

# USB LLD Quick Reference

Microcontrollers



Never stop thinking.

**Edition v4.2, 28 Feb 2006**

**Published by Infineon Technologies India pvt. Ltd.,  
ITPL,  
Bangalore, INDIA**

**© Infineon Technologies AP 2006.  
All Rights Reserved.**

#### **Attention please!**

The information herein is given to describe certain components and shall not be considered as warranted characteristics.

Terms of delivery and rights to technical change reserved.

We hereby disclaim any and all warranties, including but not limited to warranties of non-infringement, regarding circuits, descriptions and charts stated herein.

Infineon Technologies is an approved CECC manufacturer.

#### **Information**

For further information on technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies Office in Germany or our Infineon Technologies Representatives worldwide (see address list).

#### **Warnings**

Due to technical requirements components may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies Office.

Infineon Technologies Components may only be used in life-support devices or systems with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system, or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body, or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

# USB LLD Quick Reference

Microcontrollers



Never stop thinking.

---

**Table 1**

Page	Subjects (major changes since last revision)
ALL	First Draft

---

**Author:**

Jayashree Badarinath

**We Listen to Your Comments**

Is there any information in this document that you feel is wrong, unclear or missing ?  
Your feedback will help us to continuously improve the quality of our documentation.  
Please send your feedback (including a reference to this document) to:

[care\\_services@infineon.com](mailto:care_services@infineon.com)

## 1 Source file references

### 1.1 C source files

File name	Description
usb_idl.c	Implements low-level driver functionalities for USB device.
usb_iil_setup.c	Implements device enumeration functionalities.
usb_iil_rx.c	Implements reception functionality for application layer
usb_main_das.c	<i>Implements the sample program to test DAS Application</i>

### 1.2 Header Files

File name	Description
usb_iil_cfg.h	Configuration file for user as per USB1.1/1.0 specs.
usb_idl_cfg.h	USB hardware configurations file for user.
usbd_idl_macro.h	Abstracts hardware definitions for low-level driver.
usbd_idl.h	Implements the hardware definitions.
usb_iil_setup.h	Contains structure definitions for USB1.1 specs parameters.
usb_iil_common.h	Contains defines which are common to all USB HAL files
usb_iil_api.h	Contains interface definitions for application layer.

### 1.3 Other dependent files

File name	Description
compiler.h	Contains compiler dependent definitions

sys_api.h	Contains system related definitions
sys_cfg.h	System configuration file for user for setting interrupt priorities.
sys_iil.c	Contains system related functions
common.h	Contains system related functions
main.h	just main header file

## 2 Project workspace

### GNU

Copy the above mentioned files to the desired directory:

1. Common.h, compiler.h, sys\_api.h, sys\_cfg.h and sys\_iil.c from system folder
2. Usb\_idl.c, usb\_iil\_rx.c, usb\_iil\_setup.c, usb\_idl\_cfg.h, usb\_iil\_api.h, usb\_iil\_cfg.h, usb\_iil\_common.h, usb\_iil\_setup.h, usbd\_idl.h, usbd\_idl\_macro.h, usb\_main\_das.c [Version 4.2]

Note: product ID and BCD Device need to be **0x1F** CONFIG\_USB\_DEVICE\_DESCRIPTOR in usb\_iil\_cfg.h [ Note: This is required for USBIO Light driver]

0x001F, /\* field ProductId assigned by Infineon \*/\

0x001F, /\* field bcdDevice \*/\

- 4 Copy the target.ld from USB\_V4.2\GNU\GNU\_Target\_Files\GNU\_3.3.7.3.

And make the workspace with the following project settings:

### **Build Rules:**

User Flags: -mall-errata

Standard Debug

Level2

Tv1.3

Defines:

TRIBOARD\_TC1130

LINK RULES:

o/p: usb\_lnk.elf

Link Flags: -mtc13 -Wl,-Map -Wl,usb\_internal.lst

Link Script

//PATH/target.ld

Build the .elf.

While compilation comment the following line in types.h

Typedef unsigned short ushort;

### **Tasking**

Copy the following files to the desired directory:

1. Common.h, compiler.h, sys\_api.h, sys\_cfg.h and sys\_iil.c from system folder
2. Usb\_idl.c, usb\_iil\_rx.c, usb\_iil\_setup.c, usb\_idl\_cfg.h, usb\_iil\_api.h, usb\_iil\_cfg.h, usb\_iil\_common.h, usb\_iil\_setup.h, usbd\_idl.h, usbd\_idl\_macro.h, usb\_main\_das.c [Version 4.2]

### **Apply Tasking Patches**

1. In addition copy the cstart.asm, nommu.lst and nommu.opt, from USB\_V4.2\Tasking\_Patches to the current project workspace directory
2. Load the nommu.opt, from the project folder which was copied earlier.
3. Change the path to the nommu.lst to the current project directory.
4. Ensure the cstart.asm copied into project folder is read-only so that the changes are not overwritten.

The USB LLD is working fine with Tasking 2.2r2 and Tasking 2.2r3 with the MMU library off with the following workaround defined below.

>>>>

#### **MMU libraries**

Special MMU libraries have been added for derivatives that have a MMU on board. These libraries can be found in the subdirectory lib/



tc1\_mmu/ and lib/tc2\_mmu. The MMU hardware workaround is triggered by the --mmu-present option. This option is automatically set when targets are specified (with the -C option) which have a MMU. So -Ctc1130 also sets the --mmu-present option, removing the -Ctc1130 is the only option to check this, but this has some side effects. These MMU libraries contain a natural alignment for data objects. These additional MMU libraries are required, as the MMU requires natural alignment. This is causing alignment issues in the low level driver code.

Concerning this MMU matter, please have a look at the errata sheet of the TC1130, silicon bug CPU\_TC052. Here the problem is described and further information is given.

When this library is enabled, the MMU library alignment is overwriting the data alignment in the low level driver code due to which the transmission and reception of data is failing

**[Workaround]** - Select user CPU type EDE will use '-Cuserdef113' to identify a user CPU type. Its corresponding register file only contains the default CPU registers. The assembler still requires the register definitions for tc1130 which can be explicitly added to the 'include this file before source' EDITBOX of the assembler 'preprocessing' PAGE. In the Tasking\_Patches folder you will find an option file (nommu.opt) which can be imported into project to make this happen. This option file will also setup your 'script file' PAGE such that it uses external LSL file nommu.lsl, and just a definition additon in cstart.asm will make it work.

Even if the MMU library is enabled with this option file code would still work fine(Hence MMU library and USB can still be integrated in case if required), and hence this workaround does not have any side effects, only thing is we need use the .opt file, .lsf and cstart.asm for Tasking 2.2r2 onwards. User need to make sure to include the correct def files,

as well as the correct LSL file for the linker. These files are all automatically included by the -Ctc1130 option (you can verify this by using the -v option for the control program, this allows you to see which options are passed to the tools).

Compile the code

## 2.1 TC1130 USBIO Light Driver Installation

Uninstall the existing inf files using

TC1130\_LLD\_V4.00\USB\_V4.2\USB\_Driver\USBIOcw.EXE

Install the driver usbiov.inf present in driver folder.

[Note: This works only for 12 hours, and after that PC would require reboot.

Run the .elf generated and do the following steps:

1. Load the .elf present in the application folder through source navigator/crossview respectively.
2. Run the application.
3. Plug and play the device.
4. Install the driver present in Driver folder usbiov.inf.
5. after successful enumeration, windows finishes installing the driver, you may check that the four entries shown below are in the USB Section of Windows Device Manager (To open the Device Manager, right click on the "My Computer" icon on your desktop and click on "Manage". In the windows that pops up click on "Device Manager").

**"Infineon - USB - IF0"**

**"Infineon - USB - IF1"**

**"Infineon - USB - IF2"**

**"Infineon - USB - IF3"**

Run the application found in the

"USB\_V4.2\Demo\_Application\Quick\_LLD\_Reference\TC1130\_DAS\_TEST\_GUI"

folder, which pops up the following executable in the below mentioned diagram.

=====

**Check for Manual Mode:**

1. **Buffer Count, Size: 2,64**
2. **0Fill = not enabled X, 52, 170, 210, 500, 64, 128, 129, 256, 512**

**Check for Automatic Mode:**

1. **Buffer Count, Size: 2,512**
  2. **0Fill = enabled X, 52, 170, 210, 500, 982, 64, 128, 129, 256, 512, 2048**
- =====

**Automatic mode Test**

1. Tick the Interface number and alternate setting number and add the EP number in decimal, for example, with the default, usb\_main\_das.c, EP1(129), EP2(2), have been configured on Interface 1(IF1) and Alternate Setting(AS1), and hence the configuration in the application are selected accordingly.

2. After selecting the configuration, click on **Do Button** which Register Device Notification, Create Event, builds the device list, and open the device and binds the in and out EP for communication. On successful enumeration and identification of the device, the following message would be displayed:

RegisterDeviceNotification done

CreateEvent passed!

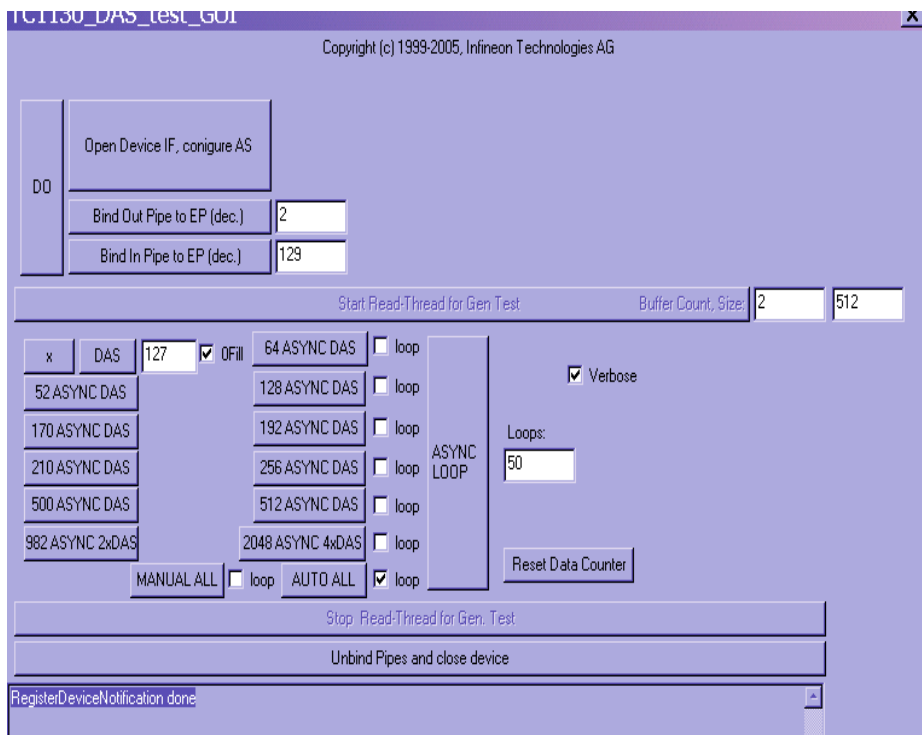
Building of device list passed!

Device open Passed!

Binding to pipe for Out EP Passed!

Binding to pipe for In EP Passed!

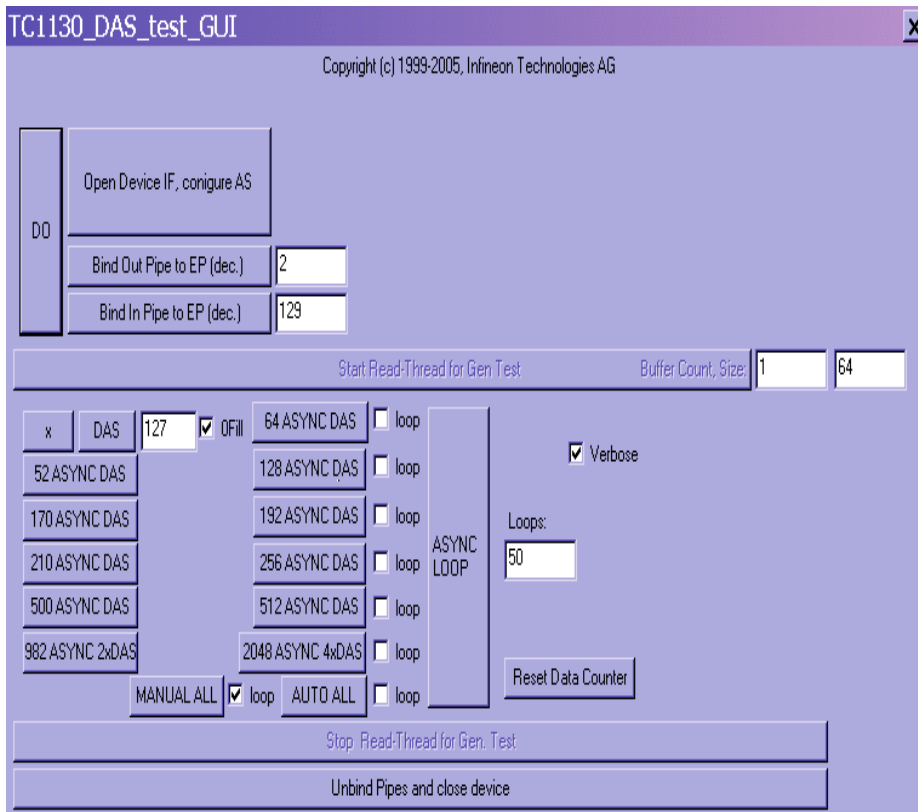
or either click on Open Device IF, Configure AS, followed by Binding Out pipe to EP 2 and Bind IN pipe to EP 1 i.e, 129



3. After selecting the Buffer count, **“Start read - Thread for Gen Test”**
4. Select the type of **“ASYNC DAS”** transmission to be tested, and perform the asynchronous transmission. Tick the box next to the tests, and Loops count and then click on Async loop to test the Async transmission in loop.
5. **“TC1130 Test flow”** tests all the asynchronous tests at one shot.
6. After the successful test **“Stop Read Thread for Gen Test”** and **“Unbind Pipes and Close Device”**.

## Manual mode Test

1. Select the **USB\_MANUAL\_MODE** in usb\_idl\_cfg.h file and recompile the project workspace.
2. Run the TC1130\_DAS\_TEST\_GUI.exe



TC1130\_DAS\_test\_GUI

Copyright (c) 1999-2005, Infineon Technologies AG

DO

Open Device IF, configure AS

Bind Out Pipe to EP (dec.) 2

Bind In Pipe to EP (dec.) 129

Start Read-Thread for Gen Test

Buffer Count, Size: 1 64

x	DAS	loop
127	64 ASYNC DAS	<input type="checkbox"/> loop
52 ASYNC DAS	128 ASYNC DAS	<input type="checkbox"/> loop
170 ASYNC DAS	192 ASYNC DAS	<input type="checkbox"/> loop
210 ASYNC DAS	256 ASYNC DAS	<input type="checkbox"/> loop
500 ASYNC DAS	512 ASYNC DAS	<input type="checkbox"/> loop
982 ASYNC 2xDAS	2048 ASYNC 4xDAS	<input type="checkbox"/> loop
MANUAL ALL	AUTO ALL	<input checked="" type="checkbox"/> loop

ASYNC LOOP

Verbose

Loops: 50

Reset Data Counter

Stop Read-Thread for Gen Test

Unbind Pipes and close device

2. Tick the Interface number and alternate setting number and add the EP number in decimal, for example, with the default, usb\_main\_das.c, EP1(129), EP2(2), have been configured on Interface 1(IF1) and Alternate Setting(AS1), and hence the configuration in the application are selected accordingly.

3. After selecting the configuration, click on **Do Button** which Register Device Notification, Create Event, builds the device list, and open the device and binds the in

and out EP for communication. On successful enumeration and identification of the device, the following message would be displayed:

RegisterDeviceNotification done

CreateEvent passed!

Building of device list passed!

Device open Passed!

Binding to pipe for Out EP Passed!

Binding to pipe for In EP Passed!

or either click on Open Device IF, Configure AS, followed by Binding Out pipe to EP 2 and Bind IN pipe to EP 1 i.e, 129

4. Select the Buffer count to 2, 512 for automatic test and 1,64 for manual mode test, [ Please note the Limitation as mentioned in Release\_Note\_USB\_LLD\_v4\_2.pdf with the release notes for selection of buffer count]

Default Buffer count is configured to 2, size to 512.

5. After selecting the Buffer count, **“Start read - Thread for Gen Test”**

6. Select the type of **“ASYNCR DAS”** transmission to be tested, and perform the asynchronous transmission. Tick the box next to the tests, and Loops count and then click on Async loop to test the Async transmission in loop.

7. **“TC1130 Test flow”** tests all the asynchronous tests at one shot.

7. After the successful test **“Stop Read Thread for Gen Test”** and **“Unbind Pipes and Close Device”**.

### **GUI for a EP0 read check**

This GUI has been developed to detect & decode a classor vendor request.

To test the EP0 bi-directionality, we have to work with

USBIO\_CLASS\_OR\_VENDOR\_REQUEST. (Page 98 on usbioman.pdf in the Thesycon directory)

1. Load the .elf present in the application folder through source navigator/crossview respectively.
2. Run the application.
3. Plug and play the device.
4. Install the driver present in Driver folder usbiov.inf.
5. after successful enumeration, windows finishes installing the driver, you may check that the four entries shown below are in the USB Section of Windows Device Manager (To open the Device Manager, right click on the "My Computer" icon on your desktop and click on "Manage". In the windows that pops up click on "Device Manager").

**"Infineon - USB - IF0"**

**"Infineon - USB - IF1"**

**"Infineon - USB - IF2"**

**"Infineon - USB - IF3"**

Run the application found in the

"USB\_V4.2\Demo\_Application\Quick\_LLD\_Reference\TC1130\_EP0\_CLASS\_VENDO R.exe" folder, which pops up the following executable in the below mentioned diagram.

6. Click on Open Device IF0, followed by CLASS\_OR\_VENDOR 0x2 and then click on close device IF0



On successful execution of application, the following result is obtained in the output window

```
CreateEvent passed!  
Building of device list passed!  
Device open passed!  
Sending Vendor Request...  
DATA UPLOAD passed!  
10000000  
01000000  
11000000  
00100000  
10100000  
01100000  
11100000  
00010000  
: 0c0b0a09  
: 100f0e0d  
: 14131211  
: 18171615  
: 1c1b1a19  
: 201f1e1d  
: 24232221  
: 28272625  
: 2c2b2a29  
: 302f2e2d  
: 34333231  
: 38373635  
: 3c3b3a39  
: 003f3e3d  
Device and Pipes successfully closed!
```



<http://www.infineon.com>