SIDE Driver API Technical Reference

(Draft 2)

Introduction

The SIDE interface for Atari 8-bit computers employs the APT (Atari Partition Table) standard, and a software API compatible with that of the IDE Plus 2.0 interface. Both the APT and the corresponding driver API were designed by KMK with additional input from myself and certain other Atari 8-bit developers. The SIDE driver implementation follows the APT standard closely. This document describes the specifics of the API common to both interfaces, but with particular reference to SIDE.

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

drac030.krap.pl/APT_spec.pdf

It should be noted that information presented in this 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 BIOS version 0.9.

In this documentation, we shall not cover the entire gamut of SIO commands which can be used with the SIDE interface, such as sector read/write. Instead, we will describe the calls which have specific relevance to the implementation.

The XDCB

IDE Plus and SIDE implement 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

The DAUX1/2 contain low word and the DAUXA/B contain 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***
$6E	Device / Disk Info	(‘n’)
$4D	Mount Partition	(‘M’)
$46	Force Media Change	(‘F’)
$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)

We’ll first consider the DEVINFO command, invoked with $6E in DCOMND ($302), and $20 or $A0 in DDEVIC.

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-$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 (ASCII, NUL padded)
$38-$FF	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.

The next three 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.

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

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

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	$53
$0303	DSTATS	$40
$0304/5	DBUFA	buffer address
$0308/9	DBYT	4 (buffer length)

The returned block consists of four bytes:

$00, $00, ID, Version

where *ID* is the handler identification number ($02 for SIDE) and *version* is its BCD-coded version number (e.g. $10 = 1.0).

DISKINFO (Disk Information)

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 harddisk).

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	Reserved
$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.

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 are reserved and should all read zero.

Next is a 40 byte NUL-padded partition name. If the partition has no name, this field should contain 40 zeros.

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

Disk Status

The status command (code $53), when issued in the context of device $31, tests for a partition’s presence. The DCB should be set up exactly as with device $20, and the contents of the returned buffer are the same.

Mount Partition

The SIDE driver supports dynamic mounting of partitions by their *Partition ID* using command $4D. Mounting is accomplished via a control block. The DCB should be 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	$4D
$0303	DSTATS	$40
$0304/5	DBUFA	control block address
$0308/9	DBYT	4 (buffer length)

The format of the mount control block is as follows:

Offset	***Function***
$00	Flags
$01	Drive Number
$02/$03	Partition ID

Bits 6 and 7 of *Flags* have dictate the behaviour of the operation:

Bit 6: If set, make drive assignment permanent
Bit 7: if set, unmount specified drive

If bit 6 is set, the result of the mount operation will be written to the drive mapping table on the disk and will be permanent. Otherwise (if the bit is clear), changes will not be written to the disk (only to the partition table in RAM), and will therefore be lost when the machine is rebooted.

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, 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, if the specified partition is already mounted on a different drive number, an error will be returned and the partition will not be mounted on the second drive number. Unmount the other drive number first before mounting the same partition elsewhere.

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.

Force Media Change

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 SIDE, this is always the master disk, but with other implementations, it's merely 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.

Get Device Info

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

6 November 2011