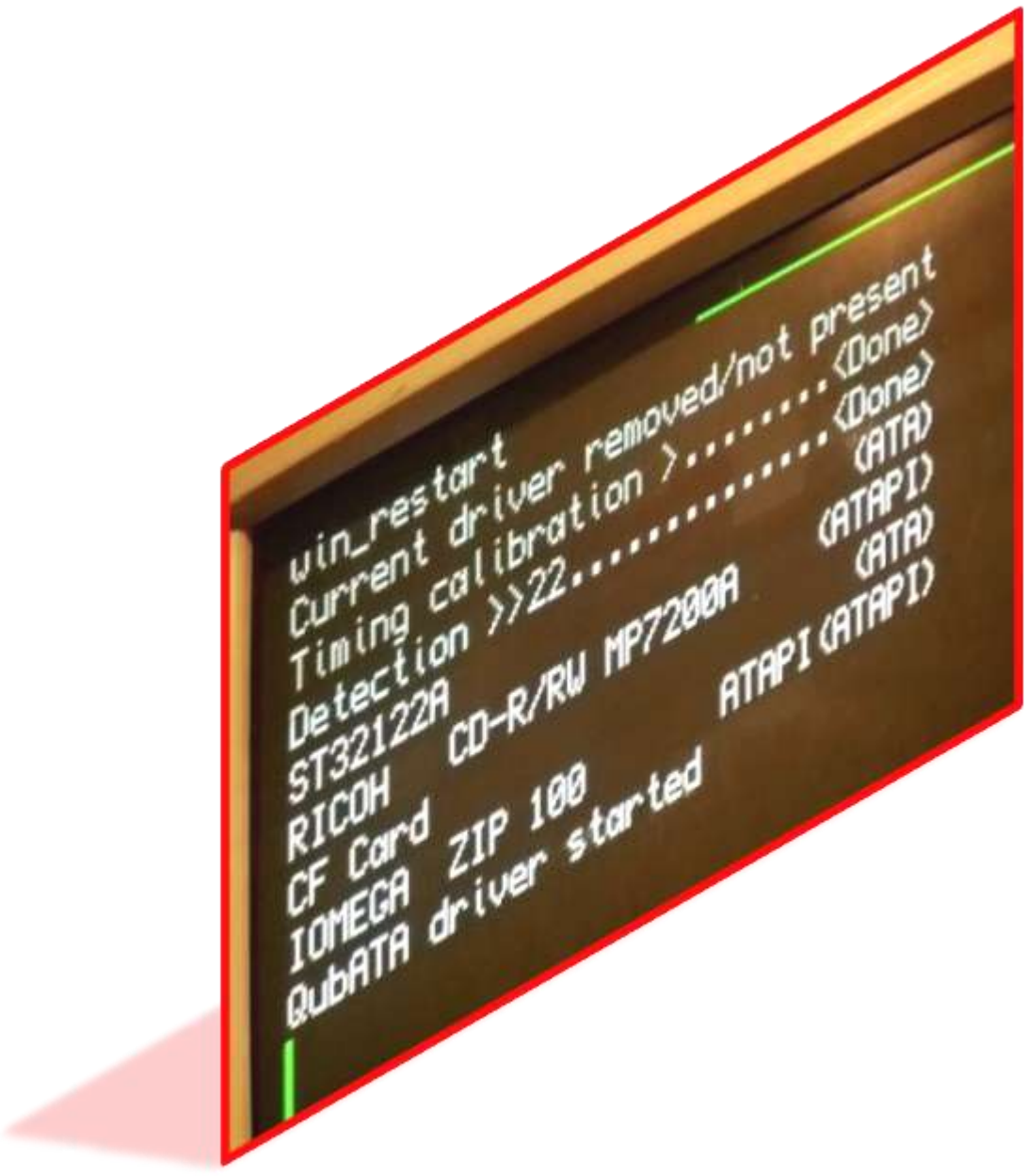
QUBATA DRIVER FOR QL

Manual (revised edition) – Alain HAOUI



CONTENTS

INTRODUCTION

This (2nd) manual revision is relative to the release of QubATA driver version 3.10. It comes in complement of concerned hardware documents (fitting interface, jumpers…) but covers all (old and new) software features for the driver, commands and associated utilities.

QubATA is a worth replacement driver of the original QubIDE written by Ph.Borman. It is now supporting a various hardware & software platforms.

Like the original QubIDE driver, QubATA uses its own powerful QLW1 format data structure for partitions/disks and offers a very complete raw data access for large variety of storage Medias (HD, CF/SD cards, ZIP, CDROM…).

This driver was initially started to improve detecting and handling multiple controllers/devices on QubIDE Expander hardware add-on. Finally, the almost entire driver was rewritten or optimized and some new features introduced. New code is arranged to get maintenance easier (at least possible) and enough slim to fit into limited ROMs space. Dependant stuff has been separated to permit implementation on several systems.

Major improvements :

- Very fast and reliable devices detection with efficient time calibration on the target system
- Better multi-controllers support with up to 16x2 devices on QUBIDE Expander (not very realistic but it is enough good for 4 devices, per example) – Need some hardware add-on
- Full compliance with ATA/ATAPI-4 protocol and removable medias support
- Fully functioning Trashcan, interfaces and associated utilities
- Enhanced direct raw access and “Alien” devices support
- Integrated commands for managing devices, partitions and format without need to any extra tool utility
- Complete set of vectorized routines to permit access to driver from other programs at low level
- CD Data support and Audio playing set commands (mimics of QPC CD-Audio module)
- Optimized and working slave blocks buffering with various sizes (up to 64 sectors per block)
- Full unsigned arithmetic permitting to handle big partitions/files up to 2GBytes (upper admissible limit for QL Files System).

REQUIREMENTS AND COMPATIBILITY

This driver is suitable for the original QubIDE Card Controller and compatible cloned interfaces on QL systems. It is also fully implemented for Qx0 platforms with ISA IDE Multi I/O cards.

To work properly, QubIDE based hardware must be fitted with the last GALs V2 codes and this is mandatory.

This driver should work under any QL OS : QDOS (any ROM version), Classic QDOS or SMSQ/E.

This driver still compatible with QubIDE partition format from V2.xx (may be from V1.56) and may be used on existing QubIDE disks and partitions. However, it would be better to initialize disks and make new partitions with the QubATA driver after having ensured it is working on the target system.

All original QubIDE commands are supported with this new driver. Some parameters were extended but still compatible with originals. Some other commands are new. All commands (old and new) are described in this manual.

Some utilities supplied with various original QubIDE packages should not be used with this driver (like partition_exe program, per example - see troubleshooting at the end of document).

INSTALLATION AND CONFIGURATION

The driver itself is supplied in two binary forms : one as ROM binary format (*_ROM) and the other as loadable RAM extension (*_REXT).

ROM version must be burned on an appropriate EPROM type according to hardware requirement. It should not be executed from file with any pseudo ROM emulation command (E.g. LOAD_EPROM).

RAM version should be configured correctly before using. This could be done with the standard “config” or “menuconfig” utilities.

Refer to the Release_Notes file supplied in the software package version for more information about configurable items and valid values for different systems.

RAM version can be started safely over any existent ROMed driver or another RAM version as per example:

LRESPR *flp1_QubATA_REXT* : REMark load and start new driver (TK2 present)

Or

***base*=RESPR(16128) : LBYTES *flp1_QubATA_REXT*,*base* : CALL *base :* REMark do it old way**

Driver will check configuration and simply complain and not load at all if something was wrong. You may have to correct some parameters in the config block and try again.

STARTING SEQUENCES

If started from RAM version (REXT file), the driver setup routine will remove firstly any WIN driver currently loaded in the system. If there are files open on any WIN drive, driver will not load.

Driver init routine proceeds to calibrate timing parameters according to your system characteristics and then starts scanning and detecting devices connected to IDE controllers.

The driver will report detected devices on each controller or expander position (from 1st to 16th slot) like that :

- (.) a dot indicates no devices detected on this position
- (1) indicates Master device detected here
- (2) indicates Master & Slave detected

Slaves alone without Masters are not supported and will not be detected.

After detection steps, driver will collect characteristics and report some information for each device.

Devices haven't passed diagnostic correctly will be reported.

Finally, it tries to link in first partition from first master device as WIN1_ if it is in valid format (QLW1).

When started from RAM version, output defaults to channel #0.

Refer to the illustration on the cover page to have an idea.

If driver is restarted later, all sequences are re-executed except that no WIN1_ drive will be automatically linked in again. So, if master 1 or partition 1 has troubles, we can still using driver and see what is happening after having reset all.

Important notice

With the 3.10 version, the driver has the ability to be started as WIN or QUB driver. If the driver was configured/started as QUB driver (and not as default WIN), please replace any WIN reference by QUB everywhere in the following sections.

MANAGING DRIVER SECTION

WIN_REMOVE — Remove driver

Syntax: **WIN_REMOVE** *Type:* **PROCedure**

Parameters:

- *none*

Description:

This command unloads all WINn provided they aren't in use and remove the current WIN driver from system if present. It frees all memories allocated by the WIN driver and unlinks all its service tasks (polling, scheduler…).

QubATA WIN driver can be restarted safely at any time without system reboot using the WIN_RESTART command (or after reset/reboot as usual from ROM).

After having removed the driver, all WIN commands, except WIN_RESTART, will report Not Found Error until the driver is restarted again.

Remarks:

This command will not return any error if no WIN driver is present or no WINn are linked in. It can be executed several times without risk.

WIN_ RESTART — Restart driver

Syntax: **WIN_RESTART** *Type:* **PROCedure**

Parameters:

- *none*

Description:

This command installs a new fresh WIN driver from start. It executes firstly an equivalent code to WIN_REMOVE command to make sure all traces relative to any previous driver are removed.

On restart, all steps realized by the driver on system boot are done (detection, calibration, diagnostic…) except that WIN1_ will not be linked in automatically. So, if master 1 or partition 1 has troubles, we can still use the driver commands and see what is happening after resetting all.

This command is the only WIN command still working after having removed driver with the previous WIN_REMOVE command.

This command executes same sequences as the ROM init routine at boot time and hence it produces same outputs and errors (on channel #0).

BASIC COMMANDS SECTION

WIN_DRIVE — Load partition

Syntax: **WIN_DRIVE *n,d,p***

Type: **PROCedure**

Parameters:

- ***n*** WIN number (1 to 8)
- ***d*** device number (1 to 32) - Masters (Odd), Slaves (Even)
- ***p*** partition number (1 to 32) or (0) for special partition Alien/Raw device

Description:

This command is used to link in a valid QDOS (QLW1) disk partition or a special RAW disk (Alien).

Examples:

WIN_DRIVE 1,1,1 : REMark link in WIN1_ from first partition on first master device
WIN_DRIVE 2,2,3 : REMark link in WIN2_ from third partition on second device (slave)
WIN_DRIVE 3,3,0 : REMark make WIN3_ as Alien partition for third device (master on slot 2)

Remarks:

This command will fail if ***n*** is already used for another present link or if the device/partition doesn't exist or wasn't detected correctly by the driver. An “Alien” disk hasn’t obviously to be initialized or partitioned as it is linked in with a temporary “fake” FAT anyway.

WIN_DRIVE — Unload partition(s)

Syntax: **WIN_DRIVE *n***

Type: **PROCedure**

Parameters:

- ***n*** WIN number (1 to 8) or (-1) for all

Description:

This command is used to unlink a disk partition which was loaded with previous form of this command. If the ***n*** parameter is given as -1, unlink all WINn present provided they aren't in use.

Examples:

WIN_DRIVE 1 : REMark unlink WIN1_
WIN_DRIVE -1 : REMark unlink all WINn present

Remarks:

This command will fail if WINn_ is In Use (files open).

When used to unlink all WINn_ (with ***n***=-1), it starts from top (WIN8_ to WIN1_) and will stop at the first WIN that cannot be unlinked (In Use). If no WINn_ found/present, it doesn't return any error.

WIN_ CTRL — Set drive options

Syntax: **WIN_CTRL *n,flags%***

Type: **PROCedure**

Parameters:

- ***n*** WIN number (1 to 8)
- ***flags%*** *: (0 to 15)*
  - ✓ bit 0 turns ON checksum for FAT if set to one
  - ✓ bit 1 turns ON name case conversion from parent path when creating files
  - ✓ bit 2 allows moving files having same path when creating directory
  - ✓ bit 3 moves files into trashcan when deleted

Description:

This command turns ON/OFF options for WINn_

WINn_ should have been linked in before using this command. Options will apply only for this partition and not for whole disk.

Flags are saved on the FAT disk and still in place after system stop/reboot until they are modified again by another WIN_CTRL command.

Current settings may be obtained with the WIN_CTRL% function command.

Examples:

WIN_CTRL 1,8 : REMark turn ON trashcan for WIN1_
WIN_CTRL 1,14 : REMark turn ON all options except FAT checksum
WIN_CTRL 1,0 : REMark turn OFF all options (default)

WIN_ CTRL% — Get current drive options

Syntax: ***flags%*=WIN_CTRL%*(n)***

Type: **FUNCtion**

Parameters:

- ***n*** WIN number (1 to 8)

Description:

This function returns current options setting for WINn_

See WIN_CTRL command for bits rules.

Examples:

***flags%*=WIN_CTRL%(*1*) : REMark get current flags for WIN1_**
WIN_CTRL *1*,(*flags%* ||*2*): REMark add name case conversion from parent path for WIN1_

WIN_ VER$ — Get driver version

Syntax: ***version$***=**WIN_VER$** *Type:* **FUNCtion**

Parameters:

- *none*

Description:

This function will return the WIN driver version present in the system.

Returned value is 4 chars length string (as '3.06').

Last known QubIDE version was 2.02.

QubATA version starts from 3.xx

WIN_ USE — Change driver use name

Syntax: **WIN_USE *[name$]*** *Type:* **PROCedure**

Parameters:

- ***name$*** *string of 3 chars length (E.g. 'FLP')*

Description:

This command modifies the WIN name used by the driver or reset to default name if no name passed as parameter (default is 'WIN').

Any error resets default.

Examples:

WIN_USE '*FLP*' : REMark change WIN to FLP, so FLP1_ will refer to WIN1_
WIN_USE : REMark reset default name (WIN again)

MAKE_ DIR — Make directory

Syntax: **MAKE_DIR *dirname$***

Type: **PROCedure**

FMAKE_ DIR — Make directory

Syntax: ***err*=FMAKE_DIR*(dirname$)***

Type: **FUNCtion**

Parameters:

- ***dirname$*** *name of directory to create*

Description:

These commands are implemented for completeness and are equivalent to those present in recent TK2 (from V2.24) or other extensions supporting level 2 directories.

These WIN commands shall work for creating directories for other drivers with level 2 support and vice versa. So, don't worry which ones are really used.

When creating a new directory, and depending on current options flags set for this WIN drive (see WIN_CTRL command bit #2), if there are existing files with the same path name, then these files can be moved automatically to the new created directory.

If this occurs, name case conversion from parent will be applied automatically to the moved files (see WIN_CTRL command bit #1).

In the case files moving option is not set for the drive and any of these files exist, then creating directory will fail with In Use Error.

Obviously, this option must have been set/unset according your wish before creating directory. Files moving can't be done automatically by the driver after directory has been created.

As usual, directory name may have up to 36 chars (without drive name WINn_) but it may be useful to leave some length for files name when added to directory.

Remarks :

You may have noticed that when using TK2 commands to create directory, if the file created can't be turned to directory type for any reason, a new file of "length 0" and type "data" will be left on the target device. QubATA commands don't.

If the ***dirname$*** ends with a trailing underscore, it will be removed.

PARTITIONS AND FORMAT SECTION

Normal BASIC FORMAT command has to be used to format any partition on any disk, provided partition exists and is healthy linked in with WIN_DRIVE command. BASIC FORMAT command will call automatically the Format trap routine supplied by QubATA driver.

FORMAT will only format partition on a disk which was previously initialized and partitioned. Each partition is formatted alone and becomes an independent WIN drive when linked in the system.

To initialize disk, create or modify partitions, refer to WIN_DISK command. See also TECHNICAL NOTES AND LIMITS section at the end of document.

Important : FORMAT will be done according to the current formatting parameters set with the following command :

WIN_ FORMAT — Set Format parameters

Syntax: **WIN_FORMAT *[flags%[,bsf%]]*** *Type:* **PROCedure**

Parameters:

- ***flags%*** *: permission switches (0 to 15), default (0)*
  - ✓ bit 0 => Lock, if not set format refuses
  - ✓ bit 1 => Quick, if set format quickly
  - ✓ bit 2 => Quiet, if set don't verbose
  - ✓ bit 3 => Query, if set don't ask confirmation

  -1 => only rename partition when format (preserve all data)

- ***bsf%*** *:* Block Size Factor *(0 to 8), default (0)*
  - 0 => keep current factor
  - 1 => optimize FAT size
  - 2 => 1 sector per block
  - 3 => 2 sectors per block
  - 4 => 4 sectors per block
  - 5 => 8 sectors per block
  - 6 => 16 sectors per block
  - 7 => 32 sectors per block
  - 8 => 64 sectors per block

Description:

This command will set current parameters applicable to next WIN_DISK or FORMAT commands for WIN drives.

These parameters are global for all WIN drives and remain actives until they are modified again or the driver restarted/system reboot.

flags% is used to set permission switches, while ***bsf%*** is used to set/modify the number of Sectors per Block when creating and formatting partitions. See TECHNICAL NOTES AND LIMITS section at the end of document for some details about this.

The Sectors per Block value (derivated from ***bsf%*** parameter) is normally used before making partition with the WIN_DISK command. However, it can be used when formatting to change actual value on existing partition without need to modify partition (unless you need change the partition size).

FORMAT doesn't create or make any change to partition limits but may change only block size and therefore the FAT size for partition.

Before issuing a FORMAT command, at least the following setting must have be done to unlock partitions protection :

WIN_FORMAT 1 : REMark allow format for WIN drives

FORMAT command has to be confirmed by the user before commit unless the Quiet flag was turned On with the WIN_FORMAT settings.

On normal formatting (not Quick Format), all blocks in the partition will be written/read/checked and any faulty block will be marked as unusable. Quick format recreate only the data structure without control of good/bad blocks.

Examples:

WIN_FORMAT 1 : REMark allow change or format partitions
WIN_DRIVE 2,2,1 : REMark link in WIN2_ from device 2, partition 1
FORMAT "WIN2_NameOfPart": REMark format partition and give it name
WIN_FORMAT 0 : REMark reset protection after format
…
WIN_FORMAT 3,4 : REMark unlock protection/format quickly, use 4 Sectors/Block
WIN_DRIVE 3,2,2 : REMark link in WIN3_ from device 2, partition 2
FORMAT "WIN3_Vroom": REMark format as quick as possible
…
WIN_FORMAT -1 : REMark allow name change only when format
WIN_DRIVE 2,2,1 : link in WIN2_
FORMAT "WIN2_NiceName" : REMark rename only WIN2_

QubATA driver will support existing partitions from QubIDE driver V2.xx (may be >V1.56) created with an appropriate working version of **partition_exe** utility supplied on QubIDE support disks.

Important : **partition_exe** utility needs some information from the original QubIDE Driver Definition Block in memory, so it can't/shouldn't be used if the QubATA driver is currently loaded.

There is no reason to use this utility any more as QubATA driver comes with all necessary commands embedded in the driver itself.

Following is the main command to deal with partitions under QubATA driver :

WIN_ DISK — Manage partitions

Syntax: **WIN_DISK *[#chan,]op$[,dev[,part[,size]]]*** *Type:* **PROCedure**

Parameters:

- ***#chan*** *: output channel, default (#1)*
- ***op$*** *: operation (4 chars significant, case insensitive)*
  - "***SHOW***" => show all detected devices
  - "***DETAILS***" => display details for device given in ***dev***
  - "***INIT***" => initialize device given in ***dev*** (destroy all data structures)
  - "***SUMMARY***" => list all partitions on device given in ***dev***
  - "***REMOVE*** " => remove (suppress) partition ***part*** on device ***dev***
  - "***CREATE***" => create new partition ***part*** on device ***dev*** with ***size*** MBytes
- ***dev*** *: device* number (1 to 32) Masters(Odd)/Slaves(Even)
- ***part*** *:* partition number (1 to 32)
- ***size*** *:* size of partition in MBytes (1 to 2047=2GO)

Description:

This major command will permit all operations to deal with devices and partitions.

Refer to the previous WIN_FORMAT command for some important settings.

SHOW and ***DETAILS*** are used to display all already detected devices and their characteristics.

SUMMARY is used to display the list of existing/created partitions on any disk and their geometry.

INIT must be used once for each disk to initialize partitions table before creating partitions. It scratches any existing partitions table and reset all data structures on the disk. It must be used only to initialize a new disk or to reset entirely/quickly an old disk.

REMOVE and ***CREATE*** are used to suppress existing partition or to create a new one.

An existing partition can't be modified (shifted, moved, resized…) and has to be suppressed and recreated again and this implies data lose.

When creating a new partition, this command will use the Block Size Factor set with the WIN_FORMAT command. If the current value is not suitable for the requested partition size, then the command will shift to the nearest adequate value.

To permit any change on partitions, the write lock must be unlocked with the command WIN_FORMAT (E.g. WIN_FORMAT 1).

For obvious reasons, some changes on partitions require that other partitions on the same disk aren't in use or currently linked in the system. For simplicity, do "WIN_DRIVE -1" command to unlink all WIN drive before using WIN_DISK command with INIT/CREATE/REMOVE operation.

Each modification has to be confirmed by the user before commit unless the Quiet flag was turned On with the WIN_FORMAT command.

Partitions are inserted in sequential order, so sufficient free place must be available when creating a partition in the middle of other partitions (E.G. create partition 3 when partitions 2 and 4 exist) but if this occurs, the command will report an error and will not overlap partitions.

Examples:

WIN_DRIVE -1 : REMark unlink all WIN drives
WIN_FORMAT 1,4 : REMark allow change partitions and set Block Size to 4 Sectors
WIN_DISK "REMOVE",1,3 : REMark remove part 3 on device 1
WIN_DISK "CREATE",1,3,128 : REMark create a new part 3 on device 1 with 128MBytes of total size
WIN_DRIVE 3,1,3 : load the new partition as WIN3_
FORMAT "WIN3_NewName" : REMark format WIN3_ and give it this new name

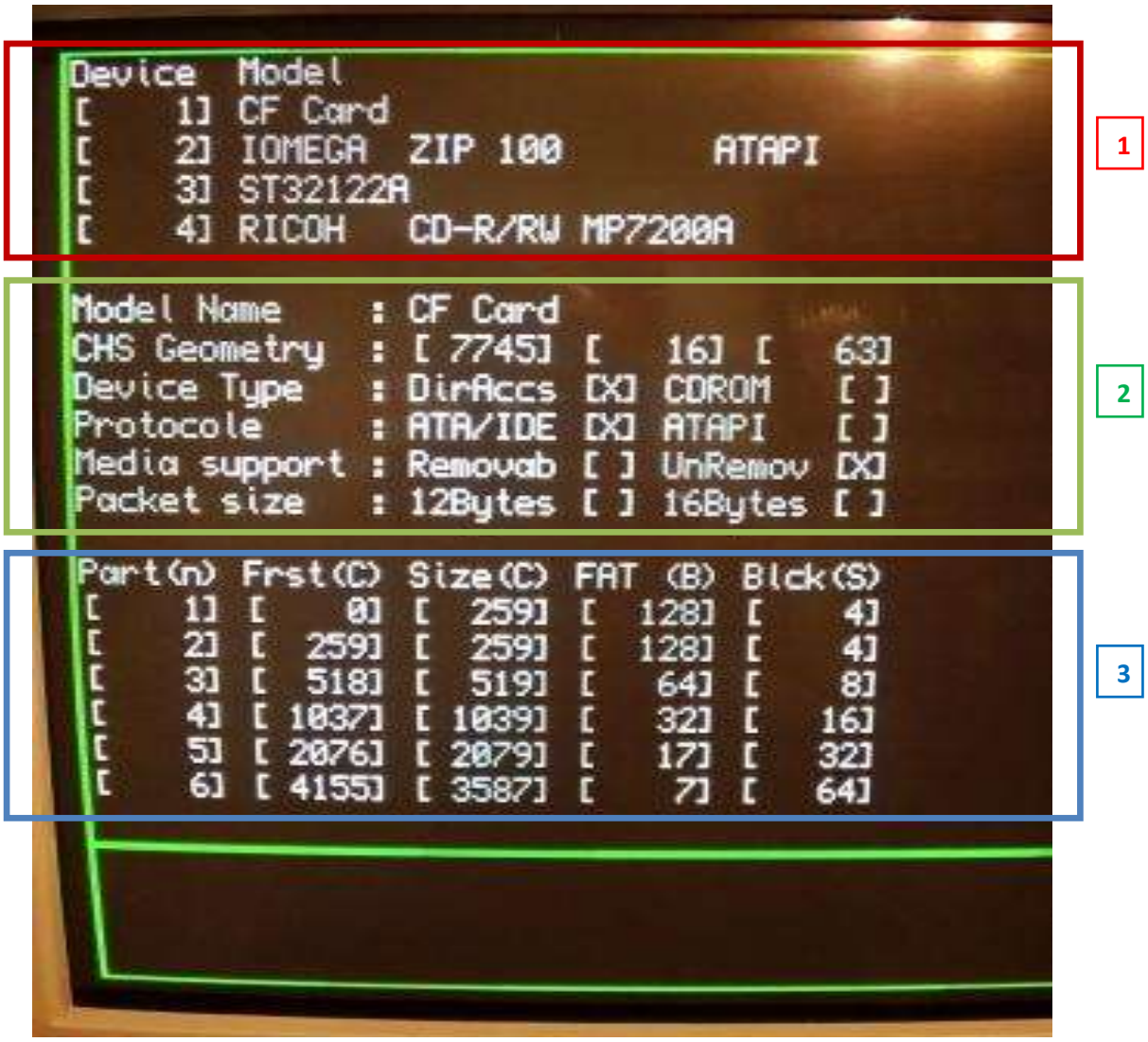


Illustration from some WIN_DISK command reports

First part is the output of **WIN_DISK *SHOW*** command. It reports 4 devices detected and for each: the device number and the manufacturer model name.

Second part is the output of **WIN_DISK *DETAILS,1*** command and it reports details about device 1. CHS geometry gives the total number of cylinders on disk (here 7745), the total number of heads (here 16) and the number of sectors per track (here 63).
The capacity of this disk is : 7745 x 16 x 63 = 7,806,960 Sectors (of 512 bytes) = 3,7 GigaBytes
The cylinder size is : 16 x 63 = 1,008 Sectors = 504 MegaBytes

Last part shows the output of **WIN_DISK *SUMMARY,1*** command which displays a list of all partitions on device 1 :

- Column 1 is the partition number
- Column 2 is the start cylinder number for partition
- Column 3 is the number of cylinders for partition (partition size in cylinders)
- Column 4 is the number of blocks for FAT (FAT size in blocks)
- Column 5 is the number of sectors per block (block size in sectors)

CDROM AUDIO COMMANDS SECTION

The QubATA driver comes with a complete new set of commands for playing Audio CDROM.

These commands are almost (mimics) very similar to CD Audio command in QPC except that, for simplicity reason and to save place in ROM, time parameters are always in Seconds units (no frames).

To make sense for these commands, a CDROM device unit should be connected to an IDE controller, a CD Audio (not Data) inserted in the unit and some speakers or headset connected to the CDROM panel unit. Sorry, no sound will be heard from the QL beeper.

Commands in this section are not available by default when QubATA driver is started and have to be explicitly installed if needed. This should be made with the following command WIN_CDEXT.

Also, these commands may not be available on hardware can't have a CDROM device unit like the Tetroid interface, per example. In this case, WIN_CDEXT will return "Not Complete" error.

WIN_CDEXT — Extend CDROM commands

Syntax: **WIN_CDEXT**

Type: **PROCedure**

Parameters:

- *none*

Description:

This command extends new CDROM Audio keywords and installs them in system. It must be executed once before using any of the following commands in this section.

CD_INIT — Initialize CDROM

Syntax: **CD_INIT *[dev|name$]***

Type: **PROCedure**

Parameters:

- ***dev*** is the device number for the wanted CDROM unit (1 to 32)
- ***name$*** ignored, only for compatibility with QPC1 (as MK said)

Description:

This command is used to select or change the CDROM device and must be used to initialize parameters in driver before calling other CDROM commands. If a ***dev*** number is given as command parameter, only this device will be checked and taken if ok, otherwise all devices will be checked and the first CDROM unit found is taken.

CD_PLAY — Play CD

Syntax: **CD_PLAY *[start[,end]]***

Type: **PROCedure**

Parameters:

- ***start*** First track number to play (default to 1 if not supplied)

  if negative or zero => play CDROM from the current position to the end

- ***end*** Last track number to play- inclusive (default to max=62 if not supplied)

Description:

This command is used to play an audio CD.

CD_ PAUSE — Do Pause

Syntax: **CD_ PAUSE**

Type: **PROCedure**

Parameters:

- *none*

Description:

This command is used to pause playing.

CD_ RESUME — Resume playing

Syntax: **CD_ RESUME**

Type: **PROCedure**

Parameters:

- *none*

Description:

This command is used to resume playing.

CD_ STOP

Stop playing

Syntax: **CD_ STOP**

Type: **PROCedure**

Parameters:

- *none*

Description:

This command is used to stop playing.

CD_ EJECT

Open Tray/Eject CD

Syntax: **CD_ EJECT**

Type: **PROCedure**

Parameters:

- *none*

Description:

This command is used to eject a CD (open the tray).

CD_ CLOSE

Close Tray/Load CD

Syntax: **CD_CLOSE**

Type: **PROCedure**

Parameters:

- *none*

Description:

This command is used to load a CD (close the tray).

CD_ ISREADY — Get Ready status

Syntax: ***current_ready%***=**CD_ ISREADY()** *Type:* **FUNCtion**

Parameters:

- *none*

Description:

This function returns current ready status (0=No, or 1=Yes=Ready).

CD_ ISPLAYING — Get Playing status

Syntax: ***current_play%***=**CD_ ISPLAYING()** *Type:* **FUNCtion**

Parameters:

- *none*

Description:

This function returns current playing status (0=No, or 1=Yes).

CD_ ISPAUSED — Get Pause status

Syntax: ***current_pause%***=**CD_ ISPAUSED()** *Type:* **FUNCtion**

Parameters:

- *none*

Description:

This function returns current pause status (0=No, or 1=Yes).

CD_ LENGTH — Get CD length

Syntax: ***cdlength%***=**CD_ LENGTH()** *Type:* **FUNCtion**

Parameters:

- *none*

Description:

This function returns total length of CD in Seconds.

CD_ALLTIME — Get CD elapsed time

Syntax: ***elapsedCD_time%***=**CD_ALLTIME()** *Type:* **FUNCtion**

Parameters:

- *none*

Description:

This function returns elapsed time of CD in Seconds (no Frames).

CD_ TRACK — Get track number

Syntax: ***trk_num%***=**CD_ TRACK()** *Type:* **FUNCtion**

Parameters:

- *none*

Description:

This function returns current track number being played.

CD_ FIRSTTRACK — Get first track number

Syntax: ***first_track%***=**CD_ FIRSTTRACK()** *Type:* **FUNCtion**

Parameters:

- *none*

Description:

This function returns the first track number on CD.

CD_ LASTTTRACK — Get last track number

Syntax: ***last_track%*=CD_ LASTTTRACK()** *Type:* **FUNCtion**

Parameters:

- *none*

Description:

This function returns the last track number on CD.

CD_ TRACKSTART — Get track start time

Syntax: ***startime%*=CD_TRACKSTART*(track)*** *Type:* **FUNCtion**

Parameters:

- ***track*** is the track number (1 to max=62)

Description:

This function returns the track start time in Seconds (no Frames).

CD_ TRACKLENGTH — Get track length time

Syntax: ***lenghtime%*=CD_ TRACKLENGTH *(track)*** *Type:* **FUNCtion**

Parameters:

- ***track*** is the track number(1 to max=62)

Description:

This function returns the track length time in Seconds (no Frames).

CD_ TRACKTIME — Get track elapsed time

Syntax: ***elapsedTRK_time%*=CD_ TRACKTIME()** *Type:* **FUNCtion**

Parameters:

- *none*

Description:

This function returns elapsed time within the current track. Time is returned in Seconds (no Frames).

CD_ FFWD — Fast Forward

Syntax: **CD_ FFWD**

Type: **PROCedure**

Parameters:

- *none*

Description:

This command is used to do a fast forward. Fast advance will continue until a next position command is issued.

CD_ FREW — Fast Rewind

Syntax: **CD_ FREW**

Type: **PROCedure**

Parameters:

- *none*

Description:

This command is used to do a fast reverse. Fast rewind will continue until a next position command is issued.

Sample CD Player

A slightly modified CD Player program from QPC1 (with Marcel Kilgus permission, Thanks) is included as demonstration to show how all above commands work with the QubATA driver.

After starting the program, use the numeric keypad to control the Player (switch NumLock ON or use numeric keys from 0 to 7) as showed in the program windows on the screen.

Playing may continue after quitting or sleeping the program until all given tracks played. If the Player is restarted or waked, it will resume at the current status.

This program is supplied in Turbo compiled task which could be directly Executed and in S*BASIC source which could be LRUNed under QDOS or EXecuted under SMSQ/E.

It needs TK2 under QDOS (Extended with TK2_EXT if necessary) and QubATA Audio CD commands Extended with the WIN_CDEXT command before launching.

DIRECT RAW ACCESS & ALIEN SECTION

QubATA driver provides an enhanced interface to access data on any disk or media support using direct addressing whatever the format of the disk (QDOS or Not) or the media type (CD, CF Card…).

In direct access mode, data is read/written by whole Block at once from/to specified Position on the disk. With QubATA direct access mode, Block is logical and its size is a multiple of sectors (of 512 bytes) which has to be specified when opening channel.

- ✓ Supported Block sizes are 512, 1024, 2048, 4096,..., 32768 Bytes.
- ✓ Blocks are numbered from 0 to $3FFFFF (22 bits)
- ✓ Position is always at Block boundary (block_number x block_size)

To do direct access, a special file have to be open with a special name : "***Wnd***" (default is "***W2d***") :

- ***W** : Special direct access name - mandatory
- ***n*** : Block Size Factor (2=>512,3=>1024,4=>2048,...,8=>32768 bytes) – default 512
- ***d*** : Swap bytes in words key (d => No Swap, D => do Swap) – default No

CDROM Data must be open with block size of 2048 Bytes (or multiple) and bytes swapping ***("*W4D"***).

When opening a disk for direct access from BASIC, the name should be enclosed in quotes, E.g.

OPEN #chan,"WIN2_*W4D"

Direct access is exclusive and can't be done if any file is open on the same WIN drive. While direct access is occurring, no other files can be open on the same WIN drive.

Before opening channel with the special file name, the WIN drive has to be linked in under 2 ways, depending if we access a valid QDOS partition or a whole "Alien" disk :

WIN_DRIVE 2,2,1 : REMark load WIN2_ from QDOS partition 1 on device number 2
WIN_DRIVE 2,2,0 : REMark load WIN2_ from device number 2 as Alien whole disk

At BASIC level, following commands can be used after having open channel ***#chan*** :

GET *#chan[\position],one_block_item$*
PUT *#chan[\position],one_block_item$*
***pos*=FPOS(*#chan[\position*])**

Take into account that the length of a string variable under BASIC can't exceed 32766 Bytes.

INPUT and **INKEY$** BASIC commands are not suitable for this usage. **PRINT** command may work if used properly. Read (**GET**) and Write (**PUT**) don't change current position when complete.

Writing a Block (**PUT**) will be flushed immediately to disk without any buffering delay.

Direct access should be finished with a normal CLOSE command as : **CLOSE *#chan***

At assembler level, following traps are implemented in a way to ensure compatibility with TK2.

IO.FSTRG (trap #3, D0=3) Read Block as String

This routine doesn't change current position pointer.
Compliant to QJUMP TK2 implementation for GET Basic Keyword.

Call parameters :

a1	base of buffer to be filled (updated to end+1 on return)
d2.w	length of buffer (Unsigned)

Return :

d1.w	number of bytes fetched

Rules :

- d2= block size (n x sectors) => copy block to (a1)+ and return d1= block size, d0=OK
- d2= 2 => block size to (a1)+, copy nothing and return d1=2, d0=OK
- d2= 2+block size => length to (a1), copy block to 2(a1)+ and return d1=2+block size, d0=OK
- Anything else : return d0=Bad Parameter error

IO.SSTRG (trap #3, D0=7) Write Block as String

This routine doesn't change current position pointer.
Compliant to QJUMP TK2 implementation for PUT Basic Keyword.

Call parameters :

a1	base of buffer to be send to disk (updated to end+1 on return)
d2.w	number of bytes to write (Unsigned)

Return :

d1.w	number of bytes sent

Rules :

- d2=block size (n x sectors) => copy block from (a1)+ and return d1=block size, d0=OK
- d2=2 => copy nothing (ignore) and return d1=0, d0=OK
- d2=2+block size => copy block from 2(a1)+ and return d1=2+block size, d0=OK
- Anything else : return d0=Bad Parameter error

IO.EXTOP (trap#3, d0=$9) Do externally operation

Call parameters :

a2	address of the user routine to be called
d1/d2/a1	parameters supplied to the user routine passed in a2

Return :

d0/d1/a1	as returned from user routine

When control is passed to the user routine, following registers are supplied as:

a0	pointer to the Channel Definition Block
a2	pointer to the FAT for the drive
a3	pointer to the Driver Definition Block
a4	pointer to the Physical Definition Block
a5	pointer to the base of the drive IDE registers (drive selected)
a6	base of system variables

User routine is called in Supervisor mode. All registers (other than d0/d1/a1) may be smashed.

FS.POSA (trap #3, d0=$42) Set absolute position

Call parameters :

d1.l	requested new absolute position (byte number - start from zero)

Return :

d1.l	new byte absolute position
	d1= block_number x block_size
	block_number (start from 0)
	block_size (512, 1024, 2048,... bytes)
d0=OK or EOF error	

Pointer is always set to the lower block boundary.

FS.POSRE (trap#3, d0=$43) Get current position

In direct access mode, this call is used to get the current position and no change is made to the current position. On entry, d1 is ignored for this call.

Return :

d1.l	current byte absolute position
	d1= current block_number x block_size
	block_number (start from 0)
	block_size (512, 1024, 2048, ... bytes)

Pointer returned is always at block boundary.

Sample CD Explorer

A sample rudimentary Data CD Explorer is included as demonstration to show how some of the above direct access commands work with the QubATA driver.

This utility is supplied in S*BASIC source. Some few lines at the beginning of the program have to be amended before running to suit your configuration. Menu extensions and TK2 (extended) are needed for this program.

This utility is used to browse an ISO 9660 DATA CD from CDROM reader unit connected to an IDE controller. It allows copying selected files to some destination device (E.g. RAM1_).

After starting, program displays some information about CD and a popup with a first list of files and directories from root level on the browsed CD.

Items with (->) flag indicate a sub-directory. Click on the item to open a new popup list with files under this sub-directory. Return to previous level is made with the item (<-)

Unflagged items indicate simply files. Click on the item to select the file (one file at one time) and copy it to the destination device as configured in the program.

TRASHCAN MANAGEMENT SECTION

QubATA driver comes with a fully functional trashcan started with QubIDE ROM from V2.0. Each partition can have its own trashcan. Trashcan is switched On or Off with the WIN_CTRL command.

As trashcan flag is set in FAT and flushed to disk, it will remain in place over load/unload partition, system reboot... until is changed again with a new setting.

With the trashcan active, deleted files are not really deleted but moved into trashcan hidden directory. When trashcan is switched back to Off, files already stored in the trashcan remain there but next files deleted are really suppressed definitely.

Files in trashcan are associated to their entry number and not to their name, so successive deleted files having the same name may reside together in trashcan as their entry numbers differ.

Directories are never moved to trashcan and obliviously can't be deleted if not empty.

Trashcan can't be accessed by the file system and has to be managed with a special utility program to show contents, recover deleted files or suppress definitely files for good. This can be made with the original ***trashcan_obj*** program but another tool comes with QubATA driver to deal with this feature.

This tool, called ***TrashMan_task*** (Trashcan Manager) still in some basic state (no PE, no colors…) but does fully the job and it can be used to :

- List/Recover/Suppress files in trashcan
- Export files from trashcan to another device (leave files in trashcan)
- Empty trashcan (suppress all files)
- Compress really trashcan (suppress holes & shrink directory => reduce size)
- Turn On/off options for the WIN drive (all options individually)

This program is menu driven and self explanatory. It is Turbo compiled with ToolKit runtime embedded. It requires only a decent/recent TK2 to be present. It could be executed as :

EX *WIN1_TOOLS_TrashMan_task*

Remarks :

Active Trashcan has to be delighted from time to time and its directory has to be compressed. Otherwise WIN drive will continue to grow even when unwanted files are regularly deleted.

TECHNICAL NOTES AND LIMITS

This driver shall work with any ATA or ATAPI device compliant to ATA/ATAPI-4 protocol. It is also removable Medias aware.

This includes hard disks, ZIP units, CDROM readers and CF Card memories with adapter. As mentioned, QubIDE based hardware must be fitted with the last GALs version 2 (mandatory).

Compatible devices must fire up in PIO modes 0, 1 or 2 even if they support higher mode 3 or 4. This is generally the case and there is no problem, but some existing (bad) devices (specially SD card adapters) fire up in mode 3 or 4 with using IOREADY line (data flux control on bus) which is not always supported on "old" IDE hardware. In this case, driver can do nothing and problems may occur in data transfer (probably device will get locked until is reset).

To use a CF Card memory on IDE controller connector, you need an "IDE to CF Card adapter" which can be easily found in the market at very reasonable price. Beware to choose a model with Master/Slave select (by switch or jumper) and a power connector (type Floppy 4 pins). Otherwise your hardware must support "Cable Select" feature and supply power on Pin 20.

To use SD cards memory, you need an "IDE to CF Card adapter" and a "True IDE CF Card to SD adapter" but this may not work at all, depending on used adapters because voltage ranges are different for CF and SD cards.

Please note that memory cards (CF, SD…) are not "Hot" removable Medias and should not be unplugged/plugged unless the system is really powered down to prevent device from any damage.

The following limits apply to this driver :

- 1 to 8 partitions can be loaded at the same time (WIN1_ to WIN8_) provided there is enough free memory to load FATs.
- 1 to 32 physical devices (16 couples of Master/Slave) can be handled on the original QUBIDE card provided that hardware was modified to use Expander. Please, ask for mod information (published information insufficient).
- 1 to 4 physical devices (2 couples of Master/Slave) are supported on Qx0 platforms on Primary/Secondary or Tertiary/Quaternary IDE controllers.
- Device number starts from 1 (Masters Odd, Slaves Even).
- 1 to 32 partitions can be used on each Media support. Partition numbers start from 1. Partition number 0 is a special case for use with "Alien" devices. The number of partitions doesn't impact the system performance as they aren't all loaded simultaneous.
- Each partition can be sized up to 2 GigaBytes depending on the number of blocks and the number of sectors per block used (see below).
- Each partition can handle up to 65275 different files in one FAT.
- Block size is independent for each partition and can be from 1 up to 64 sectors (512 bytes) per block.
- The biggest file size supported and working under QDOS File System may be a whole maximum partition size (minus FAT size) so approximately 2 GigaBytes.

The block size is set with FORMAT parameters using the WIN_FORMAT command. The block size can be changed after partition creation but before formatting.

The size of partition is defined when creating the partition with WIN_DISK CREATE command.

As the whole FAT of each partition is loaded into memory when the partition is linked in with WIN_DRIVE command, the size of the partition may have importance regarding the available memory on used system. Partition size and Block size (number of sectors/block) must be chosen carefully when creating and formatting partition. Any change for this implies a reset of partition and lose of all data on the concerned partition.

As general rules, consider following :

Each block in the partition, whatever the block size used, takes 4 bytes as 2 index entries in the loaded FAT in memory. Each index in the FAT is an unsigned short integer (0 to 65535 or $FFFF).

The size of FAT doesn't depend on the size of partition but mainly depends on the number of blocks in partition.

The size of partition (in bytes) depends on the number of blocks and the number of sectors per block.

So the equation will be :

- Take the smallest block size possible (don't waste bytes on disk) – nearest 2, 4 or 8 sectors/block
- Take the maximum number of blocks possible (to have the biggest partition size possible) – nearest 65535 blocks
- Produce a reasonable FAT size regarding available memory on the system and the number of FATs you have to load at one time – 128 or 256 KB per FAT is a good compromise.

Here is a little summary of different FAT size requirements regarding block sizes used :

- 8 KBytes per 1 MBytes of storage with 1 Sec/Blk
- 4 KBytes per 1 MBytes of storage with 2 Sec/Blk
- 2 KBytes per 1 MBytes of storage with 4 Sec/Blk
- 1 KBytes per 1 MBytes of storage with 8 Sec/Blk

- 1 KBytes per 2 MBytes of storage with 16 Sec/Blk
- 1 KBytes per 4 MBytes of storage with 32 Sec/Blk
- 1 KBytes per 8 MBytes of storage with 64 Sec/Blk

A partition of 2 GBytes size with 64 Sectors/Block may be made with a 256 Kbytes FAT size only, but the smallest file with 1 Byte per example will take a whole block of 64 sectors (32768 Bytes) of storage.

TROUBLESHOOTING

FAT check summing

This option set with the WIN_CTRL command for each partition is CPU consummating and may have some impact on the system performance. It must be clear that this option doesn't prevent FAT from being corrupted (by other programs or commands, per example) but it will only detect that FAT is corrupted. If this happens, try to reboot the system immediately without making any change to any file. But as the FAT is flushed regularly from memory to disk, there is really a little chance that the FAT on the disk could be safe and not corrupted.

When the FAT present on the disk partition becomes corrupted, it is too late and data is lost (unless using a special tool). You probably have to format the partition and restore data from the last backup you have made last evening (haven't you ?).

In the worst case with the first partition on the disk being corrupted, all partitions may be lost as the partitions table resides here. This may technically be improved but needs change to QubIDE partition format and compatibility will be probably broken.

After an observation period and if you don't use bad behaving programs, the FAT checksum can be reasonably turned Off. Major FAT problems observed are caused by electrical interferences particularly when used with a QPlane backplane.

Note also that when modifying partitions on a disk with the WIN_DISK command, checksum option will be unset automatically for the first partition of the concerned disk to permit writing of the partitions table. You have to set this option again on partition 1 if wished. This is not a bug but may be improved in the future.

Store utilities

Storing drives parameters inside SuperHermes or I2C on the Minerva MK2 ships are not supported by this driver. The main reason is that QubATA driver initializing and detection devices on startup are enough quick and reliable and doesn't really need this glue to bypass the startup delay and the geometry identification problem.

Another reason is the storing parameters only for one Master/Slave pair which is not adequate with Multi-controllers and Expander use. Also, saving drives parameters with removable Medias is not accurate and may produce unpredictable behaviors.

Other QubIDE utilities

Some of the software utilities supplied on QubIDE disks should not be used with QubATA driver specially those using information from Driver Definition Block in memory. As this has changed, compatibility can't be made in this way. This applies particularly to **partition_exe** and **WinEditor_obj** by example.

However, previous vectorized routines entries (link/unlink/rename) are maintained and still supported with this driver.

Trashcan_obj (used to access Trashcan) can be used safely with no problem, however it is more recommended to use the new **TrashMan_task** utility supplied with this driver.

Backup utilities shall still work normally but not all tested.

Other utilities may be updated and supplied in the future if still having any interest.

Also, a separate extension toolkit will be supplied to access major (old and new) driver features with complete programming interface description.

CREDITS AND PERMISSIONS

First thanks and credits go obviously to Nasta for designing QubIDE, Ph.Borman for writing the QubIDE driver and QubbeSoft/QBranch for producing, distributing, supplying manuals and maintaining QubIDE hardware over many years.

Thanks also to Dilwyn Jones, Rich Mellor, Marcel Kilgus and Wolgang Lenerz for feedbacks, helps and permissions.

Feedbacks, suggestions and comments are welcome via any channel (QL users list, Forums, Blogs…)

MANUAL REVISION HISTORY

Revision 1.00

First manual version. Need probably (surely) English review.

Revision 2.00

Revised manual version relative to V3.10 package.