

USB ISP1362 Host Controller User Guide

Version 1.10

For use with USBH ISP1362 Host Controller versions
1.02 and above

Date: 19-Jun-2017 17:22

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

System Overview	3
Introduction	3
Feature Check	4
Packages and Documents	5
Packages	5
Documents	5
Change History	6
Source File List	7
API Header File	7
Configuration File	7
Source Code	7
Version File	8
Platform Support Package (PSP) Files	8
Configuration Options	9
Starting the Host Controller	10
usbh_isp_hc	10
Host Controller Task	10
Code Example	11
Integration	12
OS Abstraction Layer	12
PSP Porting	13
isp_hwinit	14

1 System Overview

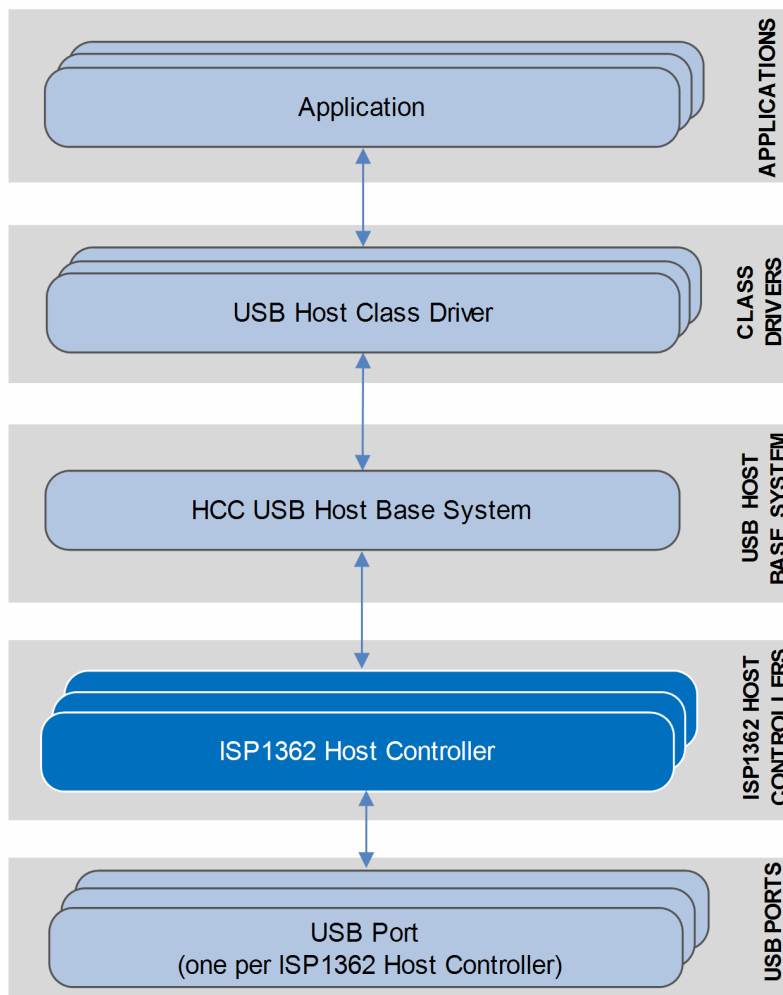
1.1 Introduction

This guide is for those who want to implement HCC Embedded's USB host stack with NXP Semiconductors' ISP1362 USB host controller.

The ISP1362 includes a USB host controller and a USB device controller. The USB device controller is actually an ISP1181, for which we provide a separate low level driver. These two drivers can be used together as required. This manual only covers the USB host functionality of the ISP1362. The ISP1181 is covered in the *USB Device Low Level Driver for ISP1181 User Guide*.

The ISP1362 module provides a high speed USB 2.0 host controller that 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 Semiconductor ISP1362 USB host controllers.
- Supports multiple simultaneous ISP1362 controllers, each with multiple devices attached.
- Can work together with HCC's USB device controller for the ISP1181.
- Supports all USB transfer types: control, bulk, interrupt, and isochronous.

1.3 Packages and Documents

Packages

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

Package	Description
<code>hcc_base_doc</code>	This contains the two guides that will help you get started.
<code>usbh_base</code>	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.
<code>usbh_drv_isp1362</code>	The USB host controller package described by this document.

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 ISP1362 Host Controller User Guide

This is this document.

1.4 Change History

This section describes past changes to this manual.

- To view or download earlier manuals, see [Archive: USB ISP1362 Host Controller User Guide](#).
- For the history of changes made to the package code itself, see [History: usbh_drv_isp1362](#).

The current version of this manual is 1.10. The full list of versions is as follows:

Manual version	Date	Software version	Reason for change
1.10	2017-06-19	1.02	New <i>Change History</i> format.
1.00	2015-12-21	1.01	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 and PSP files.

2.1 API Header File

The file `src/api/api_usbh_isp1362.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_isp1362.h` contains all the configurable parameters. Configure these as required. For details, see [Configuration Options](#).

2.3 Source Code

The source code files are in the directory `src/usb-host/usb-driver/isp1362`. **These files should only be modified by HCC.**

File	Description
<code>isp1362.c</code>	Source file for ISP1362 code.
<code>isp1362.h</code>	Header file for ISP1362 public functions.
<code>isp1362_hc.c</code>	Source file for the ISP1362 HC descriptor.
<code>isp1362_hc.h</code>	HC descriptor header file.
<code>isp1362_hub.c</code>	Source file for public hub functions.
<code>isp1362_hub.h</code>	Header file for public hub functions.
<code>isp1362_reg.h</code>	Header file for ISP1362 registers.
<code>isp1362_rw.c</code>	Source code for functions including read/write.
<code>isp1362_rw.h</code>	Header file for functions including read/write.

2.4 Version File

The file `src/version/ver_usbh_isp1362.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.

2.5 Platform Support Package (PSP) Files

These files are in the directory named `src/psp/target/usbh_isp1362`. These provide functions and elements the core code may need to use.

Note: These are PSP implementations for the specific microcontroller and board; you may need to modify these to work with a different microcontroller and/or development board. See [PSP Porting](#) for details.

The files are as follows:

File	Description
<code>psp_usbh_isp1362.c</code>	Source code of functions.
<code>psp_usbh_isp1362.h</code>	Header file for functions.

3 Configuration Options

Set the following system configuration options in the file `src/config/config_usbh_isp1362.h`.

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

MAX_DEVICE

The maximum number of devices supported. The default is 4.

MAX_EP

The maximum number of endpoints; the total of bulk, isochronous, and interrupt endpoints. The default is 20.

ISP_HOST_ISR

The ISR. The default is 0x0001176F.

ISP_HOST_IT_PRIO

The priority of the ISP device interrupt. The default is 0.

USBH_ISP_TRANSFER_STACK_SIZE

The stack size of the transfer task(s). The default is 1024.

4 Starting the Host Controller

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

4.1 `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
```

4.2 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 1024.

4.3 Code Example

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

- There is only one external interface function, **usbh_isc_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 ISP1362 host controller */
        rc = usbh_hc_init( 0, usbh_isc_hc, 0 );
    }

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

    if ( rc == 0 )
    {
        rc = usbh_hc_start( 0 ); /* Start ISP1362 Host controller */
    }

    .....
}
```

5 Integration

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

5.1 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	0

5.2 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 module makes use of the following standard PSP function:

Function	Package	Element	Description
<code>psp_memset()</code>	psp_base	psp_string	Sets the specified area of memory to the defined value.

The host controller makes use of the following function provided by the PSP template file `psp_usbh_isp1362.h`. This is target-specific.

Function	Description
<code>isp_hwinit()</code>	Initializes the device.

This function is described in the following section.

Note: HCC can provide samples for different configurations; contact support@hcc-embedded.com.

isp_hwinit

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 isp_hwinit ( void )
```

Arguments

None.

Return Values

Return value	Description
0	Successful execution.
Else	Operation failed.