SIDE / Incognito / Ultimate 1MB

APT Driver API

Technical Reference

(Draft 10)

Introduction

The APT hard disk partitioning system was devised to provide broad compatibility between a number of incipient Atari 8-bit hard disk interfaces. APT was originally devised for SIDE and IDE Plus, but it has since been adopted by the Ultimate 1MB and Incognito PBI hard disk controllers (the former working in conjunction with the SIDE cartridge). A version of the APT soft-driver for SIDE is also available for the MyIDE and MyIDE 2 interfaces. Both the APT and the corresponding driver API were designed by KMK with additional input from me and certain other Atari 8-bit developers. API extensions for Ultimate 1MB and Incognito were designed by me and Sebastian Bartkowicz. This document describes the specifics of the API common to all compliant interfaces. Deviations from or augmentations to the original design as they pertain to individual implementations will be discussed at the appropriate points.

Programmers should consider this documentation in conjunction with the APT Specification by KMK, hosted here:

drac030.krap.pl/APT_spec.pdf

This URL is a permanent link to the most current version of the APT specification.

It should be noted that information presented in this technical documentation includes material (in an abridged or edited form) previously published and written by KMK – specifically the DEVINFO.TXT, DISKINFO.TXT and XDCB.TXT files supplied with the IDE Plus 2.0 APT BIOS updates.

In this text, we will refrain from covering the entire gamut of SIO commands which can be used with the disk devices, such as sector read/write, status, etc. Instead, we will describe the calls which have specific relevance to APT.

The XDCB

IDE Plus, SIDE, and other compliant implementations support a DCB extension called XDCB. The extension allows programs to address sectors on large disks (where the sector number is a 32-bit value) and on 65C816 machines it also allows the transfer of sectors to and from the entire 16 MB address space (assuming the disk handler implements 65C816 instructions to do that).

A program requests XDCB operation by setting the 7th bit of DDEVIC. The device code for a disk is then $B1, not $31. Similarly, when using the physical disk handler (device $20), the XDCB device code will be $A0.

When either of these device codes are detected, the following DCB structure is assumed:

$0300	DDEVIC
$0301	DUNIT
$0302	DCMND
$0303	DSTATS
$0304/5	DBUFA - low 16 bits
$0306	DTIMLO
$0307	DUNUSE
$0308/9	DBYT
$030A	DAUX1
$030B	DAUX2
$030C	DAUXA
$030D	DAUXB
$030E	DBFX1
$030F	DBFX2

DAUX1/2 contains the low word and DAUXA/B contains the high word of an 32-bit sector number.

The DBFX1 $030E contains the highest byte of the buffer address. On a 6502 machine, this byte should be kept zero or an error will occur.

The DBFX2 $030F should be kept zero for upward compatibility.

The 6502 XL OS uses the addresses $030C-$030F as some temporary registers of the tape recorder handler. Thus all these should be always carefully initialized before calling the SIO, to make sure that there are no random values put there in meantime by another OS call.

Commands

The SIDE API introduces four new SIO commands over and above those provided by the Atari Operating System. The new commands are:

Code	***Command***	***ASCII***
$46	Force Media Change	(‘F’)
$48	Host File System	(‘H’)
$4D	Mount Partition	(‘M’)
$53	Device/Disk Status	(‘S’)
$6E	Device / Disk Info	(‘n’)
$EC	Get Device Info	

The Device / Disk Info command context-dependant on the contents of DDEVIC ($300). Two devices are supported:

Code	***Purpose***	***Meaning of command code $6E***
$31	Logical Disk (i.e. partitions)	DISKINFO
$20	Physical disk (i.e. raw hard disk)	DEVINFO

In addition, bit 7 of DDEVIC when set invokes use of the XDCB (see later). Therefore, device codes $B1 and $A0 are also permitted.

DEVINFO (Device Info, Command $6E)

We'll first consider the DEVINFO command, invoked with $6E in DCOMND ($302), and $20 or $A0 in DDEVIC. (Note that the DEVINFO command is considered the DISKINFO command when DDEVIC contains $31 or $B1: see the second on DISKINFO.)

The DCB should be set up as follows:

Address	***OS Label***	***Value***
$0300	DDEVIC	$20
$0301	DUNIT	unit number (1 = master, 2 = slave)
$0302	DCMND	$6E
$0303	DSTATS	$40
$0304/5	DBUFA	buffer address
$0308/9	DBYT	512 (buffer length)

The returned block consists of 512 bytes:

Offset	***Function***
$00	Bus ID
$01	Controller ID
$02	Device ID
$03	Feature Table
$04-$05	Reserved
$06-$07	Sector Size (low / high)
$08-$0B	Number of Sectors (32 bit unsigned little endian)
$0C-$0F	Reserved
$10-$37	Device Name (40 bytes, ASCII, NUL padded)
$38-$5F	Hardware / Driver Name (40 bytes, ASCII, NUL padded)
$60-$1FF	Reserved

The first three bytes are intended for device identification:

The *Bus ID* is an arbitrary number identifying the I/O bus the device is physically attached to. $00 represents the Serial Bus (SIO), $01 the parallel bus, and $02 the SpartaDOS X internal LSIO interface. SIDE returns $02 here.

The *Controller ID* is a number identifying the controller attached to the bus. SIO devices should return $00 here. For PBI devices the PBI device ID is returned here; in fact this is the number of the bit in the PBI enable register that activates the device. The value for PBI devices is thus ranged from 0 to 7.

The *Device ID* is a number identifying the physical device itself. For IDE/ATA this will be $01 for the master drive and $02 for the slave drive. Floppies return unit number, and if a disk drive contains more than one physical drive, the same DEVICE ID should be returned for all of them.

Feature Table is a bit field describing the facilities implemented by the particular hardware / driver combination. Bit meanings are as follows (a set bit in the relevant position indicates that the described feature IS implemented):

Bit 0: Dynamic partition mounting
Bit 1: Drive mapping
Bit 2: ATR mounting

Bit 3: ATR mounting uses FAT32 instead of APT partition (always clear if bit 2 is clear)

Note that the information returned in the feature table dictates the capabilities of the MOUNT command. No assumption should be made regarding the capabilities of a particular driver based on the hardware configuration: always check the feature table first.

The next two bytes are reserved and should not be used.

The next two bytes contain information about the physical sector size in bytes. The order is little endian (low / high).

The next four bytes contain information about the total sector count for the specified device. This number multiplied by the sector size should yield the device's capacity in bytes as a result. The order of bytes of this field is little endian.

The following four bytes are reserved and should be zero.

Following this is a 40 byte buffer containing a NUL-padded device name. This is the same string the SIDE driver reports when it finds a compact flash card when it first initializes.

Finally, we have another 40 byte string describing the hardware / driver combination, NUL-padded like the device name. For example:

“Ultimate 1MB PBI BIOS for SIDE”

The rest of the buffer is reserved for future use.

If DEVINFO returns error 139, it may mean that the device doesn’t support the command.

Device Status (Command $53)

The status command (code $53), when issued in the context of device $20 or $A0, tests for a device's presence.

The DCB should be set up as follows:

Address	***OS Label***	***Value***
$0300	DDEVIC	$20 / $A0
$0301	DUNIT	unit number (1 = master, 2 = slave)
$0302	DCMND	$53
$0303	DSTATS	$40
$0304/5	DBUFA	buffer address
$0308/9	DBYT	4 (buffer length)

The returned block consists of four bytes:

$40, $00, ID, Version

Where "ID" is the handler identification number, and version is the BCD-coded version number (e.g. $10 = 1.0).

Handler Identification Numbers have the following meanings:

IDE Plus 2.0:	$01
IDEa (APT):	$02
Ultimate 1MB	$03
Incognito	$04
SIDE1	$05
SIDE2	$06
Colleen	$07
MyIDE	$08
MyIDE II	$09

DISKINFO (Disk Information, Command $6E)

The DISKINFO command is used to obtain information about logical disks (partitions), and is invoked with $6E in DCOMND ($302), and $31 or $B1 in DDEVIC.

If error 139 is returned, it may mean that the device doesn't support this command.

Generally it is not expected that this command could be supported by floppy drives, and so it is advisable to send a standard "Read PERCOM" ($4E) command first, and then try out the DISKINFO only if the PERCOM returns $01 as the first byte of the returned data (i.e. if the device's reply allows to assume that it is a hard disk).

The correct DCB parameters for DISKINFO are as follows:

Address	***OS Label***	***Value***
$0300	DDEVIC	$31
$0301	DUNIT	unit number
$0302	DCMND	$6E
$0303	DSTATS	$40
$0304/5	DBUFA	buffer address
$0308/9	DBYT	64 (buffer length)

The returned block consists of 64 bytes:

Offset	***Function***
$00	Bus ID
$01	Controller ID
$02	Device ID
$03-$04	Device Sub-ID
$05	Reserved
$06-$07	Sector size (low / high)
$08-$0B	Number of sectors (32 bit unsigned little endian)
$0C-$0F	Physical starting sector number (or first cluster, if ATR). 32 bit.
$10-$37	Partition name
$38-$3F	Reserved

The first five bytes are intended for device identification. For example, DOS may want to know if two partitions mounted on different drive numbers aren't in fact the same partition mounted twice by a mistake.

Bus ID, Controller ID and *Device ID* are identical in format and meaning to those returned by the DEVINFO command.

The *Device Sub-ID* is a unique number that identifies a partition. When, for example, partition number 10 is mounted on D1:, the DISKINFO returned by D1: should have a value of 10 in this field. This numbering should start at $01. $00 means that there is no SUB-ID. For mounted disk images (see the *Mount Partition* command), the Sub-ID will always read $FFFF.

All five identification bytes constitute a hierarchy, with BUS ID being the most significant value and the DEVICE SUB-ID being the least significant value. In other words, a device can be uniquely identified as DEVICE.SUB-ID of DEVICE.ID of CONTROLLER.ID of BUS.ID.

Next byte is reserved and should read $00.

Next two bytes contain information about sector size in bytes. The order is little endian (low / high).

The next four bytes contain information about the total number of these sectors in the partition. This number multiplied by the sector size should yield the size of the partition in bytes. The order of bytes of this field is little endian (low first).

The next four bytes represent the starting LBA sector number of a partition on the physical disk. If the mounted volume is an ATR disk image in the FAT32 area of the disk, this value will be a 32-bit cluster number, corresponding to the cluster number in the disk image's FAT32 directory entry.

Next is a 40 byte NUL-padded partition name. If the partition has no name, this field should contain 40 zeros. For mounted ATRs, a NUL string will (currently) be returned.

The last 8 bytes are reserved for future definition and should contain zeros.

Disk Status (Command $53)

The status command (code $53), when issued in the context of device $31 or $B1, tests for a partition or disk image's presence.

The DCB should be set up as follows:

Address	***OS Label***	***Value***
$0300	DDEVIC	$31 or $B1
$0301	DUNIT	drive number
$0302	DCMND	$53
$0303	DSTATS	$40
$0304/5	DBUFA	buffer address
$0308/9	DBYT	4 (buffer length)

The returned block consists of four bytes. The first describes the partition or ATR density. Bit usage is as follows:

7: 1 = Enhanced density ATR (128bps and size = 1040 sectors)
65
---
00 – 128 bytes per sector
01 – 256 bytes per sector
10 – 512 bytes per sector

4: 1 = Motor on (ATRs only)
3: 1 = Write protected

The second byte is the value of the IDE error register XORed with $FF. For disk images, the second byte always reads $FF.

The third and fourth bytes contain the Handler ID and version number, as per the status command when issued for device $20 and $A0.

Mount Partition or Disk Image / Map Drives (Command $4D)

The Ultimate 1MB and Incognito APT implementations (not SIDE, MyIDE, Colleen, or IDEa) support dynamic mounting of partitions and ATR disk images. ATR disk images should be placed in a FAT32 partition on the HDD media (the APT partition editor *FDISK* allows the creation of a FAT32 partition alongside the APT segment of the disk). Normal APT partitions are mounted by referencing their *Partition ID* using command $4D, while ATRs are mounted by referencing their FAT32 starting cluster using the same command. Mounting is accomplished via a control block. The DCB should be set up as follows:

Address	***OS Label***	***Value***
$0300	DDEVIC	$20 or $A0
$0301	DUNIT	unit number (1 = master, 2 = slave)
$0302	DCMND	$4D
$0303	DSTATS	$C0 (read/write)
$0304/5	DBUFA	control block address
$0308/9	DBYT	6 (buffer length)

The format of the mount control block is as follows:

Offset	***Function***
$00	Flags
$01	Drive Number
$02-$05	Partition ID or ATR Start Cluster

Individual bits of *Flags* – when set - dictate the behaviour of the operation:

Bit 0: Operation is a disk image mount, otherwise a normal partition (*)
Bit 1: Make *drive number* the boot drive
Bit 2: FAT Slot Number is encoded in LBA address (*)
Bit 4: Disable internal BASIC on next boot (*)
Bit 5: Return drive map in *ID* (see *Drive Mapping* below) (**)
Bit 6: Reserved (***)
Bit 7: Unmount specified drive

**: Only applicable if the driver supports disk image mounting (See DEVINFO: Feature table)*
***: Only applicable if the driver supports drive mapping*
****: Previously "write to disk"; abandoned as of BIOS 1.0*

If the device supports drive mapping but not dynamic partition/ATR mounting, then all but bit 5 will return an error.

If bit 0 is set, the 32 bit value at offset $02 will be interpreted as the starting cluster of a disk image (ATR file) in the FAT32 area of the disk, rather than an APT partition ID. The cluster number must be obtained by analysis of the FAT32 disk directory, while the partition ID must be obtained by inspection of the APT partition table entries.

If bit 1 is set, as well as mounting the partition or disk image, *drive number* will become the boot drive number. This boot drive number overrides any boot value defined in the APT record on disk, but is not written to the disk. Therefore, the computer will continue to boot from the newly defined boot drive until the RAM-based partition table is refreshed, the boot drive is changed again, or the machine is power-cycled.

If bits 2 and 0 are set, bits 28-31 of *ATR Start Cluster* are interpreted as the host file system slot number (i.e. FAT Slot), currently in the range 0-3. This should be the same number returned by command $40 when the FAT partition was registered. Bit 2 is of Flags is ignored if bit 0 is clear. If bit 2 is clear and bit 0 is set, bits 28-31 of *ATR Start Cluster* are ignored and assumed to refer to the default FAT Slot 0.

Bit 5 overrides all others. If this bit is set, command $4D is considered a drive mapping query, and *returns* information to the mount control block (see *drive mapping* below).

Bit 6 is reserved as of BIOS v.1.0.

If bit 7 is set, the specified drive number will be rendered inactive and whichever partition is attached to it will go offline. If bit 7 is clear, the operation is assumed to be a mount operation, and the driver attempts to mount partition ID on the specified drive number.

Bits 6 and 7 may both be set (providing bit 0 is clear), resulting in the specified partition being permanently removed from the disk's drive mapping table (but not from the partition table itself).

When mounting a partition, no checks are made by the driver to guard against an existing mounted partition being unmounted if another partition is mounted on the same drive number, nor that a partition is not mounted on more than one drive number. The mounting software should be responsible for these kinds of checks.

So, if a drive number already has a partition assigned to it, the current partition for that drive number will go offline and be replaced by the partition specified in the mount block.

If no partition with the specified ID exists on the specified disk, an error will occur. Similarly – when mounting disk images - if there is no ATR file with the specified starting cluster, an error will be returned.

Drive Mapping

As mentioned above, setting bit 5 of *flags* in the mount control block turns command $4D into a drive mapping query command. When command $4D is issued with bit 5 of *flags* set, the mount control block should be set up as follows:

Offset	***Function***	***Contents***
$00	Flags	$20
$01	Drive Number	Doesn't matter – ignored
$02-$05	ID	Doesn't matter – ignored

In addition, DUNIT should be set to 1 to obtain a list of hosted drive numbers on the master device, and set to 2 to obtain a list for the slave device.

The drive mapping command, upon completion, will leave a 16 bit drive map in the first word of the mount control block. The lower 15 bits of this word correspond to 15 drive positions (bit 0 being drive 1, and bit 14 being drive 15; bit 15 will always be 0). Bits in the corresponding positions will only be set when there is an APT partition or disk image mounted on that drive number, and on the device being queried.

The main purpose of the drive map query is to avoid sending DISKINFO commands to offline drives or serial devices (floppy drives, etc) when obtaining information about all of the

online hard disk partitions or disk images. One can first build a list of drive numbers which are assigned to the hard disk, and confine DISKINFO calls to only those drives. This avoids often lengthy timeouts when DISKINFO queries are sent to non-existent drives or serial devices which do not recognize command $6E.

Host File System Support (Command $48)

The Ultimate 1MB and Incognito PBI BIOS implementations provide special support for the host file system, i.e. that *outside* of the APT segment of the hard disk. Usually this is a FAT partition which is not directly referenced by the APT partition table, but is used by an application such as the FAT XEX loader built into the Ultimate 1MB BIOS. Since the PBI BIOS needs to be able to handle ATR disk images mounted by the FAT loader, some way was needed for an application to "register" a partition described in the host partition table (i.e. the MBR of the hard disk).

To register a host partition, the command $48 should be used, with the DCB set up as follows:

Address	***OS Label***	***Value***
$0300	DDEVIC	$31 or $20 (doesn't matter)
$0301	DUNIT	unit number (1 = master, 2 = slave)
$0302	DCMND	$48
$0303	DSTATS	$C0 (read/write)
$0304/5	DBUFA	host control block address
$0308/9	DBYT	12 (buffer length)

The format of the host control block (HCB) is as follows:

Offset	***Function***
$00	Slot Number
$01	Aux
$02	File System
$03	Sectors Per Cluster
$04-$07	FAT LBA Start
$08-$0B	Cluster LBA Start

Individual bits of *Aux* – when set - dictate the behaviour of the operation. When Aux is 0 (no bits set), a register operation is assumed. The application should populate *FAT LBA Start*, *Cluster LBA Start*, *File System*, and *Sectors Per Cluster* before invoking command $48 with the buffer address pointing to the HCB. The command will register the partition in the next free slot, and populate the *slot* field of the HCB with the corresponding slot number. When mounting ATR disk images, it's this number which should be placed in the unused bits 28-31 of *ATR Start Cluster*. This tells the mount command to use the corresponding FAT parameters which were previously passed via command $48.

File System currently supports the following values:

0: FAT32
1: FAT16

To unregister (delete) a FAT partition entry, set bit 7 of *Aux*:

Bit 7: Unregister (delete) partition *slot*

Therefore: to delete a partition entry, set Aux to 128 and Slot to the number of the entry to be removed. Slot currently ranges from 0-3.

To check if a host partition is already registered, set Aux to 64 and populate *FAT LBA Start* with the desired LBA value. The function will then return the slot in which the host partition is registered (0-3) in *Slot Number*, or $FF if the host partition is not registered.

If command $48 fails for any reason (such as a lack of free slots), an error will be returned.

Force Media Change (Command $46)

This command (code $46) does the same job with devices $20 and $31: namely, it causes the driver to re-read the partition table from disk. With single drive interfaces, this is always the master disk, but with other adapters, it's necessary to specify 1 or 2 in DUNIT to specify which partition table to update. [Check this]

Force Media Change is most commonly issued by FDISK after writing out a modified partition table to disk and exiting to DOS, to ensure that the partition table in RAM immediately reflects the state of the partition table(s) on disk.

Ultimate 1MB, Incognito, SIDE and IDEa implementations of APT issue a Force Media Change command every time Reset is pressed with the Shift key held down.

Get Device Info (Command $EC)

The Get Device Info command ($EC) works the same way with devices $20 and $31, and simply returns a 512 byte buffer identical to that returned by the low-level IDE command $EC (Identify Device).

Please refer to the ATA specs for details of the contents of the Identify Device buffer.

Jonathan Halliday

21 February 2014