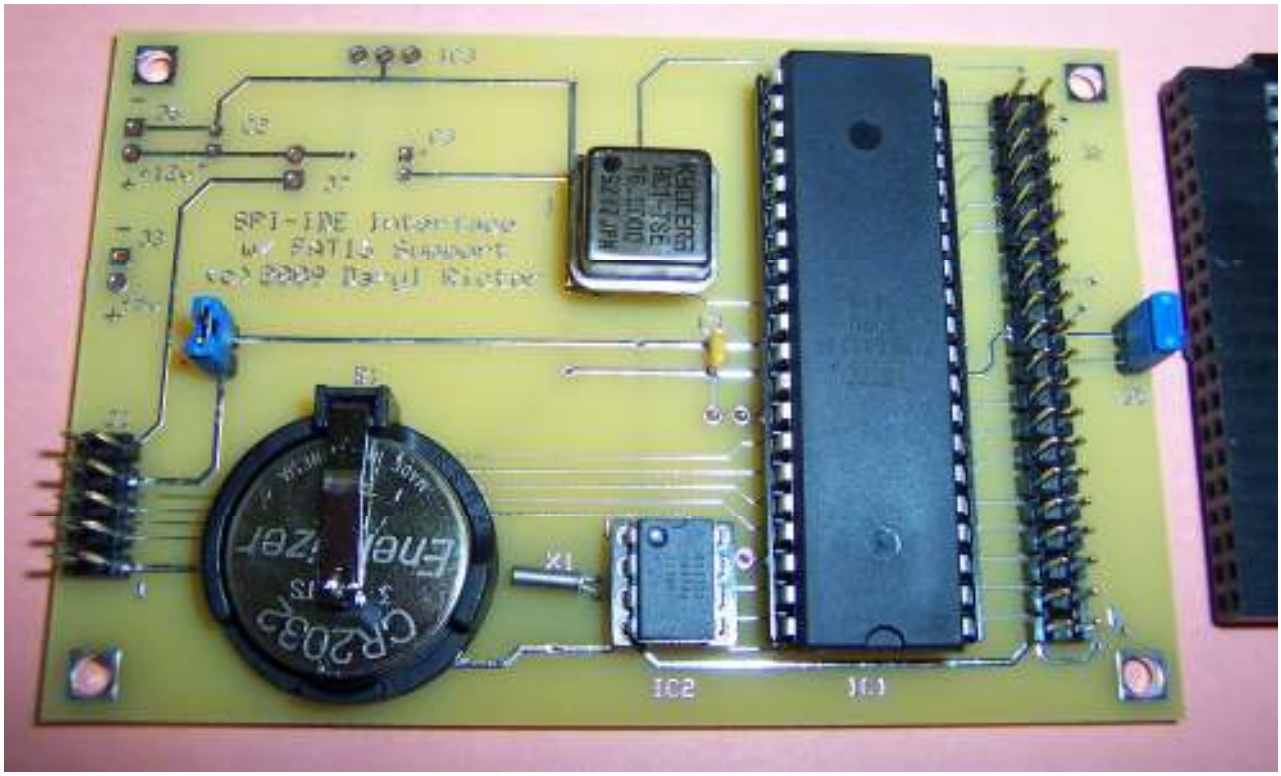
SPI to IDE/CF Interface with FAT16 support

V2.1

Designed by Daryl Rictor



Version History

1.0 – May 5, 2009 – First Release
1.1 – May 31, 2009 – Removed 65SIB connectors from PCB
1.2 - June 5, 2009 – Added more SPI diagrams
1.3 - June 7, 2009 – Change microcontroller to the ATMega324P
1.4 - Jan 21, 2010 – Fixed a bug in the Host Get File Pointer command
1.5 - March 20, 2010 – fixed bug in the FatFSeek routine - added SEEK_END mode
1.6 – no date - another fix to the FatFSeek routine
1.7 – no date - fixed "file open append" to properly point to the end of file
1.8 – Jan 23, 2012 - Added init for "dessectorHasChanged" in FatInit - fixed lockup on DumpDriveInfo also fixed delay timing for 20MHz clock vs. 16MHz
1.9 – Aug 25, 2012 - Fixed Get RTC Register command to match the datasheet {e2}{dd} now works.
2.0 – Sep 20, 2012 – Added busy flag/LED driver to pin 4 (PB3) – Active low when busy
2.1 – Feb 16, 2013 – Allows CF card larger than to 2GB to be formatted with a 2GB first partition.

Contents

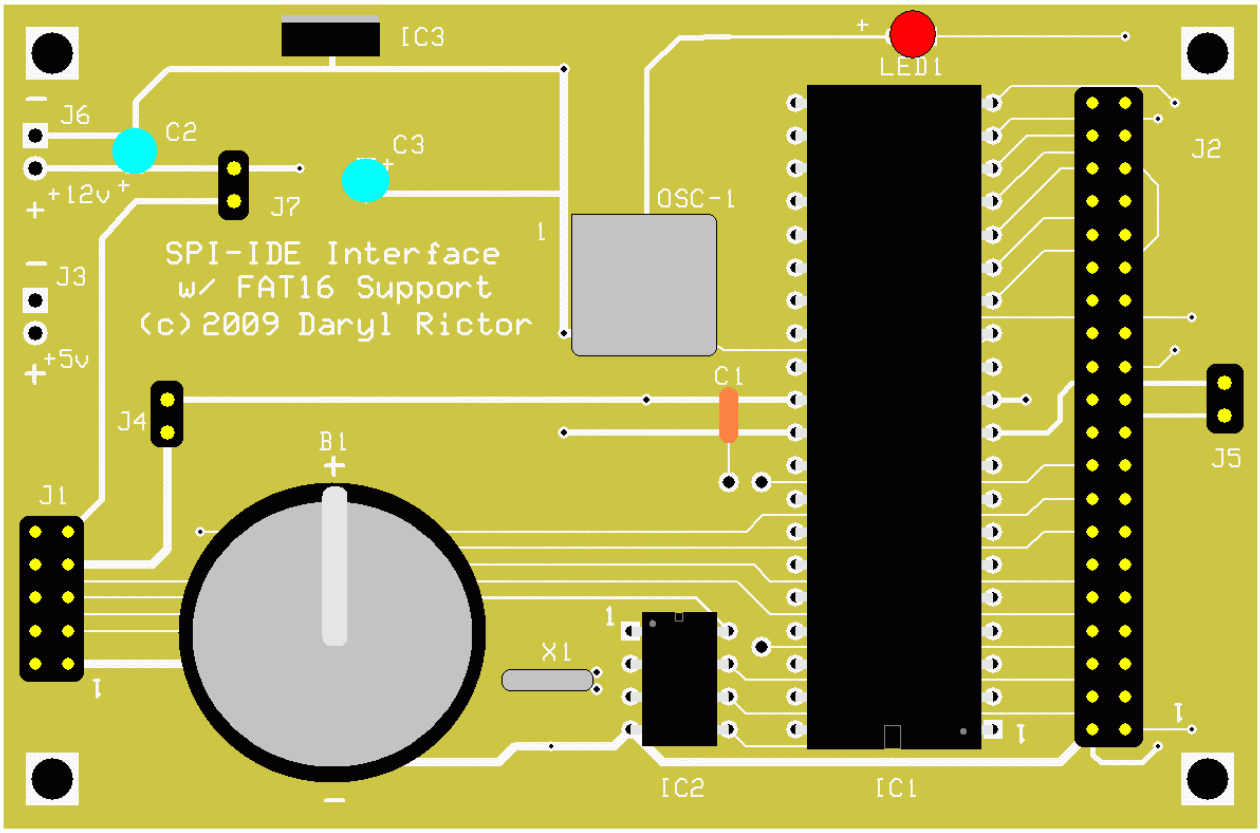
Introduction

This package was designed to provide an easy interface to an ATA hard drive or Compact Flash (CF). It further provides the software drivers to support the FAT16 data format. This allows removable CF media to be exchanged between any PC and the host system.

The FAT16 and ATA drivers were inspired and adapted from Angelo Bannack and Giordano Bruno Wolaniuk's FAT 16/32 File System Driver for Atmel AVR, from Circuit Cellar Magazine's AVR 2004 Design Contest. Their firmware was written in C while mine is written in assembly. This increases speed and reduces memory requirements.

Disclaimer and copyright notice

Hardware



The hardware required for this interface is an Atmel ATMega324P micro controller, 20MHz TTL oscillator, a 40-pin IDE header, and a 10-pin SPI header. Both headers connect directly to I/O pins on the ATMega324P. You can connect an ATA hard drive using a standard 40-pin ribbon cable, or use a Compact Flash to IDE adapter and use Compact Flash media. The CF media can run on +5v supplied through the SPI connector. An ATA hard Drive will require an external +5VDC, and possibly a +12VDC supply, to function correctly.

Optional equipment includes:

- On-board 5v regulator
- A disk activity LED with built-in resistor
- A real-time clock circuit; consisting of a DS1302 RTC IC, 32khz crystal, and a CR2032 battery and battery holder.

ATMEga324P

The micro controller used is an Atmel ATMEga324P in a 40-pin DIP package. This micro controller can be clocked up to 20MHz and includes 32K of Flash program space, 2K of SRAM, an SPI interface port, and enough I/O pins to connect the IDE and RTC circuits directly. The current firmware is only using about 11k of program space so there is plenty of room to add custom code if desired.

DS1302

The real-time clock uses a Dallas Semiconductor DS1302 in an 8-pin DIP package. This clock supports date and time functions via a 3-wire interface. It requires a 32khz watch crystal to provide the time source and a CR2032 battery to keep it running when power is removed. The firmware does simple bit banging of the I/O pins to communicate with the DS1302.

IDE-CF adapter

I recommend the use of an IDE to CF adapter and CF media to simplify your circuit. CF media power requirements are low enough to allow the whole board to draw power from the Host through the SPI port. There are several inexpensive adapters available online. Don't forget to check E-bay!

SPI Ports

There is a 10-pin SPI header for connection to a Host System. Here is the pinout:

10- +9 to +12v	9- +9 to +12v
8- +5V	7- +5V
6- SS	5- MOSI
4- MISO	3- SCLK
2- Ground	1- Ground

Jumper Bocks and External Power

There are also a few jumpers on the board. J4 is used to provide the system +5V from the SPI host. Alternatively, you can remove J4 and use J3 to supply +5V from an external source.

J5 can be used to supply +5V to pin 20 of the IDE connector. Pin 20 is normally the missing pin (keyed pin) used to prevent you from installing the cable backwards. Some CF adapters and 3.5" to 2.5" IDE adapters use pin 20 to pass +5V to the drive. Consult your hardware documents to find out if your devices support this. If in doubt, do NOT install J5.

You can use the optional on-board voltage regulator by using J7 to get +9 to +12 volts from the SPI port. Alternatively, you can use J6 to provide external +9 to +12 volts. Do not use J7.

To prevent damage, be sure that power is provide by only one source and that the other option blocks are not installed.

Software

I will use the following template to display data in this document:

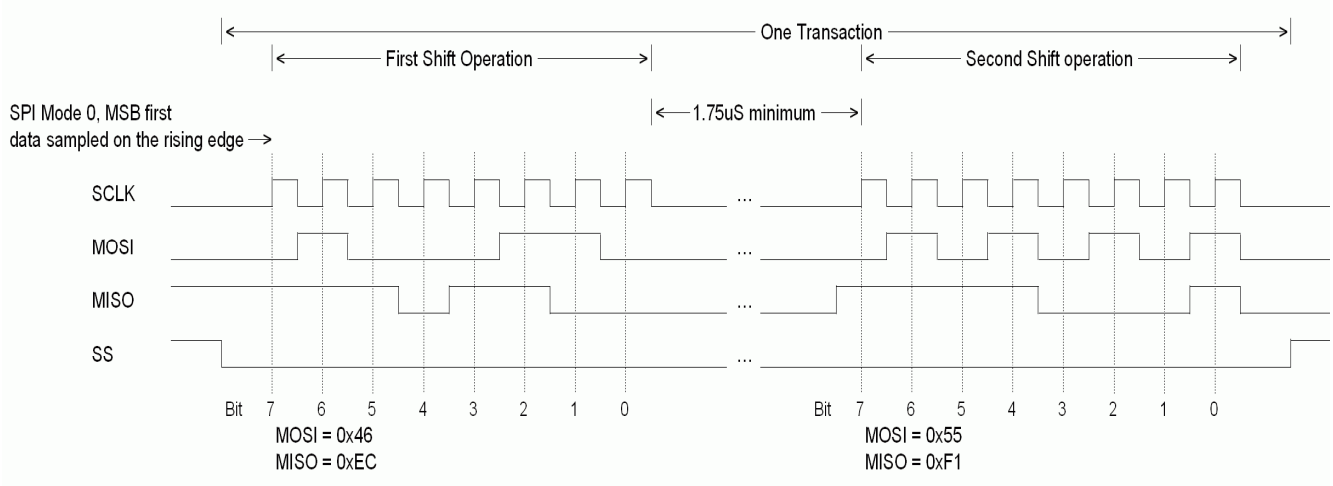
{xx} = Hexadecimal byte
[xxx] = ASCII text byte(s)

The primary host connection uses a standard SPI interface. The SPI port uses Mode 0 with the most-significant bit sent first. All SPI transactions consist of two shift operations. The first sends the command from the host to the interface. The second either sends data to or receives data from the interface. The maximum frequency of the shift clock is 5MHz, using a 20MHz clock for the Atmega324P. You will need to wait at least 1.75 microseconds after the first shift operation ends before initiating the second shift cycle. This gives the interface time to decode the command.

This is the SPI Transaction Map. It shows the relationship between the first and second shift operation for each of the three transaction types.

	1st Shift operation		2nd Shift operation	
Transaction	**Transmitted**	**Received**	**Transmitted**	**Received**
Send Data	0x01	Undefined	data out	undefined
Get Status	0x02	Undefined	0x00	status byte
Get Data	0x03	Undefined	0x00	data in

The diagram below shows the timing relationships.



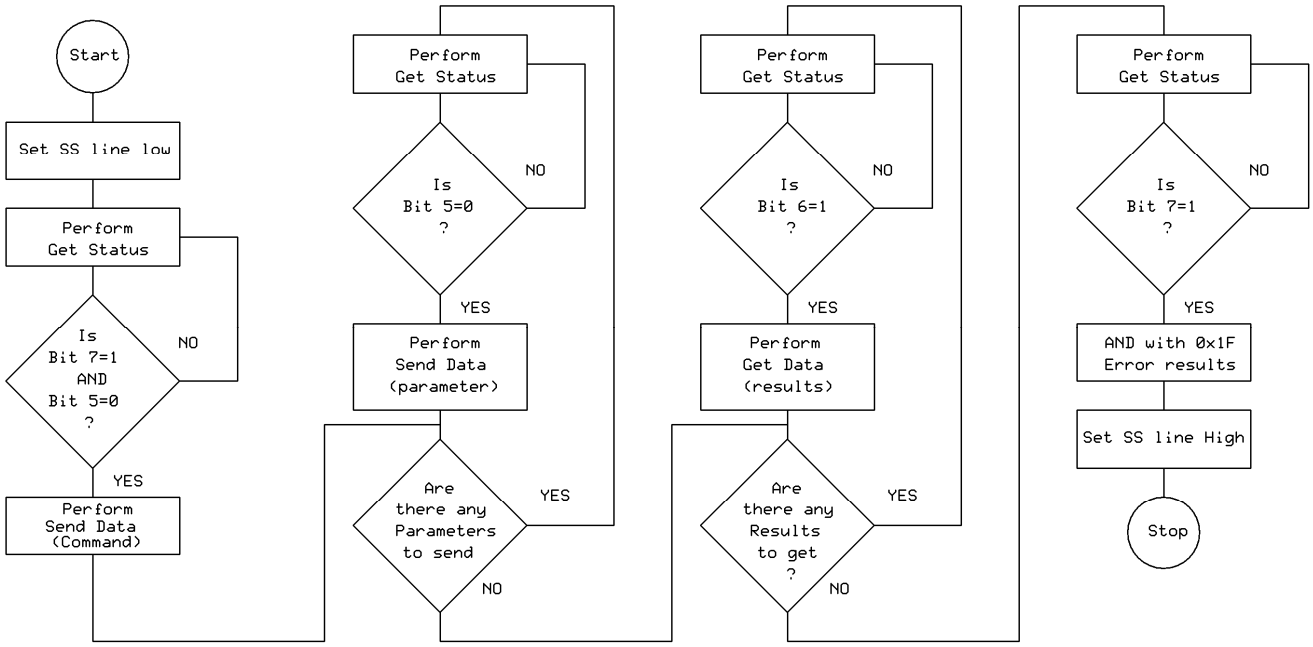
The Status Byte contains both SPI port status along with any error codes returned from the IDE interface. The status byte is formatted like this:

Bit 7 - Ready for Command if 1
Bit 6 - Data Ready to send to the Host when 1
Bit 5 - Ready to receive data from the host when 0

bit 4-0 contain a binary error code. These are the error codes:

```
Binary  Hex  Definition
43210
00000 - 00 - No Error, command successful
00001 - 01 - ATA init error
00010 - 02 - missing file name
00011 - 03 - format error
00100 - 04 - file not found
00101 - 05 - read-only file
00110 - 06 - invalid file type/open error
00111 - 07 - unable to erase all files
01000 - 08 - file not open
01001 - 09 - file already open in another handle
01010 - 0A - file protected
01011 - 0B - disk full
01100 - 0C - root directory full
01101 - 0D - invalid command
01110 - 0E - handle in use
01111 - 0F - read past EOF
10000 - 10 - File too large > 2^32
10001 - 11 - Cluster size too big
```

Here is a flowchart showing the basic communication process:



The ATMega324P firmware supports the following simple commands: (see the Command Descriptions section for a detailed description of these commands)

{C0} - Close all files
{C1} - Return Directory listing
{C2} - Make Directory
{C3} - Change Directory
{C4} - Erase File or Dir
{C5} - Erase all
{C6} - Protect file/dir
{C7} - Unprotect file/dir
{C8} - Rename File or Dir
{C9} - Open File
{CA} - Read File
{CB} - Write File
{CC} - Set file pointer
{CD} – Get file pointer
{CE} - Close File
{CF} - Test File
{D0} - File Size
{D1} - Read sector (direct mode)
{D2} - Write sector (direct mode)
{E0} - Get Calendar
{E1} - Set calendar
{E2} - Get RTC register direct
{F0} - Return Volume Name
{F1} - Return free clusters
{F2} - Format CF
{F3} - Return drive info
{F4} - Device reset (soft)

You access files for read, write, or append using handles. You can have up to 16 files opened at once, using handles {0} through {F}. The handles keep track of which file you want interact with. So, once you open a file, the handle number controls the subsequent reads and/or writes. It works like this:

Open file "test1.dat" for reading using handle 1
Open file "test2.dat" for reading using handle 2
Open file "test3.dat" for reading using handle 3
Open file "test4.dat" for reading using handle 4
Open file "test5.dat" for reading using handle 5
Open file "test6.dat" for reading using handle 6
Open file "output.dat" for writing using handle 7
read 10 bytes from handle 1
write 5 bytes to handle 7
read 20 bytes from handle 2
write 5 bytes to handle 7
read 12 bytes from handle 3
write 5 bytes to handle 7
read 1 bytes from handle 4
write 5 bytes to handle 7
read 100000 bytes from handle 5
write 50000 bytes to handle 7
read 1 bytes from handle 6
write 200000 bytes to handle 7
close handle 1
close handle 2
close handle 3
close handle 4
close handle 5
close handle 6
close handle 7

Directory operations and rename/delete file operations do not use handles.

The calendar functions are used to read and write the data & time. They can be used by the host for non-disk functions. When creating or modifying files or directories, the firmware will automatically get the current date and time and store it. If the DS1302 is missing or not functioning, the firmware will use the default date of April 11, 2009 2:04pm.

There are a few limitations to keep in mind. Media larger than 2GB can now be installed. However, you will need to format it so a 2GB FAT16 partition can be created before using it. The remainder of the free space will be unallocated. The current version only supports one (the first) partition. You may connect two devices to the IDE cable, and access both, but only one at a time as there is not enough on-board RAM to support both. You may have up to 16 files open at a time. Again, this is limited due to RAM size. FAT16 supports the old 8.3 file name convention. That is 8 characters with a 3-character extension. Examples are "sample.txt", "1.gif", "mainpage.dat", and "datafile". I do not enforce all of the character limitations that DOS does. For example, characters + & " ' ` : ; > < are not valid DOS characters but my firmware will allow them. Please keep in mind that if you move the media to a DOS/Windows machine, you may have trouble with the OS reading it if you do not follow the DOS rules. Finally, when changing directories, you must do so one level at a time. To move back, you use ".." You can move to the Root directory by using the "/" character as the name. Please note, use the forward slash, not the backward slash.

Command Descriptions

Change Directory {C3}

This command will move you to a different directory.
Note, you can only go one level forward or backward at a time.
You may also go directly to the Root Directory, by using the "/" character as the filename.

The syntax is:
{C3}[directory name]{00} moves into subdirectory
{C3}[..]{00} moves back one level
{C3}[.]{00} does nothing
{C3}[/]{00} moves to the Root directory

The return status can be {00}, {02}, {04}, {06}

Close all files {C0}

This command is issued to close all open files. This should be used before you issue the Mount Volume {F4} command to ensure no files are corrupted.

The syntax is {C0}
The return status can be {00} or {08}

Close File {CE}

This command will close the file of the handle given.

The syntax is: {CE}{handle}

The return status can be {00}, {08}

Erase all {C5}

Caution! Use with extreme care!

This command will erase ALL files and directories from the current directory.
It will erase protected files and remove directories, even if they are not empty!

The syntax is: {C5}

The return status can be {00}, {07}

Erase File or Dir {C4}

This command will erase a file or directory from the current directory. The file must be closed and not protected.
A directory must be empty before it can be deleted.

The syntax is: {C4}[file or directory name]{00}

The return status can be {00}, {02}, {04}, {06}, {07}, {09}, {0A}

File Size {D0}

This command will return the size in bytes of the handle given.

The syntax is: {D0}{handle}

The controller will return the 4-byte size: {w}{x}{y}{z}
Where:
w = the least significant byte of the 4 byte length
x = is the 2nd significant byte of the 4 byte length
y = is the 3rd significant byte of the 4 byte length
z = is the most significant byte of the 4 byte length

The return status can be {00}, {08}

Format CF {F2}

This command will format a device with a FAT16 file format. This is a quick format, where only the header sectors and FAT tables are initialized. You can still use the raw sector access commands to read any old data stored on the device.

The syntax is: {F2}

The return status can be {00}, {01}, {03}

Get Calendar {E0}

This command will return the current date & time in null-terminated text string. If the DS1302 is missing or not functioning, the firmware will return the default date of 4/11/09 14:04:00, or the last value entered using the Set Calendar function.

The syntax is: {E0}

It returns the 19-byte value [mm/dd/yy hh:mm:ss]{0}

The return status is {00}

Get file pointer {CD}

This command gets to current file pointer of the handle given. The first byte of a file is referenced by using {00}{00}{00}{00}.

The syntax is: {CD}{handle}

The data returned is the 4-byte file pointer value {w}{x}{y}{z}
Where:
w = the least significant byte of the 4 byte count
x = is the 2nd significant byte of the 4 byte count
y = is the 3rd significant byte of the 4 byte count
z = is the most significant byte of the 4 byte count

The return status can be {00}, {08}

Get RTC register direct {E2}

This command will read one of the RTC registers directly. The host can use this for applications that don't need the whole date & time string. You can also use this to read the day of the week.

The syntax is: {E2}{dd}
Where {dd} is the register address, as defined here:

```
00=seconds
01=minutes
02=hours (in 24 hour format)
03=day
04=month
05=day of the week
06=year
```

All returned values are a packed decimal byte <00>.

The return status is {00}

Make Directory {C2}

This command will make a new Directory in the current directory.

The syntax is {C2}[directory name]{00}

The return status can be {00}, {02}, {04}, {06}, {09}

Mount Volume (Device reset) {F4}

This command will mount either drive 0 or 1, and initialize the FAT file structure. Users should issue the close all command {C0} before this one to ensure any data is flushed to the device before it is un-mounted.

The syntax is: {F4}{00} or {F4}{01}

The return status can be {00}, {01}, {03}

Open File {C9}

This command will open a file in the current directory. There are three ways to open a file (modes):

```
1 - read only
2 - read/write
3 - read/append
```

If opened for read only and file does not exist, you will get an error.
If opened for read/write, an existing file will be overwritten; a new file will be created.
If opened for read/append, an existing file will be appended to; a new file will be created.
If the file is protected, write or append mode will return an error.

The file pointer is set to the first byte for modes 1 & 2 and set to the next byte after the last byte for mode 3.

The syntax is: {C9}{Mode:handle}[file, directory, or volume name]{00}
Where mode is either 1,2,or 3 and handle is 0-F. Examples:

```
Open "data" for read with handle 1        {C9}{11}[data]{00}
Open "data.txt" for write with handle 9   {C9}{29}[data.txt]{00}
Open "data.asc" for append with handle B  {C9}{3B}[data.asc]{00}
```

The return status can be {00}, {02}, {04}, {05}, {06}, {09}, {0A}, {0E}

Protect file/dir {C6}

This command will mark the selected file or directory protected. It will prevent the Erase File command {C4} from removing the file and prevent overwrites as well. An * will appear to the right of the file/directory name in the directory listing. On DOS machines, this is equivalent to setting the Read-only attribute.

The syntax is: {C6}[file or directory name]{00}

The return status can be {00}, {02}, {04}, {06}, {07}, {09}

Read File {CA}

This command will sequentially read a number of bytes from a file handle. If more bytes are read than are left in the file, the data will have {00}'s appended and a read past EOF error {0F} will be returned. In other words, it will always return the number of bytes requested.

The syntax is: {CA}{handle}{w}{x}{y}{z}
Where:
w = the least significant byte of the 4 byte count
x = is the 2nd significant byte of the 4 byte count
y = is the 3rd significant byte of the 4 byte count
z = is the most significant byte of the 4 byte count

… followed by that many number of bytes being returned. Use the Read Data procedure to retrieve them.
{dd}...

Example:
To read 5 bytes from handle F: {CA}{0A}{05}{00}{00}{00}, you will get {dd}{dd}{dd}{dd}{dd}
To read 1,000,000 bytes from handle A: {CA}{0F}{40}{42}{0F}{00} (1,000,000 = 0x000F4240)

The return status can be {00}, {08}, {0F}, {10}

Read sector (direct mode) {D1}

This command will read a raw sector from the ATA drive. There are two internal buffers that can be used to store the data, 0 or 1.

The syntax is: {D1}{buffer}{w}{x}{y}{z}

It returns 512 bytes {dd}{dd}...{dd}
Where:
Buffer = {00} or {01}
w = the least significant byte of the 4 byte sector
x = is the 2nd significant byte of the 4 byte sector
y = is the 3rd significant byte of the 4 byte sector
z = is the most significant byte of the 4 byte sector
dd = 512 data bytes

The return status will be {00}

Rename File or Dir {C8}

This command will rename a file, directory, or the volume. If the item is protected, it will return an error.

The syntax is: {C8}[file, directory, or volume name]{00}[new name] {00}

The return status can be {00}, {02}, {04}, {06}, {09}, {0A}

Return Directory listing {C1}

This command will return an ASCII dump of the current directory.

The syntax is {C1}

The return data will be variable length with a null {00} character. Each entry displays 40 characters, formatted like this:

```
ABCDEFGHIJK    <DIR>   12/31/08 22:23:59 (directory with DOS name
                                          ABCDEFGH.IJK)
ABCD    IJK *  <DIR>   12/31/08 22:23:59 (dir with DOS name ABCD.IJK,
                                          protected)
A              FFFFFFFF 12/31/08 22:23:59 (file named A)
ABCDEFGH     * FFFFFFFF 12/31/08 22:23:59 (file named ABCDEFGH, protected)
```

The return status can be {00}, {02}, {04}, {06}

Return drive info {F3}

This command will return make, model, and size information about the mounted volume.

The syntax is: {F3}

The returned values will be a variable-length, null terminated string.
[Drive:x yyyyMB Model: zzzzzzzzzzzzzzzzzzzz]{0d}{0a}
[Total Sectors:hhhhhhhh]{0}
Where:
x = Drive number 0 or 1
y = decimal MB of media capacity
z = model name of media
h = total sectors on media in hexadecimal

The return status is {00}

Return free clusters {F1}

This command will return a null-terminated string containing the number of free bytes on the drive.

The syntax is: {F1}

The returned values will be a variable-length, null terminated string.
[length in bytes]{00}

The return status is {00}

Return Volume Name {F0}

This command will return the drive Volume label in a null-terminated string.

The syntax is: {F0}

The returned values will be a variable-length, null terminated string. {dd}...{00}
The format will be:
[Drive:x Volume:xxx…]{00}

The return status is {00}

Set calendar {E1}

This command will set the onboard Real-Time Clock. You send 14 bytes containing date and time info, as described below. Even if the DS1302 is missing, the date & time will be updated. However, it will not advance and any power loss will result in the date & time reverting back to the default.

The syntax is: {E1}
Where:

```
[dw]  is a two character field containing the day of the week.  While
      this is not used by the FAT16 calendar functions, you may want it
      for your host's applications.  If not, it can be set to any value.
      [01]=Sunday
      [02]=Monday
      [03]=Tuesday
      [04]=Wednesday
      [05]=Thursday
      [06]=Friday
      [07]=Saturday
[yy]  2-digit year, [00]-[99]
[mm]  2-digit month, [01]=January and [12]=December
[dd]  2-digit day, [01]-[31]
[HH]  2-digit hour in 24 hour time, [00]-[23]
[MM]  2 digit minutes, [00]-[59]
[SS]  2 digit seconds, [00]-[59]
```

An example would be:
{E1}[03100412145230]
Which will set the calendar to Tuesday, April 12, 2010 at 2:52:30 pm

The return status is {00}

Set file pointer {CC}

This command will move the file pointer for an open file to the position requested. If file is opened read only and the pointer is moved past the end of the file, an error is returned. If moved past the end of file for write or append, the file is enlarged and filled will null bytes {00}.

There are three modes used to set the pointer: (0) SEEK_CURRENT counts from the current location of the file pointer, (1) SEEK_END counts from the end of the file, appending $00 data bytes as needed, and (2) SEEK_SET counts from the first byte of the file. The first byte of a file is referenced by using {00}{00}{00}{00}.

The syntax is: {CC}{handle}{mode}{w}{x}{y}{z}
Where:
mode is either 0, 1, 2
w = the least significant byte of the 4 byte count
x = is the 2nd significant byte of the 4 byte count
y = is the 3rd significant byte of the 4 byte count
z = is the most significant byte of the 4 byte count

Examples are:
Move to beginning of handle 3 {CC}{03}{02}{00}{00}{00}{00}
Move to 100th byte from the beginning of handle D {CC}{0D}{02}{63}{00}{00}{00}
Move forward 10 bytes from the current location of handle 6 {CC}{06}{00}{0A}{00}{00}{00}
Move forward 6 bytes from the end of handle 4 {CC}{04}{01}{06}{00}{00}{00}

The return status can be {00}, {08}, {0F}, {10}, {11}

Test File {CF}

This command will test to see if the file name given is in the current directory. It does not require a handle and does not open the file.

The syntax is: {CF}[file name]{00}

The return status can be {00}, {02}, {04}, {06}, {09}

Unprotect file/dir {C7}

This command will mark the selected file or directory as unprotected. It will allow the Erase File command {C4} to remove the file and allow overwrites as well. No * will appear to the right of the file/directory name in the directory listing. On DOS machines, this is equivalent to clearing the Read-only attribute.

The syntax is: {C7}[file or directory name]{00}

The return status can be {00}, {02}, {04}, {06}, {07}, {09}

Write File {CB}

This command will sequentially write a number of bytes to a file handle. If more bytes are written than are left on the drive, an error {0B} will be returned. You must write the exact number of bytes requested. The firmware will wait for those bytes indefinitely (i.e. freeze).

The syntax is: {CB}{handle}{w}{x}{y}{z} {dd}{..}
Where:
w = the least significant byte of the 4 byte count
x = is the 2nd significant byte of the 4 byte count
y = is the 3rd significant byte of the 4 byte count
z = is the most significant byte of the 4 byte count
dd = first data byte to send
.. = all other data bytes to send

Example:
To write 5 bytes to handle F: {CB}{0F}{05}{00}{00}{00}{dd}{dd}{dd}{dd}{dd}
To write 1,000,000 bytes to handle A: {CB}{0A}{40}{42}{0F}{00}{dd}{.. 999,999 more bytes}

The return status can be {00}, {08}, {0B}, {10}

Write sector (direct mode) {D2}

This command will write 512 bytes to a sector on the ATA drive.

The syntax is: {D2}{buffer}{w}{x}{y}{z} {dd}{dd}...{dd}
Where:
Buffer = {00} or {01}
w = the least significant byte of the 4 byte sector
x = is the 2nd significant byte of the 4 byte sector
y = is the 3rd significant byte of the 4 byte sector
z = is the most significant byte of the 4 byte sector
dd = is 512 data bytes

The return status can be {00}, {02}, {04}, {06}, {09}

Parts List

Part #	Description	Digikey.com Part#
C1	.1uF Ceramic Cap	BC1160CT-ND
IC1	ATMEga324P (40-pin DIP)	ATMEGA324P-20PU-ND
J1	SPI port (2x5 header)	* Note 1
J2	IDE port (2x20 header)	* Note 1
J4	Jumper	* Note 2
J5	Jumper	* Note 2
OSC-1	20.0 MHz TTL Oscillator (1/2 can)	X964-ND
	(optional components for RTC)	
B1	3V CR2032	P189-ND
IC2	DS1302 (8-pin DIP)	DS1302+-ND
X1	32.768khz watch crystal	300-8301-ND
Battery Holder		BH32T-C-ND
	(optional components for on-board regulated +5V)	
C2	220uF Electrolytic Cap	493-1492-ND
C3	220uF Electrolytic Cap	493-1492-ND
J7	Host +9 to +12VDC input	* Note 2
IC3	LM7805	497-1443-5-ND
	(optional components for disk activity LED)	
LED1	Drive Activity LED with internal current limiting resistor	516-1339-ND

Note 1 - order one 1SAM1028-36-ND for J1 and J2
break off appropriate number if pins for each
Note 2 - order one TSW-106-07-L-S-ND for J4,J5, and J7
break into three 1x2 headers.
Note 3 – J3 & J6 - solder wire to the pads from external connectors.