µALFAT™ User Manual
Revision 2.00

GHI Electronics, LLC

** Preliminary document **

Updated – July 19, 2006

Table of Contents

1. Introduction

1.1. FAT file system

Memory cards are very popular these days. Their small size and rather huge storage makes them ideal for digital cameras and MP3 players. Managing and organizing data on a memory card is not an easy task. It requires a robust file system. There are many file systems out there but FAT, which stands for File Allocation Table, is the most popular one. FAT was introduced by Microsoft with DOS operating system and FAT is still supported by all versions of Windows operating systems, even Windows XP. To be compatible with PCs, many other companies developed their own FAT code for different applications, such as digital cameras. There are three versions of FAT file system, FAT12, FAT16 and FAT32.

1.2. Adding a file system to your product

Adding a file system, such as FAT, to a product can cost a company thousands of dollars. A reliable FAT stack source code costs between $4,000 and $10,000. That is not all, FAT is useless without driver for the connected media, an SD card for example. Understanding the media, porting, compiling and linking are also other complicated steps. Further more, adding USB memory support adds complication to the process. Our own research shows that adding a file system to a product would take about 6 months or cost about $10,000!!!

1.3. µALFAT™ general description

µALFAT is a full FAT file system on a chip. µALFAT requires very few external components to run. For communication, a simple microcontroller (PIC, AVR, basic stamp…etc.) with UART, SPI or I2C can be used. µALFAT can access MMC and SD memory cards directly and USB Mass Storage Devices through a USB Host Controller (MAX3421E)

1.4. Key Features

- FAT16 and FAT32 support.
- Runs on UART, SPI or I2C.
- Programmable UART (serial port) bud-rate.
- Can open 4 files at the same time using file handles.
- Fast startup and media reconnect, few milliseconds.
- Fast file write and read, average to 60 Kbytes/Sec or more, including interface overhead.
- Supports Secure Digital (SD) and Multi Media Card (MMC)
- Supports USB Mass Storage Devices like thumb flashes and card readers
- No SD license is required.
- Field upgradeable firmware through UART, SPI, or I2C.
- Field upgradeable firmware through a file on the connected media!

- Very few external components.
- 10-bit ADC.
- RTC (Real Clock Time) capable of running of external battery.
- Low power consumption, 12mA.
- Three power modes.
- All I/O pins are 5 volt tolerant.
- Small surface mount package, LQFP 48 pin.
- -40˚C to +85˚C temperature operating range.
- Lead free.

1.5. Some Example Applications

- Digital cameras
- Printers
- Digital picture viewer
- MP3 players
- Data logger
- Automated machines

2. µALFAT™ Pin-Out and Description

Pin	Name	Description
1	SD_MISO	SD card signal
2	SD_MOSI	SD card signal
4	VBAT	Power source for the internal RTC. Connect to 3V battery or VCC. Always use 2 diodes to connect a battery and VCC in case the battery runs out of power.
5	VDD 1.8V	1.8V power source
6	RESET#	µALFAT reset signal, active low.
7	VSS	Ground
8	TRST#	Do not connect
9	TMS	Do not connect
10	TCK	Do not connect
11	X1	Pin 1 for 10 MHz oscillator
12	X2	Pin 2 for 10 MHz oscillator
13	UART_TX SPI_DATARDY I2C_DATARDY	UART mode: Transmit pin of UART (output) SPI and I2C modes: When high, it signals that µALFAT wants to send some data and the host must read it
14	UART_RX SPI_BUSY I2C_BUSY	Receive pin of UART (input) SPI and I2C: When high, it signals that µALFAT is being busy and not ready for new data.
15	TDI	Do not connect
16	TDO	Do not connect
17	VCC	3.3V power source
18	I2C_SCL	Clock pin for I2C
19	VSS	Ground
20	RTXC1	Pin 1 for 32.768 KHz oscillator. Optional for RTC with backup battery.
21	I2C_SDA	Data pin for I2C
22	SPI_SCK	Clock pin for SPI
23	SPI_MISO	Master In Slave out for SPI (output)
24	SPI_MOSI	Master Out Slave In for SPI (input)
25	RTXC2	Pin 2 for 32.768 KHz oscillator. Optional for RTC with backup battery.
26	RTCK	Do not connect
27	DBGSEL	Do not connect
28	SPI_SSEL#	Slave Select for SPI (input)
31	VSSA	Analog Ground
36	CD_SSEL#	SD card signal
37	MISC	A miscellaneous pin. Functionality will be determined in special releases

Pin	Name	Description
38	UH_SSEL#	USB host slave select
39	UH_RESET#	USB host reset
42	VCC	3.3V Power source
43	VSS	Ground
44	SD_SCK BL# WAKE	SD card signal. This pin must be high when RESET# is low. Also, when sending deep sleep command, a toggle on this pin will wake up µALFAT
45	UH_GPX	USB host GPX
46	UH_INT	USB host interrupt

All pins that are not listed in the table above must be left unconnected. Always, consult µALFAT-SD schematics. The pins ending with the symbol '#' are active low.

3. µALFAT™ Boot Loader

3.1. µALFAT™ General Description

The boot loader is used to update the firmware of µALFAT. When there is a new firmware release, you can simply download the file from our website and, using simple commands, µALFAT can update itself. At power up µALFAT will send 2 characters 'B' and 'L' indicating that the boot loader is ready to load new file. To exit the boot loader and start µALFAT firmware, send 'R' character. If µALFAT chip detects invalid firmware then it returns 'BL' again. In such case, reprogramming µALFAT chip is required. If µALFAT powers up in standalone mode, it will execute the firmware immediately, no BL will be sent.

3.2. Using the Boot Loader

The easiest way to update µALFAT is by placing the new firmware on any SD card using any PC and a card reader. Once you have the firmware on the card, connect the same card to µALFAT. The file MUST be placed in the root directory, not in any folder. We recommend using newly formatted media. Now you are ready to send the update command. The command is 3 characters, LOK (Load OK)

The boot loader can load firmware from SD cards or from the interface (UART, SPI or I2C) but not from USB memory.

If needed, a user can update the new firmware by sending it over SPI, I2C or UART. All commands return '!' followed by the error number. Also, the boot loader will respond with 'Wxx<CR>' on every sector write, where xx is the sector number.

The firmware file is encrypted file. Loading an incorrect file on µALFAT can damage the chip. Never tamper with the firmware files.

3.3. Boot Loader Commands

Command	Use	Notes
R	Load and run µALFAT firmware	If Boot loader return BL then reprogramming µALFAT is required
LOK	Load firmware file from connected media	Returns !00 if pass or !xx error code
W	Write one sector to internal FLASH	Follow 'W' by the sector number then 512 bytes of sector data. Transaction must be terminated by a checksum byte. Checksum byte is calculated by adding all 512 data bytes
V	Returns the loader version	Returned value is ASCII

Note: *The boot loader is entirely separate program that loads µALFAT firmware. The version number of the boot loader may not mach the version number of µALFAT. New*

µALFAT chips come with no firmware loaded and must load the latest firmware available on our website.

3.4. Communication Interfaces in boot Loader

The boot loader runs simpler drivers for UART, SPI and I2C. All drivers run in pooled mode. Also, there are few things to keeping mind:

1. The processor runs at 10 MHz versus 70 MHz in firmware.
2. SPI clock has to be lower than 1.25 MHz
3. SPI driver is half duplex.

4. Commanding with µALFAT™

4.1. Selecting an Interface

µALFAT uses UART, I2C or SPI to communicate with any external microcontroller. At power up, µALFAT samples SPI_SSEL# and SPI_SCK pins to determine what interface to use. The following table describes the states

SPI_SSEL#	SPI_SCK	Interface
0	0	UART
0	1	Skip boot loader*
1	0	I2C
1	1	SPI

* Do not use.

4.2. UART Interface

In UART interface mode, UART_TX pin is used to send data to your microcontroller and UART_RX pin to receive commands from your microcontroller. The default baud rate for UART is 9600. Always use 8 bit with no parity and 1 stop bit. Baud rate is changeable through BR command. CTS and RTS lines must be used to insure not loss of data at high speeds. CTS pin is an input to µALFAT and when it is high µALFAT will not send data and will wait for it to go low. CTS should be high as long as possible to not slow down µALFAT. RTS pin is output from µALFAT and it is set high when µALFAT FIFO is full. Depending on data transfer speed, RTS pin may never go high because µALFAT is contentiously emptying the FIFO.

Note: The internal UART have hardware TX FIFO that is 16 byte long. After asserting CTS, µALFAT may still send the internal FIFO, up to 16 bytes.

Important: µALFAT will NOT send any data if CTS pin is high! If this pin is not used then it must be connected to ground.

4.3. SPI Interface Mode

In SPI mode six pins are used for communication, to implement slave SPI. Two pins are used for handshaking. SPI_SSEL, SPI_SCK, SPI_MISO, and SPI_MOSI are the standard SPI pins where SSEL is used for Slave Select, SCK is the Serial Clock, MISO is the data line going from µALFAT to your microcontroller, and MOSI is the data line going from your microcontroller to µALFAT.

The Handshaking lines are SPI_DATARDY and SPI_BUSY. When µALFAT has data to send to the PC, it raises DATARDY line. The host (SPI master) must read the data from µALFAT as fast as possible. Data and commands can be sent to µALFAT at any time except when SPI_BUSY is high.

In the nature of SPI, the data flow is full duplex. On every SPI transaction, a data is swapped between the master (your system) and the slave (µALFAT). This is great until

you need to read data from µALFAT but you don't want to send a command. µALFAT implements a software mechanism to handle this issue. Two special numbers are used to handle the flow control. 0xFF and we will call it NDT (No Data Token) and 0xFE and will call it HDT (Half Data Token)

Whenever µALFAT SPI sees NDT (again, it is 0xFF,) it will ignore it. This allows you to read µALFAT without sending data. Of course you are sending NDT but it will be ignored by µALFAT SPI driver.

This is everything you need to do if you are using numbers from 0 to 0xFD. Usually, this is good in most cases as µALFAT commands are ASCII based and most users save files as ASCII text. If you need to send 0xFF to µALFAT, you have to send HDT followed by NDT. µALFAT will understand this as real 0xFF number and will not ignore it. For 0xFE, you have to send HDT followed by HDT. Here is a simple example in 'C' language on how your transmit routine should work:

```
SendData(char c)
{
    if( (c==FF) || (c==0xFE) )
        SendSPI(0xFE);
    SendSPI(c);
}
```

If pooling is a preferred, it is possible to keep pooling µALFAT and not check DATARDY pin at all. When there is no data, µALFAT will return NDT (0xFF)

Important: µALFAT **requires the following in order for SPI to work:**

- SCK is output from your system.
- SCK is idle high.
- SCK is slower than 8 MHz in full power mode and slower than 1.25 in reduced power mode
- Data is shifted out MSB first.
- Data is shifted on the rising edge.

4.4. I2C Interface Mode

Four pins are needed for I2C communication. The USER_I2C_SCL and USER_I2C_SDA are the two I2C bus lines. I2C_DATARDY and I2C_BUSY lines work exactly the same way as SPI_DATARDY and SPI_BUSY work except the interface is half duplex. When DATARDY is high, you can't send data to µALFAT until all the data is read and DATARDY is back low.

µALFAT runs in slave I2C mode always. The slave address of µALFAT is 0xA4. This address is fixed and can't be changed.

5. µALFAT™ Function

The commands and response in µALFAT are made in a way where they can be understood and read by a human and can be easily parsed by any simple 8-bit micro. Each command is one character. Some commands take parameters and others don't. For example, V command doesn't take any parameters and it returns the version number. On the other hand, M requires parameter to run. M creates (makes) a folder on the accessed media device. 'M LOG' creates a folder with the name LOG.

Also, every command must be terminated with Carriage return. This is the enter key on your keyboard. When programming in 'C', it is '\r' or 0x0D.The *backspace* key is supported in case there is a need to discard the last entry. There are many restrictions that must be noticed or µALFAT will not accept the command.

- Commands must be one character.
- Every command must have the exact number of arguments.
- Spaces must be used whenever is required.
- Extra spaces count as errors.
- All numbers are hexadecimal.

So all commands are text based. User can use a terminal program to type in the commands (if using a PC and RS323 level converter.) Also, any microcontroller can send commands to write/read files/folders. In Hyper terminal, you will find it at "properties→Settings→ASCII Setup→Append Line Feed"

***Important**: µALFAT doesn't echo back the data by default. Use '# 1' command to enable echo if needed.*

At power up µALFAT will send BL (Read boot loader section for more details.)

Follow BL by 'R' and you should get the following. No carriage return is needed.

GHI Electronics, LLC
uALFAT(TM) 1.xx

FAT Storage Media with µALFAT

µALFAT can connect to two kinds of storage media types. The media types are SD/MMC cards and USB Mass Storage device (SCSI command subclass, bulk only protocol) which includes thumb flash, USB hard drives and card readers. Keep in mind that all devices must be formatted FAT16 or FAT32.

µALFAT can mount only one File System Media at a time, which means that all opened files and operations in one file system will be terminated if you mount other file system media. For example, if one file is currently open on SD card and then it was switched to a thumb flash, this file handle will be automatically closed before mounting File system on thumb flash.

To access FAT Storage Media, two commands can be used. The first commands is I to access SD/MMC, the other one is U to access USB Mass Storage Device Storage device.

Example1: Mount File System on MMC and make a folder:

I
!00
M FOLDER
!00

Example2: Mount File System on USB Mass Storage Device and make a folder:

U
!00
M FOLDER
!00

µALFAT doesn't detect card removal or replacement. The final application must detect card removal. Also, µALFAT doesn't check the protection switch on SD cards. This means µALFAT will write on protected cards.No harm will be caused to the card if removed as long as no files were open for write.

For step-by-step on how to use the commands, consult µALFAT Tutorial.

µALFAT supports the original FAT file system where files are 8 characters long with extension that is 3 characters long. No long file name is supported. This allows us to speed up the file access time and simplify the user's work.

- Directories (folders)

Folders are supported by µALFAT. And it is possible to move over Folders by CD command.
User must be sure about the current folder that is working in be cause there is no current way to retrieve the current location in folder tree

M FLDR ← this command will create "FLDR" folder

- Files

Files can be opened for read, write or append. Opening a file for read requires that the file exists on the media. Opening a file for write requires that the file doesn't exist on the media. If the file that is being open for read already exists on the media, µALFAT will erase the old folder. Opening a file for append will add data to a file if it exists. If the file doesn't exist then a new file will be made. With µALFAT you can open up to 4 files at the same time using file handles.

Handles are used for fast access to a file. If a user needs to log data to 2 files at the same time, "*VOLTAGE.LOG*" and "*CURRENT.LOG*" file handles become very useful. To do so, open VOLTAGE.LOG under handle 1 and CURRENT.LOG under handle 2. Now start sending your data to handle 1 and 2 instead to the file name.

Example:

To do so, open *VOLTAGE.LOG* under handle 1 and *CURRENT.LOG* under handle 2. Now start sending your data to handle 1 and 2 instead to the file name.
To open a file use,

O 1W VOLTAGE.LOG

The previous command will create new file at the current directory with name "*VOLTAGE.LOG*" and the file handle is *1*. Now to save data to that file use

W 1>10

Now the "*W*" will write data to a file that is open at handle 1 and will write 16 bytes (10 hexadecimal = 16 decimal) to it. After sending carriage return (enter,) uALFAT will return !00. This indicates that µALFAT is ready for your data. Now, start sending your 16 byte of data. When 16 bytes are sent, µALFAT will return another error code.
Note: All numbers given to µALFAT commands are hexadecimal numbers.

This is an example session:

```
>M LOG_DATA          ← create new directory
!00
>A LOG_DATA          ← change directory
!00
>O 1W>VOLTAGE.LOG    ← create new file for write inside LOG_DATA folder
!00
>O 2W>CURRENT.LOG    ← create new file for write inside LOG_DATA folder
!00
>W 1>7               ← command to write 7 bytes (characters) to file at handle 1
!00                  ← µALFAT accepted the write command
12.123V              ← enter the data to go to the file
!00                  ← data written successfully
>W 7>A               ← writes 10 bytes to file at handle 7
!00                  ← µALFAT accepted the write command
1234567890           ← enter your data
!00                  ← data written successfully
W 1>7                ← add 7 bytes(characters) to file at handle 1
!00                  ← µALFAT accepted the write command
10.194V              ← enter the data to go to the file
!00                  ← data written successfully
>                    ← µALFAT is ready for more commands.
```

Note1: file names must **not** contain its path (i.e. "\sub1\file.txt"). **Only** pure names are allowed (i.e. "file.txt").
***Note2*:** Folders and files use naming convention that is same of DOS naming convention. Names must be 8 characters or less and 3 characters or less for the extension and they must be upper case. All upper case letters, numbers and underscore'_' can be used for names and extensions.
Example "DATA.LOG", "NOEXT", "FOLDER", "FOLDER.123", "NO_SPACE.TXT", "12345678.123"

If an error occurred, µALFAT returns the symbol '!' followed by the error number.

6. µALFAT™ Commands Set

All commands below are entered in ASCII. We choose to use ASCII to simplify troubleshooting and to allow humans to enter commands easily. A special case is when accessing the data inside or outside a file. When writing/reading to/from a file or USB Pipe, µALFAT will use any kind of data. Basically, what you send is what goes on the file. It doesn't have to be ASCII.

When µALFAT is done processing a command, it will return an error code in the form "!xx<CR>" where xx is the error number. Also, some commands require returning some extra information. Returned data will come after the symbol $, unless noted otherwise.

You can send multiple commands to µALFAT until its FIFO is full (indicated by BUSY or RTS.) µALFAT will take the commands in one at the time, process them and send responses for each one. Always terminate commands with *carriage return* character.

Note: in all commands descriptions of their outputs will consider the successful executing of the command

6.1. V Get Version Number

It prints the version number of µALFAT firmware. Note that this version is not same or related to the version number of the boot loader. The return value is always in the form "µALFAT x.xx"

Format: V

6.2. # Enable Echo Version Number

µALFAT by default does not echo back the data that it received from the host, it Echo is required this command can be used.

Format: # *n* *n* = 0 Disable echo

n = 1 Enable echo

Example: # 1 Enable echo

6.3. Z µALFAT Sleep

Read Power Modes Section for details.

Format: Z F Full power consumption

Z R Reduced Mode

Z H Hibernate Mode

6.4. T Initialize Timer

Format: T S Share Mode. the RTC runs from "shared" clock and power like processor

T B Backup Mode. it can run from a battery with a separate oscillator so it will keep the time even if the processor is off

6.5. S Set Current Time and Date

Format: S *ddddtttt* *ddddtttt* time and date 32bit structure*

Example: S 34210000 Set 1/1/2006 00:00:00

6.6. G Get Current Time and Date

Format: T X Get 32bit standard time and date structure*

T F Get time and Date in a formatted way to be understandable for human

** Time and Date structure is a DWORD Standard structure in FAT system.*

Bits(s)	*Field*	*Description*
31..25	*Year1980*	*Years since 1980*
24..21	*Month*	*1..12*
20..16	*Day*	*1..31*
15..11	*Hour*	*0..23*
10..5	*Minute*	*0..59*
4..0	*Second2*	*Seconds divided by 2 (0..30)*

6.7. B Set UART Baud Rate

µALFAT contains a fancy divider for baud rate. Using the divider we can set the UART to almost any possible baud rate. Below is a table with some standard baud rates. Note that the values do not work in boot loader. Always run boot loader at default 9600 and after firmware execution, change the baud rate. The table contains two sets of divider values, at 10 MHz and 70 MHz. Use the 70 MHz when run in full power mode and the 10 MHz when running reduced power mode.

Baud Rate	Divider at 70 MHz	Divider at 10 MHz
9600	DCEF	1FAB
119200	6EEF	0386
38400	37EF	067C
57600	43F2	08E5
115200	1EF4	04E5
230400	0FF4	02E5
460800	05A9	01E5
921600	028B	Not possible!

For example: B 1EF4 will set the baud rate to 115200 when using full power mode. µALFAT responds with !00 if command is accepted at the old baud rate and then it changes the baud rate. You will then receive a second !00 at the new baud rate.

Format: B *vvvv* *vvvv:* WORD HEX Baud Rate Divider

Example: B 1EF4 Baud Rate is 115200

6.8. I Initialize and Mount File System on MMC/SD

It is a major initializing Command which is used to mount File System on a newly attached MMC/SD, or to re-mount it. All file handles will be close automatically when using this command.

Format: I

6.9. U Initialize and Mount File System on USB Mass Storage

It is a major initializing Command which is used to mount File System on a newly attached USB Mass Storage Device like USB thumb flash, or to re-mount it. All file handles will be close automatically when using this command.

In case that the Mass Storage device has more than one Logical Unit – like Card Readers – the first Unit will be used.

Format: U

6.10. K Get Media Statistics

Format: K *ssssssss* HEX DWORD Media Size
!00 *ffffffff* HEX DWORD Free Size
$*ssssssss* $*ffffffff*
!00

6.11. Q Quick Format Media

This command resets File Allocation Only. No change occurs to Boot Sector or MBR

Format: Q CONFIRM FORMAT

Note: the function will not be executed till the right confirming string follows the command

6.12. @ Initialize List Files and Folders

It sets List Files and Folders Function pointer to the first Directory entry in the current root.

Format: @

Related Command: N

6.13. N Get Next Directory Entry

This command will print out the Directory Entry “File or Folder” and increments List Files and Folders pointer.

Format:	N !00 *NNNNNNNN.EEE AA ssssssss* !00	*NNNNNNNN* File Name *EEE* File Extension *AA* HEX Byte File Attributes* *ssssssss* HEX DWORD File Size
Example:	N !00 TEST0001.TXT 00 0000FE23 !00 N !00 TEST0002.TXT 00 00001234 !00	Passing N command two times and get the results.

Related Command: @

** File Attributes are one byte Standard Attribute Structure in FAT system.*

7	*6*	*5*	*4*	*3*	*2*	*2*	*0*
Reserved		*Archive*	*Folder*	*Volume ID*	*System*	*Hidden*	*Read Only*

6.14.M Make Directory

It creates a folder

Format: M *foldername* *foldername* follows the short name formation

Example: M MYFOLDER Create a folder with name MYFOLDER

6.15.A Change Directory

It changes the current folder access. Folder must exist.

Format: A *foldername* *foldername* follows the short name formation

Example: A MYFOLDER The current root is in MYFOLDER

6.16.E Remove Directory

This command removes Directory. The directory must be empty from any files or subdirectories.

Format: E *foldername* *Foldername* follows the short name formation

Example: E MYFOLDER Remove the folder with name MYFOLDER

6.17.O Open a file for read, write or append

To open a file for read, write or append in the current Folder, use OF command. The commands require a file handle and the access mode.

Open Modes are:

- 'R' Open for read requires the file to already exist in the current media and the current accessed folder.
- 'W' Open for read will create a new file and give write privilege to it. If the file already exists, it will be erased first then will open a new one for write.
- 'A' Open for append is similar to write with one exception. If the file already existed, it will be opened and the incoming data will be appended at the end.

Important Note: The file name must be in standard short name "8.3" formation.

Note: µALFAT currently has 4 available file handles.

Format:	O *nM*>*foldername*	Open file *foldername* to file handle *n* with access mode *M* which can be R,W or A.
Example:	O 1R>VOLTAGE.LOG	Open file VOLTAGE.LOG to file handle 1 with READ access mode.
	O 0W>CURRENT.LOG	Open file CURRENT.LOG to file handle 0 with WRITE access mode.

Related Commands: C, F, R, P and W

6.18.C Close File Handle

This command closes the opened file and updates file parameters in the file system and confirm that all data in file buffer is written to the media. Then it releases the file handle to be available for new file opening.

It is an important command, especially for file functions that add or modify on files to confirm that data is written on the media and file parameters are updated.

Format:	C *n*	Close File handle *n*
Example:	C 0	Close File handle 0

Related Commands: O, F, R and W

6.19.F Flush File Data

This command does the same function of CF function except releasing the file handle. So it updates file parameters and flushes file buffer data into storage media and keep file handle ready for another write command.

It is made especially for file functions that add or modify on files to confirm that data is written on the media and file parameters are updated.

Format:	F *n*	Close File handle *n*
Example:	F 0	Close File handle 0

Related Commands: O, C and W

6.20.R Read from File

Sending R with the file handle and the byte count and µALFAT will return your data. The file must be opened for read first. To read more data from the file, send another R. If µALFAT couldn't get all the data it promised to return, it will send a filler symbol instead. It is up to the user to decide what the filler is going to be.

Format:	R *nM*>*ssssssss*	*n* File Handle
	!00	*M* Filler Character
	ssssssss Bytes is returned	*ssssssss* HEX DWORD desirable data size to be read
	$*aaaaaaaa*	*aaaaaaaa* HEX DWORD actual read data from file size
	!00	

Example: We have a file with 8 bytes (ABCDEFGH) in it, and it is opened for read with handle number 2.

R 2Z>5	Read 5 bytes from a file #2 with filler Z
!00	µALFAT accepted the command
ABCDE	This is the data coming from the file
$00000005	All 5 bytes are valid
!00	No errors has occurred
R 2Z>5	Read more data. We will request 5 but only 3 are left
!00	No errors has occurred
FGHZZ	µALFAT returned the last 3 bytes but added 2 fillers
$00000003	µALFAT indicating it was able to read 3 bytes only
!00	µALFAT no error indication

Related Commands: O, C, and P

6.21.W Write to File

After a file is opened for write, you can use W to write to that file handle. After W command, µALFAT will respond with error code. If the error code is !00 then writing data to the file is ready. Now, everything goes into the interface goes directly to the file with no interpretation what so ever. After sending all requested data, µALFAT will return the actual write count. In some instances µALFAT could fail writing all the data and it will inform the user of the data loss. Finally, W returns another error code. You can send as many W as you need to write more data to the file. We recommend sending small block of data, around 100 bytes.

Format:	W *n*>*ssssssss*	*n* File Handle
	!00	*ssssssss* HEX DWORD desirable data size to be written
	User sends data	
	$*aaaaaaaa*	*aaaaaaaa* HEX DWORD actual written data to file size
	!00	
Example:	W 1>10	Write 16 bytes to the file opened for handle 1
	!00	µALFAT accepted the command
	1234567890abcdef	16 bytes of data to go into the file
	$00000010	µALFAT was able to write 16 bytes

	!00	No errors has occurred

Related Commands: O, C and F

6.22.P Seek File

This command changes the file pointer position. File must be opened in Read-Mode.

Format:	P *n*>*ssssssss*	*n* File Handle *ssssssss* HEX DWORD new position
Example:	P 1>10	Set file pointer at position the 16th byte in file, supposing that its size is less than 16 bytes

Related Commands: O and R

6.23.D Delete File

Format:	D *filename*	*filename* follows the short name formation
Example:	D TEST.TXT	Remove the file with name TEST.TXT

6.24.? Find File or Folder

This command search for a specific file or folder name in the current folder and print out file's major information which includes size, attributes and date & time of modification.

Format:	? *filename* !00 $*ssssssss* $*AA* $*ddddtttt* !00	*filename* follows the short name formation *ssssssss* HEX DWORD File Size *AA* HEX Byte File Attributes* *ddddtttt* HEX DWORD time and date structure**
Example:	? TEST.TXT !00 $00000F34 $00 $ 34210000 !00	File has been found and its size is 3892 bytes with no special attributes. Last modification time is 00:00:00 date is 1/1/2006

** File Attributes are one byte Standard Attribute Structure in FAT system.*

7	6	5	4	3	2	2	0
Reserved		Archive	Folder	Volume ID	System	Hidden	Read Only

*** Time and Date structure is a DWORD Standard structure in FAT system.*

Bits(s)	Field	Description
31..25	Year1980	Years since 1980
24..21	Month	1..12
20..16	Day	1..31
15..11	Hour	0..23
10..5	Minute	0..59
4..0	Second2	Seconds divided by 2 (0..30)

7. Power Modes

In some applications power consumption is very critical. µALFAT is designed with low power in mind. Although you can completely shut off µALFAT's power for zero power consumption, some user may prefer to put µALFAT to sleep instead of power off to keep the file handles open. Also, µALFAT can run at 2 different speeds.

µALFAT oscillator is 10 MHz. When executing the boot loader or the firmware, µALFAT runs in reduced power mode. In this mode, µALFAT draws about 8mA including 1.8V regulator.
If high performance is required, you can run µALFAT at full power where the core runs at 70 MHz. In this mode, the complete µALFAT-SD board draws about 38mA.
Hibernation is also another option. If the system doesn't need and file operations, µALFAT can hibernate and the core draws about 10uA. Keep in mind that regulators have their own current draw and a minimum draw. µALFAT-SD board draws 1mA when in hibernate.

The command to change the power mode is 'Z'. When changing the clock speed, you have to keep many things in mind. For example, the SPI clock can be the system clock divided by 8 maximum. So if the system is in reduced power mode running at 10 MHz, the SPI SCK can be 1.25 MHz maximum. Also, the UART baud rate needs to be adjusted to hold the correct divider value. Look at "UART Baud Rate" section for more details on the divider values.

Z F>DCEF ← switch to run at 70 MHz and set the UART baud rate to 9600.

Z R>1FAB ← set the system to run at 10 MHz and baud rate of 9600

Now, what if we are using SPI or I2C interfaces? Even in this case you are required to set the UART divider and you can use any one of the values.

When µALFAT goes in hibernation mode, it can be woken up by toggling the WAKE pin. This in is multi purpose and must always be left disconnected from your system except when there is a need to wake µALFAT. This can be accomplished by setting the pin of your micro that connects to WAKE to be input. When there is a need to wake µALFAT, set your pin to low then make the pin output, wit for few micro seconds and then set the pin back to input.

8. Power and Oscillator

µALFAT is very simple to add to your new or existing designs, it requires very few components. Two voltages are needed; 3.3V and 1.8V, crystal and very few RCs, Check the pin description and schematics for more details. You can also find more details on the chipset base (LPC2103) data sheet on Philips website www.semiconductors.philips.com and the schematics of µALFAT-SD and µALFAT-USB on our website www.ghielectronics.com µALFAT requires MAX3421E to run if USB access is neede.

9. Special Firmware

µALFAT's standard firmware contains many features to suite any application. If your application requires a special need, we can implement a special firmware to suite your needs. Loading the updated firmware or special firmware is extremely easy. Look at the boot loader section.

10. Development and OEM Circuit Boards

µALFAT-SD is our µALFAT chip on a small size board. The low cost and the ease of use make µALFAT-SD the ultimate solution for OEMs or hobbyists. The SD/MMC media connector is placed on one side of the board, making it possible to mount the µALFAT-SD board inside your product case using 90 degree brackets. The full schematic and board layout is on our website www.ghielectronics.com.
µALFAT-SD board contains 1.8 V regulator onboard. You will only need 3.3V for power. The pin out of the board gives you access to SD card detect and pins to all µALFAT's signals.

µALFAT-USB is the USB version of µALFAT-SD.

We are also capable of designing special boards to fit your needs.

Appendix A: Firmware Error Codes

Error Number	Description
0x00	No Error
0x01	ERROR_READ_SECTOR
0x02	ERROR_WRITE_SECTOR
0x03	ERROR_ERASE_SECTOR
0x11	ERROR_MBR_SIGNATURE_MISSMATCH
0x12	ERROR_BS_SIGNATURE_MISSMATCH
0x13	ERROR_SECTOR_SIZE_NOT_512
0x14	ERROR_FSINFO_SIGNATURE_MISSMATCH
0x15	ERROR_FAT12_NOT_SUPPORTED
0x21	ERROR_CLUSTER_OVER_RANGE
0x22	ERROR_CLUSTER_UNDER_RANGE
0x23	ERROR_NEXT_CLUSTER_VALUE_OVER_RANGE
0x24	ERROR_NEXT_CLUSTER_VALUE_UNDER_RANGE
0x25	ERROR_NO_FREE_CLUSTERS
0x31	ERROR_FILE_NAME_FORBIDDEN_CHAR
0x32	ERROR_FILE_NAME_DIR_NAME_OVER_8
0x33	ERROR_FILE_NAME_DIR_EXTENSION_OVER_3
0x34	ERROR_FILE_NAME_FIRST_CHAR_ZERO
0x35	ERROR_MEDIA_FULL
0x40	DIR_ENT_FOUND
0x41	DIR_ENT_NOT_FOUND
0x42	ERROR_FOLDER_IS_CORRUPTED_FIRST_CLUSTER
0x43	ERROR_FOLDER_IS_CORRUPTED_DIR_DOT_NOT_FOUND
0x44	ERROR_FOLDER_IS_CORRUPTED_DIR_DOTDOT_NOT_FOUND
0x45	ERROR_ROOT_DIRECTORY_IS_FULL
0x46	ERROR_OPEN_FOLDER_FILE
0x47	ERROR_WRTIE_TO_READ_MODE_FILE
0x48	ERROR_SEEK_REQUIER_READ_MODE
0x49	ERROR_INVALID_SEEK_POINTER
0x4A	ERROR_FOLDER_NOT_EMPTY
0x4B	ERROR_IS_NOT_FOLDER
0x4C	ERROR_READ_MODE_REQUIRED
0x4D	ERROR_END_OF_DIR_LIST
0x4E	ERROR_FILE_PARAMETERS
0x4F	ERROR_INVALID_HANDLE or ERROR_HANDLE_IN_USE
0x61	ERROR_COMMANDER_BAD_COMMAND
0x62	ERROR_COMMANDER_STR_LEN_TOO_LONG
0x63	ERROR_COMMANDER_NAME_NOT_VALID
0x64	ERROR_COMMANDER_NUMBER_INVALID
0x65	ERROR_COMMANDER_WRITE_PARTIAL_FAILURE
0x66	ERROR_COMMANDER_UNKNOWN_MEDIA_LETTER
0x67	ERROR_COMMANDER_FAILED_TO_OPEN_MEDIA
0x68	ERROR_COMMANDER_INCORRECT_CMD_PARAMETER
0x69	ERROR_CHECK_SUM

0x7A	ERROR_USBD_DESCRIPTOR_CORRUPTED
0x7B	ERROR_DESCRIPTOR_NOT_FOUND
0x7C	ERROR_USB_HUB_NOT_FOUND
0x7D	ERROR_HCD_USB_DEVICE_NOT_CONNECTED
0x7E	ERROR_HCD_INIT_FAIL
0x7F	ERROR_HCD_INDEFINED_CASE
0x80	ERROR_SD_UNEXPECTED_VALUE
0x81	ERROR_MMC_INIT_TIMEOUT
0x82	ERROR_SET_BLOCK_SIZE_FAIL
0x83	ERROR_MMC_SEND_COMMAND_FAIL
0x91	ERROR_BOMS_CSW_COMMAND_FAILD or ERROR_USB_MASSSTORAGE_COMMAD_FAILED
0x92	ERROR_BOMS_CSW_STATUS_PHASE_ERROR
0x93	ERROR_BOMS_CSW
0x94	ERROR_BOMS_WORNG_LUN_NUMBER
0x95	ERROR_BOMS_WORNG_CSW_SIGNATURE
0x96	ERROR_BOMS_WORNG_TAG_MISSMATCHED
0xA0	ERROR_USB_MASS_STORAGE_DEVICE_NOT_READY
0xA1	ERROR_USB_MASSSTORAGE_NOT_FOUND
0xB1	ERROR_HCD_BUSY
0xB2	ERROR_HCD_BADREQ
0xB3	ERROR_HCD_UNDEF
0xB4	ERROR_HCD_NAK
0xB5	ERROR_HCD_STALL
0xB6	ERROR_HCD_TOGERR
0xB7	ERROR_HCD_WRONGPID
0xB8	ERROR_HCD_BADBC
0xB9	ERROR_HCD_PIDERR
0xBA	ERROR_HCD_PKTERR
0xBB	ERROR_HCD_CRCERR
0xBC	ERROR_HCD_KERR
0xBD	ERROR_HCD_JERR
0xBE	ERROR_HCD_TIMEOUT
0xBF	ERROR_HCD_BABBLE
0xFD	ERROR_COMMANDER_UNKNOWN_ERROR

Appendix B: Boot Loader Error Codes

Error Number	Description
0xD2	ERROR_VERIFY
0xD3	ERROR_INTERNAL
0xD4	ERROR_CHECKSUM
0xD7	ERROR_INVALID_FRMWARE
0xD8	ERROR_BR_COMMAND
0xDA	ERROR_FILE_IS_EMPTY
0xDB	ERROR_FILE_NOT_FOUND
0xDE	ERROR_UNKNOWN_COMMAND

Appendix C: Licensing

Each µALFAT chip comes with unconditional license of use from GHI Electronics, LLC. There are many patented technologies utilized in µALFAT that must be account for.

- The SD card is used in MMC compatibility mode; therefore, no license is required from the SD organization.
- FAT file system is a patent of Microsoft Corporation. Licensing fee for using FAT file system must be paid by companies who wish to use FAT file system in their products. For more information, visit Microsoft's website.
http://www.microsoft.com/mscorp/ip/tech/fat.asp

Please note that our products don't support long file names and most likely you don't need to pay any licensing but you should contact Microsoft for more details.

µALFAT uses USB through USB host controllers, no USB licensing is necessary.