

HBBRUSB USB Module

Version 2.000.402

(PRELIMINARY)

Copyright

This document is Copyright © 2009 by Hobby-Robotics, LLC

All trademarks within this guide belong to their legitimate owners.

Table of Contents

Copyright.....	2
Introduction.....	4
HBBRUSB USB Module Specification.....	4
Supported USB device classes.....	4
Supported MCUs.....	4
Supported Operating Systems.....	4
Limitations	4
HBBRUSB USB Module Requirements.....	4
Using HBBRUSB.....	5
Initialization.....	5
Managing USB.....	5
Driver installation.....	6
HBBRUSB Configuration.....	7
AT91SAM7S and AT91SAM7X.....	8
LPC2000.....	9
HBBRUSB USB Module API.....	10
Functions.....	10
Constants.....	10
Errors.....	10
Function HBBRUSBInit.....	10
Function HBBRUSBTTest.....	11
Function USB.Open	11
Function USB.Read.....	11
Function USB.ReadLine.....	12
Function USB.ReadString.....	12
Function USB.Reset	13
Function USB.Write.....	13
Function USB.WriteLine.....	13
Function USB.WriteString	14

Introduction

HBBRUSB is an HBBR Basic extension module providing support for USB 2.0 peripheral.

HBBRUSB USB Module Specification

Supported USB device classes

CDC Serial

Supported MCUs

AT91SAM7S - AT91SAM7S256, AT91SAM7S128, AT91SAM7S64

AT91SAM7X - AT91SAM7X256, AT91SAM7X128

LPC21xx - LPC2148, LPC2146, LPC2144

Supported Operating Systems

Windows – 2000,XP,Vista, Windows 7 Beta

Linux - TBD

Limitations

TBD

HBBRUSB USB Module Requirements

HBBR Basic version v2.000.402 or later

Board with USB interface and supported MCU.

Using HBBRUSB

Initialization

HBBRUSB USB module needs to be initialized by calling **HBBRUSBInit** function, return value greater than 0 indicates successful initialization while 0 means 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 = HBBRUSBInit()
If res > 0 Then
    ' successful
Else
    ' failed
    error = __hbbr_mod_get_last_error()
End If
Call __hbbr_enable_irq()
```

Managing USB

After initializing IO pins and internal registers the USB peripheral is ready to connect to USB bus by calling **USB.Open()** function. Boards with lpc214x MCU and supporting SoftConnect™ the **USB.Open()** will initiate USB bus enumeration on the host side.

Example:

```
res = USB.Open()
If res > 0 Then
    ' successful
Else
    ' failed
    error = __hbbr_mod_get_last_error()
End If
```

Driver installation

On Windows HBBRUSB requires installation of the appropriate driver. To install the driver build the example project then upload it to your board and run it. When the board is enumerated for the first time Windows will ask for driver, select manual installation then navigate to the HBBRUSB installation directory and point to the "hbbrusb.inf" file. HBBRUSB is using "usbser.sys" driver distributed with Windows.

HBBRUSB Configuration

HBBRUSB is using on-chip USB peripheral to access USB 2.0.

USB signal or MCU pin	LPC214x - USB Device = 0	AT91SAM7S - USB Device=0	AT91SAM7X - USB Device=0
D+	TBD	TBD	TBD
D-	TBD	TBD	TBD
SoftConnect	P0.31	NA	NA
VBUS	P0.23		
USB LED	User configured	User configured	User configured

AT91SAM7S and AT91SAM7X

System configuration

TBD

USB pin configuration

To configure USB 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 PA???.

LPC2000

System configuration

USB peripheral requires PCLK higher than 18MHz to function correctly. Typically boards supporting USB come with 12MHz XTAL which makes correct configuration possible. In such case typical SCB Configuration would be :

```
SCB Configuration = Manual
APBDIV = 1
MAMCR = 2
MAMTIM = 3
PLLCFG = 36
```

USB pin configuration

To configure USB 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:

HBBR IH1 board is using P0.24 as a TBD

HBBRUSB USB Module API

Functions

Function HBBRUSBInit()

Function HBBRUSBTst()

Function USB.Configure(ByVal optionmask As Integer, ByVal options As Integer) As Integer

Function USB.Reset()

Function USB.Open()

Function USB.Close()

Function USB.Write(ByVal byte As Byte) As Integer

Function USB.WriteString(ByRef str \$As String, ByVal count As Integer) As Integer

Function USB.WriteLine(ByRef str \$As String) As Integer

Function USB.Read(ByRef c As Byte) As Integer

Function USB.ReadString(ByRef str \$As String, ByVal count As Integer) As Integer

Function USB.ReadLine(ByRef str \$As String) As Integer

Constants

TBD

Errors

TBD

Function HBBRUSBInit

Function HBBRUSBInit() As Integer

Initialize USB module.

Return values:

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

= 1 - succeeded

Function HBBRUSBTest

Function HBBRUSBTest() As Integer

Test USB module.

Return values:

- <= 0 - failed, return value is an error code
- = 1 – succeeded

Function USB.Open

Function USB.Open() As Integer

Open USB device

Returns:

Integer value indicating result

Return values:

- <= 0 - failed, return value is an error code
- >= 1 – succeeded

Function USB.Read

Function USB.Read(ByRef c As Byte) As Integer

Read byte from USB device.

Parameters:[in] c Byte

Returns:

Integer value indicating result

Return values:

- <= 0 - failed, return value is an error code
- >= 1 - succeeded, return value is actual number of bytes written

Function USB.ReadLine

Function USB.ReadLine(ByRef str\$ As String) As Integer

Read bytes from USB device into the string until terminated by the end of line marker or maximum string length is reached. Characters marking end of line are stored in the string.

Parameters:[in] str\$ String

Returns:

Integer value indicating result

Return values:

- <= 0 - failed, return value is an error code
- >= 1 - succeeded, return value is actual number of bytes read

Function USB.ReadString

Function USB.ReadString(ByRef str\$ As String, ByVal count As Integer) As Integer

Read bytes from USB device into the string up to a specified count or maximum string length if count is 0 or more then maximum string length.

Parameters:[in] str\$ String

[in] count Integer, if 0 then read up to maximum string length

Returns:

Integer value indicating result

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

- >= 1 - succeeded, return value is actual number of bytes read

Function USB.Reset

Function USB.Reset() As Integer

Reset USB device.

Returns:

Integer value indicating result

Return values:

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

>= 1 - succeeded

Function USB.Write

Function USB.Write(ByVal byte As Byte) As Integer

Write byte to USB device

Parameters:[in] c Byte

Returns:

Integer value indicating result

Return values:

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

>= 1 - succeeded, return value is actual number of bytes written

Function USB.WriteLine

Function USB.WriteLine(ByRef str\$ As String) As Integer

Write characters from the input string to the USB device. End of line marker is written at the end.

Parameters:[in] str\$ String

Returns:

Integer value indicating result

Return values:

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

>= 1 - succeeded, return value is actual number of bytes written

Function USB.WriteString

Function USB.WriteString(ByRef str\$ As String, ByVal count As Integer) As Integer

Write characters from the string up to the count to the USB device.

Parameters:[in] str\$ String

[in] count Integer, if 0 then entire string is written

Returns:

Integer value indicating result

Return values:

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

>= 1 - succeeded, return value is actual number of bytes written