS machine) has to read the FAT and calculate the number of free sectors. In addition the IBM main directory has a fixed length (7 sectors), there is an extra boot sector and so on... Any following access to the same disk will be at least as quick as used from QL disks.

MS-DOS does not support update counting of files. It may happen that a disk removed from the QL and modified on another QL (with ATR) or an PC computer and then reinserted to the first QL will refuse to show the newly saved files despite they are really stored on it. A PC reads a disk completely new if it has been removed which will be recognised from the disk change signal (DC) given by the floppy drive. The QL floppy controllers do not support this hardware signal and the QL always has to verify if still the same disk is in the drive (*FLP_SEC 3* with the level-1 flp drivers). ATR instead reads the first sector of the FAT, summarizes it and stores the sum for each drive seperately. If the disk has been modified this sum will have been modified with high propability and ATR reads the complete disk information newly. In case of doubt you always have the possibility to force ATR to read a disk newly by enetering the *ATR_DEV* command.

ATR supports IBM- or ATARI disks
- with 8 sectors per track (old IBM standard)
- with 9 sectors per track (standard) or
- with 10 sectors per track (ATARI special with 1600 sectors)

for reading and writing. The disk type will be recognised in the boot sector. Old IBM disks with 5.25" and 40 tracks only may be used if your drive supports double stepping. Older 80 track drives oftenly supply a jumper "DS" to select double stepping.

"ATRconv_asm" of the enclosed disk is the assembler source to create the translation tables "ATRconv_dat". The source includes comments. If you are little familiar with assembler programming you should have no problems to replace or change the lists or exchange the boot sectors.