UniDisk 3.5
#1: UniDisk 3.5 Internals

Revised by: Matt Deatherage November 1988
Written by: Mike Askins May 1985

This Technical Note formerly described the internals of the UniDisk 3.5, and this information is now documented in the *Apple IIGS Firmware Reference*.

This Note formerly documented the internal structure of the UniDisk 3.5, primarily for those interested in providing copy protection. Apple Computer no longer supports copy protection schemes, and we strongly urge developers to make use of alternate methods to limit unauthorized duplication.

The internals of the UniDisk 3.5 are now documented in the *Apple IIGS Firmware Reference*.

Further Reference

- *Apple IIGS Firmware Reference*

Apple II
Technical Notes

Developer Technical Support

UniDisk 3.5
#2: UniDisk 3.5 ID Bytes

Revised by: Matt Deatherage November 1988
Written by: Mike Askins May 1985

This Technical Note describes the signature bytes of the UniDisk 3.5.

The signature bytes for the UniDisk 3.5 are the same as those for any SmartPort device:

$Cn01 = $20	
$Cn03 = $00	ProDOS Block Device
$Cn05 = $03	
$Cn07 = $00	SmartPort Interface

where n is the slot number of the device.

When searching the slots for a UniDisk 3.5 it is very important to check **all** the signature bytes, since there are other peripherals with similar ID bytes. Once you find a SmartPort card (or port), you should do a SmartPort `STATUS` call to determine which devices are connected to it. Any number of different devices could match the SmartPort ID bytes, so trying to identify a device without making a SmartPort `STATUS` call is very likely to produce inaccurate results.

Why the UniDisk 3.5 Does Not Auto-Boot on Older Machines

If you look carefully, you will notice that the older (][,][+ and unenhanced IIe) Autostart Monitor will not boot any SmartPort device because the ID byte at $Cn07 = $00 instead of $3C (like the old Disk II). If Apple had left the ID bytes the same as the Disk II, then older versions of Apple II Pascal (1.2 and earlier) would assume that the drive was a Disk II.

Where This Leaves You

The enhanced IIe ROMs, as well as the UniDisk 3.5 IIc ROMs and later (which you have if you are using a UniDisk 3.5 on a IIc) check only the first three ID bytes. This check means that they will not only auto-boot the UniDisk 3.5, but any SmartPort or ProDOS block device. On an older machine, you can boot one of these devices by typing `PR#n` from AppleSoft or `Cn00G` from the Monitor.

Further Reference

- *Apple IIGS Firmware Reference*

Apple II
Technical Notes

Developer Technical Support

UniDisk 3.5
#3: STATUS Call Bug

Revised by: Matt Deatherage November 1988
Written by: Mike Askins & Cameron Birse September 1984

This Technical Note documents a bug in the ProDOS `STATUS` call when used with a UniDisk 3.5.

The Bug

We have found that SmartPort does not return the `WRITE PROTECT` error on the `STATUS` call. (The `WRITE` call does return the `WRITE PROTECT` error as required.)

The bug manifests itself under ProDOS (and not under Pascal, since Pascal does not require the write protect error to be returned on the `STATUS` call). Specifically, if a write-protected disk is present in the UniDisk 3.5, and the application tries to write less than 512 bytes of data to a file that already exists on the media, it becomes impossible to finish the write or to close the file. Many applications ignore errors on close calls and try to reuse the buffer area which was presumably freed by the close call. This reuse results in further errors, even if the UniDisk 3.5 is later write-enabled, since ProDOS still thinks the file is open. This bug also decreases the maximum number of open files allowed, as the file left open is included in that number.

The bug also seems to cause the ProDOS `CREATE` call to fail. When a new file is created, opened and written to, and the write fails, the file manager does not deallocate the block that it reserved in the creation attempt. (The RAM copy of the bitmap seems to get trashed—`GET_FILE_INFO` calls at this point report that there are zero blocks available.) If you subsequently write **enable** the disk and do the save (with any size file), the file is written to the disk, and the bitmap is updated. The result is that there is a block reserved on the disk that no file owns, and that block cannot be freed through normal ProDOS file calls.

The Solution

Although this problem was fixed in later IIc revisions, the UniDisk 3.5 interface for the Apple][+ and IIe has never been modified. Therefore, if your application habitually performs the actions outlined above, you may avoid it by first checking to see if the media is write-protected instead of letting the buggy ProDOS `STATUS` call do it for you.

One way to accomplish this would be to issue a SmartPort `STATUS` call using a `statcode` = $00. This call returns four bytes of information, the first of which is the general status byte. This byte has the following format:

Bit	Meaning
7	0 = character device; 1 = block device
6	1 = write allowed
5	1 = read allowed
4	1 = device on line or disk in drive
3	0 = format allowed
2	**0 = medium write protected** (block devices only)
1	1 = device currently interrupting (Apple IIc only)
0	1 = device currently open (character devices only)

As shown in the table, bit 2 of this byte tells you what the ProDOS `STATUS` call cannot seem to figure out—the media in the drive is currently write-protected.

UniDisk 3.5
#4: Accessing Macintosh Disks

Revised by: Matt Deatherage November 1988
Written by: Mike Askins May 1985

This Technical Note formerly discussed drive-specific SmartPort calls. These calls are now documented in the *Apple IIGS Firmware Reference*. This Note now describes how to access Macintosh disks from a UniDisk 3.5 disk drive, as this information was not documented in the manual.

Macintosh Disk Access

The disk data format used in the UniDisk 3.5 is essentially identical to that used for Macintosh disks. There are three notable differences between the two formats:

- Macintosh blocks are 524 bytes; UniDisk 3.5 blocks are 512 bytes.
- Macintosh MFS disks are single sided; UniDisk 3.5 disks are double sided. (Macintosh HFS disks are double sided.)
- The Macintosh uses a 2:1 physical block interleave; the UniDisk 3.5 uses a 4:1 interleave.

Accessing Blocks on a Macintosh Disk

Reading from a Macintosh disk is accomplished with the use of the `READ` command (as opposed to the `READBLOCK` command, which enforces 512 byte data.) A call to load block zero from the Macintosh disk in Unit #1 into memory at $2000 would look like this:

```
MacRead     JSR   Dispatch          ;Normal SmartPort Entry point
            DFB   $08               ;Character READ command code
            DW    Cmd_List          ;The parameter list
            BCS   Error             ;Optional error handling...
            ...
Cmd_List    DFB   $04               ;CharRead has four parameters
            DFB   $01               ;Unit number
            DW    $2000             ;Buffer address
            DW    524               ;Always transfer 524 bytes
            DFB   $00               ;Block (lo)
            DFB   $00               ;Block (med)
            DFB   $00               ;Block (hi)
```

Writing to a Macintosh disk is accomplished with the use of the WRITE command. A call to write block zero to the Macintosh disk in Unit #1 with data at memory location $2000 would look like this:

```
MacWrite    JSR   Dispatch          ;Normal SmartPort Entry point
            DFB   $09               ;Character WRITE command code
            DW    Cmd_List          ;The parameter list
            BCS   Error             ;Optional error handling...
```

The Cmd_List is the same as in the READ example.

Formatting Macintosh Disks

The formatting routine in the UniDisk 3.5 firmware can format single- or double-sided disks of variable physical block interleave. The parameters controlling the interleave and the number of disk sides are located in the controller's zero page and are set to defaults whenever the INIT call is issued to SmartPort. These parameters can be altered by using the SET_DOWN_ADR and DOWNLOAD subcalls of the CONTROL call. Once altered, the FORMAT call uses these values in the formatting process. These zero page locations and their values are detailed below:

Parameter	Location	Values
Interleave	$0062	$02 = Mac, $04 = UniDisk 3.5
DoubleSided	$0063	$00 = Single, $80 = Double-sided

The following code example formats the media in Unit #1 as a Macintosh disk:

```
MacFormat   JSR   Dispatch          ;Set address to patch interleave
            DFB   $04               ;Control call (Set_Down_Adr)
            DW    Cmd_ListA         ;Parameter List
            BCS   Error
;
            JSR   Dispatch          ;Now patch the interleave byte
            DFB   $04               ;Control call (DOWNLOAD)
            DW    Cmd_ListB         ;Parameter List
            BCS   Error
;
            JSR   Dispatch          ;Set address to patch single sided
            DFB   $04               ;Control call (Set_Down_Adr)
            DW    Cmd_ListC         ;Parameter List
            BCS   Error
;
            JSR   Dispatch          ;Now patch the single sided byte
            DFB   $04               ;Control call (DOWNLOAD)
            DW    Cmd_ListD         ;Parameter List
            BCS   Error
;
            JSR   Dispatch          ;Finally...
            DFB   $03               ;This is the actual format call
            DW    Cmd_ListE         ;Parameter List
            BCS   Error
;
            RTS
```

The parameter lists are as follows:

```
Cmd_ListA     DFB    $03               ;All control calls are 3 parms long
              DFB    $01               ;Unit #1
              DW     Ctrl_ListA        ;This has the interleave address
              DFB    $06               ;Set_Down_Adr control code

Ctrl_ListA    DW     $02               ;Two bytes for download address
              DW     $0062             ;Interleave address

Cmd_ListB     DFB    $03               ;All control calls are 3 parms long
              DFB    $01               ;Unit #1
              DW     Ctrl_ListB        ;This has the interleave value
              DFB    $07               ;Download control code

Ctrl_ListB    DW     $01               ;Two bytes for download address
              DFB    $02               ;Mac Disk Interleave value

Cmd_ListC     DFB    $03               ;All control calls are 3 parms long
              DFB    $01               ;Unit #1
              DW     Ctrl_ListC        ;This has the sides byte address
              DFB    $06               ;Set_Down_Adr control code

Ctrl_ListC    DW     $02               ;Two bytes for download address
              DW     $0062             ;Interleave address

Cmd_ListD     DFB    $03               ;All control calls are 3 parms long
              DFB    $01               ;Unit #1
              DW     Ctrl_ListD        ;This has the sides value
              DFB    $07               ;Download control code

Ctrl_ListD    DW     $01               ;Two bytes for download address
              DFB    $00               ;Value for single sided disk

Ctrl_ListE    DFB    $01               ;Format call has just one parameter
              DFB    $01               ;Unit number
```

Note: You may encounter difficulties when switching 400K single-sided disks and 800K double-sided disks in the same drive. `STATUS` requests for the number of blocks on the disk in the drive are valid for the disk **last** accessed. Thus, when you `READ` from an 800K disk, eject it, and insert a 400K disk, a `STATUS` call will reveal a size of 800K until a `READ` or `WRITE` command is issued. Applications which intend to handle both 800K and 400K disks should do a `READ` before each `STATUS` call.

Further Reference

- *Apple IIGS Firmware Reference*
- *Apple IIc Technical Reference Manual, Second Edition*

Apple II
Technical Notes

Developer Technical Support

UniDisk 3.5
#5: Architectural Differences Between 3.5" Drives

Revised by: Matt Deatherage November 1988
Written by: Cameron Birse & Mike Askins October 1986

This Technical Note provides information of interest to those developers writing low-level software for the UniDisk 3.5 and Apple 3.5 disk drives.

Definition of Drives

It is important to understand the differences between Apple's 3.5" drives if you are considering writing low-level software for use on the Apple II family drives.

UniDisk 3.5 is an intelligent drive, meaning that it has a microprocessor-based controller inside the drive enclosure that communicates with the host computer in an intelligent fashion through the IWM port. The host sends commands to the intelligent controller in the drive and the controller manipulates the drive hardware to read or write, and sends the data back to the host in a "packet" format.

Apple 3.5 Drive is an unintelligent drive that depends on the host computer to manipulate the drive hardware to read and write data to and from the drive. Apple IIGS low-level routines for this drive will be essentially the same as those downloaded to the UniDisk 3.5 controller RAM, except they will reside in the host computer's memory. New device-specific control calls must be used for the Apple 3.5 Drive.

Tips for Low-Level Drive Access

The following calls are not guaranteed to be compatible in the future; for the highest level of compatibility, avoid disk access at this level.

- **Identifying the drives:** The drives can be identified by first searching for a device that has the SmartPort firmware. After determining that there is a SmartPort device in the machine, perform a `STATUS` call with the `statcode` = $03 (return Device Information Block (DIB)). In the DIB there is a type byte and a subtype byte. The UniDisk 3.5 has a value of $01 for the type byte and $00 for

the subtype byte. The Apple 3.5 Drive also has a value of $01 for the type byte, but its subtype byte value is $C0. Be sure to make device-specific calls to ensure drive identification. See SmartPort Technical Note #7, SmartPort Subtype Codes for more details.

- **Special routines:** In the UniDisk 3.5, there is extra RAM space in the controller's memory map for custom read, write and ID routines. These routines can be downloaded to the controller from the host and executed via the SmartPort. With the Apple 3.5 Drive, these special routines reside in the host memory. Equivalent mark and hook tables for the Apple 3.5 Drive, set by control calls through the SmartPort, are supported on the Apple IIGS , but are not guaranteed for all drives and CPUs.
- **IWM hardware differences:** On the UniDisk 3.5, the IWM registers are located in the drive's controller memory starting at $0A00. On the Apple 3.5 Drive, the IWM registers are located in host memory starting at $C0E0 (slot 6 I/O space).
- **Speed differences:** Downloaded code in the UniDisk 3.5 controller runs at slightly under 2 MHz, and the cycle times are regular. The Apple IIGS running at 1 MHz also has regular cycles, however, when running at 2.8 MHz, the timing is complicated by RAM refresh and I/O synchronization times. It is best to avoid timing critical solutions, or be sure to run at 1 MHz for the Apple 3.5 Drive.

As always, in order to promote compatibility between your software and future Apple II systems and to avoid writing utilities which will only work on one kind of drive, you should avoid low-level calls that are specific to a particular device or CPU.

Further Reference

- *Apple IIGS Firmware Reference*