USB-FileSys™

Driver-free USB embedded file system (Previously USB-Config)

Summary

USB-FileSys is a low cost USB FAT file system. Its memory area is accessed by a host controller using SPI commands, and, when plugged into a USB port, simultaneously appears on a PC like a file system on a removable disk.

For low duty configuration and diagnostic applications, USB-FileSys can use its own on-chip pre-formatted 8kByte flash memory area. For moderate file storage and transfer, USB-FileSys can be interfaced to Microchip 25AA1024 EEPROM for 128kByte storage. For large file storage and transfer applications, USB-FileSys can be interfaced to SD™ cards, and integrated circuits with SD™ card interfaces, of up to 1GBbyte. (Internal and SD options for 28-pin devices only.)

USB-FileSys uses the Mass Storage Device (MSD) USB profile. It works without drivers on PCs running Windows (ME or later), Mac (OS 9 or later) or Linux (4.0 or later). It is available pre-programmed as a 28-pin DIL package and a 20 pin SSOP package.

Features

- FAT12 / FAT16 file system, no licensing needed
- True MSD plug and play, no drivers required
- Works with internal flash memory, 25AA1024 external memory (128Kbyte) and SD compatible cards and integrated circuits up to 1GB.
- Partial support for long file names
- USB 2.0 compatible

Applications

- Consumer products, e.g. photo frames
- Data logging
- File transfer
- Device configuration & diagnostics

Mechanical Specifications

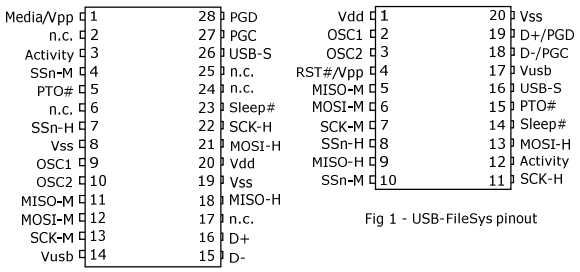


Fig 1 - USB-FileSys pinout

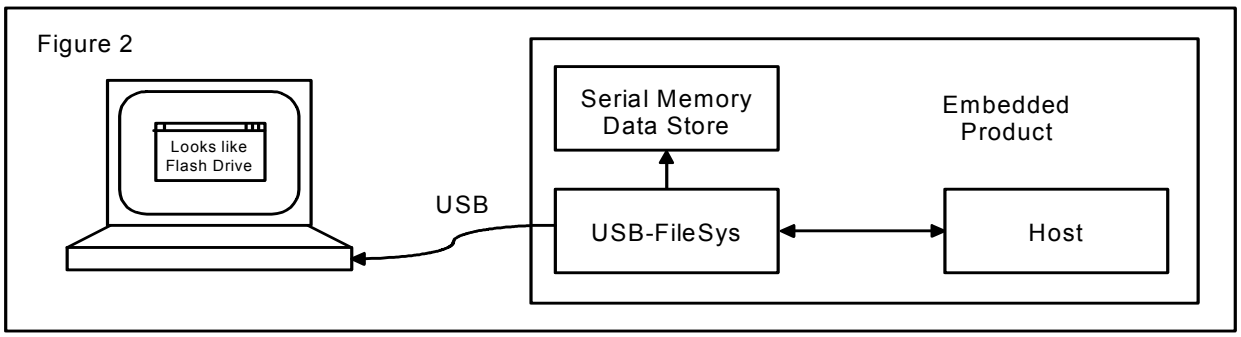
Device pinout

Table 1. Device Pinout

Pin		*Name*	*Description*
DIL	***SS***		
20	*1*	*Vdd*	Power positive input
9	*2*	*OSC1*	Oscillator input
10	*3*	*OSC2*	Oscillator output
1		*Media* *Vpp*	Media detect TEAclipper Vpp
	4	*RST#* *Vpp*	Device reset (Active low) TEAclipper Vpp
11	*5*	*MISO-M*	Data input from SPI memory
12	*6*	*MOSI-M*	Data output to SPI memory
13	*7*	*SCK-M*	Clock output to SPI memory
7	*8*	*SSn-H*	Slave select input from host
18	*9*	*MISO-H*	Data output to host
4	*10*	*SSn-M*	Slave select output to SPI memory
22	*11*	*SCK-H*	Clock input from host
3	*12*	*Activity*	USB Activity Indicator
21	*13*	*MOSI-H*	Data input from host
24	*14*	*Sleep#*	Sleep control input (Active low)
5	*15*	*PTO#*	Power take-off OK indicator (active low)
26	*16*	*USB-S*	USB voltage sense
14	*17*	*Vusb*	USB supply filter
15	*18*	*D-*	USB data -
27		*PGC*	TEAclipper PGC
16	*19*	*D+*	USB data+
28		*PGD*	TEAclipper PGD
8,19	*20*	*Vss*	Power ground reference

Firmware Factory Ltd
3 Plough Yard, Ground Floor
London EC2A 3LP, UK
sales@firmwarefactory.com
support@firmwarefactory.com

Figure 2



Electrical Specifications

Table 2. Electrical Specifications	
Operating voltage Vdd, 28-pin device	2.7V – 5.5V
Operating voltage Vdd, 20-pin device	1.8V – 5.5V
Typical/max supply current, Vdd = 5.0	10mA / 21mA
Typical/max Sleep current, Vdd = 5.0	0.1µA / 2µA
Operating Temperature	-40°C to +85°C
Maximum SPI clock rate	1MHz

Basic Operation

To the host microcontroller ('host'), USB-FileSys looks like a file system accessed with SPI commands. To the PC, it looks like a flash drive.

The use of the MSD USB profile means that no driver installation is required. It is plug-and-play compatible with PCs running Windows (ME or later), Mac (OS 9 or later) or Linux (4.0 or later).

Pin Functions

The pin functions are shown in table 1. Pins marked n.c. will be configured as inputs and, in order to minimize power consumption, should not be left floating.

The pin functions are described in detail below.

Vss, Vdd, Vusb

Vss is the power supply ground reference. *Vdd* should be connected to a regulated supply, most usually the USB power when available and a battery when USB power is not available. *Vusb* should be connected, via a 470nF capacitor, to *Vss*. See for example C8 in figure 4.

OSC1, OSC2

OSC1 and OSC2 should be connected to a 12MHz parallel cut crystal circuit with 22pF capacitors or a 12MHz resonator with 0.25% total tolerance.

Vpp, PGC, PCD

TEAclipper programming pins. Refer to the TEAclipper Programming section for details. Note that the Vpp pin may be subject to voltages as high as 13V during programming. On the 28-pin device, PGC and PGD will be configured as outputs in normal use and should be left floating when not being used for programming.

Media Detect

The Media detect input pin should be pulled low with a 33K resistor if 25AA1024 memory used. If the SCK-M pin is biased low indicating that internal memory is used, the Media detect pin is ignored.

The Media detect input pin should be high during normal operation with SD memory. If SD memory removable, Media detect should be taken low when it is removed. If the device is connected to a PC, a soft detach and re-attach will be performed on media removal. The SD memory must be physically removed so that a power-up reset is performed when it is re-inserted.

In order to allow USB-FileSys to be programmed in-circuit, it must be possible to configure the application circuit so that this input appears to be pulled high via a 22k resistor, or pulled low with a 33k resistor.

RST#

The 20-pin device has an active low pin in place of the media detect pin. This should normally be biased high with a 22k resistor.

D+, D-

USB data I/O. Refer to the USB Connectors section for details of their connections.

USB Voltage Sense

This input should be high when the device is plugged into a USB host. Refer to figure 5 for a typical method of deriving this input from the USB supply input.

USB Activity Indication

Output that can either be used for connecting to an activity indication LED, or for generating an interrupt when an USB write event occurs.

If configured as an activity indicator, it is high for approximately 100ms when USB status is changing, or a USB command is being processed other than Test Unit Ready. (Test Unit Ready is ignored because operating systems send it approximately once per second to verify the drive is still available.)

If configured as an interrupt, it can be set to output high when the PC is executing a write to any sector, or a write to the root directory entries. The primary purpose of these functions is to allow the host to capture the date and time information when the PC closes a file.

PTO#

The "Power Take-Off" output is low when power may be drawn by the application circuit from the USB supply. USB-FileSys device is configured to request up to 100mA, allowing the device to operate from unpowered hubs. It may be reconfigured to request up to 500mA using the Set Configuration command. If this is done the device will work from PCs and powered hubs only.

If PTO# is high, the device should draw no more than 100µA. In no event should an external power source inject power into the USB Vdd line.

Sleep

The Sleep input is be regularly polled. If it is low, the device and its slave memory enter a low-power sleep state. When sleep goes high, the device will resume normal operation within 70ms.

SCK-M, MISO-M, MOSI-M, SSn-M

If SD or 25AA1024 memory is used, these pins are the SPI serial interface to the memory. 4k7 pull-up resistors are required on all these lines.

If internal memory is used, SCK-M should be biased low.

SCK-H, MISO-H, MOSI-H, SSn-H

In SPI communications with the embedded host, USB-FileSys acts as an SPI slave device. USB-FileSys is selected by transitioning *SSn* from high to low. At that time, *SCLK* should be low (*CPOL=0*). On the rising edge of *SCLK*, one bit of data is read on *MOSI* (*CPHA=0*). On the falling edge of *SCLK*, the data on

MOSI may be changed by the master. (This is also known as *Mode D* or *0,0* operation.) See figure 3. The actual commands are specified in the Host Commands section.

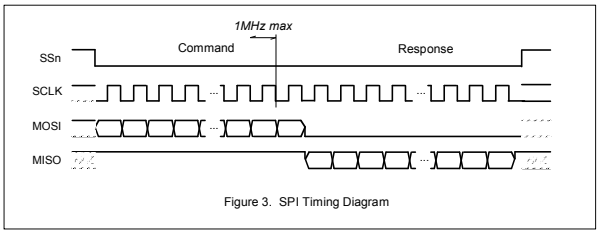


Figure 3. SPI Timing Diagram

All commands from the host are followed by a response from USB-FileSys. Since the time generated to generate a response is indeterminate, USB-FileSys will output bytes of value 0xFF until it can generate a valid response. After issuing a command, the host should repeatedly read data from USB-FileSys, waiting for a byte with a value other than 0xFF. The Write File and Set File Entry commands are each composed of two such Command-Response exchanges.

USB-FileSys has been tested with SPI clock rates up to 1MHz.

Memory

Internal Memory

In order to maximize available storage space, the internal memory is pre-formatted as a 16-sector FAT12 file system with 512 bytes per sector. (4 of these sectors are required for the file system and root directories.) Reformatting by the PC is not permitted.

The root directory can contain up to 16 entries, including the volume name. Since long file names require multiple directory entries, they are not recommended. The volume is normally supplied containing no files. If ordered in 5K+ quantities, files may be pre-loaded.

To indicate to USB-FileSys that the internal memory should be used, the SCK-M input should be pulled low.

Pages of internal memory may be erased and rewritten approximately 100K times. If file sizes change frequently, the greatest wear will occur in the FAT tables and the directory entry. Therefore if a file is to be written to frequently, its size should be fixed.

This option is only available for the 28-pin device. TQFP and SSOP packages are available, but they are not in general distribution. Contact us for further information.

External Memory – 25AA1024

For moderate storage applications, a Microchip 25AA1024 memory (128kByte storage) may be connected to the MISO-M, MOSI-M, SCK-M and SSn-M lines. A 4k7 pull-up resistor should be provided for SCK-M. The media detect input should be permanently biased low.

When first used, the memory will automatically be formatted as a 16-sector FAT12 file system with 512 bytes per sector. The PC may re-format the drive if required.

External Memory – SD card / integrated circuit

For large storage applications, an SD-compatible memory of up to 1GB may be connected to the MISO-M, MOSI-M, SCK-M and SSn-M lines. 4k7 pull-up resistors should be provided for these lines, and also for unused SD card I/O pins.

The memory can be a removable SD card or a SD-compatible chip. If the memory is removable, the media detect input should be connected to a switch to indicate when the card is present. Devices with SD card readers will be able to read the data on the SD card directly.

Memories larger than 1GB will be ignored, as will non-standard memories which do not have block sizes of 512 bytes. San-Disk products are used for testing.

Pages of SD memory are typically wear-leveled, so memory wear is not a great concern – memory wear can be compensated for by using a larger memory. For example, for a given amount of data stored, a 1GB SD card will last 10 times longer than a 100MB card.

This option is only available for the 28-pin device. TQFP and SSOP packages are available, but they are not in general distribution. Contact us for further information.

Memory Integrity

With internal memory and 25AA1024 memory, power loss during a write may result in corruption of data, possibly requiring reformatting of the disk.

With external memory, power loss protection may or may not be implemented by the memory itself.

If power is lost while a file is open, the file length information in the directory entry may be incorrect, but the data will be intact.

Application Circuits

The following circuits show a typical implementation of the USB-FileSys. Suggested component values are shown in table 3.

Table 3. Suggested component values	
Label	***Component***
R1-R3, R40-R44	*22k resistor*
R6	*1k resistor*
R21	*470Ω resistor*
T1	*P-channel Mosfet, e.g. NDS352P*
D1-D2	*Low Vf switching diode*
LED1	*Light emitting diode*
C1	*1µF capacitor*
C2, C3	*22pF capacitor*
C4, C6-C7	*100nF capacitor*
C8	*470nF capacitor*
X1	*12MHz parallel cut crystal*

In figure 4, the internal memory is used. *Vto* is gated by P-channel mosfet T1 to provide power take-off circuit, for example to provide power to a battery charger.

Oscillator X1/C2/C3 may be replaced by a low-cost resonator, provided its frequency tolerance is greater than 0.25%. C1 and C6 should be placed close to the USB connector. C7 should be placed near the Vss and Vdd pins of the USB-FileSys and is required only if it would be some distance from C6. C8 is a filter capacitor for an internal regulator and is required.

The TEAclipper connector is for in-circuit programming of devices where the firmware has been purchased from HexWax. It is recommended to allow firmware updates, even if the firmware is initially supplied pre-programmed.

LED1 is a USB activity indicator. The Sleep pin must be biased high if unused.

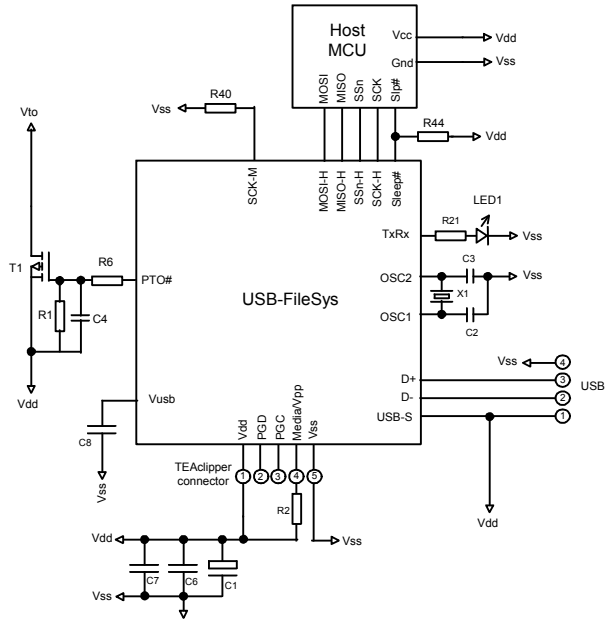


Figure 4. Typical Application Circuit, Internal Memory

In figure 5, external memory is used. SW1 acts as a media detect switch for an SD card. If 25AA1024 memory is used, the Media input should instead be biased low; if non-removable SD memory is used, the Media input should instead be biased high.

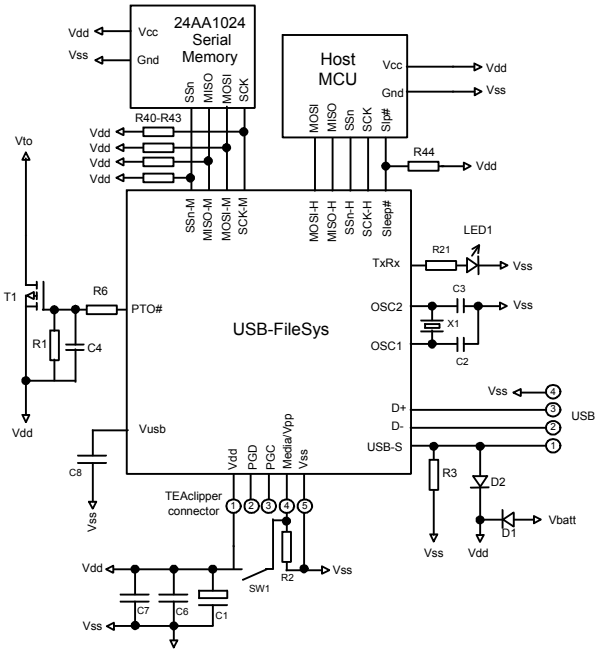


Figure 5. Typical Application Circuit, External Memory
D1 and D2 should have a very low forward voltage – ideally 0.3V.

In this implementation, two power sources are available: power from the USB power input, and a battery *Vbatt*. D1 and D2 ensure the higher of these is used to provide the voltage Vdd. R3 ensures the USB-S power sense signal is pulled low when the USB power is not present. The USB voltage must be regulated to 3.3V if the serial memory and/or host controller could not operate at 5V.

USB Connectors

Common USB connector and cable configurations are shown in figure 6 and table 4. The shield on the connector should be left unconnected. The ID pin on the mini connector permits the distinction of A and B plugs. The micro connector pin-out is the same as the mini connector.

Figure 6 Common USB pin-outs for male connectors

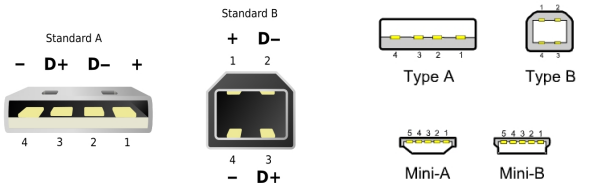


Table 4. USB Connection Key				
Pin		**Name**	**Cable color**	**Description**
Std	**Mini**			
1	1	Vcc	Red	+5V (can dip to 4.08V)
2	2	D–	White	Data –
3	3	D+	Green	Data +
–	4	ID	–	Type A: Connect to ground Type B: Not connected
4	5	Gnd	Black	Signal ground

For ultra-low cost products, it is possible to form a USB Type-A plug direct from a circuit board as shown in figure 6. This connector is only suitable for a limited number of insertions (~50 before cleaning is required).

It is unshielded and recommended only for 'dongle' type products with no cables attached.

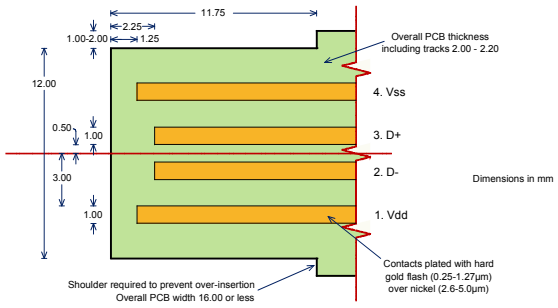


Figure 7. Integral USB connector dimensions

For further dimensional information, refer to figure 6-7 of the USB 2.0 Specification, in the development kit.

SD Card Connector

The SPI connections for SD Cards is shown in figure 8. The two pins marked BH are unused and must be biased high.

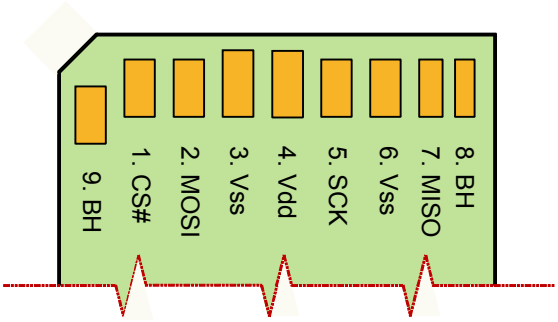


Figure 8. SD Card Connections.
Pin 9 is out of order; this is not an error.

Host Commands

All commands start with a command byte and are followed by a command-specific number of data bytes. The response from USB-FileSys will start with a status byte followed by a command-specific number of data bytes. The response is then clocked out and must be completely received before sending the next command. For multi-byte values, the byte order is little-endian, i.e. least significant byte first. *While the response is being clocked out, the value being clocked in must be 0xFF.*

The possible status byte codes are shown in table 5. Generally, USB-FileSys will respond with NOT_READY while it computes a response, and then with the status code resulting from the command. The host will have to poll USB-FileSys repeatedly to determine the result. If the result is anything other than SUCCESS, no further data bytes will follow.

Table 5. Command Status Codes		
Name	***Value***	***Meaning***
SUCCESS	*0x00*	Command successfully executed
NOT_POSSIBLE	*0x01*	Not possible to execute the command as requested
FILE_NOT_FOUND	*0x02*	File not found
VOLUME_FULL	*0x03*	No empty clusters available
ENTRY_INVALID	*0x05*	File reference is not valid
HANDLE_IN_USE	*0x06*	File handle is in use
ROOT_DIR_FULL	*0x07*	Root directory is full. (The root directory has a fixed size.)
OUT_OF_RANGE	*0x08*	Attempt to read after the last cluster
FAT_TABLE_ERROR	*0x09*	FAT table error
WRITE_FAILURE	*0x0A*	Internal memory write failure
NOT_FORMATTED	*0x0B*	Memory not formatted
NOT_MEMORY	*0x0C*	Removable SD card not present
NOT_READY	*0xFF*	Still computing result – try again later

In order to maximize the size of the available internal memory, error checking is not exhaustively performed to verify that the command sent to the host is legal. It is the host's responsibility to ensure commands are correct, e.g. only deleting empty subdirectories and using legal characters in file names.

A "current directory" is used to avoid long path names. To open a file, it is necessary to move to that file's current directory first, and then open it.

When a file is opened, the host assigns a file handle number (00-03), which is then used to refer to the file in subsequent read and write commands. When the file is closed, the file handle is freed to be used in another open command.

Long file names are only partially supported. Full support would significantly increase chip costs due to patent licensing requirements. The PC may create files and directories with long names, but they will only be visible to USB-FileSys by their short file names. USB-FileSys itself can only create files with short names, but it can work with long-name files created by the PC by referencing their short file name equivalents.

Short ("8.3") file names are composed of a name of up to eight ASCII characters followed by an extension of up to three characters. If either the name or t he extension are shorter than this, they should be padded with spaces so that the file names is 11 characters long. The following characters are not allowed:

1. Lower case letters **a** to **z**
2. Any of **\ / : * ? " < > | + , . ; = [ ]**
3. Control characters 0x00 – 0x1F
4. DEL character 0x7F
5. The first character must not be a space

Executing Commands While PC Connected

Executing commands while the PC is connected is problematic, but possible.

USB-FileSys will ensure that the PC and the host do not simultaneously modify the volume. Nevertheless, care should be taken if the host modifies any files while the device is connected to a PC.

In general the host and the PC should not attempt modify the same file. It is recommended that the host only modifies files whose directory entries have been marked as read-only for the entire time that the PC has been connected.

Operating systems may cache data locally, and so modifications made by the circuit may not necessarily immediately apparent on the PC.

Sample Host Source Code

Sample C source code for host controllers is supplied in the development kit. It was developed for the PIC18F2320, but should be readily portable to other microcontrollers. 28-pin PIC18F devices will fit in the host controller socket in the evaluation board.

Root Dir

This command sets the current directory to the root directory. The first byte of the command is the value 0x02. There are no further bytes. The response is the status byte.

Example:

Command: Go to root directory.

02

Response: Success.

00

Get File Entry By Index

This command retrieves information about a directory entry in the current directory. This could be a file, a subdirectory, or a volume label. The first byte of the command is the value 0x03. The second byte and third bytes are the index number of the subdirectory or file in the current directory being requested (least significant byte first, zero-based).

The response will be the status byte and, if the status byte is zero, the standard FAT directory entry data bytes as shown in table 6. See the Set File Entry command for an example.

Table 6. FAT directory entry

Byte	***Meaning***
0-7	File name, padded with spaces
8-10	File extension, padded with spaces
11	Bit 0: Set if item is marked as read-only Bit 1: Set if item is marked as hidden Bit 2: Set if item is marked as a system file Bit 3: Set if item is a volume label Bit 4: Set if item is a subdirectory Bit 5: Set if item is marked for archiving If bits 0-3 all set, is an ignorable VFAT entry
12	Reserved
13-17	†Creation date / time: Bits 0-7: Centisecond (1-199) Bits 8-12: Doublesecond (0-29) Bits 13-18: Minute (0-59) Bits 19-23: Hour (0-23) Bits 24-28: Date (1-31) Bits 29-32: Month (1-12) Bits 33-39: Year since 1980 (0-127 = 1980-2107)
18-19	‡Date of last access: Bits 0-4: Date (1-31) Bits 5-8: Month (1-12) Bits 9-15: Year since 1980 (0-127 = 1980-2107)
20-21	Reserved
22-25	Last modified date / time: Bits 0-4: Doublesecond (0-29) Bits 5-10: Minute (0-59) Bits 11-15: Hour (0-23) Bits 16-20: Date (1-31) Bits 21-24: Month (1-12) Bits 25-31: Year since 1980 (0-127 = 1980-2107)
26-27	Reserved
28-31	File length in bytes, LSB first

† Supported by USB-FileSys except centiseconds

‡ Supported by USB-FileSys in certain cases, see Close command

Get File Entry By Short Name

Same as Get File Entry command, except searches for a file that matches the name and attributes supplied. The first byte of the command is the value 0x04. The next 8 bytes are the file name, padded with spaces. The next 3 bytes are the file extension, padded with spaces. Last is the attribute byte, usually 0x00 for a file, 0x10 for a subdirectory, or 0x08 for the volume label, as specified by byte 11 of table 6. (Only the volume label and subdirectory bits are used when searching for a file. Other bytes are only used if a non-existent file is then created using the Open command.)

For examples, see the Change Dir and Open commands.

Change Dir

This command sets the current directory to the file returned by the immediately preceding Get File Entry By Index or Get File Entry By Short Name command. This last file must be a subdirectory; it cannot be the ".." pointer to the parent directory. The command consists of the byte 0x05 only. The response is the status byte.

Example:

Command: Find subdirectory "DATA".

04 44 41 54 41 20 20 20 20 20 20 20 10

Response: Success.

00 44 41 54 41 20 20 20 20 20 20 20 10…

Command: Move to the subdirectory.

05

Response: Success.

00

Set File Entry

This command sets directory entry of the file returned by the immediately preceding Get File Entry By Index or Get File Entry By Short Name command. The one-byte the command is the value 0x06. A response byte will be received. Assuming this the Success response 0x00, the 32 file entry data bytes are then sent, as specified in table 6. (The bytes market reserved must be left unchanged.) A second response byte will then be received.

Example:

Command: What is the 5th directory entry in the current directory?

03 04 00

Response: Success, file is a read-only file called "FRED.TXT".

00 46 52 45 44 20 20 20 20 54 58 54 01…

Command: Set file entry

06

Response: Success.

00

Data: (Renamed "TOM.TXT")

00 54 4F 4D 20 20 20 20 20 54 58 54 01…

Response: Success.

00

Example:

Delete File / Directory

This command deletes the file, subdirectory or volume label retrieved by the last Get File Entry By Index or Get File Entry By Short Name command. The first byte of the command is the value 0x07 followed by the confirmation bytes 0x59, 0x65, 0x73 ("yes").

The response will be the status byte.

Example:

Command: What is the 5th directory entry in the current directory?

03 04 00

Response: Success, file is a read-only file called "FRED.TXT"

00 46 52 45 44 20 20 20 20 54 58 54 01…

Command: Delete it

07 79 65 73

Response: Success.

00

Only empty subdirectories should be deleted; it the host's responsibility to delete all directory entries first. (The "." and ".."entries do not need to be deleted.)

Format Volume

This command deletes the entire contents of the internal volume or the 25AA1024 external memory. It frees up any clusters which may have been rendered unusable due to deletion of a non-empty directory or power failure during a file operation. The root directory will become the current directory.

The first byte of the command is the value 0x08 followed by the confirmation bytes 0x59, 0x45, 0x53 ("YES"). The response will be the status byte.

SD memories must be formatted by the PC. 24AA1024 memories can be formatted by the PC when used with the 28-pin USB-FileSys.

20-pin devices should only format using the Format Volume command.

Example:

Command: Format volume

08 59 45 53

Response: Success.

00

Set Date / Time

This command sets the date and time as used by file creation and file close commands. It does not keep time. This command should be given immediately before the open command when a file is being created and all close commands. If the command is never given, files are given the date / time 12:00 1/1/2000 when created and left unchanged otherwise.

The first byte of the command is the value 0x09. The following four bytes indicate the date and time as shown in table 7.

Example:

Command: Set date / time to 12:00, 1/1/2000

09 00 60 21 28

Response: Success.

00

Table 7. Date / time structure

Byte	***Meaning***
0-1	Time: Bits 0-4: Doublesecond (0-29) Bits 5-10: Minute (0-59) Bits 11-15: Hour (0-23)
2-3	Date: Bits 0-4: Date (1-31) Bits 5-8: Month (1-12) Bits 9-15: Year since 1980 (0-127 = 1980-2107)

Open File

This command opens an existing file, or creates it if it does not exist already, using information from the previous Get File Entry By Index or Get File Entry By Short Name command. The first byte of the command is the value 0x0A. The second is the "file handle" to associate with the file, which can be a value from 00 to 03. The response will be the status byte. The file must be closed before the file handle is re-used.

The Open command may be used for creating a subdirectory or volume label. In this case, the directory entry is created but no file is opened and the file handle is ignored. In the case of a subdirectory, the current directory is changed to the new subdirectory.

It is permitted to open a file more than once simultaneously, for example to copy data from one section of the file to another, provided only one file handle is used for writing to the file.

Files can only be created after using the Get File Entry By Short Name command.

Example 1 (open existing file):

Command: What is the 5th directory entry in the current directory?

03 04 00

Response: Success, file is a read-only file called "FRED.TXT"

00 46 52 45 44 20 20 20 20 54 58 54 01…

Command: Open the file using handle 01

0A 01

Response: Success, file is open on handle 01

00

Example 2 (create file):

Command: Find file "FRED.TXT" (note the read-only attribute is set, although this is ignored while searching for the file)

04 44 41 54 41 20 20 20 20 20 20 20 01

Response: Fail, file does not exist.

02

Command: Create the file as read-only using handle 03

0A 03

Response: Success, file is open on handle 01

00

Example 3 (create subdirectory):

Command: Find subdirectory "DATA".

04 44 41 54 41 20 20 20 20 20 20 20 10

Response: Fail, file does not exist.

02

Command: Create subdirectory (file handle ignored)

0A 00

Response: Success. (No file is opened)

00

Close File

This command closes the file associated with a file handle. The first byte of the command is the value 0x0B. The second is the "file handle" of the open file. The next four bytes (LSB first) are the desired length of the file, or 0xFFFFFFFF if the length is to be the 'expected length'. The expected length is greater of the length recorded in the directory entry and the highest position written to while open.

The main effect of this command is to update the length argument in the directory entry and, if it was shortened, to free any unused clusters at the end of the file.

If external memory is used, the creation, modify and access date/time are all fully updated as required. If internal memory is used, these fields will only be updated by USB-FileSys if the directory entry would have been modified anyway. (i.e. When the file is created and when it is closed with a different length than when it was opened.)

Example 1:

Command: Close file handle 02 with expected length

0B 02 FF FF FF FF

Response: Success

00

Read File

This command reads data from a file starting from the file pointer position. The first byte of the command is the value 0x0D. The second is the "file handle" of the open file. The next four bytes are the position in the file to read from, or 0xFFFFFFFF to continue on sequentially from the last read or write command. (If there have been no previous read or write command, the start of the file will be assumed.)

With the 28-pin device, the next two bytes are the number of bytes to read, up to a maximum of 512. The read must not cross the 512-byte sector boundary.

With the 20-pin device, the next two bytes are the number of bytes to read, up to a maximum of 64. The read must not cross the 64-byte page boundary.

Note: The file length information in the directory is ignored. The data may be read up to the last cluster itemized in the file allocation table.

The response will be the status byte followed by the data bytes read.

Example:

Command: Read 4 bytes at position 0123 from file handle 02

0D 02 23 01 00 00 04 00

Response: Success, data is 12 34 56 78.

00 12 34 56 78

Write File

This command writes data to a file starting from the file pointer position. The first byte of the command is the value 0x0E. The second is the "file handle" of the open file. The next four bytes are the position in the file to write to, 0xFFFFFFFE to append onto the end of the file, or 0xFFFFFFFF to continue on sequentially from the last read or write command. (If there have been no previous read or write commands, the start of the file will be assumed.)

With the 28-pin device, the next two bytes are the number of bytes to write, up to a maximum of 512. The write must not cross the 512-byte sector boundary.

With the 20-pin device, the next two bytes are the number of bytes to write, up to a maximum of 64. The write must not cross the 64-byte sector boundary.

Once the command has been sent, a response byte will be received. Assuming this the Success response 0x00, the data bytes are then sent. A second response byte will then be received.

Note: Writes to files are permitted even if the read-only bit is set in their directory entry; writes after the end of the existing file may cause further clusters to be allocated to the file.

Example:

Command: *Write 4 bytes at position 0123 on file handle 02*

0E 02 23 01 00 00 04 00

Response: *Success, data is 12 34 56 78.*

00

Data: *(01 02 03 04)*

01 02 03 04

Response: *Success.*

00

Get Info

This command reads device information. The command is the byte 0x0F. The first byte of the response is the status byte. The second bytes indicates the information in table 8. The third and fourth bytes are the version number, least significant byte first.

Example:

Command: *Get Info*

0F

Response: *Success, device is 500mA with a 4096-byte cluster size, version 0001*

00 61 01 00

Table 8. Info Flags

Bit	***Meaning***
0	Maximum current draw: 0: Draw a maximum of 100mA from USB (default) 1: Draw a maximum of 500mA from USB
2-1	Activity pin function: 00: High for 100ms on any USB activity 10: High on PC write to any sector 11: High on PC write to root directory entry
5-7	Cluster size: 000 - 512 bytes 001 - 1024 bytes 010 - 2048 bytes 011 - 4096 bytes 100 - 8192 bytes 101 - 16384 bytes 110 - 32768 bytes (Read only – ignored with Set Info command)

Set Info

This command sets the maximum current the device may draw from the PC, and the function of the *Activity* pin. The first byte of the command is the value 0x10. The second is as specified in table 8. The response is the status byte.

If USB-FileSys is configured for 500mA current draw, it will only work when plugged directly into PCs and powered hubs. The setting takes effect from the next plug-in. In any event, the no more than 100µA should be drawn if the PTO# pin is high.

Example:

Command: *Set 500mA operation, activity on write*

10 05

Response: *Success.*

00

Delivery and Programming

USB-FileSys is generally distributed in pre-programmed in 28-pin DIL and 20-pin SSOP packages. Contact us for other package options.

In high volumes (5K+), USB-FileSys is available reeled with your custom settings preloaded, in any available package.

TEAclipper Programming

For programming the firmware using TEAclipper/PIC HV, the *PGC*, *PGD* and *Vpp* pins must be accessible. During programming, these pins must be protected against contention. In particular, note that *Vpp* is subject to up to 13V during programming. Nothing else should be connected to *Vpp* except via a 22k pull-up resistor.

The TEAclipper connector format is shown in figure 9. Since the programming time is very fast, no programming socket is required. The TEAclipper can be 'leaned' against the plate-through holes shown.

It is strongly recommended that this connector is included in circuits even if in-circuit programming is not anticipated, since this allows you to upgrade the firmware if necessary.

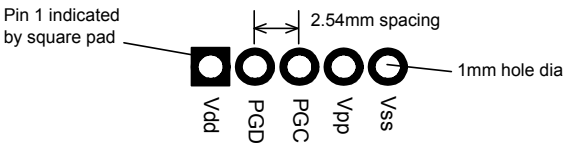


Figure 9. Recommended plate-through connector design

Evaluation Board

The USB-Eval evaluation board is available for evaluating USB-FileSys. Due to the variety of potential implementations it is supplied unpopulated and a prototyping area is provided. This will be necessary for SD memory implementation. (e.g. Figure 10.)

It is recommended that new users initially try to replicate figure 4 and then modify it to test the circuit they actually require. Note the following common omissions:

1. The Sleep pin must be biased high if unused.
2. SD memory will need 3.3V power regulation.

You may be interested in using disk sector analysis software such as Disk Investigator, which can be downloaded free from *www.theabsolute.net.*

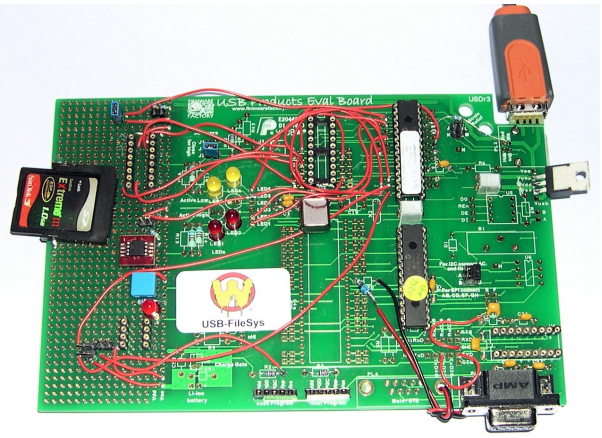


Figure 10. Evaluation board example

Development Kit

A firmware development kit is available for download from *www.hexwax.com* containing the following files:

- This data sheet.
- Sample host controller source code.
- Base controller data sheets (© Microchip Technology Inc)
- *USB 2.0 Specification* (© HP / Intel / Lucent / Microsoft / NEC / Philips 2000)
- *The FAT File System.* (Wikipedia article.)

Revision history

The following limitations exist with rev 0001 DIL parts. (All SS parts are rev 0002 or higher.)

- Files should not be truncated to length zero.
- If when a file is created, it should be immediately closed and reopened prior to writing or reading.
- Info flags reported by Get Info command are meaningless.
- Read command returns the correct only if a write command was sent more recently than a close command.
- Cannot add cluster to a FAT16 file from the SPI host. There is no easy workaround, so contact us for an upgrade if you need this capability.
- 25AA1024 memory must be formatted from the PC.

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for Hexwax ***manufacturer:***

Other Similar products are found below :

Hexwax