# ALFAT SoC Processor

Man. Rev. 2.14 Date: January 7, 2015 User Manual

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


| 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 | Covers Version 2 Releases |

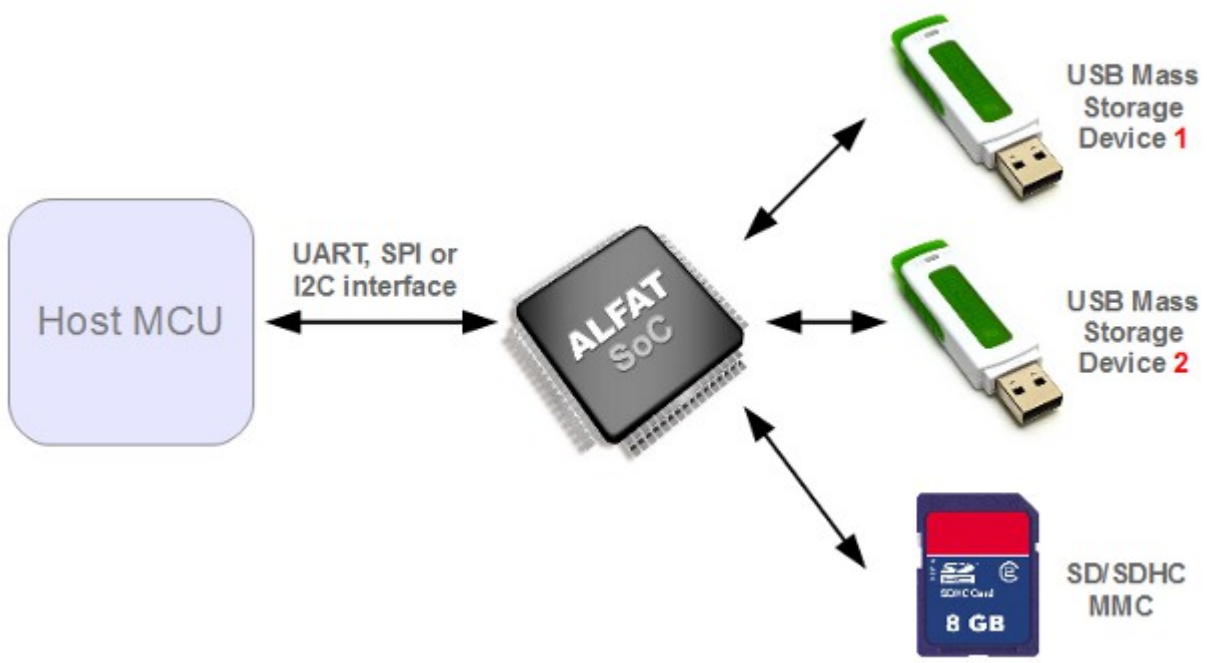
| Document Revision History | | |
|---|---|---|
| **Rev No.** | **Date** | **Modification** |
| Rev 2.14 | 01/07/15 | Simpler banner pseudo code |
| Rev 2.13 | 12/03/14 | Flow chart, cross-references fixed, OEM Module pinout clarification,... |
| Rev 2.12 | 11/03/14 | Misc. changes |
| Rev 2.11 | 08/18/14 | Micro seconds (Forum 16434) |
| Rev 2.10 | 07/21/14 | Micro seconds (Forum 16205) |
| Rev 2.09 | 06/18/14 | SPI max speed, resisters → resistors (Forum 15978) |
| Rev 2.08 | 06/18/14 | Fixed error in write command syntax description |
| Rev 2.07 | 06/02/14 | Added need for initialization to free size command (K) |
| Rev 2.06 | 05/20/14 | Moved some SPI interface sections to proper position |
| Rev 2.05 | 05/08/14 | Added ALFAT-EVAL / Evaluation Kit |
| Rev 2.04 | 04/30/14 | Changed wording in description of Z 2>1 example |
| Rev 2.03 | 04/03/14 | VCAPs from 22uF to 2.2uF |
| Rev 2.02 | 04/03/14 | ALFAT-SDR |
| Rev 2.01 | 03/25/14 | More work on SPI interface |
| Rev 2.00 | 03/10/14 | Preliminary document (for Beta firmware 2.0.0) |

# Table of Contents

# 1 Introduction

## 1.1. ALFAT System-on-a-Chip (Soc) Overview

Creating embedded systems that access FAT file-systems require precious resources to host the needed libraries; developers need to spend significant amount of time writing and testing the libraries. Add to that the hardware costs, both physical parts and circuit complexity, to interface the FAT file-system library with the storage media. In the end, investment costs of product development becomes prohibitive. With the ALFAT SoC, any system can quickly and easily access files on SD cards and USB memory devices. ALFAT uses simple serial commands over UART, SPI or I2C to manage and access FAT formatted SD Cards and USB storage devices. The ALFAT SoC processor is capable of accessing two USB mass storage devices and one SD memory card simultaneously. ALFAT's USB driver also supports USB Keyboards.




ALFAT includes two modes of operation, default and SD-Reader modes. They are covered in detail in the Operating Modes section.

ALFAT developers can take advantage of the “ALFAT Evaluation Kit,” a 100% complete set of components for investigating ALFAT. The Kit includes a number of ALFAT Off-the-shelf Circuit Boards.The included software, “ALFAT Explorer”, controls the working ALFAT directly from a Windows PC. Source code of the “ALFAT Explorer” is shipped with the Kit. Within ten minutes, the hardware can be assembled into a fully functional ALFAT.

## 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).
- Friendly user interface through UART,SPI or I2C.
- Programmable UART baud-rate.
- Up to 16 simultaneous file accesses.
- SD/SDHC card support, no 2GB limit.
- MMC card support.
- SD-Reader mode – a USB connection to a PC where the PC detects ALFAT as a USB Card reader.
- support for USB HID Keyboard Clients
- Built-in 2x USB 2.0 FS PHY USB, 12mbps.
- One USB port is capable of High-Speed 480mbps through external ULPI PHY.
- High speed 4-bit SD card interface.
- Single Pin trigger for automatic/emergency flush and close of all open files.
- Up to 4000 KBytes/sec file internal access speed on SD cards.
- Up to 4000 KBytes/sec file internal access speed on USB High-Speed.
- Up to 1000 KBytes/sec file internal access speed on USB Full-Speed.
- 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, 38 mA fully operational and 5 mA in hibernate.
- -40˚C to +85˚C operational temperature.
- RoHS Compliant/Lead free.

# 2 Terminology

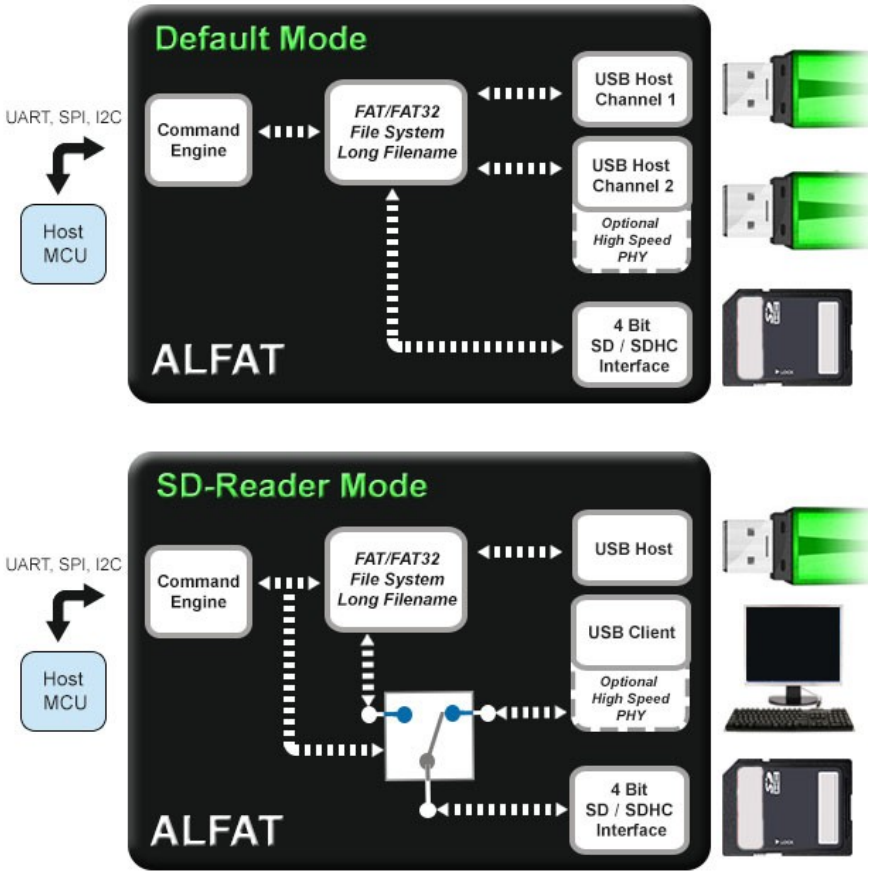
In this manual, any data written to ALFAT from the Master are colored **black;** data returned by ALFAT are colored **red**.

Definition of words and phrases used throughout this manual.

| Word | Meaning, Synonyms |
|---|---|
| Master | The micro-controller (MCU) or other device/system that communicates with, and controls, ALFAT. This could be a PC connected to ALFAT via a UART based cable or USB to Serial cable (useful for debugging and experimenting). |
| USB1 | ALFAT's USB port 1. By itself, this does not define whether that port is configured as a Host or Client. Also for USB2. |
| Command | A printable ASCII string signifying a request from Master to ALFAT; the first character of the Command is followed by one or more characters representing arguments. |
| <LF> | A place holder for the byte 0x10. On many compilation systems this is identified as the character '\n'. |
| <SP> | A place holder for the byte 0x20 (the space character). |
| **Result-string** | Any data returned to the Master in response to a Command will be followed by a 4 character Result-string that provides an indication of success or failure. The Result-*string* includes a Result-*code*. An example Result-string “!01<LF>”; the Result-code is “01” |
| **Result-code** | The two characters following the exclamation mark in the Result-string. “00” indicates success; any other value is either an error number or informational. Result-codes are listed at the end of the manual. |

# 3 Operating Modes

ALFAT includes two modes of operation. Developers need to decide which mode suits their needs and connect ALFAT accordingly. The following sections highlight the differences.




## 3.1. Default Mode

In this mode, ALFAT accepts two USB Clients and an SD card. A high level serial interface is used by the Master to manage (open, read, write, delete,... ) files and directories on the media. ALFAT handles all the tedious low-level details; such as implementing drivers for the USB and SD-Card ports, reading/writing directory tables, buffering and flushing file contents, Etc. The Master's I/O interface becomes a minor piece of development; allowing programmers to concentrate on other aspects of their devices.

## 3.2. SD-Reader Mode

With a single Command from the Master, ALFAT enters SD-Reader Mode – Where one of the USB ports is configured as a USB Client. The SD/SDHC interface is virtually connected to the USB Client port; simulating an SD card reader. This allows a PC to access the files on the SD card right through ALFAT. The Master can command ALFAT to virtually disconnect the SD card from the USB Client interface, allowing file access through commands, similar to the default mode.

This mode gives developers the flexibility of managing data on SD cards, internally and externally. An example can be a data logger device where this device run stand alone; saving data to the internal SD card. Data is collected by the master, written ALFAT; storing it on the SD card. At some point, to obtain the logged data, the logger is connected to a PC. The Master detects the PC connection through ALFAT and switches the SD to be virtually connected to the USB Client, simulating an SD reader.

Once the file transfer is complete, the Master can command ALFAT to virtually disconnect the SD card from the USB Client interface and then to continue logging data.

# 4 USB Keyboard Access

There are projects that would benefit from the ability to support USB keyboards. ALFAT is mainly designed and optimized for file system needs; however, ALFAT can also be used to read keystrokes on USB keyboards.

Details on how to read USB keyboards are covered in the “command” section, namely the Get Free Size command (K) and the Read from File command (**R**).

# 5 Architecture

ALFAT SoC's processor is an ARM Cortex-M3 that runs a robust file-system engine with SD and USB Host/Client drivers. The processor accesses storage media though its 4-bit SD interface and two USB Host 2.0 interfaces. One of the USB ports is also capable of running at High-Speed 480 Mbs with an external ULPI HS PHY chip.

Methods for interfacing ALFAT with the Master are flexible; a standard bus – UART, SPI, or I2C -- can be used; whatever fits the project best.

## 5.1. Commands

All communications between the Master and ALFAT use a well defined serial protocol; which is compact and easy to implement. The Master writes commands to ALFAT and reads data. ALFAT follows every Command with a Result-code for success, error type, or helpful information.

Other than the raw data read or written to files, Commands and their results are in human-readable ASCII. This allows for easier development and troubleshooting.

The simplicity of commands can be seen with this example to flush data written to a file: the file was opened with an associated file-handle of 3, the Master sends ALFAT the command:

**F 3**<LF>

ALFAT returns the string:

!00<LF>

the Result-code of “00” indicates the file was flushed with no error. During development this entire command sequence could be executed from a terminal program running on a PC.

## 5.2. FAT File-System Engine

The file-system engine interprets media formatted to FAT file-system standards. It has been optimized for high performance and reliability.

Here are some of the capabilities of this engine:

- supports FAT16, FAT32 standards
- Long File-name support (LFN).
- Simultaneous access to 16 opened files. There are no limits on how many files can be opened and closed.
- full directory (folder) support.
- some examples of supported I/O functions include read, write, append, seek, tell, find, delete
- No limits on media size, file size or file/folder count (other than those mandated by the FAT standard).

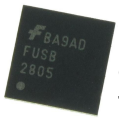
## 5.3. Memory Card Access (SDHC, SD or MMC)


ALFAT processors include a memory card driver 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/MSD cards or multimedia cards. There is no limit on the card capacity.

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

## 5.4. USB Host Connections

ALFAT is capable of accessing FAT file-system files on USB mass storage clients . The hardware provides two standard Full-Speed[1] 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[2] mode by adding a ULPI High-Speed+ PHY chips like FUSB2805. This option quadruples the file-system access speed.

[1]Full-Speed USB 2.0 is 12mbps.

[2]High-Speed USB 2.0 is 480mbps.

## 5.5. Bootloader

The bootloader is a small application that

- initializes the ALFAT processor,
- it verifies and launches the ALFAT firmware, and
- it gives the Master an interface for firmware updates.

The bootloader can only be accessed through the UART port and it uses XMODEM 1K to transfer the firmware file to ALFAT. The The Bootloader chapter explains this in greater detail.

## 5.6. Package and Pin-Out

The ALFAT SoC package is standard 10x10mm LQFP64. The following Table is a description of ALFAT's pins.

| Pin | Name | Description |
|---|---|---|
| 1 | VBAT | Power source for the internal Real Time Clock (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. VBAT works from 1.65V to 3.6V. If RTC is not needed, connect to VCC |
| 2 | NC | Should not be connected |
| 3 | OSC32_IN | Pin 1 for the 32.768 kHz crystal, for the RTC. Optional |
| 4 | OSC32_OUT | Pin 2 for the 32.768 kHz crystal, for the RTC. Optional |
| 5 | OSC_IN | Pin 1 for parallel-cut 12 MHz system crystal. 10pF to 20pF is recommended |
| 6 | OSC_OUT | Pin 2 for 12 MHz system crystal |
| 7 | RESET | Reset signal. Active Low.<br>THIS PIN IS NOT 5 VOLT TOLERANT |
| 8 | USB1_ULPI_STP | USB High-Speed PHY signal. Do not connect if PHY is not used |
| 9 | NC | Should not be connected |
| 10 | USB1_ULPI_DIR | USB High-Speed PHY signal. Do not connect if PHY is not used |
| 11 | USB1_ULPI_NXT | USB High-Speed PHY signal. Do not connect if PHY is not used |
| 12 | VSSA | Ground |
| 13 | VDDA | 3.3V Power source |
| 14 | WAKEUP/EFC | Controlled by the **Z** command. By default a wake-up signal; alternatively use as emergency flush and close (EFC). By default this pin is internally pulled low. Signal is Low to High. |
| 15 | NC | Should not be connected |
| 16 | NC | Should not be connected |
| 17 | USB1_ULPI_D0 | USB High-Speed PHY signal. Do not connect if PHY is not used |
| 18 | VSS1 | Ground |
| 19 | VDD1 | Power, 3.3V |

| Pin | Name | Description |
|---|---|---|
| 20 | DATAREADY | Indicates the status of the buffer used to hold data that may be read by the Master . When high, the buffer is contains data that may be read. |
| 21 | USB1_ULPI_CK | USB High-Speed PHY signal. Not connected if PHY is not used |
| 22 | SPI_MISO/UART_BUSY | MISO when using SPI interface; otherwise, BUSY pin for UART interface. 5 V tolerant. |
| 23 | SPI_MOSI | MOSI SPI interface, 5V tolerant. By default, this pin is internally pulled down. Note, this pin is read during ALFAT bootup to select communication interface type. |
| 24 | WP | SD Write protection signal input. Low = card not protected. Connect to ground if Write protection signal is not used in the hardware design. |
| 25 | CD60 | Memory Card Detect signal input. Low = card detected. Connect to ground if Card Detect signal is not used in the hardware design. |
| 26 | USB1_ULPI_D1 | USB High-Speed PHY signal. Do not connect if PHY is not used |
| 27 | USB1_ULPI_D2 | USB High-Speed PHY signal. Do not connect if PHY is not used |
| 28 | BOOT1 | Add a 10K resistor to ground. |
| 29 | USB1_ULPI_D3 | USB High-Speed PHY signal. Do not connect if PHY is not used |
| 30 | USB1_ULPI_D4 | USB High-Speed PHY signal. Do not connect if PHY is not used |
| 31 | VCAP_1 | Connect to a 2.2uF to ground |
| 32 | VDD1 | Power, 3.3V |
| 33 | USB1_ULPI_D5 | USB High-Speed PHY signal. Do not connect if PHY is not used |
| 34 | USB1_ULPI_D6 | USB High-Speed PHY signal. Do not connect if PHY is not used |
| 35 | USB1_DM | Data Minus, USB port 1. NC if HS PHY is used! If FS is used, add 22ohm resistor in series |
| 36 | USB1_DP | Data Plus, USB port 1. NC if HS PHY is used! If FS is used, add 22ohm resistor in series |
| 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 |

| Pin | Name | Description |
|---|---|---|
| 40 | SD_D1 | D1, 4-bit SD Bus. Add a 47K resistor between this pin and VDD |
| 41 | ULPI_19_2MHZ | This pin generates 19.2 MHz clock. It is usually used to clock the USB High-Speed PHY |
| 42 | UART_TX/U1_CONNECT | UART transmit bus line, 5V tolerant. If the SD-Reader Mode is initialized ("I<SP>D:[H]<LF>"), and communication to ALFAT is I2C, this pin gives the connection status of the USB1 device to the PC. |
| 43 | UART_RX/SPI_BUSY/I2C_BUSY | UART bus receive line. 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 |
| 47 | VCAP_2 | Connect to a 2.2uF to ground |
| 48 | VDD3 | Power, 3.3V |
| 49 | NC | Should not be connected |
| 50 | SPI_SSEL | SSEL SPI interface, 5V tolerant. By default, this pin is internally pulled down. Note, this pin is read during ALFAT bootup to select communication interface type. |
| 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. This pin is internally pulled high. If this pin is pulled low, during reset or power-up, the UART baud rate will be 9600. The default rate is 115200 baud. |
| 56 | NC | Should not be connected |
| 57 | USB1_ULPI_D7 | USB High-Speed PHY signal. Add a 10K resistor to VDD.<br>Only add10K resistor if HS PHY is not used |
| 58 | I2C_SCL/U1_CONNECT | SCL I2C Interface signal. If SD-Reader mode is initialized ("I<SP>**D:[**H]<LF>"), and communication |

| Pin | Name | Description |
|---|---|---|
| | | to ALFAT is UART or SPI, this pin gives the connection status of the USB1 Client to the USB Host[(1)]. |
| 59 | I2C_SDA | SDA I2C Interface |
| 60 | BOOT0 | Should not be connected. It is safe to have pull-up or pull-down resistor on this pin. |
| 61 | NC | Should not be connected |
| 62 | NC | Should not be connected |
| 63 | VSS2 | Ground |
| 64 | VDD4 | Power, 3.3V |

[(1)]When the SD-Reader mode is active, this pin (U1_CONNECT) signals the state of a connection between the USB Client and a PC. See the **I** command for details.

All pins have internal weak pull-up or pull-down resistors, leaving them unconnected is safe.

# 6 Selecting the ALFAT Access Interface. Reset/Booting.

## 6.1. Interface Mode Selection

The Master circuit can be connected to ALFAT using one of the following interfaces:

1. UART,
2. SPI, or
3. I2C.

During initialization (after reset or power-up), ALFAT detects which interface to use by reading SPI_SSEL and SPI_MOSI:

| SPI_SSEL | SPI_MOSI | Interface |
|---|---|---|
| low | low | UART |
| low | high | Bootloader(1) |
| high | low | I2C |
| high | high | SPI |

(1)this interface has two affects: the ALFAT is placed in bootloader mode and also the interface is UART. The bootloader is primarily used for firmware updates and can only be accessed using UART. The Bootloader chapter explains this in greater detail.

SPI_SSEL and SPI_MOSI have internal pull-down resistors.

**Example code for interfacing to ALFAT is available on the GHI Electronics' website; go to the Catalog entry for ALFAT and click on “Resources”.**

## 6.2. Boot/Reset Protocol

1. The Master holds ALFAT's RESET pin low until the power is stable.
2. The Master establishes SPI_SSEL and SPI_MOSI to indicate which bus is chosen (as shown in the table above).
3. After the interface selection pins are set, the RESET pin is set high; allowing ALFAT to boot. **Note that the RESET pin is NOT 5V tolerant.**
4. Master delays 50µs allowing ALFAT to initialize.
5. ALFAT sends the Master a banner string; which must be read, to synchronize with ALFAT.

```
BOOL ClearBanner(int timeout_ms)
{
    while(timeout_ms && ALCAM.ReadByte()!='!')
    {
        delay(1); //1ms
        timeout_ms--;
    }
    while(timeout_ms && ALCAM.ReadByte()!='0')
    {
        delay(1); //1ms
        timeout_ms--;
    }
    while(timeout_ms && ALCAM.ReadByte()!='0')
    {
        delay(1); //1ms
        timeout_ms--;
    }
    while(timeout_ms && ALCAM.ReadByte()!='\n')
    {
        delay(1); //1ms
        timeout_ms--;
    }
    if (timeout_ms>0)
        return TRUE;
    return FALSE;
}
```

## 6.3. UART Interface Mode

The UART interface uses four hardware signals:

1. UART_TX data sent from ALFAT to the Master.
2. UART_RX data sent from the Master to ALFAT.
3. UART_BUSY signal. The Master should only send data when this signal is low.
4. DATAREADY signal. ALFAT has data that can be read by the Master

### 6.3.1 UART Configuration

- The defaults are:
  - baud rate: 115200.
  - Data bits: 8
  - Parity: None
  - Stop bit: 1
- At boot time the SPI_SCK pin is internally pulled high. If this pin is pulled/set low externally during reset or power-up, the UART baud rate will default to 9600 instead of 115200 baud.
- The baud rate can be changed at run-time using the **B** command.

## 6.4. SPI Interface Mode

With the SPI interface ALFAT is a slave device. It uses four hardware signals:

1. SPI_SSEL: ALFAT Chip Select.
2. SPI_MOSI: ALFAT Data in.
3. SPI_MISO: ALFAT Data out.
4. SPI_BUSY: This can be monitored while sending data to ALFAT. When it is high, no more data should be exchanged with ALFAT until it goes low.
5. The maximum baud rate is 3 Mbits/s

### 6.4.1 SPI Bus Configurations

- The maximum SPI clock is 24 MHz.
- SPI clock Idle state is Low.
- Sampling is at the rising edge.
- Data chunks representing numbers greater than one byte are sent most significant byte (MSB) first.
- ALFAT is the slave in the SPI bus.
- SPI_SSEL should be active (low) with transfer, single and multiple bytes are allowed.
- **A minimum of 4µS delay is required between each byte**. This requirement is not needed with Fast Write to File command (L Command) that uses a SPI DMA receive channel.

## 6.4.2 SPI Frames

ALFAT SPI data is exchanged through frames:

- Frames are a stream of bytes.
- Frames are classified as either a *Write* (Master data to ALFAT) or a *Read* (ALFAT data to Master).
- The classification of a frame declares and defines its contents.
- All frames start out with a three byte *header*.
- Frame sizes are primarily based on the designer's needs. In this example, a command is sent to initialize the directory list "@ M:\TEST\T". For demonstration purposes, the command is sent using two frames (one frame or more than two frames will work as well). This is also the case for the Return-string "**!00***<LF>*", where two frames are read; a single frame will work just as well.

SPI Bus frames, in order (F1, F2, F3, F4)

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| F1 | 1 | 08 | 00 | '@' | ' ' | 'M' | ':' | '\' | 'T' | 'E' | 'S' |
| | U | 01 | RFB | RFB | RFB | RFB | RFB | RFB | RFB | RFB | RFB |

| | | | | | | |
|---|---|---|---|---|---|---|
| F2 | 01 | 2 | 00 | 'T' | 0x0A | |
| | U | 01 | RFB | RFB | RFB | |

| | | | | | | |
|---|---|---|---|---|---|---|
| F3 | 02 | 03 | 00 | 0 | 0 | 0 |
| | U | 03 | 00 | '!' | '0' | '0' |

| | | | | |
|---|---|---|---|---|
| F4 | 2 | 03 | 00 | 0 |
| | 02 | 01 | 00 | 0x0A |

RFB - Ready for Byte, U - undefined

### 6.4.2.1. Write Frames

| Frame Type | 16-bit Transaction Size N | | Payload | | | |
|---|---|---|---|---|---|---|
| 0x01 | Size LSB | Size MSB | Byte 1 | Byte 2 | ... | Byte N |
| **Synchronous Return Data** | | | | | | |
| Ignore | RFB | RFB | RFB | RFB | ... | RFB |

RFB - Ready for Byte, Frame Type – 0x1 – Write

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

1. The header consists of three bytes: the first byte is the frame type, the second and the third bytes are the 16-bit transaction size (Not including the header bytes).
2. Payload: This is the data being sent to ALFAT. The payload section size must match the transaction size value from the header.

As bytes are sent, if ALFAT becomes busy it may signal the Master to stop sending bytes. The Master can check for the busy signal in two ways:

1. by checking the SPI_BUSY pin, or
2. checking the SPI response bytes sent by ALFAT ( Ready for Byte (RFB)).

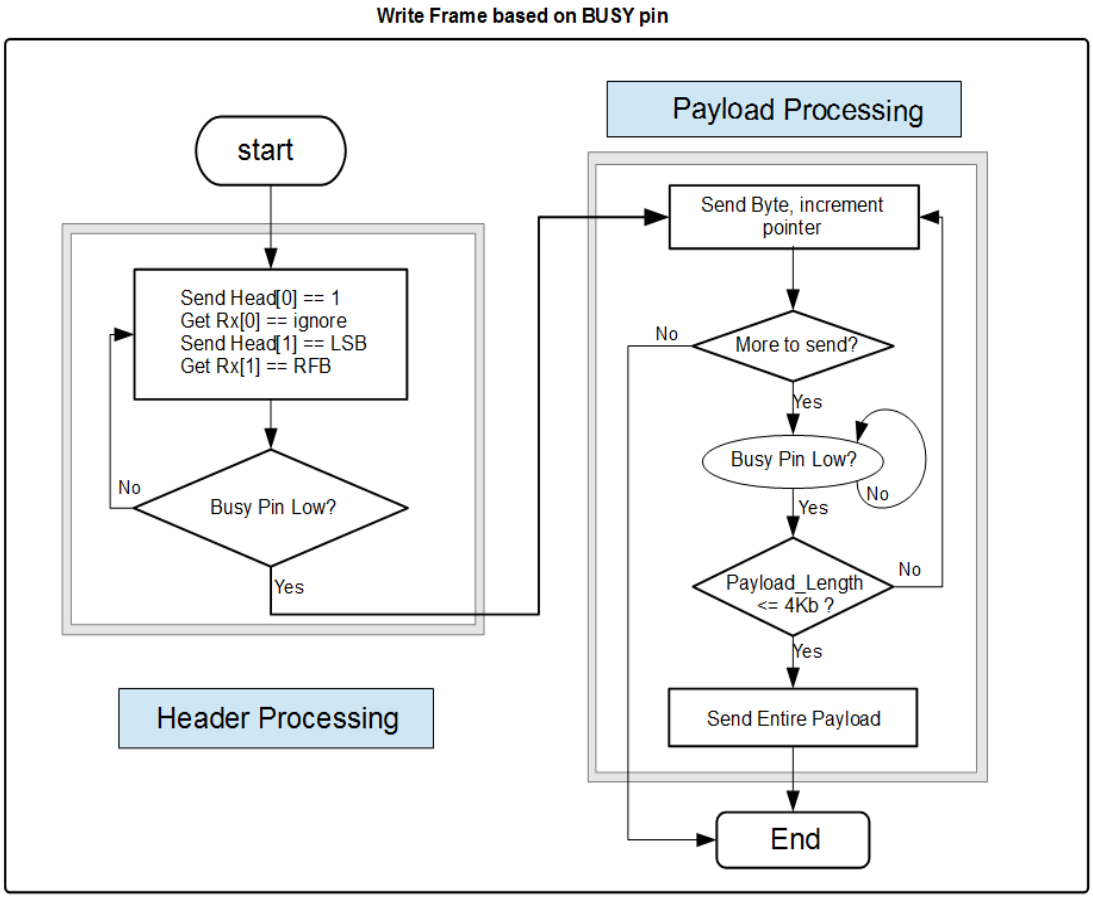
Note: regardless of the method chosen to detect busy, if the frame sent to ALFAT has a payload of 4 KB or less it can be sent with no checking. A significant speedup may be possible if large payload data (those greater than 4Kb) is divided into 4 KB chunks.

### Using SPI_BUSY

This signal is sampled before sending new frames. New frames can be sent if SPI_BUSY is low.

SPI_BUSY must also be checked during payload transmission *after each byte is sent*. If SPI_BUSY goes high:

- *during the header portion of the write*, ALFAT resets its command processor to a *ready* state. The Master must terminate the frame (send no more bytes). Then when SPI_BUSY goes low, the write frame, including header, may be sent again
- *while sending the payload*, SPI_BUSY is sampled after each byte; if it is high, transmitting pauses until it returns to low.



### Detecting Busy With RFB (Ready For Byte)

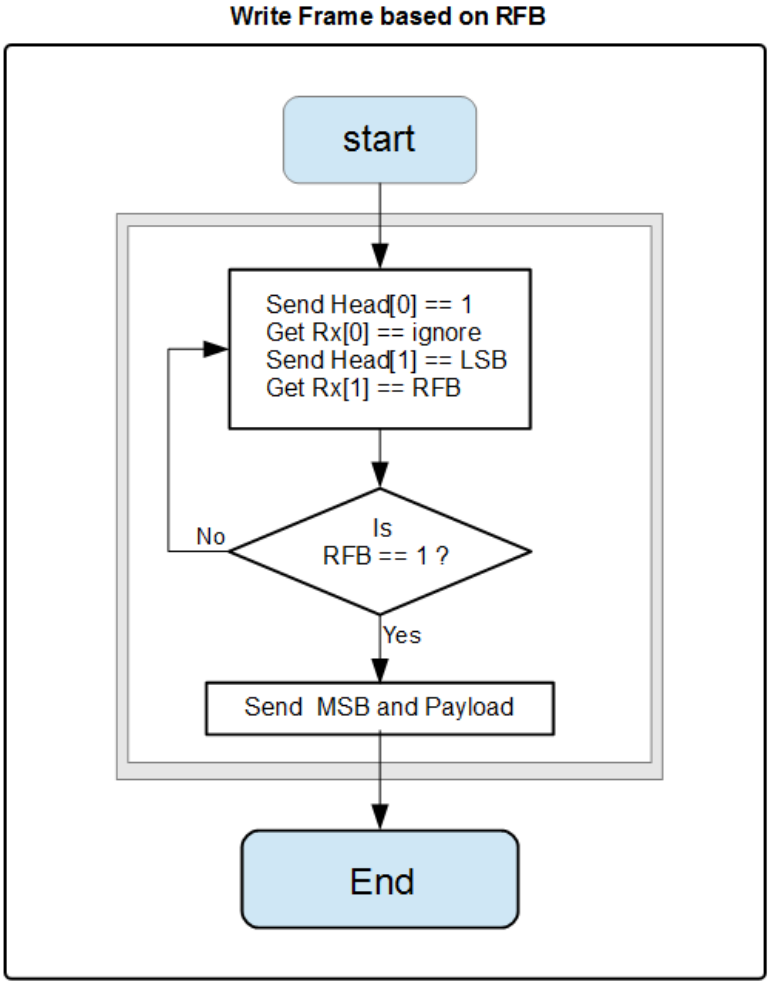
With this method, for every byte transmitted, the Master reads a byte from ALFAT which determines if ALFAT is ready for the next byte. This is the “Ready for Byte” (RFB) flag.

**To use this method, a SPI write frame can not have a payload greater than 4 KB.**

In the hardware layout, SPI_BUSY does not have to be connected to the Master.

During the sending of header bytes:

1. the frame's *Type* byte is written,




2. the LSB byte is sent. If the RFB (corresponding to the LSB) is 0, ALFAT is not ready for a byte, and the entire frame must be restarted. When the RFB is 1, ALFAT is ready; the MSB and the payload may be sent.

### 6.4.2.2. Read-Request Frames

| Frame Type | Requested Transaction 16-bit Size (*R*) | | “Filler Bytes” The length of this byte stream is *A* bytes. | | | |
|---|---|---|---|---|---|---|
| 0x02 | LSB_***R*** | MSB_***R*** | Byte 1 0x00 | Byte 2 0x00 | ... | Byte ***A*** 0x00 |
| **Synchronous Return Data, The “Read-Response ”** | | | | | | |
| Ignore | LSB_***A*** | MSB_***A*** | Byte 1 | Byte 2 | ... | Byte ***A*** |

The “Read-Response” is described in the next section.

Read-Request frames are sent by the Master to retrieve data from ALFAT; they consist of two parts, a header and filler bytes:

1. The header: consists of three bytes:
   i. the frame type (0x2 for Read),
   ii. the least significant byte of the total bytes (LSB_***R***) the Master is requesting from ALFAT
   iii. the most significant byte of the total bytes (MSB_***R***) the Master is requesting from ALFAT

   ***R*** = LSB_***R*** | (MSB_***R*** << 8)
2. Filler bytes: ***A*** bytes with value 0x00. These written by the Master to receive the data from ALFAT. The length of this byte stream is not known until the Master receives the actual size (***A***) from ALFAT.

**Read-Response Data**

| Response to frame type | Number of bytes available in ALFAT buffer | | Payload Stream *A* bytes long | | | |
|---|---|---|---|---|---|---|
| Ignore | LSB_***A*** | MSB_***A*** | Byte 1 | Byte 2 | ... | Byte ***A*** |

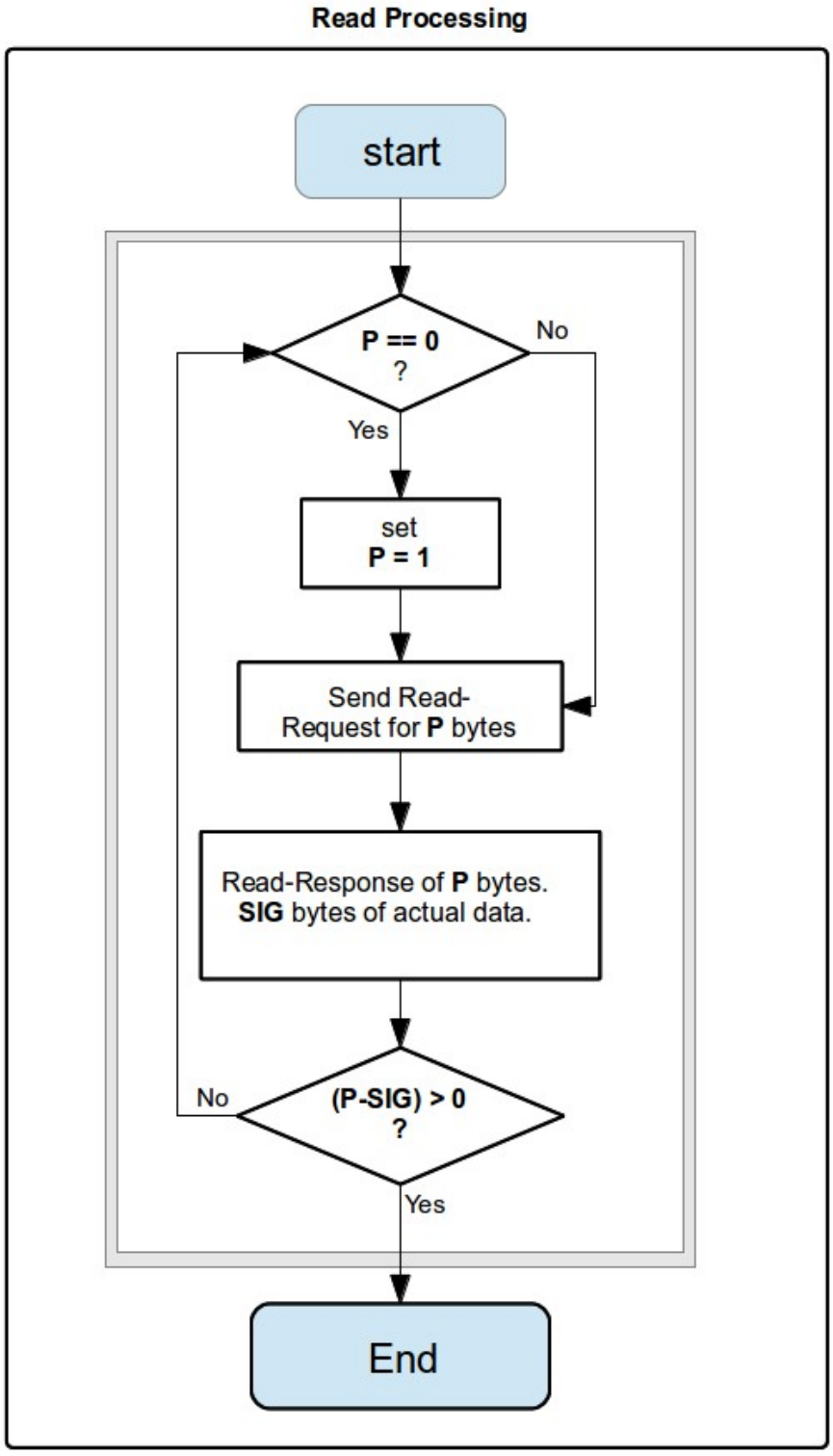
The Read-Response consists of two parts, a header and payload.

1. The header consists of three bytes:
   i. Ignored,
   ii. The least significant byte of ***A***,
   iii. The most significant byte of ***A***.

   ***A*** = LSB_***A*** | (MSB_***ABI*** << 8)

If ***A*** is 0x00, the Master will not send any Filler bytes; the SPI Read-Request frame processing is complete.

If ***A*** is less than ***R***, it does not mean that ALFAT is done with processing the *current Command*. *Commands* are fully discussed in the ALFAT Command Set chapter.


Read Processing
start
P == 0
?
No
Yes
set
P = 1
Send Read-
Request for P bytes
Read-Response of P bytes.
SIG bytes of actual data.
No
(P-SIG) > 0
?
Yes
End

## 6.5. I2C Interface Mode

I2C interface uses three hardware signals:

1. I2C_SCL: I2C clock signal.
2. I2C_SDA: I2C data signal.
3. I2C_BUSY: While sending data to ALFAT if this is high, data transmission should stop until it goes low.

### 6.5.1 I2C Bus Configuration

- ALFAT I2C slave address is 0xA4. This is a fixed address and can not be changed
- Bit zero is the RW bit, 0=Write 1= Read.
- The maximum allowed I2C clock is 400 kHz.
- The circuit must provide pull-up resistors, usually they are 2.2K, on the bus as specified in the I2C specifications.

Transmitting and receiving data to and from ALFAT is preformed through standard I2C transactions.

- Transmitting: ALFAT's I2C-address (with R/W bit set to zero) is sent; followed by the stream of bytes. ALFAT processes the assembled payload bytes sequentially as the command or the command's data (if any).
- Receiving: The Master starts by transmitting ALFAT-I2C-address (with R/W bit set to 1) followed by reading of one or more bytes. When a read request is sent to ALFAT and no data is available, a *No Data* token (0x0) is returned. This presents a problem when raw data (such as file contents) contains 0x00. A two byte sequence is used. When 0x00 is encountered, 0xFF is transmitted before 0x00; 0xFF is an *escape* token. To resolve the same problem of differentiating 0xFF (data byte vs *escape* for 0x0), if 0xFF occurs as data, it is used to *escape* itself. The following table covers these cases:.

| 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:**

- At the I2C level, the number of bytes transmitted may not match the actual size of the file contents. 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. The I2C interface processing code in the Master should remove the escape bytes so higher level application code does not have to be aware of this problem.
- WARNING the R command (read from file-handle) uses a *filler* byte to pad the return stream. For I2C only, the filler byte can not be 0xFF or 0x00.

This pseudo code outlines one possibility of a receive function:

```
UINT8 I2C_GetDataFromALFAT()
{
	UINT8 b;

	do// wait for data. 0x00 = no data
	{
		I2C_SendAddress(0xA4 + READ_BIT);/
		b=I2C_Read();
	} while(b==0);

	if(b==0xFF)
	{
	     I2C_SendAddress(0xA4 + READ_BIT);
		b=I2C_Read();
		if( (b != 0x00) && (b != 0xff))
		{
			Panic("This should never happen");
		}
		// b now equals the actual value, 0x00 or 0xFF
	}
	return b;
}
```

# 7 ALFAT Command Set

## 7.1. Introduction

The commands (written from Master to ALFAT) and responses (data read from ALFAT) intentionally use 8 bit readable/printable ASCII characters. Debugging can be performed using terminal programs running on development machines. Commands are typed in and readable responses come back. When testing, we recommend that any raw data stream associated with commands, as well as any Filler bytes for the Read command (defined below), should be readable digits.

Many examples presented below use the command “Read from File”. The full syntax (defined below) as well as the dialog (data flow between ALFAT and the Master) for the command is:

*Master to ALFAT:* R<SP>{h}{f}>{cccccccc}<LF>

*ALFAT to Master:* !00<LF>

*ALFAT to Master:* {stream of cccccccc bytes}${aaaaaaaa}<LF>

*ALFAT to Master*: ! 00<LF>

The *command* string is always the first item sent to ALFAT; in the above, the *command* is “R<SP>{h}{f}>{cccccccc}<LF>“ where “h” is a file-handle, “f” is the filler byte, “cccccccc” is the requested read size in bytes, and “aaaaaaaa” is the actual number of bytes returned. The definitions of these components are presented as necessary.

## 7.2. Terminology and Syntax of Command Definitions:

Any exceptions to the following are noted in the individual command definitions.

- “<LF>” is the ASCII line-feed byte (value 0x0A). ALFAT will accept the carriage-return (value 0x0D) in place of (0x0A).
- “<SP>” specifies the ASCII value for a space (0x20). ALFAT is white space *sensitive*; extra spaces will cause an error.
- In this document, any characters sent to ALFAT from the Master are colored **black**, and characters by the Master are red.
- All numbers are written as strings of hexadecimal digits. The digits are encoded using ASCII. For example, to send the decimal number 16 to ALFAT, the string would be “10” which is byte value 0x31 (ASCII for 1) followed by 0x30 (ASCII for 0). Hexadecimal digits A to F must be upper case.

- In syntax definitions, substitutable/changeable tokens are enclosed with curly-brackets "{}". The number of digits between the curly-brackets is the number of bytes that should be transmitted. For example: "{ssssssss}" would signify that the token will be transmitted as eight characters. For numeric *values being sent to ALFAT*, leading zeros may be omitted. Numeric values read from ALFAT will always have leading zeros. If the item being read by the Master ALFAT can change then curly-brackets are used for that item. For example, using the Read command and some of its associated dialog:

    1. Syntax of the command:

        R<SP>{h}{f}>{cccccccc}<LF>

        An example of the command string:

        "R 5W>00000002" followed by a line feed. The string "R 5W>2" is equivalent to "R 5W>00000002".

    2. Syntax of returned file contents:

        {stream of cccccccc bytes}${aaaaaaaa}<LF>

        For the Read command above, if the contents of the file associated with handle 5 is the single byte "B", the data string sent back by ALFAT for the file contents is

        "BW$00000001"<LF>

- Optional portions in command syntax are enclosed in square brackets "[ ]". For example, in the command "J[<SP>H]<LF>", "<SP>H" is optional.
- The term "Result-string" is a four byte string *sent from ALFAT to the Master* with the form "!{rr}<LF>". Where the substring "rr" is a "Result-code", either "00" for success or some other value (that is either an error indicator or is informational). For example:

    *Master to ALFAT:* I M:<LF>

    *ALFAT to Master*: !11<LF>

    in the *Result-string*, the *Result-code* "rr" is "11". *Result-codes* are listed in a table at the end of this manual.

- The phrase "command parsed" means that ALFAT has read and evaluated the command string and its arguments. After a command is parsed, ALFAT sends back the Result-string. If ALFAT returns additional data/parameters, they will be followed by a Result-string.
- If the Result-code represents an error, the transaction should be terminated. Examples:

    1. Read command

        *Master to ALFAT:* R 1F>5<LF>

        *ALFAT to Master*: !30-handle

    ALFAT will not send back any bytes for the rest of the dialog. ALFAT will reset its state to *ready and waiting for command*.

2. Write command

   *Master to ALFAT:* W 1>5<LF>

   *ALFAT to Master*: !30

   The Master terminates it's Command processing; in this case, it must not send the data portion of the write command to ALFAT.

- If the command has a return value independent of any data stream, the value will be proceeded by a dollar sign '$'. For example, if the Master issues a read command for 5 bytes from a file opened with handle A, a filler value of ASCII P; and there were only 4 more bytes in the file “ABCD”, then the dialog for a successful read would be:

  HOST → ALFAT: “R AP>5<LF>”

  ALFAT → HOST: “!00<LF>”

  ALFAT → HOST: “ABCDP$00000004<LF>”

  ALFAT → HOST: “!00<LF>”

- ALFAT stores each command in a FIFO buffer. This means that the Master can send a byte stream for one command followed by another. When the buffer is full, ALFAT will signal using BUSY. The return stream's ordering will match the FIFO processing ordering of the commands.
- The total length of an individual command may not exceed 200 bytes.
- ALFAT's storage device ports are identified by “Drive Names”:

  “**M:**” Memory Card drive

  “**U0:**” USB Flash drive 0

  “**U1:**” USB Flash drive 1

  “**K0:**” USB Client attached to USB0 is a Keyboard

  “**K1:**“ UBS Client attached to USB1 is a Keyboard

  “**D:**” refers to a combination of 2 devices for SD-Reader mode

- When a command requires a file-name, it must be *absolute* (*non-relative*) and must include the drive name. For example, if a file named “ghi” exists in the folder “\root\subdirectory” on USB0 then any command referencing “ghi.dat” must use “U0:\root\subdirectory\ghi.dat”.
- A pin protocol for when to send commands or read data is simple. BUSY and DATAREAD can be monitored by the Master and interpreted as:
  - new commands may be sent when BUSY and DATAREADY are both low
  - ALFAT may be read when BUSY is low and DATAREADY is high

Any exceptions to the above are noted in the individual command definitions.

## 7.3. Summary of All Available Commands:

| Command | Description | Command | Description |
|---|---|---|---|
| V | Get version number | I | Initialize Interfaces/Devices |
| Z | ALFAT Control | 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 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 | L | Fast Write to file (SPI only) |
| <LF> | No Operation (NOP) | | |

## 7.4. Full Syntax and Definition of Commands

### 7.4.1 V - Get Version Number

| | | |
|---|---|---|
| **Format** | V<LF><br>v{x}.{x}.{x}<LF><br>!{rr}<LF> | x.x.x - numeric version<br>rr - the Result-code |
| **Example** | V<LF><br>v2.0.0<LF><br>!00<LF> | The version number is 2.0.0 |

Notes:

- The version is not the same or related to the version number of the bootloader.
- The return data is sent before the Result-string that is normally present after command parsing and argument checking.

Prints the version number of the ALFAT SoC firmware.

### 7.4.2 # - Enable Echo

| | | |
|---|---|---|
| **Format** | #<SP>{n}<LF><br>!{rr}<LF> | n = 0 Disable echo<br>n = 1 Enable echo<br>rr - the Result-code |
| **Example** | #<SP>1<LF><br>!00<LF> | Enable echo |

Primarily for debugging enabling echo makes ALFAT send back the data it receives over UART.

Echo is disabled by default.

### 7.4.3 Z - ALFAT Control

| | | |
|---|---|---|
| **Format** | Z<SP>{n}[>{o}]<LF><br>!{rr}<LF> | n: Control mode number<br>o: 0 == Off, 1 == On, values 2...F are reserved<br>rr the Result-code |
| **Example** | Z<SP>0<LF> | Put ALFAT in Standby mode.<br>After waking up from Standby mode, program execution restarts in the same way as after a Reset. |
| **Example2** | Z<SP>1<LF><br>!00<LF> | Put ALFAT in Stop mode.<br>The Result-string will be returned after ALFAT wakes up by setting WAKEUP pin low. |
| **Example3** | Z<SP>2>1<LF><br>!00<LF> | ALFAT uses WAKEUP/EFC as Emergency Flush and Close |

Control mode number

| | |
|---|---|
| 0 | Standby Mode. (On or Off control is not allowed) |
| 1 | Stop Mode. (On or Off control is not allowed) |
| 2 | If On/OFF flag is 1 WAKEUP becomes EFC. If On/Off flag is 0 WAKEUP has its default functionality. If the optional argument is not present “J 2<LF>”, the use of the pin as EFC is disabled. |
| 3...F | Reserved |

This command controls a number of “run level” or “processor states” of ALFAT.

ALFAT supports two low power modes:

1. **Standby Mode:** This mode will completely turn off the processor.
   - Exiting Standby mode: ALFAT exits Standby Mode by resetting the processor (RESET pin) or by a rising edge on WAKEUP pin. After waking up from standby mode, the system is in reset. It is important for users to make sure that all the files are closed before setting this mode.
   - Typical Power Consumption: 3.3 µA TA = 25°C.
   - Maximum Power Consumption: 12.4 µA TA = 25°C. 20.5µA TA = 85°C.
2. **Stop Mode:** With this mode the system will maintain its state but will stop execution.
   - Exiting Stop Mode: ALFAT exits Stop Mode by a rising edge on WAKEUP pin. After waking up from Stop Mode, program execution resumes from where it stopped.
   - Typical Power Consumption: 0.5 mA $T_A$ = 25 °C.
   - Maximum Power Consumption: 1.2 mA $T_A$ = 25 °C. 11 mA $T_A$ = 85 °C.

ALFAT can change the purpose of the power control pin; when the Master is not going to use low-power, it may use WAKEUP as an **Emergency Flush and Close trigger (EFC):**

- it is an interrupt pin that triggers on low to high edge transition.
- when triggered:
  - ALFAT does a graceful internal shutdown, as quickly as possible, by flushing and closing all open files.
  - Any command in-process will be terminated.
  - The internal FIFO buffer will be flushed.
- if enabled, and a Z command for power state control is sent by the Master, then EFC is disabled and the pin is reconfigured for WAKEUP
- after an EFC event/interrupt has occurred, ALFAT *should be* reset. If it is not reset, then if a UART interface is active, !60<LF> will be sent to the Master; for SPI or I2C interfaces, ALFAT will send !60<LF> as the Result-string of the first command sent to it.
- Communication of data in SD-Reader devices is controlled by the PC associated with the SD-Reader. If ALFAT is in SD-Reader Mode, the triggering of EFC will not affect, in any way, the current state of transactions between the USB Host and ALFAT.

*Best practice* suggestions for EFC: use in power-fail circuit or other similar external event, use before a reset when the Master detects internal “can not happen errors” (asserts).

### 7.4.4 T - Initialize Real Time Clock

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

It is important to use this command before setting the time.

### 7.4.5 S - Set Current Time and Date

| | | |
|---|---|---|
| **Format** | S<SP>{ddddtttt}<LF><br>!{rr}<LF> | ddddtttt - time and date 32bit structure(1)<br>rr - the Result-code |
| **Example** | S<SP>34210000<LF><br>!00<LF> | Set 1/1/2006 00:00:00 |

(1)Time and Date structure is a 32-bits standard structure used 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) |

For example, 0x34212002 is 01/01/2006 – 04:00:04

The **I** command should be used before setting the Date and Time.

### 7.4.6 G - Get Current Time and Date

| | | |
|---|---|---|
| **Format** | G<SP>D<LF><br>{MM}-{DD}-{YYYY}<LF><br>!{rr}<LF> | Get current date.<br>rr - the Result-code<br>MM – month (01 – 12)<br>DD – day (01 – 31)<br>YYYY - year |
| | G<SP>T<LF><br>{HH}:{MM}:{SS}<LF><br>!{rr}<LF> | Get current time.<br>HH – hour (01 - 24)<br>MM – minutes<br>SS – seconds |

### 7.4.7 B - Set UART Baud Rate

| | | |
|---|---|---|
| **Format** | B<SP>{ssssssss}<LF><br>!{rr}<LF><br>!{rr}<LF> | ssssssss: The standard baud rate value in HEX(1).<br>rr - the Result-code |
| **Example** | B<SP>1C200<LF><br>!00<LF><br>!00<LF> | Set the baud rate to 115200.<br>← command parsed and is correct<br>← send success after changing rate |

(1)The first Result-string refers to correct parsing of the command, including error checking of the baud rate value. The second Result-string is sent *after* the baud rate has changed.

Under default conditions, the UART interface has a baud rate of 115200, The default value can be changed by pulling SCK_SPI low during booting/reset; in this case, the baud rate will be 9600.

### 7.4.8 I - Initialize and Configure Devices

| | | |
|---|---|---|
| **Format** | I<SP>{X}[H][K]<LF><br>!{rr}<LF> | X - is a Drive name, one of:<br>**M:** Memory Card drive(4)<br>**U0:** USB drive 0 Full-Speed<br>**U1:** USB drive 1 Full-Speed<br>**D:** USB1 SD-Reader Mode (1)<br>**K0:** USB drive 0 as Keyboard Reader(3)<br>**K1:** USB drive 1 as Keyboard Reader(3)<br><br>H – **for "U1:" and "D:" only**, use High-Speed (2)<br><br>K – Keyboard filter mode(3).<br><br>rr - the Result-code |
| **Example** | I<SP>M:<LF><br>!00<LF> | Initialize memory card with the default clock frequency.(4) |
| | I<SP>U0:<LF><br>!00<LF> | Initialize USB0 at Full-Speed mode |
| | I<SP>U1:<LF><br>!00<LF> | Initialize USB1 at Full-Speed mode(2) |
| | I<SP>U1:H<LF><br>!00<LF> | Initialize USB1 at High-Speed mode(2) |
| | I<SP>D:H<LF><br>!00<LF> | Establish High-Speed SD-Reader Mode |

(1) When ALFAT is in SD-Reader Mode, both USB1 and the Memory Card drive are used.

(2) The optional parameter **H** indicates that the USB1 device should be initialized as High-Speed (the default is Full-Speed). **H** is not supported for "**K1:**"

(3) When initializing either USB0 or USB1 as a Keyboard host:

1. an optional argument indicates whether the **R** command (Read) passes back the *raw* (unfiltered) key-code or a filtered code. Filtered key-codes take the raw key-codes and map most of the printable key-codes to their ASCII value. Possible values for K are:
   - A – return filtered (ASCII) characters
   - R – return raw key-codes.
2. For a consistent I/O model, a mapping is provided to refer to the USB keyboard using virtual file-handles (for commands where file-handles are required.
   - Virtual file-handles are implicitly opened by the **I** command.
   - **K0:** is given handle "Z"; **K1:** is assigned handle "Y"
   - Virtual file-handles can be read (get keys) and written (e.g. set LED)
   - Other commands for Keyboard Clients include: **K** - get size, **J** - read register.

(4) When using the memory Card device (**M:**), an optional argument is supported to set the bus speed of the SD Reader (described further below).

This command controls functions related to the hardware interfaces for file-storage devices, such as:

- initializing the hardware and software interfaces to a file-system device and mounting its file-system.
- Setting ALFAT to SD-Reader Mode (Mass Storage Client)
- changing the bus speed of the memory/SD card interface.

If **I** returns "11" or "70" at the Result-code, the program should check for the presence of media with the **J** command before using **I.**

If the **I** command is executed for any device that is currently mounted; that is, it has previously appeared in an **I** command, then all open files on that device are flushed and closed; the device and internal data structures are all re-initialized as though this were the first **I** command for the device.

Example:

**I M:**<LF>
…
{various files are opened on **M:**, some have been written. No files have been closed}
…
**I M:**<LF>
{all files associated with **M:** are flushed and closed, their file-handles are free}
…

This holds true for **D:** (SD-Reader mode). For example (see **J** command below), assuming a card is in **M:** and is write-enabled, **U1:** is not connected, and **U0:** holds no device:

**I D:**<LF>
!00<LF>
J<LF>
!00<LF>
$81<LF>
!00<LF>
...
{a USB Flash Drive is plugged into **U1:** (before the following)}
…
**I U1:**<LF>
!00<LF>
**J**<LF>
$49<LF>
!00<LF>

**Important Notes**

- If ULPI PHY is implemented in the design (which includes the ALFAT OEM board), then USB1 must be initialized with the High-Speed parameter (**H)**. This is true even if the attached USB Client does not support High-Speed mode.
- WP pin (Write Protect Signal) must be connected to ground if it is not used in the hardware design.
- CD pin (Card Detect Signal) must be connected to ground if it is not used in the hardware design.
- If a drive has not been initialized by this command, any use of other file related commands that reference that drive name will fail.

**ALFAT in SD-Reader Mode**

In this configuration, the SD Card (**M:)** is presented as an external file-system to the PC (or other USB Host) connected to **U1:**.

When the **I** command is successfully finished the U1_CONNECT pin goes high. When the PC has successfully connected with **U1:**, the pin is pulled low; it stays that way until a disconnect occurs or another **I** command is issued that takes ALFAT out of SD-Reader mode. This is not real-time, nor is it the same for all device connections. The pin can stay low for few milliseconds after a lost connection; it can take even longer to go from high to low (when the physical connection is established). This is due to differences between device speeds at either end of the USB bus and the speed of the USB bus.

The physical U1_CONNECT pin is different depending on the Master to ALFAT interface method. When interfaced using UART or SPI, the U1_CONNECT pin is I2C_SCL; when using I2C, the pin is UART_TX.

**Initializing the Memory Card Device with a Different Clock Frequency:**

| | | |
|---|---|---|
| **Format** | I<SP>M:>{d}<LF><br>!{rr}<LF> | d SD bus clock frequency mode:<br>0 - 24 MHz.<br>1 - 16 MHz.<br>2 - 12 MHz. **(the default value**)<br>3 - 9.6 MHz.<br>4 - 8 MHz.<br>rr - the Result-code |
| **Example** | I<SP>M:>0<LF><br>!00<LF> | Initialize memory card with 24 MHz. clock frequency. |

The default SD bus clock frequency is 12 MHz. This frequency works on the majority of the memory cards. Some cards can work with higher frequencies. Higher frequencies will enhance the access speed.

**Notes related to the physical SD Card:**

- Some SD cards work at 24 MHz; but not the majority. Cards that do not work with this high frequency may initialize without error. Despite the successful mounting there is about a 20% chance that low level write sector requests will fail. It is highly recommended to use the default frequency if the final application should work with the majority of the cards.
- If the SD card is rated by the manufacturer with a high Class number, this does not mean that it will work with higher clock frequencies.
- If ALFAT is in SD-Reader mode, **M:** uses the default value of 12 MHz.

### 7.4.9 J - Read Register

| | | |
|---|---|---|
| **Format** | J<SP>{R}<LF><br>!{rr}<LF><br>${s}<LF><br>!{rr}<LF> | R – is the register number to read (1)<br>s - is the register's value<br>rr - the Result-code |
| **Example** | J<LF><br>!00<LF><br>$11<LF><br>!00<LF><br><br>J<SP>1<LF><br>!00<LF><br>$0101E<LF><br>!00<LF> | <br><br><br>Indicating Card Detect pin is high and USB1 is in High-Speed mode.<br><br><br><br>File-handles 0 and 8 are in use (opened) |

(1)It is permissible to leave the register number out of the command. In this case, the **J** command defaults to register number 0 (Device Status Register).

ALFAT supports a number of status registers. State values are defined by bits in the registers. Registers are not all the same size. Register bit numbers are defined with 0 being the right-most bit.

| Register Number | Name and Size |
|---|---|
| 0 | Device Status Register – S (one byte). Bits showing media/device status and modes |
| 1 | File Status Register – SS (two bytes). Bits flags showing usage of file-handles |
| 2 | Auxiliary Device Status Register – SS (two bytes). |

### Bit Definitions

Device Status Register – 0

| 7 | 6 | 5 | 4 | 3[1] | 2[1] | 1 | 0 |
|---|---|---|---|---|---|---|---|
| U1: (D:)<br>0 - in Host Mode<br>1 - in SD-Reader Mode | U1:<br>0 - not attached<br>1 - attached | U0:<br>0 - not attached<br>1 - attached | U1:<br>0 - Full-Speed<br>1 - High-Speed | U1:<br>0 - not mounted[N1)]<br>1 - mounted | U0:<br>0 - not mounted[N1)]<br>1 - mounted | M: WP<br>0 - R/W<br>1 - R | M: CD<br>0 - no<br>1 - detect |

[1] for “not mounted”, either no **I** command has occurred for the device or the device was mounted and then the media was removed. WARNING: do not remove mounted media without first closing all open file-handles.

File Status Register – 1

| F | E | D | C | B | A | 9 | 8 |
|---|---|---|---|---|---|---|---|
| 0 - closed<br>1 - opened | 0 - closed<br>1 - opened | 0 - closed<br>1 - opened | 0 - closed<br>1 - opened | 0 - closed<br>1 - opened | 0 - closed<br>1 - opened | 0 - closed<br>1 - opened | 0 - closed<br>1 - opened |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 - closed<br>1 - opened | 0 - closed<br>1 - opened | 0 - closed<br>1 - opened | 0 - closed<br>1 - opened | 0 - closed<br>1 - opened | 0 - closed<br>1 - opened | 0 - closed<br>1 - opened | 0 - closed<br>1 - opened |

Each bit in this register corresponds to a file-handle. If the bit is 1, the handle is in use; otherwise it is free. Counting the number of 1 or 0 valued bits yields the number of open or closed files, respectively. WARNING: do not remove mounted media without first closing all open file-handles.

Auxiliary Device Status Register – 2

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| reserved | reserved | reserved | reserved | reserved | reserved | reserved | reserved |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| reserved | reserved | reserved | reserved | K1: Filtering<br>0 - ASCII<br>1 - Raw | K0: Filtering<br>0 - ASCII<br>1 - Raw | U1: or K1:<br>0 - U1:<br>1 - K1: | U0: or K0:<br>0 - U0:<br>1 - K0: |

Bits 0 and 1 indicate whether an **I** command was performed to initialize a USB interface as a USB Keyboard Host or as a Mass Storage Host.

Bit 2 is only defined if Bit 0 has value 1
Bit 3 is only defined if Bit 1 has value 1

### 7.4.10 K - Get Free Size

| | | |
|---|---|---|
| **Format** | K<SP>{X}<LF><br>!{rr}<LF><br>${ssssssssssssssss}<LF><br>!{rr}<LF> | X - is the Drive name, one of:<br>**M:** - Memory Card drive<br>**U0:** - USB Flash drive 0<br>**U1:** - USB Flash drive 1<br>**K0:** - USB Keyboard 0<br>**K1:** - USB Keyboard 1<br>ssssssssssssssss:<br>• For Keyboards, this is the number of bytes available from ALFAT.<br>• For all other drives, this is the number of free bytes on the drive.<br>rr - the Result-code |
| **Example** | K<SP>**U1:**<LF><br>!00<LF><br>$00000000717F0000<LF><br>!00<LF> | There are 1904148480 free bytes available on USB Flash drive 1. |

Returns the media's remaining free size. Note this command may take several seconds for calculations to finish depending on the media size and type.

For Keyboards, this is the number of bytes available from ALFAT.

Note for file system drives, the drive must be initialized (I - Initialize and Configure Devices).

### 7.4.11 @ - Initialize Files and Folders List

| | | |
|---|---|---|
| **Format** | @<SP>{full path}<LF><br>!{rr}<LF> | full path: identifies the folder who's directory entries will be read using the **N** command.<br>rr - the Result-code |
| **Example** | @<SP>M:\TEST\TMP<LF><br>!00<LF> | Prepare to list all files and folders in the path “M:\TEST\TMP” |

This command establishes internal variables needed for retrieving all directory entries for a given path. It is a required initialization that must take place before using the **N** command (described below). No command other than **N** may be sent after **@** or **N** will not work.

### 7.4.12 N - Get Next Directory Entry

| | | |
|---|---|---|
| **Format** | N<LF><br>!{rr}<LF><br>{NNNNN.EEE}<LF><br>${AA}<LF><br>${ssssssss}<LF><br>!{rr}<LF> | NNNNN - variable length file-name<br>EEE - File Extension (if any)<br>AA - File Attributes<br>ssssssss - size of file in bytes.<br>rr - the Result-code |
| **Example** | @<SP>M:\TEST\TMP<LF><br>!00<LF><br>N<LF><br>!00<LF><br>TEST0001.TXT<LF><br>$00<LF><br>$0000FE23<LF><br>!00<LF><br>N<LF><br>!00<LF><br>TEST0002.TXT<LF><br>$20<LF><br>$00001234<LF><br>!00<LF><br>N<LF><br>!04<LF> | Retrieves all (two) file-names in the folder " M:\TEST\TMP" |

File Attributes are a byte of bit flags; defined as the "Standard Attribute Structure" in FAT systems, as shown below:

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

File names returned by this command do not include the path.

The **@** command (described above) must be called before using this command. Each time a **N** command successfully finishes, ALFAT retrieves a directory entry from the path specified in the **@** command. It increments a pointer so that the next **N** command will get the next directory entry.

When the pointer reaches the end of the list, and **N** is called again, ALFAT returns a Result-code of "04".

If, during this iterative process, any command other than **N** is sent, subsequent **N** commands will fail. The **@** command can be called to re-initialize the list system (the list pointer will be reset to the first directory entry).

## 7.4.13 O - Open File for Read, Write or Append

| | | |
|---|---|---|
| **Format** | O<SP>{N}{M}>{file-name}<LF><br>!{rr}<LF> | File-name - an absolute file specification that will be opened, including the device name.<br><br>N - the file-handle that will refer to the open file(1).<br><br>M - the access mode for subsequent file operations(2)<br><br>rr - the Result-code |
| **Example** | O<SP>1R>M:\TEST\VOLTAGE.LOG<LF><br>!00<LF> | Open file VOLTAGE.LOG with file-handle 1 and read access mode. This file is located in the "TEST" folder on the SD memory card. |
| | O<SP>0W>U0:\DAT\CURRENT.LOG<LF><br>!00<LF> | Open file CURRENT.LOG, on USB0, using file-handle 0. The file-handle can be used in the **W** command (Write File). If folder DAT is not available, it will be created automatically |

(1)**N,** the file-handle

- must be a character value from '0' to 'F'.
- must be free (not in use from another **O** command).
- The virtual file-handles 'Z' (**K0:**) and 'Y' (**K1:**) are not allowed in this command.

(2)**M,** the access mode, is one of:

- '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, the default position for subsequent writes is 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, is associated with a handle. That file-handle is no longer free and can not be used in another Open command without first using a Close command (**C**) to free up the file-handle. It is easy to use an unlimited number of files by closing one file to open another.

NOTES:

- before opening a file, the Initialize command (**I**) must have been used on the device with the file.
- This command may not be used with the USB Keyboards (**K1:**, **K0:** ; virtual file-handles ('Y' and 'Z'))

### 7.4.14 R - Read from File

| | | |
|---|---|---|
| **Format** | R<SP>{N}{F}>{ssssssss}<LF><br>!{rr}<LF><br>dddddd...<br>${aaaaaaaa}<LF><br>!{rr}<LF> | N - file-handle<br><br>F - the “Filler” Byte<br><br>ssssssss - required number of return bytes<br><br>dddddd... - the return stream of data exactly ssssssss bytes<br><br>aaaaaaaa - is the number of actual bytes in the returned stream that were available in the file (less than or up to ssssssss bytes)<br><br>rr - the Result-code |
| **Example** | Given a file, opened with handle 2, it contains 8 bytes “ABCDEFGH” | |
| | R<SP>2^>5<LF><br>!00<LF><br>ABCDE$00000005<LF><br>!00<LF><br>R<SP>2Z>5<LF><br>!00<LF><br>FGHZZ$00000003<LF><br>!00<LF> | Read 5 bytes from file-handle 2 with a Filler byte of “^”, the Master requests 5 bytes. ALFAT returns 5 significant bytes “ABCDE”. The Master then uses the **R** command to request 5 more bytes, with a Filler byte “Z”. ALFAT returns the last 3 bytes of the file “FGH” and 2 filler bytes “ZZ”. The number of significant bytes that were returned was 3. |

- **N** - the file-handle
  - for files on media devices with file-systems (card reader, USB drives)
    - must be a value from '0' to 'F'.
    - must be associated with a file (the Open command associates a filename with a file-handle).
  - For virtual-handles associated with USB Keyboards
    - must be 'Z' (**K0:**) or 'Y' (**K1:**)

- **ssssssss** - the number of bytes the Master requests ALFAT to read. Even if the device does not contain enough data to satisfy the request, ALFAT will return all available bytes and then it sends the Filler byte as many times as needed to return a byte count equal to **ssssssss**. Data available from file-system based devices is based on the current position in the file (byte index).

- **F** - the “Filler” byte – when the device does not have enough data to return the requested size; after the final file data is sent, enough filler bytes are sent to make the byte stream have a length that matches the requested bytes (**ssssssss)**.

- **aaaaaaaa** - the number of actual bytes (the count of bytes in the return data (**dddddd...)** that are *not* filler bytes). This number will be less than or equal to **ssssssss.**

This command is used to read data from a device associated with the specified handle. For file-system based devices, the file must have been opened with Read access mode.

In order, ALFAT

1. sends back data from the file while incrementing its internal file pointer.
2. if the the file pointer reaches the end of the file, ALFAT returns the Filler byte until it reaches the total size required in the command.
3. sends the actual number of bytes read from the file.

WARNING for I2C only, the Filler byte can not be 0xFF or 0x00. See the I2C interface section for more details.

For UART, we advise that the Master's input buffer (for processing data from ALFAT) be as large, or larger, than the largest size of data requested by any read command.

The DATAREADY pin can be used with the Read command when the Master to ALFAT interface mode is either I2C or SPI interfaces; when using a UART interface, ALFAT sends the data when it is available. Use of DATAREADY state:

- DATAREADY will be high while ALFAT is parsing and evaluating a command string.

- The transfer of data during a read command is processed from the media in chunks (1024 bytes). Each time a chunk has been processed by ALFAT and is ready for the Master to retrieve, ALFAT sets the pin high; when data is not ready the pin will be low. For a block of less than 1024 bytes, ALFAT set the pin high when it places the last byte from the media into the buffer. For I2C and SPI, the Master must perform read(s) until the current buffer is empty.

- No new commands may be sent by the Master until the entire read command transaction is finished (the Master has read the final Result-string).

### 7.4.15 W - Write to File

| | | |
|---|---|---|
| **Format** | W<SP>{N}>{ssssssss}<LF><br>!{rr}<LF><br>dddddd...<br>${aaaaaaaa}<LF><br>!{rr}<LF> | N File Handle<br><br>ssssssss - the number of bytes to write<br><br>dddddd... a stream of raw data bytes exactly ssssssss bytes long<br><br>aaaaaaaa number of bytes actually written to the file.<br><br>rr - the Result-code |
| **Example** | W<SP>1>10<LF><br>!00<LF><br>1234567890abcdef<br>$00000010<LF><br>!00<LF> | Write 16 bytes to the file associated with handle 1 |

- **N,** the file-handle
  - must be a numeric value from 0 to F.
  - must be associated with a file (from Open command).
- **ssssssss**: the number of bytes to be written.
- aaaaaaaa: the number of actual bytes

This command is used to write data to a file associated with the specified handle.

All sizes/lengths in the command are in hexadecimal format.

The file must have been opened with write or append access mode.

- The data stream (dddddd...**)** should *not* be sent until the Master has parsed the command and returned the first Result-string indicating success.
- If the first Result-code is an error (not “00”), the Master must terminate the write command and send no more bytes.

All commands issued before using the Write command must be completed, including the read of the final Result-string. The command FIFO buffer must be empty when Write is sent.

If an error occurs while ALFAT is receiving the data stream, ALFAT will continue to read bytes until the required number has been sent by the Master. Then ALFAT returns the error in the second Result-string.

To make sure that the data is physically written to a file, the file **must** be flushed (**F** command) or closed (**C** command). Otherwise, there is a risk of losing data or corrupting the file-system when the storage media is removed, there is a power loss, or a reset.

### 7.4.16 L - Fast Write to File (SPI mode only)

| | | |
|---|---|---|
| **Data Flow:** | SPI Write Frame:<br>**L<SP>{n}>{ssssssss}<LF>**<br><br>SPI Read Frame:<br>**!00<LF>**<br><br>Host sends data with **No framing**<br><br>SPI Read Frame:<br>$aaaaaaaa<LF><br>!00<LF> | N: File Handle ( 0 to F).<br><br>ssssssss: the data size to be written.<br><br>aaaaaaaa: the number of bytes actually written. |

At a high level, this command works exactly as a Write command (**W**). It is only available when using the SPI interface; in this case, ALFAT uses internal DMA to achieve higher data receive rates (average 1.4 MByte/s) than with the regular W command. Additional information on data throughput is available in the Serial Interface Speed Overhead section.

Algorithm for **I** command:

1.Using normal SPI write frames, the Master initiates the **L** command by sending "**L<SP>{n}>{ssssssss}<LF>)**".

2.ALFAT accepts the request and sends the acknowledgment (using normal SPI Read framing).

3.ALFAT opens its SPI DMA channel.

4.Master checks SPI_BUSY. If it is not busy then it sends 8192 bytes (or less) of the data. Data is streamed with no framing. Normally when using SPI there is a minimum delay of 2µS required between each byte. With this command, the entire 8192 bytes (or less) are sent without delay between bytes.

5. The Master repeats step 4 until all ssssssss bytes of data are sent .

6. The Master checks SPI_BUSY. If it is not busy, it reads the Result-string with normal Read frames.

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

The virtual file-handles 'Z' (**K0:**) and 'Y' (**K1:**) are not allowed in this command.

### 7.4.17 F - Flush File Data

| | | |
|---|---|---|
| **Format** | F<SP>{n}<LF><br>!{rr}<LF> | N - file-handle , one of 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E or F<br>rr - the Result-code |
| **Example** | F<SP>0<LF><br>!00<LF> | Flush File-handle 0 |

This command flushes (commits) any buffered data to the specified open file. This command physically saves all data to the media.

Note the Close command will flush the file before closing the file.

The virtual file-handles 'Z' (**K0:**) and 'Y' (**K1:**) are not allowed in this command.

### 7.4.18 C - Close File

| | | |
|---|---|---|
| **Format** | C<SP>{n}<LF><br>!{rr}<LF> | n – file handle can be 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E or F<br>rr - the Result-code |
| **Example** | C<SP>0<LF><br>!00<LF> | Close File-handle 0 |

This command cause ALFAT to first perform a flush file command (**F**). After all buffers have been written, ALFAT releases the file-handle and associated resources. The file-handle is now free and can be used in an Open command.

The virtual file-handles 'Z' (**K0:**) and 'Y' (**K1:**) are not allowed in this command.

### 7.4.19 P - File Seek

| | | |
|---|---|---|
| **Format** | P<SP>{N}>{ssssssss}<LF><br>!{rr}<LF> | N - File Handle (0 through F)<br><br>ssssssss - the new position of the file's index/pointer (hexadecimal)<br><br>rr - the Result-code |
| **Example** | P<SP>1>10<LF><br>!00<LF> | Change the file pointer of file-handle '1' to start-of-file plus 10.<br><br>rr - the Result-code |

This command changes the current byte position in a file relative to the start of the file (index 0).

**For files that were opened for read mode ("R")** valid values range from 0 (start of file) to the file size (first byte past the last data byte). Subsequent Read commands will return data from this index as the starting point.

**For files opened with write mode ("W")**, the only valid value for seeking is 0. Subsequent writes will:

- overwrite any data already present (in the range of 0 to the size of the data written),
- may extend the size of the file, and
- leaves the index 1 past the last byte written.

The virtual file-handles 'Z' (**K0:**) and 'Y' (**K1:**) are not allowed in this command.

See also the Tell command (**Y**)

### 7.4.20 Y - File Tell

| | | |
|---|---|---|
| **Format** | Y<SP>{N}<LF><br>!{rr}<LF><br>${ssssssss}<LF><br>!{rr}<LF> | N - File-Handle ('0' through 'F')<br>ssssssss - the current value of the file pointer/index in hexadecimal notation.<br>rr - the Result-code |
| **Example** | Y<SP>1<LF><br>!00<LF><br>$00000003<LF><br>!00<LF> | The file pointer of file-handle 1 is at position 0x03 |

Gets the current byte index in a file. This will be a value from 0 (start of file) to file size (end of file)

The virtual file-handles 'Z' (**K0:**) and 'Y' (**K1:**) are not allowed in this command.

### 7.4.21 D - Delete File or Folder

| | | |
|---|---|---|
| **Format** | D<SP>{Name[\]}<LF><br>!{rr}<LF> | Name - The absolute (non-relative) name of the object to be deleted.<br>rr - the Result-code |
| **Example** | D<SP>M:\TMP\TEST.TXT<LF><br>!00<LF> | Remove the file with name TEST.TXT in the TMP folder on the SD memory card. This will not delete the TMP folder. |
| | D<SP>M:\TMP\<LF><br>!00<LF> | Remove the FOLDER with name TMP on the memory card. The folder must be empty. |

Deletes a file or a folder. A name that is a folder (directory) must be appended with the backslash “\”; for files there should be no “\”

- The target file or folder must exist.
- If a folder is being deleted, it must be empty.
- If the path contains a file-name, the file must be closed.

### 7.4.22 ? - Find File or Folder

| | | |
|---|---|---|
| **Format** | ?<SP>{Name}<LF><br>!{rr}<LF><br>${ssssssss}<LF><br>${AA}<LF><br>${hh}:{mm}:{ss}<SP>{mm}-{dd}-{yyyy}<LF><br>!{rr}<LF> | Name - The absolute (non-relative) name of the target to be located.<br><br>ssssssss - size of the target in bytes as a hexadecimal number<br><br>“AA” - a set of bit flags, the target's “Attributes”. Represented in hexadecimal.<br><br>hh:mm:ss is the last time the target was modified. Hours are from 0-23 (1:00 P.M. == 13:00)<br><br>mm-dd-yyyy (month-day-year) is the last date the target was modified.<br><br>rr - the Result-code |
| **Example** | ?<SP>M:\TEST.TXT<LF><br>!00<LF><br>$00000F34<LF><br>$00<LF><br>$12:00:00<SP>01-24-2011<LF><br>!00<LF> | The target TEST.TXT has been found on the SD Reader. Its size is 3892 bytes. The Attributes indicate the target is a file, it is visible, it can be written,...)<br>The last modification time and date is 12:00:00 on 1/24/2011 |

This command searches for a specific file or folder (the “Target”). If the target exists (Result-code of “00”), ALFAT returns the file size, a set of attributes for the target, and the time and date of the target's last modification.

The Attributes are defined as the “Standard Attribute Structure” in the FAT file-system. T bit meanings are:

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

### 7.4.23 M - Copy From File to Another

| | | |
|---|---|---|
| **Format** | M<SP>{S}<SP>{I}<SP>{D}<SP>{LLLLLLLL}<LF><br>!{rr}<LF><br>${xxxxxxxx}<LF><br>!{rr}<LF> | S – file-handle of source<br><br>I – file offset to use as the starting point of the data to be copied from S.<br><br>D - file-handle of the file which is the destination for the data.<br><br>LLLLLLLL - number of bytes to be copied in hexadecimal<br><br>xxxxxxxx - the number of bytes actually transferred<br><br>rr - the Result-code |
| **Example** | M<SP>0<SP>0<SP>1<SP>64<LF><br>!00<LF><br>$00000064<LF><br>!00<LF> | A successful 100 byte copy from the file opened with file-handle 0, starting from the beginning of the file. The file opened with1 receives the data. |

This command copies data from one file to another.

Files may be located on different devices (for instance one on USB1 and the other on USB0)

This command supports copying a specific piece of the source file to the destination using the combination of INDEX and LENGTH.

### 7.4.24 A - Rename file

| | | |
|---|---|---|
| **Format** | A<SP>{OLD_NAME}>{NEW_FILE_NAME}<LF><br>!{rr}<LF> | OLD_NAME - the absolute (non-relative) name of the file to rename. OLD_NAME must exist.<br><br>NEW_FILE_NAME - The new name of the file. Do not include the path to the file; this is just the filename.<br><br>rr - the Result-code |
| **Example** | A<SP>U0:\GHI\ALFAT.TXT>ALFAT001.TXT<LF><br>!00<LF> | Rename ALFAT.TXT (on USB0 in the directory GHI) to ALFAT00.TXT. The new file's full (absolute) name is U0:\GHI\ALFAT001.TXT |

### 7.4.25 E - Test Media Speed

| | | |
|---|---|---|
| **Format** | E<SP>{X}>{ssssssss}<LF><br>!{rr}<LF><br>${aaaaaaaa}<LF><br>${bbbbbbbb}<LF><br>!{rr}<LF> | X - Drive name, one of:<br>"M:" Memory Card drive<br>"U0:" USB Flash drive 0<br>"U1:" USB Flash drive 1<br>ssssssss - number of bytes of data that will be tested. It should divide evenly by 1024.<br>aaaaaaaa - total millisecond for writing (hexadecimal)<br>bbbbbbbb - total milliseconds for reading (hexadecimal)<br><br>rr - the Result-code |
| **Example** | E<SP>M:>6400000<LF><br>!00<LF><br>$00005D14<LF><br>$000039FB<LF><br>!00<LF> | Test write and read of 100MB on the Memory Card.<br>It takes 23828 milliseconds for writing and 14843 milliseconds for reading. |

A drive must be initialized and mounted before using the **E** command (see the section I - Initialize and Configure Devices )

This command may take a few seconds to minutes for calculations to finish, depending on the test size and the media. There must be at least two free handles available for this command to use internally.

### 7.4.26 Q – Format

| | | |
|---|---|---|
| **Format** | Q<SP>CONFIRM FORMAT<SP>{X}<LF><br>!{rr}<LF><br>!{rr}<LF> | X is drive name, one of:<br>“**M:**” - Memory card.<br>“**U0**:” - USB flash drive 0<br>“**U1:**“ - USB flash drive 1<br>rr - the Result-code |

Note: Depending on the media (size and access speed), this command may take a long time to finish. The first Result-string is sent before formatting starts.
The second Result-string is sent when formatting is done using the Bootloader, Firmware Updates

### 7.4.27 <LF> – No Operation (NOP)

| | | |
|---|---|---|
| **Format** | <LF><br>!{rr}<LF> | No operation.<br>rr - the Result-code |

Sending ALFAT <LF> by itself (not proceeded by any other characters) always returns a Result-string with a Result-code of “00”, indicating success.

Whenever ALFAT is reset/powered-up, it sends the Master a banner string. It is important that this banner be read by the Master. To do this the NOP command is sent. See also the section Boot/Reset Protocol.

# 8 The Bootloader

## 8.1. General Description

The bootloader is the software that runs when the ALFAT SoC is reset or powered-up. It has two primary functions:

1. Boots/initializes the state of the hardware; verifies and launches/starts-execution of the ALFAT firmware.
2. GHI Electronics maintains ALFAT firmware with improvements and bug fixes. The bootloader provides the Master or development system an interface for ALFAT firmware installation.

## 8.2. Connecting and Controlling the Bootloader

When the ALFAT SoC is reset or powered-up, if SPI_SSEL is low and SPI_MOSI is high, the bootloader supports a small set of commands transmitted over UART. There is no support for other ports.

**Unless firmware updates will never be done, hardware designs must support UART_TX and UART_RX pins.**

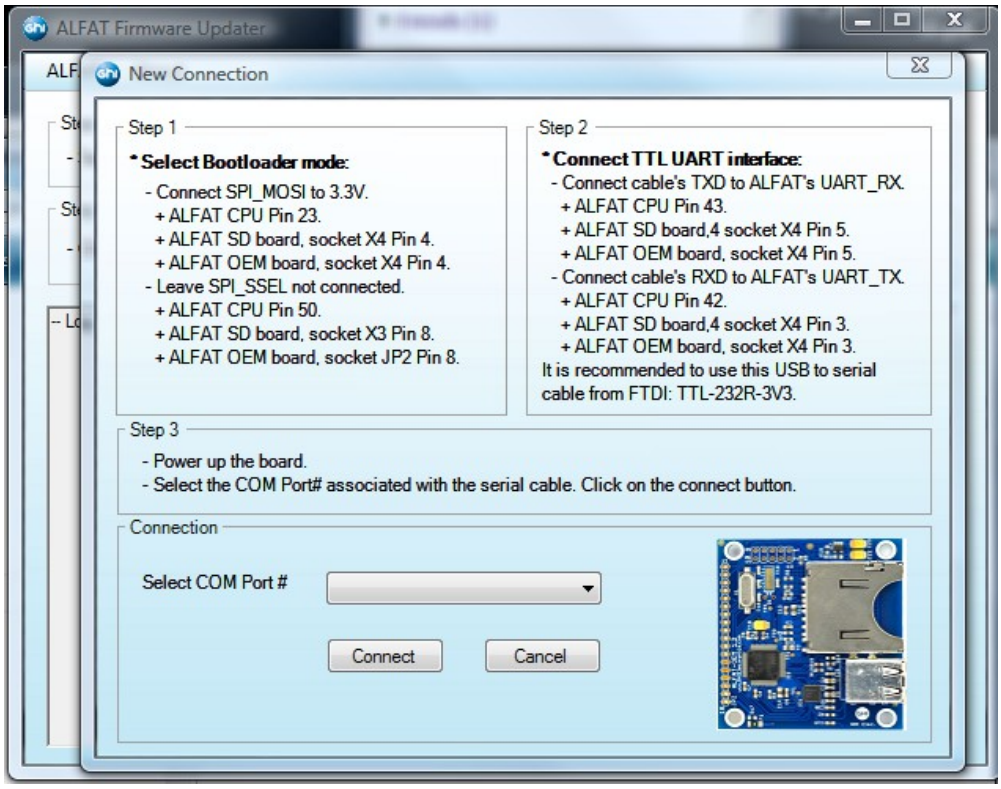
Use of UART by the bootloader is *not* related to the Interface (SPI, I2C, or UART) used for communication with the firmware. See the section "Selecting the ALFAT Access Interface Mode"

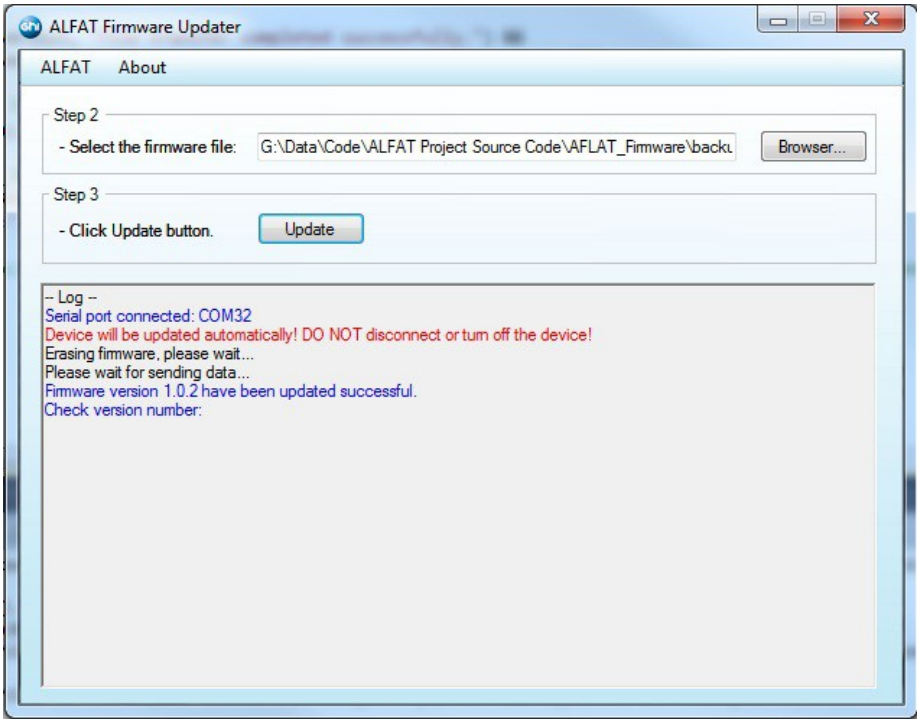
Firmware transfer to ALFAT uses the XMODEM protocol with 1 KB packets and CRC.

## 8.3. Firmware Updater Application

An updater application is provided by GHI Electronics for updating the firmware from development computers. Users may also wish to implement this functionality right into the Master so an update can be performed by the Master. To connect ALFAT, or one of the ALFAT OEM boards to a PC for update, a TTL serial connection is needed. If using a regular PC serial port then a RS232 to TTL level converter is required between ALFAT's UART interface and the PC's serial port.

We recommend using an USB TTL serial cable. Here is a part number for a FTDI cable : TTL-232R-3V3 from FTDI. The USB side of the cable is recognized the PC as a virtual serial port (COM port) and it provides TTL 3.3V levels at its pin connector side that can be connected directly to ALFAT's UART interface.


ALFAT Firmware Updater
New Connection
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 #
Connect
Cancel


ALFAT Firmware Updater
ALFAT About
Step 2
- Select the firmware file:
G:\Data\Code\ALFAT Project Source Code\AFLAT_Firmware\backu
Browser...
Step 3
- Click Update button.
Update
-- 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:

## 8.4. Bootloader 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. |

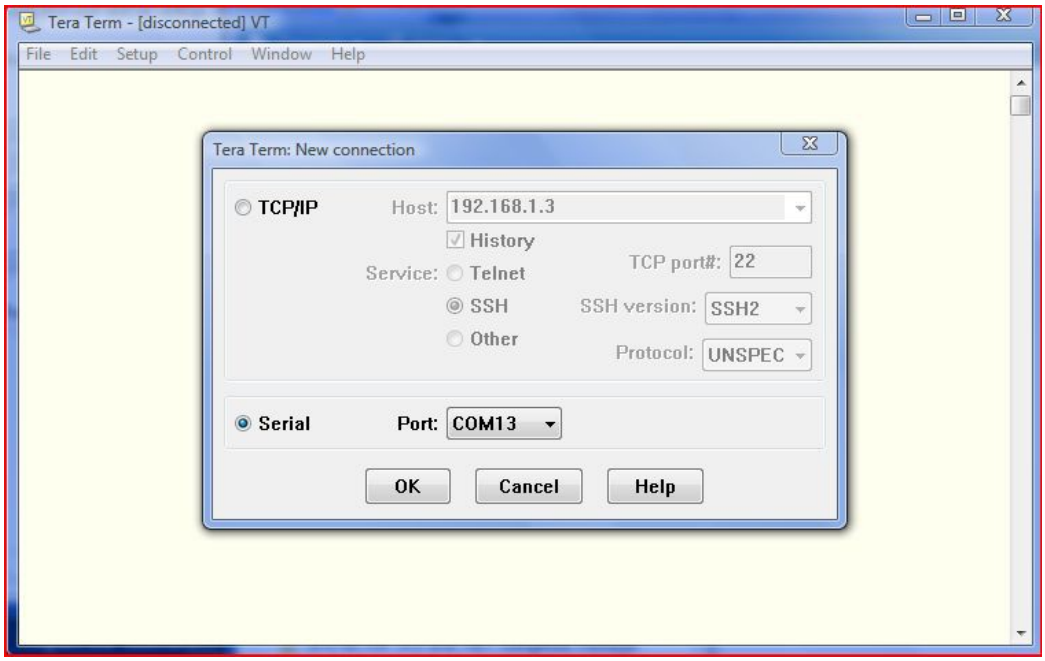
**Note:** The bootloader is entirely separate program that loads the ALFAT firmware. The version number of the bootloader may not match the ALFAT firmware version number. The bootloader can't be updated.

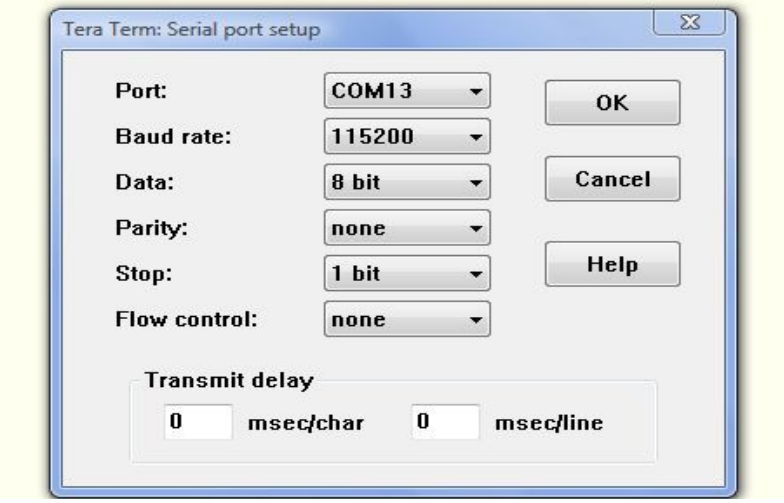
## 8.5. Updating the Firmware Using a Terminal Console

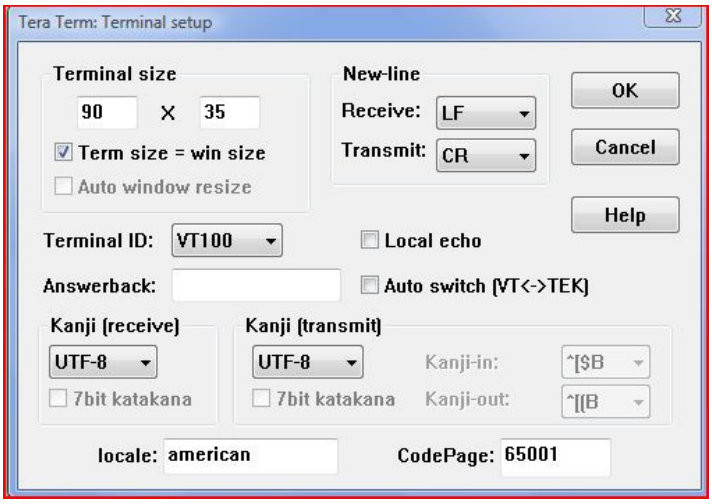
We recommend using the provided firmware updater application (described in previous session). But here is an example that uses a terminal console like TeraTerm to update the firmware instead.

Put ALFAT in the bootloader mode (SPI_SSEL low and SPI_MOSI high; reset)

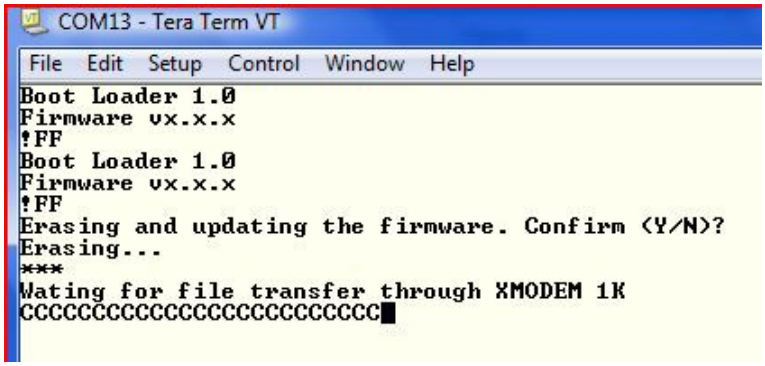
Open the relative COM port and set the baud rate to 115200 and set the New-line receive to LF.



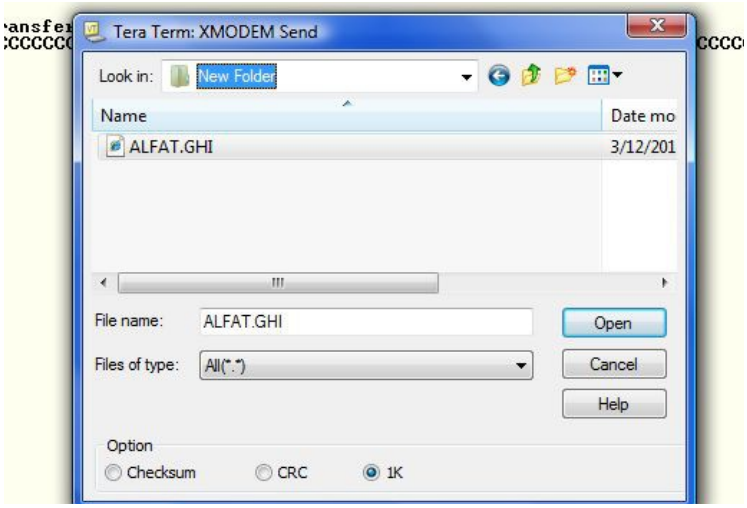


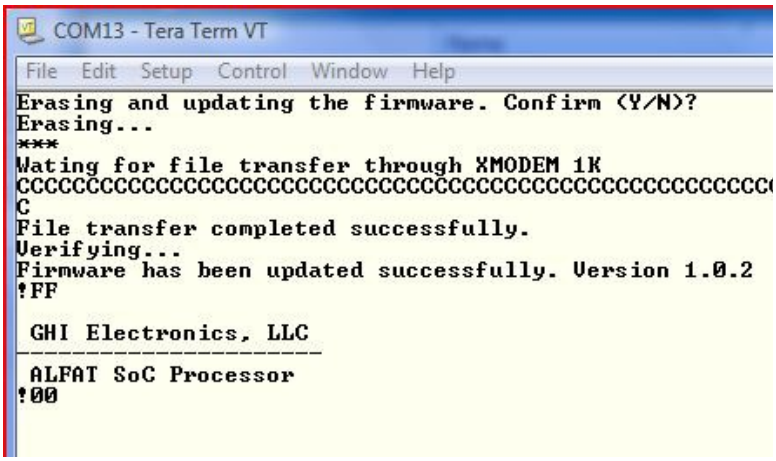

Use X command and follow the instructions




Choose to send a file using **XMODEM with 1K option** then choose ALFAT.GHI The filename on the GHI website will reflect the version number,e.x “ALFAT(2.2.0).GHI”. The firmware file is downloaded from GHI Electronics' website; click on the “Support” tab, then the “File System” button, follow the firmware link (GHIelectronics.com Support->File System)




After the file has been transferred, the bootloader will send a message similar to "Firmware has been updated successfully. Version 2.0.0." After that, release the bootloader mode and reset the chip, or use the run R command to run the new firmware.



# 9 Hardware integration guide

Reference schematics for the ALFAT Soc processor are found under its Catalog entry on GHI Electronics' website; click on the Resources tab, and find the link to ALFAT's schematic.

## 9.1. Power Source

Power sources are the cause of many problems. ALFAT is capable of running at lower voltage or somewhat noisy voltage source. However, the media devices may or may not work on an unstable power source. Make sure that the power source to the storage media is reliable and there is a large enough capacitor as close as possible to the media power pins. We recommend adding 0.1uF and 22uF capacitors.

- If 5V is connected to pin 18, there is no need 3.3V on pin 13.
- If 3.3V is connect on pin 13, then 5V is not strictly necessary on pin 18; however, USB will not work. USB requires 5V on this pin.

## 9.2. Crystals

ALFAT's main clock is provided through a 12 MHz Crystal with 500PPM or less and a load capacitance around 18pF.

Real Time Clock crystal's value should be 32.768 kHz and the load capacitance is 12.5pF.

## 9.3. Card Detect and Write Protect signals

When the SD (memory card) interface is being initialized (**I** command), ALFAT samples the card detect input (CD, pin 25) If CD is Low, ALFAT will initialize the card; otherwise, ALFAT will reject the command. Consequently, if the card detect signal is not implemented in the hardware design, the CD pin must be connected to GND.

When sent any commands that requires write access to the memory card (for example W, Q, A, …) ALFAT checks the write protection input (pin 24/ WP). If WP is low, then ALFAT proceeds with the command. If a write protection signal is not implemented in the hardware design, then the WP pin must be connected to GND.

## 9.4. Full-Speed / High-Speed with ULPI PHY

To use USB's “Full-Speed” mode (12Mbps), two 22ohm resistors must be connected in serial with the data+(DP) and data- (DM) signals of each of ALFAT's USB interfaces.

USB1 can be operated in High-Speed mode (480 Mbps) though the use of an external ULPI PHY (for example a Fairchild FUSB2805).High-Speed.

This PHY requires 19.2 MHz clock source. To save the cost of an additional crystal, ALFAT generates a 19.2 MHz clock at pin 41 that can be used with the PHY.

**Notes:**

- If ULPI PHY is used, the two 22ohm resistors are recommended (in serial with the data+(DP) and data- (DM) lines). There have been reports that, with the ULPI PHY and 22ohm resistors, some USB Flash/Thumb drives do not work.
- If ULPI PHY is implemented, when initializing USB1 (**I** command with “U1:”), the High-Speed attribute must be used (“U1:H”). This is still true even if the physical device does not support High-Speed mode. This applies to the ALFAT OEM board.

## 9.5. Real Time Clock

ALFAT tags modified or created files with the current time and date. To keep track of time, ALFAT uses the internal real time clock. There are two options for implementing the RTC.

The first option is ”Shared” mode, where the RTC runs off the ALFAT processor's clock and power. With this option,no external components are required. VBAT should be connected to VCC. With this option, the Master would normally set the time and date with every reset/power-up.

The second option is “Backup” mode, where The RTC clock uses an external 32.768 kHz crystal and runs off VBAT power (commonly provided by a 1.65 to 3.6V coin battery). Using a battery ensures that the RTC keeps operating even if the system is powered off. Using this mode, the RTC consumes about 1 uA. .

To ensure that VBAT get powered from the backup battery only when the main power is off, two diodes should be placed in the circuit.

## 9.6. Bootloader Access jj

The current ALFAT firmware is very stable. . Periodically we may release minor updates and/or major upgrades. We highly recommend using a design that supports installing firmware. This includes maintaining access to the following pins: UART RX, UART TX, loader, and RESET.

## 9.7. Electrical characteristics

ALFAT SoC is based on STM32F205RBT6. Consult with STM32F205RBT6 datasheet for electrical characteristics if needed.

# 10 ALFAT Off-the-shelf Circuit Boards

GHI Electronics offers off-the-shelf OEM boards that uses the ALFAT SoC processor. These boards expose all the 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.

## 10.1. OEM Board Pin-outs

All OEM boards expose the same signals through a 1x18 pin-mount[1]. If the desired interface is UART, all signals required are exposed through a secondary 2x5 pin-mount.

**1x18 pin-mount**

| Pin | Name | Pin | Name |
|---|---|---|---|
| 1 | UART_TX/U1_CONNECT | 10 | DATAREADY |
| 2 | UART_RX/SPI_BUSY/ I2C_BUSY | 11 | Reserved |
| 3 | I2C_SCL/U1_CONNECT | 12 | VBAT |
| 4 | I2C_SDA | 13 | Internal 3.3V (Do not connect) |
| 5 | SPI_SCK | 14 | RESET (not 5V tolerant) |
| 6 | SPI_MISO/UART_BUSY | 15 | GND |
| 7 | SPI_MOSI | 16 | Not connected[2] |
| 8 | SPI_SSEL | 17 | Not connected[1][2] |
| 9 | WAKE | 18 | 5 Volts[1] |

[1]The ALFAT SD Pin-out uses a 1x16 pin-mount. Pins 1 through 16 are the same as described in the above table

[2]The ALFAT SDR board exposes the the USB0 data pins. Pin 16 is data minus (-), Pin 17 is data plus (+)

**2x5 pin-mount**

| Pin | Name | Pin | Name |
|---|---|---|---|
| 1 | Internal 3.3V (Do not connect) | 2 | 5V |
| 3 | UART_TX/U1_CONNECT | 4 | SPI_MOSI |
| 5 | UART_RX/SPI_BUSY/I2C_BUSY | 6 | SPI_MISO/UART_BUSY |
| 7 | Reserved | 8 | VBAT |
| 9 | GND | 10 | RESET (not 5V tolerant) |

## 10.2. ALFAT OEM Board

The ALFAT OEM Board exposes all of ALFAT™ SoC's processor features. This board offers a seamless way to access files on SDHC, SD and MMC cards plus devices supporting the USB UMC protocol such as thumb 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 USB 2.0 port.
3. The board includes pads for a RTC 32.768 kHz crystal. The crystal is not included.

USB0 is the lower socket, USB1 the upper.

## 10.3. ALFAT SD Board

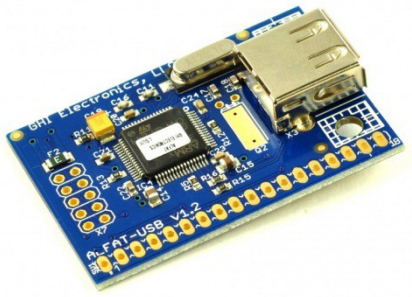
ALFAT SD board is an OEM board using the ALFAT™ SoC processor. This board offers a seamless way to access files on SD, SDHC and MMC cards. A standard SD/MMC connector with a push spring is included

The board includes pads for RTC 32.768 kHz crystal (the crystal is not included).

The other OEM boards use an 18 pin-mount; the ALFAT SD Pin-out uses 16 pins (pins 1-16 have identical corresponding signals as the 18 pin-mount)

## 10.4. ALFAT USB Board

ALFAT USB Board is an OEM board exposing all of ALFAT™ SoC's features. This board offers a seamless way to access files on USB MSC Clients such as thumb drives.
The board includes pads for RTC 32.768 kHz crystal. The crystal is not included.
ALFAT USB includes a Full-Speed USB 2.0 port.



## 10.5. ALFAT SDR Board

ALFAT SDR Board is an OEM board for designs which take advantage of ALFAT SoC's SD-Reader mode. This board has a micro-USB socket for USB1 which is connected to an external PHY supporting High-Speed; as well as, a full SD Card reader.

## 10.6. ALFAT Evaluation Kit

The ALFAT Evaluation Kit is designed to make the mastering of ALFAT's capabilities quick and easy. It works with any of the boards described in this chapter. This kit is an invaluable guide for engineers new to ALFAT. Evaluation begins by:

1. connecting the Kit's hardware,
2. installing the provided software on a Windows PC, and
3. using the GUI to run some examples

The above procedure takes less than ten minutes and is explained, step by step, in the Quick Start Guide.

The kit includes:

- a “hosting” board, ALFAT-EVAL, provides the necessary circuitry to boot and interact with an ALFAT board.
- an integral USB port connects the EVAL assembly to a standard Windows PC.
- a friendly point-and-click program “ALFAT Explorer” is hosted on the PC. It controls the attached ALFAT,
- the source code for the Explorer program is provided so you can see the simplicity of ALFAT's API,
- The Evaluation Kit includes both an ALFAT OEM Board and an ALFAT SDR Board.
- Test media (SD Card, and USB Flash memory).

The Kit is powered by the USB cable

In summary, the ALFAT Evaluation Kit contains every necessary component to exercise ALFAT's capabilities right down to the USB cable, SD-Card, and USB flash storage. Full details can be found on GHI Electronics' web site (www.ghielectronics.com).

# 11 Performance

## 11.1. Selecting the Right Storage Media

The current storage media market is flooded with low grade devices. These devices may work on a PC but that doesn't mean the device follows standards and will work with ALFAT. 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 make our best effort to support a wide range of storage media that follow published standards. But GHI Electronics does not guarantee that ALFAT will be able to access all storage media

For products using ALFAT SoC processors, it is important to test different media devices. Then maintain a list of supported media for a product.. GHI Electronics does not recommend any specific brand but always recommends selecting a well known source. If media is not supported, a failure most often occurs at initialization (I command). Less often (rarely) the media will initialize but then have problems with reading and/or writing. If the media mounts with no errors, in most cases, it is safe to assume it will function normally.

As discussed in the integration section, power source can be a big source of problems. ALFAT is capable of running at lower voltage or  with low levels of noise. However, in those cases, the media may or may not work. Make sure the voltage source to the media is reliable and there is a large enough capacitor placed as close as possible to the media connector. We recommend adding 0.1uF and 22uF.

## 11.2. File Access Speed

There are many factors that affects file access speeds. Some storage media devices have internal buffering, others have high speed rating. But even on the exact same media, speeds might differ between 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 runs. Noisy power source may cause errors while accessing the media that slows the speed down.

## 11.3. Serial Interface Speed Overhead

The actual speed of a specific media can be easily determined using the E command, which range from 1000 KBytes/sec to 4000 KBytes/sec. This is the internal speed with complete FAT file-system overhead. Such speeds are achieved when using the copy command. But when using other commands like read or write commands, the serial connection with the Master adds a major delay overhead.

When one of the serial interfaces is used to exchange the data with a Master, some command-response overhead is introduced. There is also a maximum clock and other restrictions on the interfaces. On an average, file access speed is about 230 KBytes/sec when using UART or SPI and about 25 KBytes/sec when using I2C. That is about the same when using SD, USB FS or USB HS.

To achieve higher writing speed, ALFAT command-set includes **L** Command (Fast Write to File command). Available for SPI interface only, this command is very similar to the W Command but, with **L** Command, it uses internal DMA to reach faster receiving rates. The average speed with this command is 1400 KBytes/sec with SD cards, 1200 KBytes/sec with USB HS and 750 KBytes/sec with USB FS.

# 12 Result-codes

Result-codes are always strings with 2 characters. The 2 characters are always from the set of hexadecimal digits '0' to 'F'. So the string may be interpreted as a one byte hexadecimal value ranging from decimal 0 to 255. For example: Result-code "3F" as produced in the data stream from ALFAT to the Master will be the two ASCII bytes with values of 0x33 and 0x46.

| Value | Description |
|---|---|
| 00 | Command successful. |
| 01 | Unknown command. |
| 02 | Incorrect parameters. |
| 03 | Operation failed. This error code is returned also if the user attempted to write to write-protected card. |
| 04 | Reached the end of the file/folder list. This is not an error. |
| 10 | Media does not initialize. |
| 11 | Initialize media failed. This error code can be returned if the card detect signal is high or floating, check the Card Detect and Write Protect signals section for details. Makes sure there is media in the device (see the section J - Read Register) before using the initialize command (**I**), |
| 12 | Insufficient free space on storage media. |
| 20 | File/folder doesn't exist. |
| 21 | Failed to open the file. |
| 22 | Seek only runs on files open for read. |
| 23 | Seek value can only be within the file size. |
| 24 | File-name can't be zero. |
| 25 | File-name has forbidden character. |
| 26 | File/folder name already exists. |
| 30 | Invalid handle. |
| 31 | Handle source does not open. |
| 32 | Handle destination does not open. |
| 33 | Handle source requires file open for read mode.. |
| 34 | Handle destination requires file open for write or append mode. |
| 35 | No more handle available. |
| 36 | Handle does not open. |
| 37 | Handle is already in use. |
| 38 | Open file mode invalid. |
| 39 | Handle requires write or append mode. |
| 3A | Handle requires read mode. |
| 40 | The system is busy. |
| 41 | Command is supported with SPI interface only. |

| Value | Description |
|---|---|
| 70 | When initializing SD-Reader mode, no SD Card in Socket (Use **J** to detect before **I**) |
| 71 | For **"I K0:**" or **"I K1:**", no USB keyboard found. |
| 72 | Read or Write command (**R**, **W**) for file-handle **Z (K0:)** or **Y (K1:)** when not initialized |
| FF | bootloader indication code. |

# 13 DISCLAIMER

IN NO EVENT SHALL GHI ELECTRONICS, LLC. OR ITS PARTNERS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS PRODUCT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. SPECIFICATIONS ARE SUBJECT TO CHANGE WITHOUT ANY NOTICE. PRICES ARE SUBJECT TO CHANGE WITHOUT ANY NOTICE. ALFAT SOC PROCESSOR AND ITS LINE OF OEM BOARDS ARE NOT DESIGNED FOR LIFE SUPPORT APPLICATIONS.