

HBBRFS FAT File System Module

Version 2.000

Copyright

This document is Copyright © 2008 by Hobby-Robotics, LLC

All trademarks within this guide belong to their legitimate owners.

Table of Contents

Copyright.....	2
Introduction.....	4
HBBRFS File System Module Specification.....	4
Supported cards.....	4
Supported FAT types	4
Supported MCUs.....	4
Limitations	4
Using HBBRFS.....	5
Initialization.....	5
Managing file systems.....	5
Using file system.....	7
Paths and current directory.....	7
Reading, Writing to and from file.....	7
SPI Configuration.....	8
AT91SAM7S and AT91SAM7X.....	9
LPC2000.....	10
HBBRFS File System Module API.....	11
Functions.....	11
Constants.....	11
Errors.....	12
ChDir.....	12
ChDrive.....	12
CurDir.....	13
FileClose.....	13
FileGet.....	13
FileLen.....	14
FileOpen.....	14
FilePut.....	15
FileSeek.....	15
FreeFile.....	16
HBBRFSInit	16
Kill	16
Sub MkDir	16
Mount	17
Rename	17
Rmdir	17
Umount	18
Errors.....	18

Introduction

HBBRFS is an HBBR Basic extension module providing FAT file system support for MMC and SD cards.

HBBRFS File System Module Specification

Supported cards

MMC maximum capacity 1GB

SD maximum capacity 1GB

Supported FAT types

FAT12

FAT16

FAT32

Supported MCUs

AT91SAM7S - AT91SAM7S256, AT91SAM7S128, AT91SAM7S64

AT91SAM7X - AT91SAM7X256, AT91SAM7X128

LPC21xx - LPC2104, LPC2105, LPC2106, LPC2138, LPC2136, LPC2134, LPC2148, LPC2146, LPC2144

Limitations

Card access in SPI mode.

File names in 8.3 format (DOS file-naming rule), Long File Names (LFN) are not supported.

Maximum number of open files 4.

Maximum path length is 256 characters for absolute path.

Maximum mount path length is 13.

Using HBBRFS

Initialization

HBBRFS module depends on hardware based delay generation which has to be enabled and initialized prior to initializing the module. Delay generation details are described in the runtime documentation. Delay initialization is done by calling `__hbbr_delay_init` function, return value greater than 0 indicates successful initialization while 0 means it has failed.

Example:

```
res = __hbbr_delay_init()
If res > 0 Then
    ' continue
End If
```

Once the delay is initialized HBBRFS files system module itself needs to be initialized by calling `HBBRFSInit` function, return value greater than 0 indicates successful initialization while 0 mean it has failed. In case of initialization failure the module error value is set to the error code which can be retrieved by calling `__hbbr_mod_get_last_error` function. Error codes are described in the API reference.

Example:

```
res = HBBRFSInit()
If res > 0 Then
    ' successful
Else
    ' failed
    error = __hbbr_mod_get_last_error()
End If
Call __hbbr_enable_irq()
```

Managing file systems

After initializing SPI peripheral and IO pins HBBRFS module is ready to access file system on the SD/MMC card. The first step is to mount the file system by calling **Mount** function. It takes two

arguments, first one is numeric device id identifying SPI peripheral, the second argument is a mount path used to associated it with the file system. Mount path has to include device id followed by the colon. In case of mount failure the module error value is set to the error code which can be retrieved by calling `__hbbr_mod_get_last_error` function. Error codes are described in the API reference.

Example:

```
dev = 0
mpath = "0:"
res = Mount(dev, mpath)
If res > 0 Then
    ' successful
Else
    ' failed
    error = __hbbr_mod_get_last_error()
End If
```

HBBRFS supports multiple mounted file systems simultaneously and allows switching between them using ChDrive sub.

Example:

```
dev = 0
mpath = "0:"
res = Mount(dev, mpath)
Call ChDrive(mpath)
' check for errors
error = __hbbr_mod_get_last_error()
```

File system should be unmounted before removing SD/MMC card from the socket. Unmounting files system prevents corrupting data on the card.

Example:

```
dev = 0
mpath = "0:"
res = Umount(dev, mpath)
If res > 0 Then
    ' successful
Else
    ' failed
    error = __hbbr_mod_get_last_error()
End If
```

Using file system

Once the files system has been mounted it can be accessed to traverse directories, create, remove files and directories, read, write text and binary data from files.

Following functions and subs are used to access directories and files.

ChDir

MkDir

RmDir

CurDir

Kill

Rename

Paths and current directory

HBBBRFS maintains current directory which allows to access files using relative paths.

Reading, Writing to and from file

Files are accessed using file number associated with the specific file by using **FileOpen** sub which must be called before attempting any access to the files data. Correspondingly file can be disconnected from the file number by calling **FileClose** sub.

File can be accessed in text or binary mode specified when calling **FileIOpen**.

In the text mode (ASCII text file) data can be read and written with built-in keywords **Print**, **Write**, **Input**, **Line Input**.

In the binary mode data can be read and written either with built-in keywords **Put** and **Get** as well as **FilePut** and **FileGet** subs.

SPI Configuration

HBBRFS is using existing on-chip SPI peripheral to access MMC/SD cards. Specific SPI peripheral is selected through Mount function Device parameter. When Mount function is called the SPI peripheral is initialized as SPI master and configured to access MMC/SD cards. Depending on the MCU and selected SPI peripheral following pins are used.

SPI signal	LPC2xxx - SPI0 Device = 0	LPC2xxx – SPI1 Device = 1	AT91SAM7S - SPI Device=0	AT91SAM7X - SPI0 Device=0	AT91SAM7X - SPI1 Device=1
MISO Input	MISO0 - P0.5	MISO1 - P0.18	MISO - PA12	MISO0 - PA16	MISO1 - PA24
MOSI Output	MOSI0 - P0.6	MOSI1 - P0.19	MOSI - PA13	MOSI0 - PA17	MOSI1 - PA23
SCK Output	SCK0 - P0.4	SCK1 - P0.17	SPCK - PA14	SPCK0 - PA18	SPCK1 - PA22
Card Select Output	User configured	User configured	User configured	User configured	User configured

SPI MISO, MOSI and SCK signals are hardware controlled.

SPI Card Select (Chip Select) is software controlled and can be configured by the user. It is a IOPIN_0 parameter in in the hbbfrfs_XXXXXX module.

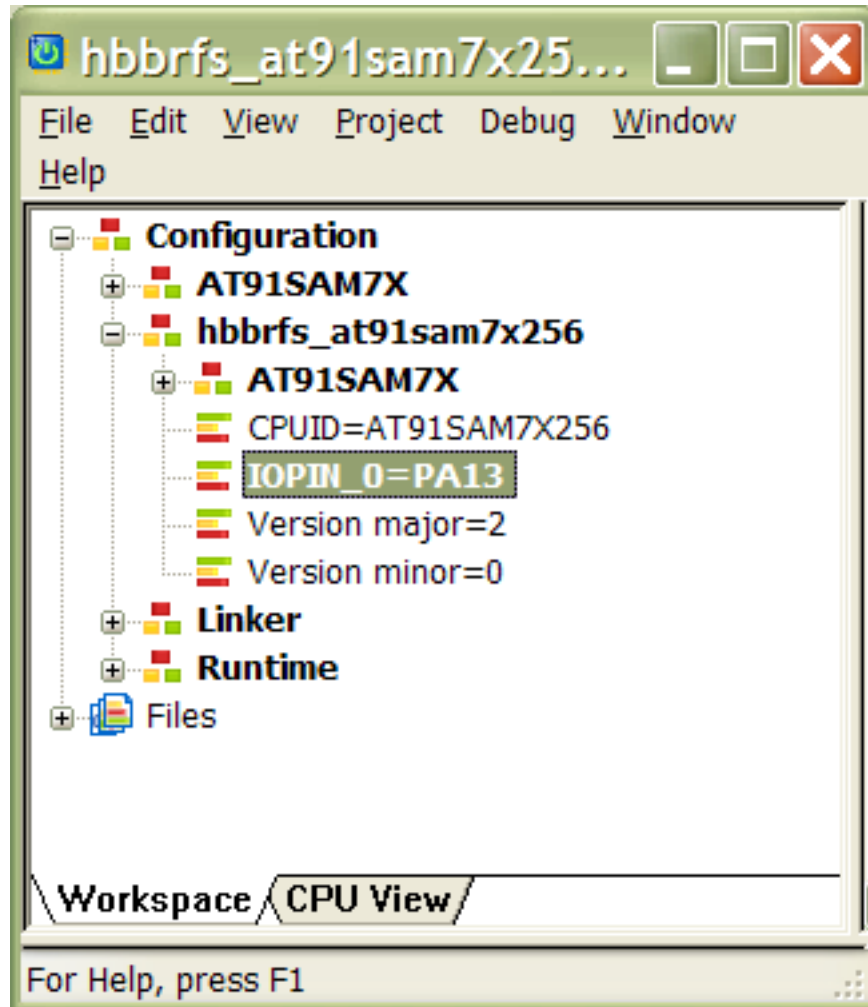
AT91SAM7S and AT91SAM7X

To configure Card Select signal follow these steps:

- in Workspace Tab select Configuration->hbbrfs_at91sam7xxx->IOPIN_0
- do right click on it to bring context menu and select Edit
- choose or type appropriate pin name for your board, the format is PAn where n is number 0 to 31

Example:

Atmel's AT91SAM7X256-EK board is using PA13 to drive Card Select signal in SD/MMC socket.



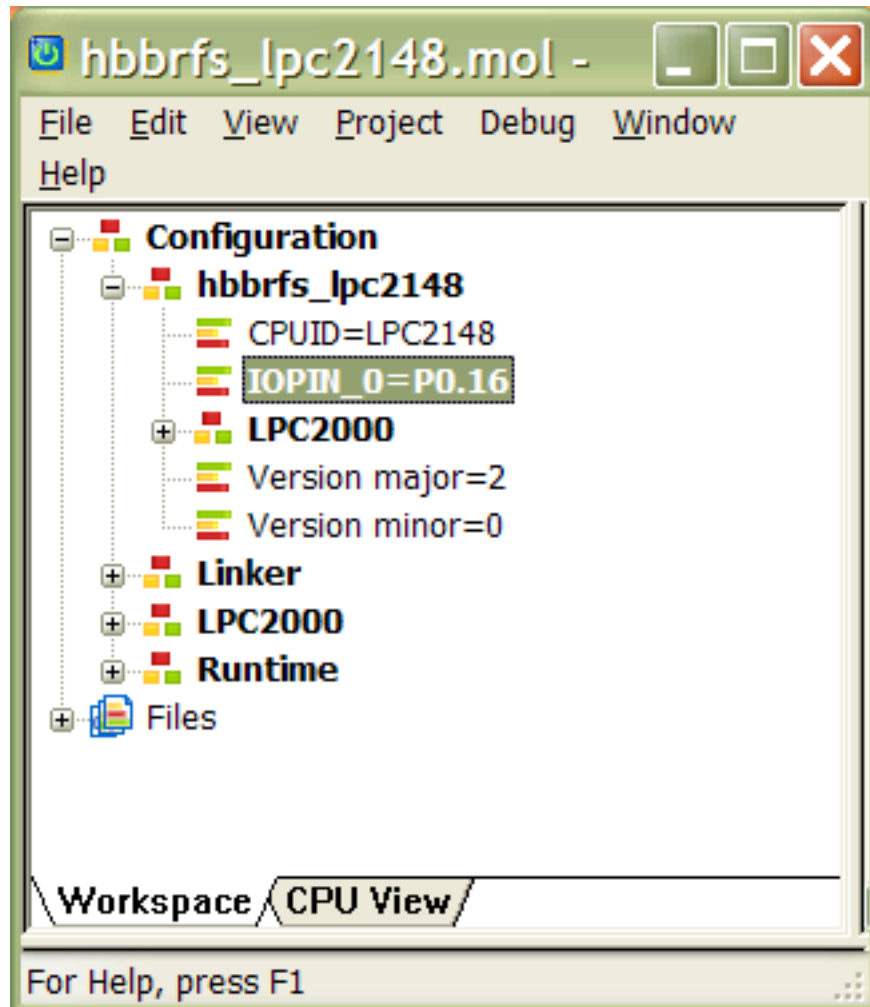
LPC2000

To configure Card Select signal follow these steps:

- in Workspace Tab select Configuration->hbbrfs_lpc2xxx->IOPIN_0
- do right click on it to bring context menu and select Edit
- choose or type appropriate pin name for your board, the format is P0.n where n is number 0 to 31

Example:

HBRR IH1 board is using P0.16 to drive Card Select signal in SD/MMC socket.



HBBRFS File System Module API

Functions

Function HBBRFSInit ()

Function Mount (ByVal Device As Integer, ByRef Path As String) As Integer

Function Umount (ByVal Device As Integer, ByRef Path As String) As Integer

Sub ChDrive (ByRef Drive As String)

Function FreeFile ()

Sub FileOpen (ByVal FileNumber As Integer, ByRef FileName As String, ByVal Mode As Integer)

Sub FileClose (ByVal FileNumber As Integer)

Function FileLen (ByRef PathName As String) As UInteger

Sub FileSeek (ByVal FileNumber As Integer, ByVal Position As UInteger)

Sub ChDir (ByRef Path As String)

Sub Mkdir (ByRef Path As String)

Sub Rmdir (ByRef Path As String)

Function CurDir ()

Sub Kill (ByRef Path As String)

Sub Rename (ByRef OldPath As String, ByRef NewPath As String)

Sub FilePut (ByVal FileNumber As Integer, ByRef Value As Type)

Sub FileGet (ByVal FileNumber As Integer, ByRef Value As Type)

Constants

CONST MODE = &h00000001

CONST MODE_INPUT = &h00000001

CONST MODE_OUTPUT = &h00000002

CONST MODE_RANDOM = &h00000003

CONST MODE_APPEND = &h00000004

CONST MODE_BINARY = &h00000008

CONST ACCESS

CONST ACCES_READ

CONST ACCES_WRITE

CONST ACCES_READWRITE

Errors

CONST ERROR_HBBRFS = -1
CONST ERROR_HBBRFS_CONFIG = -2
CONST ERROR_HBBRFS_BAD_DEVICE_ID = -32
CONST ERROR_HBBRFS_BAD_MOUNT_PATH = -33
CONST ERROR_HBBRFS_NO_FREE_FILE = -34
CONST ERROR_HBBRFS_BAD_FILE_NUM = -35
CONST ERROR_HBBRFS_BAD_FILE_MODE = -36
CONST ERROR_HBBRFS_BAD_FILE_ACCESS = -37
CONST ERROR_HBBRFS_BAD_FILE_LOCK = -38
CONST ERROR_HBBRFS_FILE_ALREADY_OPEN = -39
CONST ERROR_HBBRFS_DIR_DOES_NOT_EXIST = -40
CONST ERROR_HBBRFS_FILE_DOES_NOT_EXIST = -41
CONST ERROR_HBBRFS_DIR_HAS_FILES = -42
CONST ERROR_HBBRFS_FILE_WRITE_FAILED = -43
CONST ERROR_HBBRFS_FILE_READ_FAILED = -44

ChDir

Sub ChDir (ByRef Path As String)

Change current directory to the specified path.

Parameters:

[in] Path String

ChDrive

Sub ChDrive (ByRef Drive As String)

Change current drive.

Parameters:

[in] Drive String

CurDir

Function CurDir() As String

Returns path to the current directory.

Parameters:

none

Returns:

String with path to the current directory

FileClose

Sub FileClose(ByVal FileNumber As Integer)

Close file associated with the file number.

Parameters:

[in] FileNumber Integer

FileGet

Sub FileGet(ByVal FileNumber As Integer, ByRef Value As Type)

Read binary value from a file into Value.

Parameters:

[in] FileNumber Integer

[in] Value Type

FileLen

Function FileLen(ByRef PathName As String) As UInteger

Returns file length specified by PathName.

Parameters:

[in] PathName String

Returns:

UInteger value size of the file

FileOpen

Sub FileOpen(ByVal FileNumber As Integer, ByRef FileName As String, ByVal Mode As Integer)

Open file for access in specified mode.

Parameters:

[in] FileNumber Integer

[in] FileName String

[in] Mode Integer

Mode - Append, Input, Output, Random, Binary

Append - default

Input - input read access only

Output - output write access only

Random - random access

Binary - binary access

Append, Input, Output used for text files.

Random used for random access to file.
Binary used for binary access to file

CONST ACCES_READ - Read

CONST ACCES_READWRITE - Read and Write

CONST ACCES_WRITE - Write

CONST ACCESS - Flag file permission

FilePut

Sub FilePut(ByVal FileNumber As Integer, ByVal Value As Type)

Write binary value to a file.

Parameters:

[in] FileNumber Integer

[in] Value Type

FileSeek

Sub FileSeek(ByVal FileNumber As Integer, ByVal Position As UInteger)

Set current position in the file.

Starting byte position in a file is 1, second one is 2 and so on up to the file length.

Parameters:

[in] FileNumber Integer

[in] Position UInteger

FreeFile

Function FreeFile() As Integer

Returns free file number.

Return values:

< 0 - failed, return value is an error code

>=0 - succeeded, return value is free file number

HBBRFSInit

Function HBBRFSInit() As Integer

Initialize file system module.

Return values:

<= 0 - failed, return value is an error code

= 1 – succeeded

Kill

Sub Kill(ByRef Path As String)

Delete file from the file system.

Parameters:

[in] Path String

Sub Mkdir

Sub Mkdir(ByRef Path As String)

Make new directory.

Parameters:

[in] Path String

Mount

Function Mount(ByVal Device As Integer, ByRef Path As String) As Integer

Mount file system.

Parameters:

[in] Device Integer - device Id

[in] Path String - mount point

Return values:

<= 0 - failed, return value is an error code

= 1 – succeeded

Rename

Sub Rename(ByRef OldPath As String, ByRef NewPath As String)

Rename file.

Parameters:

[in] OldPath String

[in] NewPath String

RmDir

Sub RmDir(ByRef Path As String)

Remove existing directory.

Parameters:

[in] Path String

Umount

Function Umount(ByVal Device As Integer, ByRef Path As String) As Integer

Unmount file system.

Parameters:

[in] Device Integer - device Id

[in] Path String - mount point

Return values:

<= 0 - failed, return value is an error code

= 1 - succeeded

Errors

CONST ERROR_HBBRFS = -1

HBBRFS files system initialization error.

CONST ERROR_HBBRFS_BAD_DEVICE_ID = -32

ERROR bad device ID

CONST ERROR_HBBRFS_BAD_FILE_ACCESS = -37

ERROR bad file access

CONST ERROR_HBBRFS_BAD_FILE_LOCK = -38

ERROR bad file lock

CONST ERROR_HBBRFS_BAD_FILE_MODE = -36

ERROR bad file mode

CONST ERROR_HBBRFS_BAD_FILE_NUM = -35

ERROR bad file number

CONST ERROR_HBBRFS_BAD_MOUNT_PATH = -33

ERROR bad mount path

CONST ERROR_HBBRFS_CONFIG = -2

HBBRFS configuration error

CONST ERROR_HBBRFS_DIR_DOES_NOT_EXIST = -40

ERROR directory does not exists

CONST ERROR_HBBRFS_DIR_HAS_FILES = -42

ERROR directory is not empty

CONST ERROR_HBBRFS_FILE_ALREADY_OPEN = -39

ERROR file already open

CONST ERROR_HBBRFS_FILE_DOES_NOT_EXIST = -41

ERROR file does not exists

CONST ERROR_HBBRFS_FILE_READ_FAILED = -44

ERROR reading file failed

CONST ERROR_HBBRFS_FILE_WRITE_FAILED = -43

ERROR writing to file failed

CONST ERROR_HBBRFS_NO_FREE_FILE = -34

ERROR no free file

CONST MODE = &h00000001

Flag file mode

CONST MODE_APPEND = &h00000004

Append

CONST MODE_BINARY = &h00000008

Binary

CONST MODE_INPUT = &h00000001

Input

CONST MODE_OUTPUT = &h00000002

Output

CONST MODE_RANDOM = &h00000003

Random