ALFAT SoC Processor

Rev. 1.03 Date: March 19, 2012 User Manual

A high performance FAT file system SoC processor with dual USB Host interfaces and 4-bit SD interface. Controlled through UART, SPI or I2C.

FAT16/32 With LFN USB Host

Document Information	Description
Abstract	ALFAT SoC processor concept, pin-out, specifications, commands, hardware integration guide and full information needed to implement a solution using this processor.
Firmware	V1.02

Revision History

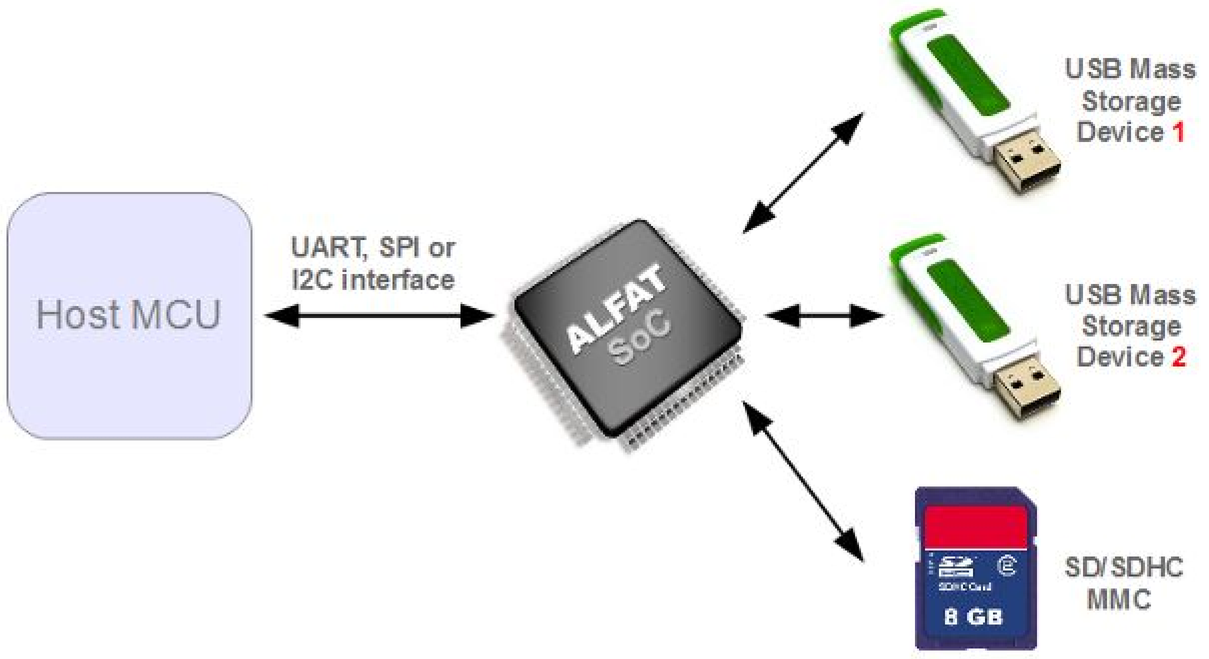
Rev No.	Date	Modification
Rev 1.03	03/19/12	Fixes to multiple typos. Clarifying some ambiguous points.
Rev 1.02	03/15/12	Adding 6.4.Updating the firmware using a terminal console Fixes to 8.2.ALFAT SD board pin-mount tables
Rev 1.01	03/08/12	Updates to 4.3.SPI interface mode section Updates to 6.2.Firmware Updater App
Rev 1.00	02/15/12	Preliminary document

Table of Contents

1. Introduction

1.1. ALFAT SoC processor Concept

Adding FAT file system to products requires a lot of resources from the system and needs intensive development efforts. This also requires USB Host drivers and SD memory drivers to be able o access storage medias such as SD cards or USB Mass Storage devices. Additionally, licensing patented technologies, such as LFN is a lengthy and expensive process. Thanks to ALFAT SoC Processor, any simple system can now access files on SD cards and USB memory drives in a very short time, with minimal resources through simple UART, SPI or I2C interface. ALFAT SoC processor is capable of accessing two USB mass storage devices and one memory card simultaneously. This gives the user unique features such as copying data from one media to another.


Host MCU
UART, SPI or I2C interface
ALFAT SoC
USB Mass Storage Device 1
USB Mass Storage Device 2
SD/SDHC MMC
8 GB

The Host MCU controls ALFAT through simple commands sent through UART (serial), SPI or I2C. The commands give the host MCU the ability to access files on the storage media. With ALFAT, file access rate can reach 4000 KBytes/sec.

An important advantage of ALFAT is that supports Long File Name **LFN** and it is licensed by Microsoft for commercial use. Solutions that depends on ALFAT can use LFN commercially without the need for any additional licensing.

1.2. Example applications:

- High speed Data loggers.
- Automated Machinery.
- Digital picture viewer.
- Consumer products.

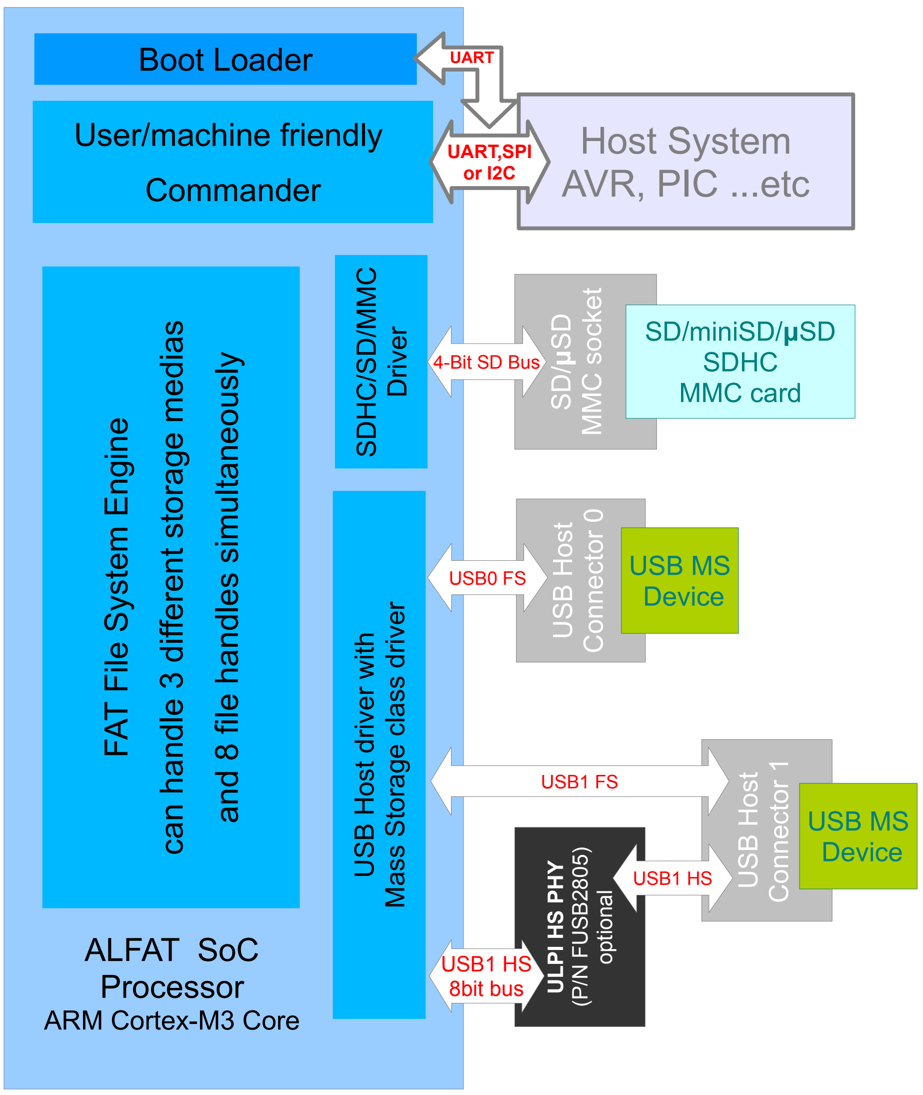
1.3. Key features

- Built-in 2-port USB Host controller.
- FAT16 and FAT32 File system.
- No limits on media size, file size or file/folder count.
- LFN (Long File Name), licensed by Microsoft.
- Friendly user interface through UART,SPI or I2C.
- Programmable UART baud-rate.
- Up to 8 simultaneous file access.
- SD/SDHC card support, no 2GB limit. **GHI electronics is an SD association member.**
- MMC card support.
- Built-in 2x USB 2.0 FS PHY USB, 12mbps.
- One USB ports is capable of HS 480mbps through external ULPI PHY.
- High speed 4-bit SD card interface.
- Up to 4000 KBytes/sec file access speed on SD cards.
- Up to 4000 KBytes/sec file access speed on USB (with ULPI HS PHY).
- Up to 1000 KBytes/sec file access speed on USB Full Speed (no ULPI HS PHY).
- RTC (Real Clock Time) with separate power domain.
- All I/O pins are 5 volt tolerant **EXCEPT RESET PIN**.
- Small surface mount package, LQFP 64 pin.
- Single 3.3V power source.
- Low power consumption, TBD mA.
- -40˚C to +85˚C operational temperature.
- RoHS Compliant/Lead free.

2. Architecture

ALFAT SoC processor is an ARM Cortex-M3 processor that runs a robust file system engine with SD and USB host (mass storage) drivers. The processor access storage media though its 4-bit SD interface and two USB host 2.0 interface. One of the USB host ports is capable running at high speed 480mbps with external ULPI HS PHY chip.

ALFAT SoC processor provides 3 different standard access interfaces for the host system, UART, SPI or I2C.


Boot Loader
UART
User/machine friendly Commander
UART,SPI or I2C
Host System AVR, PIC ...etc
FAT File System Engine can handle 3 different storage medias and 8 file handles simultaneously
SDHC/SD/MMC Driver
4-Bit SD Bus
SD/µSD MMC socket
SD/miniSD/µSD SDHC MMC card
USB Host driver with Mass Storage class driver
USB0 FS
USB Host Connector 0
USB MS Device
USB1 FS
USB Host Connector 1
USB MS Device
USB1 HS
ULPI HS PHY (P/N FUSB2805) optional
USB1 HS 8bit bus
ALFAT SoC Processor ARM Cortex-M3 Core

2.1. Commander

The function of the commander is to provide the user with a protocol and a set of commands to control the processor and access the files on the storage devices.

The physical interface that the user (the host system) can use to access the commander is UART, SPI or I2C serial port.

Commands are human-readable ASCII format. This allows for easier development and troubleshooting. But at the same time, the commands are designed to easy to parse by the host software.

2.2. FAT File System Engine

The function of the file system engine is to handle accessing the file system according to FAT standards. It has been optimized for high speed access and with high performance.

Here are some of the capabilities of this engine:

- FAT16, FAT32.
- Licensed Long File Names support. Licensed to be used on ALFAT by Microsoft.
- Access up to 8 opened files simultaneously.
- Complete directories (folders) support.
- File access functions include read, write, append, seek, tell, find, delete, remove folder ...etc.
- High speed read or write access, up to 4000KBytes/sec.
- No limits on media size, file size or file/folder count.

It is important to note here that the 8 file handle limit is only on how many simultaneous files are open. ALFAT SoC processor Has no limits on how many files can be opened and closed.

2.3. Memory Card Access (SDHC, SD or MMC)

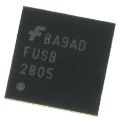
ALFAT SoC processor includes memory card driver internally that supports SD, SDHC and MMC cards. This gives ALFAT the ability to access a wide range of memory cards such as standard or high capacity SD/µSD cards or multimedia cards. There is no limit on the card capacity.

Unlike typical solutions that acces the card through SPI-based interface, ALFAT's hardware provides a 4-bit SD bus interface for higher performance.

GHI Electronics is a member of SD association.

2.4. USB Mass Storage Access

ALFAT SoC processor is capable of accessing FAT file system files on USB mass storage devices. The hardware provides two standard Full Speed* USB 2.0 compatible host interfaces (USB0 and USB1) that use the internal USB PHY. Only 22ohm resistors and USB host connector is needed.



USB1 interface is also capable of running in USB 2.0 High Speed mode by adding ULPI High Speed+ PHY chips like FUSB2805. This options quadruples the file system access speed.

**Full-Speed USB 2.0 is 12mbps.*

+High-Speed USB 2.0 is 480mbps.

2.5. Boot Loader

The boot loader is a piece of software that boots the system up. It verifies and runs ALFAT firmware. Also It gives the hosting system an interface for firmware maintenance.

GHI Electronics regularly maintains ALFAT firmware with improvements and bug fixes. ALFAT boot loader allows the hosting system to update the firmware.

The boot loader can only be accessed through the UART port and it uses XMODEM 1K to transfer the firmware file to ALFAT. Boot Loader and firmware update section explain this in more details.

3. Package and Pin-Out

ALFAT SoC package is standard 10x10mm LQFP64.

The following Table includes a brief description of ALFAT SoC processor pins.

Pin	Name	Description
1	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. **This pin must have power, even if the internal RTC is not needed.**
2	NC	Should not be connected
3	OSC32_IN	Pin 1 for the 32.768 KHz crystal for the real time clock. Optional.
4	OSC32_OUT	Pin 2 for the 32.768 KHz crystal for the real time clock. Optional.
5	OSC_IN	Pin 1 for 12 MHz system crystal.
6	OSC_OUT	Pin 2 for 12 MHz system crystal.
7	$\overline{\text{RESET}}$	Reset signal, the pin pulled up internally. Active Low. THIS PIN IS NOT 5VOLT TOLERANT.
8	USB1_ULPI_STP	USB High Speed PHY signal. Do connected if PHY is not use.
9	NC	Should not be connected.
10	USB1_ULPI_DIR	USB High Speed PHY signal. Do connected if PHY is not use.
11	USB1_ULPI_NXT	USB High Speed PHY signal. Do connected if PHY is not use.
12	VSSA	Ground
13	VDDA	3.3V Power source.
14	#WAKEUP	Wake up ALFAT from hibernate when set low. Check power mode command for more details
15	NC	Should not be connected.
16	NC	Should not be connected.
17	USB1_ULPI_D0	USB High Speed PHY signal. Do connected if PHY is not use.
18	VSS1	Ground.
19	VDD1	Power, 3.3V.

Pin	Name	Description
20	ACTIVE	Active pin
21	USB1_ULPI_CK	USB High Speed PHY signal. Do connected if PHY is not use.
22	SPI_MISO UART_BUSY	MISO SPI interface and UART BUSY pin, 5V tolerant.
23	SPI_MOSI	MOSI SPI interface, 5V tolerant.
24	NC	Should not be connected.
25	NC	Should not be connected.
26	USB1_ULPI_D1	USB High Speed PHY signal. Do connected if PHY is not use.
27	USB1_ULPI_D2	USB High Speed PHY signal. Do connected if PHY is not use.
28	BOOT1	Add a 10K resistor to ground.
29	USB1_ULPI_D3	USB High Speed PHY signal. Do connected if PHY is not use.
30	USB1_ULPI_D4	USB High Speed PHY signal. Do connected if PHY is not use.
31	VCAP_1	Connect to a 22uF to ground.
32	VDD1	Power, 3.3V.
33	USB1_ULPI_D5	USB High Speed PHY signal. Do connected if PHY is not use.
34	USB1_ULPI_D6	USB High Speed PHY signal. Do connected if PHY is not use.
35	USB1_DM	Data Minus, USB Port 1. This is only used if no HS PHY is installed. NC if PHY is used. Add 22ohm resistor in series if used.
36	USB1_DP	Data Plus, USB Port 1. This is only used if no HS PHY is installed. NC if PHY is used. Add 22ohm resistor in series if used.
37	NC	Should not be connected.
38	NC	Should not be connected.
39	SD_D0	D0, 4-bit SD Bus. Add a 47K resistor between this pin and VDD.
40	SD_D1	D1, 4-bit SD Bus. Add a 47K resistor between this pin and VDD.
41	ULPI_26MHZ	This pin generates 26MHz clock. It is usually used to clock the USB High Speed PHY.
42	UART_TX	TX UART interface, 5V tolerant.
43	UART_RX SPI_BUSY I2C_BUSY	RX UART interface BUSY pin for SPI and I2C, 5V tolerant. Add a 10K resistor between this pin and VDD.
44	USB0_DM	Data Minus, USB Port 0. Add 22ohm resistor in series.
45	USB0_DP	Data Plus, USB Port 0. Add a 22ohm resistor in series.
46	NC	Should not be connected.

Pin	Name	Description
47	VCAP_2	Connect to a 22uF to ground.
48	VDD3	Power, 3.3V.
49	NC	Should not be connected.
50	SPI_SSEL	SSEL SPI interface, 5V tolerant.
51	SD_D2	D0, 4-bit SD Bus. Add a 47K resistor between this pin and VDD.
52	SD_D3	D1, 4-bit SD Bus. Add a 47K resistor between this pin and VDD.
53	SD_CLK	CLK, 4-bit SD Bus.
54	SD_CMD	CMD, 4-bit SD Bus. Add a 47K resistor between this pin and VDD.
55	SPI_SCK	SCK SPI interface, 5V tolerant.
56	NC	Should not be connected.
57	USB1_ULPI_D7	USB High Speed PHY signal. Do connected if PHY is not use. Add a 10K resistor to VDD.
58	I2C_SCL	SCL I2C Interface.
59	I2C_SDA	SDA I2C Interface.
60	NC	Should not be connected.
61	NC	Should not be connected.
62	NC	Should not be connected.
63	VSS2	Ground.
64	VDD4	Power, 3.3V.

4. ALFAT access interface

ALFAT accepts commands over UART, SPI or I2C. 2 pins are sampled on power up to determine the interface. The pins are SPI_SSEL and SPI_MOSI.

4.1. Selecting the access interface

On power up, the system should hold ALFAT in reset state until the power is stable. This is done by holding reset pin low. Then, pins SPI_SSEL and SPI_MOSI must be set to the desired serial interface, UART, SPI, I2C or even to force the boot loader. The boot loader is needed for firmware updates. Those pins have internal pull-down resistor so an unconnected pin is considered low. After pins are set, the reset pin must be set to high state to allow ALFAT to boot up. **Note that the reset pin is NOT 5V tolerant.**

ALFAT needs TBDms (use 1second for now) to sample these pins. If the selected interface is SPI, these pins can be change to their appropriate SPI functions; otherwise, it is safe to leave the pins in the same state used on power-up.

SPI_SSEL	SPI_MOSI	Interface
low	low	UART
low	high	Boot loader
high	low	I2C
high	high	SPI

4.2. UART Interface

The UART interface use UART_TX pin to send data out of ALFAT and UART_RX pin to receive data into ALFAT. UART_BUSY pin should be monitored when sending data to ALFAT. When UART_BUSY is high, no data should be transmitted to ALFAT.

While using UART_BUSY pin is highly recommended, it is safe to ignore this signal if the command/data sent to ALFAT is less than 4K Bytes in one continuous burst.

Here are the steps needed to use ALFAT without UART_BUSY pin:

1. Send command. This can be any command and if writing to a file, the sent data size must be less than 4K Bytes in one shot. (More Data can be sent in the following command).

2. Wait till response is back before sending more data. This guarantee the internal receive buffer in ALFAT is empty.

4.3. SPI Interface Mode

SPI interface uses 4 pins: SPI_SSEL, SPI_MOSI, SPI_MISO and SPI_CLK, with optional SPI_BUSY. The busy pin goes high when the internal receive buffer is full.

Sending and receiving data from ALFAT over SPI is done through frames. A write frame and read frame. The frames are simply a layer that sits between the actual SPI data bus and the data streams to/from ALFAT. Internally, ALFAT processes the actual data payload in the frames.

Write Transaction

Request Frame

The frame consists of two parts: Header and payload.

- The header consists of three bytes. The first byte is the frame type, the second and the third bytes are 16-bit transaction length.
- Payload: This is actual transaction payload needed to be sent to ALFAT. The payload section size MUST match the declared in the transaction size field.

Frame Type	16-bit Transaction Size N		Payload			
0x01	Size LSB	Size MSB	Byte 1	Byte 2	...	Byte N

Response Frame

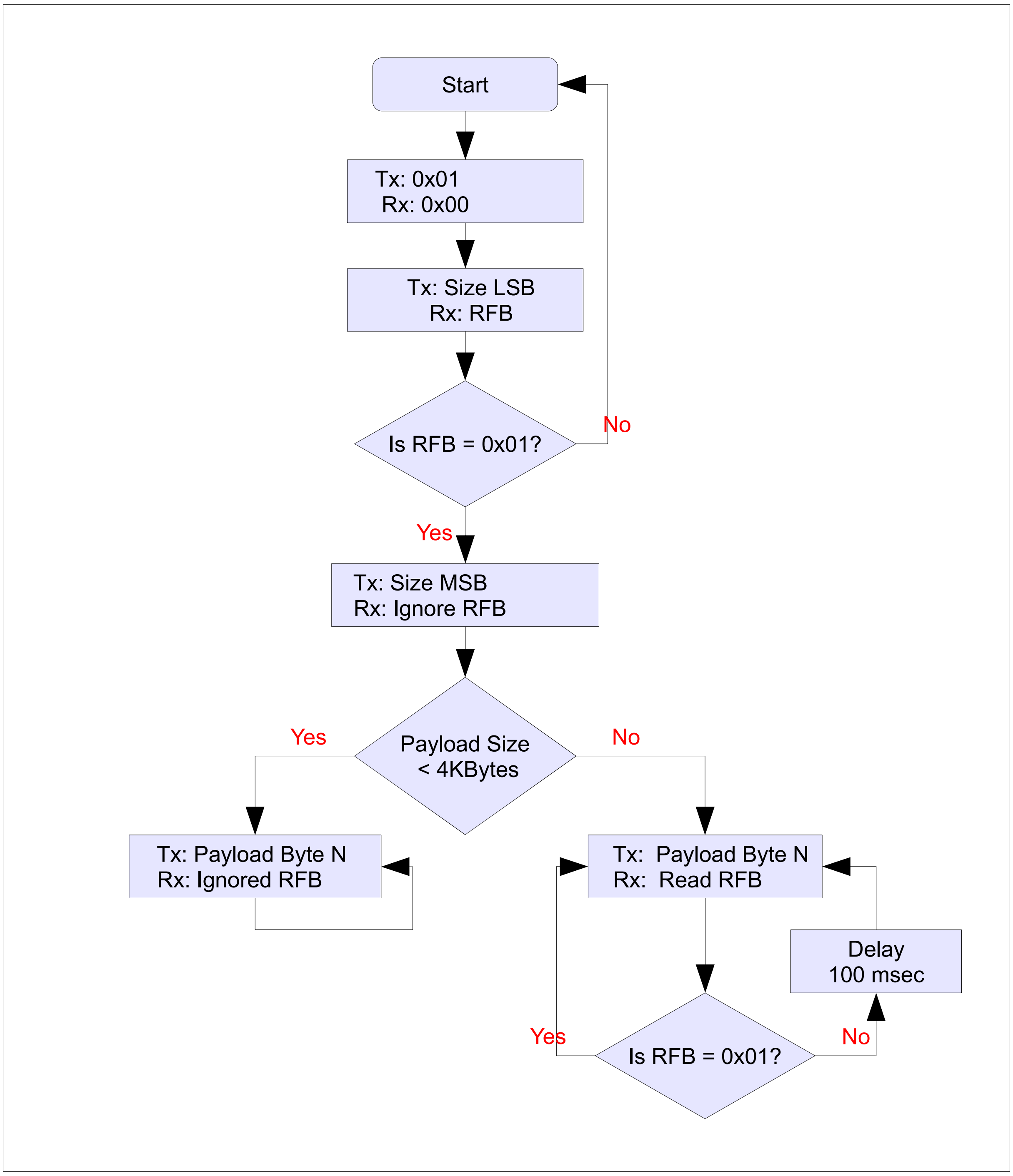
0x00	RFB	RFB	RFB	RFB	...	RFB

RFB is Ready Flag Byte.

When a write transaction request is being sent, ALFAT returns zero with the first received byte (Frame Type) and then it returns a ready flag with the received second byte (Size LSB). If the ready flag is 0x00, this means that ALFAT is not ready and the current frame is terminated and the user must not send more bytes. If the ready flag is 0x01, this means that ALFAT is ready to receive and the user can proceed and send Size MSB and the payload.

If the size of payload is less than or equal to 4096 Bytes, nothing else needs to be checked. But, if the payload is more than 4096 Bytes, the user must check back the returned ready flag with every transmitted byte. If ALFAT got not ready during data payload transmission, the frame is not terminated and the user must continue sending data but a 100 msec delay must be performed after every byte received with a not-ready flag.

SPI write frame flow chart



Read Transaction

Processing read transaction is done through sending read request frame and receiving response frame that carry the useful payload.

Request Frame

The frame consists of two parts: a header and stuffing byte payload.

- The header consists of three bytes. The first byte is the frame type, the second and the third bytes are the requested 16-bit transaction length.
- Stuffing byte payload: This should always consists of 0x00. This section size should equal P, where P is the actual payload size that ALFAT will send back.

Frame Type Resp.	**Requested Transaction 16-bit Size N**		**Stuffing Byte Payload, the size is P.**			
0x02	N LSB	N MSB	0x00	0x00	0x00	0x00

Response Frame

The frame consists of two parts: a header and payload.

- The header consists of three bytes. The first byte is the frame type response which is always 0x00, the second and the third bytes are 16-bit value ACT that reflects how many bytes ALFAT currently has ready to be sent out to the hosting system. ACT value could be more or less that N (the requested transaction size)
- Payload: This is the actual transaction payload that is being send out from ALFAT to the host. This section size is P.

That actual Payload size is calculated this way: If the requested transaction size N is larger that the size of available to be sent ACT then P should equal ACT otherwise P should equal N.

Frame Type Resp.	**Internal buffer filled portion 16-bit Size ACT**		**Payload Bytes, the size is P if N > ACT then P = ACT else P =N**			
0x00	ACT LSB	ACT MSB	Byte 1	Byte 2	...	Byte P

Important: Users shouldn't rely on ACT size because the size internally may change after the read command was sent. it used only as an estimate.

SPI Bus Configurations

- The maximum SPI clock is 24MHz.
- SPI clock Idle state is Low.
- Sampling is at the rising edge.
- SPI_BUSY active state is High.
- SPI_SSEL active state is Low. The user is free to toggle SPI_SSEL with every byte or with every chunk of bytes.

4.4. I2C Interface Mode

The I2C interface on ALFAT use I2C_SCL and I2C_SDA pins for communication. Optionally, I2C interface also provides I2C_BUSY pin.

When writing data, if I2C_BUSY is high, data shouldn't be transmitted to ALFAT. The user can ignore I2C_BUSY pin if the data transmitted to a file is less than 4K Bytes. Of course the user can write much larger files, this is only the chunk size in write command.

To transmit data to ALFAT, the user will send ALFAT's I2C address with W flag as specified in the I2C specifications. Any data after will be sent to the internal FIFO.

Reading data is similar and started by transmitting ALFAT's I2C address followed by a read of one or more bytes. When reading data, ALFAT transmits a No-Data-Token (NDT) when no data is available. If reading a file and the file contains 0x00 or 0xFF, ALFAT will return HDT followed by the actual byte. This table explains further:

Actual Data coming from File	Data transmitted from ALFAT I2C bus
0x00	0xFF followed by 0x00
0xFF	0xFF followed by another 0xFF
0x01 … 0xFE	Data is sent as is

Important: With NDT/HDT being used, the actual data size on a files maybe different than what is transmitted. For example, If a file has one byte and this byte value is zero, ALFAT will actually send 2 bytes, a 0xFF followed by 0x00. This is all in the I2C interface drivers of course, no in the commands sent to ALFAT. The user will still send read file command with one byte.

Bus Configuration

ALFAT's I2C slave address is 0xA4. This is a fixed address and can't be changed. The maximum allowed I2C clock is 400Khz. The master must provide pull up resistors on the bus as specified in the I2C specifications, normally 2.2K.

5. ALFAT Command Set

All commands are entered in human-readable ASCII. This was chosen to simplify troubleshooting and development. A terminal programs can be used to enter all commands manually. Note that the data going into a file or data being read from a file do not have to be ASCII. The data to and from the file are transferred as is.

When ALFAT is done processing a command it will always return an error code in this form "!xx\n" where xx is the error number. If a command return information, this information will be marked with $ symbol, unless otherwise is noted.

You can send multiple commands to ALFAT SoC until its FIFO is full (indicated by BUSY or RTS). ALFAT will process one at the time in the same order.

Command	Description	Command	Description
V	Get version number	I	Initialize MMC/SD or USB
Z	Low power	O	Open file to a free handle
T	Initialize Real Time Clock	W	Write to a file
S	Set current time and date	R	Read from a file
G	Get current time and date	F	Flush file
B	Change Baud rate	C	Close file
#	Enable echo	P	File seek
J	Read status register	Y	File tell
E	Test media	D	Delete file or folder
K	Get free size	?	Find file or folder
@	Initialize directory list	M	Copy From File to Another
N	Get next directory entry	A	Rename file
Q	Format		

General command notes:

- Any command with its argument can not exceed 200 bytes. This is not related to the 4096 byte data FIFO.
- Commands are terminated with line-feed and ALFAT returned data termination is line-feed as well. However, ALFAT will accept carriage-return instead of line feed, but not both.
- The user must read back the responses for each command properly and check

whether the command was successful.

- Commands must have the exact formatting. Extra spaces are not allowed.
- All numbers are Hexadecimal represented in ASCII. For example, to send the decimal number 16 to ALFAT SoC which is 10 in Hexadecimal, you send 0x31 which is ASCII for 1 and 0x30 which is ASCII for 0. Also, for Hexadecimal numbers A to F, they must be entered in upper case letters. If using a terminal to enter commands manually, then only type in the number.
- Some commands have multiple error codes. Usually, the first error code denotes the command is accepted and then it is processed. Another error code is sent when the command has finished processing successfully. If the first error code was a no success then the command is terminated. Note that the second error code will never be sent.
- All commands that accept file name, such as O and D commands, require a full file path. File name is not enough. For example, “U1:\folder\data.log” is good but “data.log” shouldn't be used.

V - Get Version Number

Prints the version number of ALFAT SoC firmware. Note that this version is not the same or related to the version number of the boot loader.

Format	V<LF> vx.x.x<LF> !00<LF>	Returns version number
Example	V<LF> v1.0.0<LF> !00<LF>	The version number is 1.0.0

# - Enable Echo

Enabling echo makes ALFAT echos back the data it receives over UART.

Format	#<SP>n<LF>
	!00<LF>
		n = 0 Disable echo
		n = 1 Enable echo

Example	#<SP>1<LF> !00<LF>	Enable echo

Echo is disabled by default.

Z - Set Power Mode

Format	Z<SP>n<LF> !00<LF>	n : Power mode number
Example	Z<SP>1<LF> !00<LF>	Put ALFAT in hibernate mode. The !00 will be returned after ALFAT wakes up by setting #WAKEUP pin low.

Power mode number

0	Reserved
1	Hibernate mode: set #WAKEUP pin low to let ALFAT get out of the hibernate mode.
2	Reserved
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Reserved
8	Reserved
9	Reserved

T - Initialize Real Time Clock

Format:	T<SP>S<LF> !00<LF>	Shared Mode. The RTC runs off the same processor clock.
	T<SP>B<LF> !00<LF>	Backup Mode. The RTC clocks using the external 32.768Khz crystal and runs VBAT power(1.65 to 3.6 V) backup coin battery. This ensures that the RTC keeps clocking even if ALFAT main power is down.

S - Set Current Time and Date

Format	S<SP>ddddtttt<LF> !00<LF>	ddddtttt time and date 32bit structure*
Example	S<SP>34210000<LF> !00<LF>	Set 1/1/2006 00:00:00

*Time and Date structure is a 32-bits standard structure used in FAT system. For example, 0x34212002 is 01/01/2006 – 04:00:04

Bits(s)	Field	Description	0x34212000 Bits in Binary
31..25	Year1980	Years since 1980	001 1010
24..21	Month	1..12	0001
20..16	Day	1..31	0 0001
15..11	Hour	0..23	0 0100
10..5	Minute	0..59	00 0000
4..0	Second2	Seconds divided by 2 (0..30)	0 0010

G - Get Current Time and Date

Format	G<SP>D<LF> MM-DD-YYYY<LF> !00<LF>	Get current date.
	G<SP>T<LF> HH:MM:SS<LF> !00<LF>	Get current time.

B - Set UART Baud Rate

Format	B<SP>sssss<LF> !00<LF> !00<LF>	sssss: The standard baud rate value in HEX. The first !00 will be sent when the baud rate value is correct.
Example	B<SP>1C200<LF> !00<LF> !00<LF>	Set the baud rate at 115200.

I - Initialize and Mount MMC/SD or USB

This command should be called to initialize and mount the file system. Any other file-related commands will fail if this was not called first.

Format	I<SP>X:<LF> !00<LF>	X: Drive name: M: Memory Card drive U0: USB Flash drive 0 Full-Speed U1: USB Flash drive 1 Full-Speed U1:H USB Flash drive 1 High-Speed. Requires an external HS PHY.
Example	I<SP>M:<LF> !00<LF>	Initialize memory card
	I<SP>U0:<LF> !00<LF>	Initialize USB 0 at Full-Speed mode
	I<SP>U1:H<LF> !00<LF>	Initialize USB 1 at High-Speed mode

J - Read Status Register

Returns system status register.

Format	J<LF> !00<LF> $ss<LF> !00<LF>	Read status register. $ss is 1 byte in HEX that shows the status of media drive. First !00 is sent before starting calculations
Example	J<LF> !00<LF> $11<LF> !00<LF>	Indicating card detect pin is high and USB 1 is in high speed mode.

This table shows the bit mapping.

Bit Number	Status
0	SD Detect pin state
1	SD Protect pin state
2,3	Reserved
4	0: USB1 is in Full Speed mode. 1: USB1 is in High Speed mode.
5,6,7	Reserved

K- Get Free Size

Gets media total remaining free size. Note this command may take several seconds for calculations to finish depending on the media size.

Format	K<SP>X<LF> !00<LF> $ssssssss<LF> !00<LF>	X: Drive name, included: M: Memory Card drive U0: USB Flash drive 0 U1: USB Flash drive 1 ssssssss 4 bytes in HEX free size in drive. First !00 is sent before starting calculations
Example	K<SP>U1:<LF> !00<LF> $717F0000<LF> !00<LF>	Size Available on USB Flash drive 0 is 1904148480 bytes

@ - Initialize Files and Folders List

To list files/folders at a certain path, first use this command to initialize the list counter, then call N command to get a directory entry. Every time N command is called, ALFAT retrieves an entry of the list.

No other commands should be called between the N commands, otherwise the user should start over and call @ command.

Format	@<SP>full path<LF> !00<LF>	Full path: the full path needed to initialize included drive name.
Example	@<SP>M:\TEST\TMP<LF> !00	Initialize folder M:\TEST\TMP

N - Get Next Directory Entry

Calling N command retrieves a directory entry from the directory list initialized using @ command. Every time N command is called, ALFAT retrieves an entry of the list and increment the list pointer.

When list pointer reaches the end of the list and N is called again, ALFAT returns error code 0x04 indicating the end of the list has been reached.

Format	N<LF> !00<LF> NNNNN.EEE<LF> $AA<LF> $ssssssss<LF> !00<LF>	NNNNN File Name EEE File Extension (if any) AA 1 byte, in HEX, File Attributes* ssssssss 4 bytes in HEX file size
Example	N<LF> !00<LF> TEST0001.TXT<LF> $00<LF> $0000FE23<LF> !00<LF> N<LF> !00<LF> TEST0002.TXT<LF> $20<LF> $00001234<LF> !00<LF>	Passing N command two times and getting the results.

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

7	6	5	4	3	2	1	0
Reserved		Archive	Folder	Volume ID	System	Hidden	Read Only

O - Open a File for Read, Write or Append

The command requires a free file handle and a access mode.

Open Modes are:

- 'R' Open for read, requires the file to exist at the specified path.
- 'W' Open for write, will create a new file and give write privileges to it. If the file already exists, it will be erased and re-written.
- 'A' Open for append, will write data to the end of the file. If the file does not exist, it will be created.

ALFAT has 16 available file handles. Each file once opened must be associated with a handle. Closing the file, would free up the file handle.

Note: ALFAT can access unlimited number of files. The limitation is 16 simultaneous opened files but a handle can be closed then used to open any other file.

Format	O<SP>nM>full path<LF> !00<LF>	Open file file name followed full path and associate it with handle n and access mode M. n is the handle number, in HEX, and can be 0, 1, 2,3,4,5,6, 7, 8, 9, A, B, C, D, E or F. M can be R, W or A
Example	O<SP>1R>M:\TEST\VOLTAGE.LOG<LF> !00<LF>	Open file VOLTAGE.LOG with file handle 1 and read access mode. This file is located in TEST older on SD memory card.
	O<SP>0W>U0:\DAT\CURRENT.LOG<LF> !00<LF>	Open file CURRENT.LOG using file handle 0 and write access mode on USB0. If folder DAT is not available, it will be created automatically.

R - Read from File

This command is called to read data through a file handle assigned to an open file with read mode. The user determines the size of data to be read and the filler byte in the command parameters.

After executing the command, ALFAT sends back the data and increments the internal file pointer. If the the file pointer reached the end of the file, then ALFAT returns filler bytes till it reaches the total size required in the R command, then it returns the actual data size.

Note: the data is sent directly to the user with no formatting (no interpretation or conversion) in UART and SPI mode. In I2C mode, some of the bytes might be replaced by two bytes. Check I2C interfaces sections for more details.

Format	R<SP>nM>ssssssss<LF> !00<LF> ssssssss Bytes are returned $aaaaaaaa<LF> !00<LF>	n File Handle 0, 1, 2,3,4,5,6, 7, 8, 9, A, B, C, D, E or F. M Filler Character ssssssss 4 bytes Max. in HEX data size to read aaaaaaaa 4 bytes always in HEX actual read size
Example	We have a file with 8 bytes (ABCDEFGH) in it and it is opened for read with handle number 2.	
	R<SP>2^>5<LF> !00<LF> ABCDE$00000005<LF> !00<LF> R<SP>2Z>5<LF> !00<LF> FGH^^$00000003<LF> !00<LF>	Read 5 bytes from file handle 2 with a filler ^ 5 bytes are read Read 5 more bytes. Only 3 bytes are available and 2 are filler bytes

W - Write to File

This command is called to write data through a file handle assigned to an open file with write or append mode:

1. Send W command with file handle and the data size.
2. Wait till you get the acknowledge.
3. Send the data.

The sent data is written directly to the file as is (no interpretation or conversion). Also, the user should make sure that the sent data size matches the size declared in the command. If an error occurs while writing, ALFAT still expects all the data then it sends back the error code.

There is no line-feed sent after the data. If writing “Hello” to a file, just send the 5 bytes and then ALFAT will respond with how many bytes were written. In this case, it should be $00000005.

To make sure that the data is written to a file, the file must be flushed (F command) or closed (C command) when done, or there will be a risk of losing data or corrupting the file system if the storage media was removed or if there was a power loss.

Format	W<SP>n>ssssssss<LF> !00<LF> User sends data (ssssssss bytes) $aaaaaaaa<LF> !00<LF>	n File Handle 0, 1, 2,3,4,5,6, 7, 8, 9, A, B, C, D, E or F. ssssssss 4 bytes Max. in HEX data size to be written aaaaaaaa 4 bytes always in HEX actual written size
Example	W<SP>1>10<LF> !00<LF> 1234567890abcdef $00000010<LF> !00<LF>	Write 16 bytes to the file associated with handle 1

F - Flush File Data

This command flushes (commits) the data of an opened file. The file will still be opened and associated with a handle, after calling the command. This command is useful to make sure all data are physically saved in the media.

To make sure that the data is written to a file, the file must be flushed (F command) or closed (C command) when done, or there will be a risk of losing data or corrupting the file system if the storage media was removed or if there was a power loss.

Format	F<SP>n<LF> !00<LF>	Flush File handle n n can be 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E or F
Example	F<SP>0<LF> !00<LF>	Flush File handle 0

C - Close File

This command issues a flush file (F – Command) internally and then release the file handle.

Format	C<SP>n<LF> !00<LF>	Close File handle n n can be 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E or F
Example	C<SP>0<LF> !00<LF>	Close File handle 0

P - File Seek

This command changes the current byte position in a file. Valid values are ranging from 0 to file size.

Format	P<SP>n>ssssssss<LF> !00<LF>	n File Handle 0 through F ssssssss 4 bytes new position
Example	P<SP>1>10<LF> !00<LF>	Set file pointer at index 0x10 (16 in decimal)

Y - File Tell

Gets the current byte index in a file. Valid values are 0 to file size.

Format	Y<SP>n<LF> !00<LF> $ssssssss<LF> !00<LF>	n File Handle 0 through F ssssssss 4 bytes in HEX position in the file
Example	Y<SP>1<LF> !00<LF> $00000003<LF> !00<LF>	The file with handle 1 has the file pointer at index 0x03

D - Delete File or Folder

Deletes a file or a folder. Appending the name with the \ symbol indicated this is a folder.

Format	D<SP>FULL PATH[\]<LF> !00<LF>	Full path: The full path where the FILE is located. [\]: if '\' is added at the end of the full path string, it means we are deleting a FOLDER, not file.
Example	D<SP>M:\TMP\TEST.TXT<LF> !00<LF>	Remove the FILE with name TEST.TXT in TMP folder on SD memory card. This will not delete the TMP folder.
	D<SP>M:\TMP\<LF> !00<LF>	Remove the FOLDER with name TMP in root folder of memory card. The folder must be empty.

? - Find File or Folder

This command searches for a specific file or folder name at the specified path. If the directory exists, ALFAT outputs the file size, attributes and date & time of modification, otherwise, it returns an error indicating that the file is not found.

Format	?<SP>FULL PATH<LF> !00<LF> $ssssssss<LF> $AA<LF> $hh:mm:ss<SP>dd-mm-yyyy<LF> !00<LF>	ssssssss 4 bytes in HEX file size AA 1 byte in HEX file Attributes* hh:mm:ss is last time the file or folder was modified. dd:mm:yyyy is last date the file or folder was modified.
Example	?<SP>M:\TEST.TXT<LF> !00<LF> $00000F34<LF> $20<LF> $12:00:00<SP>01-01-2011<LF> !00<LF>	File has been found and its size is 3892 bytes with no special attributes. $20 indicate this is a file. Last modification time is 12:00:00 date is 1/1/2011

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

7	6	5	4	3	2	1	0
Reserved		Archive	Folder	Volume ID	System	Hidden	Read Only

M - Copy From File to Another

Since ALFAT supports opening more than one file at the same time and it supports three file media, this command comes handy copy data from a file to another, even if the files are located on different media.

Format	M<SP>HANDLE_SRC<SP>INDEX<SP>HANDLE_DES<SP>LENGTH<LF> !00<LF> $xxxx<LF> !00<LF>	HANDLE_SRC: Handle source. INDEX: Copy from this index of handle source. HANDLE_DES: Handle destination. LENGTH: Length of data need to be copied. xxxx: actual size has copied.
Example	M<SP>0<SP>0<SP>1<SP>64<LF> !00<LF> $00000064<LF> !00<LF>	Copy from file handle 0 at index 0 to handle 1 with 0x64 bytes.

A - Rename file

Format	A<SP>FULL PATH>NEW FILE NAME<LF> !00<LF>	FULL PATH: The full path of file. It is also included file name. NEW FILE NAME: The new name file. Just only name of new file, no include full path here.
Example	A<SP>U0:\GHI\ALFAT.TXT>ALFAT001.TXT<LF> !00<LF>	Rename ALFAT.TXT in GHI folder from USB0 to ALFAT001.TXT.

E - Test Media Speed

Format	E<SP>X>ssssssss<LF> !00<LF> $aaaaaaaa<LF> $bbbbbbbb<LF> !00<LF>	X: Drive name, included: M: Memory Card drive U0: USB Flash drive 0 U1: USB Flash drive 1 ssssssss 4 bytes in HEX data size for testing. It should be divide 1024. aaaaaaaa 4 bytes in HEX: total milisecond for writing. bbbbbbbb 4 bytes in HEX: total milisecond for reading. First !00 is sent before starting calculations
Example	E<SP>M:>6400000<LF> !00<LF> $00005D14<LF> $000039FB<LF> !00<LF>	Test write and read on Memory Card for 100MB. It takes 23828 millisecond for writing and 14843 millisecond for reading.

Note: this command may take a few seconds or minutes for calculations to finish depending on the size for testing. It also takes 2 handles for writing and reading so the command is only success if the total handle used is smaller than 14.

Q – Format

Format	Q<SP> CONFIRM FORMAT<SP>X<LF> !00<LF> !00<LF>	X is driver name. Included: M: Memory card. U0: USB flash drive 0 U1: USB flash drive 1 First !00 is sent before formatting Second !00 is sent when done

Note this command may take a few seconds for calculations to finish depending on the media size.

6. Boot Loader and firmware update

6.1. General Description

The boot loader is a software that boots the system up. It verifies and runs the ALFAT firmware. It also gives the hosting system an interface for firmware maintenance.

GHI Electronics regularly maintains ALFAT firmware with improvements and bug fixes. ALFAT boot loader allows the hosting system to update the firmware.

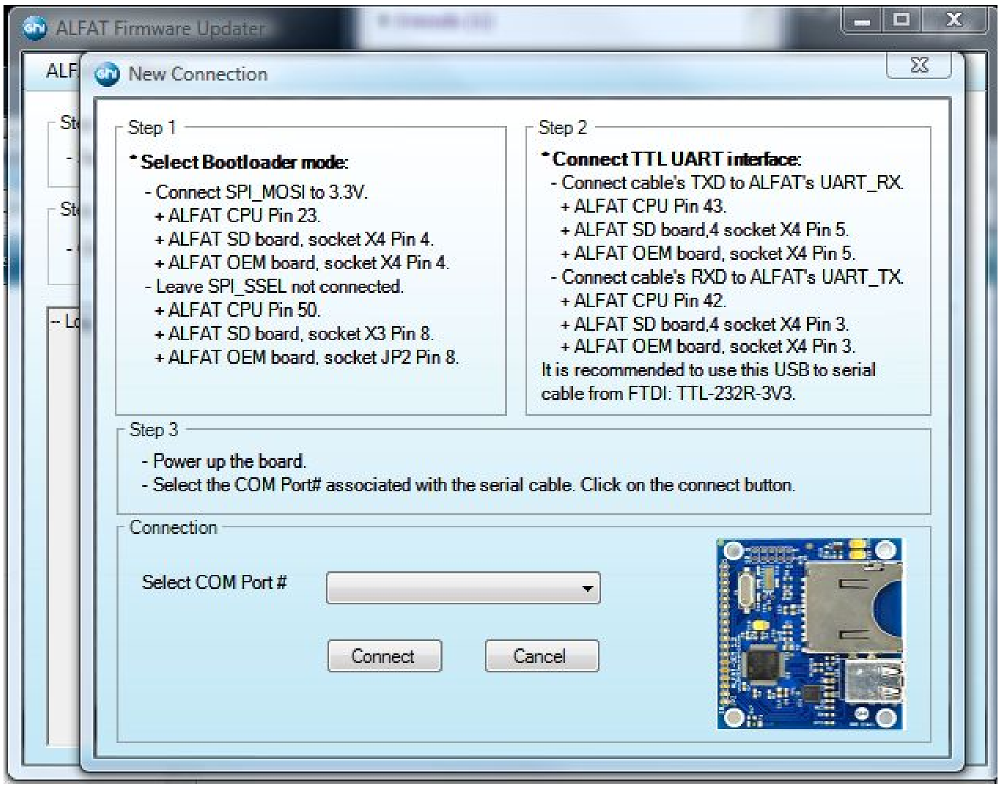
The boot loader can only be accesses through the UART port, without hardware handshaking. So it only requires UART_TX and UART_RX pins.

It is very important that this UART port is exposed on the hardware using ALFAT SoC processor even if the used ALFAT interface with the hosting system is not UART (SPI or I2C). This guarantees that ability to update ALFAT firmware when needed.

6.2. Firmware Updater App

An updater application is provided by GHI to aid in updating the firmware. Users may also wish to implement this functionality right into the host system so an update can happen within the hosting system. The firmware update uses standard XMODEM 1K CRC protocol. To connect ALFAT, or one of the OEM boards, to a PC for update, a TTL serial connection is needed. If using a regular PC serial port then RS232 to TTL level converter is required between ALFAT UART interface and the serial port.

We recommend using USB TTL serial cable. Here is the part number TTL-232R-3V3 from FTDI. This USB enumerated as a virtual serial port and it provide TTL 3.3V levels at the other side that can be connected directly to ALFAT's UART interface.



Step 1

* **Select Bootloader mode:**
  - Connect SPI_MOSI to 3.3V.
    + ALFAT CPU Pin 23.
    + ALFAT SD board, socket X4 Pin 4.
    + ALFAT OEM board, socket X4 Pin 4.
  - Leave SPI_SSEL not connected.
    + ALFAT CPU Pin 50.
    + ALFAT SD board, socket X3 Pin 8.
    + ALFAT OEM board, socket JP2 Pin 8.

Step 2

* **Connect TTL UART interface:**
  - Connect cable's TXD to ALFAT's UART_RX.
    + ALFAT CPU Pin 43.
    + ALFAT SD board,4 socket X4 Pin 5.
    + ALFAT OEM board, socket X4 Pin 5.
  - Connect cable's RXD to ALFAT's UART_TX.
    + ALFAT CPU Pin 42.
    + ALFAT SD board,4 socket X4 Pin 3.
    + ALFAT OEM board, socket X4 Pin 3.

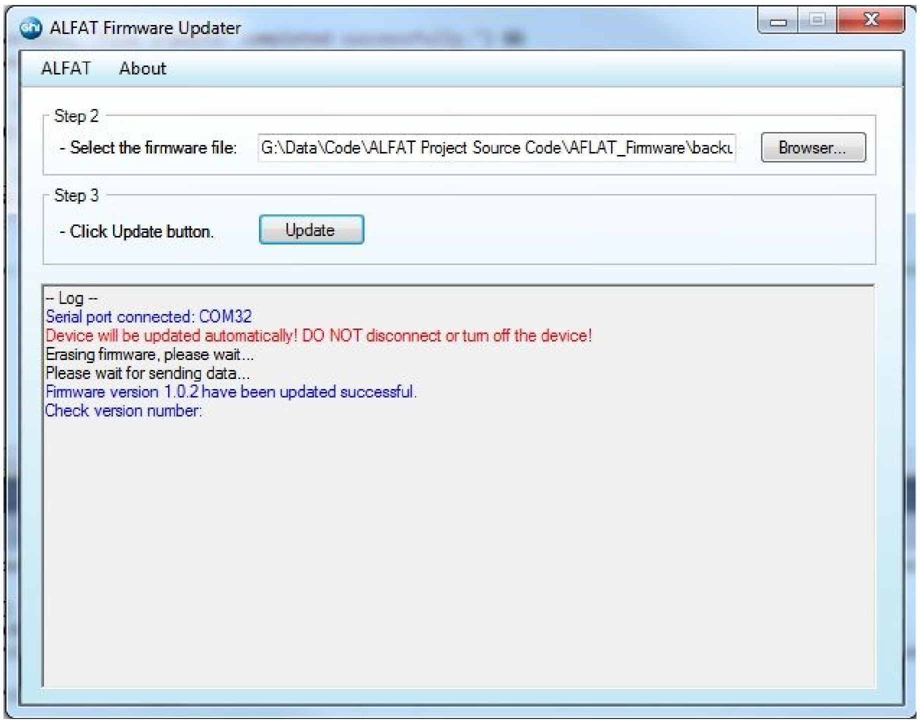
It is recommended to use this USB to serial cable from FTDI: TTL-232R-3V3.

Step 3

- Power up the board.
- Select the COM Port# associated with the serial cable. Click on the connect button.

Connection

Select COM Port #



Step 2

- Select the firmware file: G:\Data\Code\ALFAT Project Source Code\AFLAT_Firmware\backu

Step 3

- Click Update button.

```
-- Log --
Serial port connected: COM32
Device will be updated automatically! DO NOT disconnect or turn off the device!
Erasing firmware, please wait...
Please wait for sending data...
Firmware version 1.0.2 have been updated successful.
Check version number:
```

6.3. Boot Loader Commands

Command	Description
R	Run ALFAT firmware
E	Erase ALFAT firmware
X	Update ALFAT firmware (the firmware file is transferred using XMODEM 1K)
V	Returns the loader version and current ALFAT firmware version.

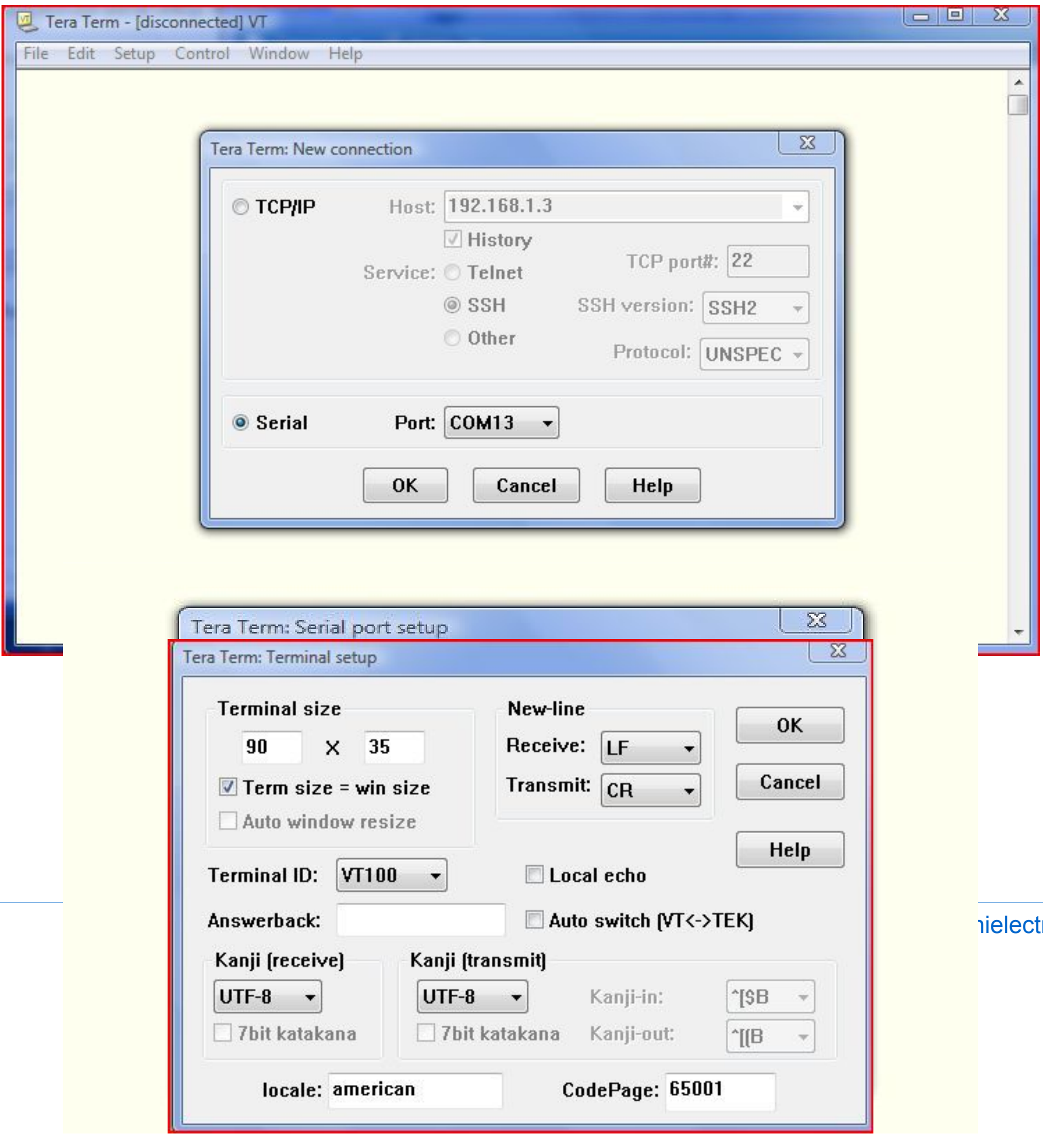
Note: The boot loader is entirely separate program that loads ALFAT SoC firmware. The version number of the boot loader may not match the ALFAT firmware version number. The boot loader can't be updated.

6.4. Updating the firmware using a terminal console

It is recommended to use the provided firmware updater application. But here is an example how to use a terminal console like TeraTerm to update the firmware instead.

Put ALFAT in the bootloader mode as explained in 4.1.Selecting the access interface

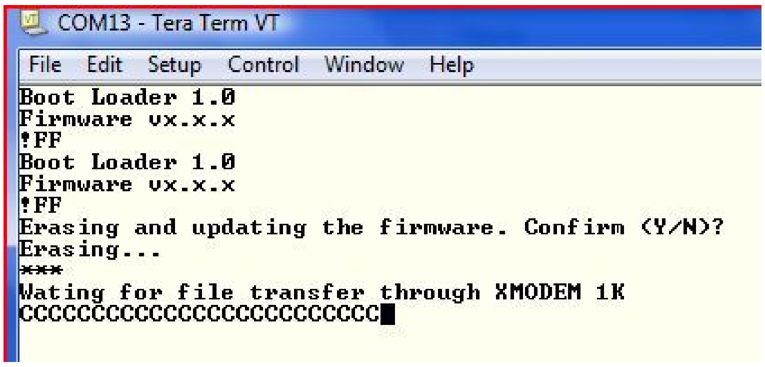
Open the relative COM port and set the baud rate to 115200 and set the New-line receive to LF.


Tera Term - [disconnected] VT
File Edit Setup Control Window Help
Tera Term: New connection
OK
Cancel
Help
Tera Term: Serial port setup
Tera Term: Terminal setup
OK
Cancel
Help

TCP/IP Host: 192.168.1.3
[✓] History
Service: Telnet TCP port#: 22
(●) SSH SSH version: SSH2
Other Protocol: UNSPEC
(●) Serial Port: COM13

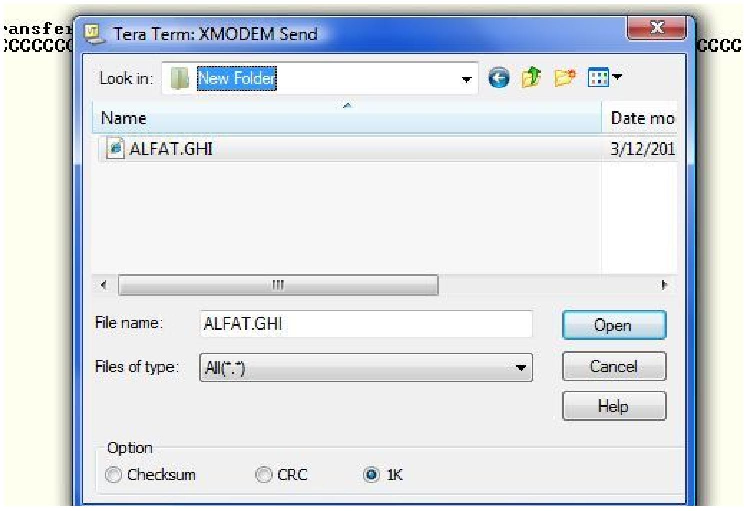
Terminal size
90 X 35
[✓] Term size = win size
[ ] Auto window resize
New-line
Receive: LF
Transmit: CR
Terminal ID: VT100
[ ] Local echo
Answerback:
[ ] Auto switch (VT<->TEK)
Kanji (receive): UTF-8
[ ] 7bit katakana
Kanji (transmit): UTF-8
[ ] 7bit katakana
Kanji-in: ^[$B
Kanji-out: ^[(B
locale: american
CodePage: 65001

Use X command and follow the instructions



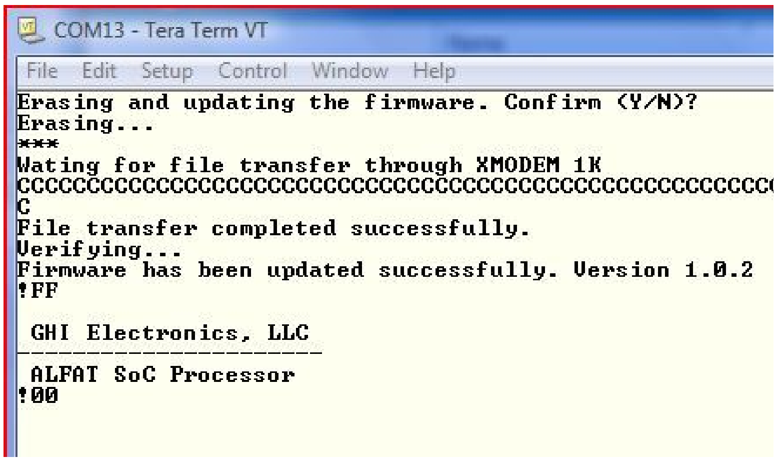
```
Boot Loader 1.0
Firmware vx.x.x
!FF
Boot Loader 1.0
Firmware vx.x.x
!FF
Erasing and updating the firmware. Confirm (Y/N)?
Erasing...
***
Wating for file transfer through XMODEM 1K
CCCCCCCCCCCCCCCCCCCCCCCCC
```

Choose to send a file using XMODEM with 1K option. and choose ALFAT.GHI (the firmware file downloaded from ALFAT page.



ALFAT.GHI 3/12/201
File name: ALFAT.GHI
Files of type: All(*.*)
Option: Checksum CRC [✓ 1K]

After the file being transferred. You will get a message like this "Firmware has been updated successfully. Version 1.0.2." After that you can release the boot loader mode and reset the chip or run R command to run the firmware.



```
Erasing and updating the firmware. Confirm (Y/N)?
Erasing...
***
Wating for file transfer through XMODEM 1K
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
File transfer completed successfully.
Verifying...
Firmware has been updated successfully. Version 1.0.2
!FF

 GHI Electronics, LLC
------------------------
 ALFAT SoC Processor
!00
```

7. Hardware integration guide

Power source is a big source of problems as well. ALFAT is capable of running at lower voltage or somewhat noisy voltage source. The media may or may not work on bad voltages. Make sure the voltage source to the media is reliable and there is large enough capacitor as close as possible to the media connector. We recommend adding 0.1uF and 22uF.

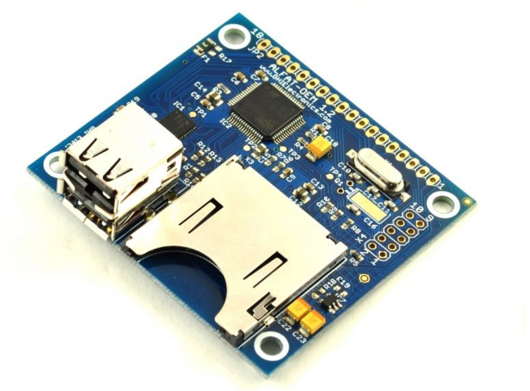
To be continued ...

8. ALFAT OEM Circuit Boards

GHI Electronics offers off-the-shelf OEM boards that uses ALFAT SoC processor. These boards expose all needed signals to interface with ALFAT over UART, SPI or I2C and provide convenient connectors like SD or USB connectors. The boards are easily mountable on existing or new product.

8.1. ALFAT OEM Board

ALFAT OEM Board is an OEM board that exposes all of ALFAT™ SoC processor features. This board offers a seamless way to access files on SDHC, SD and MMC cards plus on USB MSC devices such as tumb drives.



ALFAT OEM includes:

1. SD/MMC Connector with push spring.
2. Dual USB connector Type A with:
    - Full speed USB 2.0 port.
    - High speed USB2.0 port.

All required signals are exposed through a 1x18 pin-mount. Also, if the desired interface is UART, all signals required are exposed through a secondary 2x5 pin-mount.

ALFAT OEM Pinout

1x18 pin-mount

Pin	Name	Pin	Name
1	UART_TX	10	ACTIVE
2	UART_RX/SPI_BUSY/ I2C_BUSY	11	Reserved
3	I2C_SCL	12	VBAT
4	I2C_SDA	13	Internal 3.3V (Do not connect)
5	SPI_SCK	14	$\overline{\text{RESET}}$ (not 5V tolerant)
6	SPI_MISO/UART_BUSY	15	GND

Pin	Name	Pin	Name
7	SPI_MOSI	16	Not connected
8	SPI_SSEL	17	Not connected
9	WAKE	18	5 Volts

2x5 pin-mount

Pin	Name	Pin	Name
1	Internal 3.3V (Do not connect)	2	5V
3	UART_TX	4	SPI_MOSI
5	UART_RX/SPI_BUSY/I2C_BUSY	6	SPI_MISO/UART_BUSY
7	Reserved	8	VBAT
9	GND	10	$\overline{\text{RESET}}$ (not 5V tolerant)

8.2. ALFAT SD Board

ALFAT SD board is an OEM board uses ALFAT™ SoC processor. This board offers a seamless way to access files on SD, SDHC and MMC cards with ALFAT SoC processor. The board include a standard SD/MMC connecter that includes a push spring.

All required signals are exposed through a 1x16 pin-mount. Also, if the desired interface is UART, all signals required are exposed through a secondary 2x5 pin-mount.

ALFAT SD Pin-out

1x16 pin-mount

Pin	Name	Pin	Name
1	UART_TX	9	WAKEUP
2	UART_RX/SPI_BUSY/I2C_BUSY	10	ACTIVE

Pin	Name	Pin	Name
3	I2C_SCL	11	Reserved
4	I2C_SDA	12	VBAT
5	SPI_SCK	13	3.3V
6	SPI_MISO/UART_BUSY	14	$\overline{\text{RESET}}$ (not 5V tolerant)
7	SPI_MOSI	15	GND
8	SPI_SSEL	16	Not connected

2x5 pin-mount

Pin	Name	Pin	Name
1	3.3V	2	Not Connected
3	UART_TX	4	SPI_MOSI
5	UART_RX/SPI_BUSY/I2C_BUSY	6	SPI_MISO/UART_BUSY
7	Reserved	8	VBAT
9	GND	10	$\overline{\text{RESET}}$ (not 5V tolerant)

8.3. ALFAT-USB FS and HS Boards

Coming soon!

9. Conditions of use and Performance

9.1. Selecting a the Right Storage Media

Current storage media market is flooded with low grade devices. These devices may work on a PC but that doesn't mean the device was tested to follow the standards. Also, other devices may have advanced features not suitable for embedded devices. For example, some USB memory drives have a built in USB hub. We made our best to support a wide range of storage medias that follows the standards. But GHI Electronics does not guarantee that ALFAT will be able to access all storage media, especial if it one of the medias mentioned earlier.

For products that integrates ALFAT SoC process, it is important to test different media devices and offer a range of tested devices to end user. GHI Electronics doesn't recommend any specific brand but always recommends selecting a well known source.

The failure will happen at initialization, not at file read and write. If a media mounts then it is safe to assume it will function with no later issues.

As discussed in the integration section, Power source is a big source of problems as well. ALFAT is capable of running at lower voltage or somewhat noisy voltage source. The media may or may not work on bad voltages. Make sure the voltage source to the media is reliable and there is large enough capacitor as close as possible to the media connector. We recommend adding 0.1uF and 22uF.

Should a specific device fails to mount but this device is required, please contact GHI Electronics with details on the failed device. GHI's labs can test the media to determine if supporting this device is possible through a firmware improvement. a fee may be requested for this service.

9.2. File Access Speed

There are many factors that effect the file access speed. Some storage media devices have internal buffering for example. Others have high speed rating.

But even on the exact same media, speeds can be different through different tests. Here are some factors that affect the speed on the same media: fragmentation, media life and voltage.

Fragmented storage runs slower because the system needs to spend more time, or even read more sectors from the FAT table, to find the needed cluster. Formatting the media should take care of this fragmentation. Also, storage access speed decreases when the storage get closer to the end of life. The time needed to erase sectors increases while the device is maturing. Some sectors may even start failing at some point which causes more delays. Finally, while some devices can run on a range of voltages, the lower the voltage the slower the device usually run. nosy power source can cause errors while accessing the media that slow down the speed.

9.3. Serial Interface Speed Overhead

The actual speed of a specific media can be easily determined using the E command. This is the internal speed with complete FAT file system overhead. Such speeds are achieved when using the copy command since no serial connection is involved in transferring data. In an actual product, one of the serial interfaces will be used. Those serial interfaces have some command/response over head and also there is a maximum clock that can be used. The following benchmarks show the different results. These numbers will always be different as explained earlier.

… TBA.

10. Error Codes

Error (HEX)	Description
0x00	Command successful.
0x01	Unknown command.
0x02	Incorrect parameters.
0x03	Operation failed.
0x04	Reached the end of the file/folder list. This is not an error.
0x10	Media does not initialize.
0x11	Initialize media failed.
0x20	File/folder doesn't exist.
0x21	Failed to open the file.
0x22	Seek only runs on files open for read.
0x23	Seek value can only be within the file size.
0x24	File name can't be zero.
0x25	File name has forbidden character.
0x26	File/folder name already exists.
0x30	Invalid handle.
0x31	Handle source does not open.
0x32	Handle destination does not open.
0x33	Handle source requires file open for read mode..
0x34	Handle destination requires file open for write or append mode.
0x35	No more handle available.
0x36	Handle does not open.
0x37	Handle is already in use.
0x38	Open file mode invalid.

Error (HEX)	Description
0x39	Handle requires write or append mode.
0x3A	Handle requires read mode.
0xFF	Boot Loader indication code.

DISCLAIMER