# Chapter 3 Calls to the Protocol Converter

This chapter is about the Protocol Converter, which is a set of assembly-language routines used to support external I/O devices, such as UniDisk 3.5. To ProDOS and Pascal 1.3, the Protocol Converter appears to be a block device.

The following topics are discussed in this chapter:

- How to locate the Protocol Converter
- How to issue a call to the Protocol Converter
- The use of each call
- The parameters required for each call
- Possible errors codes returned for each call
- The possible causes of the errors

At the end of this chapter is an example of an assembly-language program that uses a Protocol Converter call.

## Locating the Protocol Converter

The code for the Protocol Converter always begins at address $C500 in the Apple IIc with 32K ROM. To ensure compatibility of your programs with the Apple IIe, however, your Protocol Converter routines should always begin with a search for the Protocol Converter by looking for the following bytes: CN01 = $20, CN03 = $00, CN05 = $03, and $CN07 = 00, where N can be from 1 to 7. The Protocol Converter entry point is then at address $CN00 + ($CNFF) + 3. The sample program at the end of this chapter illustrates such a search.

## How to Issue a Call to the Protocol Converter

MLI calls: see the *ProDos Technical Reference Manual*, Chapter 4.

Protocol Converter calls are coded in a manner similar to ProDOS Machine Language Interface (MLI) calls: The program executes a JSR (jump to subroutine) to a dispatch routine at address $C500 + ($C5FF) + 3, where ($C5FF) refers to the value of the byte located at $C5FF.

The number of the Protocol Converter call and a two-byte pointer to the call's parameter list must immediately follow the call. Here is an example of a call to the Protocol Converter:

```
IWMCALL JSR DISPATCH ;Call PC command dispatcher
        DFB CMDNUM   ;This specifies the command type
        DW  CMDLIST  ;2-byte (low,high) pointer to parameter list
        BCS ERROR    ;Carry is set on an error
```

The command number determines the Protocol Converter call to be used. The length and contents of the parameter list depend on the call, as described in Section "Descriptions of the Protocol Converter Calls."

Upon completion of the call, the program resumes execution at the statement following the pointer to the parameter list. In this example, the DFB and DW statements are skipped, and execution resumes with the BCS statement. If the call is successful, the C flag (in the Processor Status register of the 65C02 microprocessor) is cleared (0), and the accumulator (the A register) is cleared to all zeros. If the call is unsuccessful, the C flag is set (1), and the error code is placed in the A register. After the Protocol Converter call, the contents of the 65C02's registers are as follows:

| **Register:** | **Processor Status** | | | | | | | **X** | **Y** | **A** | **PC** | **S** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Bit:** | **N** | **Z** | **C** | **D** | **V** | **I** | **B** | | | | | |
| **Successful call:** | x | x | 0 | 0 | x | u | u | x | x | 0 | JSR+3 | u |
| **Unsuccessful call:** | x | x | 1 | 0 | x | u | u | x | x | Error | JSR+3 | u |

x = undefined, except in cases where index information is returned in X and Y

u = unchanged

Most Protocol Converter calls include a two-byte pointer to a parameter list, which may contain information to be used by the call, or provide space for information to be returned by the call.

## Cautions

You *must* observe the following cautions when using the Protocol Converter, or *your program will crash*:

- The Protocol Converter requires up to 35 bytes of stack space. Be sure you take this into account when calculating the stack space used by your program.

  Failure to allow for the stack space used by the Protocol Converter can result in a stack overflow, causing your program to crash when it attempts to access the data that have been overwritten.
- Data cannot be read from the Protocol Converter into RAM that is not both read-enabled and write-enabled. The Protocol Converter must be able to read from the RAM after writing to it, to obtain a checksum. Failure to observe this rule results in an error (BUSERR $06).

Reading and writing to RAM: see Section "Bank Switched Memory" in the *Apple IIc Reference Manual*.

- Do not attempt to use the Protocol Converter to put anything into zero page locations. These locations are reserved for temporary storage of data by the Protocol Converter.

## Descriptions of the Protocol Converter Calls

Calls to the Protocol Converter are used

- to obtain status information about a device
- to reset a device
- to format the medium in a device
- to read from a device
- to write to a device
- to send control information to a device.

The Protocol Converter calls, in command-number sequence, are:

**STATUS ($00)**

Returns status information about a particular device, including general status (character or block device, read or write protection, format allowed, device on line); the device control block (set with the CONTROL call); the device newline status (character devices only); and device-specific information (number of blocks, ID string, device name, device type, device firmware version).

**READ BLOCK ($01)**

Reads one 512-byte block from a disk device, and writes it to memory.

**WRITE BLOCK ($02)**

Writes one 512-byte block from memory to a disk device.

**FORMAT ($03)**

Prepares all blocks on a block device for reading and writing.

**CONTROL ($04)**

Controls some device functions, including warm resets, setting the device control block (which controls global aspects of the device's operating environment), setting newline status (character devices only), and device interrupts. Several CONTROL calls are device-specific.

**INIT ($05)**

Resets all resident devices. A global reset is done automatically on startup or system resets from the keyboard; an application should never have to reset all devices.

**OPEN ($06)**

Prepares a character device for reading or writing.

**CLOSE ($07)**

Tells a character device that a sequence of reads or writes is over.

| | |
|---|---|
| **READ ($08)** | Reads a specified number of bytes from a specified device. |
| **WRITE ($09)** | Writes a specified number of bytes from memory to a specified device. |

## Format of Call Descriptions

The following sections describe each protocol converter call, including the command number, the parameter list, and error codes. The calls are discussed in command-number order. Each call is shown in this format:

*Command Name* The name used to identify the call for descriptive purposes.

*Command Number* The number, in hexadecimal, that specifies the call to the Protocol Converter.

*Parameter List* A list of the parameters required for the call.

*General Description* The purpose and use of the call.

*Parameter Descriptions* A description of each parameter, and descriptions of data bytes pointed to by parameters. When a parameter is a status or control code, the meaning of each code number is discussed.

*Possible Errors* A list of the error codes that can be returned by this call. A complete list of Protocol Converter error codes is included at the end of this chapter.

## STATUS

| | |
|---|---|
| **Command Number** | $00 |
| **Parameter List** | $03 (parameter count)<br>Unit number<br>Status list pointer (low byte, high byte)<br>Status code |

The STATUS call returns status information about a particular device. The type of information returned is determined by the status-code parameter, and the location to which it is returned is determined by the status list pointer.

After a STATUS call has been executed, the 65C02's X and Y registers contain the number of bytes of status information returned (the low byte of this number is in the X register, and the high byte is in the Y register).

## Parameter Descriptions

**Parameter Count**
1-byte value
3 for this call.

**Unit Number**
1-byte value
The Protocol Converter assigns each device a unique number during initialization (on startup and cold reset). The numbers are in the range $01—$7E, and are assigned according to the devices' positions in the chain.

Important

Use a unit number of $00 and a status code of $00 in a STATUS call to obtain the status of the Protocol Converter itself (see the discussion under Status Code = $00, below).

**Status List Pointer**
2-byte value
Points to the buffer to which the status is to be returned. The length required for the buffer varies depending on the status request being made.

**Status Code**
1-byte value
Indicates what kind of status request is being made. Status codes are in the range $00-$FF, as follows:

*Table 3-1.* Status Codes

| Code | Status Returned |
|---|---|
| $00 | Return device status |
| $01 | Return device control block (DCB) |
| $02 | Return newline status (character devices only) |
| $03 | Return device information block (DIB) |
| $05 | Return UniDisk 3.5 status |

Status codes $01 and $02 are not supported by the UniDisk 3.5 and result in an error (BADCTL $21). Device status for the UniDisk 3.5 can be obtained by using status code $05.

***Status Code = $00, Return Device Status*** The device status consists of four bytes. The first is the general status byte:

| Bit | Description |
|---|---|
| 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 |
| 0 | 1 = device currently open (character devices only) |

If the STATUS call is for a block device, the next three bytes (low byte first) are the size in 512-byte blocks. The maximum size is 16 million ($FFFFFF) blocks (about 8 gigabytes). If the call is for a character device, these three bytes must be set to zero.

*Unit Number $00:* A STATUS call with status code = $00 and unit number = $00 returns the status of the Protocol Converter itself. In this case, the status list consists of eight bytes, as follows:

```
STAT_LIST DFB  Number_Devices    ;Devices hooked to PC
          DFB  Interrupt_Status ;Bit 6 clear = interrupt sent
          DFB                   ;Reserved
          DFB                   ;Reserved
          DFB                   ;Reserved
          DFB                   ;Reserved
          DFB                   ;Reserved
          DFB                   ;Reserved
          DFB                   ;Reserved
```

ACIA status register: see Section "Firmware Handling of Interrupts" in the *Apple IIc Reference Manual.*

The Number_Devices byte returns the total number of intelligent devices attached to the Protocol Converter. The Interrupt_Status byte is a copy of the Asynchronous Communications Interface Adapter (ACIA) status register at the time of the interrupt, and is used to indicate that a device requires interrupt servicing. If the sixth bit of this byte equals zero, one or more devices in the Protocol Converter Bus daisy chain must be serviced; your interrupt handler must poll each device on the chain to determine which ones.

> ***About Interrupts:*** Devices that require interrupt servicing must use the EXTINT line on the external disk port connector of the Apple IIc to be supported by the Protocol Converter. UniDisk 3.5, for example, does not support this line, and so cannot generate interrupts to the Protocol Converter. See Section "CONTROL" for instructions on enabling Protocol Converter interrupts. See Appendix E in the *Apple IIc Reference Manual* for more information about programming with interrupts.

***Status Code = $01, Return Device Control Block*** The device control block (DCB) is used to control various operating characteristics of a device, and is device dependent. Each device has a default DCB, which can be altered with a CONTROL call. The first byte (the *count byte*) gives the number of bytes in the control block (*not* including the count byte), so the length never exceeds 256 bytes (257 including the count byte).

UniDisk 3.5

> UniDisk 3.5 has no DCB, and returns an error (BADCTL $21) in response to this call.

Newline read mode: see Chapter 4 in the *PRODOS Technical Reference Manual.*

***Status Code = $02, Return Newline Status*** Newline status applies only to character devices. Use of statcode = $02 with a block device results in error BADCTL ($21).

***Status Code = $03, Return Device Information Block*** The device's information block contains information identifying the device, its type, and various other attributes. The returned status list has the following form:

```
STAT_LIST DFB  Device_Statbyte1 ;Same as byte 1 in Status Code=0
          DFB  Device_Size_Lo   ;Number of blocks (block device)
          DFB  Device_Size_Med  ;Number of blocks (middle byte)
          DFB  Device_Size_Hi   ;Number of blocks (high byte)
          DFB  ID_String_Length ;Length in bytes (16 max.)
          ASC  '<device name>'  ;7-bit ASCII, uppercase, padded
*                                with spaces, eighth bit always=0
*                                (16 bytes)
          DFB  Device_Type_Code
          DFB  Device_Subtype_Code
          DW   Version          ;Device firmware version number
```

*Status Code = $05, Return UniDisk 3.5 Status* This call allows the diagnostic program to get more detailed information about the cause of a read or write error, and to examine the contents of the 65C02's registers after a CONTROL Protocol Converter call with control code = $05 (see Section "CONTROL"). The returned status list has this form:

```
STAT_LIST DFB   $00
          DFB   Error     ;Soft Error byte (see below)
          DFB   Retries   ;Number of retries (see below)
          DFB   $00
          DFB   A_Value   ;Acc value after a CONTROL EXECUTE call
          DFB   X_Value   ;X value after EXECUTE
          DFB   Y_Value   ;Y value after EXECUTE
          DFB   P_Value   ;Processor Status value after EXECUTE
```

The Error byte returned by a STATUS call with status code = $05 (Return UniDisk 3.5 Status) contains the following bits:

| Bit | Description |
|---|---|
| 7 | 0 |
| 6 | 0 |
| 5 | 1 = address field mark or checksum error |
| 4 | 1 = data field checksum error |
| 3 | 1 = data field bitslip mark mismatch |
| 2 | 1 = seek error; unexpected track value found in address field |
| 1 | 0 |
| 0 | 0 |

The Retries byte returned by a STATUS call with status code = $05 (Return UniDisk 3.5 Status) specfies the number of address fields that had to be passed before the operation was completed. This information could be used, for example, to determine the number of passes necessary to read a data field correctly: If Retries is found to be greater than the number of sectors on the target track, then more than one pass was required.

The last four bytes of the status list are set only after a CONTROL call with control code = $05, and are zero after any other call (STATUS calls do not clear the status bytes).

## Possible Errors

The following errors can be returned by the STATUS call:

| | | |
|---|---|---|
| $01 | BADCMD | An unimplemented command was issued |
| $04 | BADPCNT | Bad call parameter count |
| $06 | BUSERR | Communications error |
| $21 | BADCTL | Invalid status code |
| $30-$3F | | Device-specific errors |

## READ BLOCK

**Command Number** $01

**Parameter List**

$03 (parameter count)
Unit number
Data buffer (low byte, high byte)
Block number (low byte, mid byte, high byte)

The READ BLOCK call reads one 512-byte block from the disk device specified by the unit-number parameter into memory starting at the address specified by the data-buffer parameter.

## Parameter Descriptions

**Parameter Count**
1-byte value

3 for this call.

**Unit Number**
1-byte value

The Protocol Converter assigns each device a unique number during initialization (on startup and cold reset). The numbers are in the range $01—$7E, and are assigned according to the devices' positions in the chain. A unit number of $00 in the STATUS call returns the number of devices connected to the Protocol Converter.

**Data Buffer**
2-byte value

Points to the buffer into which the data is read. The buffer must be 512 or more bytes in length.

**Block Number**
3-byte value

The logical address of a block of data to be read. There is no general connection between block numbers and the layout of tracks and sectors on the disk. The translation from logical to physical blocks is performed by the device. (The most significant byte is zero for all devices currently in use.)

## Possible Errors

The following errors can be returned by the READ BLOCK call:

| | | |
|---|---|---|
| $01 | BADCMD | An unimplemented command was issued |
| $04 | BADPCNT | Bad call parameter count |
| $06 | BUSERR | Communications error |
| $27 | IOERROR | I/O error |
| $28 | NODRIVE | No device connected |
| $2D | BADBLOCK | Invalid block number |
| $2F | OFFLINE | Device off-line or no disk in drive |

## WRITE BLOCK

**Command Number** $02

**Parameter List**

$03 (parameter count)
Unit number
Data buffer (low byte, high byte)
Block number (low byte, mid byte, high byte)

The WRITE BLOCK call writes one 512-byte block from memory to the disk device specified by the unit-number parameter. The block in memory starts at the address specified by the data-buffer parameter.

## Parameter Descriptions

**Parameter Count**
1-byte value

3 for this call.

**Unit Number**
1-byte value

The Protocol Converter assigns each device a unique number during initialization (on startup and cold reset). The numbers are in the range $01—$7E, and are assigned according to the devices' positions in the chain. A unit number of $00 in the STATUS call returns the number of devices connected to the Protocol Converter.

**Data Buffer**
2-byte value

Points to the buffer from which the data is to be written.

**Block Number**
3-byte value

The logical address of a block of data to be written. There is no general connection between block numbers and the layout of tracks and sectors on the disk. The translation from logical to physical blocks is performed by the device. (The most significant byte is zero for all devices currently in use.)

## Possible Errors

The following errors can be returned by the WRITE BLOCK call:

| | | |
|---|---|---|
| $01 | BADCMD | An unimplemented command was issued |
| $04 | BADPCNT | Bad call parameter count |
| $06 | BUSERR | Communications error |
| $27 | IOERROR | I/O error |
| $28 | NODRIVE | No device connected |
| $2B | NOWRITE | Disk write protected |
| $2D | BADBLOCK | Invalid block number |
| $2F | OFFLINE | Device off-line or no disk in drive |

## FORMAT

**Command Number** $03

**Parameter List** $01 (parameter count)
Unit number

The FORMAT call prepares all blocks on the recording medium of a block device for reading and writing. The formatting done by this call is not linked to any operating system; for example, bitmaps and catalogs are not written by this call.

### Parameter Descriptions

**Parameter Count**
1-byte value

1 for this call.

**Unit Number**
1-byte value

The Protocol Converter assigns each device a unique number during initialization (on startup and cold reset). The numbers are in the range $01—$7E, and are assigned according to the devices' positions in the chain. A unit number of $00 in the STATUS call returns the number of devices connected to the Protocol Converter.

### Possible Errors

The following errors can be returned by the FORMAT call:

| | | |
|---|---|---|
| $01 | BADCMD | An unimplemented command was issued |
| $04 | BADPCNT | Bad call parameter count |
| $06 | BUSERR | Communications error |
| $27 | IOERROR | I/O error |
| $28 | NODRIVE | No device connected |
| $2B | NOWRITE | Disk write protected |
| $2F | OFFLINE | Device off-line or no disk in drive |

## CONTROL

**Command Number** $04

**Parameter List** $03 (parameter count)
Unit number
Control list (low byte, high byte)
Control code

The CONTROL call sends control information to the device. The information can be of a general nature (such as resets or interrupts), or device-specific (such as Download to UniDisk 3.5 RAM).

Important

A CONTROL call to unit number $00 sends control information to the Protocol Converter itself. See the discussions under Control Code = $00 and Control Code = $01, below.

### Parameter Descriptions

**Parameter Count**
1-byte value

3 for this call.

**Unit Number**
1-byte value

The Protocol Converter assigns each device a unique number during initialization (on startup and cold reset). The numbers are in the range $01—$7E, and are assigned according to the devices' positions in the chain. A unit number of $00 in the STATUS call returns the number of devices connected to the Protocol Converter. Use a unit number of $00 in the CONTROL call to send control information to the Protocol Converter itself.

**Control List**
2-byte value

Points to the buffer from which the control information is read. The first two bytes (the *count bytes*, low byte first) of the control list specify the number of bytes in the list (*not* including the count bytes); the remainder of the list contains the control information passed to the device.

Every CONTROL call must have a control list; if no control information is being passed, then the control list consists of the count bytes only:

```
CTRL_LIST DW $00
```

**Control Code**
1-byte value

The number of the control request being made. Control codes are in the range $00—$FF. The following requests are not device-specific:

| Code | Control Function |
|---|---|
| $00 | Reset the device |
| $01 | Set device control block (DCB) |
| $02 | Set newline status (character devices only) |
| $03 | Service device interrupt |

Control requests to unit number $00 are sent to the Protocol Converter itself:

| Code | Control Function |
|---|---|
| $00 | Enable interrupts from Protocol Converter |
| $01 | Disable interrupts from Protocol Converter |

Specific devices may respond to some or all of these additional control requests:

| Code | Control Function |
|---|---|
| $04 | Eject disk |
| $05 | Run a 65C02 subroutine |
| $06 | Set download address |
| $07 | Download to device RAM |

*Control Code = $00, Reset the Device* Performs a warm reset of the device. Generally returns "housekeeping" values to some reset value. The control list for this call is device dependent.

UniDisk 3.5

The control list for this call for UniDisk 3.5 devices is:

```
CTRL_LIST DW $00 ;No parameters are passed
```

*Unit Number $00:* A CONTROL call with control code = $00 and unit number = $00 enables interrupts from the Protocol Converter. This call informs the firmware that external interrupts are possible, and directs it to call the user's interrupt handler if an interrupt occurs. It also turns on the Asynchronous Communications Interface Adapter (ACIA) for port 1.

When the user's interrupt handler identifies an external interrupt, you can determine if it came from the Protocol Converter by making a STATUS call with unit number = $00 and control code = $00 (see Section "STATUS"). See Appendix E in the *Apple IIc Reference Manual* for more information on handling interrupts.

*Control Code = $01, Set Device Control Block* Alters the contents of the device control block (DCB). The DCB is usually used to set global aspects of a device's operating environment. Each device has a default setting for the DCB, set on initialization. Since the length of the DCB is device dependent, you should first read in the DCB with the STATUS call, then alter the bits of interest, and finally, use the same byte string as the control block for the CONTROL call. The first byte (the *count byte*) of the DCB gives the number of bytes in the control block (*not* including the count byte), so the length never exceeds 257 bytes, including the count byte.

UniDisk 3.5

UniDisk 3.5 has no DCB; a Set DCB CONTROL call to UniDisk 3.5 returns an error (BADCTL $21).

*Unit Number = $00:* A CONTROL call with control code = $01 and unit number = $00 disables interrupts from the Protocol Converter. This call turns off the ACIA for port 1 and sets the least significant bit of the ACIA control register to zero.

Newline read mode: See Chapter 4 in the *PRODOS Technical Reference Manual.*

*Control Code = $02, Set Newline Status* Sets a character device to newline enabled or newline disabled.

*Control Code = $03, Device Service Interrupt* To be used as needed for interrupt-driven devices.

*Control Code = $04, Eject Disk* To be used for devices that support an auto-eject feature.

UniDisk 3.5

Causes UniDisk 3.5 to auto-eject a disk. There are no parameters in the control list, and no errors are returned if the disk ejected correctly or there was no disk in the drive. Error code $27 (I/O error) is returned if the eject failed, that is, a disk is still in the drive. The control list for UniDisk 3.5 is:

```
CTRL_LIST DW $00 ;No parameters are passed
```

▲Warning [illegible]

## Possible Errors

The following errors can be returned by the CONTROL call:

| | | |
|---|---|---|
| $01 | BADCMD | An unimplemented command was issued |
| $04 | BADPCNT | Bad call parameter count |
| $06 | BUSERR | Communications error |
| $21 | BADCTL | Invalid control code |
| $22 | BADCTLPARM | Invalid parameter list |
| $30-$3F | | Device-specific errors |

## INIT

| | |
|---|---|
| **Command Number** | $05 |
| **Parameter List** | $01 (parameter count)<br>$00 (unit number) |

The INIT call resets all intelligent devices attached to the Protocol Converter. The Protocol Converter goes through an initialization sequence, cold-resetting all devices and sending each its unit number. This call is made automatically on startup; an application should never have to make this call.

### Parameter Descriptions

**Parameter Count**
1-byte value

1 for this call.

**Unit Number**
1-byte value

The unit number used in this call is always $00.

### Possible Errors

The following errors can be returned by the INIT call:

| | | |
|---|---|---|
| $01 | BADCMD | An unimplemented command was issued |
| $04 | BADPCNT | Bad call parameter count |
| $06 | BUSERR | Communications error |
| $28 | NODRIVE | No device connected |

## OPEN

| | |
|---|---|
| **Command Number** | $06 |
| **Parameter List** | $01 (parameter count)<br>Unit number |

The OPEN call prepares a character device for reading or writing.

UniDisk 3.5

Since UniDisk 3.5 is a block device, it does not accept this call. An attempt to use an OPEN call with UniDisk 3.5 will result in an error (BADCMD $01).

### Parameter Descriptions

**Parameter Count**
1-byte value

1 for this call.

**Unit Number**
1-byte value

The Protocol Converter assigns each device a unique number during initialization (on startup and cold reset). The numbers are in the range $01—$7E, and are assigned according to the devices' positions in the chain. A unit number of $00 in the STATUS call returns the number of devices connected to the Protocol Converter.

### Possible Errors

The following errors can be returned by the OPEN call:

| | | |
|---|---|---|
| $01 | BADCMD | An unimplemented command was issued |
| $04 | BADPCNT | Bad call parameter count |
| $06 | BUSERR | Communications error |
| $28 | NODRIVE | No device connected |
| $2F | OFFLINE | Device off-line or no disk in drive |

## CLOSE

| | |
|---|---|
| **Command Number** | $07 |
| **Parameter List** | $01 (parameter count)<br>Unit number |

The CLOSE call tells a character device that a sequence of reads or writes is over.

UniDisk 3.5 | Since UniDisk 3.5 is a block device, it does not accept this call. An attempt to use a CLOSE call with UniDisk 3.5 will result in an error (BADCMD $01).

### Parameter Descriptions

**Parameter Count**
1-byte value

1 for this call.

**Unit Number**
1-byte value

The Protocol Converter assigns each device a unique number during initialization (on startup and cold reset). The numbers are in the range $01—$7E, and are assigned according to the devices' positions in the chain. A unit number of $00 in the STATUS call returns the number of devices connected to the Protocol Converter.

### Possible Errors

The following errors can be returned by the CLOSE call:

| | | |
|---|---|---|
| $01 | BADCMD | An unimplemented command was issued |
| $04 | BADPCNT | Bad call parameter count |
| $06 | BUSERR | Communications error |
| $28 | NODRIVE | No device connected |
| $2F | OFFLINE | Device off-line or no disk in drive |

## READ

**Command Number** $08

**Parameter List**

$04 (parameter count)
Unit number
Buffer pointer (low byte, high byte)
Byte count (low byte, high byte)
Address pointer (low byte, mid byte, high byte)

The READ call reads the number of bytes specified by the byte-count parameter into memory starting at the address specified by the buffer-pointer parameter.

*Macintosh:* This call can be used by UniDisk 3.5 devices to read 524-byte data blocks written by an Apple Macintosh™ Computer.

## Parameter Descriptions

**Parameter Count**
1-byte value

4 for this call.

**Unit Number**
1-byte value

The Protocol Converter assigns each device a unique number during initialization (on startup and cold reset). The numbers are in the range $01—$7E, and are assigned according to the devices' positions in the chain. A unit number of $00 in the STATUS call returns the number of devices connected to the Protocol Converter.

**Buffer Pointer**
2-byte value

Points to the buffer into which the data is read. The buffer must be large enough to contain the number of bytes requested by the byte-count parameter.

**Byte Count**
2-byte value

Specifies the number of bytes to be transferred.

*Macintosh:* The byte count used to read Macintosh disks with a UniDisk 3.5 is always 524 bytes ($020C).

**Address Pointer**
3-byte value

Specifies the address to start reading from. The meaning of this parameter depends on the device being read.

*Macintosh:* When using a UniDisk 3.5 to read Macintosh disks, the address pointer specifies the number of the 524-byte Macintosh block to be read (from $00 to $031F for a single-sided disk).

## Possible Errors

The following errors can be returned by the READ call:

| | | |
|---|---|---|
| $01 | BADCMD | An unimplemented command was issued |
| $04 | BADPCNT | Bad call parameter count |
| $06 | BUSERR | Communications error |
| $27 | IOERROR | I/O error |
| $28 | NODRIVE | No device connected |
| $2D | BADBLOCK | Invalid block number |
| $2F | OFFLINE | Device off-line or no disk in drive |

## WRITE

**Command Number** $09

**Parameter List**

$04 (parameter count)
Unit number
Buffer pointer (low byte, high byte)
Byte count (low byte, high byte)
Address pointer (low byte, mid byte, high byte)

The WRITE call writes the number of bytes specified by the byte-count parameter to the specified unit from memory starting at the address indicated by the buffer-pointer parameter. The meaning of the address pointer depends on the type of device (see the parameter descriptions, below).

*Macintosh:* This call can be used by UniDisk 3.5 devices to write 524-byte blocks for use by an Apple Macintosh computer.

## Parameter Descriptions

**Parameter Count**
1-byte value

4 for this call.

**Unit Number**
1-byte value

The Protocol Converter assigns each device a unique number during initialization (on startup and cold reset). The numbers are in the range $01—$7E, and are assigned according to the devices' positions in the chain. A unit number of $00 in the STATUS call returns the number of devices connected to the Protocol Converter.

**Buffer Pointer**
2-byte value

Points to the buffer from which the data is to be written.

**Byte Count**
2-byte value

Specifies the number of bytes to be transferred.

*Macintosh:* The byte count used to write Macintosh disks with a UniDisk 3.5 is always 524 bytes ($020C).

**Address Pointer**
3-byte value

Specifies the address to start writing from. The meaning of this parameter depends on the device being written to.

*Macintosh:* When using a UniDisk 3.5 to write Macintosh disks, the address pointer specifies the number of the 524-byte Macintosh block to be written (from $00 to $031F for a single-sided disk).

## Possible Errors

The following errors can be returned by the WRITE call:

| | | |
|---|---|---|
| $01 | BADCMD | An unimplemented command was issued |
| $04 | BADPCNT | Bad call parameter count |
| $06 | BUSERR | Communications error |
| $27 | IOERROR | I/O error |
| $28 | NODRIVE | No device connected |
| $2D | BADBLOCK | Invalid block number |
| $2F | OFFLINE | Device off-line or no disk in drive |

## An Example: Issuing a Protocol Converter Call

Here is an example of a program that issues a STATUS call to the Protocol Converter to obtain information about a device.

**Apple IIe**

The code for the Protocol Converter in the Apple IIc with 32K ROM always begins at address $C500; however, to ensure compatibility with the Apple IIe, your programs should always do a search for the Protocol Converter, as in the following example.

```
0000:                    1 *
0000:                    2 *
0000:                    3 *
0000:                    4 * This example shows how to find
0000:                    5 *  and use a PC interface.  A search
0000:                    6 *  is made for a PC, and when one is
0000:                    7 *  found, a vector is set up which
0000:                    8 *  points to the PC entry.  Then a
0000:                    9 *  Device Information Block STATUS call
0000:                   10 *  is made, and if successful, the name
0000:                   11 *  string embedded in the DIB is output
0000:                   12 *  to the screen. Only the first device
0000:                   13 *  in the chain is accessed.
0000:                   14 *
0000:                   15 *
0000:                   16          MSB   ON
0000:                   17 *
0000:                   18 *
0000:          0006     19 ZPTempL  equ   $0006     ;Temporary zero
0000:                   20 *                          page storage
0000:          0007     21 ZPTempH  equ   $0007
0000:                   22 *
0000:          FDED     23 COut     equ   $FDED     ;Console output
0000:          FD8E     24 CROut    equ   $FD8E     ;Carriage return
0000:                   25 *
0000:          0000     26 StatusCmd equ  0
0000:                   27 *
0000:                   28 *
0300:          0300     29          org   $300
0300:                   30 *
0300:                   31 * Find a Protocol Converter in one of the
0300:                   32 *  slots.
0300:                   33 *
0300:20 43 03           34          jsr   FindPC
0303:B0 1C   0321       35          bcs   Error
0305:                   36 *
0305:                   37 * Now make the DIB call to the first guy
0305:                   38 *
```

```
0305:20 67 03        39              jsr   Dispatch
0308:00              40              dfb   StatusCmd
0309:6A 03           41              dw    DParms
030B:B0 14   0321    42              bcs   Error
030D:                43 *
030D:                44 * Got the DIB; now print the name string
030D:                45 *
030D:A2 00           46              ldx   #0
030F:          030F  47 morechars    equ   *
030F:BD 74 03        48              lda   DIBName,x
0312:09 80           49              ora   #$80      ;COut wants high
0314                 50 *                            Bit set
0314:                51 *
0314:20 ED FD        52              jsr   COut
0317:E8              53              inx
0318:EC 73 03        54              cpx   DIBNameLen
031B:90 F2   030F    55              blt   morechars
031D:                56 *
031D:20 8E FD        57              jsr   CROut     ;Finish it off
0320:                58 *                            with a return
0320:                59 *
0320:60              60              rts
0321:                61 *
0321:                62 *
0321:          0321  63 Error        equ   *
0321:                64 *
0321:                65 * There's either no PC around, or there
0321:                66 * was no Unit #1... give message
0321:                67 *
0321:A2 00           68              ldx   #0
0323:          0323  69 err1         equ   *
0323:BD 2F 03        70              lda   Message,x
0326:F0 06   032E    71              beq   errout
0328:20 ED FD        72              jsr   COut
032B:E8              73              inx
032C:D0 F5   0323    74              bne   err1
032E:                75 *
032E:          032E  76 errout       equ   *
032E:60              77              rts
032F:                78 *
032F:CE CF A0 D0     79 Message      asc   'NO PC OR NO DEVICE'
0341:8D 00           80              dfb   $8D,0
0343:                81 *
0343:                82 *
0343:          0343  83 FindPC       equ   *
0343:                84 *
0343:                85 * Search slot 7 to slot 1 looking for
0343:                86 *  signature bytes
0343:                87 *
0343:A2 07           88              ldx   #7        ;Do for seven
0345:                89 *                            slots
0345:A9 C7           90              lda   #$C7
```

```
0347:85 07          91              sta   ZPTempH
0349:A9 00          92              lda   #$00
034B:85 06          93              sta   ZPTempL
034D:               94 *
034D:         034D  95 newslot      equ   *
034D:A0 07          96              ldy   #7
034F:               97 *
034F:         034F  98 again        equ   *
034F:B1 06          99              lda   (ZPTempL),y
0351:D9 70 03       100             cmp   sigtab,y      ;One of four
0354:               101 *                               byte signature
0354:F0 07    035D  102             beq   maybe         ;Found one
0356:               103 *                               signature byte
0356:C6 07          104             dec   ZPTempH
0358:CA             105             dex
0359:D0 F2    034D  106             bne   newslot
035B:               107 *
035B:               108 * If we get here, it's because we couldn't
035B:               109 *  find a Protocol Converter.
035B:               110 * Exit with the carry set.
035B:               111 *
035B:38             112             sec
035C:60             113             rts
035D:               114 *
035D:               115 * If we get here, it means that one or
035D:               116 *  more of the signature bytes
035D:               117 *  for this card are what we're looking
035D:               118 *  for.  Decrement the byte
035D:               119 *  counter and branch back to verify any
035D:               120 *  remaining bytes.
035D:               121 *
035D:         035D  122 maybe       equ   *
035D:88             123             dey
035E:88             124             dey                 ;If N=1 then
035F:               125 *                               all sig bytes okay
035F:10 EE    034F  126             bpl   again
0361:               127 *
0361:               128 * Found a Protocol Converter interface.
0361:               129 *  Set up the call address.
0361:               130 * We already have the high byte ($CN);
0361:               131 *  we just need the low byte.
0361:               132 *
0361:         0361  133 foundPC     equ   *
0361:A9 FF          134             lda   #$FF
0363:85 06          135             sta   ZPTempL
0365:A0 00          136             ldy   #0            ;For
0367:               137 *                               indirect load
0367:B1 06          138             lda   (ZPTempL),y   ;Get the
0369:               139 *                                 byte
0369:               140 *
0369:               141 * Now the Acc has the low order ProDOS
0369:               142 *  entry point.  The PC entry is
```

```
0369:                   143 *  three locations past this...
0369:                   144 *
0369:18                 145            clc
036A:69 03              146            adc   #3
036C:85 06              147            sta   ZPTempL
036E:                   148 *
036E:                   149 * Now ZPTempL has the PC entry point.
036E:                   150 *  Return with carry clear.
036E:                   151 *
036E:18                 152            clc
036F:60                 153            rts
0370:                   154 *
0370:                   155 *
0370:                   156 * These are the PC signature bytes in
0370:                   157 *  their relative order.
0370:                   158 *  The $FF bytes are filler bytes and
0370:                   159 *  are not compared.
0370:                   160 *
0370:FF 20 FF 00        161 sigtab    dfb   $FF,$20,$FF,$00
0374:FF 03 FF 00        162           dfb   $FF,$03,$FF,$00
0378:                   163 *
0378:                   164 *
0378:         0378      165 Dispatch  equ   *
0378:6C 06 00           166           jmp   (ZPTempL)     ;Simulate
037B:                   167 *                   an indirect JSR to PC
037B:                   168 *
037B:                   169 *
037B:         037B      170 DParms    equ   *
037B:03                 171 DPParmCt  dfb   3             ;Status
037C:                   172 *             calls have three parameters
037C:01                 173 DPUnit    dfb   1
037D:80 03              174 DPBuffer  dw    DIB
037F:03                 175 DPStatCode dfb  3
0380:                   176 *
0380:                   177 *
0380:         0380      178 DIB       equ   *
0380:00                 179 DIBStatByte1 dfb 0
0381:00 00 00           180 DIBDevSize dfb  0,0,0
0384:00                 181 DIBNameLen dfb  0
0385:         0010      182 DIBName   ds    16,0
0395:00                 183 DIBType   dfb   0
0396:00                 184 DIBSubType dfb  0
0397:00 00              185 DIBVersion dw   0
0399:                   186 *
0399:                   187 *
```

## Summary of Commands and Parameters

This is a summary of Protocol Converter calls. In each case, byte 0 of the command parameter list (CMDLST) specifies the number of parameters in the command list (not including byte 0). Parameters that require more than one byte (the status list pointer, for example) are entered low byte first. The meaning of the address-pointer parameter is device specific. See the sections on the individual calls in this chapter for a discussion of each parameter.

*Figure 3-1.* Summary of Protocol Converter Commands and Parameters

| **Command** | **STATUS** | **READBLOCK** | **WRITEBLOCK** | **FORMAT** | **CONTROL** |
|---|---|---|---|---|---|
| **CmdNum** | $00 | $01 | $02 | $03 | $04 |
| **CmdList Byte** | | | | | |
| 0 | $03 | $03 | $03 | $01 | $03 |
| 1 | Unit Num | Unit Num | Unit Num | Unit Num | Unit Num |
| 2 | Stat List Ptr | Buffer Ptr | Buffer Ptr | | Ctl List Ptr |
| 3 | | | | | |
| 4 | Stat Code | Block Num | Block Num | | Ctl Code |
| 5 | | | | | |
| 6 | | | | | |

| **Command** | **INIT** | **OPEN** | **CLOSE** | **READ** | **WRITE** |
|---|---|---|---|---|---|
| **CmdNum** | $05 | $06 | $07 | $08 | $09 |
| **CmdList Byte** | | | | | |
| 0 | $01 | $01 | $01 | $04 | $04 |
| 1 | $00 | Unit Num | Unit Num | Unit Num | Unit Num |
| 2 | | | | Buffer Ptr | Buffer Ptr |
| 3 | | | | | |
| 4 | | | | Byte Count | Byte Count |
| 5 | | | | | |
| 6 | | | | | |
| 7 | | | | Address Ptr | Address Ptr |
| 8 | | | | | |

Unused bytes

## Summary of Error Codes

This is a summary of Protocol Converter call error codes, including a brief description of the possible causes for each. If there is no error, the C flag (in the Processor Status register of the 65C02 microprocessor) is cleared (0), and the accumulator (the A register) contains zeros. If the call was unsuccessful, the C flag is set (1), and the A register contains the error code.

| | | |
|---|---|---|
| $00 | | No error. |
| $01 | BADCMD | A nonexistent command was issued. Check the command number in the Protocol Converter call. |
| $04 | BADPCNT | Bad call parameter count. The call parameter list was not properly constructed. Make sure the parameter list has the correct number of parameters. |
| $06 | BUSERR | A communications error between the device controller and the host. Make sure that RAM is both read-enabled and write-enabled. Check the hardware (cables and connectors) between the device and the host. Check for noise sources; make sure the cable is properly shielded. |
| $11 | BADUNIT | Unit number $00 was used in a call other than STATUS, CONTROL, or INIT. |
| $21 | BADCTL | The control or status code is not supported by the device. |
| $22 | BADCTLPARM | The control parameter list contains invalid information. Make sure each value is within the range allowed for that parameter. |
| $27 | IOERROR | The device encountered an I/O error when trying to read or write to the recording medium. Make sure that the medium in the device is formatted and not defective. Make sure the device is operating correctly. |
| $28 | NODRIVE | The device is not connected. This can occur if the device is not connected but its controller is, or if there is no device with the unit number specified. |

| | | |
|---|---|---|
| $2B | NOWRITE | The medium in the device is write protected. |
| $2D | BADBLOCK | The block number is outside the range allowed for the medium in the device. Note that this range depends on the type of device and the type of medium in the device (single-sided vs. double-sided disk, for example). |
| $2F | OFFLINE | Device off-line or no disk in drive. Check the cables and connections; make sure the medium is present in the drive, and that the drive is functioning correctly. |
| $30-$3F | DEVSPEC | Errors which differ from device to device. See the technical manual for the device in question for details. |
| $40-$4F | | Reserved for future expansion. |
| $50-$7F | NONFATAL | A device-specific *soft* error. The operation completed successfully, but some *exception* condition was detected. See the technical manual for the device in question for details. |