



# USB SAF1761 Host Controller User Guide

Version 1.30

For use with USBH SAF1761 Host Controller versions 2.11 and above

Exported on 10/24/2018

All rights reserved. This document and the associated software are the sole property of HCC Embedded. Reproduction or duplication by any means of any portion of this document without the prior written consent of HCC Embedded is expressly forbidden.

HCC Embedded reserves the right to make changes to this document and to the related software at any time and without notice. The information in this document has been carefully checked for its accuracy; however, HCC Embedded makes no warranty relating to the correctness of this document.

---

## Table of Contents

<b>1</b>	<b>System Overview.....</b>	<b>4</b>
1.1	Introduction .....	5
1.2	Feature Check .....	6
1.3	Compatible Devices .....	7
1.4	Packages and Documents .....	8
	Packages.....	8
	Documents .....	8
1.5	Change History .....	9
<b>2</b>	<b>Source File List .....</b>	<b>10</b>
2.1	API Header File .....	10
2.2	Configuration File.....	10
2.3	Source Code .....	10
2.4	Version File .....	10
<b>3</b>	<b>Source Files in the Common Package.....</b>	<b>11</b>
3.1	Configuration File.....	11
3.2	Source Code .....	11
3.3	Version File .....	11
3.4	Platform Support Package (PSP) Files.....	12
<b>4</b>	<b>Configuration Options .....</b>	<b>13</b>
4.1	SAF1761 Configuration File .....	13
4.2	SAF1761/ISP1582 Common Package Configuration File .....	13
<b>5</b>	<b>Starting the Host Controller .....</b>	<b>15</b>
5.1	usbh_isp_hc .....	15
5.2	Host Controller Task .....	15
5.3	Code Example .....	16
<b>6</b>	<b>Integration.....</b>	<b>17</b>
6.1	OS Abstraction Layer .....	17
6.2	PSP Porting .....	18
	psp_isp_init.....	20
	psp_isp_r16.....	21

---

psp_isp_r32.....	22
psp_isp_w16.....	23
psp_isp_w32.....	24
psp_isph_read_mem.....	25
psp_isph_read_ptd.....	26
psp_isph_write_mem.....	27
psp_isph_write_ptd.....	28

# 1 System Overview

This chapter contains the fundamental information for this module.

The component sections are as follows:

- [Introduction](#) – describes the main elements of the module.
- [Feature Check](#) – summarizes the main features of the module as bullet points.
- [Packages and Documents](#) – the *Packages* section lists the packages that you need in order to use this module. The *Documents* section lists the relevant user guides.
- [Change History](#) – lists the earlier versions of this manual, giving the software version that each manual describes.

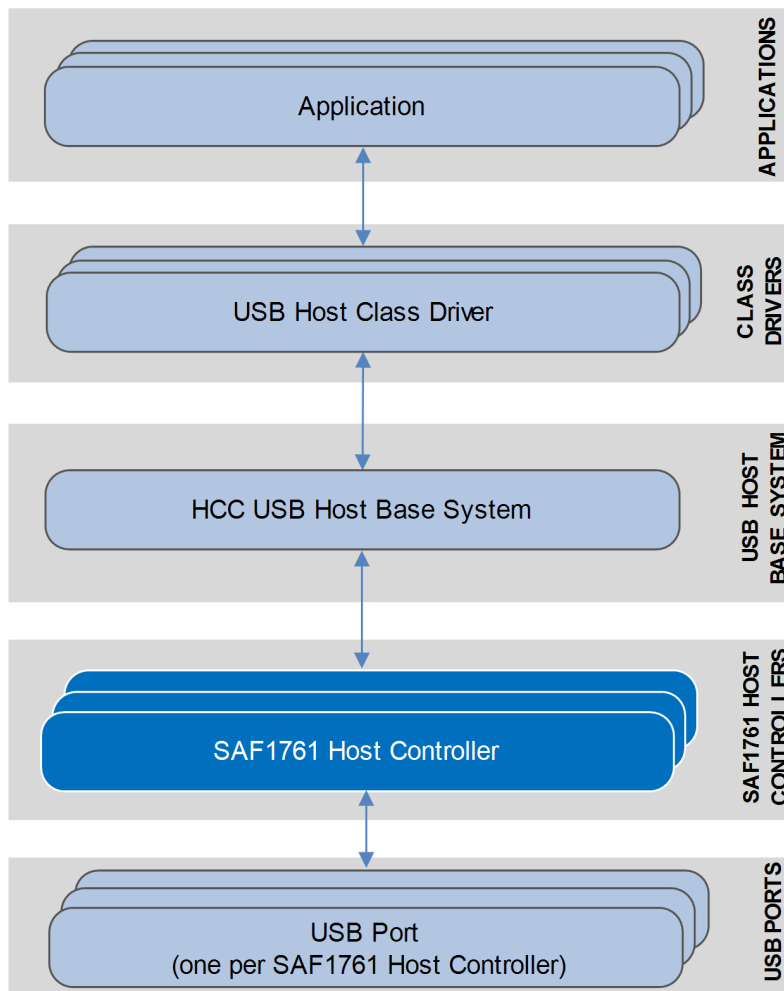
## 1.1 Introduction

This guide is for those who want to implement HCC Embedded's USB host stack with NXP Semiconductors' SAF1761 USB host controllers. The NXP devices supported are the SAF1761 and also the legacy products ISP1760, ISP1761, and ISP1763.

The SAF1761 includes a USB host controller and a USB device controller. The USB device controller is actually an ISP1582, for which we provide a separate driver. These two drivers can be used together as required. This manual only covers the USB host functionality of the SAF1761. The ISP1582 is covered in the *USB Device Low Level Driver for ISP1582 User Guide*.

The SAF1761 module provides a high speed USB 2.0 host controller which provides both full and low speed USB functions. The controller can handle all USB transfer types and, in conjunction with the USB host stack, can be used with any USB class driver.

The position of the host controller within the USB stack is shown below:



## 1.2 Feature Check

The main features of the host controller are the following:

- Conforms to the HCC Advanced Embedded Framework.
- Designed for integration with both RTOS and non-RTOS based systems.
- Integrated with the HCC USB Host stack and all its class drivers.
- Supports NXP Semiconductors' SAF1761 host controllers and also the legacy ISP1760, ISP1761, and ISP1763 host controllers.
- Supports multiple simultaneous SAF1761 controllers, each with multiple devices attached.
- Can work together with HCC's USB device controller for the ISP1582.
- Supports all USB transfer types: control, bulk, interrupt, and isochronous.

## 1.3 Compatible Devices

The module supports the following devices:

Device	Notes
NXP ISP1760	This is no longer marketed by NXP.
NXP ISP1761	This is no longer marketed by NXP.
ST ISP1763	This is now produced by ST Microelectronics.
NXP SAF1761	This is automotive grade, compliant with AEC-Q100.

## 1.4 Packages and Documents

### Packages

The table below lists the packages that you need in order to use this module:

Package	Description
<b>hcc_base_doc</b>	This contains the two guides that will help you get started.
<b>usbh_base</b>	The USB host base package. This is the framework used by USB class drivers to communicate over USB using a specific USB host controller package.
<b>usbh_drv_isp176x</b>	The USB host controller package described by this document.
<b>usbc_drv_isp1582_isp176x</b>	Common code used by both the SAF1761 module and the ISP1582 device low level driver module.
<b>usbh_cd_hub</b>	The USB Host Hub class driver.

### Documents

For an overview of HCC's embedded USB stacks, see [Product Information](#) on the main HCC website.

Readers should note the points in the [HCC Documentation Guidelines](#) on the HCC documentation website.

#### **HCC Firmware Quick Start Guide**

This document describes how to install packages provided by HCC in the target development environment. Also follow the *Quick Start Guide* when HCC provides package updates.

#### **HCC Source Tree Guide**

This document describes the HCC source tree. It gives an overview of the system to make clear the logic behind its organization.

#### **HCC USB Host Base System User Guide**

This document defines the USB host base system upon which the complete USB stack is built.

#### **HCC USB SAF1761 Host Controller User Guide**

This is this document.



## 1.5 Change History

This section describes past changes to this manual.

- To download this manual as a PDF, see [USB Host PDFs](#).
- For the history of changes made to the package code itself, see [History: usbh\\_drv\\_isp176x](#) and [History: usbc\\_drv\\_isp1582\\_isp176x](#).

The current version of this manual is 1.30. The full list of versions is as follows:

Manual version	Date	Software version	Reason for change
1.30	2018-10-24	2.11 and 1.02	Added PSP version file.
1.20	2017-08-29	2.11 and 1.02	Corrected <i>Packages</i> list.
1.10	2017-06-19	2.11 and 1.02	New <i>Change History</i> format.
1.00	2015-12-22	2.11 and 1.02	First release.

## 2 Source File List

This section describes all the source code files included in the system. These files follow the HCC Embedded standard source tree system, described in the *HCC Source Tree Guide*. All references to file pathnames refer to locations within this standard source tree, not within the package you initially receive.

**Note:** Do not modify any of these files except the configuration file.

### 2.1 API Header File

The file `src/api/api_usbh_isp176x.h` is the only file that should be included by an application using this module. It declares the `usbh_isp_hc()` function. For details, see [Starting the Host Controller](#).

### 2.2 Configuration File

The file `src/config/config_usbh_isp176x.h` contains all the configurable parameters. Configure these as required. For details of these options, see [Configuration Options](#).

### 2.3 Source Code

The source code files are in the directory `src/usb-host/usb-driver/isp176x`. **These files should only be modified by HCC.**

File	Description
<code>isp176x.c</code>	Source file for SAF1761 code.
<code>isp176x.h</code>	Header file for SAF1761 public functions.
<code>isp176x_hc.c</code>	Source file for the SAF1761 HC descriptor.
<code>isp176x_hc.h</code>	HC descriptor header file.
<code>isp176x_hub.c</code>	Source file for SAF1761 hub.
<code>isp176x_hub.h</code>	Header file for SAF1761 hub public functions.
<code>isp176x_reg.h</code>	Header file for SAF1761 registers.

### 2.4 Version File

The file `src/version/ver_usbh_isp176x.h` contains the version number of this module. This version number is checked by all modules that use this module to ensure system consistency over upgrades.

## 3 Source Files in the Common Package

This section describes the source code files included in the common package used by both the SAF1761 module and the ISP1582 device low level driver. These files follow the HCC Embedded standard source tree system, described in the *HCC Source Tree Guide*. All references to file pathnames refer to locations within this standard source tree, not within the package you initially receive.

**Note:** Do not modify any of these files except the configuration file.

### 3.1 Configuration File

The file `src/config/config_usbc_isp1582_isp176x.h` contains the common configurable parameters. Configure these as required. For details of these options, see [Configuration Options](#).

### 3.2 Source Code

The source code files are in the directory `src/usb-common/usb-drivers/isp_1582_isp176x`. **These files should only be modified by HCC.**

File	Description
<code>usbc_isp1582_isp176x.c</code>	Source file for common code.
<code>usbc_isp1582_isp176x.h</code>	Header file for common public functions.

### 3.3 Version File

The file `src/version/ver_usbh_isp1582_isp176x.h` contains the version number of this module. This version number is checked by all modules that use this module to ensure system consistency over upgrades.

## 3.4 Platform Support Package (PSP) Files

These files are in the directory **src/psp/target/usbh\_usbd\_isp**. They provide functions and elements the core code may need to use, depending on the hardware.

**Note:**

- These are PSP implementations for the specific microcontroller and development board; you may need to modify these to work with a different microcontroller and/or board. See [PSP Porting](#) for details.
- In the package these files are offset to avoid overwriting an existing implementation. Copy them to the root **hcc** directory for use.

File	Description
<b>psp_isp1582_isp176x.c</b>	Functions source code.
<b>psp_isp1582_isp176x.h</b>	Functions header file.

The PSP has a version file, **ver\_psp\_isp1582\_isp176x.h**.

## 4 Configuration Options

This section lists the available configuration options and their default values.

### 4.1 SAF1761 Configuration File

Set the following system configuration options in the file **src/config/config\_usbh\_isp176x.h**.

#### **MAX\_DEVICE**

The maximum number of devices supported. The default is 2.

#### **MAX\_EP**

The maximum number of endpoints (bulk, isochronous, and interrupt). The default is 4.

#### **USBH\_ISP\_TRANSFER\_STACK\_SIZE**

The stack size of the transfer task(s). The default is 4096.

### 4.2 SAF1761/ISP1582 Common Package Configuration File

Set the following system configuration options in the common package's file **src/config/config\_usbc\_isp1582\_isp176x.h**.

**Note:** For full details of these options, refer to the manufacturer's manual for the device.

#### **ISP\_USE\_HOST**

Keep the default of 1 to enable the host (if one is available).

#### **ISP\_USE\_DEVICE**

Set this to 0.

#### **ISP\_CHIP\_VERSION**

This specifies the ISP device used:

- 0 = ISP1760.
- 1 = ISP1761.
- 3 = ISP1763 (the default).

**ISP\_CHIP\_VER\_SAF**

Specify the device type; see [Compatible Devices](#):

- 0 = ISP176x (the default).
- 1 = SAF176x

**ISP\_32BIT\_IF**

This specifies how ISP is interfaced:

- 0 = 16bit mode (the default).
- 1 = in 32 bit mode.

**ISP\_INT\_LEVEL**

The interrupt trigger type:

- 0 - level triggered.
- 1 - edge triggered (the default).

**ISP\_INT\_POL**

The interrupt polarity:

- 0 - active low (the default).
- 1 - active high.

**ISP\_ISR\_ID**

The ISR ID. The default is ( `ISR_ID_XINT1` ).

**ISP\_INT\_PRIO**

The interrupt priority. This is local to the ISP Controller. The default is 0.

## 5 Starting the Host Controller

This section shows how to start the host controller and describes the task created. It includes a code example.

### 5.1 `usbh_isp_hc`

This external interface function provides the host controller descriptor required by the `usbh_hc_init()` function.

#### Format

```
extern void * const usbh_isp_hc
```

### 5.2 Host Controller Task

The host controller task handles all completed transfers. Callback requested for the transfer is executed from this task.

The task has the following attributes:

Attribute	Description
Entry point	<code>usbh_isp_transfer_task</code>
Priority	<code>USBH_TRANSFER_TASK_PRIORITY</code>
Stack size	<code>USBH_ISP_TRANSFER_STACK_SIZE</code> . The default is 4096.

## 5.3 Code Example

This example shows how to initialize the host controller. Note the following:

- There is only one external interface function, **usbh\_isp\_hc()**. To link this host controller to the system, you call the **usbh\_hc\_init()** function with this function as a parameter.
- The last parameter in the **usbh\_hc\_init()** call is the number of the host controller.

```
void start_usb_host_stack ( void )
{
  int rc;
  rc = hcc_mem_init();

  if ( rc == 0 )
  {
    rc = usbh_init();      /* Initialize USB host stack */
  }

  if ( rc == 0 )
  {
    /* Attach SAF1761 host controller */
    rc = usbh_hc_init( 0, usbh_isp_hc, 0 );
  }

  if ( rc == 0 )
  {
    rc = usbh_start(); /* Start USB host stack */
  }

  if ( rc == 0 )
  {
    rc = usbh_hc_start( 0 ); /* Start SAF1761 Host controller */
  }

  .....
}
```



## 6 Integration

This section specifies the elements of this package that need porting, depending on the target environment.

### 6.1 OS Abstraction Layer

All HCC modules use the OS Abstraction Layer (OAL) that allows the module to run seamlessly with a wide variety of RTOSes, or without an RTOS.

This module requires the following OAL elements:

OAL Resource	Number Required
Tasks	1
Mutexes	1
Events	1
ISRs	1

## 6.2 PSP Porting

The Platform Support Package (PSP) is designed to hold all platform-specific functionality, either because it relies on specific features of a target system, or because this provides the most efficient or flexible solution for the developer.

The host controller makes use of the following functions provided by the common ISP device package **usbc\_isp1582\_isp176x.h** file.

Function	Description
<b>usbc_isp_init()</b>	Initializes the device.
<b>usbc_isp_delete()</b>	Deletes the device.
<b>usbc_isp_isr_install()</b>	Installs the ISR.
<b>usbc_isp_isr_enable()</b>	Enables the ISR.
<b>usbc_isp_isr_disable()</b>	Disables the ISR.
<b>usbc_isp_isr_delete()</b>	Deletes the ISR.
<b>usbc_isp_lock()</b>	Locks the device.
<b>usbc_isp_unlock()</b>	Unlocks the device.

The host controller makes use of the following functions provided by the common PSP template file **psp\_isp1582\_isp176x.h** file. These are target-specific.

Function	Description
<b>psp_isp_init()</b>	Initializes the device.
<b>psp_isp_r16()</b>	Reads a 16 bit ISP address.
<b>psp_isp_r32()</b>	Reads a 32 bit ISP address.
<b>psp_isp_w16()</b>	Writes a 16 bit ISP address.
<b>psp_isp_w32()</b>	Writes a 32 bit ISP address.
<b>psp_isph_read_mem()</b>	Reads from the host memory area.
<b>psp_isph_write_mem()</b>	Writes to the host memory area.
<b>psp_isph_read_ptd()</b>	Reads from the PTD.
<b>psp_isph_write_ptd()</b>	Writes to the PTD.

These functions are described in the following sections.

**Note:** HCC can provide samples for different configurations; contact [support@hcc-embedded.com](mailto:support@hcc-embedded.com).

## psp\_isp\_init

This function is provided by the PSP to initialize the device.

This enables the clocks, GPIO pin, external memory interface, and so on.

### Format

```
int psp_isp_init ( void )
```

### Arguments

None.

### Return Values

Return value	Description
0	Successful execution.
Else	Operation failed.

## psp\_isp\_r16

This function is provided by the PSP to read a 16 bit value from the specified address.

This macro is configurable for 16 and 32 bit addressing, based on the setting of [ISP\\_32BIT\\_IF](#).

### Format

```
psp_isp_r16 ( d, a )
```

### Arguments

Parameter	Description	Type
d	The destination; where to put the data read.	uint16_t
a	The address; where to read the data from.	uint16_t

## psp\_isp\_r32

This function is provided by the PSP to read a 32 bit value from the specified address.

This macro is configurable for 16 and 32 bit addressing, based on the setting of [ISP\\_32BIT\\_IF](#).

### Format

```
psp_isp_r32 ( d, a )
```

### Arguments

Parameter	Description	Type
d	The destination; where to put the data read.	uint32_t
a	The address; where to read the data from.	uint32_t

## psp\_isp\_w16

This function is provided by the PSP to write a 16 bit value.

### Format

```
psp_isp_w16 ( a, v )
```

### Arguments

Parameter	Description	Type
a	Where to write the value.	uint16_t
v	The value to write.	uint16_t

## psp\_isp\_w32

This function is provided by the PSP to write a 32 bit value.

### Format

```
psp_isp_w32 ( a, v )
```

### Arguments

Parameter	Description	Type
a	Where to write the value.	uint32_t
v	The value to write.	uint32_t



## psp\_isph\_read\_mem

This function is provided by the PSP to read from the Host memory area.

### Format

```
void psp_isph_read_mem (  
    uint8_t *   p_dst,  
    uint32_t   addr,  
    uint32_t   size )
```

### Arguments

Parameter	Description	Type
p_dst	Where to put the data read.	uint8_t *
addr	The address to read from.	uint32_t
size	The number of bytes to read.	uint32_t

### Return Values

None.

## psp\_isph\_read\_ptd

This function is provided by the PSP to read from the PTD.

### Format

```
void psp_isph_read_ptd (  
    uint32_t * p_ptd,  
    uint32_t  addr,  
    uint16_t  size )
```

### Arguments

Parameter	Description	Type
p_ptd	Where to put the data read.	uint32_t *
addr	The address to read from.	uint32_t
size	The number of bytes to read.	uint16_t

### Return Values

None.

## psp\_isph\_write\_mem

This function is provided by the PSP to write to the Host memory area.

### Format

```
void psp_isph_write_mem (  
    uint32_t    addr,  
    uint8_t *   p_src,  
    uint32_t    size )
```

### Arguments

Parameter	Description	Type
addr	Where to write to.	uint32_t
p_src	A pointer to the data to write.	uint8_t *
size	The number of bytes to write.	uint32_t

### Return Values

None.

## psp\_isph\_write\_ptd

This function is provided by the PSP to write to the PTD.

### Format

```
void psp_isph_write_ptd (  
    uint32_t    addr,  
    uint32_t *  p_ptd,  
    uint16_t    size )
```

### Arguments

Parameter	Description	Type
addr	The address in the PTD to write to.	uint32_t
p_ptd	A pointer to the data to write.	uint32_t *
size	The number of bytes to write.	uint16_t

### Return Values

None.