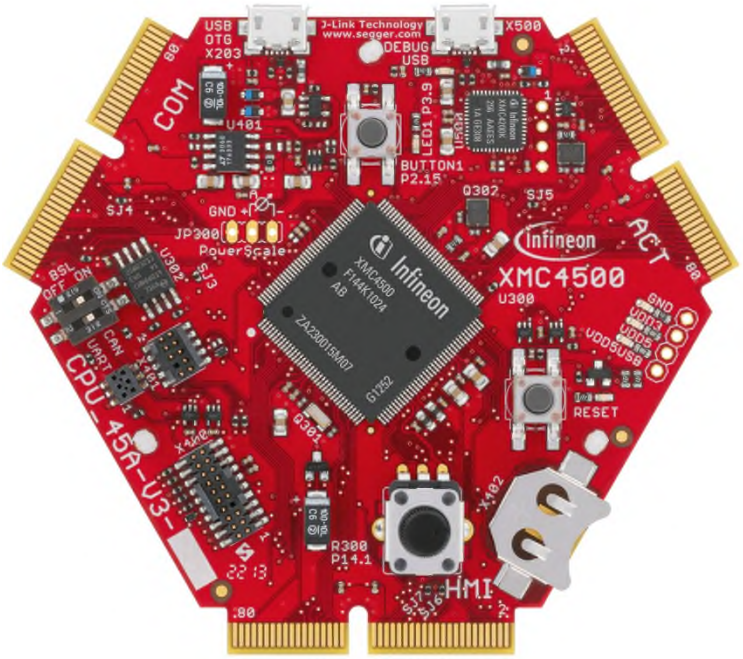
XMC4500 USB Host Virtual COM Port (VCOM) Example

Getting Started V1.0

Contents

Contents

Overview



- This example demonstrates the implementation of USB host virtual COM (VCOM) port functionality. It is based on the free LUFA USB host stack implementation which is ported to XMC4500 family and provided within this example.
- Within this documentation you will be guided through all the building blocks of a typical USB host VCOM application on the XMC4500 family. You will be able to connect a VCOM device to the host and perform data transfers access to and from the device.
- As a result you will be enabled to implement your own USB host VCOM functionality on the XMC4500 family.

Contents

Requirements - hardware

Two XMC4500 General Purpose CPU_45A-V3 boards

- One will behave as a USB host and another will behave as a USB device

USB cable micro-A plug to micro-B plug for host example

USB cable micro-B plug to standard-A plug for debugging and serial port terminal

Requirements – hardware and software

- Windows laptop
- DAVE™ installed

- DAVE ™ (v4.1.4 or Higher)
- Download DAVE™ free of charge

  Link : DAVE™ 4 Download
- Download USB device VCOM app from DAVE™
- Serial port terminal software like Tera Term

  Link : http://ttssh2.osdn.jp/

Requirements – free software download

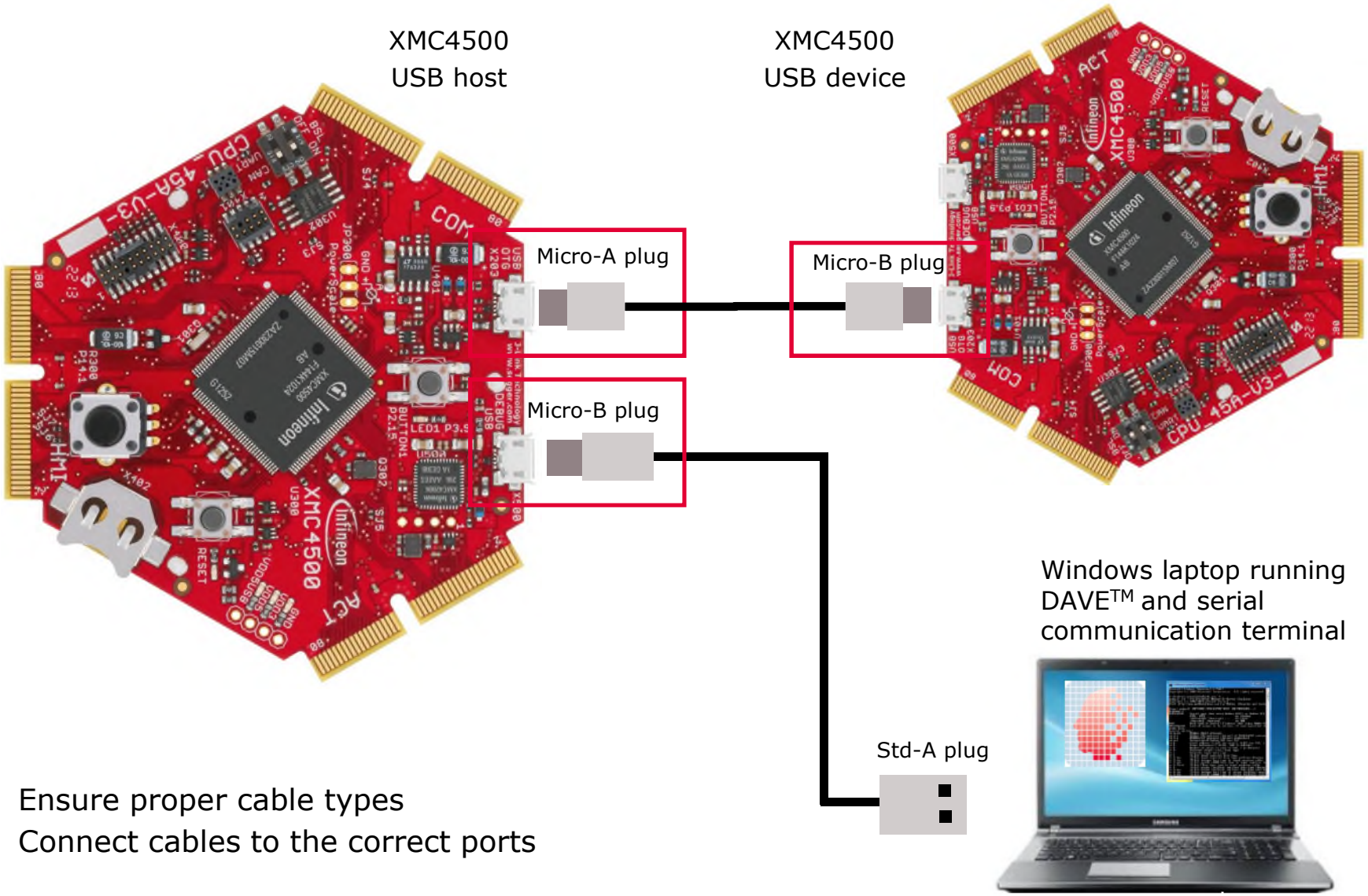
- DAVE ™ (v4.1.4 or Higher)
- Download DAVE™

  Link : DAVE™ 4 Download
- Download USB device VCOM app from DAVE™
- Serial port terminal software like Tera Term

  Link : http://ttssh2.osdn.jp/

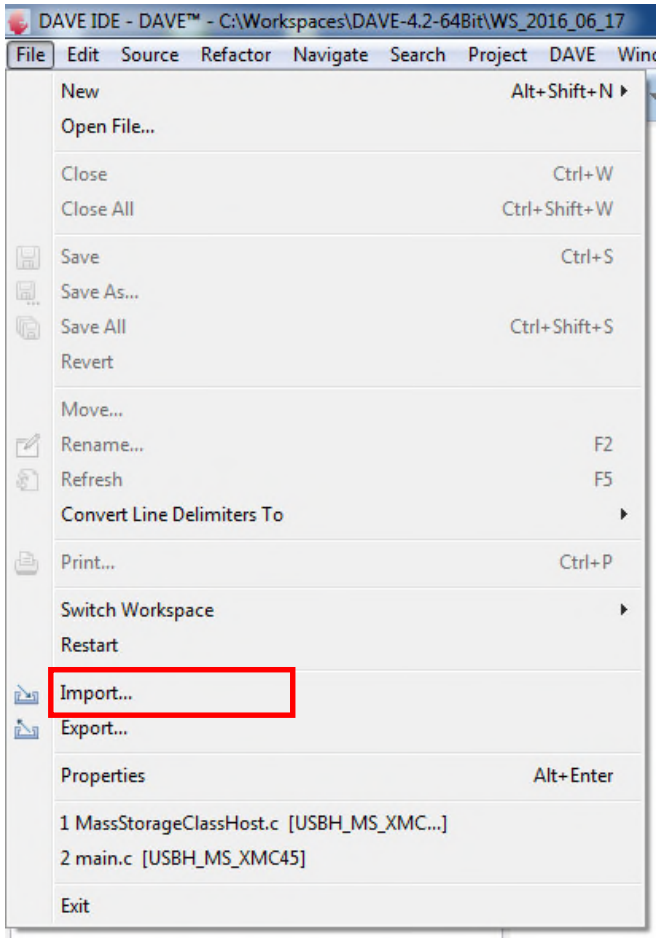
Contents

Setup - hardware



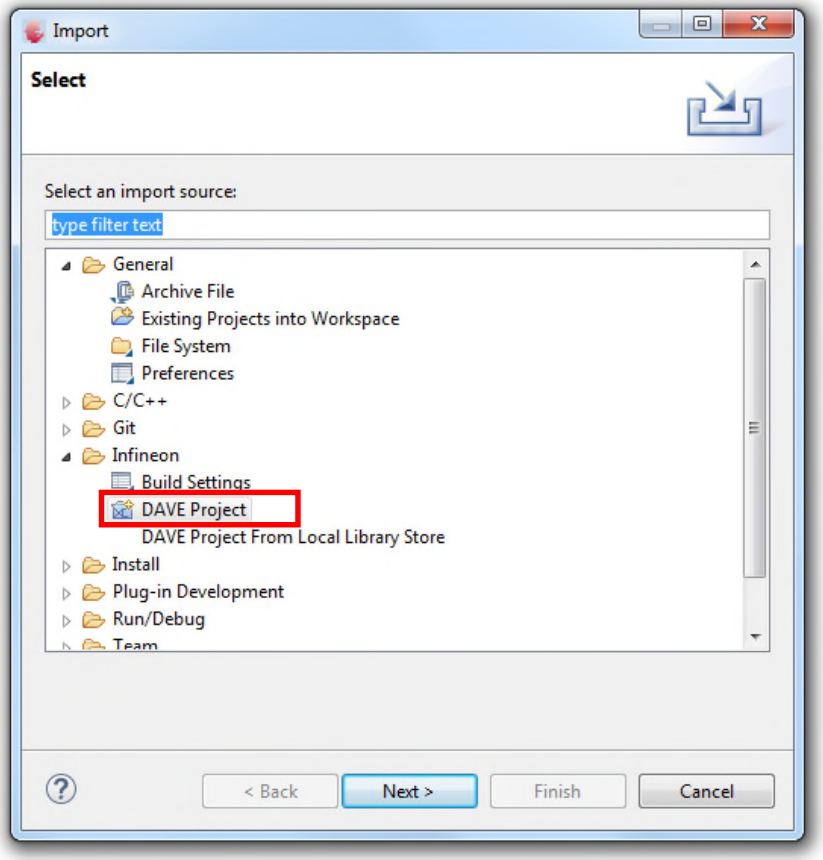
- Ensure proper cable types
- Connect cables to the correct ports

Setup – import VCOM example project in to DAVE™

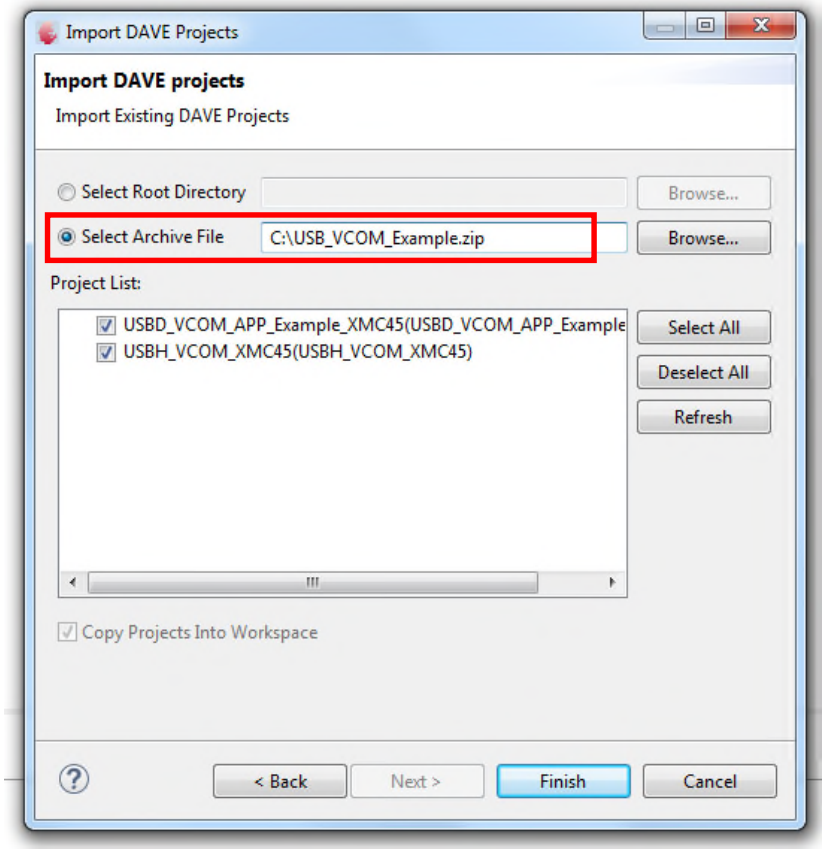


1

Setup – import VCOM example project in to DAVE™

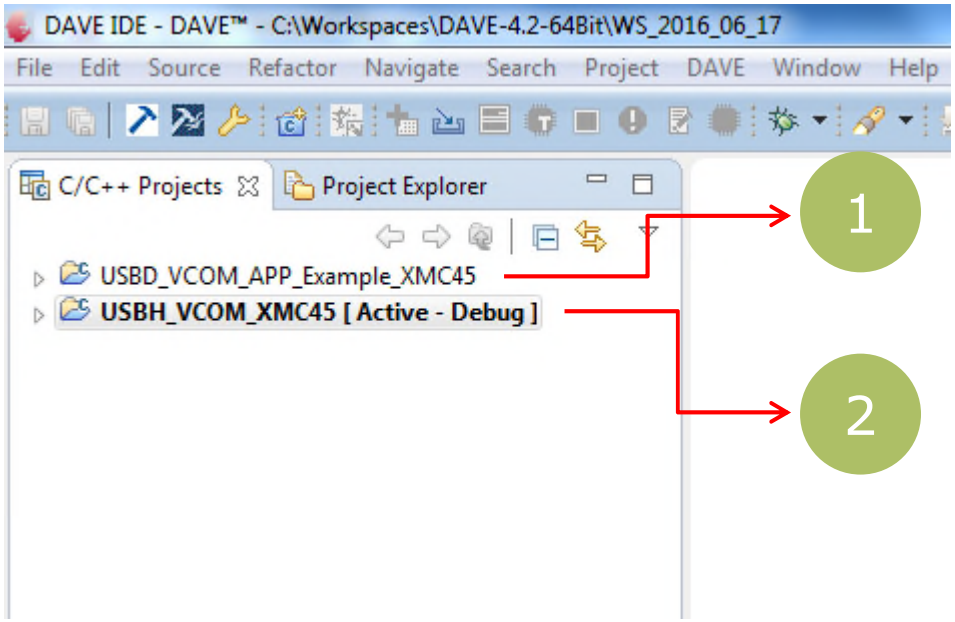


2



3

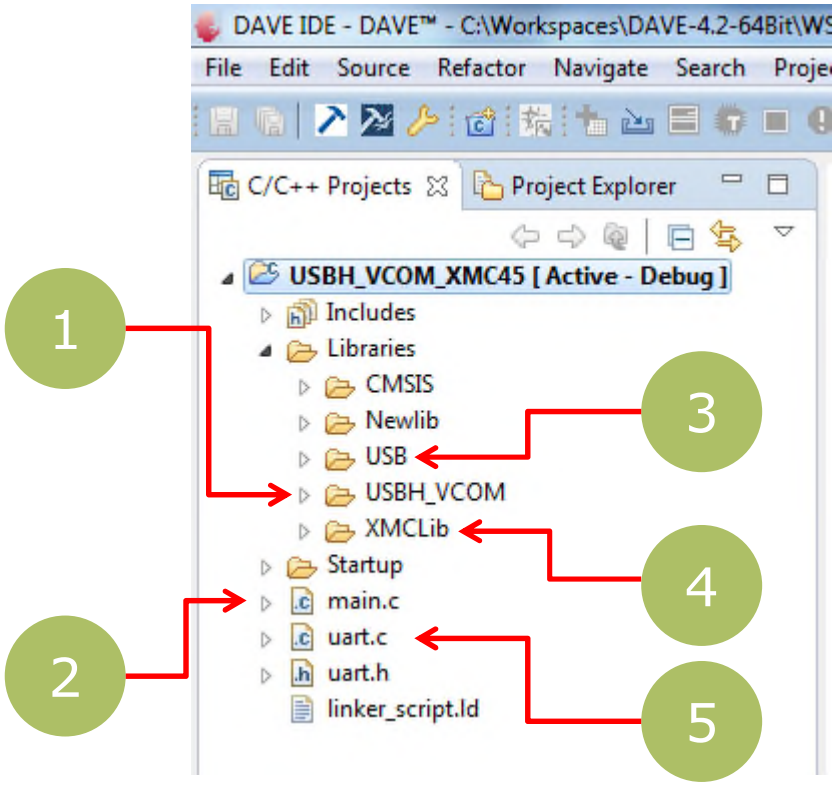
Setup –import VCOM example project in to DAVE™



1 USB VCOM Device project

2 USB VCOM Host project

Setup – import VCOM example project in to DAVE™



› Check the folder structure of the imported project.

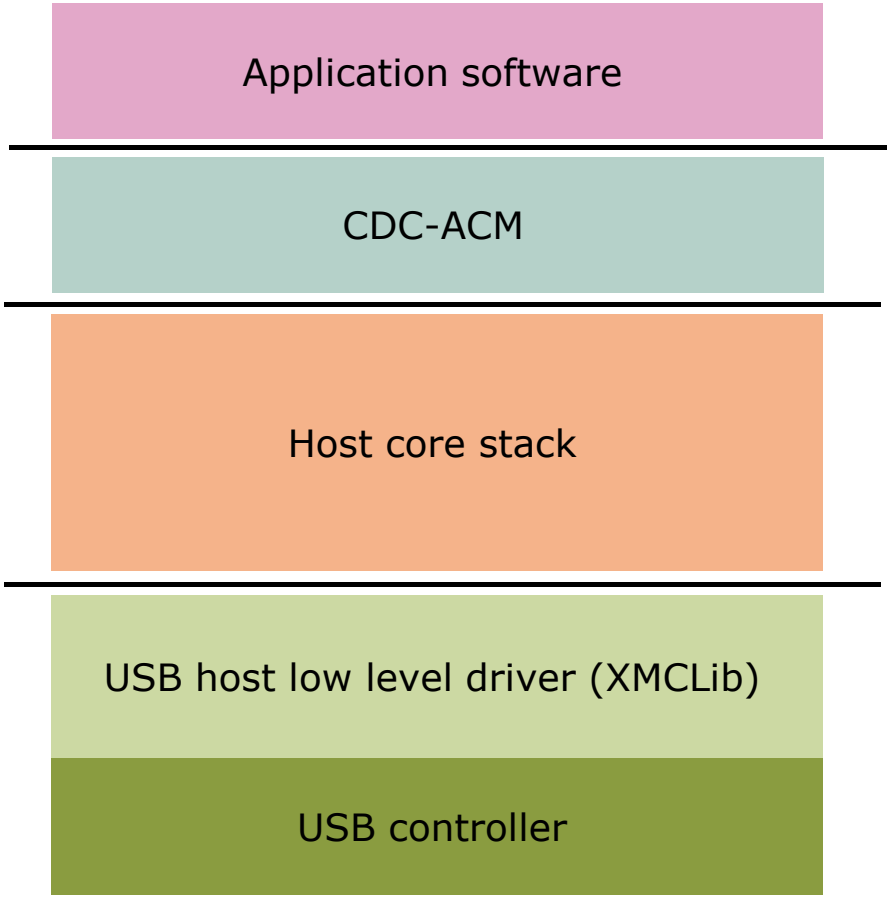
1. VCOM application code implementing USB data transmit and receive
2. The main file implementing the application task and echo data on UART
3. Free LUFA USB stack, CDC-ACM class and glue layer for XMC low level driver
4. XMCLib folder contains USB low level driver in file xmc_usb.c
5. Code for redirecting data to UART

Contents

Example application behavior

- First XMC4500 CPU_45A-V3 board is initialized as a USB host and provides VBUS on the USB port. This board runs host VCOM software.
- Second XMC4500 CPU_45A-V3 board is initialized as USB device. This board runs VCOM device software.
- When successfully enumerated, USB host recognizes the connected device as a CDC-ACM capable device (VCOM)
- First XMC4500 as USB host receives data from Windows laptop running serial terminal software like Tera Term and sends data to the device XMC4500 over USB
- Second XMC4500 as USB device loops back the received data on the USB
- First XMC4500 as USB host redirects received loop back data to debug port connected to Windows laptop running serial terminal software like Tera Term

Simplified example application architecture



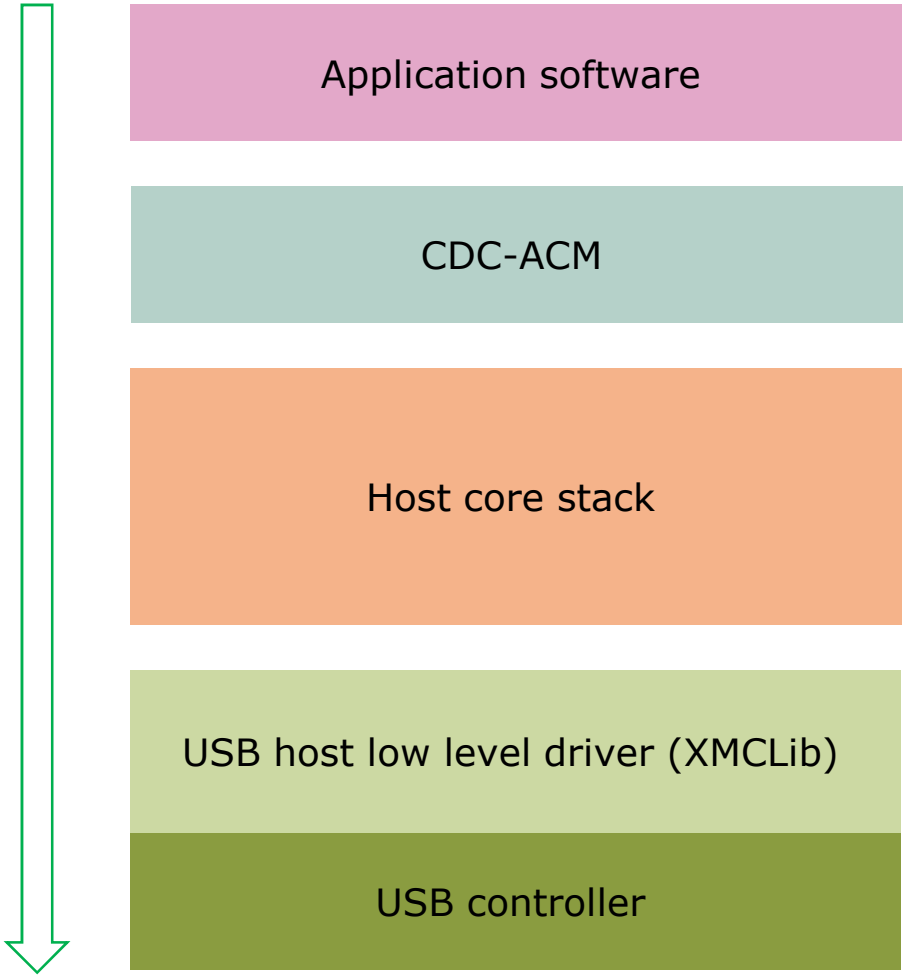
CDC_ACM send, receive data and notification interface

Bulk and interrupt pipe APIs

Low level driver based on USB host CMSIS API

USB stack and CDC-ACM sub-class implementation is based on a freely available stack known as LUFA

Example application data flow



Application software transfers byte received over UART to the CDC-ACM layer using send and receive APIs

CDC-ACM layer transfers the data to the host core stack. It also prepares to receive any notification messages on the interrupt endpoint

Host stack manages connections and enumeration. Provides mechanisms to transfers data through pipes

Low level driver based on USB host CMSIS API

Application – overview of main.c

main.c

```
48  */
49  int main(void)
50  {
51    uint8_t USBHostState;
52    int8_t ReceivedByte;
53    uint8_t *RxPtr;
54    uint32_t Rxlen;
55    uint32_t uart_data_count;
56
57
58    /*Configure USB and CPU clocks*/
59    ClockSetup();
60    /*Initialize UART channel*/
61    UART_init();
62    /*Select VBUS pin as P3.2*/
63    XMC_USBH_Select_VBUS(XMC_GPIO_PORT3, 2U);
64    /* Initializes the USB host driver. */
65    USB_Init_Host(&USBVCH_CB);
66
67    /*LED pin to indicate USB status*/
68    XMC_GPIO_SetMode(XMC_GPIO_PORT3, 9, XMC_GPIO_MODE_OUTPUT_PUSH_PULL);
69    XMC_GPIO_SetOutputHigh(XMC_GPIO_PORT3, 9);
70
```

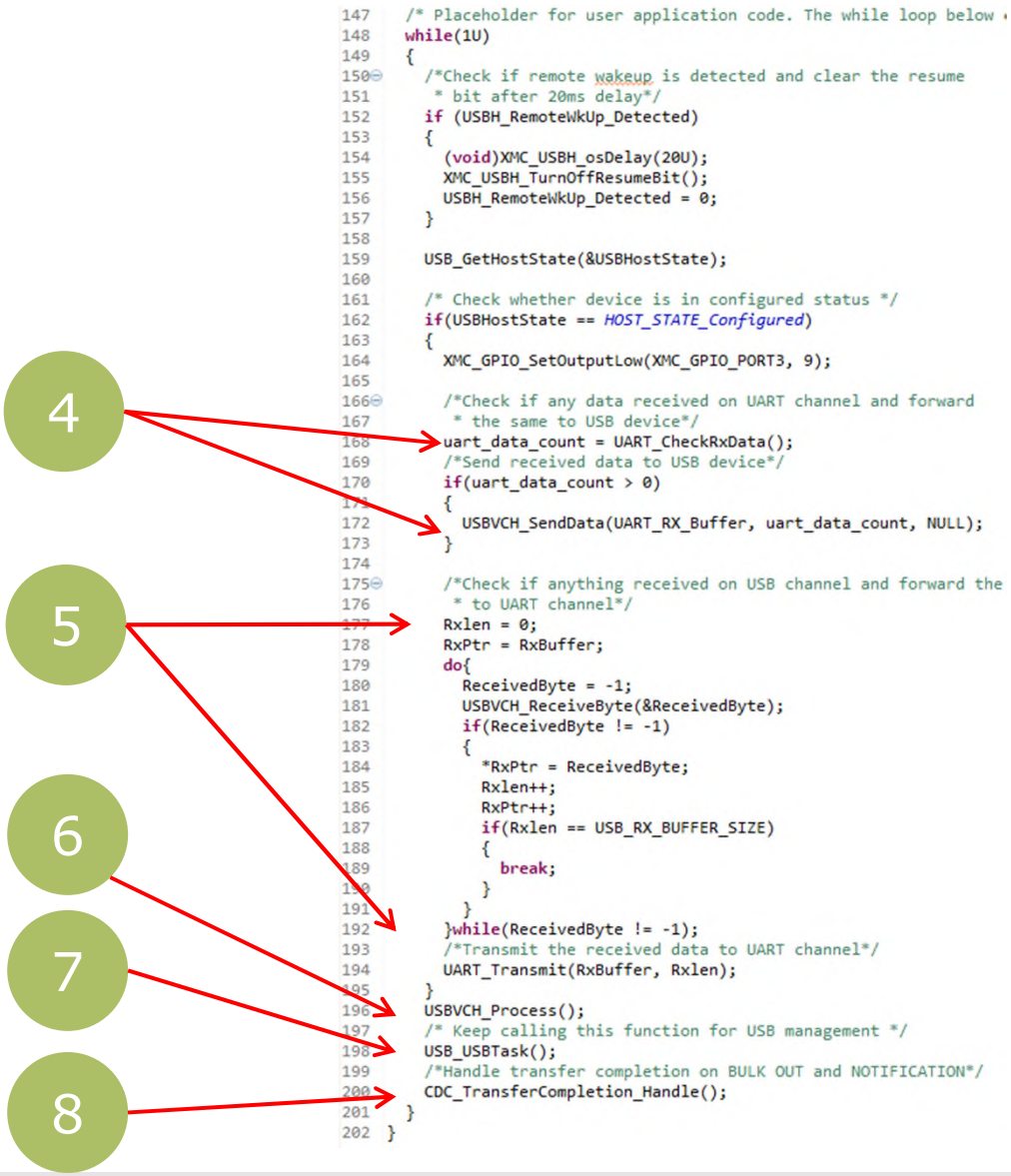
1

2

3

1. Clock setup for XMC4500
2. Select VBUS pin P3.2
3. Initialize USB host

Application – overview of main.c




```
147   /* Placeholder for user application code. The while loop below
148   while(1U)
149   {
150     /*Check if remote wakeup is detected and clear the resume
151      * bit after 20ms delay*/
152     if (USBH_RemoteWkUp_Detected)
153     {
154       (void)XMC_USBH_osDelay(20U);
155       XMC_USBH_TurnOffResumeBit();
156       USBH_RemoteWkUp_Detected = 0;
157     }
158
159     USB_GetHostState(&USBHostState);
160
161     /* Check whether device is in configured status */
162     if(USBHostState == HOST_STATE_Configured)
163     {
164       XMC_GPIO_SetOutputLow(XMC_GPIO_PORT3, 9);
165
166       /*Check if any data received on UART channel and forward
167        * the same to USB device*/
168       uart_data_count = UART_CheckRxData();
169       /*Send received data to USB device*/
170       if(uart_data_count > 0)
171       {
172         USBVCH_SendData(UART_RX_Buffer, uart_data_count, NULL);
173       }
174
175       /*Check if anything received on USB channel and forward the
176        * to UART channel*/
177       Rxlen = 0;
178       RxPtr = RxBuffer;
179       do{
180         ReceivedByte = -1;
181         USBVCH_ReceiveByte(&ReceivedByte);
182         if(ReceivedByte != -1)
183         {
184           *RxPtr = ReceivedByte;
185           Rxlen++;
186           RxPtr++;
187           if(Rxlen == USB_RX_BUFFER_SIZE)
188           {
189             break;
190           }
191         }
192       }while(ReceivedByte != -1);
193       /*Transmit the received data to UART channel*/
194       UART_Transmit(RxBuffer, Rxlen);
195     }
196     USBVCH_Process();
197     /* Keep calling this function for USB management */
198     USB_USBTask();
199     /*Handle transfer completion on BULK OUT and NOTIFICATION*/
200     CDC_TransferCompletion_Handle();
201   }
202 }
```

4. Receive data from UART and send to USB
5. Receive data from USB and send to UART
6. Fetch descriptors and match it to CDC-ACM sub-class and start notification pipe and bulk IN pipe
7. Manage connection events and initiate enumeration
8. Manage transfer completion events

Application – configure time delay API

- The USB stack uses timing delay of the order of several tens of milliseconds to comply with the specification
- Port the time delay API if required. In the example code, SysTick timer is used
- Ensure timer is calibrated

USB_Glue.c

```
622  uint8_t XMC_USBH_osDelay(uint32_t MS)
623  {
624    USBH_GLUE_StartTimer(MS);
625    while(USBH_GLUE_IsTimerRunning());
626
627    return 0;
628  }
```

Application - callbacks

- Application registers for the following event callbacks

USB_Glue.h

```
/*Structure has members to interface the LUFA stack with
 * class implementation and the application*/
typedef struct USBH_GLUE_APP_IF
{
  USBH_Port_Event_Handler_cb PortEventHandler;        /*Callback function to be executed on
                                                        occurance of a port event*/
  USBH_Pipe_Event_Handler_cb PipeEventHandler;        /*Callback function to be executed on
                                                        occurance of a pipe event*/
  USBH_EnumerationComplete_cb EnumerationCompleteCb;  /*Callback function to be executed after
                                                        device enumeration is complete*/
  /*Following function pointers will be called from the implementation of certain LUFA APIs*/
  USBH_BytesInPipe GetBytesInPipe;                    /*Function will be executed when a call
                                                        to LUFA API Pipe_BytesInPipe is made.*/

  USBH_PipeRead GetReadByte;                          /*Function will be executed when a call
                                                        to LUFA API Pipe_Read_8 is made.*/

  USBH_PipeIsINReceived IsINReceived;                 /*Function will be executed when a call
                                                        to LUFA API Pipe_IsINReceived is made.*/
}USBH_GLUE_APP_IF_t;
```

Application – Implement and register required callbacks

1 Implement call back function

1

```
157  /*Callback function executed on port interrupt*/
158  void CDC_USB_PortCb(uint8_t port, uint32_t event)
159  {
160    if(event & XMC_USBH_EVENT_DISCONNECT)
161    {
162      XMC_GPIO_SetOutputHigh(XMC_GPIO_PORT3, 9);
163      Driver_USBH0.PipeDelete(USBHost_Pipe_State[0].pipe_handle);
164      Driver_USBH0.PipeDelete(USBHost_Pipe_State[1].pipe_handle);
165      Driver_USBH0.PipeDelete(USBHost_Pipe_State[2].pipe_handle);
166      Driver_USBH0.PipeDelete(USBHost_Pipe_State[3].pipe_handle);
167        /*Reset the data handling indices*/
168      USBH_VCOM_RX_cur_index = 0;
169      USBH_VCOM_RX_prev_index = 0;
170    }
171    if(event & XMC_USBH_EVENT_REMOTE_WAKEUP)
172    {
173      /*This flag is set to remember the occurrence of remote wakeup event and
174       * to return from ISR immediately. This helps to time a 20ms delay in
175       * the context of the application main loop rather than inside the ISR context.
176       * The reason for this is that the example uses a timer interrupt whose priority is
177       * lower than the USB interrupt and therefore would result in a deadlock.*/
178      USBH_RemoteWkUp_Detected = 1;
179    }
180  }
```

Application – Implement and register required callbacks

2

```
31 /*Callback functions to be called from USB glue layer*/
32 USBH_GLUE_APP_IF_t USBVCH_CB =
33 {
34     .GetBytesInPipe = USBVCH_Pipe_BytesInPipe,
35     .GetReadByte = USBVCH_Pipe_Read_8,
36     .IsINReceived = USBVCH_Pipe_IsINReceived,
37     .PipeEventHandler = USBH_VCOM_Rx_Data_Handler,
38     .PortEventHandler = CDC_USB_PortCb
39 };
```

3

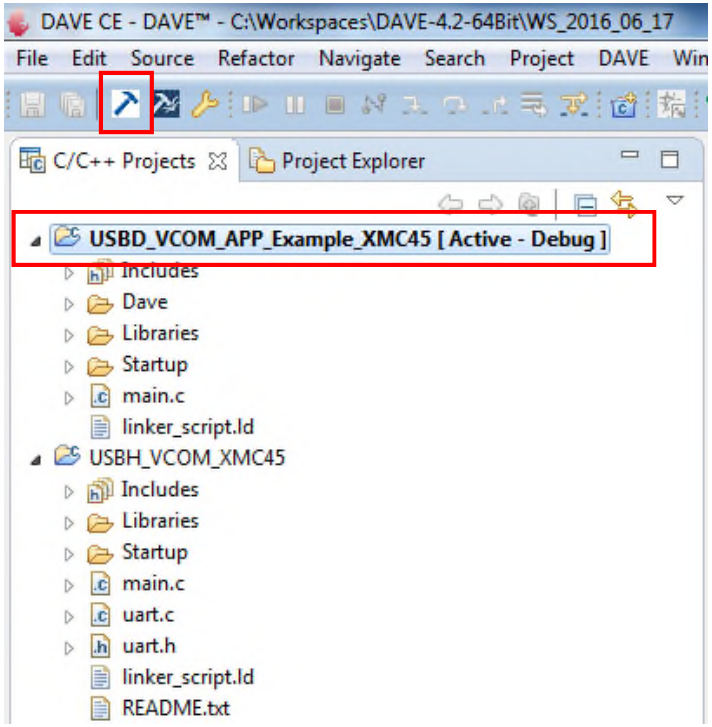
```
49 int main(void)
50 {
51   uint8_t USBHostState;
52   int8_t ReceivedByte;
53   uint8_t *RxPtr;
54   uint32_t Rxlen;
55   uint32_t uart_data_count;
56
57
58   /*Configure USB and CPU clocks*/
59   ClockSetup();
60   /*Initialize UART channel*/
61   UART_init();
62   /*Select VBUS pin as P3.2*/
63   XMC_USBH_Select_VBUS(XMC_GPIO_PORT3, 2U);
64   /* Initializes the USB host driver. */
65   USB_Init_Host(&USBVCH_CB);
66
```

2 Register call back function

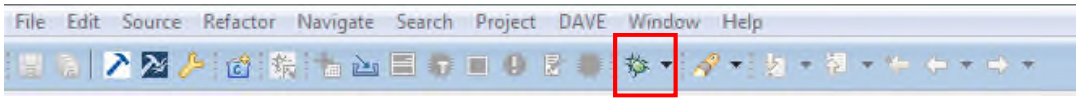
3 Call initialization function

Contents

How to test – build and download device project



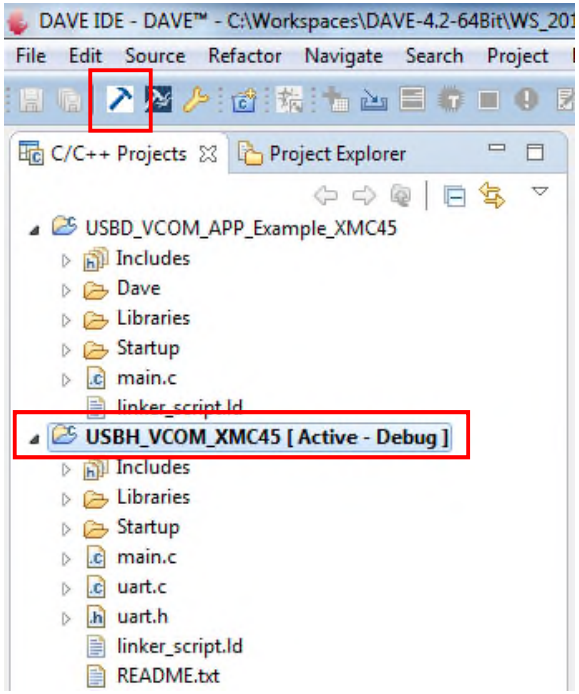
1



2

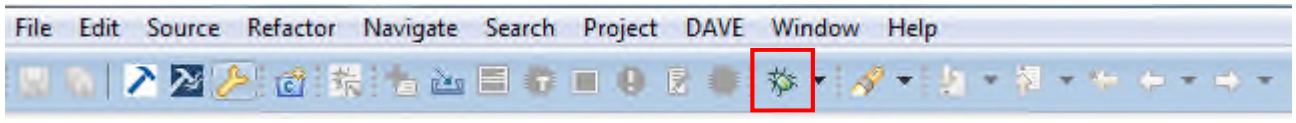
1. Make the device project 'Active' and build **device project**
2. Download image into **second** XMC4500 General Purpose board. Image now resides in flash memory
3. Reset XMC4500 General Purpose board to start executing the image

How to test – build and download host project

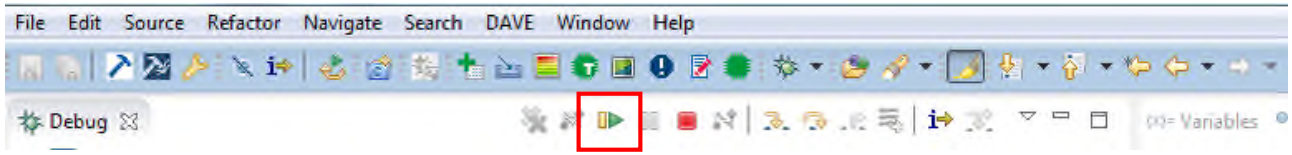


1

1. Make the host project 'Active', build and download **host project** into **first** General Purpose board.
2. Start debugger
3. Start the execution by pressing the run button



2



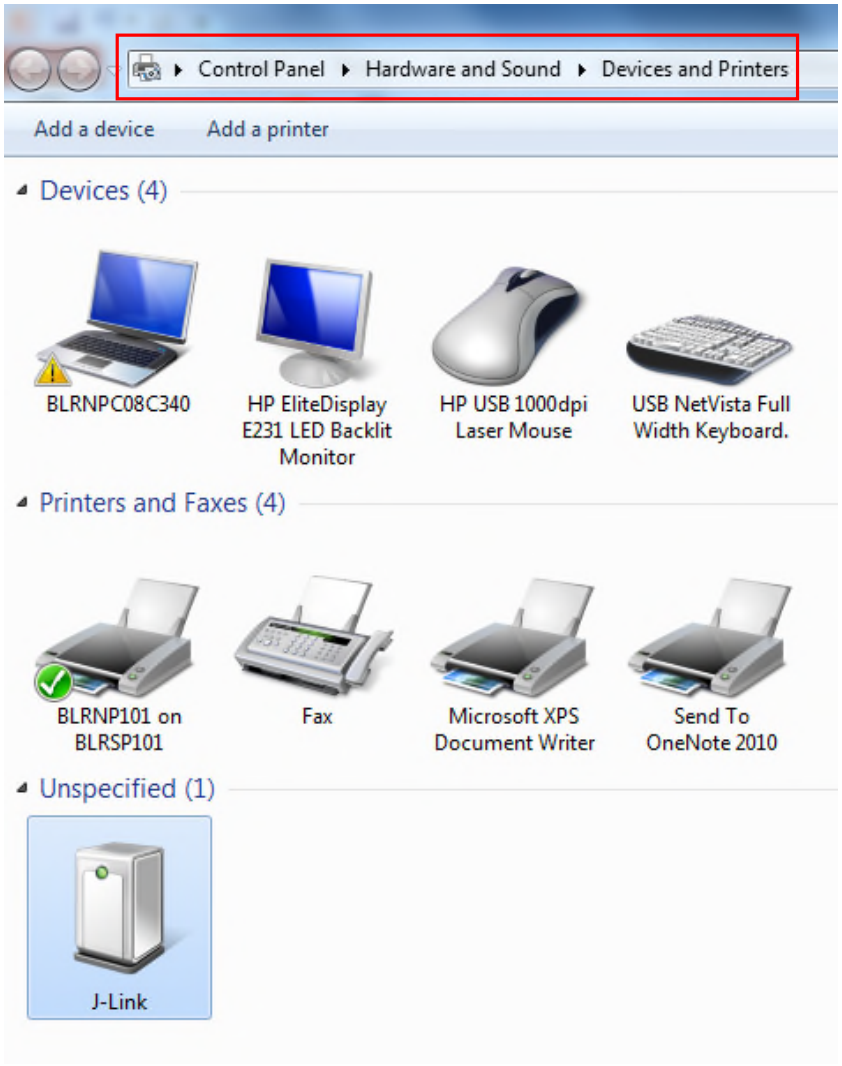
3

How to test

1. Connect the USB cable between the host and device XMC4500 General Purpose boards. Refer slide 10
2. Open serial port terminal software on Windows laptop (example Tera Term)
   - See next slide on how to locate associated COM port
3. Configure the terminal software for 19200 baud rate, 8 bit data, 1 stop bit, no parity
4. Type some characters. The typed characters will be looped back and visible on the terminal.

Finding associated COM port on Windows

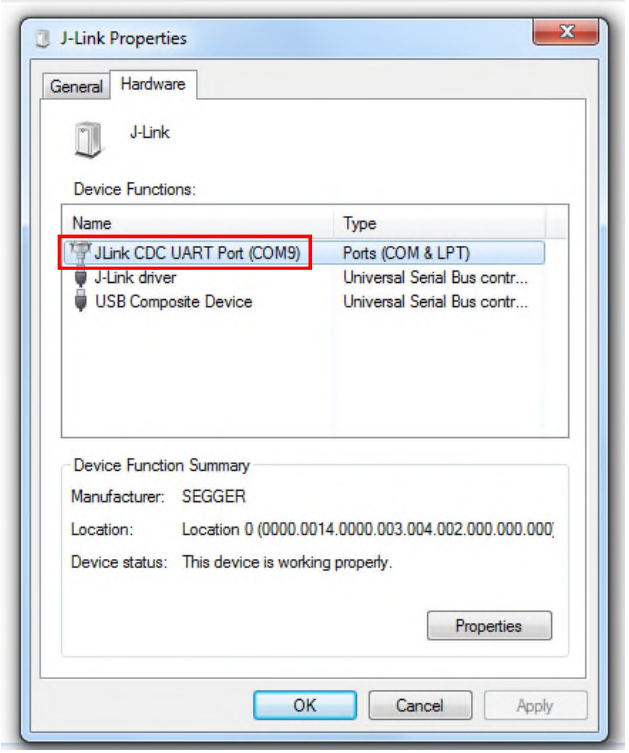
1



2

1 Open Control Panel -> Hardware and Sound -> Devices and Printers

2 Right click on J-Link and select Properties, then select Hardware tab. COM port is indicated.



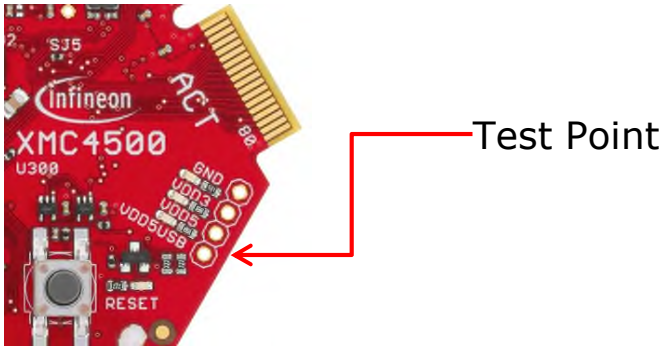
Supported features

- ACM Requests – SetControlLineState, SetLineCoding, SendBreak
- Only ACM Notification - Serial State is implemented

Contents

Debug Hints

- Check VBUS is available and within permissible range (4.4V to 5.25V) on the USB host port receptacle. Lower VBUS voltages due to excessive current draw may cause USB devices to disconnect intermittently or fail enumeration. Try connecting a different USB device.



- Long cables may cause malfunction due to signal quality issues. Use short cables if possible.
- Check if XMC4500 VCOM device example application is running properly by connecting device to a Windows host running Tera Term. XMC4500 device will loop back characters typed on terminal.

Contents

Further reading

- USB specifications and Micro-USB cables and connectors specification

  http://www.usb.org/developers/docs/usb20_docs/

- Communication devices class specifications
  - Class definitions for communication devices v1.2
  - Communications class, subclass specifications for PSTN devices

  http://www.usb.org/developers/docs/devclass_docs/

- CMSIS USB host API specification

  https://www.keil.com/pack/doc/CMSIS/Driver/html/group__usb__interface__gr.html

Further reading

- LUFA USB stack

  http://www.fourwalledcubicle.com/index.php

- Books
  - USB Complete: The Developer's Guide, Jan Axelson
  - USB Embedded Hosts, Jan Axelson

Contents

Limitations

- Hubs are not supported
- Over current protection is not supported
- DMA Transfers are not supported
- Low level driver does not support isochronous transfers

Part of your life. Part of tomorrow.

Infineon