

# Freescale Processor Expert USB Components Quick Start Guide

---

**PRODUCT:** Freescale USB Stack

---

**PRODUCT VERSION:** 4.1.0

---

**DESCRIPTION:** Freescale Processor Expert USB Components Quick Start Guide

---

**RELEASE DATE:** March 20<sup>th</sup>, 2013

## **How to Reach Us:**

### **Home Page:**

[www.freescale.com](http://www.freescale.com)

### **Web Support:**

<http://www.freescale.com/support>

### **USA/Europe or Locations Not Listed:**

Freescale Semiconductor, Inc.  
Technical Information Center, EL516  
2100 East Elliot Road  
Tempe, Arizona 85284  
1-800-521-6274 or +1-480-768-2130  
[www.freescale.com/support](http://www.freescale.com/support)

### **Europe, Middle East, and Africa:**

Freescale Halbleiter Deutschland GmbH  
Technical Information Center  
Schatzbogen 7  
81829 Muenchen, Germany  
+44 1296 380 456 (English)  
+46 8 52200080 (English)  
+49 89 92103 559 (German)  
+33 1 69 35 48 48 (French)  
[www.freescale.com/support](http://www.freescale.com/support)

### **Japan:**

Freescale Semiconductor Japan Ltd.  
Headquarters  
ARCO Tower 15F  
1-8-1, Shimo-Meguro, Meguro-ku,  
Tokyo 153-0064  
Japan  
0120 191014 or +81 3 5437 9125  
[support.japan@freescale.com](mailto:support.japan@freescale.com)

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.



Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. ARC, the ARC logo, ARCanget, ARCform, ARChitect, ARCompact, ARCTangent, BlueForm, CASSEIA, High C/C++, High C++, iCon186, MetaDeveloper, MQX, Precise Solution, Precise/BlazeNet, Precise/EDS, Precise/MFS, Precise/MQX, Precise/MQX Test Suites, Precise/RTCS, RTCS, SeeCode, TotalCore, Turbo186, Turbo86, V8 µ RISC, V8 microRISC, and VAutomation are trademarks of ARC International. High C and MetaWare are registered under ARC International.

All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2013. All rights reserved.

Rev. 09  
03/2013

## Table of Contents

<b>Freescale Processor Expert USB Components Quick Start Guide .....</b>	<b>i</b>
<b>1 USB_HOST_STACK.....</b>	<b>3</b>
1.1 Folder structure .....	3
1.2 Running HID mouse host example .....	4
1.2.1 Importing component .....	4
1.2.2 Generating Processor Expert source code.....	5
1.2.3 Creating USB_HID_HOST_STACK project for K60 using Processor Expert source code.	7
1.2.4 Running the example .....	17
<b>2 USB_OTG_STACK .....</b>	<b>22</b>
2.1 Reference material .....	22
2.2 Folder structure .....	22
2.3 Running USB_HID_OTG example .....	23
2.3.1 Generating Processor Expert source code.....	23
2.3.2 Creating USB_HID_OTG_STACK project for K60 using Processor Expert .....	24
2.3.3 Running the example .....	42
<b>3 USB_HID_CLASS.....</b>	<b>46</b>
3.1 Architecture overview .....	46
3.2 Folder structure .....	46
3.2.1 Delivered package structure .....	46
3.2.2 USB HID example structure .....	48
3.3 Running USB HID example .....	49
3.3.1 USB HID mouse device example .....	49
3.3.2 USB HID mouse host example.....	54
3.3.3 Running the USB HID mouse host example.....	58
3.3.4 USB HID keyboard OTG example.....	62
<b>4 USB_CDC_CLASS .....</b>	<b>73</b>
4.1 Architecture overview .....	73
4.2 Folder structure .....	73
4.2.1 USB CDC component package.....	73
4.2.2 CDC device example structure.....	74
4.2.3 CDC host example structure .....	75
4.3 USB CDC example.....	75
4.3.1 Importing component .....	75
4.3.2 CDC device example .....	77
4.3.3 CDC host example.....	87
<b>5 USB_MSD .....</b>	<b>93</b>
5.1 Folder structure .....	93
5.1.1 USB MSD component package.....	93
5.1.2 USB MSD example structure .....	94
5.2 USB MSD example .....	94
5.2.1 Importing component .....	94
5.2.2 MSD device example .....	96
5.3 Running MSD device example .....	101
<b>6 USB_AUDIO_CLASS.....</b>	<b>103</b>
6.1 Architecture overview .....	103
6.2 Folder structure .....	103
6.2.1 USB Audio component package structure.....	103
6.2.2 AUDIO device example structure .....	105
6.2.3 AUDIO host example structure.....	105

6.3 USB AUDIO example .....	105
6.3.1 Importing component .....	105
6.3.2 AUDIO device example .....	107
6.3.3 AUDIO host example .....	116
<b>7 USB HID Device Example with MQX Lite.....</b>	<b>130</b>
7.1.1 Creating Processor Expert Project .....	130
7.1.2 Configure the Processor Expert Components .....	132
7.1.3 Edit User Code and Add Application Code.....	137
7.1.4 Running the USB HID Mouse Device Example .....	139

# 1 USB\_HOST\_STACK

This section describes how to use the USB\_HOST\_STACK component in the Processor Expert of Code Warrior. It also guides you how to run an USB Host example using this component.

## 1.1 Folder structure

This section describes the functionality of the USB Mouse Host example running on Tower K60 board. The following picture shows the structure of the example.

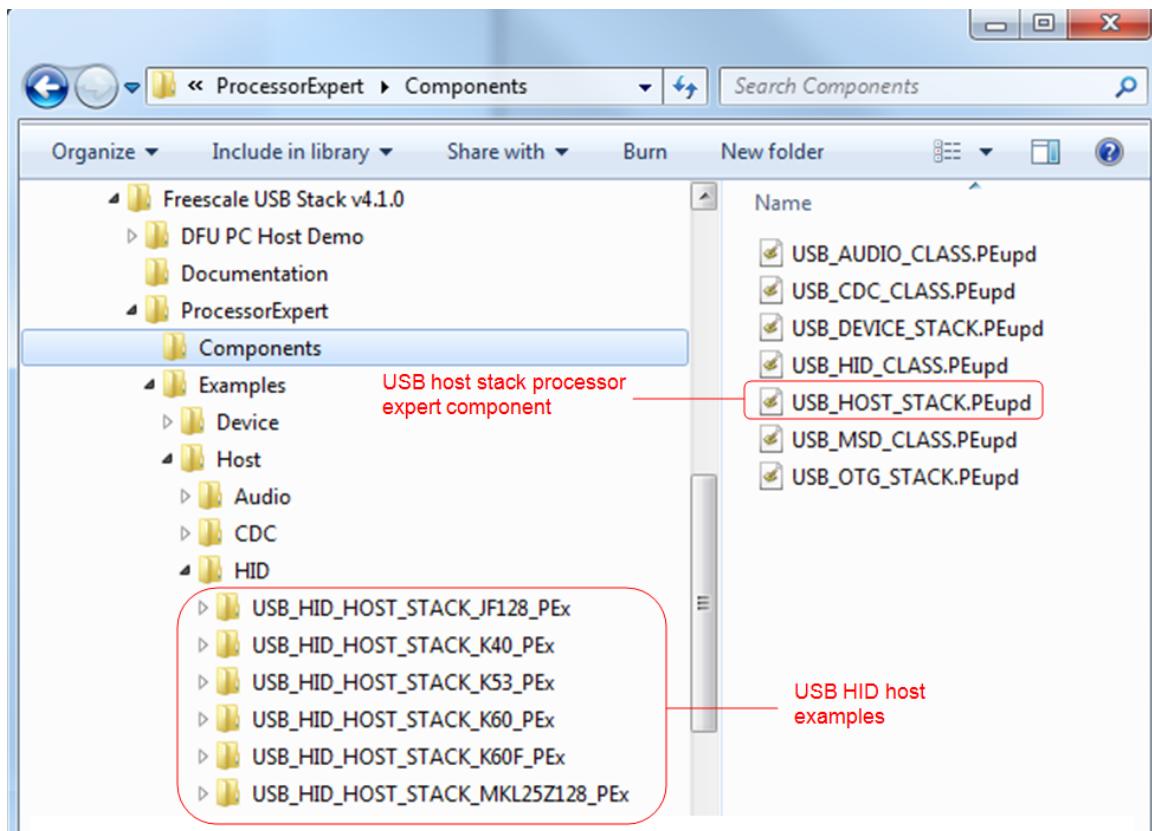


Figure 1-1 HID Mouse Host folder structure

## 1.2 Running HID mouse host example

### 1.2.1 Importing component

This section describes the steps to import the USB\_HOST\_STACK component used to run the USB HID Mouse Host example.

**Step1.** In CodeWarrior Development Studio, chose Processor Expert → Import Package.

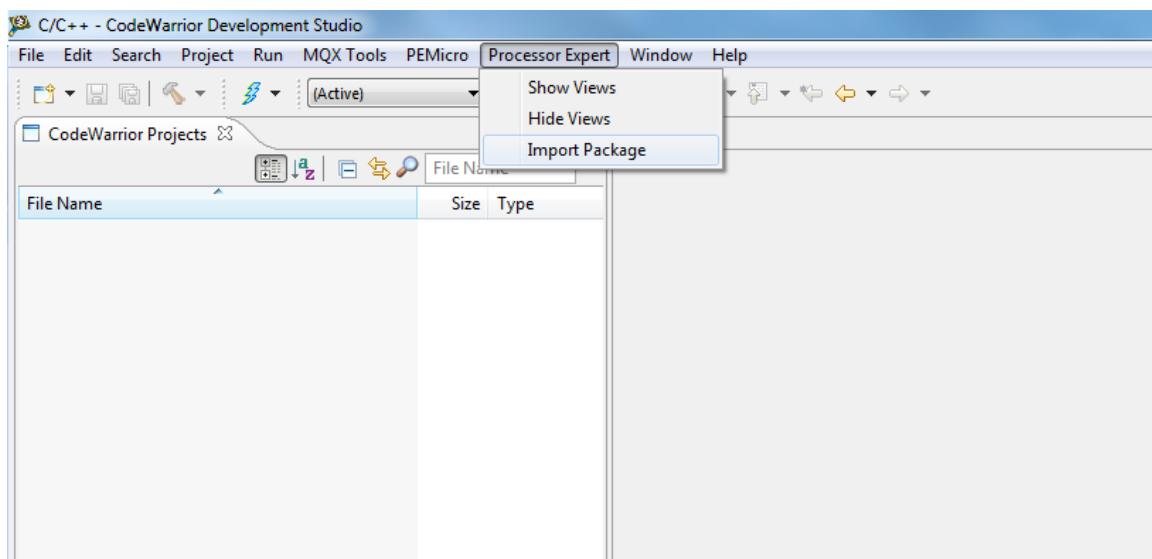


Figure 1-2 Processor Expert – Import Package

**Step2.** A browser window appears. Navigate to the folder that contains the USB\_HOST\_STACK component, chose it and click open.

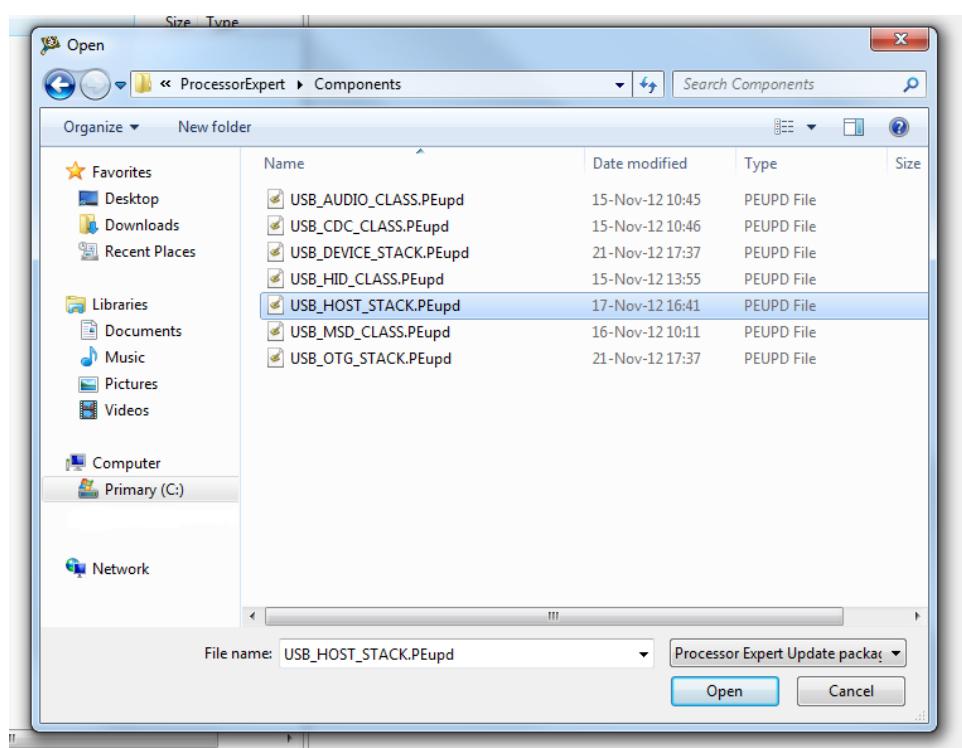


Figure 1-3 Delivered package

**Step3.** An Import Complete dialog box appears. Click OK for confirmation.

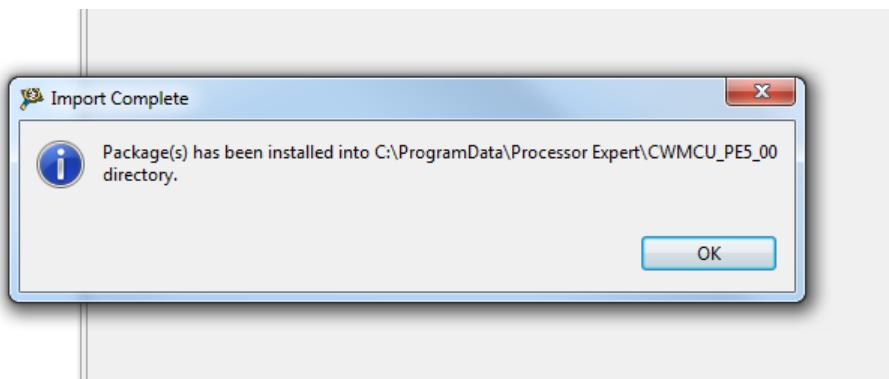


Figure 1-4 Import Complete

### 1.2.2 Generating Processor Expert source code

This section describes the steps to generate Processor Expert source code that is used to run the USB HID Mouse Host example.

**Step1.** Open USB\_HID\_HOST\_STACK\_K60\_PEx example with CW10.3 → From the CW10.3 toolbar, select File -> Import to open the Import window -> Open General folder -> Select Existing Projects into Workspace -> Click Next.

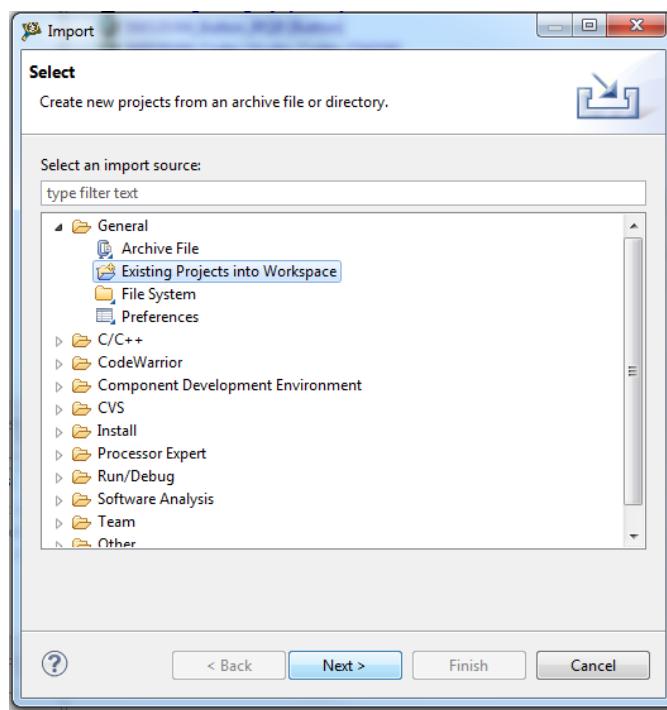


Figure 1-5 Import existing projects into Workspace

**Step2.** Click Browse button → Select USB\_HID\_HOST\_STACK\_K60\_PEx folder -> Click OK -> then click Finish button.

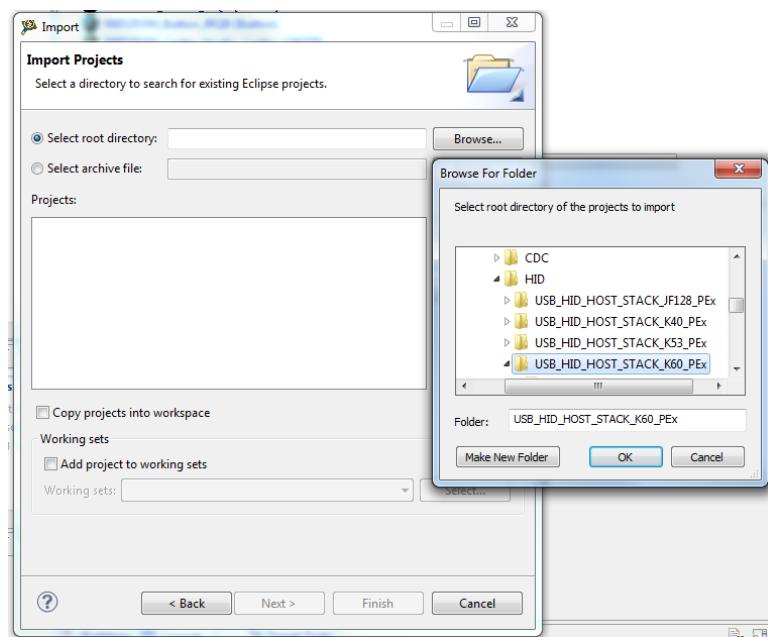


Figure 1-6 Import existing projects into Workspace

**Step3.** Generate Processor Expert code.

Open the project folder and select ProcessorExpert.pe -> Right click then select Generate Processor Expert Code field to generate source code.

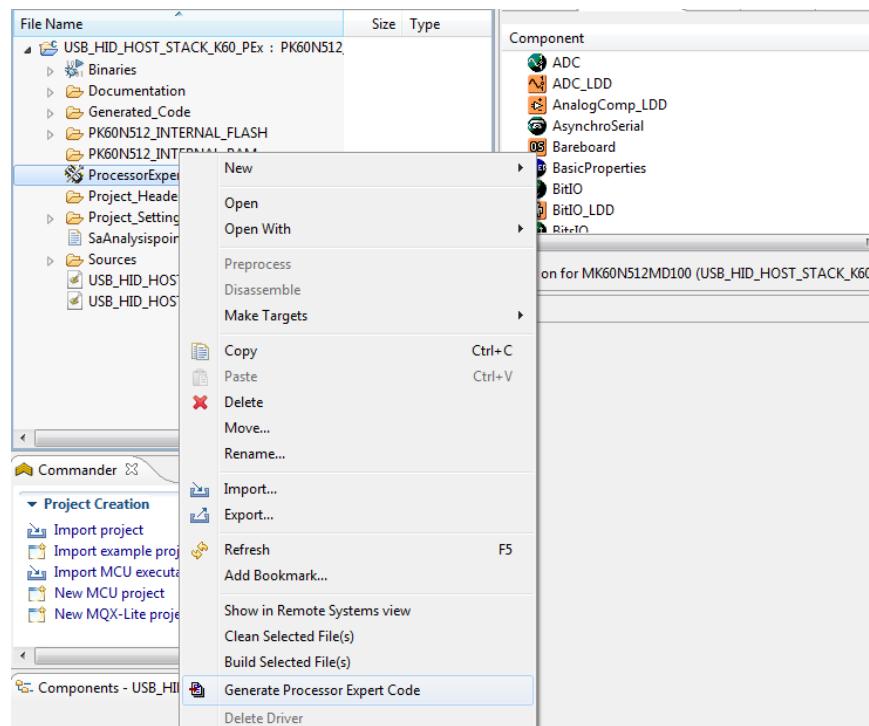


Figure 1-7 Generate Processor Expert code

### 1.2.3 Creating USB\_HID\_HOST\_STACK project for K60 using Processor Expert

**Step1.** Open CodeWarrior Development Studio 10.3 -> from the toolbar, select File -> New -> Bareboard Project. In the Project name text box type the desired name for the project -> Next.

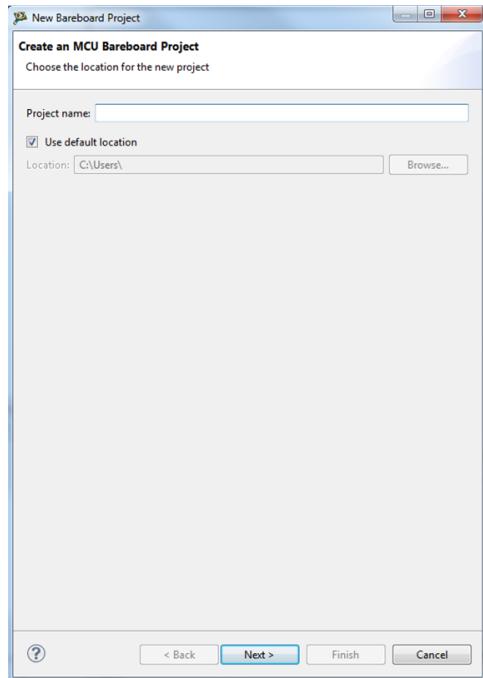


Figure 1-8 New Bareboard project

**Step2.** Select the device MK60DN512Z -> Next.

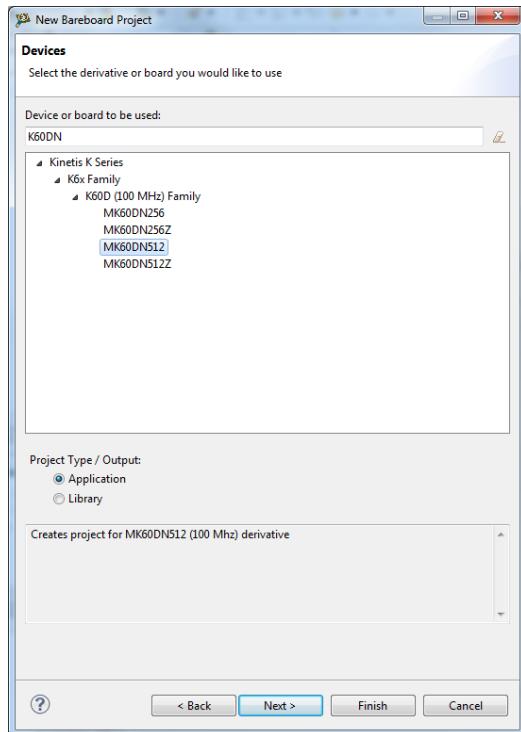


Figure 1-9 Devices window.

**Step3.** Select the connection used for this project -> Next to the Language and Build Tools Options window -> Next.

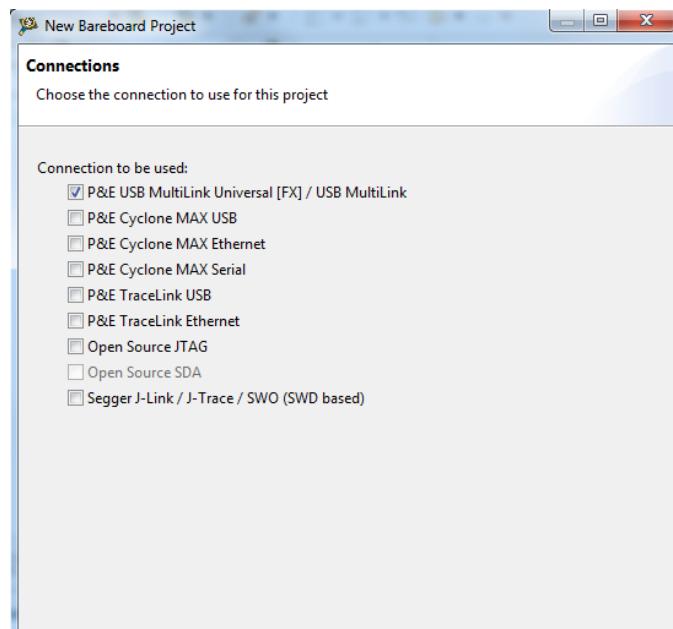


Figure 1-10 Connections window.

**Step4.** In Rapid Application Development window select Processor Expert and then Finish.

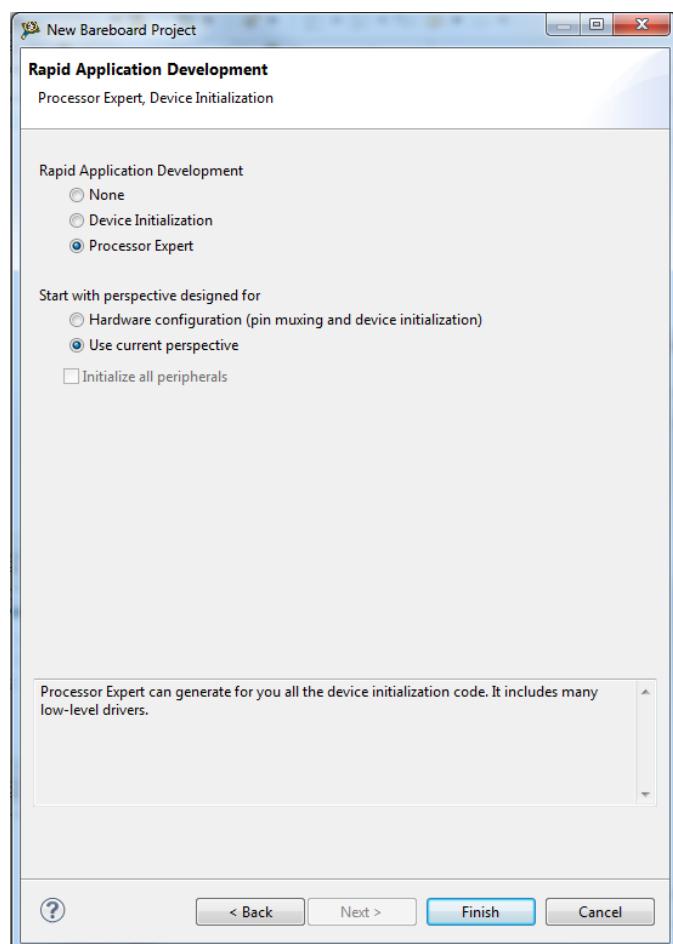


Figure 1-11 Rapid Application Development window.

**Step5.** From the CW10.3 toolbar, select Processor Expert → Select Show Views to open Components Library → Select the USB\_HOST\_STACK component in the Components Library and add it to project.

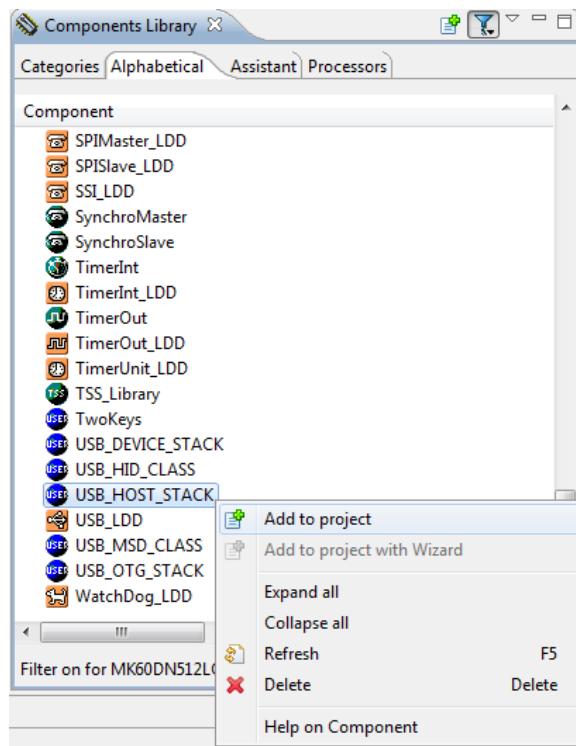


Figure 1-12 Component Library

**Step6.** Implement same as step 5 to add the TWR\_SER\_K60\_UART component into the project.

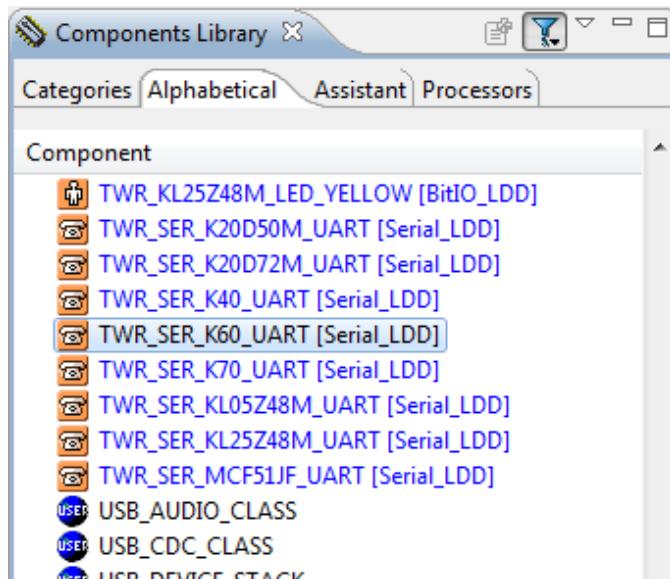


Figure 1-13 Serial\_LDD component

## Step7. Configure CPU clock to run USB module:

- Select CPU target from Project Panel window to setup clock source as in the following figure:
  - o Target: Cpu:MK60N512VMD100
  - o Clock source: External reference clock 50MHz
  - o MCG mode: PEE
  - o MCG output: 48 MHz
  - o PLL output: 48MHz
  - o Core clock: 48MHz
  - o Bus clock: 48MHz
  - o External bus clock: 24MHz
  - o Flash clock: 24MHz

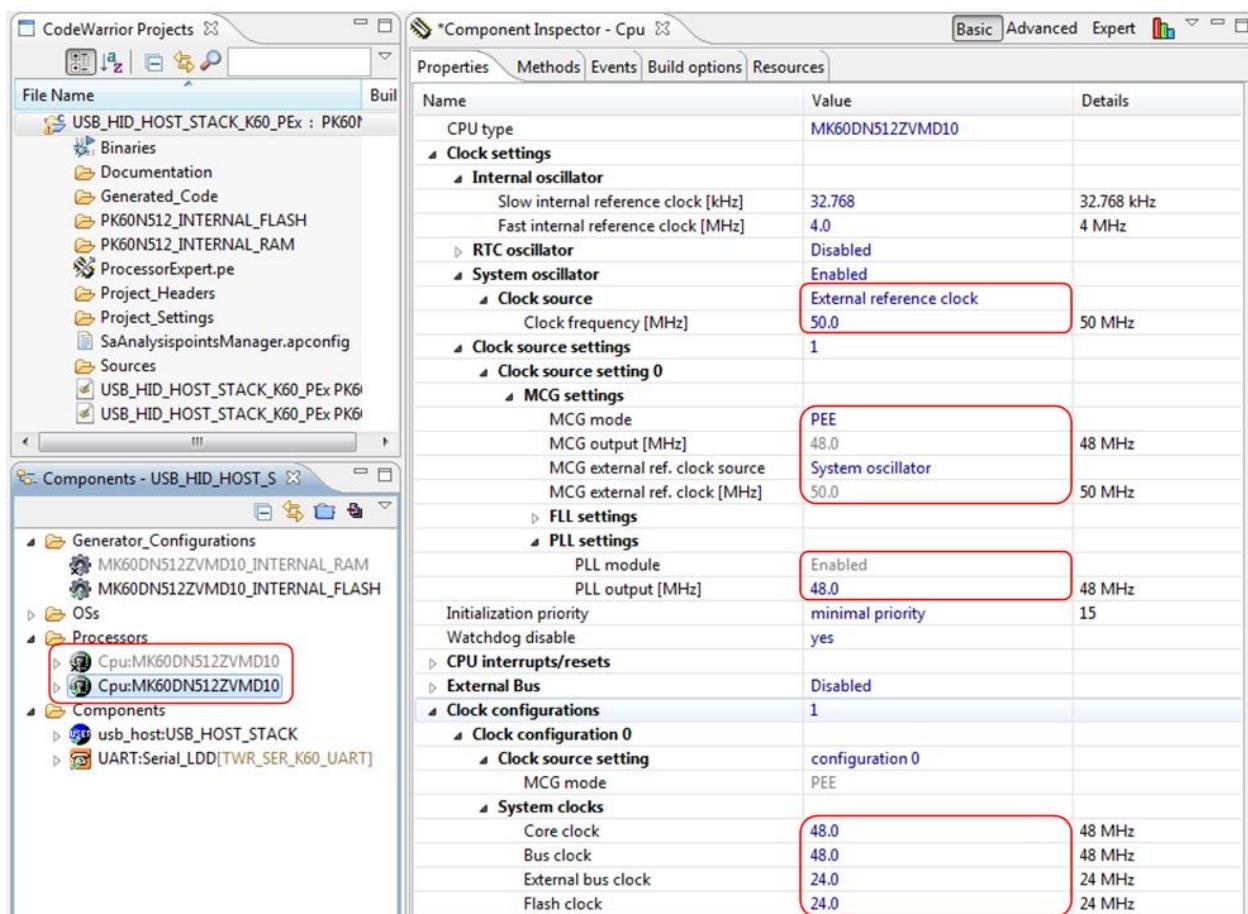


Figure 1-14 USB Clock source settings

- Chose advanced tab

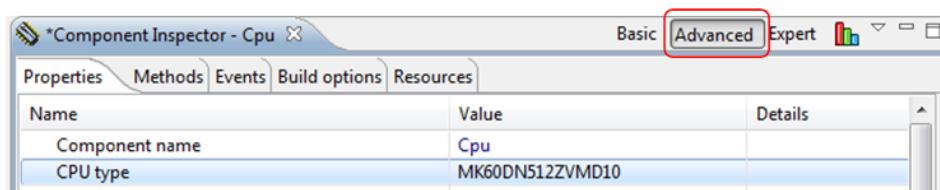


Figure 1-15 Open Advanced Tab

- PLL/FLL clock selection:
  - o PLL clock

Bus clock	48.0	48 MHz
External clock prescaler	Auto select	2
External bus clock	24.0	24 MHz
Flash clock prescaler	Auto select	2
Flash clock	24.0	24 MHz
<b>PLL/FLL clock selection</b>		
Clock frequency [MHz]	<b>48.0</b>	48 MHz
<b>USB clock settings</b>		
USB clock divider	Auto select	1
USB clock multiply	Auto select	1
USB clock	48.0	48 MHz

Figure 1-16 USB Clock configuration

#### Step8. Setup the USB\_HOST\_STACK component:

- Click on the USB\_HOST\_STACK component to select this target.
- Configure this component as shown below:

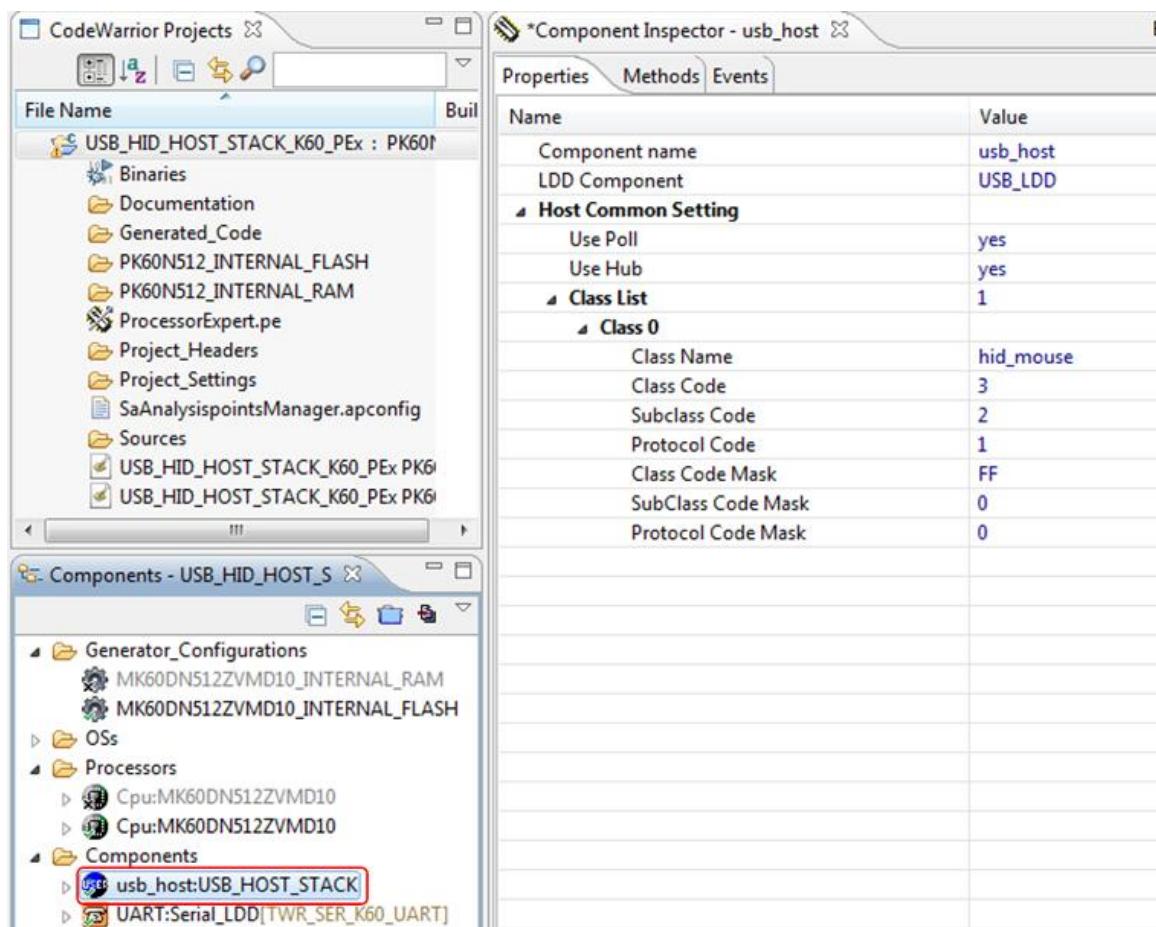


Figure 1-17 Host common settings

**Step9.** Setup the USB\_LDD component corresponding with the USB\_HOST\_STACK component.

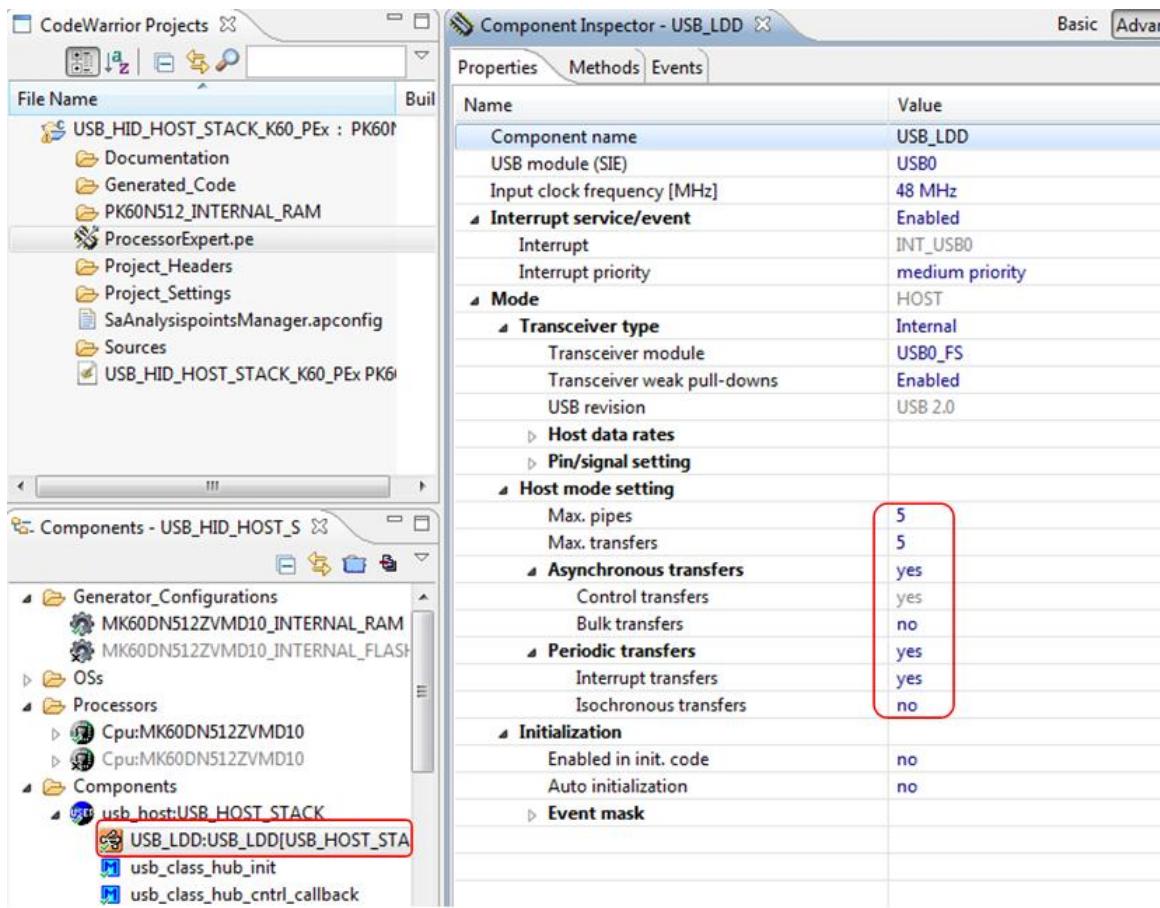


Figure 1-18 USB\_LDD component settings

**Step10.** Setup the TWR\_SER\_K60\_UART component:

- Click on the UART component to select this target.
- Configure this component as shown below:

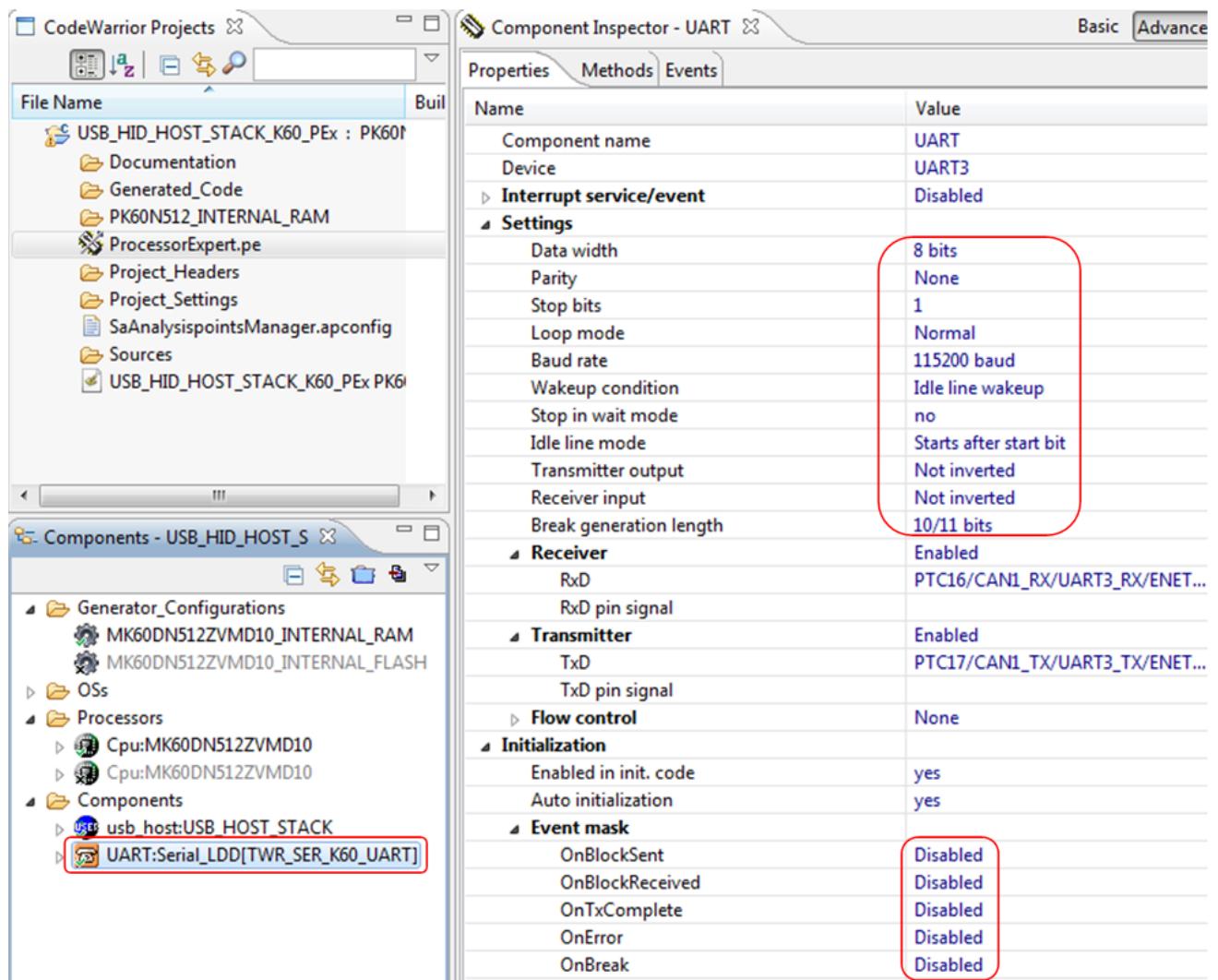


Figure 1-19 Serial\_LDD component settings

Component Inspector - UART

Name	Value	Details
Init	generate code	
Deinit	generate code	
Enable	don't generate code	
Disable	don't generate code	
SendBlock	generate code	
ReceiveBlock	generate code	
CancelBlockTransmission	don't generate code	
CancelBlockReception	don't generate code	
GetError	don't generate code	
GetSentDataNum	generate code	
GetReceivedDataNum	generate code	
GetTxCompleteStatus	generate code	
SetEventMask	don't generate code	
GetEventMask	don't generate code	
SelectBaudRate	don't generate code	
GetSelectedBaudRate	don't generate code	
SetLoopMode	don't generate code	
GetLoopMode	don't generate code	
GetStats	don't generate code	
ClearStats	don't generate code	
SendBreak	don't generate code	
GetBreak	don't generate code	
Main	generate code	

Figure 1-20 Serial\_LDD component methods

Component Inspector - UART

Name	Value
Event module name	Events
OnBlockReceived	don't generate code
Event procedure name	UART_OnBlockReceived
OnBlockSent	don't generate code
Event procedure name	UART_OnBlockSent
OnBreak	don't generate code
OnError	don't generate code
OnTxComplete	don't generate code

Figure 1-21 Serial\_LDD component events

**Step11.** Generate Processor Expert code:

After setting the components, select ProcessorExpert.pe in the project folder → Right click → Choose Generate Processor Expert Code field to generate source code.

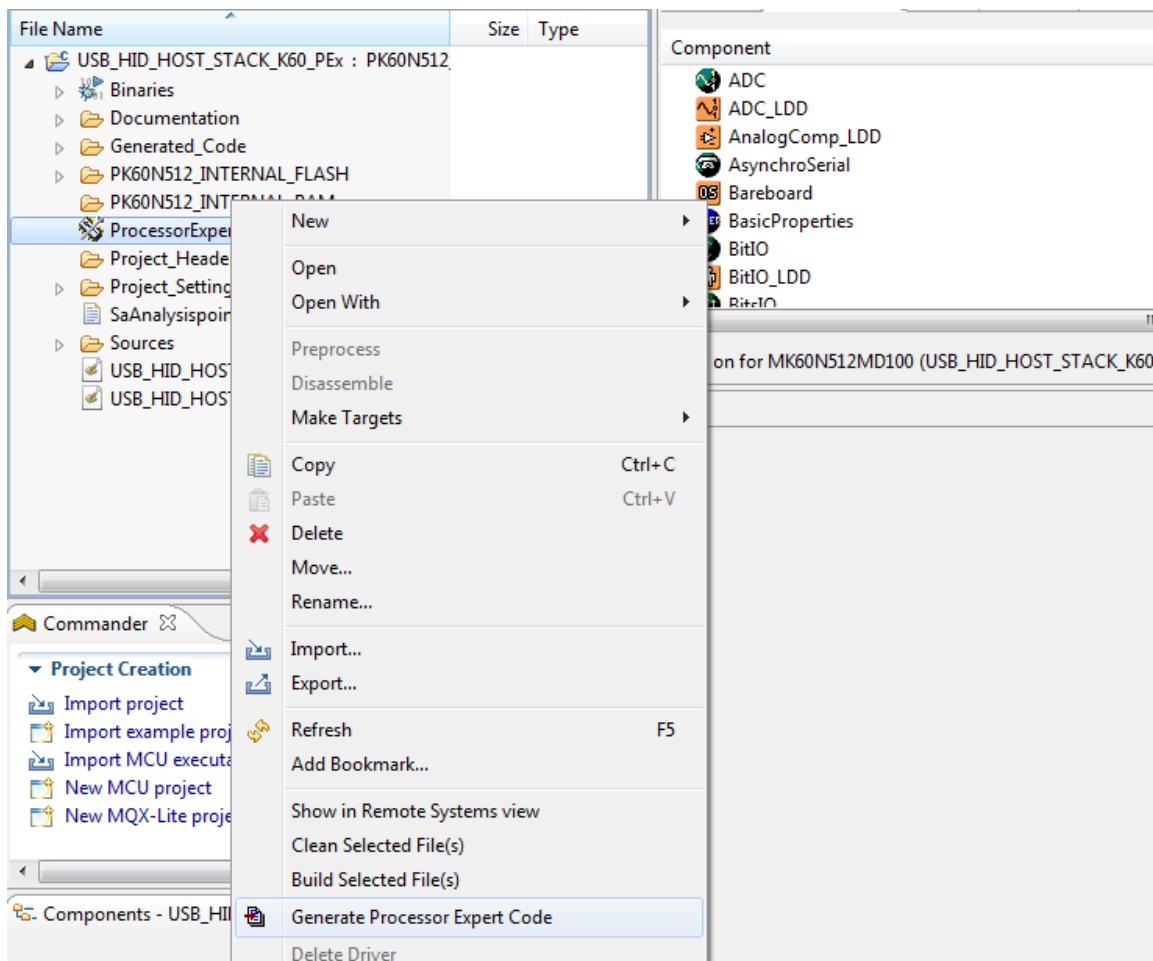


Figure 1-22 Generate Processor Expert code

**Step12.** After source code has been generated, we must add code for the transfer with the application layer.

- Add code to define `USB_CLASS_HID_MOUSE_INTF_STRUCT` in the “`usb_classes.h`” file:

```
typedef struct {
    GENERAL_CLASS                                     G;
/* End <struct_begin_1>, DO NOT MODIFY LINES ABOVE */
/* Write your own fields of struct 1 below */
/* Only 1 command can be issued at one time */
    boolean                                         IN_SETUP;

    /* Here we store callback and parameter from higher level */
    tr_callback                                      USER_CALLBACK;
    pointer                                           USER_PARAM;
/* Begin of <struct_end_1>, DO NOT MODIFY LINES BELOW */
} USB_CLASS_HID_MOUSE_INTF_STRUCT, _PTR_
USB_CLASS_HID_MOUSE_INTF_STRUCT_PTR;
```

- Add code to register “***usb\_class\_hid\_init***” function into the “***usb\_classes.c***” file:

```
void usb_class_hid_mouse_init(PIPE_BUNDLE_STRUCT_PTR pbs_ptr,
CLASS_CALL_STRUCT_PTR ccs_ptr)
{
/* End <function_begin_1>, DO NOT MODIFY LINES ABOVE */
/* Write your own code of usb_class_hid_mouse_init function below */
    usb_class_hid_init(pbs_ptr,ccs_ptr);
/* End of <function_end_1>, DO NOT MODIFY LINES BELOW */
} /* End of usb_class_hid_mouse_init function */
```

- Add code into the “***ProcessorExpert.c***” file to initialize and run the “***Mouse\_Task***” function.

```
/* User includes (#include below this line is not maintained by
Processor Expert) */
#include "hidmouse.h"
#include "usb_host_hid.h"
#include "poll.h"
uchar main_buffer[HID_BUFFER_SIZE];
HID_COMMAND          main_hid_com;
void main(void)
{
/* Write your local variable definition here */

/** Processor Expert internal initialization. DON'T REMOVE THIS
CODE!!! ***/
PE_low_level_init();
/** End of Processor Expert internal initialization. **/


/* Write your code here */
hid_mouse_init();
for(;;){
    Mouse_Task(main_buffer,&main_hid_com);
    Poll();
}
```

- Also HID Class and mouse application files should be created or added to project from the provided example.

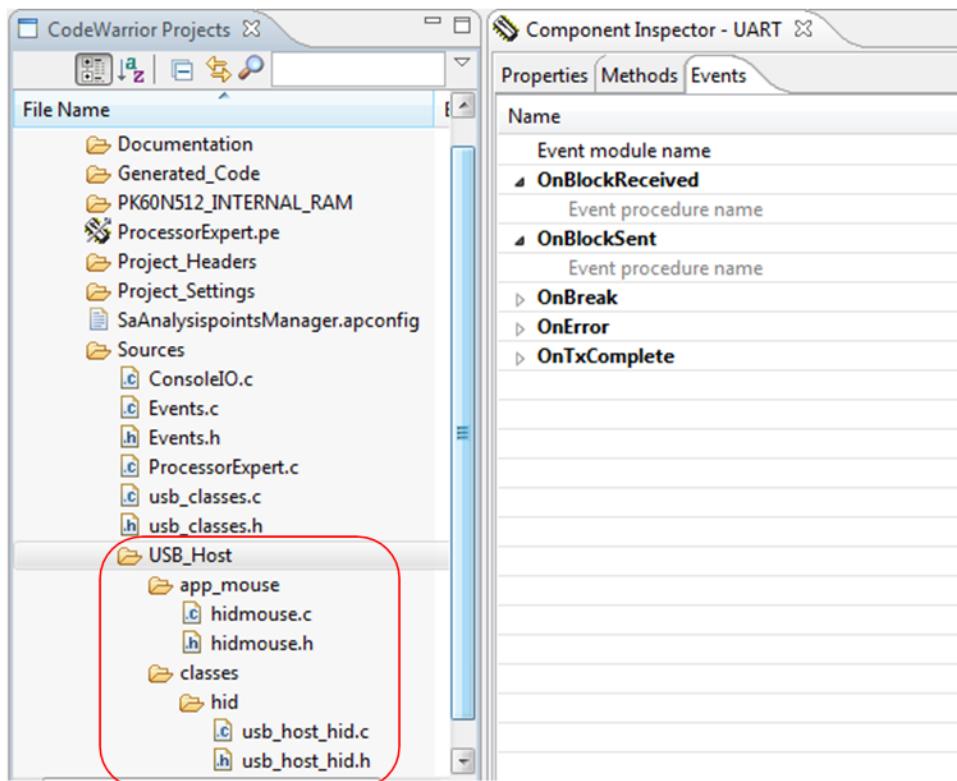


Figure 1-23 USB HID Host included files

#### 1.2.4 Running the example

Implement the following steps to run HID mouse host example:

**Step1.** Setup the hardware to run the example:

- The computer uses one USB cable to supply power to the board and program USB HID Host image to the flash. One COM cable is used to get events from the HID mouse host.
- The USB HID mouse device is plugged to the USB HID mouse host by one USB cable.

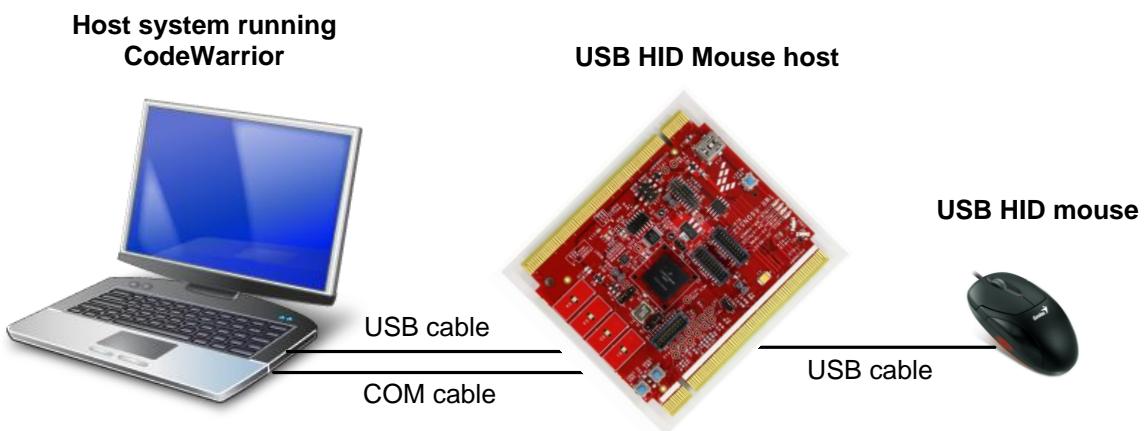


Figure 1-24 Hardware setup

**Step2.** To ensure that the application runs correctly, the HyperTerminal is used to get events from the USB mouse host device. This software is configured as shown below:

- Open HyperTerminal application:



Figure 1-25 Run HyperTerminal application

- The HyperTerminal opens as shown below. Enter the name of the connection and click the OK button.

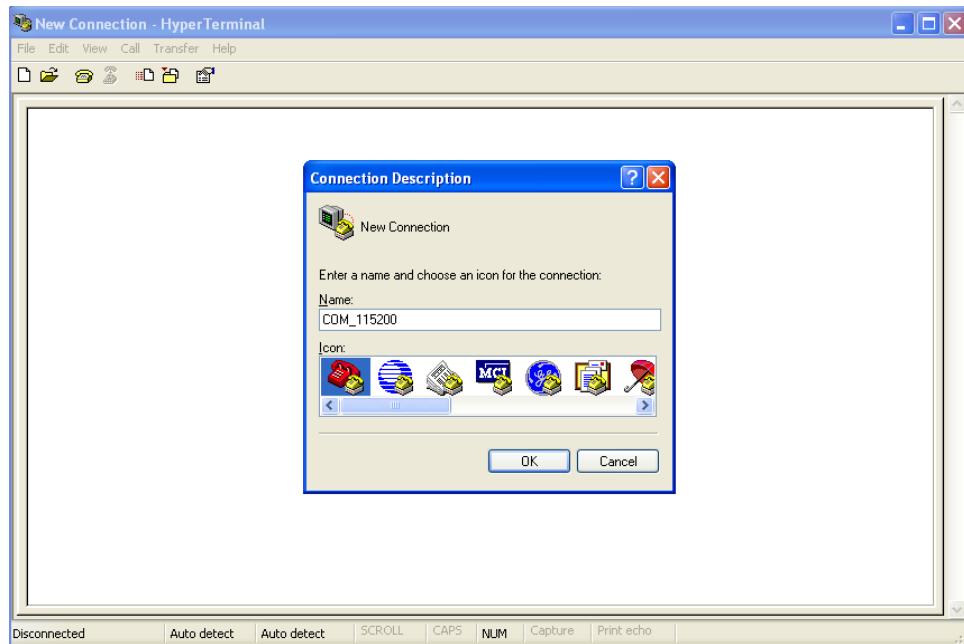


Figure 1-26 HyperTerminal startup

- The window shown in the following picture appears. Select the COM port.

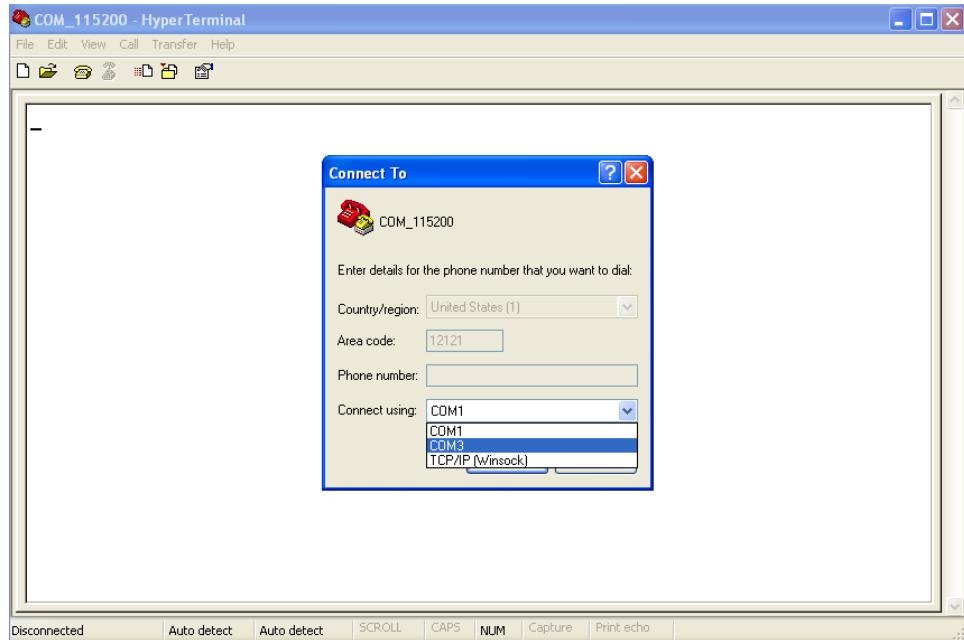


Figure 1-27 Connect using COM port

- Configure the COM port baud rate and other properties as shown below:

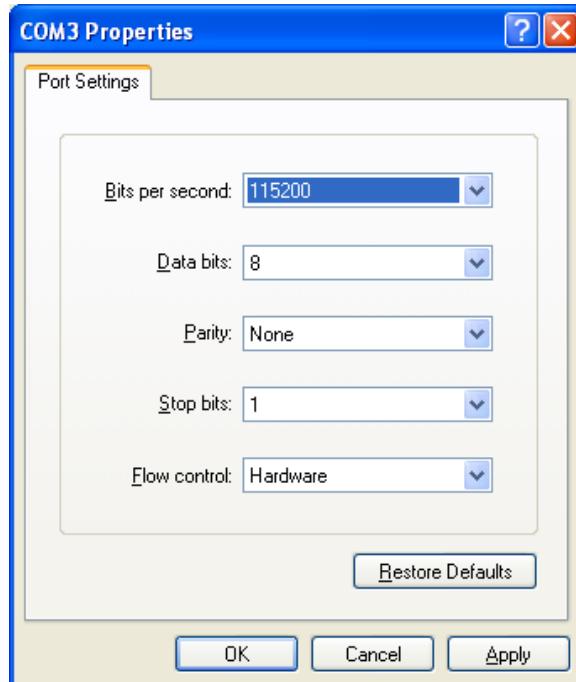


Figure 1-28 COM properties

- The HyperTerminal is now configured as shown below:

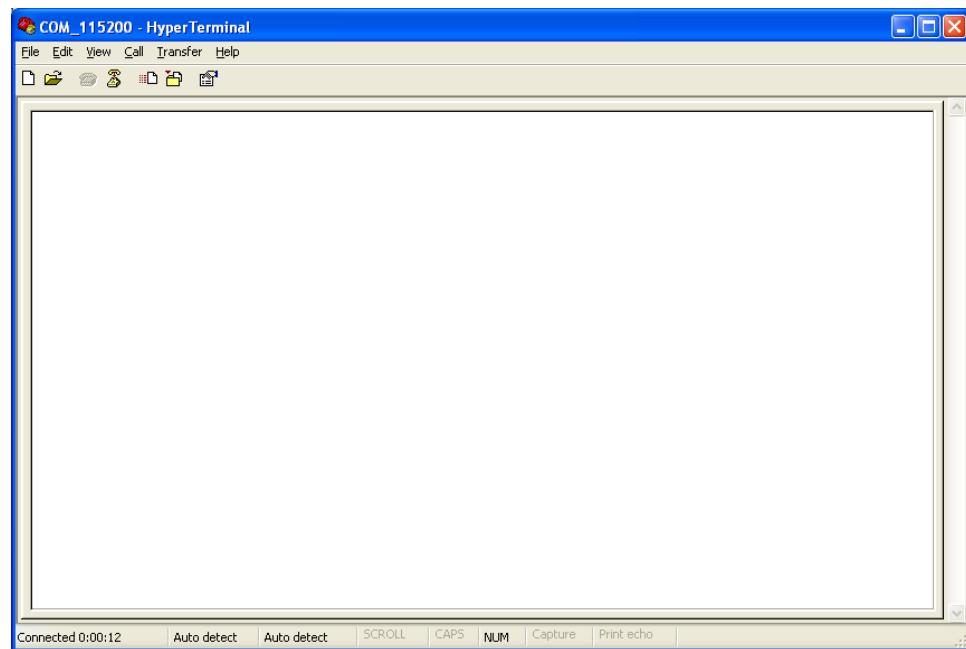


Figure 1-29 Hyper Terminal

**Step3.** Build and load the image of USB HID Host application to the Kinetis K60 Tower board. The HyperTerminal shows the USB HID Host event as shown in the following figure:

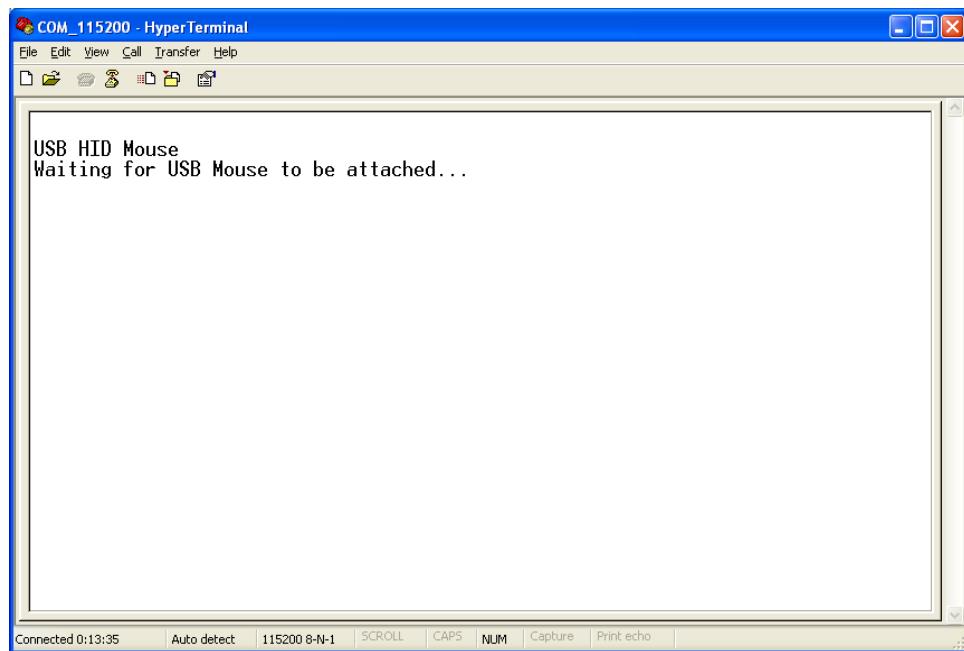


Figure 1-30 USB HID mouse host initializing

**Step4.** Plug the USB HID mouse device into the USB HID host, the HyperTerminal appears as shown in the following figure:

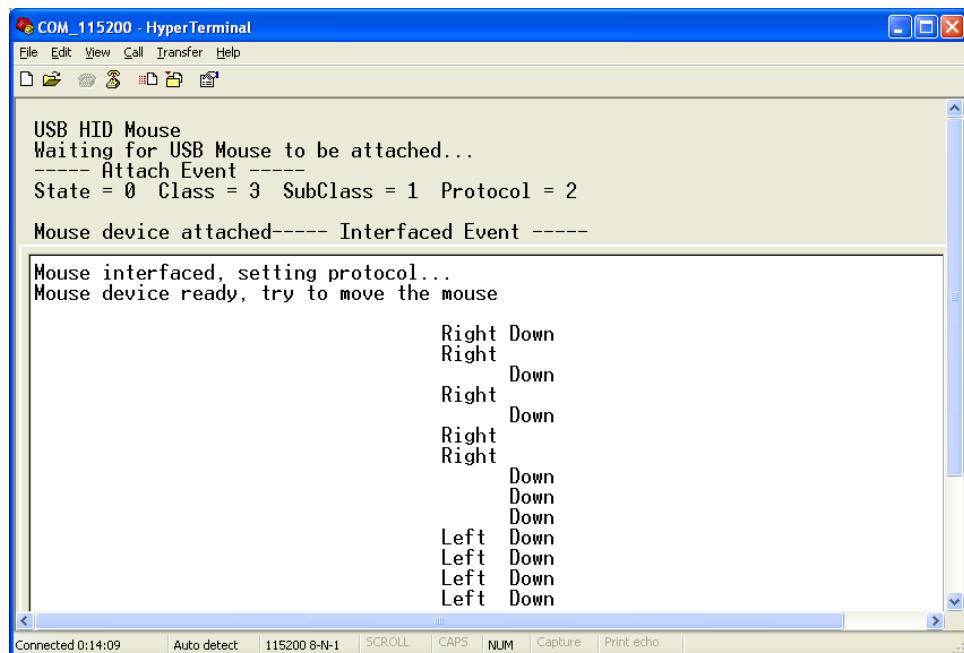


Figure 1-31 USB HID mouse host operating

## 2 USB\_OTG\_STACK

This section describes how to run the USB OTG example that uses the USB\_OTG\_STACK component in the Processor Expert of CodeWarrior.

### 2.1 Reference material

Refer also to the following materials:

- USB HOST STACK component section
- USB Specification Revision 2.0
- USB On-The-Go and Embedded Host Revision 2.0

### 2.2 Folder structure

In this section, we'll create the USB HID OTG example project to verify the operation of the USB OTG STACK component. It has the structure as shown below:

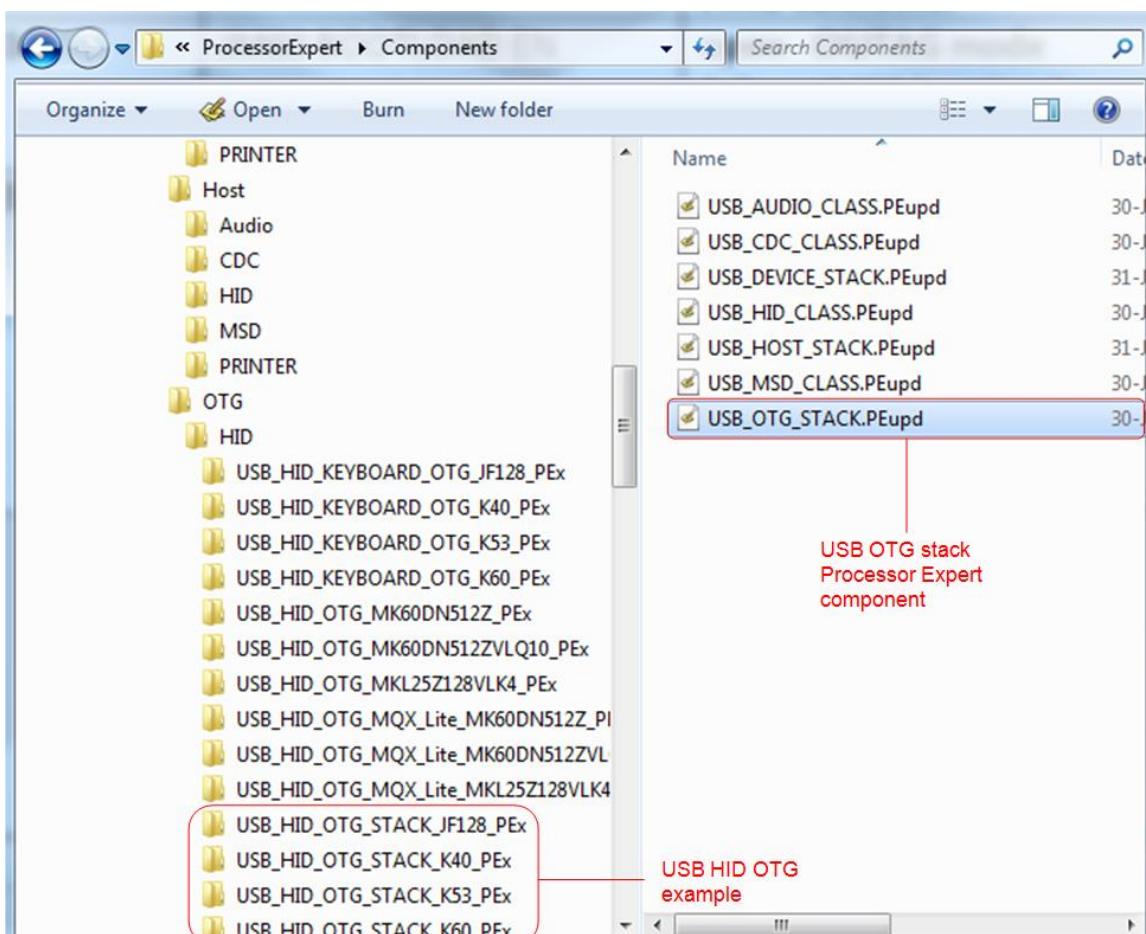


Figure 2-1 USB\_HID\_OTG folder structure

## 2.3 Running USB\_HID\_OTG example

### 2.3.1 Generating Processor Expert source code

This section describes the steps to generate Processor Expert source code that is used to run the USB HID OTG example.

**Step1.** Open USB\_HID\_OTG\_STACK\_K60\_PEx example with CW10.3 → From the CW10.3 toolbar, select File -> Import to open the import window -> Open General folder -> Select Existing Projects into Workspace -> Click Next.

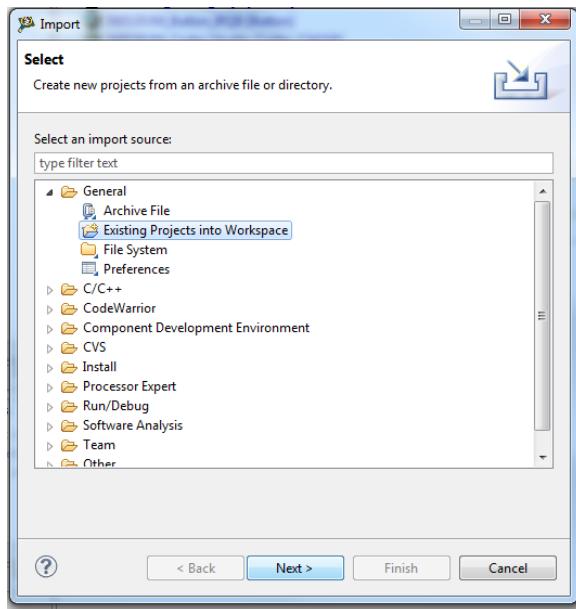


Figure 2-2 Import existing projects into Workspace

**Step2.** Click Browse button → Select USB\_HID\_OTG\_STACK\_K60\_PEx folder -> Click OK -> then click Finish button.

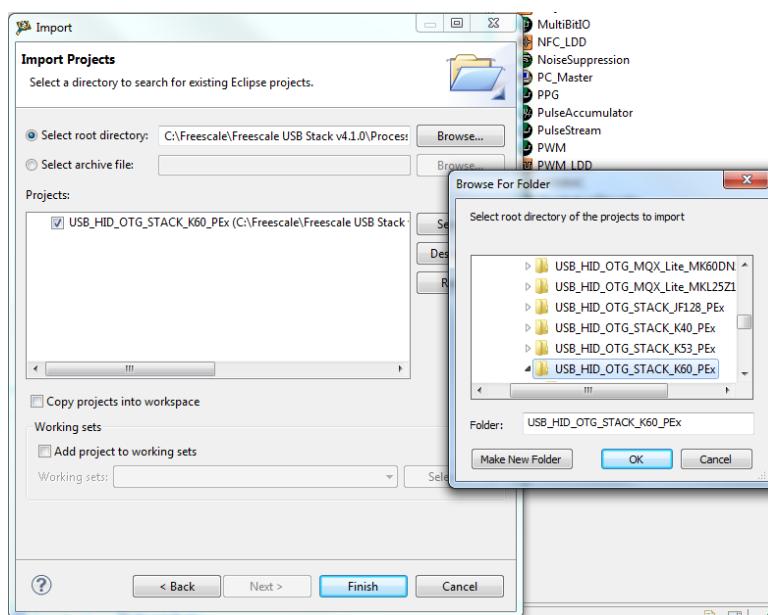


Figure 2-3 Import existing projects into Workspace

**Step3.** Generate Processor Expert code:

Open the project folder and select ProcessorExpert.pe -> Right click then select Generate Processor Expert Code field to generate source code.

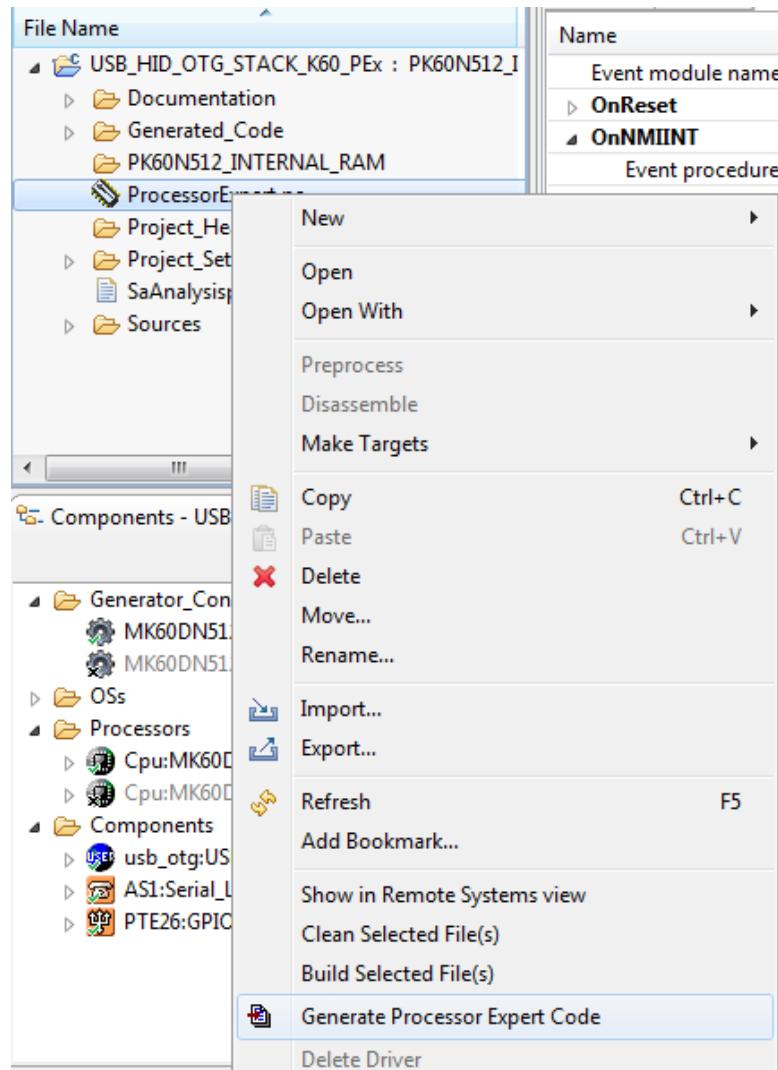


Figure 2-4 Generate Processor Expert code

### 2.3.2 Creating USB\_HID\_STACK project for K60 using Processor Expert

**Step1.** Open CodeWarrior Development Studio 10.3 -> from the toolbar, select File -> New -> Bareboard Project. In the Project name text box type the desired name for the project -> Next.

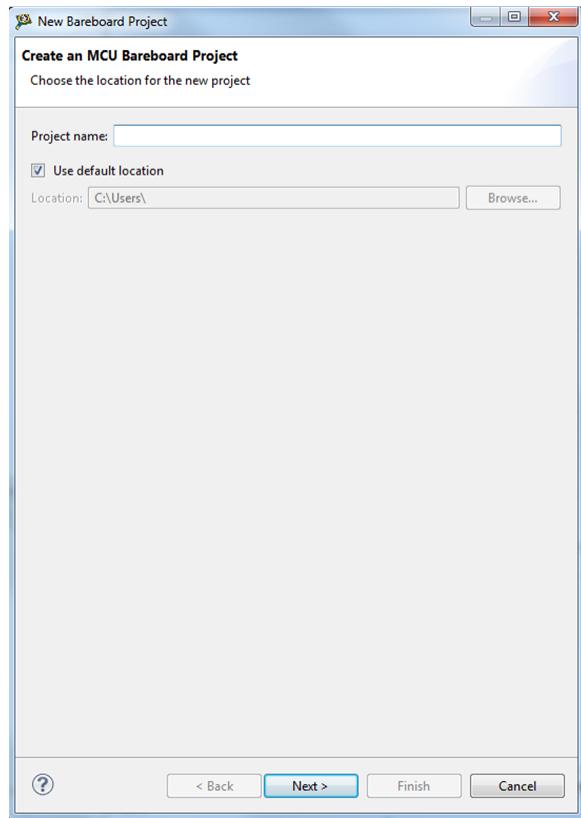


Figure 2-5 New Bareboard Project

**Step2.** Select the device MK60DN512 -> Next.

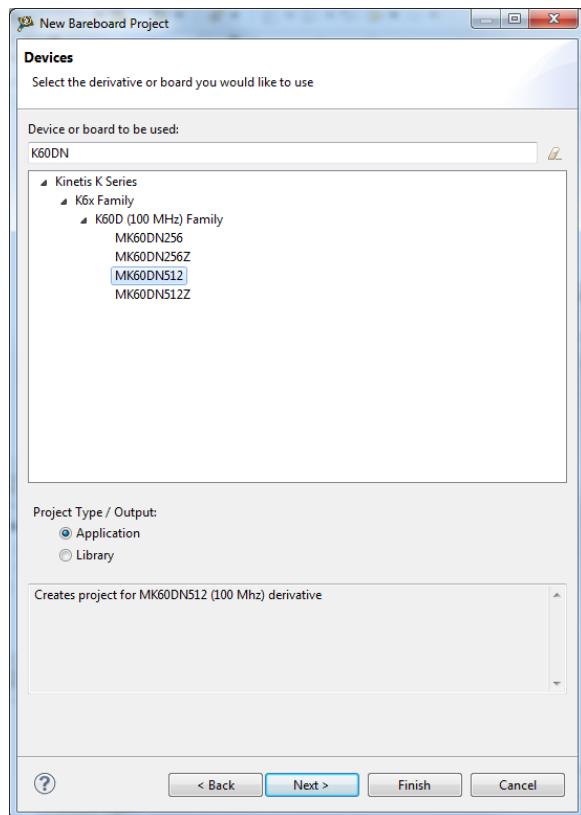


Figure 2-6 Devices window.

**Step3.** Select the connection used for this project -> Next, Language and Build Tools Options windows -> Next.

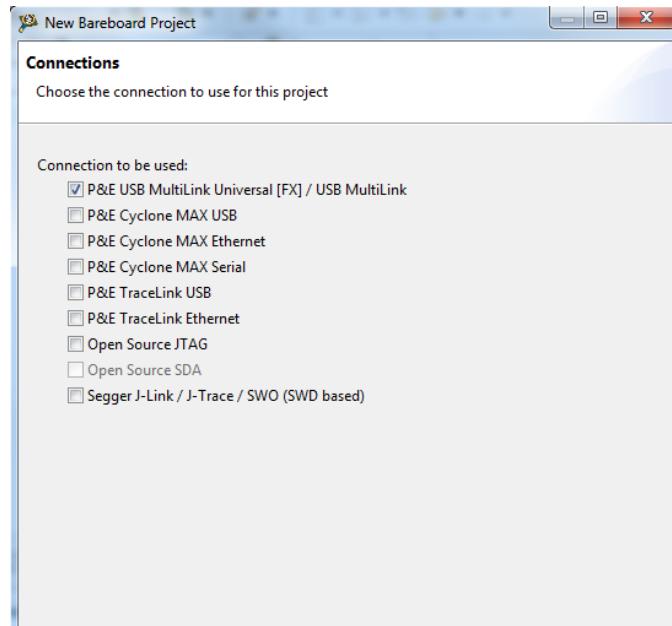


Figure 2-7 Connections window.

**Step4.** In Rapid Application Development window select Processor Expert and then Finish.

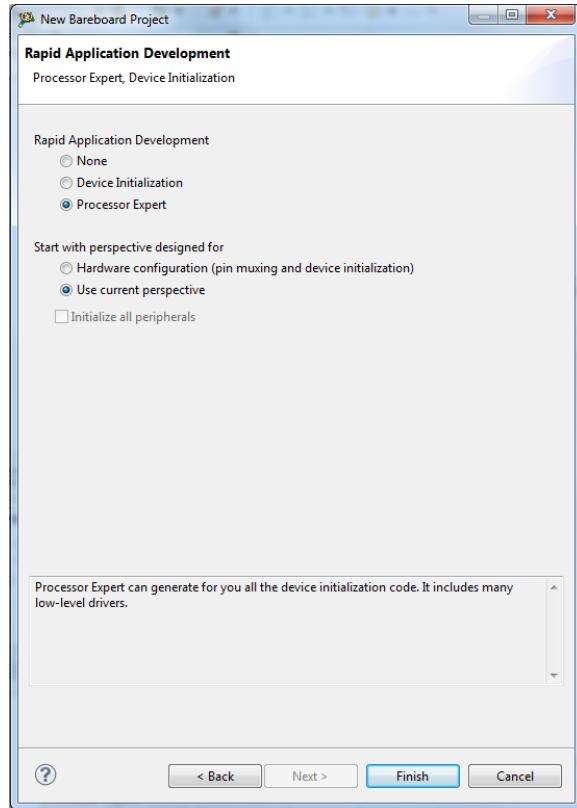


Figure 2-8 Rapid Application Development window.

**Step5.** From the CW10.3 toolbar, select Processor Expert → Select Show Views to open Components Library → Choose and add the USB\_OTG\_STACK component to the project.

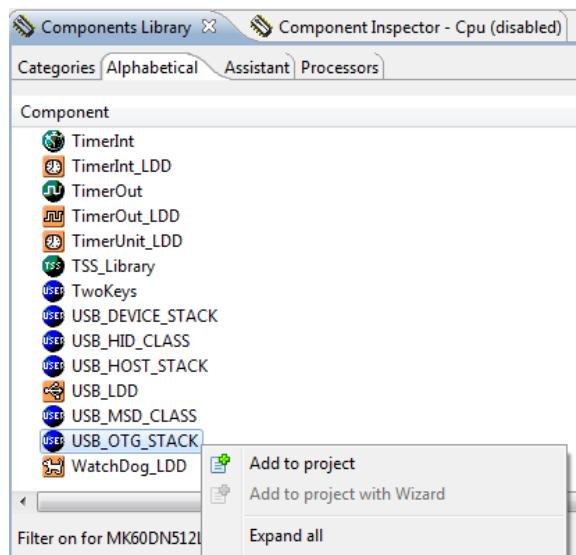


Figure 2-9 Components Library

The USB\_OTG\_STACK component has the user interface as shown below:

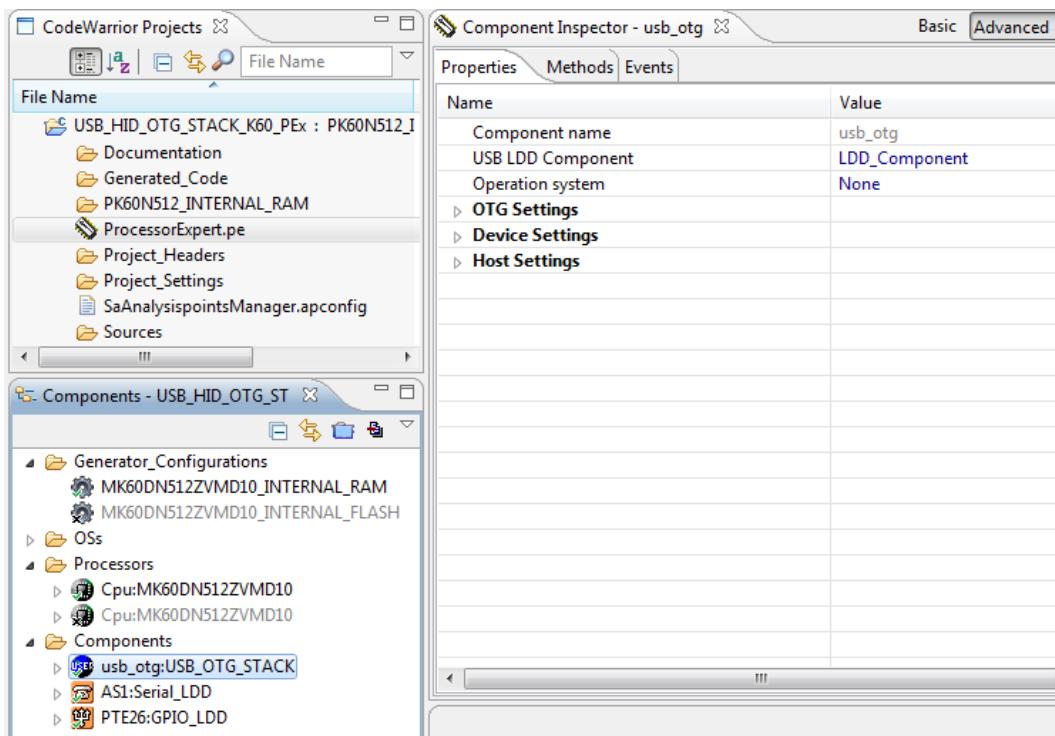


Figure 2-10 USB\_OTG\_STACK component user interface

**Step6.** Implement same as step 5 to add Serial\_LDD and GPIO\_LDD components into the project.

## Step7. Configure CPU clock to run USB module:

- Select CPU target from Project Panel window to setup clock source as the following:
  - o Target: Cpu:MK60N512VMD100
  - o Clock source: External reference clock 50MHz
  - o MCG mode: PEE
  - o MCG output: 48 MHz
  - o PLL output: 48MHz
  - o Core clock: 48MHz
  - o Bus clock: 48MHz
  - o External bus clock: 24MHz
  - o Flash clock: 24MHz

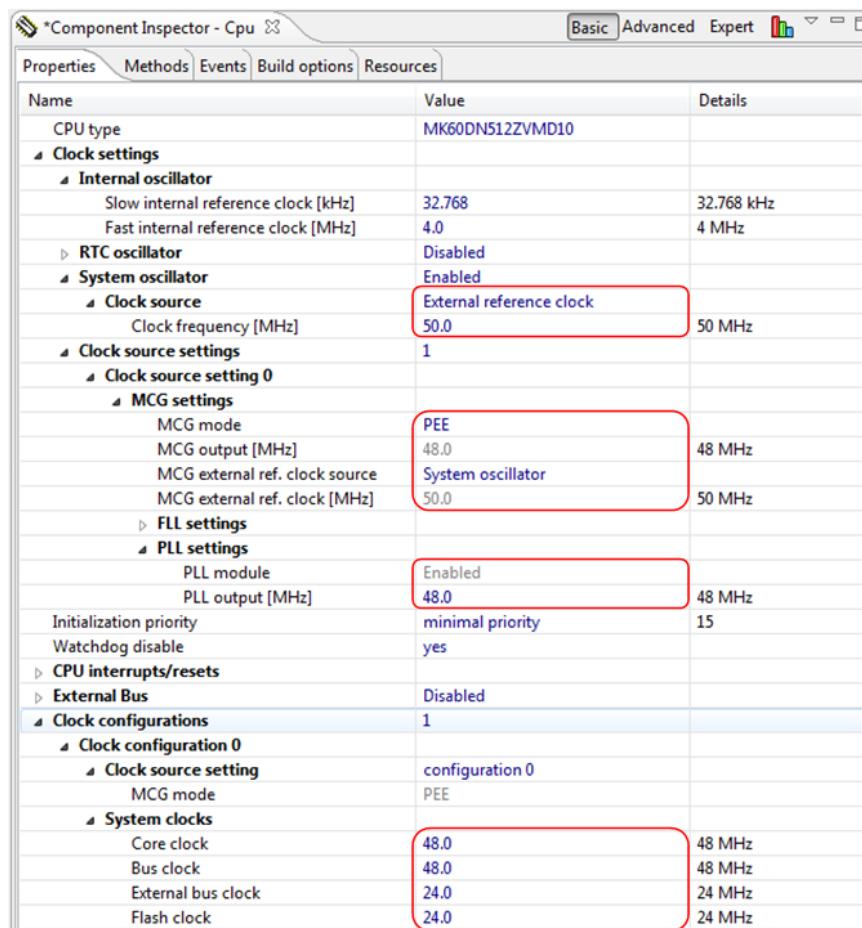


Figure 2-11 USB Clock source settings

- Chose advanced tab:

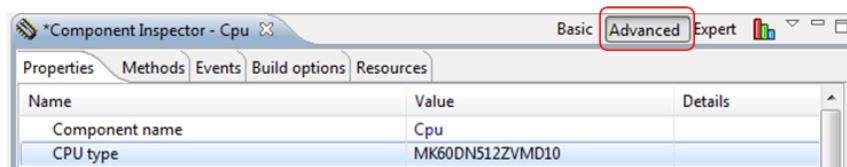


Figure 2-12 Open Advanced Tab

- PLL/FLL clock selection:
  - o PLL clock

The screenshot shows a table from a software interface. The columns are 'Name', 'Value', and 'Details'. The rows include:

Bus clock	48.0	48 MHz
External clock prescaler	Auto select	2
External bus clock	24.0	24 MHz
Flash clock prescaler	Auto select	2
Flash clock	24.0	24 MHz
<b>PLL/FLL clock selection</b>	<b>PLL clock</b>	
Clock frequency [MHz]	48.0	48 MHz
<b>USB clock settings</b>		
USB clock divider	Auto select	1
USB clock multiply	Auto select	1
USB clock	48.0	48 MHz

Figure 2-13 USB Clock configuration

- Choose Internal Peripherals field → Enable Flexible memory controller → choose Master 4(USB OTG) → Choose Access protection - Read-Write.

The screenshot shows the 'Component Inspector - Cpu' window with the 'Advanced' tab selected. The 'Properties' tab is active. The table shows the following settings for Master 4 (USB OTG):

Name	Value	Details
Initialization priority	minimal priority	15
Watchdog disable	yes	
<b>Internal peripherals</b>		
<b>NMI pin</b>	Enabled	
NMI Pin	PTA4/FTM0_CH1/NMI_b/EZP_CS_...	PTA4/FTM0_CH1/NM
NMI Pin signal		
<b>Reset</b>		
<b>Debug interface (JTAG)</b>		
<b>Flash memory organization</b>		
<b>Flexible memory controller</b>	Enabled	
<b>Access Protection</b>		
<b>Master 0 (ARM core code bus)</b>		
<b>Master 1 (ARMcore system bus)</b>		
<b>Master 2 (DMA / EzPort)</b>		
<b>Master 3 (Ethernet)</b>		
<b>Master 4 (USB OTG)</b>		
Prefetch	Disabled	
Access protection	Read-Write	
<b>Master 5 (SDHC)</b>		
<b>Cache settings</b>		

Figure 2-14 Flexible memory controller settings

- Enable AIPS1 setting → Choose USB OTG → Configure as shown below:

Component Inspector - Cpu			Basic	Advanced	Expert
Properties		Methods	Events	Build options	Resources
Name	Value	Details			
Internal peripherals					
NMI pin	Enabled				
Reset					
Debug interface (JTAG)					
Flash memory organization					
Flexible memory controller	Enabled				
Flash configuration field	Disabled				
MPU settings	Enabled				
AXBS settings	Enabled				
AIPSO settings	Enabled				
AIPS1 settings	Enabled				
Master privilege settings					
ARM core code bus					
ARMcore system bus					
DMA / EzPort					
Ethernet					
USB OTG					
Trusted for read	yes				
Trusted for write	yes				
Privilege level	no				

Figure 2-15 AIPS1 settings

- Enable LVD interrupt in the Power management controller field.

Component Inspector - Cpu			Basic	Advanced	Expert
Properties		Methods	Events	Build options	Resources
Name	Value	Details			
Flexible memory controller	Enabled				
Flash configuration field	Disabled				
MPU settings	Enabled				
AXBS settings	Enabled				
AIPSO settings	Enabled				
AIPS1 settings	Enabled				
MCM settings	Disabled				
System control block settings	Disabled				
Power management controller					
LVD reset	Enabled				
LVD voltage threshold	Low				
LVW voltage threshold	Low				
Traditional RAM power	Not powered in VLLS2				
Bandgap buffer	Disabled				
LVD interrupt					
Interrupt	INT_LVD_LVW				
Interrupt request	Enabled				
Interrupt priority	0 (Highest)				
LVD interrupt	Disabled				
LVW interrupt	Disabled				
System Integration Module					

Figure 2-16 Power management controller settings

**Step8.** Configure the USB\_OTG\_STACK component to run USB\_HID\_OTG example.

Select the USB\_OTG\_STACK component to configure it as in the following figure:

- OTG settings:
  - o Choose GPIO LDD component to configure interrupt pin for MAX3353 as in the following figure:

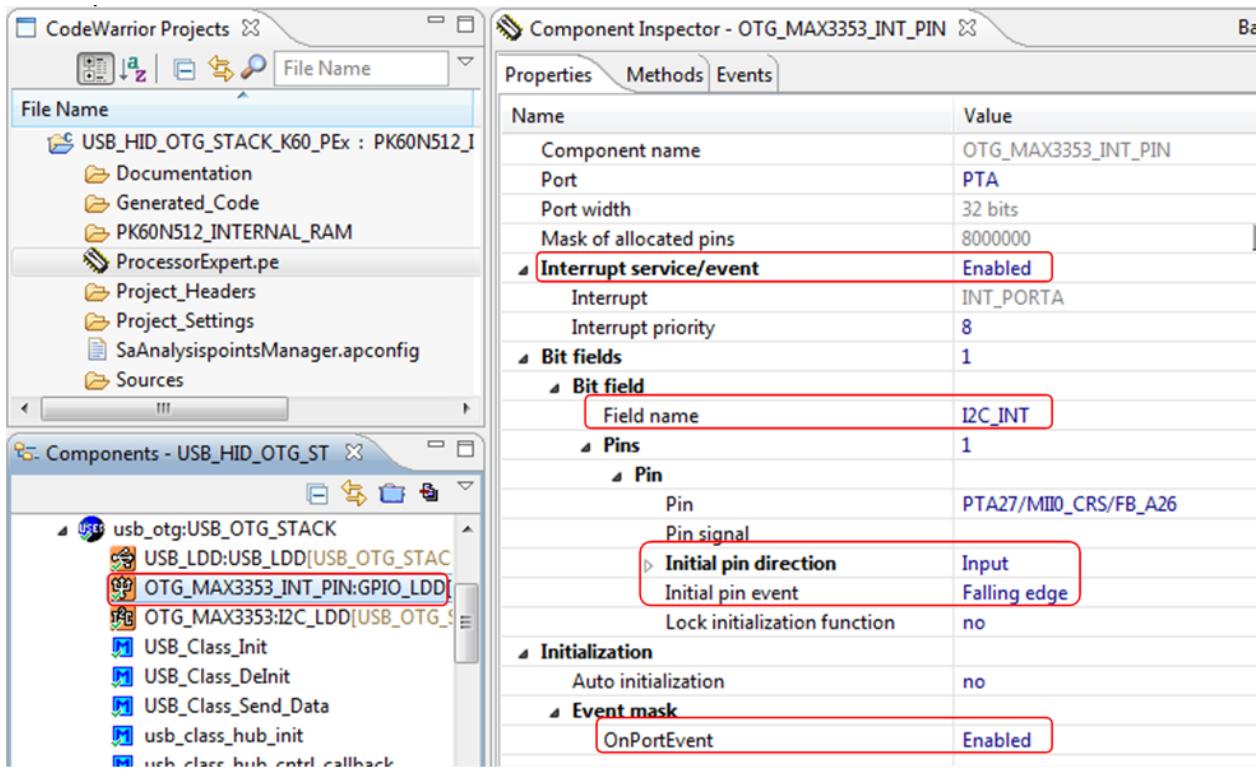


Figure 2-17 MAX3353 Interrupt pin settings

- Choose I2C LDD component to configure the features for MAX3353

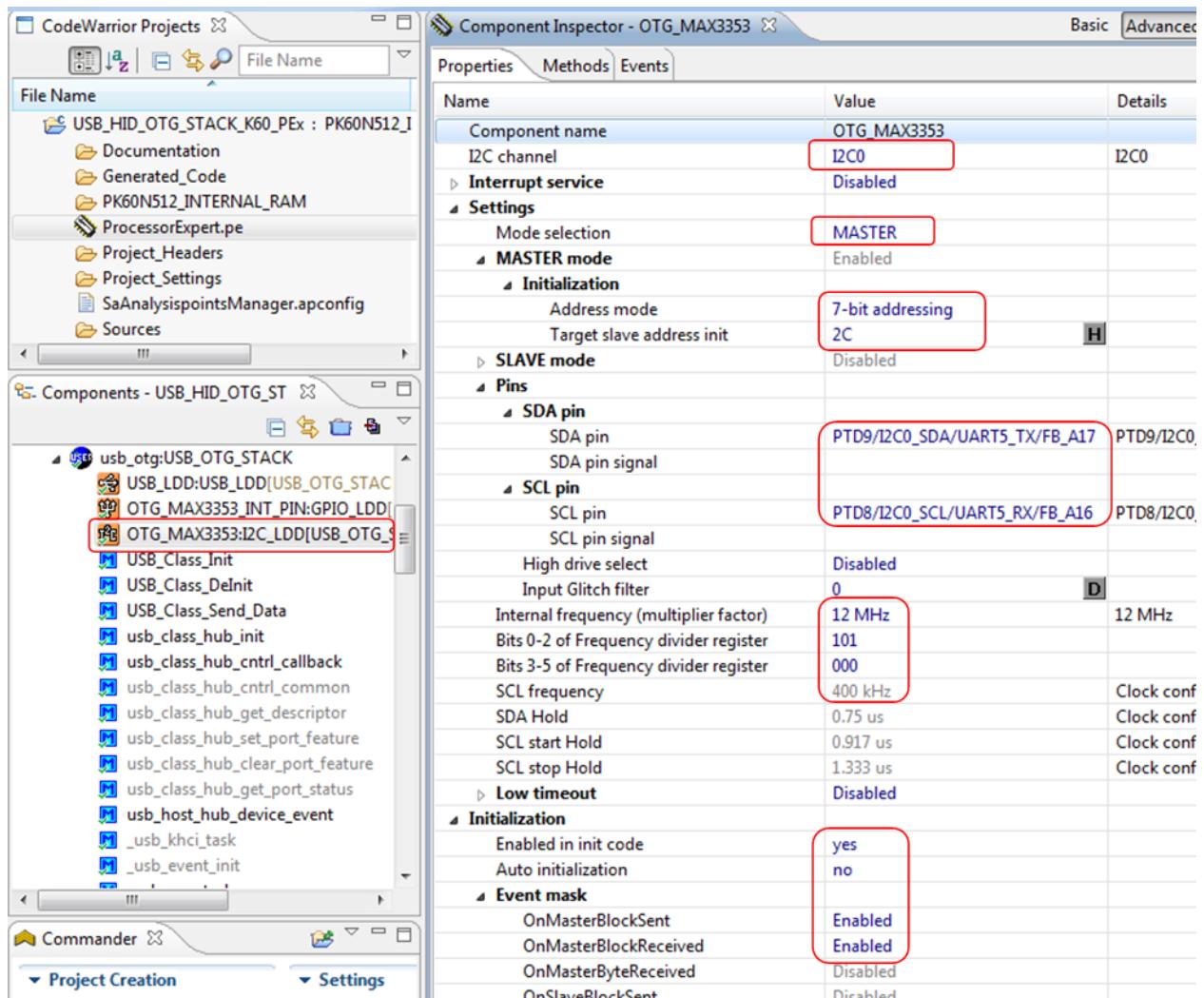


Figure 2-18 MAX3353 settings

- Device settings:
  - o Configure general descriptors as in the following figure:

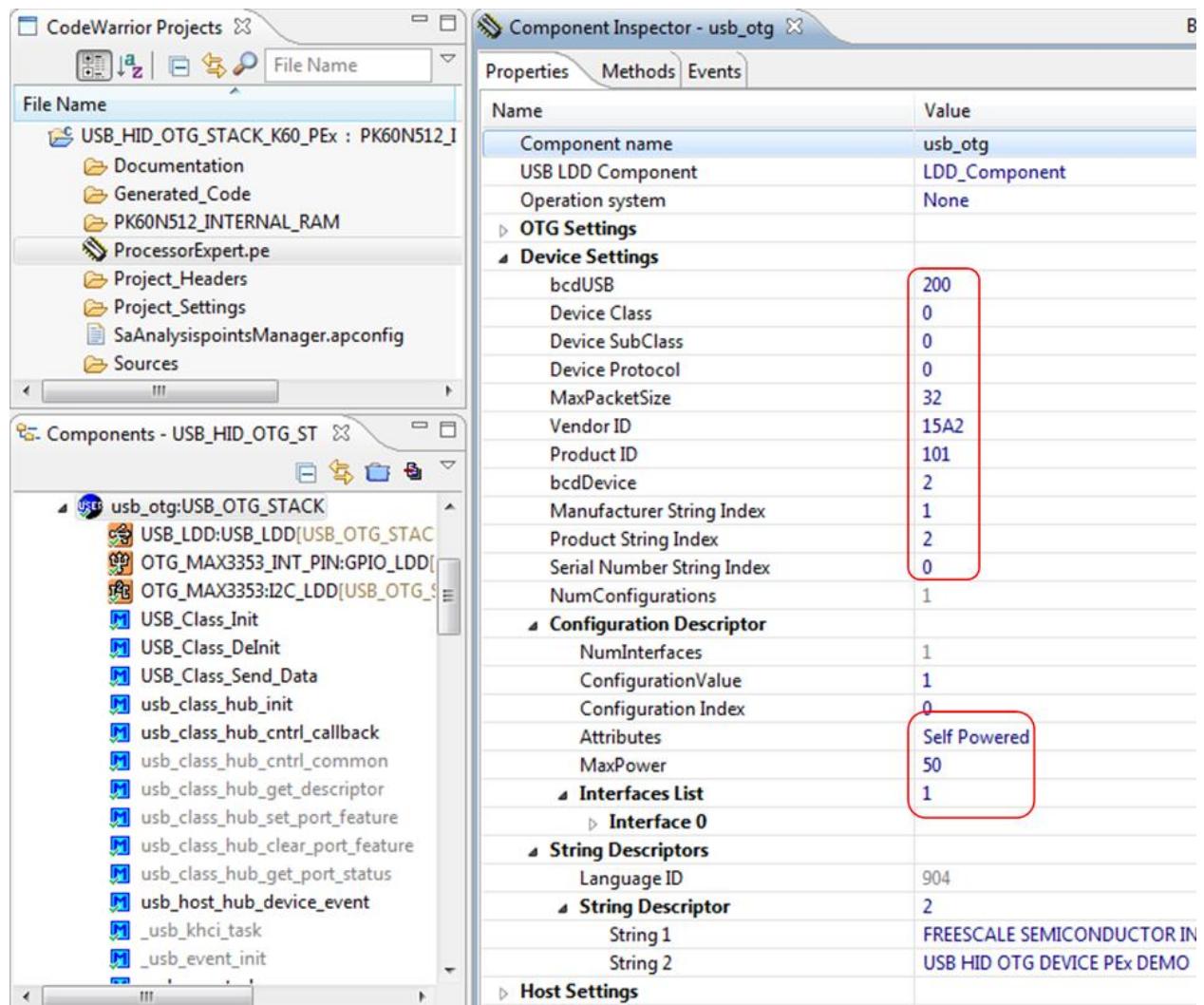


Figure 2-19 General descriptors settings

- Configure HID descriptor as in the following figure:

Interfaces List	
InterfaceNumber	0
AlternateSetting	0
NumEndpoints	1
InterfaceClass	3
InterfaceSubClass	1
InterfaceProtocol	1
Interface String Descriptor Index	0
Endpoints List	
EndpointNumber	1
EndpointDirection	IN
Transfer Type	Interrupt
Max packet size	8
Interval	10

String Descriptors	
Language ID	904
String Descriptor	
String 1	2
String 2	FREESCALE SEMICONDUCTOR INC USB HID OTG DEVICE PEx DEMO

Figure 2-20 HID descriptor settings

- Host settings:
  - Setup the Host settings field as in the following figure:

Name	Value
Component name	usb_otg
USB LDD Component	LDD_Component
Operation system	None
OTG Settings	
Device Settings	
Host Settings	
Common Class	yes
Use Poll	yes
Use Hub	yes
Class List	
Class 0	
Class Name	hid_keyboard
Class Code	3
Subclass Code	0
Protocol Code	0
Class Code Mask	FF
SubClass Code Mask	0
Protocol Code Mask	0

Figure 2-21 Host settings

**Step9.** Configure the USB\_LDD component as shown below:

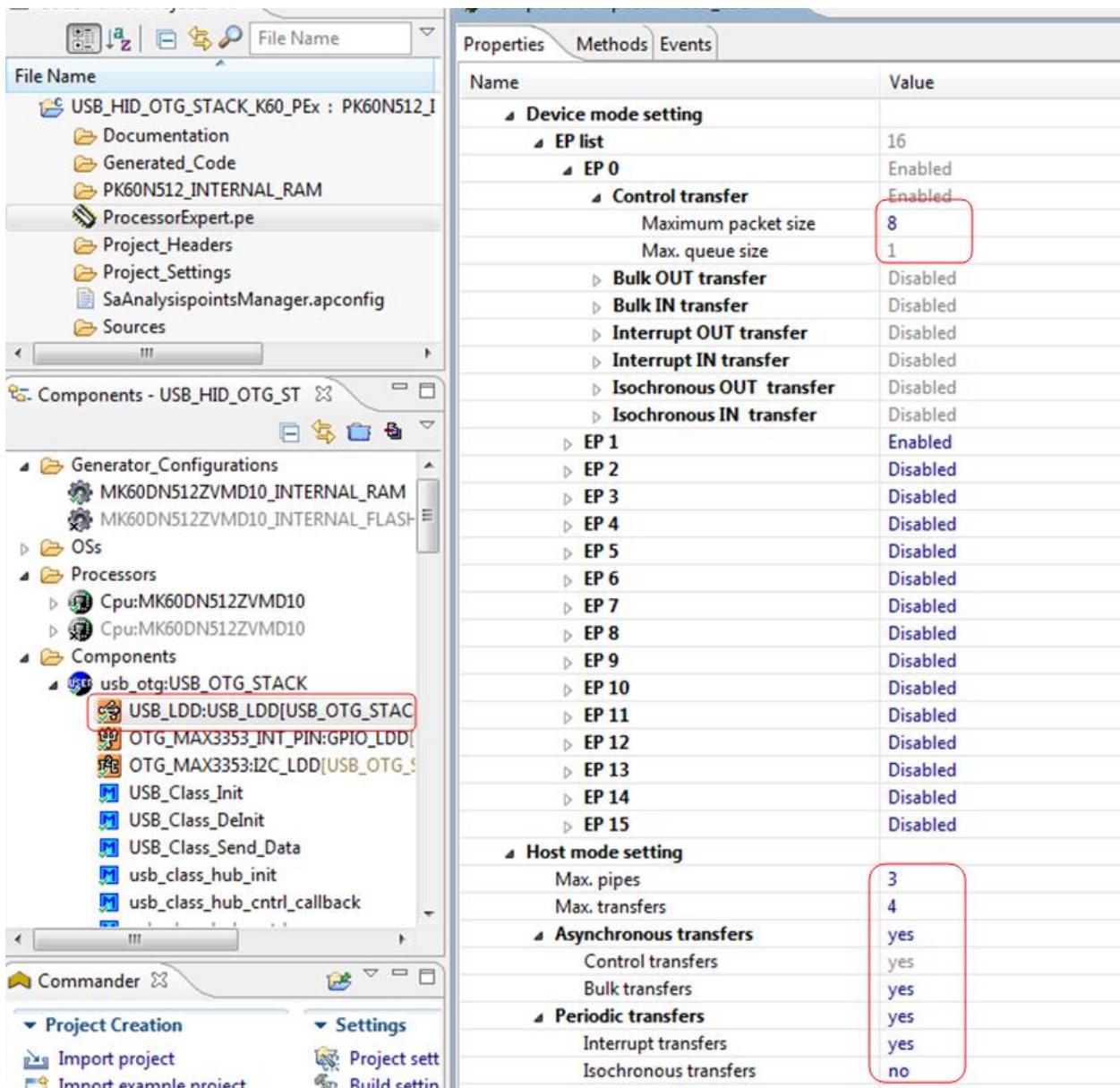


Figure 2-22 USB\_LDD component settings

**Step10.** To run HID OTG component we have to add two components:

- Serial LDD component is used to show the status and operation of the device.
- GPIO LDD component is used as a demo for pressing the button.

Chose components from component library and add them to the project as the step 5 → configure the components as in the following figure:

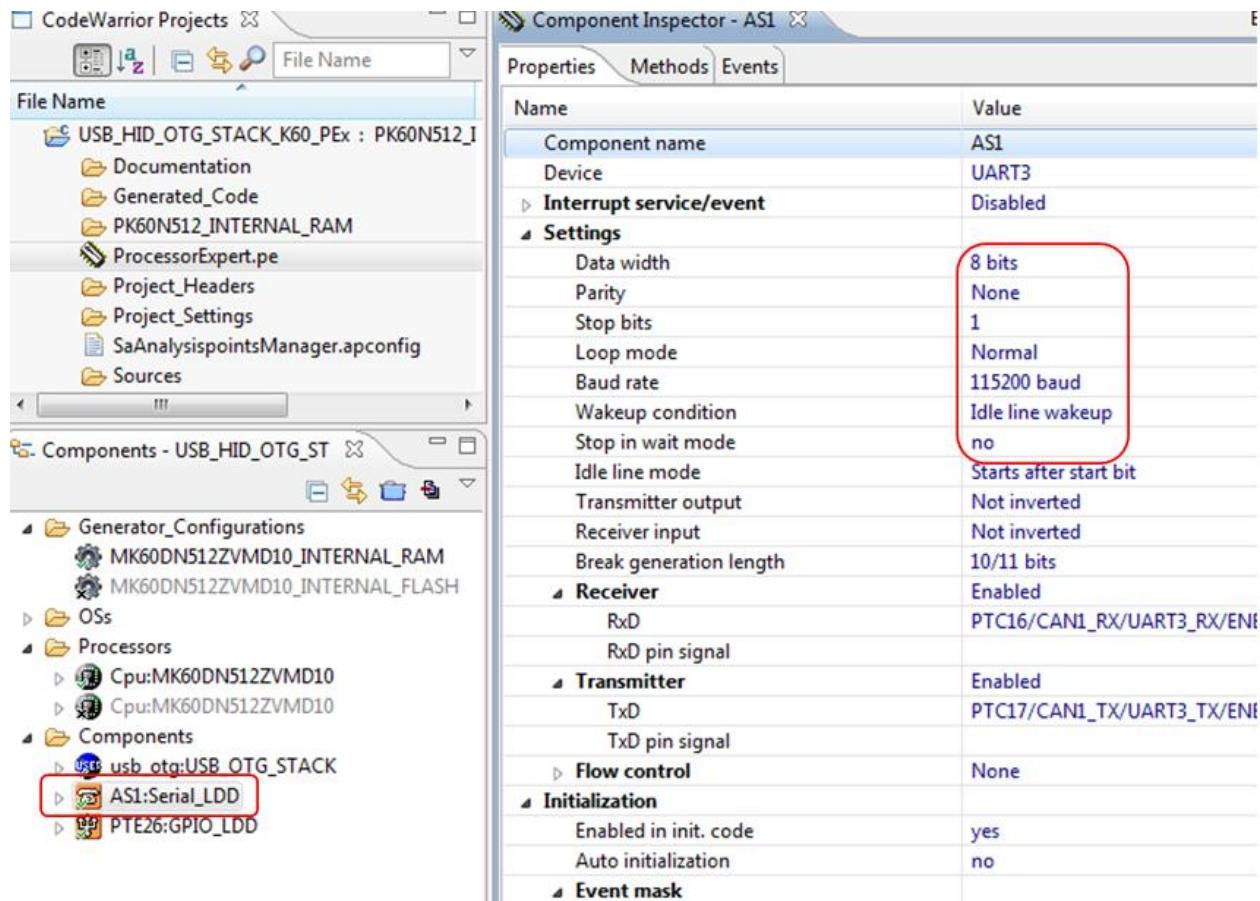


Figure 2-23 Serial\_LDD component settings

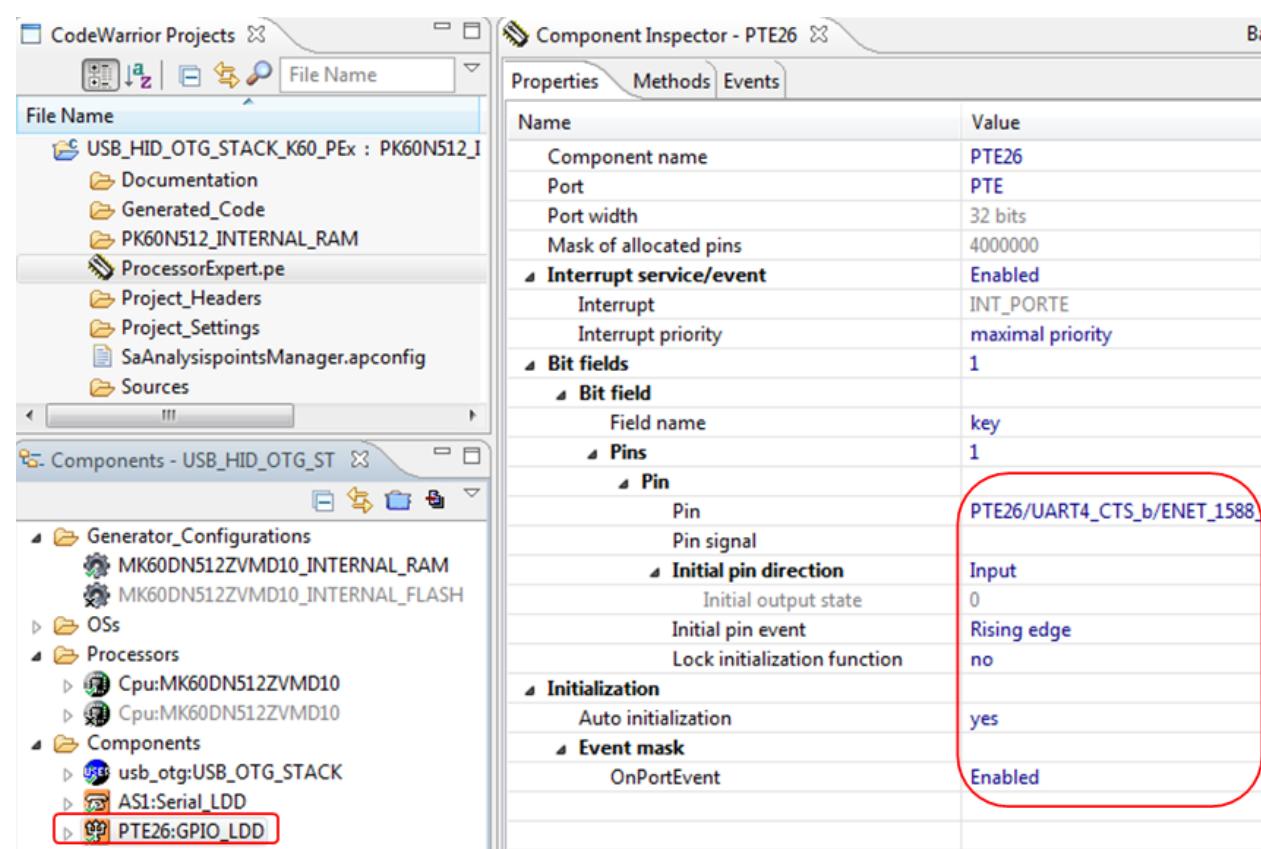


Figure 2-24 GPIO\_LDD component settings

### Step11. Generate Processor Expert code:

After setting the components, select ProcessorExpert.pe in the project folder → Right click → Choose Generate Processor Expert Code field to generate the source code.

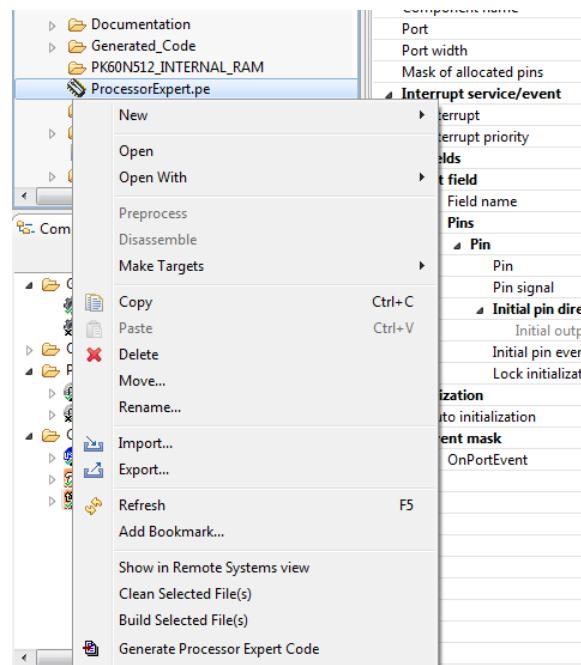


Figure 2-25 Generate Processor Expert code

**Step12.** After source code has been generated, we must add code for the transfer with the application layer.

- HID Device application: The USB\_OTG\_STACK component generates only a general descriptor for user. To run USB HID device, user must add HID specific descriptor as the following:

Add “**usb\_descriptor.c**” file:

- Write specific descriptor into the “**g\_config\_descriptor**” array:

```
uint_8 g_config_descriptor[CONFIG_DESC_SIZE] =
{
    /* some generated code */

    /* End code block <1> auto generated code, DO NOT MODIFY LINES ABOVE */
    /* Specific Interface Descriptor */
    /* Write code to define specific descriptor here */
    HID_ONLY_DESC_SIZE,
    USB_HID_DESCRIPTOR,
    0x00,0x01,
    0x00,
    0x01,
    0x22,
    0x3F,0x00, /* report descriptor size to follow */
    /* Begin of code block <2> auto generated code, DO NOT MODIFY LINES
    BELOW */ */

    /* some generated code */

};
```

- Write specific descriptor size into the “**g\_std\_desc\_size**” array:

```
USB_PACKET_SIZE const g_std_desc_size[USB_MAX_STD_DESCRIPTOR+1] = {
    0,
    DEVICE_DESCRIPTOR_SIZE,
    CONFIG_DESC_SIZE,
    0,                                /* string */
    0,                                /* Interface */
    0,                                /* Endpoint */
    0,                                /* Device Qualifier */
    0                                 /* other speed config */

    /* End code block <3> auto generated code, DO NOT MODIFY LINES ABOVE */
    /* write the specific descriptor size here*/
    ,REPORT_DESC_SIZE
    /* Begin of code block <4> auto generated code, DO NOT MODIFY LINES
    BELOW */
};
```

- Write specific descriptor structure pointer into the “**g\_std\_descriptors**” array:

```
uint_8_ptr const g_std_descriptors[USB_MAX_STD_DESCRIPTOR+1] ={ 
    NULL,
    (uint_8_ptr)g_device_descriptor,
    (uint_8_ptr)g_config_descriptor,
    NULL,                                /* string */
    NULL,                                /* Interface */
```

```

        NULL,                                /* Endpoint */
        NULL,                                /* Device Qualifier */
        NULL                                 /* other speed config*/
/* End code block <4> auto generated code, DO NOT MODIFY LINES ABOVE */
/* write the specific descriptor pointer here*/
, (uint_8_ptr)g_report_descriptor
/* Begin of code block <5> auto generated code, DO NOT MODIFY LINES
BELOW */
};

```

- Add code for the “**USB\_Desc\_Get\_Descriptor**” function:

```

uint_8 USB_Desc_Get_Descriptor(uint_8 controller_ID, uint_8 type, uint_8
str_num, uint_16 index, uint_8_ptr *descriptor, USB_PACKET_SIZE *size)

{
    UNUSED (controller_ID)
    switch(type)
    {
/* End code block <5> auto generated code, DO NOT MODIFY LINES ABOVE */

        /* Write your own code here ... */
        /* ex
           case aaa:
           {
               // some code
           }
        */
    case USB_REPORT_DESCRIPTOR:
    {
        type = USB_MAX_STD_DESCRIPTORS;
        *descriptor = (uint_8_ptr)g_std_descriptors [type];
        *size = g_std_desc_size[type];
    }
    break;
    case USB_HID_DESCRIPTOR:
    {
        type = USB_CONFIG_DESCRIPTOR ;
        *descriptor = (uint_8_ptr)(g_std_descriptors [type]+
CONFIG_ONLY_DESC_SIZE+IFACE_ONLY_DESC_SIZE);
        *size = HID_ONLY_DESC_SIZE;
    }
    break;
    case USB_OTG_DESCRIPTOR:
    {
        type = USB_CONFIG_DESCRIPTOR ;
        *descriptor = (uint_8_ptr)(g_std_descriptors [type]+
CONFIG_ONLY_DESC_SIZE+
IFACE_ONLY_DESC_SIZE+
HID_ONLY_DESC_SIZE+
ENDP_ONLY_DESC_SIZE
);
        *size = OTG_ONLY_DESC_SIZE;
    }
    break;
}

```

```

/* Begin of code block <6> auto generated code, DO NOT MODIFY LINES BELOW */
*/
/* some generated code */

}
/* End code block <6> auto generated code, DO NOT MODIFY LINES ABOVE */

```

- “***usb\_descriptor.h*** file:

- o Add specific definitions:

```

/* End code block <1> auto generated code, DO NOT MODIFY LINES
ABOVE */

/* Write code here ... */
/* Add your custom macro here*/
#define HID_ONLY_DESC_SIZE (9)
#define REPORT_DESC_SIZE (63)
#define HID_DESC_ENDPOINT_COUNT (1)
#define HID_ENDPOINT (1)
#define HID_ENDPOINT_PACKET_SIZE (8)
#define USB_HID_DESCRIPTOR (0x21)
#define USB_REPORT_DESCRIPTOR (0x22)
/* The following macro is used for specific descriptor and default
0. If need,
you can redefine this value belongs to the application*/
#define SPECIFIC_DESC_SIZE HID_ONLY_DESC_SIZE
/* Begin of code block <2> auto generated code, DO NOT MODIFY LINES
BELOW */

```

- HID Host application: The USB\_OTG\_STACK component generates source code for common host class. To run USB HID Host, user must add USB host specific class definition.

- o Write code to define “***USB\_CLASS\_HID\_KEYBOARD\_INTF\_STRUCT***” in the “***usb\_classes.h*** file:

```

typedef struct {
    GENERAL_CLASS G;
/* End <struct_begin_1>, DO NOT MODIFY LINES ABOVE */
/* Write your own fields of struct 1 below */
    /* Only 1 command can be issued at one time */
    boolean IN_SETUP;

    /* Here we store callback and parameter from higher level */
    tr_callback USER_CALLBACK;
    pointer USER_PARAM;
/* Begin of <struct_end_1>, DO NOT MODIFY LINES BELOW */
} USB_CLASS_HID_KEYBOARD_INTF_STRUCT, _PTR_
USB_CLASS_HID_KEYBOARD_INTF_STRUCT_PTR;

```

- o Write code to register “***usb\_class\_hid\_init***” function into the “***usb\_classes.c***” file:

```

/* Write your own includes here ...*/
#include "usb_host_hid.h"

void usb_class_hid_keyboard_init(PIPE_BUNDLE_STRUCT_PTR pbs_ptr,
CLASS_CALL_STRUCT_PTR ccs_ptr)
{

```

```

/* End <function_begin_1>, DO NOT MODIFY LINES ABOVE */
/* Write your own code of usb_class_hid_keyboard_init function below
 */
    usb_class_hid_init(pbs_ptr, ccs_ptr);
/* End of <function_end_1>, DO NOT MODIFY LINES BELOW */
} /* End of usb_class_hid_keyboard_init function */
/* DO NOT MODIFY LINES ABOVE */

```

- Also the class and application specific files should be created or added to project from the provided example.

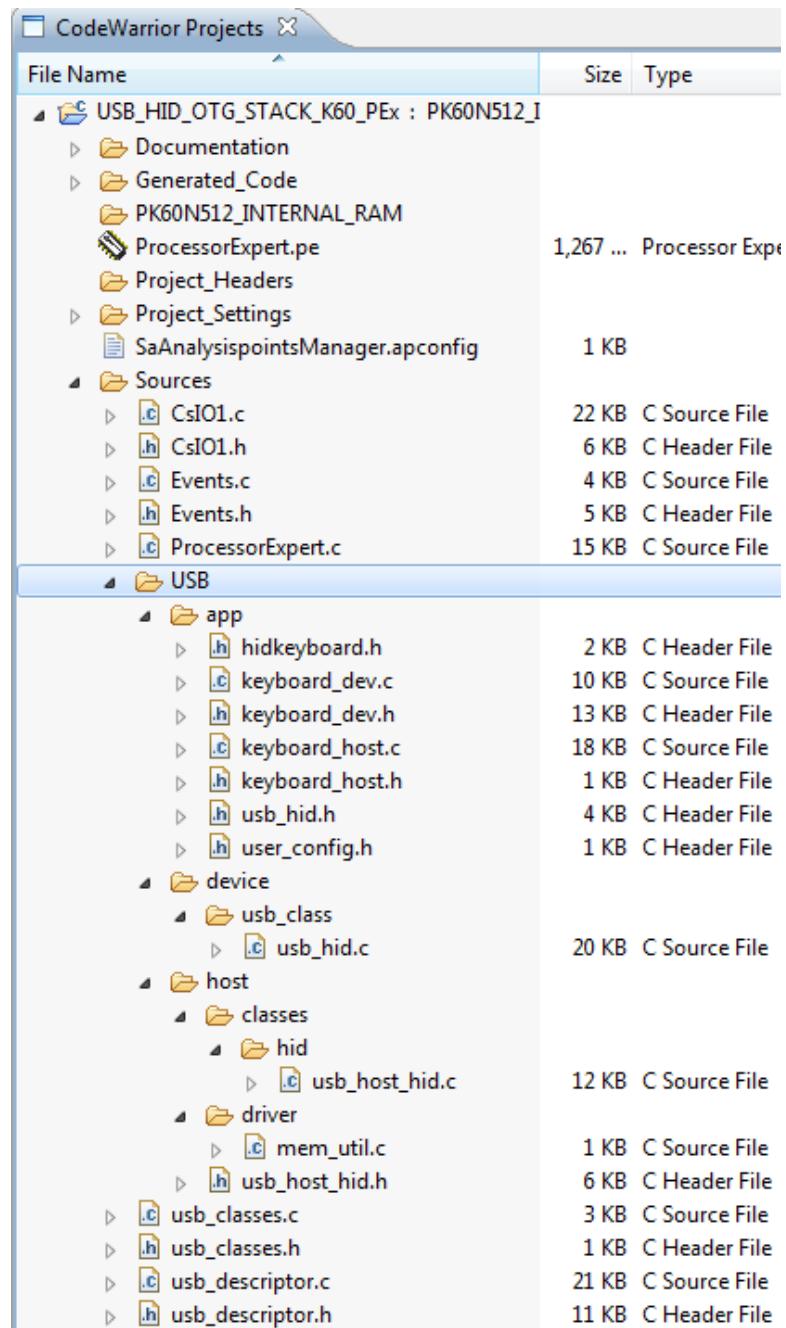


Figure 2-26 USB\_HID\_OTG included files

### 2.3.3 Running the example

This section describes the steps to run USB HID OTG example:

**Step1.** Setup the hardware to run example:

- The computer uses one USB cable to supply power to the board and program USB HID OTG image to the flash. One COM cable is used to get events from the HID OTG device.

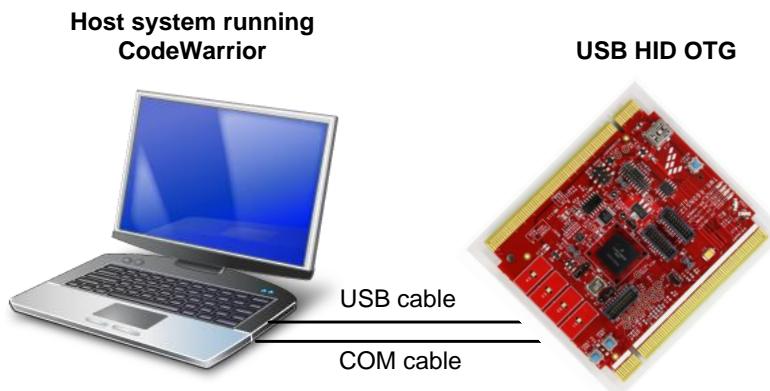


Figure 2-27 Hardware setup

**Step2.** To ensure that the application runs correctly, the HyperTerminal is used to get events from the USB HID OTG device. This software is configured as the following:

- Open HyperTerminal application, enter the name of the connection and click on the OK button.

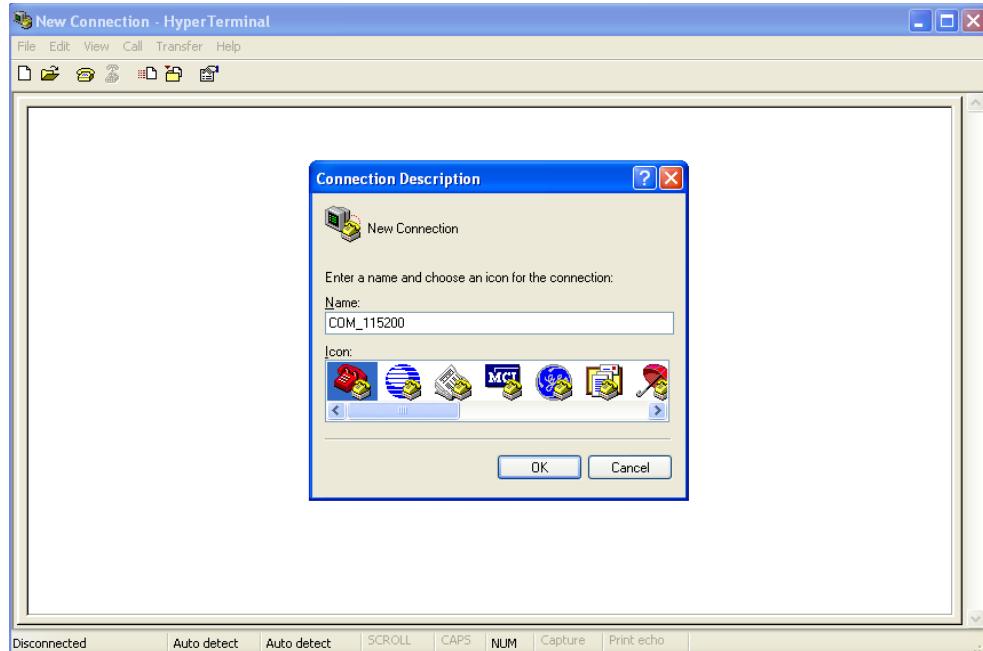


Figure 2-28 HyperTerminal startup

- The window shown in the following figure appears. Select the COM port.

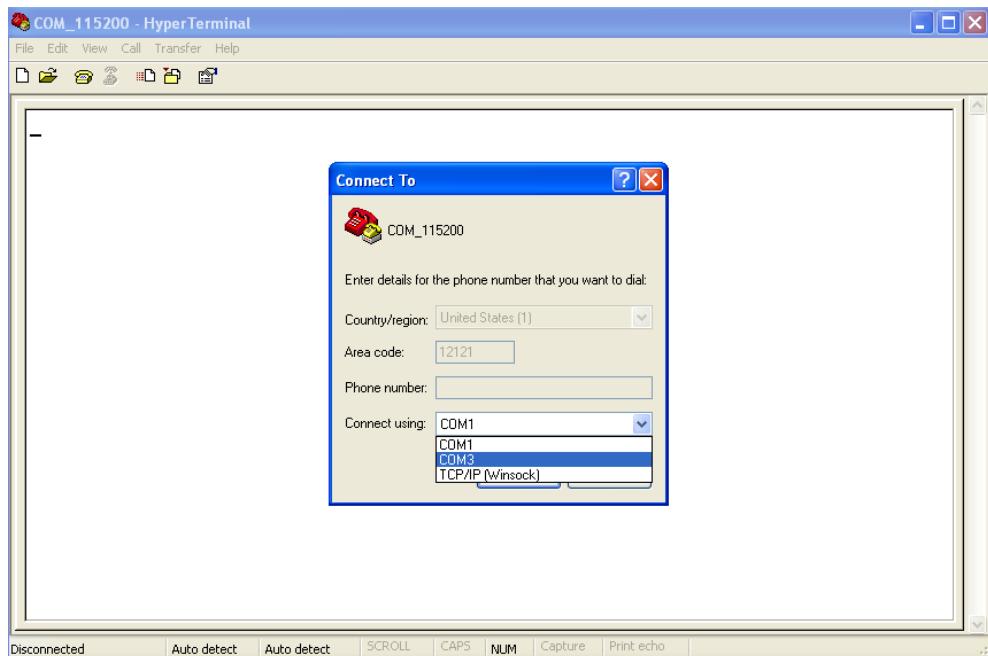


Figure 2-29 Connect using COM port

- Configure the COM port baud rate and other properties as shown below:

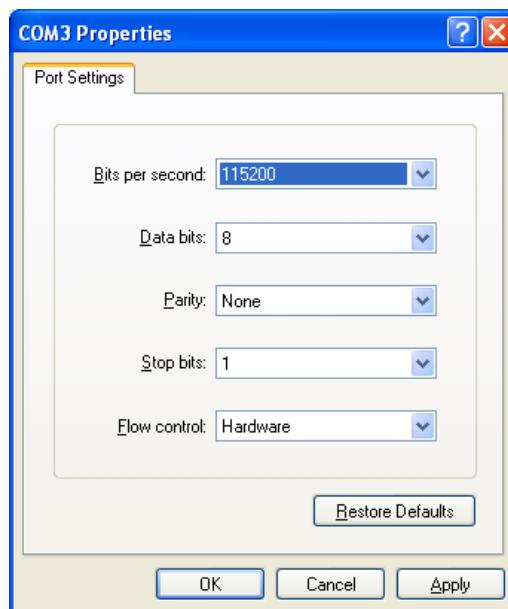


Figure 2-30 COM properties

- The HyperTerminal is now configured as shown below:

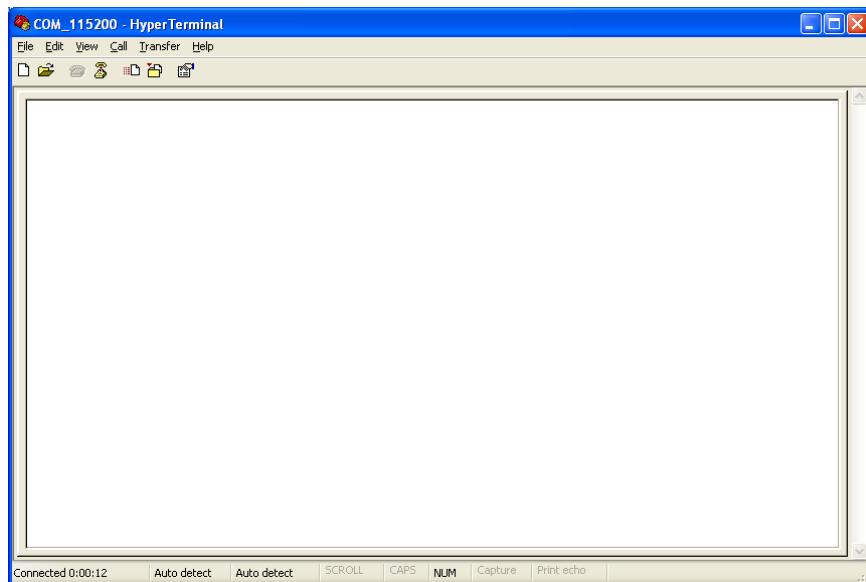


Figure 2-31 Hyper Terminal

**Step3.** Build and load the image of USB HID OTG application to the Kinetis K60 tower board. The HyperTerminal shows the USB HID OTG status as in the following figure:

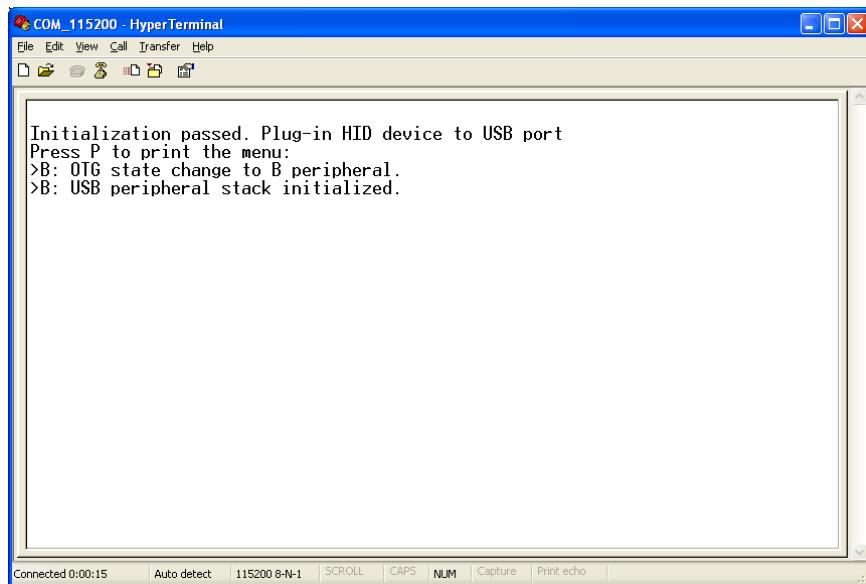


Figure 2-32 USB HID OTG initializing

**Step4.** Connect the USB HID OTG device to the PC, press the switch 2, the data transfer is captured as shown below:

Sp	Index	m:s.ms.us	Len	Err	Dev	Ep	Record	Data
FS	162	0:03.100.678	18 B		02	00	[+] Get Device Descriptor	Index=0 Length=18
FS	175	0:03.103.666	3.00 ms				[4 SOF]	[Frames: 910 - 913]
FS	176	0:03.104.678	9 B		02	00	[+] Get Configuration Descriptor	Index=0 Length=9
FS	189	0:03.107.667	4.00 ms				[5 SOF]	[Frames: 914 - 918]
FS	190	0:03.108.678	39 B		02	00	[+] Get Configuration Descriptor	Index=0 Length=39
FS	207	0:03.112.667	2.00 ms				[6 SOF]	[Frames: 919 - 921]
FS	208	0:03.113.679	0 B		02	00	[+] Set Configuration	Configuration=1
FS	217	0:03.115.668	18.0 ms				[19 SOF]	[Frames: 922 - 940]
FS	218	0:03.132.682	0 B		02	00	[+] Set Idle	Duration=Indefinite Report=0
FS	227	0:03.134.670	5.00 ms				[6 SOF]	[Frames: 941 - 946]
FS	228	0:03.135.683	63 B		02	00	[+] Get Report Descriptor	Index=0 Length=127
FS	245	0:03.140.671	1.01 s				[1015 SOF]	[Frames: 947 - 1961]
FS	246	0:04.152.819	1 B		02	00	[+] Set Output Report	Length=1
FS	259	0:09.272.334	1.99 s		01	03	[+] [193467 ORPHANED]	[Periodic Timeout]
FS	261	0:09.272.652	1.99 s		01	01	[+] [2000 ORPHANED]	[Periodic Timeout]
FS	263	0:03.145.675	1.99 s		02	01	[+] [250 IN-NAK]	[Periodic Timeout]
FS	264	0:04.155.808	1.67 s				[1671 SOF]	[Frames: 1962 - 1584]
FS	265	0:05.145.944	8 B		02	01	[+] Input Report	Keys=[PgUp]
FS	271	0:05.827.033	7.00 ms				[8 SOF]	[Frames: 1585 - 1592]
FS	272	0:05.834.037	8 B		02	01	[+] Input Report	
FS	277	0:04.972.328	1.99 s		01	03	[+] [193144 ORPHANED]	[Periodic Timeout]
FS	279	0:04.972.921	1.99 s		01	01	[+] [2000 ORPHANED]	[Periodic Timeout]
FS	281	0:05.835.034	1.99 s				[2000 SOF]	[Frames: 1593 - 1544] [Periodic Timeout]
FS	282	0:05.842.038	1.99 s		02	01	[+] [250 IN-NAK]	[Periodic Timeout]
FS	283	0:05.872.321	1.99 s		01	03	[+] [19457A ORPHANED]	[Periodic Timeout]

Figure 2-33 USB HID OTG operation

**Step5.** Unplug the USB HID OTG device and then plug the USB keyboard board device to the USB HID OTG device. Press keyboard's button, the HyperTerminal shows the status changing of the USB HID OTG device.

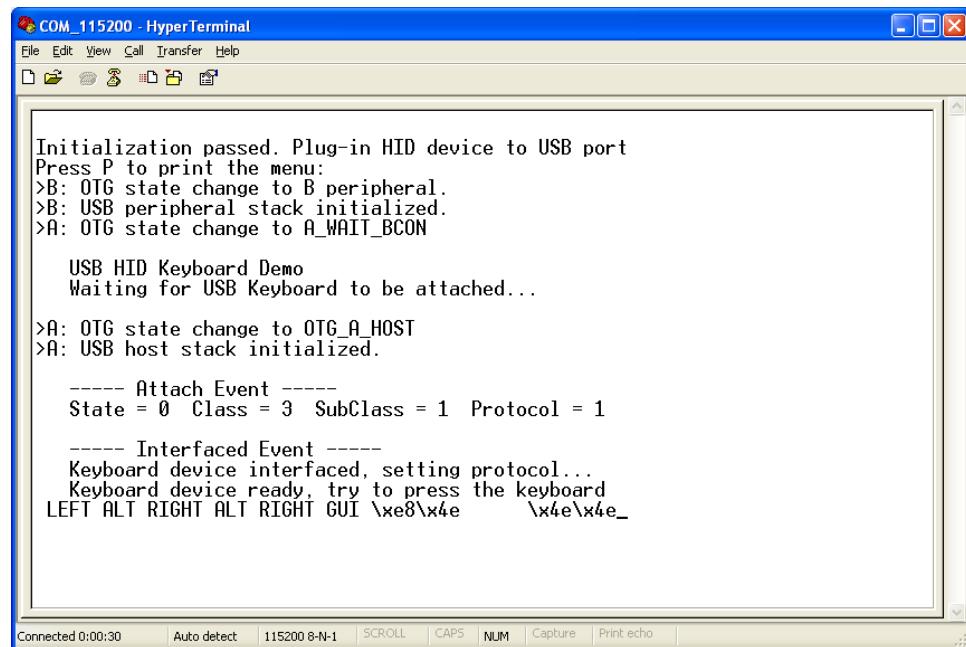


Figure 2-34 USB HID OTG operation

## 3 USB\_HID\_CLASS

This section describes how to run an USB HID example that uses the USB\_HID\_CLASS component in the Processor Expert of Code Warrior.

### 3.1 Architecture overview

The USB\_HID\_CLASS component makes use of the USB\_DEVICE\_STACK, USB\_HOST\_STACK and USB\_OTG\_STACK existing components to provide USB HID class APIs for user.

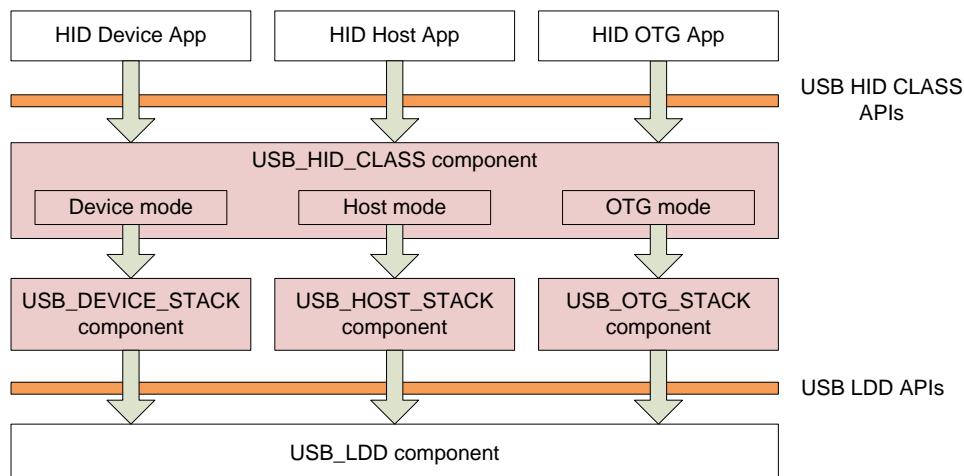


Figure 3-1 Architecture overview

### 3.2 Folder structure

#### 3.2.1 Delivered package structure

The delivered package has the structure as the following:

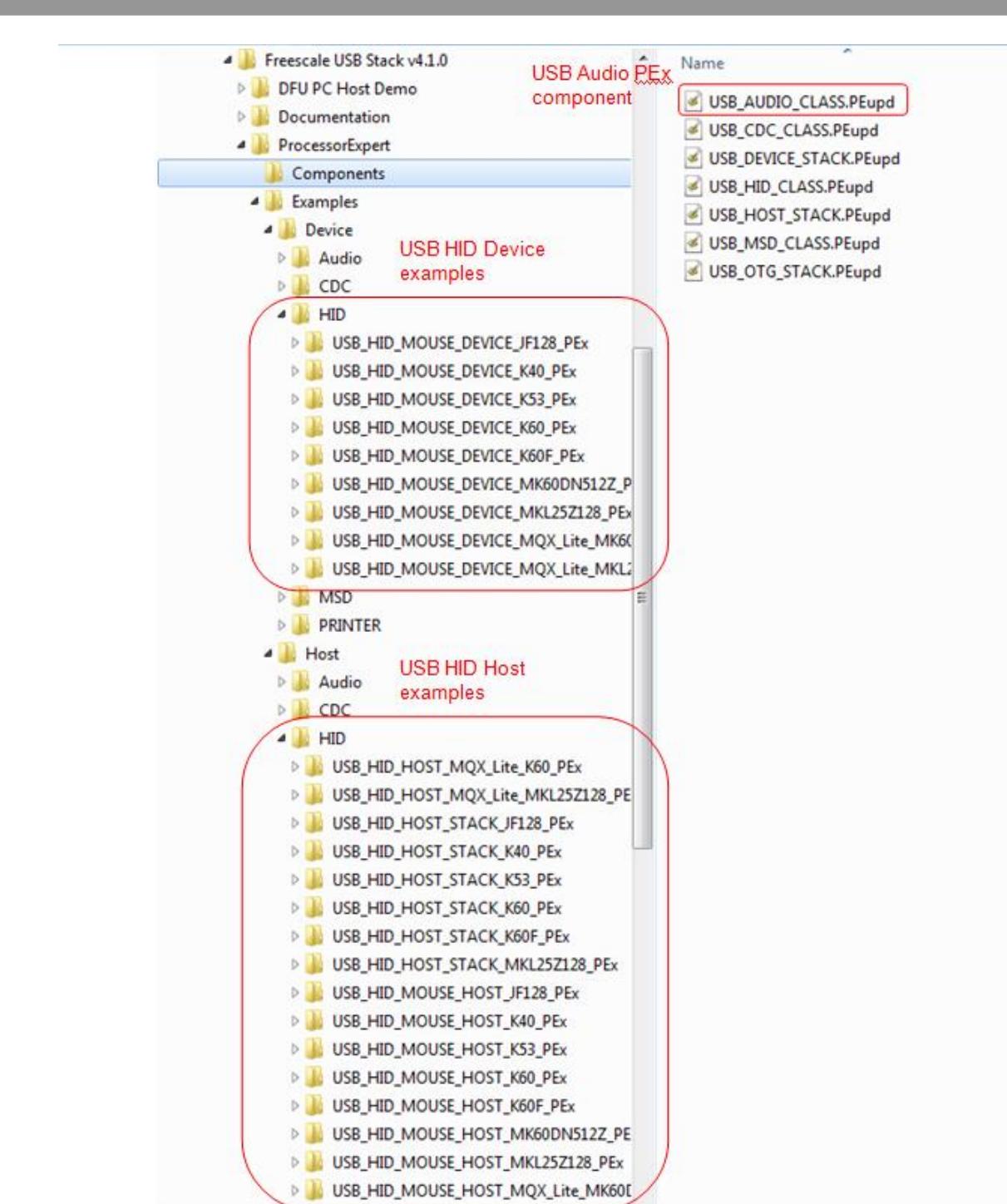


Figure 3-2 Delivered package folder structure

- ❖ **USB\_DEVICE\_STACK** component: This component will encapsulate all the common USB device stack functionality.
- ❖ **USB\_HOST\_STACK** component: This component will encapsulate all the common USB host stack functionality.
- ❖ **USB\_OTG\_STACK** component: This component will encapsulate all the common USB OTG stack functionality.
- ❖ **USB\_HID\_CLASS** component: This component will encapsulate all the Human Interface Device Class functionality.

### 3.2.2 USB HID example structure

- The USB HID mouse device examples have the structure as shown below:

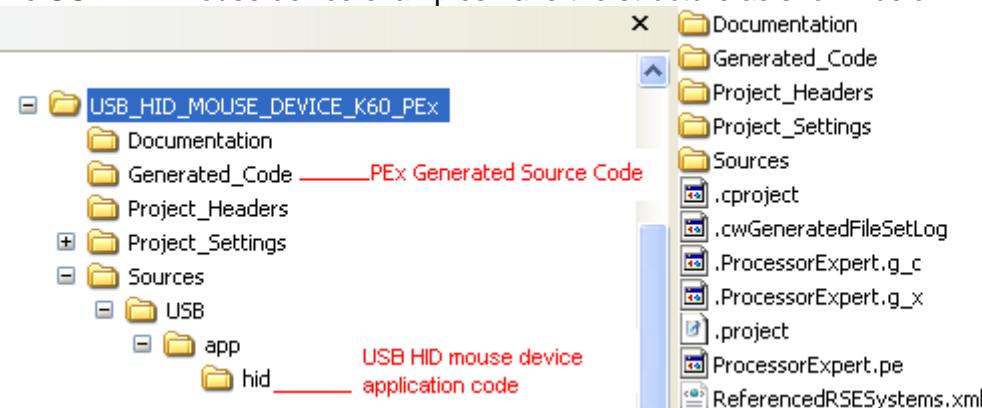


Figure 3-3 USB HID mouse device example folder structure

- The USB HID mouse host examples have the structure as shown below:

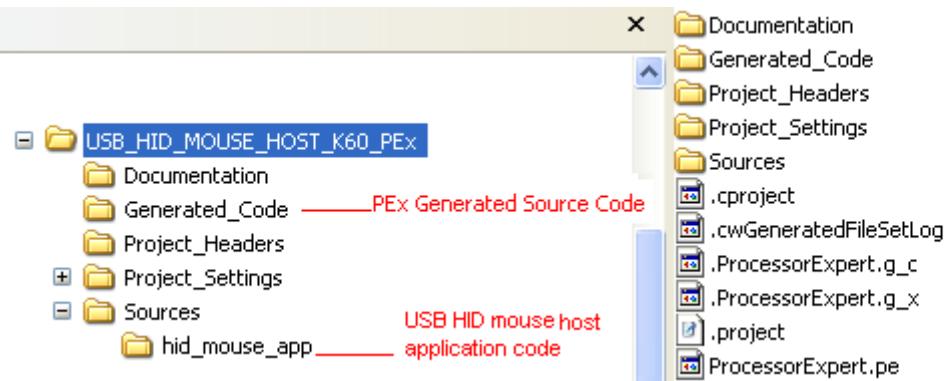


Figure 3-4 USB HID mouse host example folder structure

- The USB HID keyboard OTG examples have the structure as shown below:

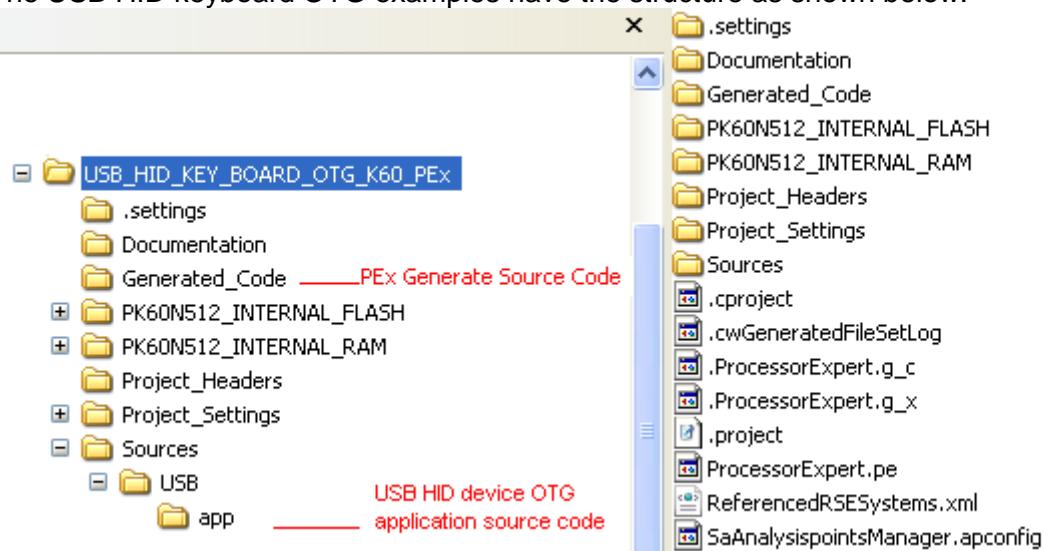


Figure 3-5 USB HID keyboard OTG example folder structure

## 3.3 Running USB HID example

### 3.3.1 USB HID mouse device example

#### 3.3.1.1 Generating Processor Expert source code

This section describes the steps to generate Processor Expert source code that is used to run the USB HID mouse device example.

**Step1.** Import USB\_DEVICE\_STACK, USB\_HOST\_STACK, USB\_OTG\_STACK and USB\_HID\_CLASS components from the package to Components Library.

**Step2.** Open the USB\_HID\_MOUSE\_DEVICE\_K60\_PEx example with CW10.3 → open Components Library and add the USB\_HID\_CLASS component to project → Select HID device mode → the user interface is shown below:

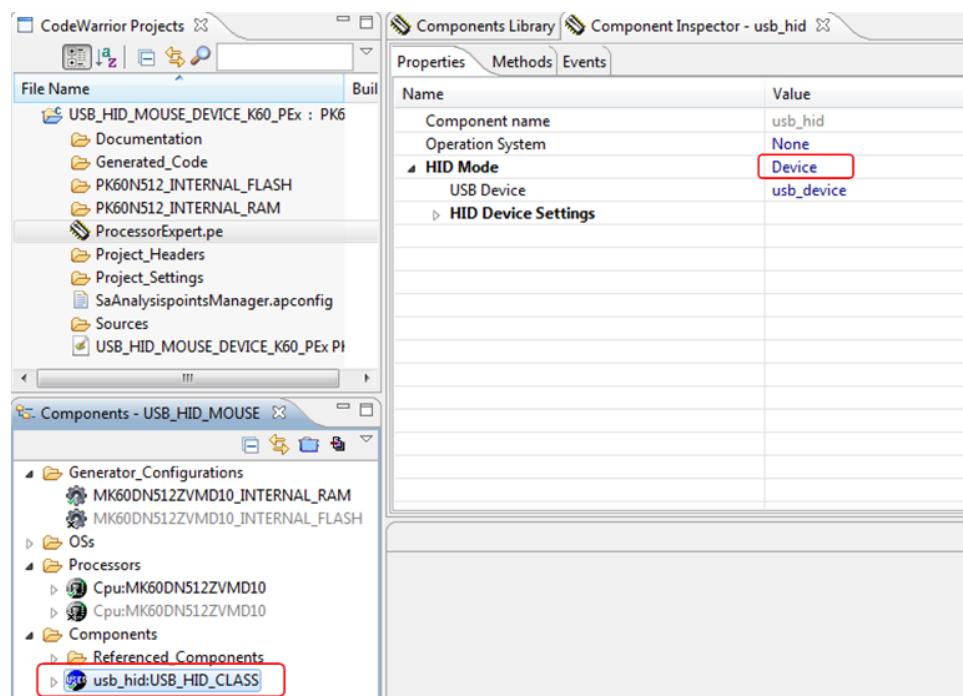


Figure 3-6 USB HID device mode

**Step3.** Configure CPU clock to run USB module:

- Select CPU target from Project Panel window to setup clock source as in the following figure:
  - o Target: Cpu:MK60N512VMD100
  - o Clock source: External reference clock 50MHz
  - o MCG mode: PEE
  - o MCG output: 48 MHz
  - o PLL output: 48MHz
  - o Core clock: 48MHz
  - o Bus clock: 48MHz
  - o External bus clock: 24MHz
  - o Flash clock: 24MHz

\*Component Inspector - Cpu

Properties Methods Events Build options Resources		
Name	Value	Details
CPU type	MK60DN512ZVMD10	
Clock settings		
Internal oscillator		
Slow internal reference clock [kHz]	32.768	32.768 kHz
Fast internal reference clock [MHz]	4.0	4 MHz
RTC oscillator	Disabled	
System oscillator	Enabled	
Clock source	External reference clock	
Clock frequency [MHz]	50.0	50 MHz
Clock source settings	1	
Clock source setting 0		
MCG settings		
MCG mode	PEE	
MCG output [MHz]	48.0	48 MHz
MCG external ref. clock source	System oscillator	
MCG external ref. clock [MHz]	50.0	50 MHz
FLL settings		
PLL settings		
PLL module	Enabled	
PLL output [MHz]	48.0	48 MHz
Initialization priority	minimal priority	
Watchdog disable	yes	
CPU interrupts/resets		
External Bus	Disabled	
Clock configurations		
Clock configuration 0		
Clock source setting		
MCG mode	PEE	
System clocks		
Core clock	48.0	48 MHz
Bus clock	48.0	48 MHz
External bus clock	24.0	24 MHz
Flash clock	24.0	24 MHz

Figure 3-7 USB Clock settings

#### Step4. Configure the USB\_HID\_CLASS component:

- ❖ Configure the USB HID standard descriptor as shown below:

CodeWarrior Projects

File Name: USB\_HID\_MOUSE\_DEVICE\_K60\_PEx : PK6

Components Library Component Inspector - usb\_hid

Properties Methods Events	
Name	Value
Component name	usb_hid
Operation System	None
HID Mode	Device
USB Device	usb_device
HID Device Settings	
Protocol code	USB_PROTOCOL_HID_MOUSE
Subclass code	USB_SUBCLASS_HID_BOOT
Interface number	0
Endpoint 0 max packet size	16
Speed	Full Speed
Endpoints	
USB_INTERRUPT IN Endpoint	
Endpoint number	1
Max packet size	8
HID Report Descriptor	
Report Descriptor Item List	27
Item0	

Components - USB\_HID\_MOUSE

Components

Referenced Components

usb\_deviceUSB\_DEVICE\_STACK[Usb\_D]  
usb\_hid:USB\_HID\_CLASS

Figure 3-8 USB HID standard descriptor settings

- ❖ Click the plus button to configure the USB HID Report descriptor:

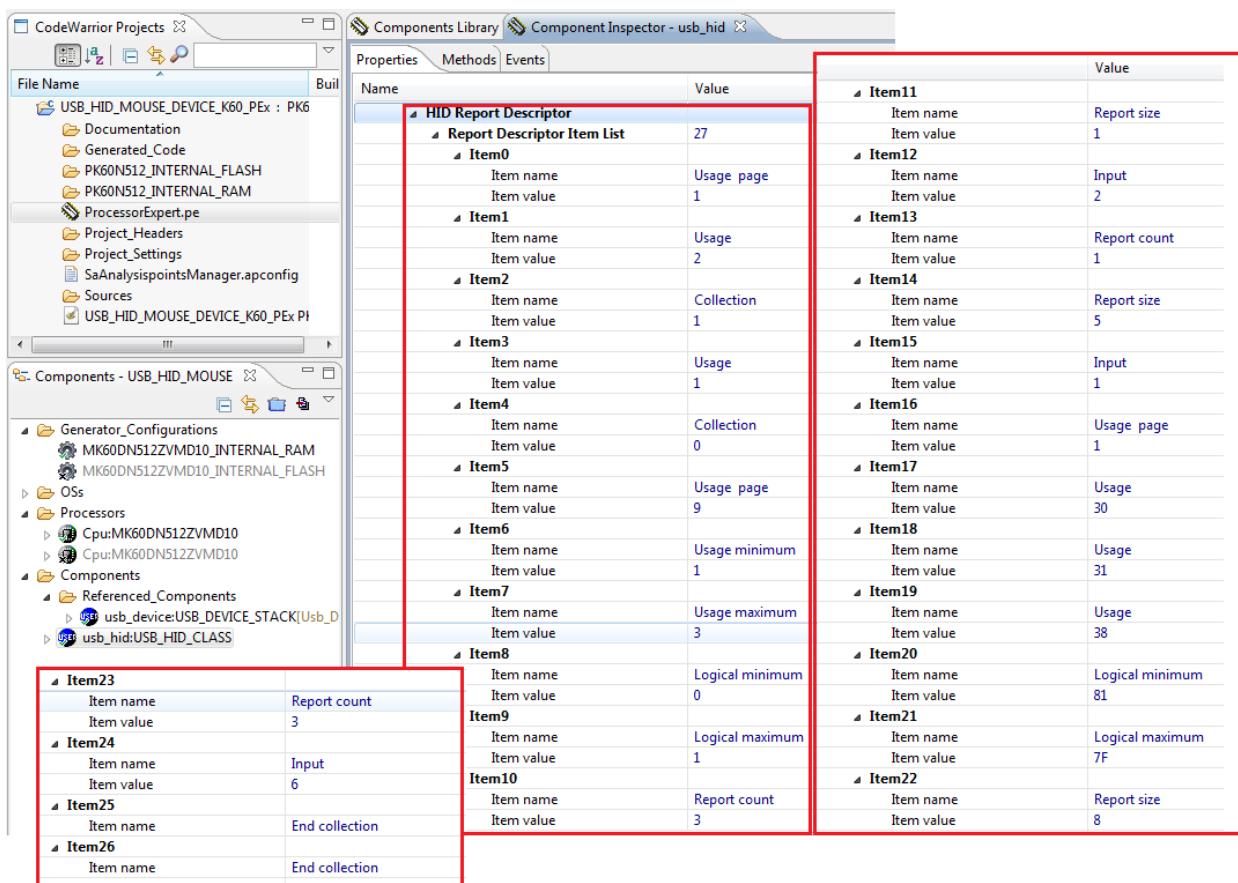


Figure 3-9 USB HID mouse report descriptor settings

- Step5.** Configure the USB\_DEVICE\_STACK to run USB HID example
- ❖ Configure the USB device descriptor as in the following figure:

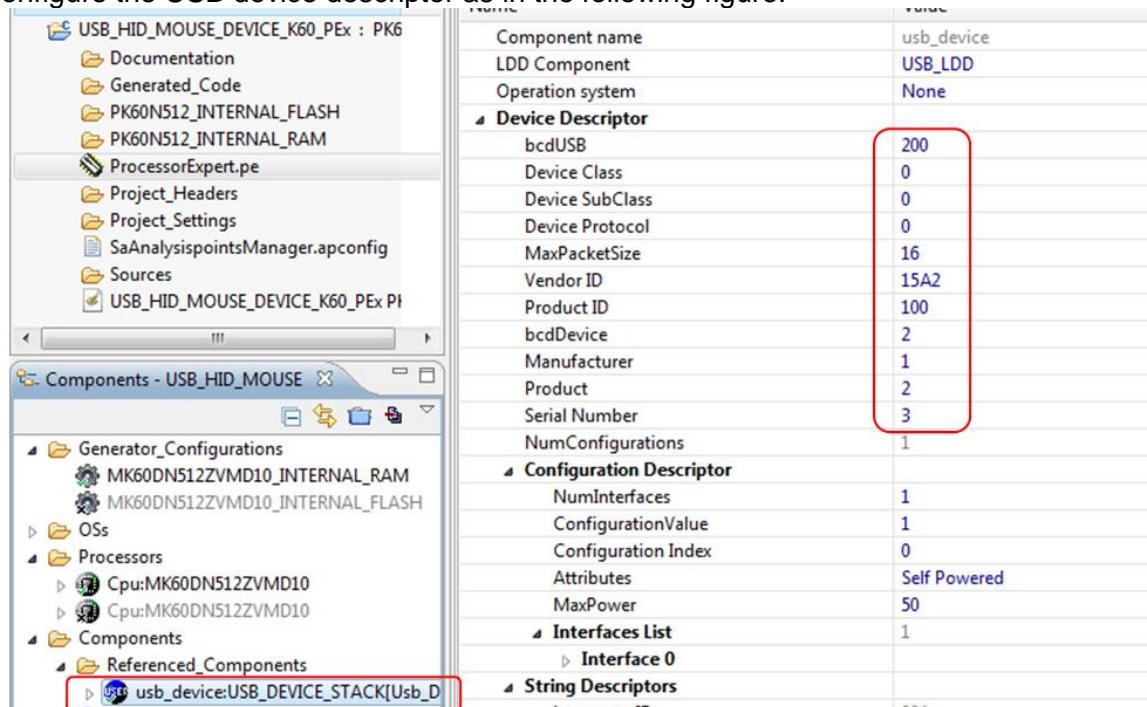


Figure 3-10 USB Device Descriptor settings

Freescale Processor Expert USB Components Quick Start Guide

- ❖ Configure the USB HID class specific descriptor. This descriptor is inherited from the USB\_HID\_CLASS component.

Name	Value
NumConfigurations	1
<b>Configuration Descriptor</b>	
NumInterfaces	1
ConfigurationValue	1
Configuration Index	0
Attributes	Self Powered
MaxPower	50
<b>Interfaces List</b>	
<b>Interface 0</b>	1
<b>String Descriptors</b>	
Language ID	904
<b>String Descriptor</b>	
String 1	2
String 2	FREESCALE SEMICONDUCTOR INC. USB HID MOUSE DEVICE PEx DEMO

Figure 3-11 USB HID Class specific descriptor settings

- ❖ Configure the USB String descriptor. Click the plus button and then fill the string values as shown below:

Name	Value
<b>Interfaces List</b>	1
<b>Interface 0</b>	
<b>String Descriptors</b>	
Language ID	904
<b>String Descriptor</b>	
String 1	2
String 2	FREESCALE SEMICONDUCTOR INC. USB HID MOUSE DEVICE PEx DEMO

Figure 3-12 USB String Descriptor settings

**Step6.** Configure the USB\_LDD component to run USB HID mouse device example as in the following figure:

Name	Value
Component name	USB_LDD
USB module (SIE)	USB0
Input clock frequency [MHz]	48 MHz
<b>Interrupt service/event</b>	Enabled
Interrupt	INT_USB0
Interrupt priority	medium priority
<b>Mode</b>	DEVICE
<b>Transceiver type</b>	Internal
Transceiver module	USB0_FS
Transceiver weak pull-downs	Enabled
USB revision	USB 2.0
Device data rate	Full speed
<b>Pin/signal setting</b>	Disabled
Clock pin	
USB data pins	
VBUS pins	
Pullup/Pulldown pins	
<b>Device mode setting</b>	
<b>EP list</b>	16
<b>EP 0</b>	Enabled
<b>EP 1</b>	Enabled
Control transfer	Disabled
Bulk OUT transfer	Disabled
Bulk IN transfer	Disabled
Interrupt OUT transfer	Disabled
Interrupt IN transfer	Enabled
Max. packet size	8
Max. queue size	1
<b>Isochronous OUT transfer</b>	Disabled
<b>Isochronous IN transfer</b>	Disabled
EP 2	Disabled

Figure 3-13 USB\_LDD component device mode settings

**Step7.** Generate Processor Expert code:

After the setting of the components, select ProcessorExpert.pe in the project folder → Right click → Choose Generate Processor Expert Code field to generate source code.

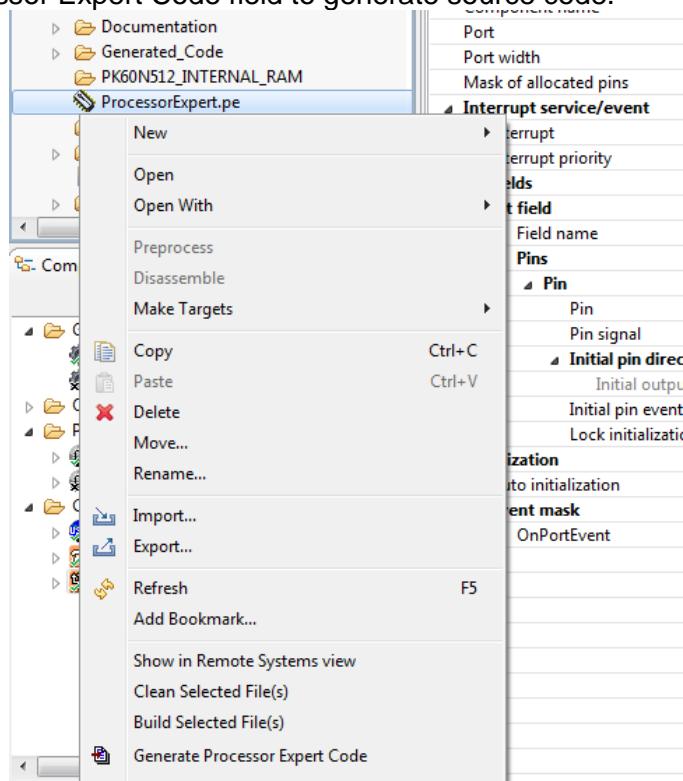


Figure 3-14 Generate Processor Expert Code

### 3.3.1.2 Running the USB HID mouse device example

This section describes the steps to run the USB HID mouse device example.

**Step1.** Setup the hardware as in the following figure:

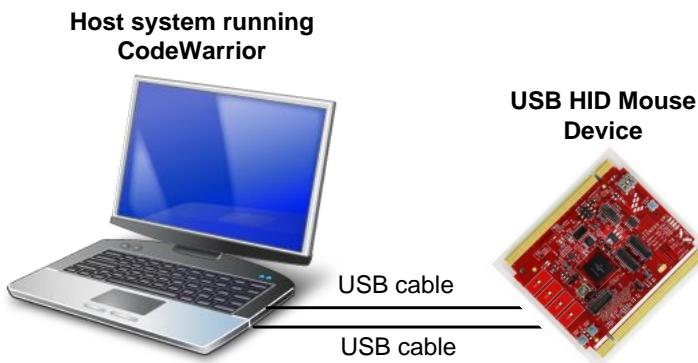


Figure 3-15 HID mouse device demo setup

**Step2.** After source code has been generated, build and program the USB HID mouse device image to the flash.

**Step3.** Connect the USB HID mouse device to the PC by one USB cable. After the USB HID mouse device completed the enumeration, it will send data to move mouse pointer automatic. The transferred data is captured as in the following figure:

\*Untitled - Total Phase Data Center

Sp	Index	m:s.ms.us	Len	Err	Dev	Ep	Record	Data
FS	37	0:02.239.203	18 B		01	00	[4 SOF]	Index=0 Length=18 [Frames: 1967 - 1970]
FS	54	0:02.243.201	3.00 ms				[Get Configuration Descriptor]	Index=0 Length=9 [Frames: 1971 - 1977]
FS	55	0:02.244.204	9 B		01	00	[7 SOF]	Index=0 Length=255 [Frames: 1978 - 1980]
FS	68	0:02.247.201	6.00 ms				[Get Configuration Descriptor]	Index=0 Length=10 [Frames: 1981 - 1985]
FS	69	0:02.248.204	34 B		01	00	[3 SOF]	Index=0 Length=255 [Frames: 1986 - 1993]
FS	90	0:02.254.202	2.00 ms				[Control Transfer (STALL)]	Index=0 Length=10 [Frames: 1994 - 1996]
FS	91	0:02.255.205	0 B		01	00	[5 SOF]	Index=0 Length=255 [Frames: 1999 - 2006]
FS	99	0:02.257.202	4.00 ms				[Get String Descriptor]	Index=0 Length=255 [Frames: 2007 - 2028]
FS	100	0:02.258.206	4 B		01	00	[8 SOF]	Index=0 Length=18 [Frames: 2029 - 2032]
FS	113	0:02.262.203	7.00 ms				[Get String Descriptor]	Index=2 Length=255 [Frames: 2033 - 2040]
FS	114	0:02.263.206	60 B		01	00	[5 SOF]	Index=0 Length=34 [Frames: 2041 - 2043]
FS	139	0:02.270.204	4.00 ms				[Get String Descriptor]	Index=0 Length=10 [Frames: 2044 - 15]
FS	140	0:02.271.208	4 B		01	00	[8 SOF]	Configuration=1 [Frames: 24 - 40]
FS	153	0:02.275.205	7.00 ms				[Get String Descriptor]	Duration=Indefinite Report=0 Axes=[X:8 Y:0 VWheel:0] [Frames: 41 - 48]
FS	154	0:02.276.208	60 B		01	00	[22 SOF]	Axes=[X:8 Y:0 VWheel:0] [Frames: 49 - 51]
FS	179	0:02.283.206	21.0 ms				[Get Device Descriptor]	Index=0 Length=18 [Frames: 52 - 54]
FS	180	0:02.301.212	18 B		01	00	[4 SOF]	Index=0 Length=9 [Frames: 55 - 57]
FS	197	0:02.305.209	3.00 ms				[Get Configuration Descriptor]	Index=0 Length=9 [Frames: 58 - 60]
FS	198	0:02.306.212	9 B		01	00	[8 SOF]	Index=0 Length=34 [Frames: 61 - 63]
FS	211	0:02.309.209	7.00 ms				[Get Configuration Descriptor]	Index=0 Length=10 [Frames: 64 - 66]
FS	212	0:02.312.213	34 B		01	00	[Set Idle]	Index=0 Length=116 [Frames: 67 - 69]
FS	233	0:02.317.210	2.00 ms				[8 SOF]	Duration=Indefinite Report=0 Axes=[X:8 Y:0 VWheel:0] [Frames: 70 - 72]
FS	234	0:02.318.214	0 B		01	00	[Set Configuration]	Index=0 Length=116 [Frames: 73 - 75]
FS	243	0:02.320.211	19.0 ms				[17 SOF]	Index=0 Length=116 [Frames: 76 - 78]
FS	244	0:02.338.216	0 B		01	00	[Input Report]	Index=0 Length=116 [Frames: 79 - 81]
FS	253	0:02.340.213	7.00 ms				[8 SOF]	Index=0 Length=116 [Frames: 82 - 84]
FS	254	0:02.341.217	52 B		01	00	[Input Report]	Index=0 Length=116 [Frames: 85 - 87]
FS	279	0:02.348.215	16.0 ms				[10 SOF]	Index=0 Length=116 [Frames: 88 - 90]
FS	280	0:02.364.220	4 B		01	01	[Input Report]	Index=0 Length=116 [Frames: 91 - 93]
FS	285	0:02.365.217	7.00 ms				[8 SOF]	Index=0 Length=116 [Frames: 94 - 96]
FS	286	0:02.372.221	4 B		01	01	[Input Report]	Index=0 Length=116 [Frames: 97 - 99]
FS	287	0:02.372.218	7.00 ms				[10 SOF]	Index=0 Length=116 [Frames: 100 - 102]

Figure 3-16 USB HID mouse device operation

### 3.3.2 USB HID mouse host example

#### 3.3.2.1 Generating Processor Expert source code

This section describes the steps to generate Processor Expert source code that is used to run the USB HID mouse host example.

**Step1.** Import components from the package to the Components Library → Open the `USB_HID_MOUSE_HOST_K60_PEx` example with CW10.3 → Open Components Library and add the `USB_HID_CLASS` component to project → Select HID host mode, the user interface is shown below:

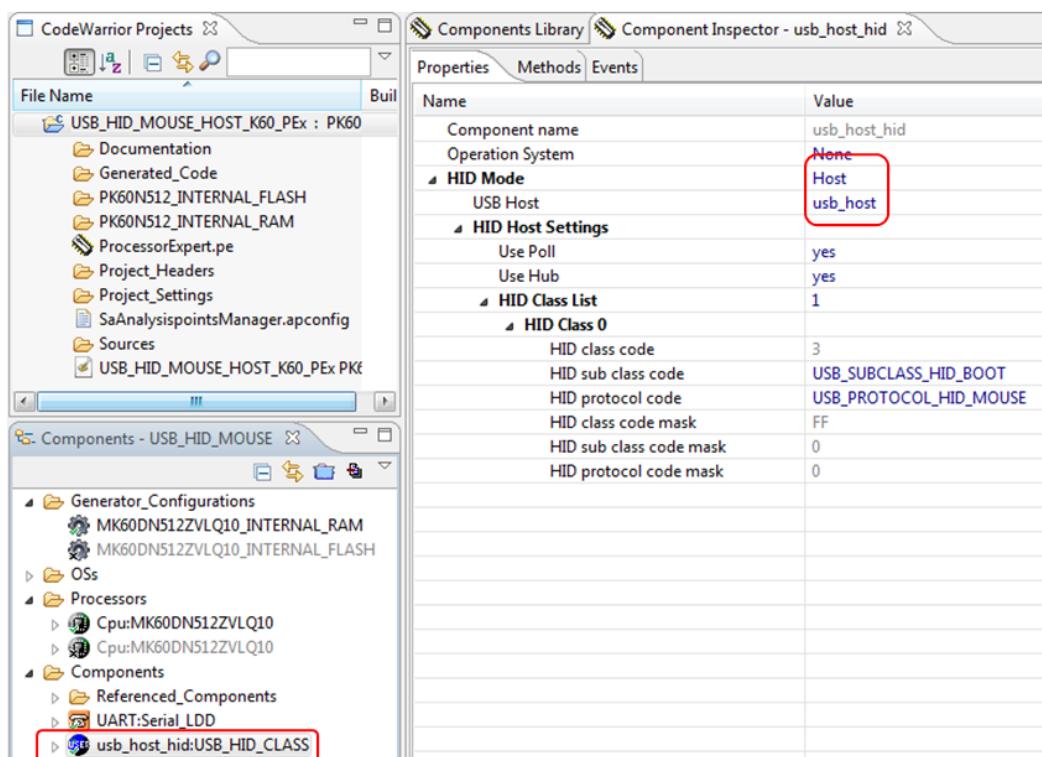


Figure 3-17 USB HID Host mode

**Step2.** Configure the clock to run USB module. Refer to the USB HID mouse device example above.

**Step3.** Configure the USB\_HID\_CLASS component:

Click the plus button to add the HID class to the HID class list, configure the HID class as in the following figure:

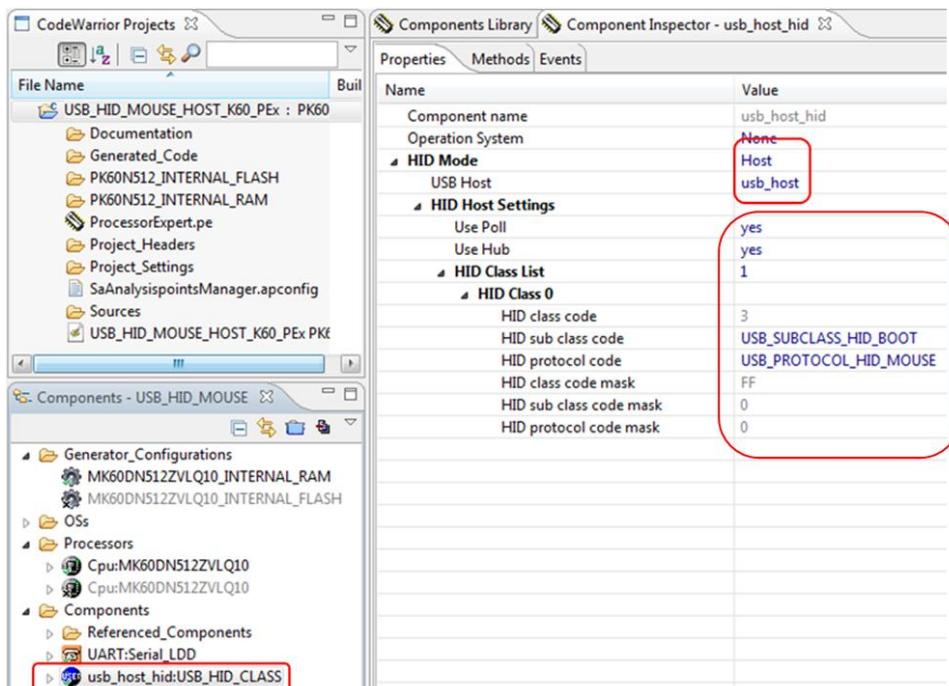


Figure 3-18 USB HID Host settings

Freescale Processor Expert USB Components Quick Start Guide

**Step4.** Configure the USB\_HOST\_STACK to run the USB HID mouse host example. Those settings are inherited from the USB HID CLASS component.

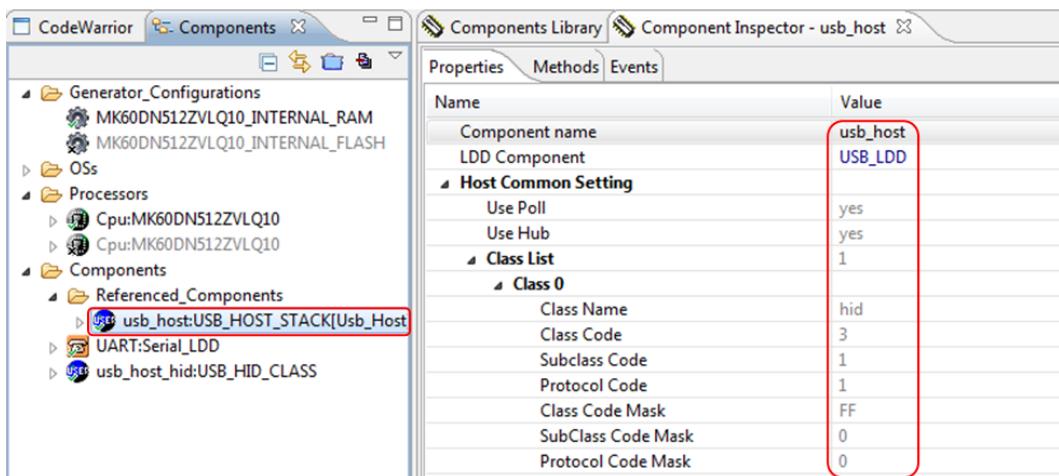


Figure 3-19 USB\_HOST\_STACK component settings

**Step5.** Configure the USB\_LDD component to run the USB HID mouse host example as in the following figure:

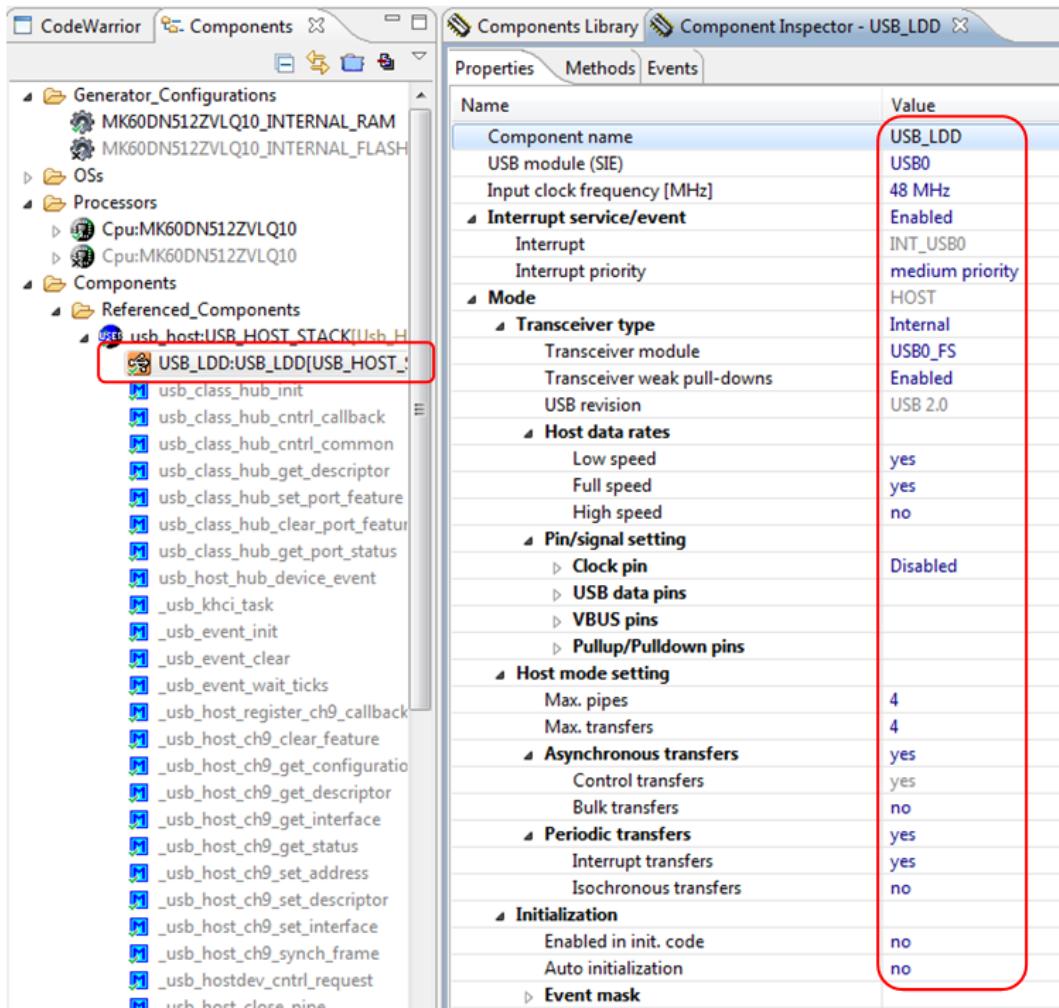


Figure 3-20 USB\_LDD component host mode settings

**Step6.** To get USB HID mouse host events, we must use the Serial\_LDD component to configure the UART module.

- ❖ Select the Serial\_LDD component from the Components Library and add it into project.
- ❖ Configure the Serial\_LDD component as in the following figure:

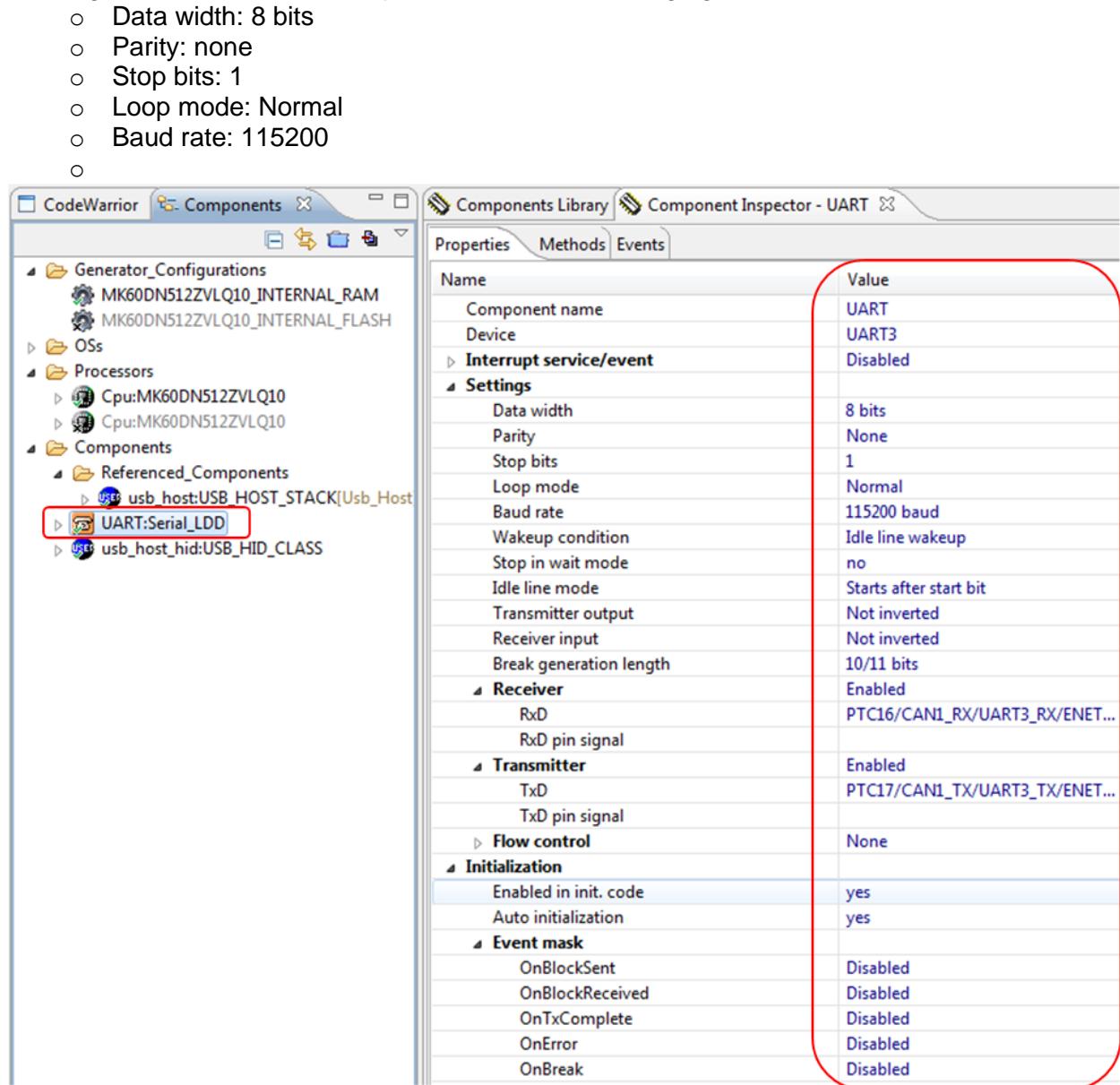


Figure 3-21 Serial\_LDD component settings

**Step7.** Generate USB\_HID\_CLASS source code to run USB HID mouse host example. Refer to the USB HID mouse device example above.

### 3.3.3 Running the USB HID mouse host example

This section describes steps to run the USB HID host example.

#### Step1. Hardware settings

- The computer uses one USB cable to supply power to the board and program USB HID OTG image to the flash. One COM cable is used to get events from the USB HID mouse host.

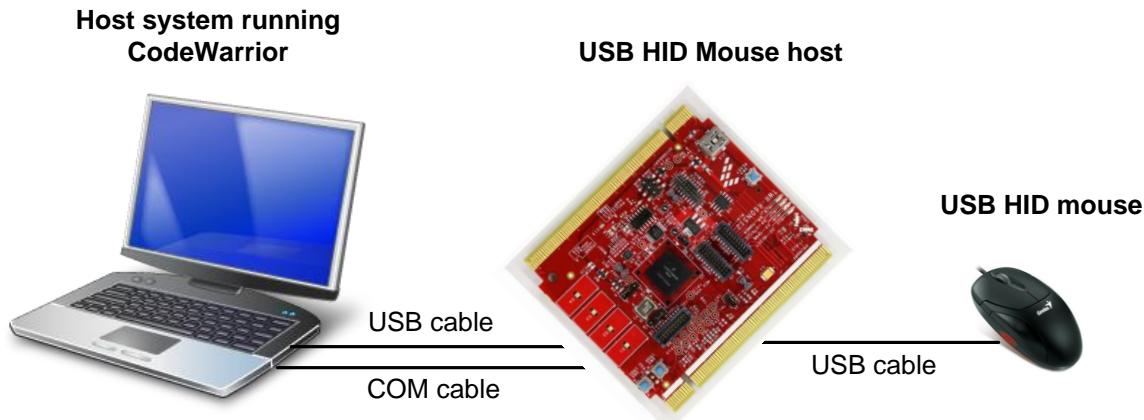


Figure 3-22 HID mouse host demo setup

Step2. To run the USB HID mouse host example, after the generation of the source code, we must add the code to transfer between the USB generated code and the USB mouse host application.

- ❖ Add code into “**device\_driver\_info.c**” file to register USB Host event callback functions as shown below:

```
#include "hidmouse.h"

void usb_host_hid_mouse_event
(
    /* [IN] pointer to device instance */
    usb_device_instance_handle    dev_handle,
    /* [IN] pointer to interface descriptor */
    usb_interface_descriptor_handle intf_handle,
    /* [IN] code number for event causing callback */
    uint_32                      event_code
)
{
    /* Write code here ... */
    usb_host_hid_mouse_app_event(dev_handle, intf_handle, event_code);
}
```

**Step3.** To ensure that the application runs correctly, the HyperTerminal is used to get events from the USB HID mouse host device. This software is configured as in the following steps:

- Open HyperTerminal application, enter the name of the connection and click on the OK button

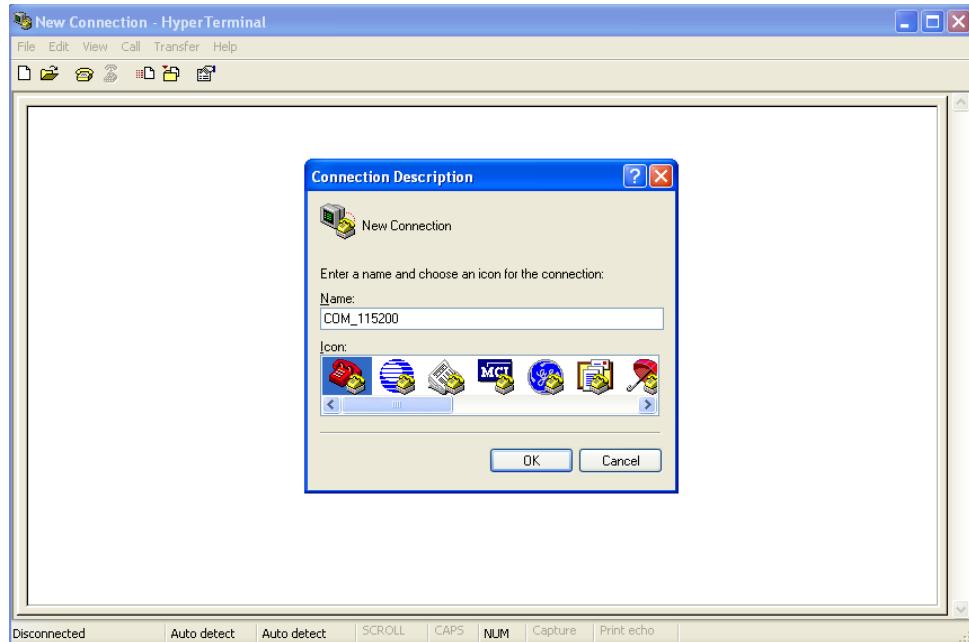


Figure 3-23 HyperTerminal startup

- The window shown in the following figure appears. Select the COM port.

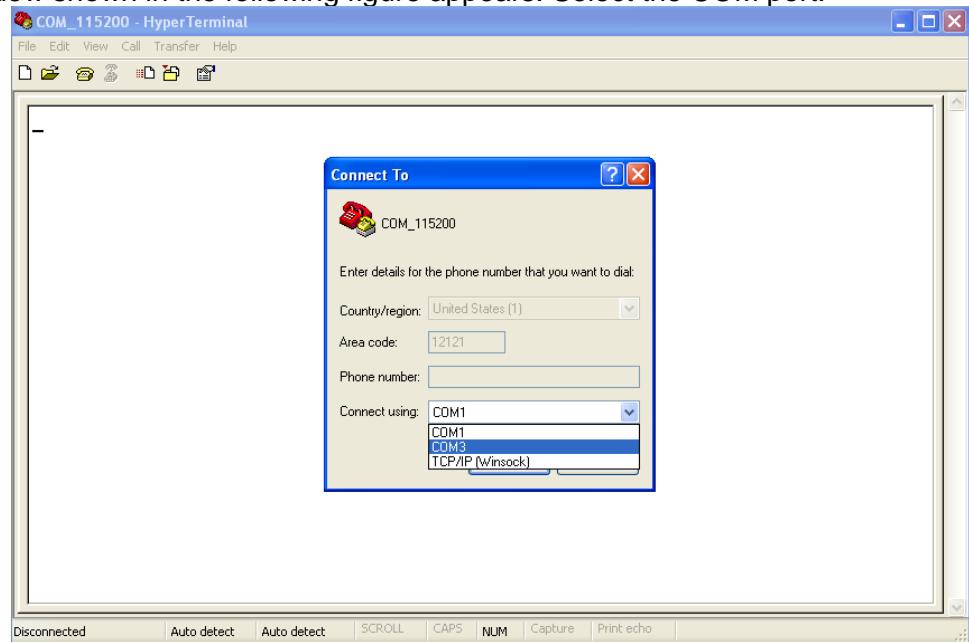


Figure 3-24 Connect using COM port

- Configure the COM port baud rate and other properties as in the figure below:

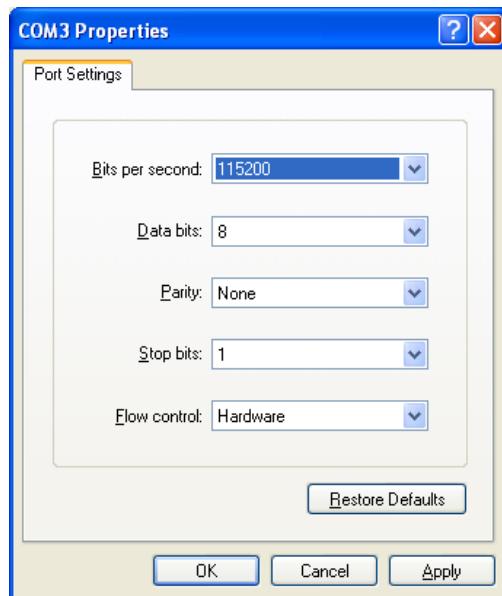


Figure 3-25 COM properties

- The HyperTerminal is now configured as shown in the figure below:

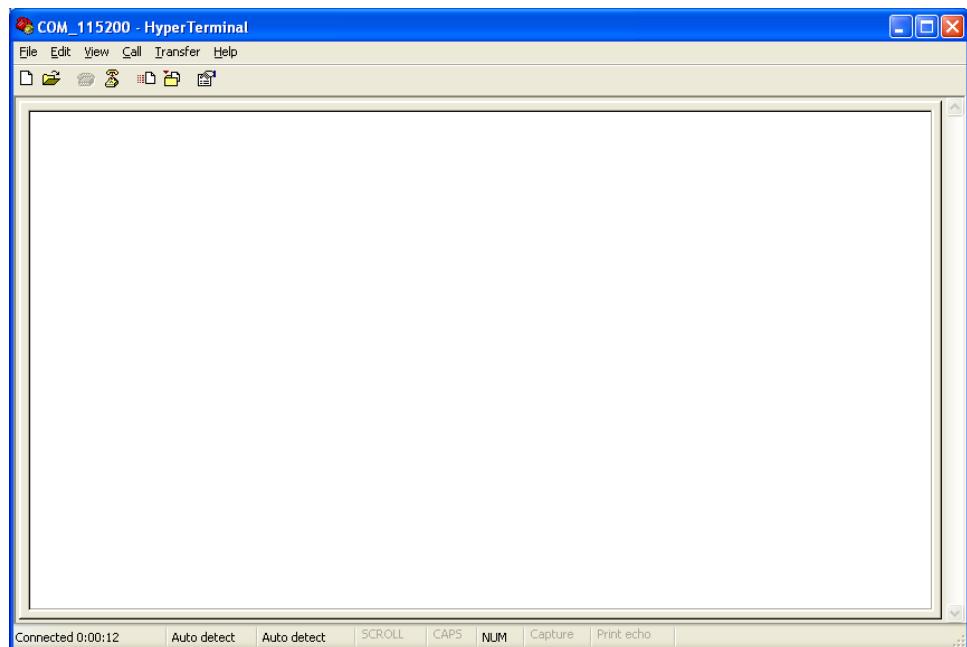


Figure 3-26 Hyper Terminal

**Step4.** Build and load the image of USB HID mouse host application to the Kinetis K60 tower board. The HyperTerminal shows the USB HID mouse host status as in the following figure:

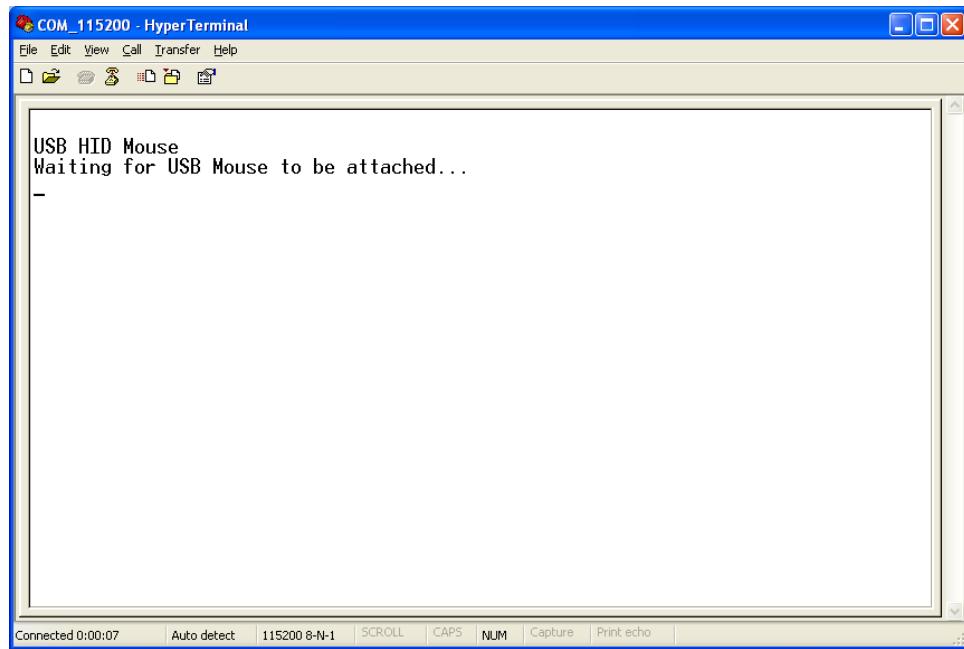


Figure 3-27 USB mouse host initialize

**Step5.** Plug the mouse device to the USB mouse host, the HyperTerminal shows the operation as in the following figure:

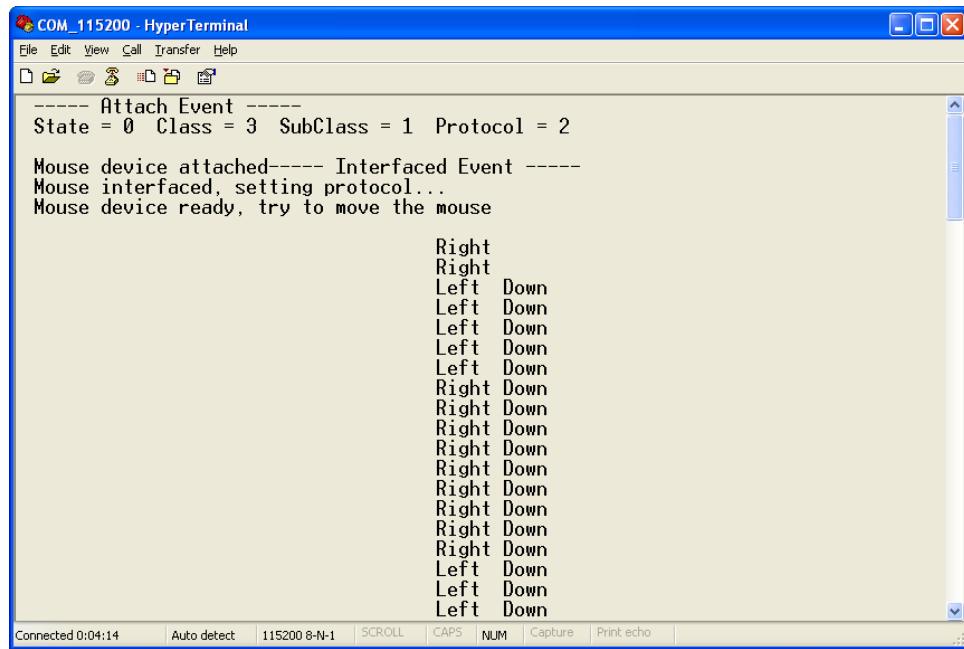


Figure 3-28 USB mouse host operation

### 3.3.4 USB HID keyboard OTG example

#### 3.3.4.1 Generating Processor Expert source code

This section describes the steps to generate Processor Expert source code that is used to run the USB HID keyboard OTG example.

**Step1.** Import components from the package to the Components Library → Open the USB\_HID\_KEYBOARD\_OTG\_K60\_PEx example with CW10.3 → Open Components Library and add the USB\_HID\_CLASS component to project → select HID OTG mode, the user interface is shown below

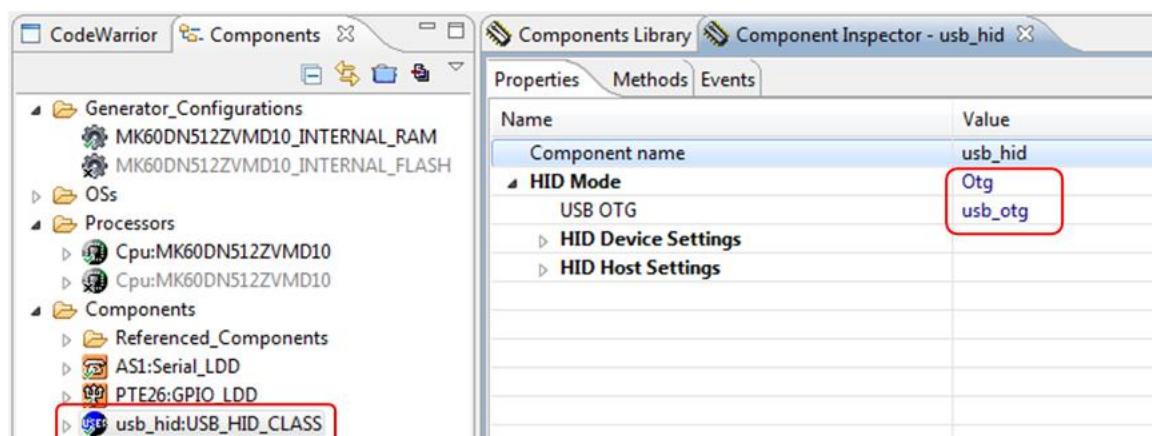


Figure 3-29 USB HID OTG mode

**Step2.** Configure the clock to run USB module

- ❖ Select CPU target from Project Panel window to setup the USB clock: Refer to the USB HID mouse device example above.
- ❖ From Component Inspector window chose advanced tab button to configure internal peripherals that are used to run OTG example.

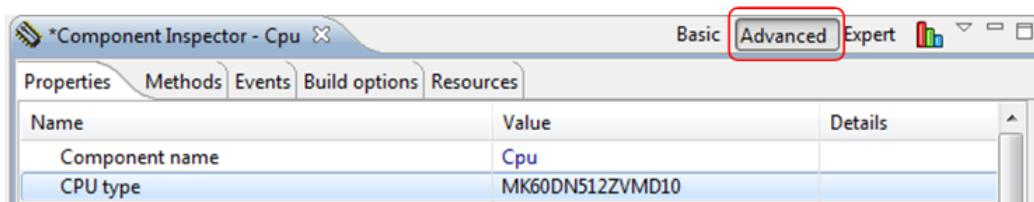


Figure 3-30 Open Advanced Tab

- Chose Internal Peripherals field → enable Flexible memory controller → chose Master 4(USB OTG) → chose Access protection - Read-Write.

Component Inspector - Cpu

Basic Advanced Expert

Properties Methods Events Build options Resources

Name	Value	Details
Initialization priority	minimal priority	15
Watchdog disable	yes	
Internal peripherals		
NMI pin	Enabled	
NMI Pin	PTA4/FTM0_CH1/NMI_b/EZP_CS...	PTA4/FTM0_CH1/NM
NMI Pin signal		
Reset		
Debug interface (JTAG)		
Flash memory organization		
Flexible memory controller	Enabled	
Access Protection		
Master 0 (ARM core code bus)		
Master 1 (ARMcore system bus)		
Master 2 (DMA / EzPort)		
Master 3 (Ethernet)		
Master 4 (USB OTG)		
Prefetch	Disabled	
Access protection	Read-Write	
Master 5 (SDHC)		
Cache settings		

Figure 3-31 Flexible memory controller settings

- Enable AIPS1 setting → chose USB OTG → configure as in the following figure:

Component Inspector - Cpu

Basic Advanced Expert

Properties Methods Events Build options Resources

Name	Value	Details
Internal peripherals		
NMI pin	Enabled	
Reset		
Debug interface (JTAG)		
Flash memory organization		
Flexible memory controller	Enabled	
Flash configuration field	Disabled	
MPU settings	Enabled	
AXBS settings	Enabled	
AIPSO settings	Enabled	
AIPS1 settings	Enabled	
Master privilege settings		
ARM core code bus		
ARMcore system bus		
DMA / EzPort		
Ethernet		
USB OTG		
Trusted for read	yes	
Trusted for write	yes	
Privilege level	no	

Figure 3-32 AIPS1 settings

- Enable LVD interrupt in the Power management controller field.

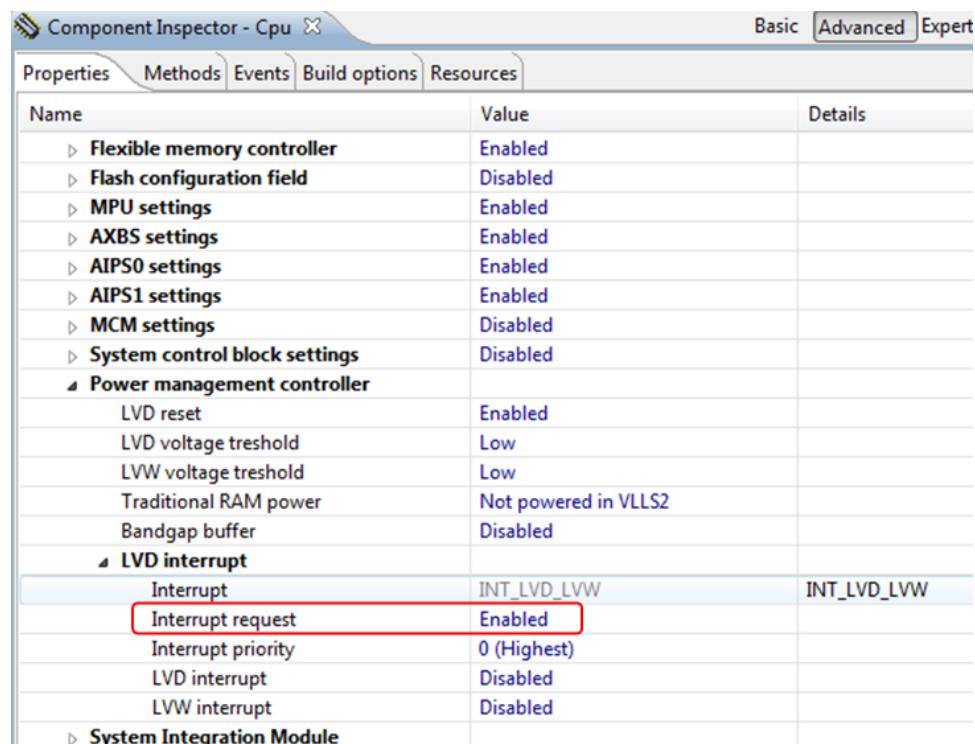


Figure 3-33 Power management controller settings

### Step3. Configure the USB\_HID\_CLASS component:

- HID Device Settings field. Refer to the USB HID mouse device example above.

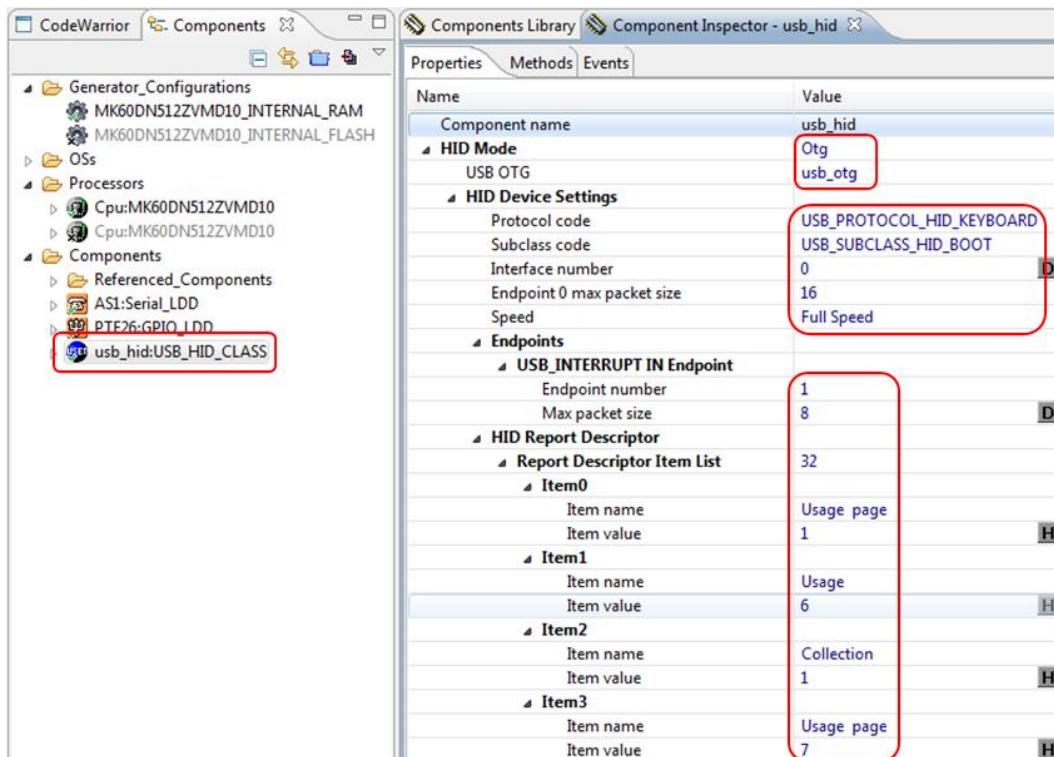


Figure 3-34 HID Device settings

- ❖ HID Host Settings field. Refer to the USB HID mouse example above.

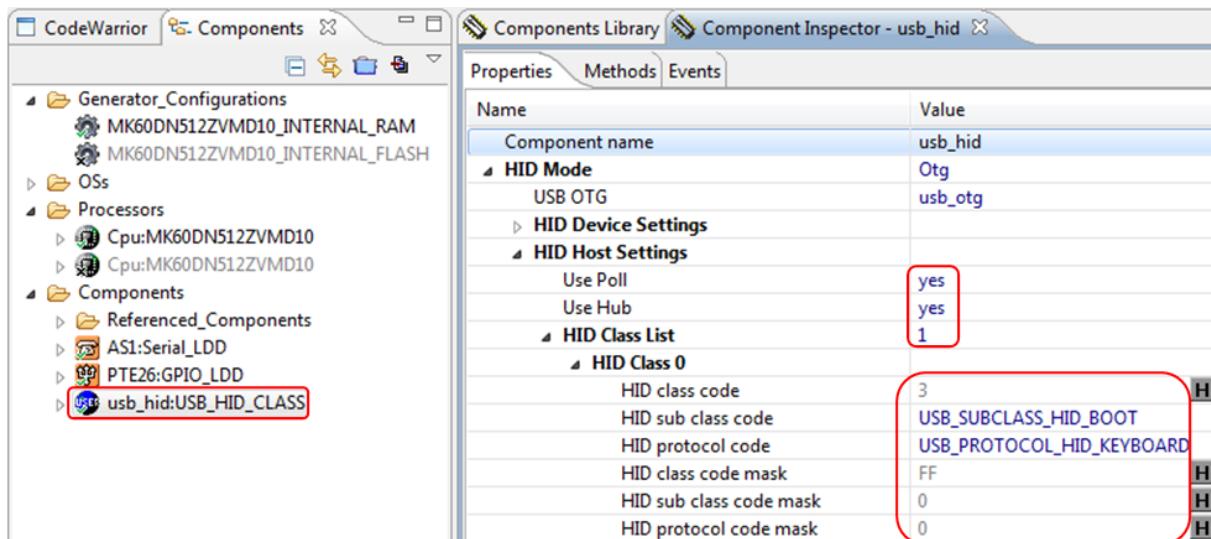


Figure 3-35 HID Host settings

#### Step4. Configure the USB\_OTG\_STACK component:

- OTG settings:
  - o Chose the GIPO\_LDD component to configure interrupt pin for MAX3353 as the following figure:

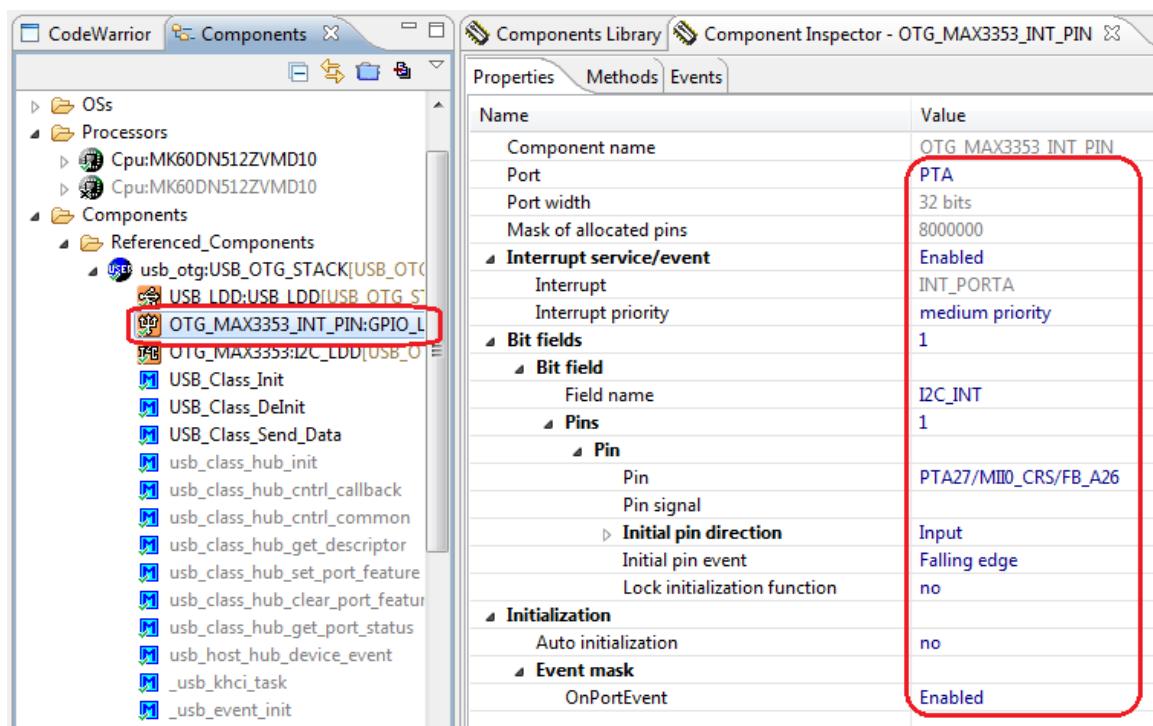


Figure 3-36 MAX3353 Interrupt pin settings

- Choose the I2C\_LDD component to configure the features for MAX3353

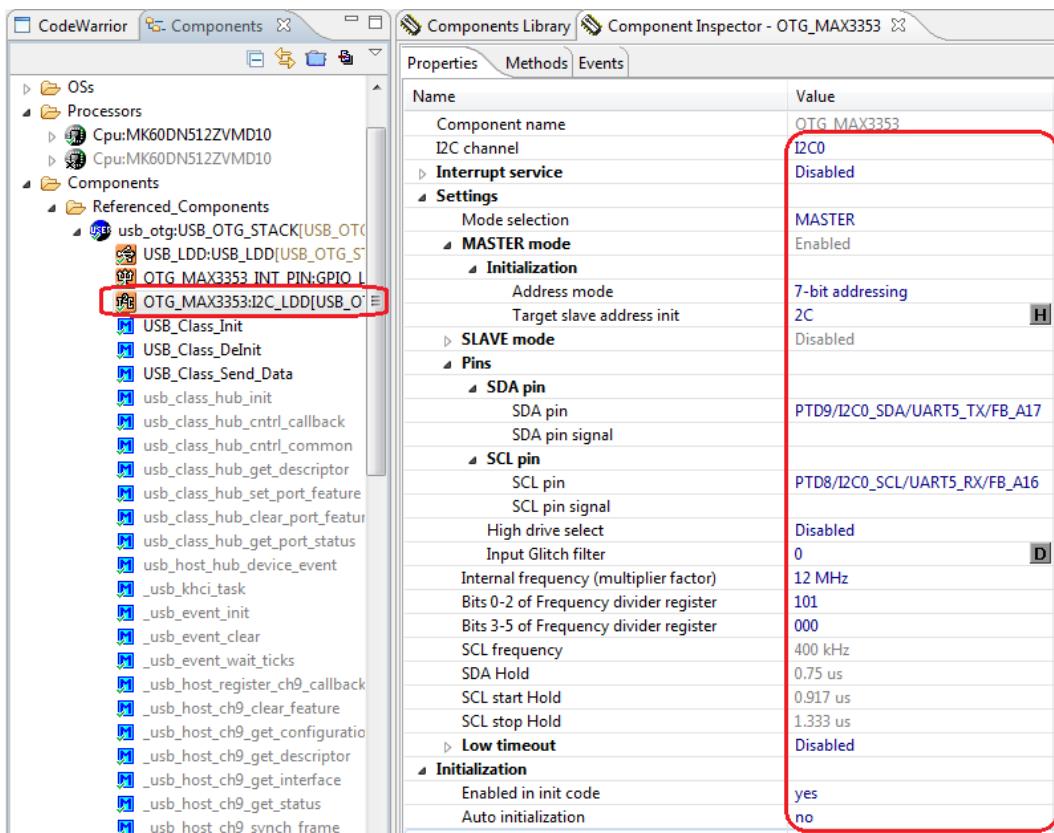


Figure 3-37 MAX3353 settings

- Device settings:
  - Configure general descriptor as in the following figure:

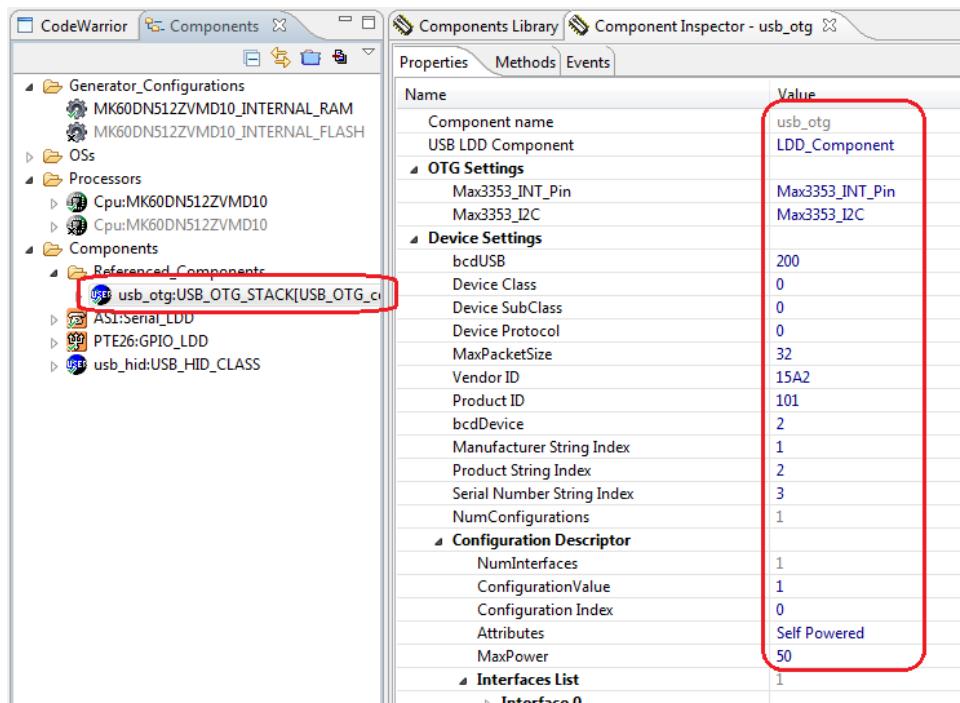


Figure 3-38 General descriptor settings

- Configure HID descriptor as shown below:

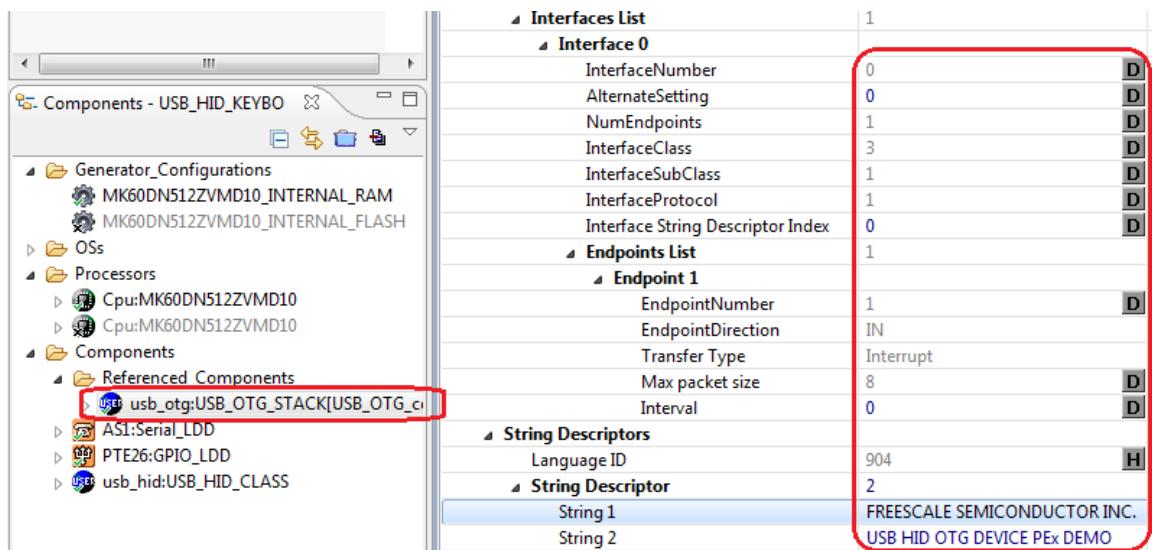


Figure 3-39 HID descriptor settings

- Host settings:
  - Setup the Host settings field as in the following figure:

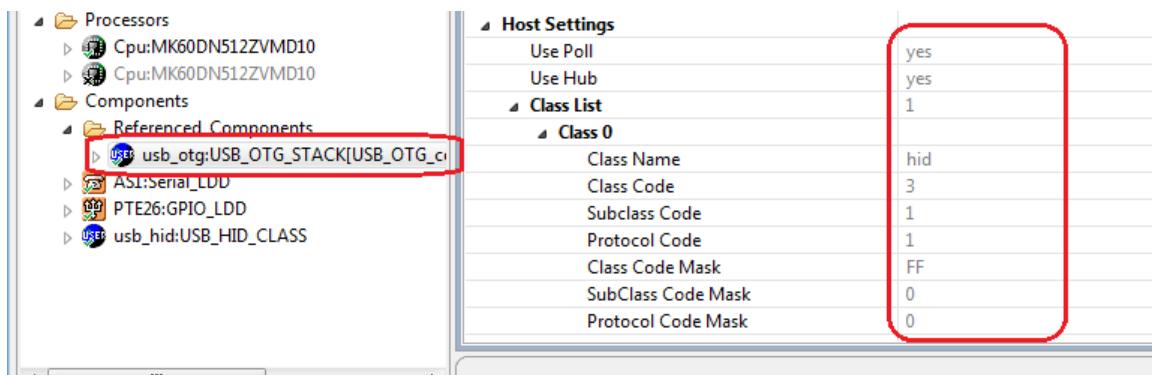


Figure 3-40 Host settings

**Step5.** Configure the USB\_LDD component as shown below. Refer to the USB HID mouse device and USB HID mouse host examples.

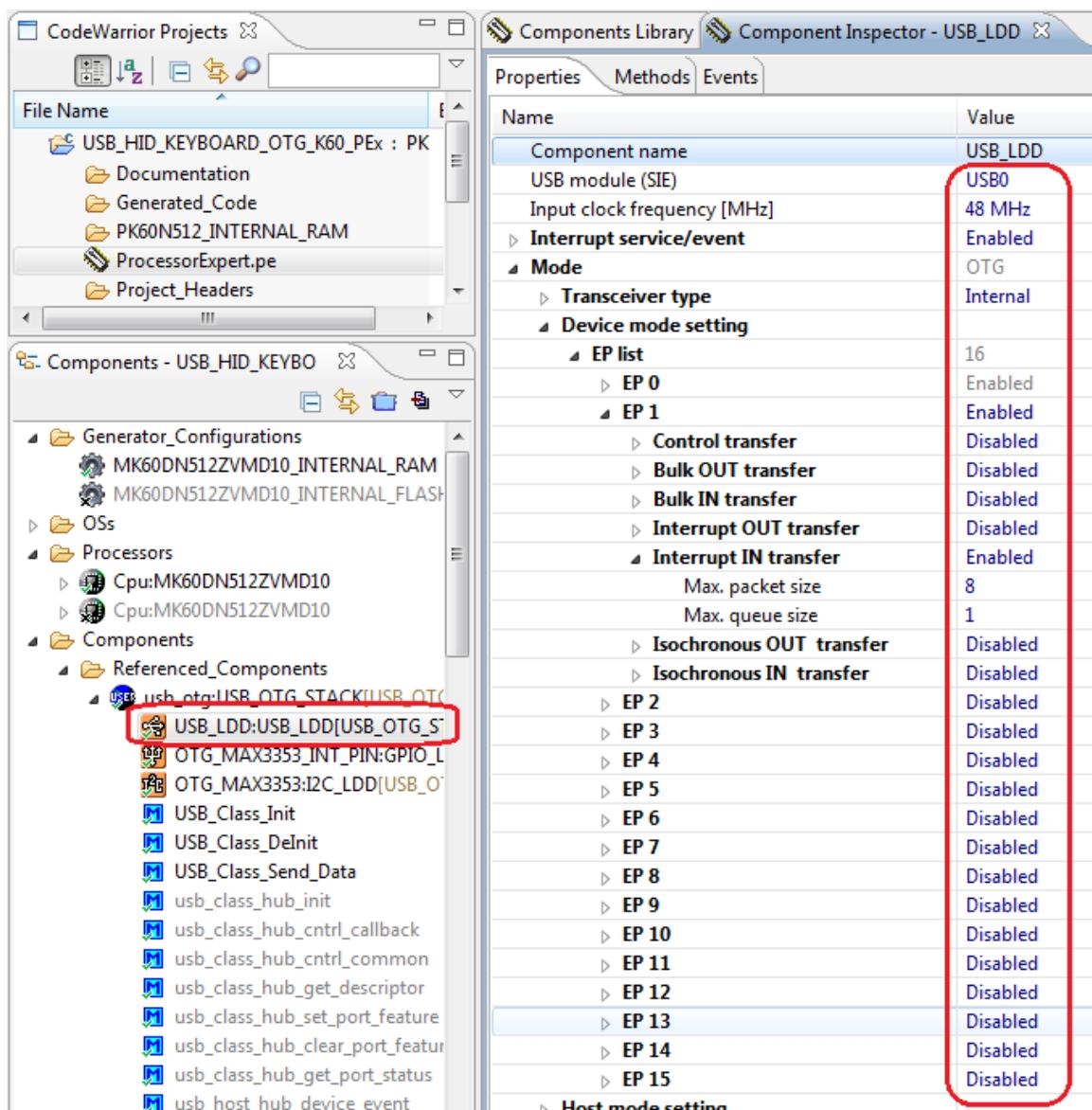


Figure 3-41 USB\_LDD component OTG mode settings

**Step6.** To run HID OTG component we have to add two components:

- Serial\_LDD component is used to show the status and operation of the USB HID keyboard OTG device.
- GPIO\_LDD component is used for the demo of pressing the button.

Choose the components from Components Library and add them to the project → configure the components as in the following figures:

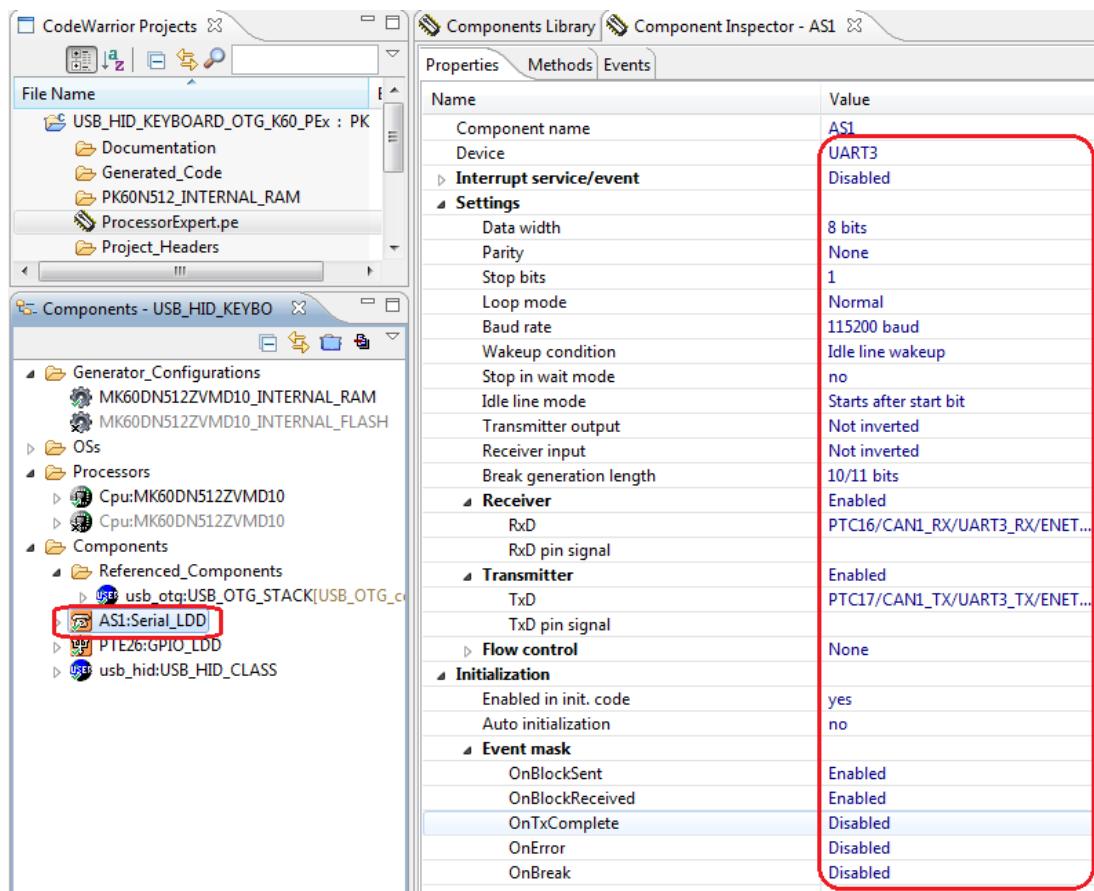


Figure 3-42 Serial\_LDD component settings

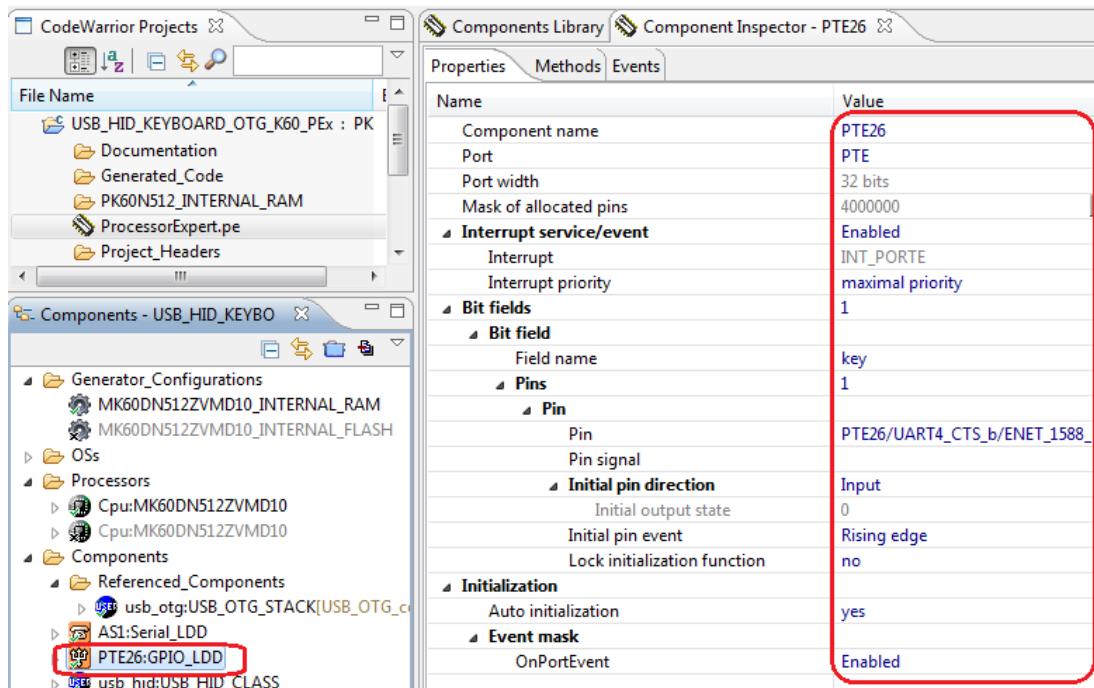


Figure 3-43 GPIO\_LDD component settings

**Step7.** Generate the USB\_HID\_CLASS source code. Refer to the USB HID mouse device example above.

### 3.3.4.2 Running the USB HID keyboard OTG example

This section describes the steps to run the USB HID keyboard OTG example.

#### Step1. Hardware settings:

- The computer uses one USB cable to supply power to the board and program USB HID keyboard OTG image to the flash. One COM cable is used to get events from the USB HID keyboard OTG device.



Figure 3-44 HID keyboard OTG demo setup

**Step2.** To run the USB HID keyboard OTG example, after the source has been generated, we must add the code to transfer between the USB generated code and the USB keyboard OTG application code.

- write code into "**device\_driver\_info.c**" file to register USB OTG event callback function as the following:

```
#include "hidkeyboard.h"

void usb_host_hid_keyboard_event
(
    /* [IN] pointer to device instance */
    usb_device_instance_handle dev_handle,
    /* [IN] pointer to interface descriptor */
    usb_interface_descriptor_handle intf_handle,
    /* [IN] code number for event causing callback */
    uint_32 event_code
)
{
    /* Write code here ... */
    usb_host_hid_keyboard_app_event(dev_handle,intf_handle,event_code);
}
```

**Step3.** To ensure that application runs correctly, the HyperTerminal is used to get events from the USB HID keyboard OTG device. Refer to the USB HID mouse host example to configure this software.

**Step4.** Build and load the image of the USB HID keyboard OTG application to the Kinetis K60 tower board. The HyperTerminal shows the USB HID keyboard OTG status as in the following figure:

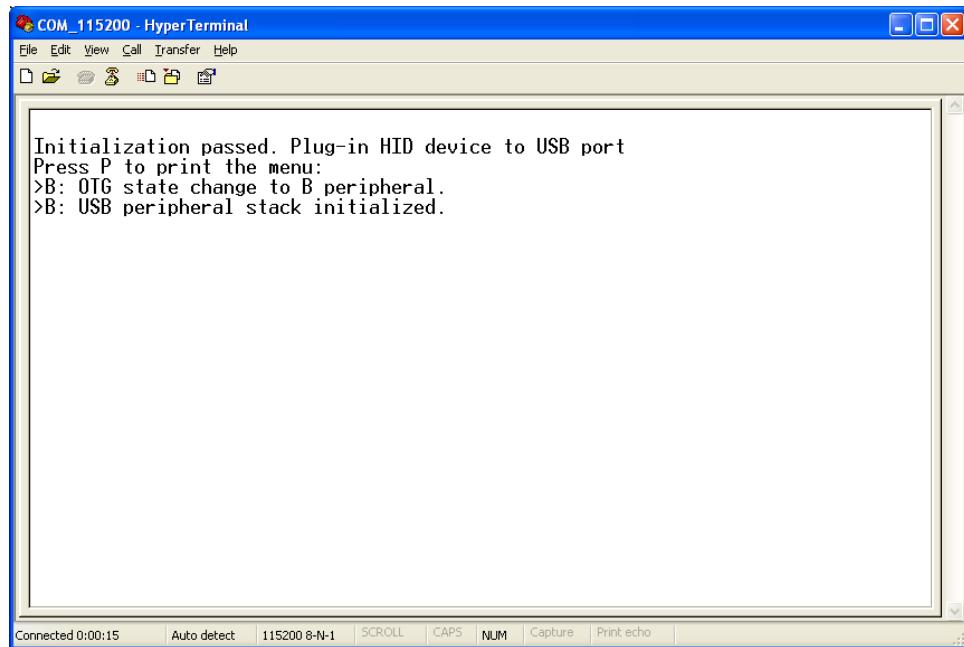


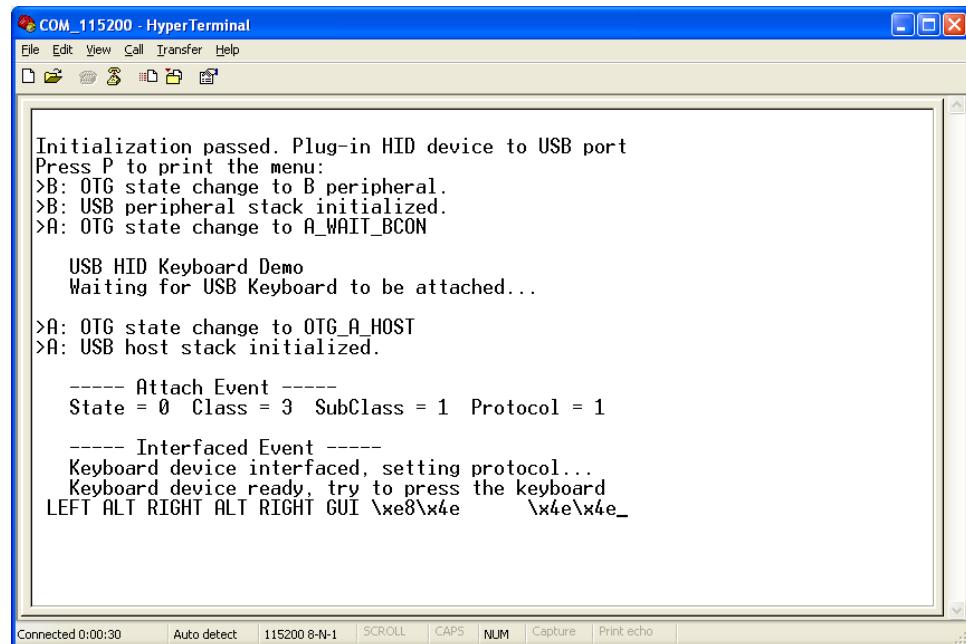
Figure 3-45 USB HID keyboard OTG initializing

**Step5.** Connect the USB HID keyboard OTG device with the PC by one USB cable. Press switch 2, the transfer data is captured as the following figure:

*Untitled - Total Phase Data Center								
Sp	Index	ms.ms.us	Len	Err	Dev	Ep	Record	Data
FS	162	0:03.100.678	18 B		02	00	[+] Get Device Descriptor	Index=0 Length=18
FS	175	0:03.103.666	3.00 ms				[4 SOF]	[Frames: 910 - 913]
FS	176	0:03.104.678	9 B		02	00	[+] Get Configuration Descriptor	Index=0 Length=9
FS	189	0:03.107.667	4.00 ms				[5 SOF]	[Frames: 914 - 918]
FS	190	0:03.108.678	39 B		02	00	[+] Get Configuration Descriptor	Index=0 Length=39
FS	207	0:03.112.667	2.00 ms				[3 SOF]	[Frames: 919 - 921]
FS	208	0:03.113.679	0 B		02	00	[+] Set Configuration	Configuration=1
FS	217	0:03.115.668	18.0 ms				[19 SOF]	[Frames: 922 - 940]
FS	218	0:03.132.682	0 B		02	00	[+] Set Idle	Duration=Indefinite Report=0
FS	227	0:03.134.670	5.00 ms				[6 SOF]	[Frames: 941 - 946]
FS	228	0:03.135.683	63 B		02	00	[+] Get Report Descriptor	Index=0 Length=127
FS	245	0:03.140.671	1.01 s				[1015 SOF]	[Frames: 947 - 1961]
FS	246	0:04.152.819	1 B		02	00	[+] Set Output Report	Length=1
FS	259	0:02.972.334	1.99 s	01	03	[+]	[193467 ORPHANED]	[Periodic Timeout]
FS	261	0:02.972.652	1.99 s	01	01	[+]	[2000 ORPHANED]	[Periodic Timeout]
FS	263	0:03.145.675	1.99 s	02	01		[250 IN-NAK]	[Periodic Timeout]
FS	264	0:04.155.808	1.67 s				[1671 SOF]	[Frames: 1962 - 1584]
FS	265	0:05.145.944	8 B	02	01	[+]	[Input Report]	Keys=[PgUp]
FS	271	0:05.827.033	7.00 ms				[8 SOF]	[Frames: 1585 - 1592]
FS	272	0:05.834.037	8 B	02	01	[+]	[Input Report]	
FS	277	0:04.972.328	1.99 s	01	03	[+]	[193144 ORPHANED]	[Periodic Timeout]
FS	279	0:04.972.921	1.99 s	01	01	[+]	[2000 ORPHANED]	[Periodic Timeout]
FS	281	0:05.835.034	1.99 s				[2000 SOF]	[Frames: 1593 - 1544] [Periodic Timeout]
FS	282	0:05.842.038	1.99 s	02	01		[250 IN-NAK]	[Periodic Timeout]
FS	292	0:05.872.221	1.00 s	01	03	[+]	[193471 ORPHANED]	[Periodic Timeout]

Figure 3-46 USB HID keyboard OTG operation – Device-B Peripheral

**Step6.** Unplug the USB HID keyboard OTG device and then connect the USB keyboard device to it. Press keyboard's button, the HyperTerminal shows the status changing of the USB HID keyboard OTG device.



The screenshot shows a window titled "COM\_115200 - HyperTerminal". The menu bar includes File, Edit, View, Call, Transfer, Help. The toolbar has icons for Cut, Copy, Paste, Find, Replace, etc. The main text area displays the following log output:

```
Initialization passed. Plug-in HID device to USB port
Press P to print the menu:
>B: OTG state change to B peripheral.
>B: USB peripheral stack initialized.
>A: OTG state change to A_WAIT_BCON

USB HID Keyboard Demo
Waiting for USB Keyboard to be attached...

>A: OTG state change to OTG_A_HOST
>A: USB host stack initialized.

----- Attach Event -----
State = 0 Class = 3 SubClass = 1 Protocol = 1

----- Interfaced Event -----
Keyboard device interfaced, setting protocol...
Keyboard device ready, try to press the keyboard
LEFT ALT RIGHT ALT RIGHT GUI \xe8\x4e \x4e
```

The status bar at the bottom shows "Connected 0:00:30" and various control buttons: Auto detect, 115200 8-N-1, SCROLL, CAPS, NUM, Capture, Print echo.

Figure 3-47 USB HID keyboard OTG operation – Device-A Host

## 4 USB\_CDC\_CLASS

This section describes how to run USB CDC device and USB CDC host examples with the USB\_CDC\_CLASS component in the Processor Expert on Code Warrior IDE.

### 4.1 Architecture overview

The USB\_CDC\_CLASS component makes use of the USB\_DEVICE\_STACK and USB\_HOST\_STACK existing components to provide CDC APIs for user.

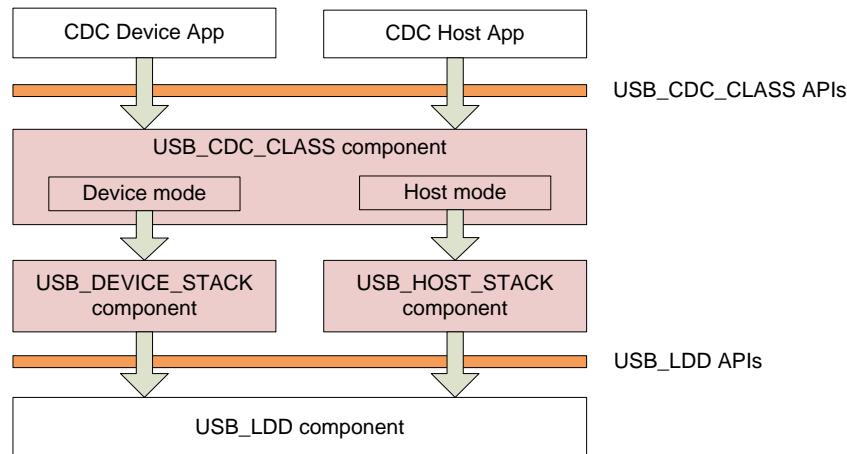


Figure 4-1 Architecture overview

### 4.2 Folder structure

#### 4.2.1 USB CDC component package

The delivered package includes items as in the following figure:

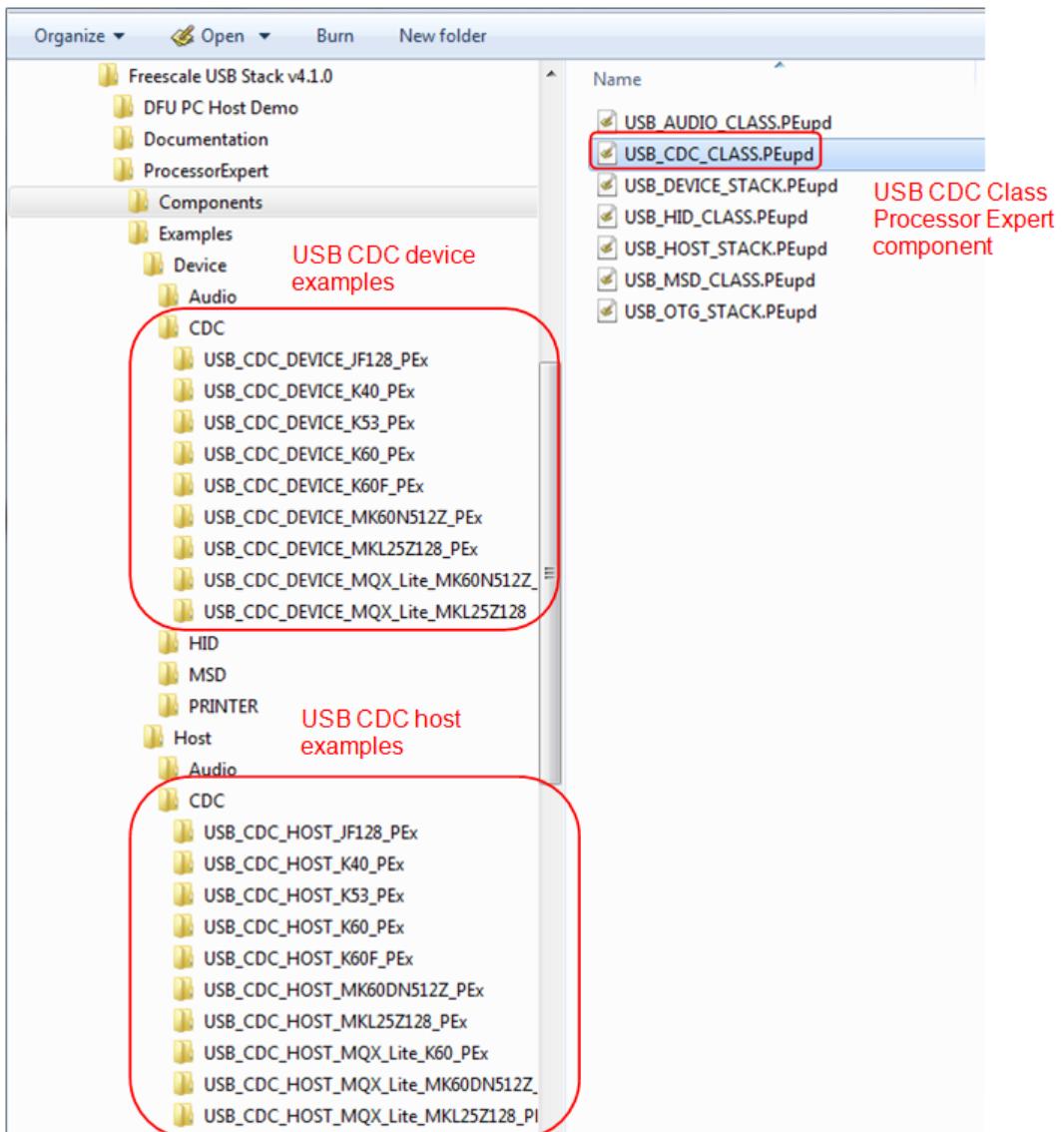


Figure 4-2 Deliver package directory structure

- USB\_HOST\_STACK component: This component will encapsulate all the common USB host stack functionality.
- USB\_CDC\_CLASS component: This component will encapsulate the Communication Device Class functionality.

#### 4.2.2 CDC device example structure

The USB CDC device example has the structure as in the following figure:

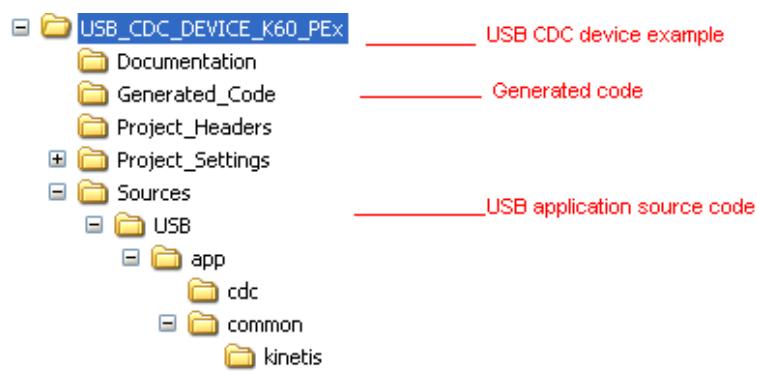


Figure 4-3 CDC device directory structure

#### 4.2.3 CDC host example structure

The USB CDC host example has the structure as in the following figure:

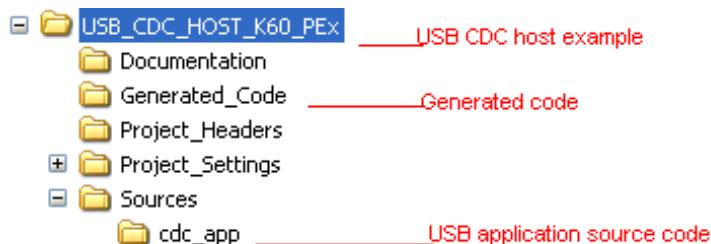


Figure 4-4 CDC host directory structure

### 4.3 USB CDC example

This section describes the functionality of the USB CDC example on the Tower K60 board.

#### 4.3.1 Importing component

This section describes the steps to import the USB\_CDC\_CLASS component that is used to run the USB CDC examples.

**Step1.** On CodeWarrior Development Studio, chose Processor Expert → Import Package

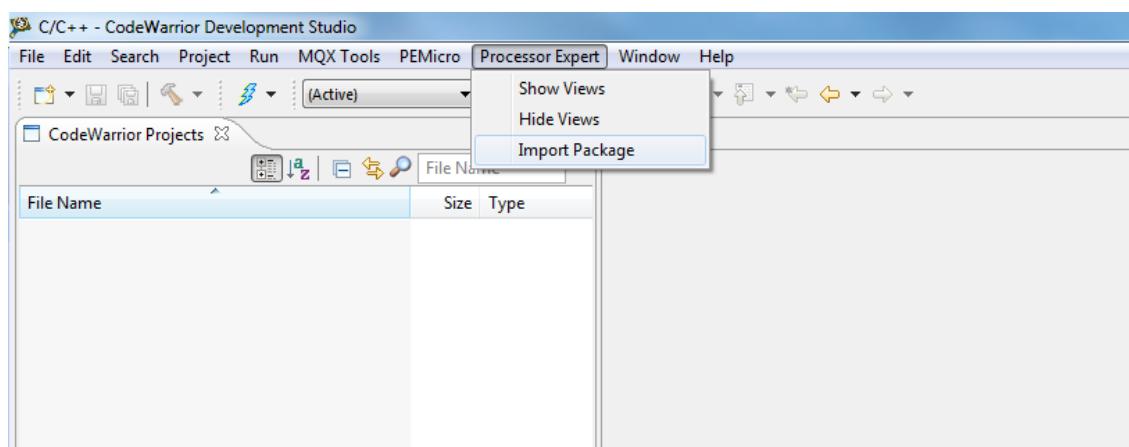


Figure 4-5 Processor Expert – Import Package

**Step2.** A browser window appears. Navigate to the folder that contains the USB\_CDC\_CLASS component, chose it and click open.

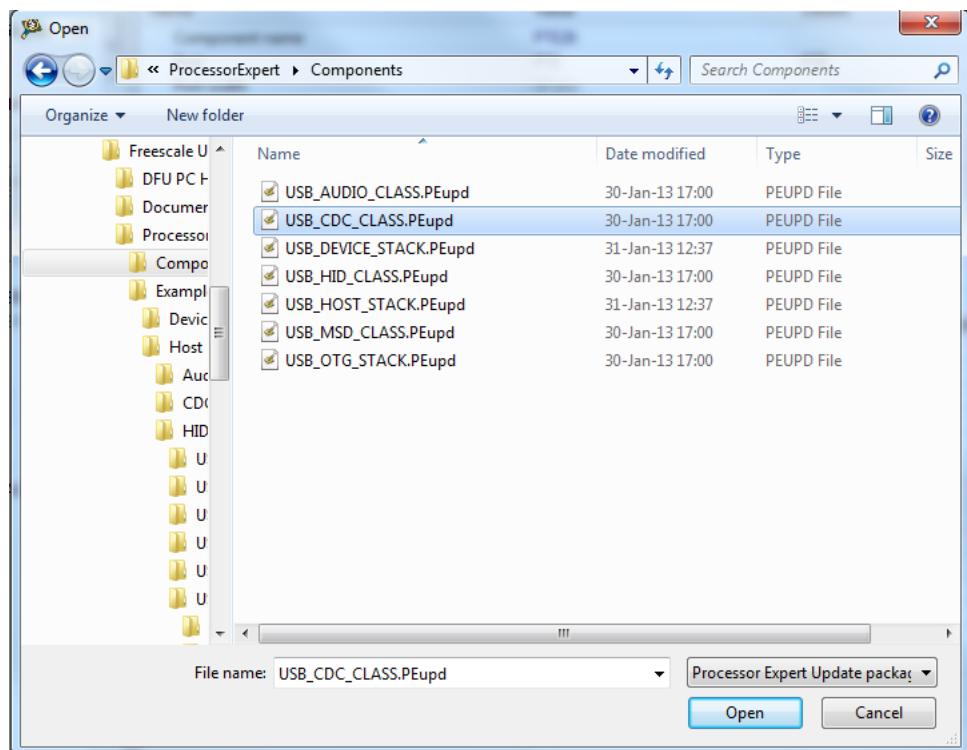


Figure 4-6 Delivered package navigation

**Step3.** An Import Complete dialog box appears. Click OK for confirmation.

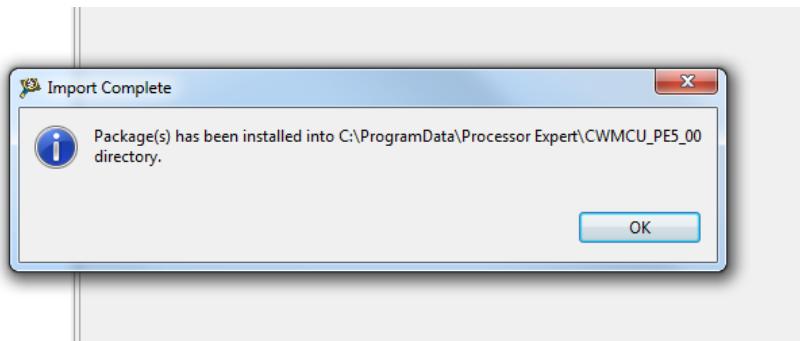


Figure 4-7 Import USB components

**Step4.** Implement the same steps to import the USB\_HOST\_STACK and USB\_DEVICE\_STACK components.

## 4.3.2 CDC device example

### 4.3.2.1 Generating CDC device source code

This section describes the steps to generate USB CDC source code to run USB CDC device example.

**Step1.** Open USB\_CDC\_DEVICE\_K60\_PEx example with CW10.3 → from the CW10.3 toolbar, select Processor Expert → Select Show Views to open Components Library → Select USB\_CDC\_CLASS component and add to project.

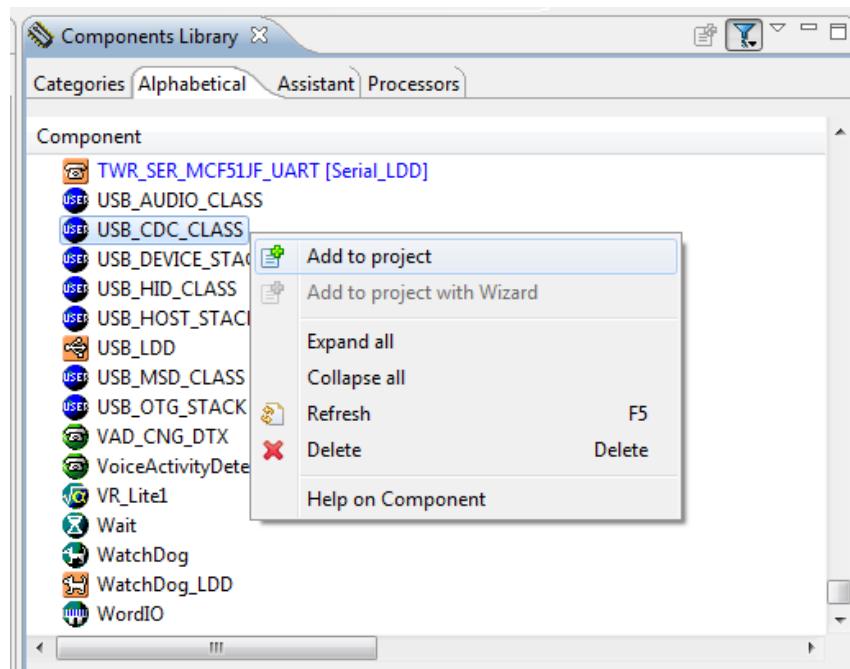


Figure 4-8 Component show views

**Step2.** Configure CPU clock to run USB module

- Select CPU target from Project Panel window to setup clock source as the following:
  - o Target: Cpu:MK60N512VMD100
  - o Clock source: External reference clock 50MHz
  - o MCG mode: PEE
  - o MCG output: 48 MHz
  - o PLL output: 48MHz
  - o Core clock: 48MHz
  - o Bus clock: 48MHz
  - o External bus clock: 24MHz
  - o Flash clock: 24MHz

\*Component Inspector - Cpu

Properties Methods Events Build options Resources		
Name	Value	Details
CPU type	MK60DN512ZVMD10	
<b>Clock settings</b>		
<b>Internal oscillator</b>		
Slow internal reference clock [kHz]	32.768	32.768 kHz
Fast internal reference clock [MHz]	4.0	4 MHz
<b>RTC oscillator</b>	Disabled	
<b>System oscillator</b>	Enabled	
<b>Clock source</b>	External reference clock	
Clock frequency [MHz]	50.0	50 MHz
<b>Clock source settings</b>	1	
<b>Clock source setting 0</b>		
<b>MCG settings</b>		
MCG mode	PEE	
MCG output [MHz]	48.0	48 MHz
MCG external ref. clock source	System oscillator	
MCG external ref. clock [MHz]	50.0	50 MHz
<b>FLL settings</b>		
<b>PLL settings</b>		
<b>PLL module</b>	Enabled	
PLL output [MHz]	48.0	48 MHz
Initialization priority	minimal priority	
Watchdog disable	yes	
<b>CPU interrupts/resets</b>		
<b>External Bus</b>	Disabled	
<b>Clock configurations</b>	1	
<b>Clock configuration 0</b>		
<b>Clock source setting</b>	configuration 0	
MCG mode	PEE	
<b>System clocks</b>		
Core clock	48.0	48 MHz
Bus clock	48.0	48 MHz
External bus clock	24.0	24 MHz
Flash clock	24.0	24 MHz

Figure 4-9 Clock source settings

- Chose advanced tab

\*Component Inspector - Cpu

Properties Methods Events Build options Resources		
Name	Value	Details
Component name	Cpu	
CPU type	MK60DN512ZVMD10	

Figure 4-10 Open Advanced Tab

- PLL/FLL clock selection:

- o PLL clock

Bus clock	48.0	48 MHz
External clock prescaler	Auto select	2
External bus clock	24.0	24 MHz
Flash clock prescaler	Auto select	2
Flash clock	24.0	24 MHz
<b>PLL/FLL clock selection</b>	PLL clock	
Clock frequency [MHz]	48.0	48 MHz
<b>USB clock settings</b>		
USB clock divider	Auto select	1
USB clock multiply	Auto select	1
USB clock	48.0	48 MHz

Figure 4-11 Clock configuration

### **Step3.** Setup the USB\_CDC\_CLASS component

- Click on the USB\_CDC\_CLASS component to select this target.
- Show in the Component Inspector window → Select CDC Mode field → Select Device mode.

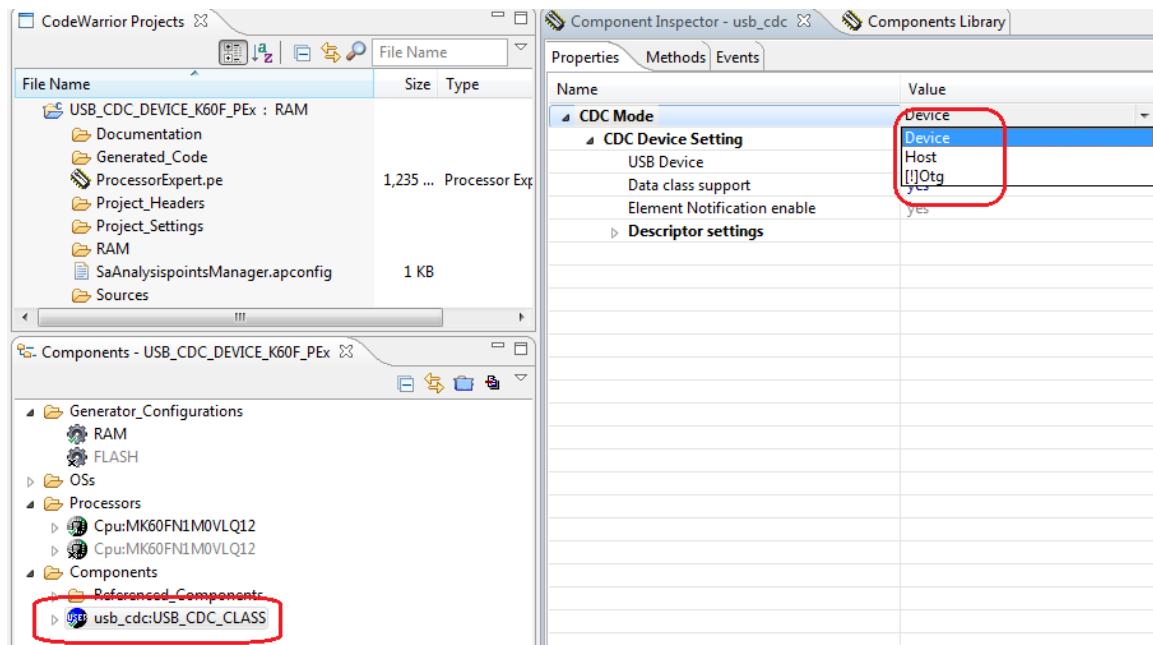


Figure 4-12 CDC device mode settings

- Fill input parameters of the USB\_CDC\_CLASS component as in the following figures:

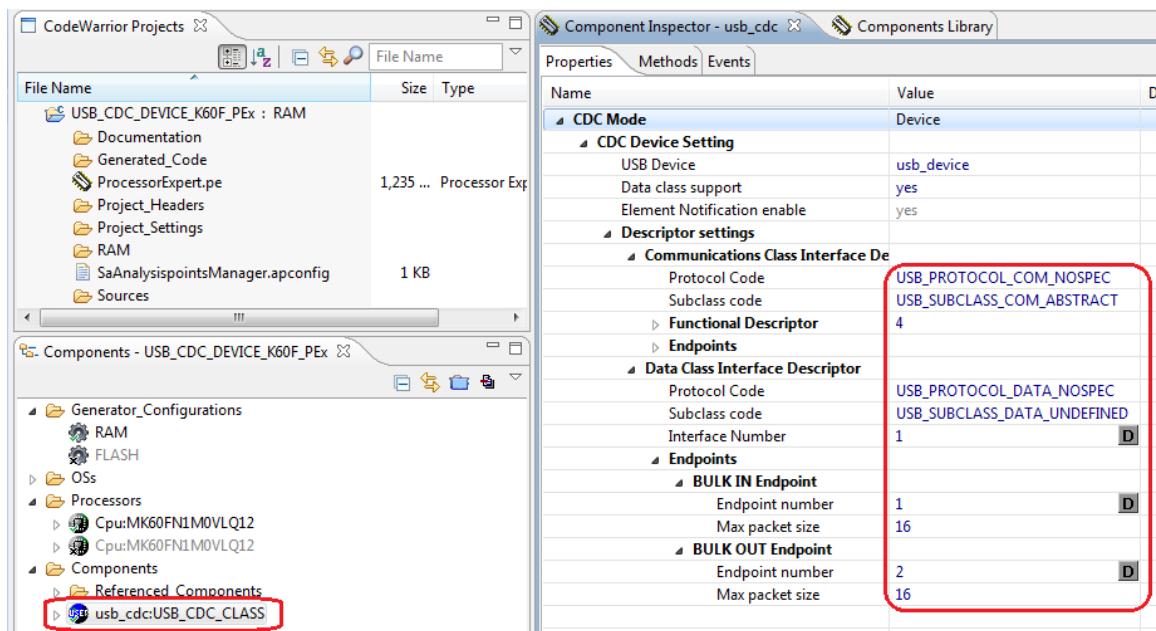


Figure 4-13 CDC Device settings

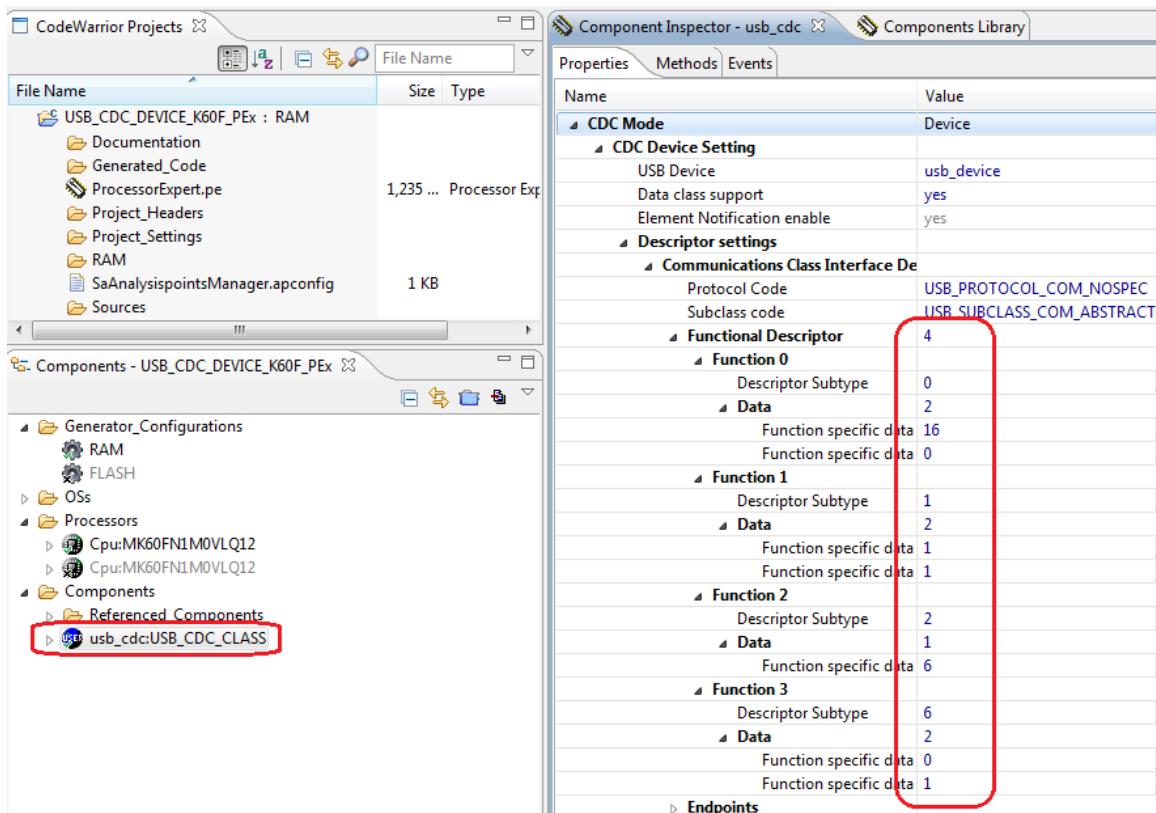


Figure 4-14 Function Descriptor settings

#### Step4. Setup the USB\_DEVICE\_STACK component:

- Click on the USB\_DEVICE\_STACK component to select this target.
- Fill input parameters as in the following figures:
  - o Device Descriptor

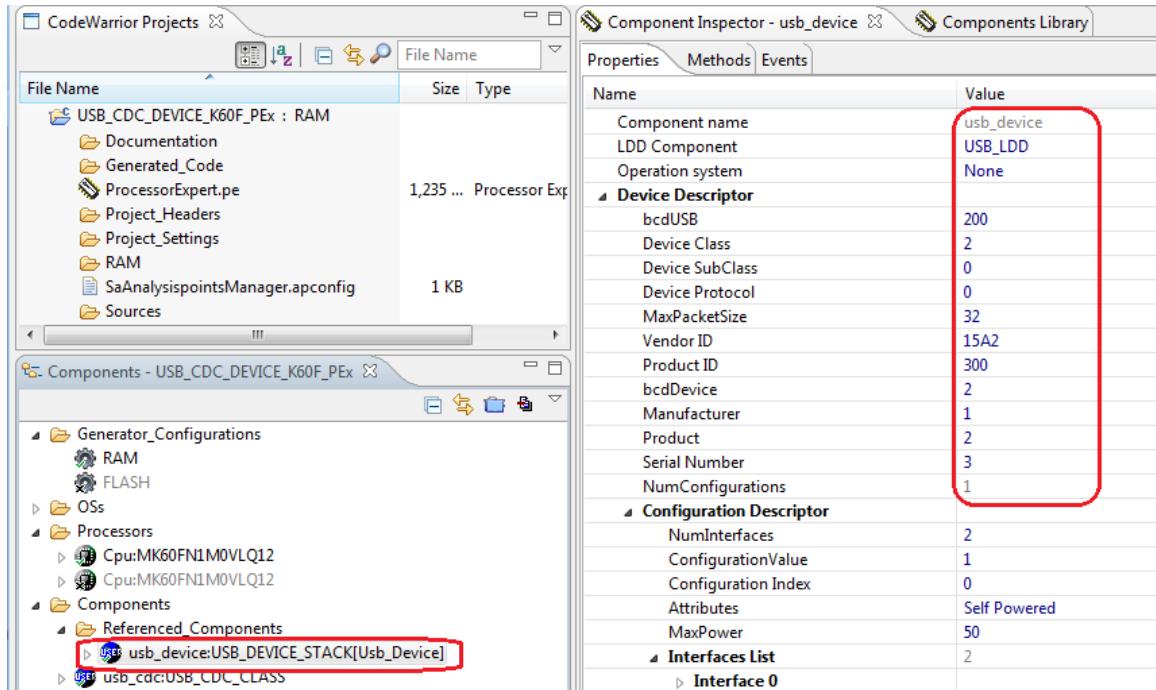


Figure 4-15 Device Descriptor settings

- Configuration Descriptor: This group inherited the USB\_CDC\_CLASS component.

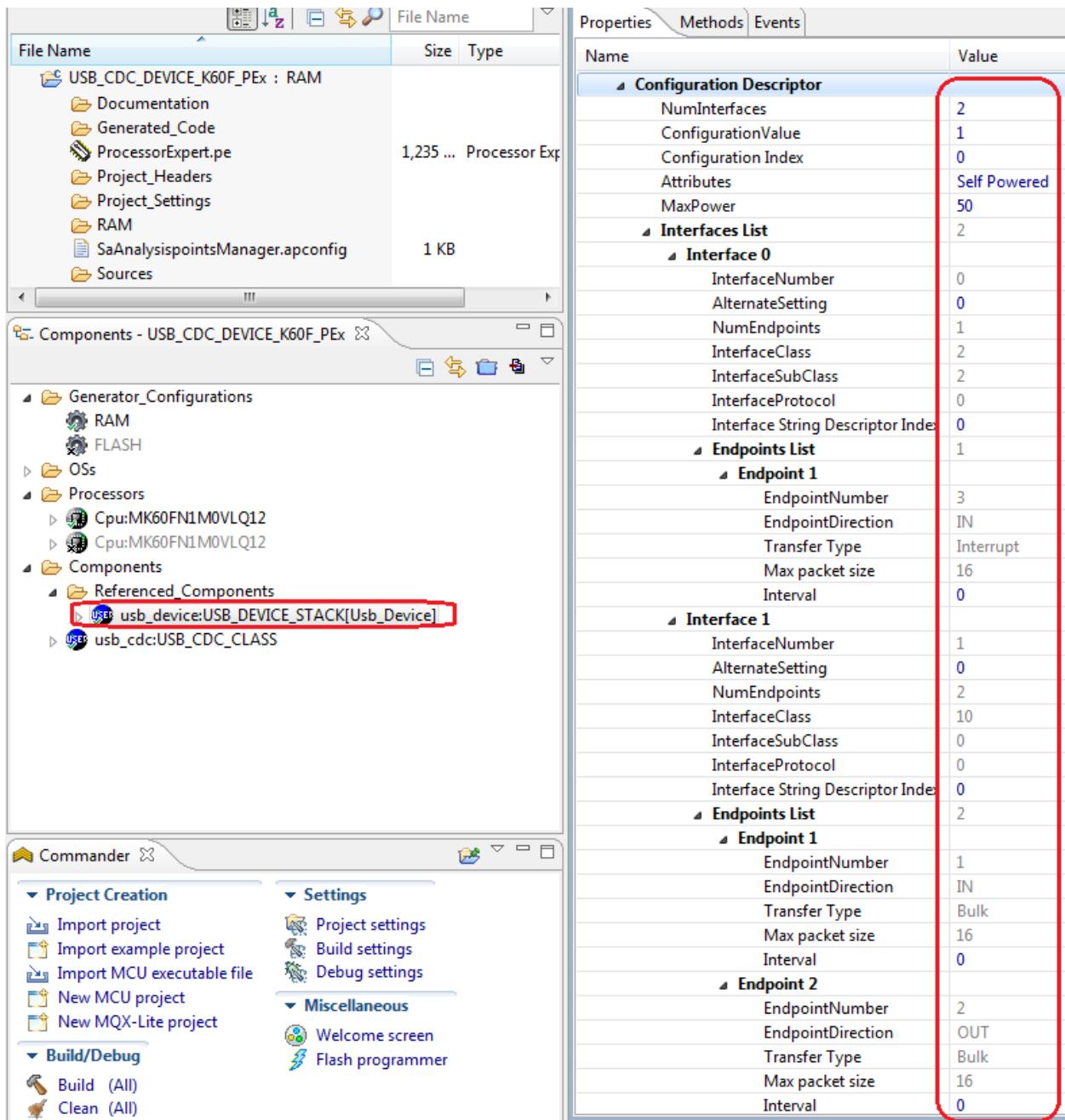


Figure 4-16 Configuration Descriptor settings

- String Descriptors: Click plus button to add string field → Fill string values

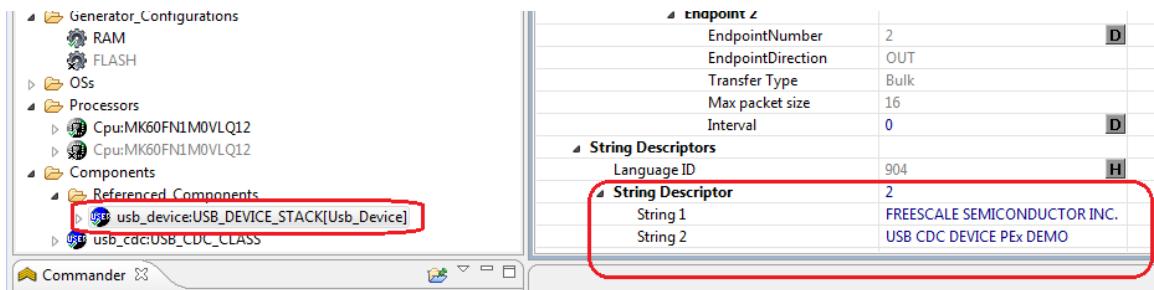


Figure 4-17 String Descriptor settings

### Step5. Setup the USB\_LDD component

- Click on the USB\_LDD component to select this target.
- Setup input parameters corresponding to USB\_CDC\_CLASS component.

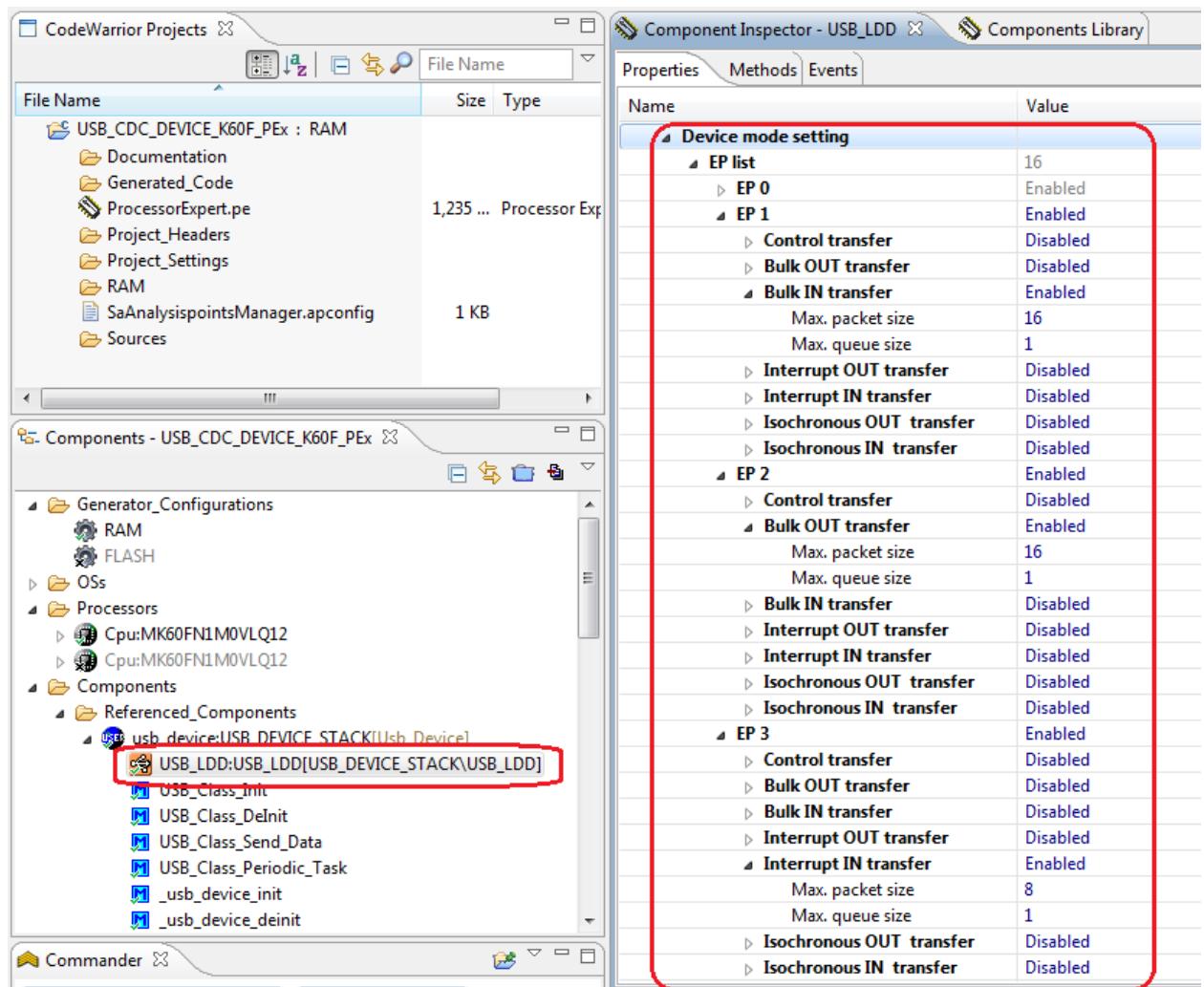


Figure 4-18 USB\_LDD component settings

## **Step6.** Generate Processor Expert code:

After setting the components, select ProcessorExpert.pe in the project folder → Right click → Choose Generate Processor Expert Code field to generate source code.

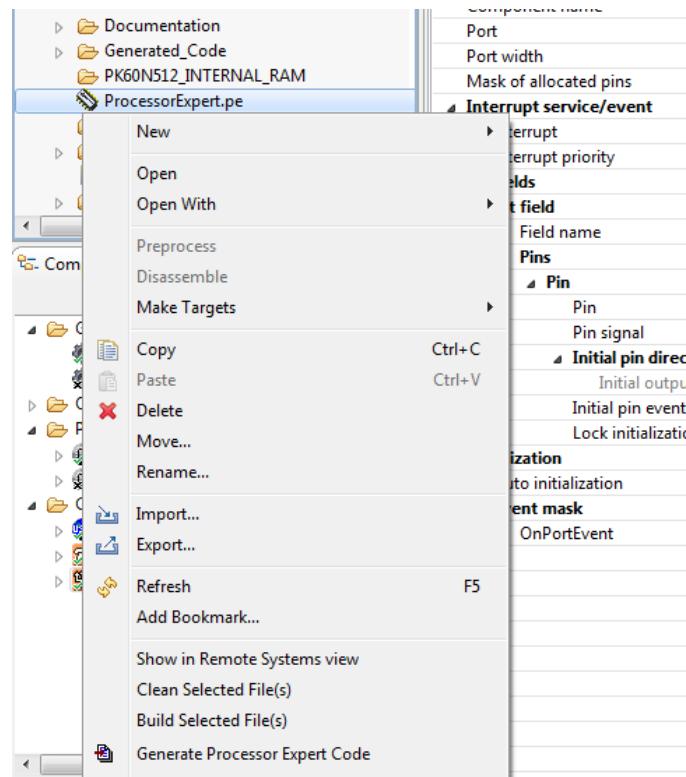


Figure 4-19 Generate Processor Expert code

### **4.3.2.2 Running CDC device example**

Implement the following steps to run the CDC device example:

**Step1.** Build and load the image of CDC device application to the Kinetis K60 Tower Board.

**Step2.** After the image has been loaded successfully, connect the USB port of the board to a USB port of the computer. You can see the device entry on device manager of Windows



Figure 4-20 Device entry

**Step3.** To ensure that application runs correctly, the HyperTerminal is used to transact with the CDC device. These are the steps to configure HyperTerminal.

- Open HyperTerminal application as the following figure:

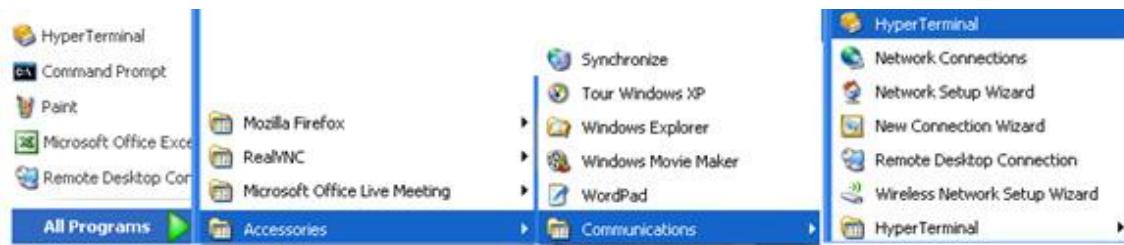


Figure 4-21 Launch HyperTerminal application

- The HyperTerminal opens as in the figure below. Enter the name of the connection and click on the OK button.

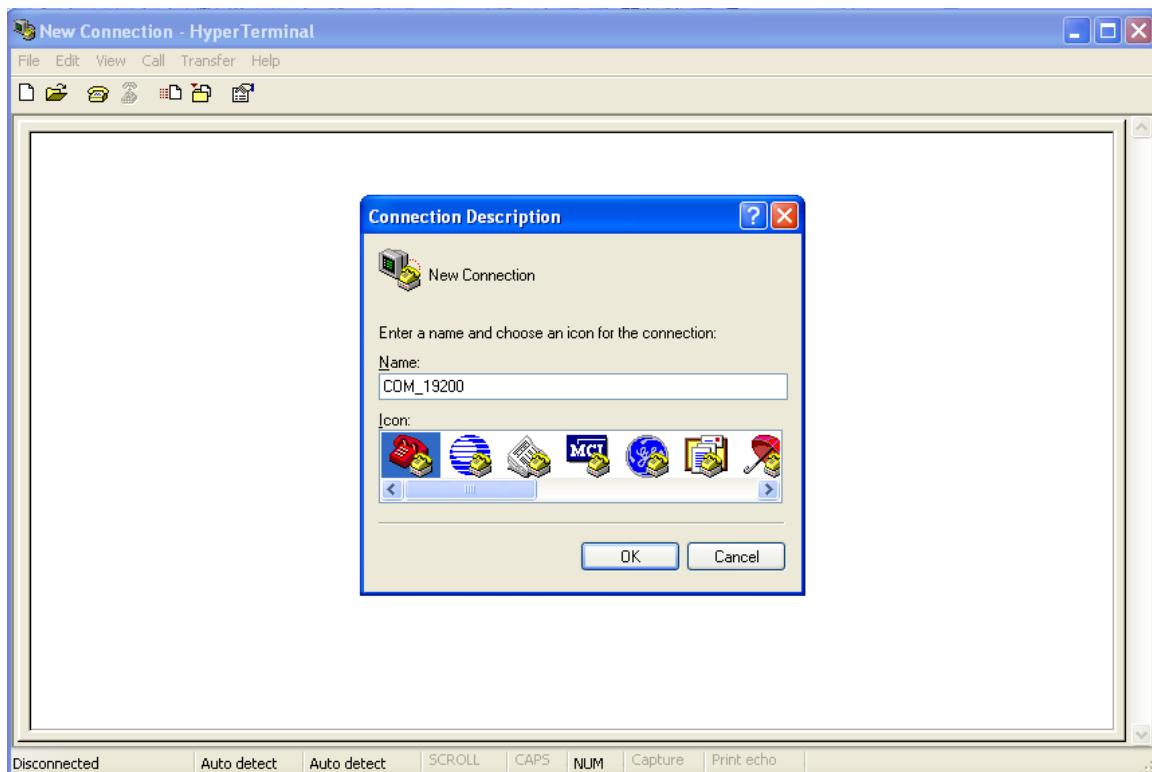


Figure 4-22 HyperTerminal startup

- The window shown in the following figure appears. Select the COM port.

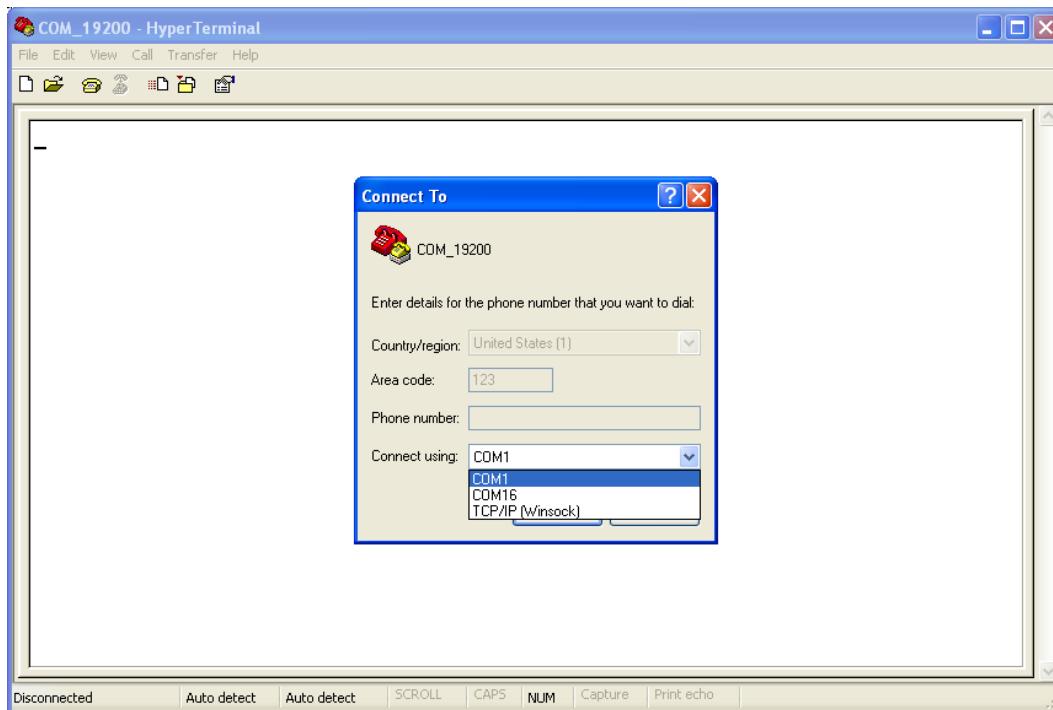


Figure 4-23 Connect using COM port

- Configure the COM port baud rate and other properties as in the figure below.

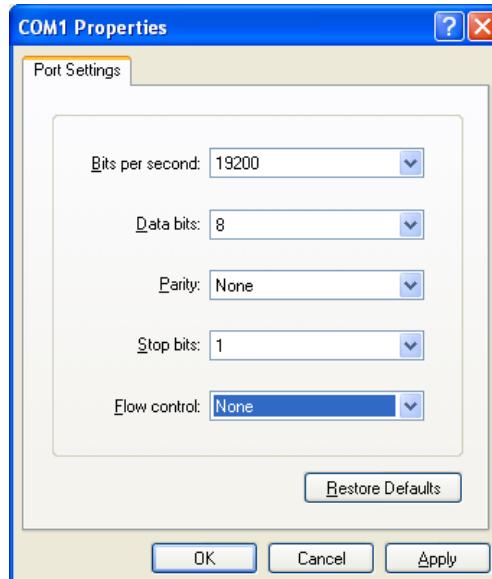


Figure 4-24 COM properties

- The HyperTerminal is now configured as shown in the figure below:

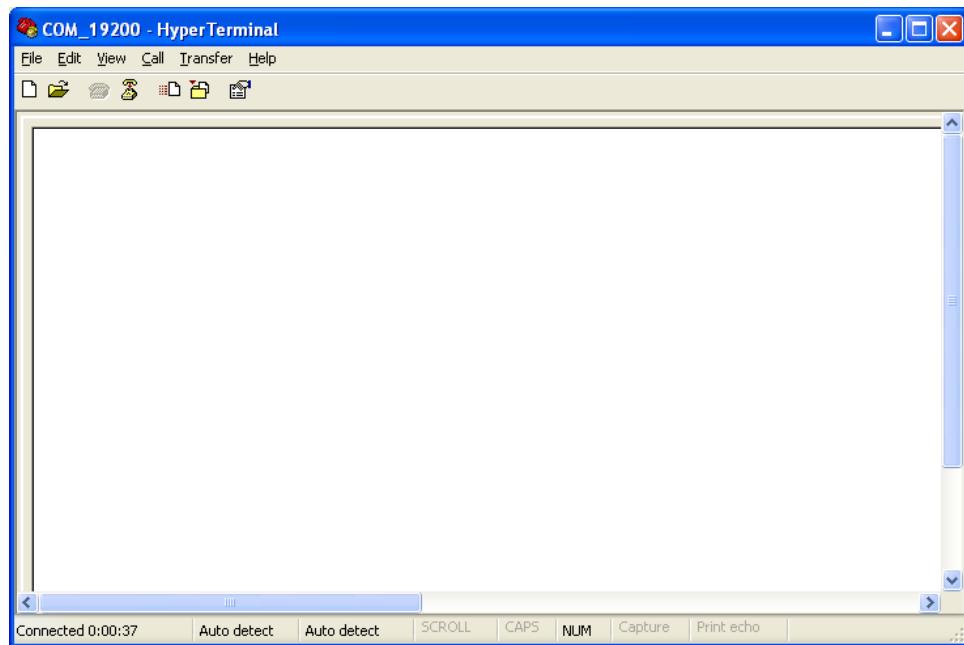


Figure 4-25 Hyper Terminal

**Step4.** Type strings to the HyperTerminal window to verify the CDC device operating.

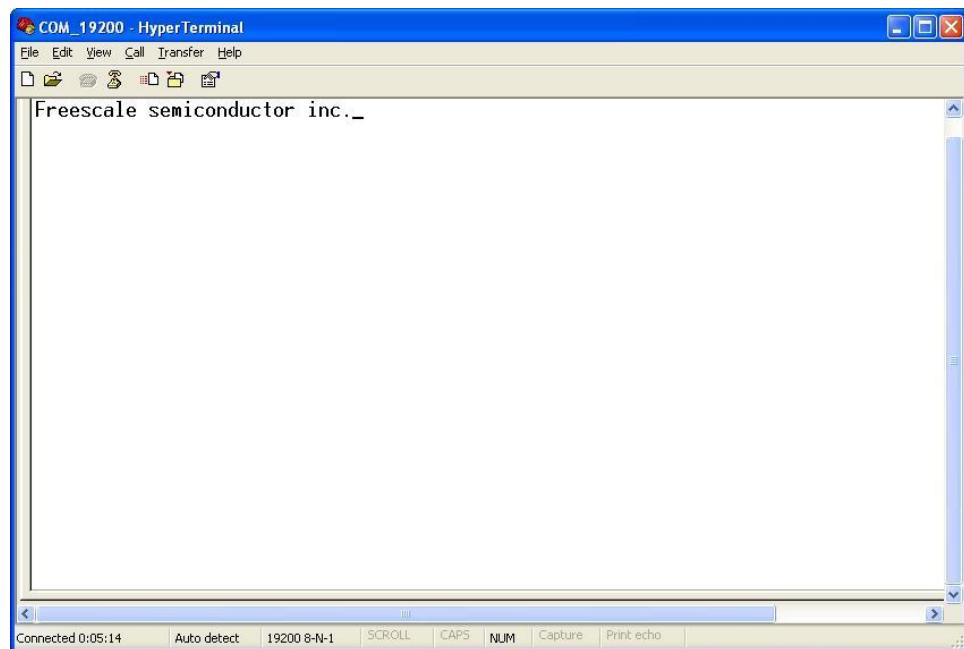


Figure 4-26 CDC device operating

### 4.3.3 CDC host example

#### 4.3.3.1 Generating CDC host source code

This section describes the steps to generate USB CDC source code to run USB CDC host example.

**Step1.** Open the USB\_CDC\_HOST\_K60\_PEx example with CW10.3 → from CW10.3 toolbar → Select Processor Expert → Select Show Views to open Components Library. Select USB\_CDC\_CLASS component and add to project, same as the CDC device example above.

**Step2.** Implement same as step1 to add the TWR\_SER\_K60\_UART component into the project.

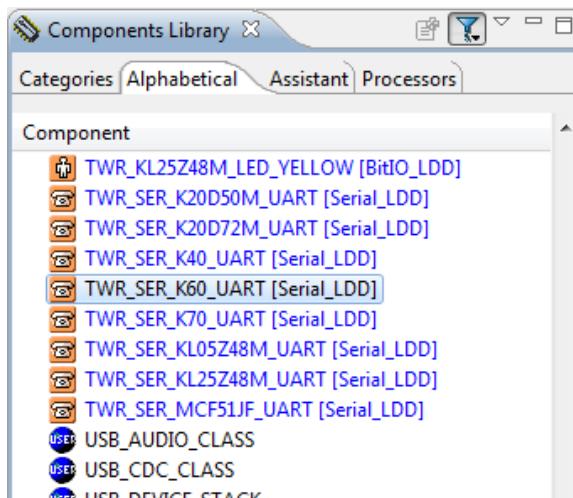


Figure 4-27 Components Library

**Step3.** Setup the CPU clock to run USB module same as the CDC device example above.

**Step4.** Setup the USB\_CDC\_CLASS component:

- Click on the USB\_CDC\_CLASS component to select this target.
- Show in the Component Inspector window → Select CDC Mode field → Select Host mode.

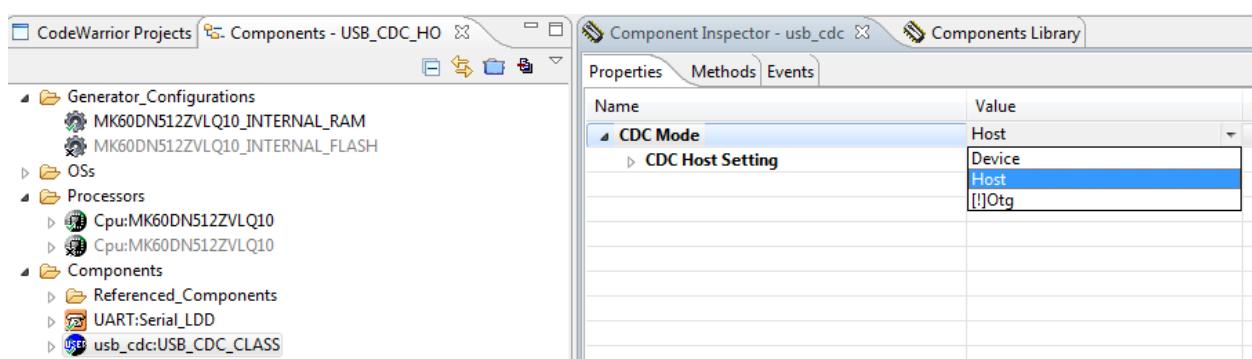


Figure 4-28 CDC host mode setting

- Fill input parameters into CDC class list settings:

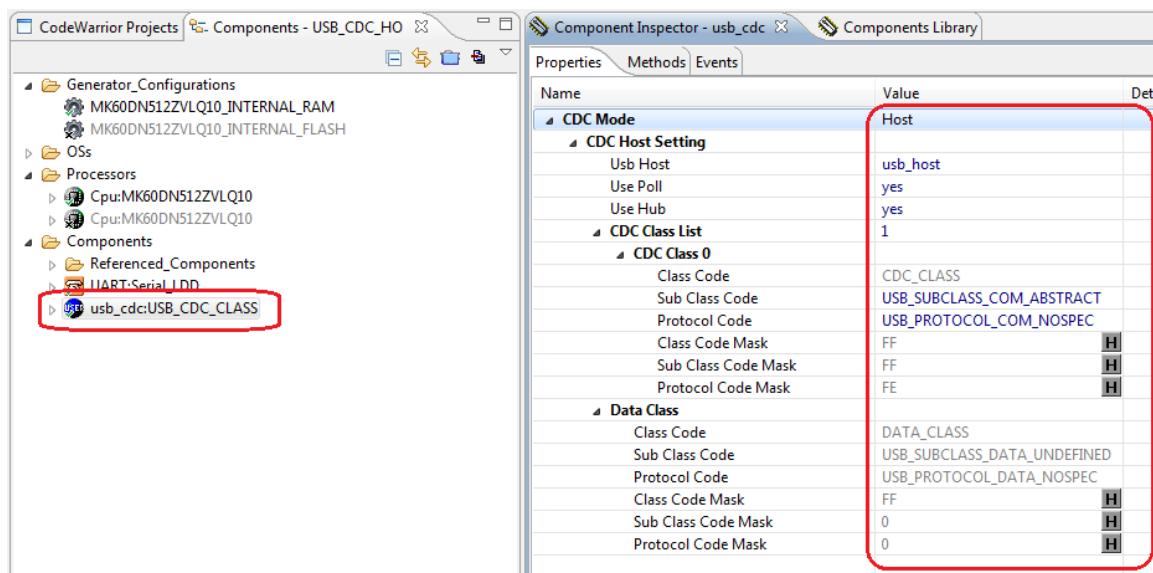


Figure 4-29 USB\_CDC\_CLASS component settings

#### Step5. Setup the USB\_HOST\_STACK component:

- Click on the USB\_HOST\_STACK component to select this target.
- This component is inherited in the USB\_CDC\_CLASS component.

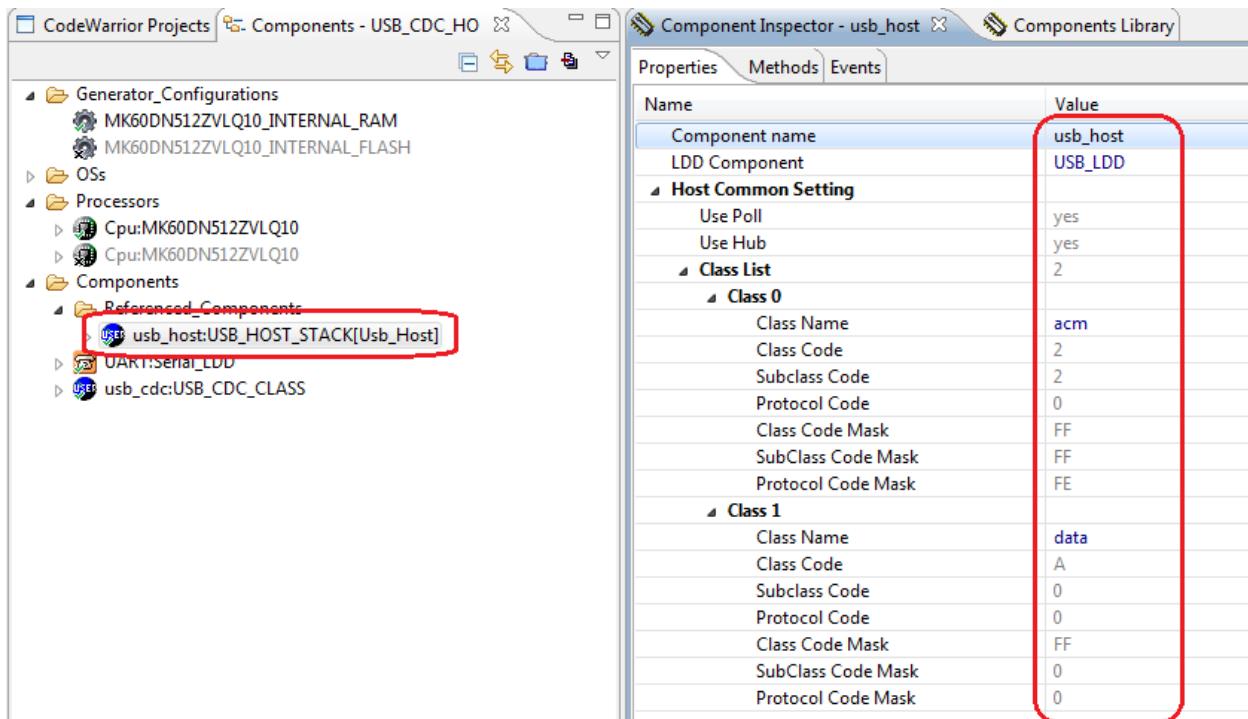


Figure 4-30 USB\_HOST\_STACK component settings

### Step6. Setup the USB\_LDD component:

- Click on the USB\_LDD component to select this target.
- Setup input parameters corresponding with USB\_CDC\_CLASS component.

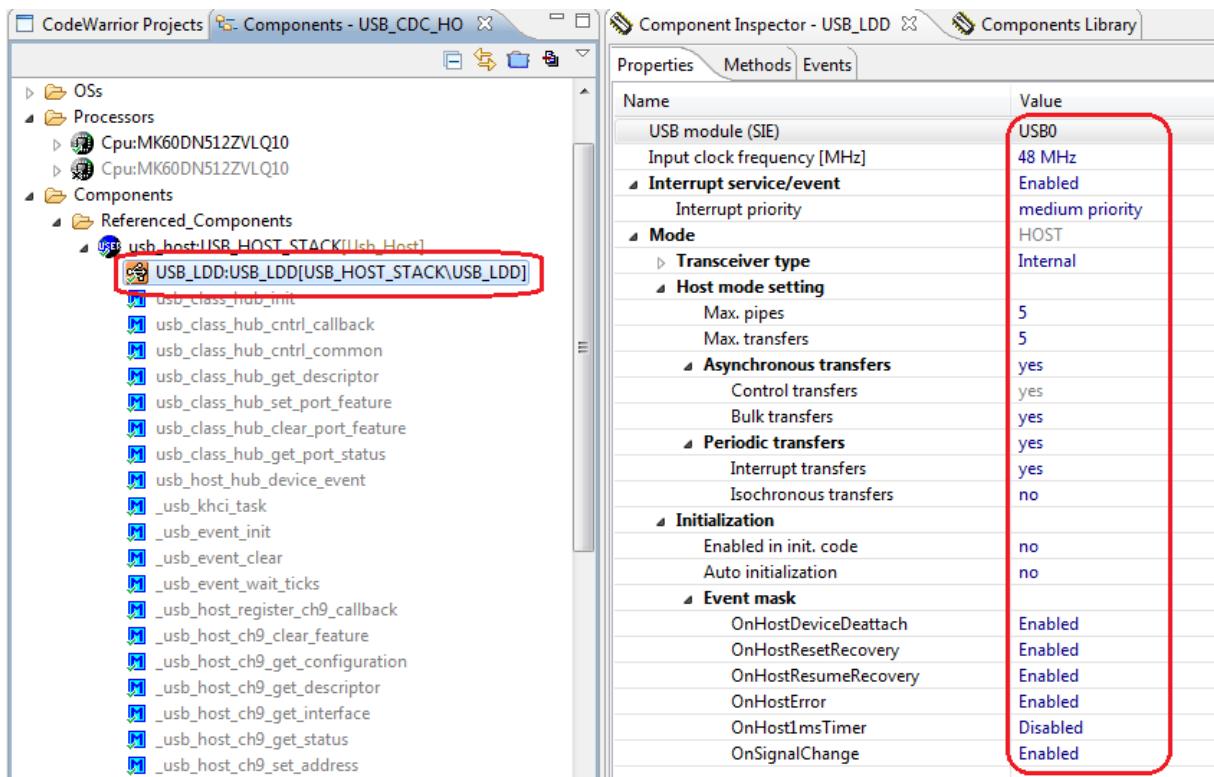


Figure 4-31 USB\_LDD component settings

### Step7. Setup the TWR\_SER\_K60\_UART component as following figure:

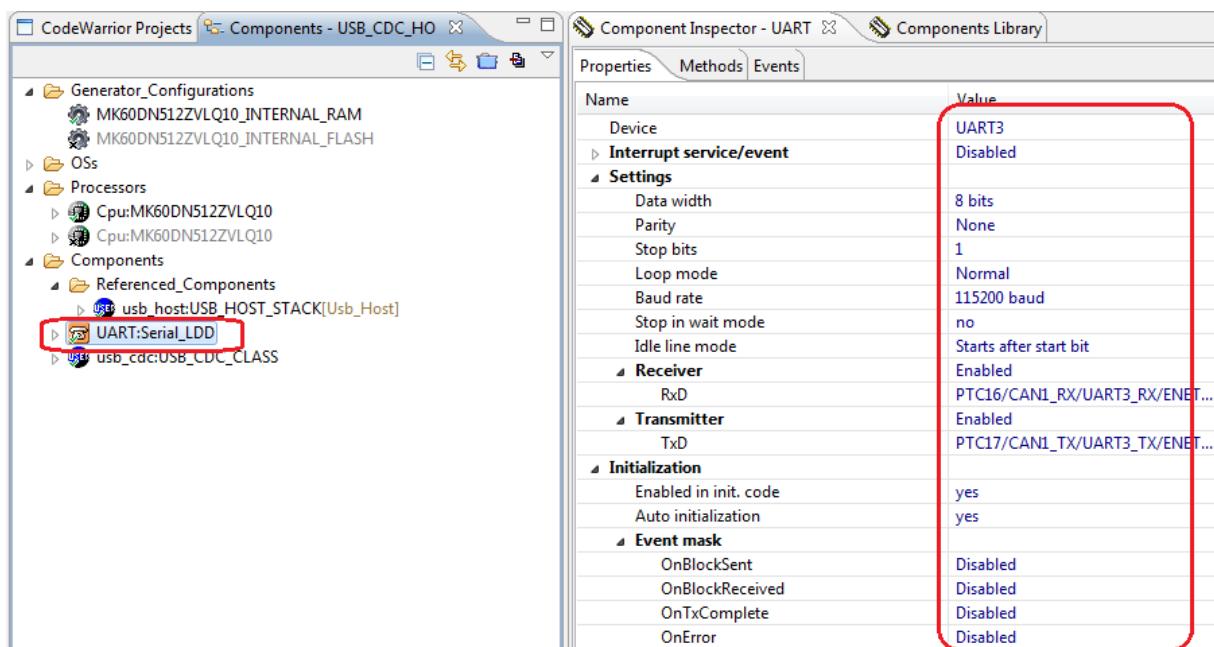


Figure 4-32 UART settings

**Step8.** Generate the CDC host source code same as the CDC device example above.

#### 4.3.3.2 Running CDC host example

Implement the following steps to run the CDC host example:

**Step1.** Setup the hardware to run example.

- The computer uses one USB cable to supply power to the board and program USB CDC host image to the flash. One COM cable is used to get events from the USB CDC host.
- The USB CDC device is plugged to the USB CDC host by one USB cable.

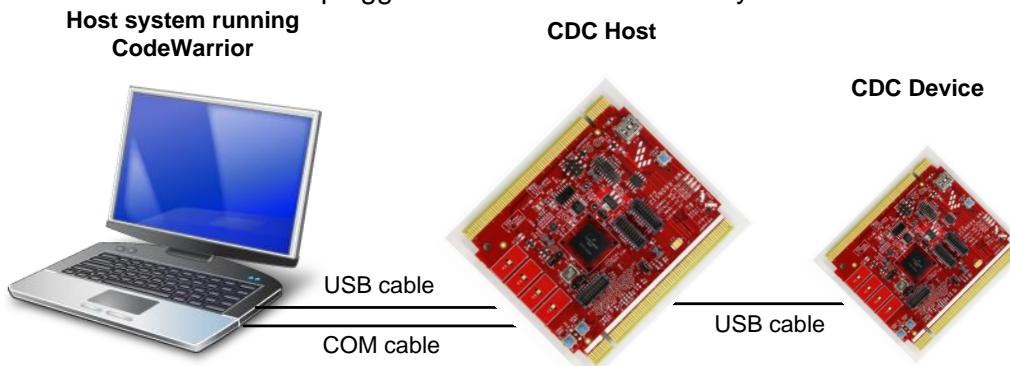


Figure 4-33 Hardware setup

**Step2.** To verify USB CDC host operating, The HyperTerminal is used and configured same as the USB CDC device example with baud rate 115200.

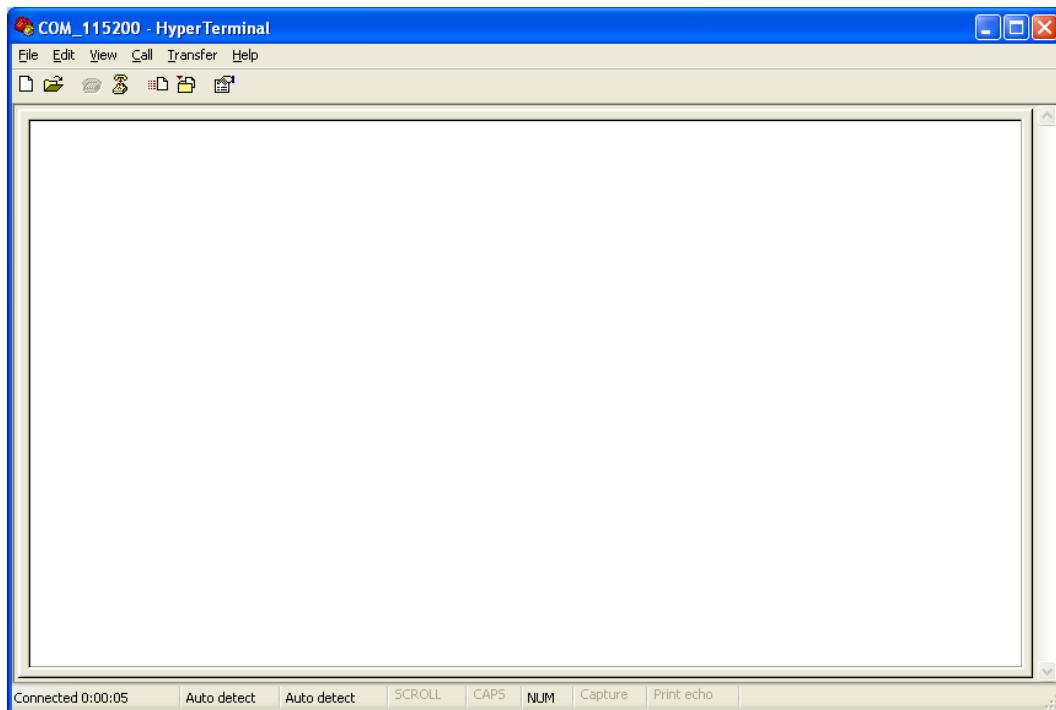


Figure 4-34 Hyper Terminal

**Step3.** After source code has been generated, we must add the code to register application callback function into the “**device\_driver\_info.c**” file.

```

/* Application call back function */
void usb_host_cdc_acm_event
(
    /* [IN] pointer to device instance */
    _usb_device_instance_handle dev_handle,
    /* [IN] pointer to interface descriptor */
    _usb_interface_descriptor_handle intf_handle,
    /* [IN] code number for event causing callback */
    uint_32 event_code
)
{
    /* Write code here ... */
    usb_host_cdc_acm_app_event(dev_handle,
                                intf_handle, event_code);
}

void usb_host_cdc_data_event
(
    /* [IN] pointer to device instance */
    _usb_device_instance_handle dev_handle,
    /* [IN] pointer to interface descriptor */
    _usb_interface_descriptor_handle intf_handle,
    /* [IN] code number for event causing callback */
    uint_32 event_code
)
{
    /* Write code here ... */
    usb_host_cdc_data_app_event(dev_handle,
                                intf_handle, event_code);
}
/*EOF*/

```

Figure 4-35 Driver Table Information

**Step4.** Build and load the image of CDC host application to the Kinetis K60 Tower Board. The HyperTerminal show the USB CDC host event as in the figure below:

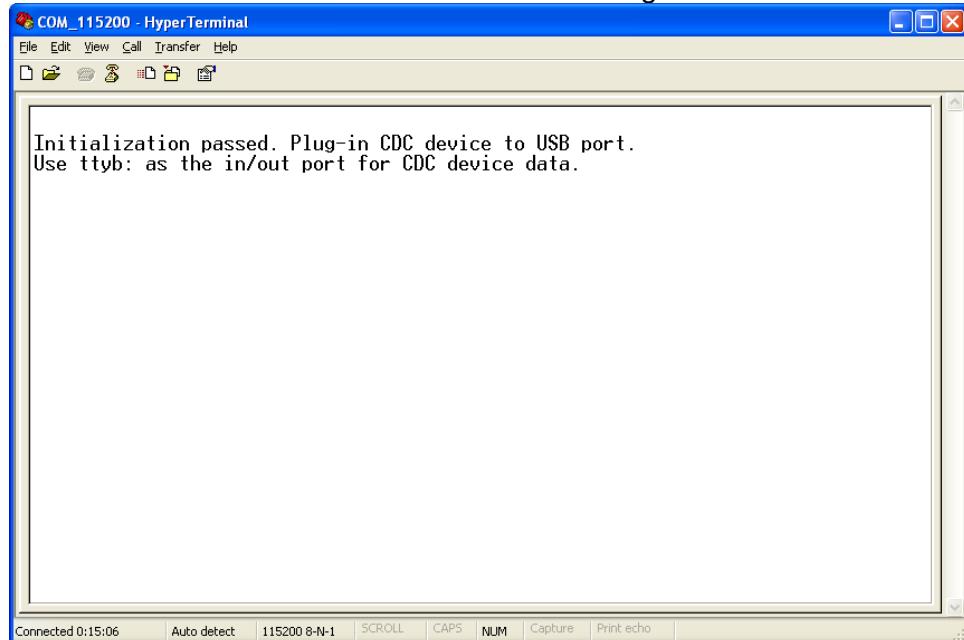


Figure 4-36 CDC host initializing

**Step5.** Plug the USB CDC device into the USB CDC host, the HyperTerminal shows transaction operation.

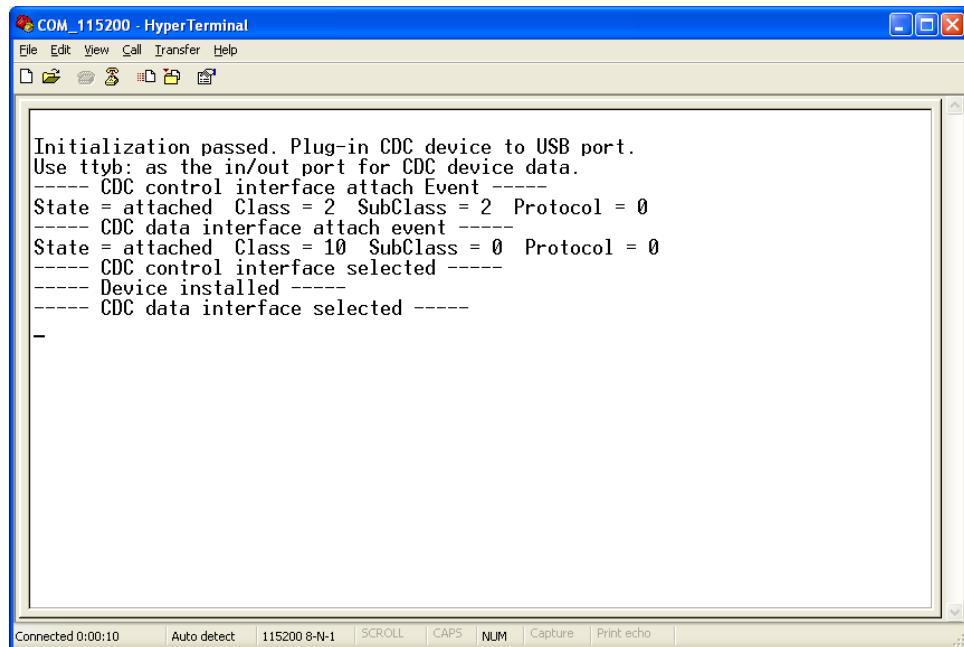


Figure 4-37 CDC host operating

## 5 USB\_MSD

This section describes how to use the USB\_DEVICE\_STACK and USB\_MSD components in the Processor Expert on Code Warrior IDE. It also guides you how to create and run the USB MSD example using those components.

### 5.1 Folder structure

#### 5.1.1 USB MSD component package

The USB MSD component package includes components as in the following figure:

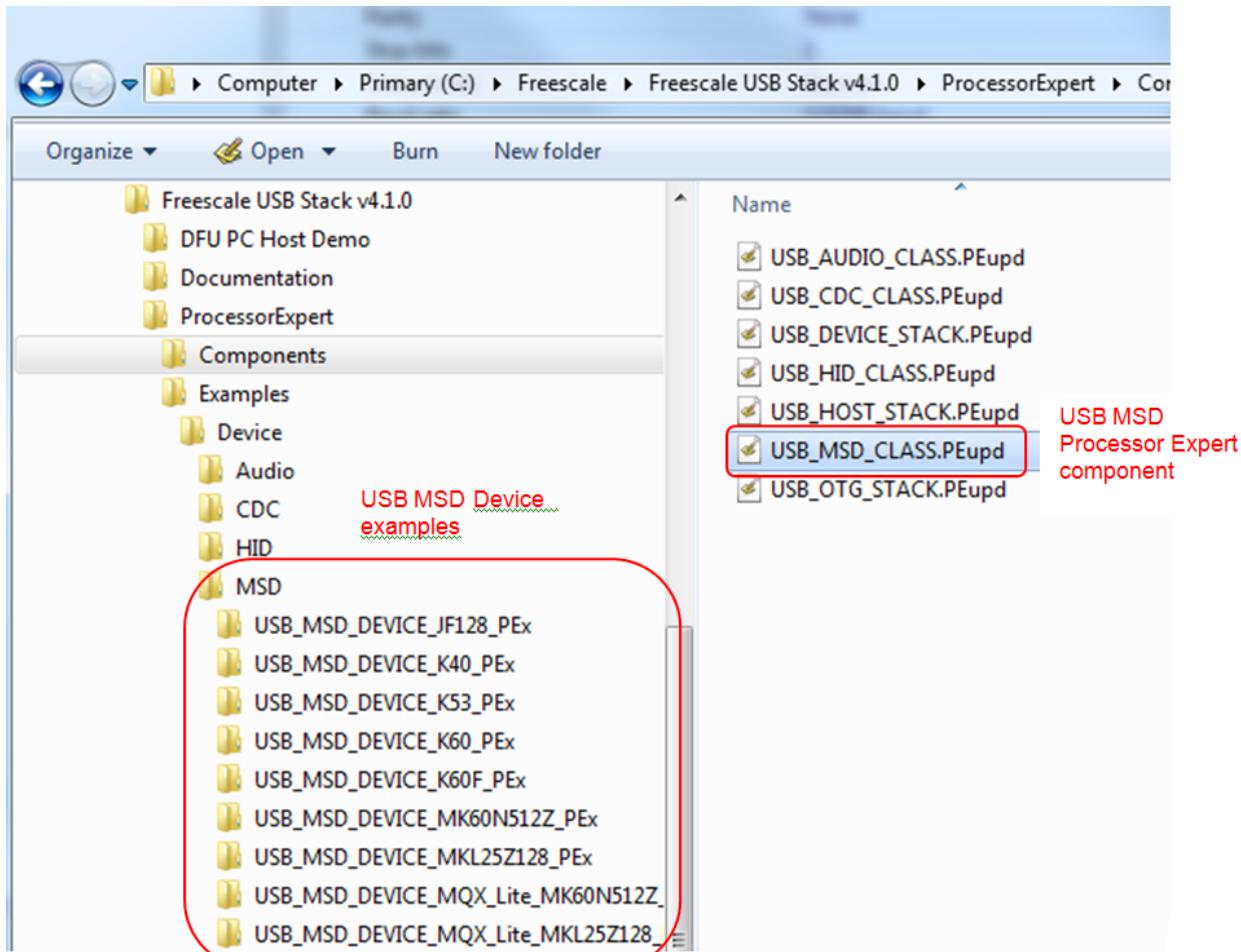


Figure 5-1: USB MSD Component packet

- USB\_DEVICE\_STACK component: This component will encapsulate all the common USB device stack functionality.
- USB\_MSD component: This component will encapsulate the Mass Storage Device Class functionality.

### 5.1.2 USB MSD example structure

The USB MSD example has the structure as in the following figure:

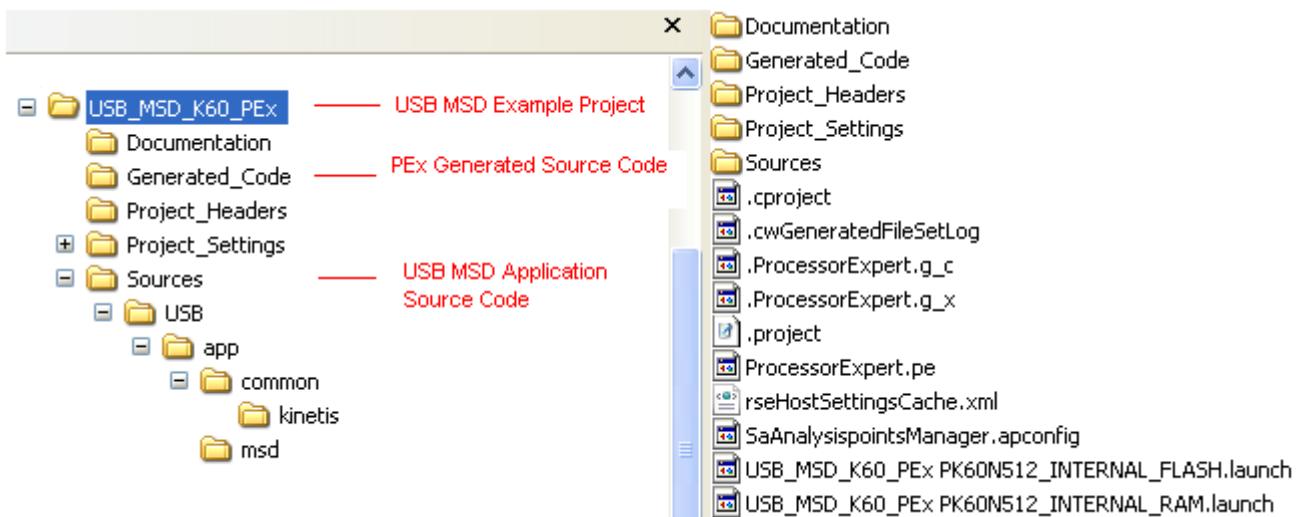


Figure 5-2: USB MSD example package

## 5.2 USB MSD example

This section describes the functionality of the USB MSD example using USB\_DEVICE\_STACK and USB\_MSD\_CLASS components on the Tower K60 Board.

### 5.2.1 Importing component

This section describes the steps to import the USB\_DEVICE\_STACK and USB\_MSD\_CLASS components used to run the USB MSD example.

**Step1.** In CodeWarrior Development Studio, chose Processor Expert → Import Package

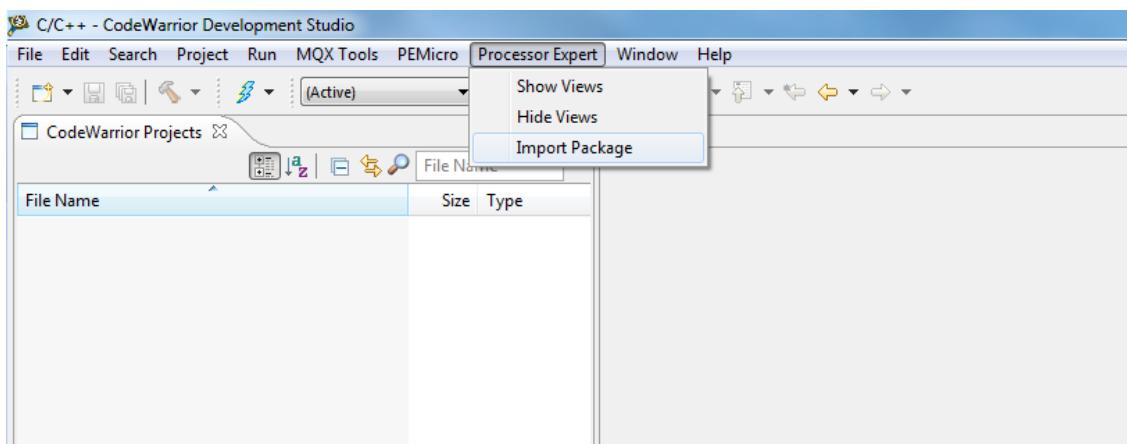


Figure 5-3 Processor Expert – Import Package

**Step2.** A browser window appears. Navigate to the folder that contains the USB\_MSD\_CLASS component, chose it and click open.

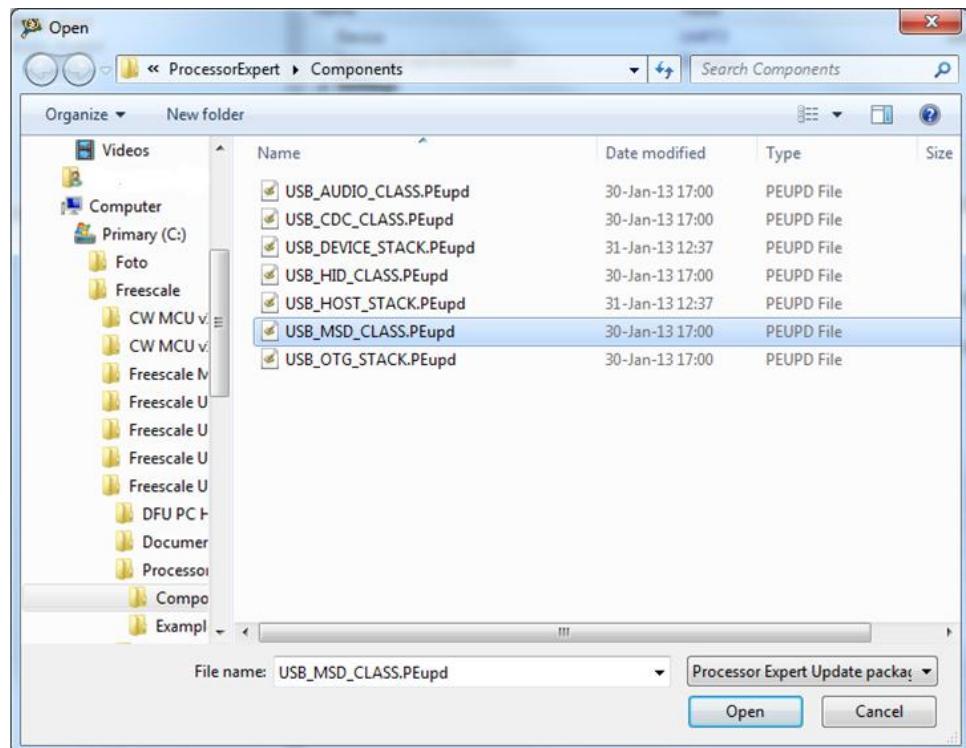


Figure 5-4: Open USB MSD package

**Step3.** An Import Complete dialog box appears. Click OK for confirmation.

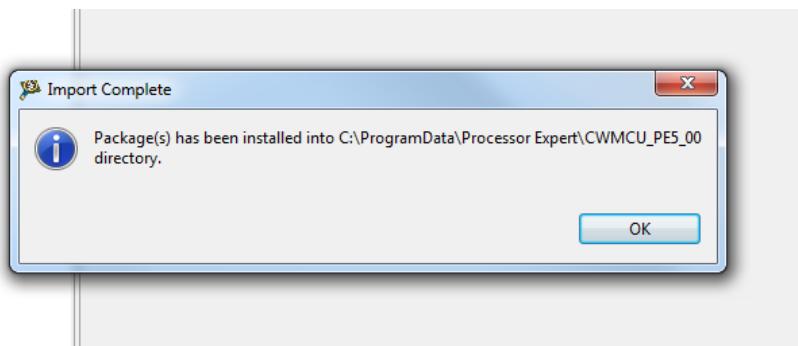


Figure 5-5: Import USB components

**Step4.** Implement the same steps to import the USB\_DEVICE\_STACK component.

## 5.2.2 MSD device example

### 5.2.2.1 Generating MSD device source code

This section describes the steps to generate USB MSD source code to run USB MSD device example.

**Step1.** Open USB\_MSD\_DEVICE\_K60\_PEx example with CW10.3 → from the CW10.3 toolbar, select Processor Expert → Select Show Views to open Components Library → Select USB\_MSD\_CLASS component and add to project.

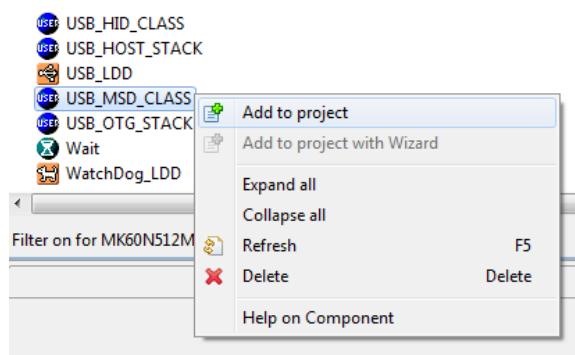


Figure 5-6: Components Library

**Step2.** Configure CPU clock to run USB module

- Select CPU target from Project Panel window to setup clock source as in the following figure:
  - o Target: Cpu:MK60N512VMD100
  - o Clock source: External reference clock 50MHz
  - o MCG mode: PEE
  - o MCG output: 48 MHz
  - o PLL output: 48MHz
  - o Core clock: 48MHz
  - o Bus clock: 48MHz
  - o External bus clock: 24MHz
  - o Flash clock: 24MHz

\*Component Inspector - Cpu

Properties Methods Events Build options Resources			Basic	Advanced	Expert	Color
Name	Value	Details				
CPU type	MK60DN512ZVMD10					
<b>Clock settings</b>						
<b>Internal oscillator</b>						
Slow internal reference clock [kHz]	32.768	32.768 kHz				
Fast internal reference clock [MHz]	4.0	4 MHz				
<b>RTC oscillator</b>	Disabled					
<b>System oscillator</b>	Enabled					
<b>Clock source</b>	External reference clock					
Clock frequency [MHz]	50.0	50 MHz				
<b>Clock source settings</b>	1					
<b>Clock source setting 0</b>						
<b>MCG settings</b>						
MCG mode	PEE					
MCG output [MHz]	48.0	48 MHz				
MCG external ref. clock source	System oscillator					
MCG external ref. clock [MHz]	50.0	50 MHz				
<b>FLL settings</b>						
<b>PLL settings</b>						
PLL module	Enabled					
PLL output [MHz]	48.0	48 MHz				
Initialization priority	minimal priority					
Watchdog disable	yes					
<b>CPU interrupts/resets</b>						
<b>External Bus</b>	Disabled					
<b>Clock configurations</b>	1					
<b>Clock configuration 0</b>						
<b>Clock source setting</b>	configuration 0					
MCG mode	PEE					
<b>System clocks</b>						
Core clock	48.0	48 MHz				
Bus clock	48.0	48 MHz				
External bus clock	24.0	24 MHz				
Flash clock	24.0	24 MHz				

Figure 5-7 Clock source settings

- Chose advanced tab

\*Component Inspector - Cpu

Properties Methods Events Build options Resources			Basic	Advanced	Expert	Color
Name	Value	Details				
Component name	Cpu					
CPU type	MK60DN512ZVMD10					

Figure 5-8: Open Advanced Tab

- PLL/FLL clock selection:
  - o PLL clock

Bus clock	48.0	48 MHz
External clock prescaler	Auto select	2
External bus clock	24.0	24 MHz
Flash clock prescaler	Auto select	2
Flash clock	24.0	24 MHz
<b>PLL/FLL clock selection</b>	PLL clock	
Clock frequency [MHz]	48.0	48 MHz
<b>USB clock settings</b>		
USB clock divider	Auto select	1
USB clock multiply	Auto select	1
USB clock	48.0	48 MHz

Figure 5-9: Clock configuration

Freescale Processor Expert USB Components Quick Start Guide

### **Step3.** Setup the USB\_MSD\_CLASS component

- Select USB\_MSD component → Fill all values into component value fields:

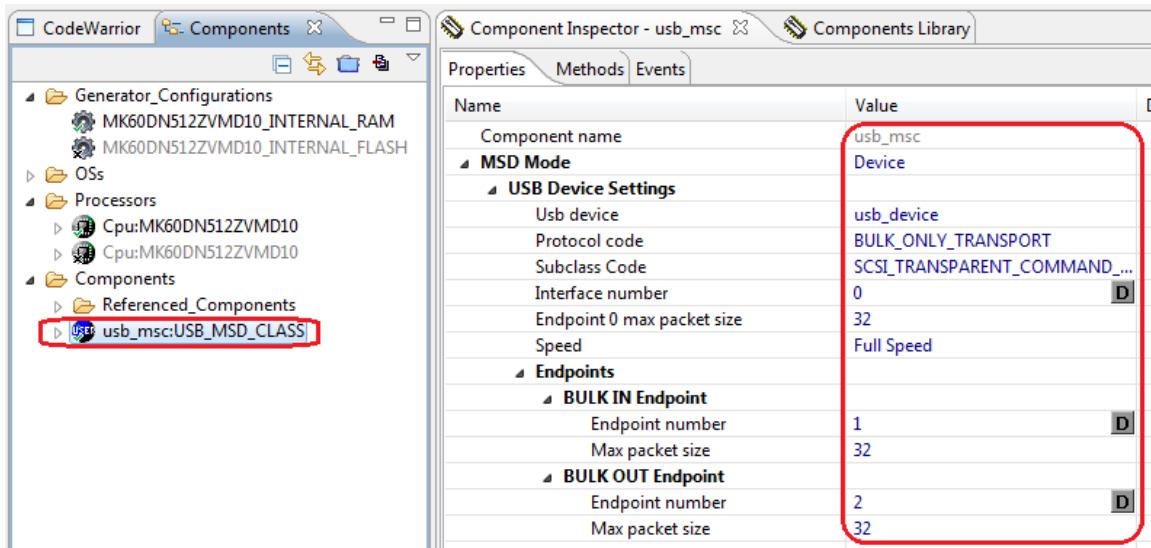


Figure 5-10: MSD device mode settings

### **Step4.** Setup the USB\_DEVICE\_STACK component

- Click on the USB\_DEVICE\_STACK component to select this target.
- Fill input parameters as in the following figures.
  - o Device Descriptor:

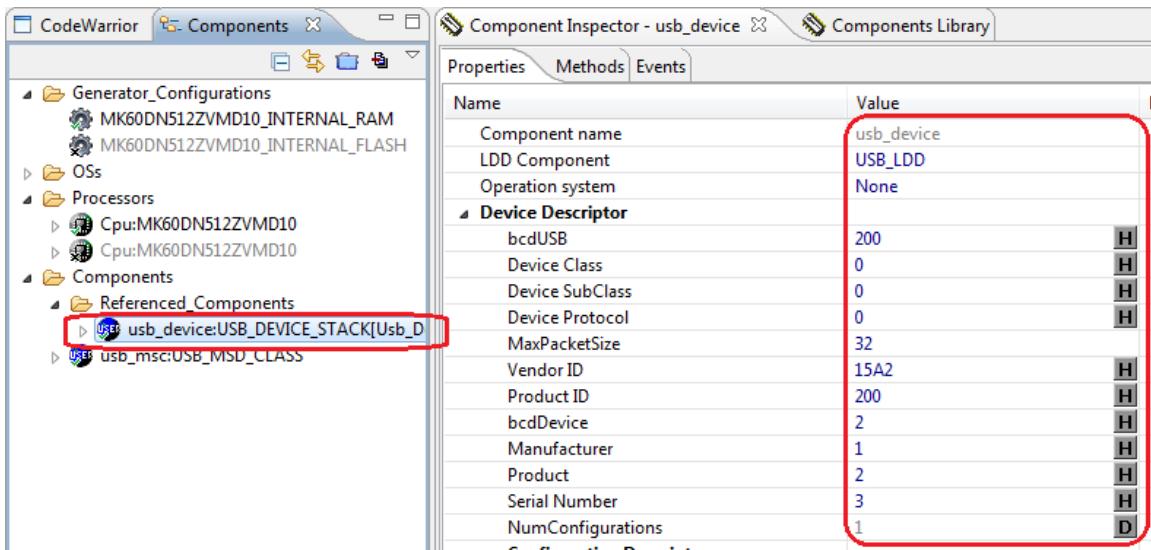


Figure 5-11: Device Descriptor settings

- Configuration Descriptor: This group is inherited in the USB\_MSD\_CLASS component.

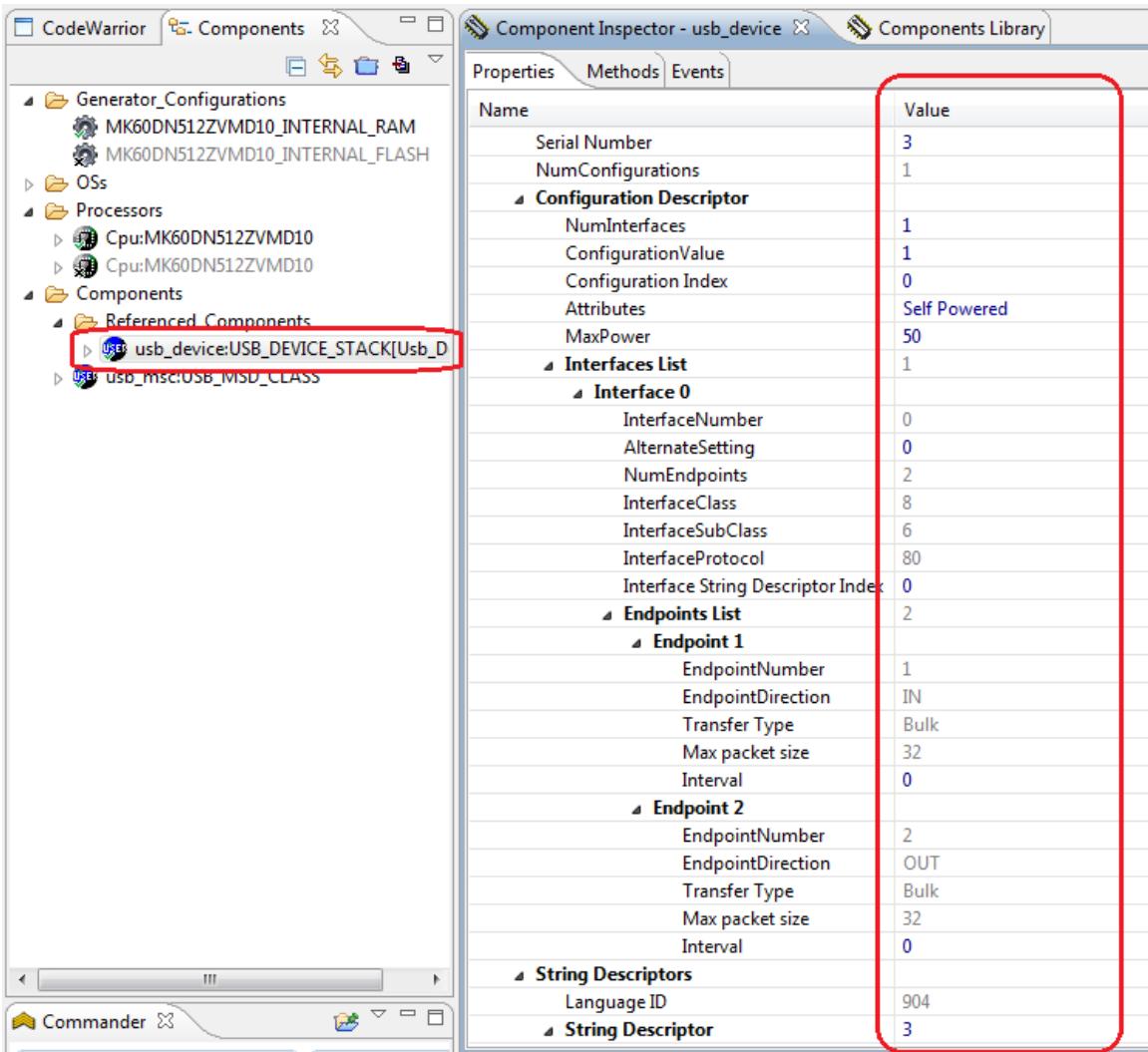


Figure 5-12: Configuration Descriptor settings

- String Descriptors: Click plus button to add string field → Fill string values

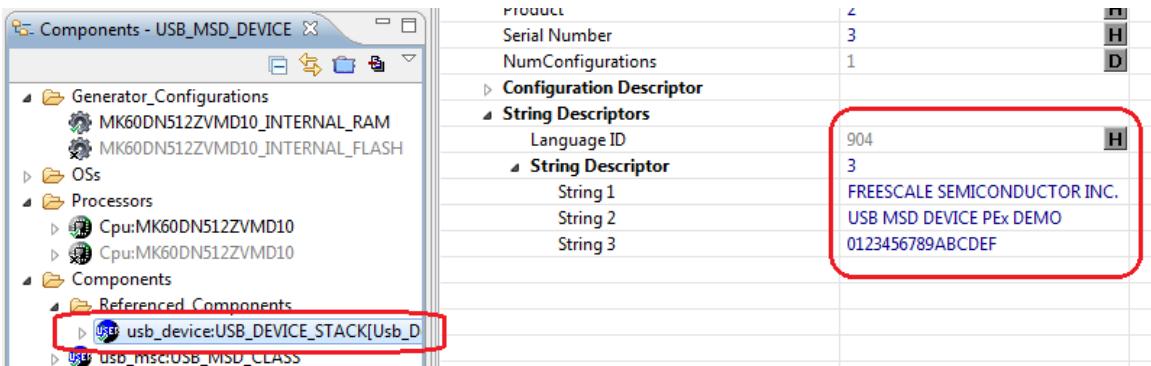


Figure 5-13: String Descriptor settings

## Step5. Setup the USB\_LDD component

- Click on the USB\_LDD component to select this target.
- Setup input parameters which corresponds with USB\_MSD\_CLASS component

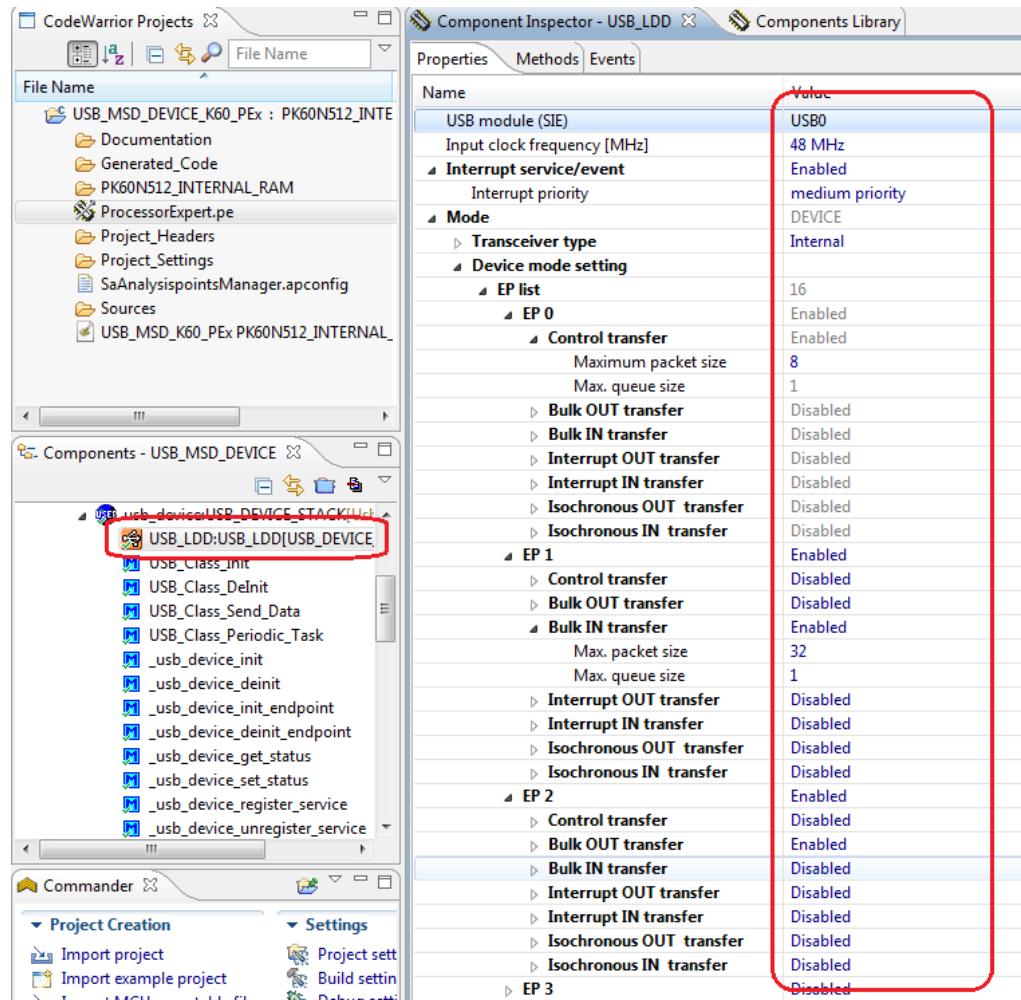


Figure 5-14: USB\_LDD component

- Enable events

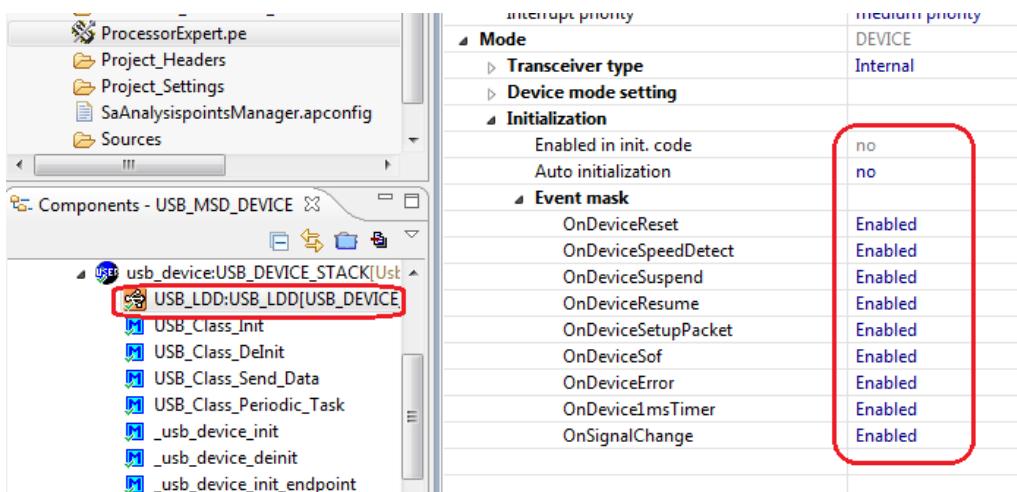


Figure 5-15: Enable events

### **Step6.** Generate Processor Expert code:

After setting the components, select ProcessorExpert.pe in the project folder → Right click → Choose Generate Processor Expert Code field to generate source code.

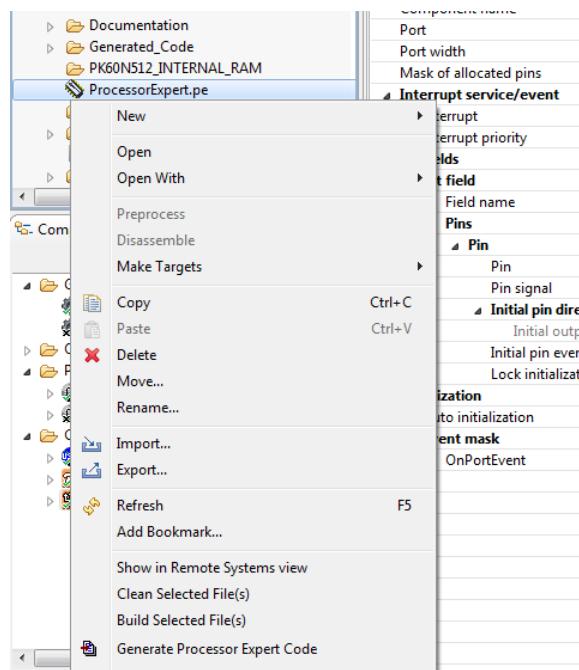


Figure 5-16: Generate Processor Expert code

## 5.3 Running MSD device example

Implement the following steps to run the CDC device example

**Step1.** Build and load the image of CDC device application to the Kinetis K60 Tower Board.

**Step2.** After the image has been loaded successfully, connect the USB port of the board to a USB port of the computer. You can see the device entry on device manager of Windows

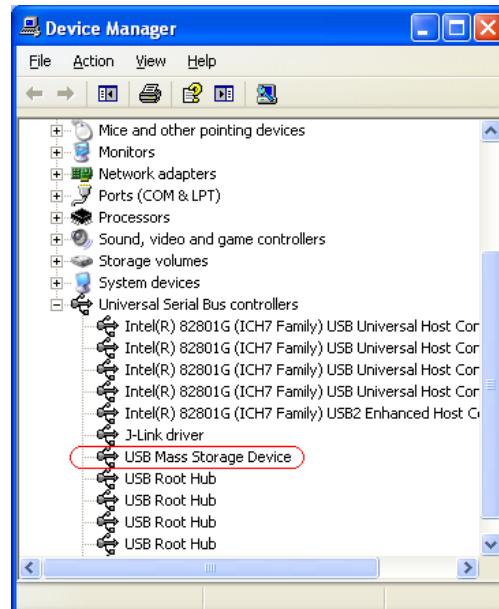


Figure 5-17: Device entry

- The MSD device is displayed as a removable disk in My Computer.

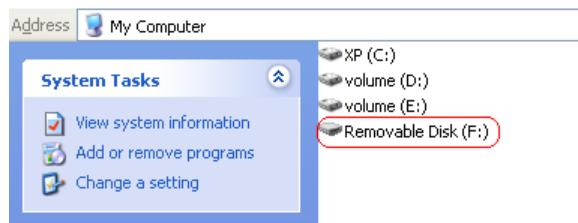


Figure 5-18: Removable disk

- You can use Total Phase Data Center to view data transferred between the computer and the device.

Sp	Index	m:s.ms.us	Len	Err	Dev	Ep	Record
FS	257	0:02.435.215	3.00 ms				[4 SOF]
FS	258	0:02.436.219	2 B		01	00	[+ Get String Descriptor]
FS	271	0:02.439.216	3.00 ms				[4 SOF]
FS	272	0:02.440.219	4 B		01	00	[+ Get String Descriptor]
FS	285	0:02.443.216	3.00 ms				[4 SOF]
FS	286	0:02.444.220	2 B		01	00	[+ Get String Descriptor]
FS	299	0:02.447.217	5.00 ms				[6 SOF]
FS	300	0:02.448.220	34 B		01	00	[+ Get String Descriptor]
FS	321	0:02.453.218	2.00 ms				[3 SOF]
FS	322	0:02.454.221	0 B		01	00	[+ Set Configuration]
FS	331	0:02.456.218	72.0 ms				[73 SOF]
FS	332	0:02.526.231	1 B		01	00	[+ Get Max LUN]
FS	345	0:02.529.228	5.00 ms				[6 SOF]
FS	346	0:02.530.231	36 B		01	02	[+ Inquiry [0]]
FS	366	0:02.535.229	4.00 ms				[5 SOF]
FS	367	0:02.536.232	12 B		01	02	[+ Read Format Capacities [0]]
FS	383	0:02.540.229	10.0 ms				[11 SOF]
FS	384	0:02.546.234	8 B		01	02	[+ Read Capacity [0]]
FS	400	0:02.551.231	5.00 ms				[6 SOF]
FS	401	0:02.552.235	512 B		01	02	[+ Read [0]]
FS	477	0:02.557.232	4.00 ms				[5 SOF]
FS	478	0:02.558.235	8 B		01	02	[+ Mode Sense [0]]

Figure 5-19: Total Phase Data Center

## 6 USB\_AUDIO\_CLASS

This section describes how to run USB AUDIO device and USB AUDIO host examples with the USB\_AUDIO\_CLASS component in the Processor Expert on Code Warrior IDE.

### 6.1 Architecture overview

The USB\_AUDIO\_CLASS component makes use of the USB\_AUDIO\_DEVICE and USB\_AUDIO\_HOST existing components to provide AUDIO APIs for user.

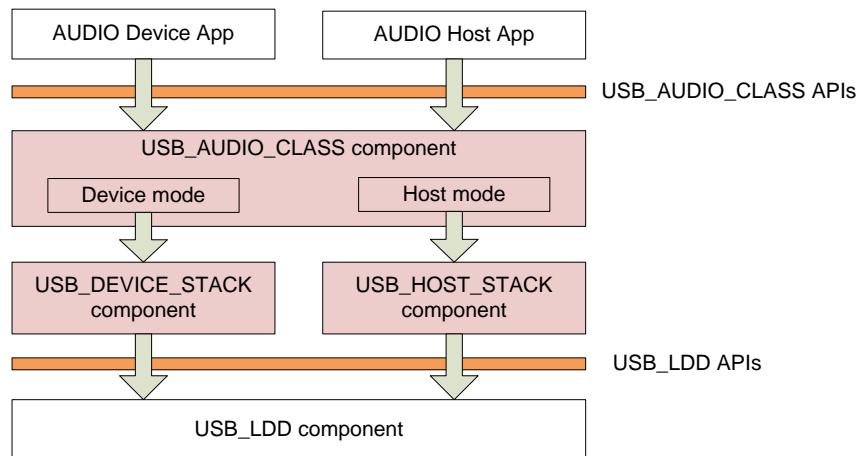


Figure 6-1 Architecture overview

### 6.2 Folder structure

#### 6.2.1 USB Audio component package structure

The deliver package includes items as the following figure:

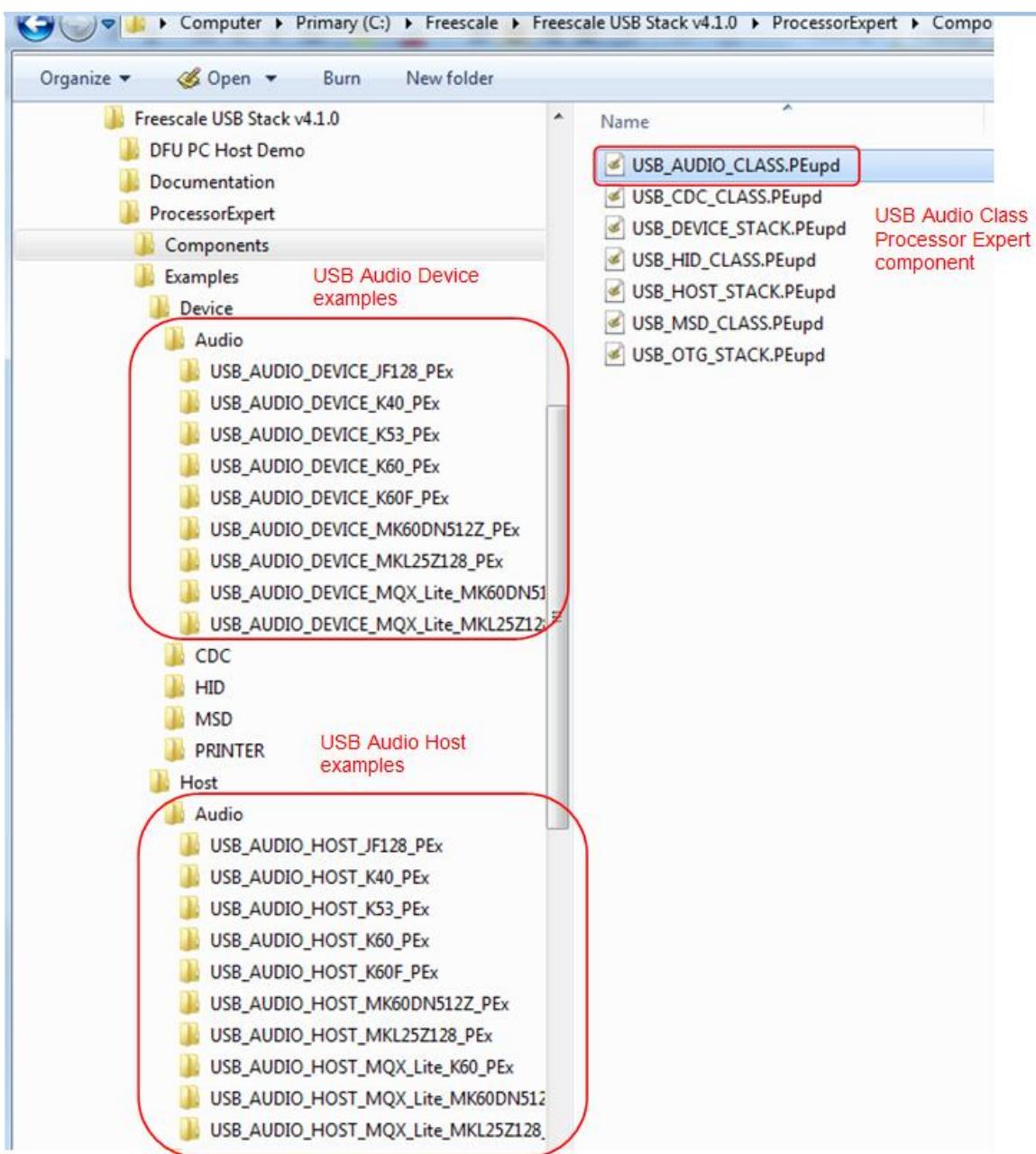


Figure 6-2 Delivered Package Directory Structure

- USB\_HOST\_STACK component: This component will encapsulate all the common USB host stack functionality.
- USB\_AUDIO\_CLASS component: This component will encapsulate the Communication Device Class functionality.

## 6.2.2 AUDIO device example structure

The USB AUDIO device example has the structure as the following figure:

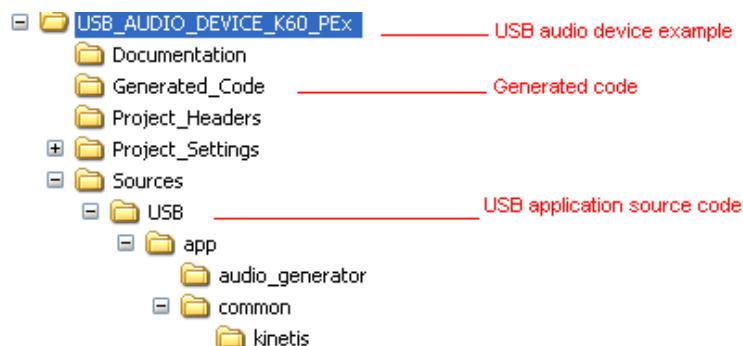


Figure 6-3 AUDIO device folder structure

## 6.2.3 AUDIO host example structure

The USB AUDIO host example has the structure as the following figure:



Figure 6-4 AUDIO host folder structure

## 6.3 USB AUDIO example

This section describes the functionality of the USB AUDIO example on the Tower K60 board.

### 6.3.1 Importing component

This section describes steps to import components used to run USB AUDIO example.

**Step1.** In CodeWarrior Development Studio, chose Processor Expert → Import Package

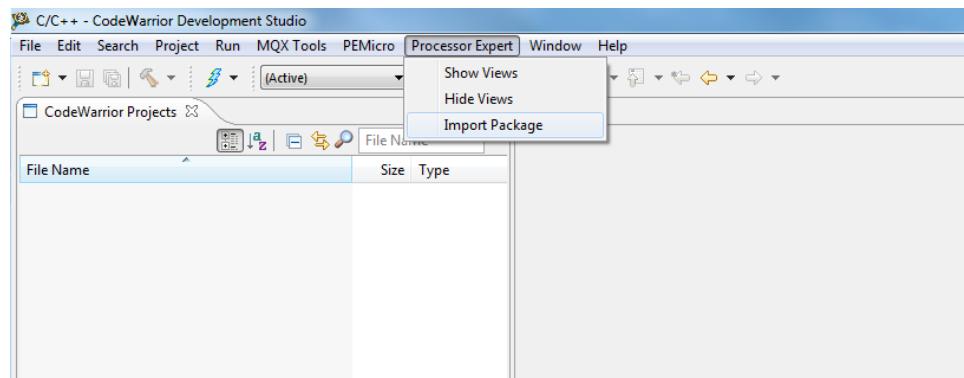


Figure 6-5 Processor Expert Import Package

**Step2.** A browser window appears. Navigate to the folder that contains the USB\_AUDIO\_CLASS component, chose it and click open.

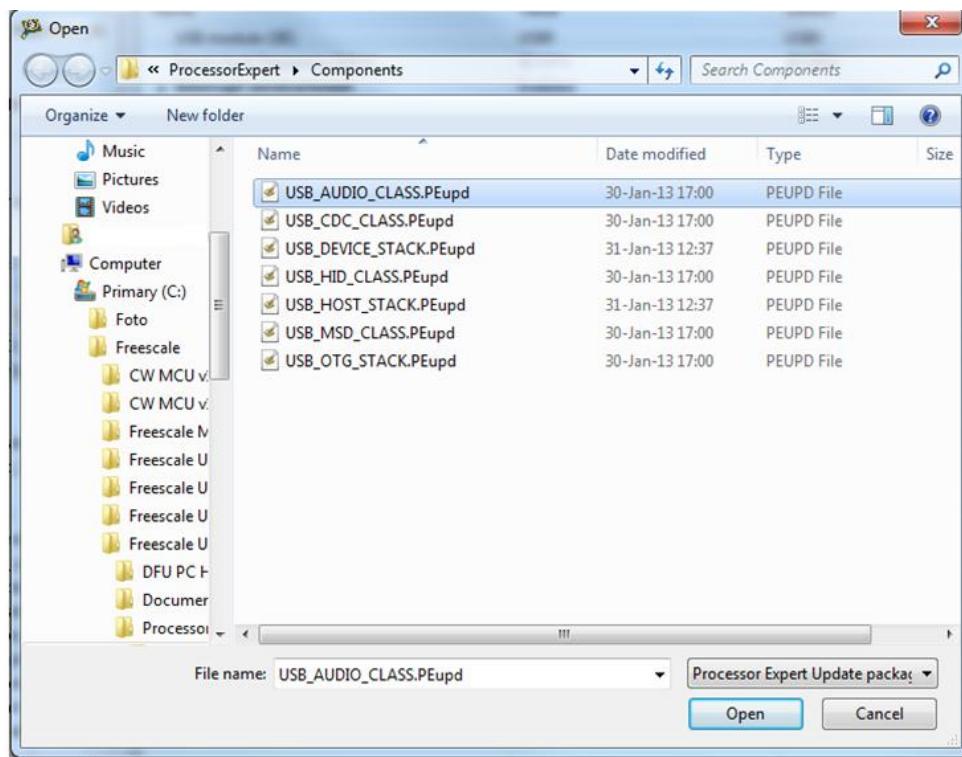


Figure 6-6 Deliver package navigation

**Step3.** An Import Complete dialog box appears. Click OK for confirmation.

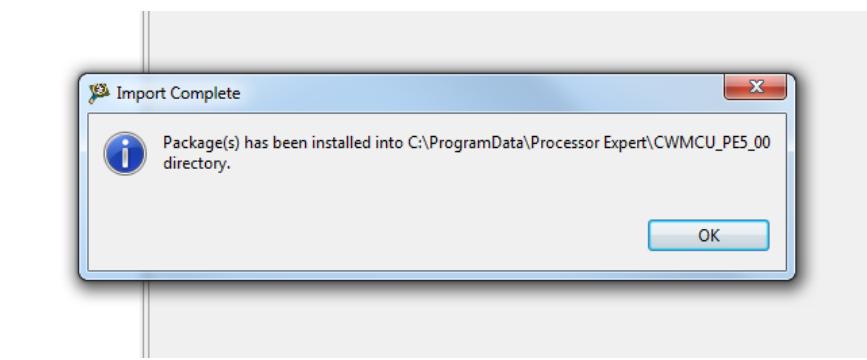


Figure 6-7 Import USB components

**Step4.** Implement the same steps to import the USB\_HOST\_STACK and USB\_DEVICE\_STACK components.

## 6.3.2 AUDIO device example

### 6.3.2.1 Generating AUDIO device source code

This section describes the steps to generate USB AUDIO source code to run USC AUDIO device example.

**Step1.** Open USB\_AUDIO\_DEVICE\_K60\_PEx example with CW10.3 → from the CW10.3 toolbar, select Processor Expert → Select Show Views to open Components Library → Select USB\_AUDIO\_CLASS component and add to project.

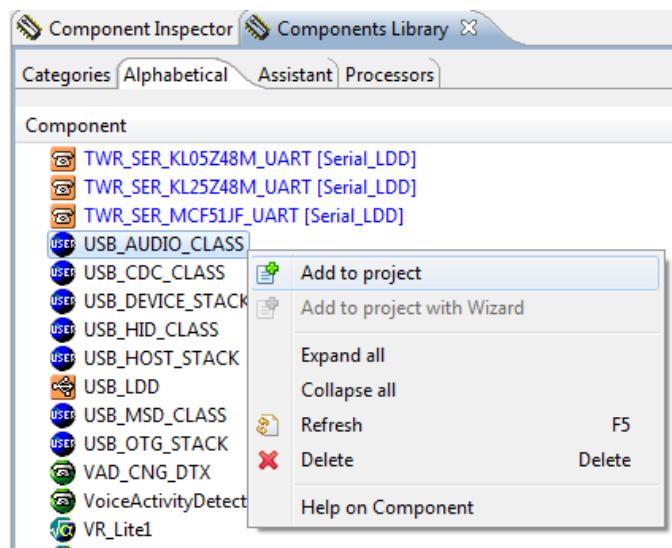


Figure 6-8 Components Library

### Step2. Configure CPU clock to run USB module

- Select CPU target from Project Panel window to setup clock source as in the following figure:
  - o Target: Cpu:MK60N512VMD100
  - o Clock source: External reference clock 50MHz
  - o MCG mode: PEE
  - o MCG output: 48 MHz
  - o PLL output: 48MHz
  - o Core clock: 48MHz
  - o Bus clock: 48MHz
  - o External bus clock: 24MHz
  - o Flash clock: 24MHz

\*Component Inspector - Cpu

Basic Advanced Expert

Properties Methods Events Build options Resources

Name	Value	Details
CPU type	MK60DN512ZVMD10	
<b>Clock settings</b>		
<b>Internal oscillator</b>		
Slow internal reference clock [kHz]	32.768	32.768 kHz
Fast internal reference clock [MHz]	4.0	4 MHz
<b>RTC oscillator</b>	Disabled	
<b>System oscillator</b>	Enabled	
<b>Clock source</b>	External reference clock	
Clock frequency [MHz]	50.0	50 MHz
<b>Clock source settings</b>	1	
<b>Clock source setting 0</b>		
<b>MCG settings</b>		
MCG mode	PEE	
MCG output [MHz]	48.0	48 MHz
MCG external ref. clock source	System oscillator	
MCG external ref. clock [MHz]	50.0	50 MHz
<b>FLL settings</b>		
<b>PLL settings</b>		
<b>PLL module</b>	Enabled	
<b>PLL output [MHz]</b>	48.0	48 MHz
Initialization priority	minimal priority	15
Watchdog disable	yes	
<b>CPU interrupts/resets</b>		
<b>External Bus</b>	Disabled	
<b>Clock configurations</b>	1	
<b>Clock configuration 0</b>		
<b>Clock source setting</b>		
MCG mode	PEE	
<b>System clocks</b>		
Core clock	48.0	48 MHz
Bus clock	48.0	48 MHz
External bus clock	24.0	24 MHz
Flash clock	24.0	24 MHz

Figure 6-9 Clock source settings

- Chose advanced tab.
- PLL/FLL clock selection:
  - o PLL clock

Bus clock	48.0	48 MHz
External clock prescaler	Auto select	2
External bus clock	24.0	24 MHz
Flash clock prescaler	Auto select	2
Flash clock	24.0	24 MHz
<b>PLL/FLL clock selection</b>		
<b>PLL clock</b>	48.0	48 MHz
<b>USB clock settings</b>		
USB clock divider	Auto select	1
USB clock multiply	Auto select	1
USB clock	48.0	48 MHz

Figure 6-10 Clock configuration

### Step3. Setup the USB\_AUDIO\_CLASS component:

- Click on the USB\_AUDIO\_CLASS component to select this target.
- Show in the Component Inspector window → Select AUDIO Mode field → Select Device mode.

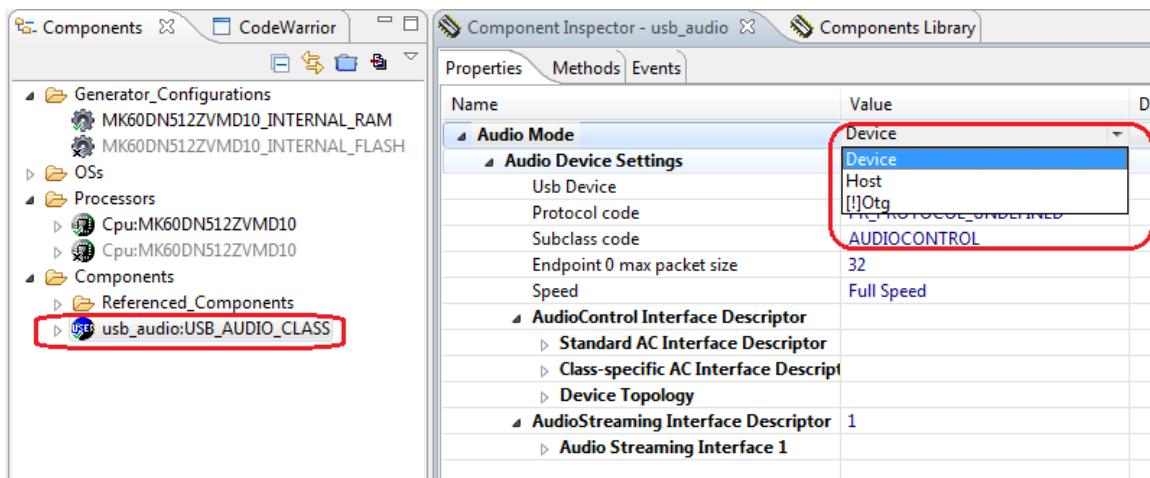


Figure 6-11 AUDIO device mode setting

- Fill input parameters of the USB\_AUDIO\_CLASS component as in the following figure:

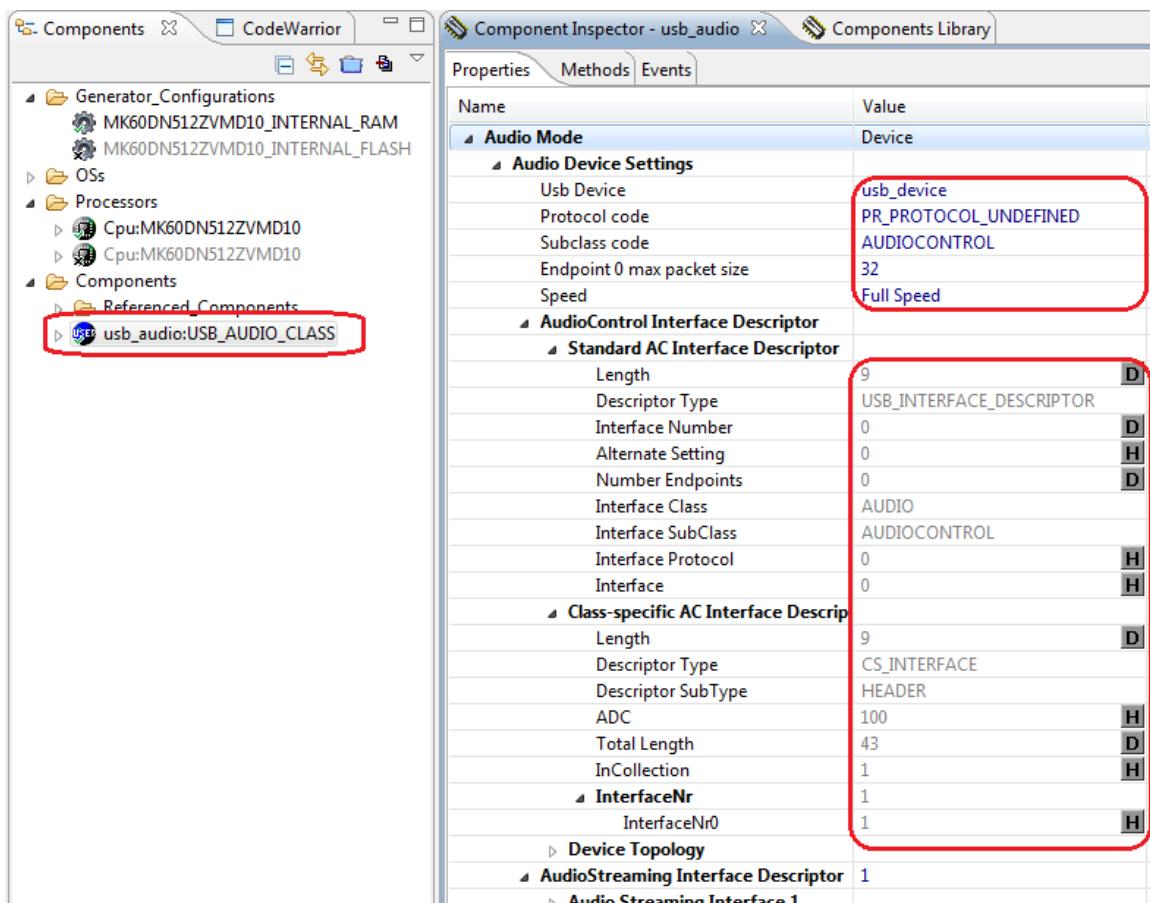


Figure 6-12 Audio Control Interface settings

<b>Device Topology</b>		
<b>Input Terminal Descriptor</b>	1	
<b>Input Terminal0</b>		
Length	12	D
Descriptor Type	CS_INTERFACE	
Descriptor SubType	INPUT_TERMINAL	
Terminal ID	1	H
Terminal Type	Microphone	
AssocTerminal	0	H
Number Channels	1	D
Channel Config	0	H
Channel Names	0	H
Terminal	0	H

Figure 6-13 Input terminal settings

<b>Device Topology</b>		
<b>Input Terminal Descriptor</b>	1	
<b>Feature Unit Descriptor</b>	1	
<b>Feature Unit0</b>		
Length	10	D
Descriptor Type	CS_INTERFACE	
Descriptor SubType	FEATURE_UNIT	
Unit ID	2	H
Source ID	1	H
Control Size	2	D
<b>Supported Controls</b>	1	
<b>Supported Controls0</b>		
Mute Control	yes	
Volume Control	yes	
Bass Control	yes	
Mid Control	yes	
Treble Control	yes	
Graphic Equalizer Control	yes	
Automatic Gain Control	yes	
Delay Control	yes	
Bass Boost	yes	
Loudness	yes	
Master Control	0	H
Feature	0	H

Figure 6-14 Feature Unit settings

<b>Device Topology</b>	
<b>Input Terminal Descriptor</b>	1
<b>Feature Unit Descriptor</b>	1
<b>Output Terminal Descriptor</b>	1
<b>Output Terminal0</b>	
Length	9
Descriptor Type	CS_INTERFACE
Descriptor SubType	OUTPUT_TERMINAL
Terminal ID	3
Terminal Type	USB streaming
Associated Terminal	0
Source ID	2
Terminal	0

Figure 6-15 Output terminal settings

<b>AudioStreaming Interface Descriptor</b>	1
<b>Audio Streaming Interface 1</b>	
<b>Alternate Setting</b>	2
<b>Class-specific AS General Interface Descriptor</b>	
Length	7
Descriptor Type	CS_INTERFACE
Descriptor SubType	AS_GENERAL
Terminal Link	3
Delay	0
FormatTag	PCM8
<b>Type I Format Type Descriptor</b>	
Length	11
Descriptor Type	CS_INTERFACE
DescriptorSubType	FORMAT_TYPE
FormatType	FORMAT_TYPE_I
NrChannels	1
SubFrame Size	1
BitResolution	8
SamFreqType	1
SamFreq	1F40
<b>Audio Standard Endpoint Descriptor</b>	
Length	9
Descriptor Type	USB_ENDPOINT_DESCRIPTOR
Endpoint Address	81
Attributes	1
MaxPacketSize	16
Interval	1
Refresh	0
SynchAddress	0

Figure 6-16 Audio Streaming Interface settings

**Step4.** Setup the USB\_DEVICE\_STACK component:

- Click on the USB\_DEVICE\_STACK component to select this target.
- Fill input parameters as in the following figures:
  - o Device Descriptor :

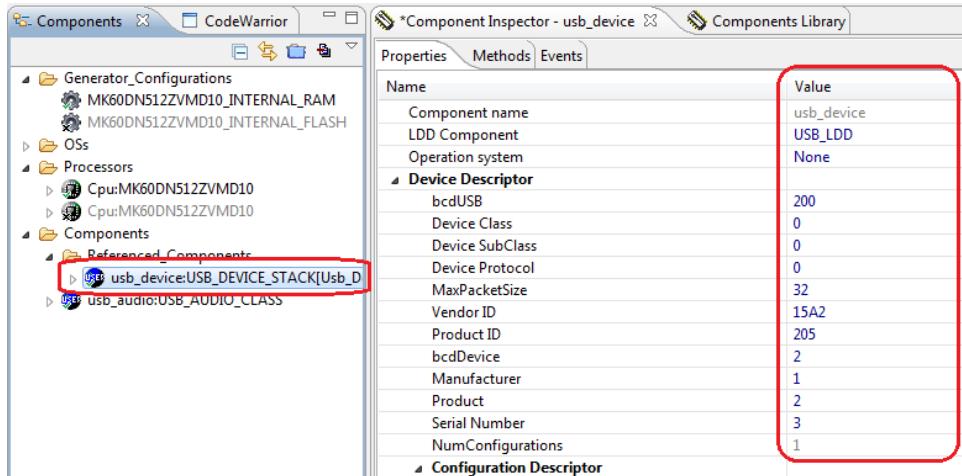


Figure 6-17 Device Descriptor settings

- o Configuration Descriptor: This group is inherited the USB\_AUDIO\_CLASS component.

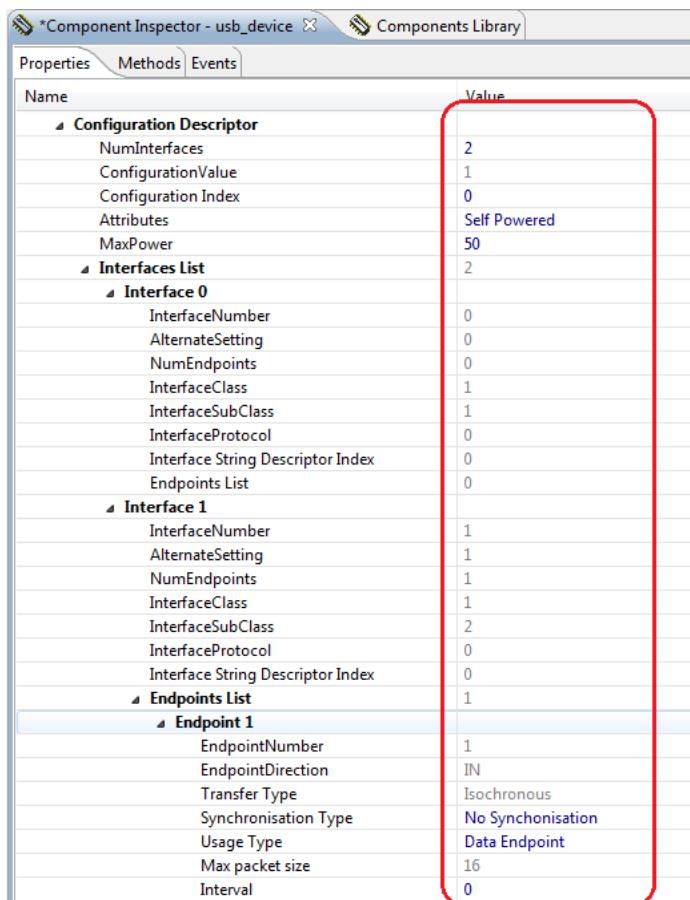


Figure 6-18 Configuration Descriptor settings

- String Descriptors: Click plus button to add string field → Fill string values.

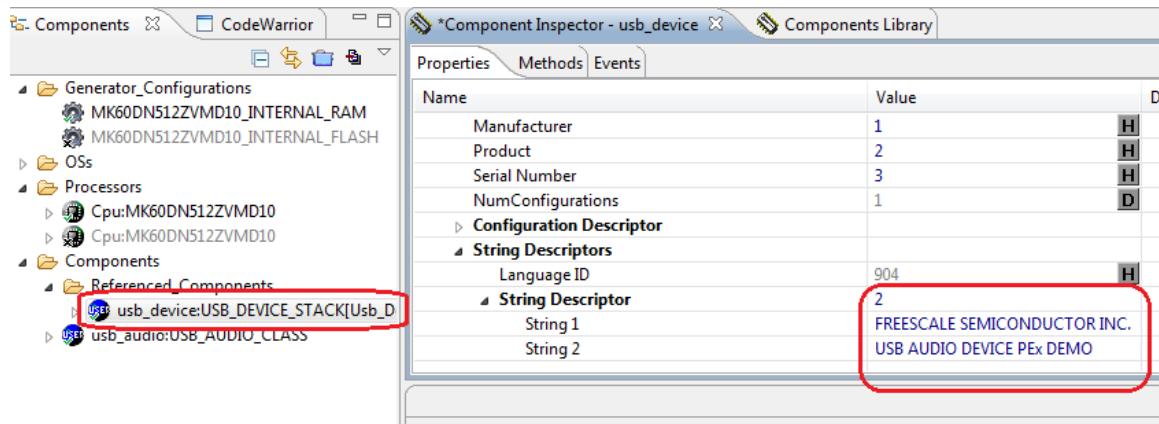


Figure 6-19 String Descriptor settings

#### Step5. Setup the USB\_LDD component:

- Click on the USB\_LDD component to select this target.
- Setup input parameters corresponds with USB\_AUDIO\_CLASS component:

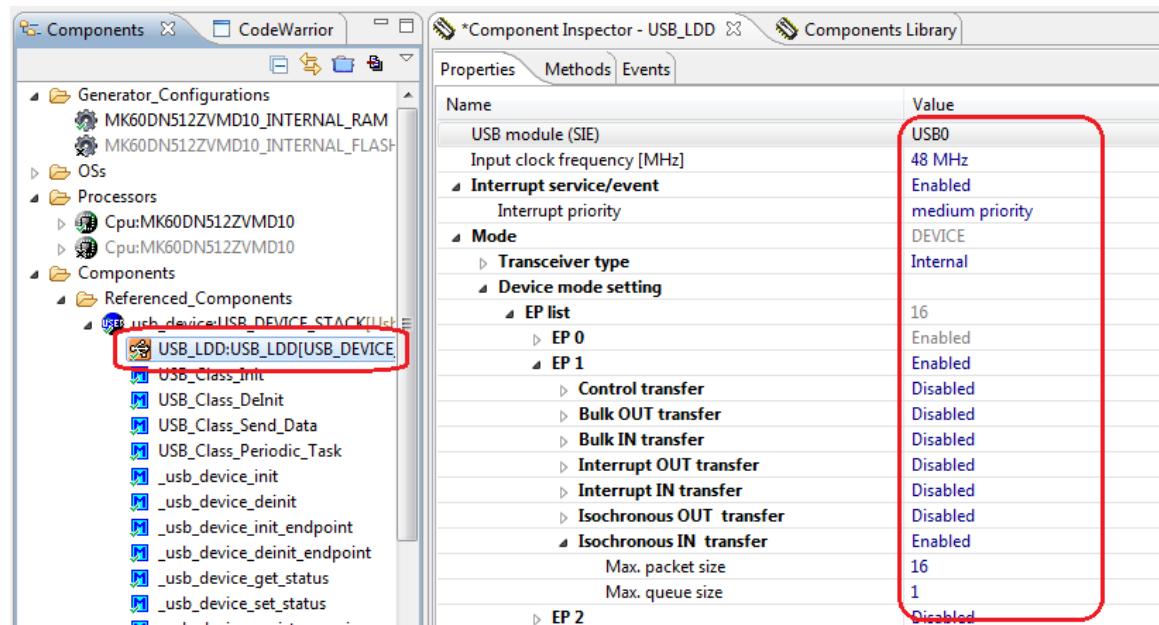


Figure 6-20 USB\_LDD component settings

#### Step6. Generate Processor Expert code:

After setting the components, select ProcessorExpert.pe in the project folder → Right click → Choose Generate Processor Expert Code field to generate source code.

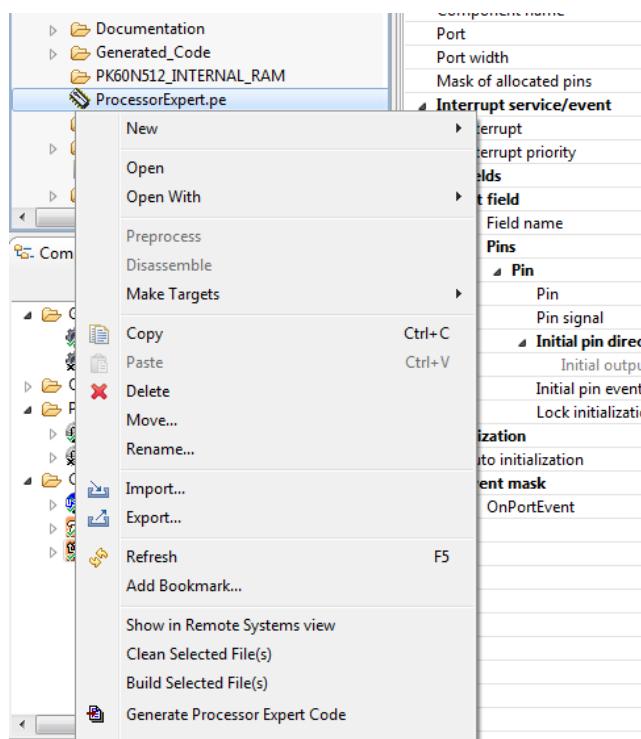


Figure 6-21 Generate Processor Expert code

### 6.3.2.2 Running AUDIO device example

Implement the following steps to run the AUDIO device example

**Step1.** Build and load the image of AUDIO device application to the Kinetis K60 Tower board.

**Step2.** After the image has been loaded successfully, connect the USB port of the board to a USB port of the computer. You can see the device entry on device manager of Windows:

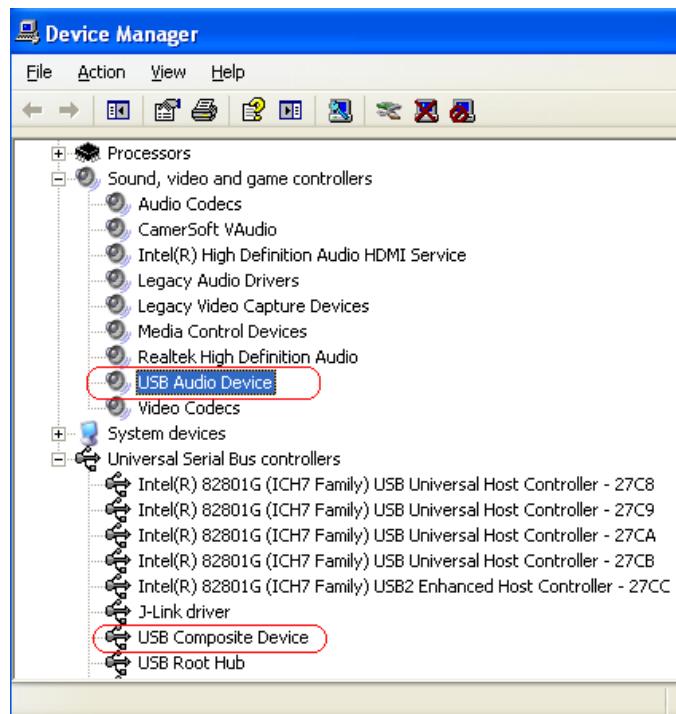


Figure 6-22 Device entry

Freescale Processor Expert USB Components Quick Start Guide

**Step3.** To ensure that application runs correctly, the Sound Recorder is used to record audio data from the AUDIO device.



Figure 6-23 Launch Sound Recorder application

- The Sound Recorder opens as in the figure below.



Figure 6-24 Sound Recorder startup

- To start, click on the button as in the following figure:



Figure 6-25 Start recording data



Figure 6-26 Sound Recorder Operating

- Choose **Save As...** from File taskbar to save audio data to .wav file

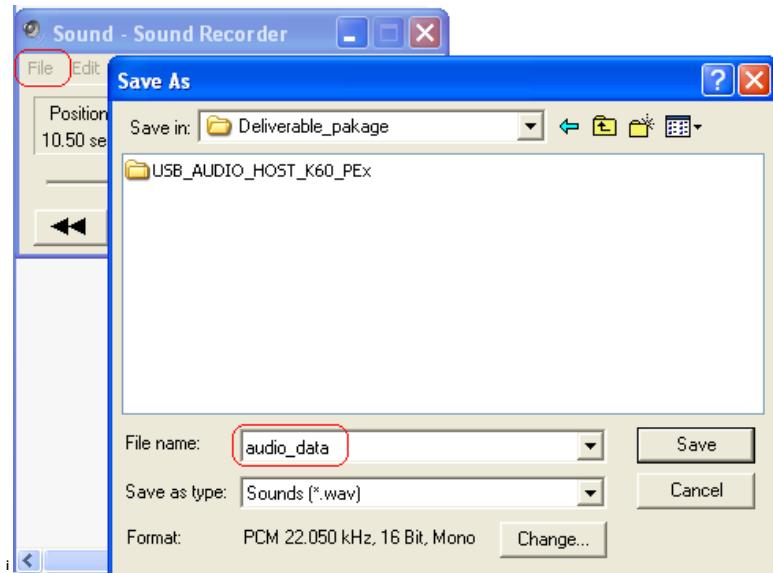


Figure 6-27 Sound Recorder Operating

### 6.3.3 AUDIO host example

#### 6.3.3.1 Generating AUDIO host source code

This section describes the steps to generate USB AUDIO source code to run USB AUDIO host example.

**Step1.** Open the USB\_AUDIO\_HOST\_K60\_PEx example with CW10.3 → from CW10.3 toolbar → Select Processor Expert → Select Show Views to open Components Library. Select USB\_AUDIO\_CLASS component and add to project same as the AUDIO device example above.

**Step2.** Implement same as step1 to add the TWR\_SER\_K60\_UART component in to the project.

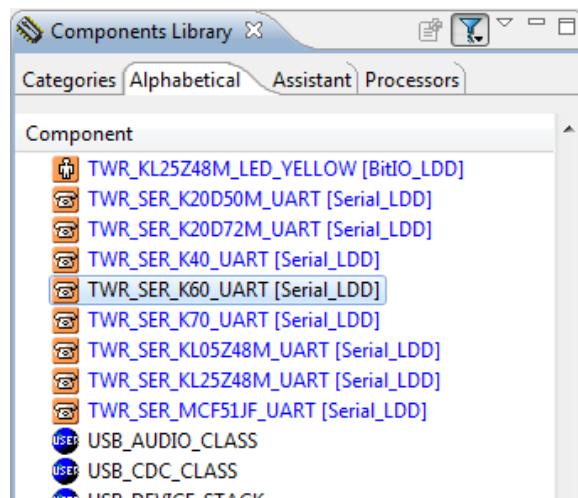


Figure 6-28 Component show views

**Step3.** Setup the CPU clock to run USB module same as the AUDIO device example above.

**Step4.** Setup the USB\_AUDIO\_CLASS component:

- Click on the USB\_AUDIO\_CLASS component to select this target.
- Show in the Component Inspector window → Select AUDIO Mode field → Select Host mode.

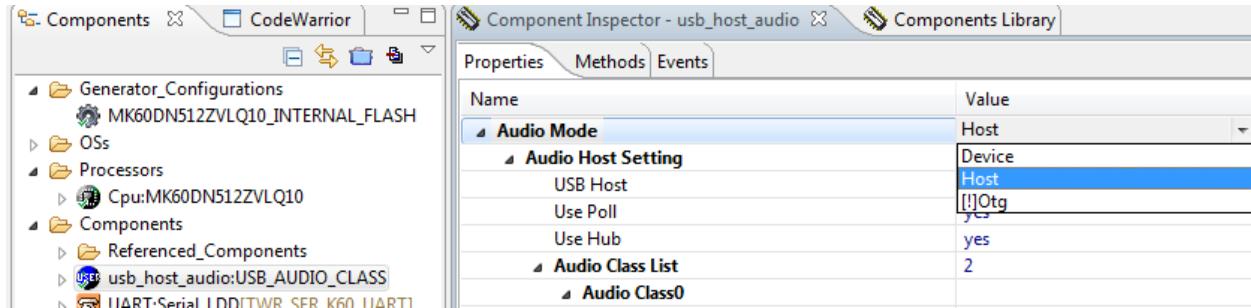


Figure 6-29 AUDIO host mode setting

- Fill input parameters into AUDIO class list settings:

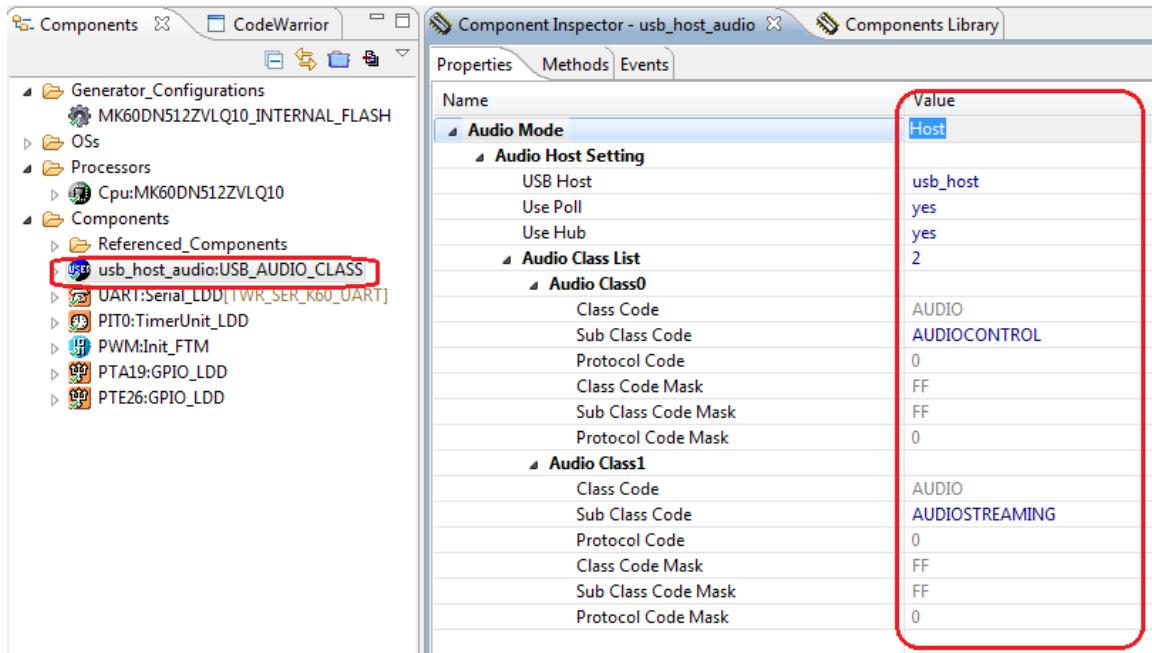


Figure 6-30 USB\_AUDIO\_CLASS component settings

### Step5. Setup the USB\_HOST\_STACK component:

- Click on the USB\_HOST\_STACK component to select this target.
- This component is inherited in the USB\_AUDIO\_CLASS component.

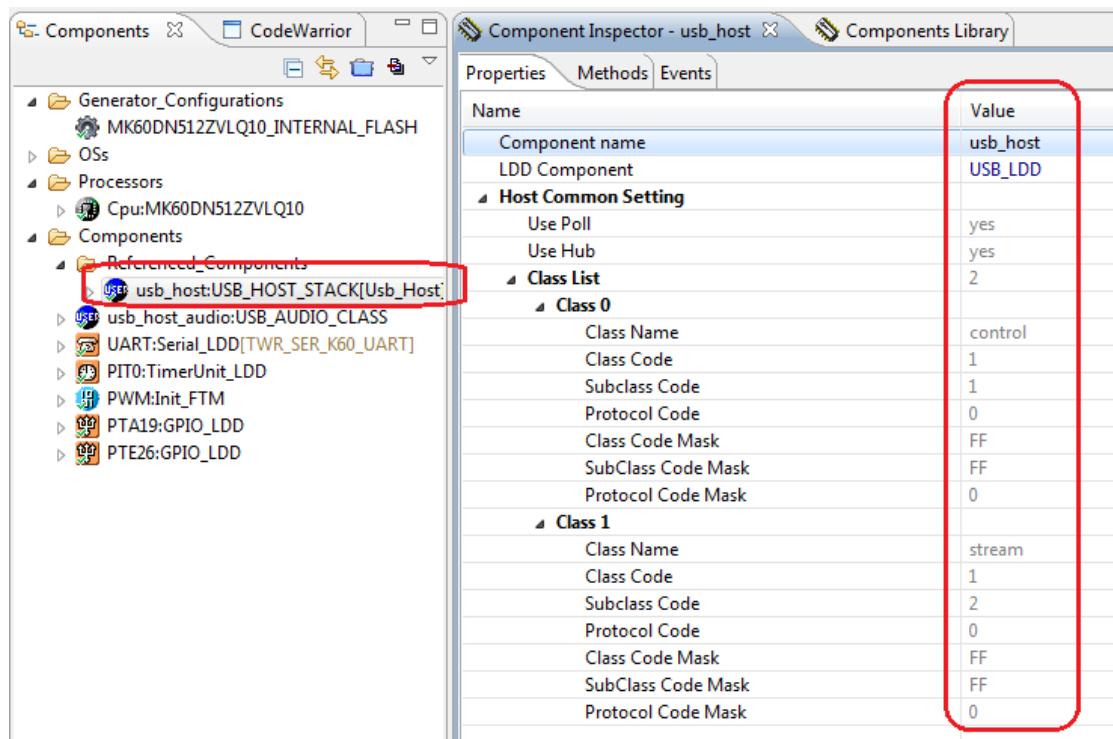


Figure 6-31 USB\_HOST\_STACK component settings

### Step6. Setup the USB\_LDD component:

- Click on the USB\_LDD component to select this target.
- Setup input parameters which corresponds with USB\_AUDIO\_CLASS component:

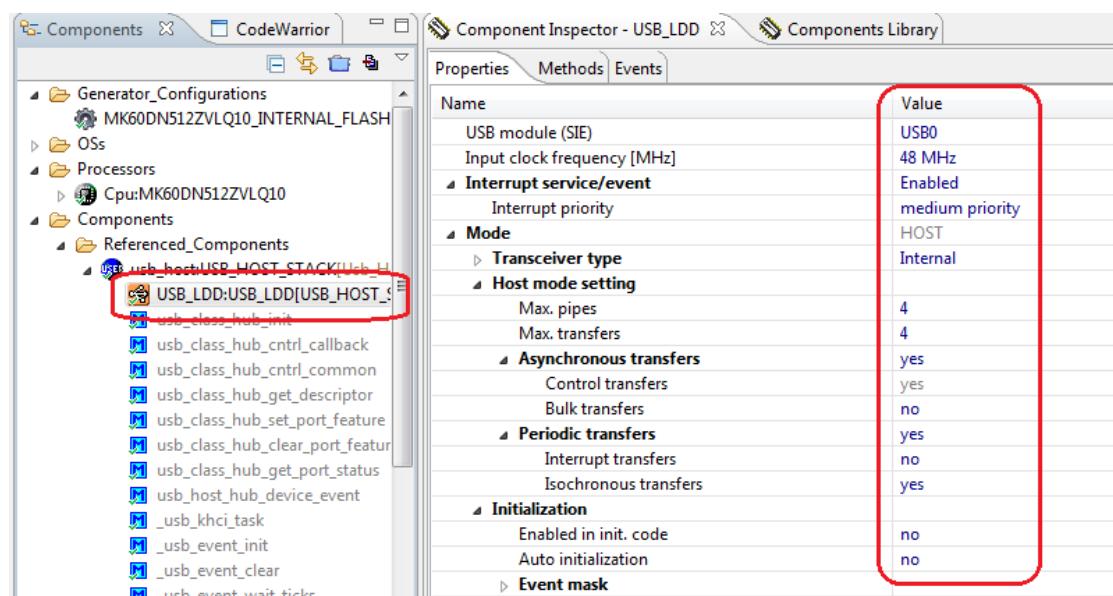


Figure 6-32 USB\_LDD component settings

**Step7.** Setup the TWR\_SER\_K60\_UART component as in the following figure:

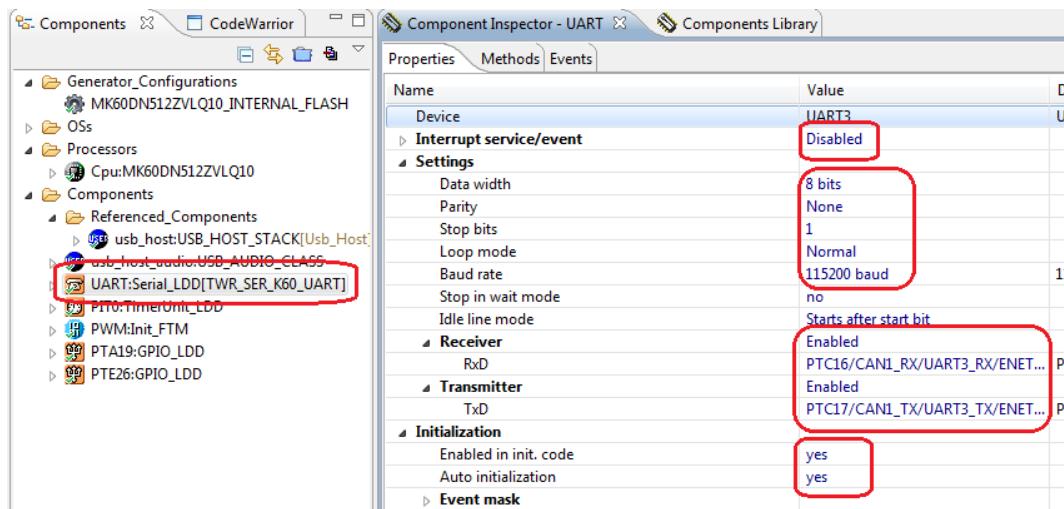


Figure 6-33 UART settings

**Step7.** Setup the TimerUnit\_LDD component for PIT0 as in the following figures:

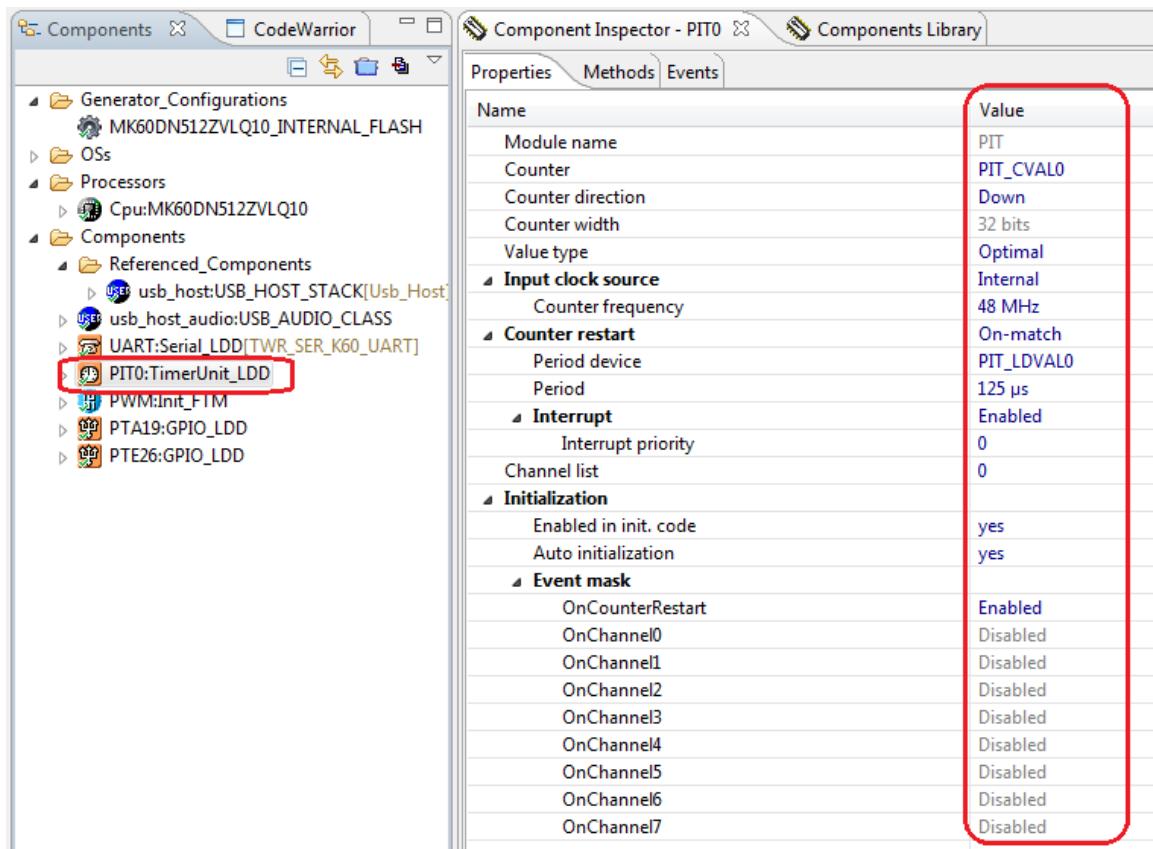


Figure 6-34 PIT 0 Properties Settings

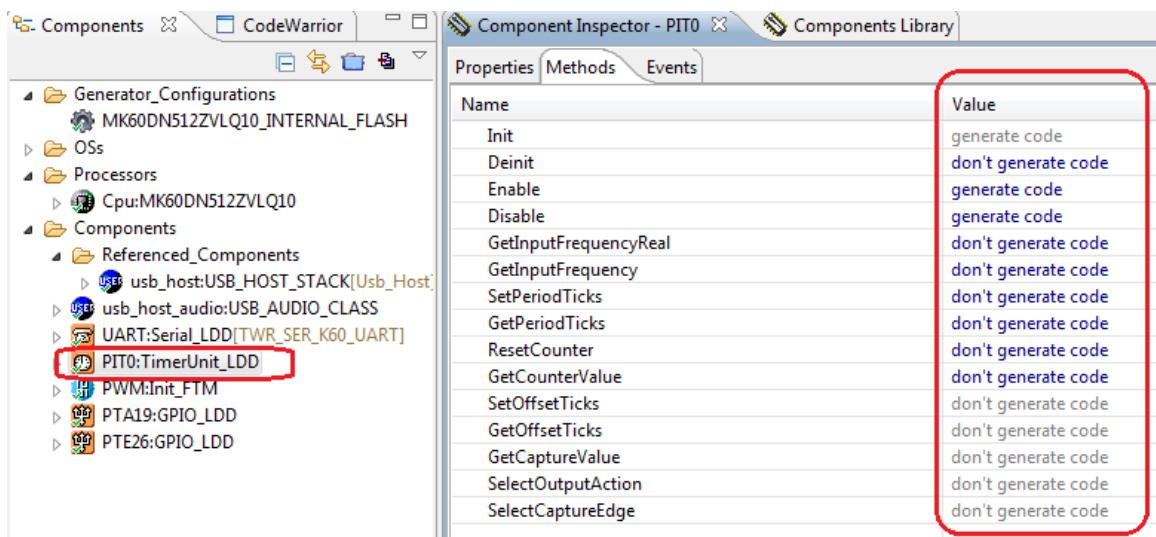


Figure 6-35 PIT 0 Methods Settings

**Step8.** Setup the Init\_FTM component for PWM as in the following figure:

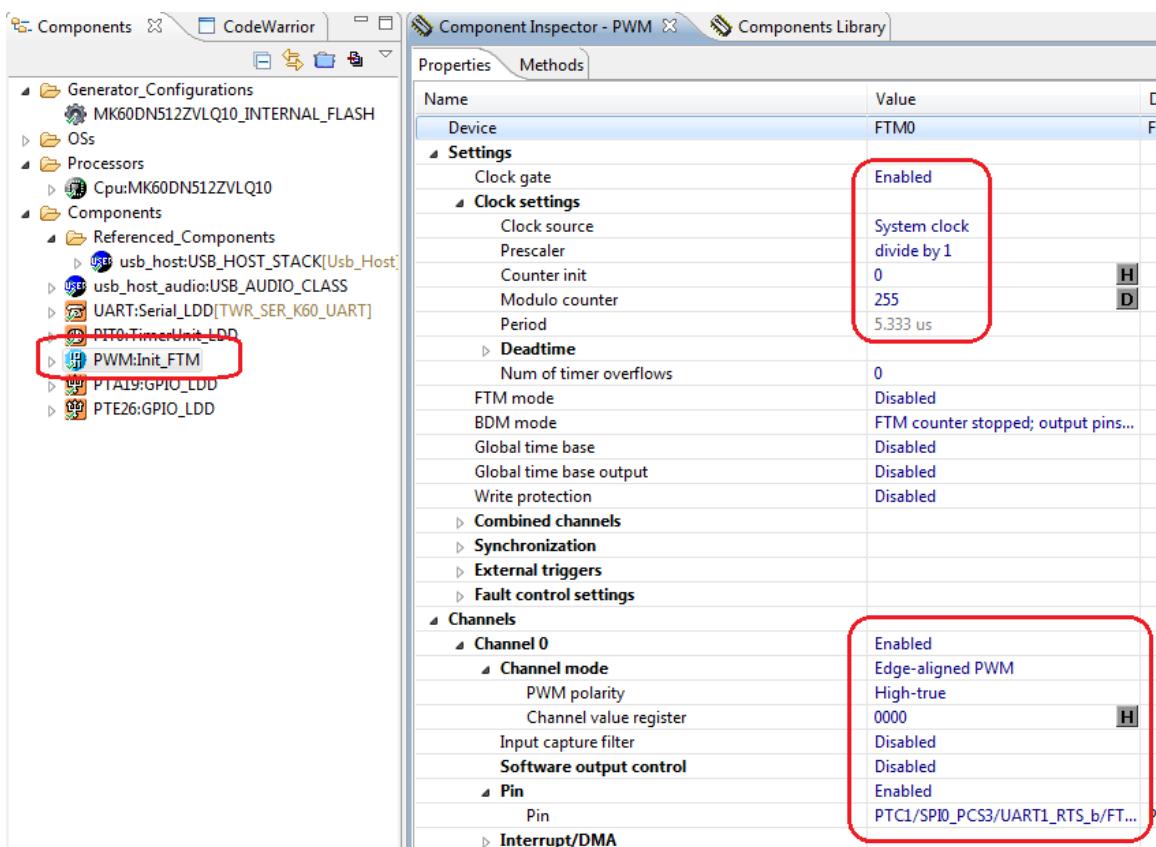


Figure 6-36 PWM settings

**Step9.** Setup the GPIO\_LDD component for PTA19 as in the following figures:

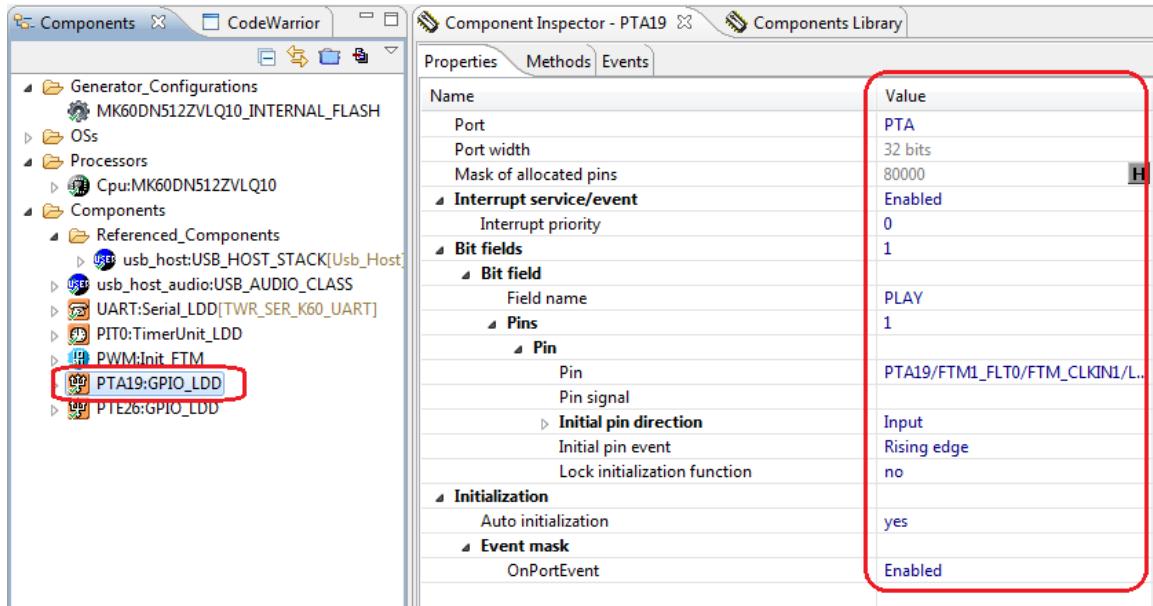


Figure 6-37 PTA19-“PLAY” Button Properties settings

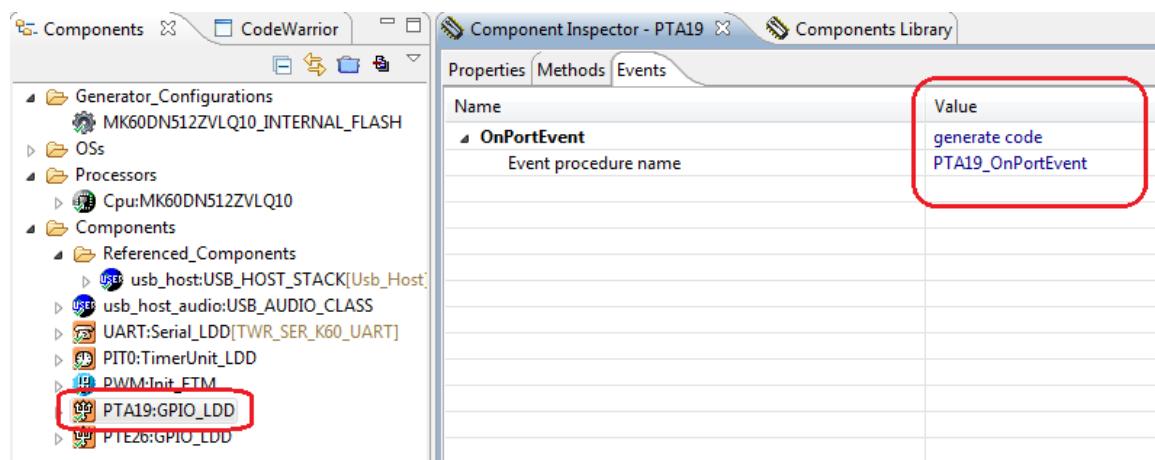


Figure 6-38 PTA19-“PLAY” Button Events settings

**Step9.** Setup the GPIO\_LDD component for PTE26 as in the following figures:

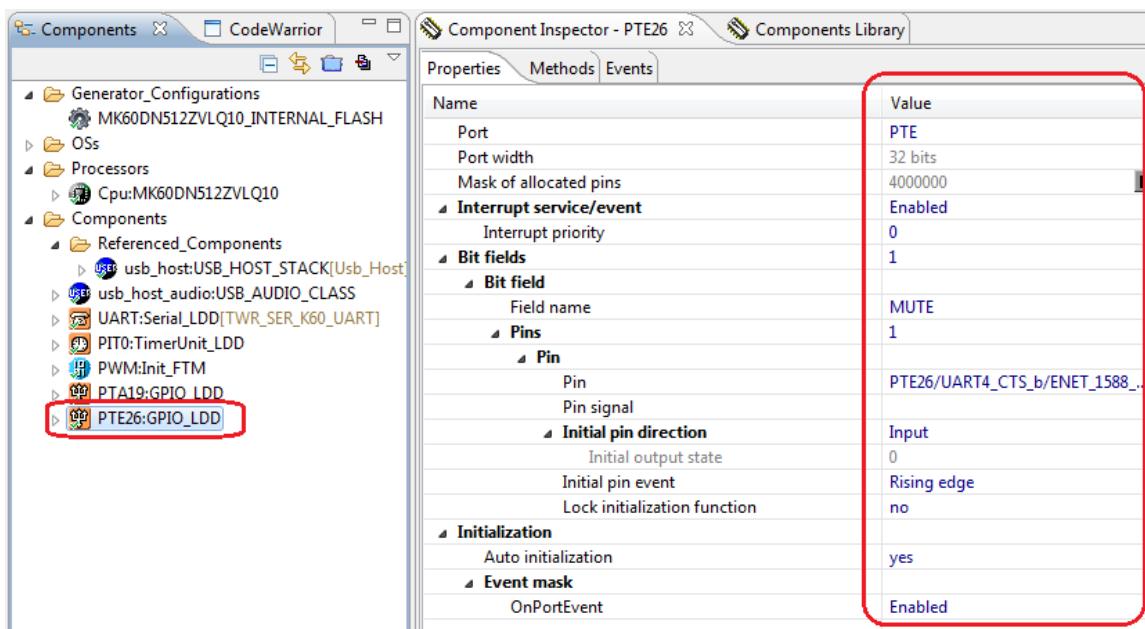


Figure 6-39 PTE26-“MUTE” button Properties settings

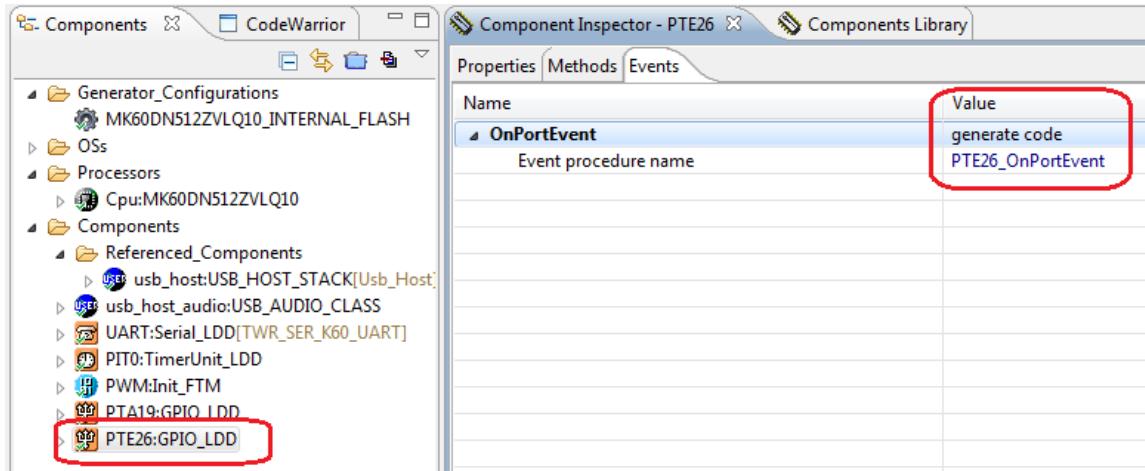


Figure 6-40 PTE26-“MUTE” button Events settings

**Step8.** Generate the AUDIO host source code same as the AUDIO device example above.

### 6.3.3.2 Running AUDIO host example

Implement the following steps to run the AUDIO host example:

**Step1.** Setup the hardware to run example.

- The computer uses one USB cable to supply power to the board and program USB AUDIO host image to the flash. One COM cable is used to get events from the USB AUDIO host.
- The USB AUDIO device is plugged to the USB AUDIO host by one USB cable.

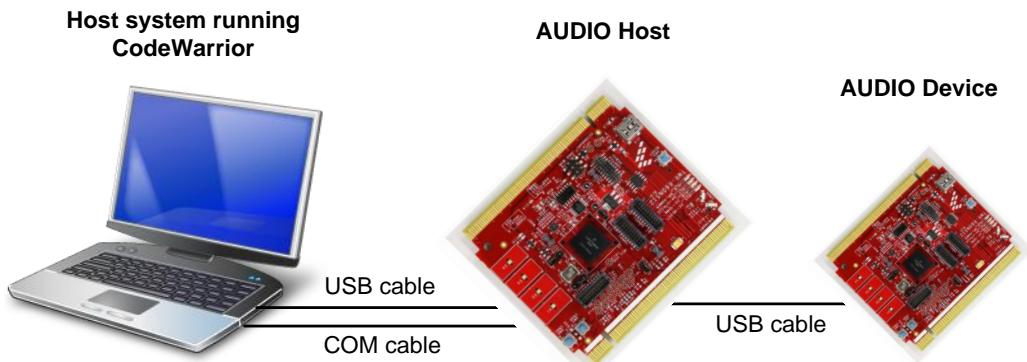


Figure 6-41 Hardware setup

**Step2.** To verify USB AUDIO host operating, The HyperTerminal is used and configured same as the USB AUDIO device example with baud rate is 115200.

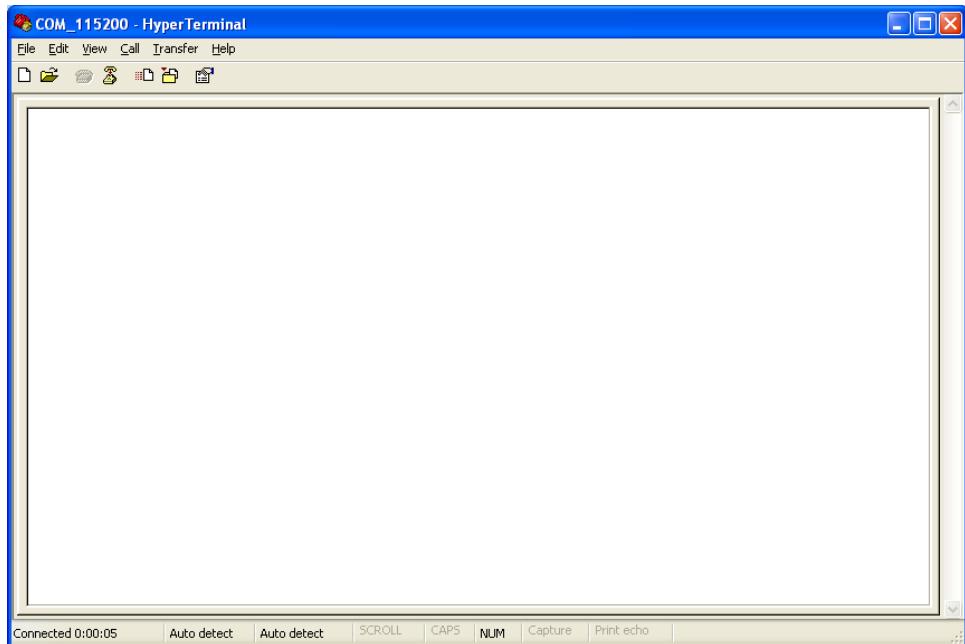


Figure 6-42 Hyper Terminal

**Step3.** After source code has been generated, we must add code to register application callback function into the "**device\_driver\_info.c**" file.

- Include the application:

```
/* Write your own includes here ... */
#include "audio.h"
```

- The "**usb\_host\_audio\_control\_event**" function:

```
void usb_host_audio_control_event
(
    /* [IN] pointer to device instance */
    _usb_device_instance_handle      dev_handle,
    /* [IN] pointer to interface descriptor */
```

```

    _usb_interface_descriptor_handle intf_handle,
/* [IN] code number for event causing callback */
    uint_32                                event_code
)
{
/* End code block <2>, DO NOT MODIFY LINES ABOVE */
/* Write your own code of usb_host_audio_control_event function
below */
    usb_host_audio_control_app_event(dev_handle,intf_handle,event_code);
/* End of code block <3>, DO NOT MODIFY THE LINE BELOW */
} /* End of usb_host_audio_control_event function */

```

And the “***usb\_host\_audio\_stream\_event***” function:

```

void usb_host_audio_stream_event
(
/* [IN] pointer to device instance */
    _usb_device_instance_handle      dev_handle,
/* [IN] pointer to interface descriptor */
    _usb_interface_descriptor_handle intf_handle,
/* [IN] code number for event causing callback */
    uint_32                                event_code
)
{
/* End code block <4>, DO NOT MODIFY LINES ABOVE */
/* Write your own code of usb_host_audio_stream_event function below
*/
    usb_host_audio_stream_app_event(dev_handle,intf_handle,
event_code);
/* End of code block <5>, DO NOT MODIFY THE LINE BELOW */
} /* End of usb_host_audio_stream_event function */

```

**Step4.** Build and load the image of AUDIO host application to the Kinetis K60 Tower board. To ensure that application runs correctly, the HyperTerminal is used to transact with the AUDIO device. There are steps to configure HyperTerminal.

- Open HyperTerminal application as the following figure:



Figure 6-43 Launch HyperTerminal application

- The HyperTerminal opens as figure below. Enter the name of the connection and click on the OK button.

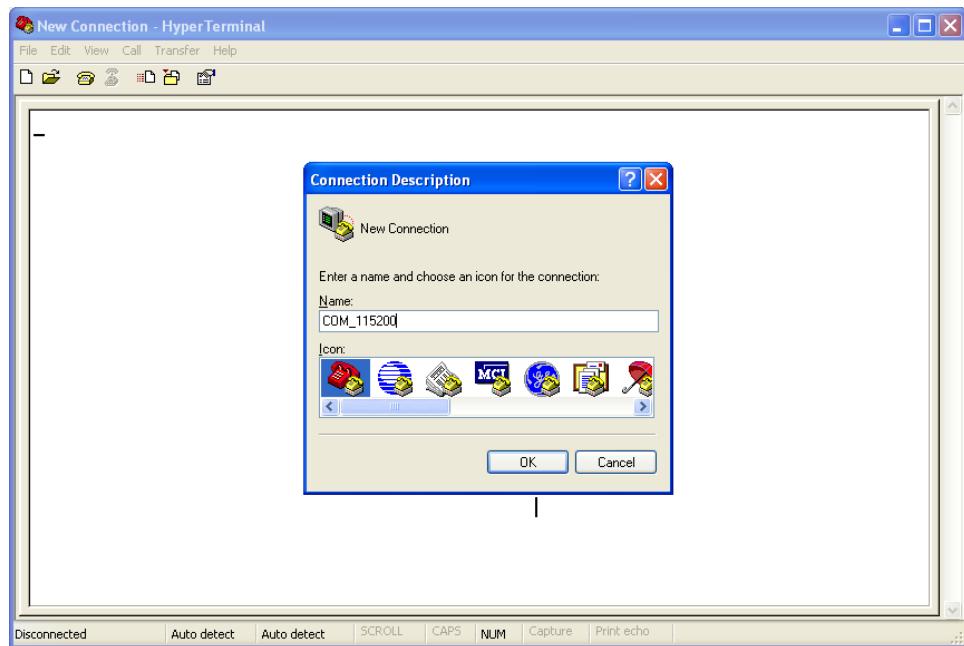


Figure 6-44 HyperTerminal startup

- The window shown in the following figure appears. Select the COM port.

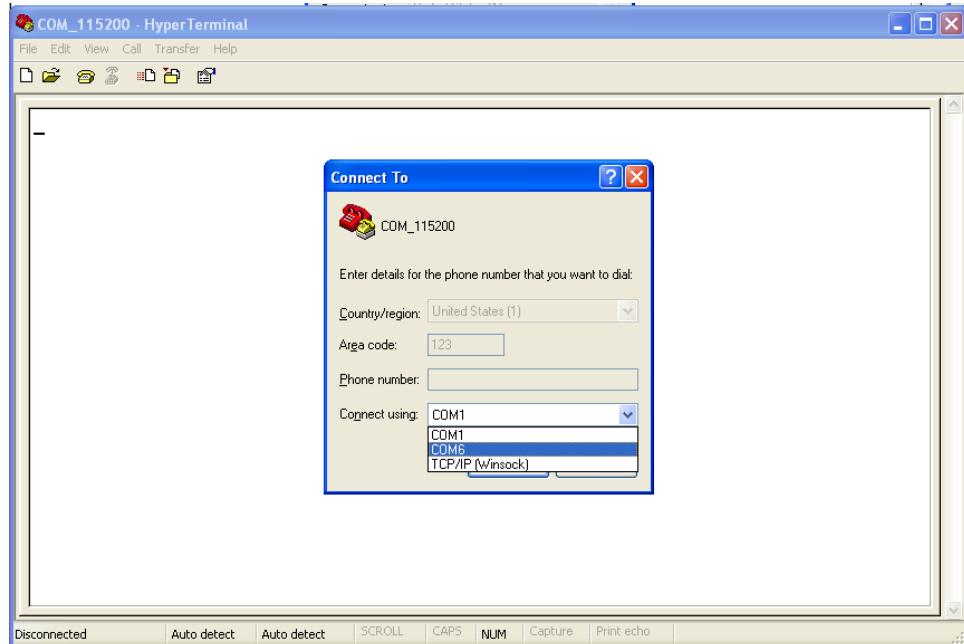


Figure 6-45 Connect using COM port

- Configure the COM port baud rate and other properties as in the figure below:

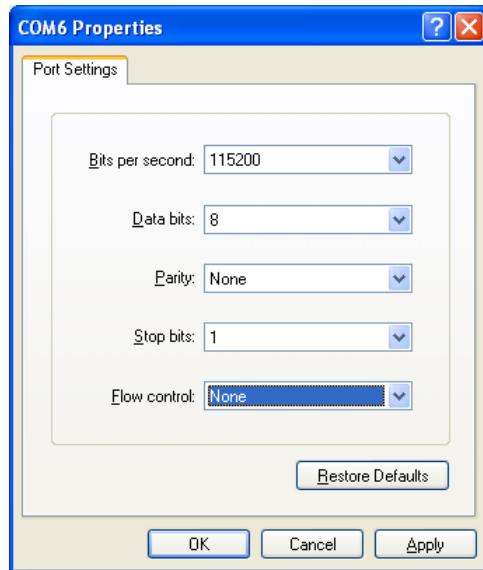


Figure 6-46 COM properties

- The HyperTerminal is now configured as shown in the figure below:

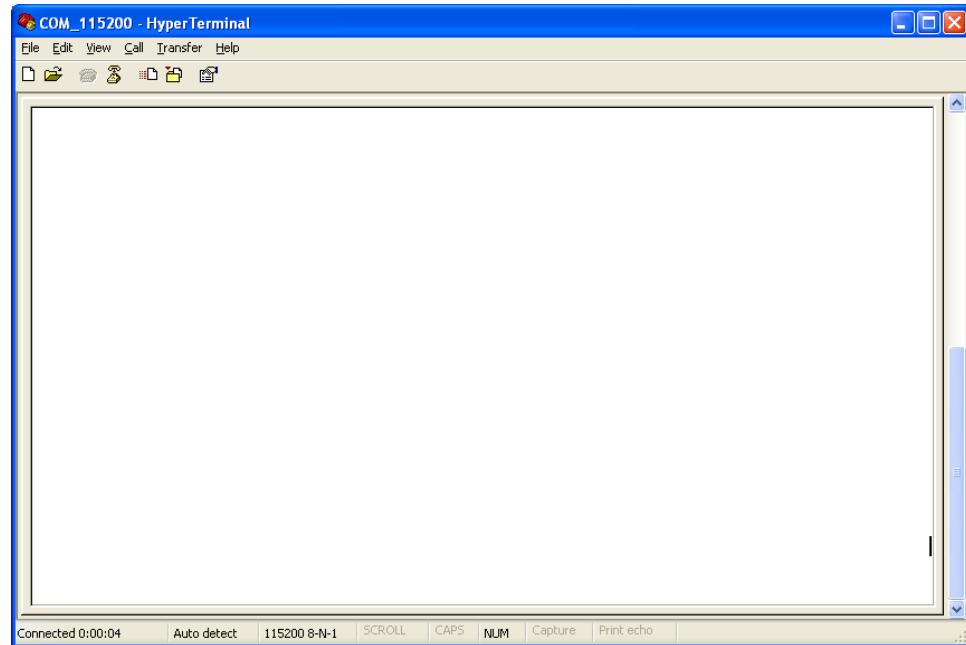


Figure 6-47 Hyper Terminal

The HyperTerminal shows the USB AUDIO host event as shown below:

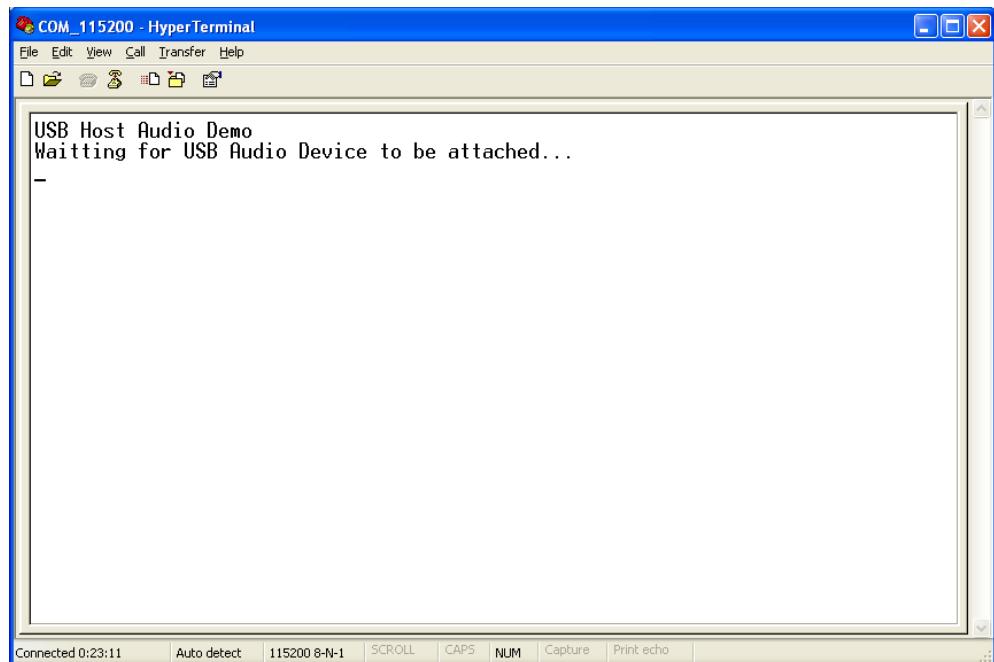


Figure 6-48 USB Host waiting for audio device attachment event

**Step5.** Plug the USB AUDIO device into the USB AUDIO host, the HyperTerminal shows the device information, if audio device is microphone (generator).

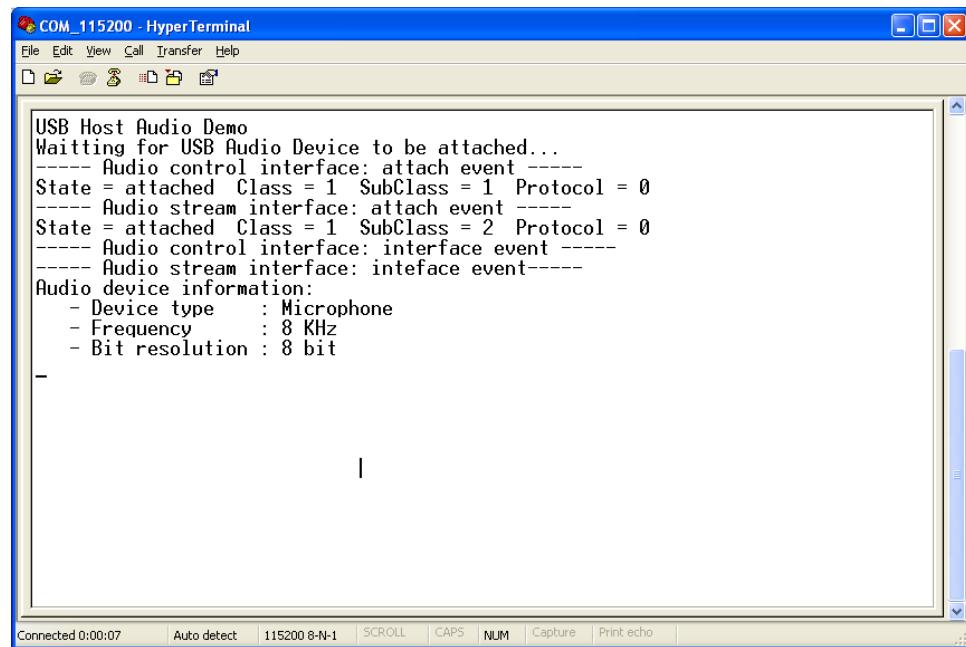


Figure 6-49 Attached device is Speaker type

**Step6.** Press *play* button to set Mute ON/OFF.

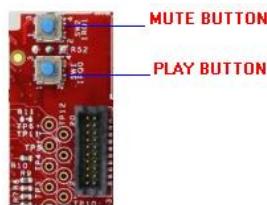


Figure 6-50 AUDIO functional button

```
USB Host Audio Demo
Waitting for USB Audio Device to be attached...
---- Audio control interface: attach event ----
State = attached Class = 1 SubClass = 1 Protocol = 0
---- Audio stream interface: attach event ----
State = attached Class = 1 SubClass = 2 Protocol = 0
---- Audio control interface: interface event ----
---- Audio stream interface: interface event ----
Audio device information:
- Device type : Microphone
- Frequency   : 8 KHz
- Bit resolution : 8 bit

Set Mute OFF
Set Mute successfully

Set Mute ON
Set Mute successfully
```

Figure 6-51 Set Mute ON/OFF

**Step7.** Press Switch 2 to Start/Stop transferring audio data stream between the Audio Host and the Audio Device.

```
USB Host Audio Demo
Waitting for USB Audio Device to be attached...
---- Audio control interface: attach event ----
State = attached Class = 1 SubClass = 1 Protocol = 0
---- Audio stream interface: attach event ----
State = attached Class = 1 SubClass = 2 Protocol = 0
---- Audio control interface: interface event ----
---- Audio stream interface: interface event ----
Audio device information:
- Device type : Microphone
- Frequency   : 8 KHz
- Bit resolution : 8 bit

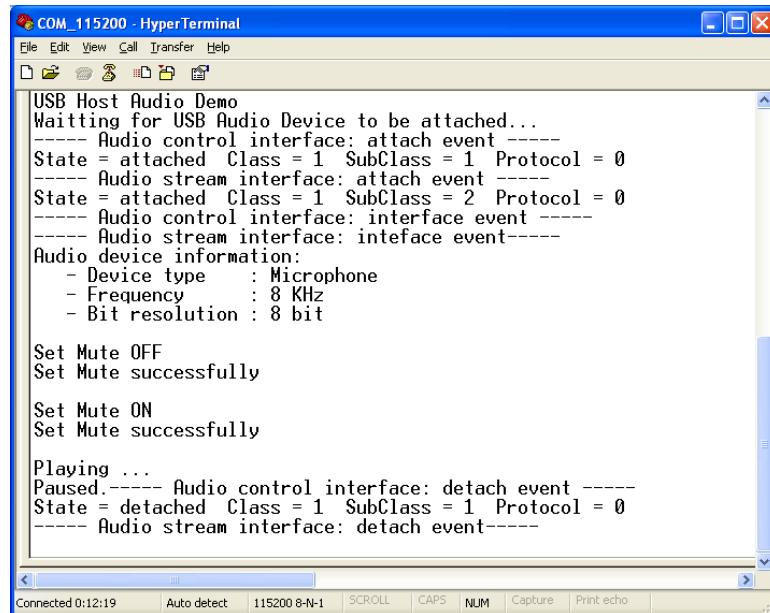
Set Mute OFF
Set Mute successfully

Set Mute ON
Set Mute successfully

Playing ...
Paused.
```

Figure 6-52 Start/Stop transferring audio data

**Step8.** Unplug the Audio Device.



The screenshot shows a window titled "COM\_115200 - HyperTerminal". The terminal window displays the following text:

```
USB Host Audio Demo
Waiting for USB Audio Device to be attached...
----- Audio control interface: attach event -----
State = attached Class = 1 SubClass = 1 Protocol = 0
----- Audio stream interface: attach event -----
State = attached Class = 1 SubClass = 2 Protocol = 0
----- Audio control interface: interface event -----
----- Audio stream interface: interface event -----
Audio device information:
- Device type : Microphone
- Frequency : 8 KHz
- Bit resolution : 8 bit

Set Mute OFF
Set Mute successfully

Set Mute ON
Set Mute successfully

Playing ...
Paused.----- Audio control interface: detach event -----
State = detached Class = 1 SubClass = 1 Protocol = 0
----- Audio stream interface: detach event -----
```

The status bar at the bottom of the terminal window shows "Connected 0:12:19" and other serial port parameters.

Figure 6-53 Audio Device detached

## 7 USB HID Device Example with MQX Lite

This section describes the steps to create an USB HID device example using USB Device stack and MQX Lite Processor Expert components.

### 7.1.1 Creating Processor Expert Project

This section describes the steps to create the Processor Expert project.

**Step1.** Open CodeWarrior Development Studio, chose Processor Expert → Import Package.

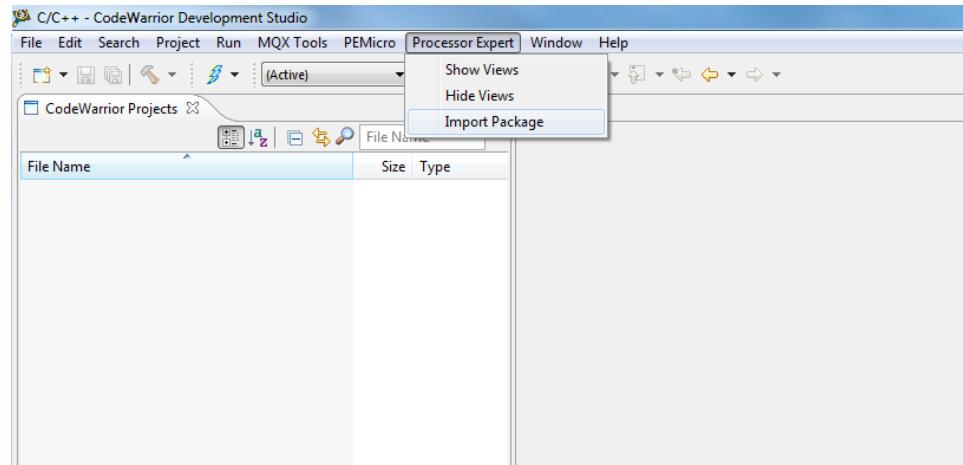


Figure 7-1 Processor Expert - Import package

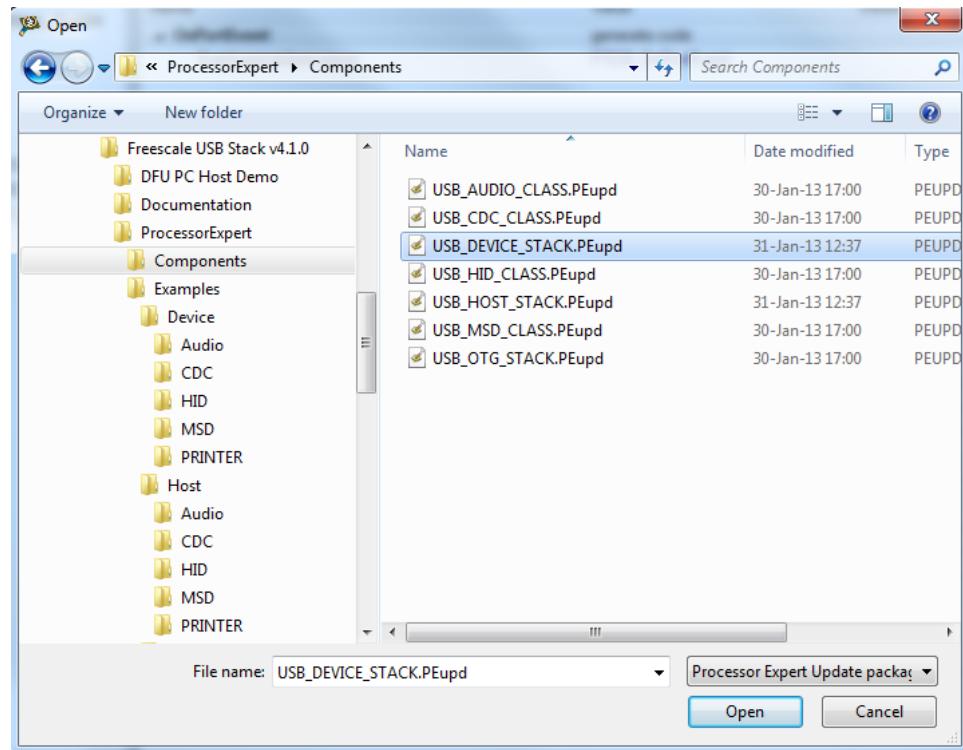


Figure 7-2 Select USB Device stack package

**Step2.** Create the Processor Expert project and add components to the project:

- Create a Processor Expert project, the project have the processor MKL25VLK4.
- Add the USB Device stack and the MQX Lite components from Component Library to the project.

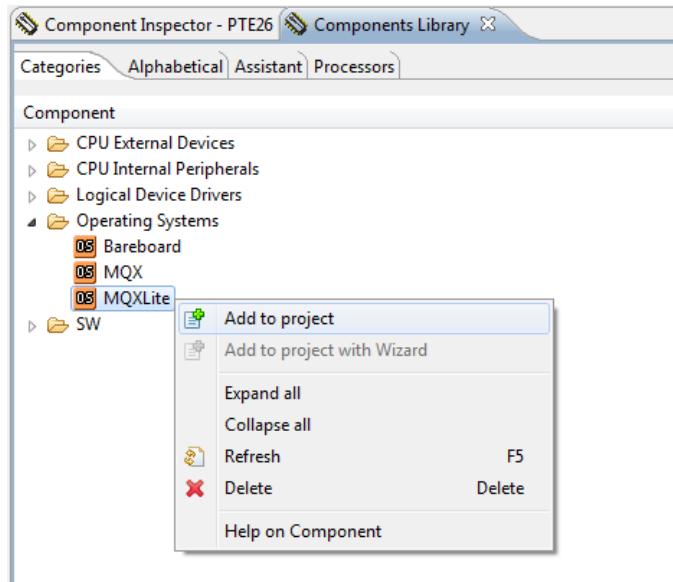


Figure 7-3 Add the MQX Lite component

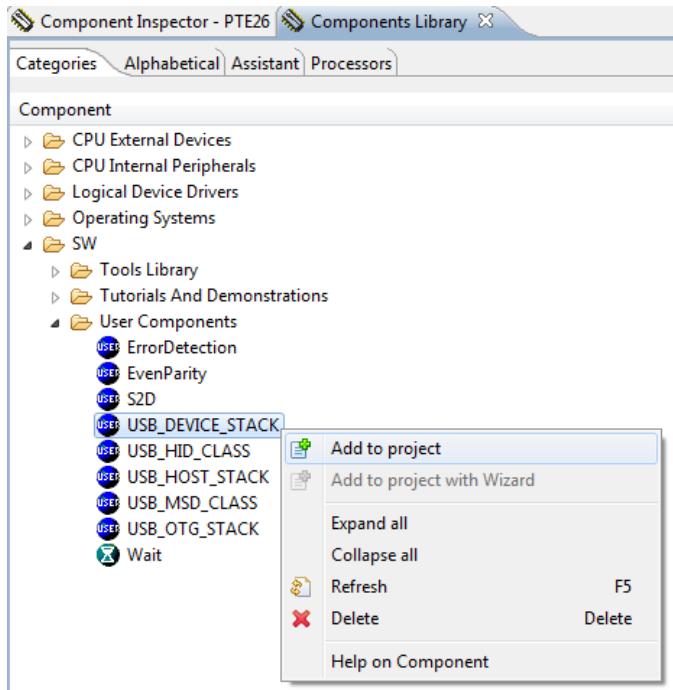


Figure 7-4 Add the USB Device stack component

## 7.1.2 Configure the Processor Expert Components

This section describes the steps to configure the Processor Expert components that you have added into the project.

### Step1. Configure the CPU:

- Configure clock to run USB module:
  - o CPU type: MKL25Z128VLK4
  - o Clock source: External crystal
  - o Clock frequency: 8.0MHz
  - o MCG mode: PEE
  - o MCG output: 96MHz
  - o PLL output: 96MHz

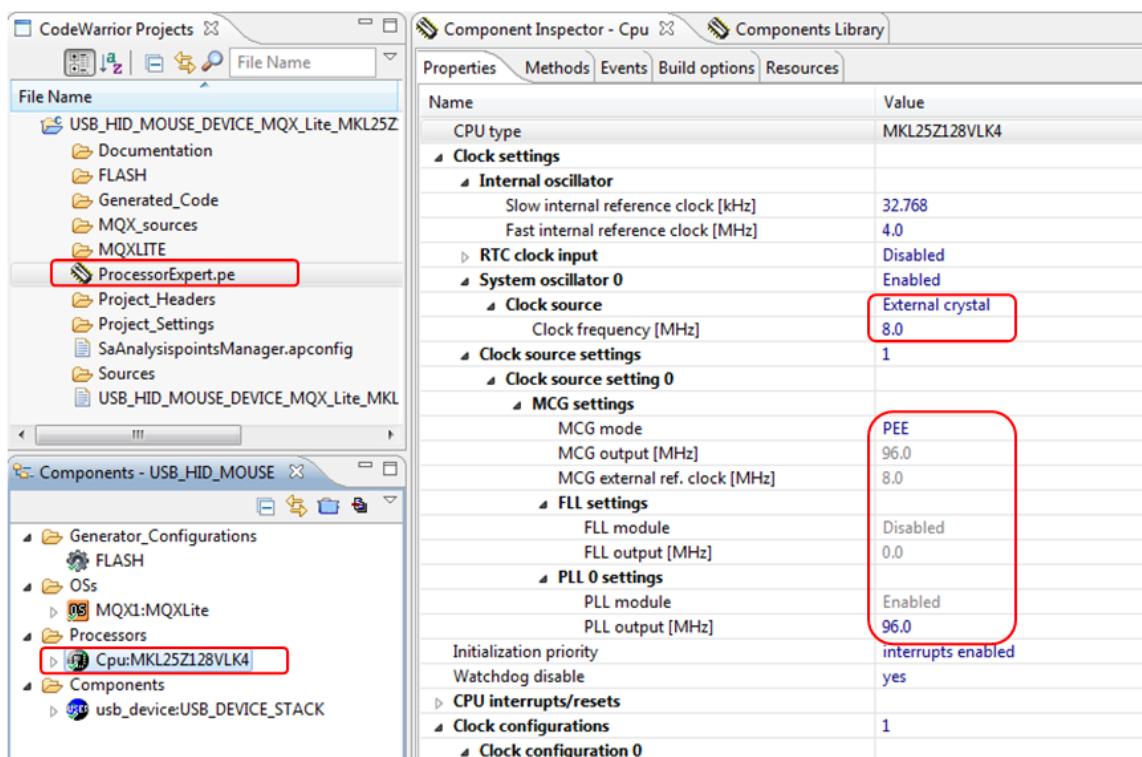


Figure 7-5 CPU Clock settings

- Core clock: 48MHz
- Bus clock: 24MHz

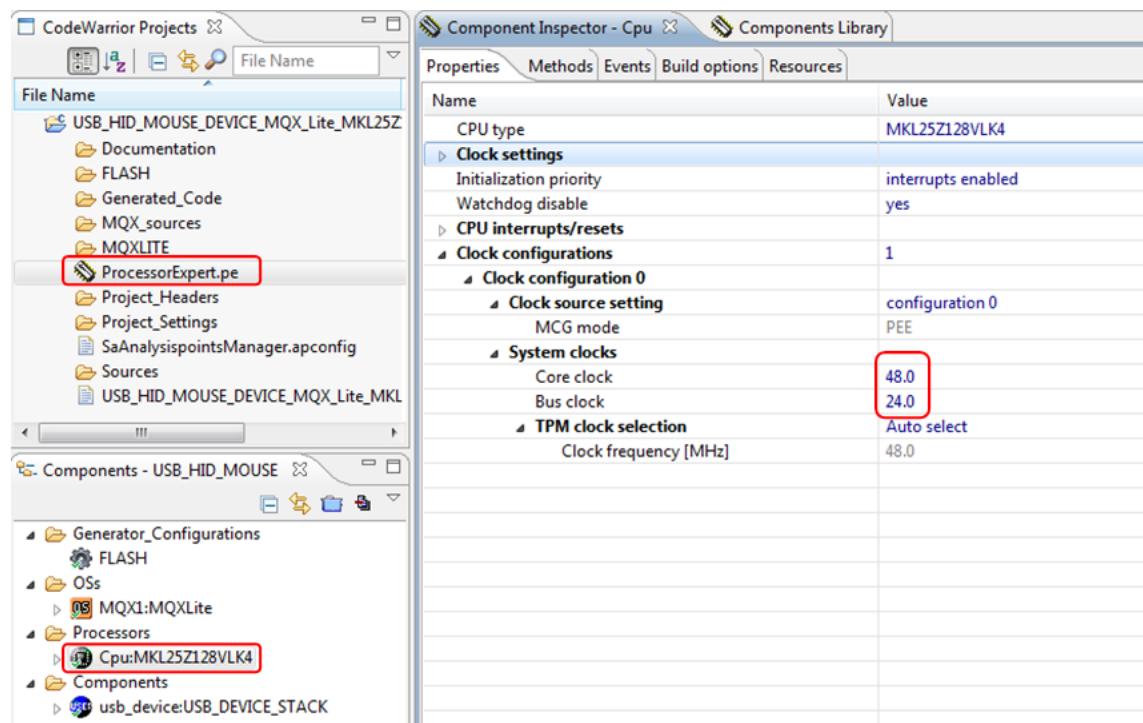


Figure 7-6 CPU Clock Configurations

### Step2 Configure the MQX Lite component:

- Configure the System timer:
  - Input clock source: Internal
  - Clock frequency: 48 MHz

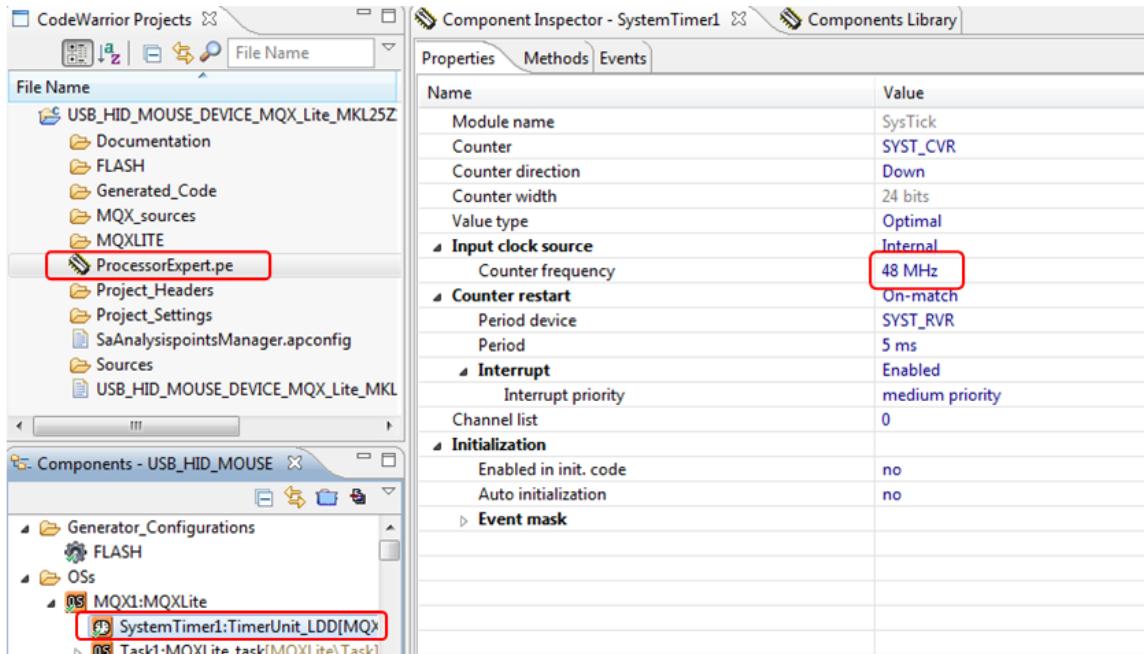


Figure 7-7 System Timer Configurations

- Configure the Tasks:
  - o Name: usb\_task
  - o Entry point function: usb\_task

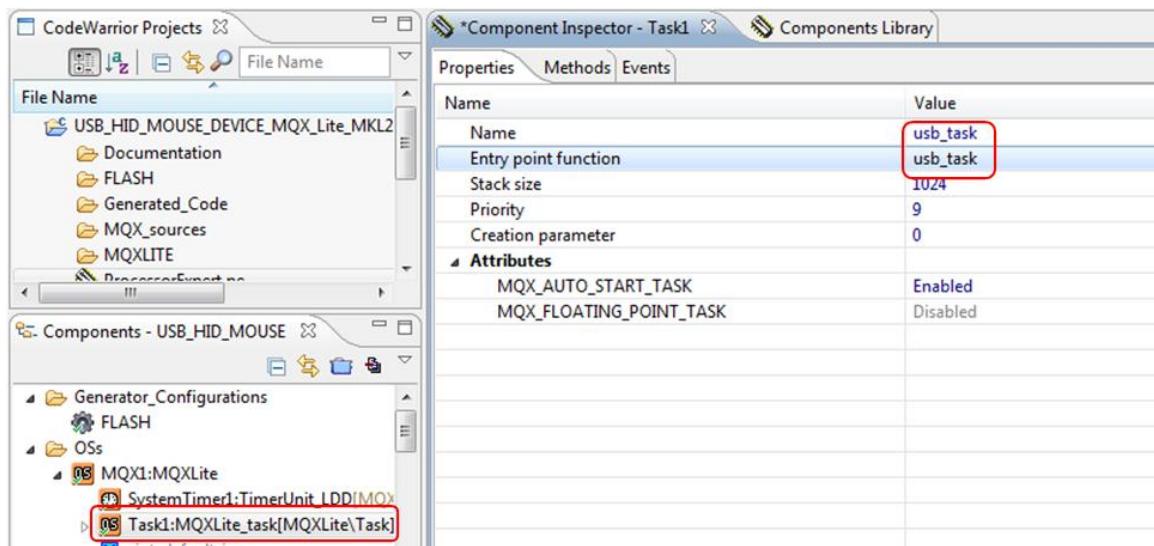


Figure 7-8 Task Configuration

### Step3. Configure the USB device stack component:

- Configure the Device descriptor:

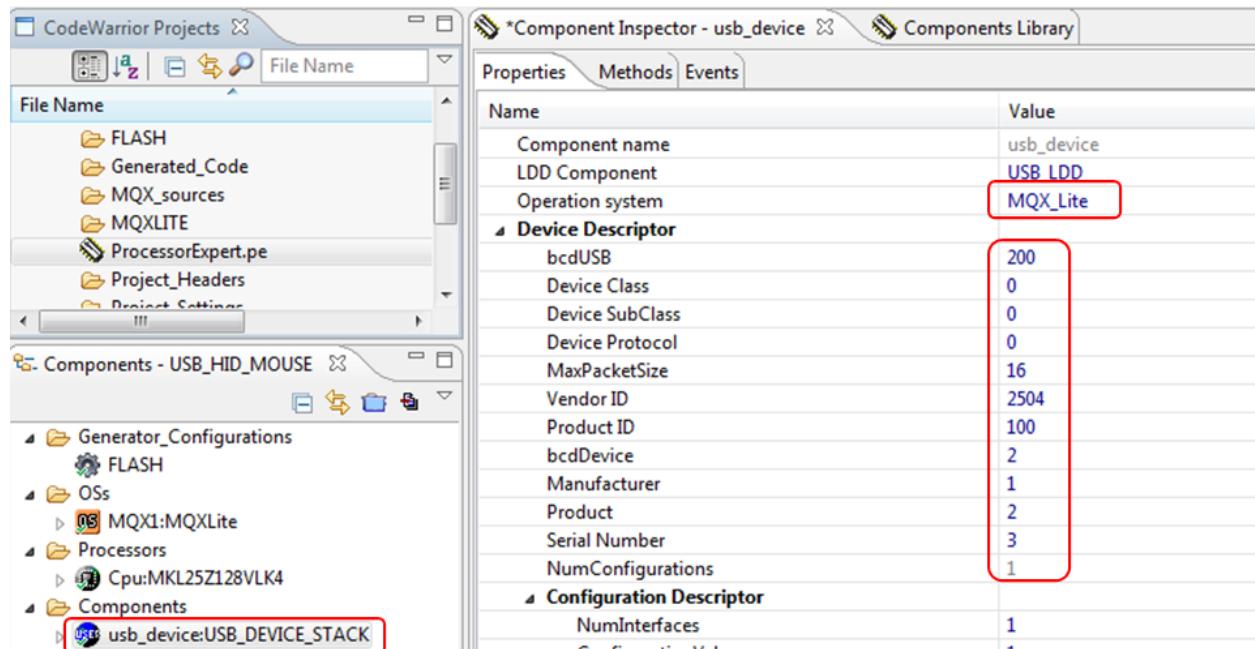


Figure 7-9 Device Descriptor configuration

- Configure the Configuration Descriptor:

- o Click on the Interfaces List and click plus to add an interface to the configuration.
- o Click on the Endpoints List and click plus to add an endpoint to the interface.

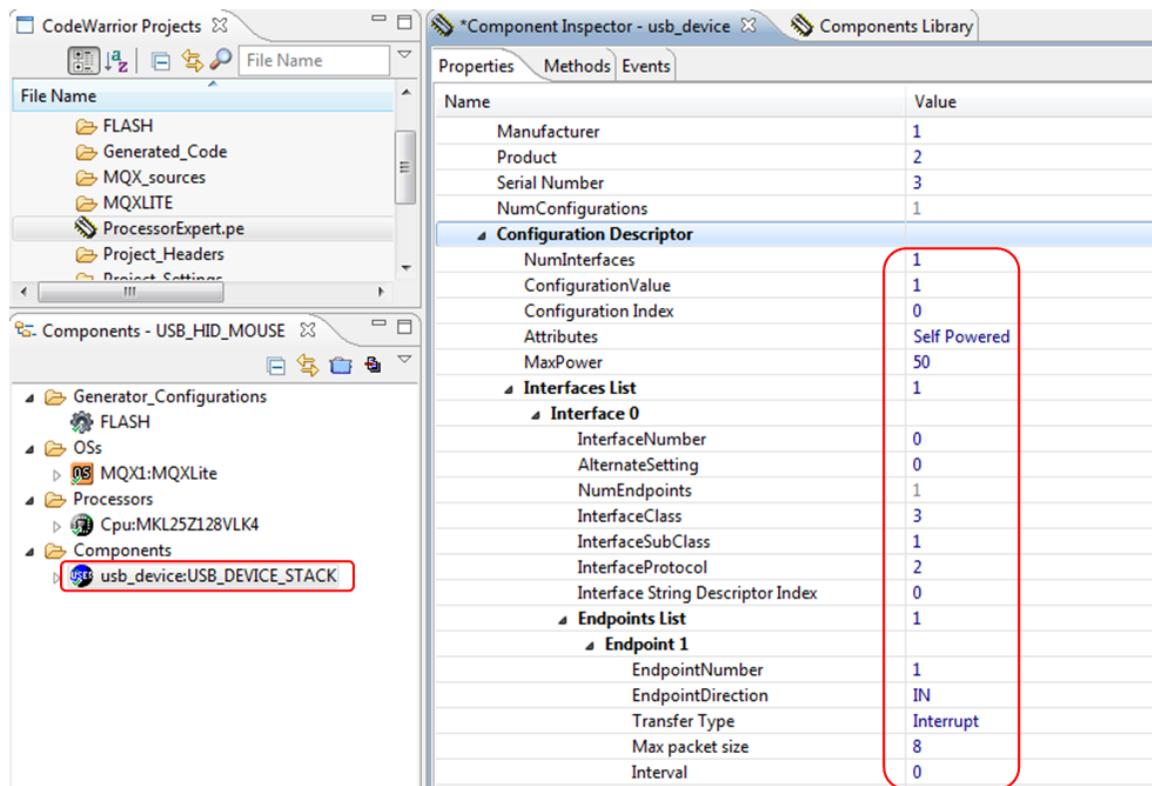


Figure 7-10 Configuration Descriptor configuration

- Configure the String Descriptor:

- o Click on the String Descriptor and click plus to add an string to the string descriptor.

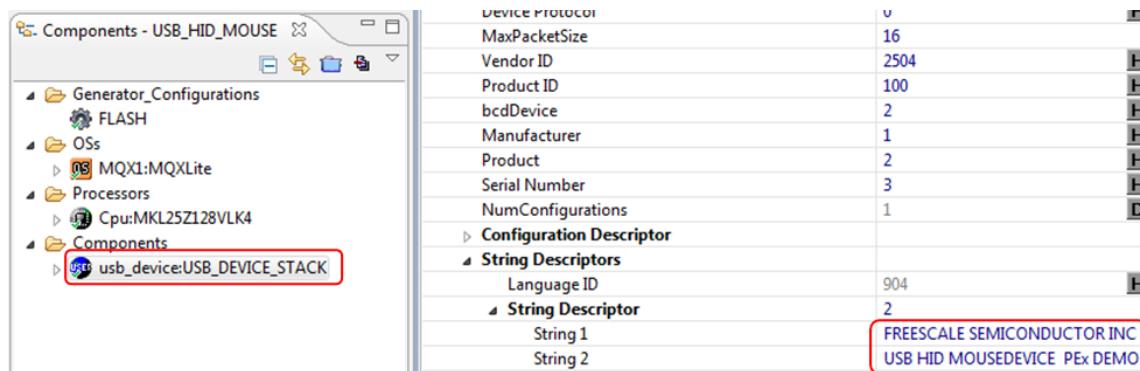


Figure 7-11 String Descriptor configuration

- Configure the USB\_LDD :
  - o Enable the EP1, enable Interrupt IN transfer, Max. packet size 8, Max. queue size 1.

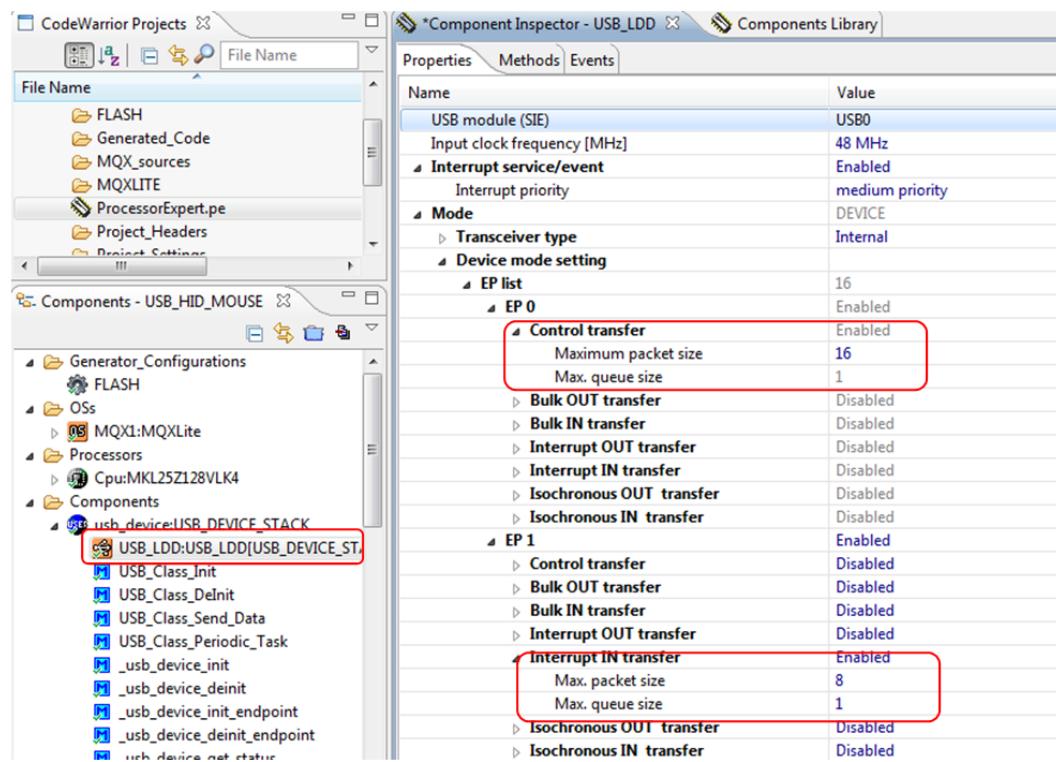


Figure 7-12 USB\_LLD configuration

#### Step4. Generate the USB\_HID\_CLASS source code:

- On components window, click “generate processor expert code” button, the Processor Expert will generate the MQX lite and the USB device stack code for you.

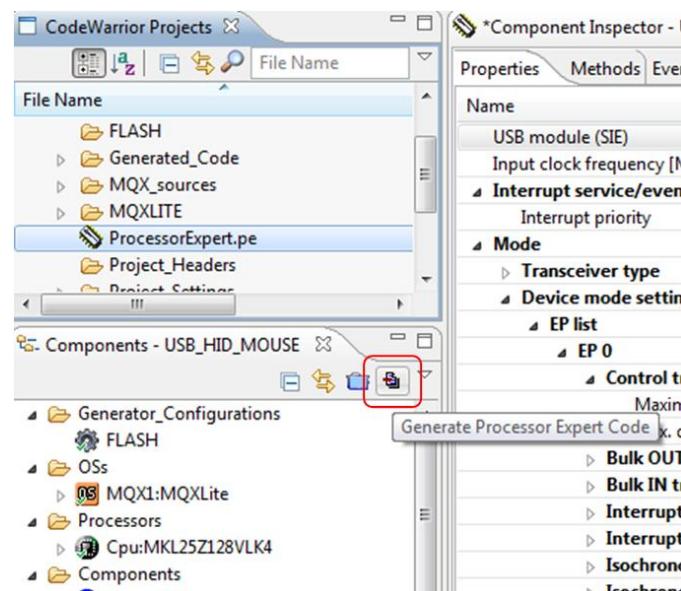


Figure 7-13 Generate Processor Expert Code

### 7.1.3 Edit User Code and Add Application Code

- Copy the USB HID mouse device application code and the USB HID Class code to the source folder of your project. That code is in the location:

\USB\_HID\_MOUSE\_DEVICE\_MQX\_Lite\_MKL25Z128\_PEx\Sources\USB\app\hid

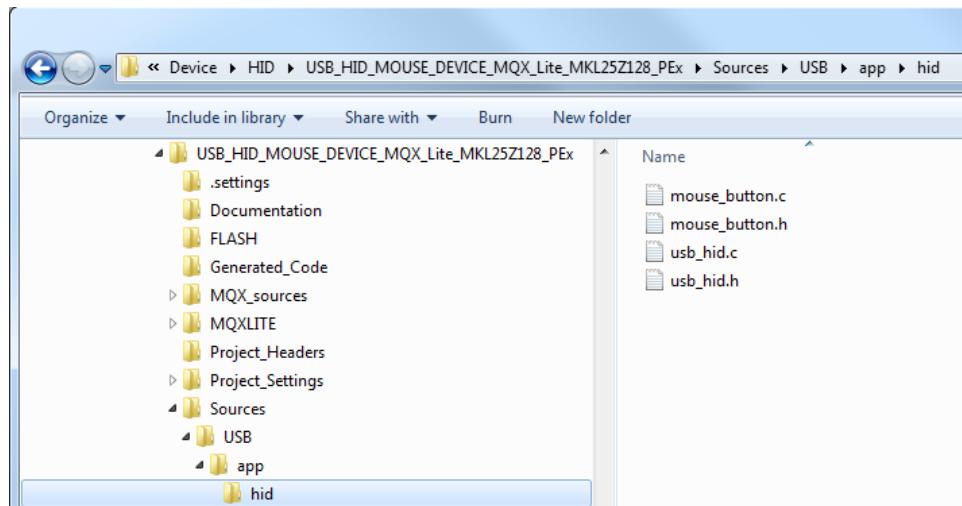


Figure 7-14 The USB Mouse Device and the HID Class code

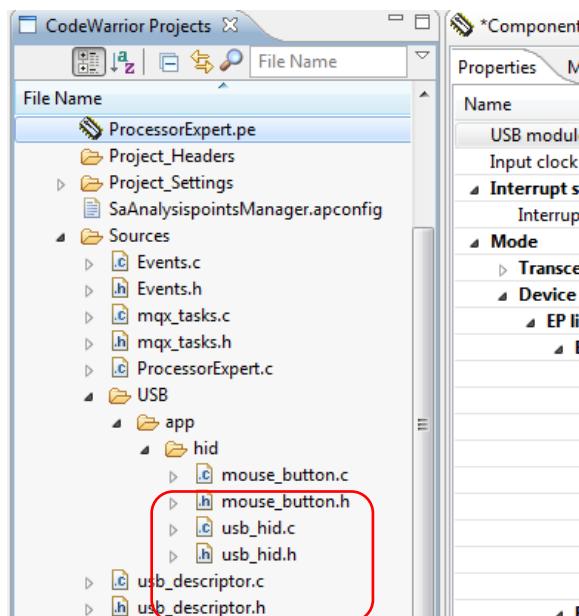


Figure 7-15 Add the USB Mouse Device and the HID Class to the project

- Edit the “**usb\_descriptor.c**”:

- o Edit the “**g\_config\_descriptor**” array:

```
/* End code block <1> auto generated code, DO NOT MODIFY LINES ABOVE */

/* Write your own specific descriptor code here ... */
/* HID descriptor */
HID_ONLY_DESC_SIZE,
USB_HID_DESCRIPTOR,
0x00,0x01,
```

```

0x00,
0x01,
0x22,
0x34,0x00,           /* report descriptor size to follow */

/* Begin of code block <2> auto generated code, DO NOT MODIFY LINES BELOW */

```

- Write the “***g\_report\_descriptor***” array:

```
/* End code block <2> auto generated code, DO NOT MODIFY LINES ABOVE */
```

```

/* Write your own specific descriptor struct here ... */
uint_8 g_report_descriptor[REPORT_DESC_SIZE] =
{
    0x05,0x01,
    0x09,0x02,
    0xA1,0x01,
    0x09,0x01,
    0xA1,0x00,
    0x05,0x09,
    0x19,0x01,
    0x29,0x03,
    0x15,0x00,
    0x25,0x01,
    0x95,0x03,
    0x75,0x01,
    0x81,0x02,
    0x95,0x01,
    0x75,0x05,
    0x81,0x01,
    0x05,0x01,
    0x09,0x30,
    0x09,0x31,
    0x09,0x38,
    0x15,0x81,
    0x25,0x7F,
    0x75,0x08,
    0x95,0x03,
    0x81,0x06,
    0xC0,
    0xC0,
};

/* Begin of code block <3> auto generated code, DO NOT MODIFY LINES BELOW */

```

- Edit the “***g\_std\_desc\_size***” array:

```
/* End code block <3> auto generated code, DO NOT MODIFY LINES ABOVE */
```

```

/* Write your own specific descriptor size here ... */
        ,REPORT_DESC_SIZE
/* Begin of code block <4> auto generated code, DO NOT MODIFY LINES BELOW */

```

- Edit the “***g\_std\_descriptors***” array:

```
/* End code block <4> auto generated code, DO NOT MODIFY LINES ABOVE */
```

```

/* Write your own specific descriptor pointer struct here ... */

        ,(uint_8_ptr)g_report_descriptor
/* Begin of code block <5> auto generated code, DO NOT MODIFY LINES BELOW */

```

- Edit the “***USB\_Desc\_Get\_Descriptor***” function:

```
/* End code block <6> auto generated code, DO NOT MODIFY LINES ABOVE */
```

```
/* Write your own code here ... */
```

```

        case USB_REPORT_DESCRIPTOR:
    {
        type = USB_MAX_STD_DESCRIPTORS;
        *descriptor = (uint_8_ptr)g_std_descriptors [type];
        *size = g_std_desc_size[type];
    }
    break;
    case USB_HID_DESCRIPTOR:
    {
        type = USB_CONFIG_DESCRIPTOR ;
        *descriptor = (uint_8_ptr)(g_std_descriptors [type]+
                                CONFIG_ONLY_DESC_SIZE+IFACE_ONLY_DESC_SIZE);
        *size = HID_ONLY_DESC_SIZE;
    }
    break;

/* Begin of code block <7> auto generated code, DO NOT MODIFY LINES BELOW */

```

- Edit the “***usb\_descriptor.h***:

- o Write the macros:

```

/* End code block <1> auto generated code, DO NOT MODIFY LINES ABOVE */

/* Write code here ... */

/* redefine the USER_DESC_SIZE parameter here
 * example
 * #undef USER_DESC_SIZE
 * #define USER_DESC_SIZE           (10)
 */
#undef USER_DESC_SIZE
#define USER_DESC_SIZE           (9)
/* Add your custom macro here*/
#define CONFIG_DESC_SIZE          (34)
#define REMOTE_WAKEUP_SUPPORT     (1) /*1: TRUE, 0: FALSE*/
#define REMOTE_WAKEUP_SHIFT        (5)
#define REPORT_DESC_SIZE          (52)
#define HID_ONLY_DESC_SIZE         (9)
#define USB_HID_DESCRIPTOR        (0x21)
#define USB_REPORT_DESCRIPTOR      (0x22)
#define HID_DESC_ENDPOINT_COUNT   (1)
#define HID_ENDPOINT               (1)
#define HID_ENDPOINT_PACKET_SIZE  (8)

/* Begin of code block <2> auto generated code, DO NOT MODIFY LINES BELOW */

```

#### 7.1.4 Running the USB HID Mouse Device Example

This section describes the steps to run the USB HID mouse device example.

**Step1.** Setup the hardware as in the following figure:

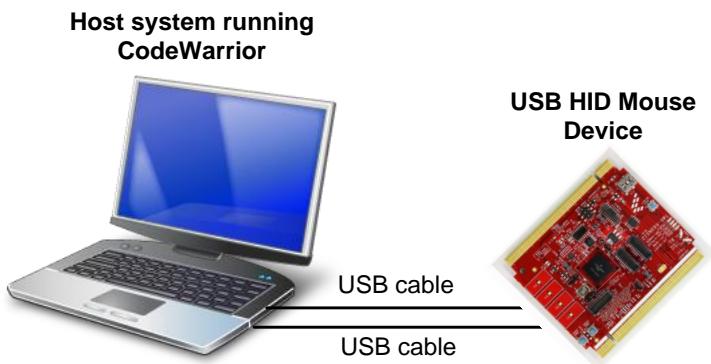


Figure 7-16 HID mouse device demo setup

**Step2.** After generating Processor Expert source code is completed, build and program the USB HID mouse device image to the flash.

**Step3.** Connect the USB HID mouse device to the PC by one USB cable. After the USB HID mouse device completed the enumeration, the mouse cursor will move around on the screen.