AK-SDFS-UART

Reference manual

Contents

About this document

Revision history

The table below displays the revision history for the chapters in this manual.

Chapter	Date	Revision	Changes made
All	November 2011	1.0	First publication

Contact information

For the latest news, upgrades and information about Artekit products, visit the Artekit web site at http://www.artekit.eu
For technical support on this product, visit the support page at http://www.artekit.eu/support
For additional information about Artekit products, consult the sources below.

Information type	Resource
Technical support	support@artekit.eu
Literature	www.artekit.eu
Sales	sales@artekit.eu
Products forum	www.artekit.eu

Life support policy

Artekit Italy products are not indented or authorized for use as critical components in life support devices or systems without the express written approval from Artekit Italy. Those devices may include devices for supporting or sustaining life, devices for surgical implant into the body or any other device whose failure to perform correctly could result in life support failure.

Copyright information

Specifications

Product description

The Artekit AK-SDFS-UART board is a low-power, low-cost, small-footprint data storage board for reading and writing data to SD cards using the FAT32 and FAT16 file system. It can be operated through a RS232 serial port at different baud rates, managing concurrently up to 4 files.

The main goal of the AK-SDFS-UART device is to provide a complete file system and SD access to those boards or products that by design have not the capacity or the resources to manage a SD and a file system. You may write and read files from any board that has an available serial port. The resulting written data can be read on a standard PC because it uses the same standard file system.

You may visit the Artekit website at www.artekit.eu and download the latest examples and a library to quickly start working with the SD card.

Main components

The AK-SDFS-UART board has the following main components:

- MicroSD socket.
- ARM microcontroller running the complete file system.
- Protocol activity LED.
- Opened files indication LED.

Environmental requirements

The AK-SDFS-UART board must be stored between -40° C and +100° C. The recommended operating temperature is between 0° C and +70° C.

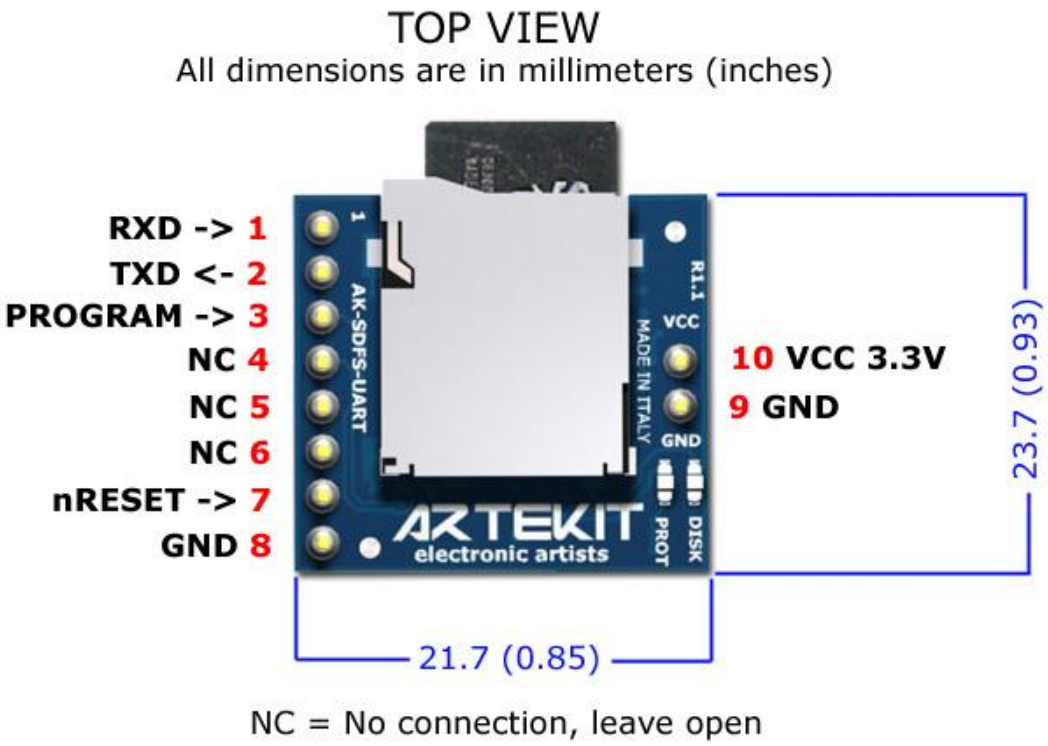
The AK-SDFS-UART board may be damaged without proper anti-static handling.

Handling the board

When handling the board, it is important to observer the following precaution:

Static discharge precaution – Without proper anti-static handling the board can be damaged. Therefore, take anti-static precautions when handling the board.

Board overview



Pins description

1	Input	UART RX, for external communication. See the Communication Protocol chapter.
2	Output	UART TX, for external communication. See the Communication Protocol chapter.
3	Input	Unconnected for normal operation. Set to logical 1 for firmware upgrade mode.
4	NC	Leave this pin unconnected.
5	NC	Leave this pin unconnected.
6	NC	Leave this pin unconnected.
7	Input	Reset. Active low. Apply a >5ms pulse to reset the board.
8	GND	Ground.
9	GND	Ground.
10	VCC	VCC 3.3V

Electrical characteristics

Test conditions

Unless specified, all voltages are referenced to GND.

Minimum and maximum values

Unless otherwise specified, the minimum and maximum values are guaranteed in the worst conditions of ambient temperature, supply voltage and frequencies by tests in production on the 100% of the devices with an ambient temperature $T_A = 25\ °C$.

Typical values

Unless otherwise specified, typical data are based on TA = 25 °C, VCC = 3.3V (for the 2V < VCC < 3.6V voltage range). They are given only as design guidelines and are not tested.

Absolute maximum ratings

WARNING Exceeding values beyond these absolute maximum values may cause permanent damage to the device and/or SD card. Operating at absolute maximum rating conditions for extended periods may affect the device and/or card reliability.

Symbol	Ratings	Min.	Max.	Unit
VCC-GND	External main supply voltage.	-0.3	4	V
Vin	Input voltage on 5V tolerant pins.	GND-0.3	+5.5	
	Input voltage on other pins.	GND-0.3	VCC+0.3	

Current consumption in operating mode

Symbol	Parameter	Max.	Unit
Icc	Supply current in RUN mode without micro SD	50	mA
	Supply current in RUN modem with micro SD	125	

Note: all signal pins are 5V tolerant.

Normal operating parameters

Symbol	Parameter	Value	Unit
Vcc	Power supply applied to VCC pin	3.3	V

Supported file systems

The AK-SDFS-UART board supports the standard FAT16 and FAT32 file systems, including long file names. The board and the communication with the SD card comply with the SDHC standard.

Communication protocol

Notations

Hexadecimal value will be shown in the format 0x00. For example the decimal number 255 can be shown as 0xFF.

All the values are little-endian.

Protocol specification

The AK-SDFS-UART board communicates with the host using a robust binary protocol with error checking. Unless other products, error checking proves to add an extra layer of security on the transport protocol, especially in noisy environments, when using TTL lines.

Packet

Each message going to and from the AK-SDFS-UART is called a packet, and for every packet there is an ACK flag that ensures the host the AK-SDFS-UART board has received the packet without any errors.

Every packet has the following format:

Preamble	Command	File handle / opt	Data length	Data	CRC16

Explanation of the packet fields

- **Preamble:** the preamble is a two bytes length code meant to detect the start of a packet. For every packet the preamble is 0x41 and 0x4B. ('AK' letters). See the example packet below, in the next section, for byte ordering.
- **Command:** one byte length code indicating the command. The list of commands is in the next chapter.
- **File handler/opt:** this field is one byte length and carries the file handler ID. When an "Open file" command is issued, the AK-SDFS-UART device returns an ID number that represents the opened file. Since a maximum of four files can be opened at the same time, this field helps to determine on which file the command is directed. For example a packet intended to write some data on the opened file with ID = 1 should have this field set to 1. In some cases this byte represents an option value (depending on the issued command, more on this in the next chapter).
- **Data length:** this field is two bytes length and is the length in bytes of the packet Data field.
- **Data:** this is the data body of the packet and its length is determined by the Data length field.
- **CRC16:** a two bytes checksum. The formula for calculating is described below.

CRC function

The following is the C function to calculate the CRC16 checksum:

```
static unsigned short fprot_crc16(void* data, unsigned long len, unsigned short partial)
{
	unsigned short crc = partial;
	unsigned long i;
	unsigned char j, c;
	unsigned char* ptr = (unsigned char*) data;

	for(i = 0; i < len; i++)
	{
		c = ((crc>>8) & 0x00FF);
		c ^= *ptr++;
		crc = ((unsigned short) c << 8) | (crc & 0x00FF);
		for (j = 0; j < 8; j++) {
			if (crc & 0x8000) {
				crc <<= 1;
				crc ^= 0x1021;
			} else {
				crc <<= 1;
			}
		}
	}

	return crc;
}
```

Example of a packet

Being the following the packet format:

Preamble	Command	File handle / opt	Data length	Data	CRC16

A resulting packet for opening the "\File.txt" could be:

0x41 'A'	0x4B 'K'	0x01	0	0	10	"\File.txt"+0x00		

Communication flow

The communication is always started by the host: the AK-SDFS-UART board will never send spontaneous packets.

For every sent command there is a packet with the response (plus ACK) or a packet containing an error code: every packet sent by host is acknowledged with a packet containing the ACK flag (0x80), that is, a response packet from the AK-SDFS-UART board will contain in the *Command* field the original command sent by the host plus 0x80.

An example Open File command sequence is shown below:

- To open a file the host sends:

0x41 'A'	0x4B 'K'	0x01	0	0	10	"\File.txt" + 0x00		

- And the AK-SDFS-UART board ACKs the packet with:

0x41 'A'	0x4B 'K'	0x81	1	0	0			

Note that the response to the Open File command is included (the *File handle / opt* field).

The AK-SDFS-UART device may return error codes as a response to commands. For example, when trying to open a file when there is no card in the SD slot.

Commands listing

The following list will enumerate the codes that can be sent in the packet Command field, the values of the File Handle/Opt field and the format of the Data field (if there is data to be sent). The packet Data length field must be set accordingly to the length of the Data field, including the null-termination byte for strings.

The returned error codes are not listed here but in the Error handling chapter.

Quick overview

Command	Code	Description
Open file	0x01	Opens a file on the SD
Close file	0x02	Closes a file
Read	0x03	Reads up to 512 bytes of data
Read line	0x04	Reads a line of text (up to 512 bytes)
Write	0x05	Writes up to 512 of data
Flush	0x06	Flushes unsaved data into the file
File info	0x07	Returns the status of the length and the position pointer.
Seek	0x08	Moves the position pointer to a fixed position
Delete	0x09	Deletes a file
Create directory	0x0A	Creates a directory
List directory	0x0B	Lists the contents of a directory
Check for SD	0x0C	Returns the status of the SD slot (if there is a card present)
Fat information	0x0D	Returns the state of the file system (free space/total space)
Get status	0x0E	Returns the count of the opened files and the maximum quantity
Set baud rate	0x0F	Sets the baud rate
Close all	0x10	Closes all opened files
Save parameters	0x11	Saves the parameters permanently
Set Date/Time	0x12	Sets the current date and time.
Error	0x7F	The AK-SDFS-UART device declares that an error has occurred

Open File command

Send this command to open a file. The packet format is the following:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x01	Open mode		File name	

Fields

- Open mode: a combination of the following values:
  - 0x01 = Read mode. Opens the file in read mode.
  - 0x02 = Write mode. Opens the file in write mode.
  - 0x04 = Create new. If the file does not exist, it will be created. Otherwise an error code is returned.
  - 0x08 = Create always. Creates the file whenever the file exists or not.

  For example, the value 0x0B can be used to create a file in read/write mode.

- File name: a null-terminated string indicating the file to be opened. Remember the paths are not relative (i.e. a full path must be used) and every path must be preceded by a slash ("\"). For example a valid file name can be "\Diretory1\log.txt".

On success, the AK-SDFS-UART device will answer with the following packet:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x81	File handle			

Fields

- File handle: a number representing an opened file. This number can be used with the rest of the file commands.

Close File command

Send this command to close a file. The packet format is the following:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x02	File handle			

Fields

- File handle: a number representing an opened file (with the Open File command).

On success, the AK-SDFS-UART device will answer with the following packet:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x82	File handle			

Fields

- File handle: the recently closed file handle.

Read command

Send this command to read up to 512 bytes at a time. The packet format is the following:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x03	File handle	2 bytes	Quantity of data	

Fields

- File handle: a number representing an opened file.
- Data: two bytes indicating the quantity of data to be read (1-512).

On success, the AK-SDFS-UART device will answer with the following packet:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x83	File handle	Max. 512 bytes	Data	

Fields

- File handle: the handle of the file on which the read operation was performed.
- Data length: the quantity of bytes returned for the read operation. If this number is lesser than the quantity specified in the Read file command, it means the End Of File was reached. Use the File Info command to check the status of the file pointer and the length of the file.
- Data: the read data. The length is specified in the Data length field.

Read Line command

Send this command to read a line of text (up to 512 bytes at a time). The AK-SDFS-UART board will read the file until a CR LF sequence is found or the 512 bytes limit or the quantity of data value is reached.

The packet format is the following:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x04	File handle	2 bytes	Quantity of data	

Fields

- File handle: a number representing an opened file (with the Open File command).
- Data: two bytes indicating the quantity of data to be read (1-512).

On success, the AK-SDFS-UART device will answer with the following packet:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x84	File handle	Max. 512 bytes	Data	

Fields

- File handle: the handle of the file on which the read operation was performed.
- Data length: the quantity of bytes returned for the read operation. If this number is lesser than the quantity specified in the Read file command, it means the End Of File or End Of Line was reached. Use the File Info command to check the status of the file pointer and the length of the file.
- Data: the read data. The length is specified in the Data length field.

Write command

Send this command to write up to 512 bytes at a time. The packet format is the following:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x05	File handle	Data length	Data	

Fields

- File handle: a number representing an opened file.
- Data length: the quantity of bytes to be written (1-512).
- Data: the data to be written.

On success, the AK-SDFS-UART device will answer with the following packet:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x85	File handle	2 bytes	Quantity of data	

Fields

- File handle: the handle of the file on which the write operation was performed.
- Data: two bytes indicating the quantity of data actually written to the file.

Remarks: the data may not be actually written to the file for every Write file command issued. This is because the AK-SDFS-UART board may temporary buffer the incoming data into RAM (to speed up the write process). It is recommendable to issue the File Flush command from time to time to avoid data loss. Also closing the file with the File Close command ensures no data loss. Removing the card between or while performing writing operations may corrupt the file and/or the file system (as with any other data media/file system).

File Flush command

Send this command to flush the unsaved data into the file system. The packet format is the following:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x06	File handle			

Fields

- File handle: a number representing an opened file (with the Open File command).

Note: see the remarks section for the Write File command.

On success, the AK-SDFS-UART device will answer with the following packet:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x86	File handle			

Fields

- File handle: the handle of the file on which the flush operation was performed.

File Info command

Send this command to retrieve the length of an opened file and the current pointer position within the file.

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x07	File handle			

Fields

- File handle: a number representing an opened file (with the Open File command).

On success, the AK-SDFS-UART device will answer with the following packet:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x87	File handle	8 bytes	File information	

Fields

- File information: 8 bytes. The first four bytes are the position of the pointer within the file. The last four bytes indicates the current file length.

File Seek command

Send this command to move the pointer within an opened file.

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x08	File handle	4 bytes	New position	

Fields

- File handle: a number representing an opened file (with the Open File command).
- New position: new position for the pointer, in bytes.

Remarks: If the pointer is set beyond the current file length, the file is expanded to reach the new pointer position. It is useful to quickly create large files. The contents of the new expanded length may contain garbage. That's it; the file is not filled with any fixed value.

On success, the AK-SDFS-UART device will answer with the following packet:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x88	File handle	4 bytes	New position	

Fields

- New position: new position for the pointer, in bytes.

File Delete command

Send this command to delete files or directories. The file or directory must exist and must not be in use; otherwise the AK-SDFS-UART device will return an error packet. When deleting a directory, it must be empty.

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x09		Max. 512 bytes	Path	

Fields

- Data length: length in bytes of the Data field.
- Path: null-terminated string of the full path of the file or directory.

On success, the AK-SDFS-UART device will answer with the following packet:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x89				

Create Directory command

Send this command create a new directory

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x0A		Max. 512 bytes	Directory path	

Fields

- Data length: length in bytes of the Data field.
- Directory path: null-terminated string of the full path of the directory to be created.

On success, the AK-SDFS-UART device will answer with the following packet:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x8A				

Dir command

Send this command to start or continue a directory list sequence.

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x0B		Data length	Path	

Fields

- Data length: length in bytes of the Data field.
- Path: null-terminated string of the full path.

Remarks: use this command to retrieve a list of files and directories from the specified path. For every Dir command issued the AK-SDFS-UART device will send back a packet containing **one** directory or file name. You can retrieve

the complete list of files and directories in the specified path by sending successive Dir commands. The AK-SDFS-UART device will answer with a Dir command packet (shown below) with the file or directory name contained in the Data field. If there are no more files or directories in the path, the AK-SDFS-UART device will answer with the same command packet but with the Data length field set to zero and with an empty Data field.

To reset the listing sequence and start over again, send the same packet but will *Data length* field set to zero and without *Data* field.

On success, the AK-SDFS-UART device will answer this command with the following packet:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x8B		Data length	File/Directory Name	

Fields

- Data length: length in bytes of the Data field. Zero if there are no more files/directories to list.
- File/Directory name: null-terminated string containing the name of a file or directory. A directory can be differentiated from a file because it is enclosed between the <> symbols, for example “<directory>”. If there are no more files/directories to list, this field will be empty.

Check Card Presence command

Send this command to retrieve the slot status (card presence/absence).

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x0C				

Fields

On success, the AK-SDFS-UART device will answer with the following packet:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x8C		1 byte	Status	

Fields

- Status: one byte. If 1 the card is present. If zero there is no card in the SD slot.

FAT Information command

Send this command to retrieve the total file system space and the current free space.

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x0D				

On success, the AK-SDFS-UART device will answer with the following packet:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x8D		8 bytes	FAT information	

Fields

- FAT information: 8 bytes. The first 4 bytes represents the total capacity (in bytes) of the SD card partition. The last 4 bytes indicates the current available free space.

Status command

Send this command to retrieve the count of the currently opened files

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x0E				

On success, the AK-SDFS-UART device will answer with the following packet:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x8E		2 bytes	Status	

Fields

- Status: 2 bytes. The first byte is the maximum quantity of opened files. The second byte is the quantity of currently opened files.

Set Baud Rate command

Use this command to set the AK-SDFS-UART device baud rate.

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x0F		4 bytes	Baud rate	

Fields

- Baud rate: a 32 bit value representing the baud rate. It can be one of the following values:
  - 1200
  - 2400
  - 4800
  - 9600
  - 19200
  - 38400
  - 57600
  - 115200

On success, the AK-SDFS-UART device will answer with the following packet:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x8F				

Note: the answer will be given using the old baud rate. After the response is sent, all the successive commands will be sent using the new baud rate.

Close All Files command

Send this command to close all the opened files. Use this command to ensure there are no opened files after a (wanted or not) host reset.

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x10				

On success, the AK-SDFS-UART device will answer with the following packet:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x90				

Save Parameters command

Send this command to save permanently the baud rate configuration into the internal flash memory. This way, the next time the AK-SDFS-UART device is reset; it will use the configured baud rate (with the Set Baud Rate command).

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x11				

On success, the AK-SDFS-UART device will answer with the following packet:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x91				

Set Date/Time command

Use this command to set the current date and time into the AK-SDFS-UART board. This date and time will be written to the file system when creating of modifying files.

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x12		6 bytes	Date and time	

Fields

- Date and time: six bytes with the following format:
  - 1st byte: year minus 2000 (for example 11 for 2011).
  - 2nd byte: month.
  - 3rd byte: day.
  - 4th byte: hour.
  - 5th byte: minutes.
  - 6th byte: seconds.

On success, the AK-SDFS-UART device will answer with the following packet:

Preamble	Command	File handle / opt	Data length	Data	CRC16
	0x92				

Error codes

The following is the list of the possible error codes the AK-SDFS-UART device may answer in the presence of an error. The values described here may be found in the *File handle / opt* field when the *Command* field is 0x7F (Error command).

Value	Error	Cause
1	No more files	When trying to open more than 4 files concurrently
2	File not found	When trying an operation on a non existing file or directory
3	Invalid handle	The provided file handle is invalid
4	No file system	The file system cannot be found
5	Internal error	Internal error in the AK-SDFS-UART device
6	Disk error	Low level IO error
7	Disk not ready	Physical driver not ready
8	Timeout	The current operation resulted in timeout
9	Invalid name	The provided file or directory name is invalid
10	Path not found	The provided path cannot be found
11	Write-protected	The physical drive is write-protected (not implemented)
12	Already exists	The file or directory already exists
13	Locked	The file or directory is locked
14	Denied	Access denied
15	Invalid length	The length for the current operation is invalid (0 or greater than 512)
16	Packet error	A error was detected in the received packet (bad length or parameter)
17	No card	There is not a SD card on the reader
18	Invalid parameters	One or more parameters of the packet are invalid
19	Unknown command	The receive command code is unknown

Communications Protocol library

Description

With the scope of making easy the integration of the AK-SDFS-UART device into existing or new projects, Artekit makes available for download a Communication Protocol library, written in C language, with full source code.

You may download the library from the Artekit website at http://www.artekit.eu

The library may be used in commercial and/or non-commercial products, and redistributed in any way as long as the copyright information in the header of each file is maintained.

The support for this library is limited and is released in good faith. Artekit Italy should not be held liable for any direct, indirect, punitive damages or any damages associated with use of this library or code.

Usage

The library is contained in two files: fprot.c and fprot.h files.

The following is a list of the available functions in the library:

```
unsigned char fprot_init(txdatafunc* tx, rxdatafunc* rx, delayfunc* delay);
unsigned char fprot_open(char* file, unsigned char options, FPROT_FILE* handle);
unsigned char fprot_close(FPROT_FILE file);
unsigned char fprot_read(FPROT_FILE file, void* buf, unsigned long qty, unsigned long* read);
unsigned char fprot_read_line(FPROT_FILE file, void* buf, unsigned short qty, unsigned short* read);
unsigned char fprot_write(FPROT_FILE file, void* buf, unsigned long qty, unsigned long* written);
unsigned char fprot_flush(FPROT_FILE file);
unsigned char fprot_delete(char* file);
unsigned char fprot_check_card(void);
unsigned char fprot_file_info(FPROT_FILE file, unsigned long* pos, unsigned long* size);
unsigned char fprot_seek(FPROT_FILE file, unsigned long pos, unsigned long* new_pos);
unsigned char fprot_fat_info(unsigned long* capacity, unsigned long* free);
unsigned char fprot_status(unsigned char* max, unsigned char* used);
unsigned char fprot_mkdir(char* path);
unsigned char fprot_dir(char* path, char* entry, unsigned short entrylen);
unsigned char fprot_close_all(void);
unsigned char fprot_set_baudrate(unsigned long baudrate);
unsigned char fprot_save_parameters(void);
unsigned char fprot_set_time(unsigned char day, unsigned char month, unsigned char year,
                             unsigned char hour, unsigned char min, unsigned char seconds);

unsigned char fprot_get_last_error(void);
```

To start using the library call the **fprot_init()** function and pass the 3 parameters as indicated. These 3 parameters are pointer to functions: since this library can be used on many platforms and MCUs, the transmission, reception and delay functions are declared internally as pointer to functions, that you may declare in your program.

- The **txdatafunc** type is a function pointer used for the transmission of characters.
- The **rxdatafunc** type is a function pointer used for the reception of characters.
- The **delayfunc** type is a function pointer used to delay the reception and to detect timeout.

Here is the definition of these 3 types:

```
typedef void (txdatafunc)(unsigned char*, unsigned long);
typedef unsigned long (rxdatafunc)(unsigned char*, unsigned long);
typedef void (delayfunc)(unsigned long);
```

You may define these functions in your code. The following is an example of a typical initialization:

```
#include "fprot.h"

/*
 * This function is used to transmit characters through an UART.
 * Note that the uart_put_char() function is an example function and may
 * not be available in your code.
 *
 * Parameters:
 * unsigned char* data        Pointer to an array of bytes to transmit.
 * unsigned long data_len     The quantity of bytes to send.
 */
void my_tx_function(unsigned char* data, unsigned long data_len)
{
        unsigned long i;

        for (i = 0; i < data_len; i++)
        {
                uart_put_char(data[i]);
        }
}

/*
 * This function is used to receive characters from an UART.
 * Note that the uart_get_char() function is an example function and may
 * not be available in your code.
 *
 * Parameters:
 * unsigned char* data        Pointer to an array of bytes to receive the data.
 * unsigned long data_len     The quantity of bytes to receive.
 *
 * This function must return the quantity of bytes read.
 */
unsigned long my_rx_function(unsigned char* data, unsigned long data_len)
{
        unsigned long read = 0;

        while (read < data_len)
        {
                data[read] = uart_get_char();
                read++;
        }

        return read;
}

/*
 * This function is used to perform a delay in milliseconds.
 * Note that the delay() function is an example function and may
 * not be available in your code.
 *
 * Parameters:
 * unsigned long ms    The quantity of milliseconds to delay.
 */
void my_delay_function(unsigned long ms)
{
        delay(ms);
}

/* This function is an example of the main entry point of your program. */
int main(void)
{
        /*
         * Your initialization here
         */

        /* Call fprot_init() with the three functions declared above */
        fprot_init(my_tx_function, my_rx_function, my_delay_function);

        /*
         * Your program here
         */

        return 0;
}
```

After calling **fprot_init()** you can use any other function of the library.

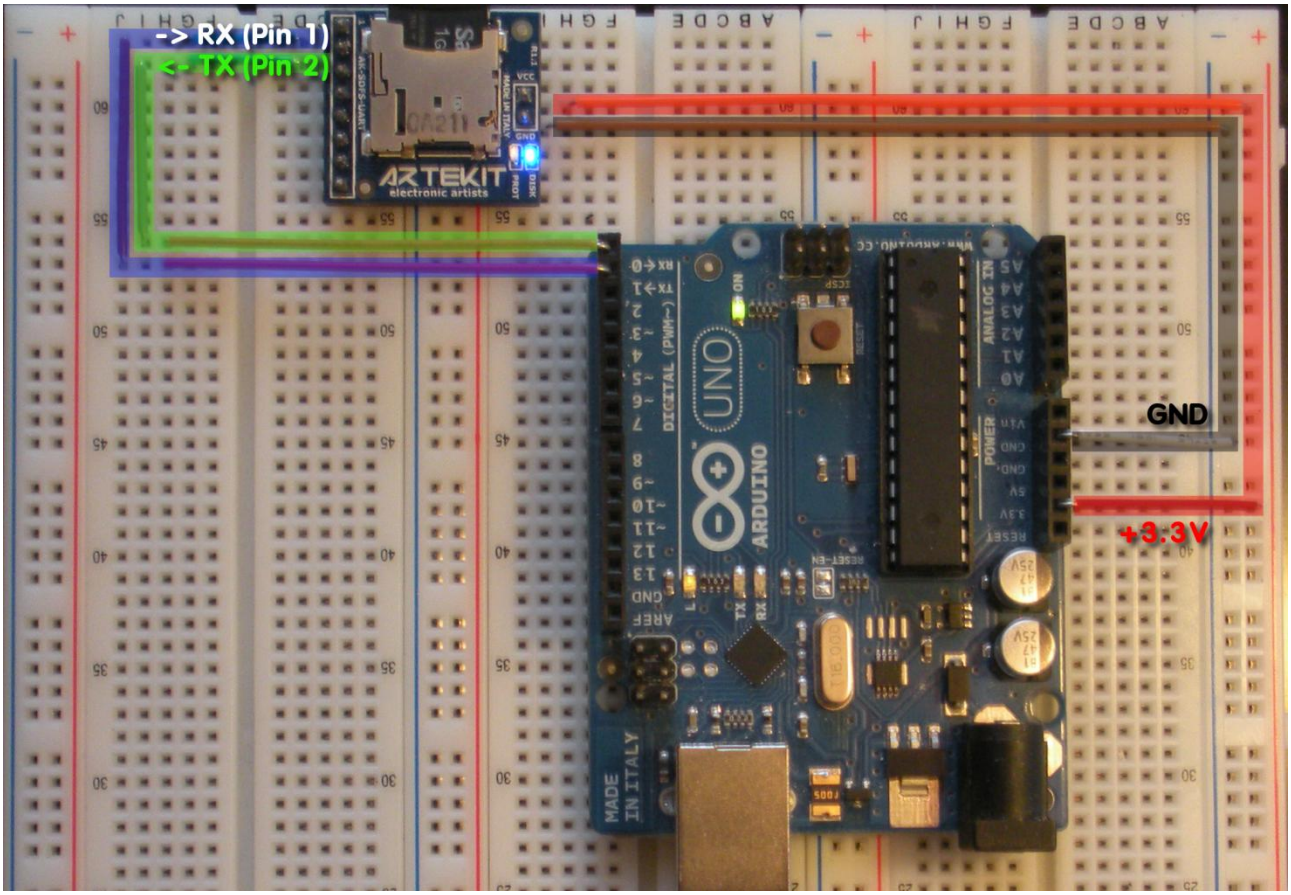
You can customize the timeout settings by modifying the `FPROT_TIMEOUT_TIME` constant in the fprot.h file.

The timeout is the time in milliseconds the library should wait when receiving characters from the UART. If the time between characters exceeds the `FPROT_TIMEOUT_TIME` value, then a timeout is declared, and the reading is stopped.

Using the AK-SDFS-UART with an Arduino board

The AK-SDFS-UART can be easily connected to a wide variety of boards since it requires only four wires.

The next image shows an AK-SDFS-UART device actually working with an Arduino board.



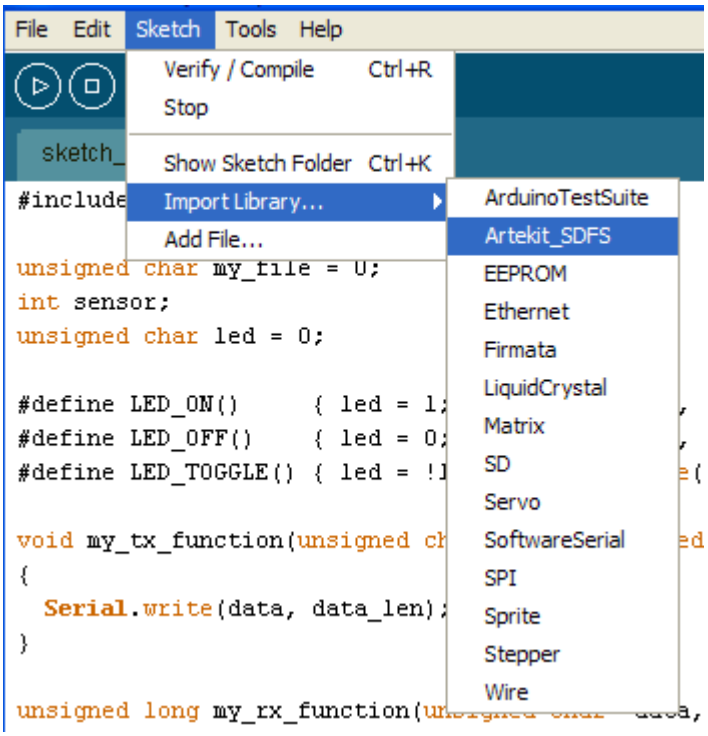
Connections

AK-SDFS-UART	Arduino UNO
Pin 1 (RX)	Pin 1 (TX)
Pin 2 (TX)	Pin 0 (RX)
Pin 10 (VCC)	3.3V
Pin 9 (GND)	GND

Using the AK-SDFS-UART library with the Arduino environment

The provided library can be easily used with the current Arduino development environment. Just create a “Artekit_SDFS” folder inside the Arduino libraries folder and copy fprot.c and fprot.h. Then rename fprot.c to fprot.cpp (so you can use it as a library). You can download the library directly from the Artekit website that includes the folder with all the files.

After this you can use the library with your Arduino projects. Remember to include the library by selecting the Sketch menu and then “Import Library…” and selecting “Artekit_SDFS”.



Example sketch

The following is an example sketch (can be downloaded from the Artekit website) that saves the value of an analogical input into a file called “log.txt” in the SD.

```
#include <fprot.h>

unsigned char my_file = 0;
int sensor;
unsigned char led = 0;

#define LED_ON()     { led = 1; digitalWrite(13, led);      }
#define LED_OFF()    { led = 0; digitalWrite(13, led);      }
#define LED_TOGGLE() { led = !led;  digitalWrite(13, led);  }

void my_tx_function(unsigned char* data, unsigned long data_len)
{
  Serial.write(data, data_len);
}

unsigned long my_rx_function(unsigned char* data, unsigned long data_len)
{
  unsigned long read = 0;

  while (read < data_len)
  {
    if (Serial.available() > 0) {
      data[read++] = Serial.read();
    }
  }

  return read;
}

void my_delay_function(unsigned long ms)
{
  delay(ms);
}

void setup()
{

  pinMode(13, OUTPUT);

  Serial.begin(9600);

  if (fprot_init(my_tx_function, my_rx_function, my_delay_function))
  {
    // AK-SDFS-UART library initialized

    // Close all files that may be opened before reset
    fprot_close_all();

    // Open a file called log.txt
    if (fprot_open("\\log.txt", FPROT_MODE_RW | FPROT_MODE_CREATE_ALWAYS, &my_file) !=
FPROT_NO_ERROR ||
        my_file == FPROT_INVALID_FILE)
    {

      // Indicate failure keeping the LED ON
      LED_ON();
      my_file = 0;
    }
  }
}

void loop()
{
  char str[32];
  unsigned long written;

  // If the file was opened
  if (my_file)
  {
```

```
    // Read a sensor
    sensor = analogRead(A0)/4;

    // Create a string to write
    sprintf(str, "Analog=%i\r\n", sensor);

    // Write the file with the str contents
    if (fprot_write(my_file, str, strlen(str), &written) == FPROT_NO_ERROR)
    {

      // If fprot_write() functions succeeds, flush the file to save
      // the data permanently
      fprot_flush(my_file);

      // Toggle the LED to indicate activity
      LED_TOGGLE();

    } else {

      // On failure, keep the led ON and set my_file = 0
      LED_ON();
      my_file = 0;

    }
  }

  // Do the above each second
  delay(1000);
}
```

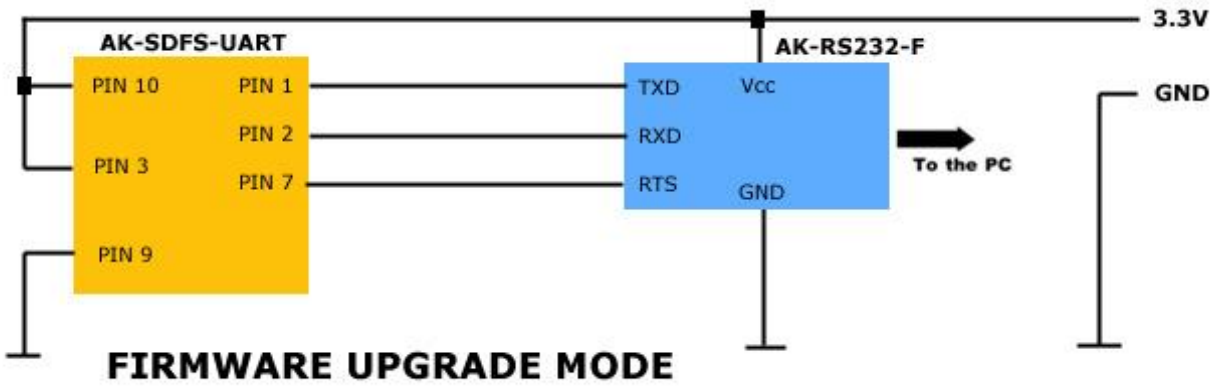
Operation modes

Firmware upgrade mode

It is possible to update the internal AK-SDFS-UART firmware using the application provided by Artekit. You can freely download this application from the Artekit website.

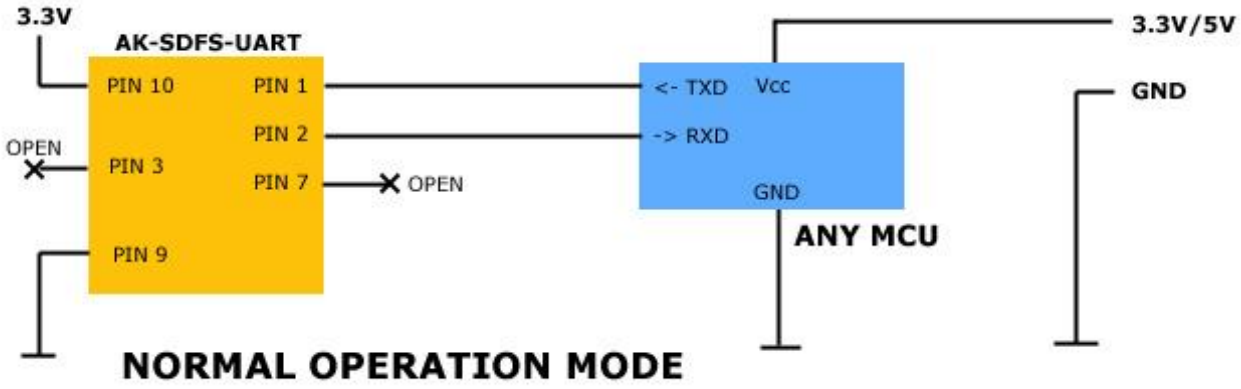
To connect the PC to the AK-SDFS-UART device is necessary to use a RS232 to TTL adapter, as the AK-RS232-F from Artekit or similar. You may construct your own adapter using a MAX3232 or similar chip. In the next sample we will use the AK-RS232-F adapter.

WARNING Connecting the AK-SDFS-UART device directly to a PC serial port will damage the board. Use a RS232 level translator (the AK-SDFS-UART communication port lines are TTL).



Normal operation mode

The next figure shows the normal operating circuitry at TTL levels. The AK-SDFS-UART device can be connected to any MCU using a standard UART communication port.



NOTES: