USB LLD Demo Application Readme File

Microcontrollers

Never stop thinking.

Edition v4.2, 28 Feb 2006

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

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 Demo Application Readme File

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

1 Folder contents

DAvE - This contains preconfigured DAvE settings for GNU and Tasking for reference purpose

GNU - This contains elf generated for Blink LED using the GNU Workspace

Tasking - This contains elf generated for Blink LED using Tasking compiler workspace.

TC1130_USB_BlinkyLED - This contains the VC++ Version 6 Sources for Blink LED Folder

Thesycon - This folder is the USBIO Driver folder which is required for the compilation of above VC++ Application. Note this folder need to be copied to C:\ prior to compilation of VC++ Workspace.

TC1130_USB_BlinkyLED.exe - Application

USB_Demo_App_Readme - Readme file

2 USB Blinking DEMO Example

2.1 DAvE Settings:

Select the processor as **TC1130**

2.1.1 General settings

Select the compiler GNU/Tasking

System Clock:
External clock freq: 20 MHz
P = 4
VCO Range = 400-500Mhz
N =96
K = 5
Ratio of Fcpu/Fsys = 1:1

Interrupts
Enable the global interrupts system[IE]

2.1.2 ASC0

Enable ASC0
Module clock:
Clock divider for normal operation mode[RMC] – 96MHz

Control
Mode control: 8-Bit data(Asynchronous)
Transmit pin Selection -> Tx Pin {P2.1) Selected.
Ë Rx Pin(P2.0) Selected
Receiver enable (REN) -> Select the Enable Receiver
Stop Bit Selection [STP] -> Select one stop bit

Interrupts -> Select Enable Receive interrupt[RSRC]

Baud Rate
Select the Use fractional divider as prescalar for baud rate timer[FDE]
Required baud rate [kbaud] – 115.200.
Baud Rate Generator Run Control [R] – Enable baud rate generator
Reload value[RL] – 0x002E

Interrupts
Select the ASC0 receive SRN to level 75 onto left tab

Functions
Select ASC0_vInit
Select following function library
ASC0_vSendData
ASC0_usGetData
ASC0_viRx
ASC0_ubTxDataReady
ASC0_ubTxBufFree

2.1.3 GPTU

Module Clock
Module Run Mode Clock Control: Clock divider for normal operation mode[RMC] – System clock / 1 [= 96.0000MHz]
T0/1
T0 and T1 Global Input 0 Selection [T01IN0]
Select Timer T2A over-/underflow
T0 and T1 Global Input 0 Selection [T01IN1]
Select Timer T2B over-/underflow

Timer 0/Timer 1

Configure T0A –
Select Clock input Selection [T0AINS] Clock input [=96.0000MHz]
Timer Reload Selection [T0AREL] Reload on overflow of timer T0A
Timer Run Control – Select T0A after initialization [T0ARUN]
Timer Registers
Timer T0A reload register – 0x00
Timer T0A register 0x00
Overflow [us] – 2.6667
Configure T0B
Select Clock input Selection [T0BINS] – Carry input from T0A [Concatenation]
Timer Reload Selection [T0BREL] Reload on overflow of timer T0B
Timer Run Control – Select T0B after initialization [T0BRUN]
Timer Registers
Timer T0A reload register – 0x00
Timer T0A register 0x00
Overflow [us] – 682.6667
Configure T0C
Select Clock input Selection [T0CINS] – Carry input from T0B [Concatenation]
Timer Reload Selection [T0CREL] Reload on overflow of timer T0C
Timer Run Control – Select T0C after initialization [T0CRUN]
Timer Registers
Timer T0A reload register – 0x80
Timer T0A register 0x80
Overflow [us] – 87.3813
Configure T0D
Select Clock input Selection [T0DINS] – Carry input from T0C [Concatenation]
Timer Reload Selection [T0DREL] Reload on overflow of timer T0D
Timer Run Control – Select T0D after initialization [T0DRUN]
Timer Registers
Timer T0A reload register – 0xF8
Timer T0A register 0xF8
Overflow [us] – 699.0507

T0/T1
Configure Timer 0 outputs
Output Source selection OUT0x(SOUT0x)
Signal OUT00: Timer T0D overflow generates the signal OUT00

External Outputs -
Configure GPTU External port 7
Output 7 pin selection
Select Use pin 0.7 as GPTU_PUT7 output

Function:
Select GPTU_vInit

Select following Function library
GPTU_vStartTmr
GPTU_vStopTmr
GPTU_vSetReload
GPTU _uwGetReload

2.1.4 SCU

Control - > Enable CSG [CSGEN]
USB Clock Divider – Select USB Clock Divider [2:1]
USB Clock Select – Select USB Clock from Internal CGU

Functions: Select SCU_vInit

Can even refer to the .dav files present in DAvE folder

2.2 User Code

The following is the user code to be added in the generated DAvE files:

ASC0.c

```
// USER CODE BEGIN (Rx,1)

void PrintOut(const char *format, ...);
void SendStatus();
extern unsigned char BlinkingLEDSpeed;
extern int BlinkingLEDState;
// USER CODE END
```

In the ASC0_viRx(void) function add the following code in the User code

```
{

// USER CODE BEGIN (Rx,2)
unsigned char c = ASC0_usGetData();
switch (c)
      {
      case '0':
         PrintOut("Request to Stop the Blinking LED\n\r");
         GPTU_vStopTmr(GPTU_T0A_D);
         BlinkingLEDState = 0;
         break;
      case '1':
         PrintOut("Request to Start the Blinking LED\n\r");
         GPTU_vStartTmr(GPTU_T0A_D);
         BlinkingLEDState = 1;
         break;
      case '+':
         if (BlinkingLEDSpeed<0xff)
```

```
       {
       BlinkingLEDSpeed ++;
       GPTU_vSetReload(GPTU_T0D,BlinkingLEDSpeed);
       PrintOut("Request to Increase the Blinking LED Speed:\t0x%.2x\n\r",Blink-
ingLEDSpeed);
       }
       break;

     case '-':
      if (BlinkingLEDSpeed>0xf0)
      {
      BlinkingLEDSpeed --;
      GPTU_vSetReload(GPTU_T0D,BlinkingLEDSpeed);
     PrintOut("Request to Decrease the Blinking LED Speed:\t0x%.2x\n\r",Blink-
ingLEDSpeed);
     }
     break;
  case 'i':
      PrintOut("Status:\n\rT0RDBCA: 0x%.8x\n\r",GPTU_T0RDCBA);
      PrintOut("T0DBCA: 0x%.8x\n\r",GPTU_T0DCBA);
      break;
  default:
      break;
}
SendStatus();

  // USER CODE END

} //  End of function ASC0_viRx
```

Main.c

```
// USER CODE BEGIN (MAIN_General,2)
#include "usb_iil_api.h "
#include "compiler.h"
#include "sys_api.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stdarg.h>

// USER CODE END

// USER CODE BEGIN (MAIN_General,7)

        int BlinkingLEDState = 1;
        unsigned char BlinkingLEDSpeed = 0xf8;

// USER CODE END

// USER CODE BEGIN (Main,1)

void PrintOut(const char *format, ...)
{
  int i;
  const int buffersize = 2048;
  va_list argptr;

  char buffer[buffersize];

  // print to buffer
  va_start(argptr,format);
  vsprintf(buffer,format,argptr);
  buffer[buffersize-1] = 0;
```

```
  // send buffer to output window (synchronous behavior)
  for (i=0;i<strlen(buffer);i++)
  {
  while (!ASC0_ubTxBufFree());
  ASC0_vSendData(buffer[i]);
  }


  va_end(argptr);

}

void SendStatus()
{
PrintOut("Transmitting status:\n\r\tState is %s\n\r\tSpeed is:
0x%.2x\n\r\t",BlinkingLEDState==0?"Not Blinking":"Blinking",BlinkingLED-
Speed);
}
// USER CODE END

sword main(void)
{
  sword swReturn;

  // USER CODE BEGIN (Main,2)
unsigned char Buffer[64];
int nBytesTransmit = 64;
int val;
  // USER CODE END


  // USER CODE BEGIN (Init,3)
```

```
  //   initializes the Universal Serial Bus(USB)
  USBD_device_initialize();

  // USER CODE END

  // USER CODE BEGIN (Main,9)

PrintOut("\n\r Welcome to Blinky LED USB Application\n\r");
PrintOut("\n\r Open the GUI Application\n\r");
PrintOut("\n\r Click on Open&Configure\n\r");
PrintOut("\n\r Click on Bind Out Pipe for binding the Out Pipe \n\r");
PrintOut("\n\r Click on Bind IN Pipe for binding the IN Pipe \n\r");
PrintOut("\n\r Click on Start Read-Thread \n\r");
PrintOut("\n\r Click on Faster/Slower to vary the speed of LED\n\r");
while (1)
{
static int count = 0;
IFX_UINT8 pData[64];
IFX_SINT16 nBytesRequest = 64, nBytesReceived;
if (USBD_get_device_state() == USB_CONFIGURED)
{
nBytesReceived  = USB_receive(pData, nBytesRequest, 0x02);
count ++;

if(nBytesReceived > 0)
{
switch (pData[0])
{
case '0':
PrintOut("Request to Stop the Blinking LED\n\r");
GPTU_vStopTmr(GPTU_T0A_D);
BlinkingLEDState = 0;
break;
```

```
case '1':
PrintOut("Request to Start the Blinking LED\n\r");
GPTU_vStartTmr(GPTU_T0A_D);
BlinkingLEDState = 1;
break;
case '+':
if (BlinkingLEDSpeed<0xff)
{
BlinkingLEDSpeed ++;
GPTU_vSetReload(GPTU_T0D,BlinkingLEDSpeed);
PrintOut("Request to Increase the Blinking LED Speed:\t0x%.2x\n\r",Blinking-
LEDSpeed);
}
break;
case '-':
if (BlinkingLEDSpeed>0xf0)
{
BlinkingLEDSpeed --;
GPTU_vSetReload(GPTU_T0D,BlinkingLEDSpeed);
PrintOut("Request to Decrease the Blinking LED Speed:\t0x%.2x\n\r",Blinking-
LEDSpeed);
}
break;
default:
break;
}
Buffer[0] = (unsigned char)BlinkingLEDState;
Buffer[1] = BlinkingLEDSpeed;

val = USBD_transmit(0x01, Buffer, nBytesTransmit, 0);

}
//SendStatus();
```

```
}
}

  // USER CODE END
```

2.3 Project workspace

GNU

Copy the following files to the desired directory:

1. ASC0.c, ASC0.h, GPTU.c, GPTU.h, MAIN.c, MAIN.h, SCU.c, SCU.h, TC1130Regs.h (DAvE generated code with the above user code added)
2. Common.h, compiler.h, sys_api.h, sys_cfg.h and sys_iil.c from system folder
3. 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 [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 */\
```

In the usb_idl_cfg.h, ensure to set the mode to **USB_MANUAL_MODE**

4 Copy the target.ld

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. ASC0.c, ASC0.h, GPTU.c, GPTU.h, MAIN.c, MAIN.h, SCU.c, SCU.h, TC1130Regs.h (DAvE generated code with the above user code added)
2. Common.h, compiler.h, sys_api.h, sys_cfg.h and sys_iil.c from system folder
3. 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 [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 */\

In the usb_idl_cfg.h, ensure to set the mode to **USB_MANUAL_MODE**

Apply Tasking Patches

1. In addition copy the cstart.asm, nommu.lsl 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.lsl 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 trans-

mission 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, .lsl 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

2.4 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.

Copy the folder USB_V4.2\Demo_Application\Blink_LED\Thesycon to C:\

Build the VC++ Project workspace from USB_V4.2\Demo_Application\Blink_LED\TC1130_USB_BlinkyLED folder. This generates the TC1130_USB_BlinkyLED in USB_V4.2\Demo_Application\Blink_LED\TC1130_USB_BlinkyLED\Debug folder.

Run the .elf generated and do the following steps:

1. Load the .elf present in the application folder through source navigator/ crossview respectively.
2. Open the MTTY using the baud rate 115200.
3. Run the application.
4. Plug and play the device.
5. Install the driver present in Driver folder usbiov.inf.
6. Open the BLINK LED application in GUI folder and follow the following steps.

Click on **Open Device IF1, Configure AS1**
Click on **Bind Out pipe to EP2**
Click on **Bind IN pipe to EP1**
Click on **Start Read-thread.**

Click on **Start blink, stop Blink, Start blink, faster and slower** and see the LED blinking accordingly.

http://www.infineon.com

Published by Infineon Technologies AG