APT Driver API Technical Reference

For Ultimate 1MB, Incognito, SIDE and the 1088XEL

Contents

Introduction

The APT hard disk partitioning format was devised to provide broad compatibility between a broad range of Atari 8-bit hard disk adapters. APT was originally devised for SIDE and IDE Plus, but it has since been adopted by the Ultimate 1MB (U1MB) and Incognito PBI hard disk controllers (the former working in conjunction with the SIDE cartridge) and has proliferated via SpartaDOS X (SDX) 'soft drivers' to MYIDE and MYIDE II. Latterly, the 1088XEL mini-ATX motherboard was built around U1MB and APT.

While APT is a partitioning standard rather than a driver framework, the standard has become married to KMK's 'XDCB' protocol, which implements the DEVINFO and DISKINFO commands to provide a standardised method of obtaining geometry information from mass storage devices. Because all APT implementations support XDCB, we will consider the corresponding driver API as the 'APT driver API'.

Unfortunately, since no agreed standard was yet decided upon for dynamic partition and ATR disk image mounting, APT compliant devices which support these facilities do so in a bespoke manner. IDE Plus 2.0, for instance, has no published API for the ATR mounting of which it is capable, and to the best of my knowledge does not possess any method of dynamically mounting partitions. U1MB and Incognito, meanwhile, implement a public API for partition and ATR mounting which may or may not be adopted by future devices offering similar functionality.

Aside from dynamic mounting (and host partition registration with regard to mounting ATRs in FAT volumes), the API described here is common to the U1MB PBI BIOS, Incognito, XEL-CF, IDE Plus 2.0, and all SDX 'soft-drivers'. Because of this, the same partition editor (APT FDISK) and partition management tools (from both the IDE Plus and APT toolkits) work with all currently available APT devices.

The APT and XDCB were designed by Konrad Kokoszkiewicz (KMK, Drac030). Original API extensions for U1MB, SIDE and Incognito were designed and implemented by myself and Sebastian Bartkowicz (Candle O'Sin). Further revisions to the same API were designed and implemented by me.

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 disk devices (such as sector read/write, etc). Instead, we will describe the SIO functions 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:

Address	Label
$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 a 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, these should always be 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 = primary, 2 = secondary)
$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	Meaning
0	Dynamic partition mounting
1	Drive mapping
2	ATR mounting
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 = primary, 2 = secondary)
$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:

Device	ID
IDE Plus 2.0	$01
KMK/JZ 'IDEa'	$02
Ultimate 1MB	$03
Incognito	$04
SIDE1	$05
SIDE2	$06
Colleen	$07
MyIDE	$08
MyIDE II	$09
XEL-CF (U1MB PBI)	$0A
XEL-CF (SDX driver)	$0B

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. In that case, bits 0-27 describe the cluster number of the ATR, and bits 28-29 hold the "host partition ID" (numbered 0-3) of the FAT partition in which the ATR resides. Bits 30-31 are reserved. See the section on host file system support later in this document.

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:

Bit(s)	Meaning
7	1 = Enhanced density ATR (128bps and size = 1040 sectors)
6-5	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 FAT16 or FAT32 partition on the HDD media (the APT partition editor *FDISK* allows the creation of a FAT partition alongside the APT segment of the disk). Normal APT partitions are mounted by referencing their *Device Sub-ID* using command $4D, while ATRs are mounted by referencing their FAT starting cluster and *Host Partition ID* using the same command. Mounting is accomplished via a Mount Control Block. The DCB should be set up as follows:

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

DUNIT must be set prior to mounting an APT partition, but when mounting an ATR, its value should be left at 1 (the value is ignored, but still must hold a valid value). Since ATRs are mounted with reference to their host partition ID, the DUNIT value is not used.

The format of the MCB is as follows:

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

Bytes 2-5 form a little-endian 32-bit value which contains the partition Sub-ID (for a partition mount), or the ATR file's starting cluster number (bits 0-27) and Host Partition ID (bits 28-29). Unused bits should be set to 0.

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

Bit	Meaning
0	1 = Disk image mount
1	1 = Partition mount
2	1 = Set state of BASIC on next boot according to bit 3
3	If bit 2 set, 1 = BASIC on, 2 = BASIC off
4	1 = boot from drive number in MCB on next boot
5	1 = Return drive map in *ID* (see *Drive Mapping* below) (*)
6	1 = Rotate disk images
7	1 = Unmount specified drive

** Only applicable if the driver supports drive mapping*

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

Disk Image Mount (bit 0)

If bit 0 is set, the 28-bit value at offset $02 will be interpreted as the starting cluster of a disk image (ATR file) in the registered FAT partition whose host partition ID is represented by bits 28-29 of the cluster address. Bits 30-31 should be set to 0. DUNIT should be set to 1 although it is ignored, since the host partition will already be registered on one or the other physical disks (assuming two disks are present).

Prior to mounting an ATR, the mounting application must register the host partition in which the ATR disk image resides. The registration API returns an ID (0-3) upon registering a partition, and this same ID should be supplied in bits 28-29 of an ATR cluster number when ATRs residing in that partition are subsequently mounted. Once a FAT partition is registered, its ID remains valid until a media change is called (which reloads all partition tables and unmounts any dynamically mounted partitions and disk images). After a cold boot or immediately following a media change operation, no host partitions are registered. Prior versions of the PBI BIOS used to auto-register the first FAT on the disk, but since newer versions of the loader handle FAT registration transparently, this was dropped in the most recent PBI BIOS versions.

For the reasons described above, a program which mounts ATRs should check that the FAT partition containing the ATR to be mounted is actually registered before attempting to mount a disk image. The best way to accomplish this is to routinely check whether the FAT is registered immediately prior to mounting any disk image. If the partition isn't registered, it can then be registered prior to the mount operation. If the partition is already registered, the host partition ID will be returned and can be supplied via the mount call. Host partition registration is described in the next section.

Partition Mount (bit 1)

If bit 1 is set, the operation is assumed to be a partition mount, and the 32-bit value at offset $02 is interpreted as a Device Sub-ID, which is to say a partition sequence number or ID. In this instance, the value of DUNIT is observed, since it tells the driver on which physical disk (1 or 2 for primary or secondary) the partition resides. No checks are made by the driver whether a partition or disk image is already mounted at the specified drive number: the newly mounted partition will replace it, causing any previous partition to be unmounted. Likewise, no checks are made that a partition is not mounted on multiple drive numbers at once: the calling program should ensure this doesn't happen.

BASIC State (bits 2 and 3)

When bit 2 is set, bit 3 specifies the state of internal BASIC on the next boot after the command is issued. If bit 3 = 1, BASIC is left enabled (or in the default OS state, whichever that is), and if bit 3 = 0, BASIC is suppressed.

Note that one may specify the BASIC state as part of a partition or ATR mount operation, since the specified BASIC state will be set during the mount operation.

Set Boot Drive (bit 4)

When bit 4 is set, the boot drive (i.e. the drive number the system will boot from on the next OS restart) is set to the value stored in byte 1 of the MCB (Drive Number). If the desired boot drive number is the same as that of the current mount operation, the setting bit 4 during the mount operation will case the system to boot from the mounted ATR without.

Drive Mapping (bit 5)

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

The DCB should be set up thus:

Address	OS Label	Value
$0300	DDEVIC	$20 or $A0
$0301	DUNIT	unit number (1 = primary, 2 = secondary)
$0302	DCMND	$4D
$0303	DSTATS	$40 (read)
$0304/5	DBUFA	control block address
$0308/9	DBYT	6 (buffer length)

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 the unit specified in DUNIT.

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.

Commonly, one will wish to obtain a bitmap of all mounted drives on both the master and slave devices (where applicable). In that case, simply call the drive mapping function twice (once for the master, and again for the slave) and OR the two words together.

Rotate ATRs (bit 6)

When bit 6 is set, any mounted ATRs are rotated left one bit in the drive number order. This is to say an ATR on D1: moves to D15:, an ATR on D2: moves to D1:, etc. Regular partitions are not affected.

Unmount Drive (bit 7)

If bit 7 is set, the partition or disk image on the specified drive number will be unmounted.

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 employ the tools provided by the API to ensure this does not happen.

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.

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 firmware. 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	$20 or $A0
$0301	DUNIT	unit number (1 = primary, 2 = secondary)
$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 0-3 (returned value)
$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 = 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 S*lot* field of the HCB with the corresponding slot number. When mounting ATR disk images, it's this returned slot number which should be placed in the unused bits 28-29 of *ATR Start Cluster*. This tells the mount command to use the corresponding FAT parameters (device number, cluster start, FAT width, etc) 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. The range for valid slot values is currently 0-3 (four host slots).

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.

Note: Media Change does not reset the controller, so if a hard reset is required, the Reset All command should be issued first.

Reset All (Command $E7)

The Reset All command simply performs a hardware reset on all controllers on the bus where appropriate. If the hardware does not support a hard reset (Incognito, for example), the driver will simply attempt to put the disk into 8-bit PIO mode regardless, possibly returning an error if the device fails to respond.

The Reset All command ignores all DCB parameters aside from DDEVIC, so you may call it with $2x, 3x, Ax or Bx in DDEVIC. Unit numbers and buffer addresses are ignored.

On the 1088XEL, the Reset All command may be used to reset a disk following a media hot-swap.

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.

Feedback

If you notice any errors or typos in this document, please feel free to report them for correction by email.

Jonathan Halliday

July 2018