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 20th, 2013

How to Reach Us:

Home Page:
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

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

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

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Rev. 09
03/2013

Table of Contents

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.

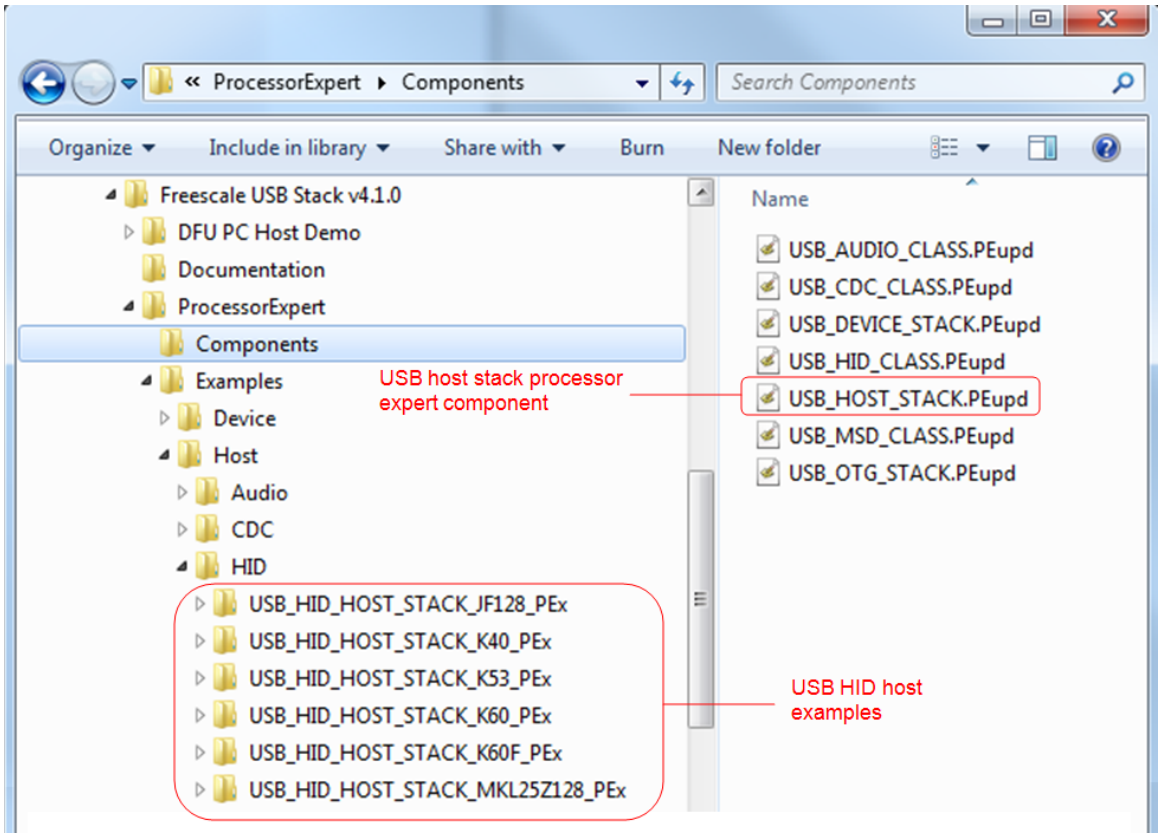


Figure 1-1 HID Mouse Host folder structure

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.

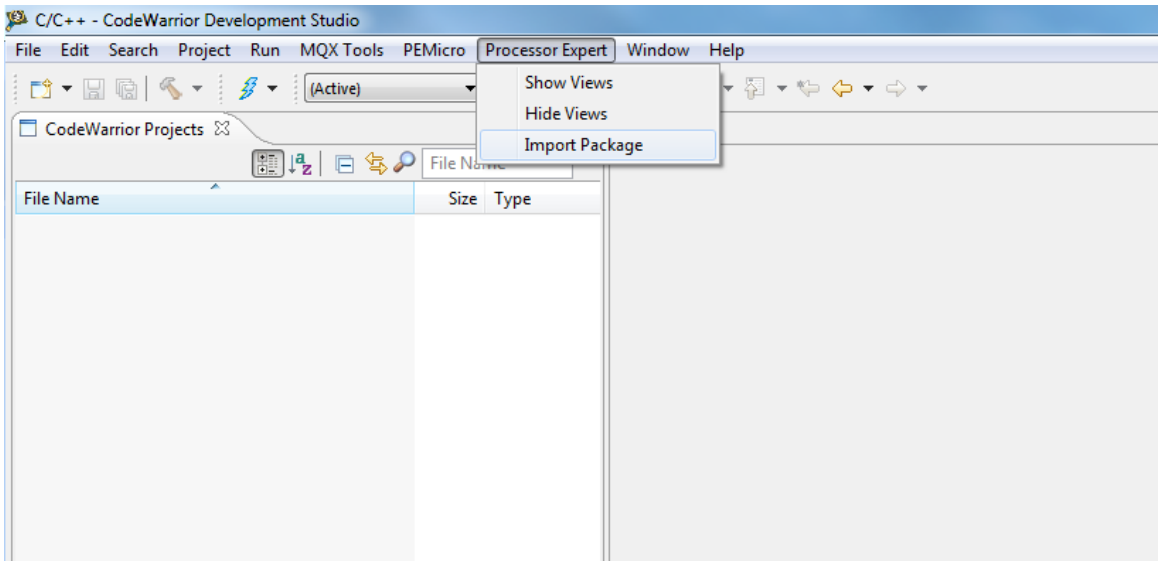


Figure 1-2 Processor Expert – Import Package

Step2. A browser window appears. Navigate to the folder that contains the USB_HOST_STACK component, chose it and click open.

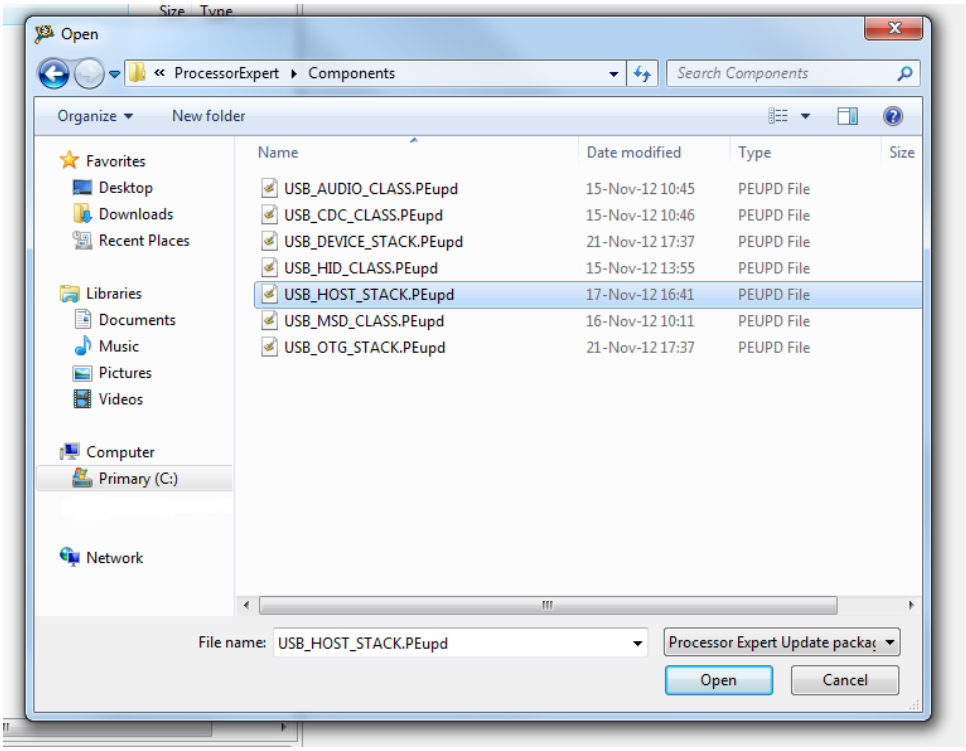


Figure 1-3 Delivered package

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

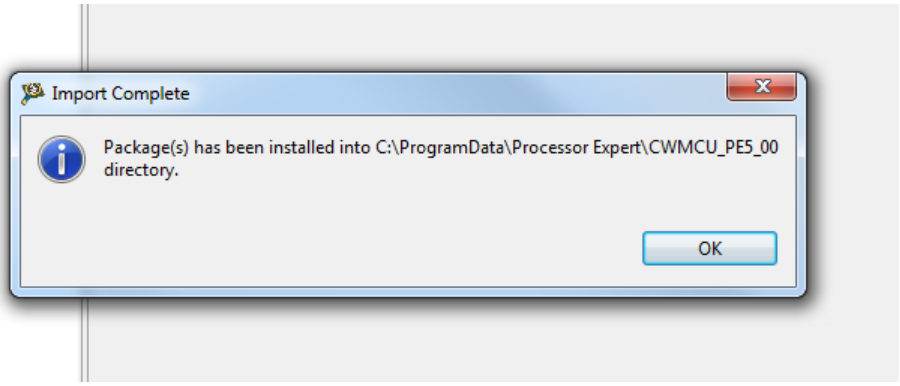


Figure 1-4 Import Complete

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.

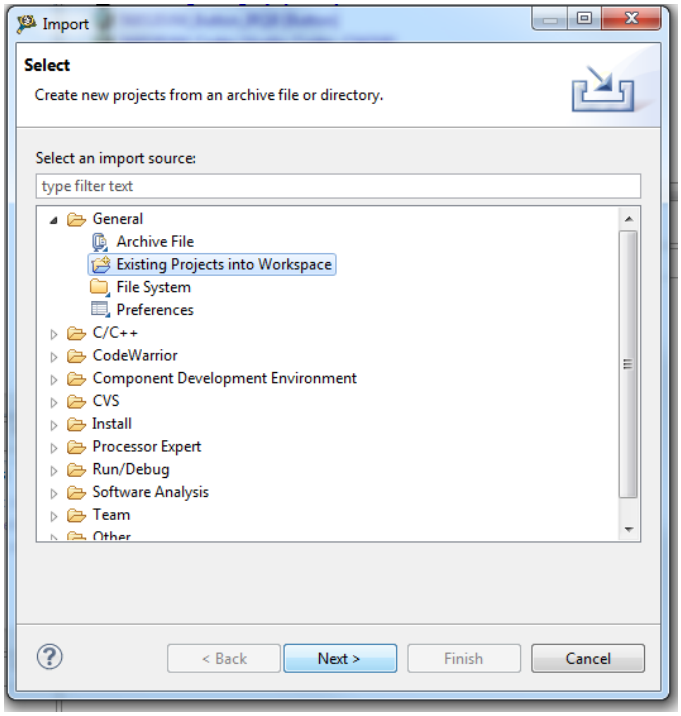


Figure 1-5 Import existing projects into Workspace

Step2. Click Browse button → Select USB_HID_HOST_STACK_K60_PEx folder -> Click OK -> then click Finish button.

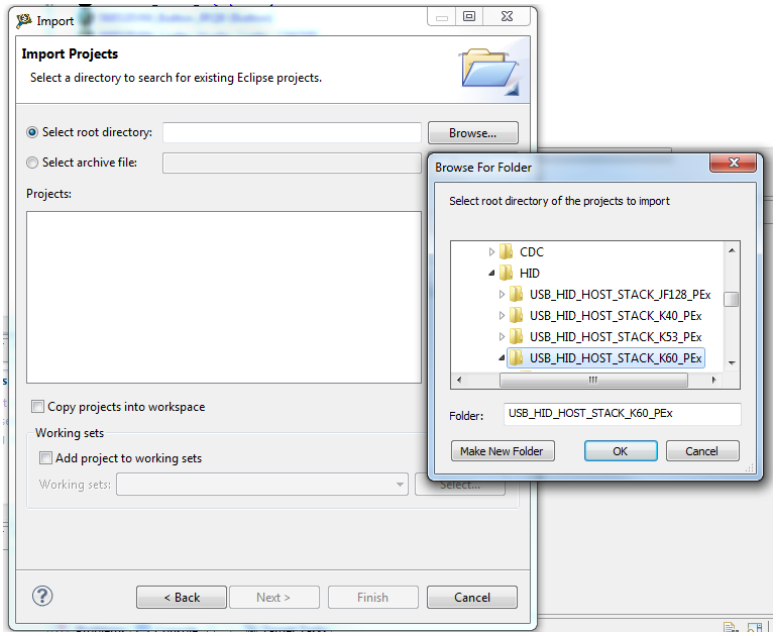


Figure 1-6 Import existing projects into Workspace

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.

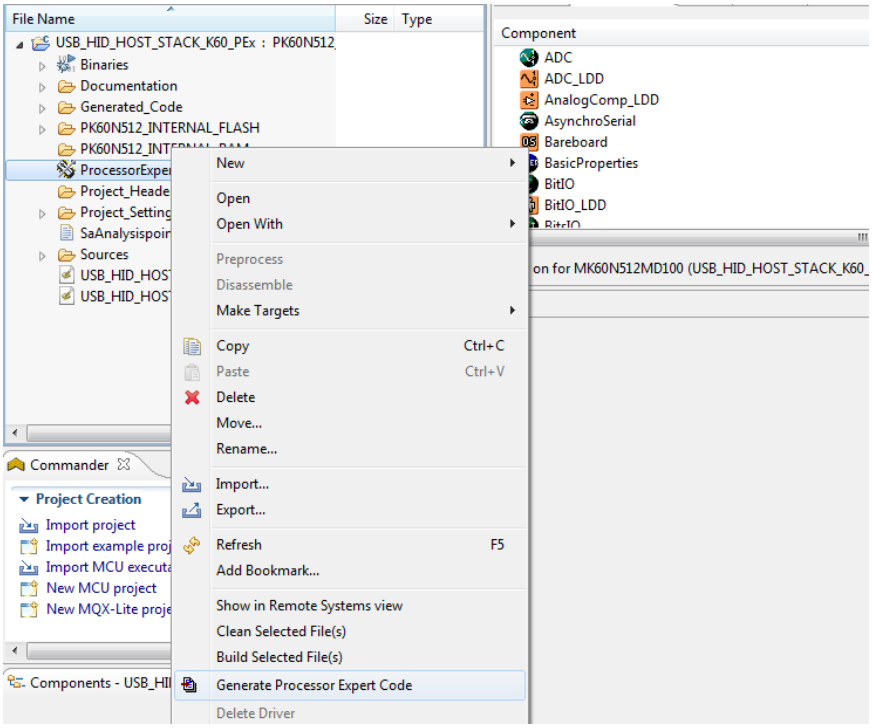


Figure 1-7 Generate Processor Expert code

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.

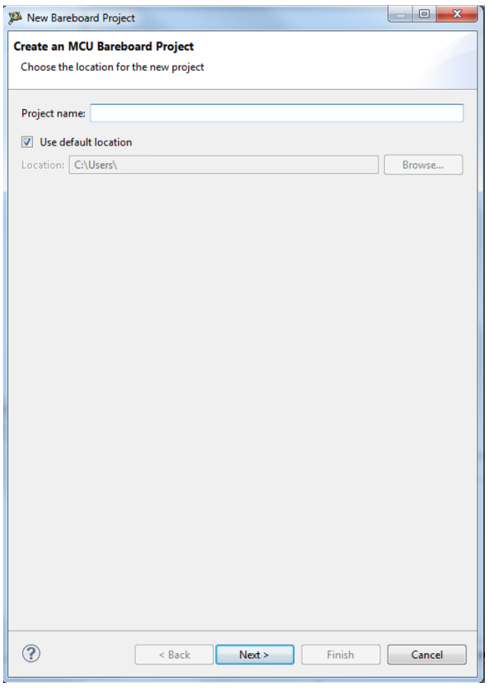


Figure 1-8 New Bareboard project

Step2. Select the device MK60DN512Z -> Next.

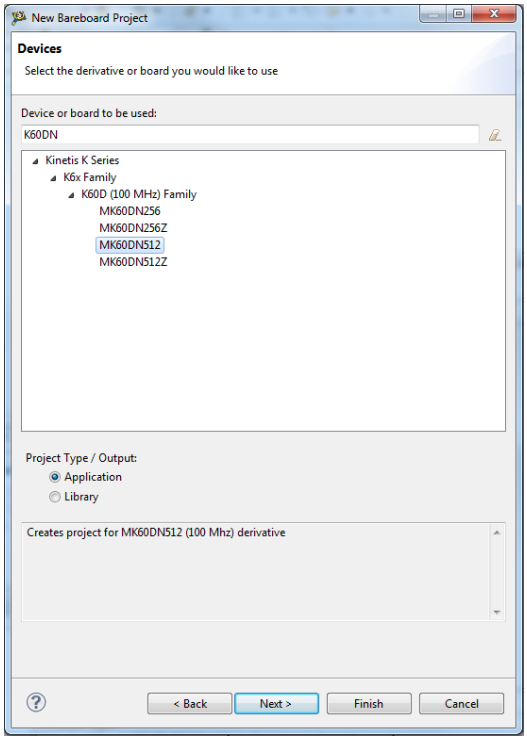


Figure 1-9 Devices window.

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

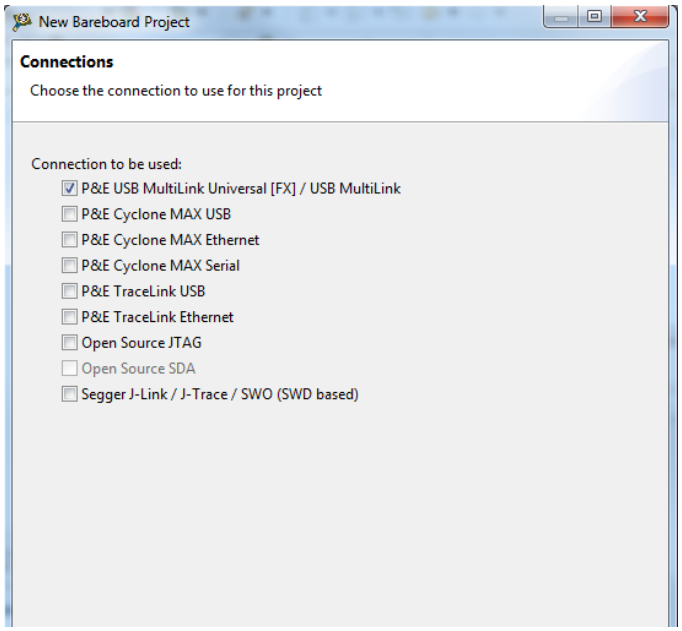


Figure 1-10 Connections window.

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

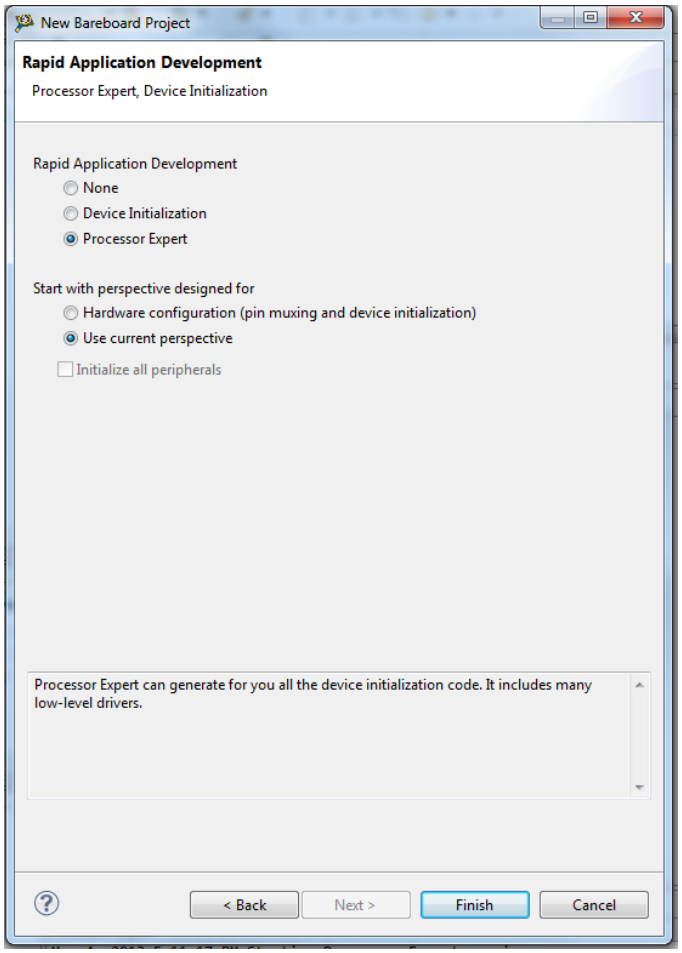


Figure 1-11 Rapid Application Development window.

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.

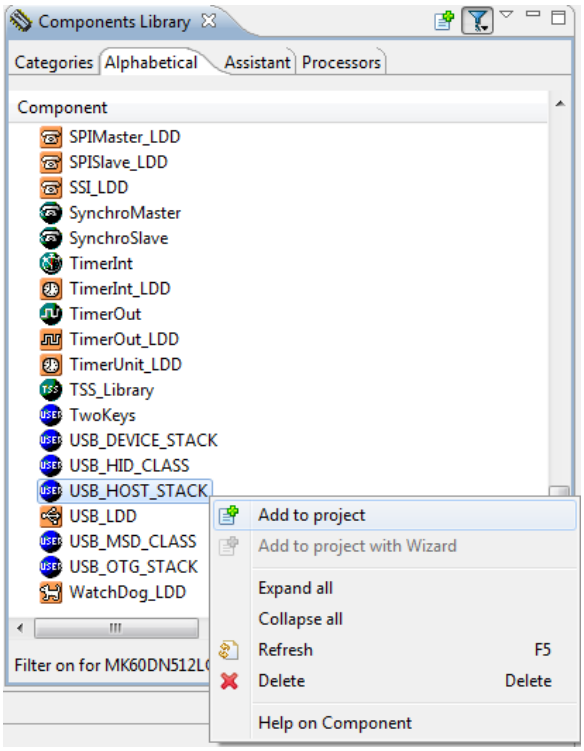


Figure 1-12 Component Library

Step6. Implement same as step 5 to add the TWR_SER_K60_UART component into the project.

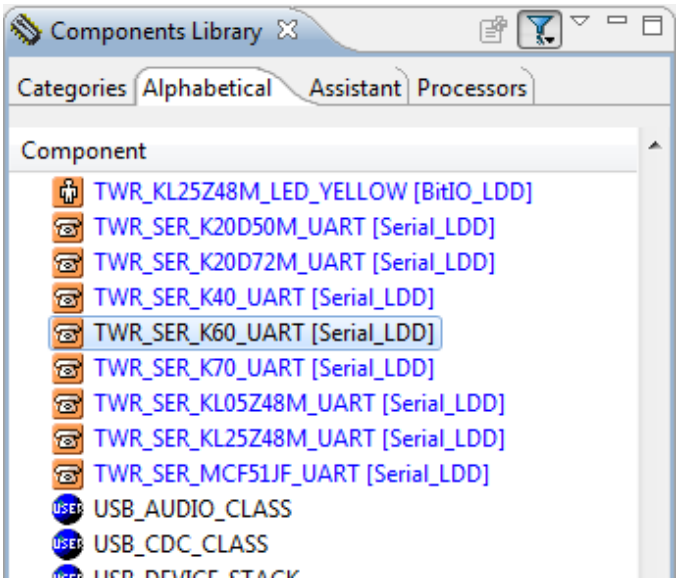


Figure 1-13 Serial_LDD component

Step7. Configure CPU clock to run USB module:

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

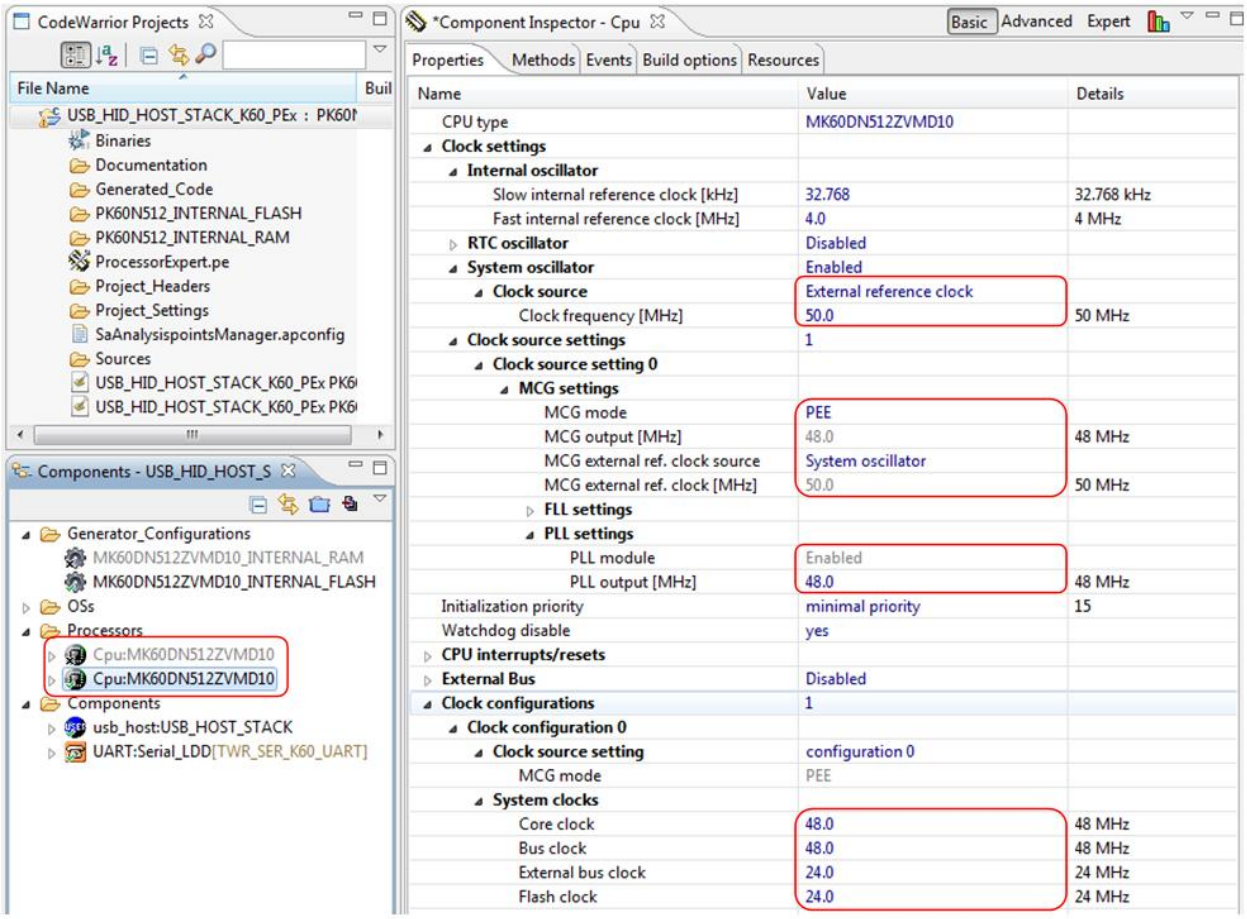


Figure 1-14 USB Clock source settings

- Chose advanced tab

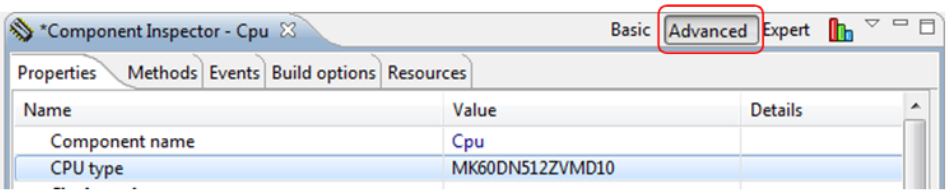


Figure 1-15 Open Advanced Tab

- PLL/FLL clock selection:
    - PLL clock

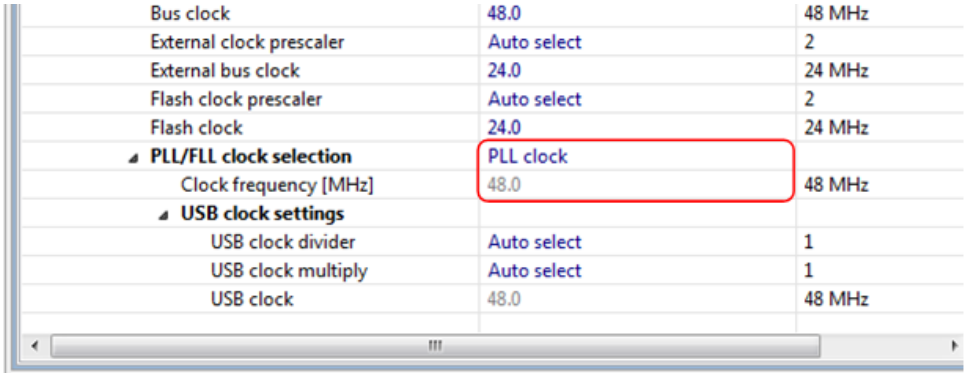


Figure 1-16 USB Clock configuration

Step8. Setup the USB_HOST_STACK component:

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

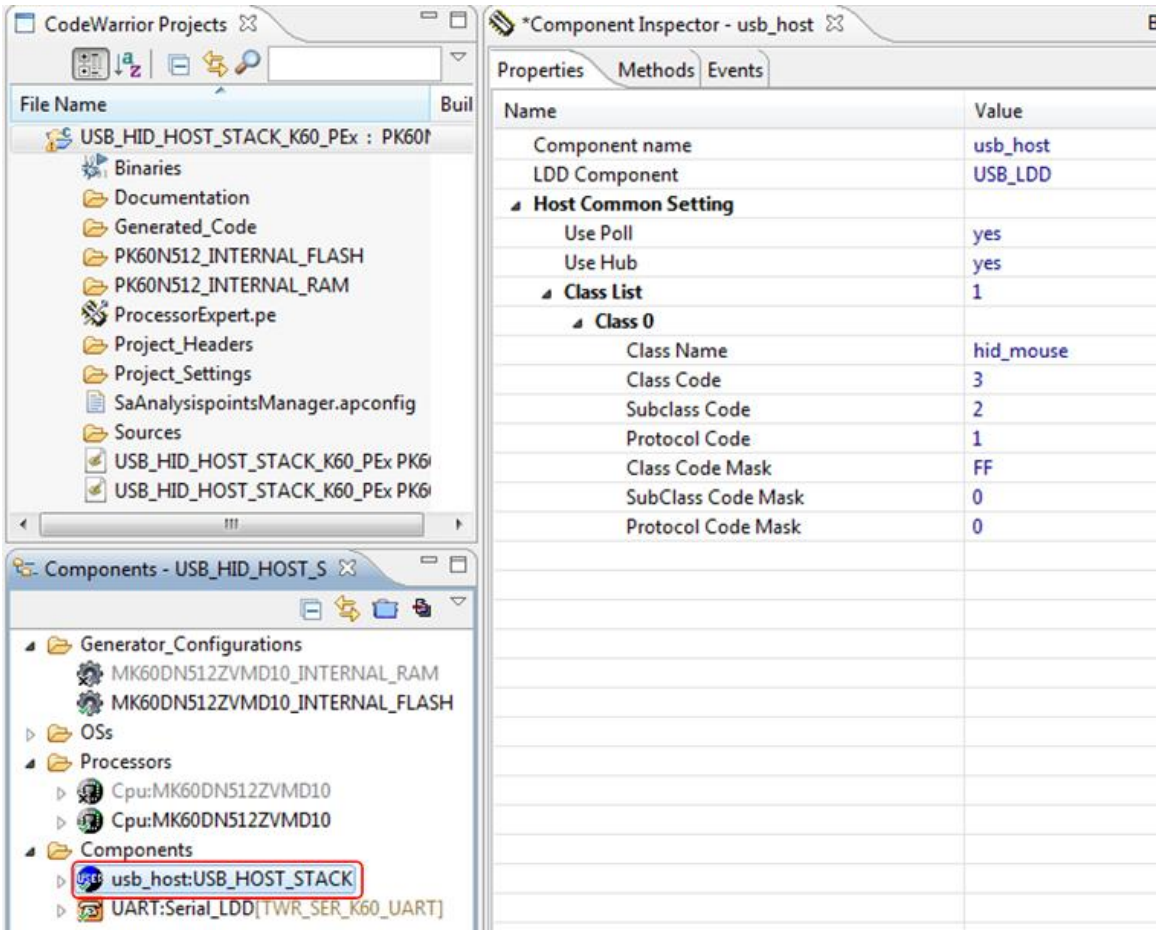


Figure 1-17 Host common settings

Step9. Setup the USB_LDD component corresponding with the USB_HOST_STACK component.

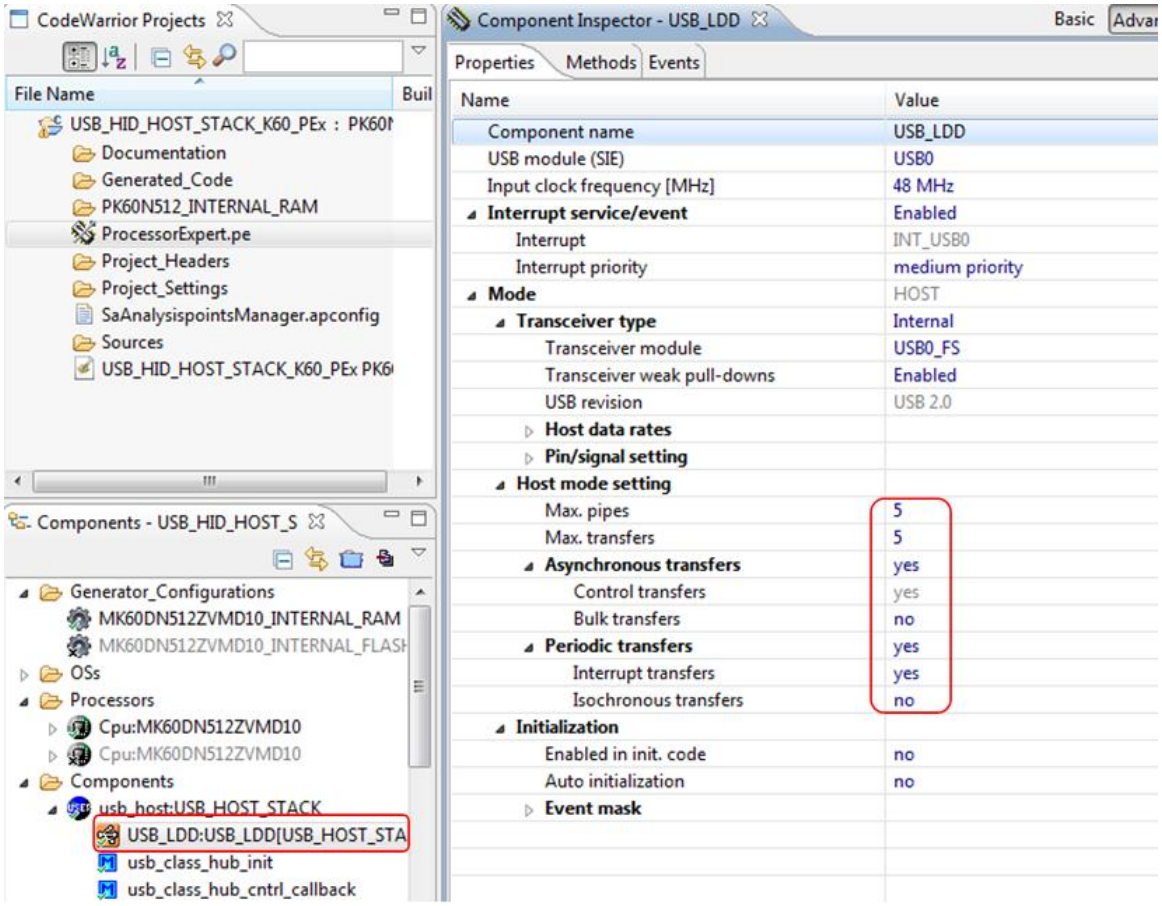


Figure 1-18 USB_LDD component settings

Step10. Setup the TWR_SER_K60_UART component:

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

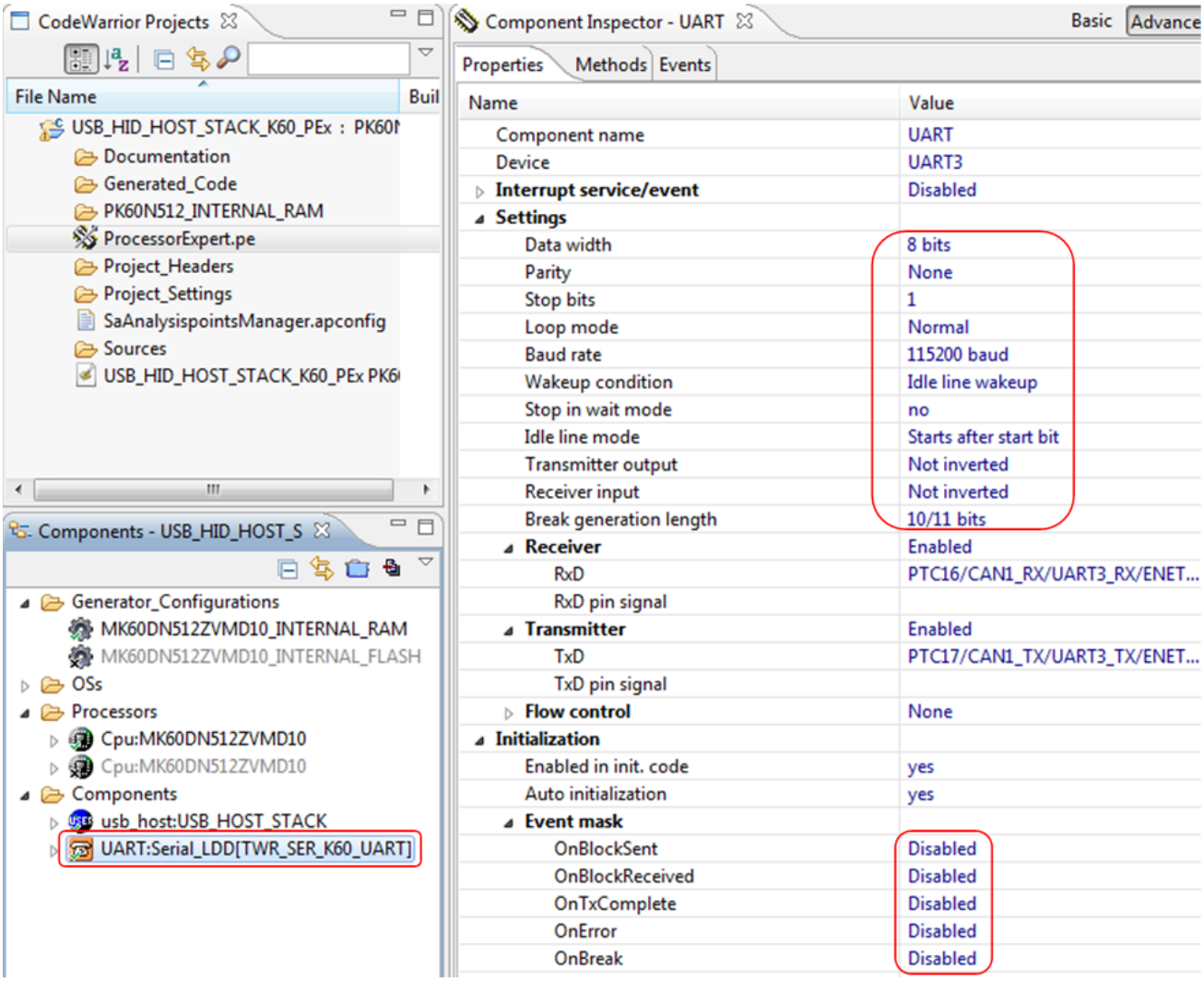


Figure 1-19 Serial_LDD component settings

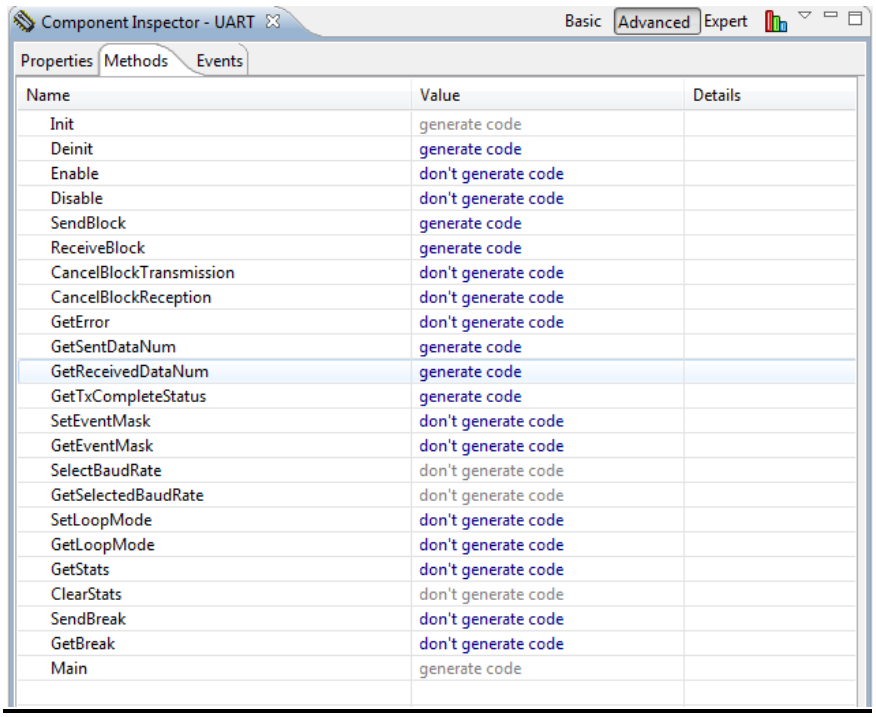


Figure 1-20 Serial_LDD component methods

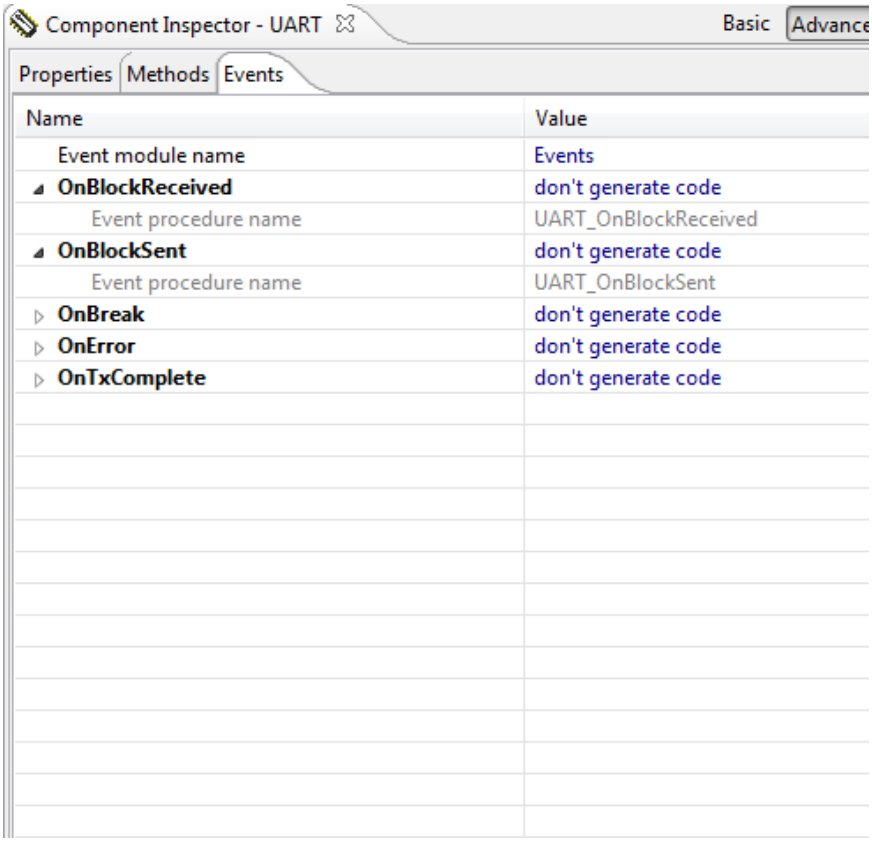


Figure 1-21 Serial_LDD component events

Step11. Generate Processor Expert code:
After setting the components, select ProcessorExpert.pe in the project folder → Right click → Chose Generate Processor Expert Code field to generate source code.

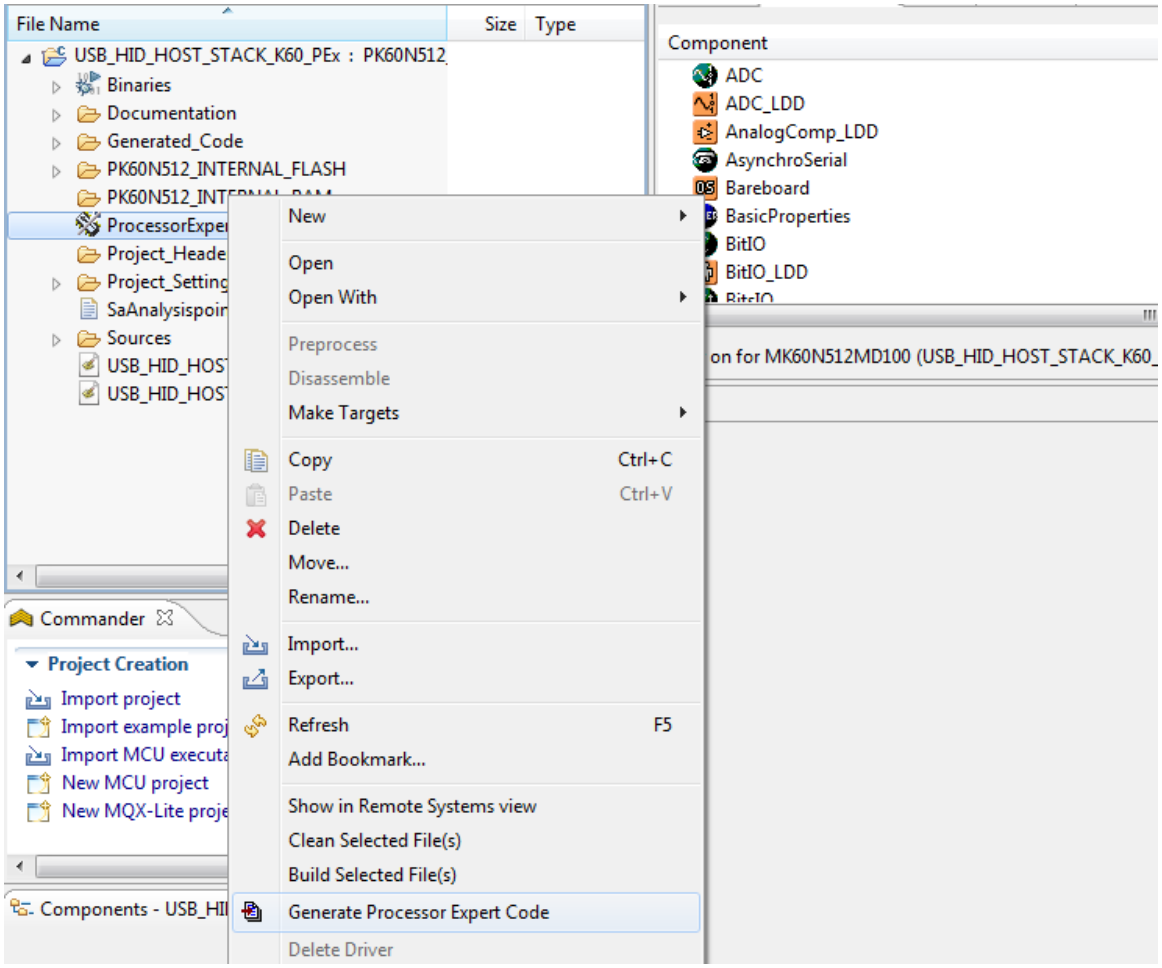


Figure 1-22 Generate Processor Expert code

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.

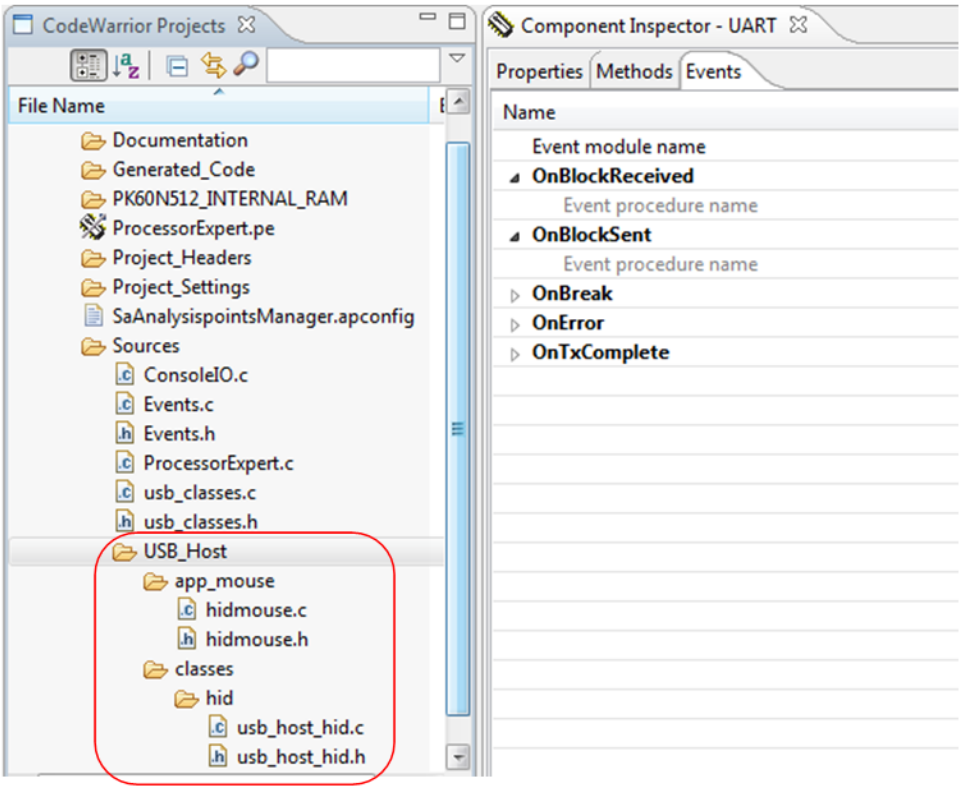


Figure 1-23 USB HID Host included files

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.

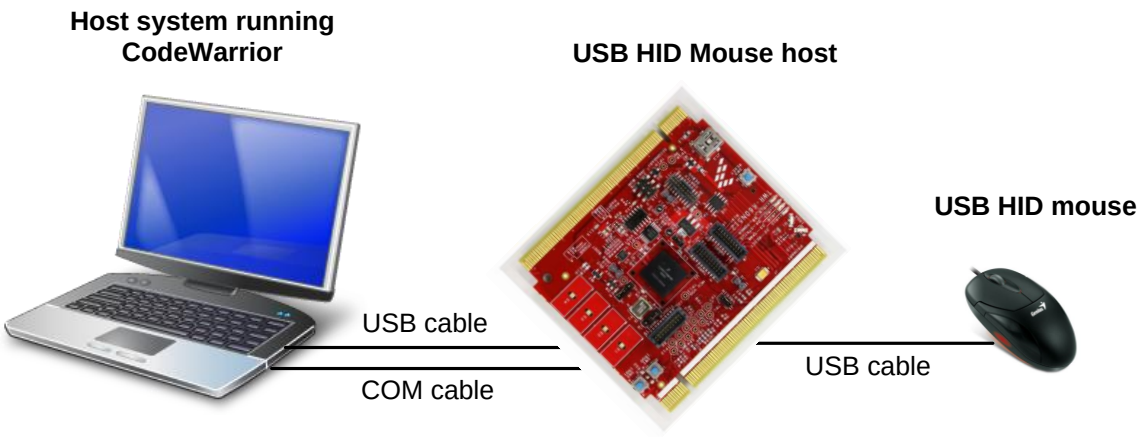


Figure 1-24 Hardware setup

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:

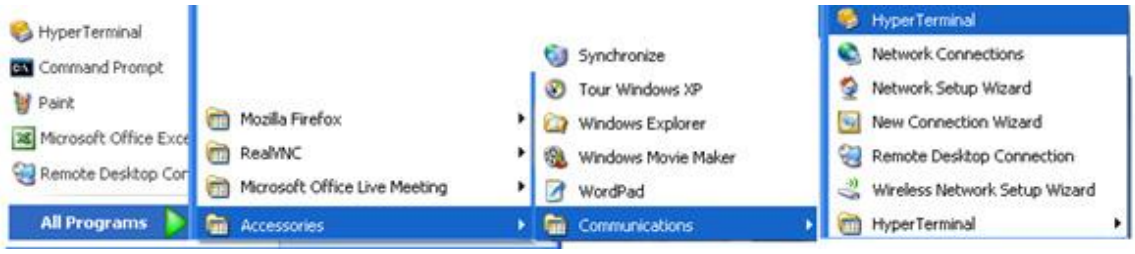


Figure 1-25 Run HyperTerminal application

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

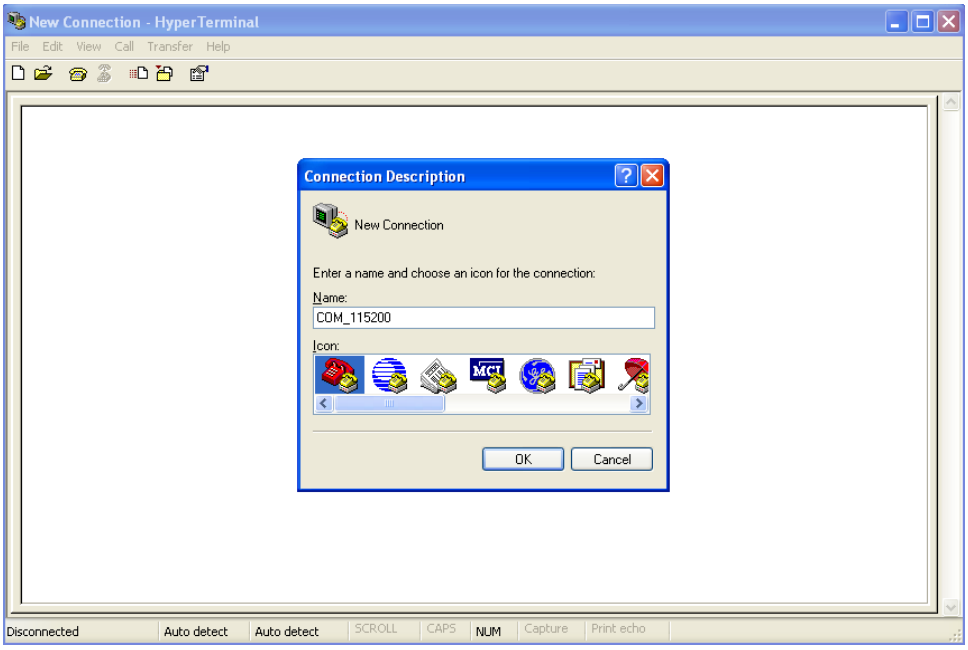


Figure 1-26 HyperTerminal startup

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

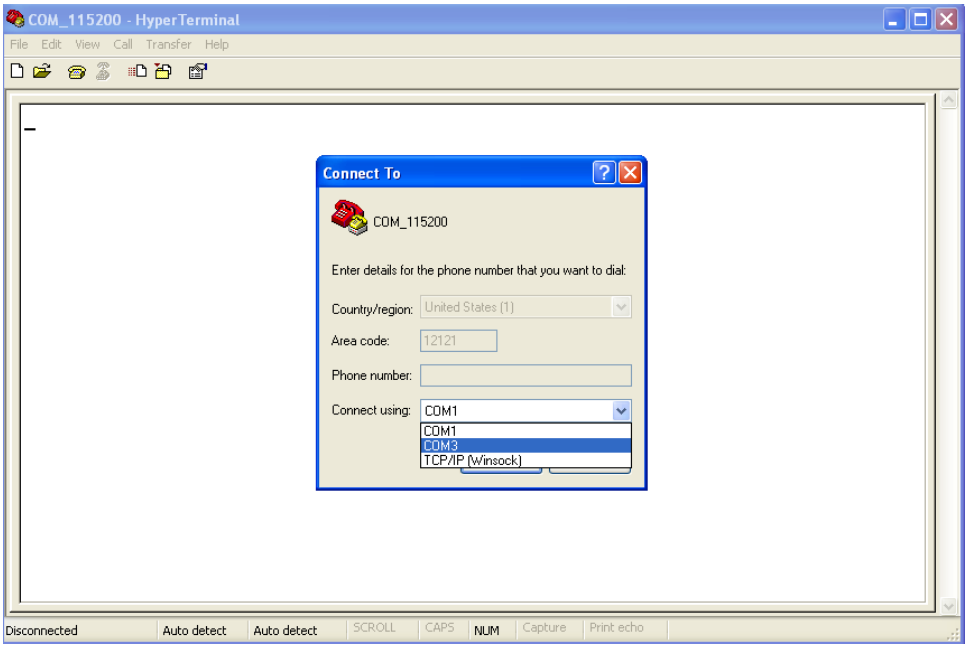


Figure 1-27 Connect using COM port

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

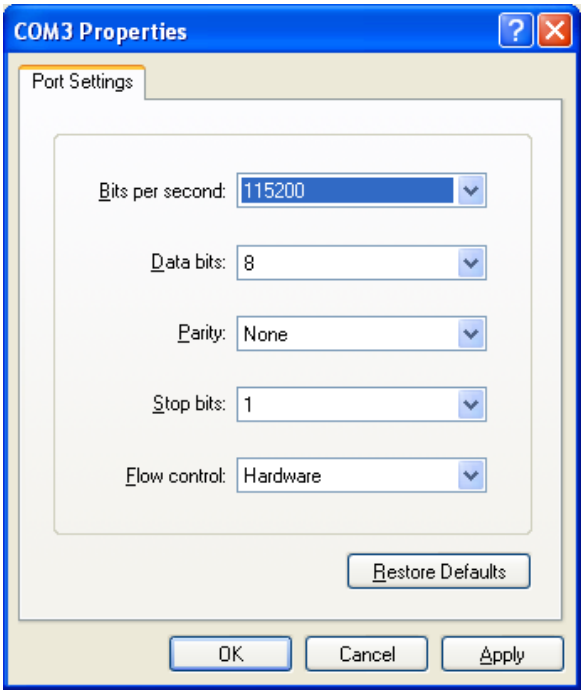


Figure 1-28 COM properties

- The HyperTerminal is now configured as shown below:

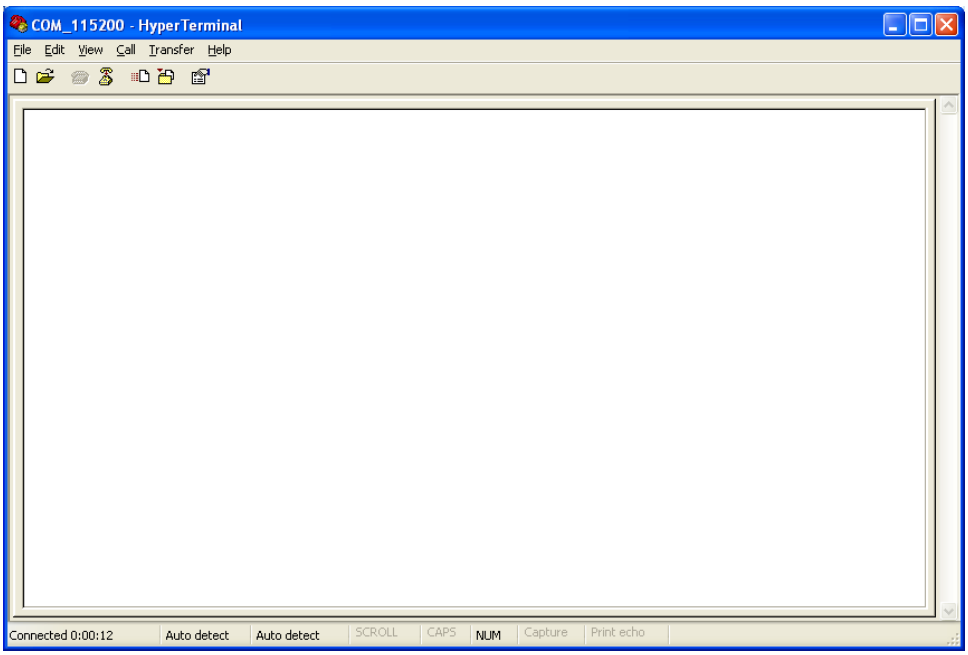


Figure 1-29 Hyper Terminal

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:

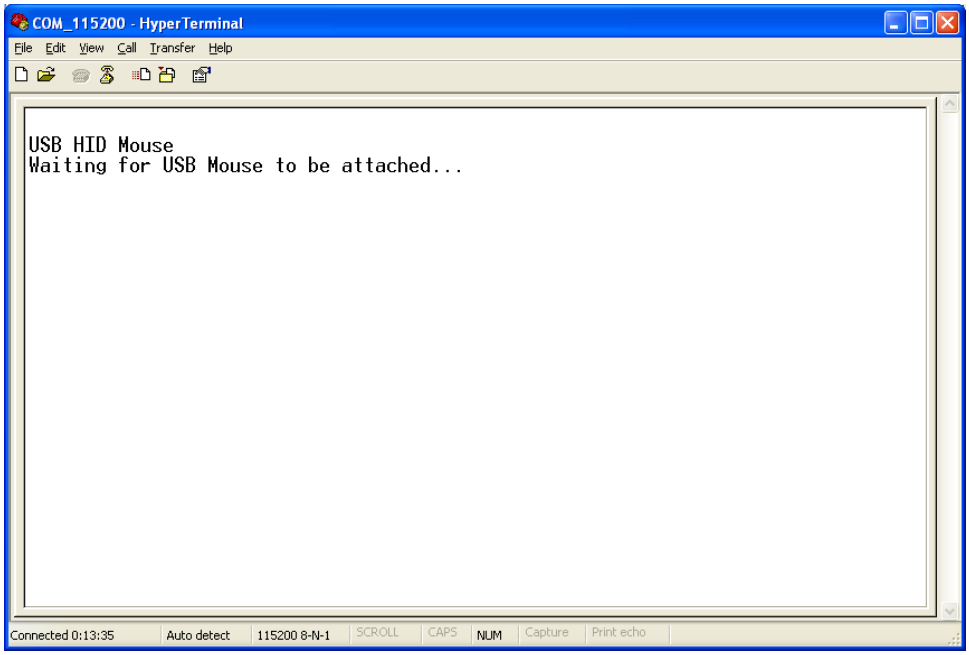


Figure 1-30 USB HID mouse host initializing

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

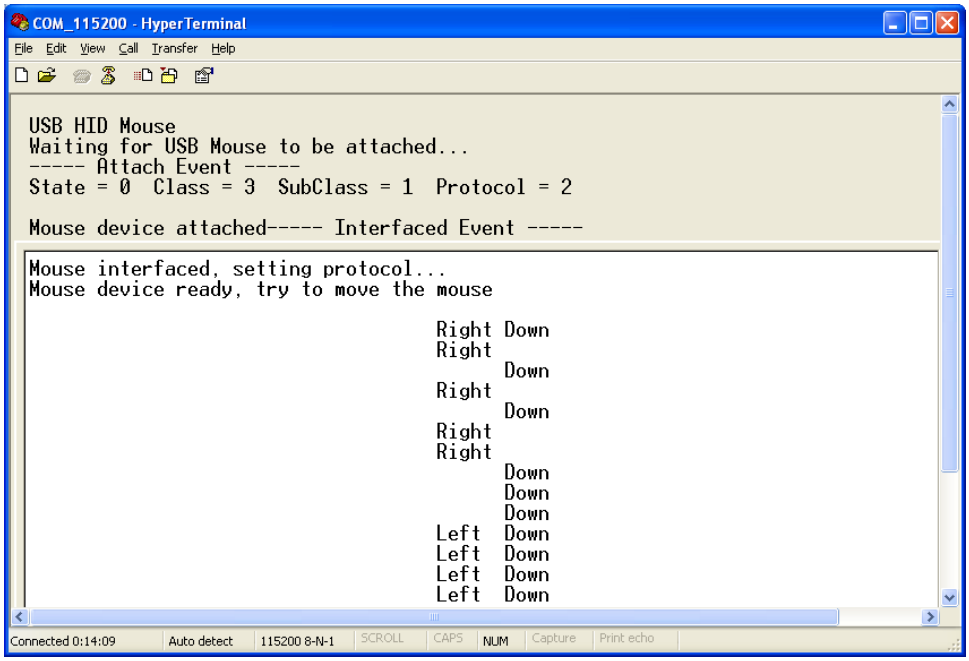


Figure 1-31 USB HID mouse host operating

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:

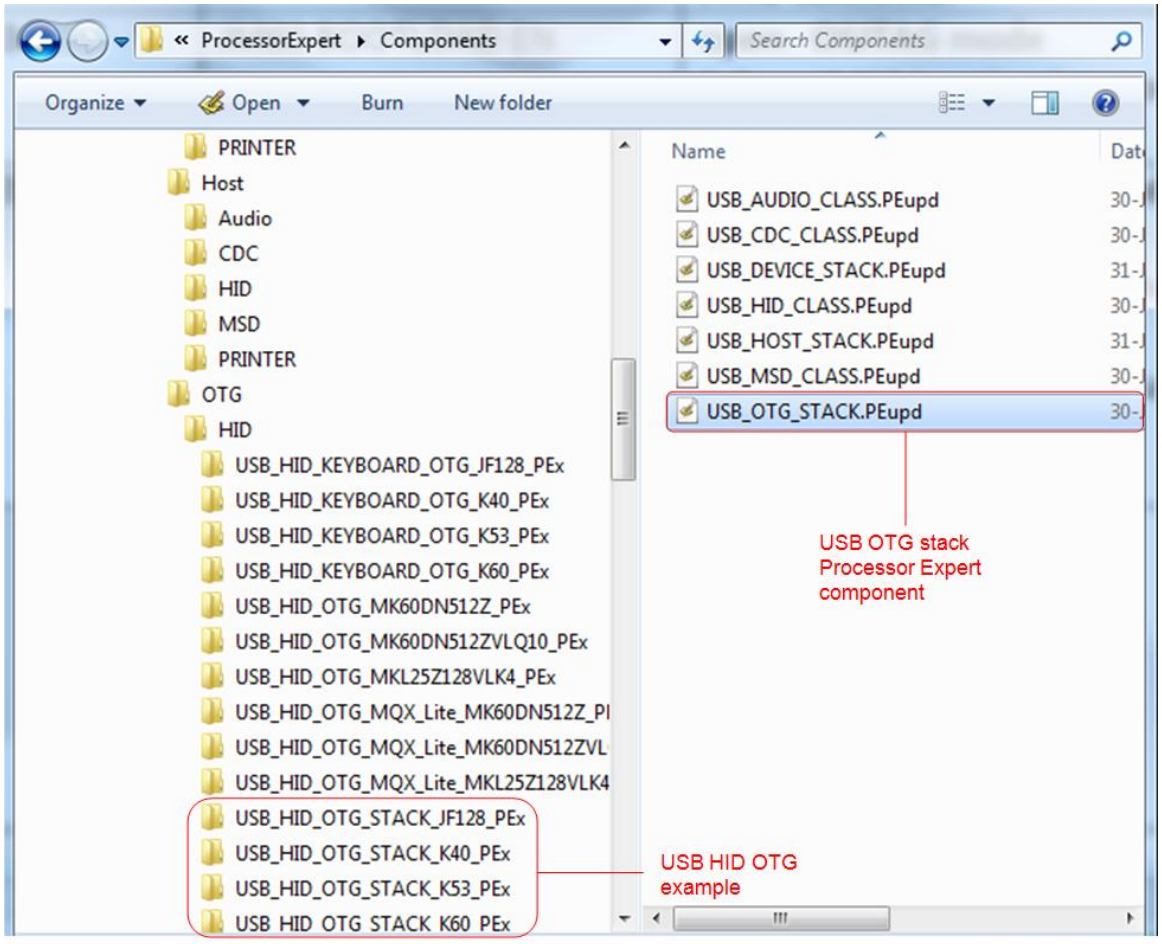


Figure 2-1 USB_HID_OTG folder structure

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.

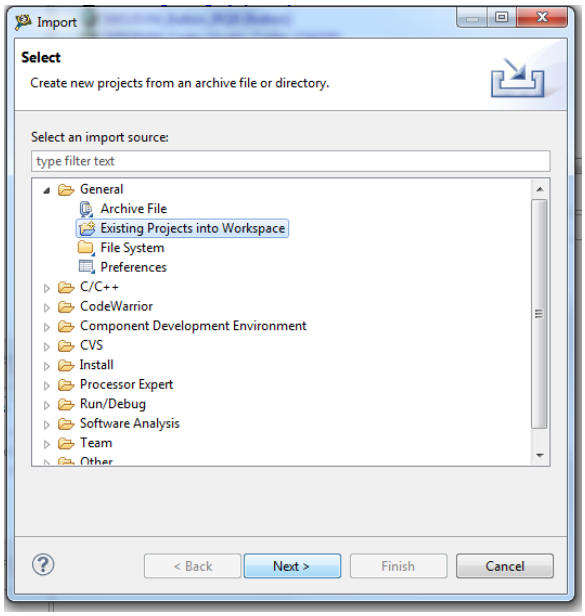


Figure 2-2 Import existing projects into Workspace

Step2. Click Browse button → Select USB_HID_OTG_STACK_K60_PEx folder -> Click OK -> then click Finish button.

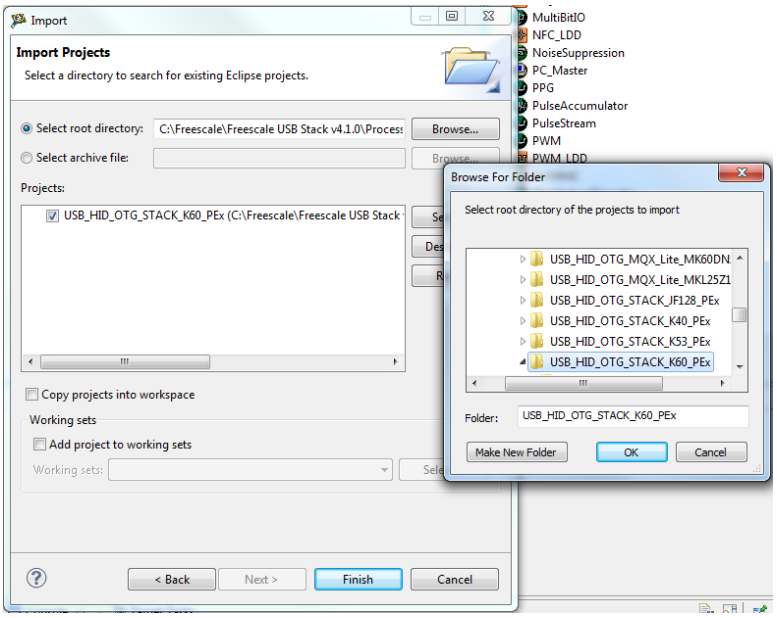


Figure 2-3 Import existing projects into Workspace

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.

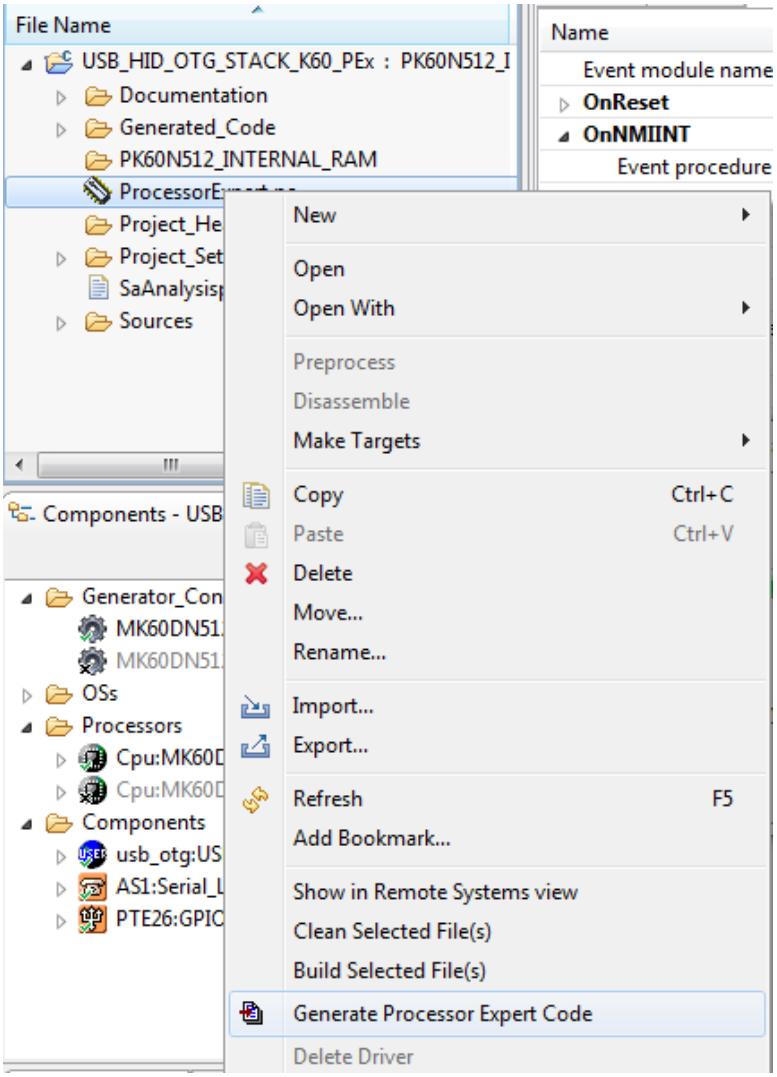


Figure 2-4 Generate Processor Expert code

2.3.2 Creating USB_HID_OTG_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.

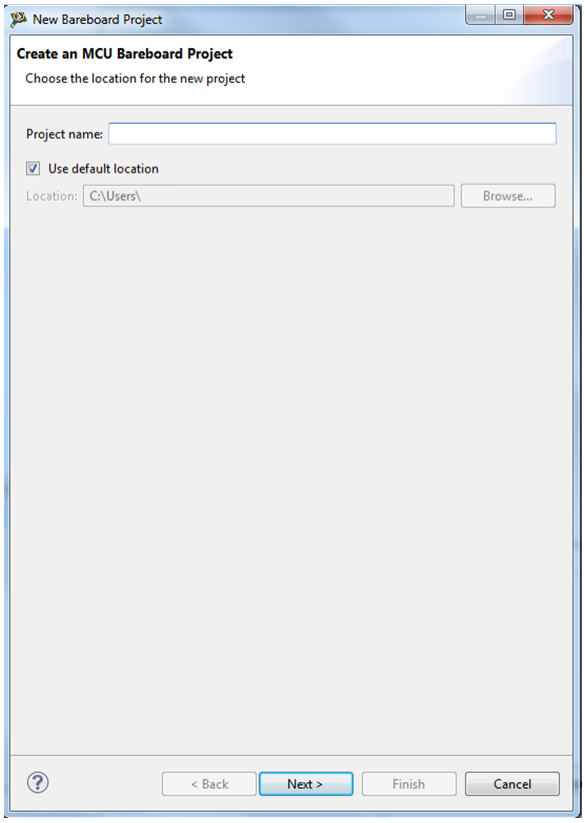


Figure 2-5 New Bareboard Project

Step2. Select the device MK60DN512Z -> Next.

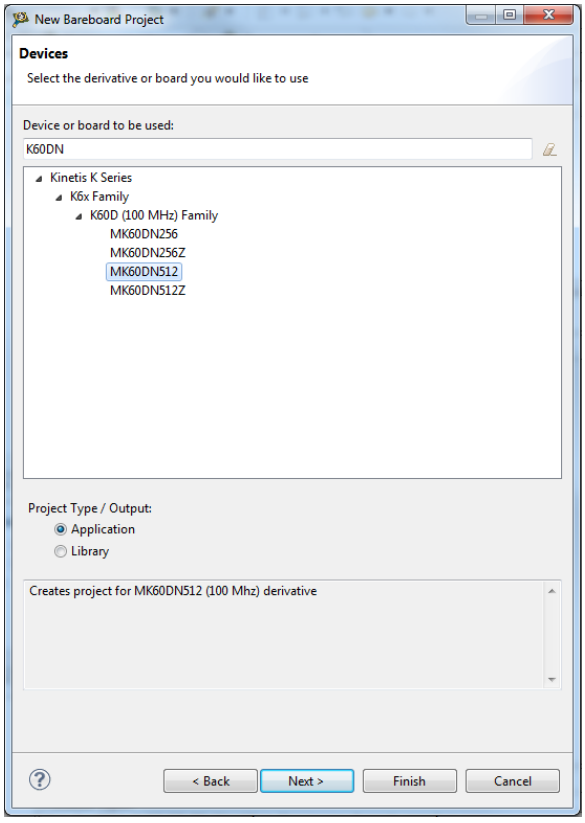


Figure 2-6 Devices window.

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

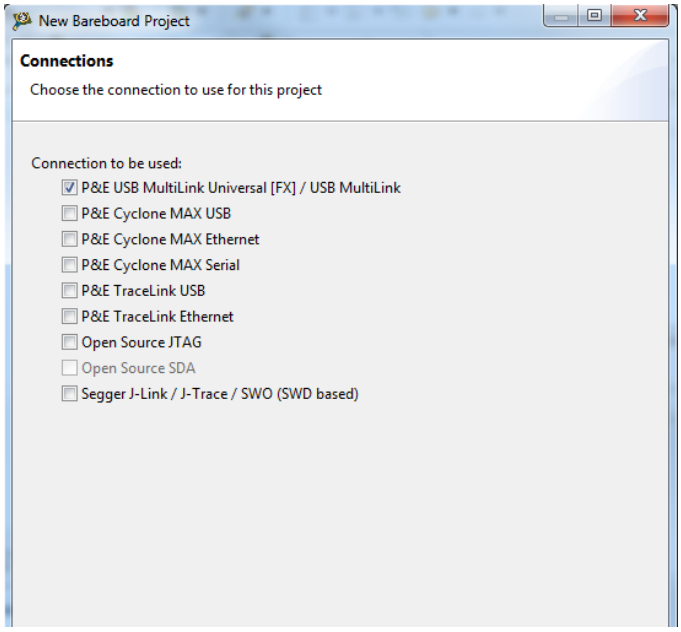


Figure 2-7 Connections window.

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

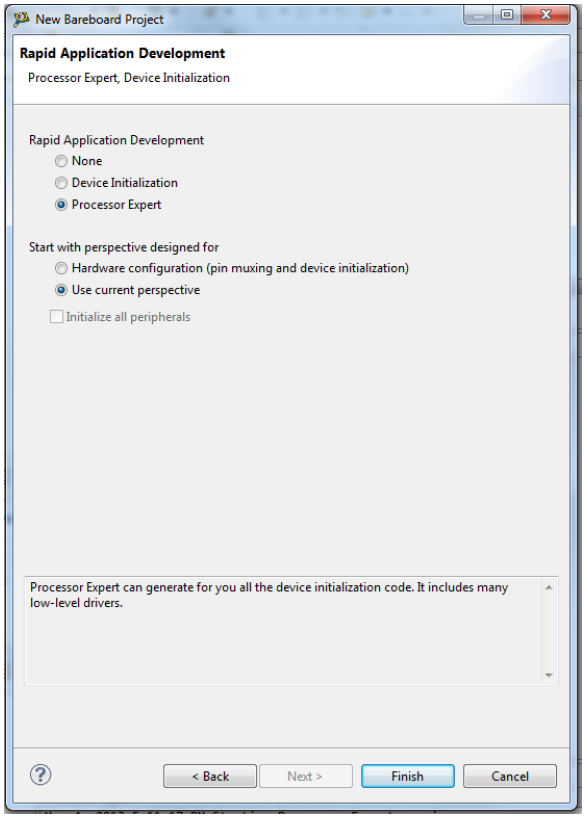


Figure 2-8 Rapid Application Development window.

Step5. From the CW10.3 toolbar, select Processor Expert → Select Show Views to open Components Library → Chose and add the USB_OTG_STACK component to the project.

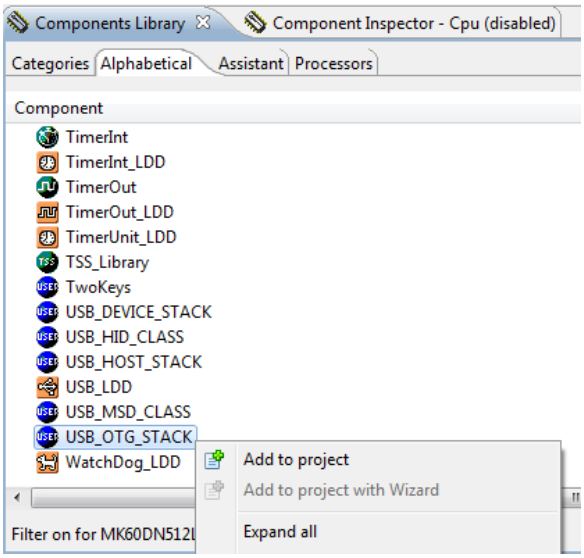


Figure 2-9 Components Library

The USB_OTG_STACK component has the user interface as shown below:

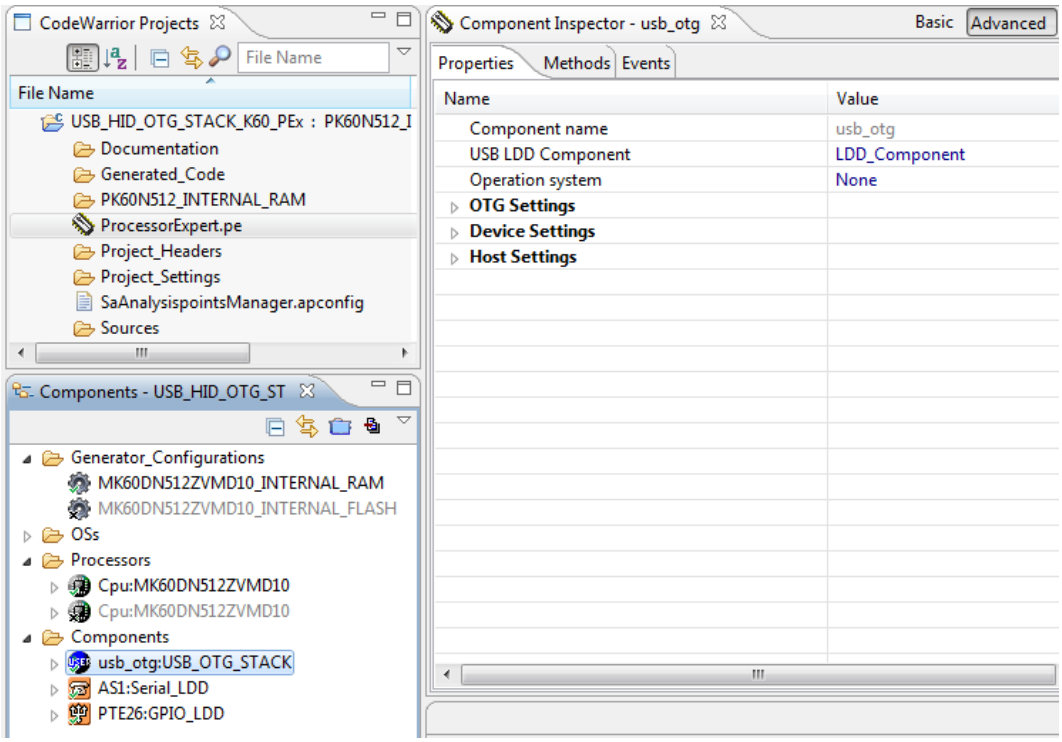


Figure 2-10 USB_OTG_STACK component user interface

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:
  - Target: Cpu:MK60N512VMD100
  - Clock source: External reference clock 50MHz
  - MCG mode: PEE
  - MCG output: 48 MHz
  - PLL output: 48MHz
  - Core clock: 48MHz
  - Bus clock: 48MHz
  - External bus clock: 24MHz
  - Flash clock: 24MHz

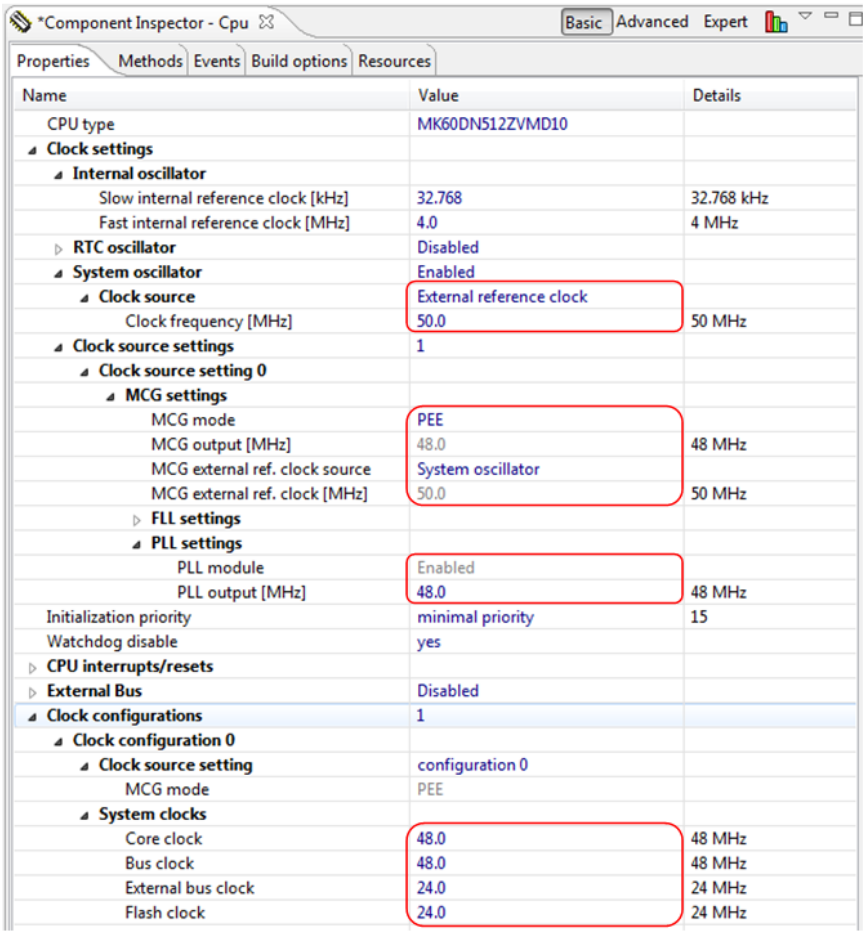


Figure 2-11 USB Clock source settings

- Chose advanced tab:

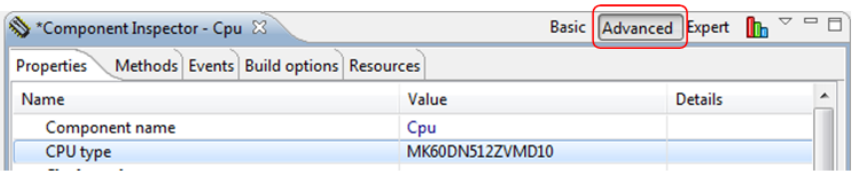


Figure 2-12 Open Advanced Tab

- PLL/FLL clock selection:
    - PLL clock

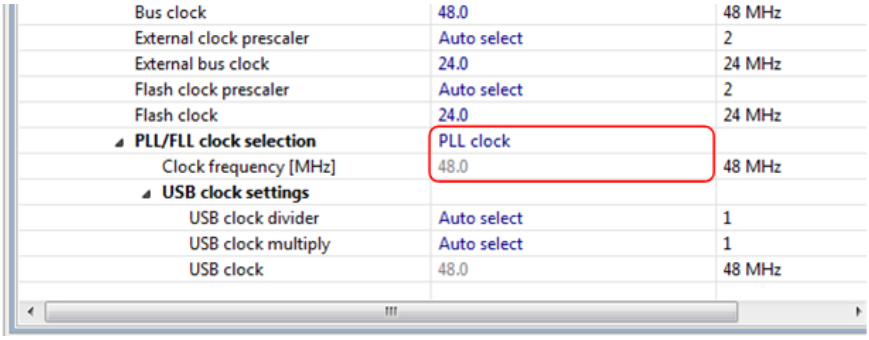


Figure 2-13 USB Clock configuration

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

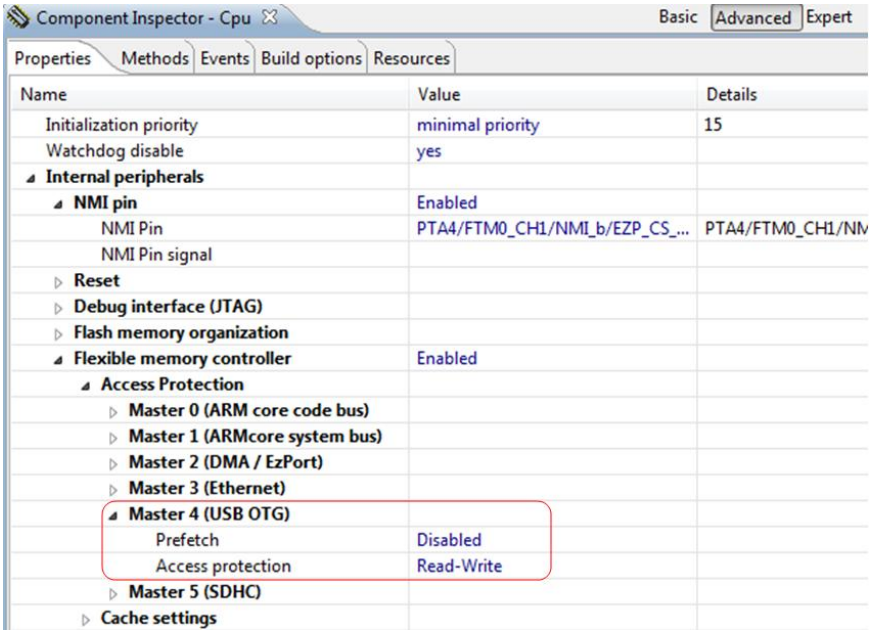


Figure 2-14 Flexible memory controller settings

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

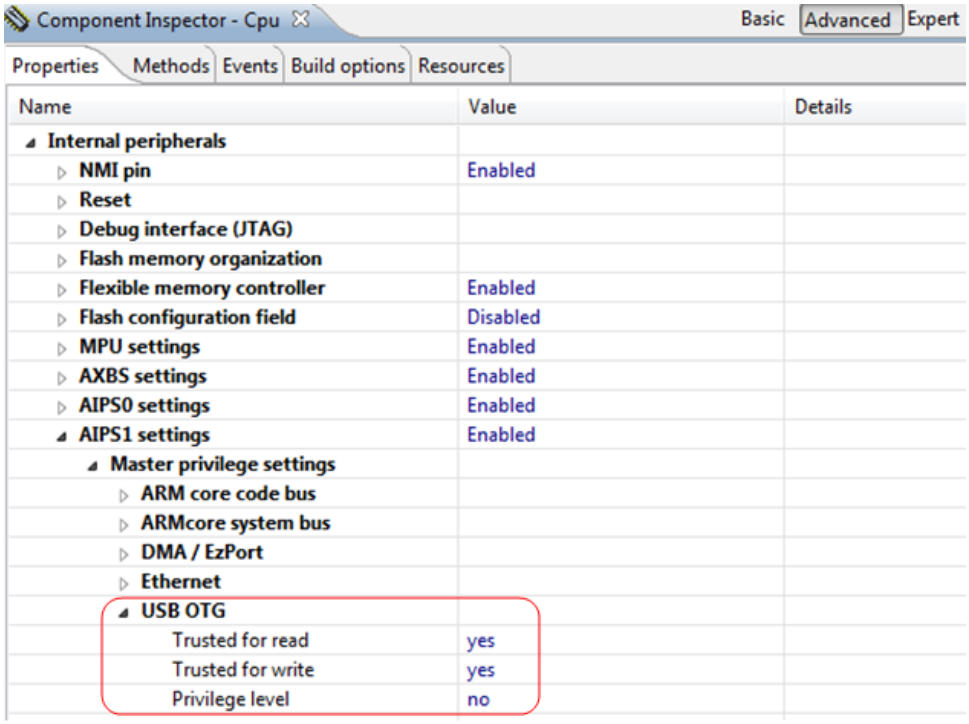


Figure 2-15 AIPS1 settings

- Enable LVD interrupt in the Power management controller field.

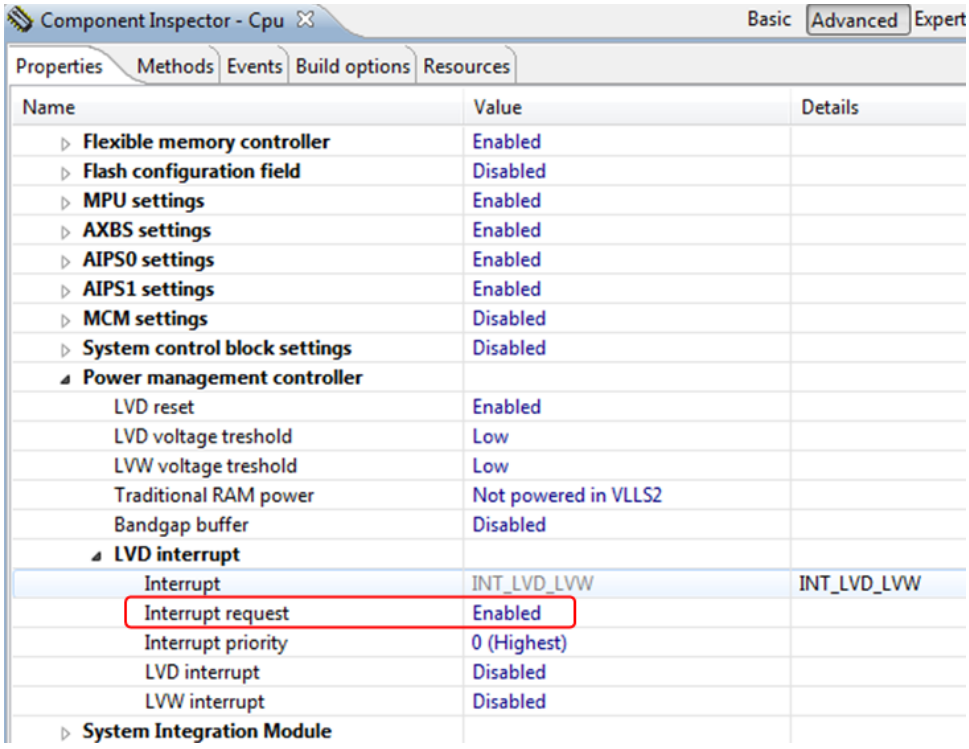


Figure 2-16 Power management controller settings

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:
  - Chose GIPO LDD component to configure interrupt pin for MAX3353 as in the following figure:

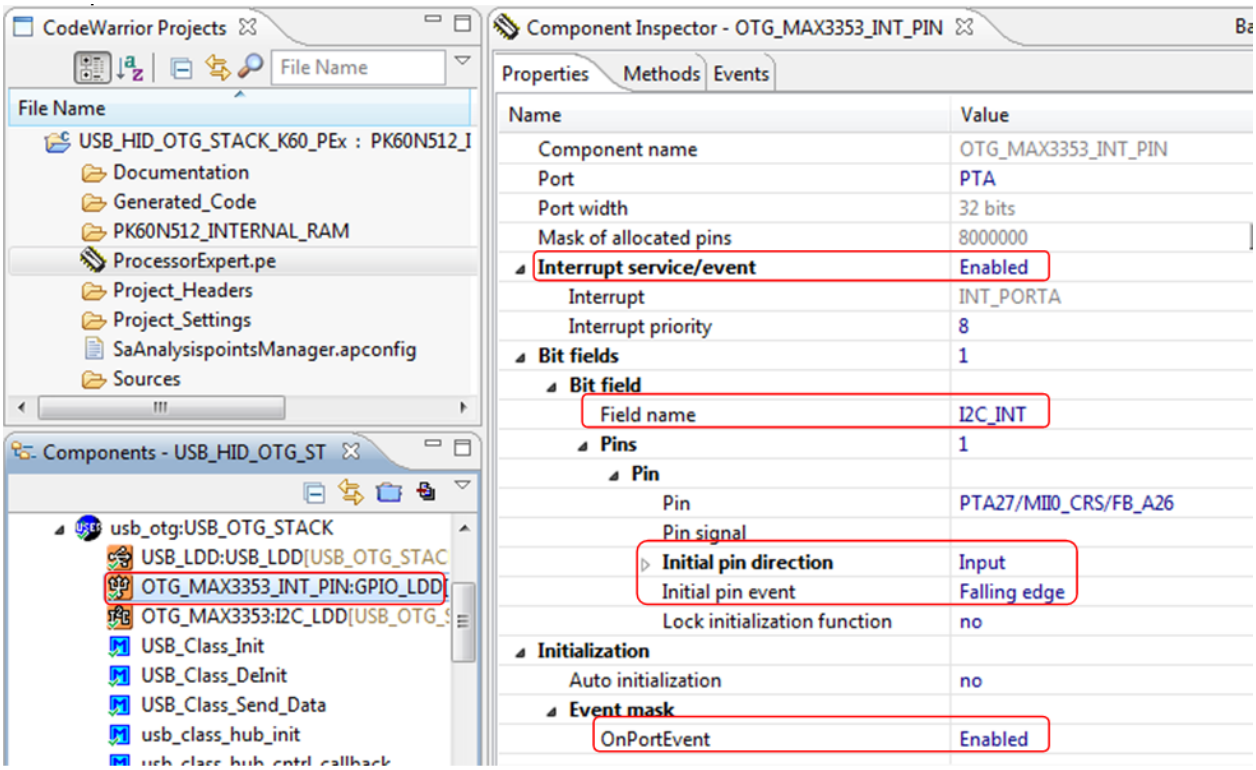


Figure 2-17 MAX3353 Interrupt pin settings

- Chose I2C LDD component to configure the features for MAX3353

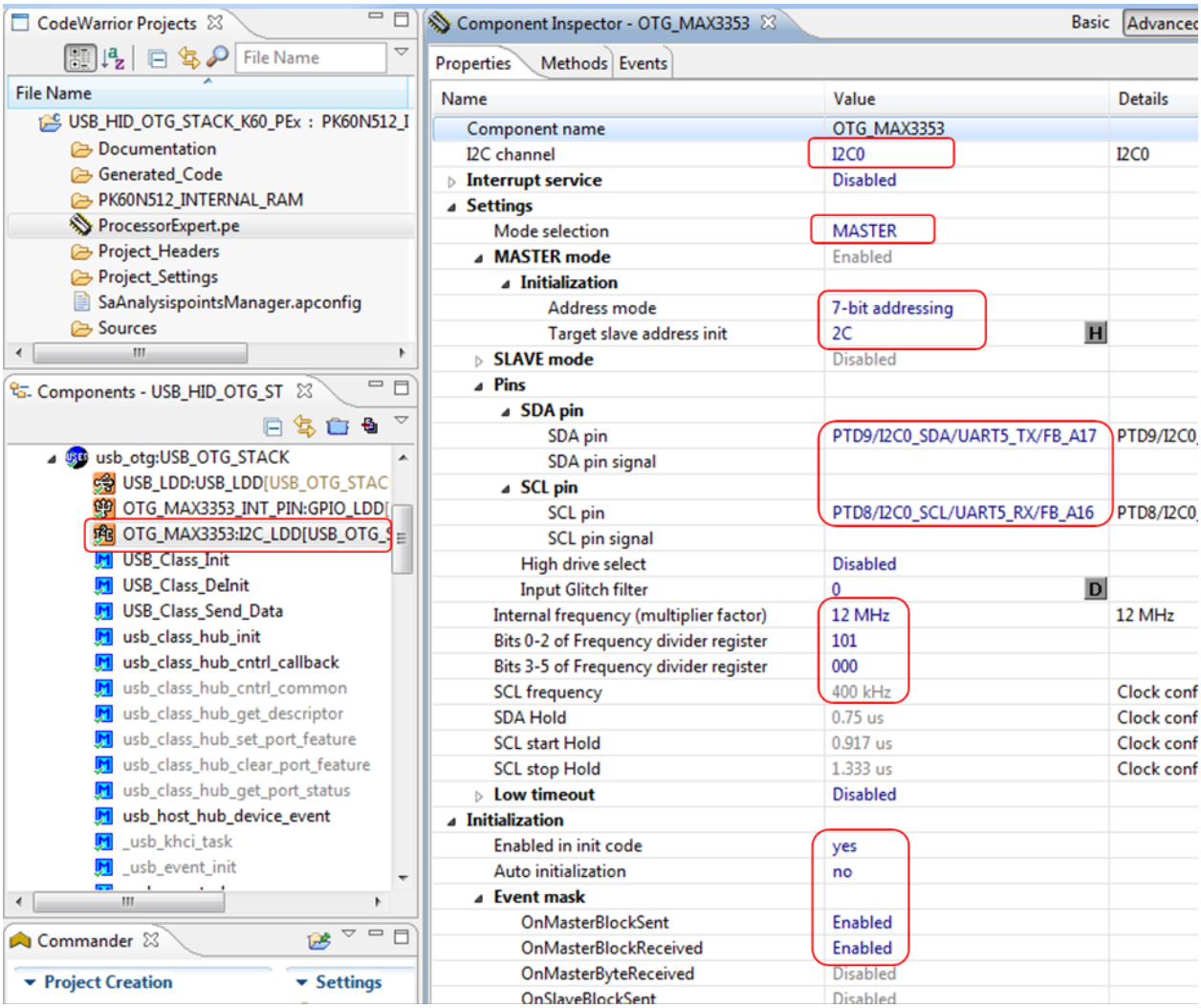


Figure 2-18 MAX3353 settings

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

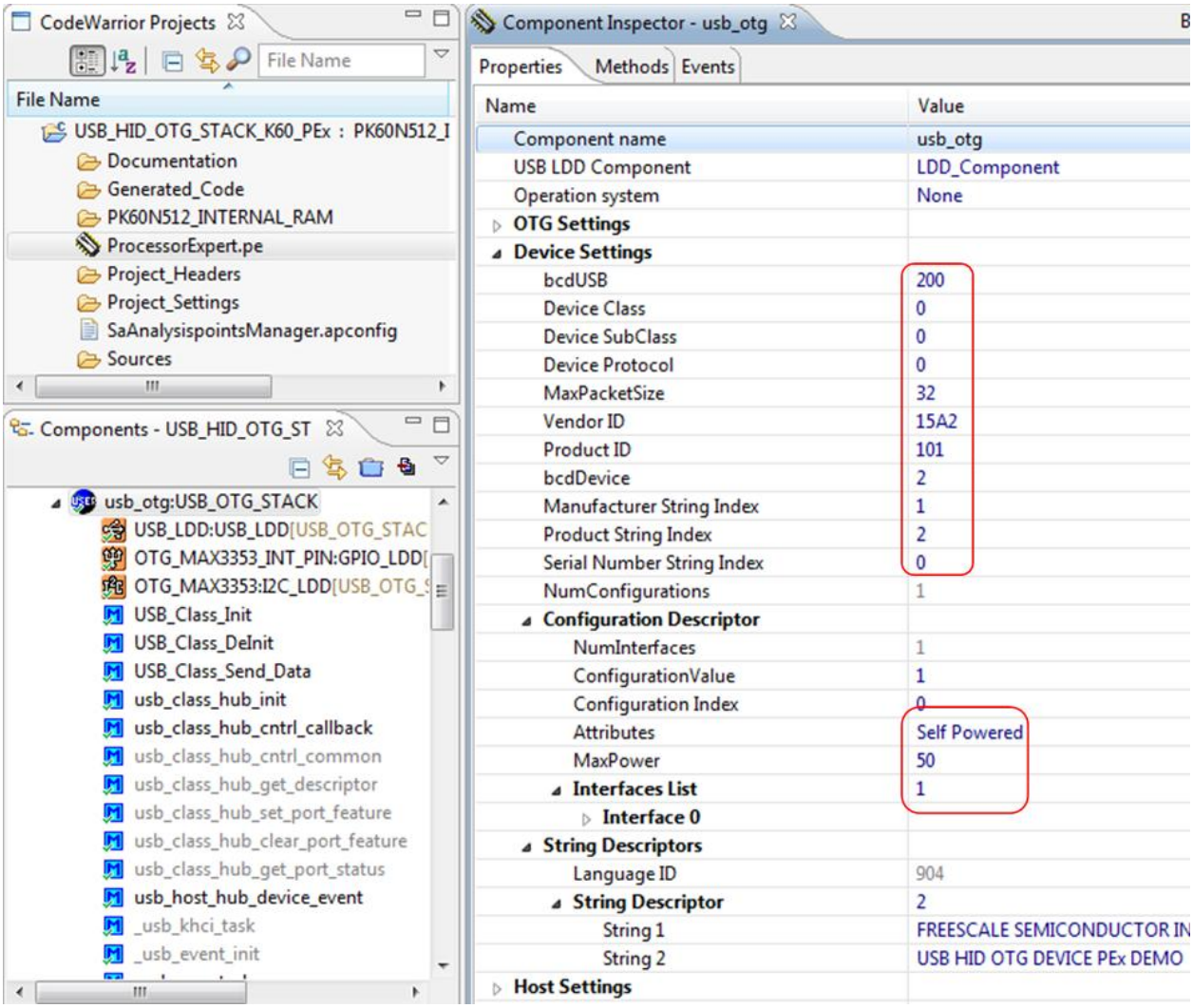


Figure 2-19 General descriptors settings

- Configure HID descriptor as in the following figure:

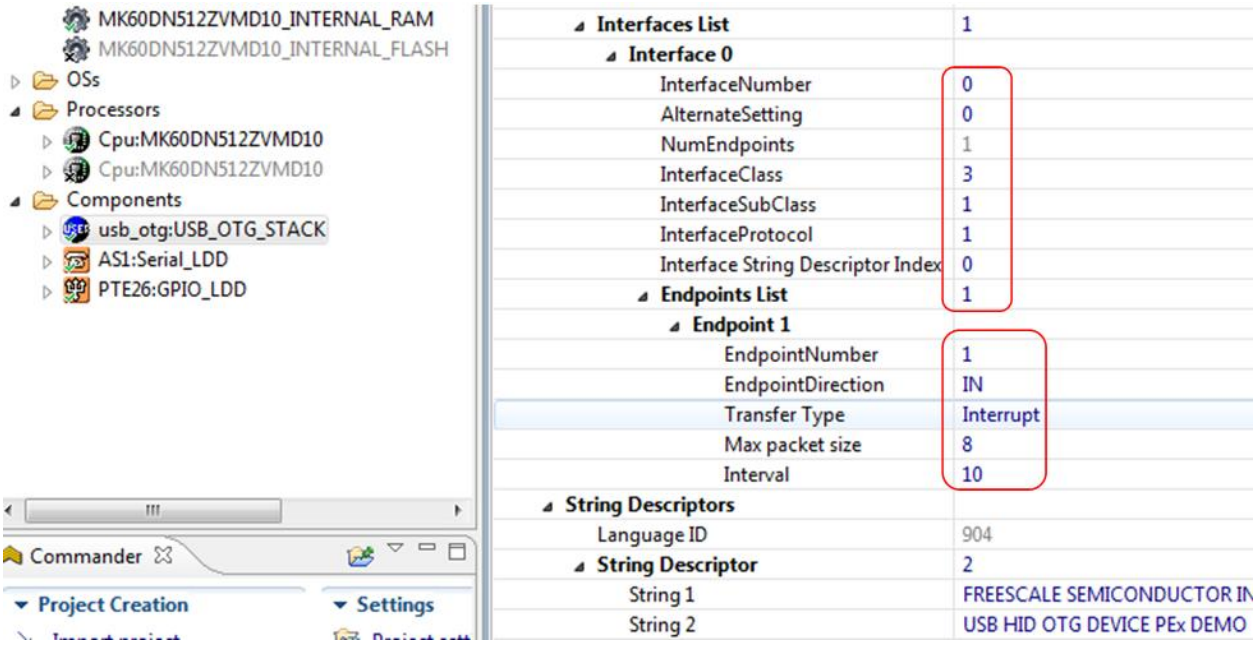


Figure 2-20 HID descriptor settings

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

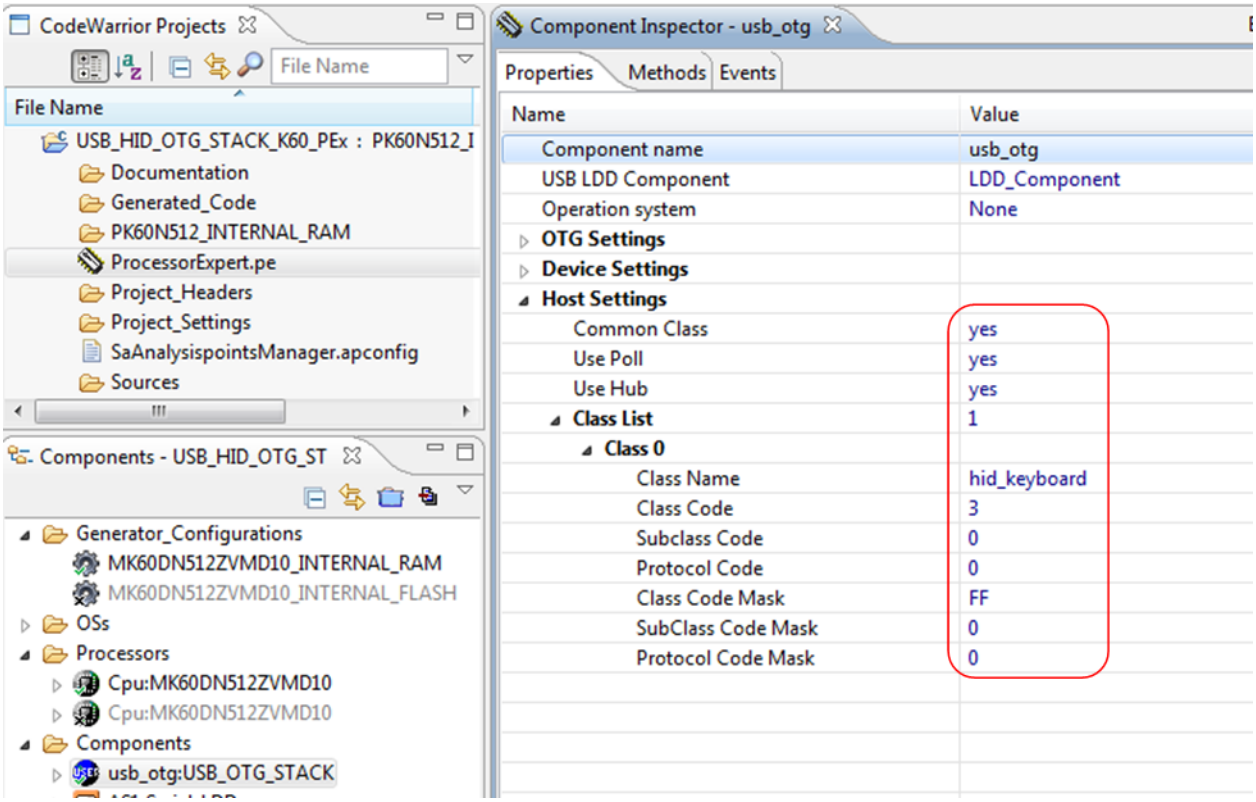


Figure 2-21 Host settings

Step9. Configure the USB_LDD component as shown below:

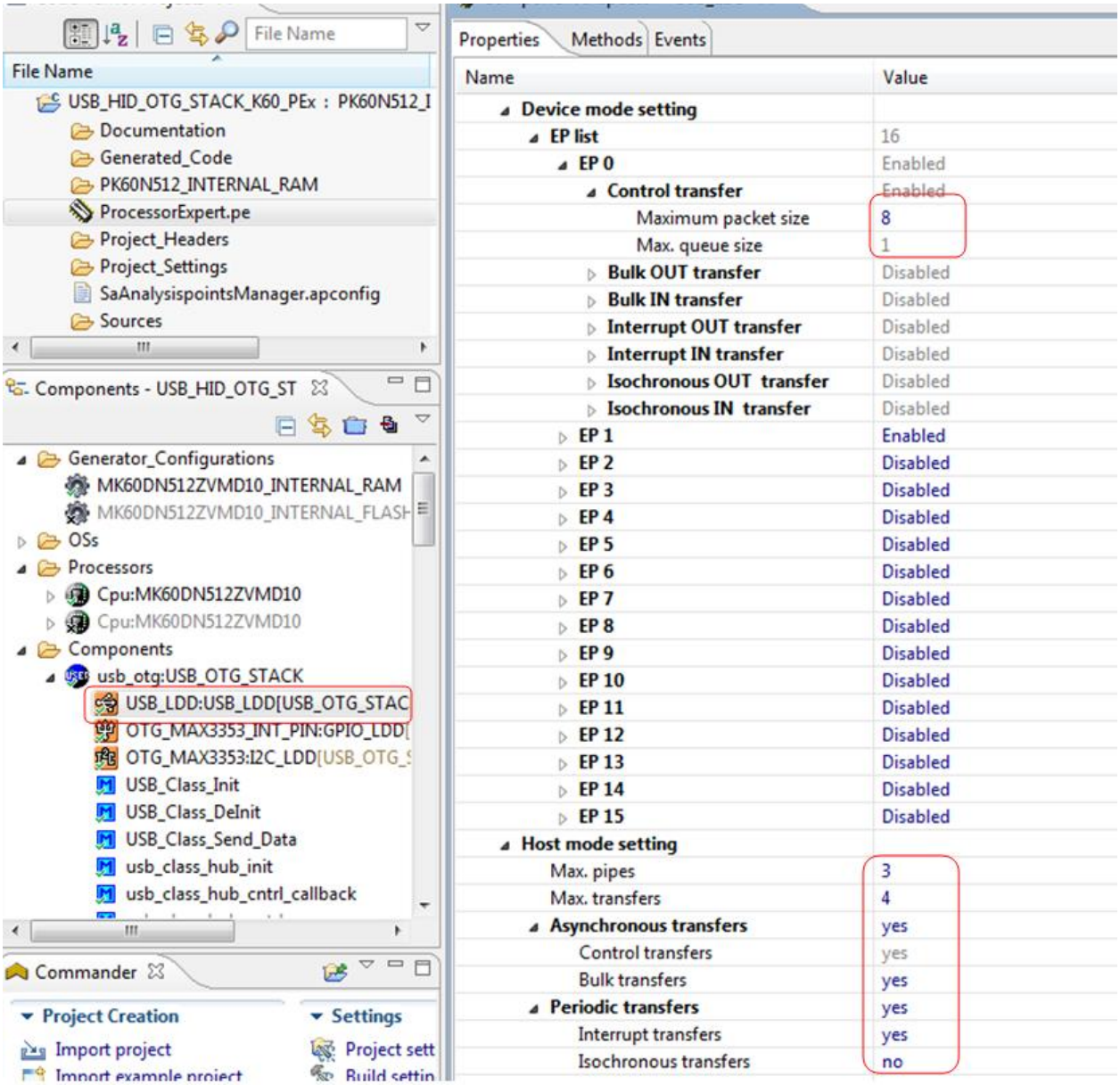


Figure 2-22 USB_LDD component settings

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:

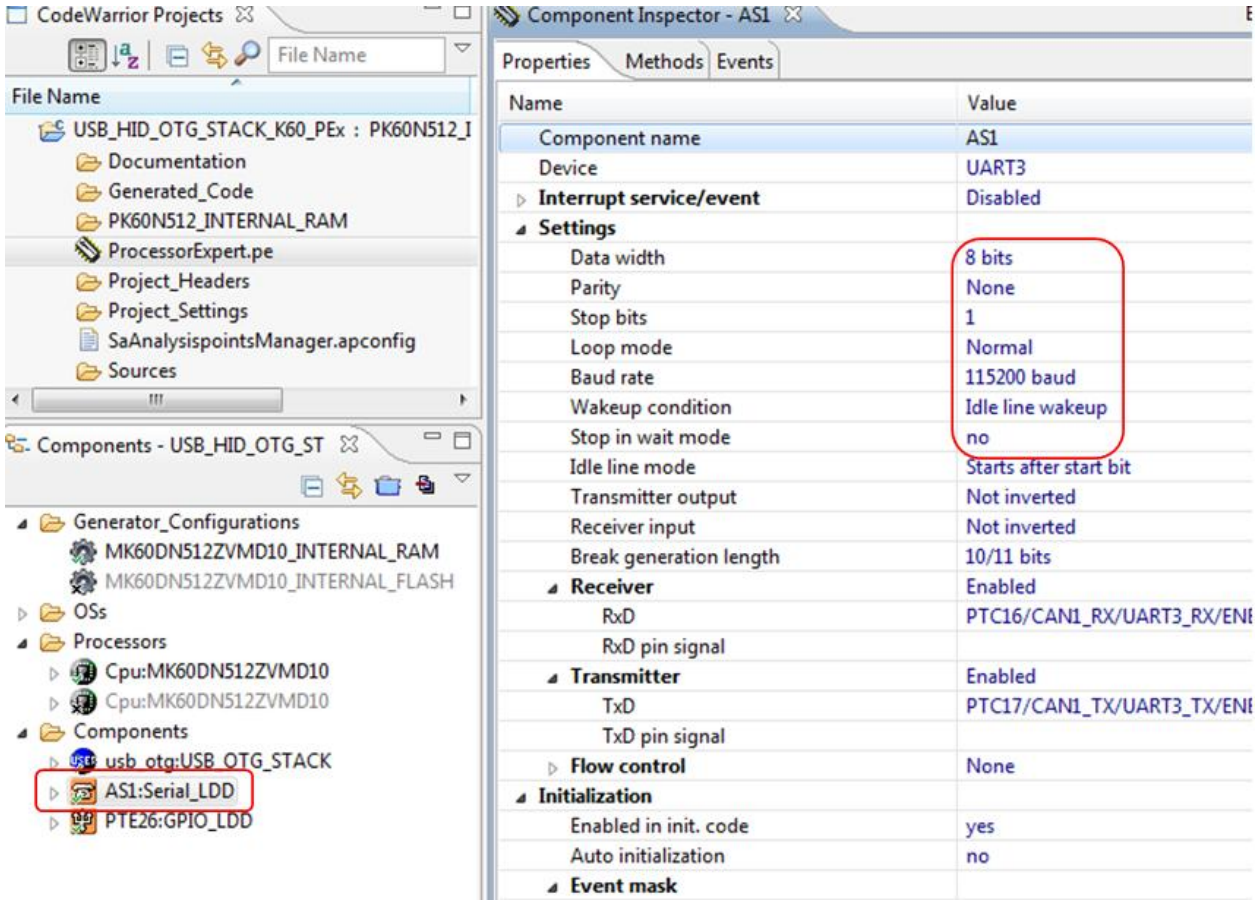


Figure 2-23 Serial_LDD component settings

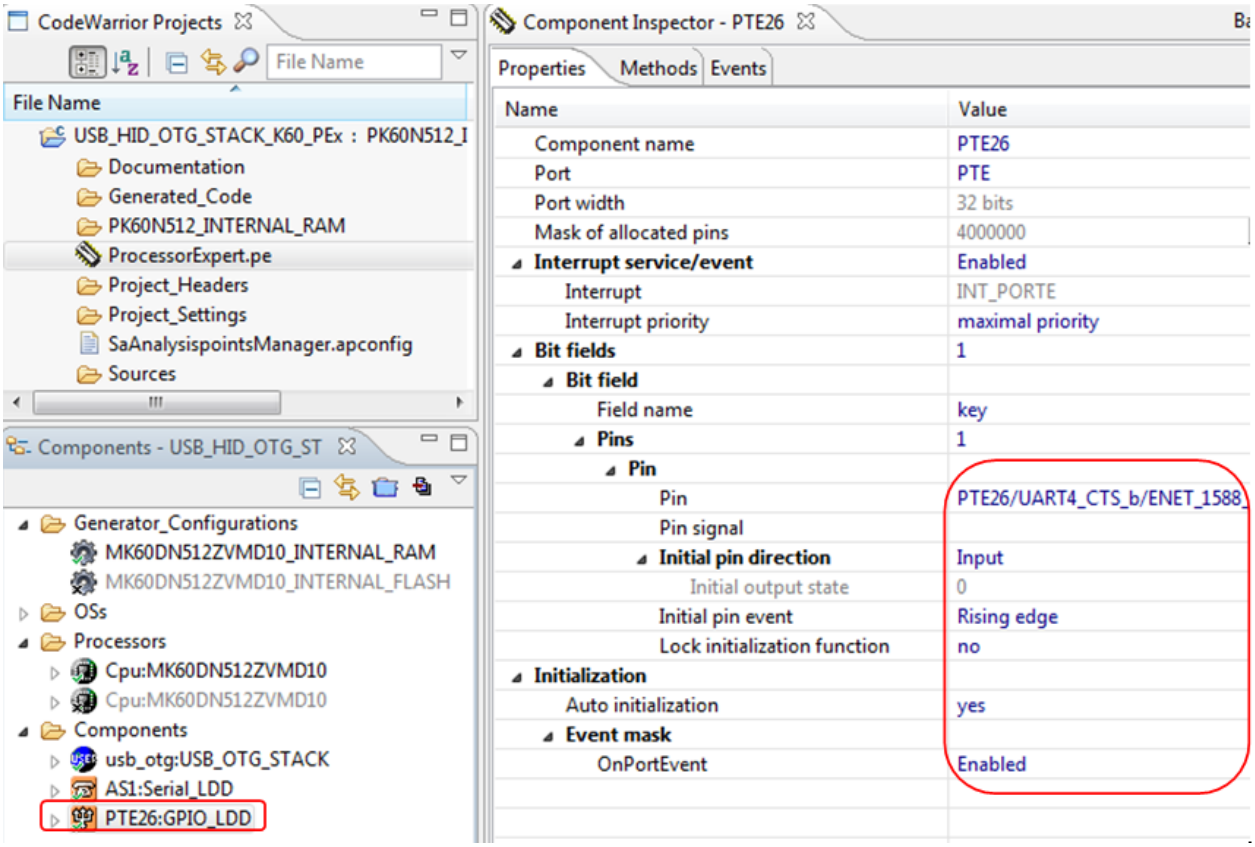


Figure 2-24 GPIO_LDD component settings

Step11. Generate Processor Expert code:

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

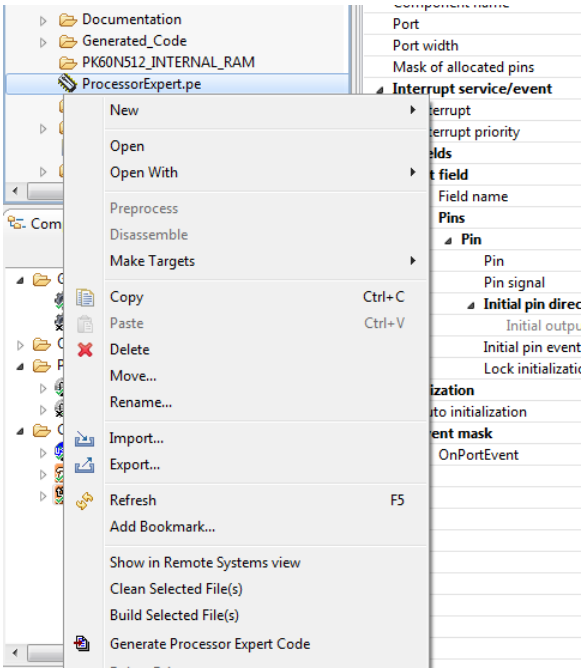


Figure 2-25 Generate Processor Expert code

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_DESCRIPTORS+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_DESCRIPTORS+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:
    - 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.
    - 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;
```

    - 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.

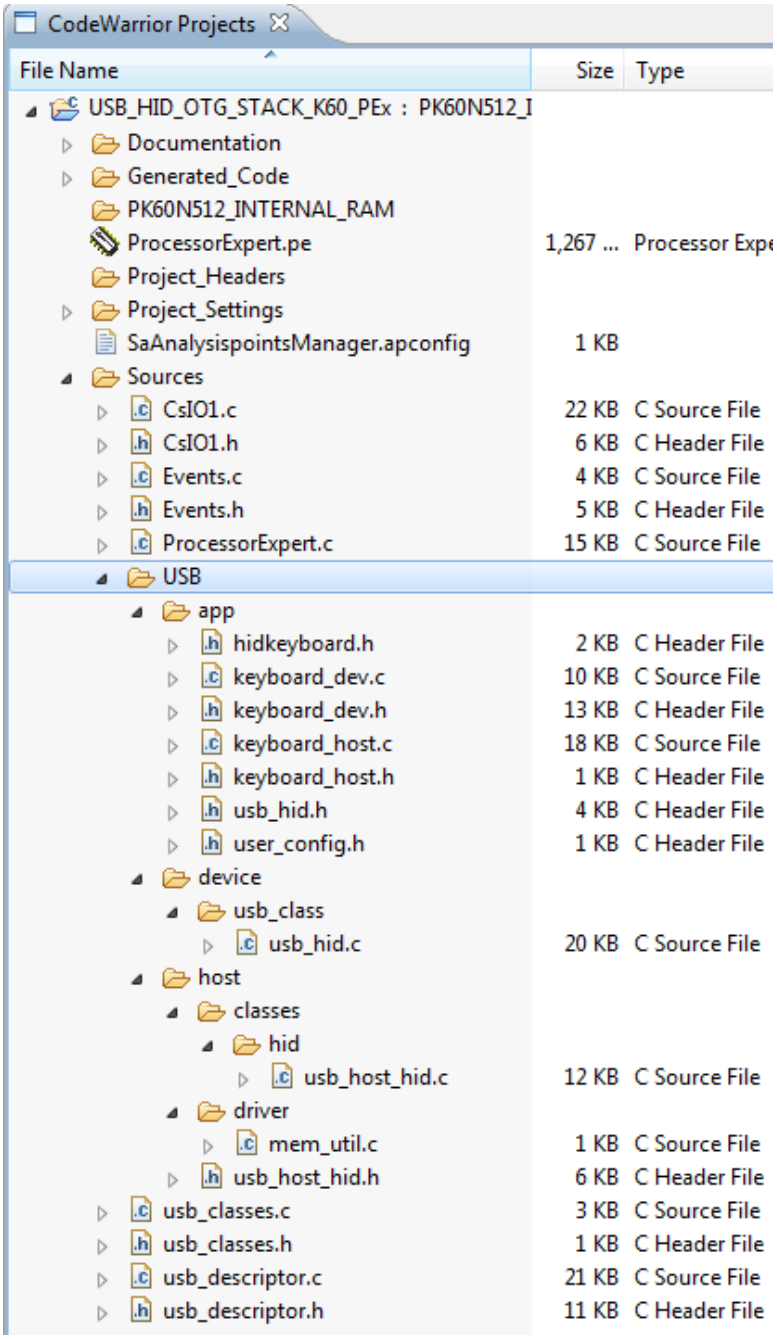


Figure 2-26 USB_HID_OTG included files

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.

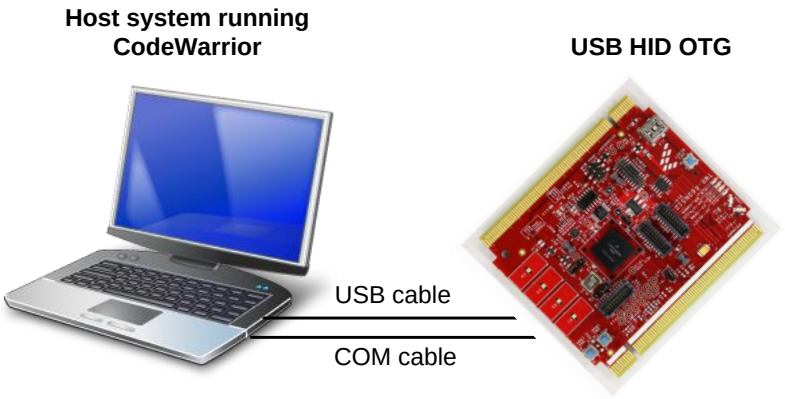


Figure 2-27 Hardware setup

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.

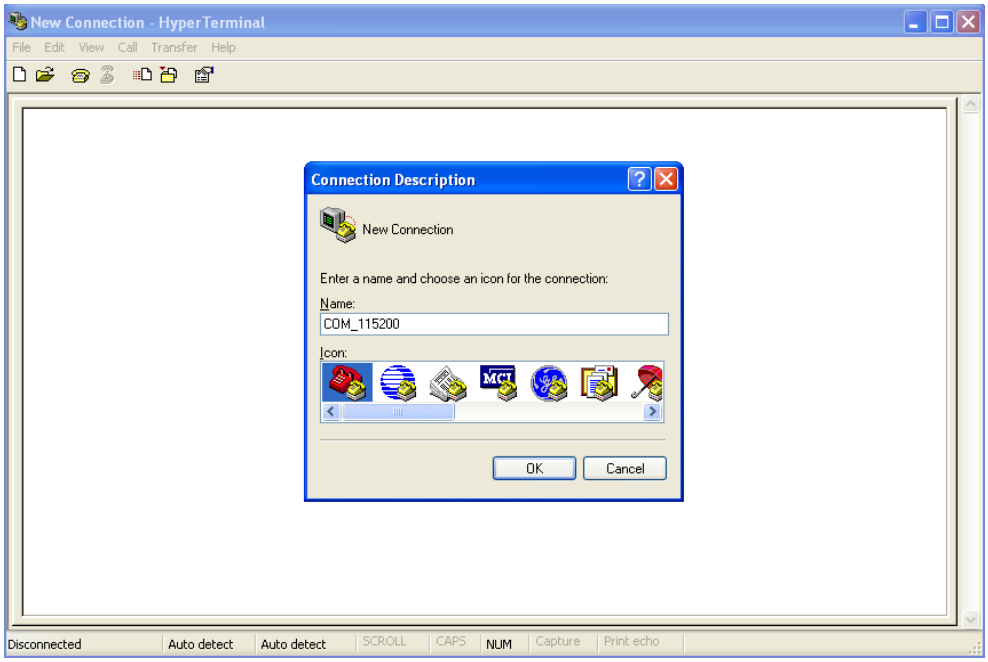


Figure 2-28 HyperTerminal startup

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

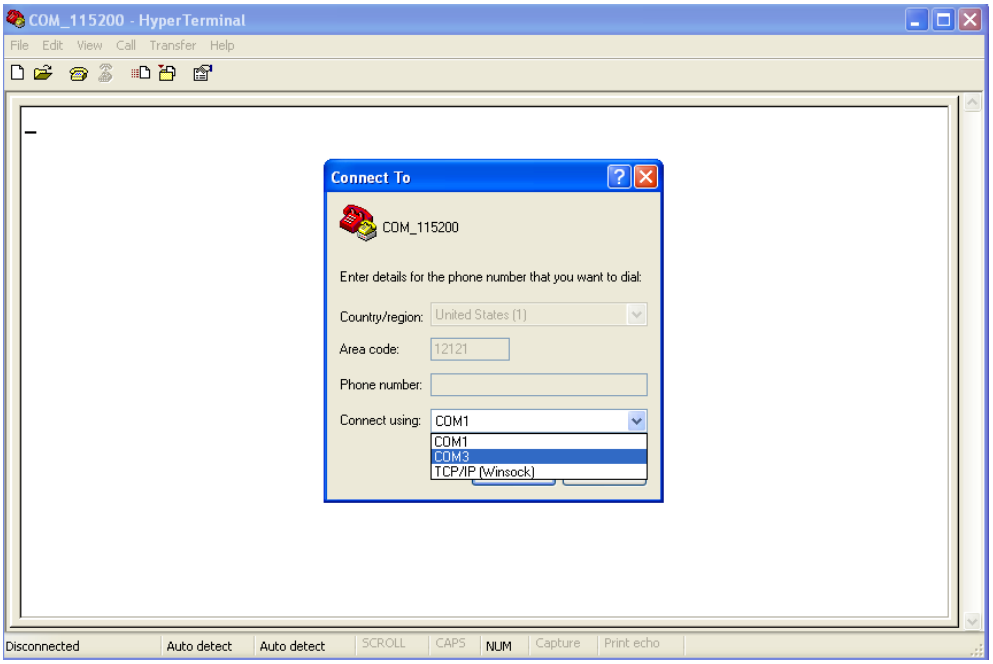


Figure 2-29 Connect using COM port

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

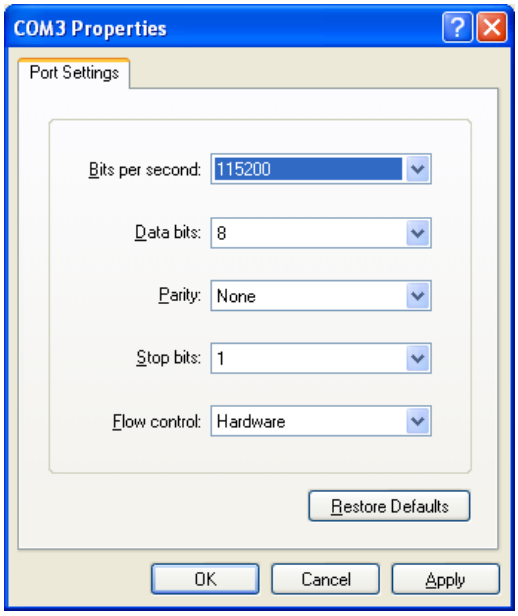


Figure 2-30 COM properties

- The HyperTerminal is now configured as shown below:

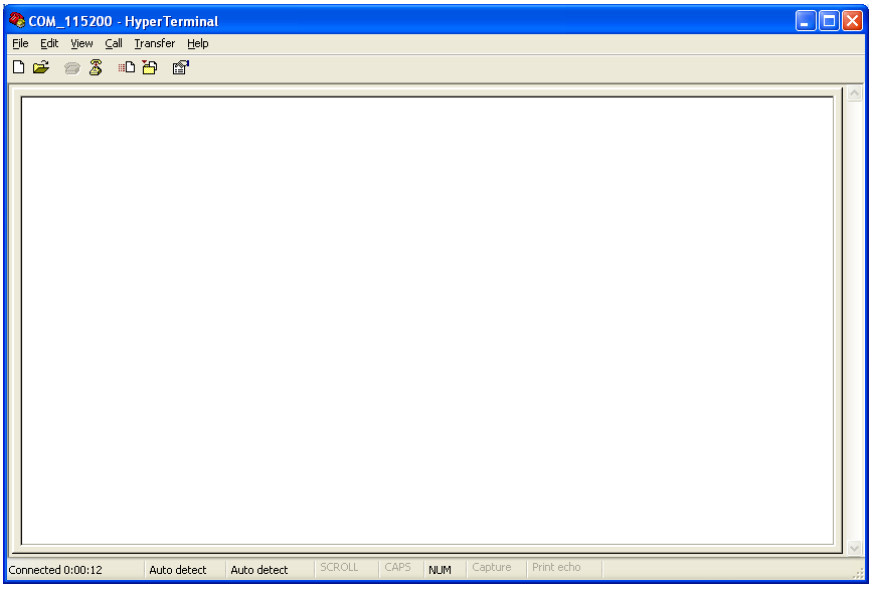


Figure 2-31 Hyper Terminal

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:

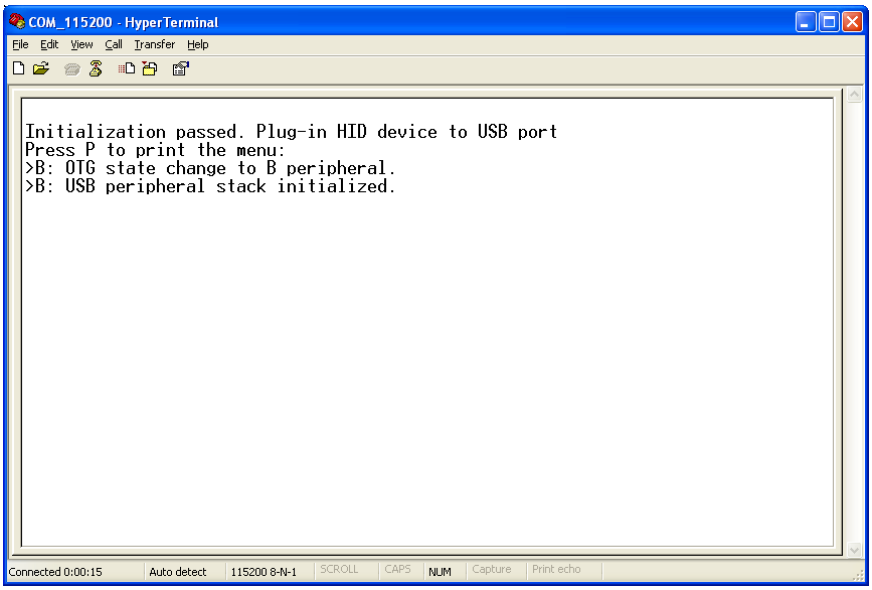
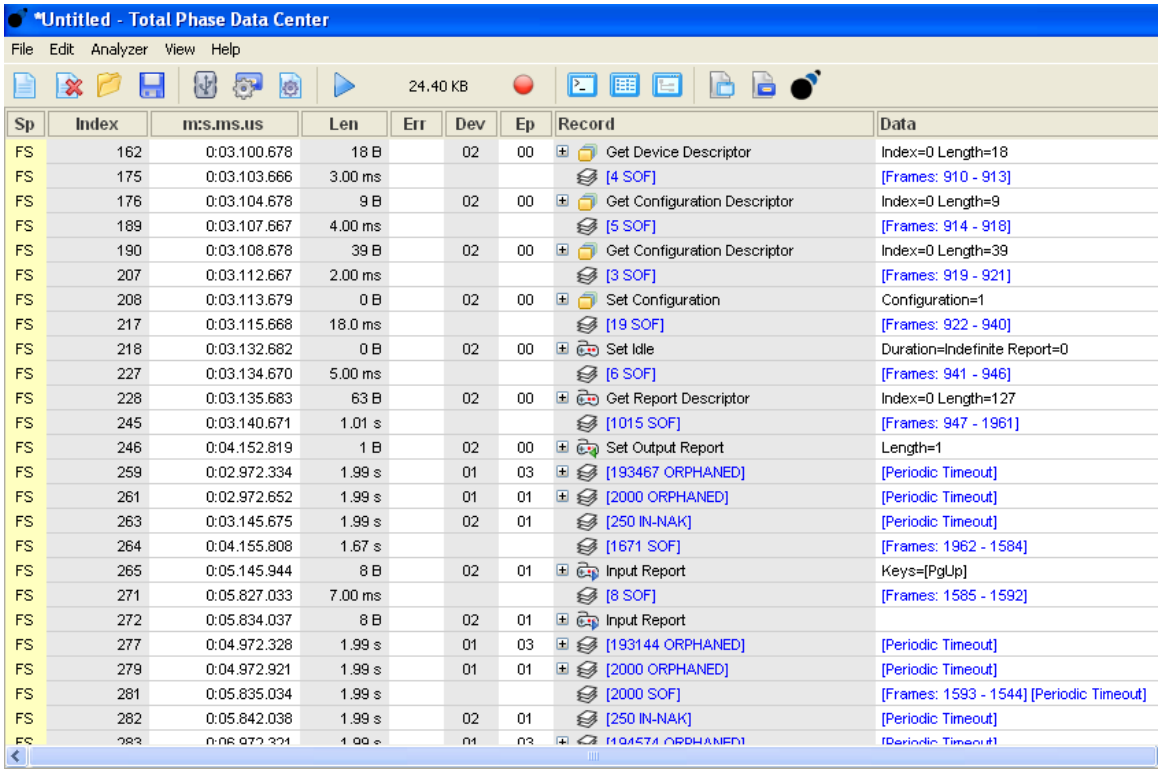


Figure 2-32 USB HID OTG initializing

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				[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]

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.

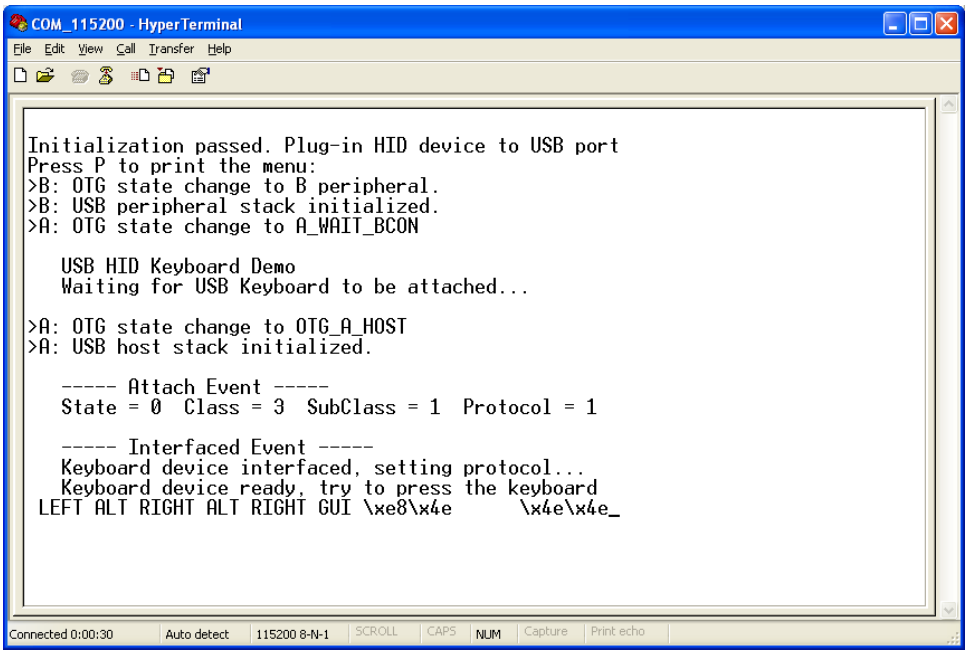


Figure 2-34 USB HID OTG operation

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.

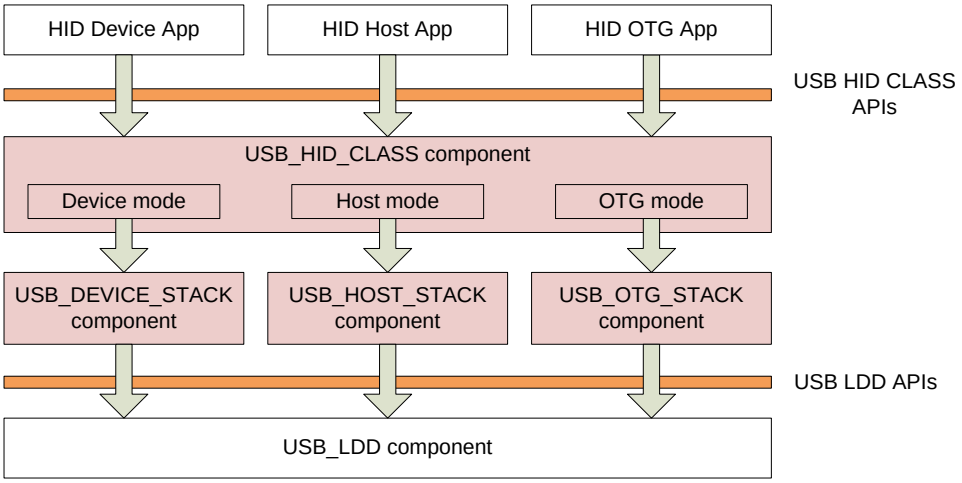


Figure 3-1 Architecture overview

3.2 Folder structure

3.2.1 Delivered package structure

The delivered package has the structure as the following:

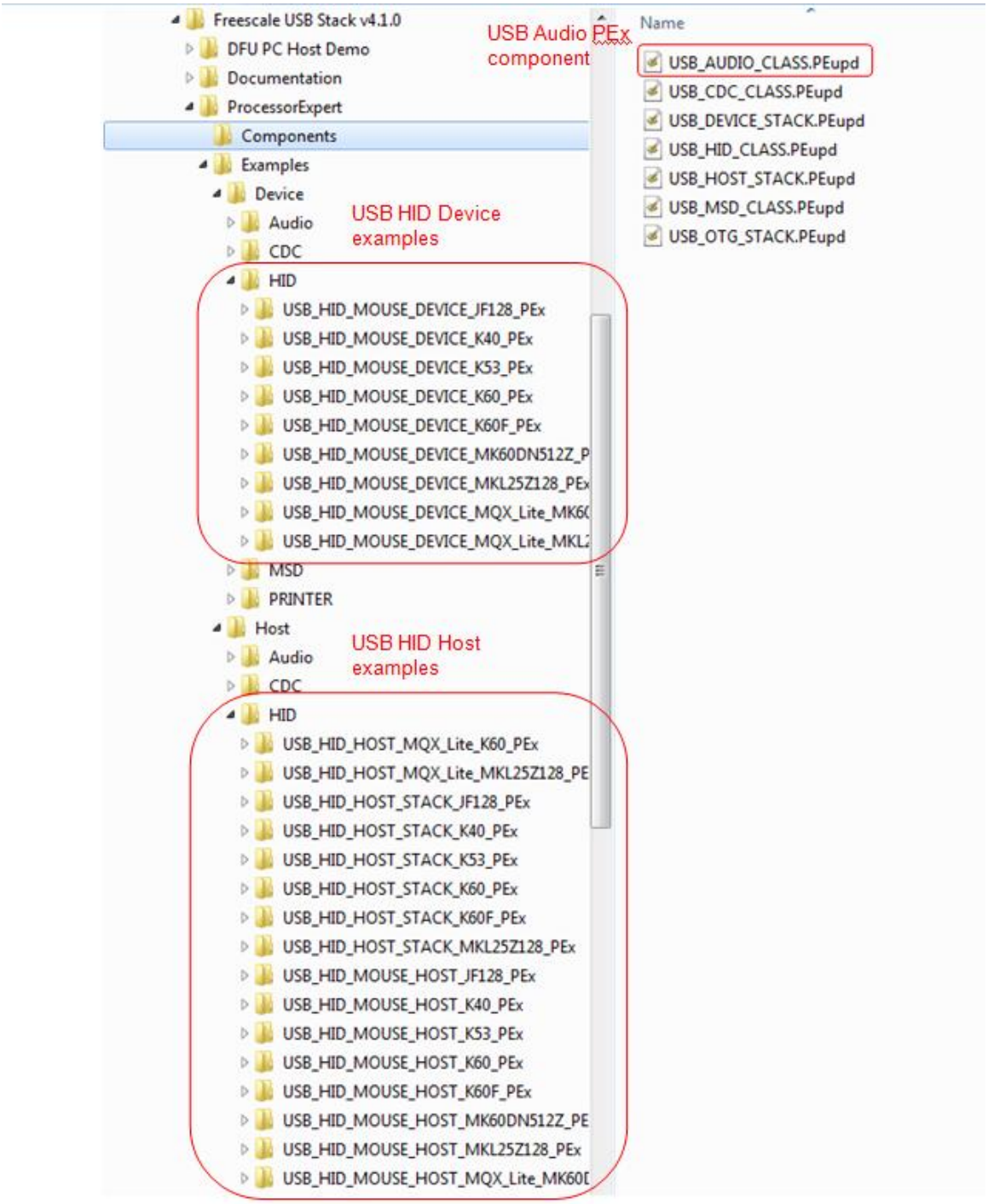


Figure 3-2 Delivered package folder structure

- 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:

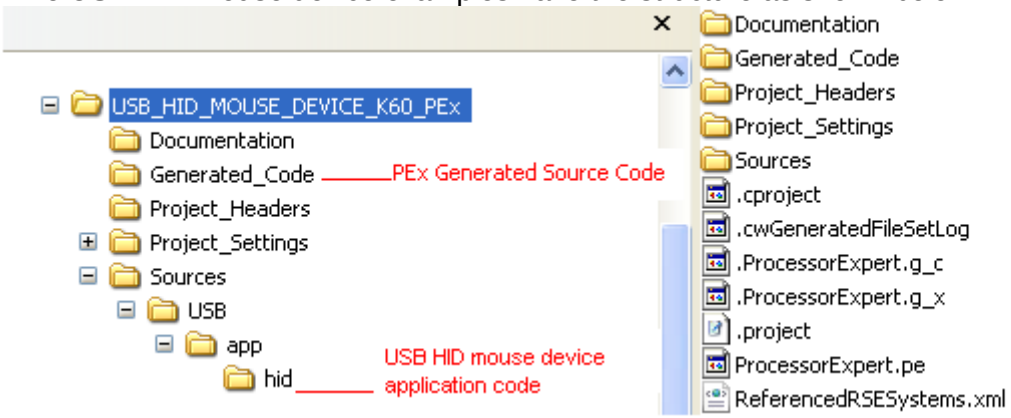


Figure 3-3 USB HID mouse device example folder structure

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

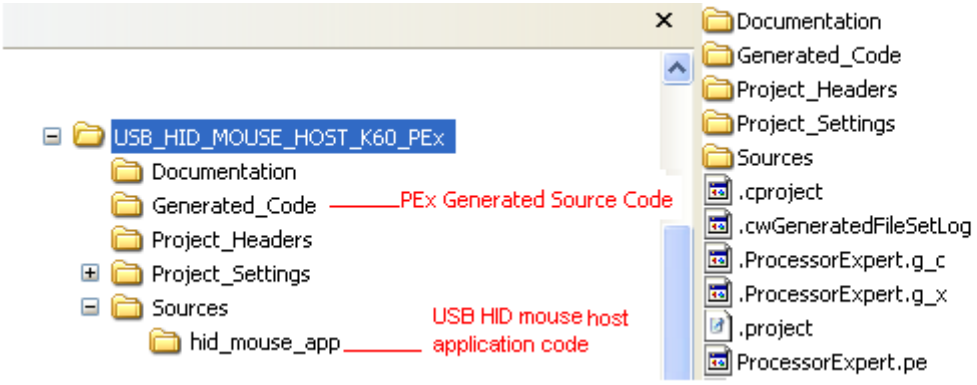


Figure 3-4 USB HID mouse host example folder structure

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

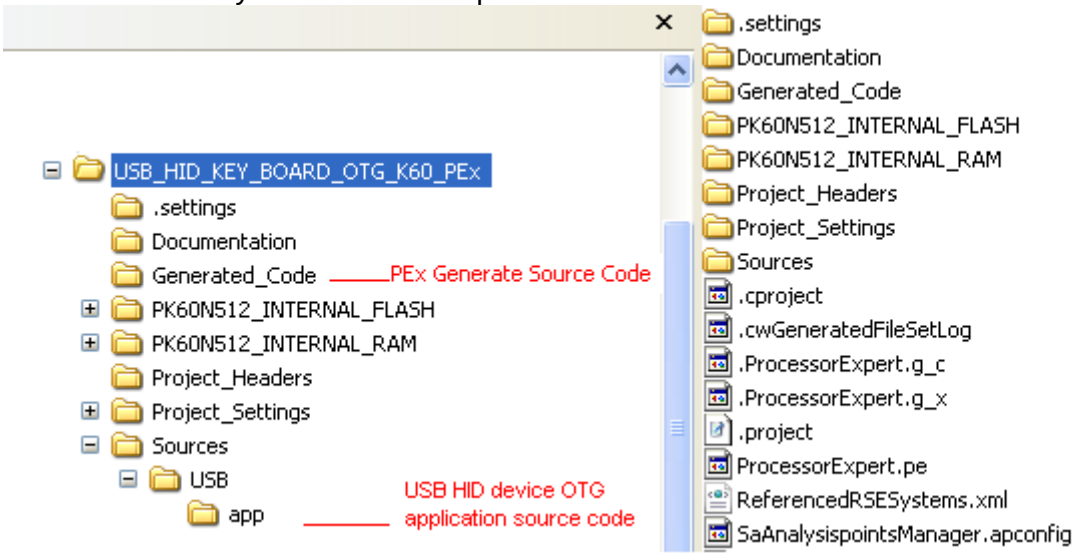


Figure 3-5 USB HID keyboard OTG example folder structure

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:

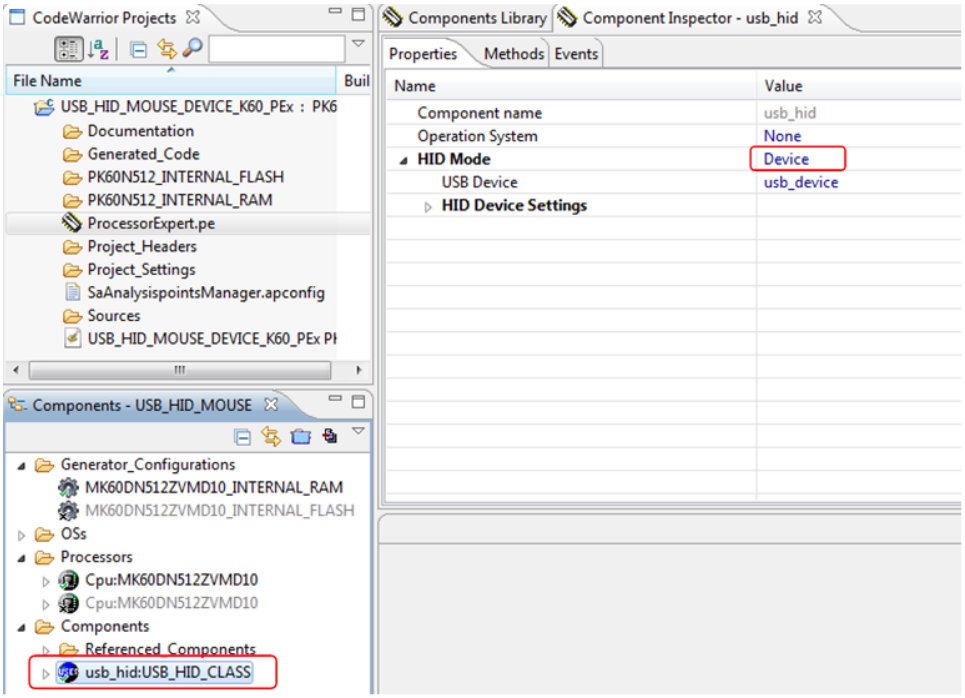


Figure 3-6 USB HID device mode

Step3. Configure CPU clock to run USB module:

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

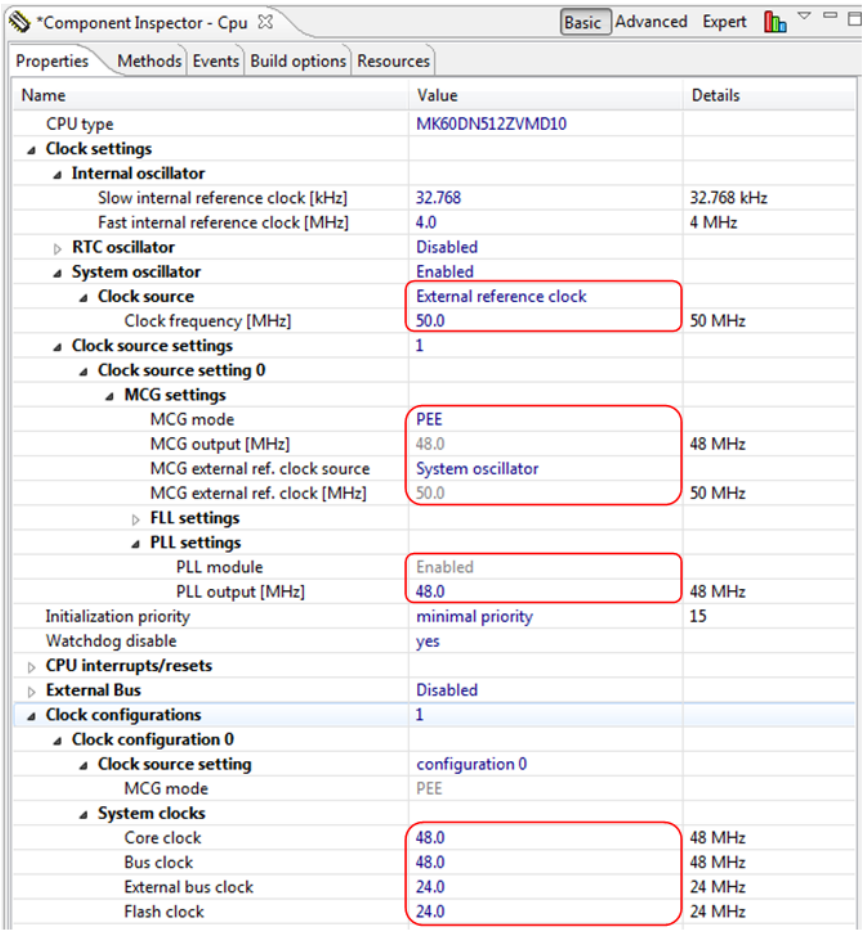


Figure 3-7 USB Clock settings

Step4. Configure the USB_HID_CLASS component:

- Configure the USB HID standard descriptor as shown below:

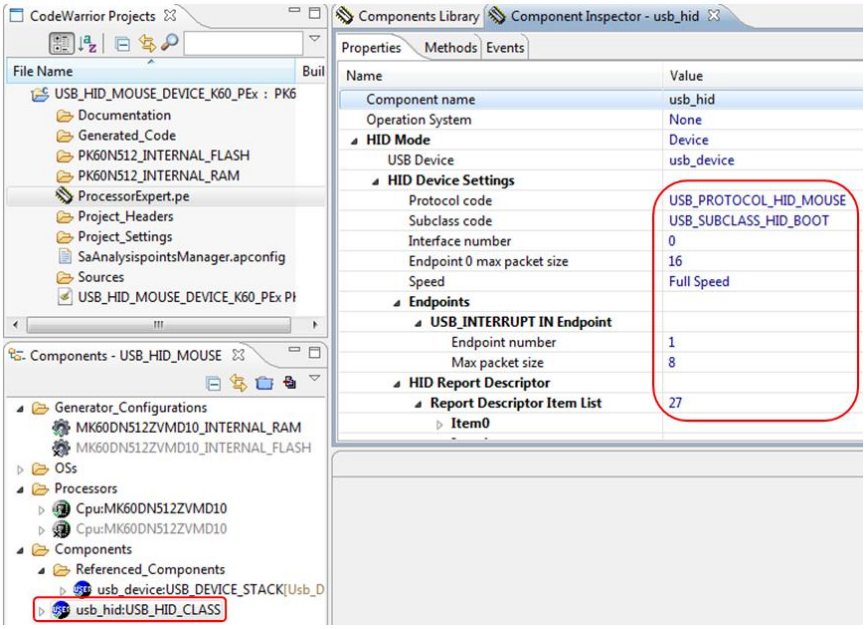


Figure 3-8 USB HID standard descriptor settings

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

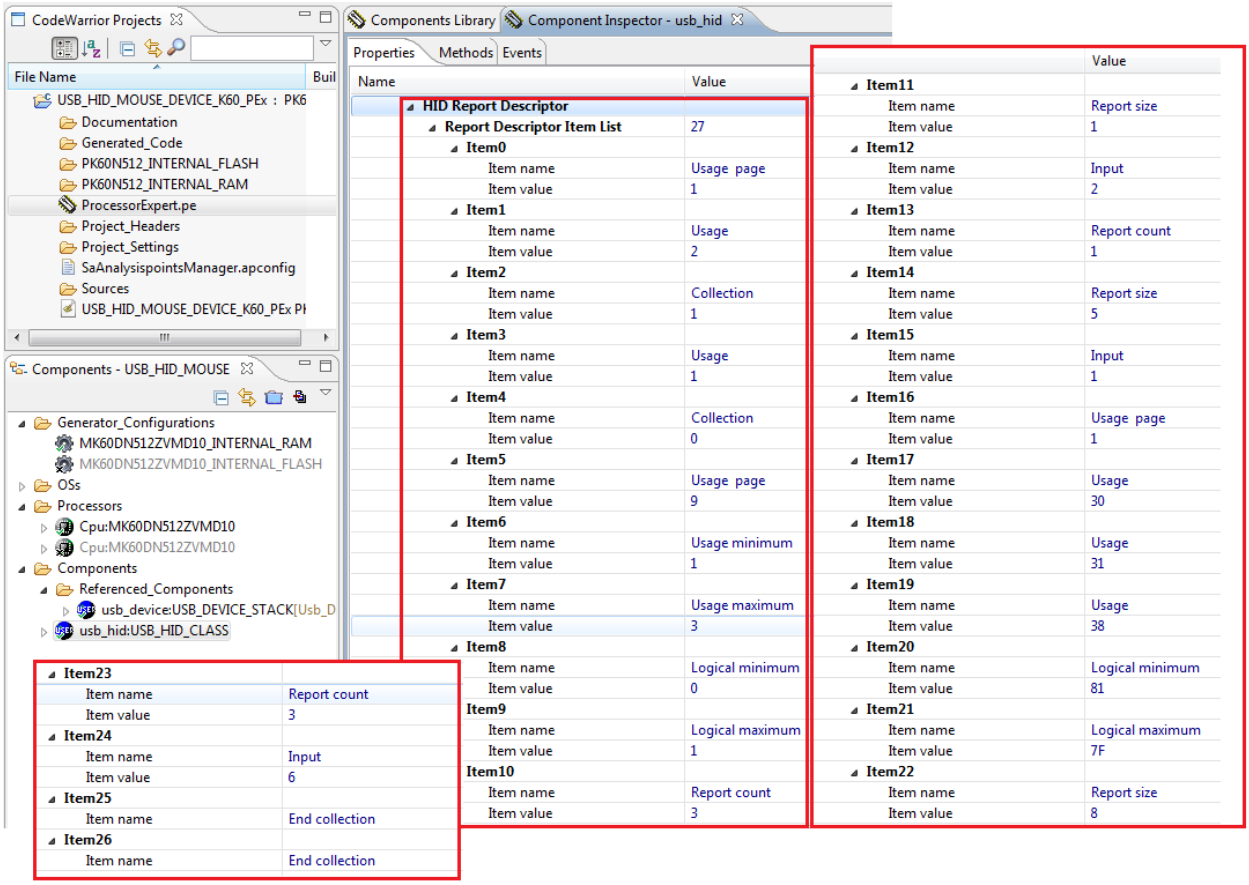


Figure 3-9 USB HID mouse report descriptor settings

Step5. Configure the USB_DEVICE_STACK to run USB HID example

❖ Configure the USB device descriptor as in the following figure:

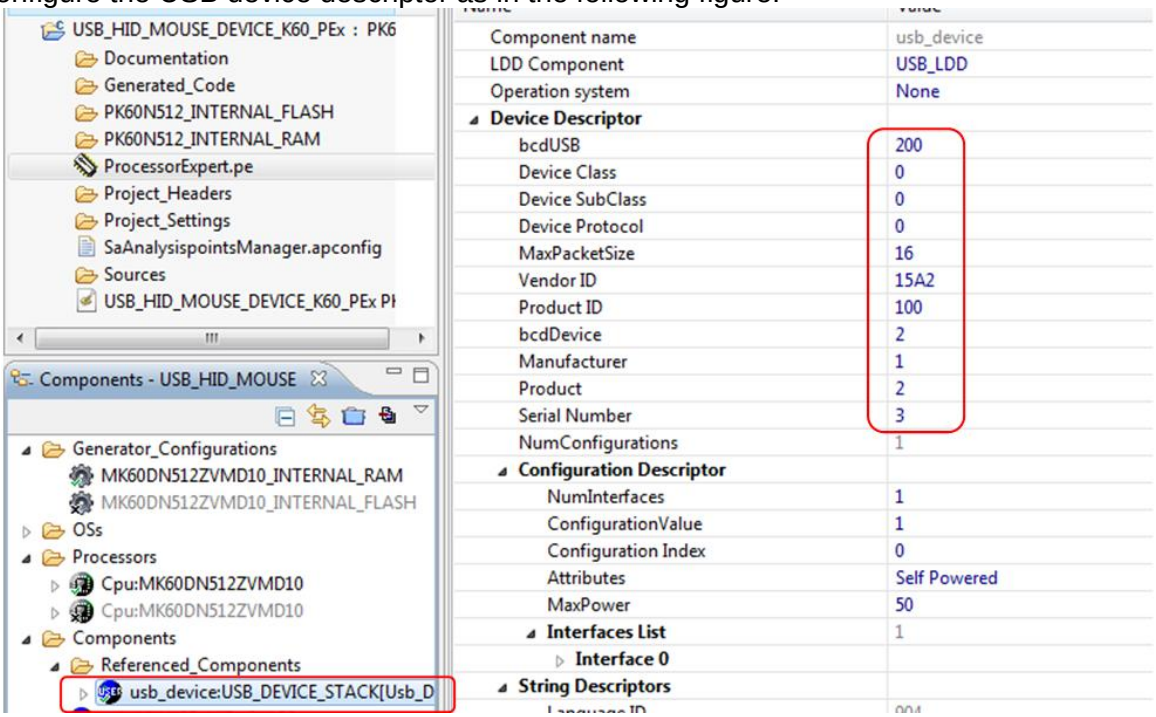


Figure 3-10 USB Device Descriptor settings

❖ Configure the USB HID class specific descriptor. This descriptor is inherited from the USB_HID_CLASS component.

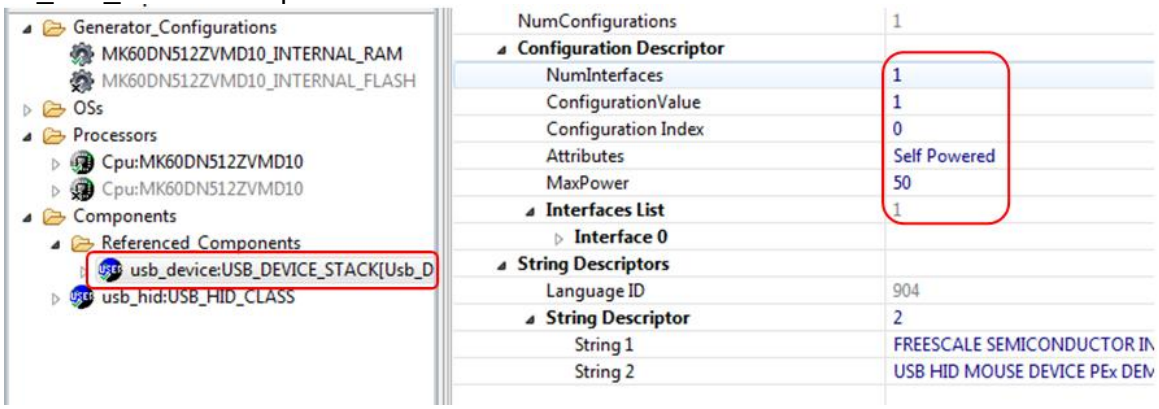


Figure 3-11 USB HID Class specific descriptor settings

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

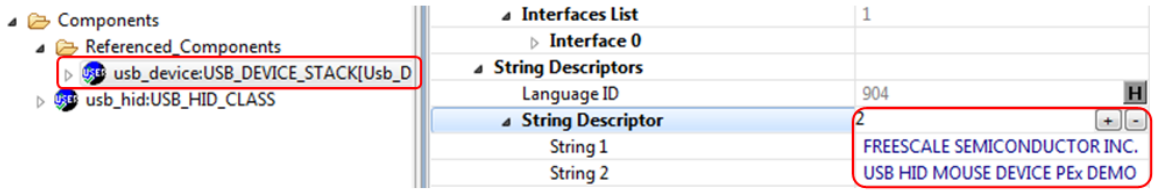


Figure 3-12 USB String Descriptor settings

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

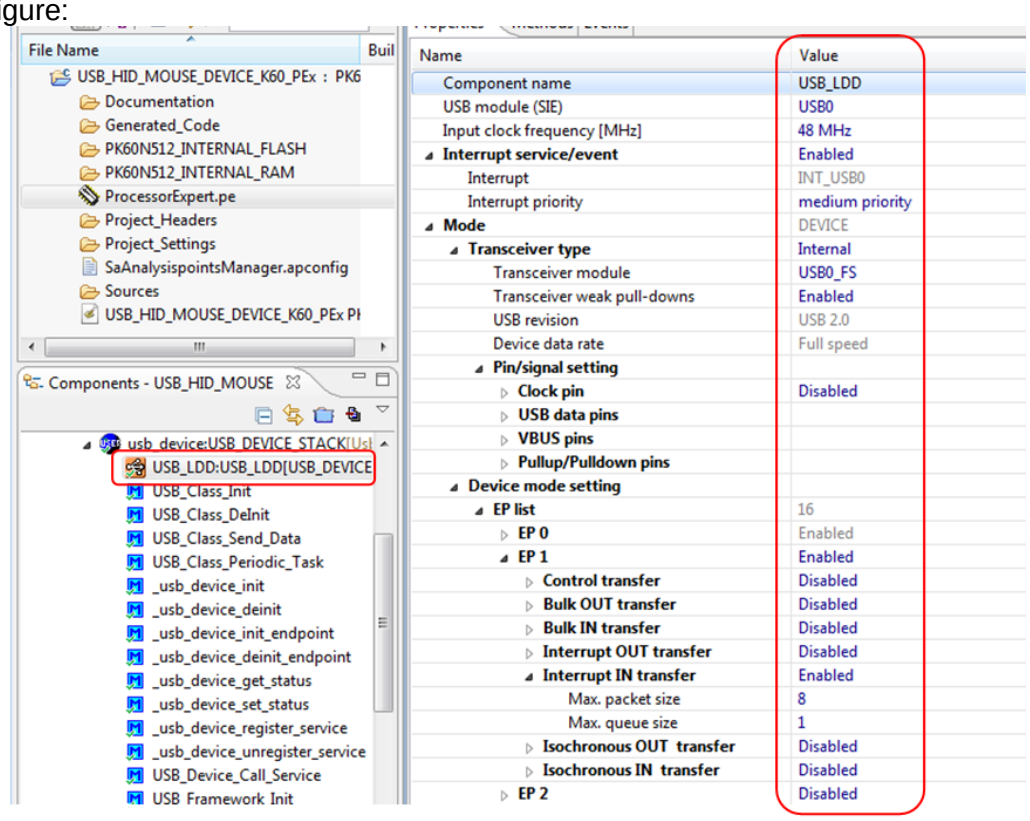


Figure 3-13 USB_LDD component device mode settings

Step7. Generate Processor Expert code:
After the setting of the components, select ProcessorExpert.pe in the project folder → Right click → Chose Generate Processor Expert Code field to generate source code.

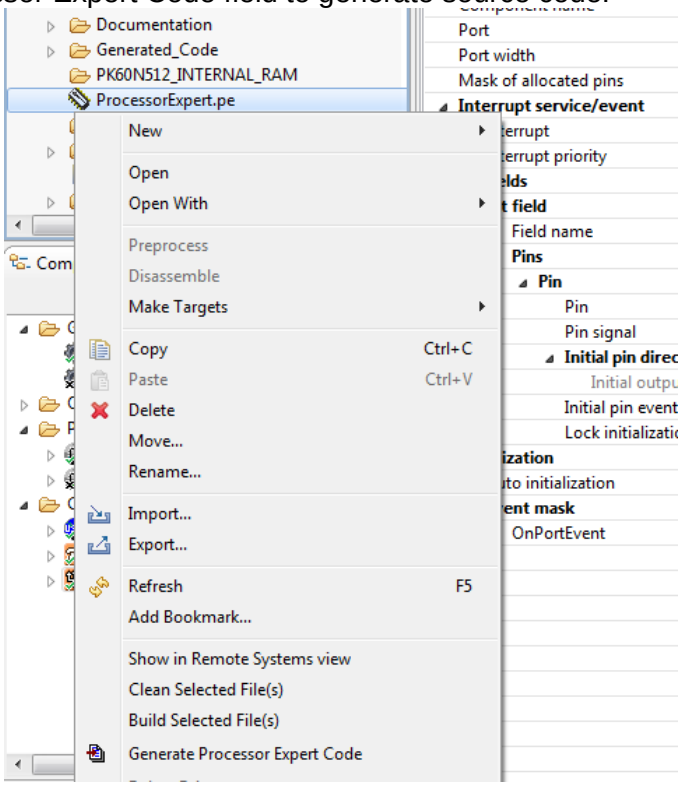


Figure 3-14 Generate Processor Expert Code

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:

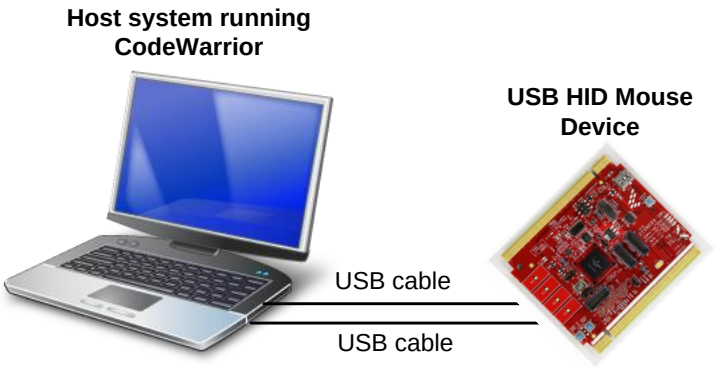
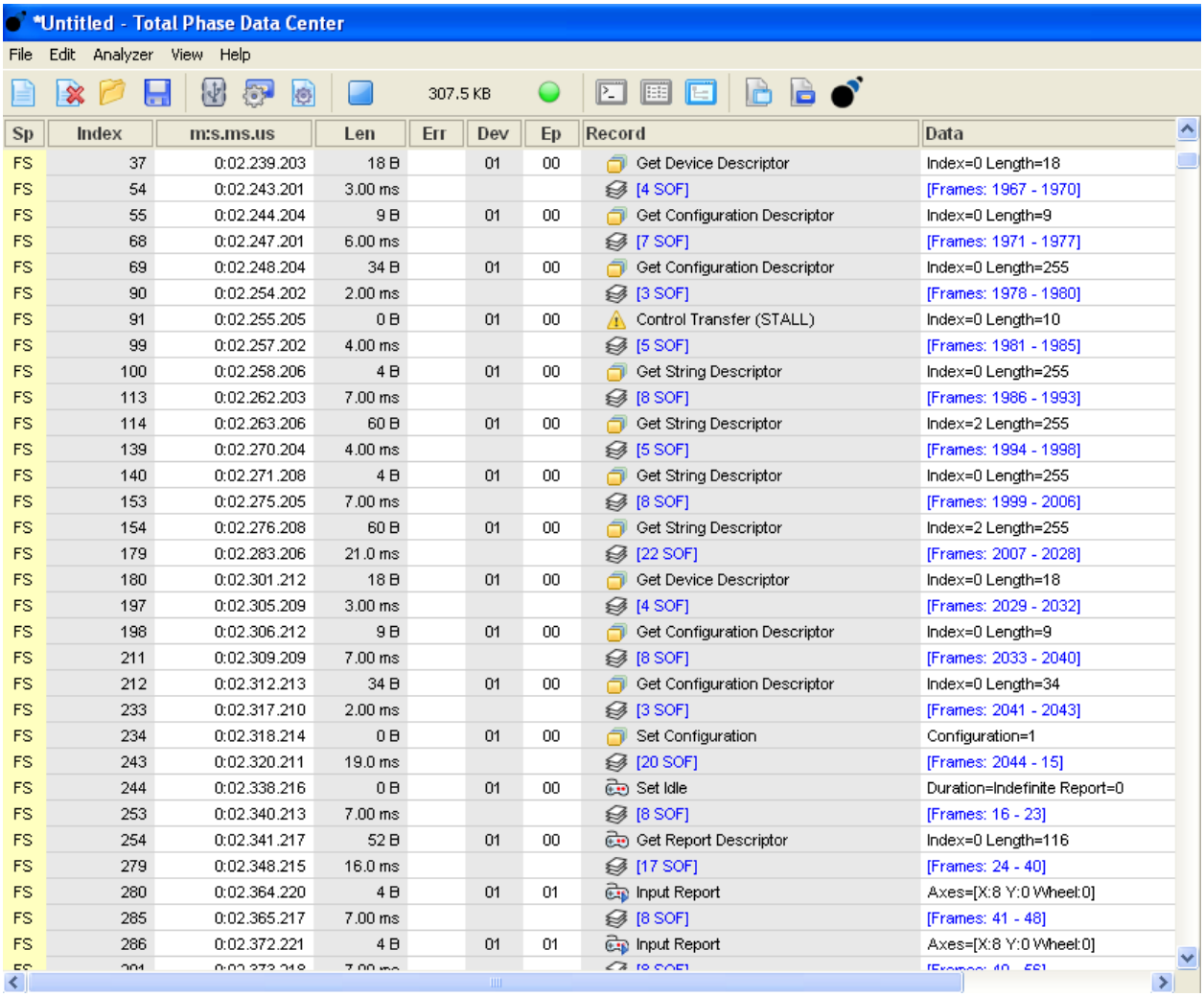


Figure 3-15 HID mouse device demo setup

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:



Sp	Index	m:s.ms.us	Len	Err	Dev	Ep	Record	Data
FS	37	0:02.239.203	18 B		01	00	Get Device Descriptor	Index=0 Length=18
FS	54	0:02.243.201	3.00 ms				[4 SOF]	[Frames: 1967 - 1970]
FS	55	0:02.244.204	9 B		01	00	Get Configuration Descriptor	Index=0 Length=9
FS	68	0:02.247.201	6.00 ms				[7 SOF]	[Frames: 1971 - 1977]
FS	69	0:02.248.204	34 B		01	00	Get Configuration Descriptor	Index=0 Length=255
FS	90	0:02.254.202	2.00 ms				[3 SOF]	[Frames: 1978 - 1980]
FS	91	0:02.255.205	0 B		01	00	Control Transfer (STALL)	Index=0 Length=10
FS	99	0:02.257.202	4.00 ms				[5 SOF]	[Frames: 1981 - 1985]
FS	100	0:02.258.206	4 B		01	00	Get String Descriptor	Index=0 Length=255
FS	113	0:02.262.203	7.00 ms				[8 SOF]	[Frames: 1986 - 1993]
FS	114	0:02.263.206	60 B		01	00	Get String Descriptor	Index=2 Length=255
FS	139	0:02.270.204	4.00 ms				[5 SOF]	[Frames: 1994 - 1998]
FS	140	0:02.271.208	4 B		01	00	Get String Descriptor	Index=0 Length=255
FS	153	0:02.275.205	7.00 ms				[8 SOF]	[Frames: 1999 - 2006]
FS	154	0:02.276.208	60 B		01	00	Get String Descriptor	Index=2 Length=255
FS	179	0:02.283.206	21.0 ms				[22 SOF]	[Frames: 2007 - 2028]
FS	180	0:02.301.212	18 B		01	00	Get Device Descriptor	Index=0 Length=18
FS	197	0:02.305.209	3.00 ms				[4 SOF]	[Frames: 2029 - 2032]
FS	198	0:02.306.212	9 B		01	00	Get Configuration Descriptor	Index=0 Length=9
FS	211	0:02.309.209	7.00 ms				[8 SOF]	[Frames: 2033 - 2040]
FS	212	0:02.312.213	34 B		01	00	Get Configuration Descriptor	Index=0 Length=34
FS	233	0:02.317.210	2.00 ms				[3 SOF]	[Frames: 2041 - 2043]
FS	234	0:02.318.214	0 B		01	00	Set Configuration	Configuration=1
FS	243	0:02.320.211	19.0 ms				[20 SOF]	[Frames: 2044 - 15]
FS	244	0:02.338.216	0 B		01	00	Set Idle	Duration=Indefinite Report=0
FS	253	0:02.340.213	7.00 ms				[8 SOF]	[Frames: 16 - 23]
FS	254	0:02.341.217	52 B		01	00	Get Report Descriptor	Index=0 Length=116
FS	279	0:02.348.215	16.0 ms				[17 SOF]	[Frames: 24 - 40]
FS	280	0:02.364.220	4 B		01	01	Input Report	Axes=[X:8 Y:0 Wheel:0]
FS	285	0:02.365.217	7.00 ms				[8 SOF]	[Frames: 41 - 48]
FS	286	0:02.372.221	4 B		01	01	Input Report	Axes=[X:8 Y:0 Wheel:0]

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:

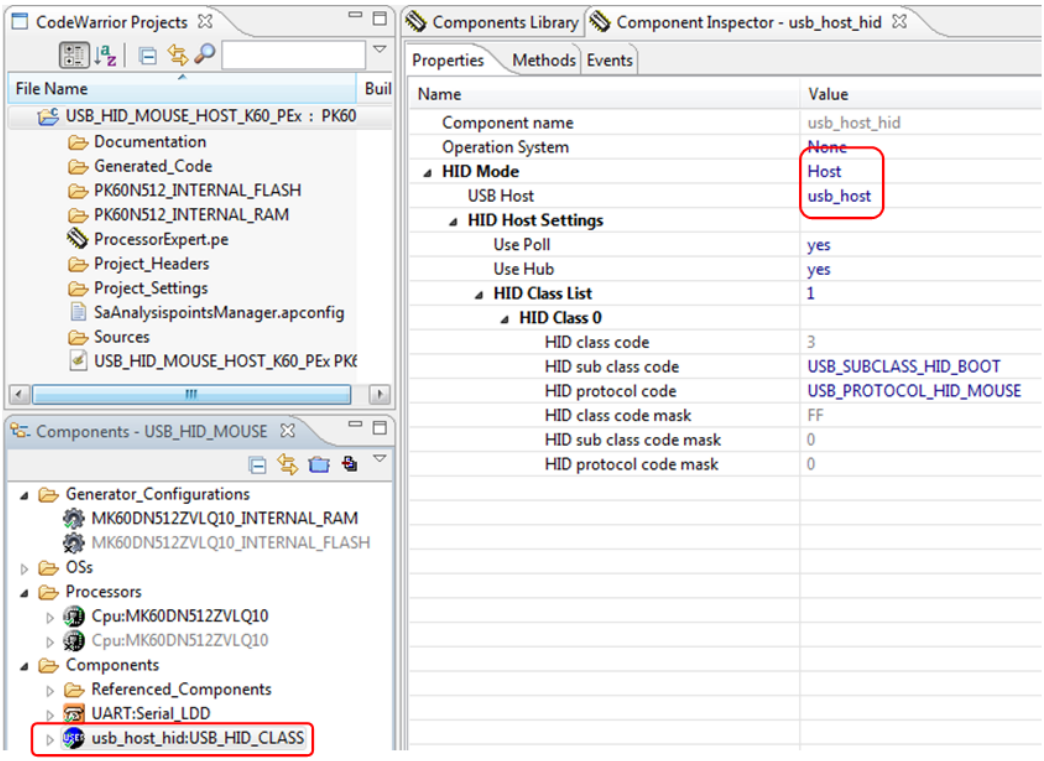


Figure 3-17 USB HID Host mode

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:

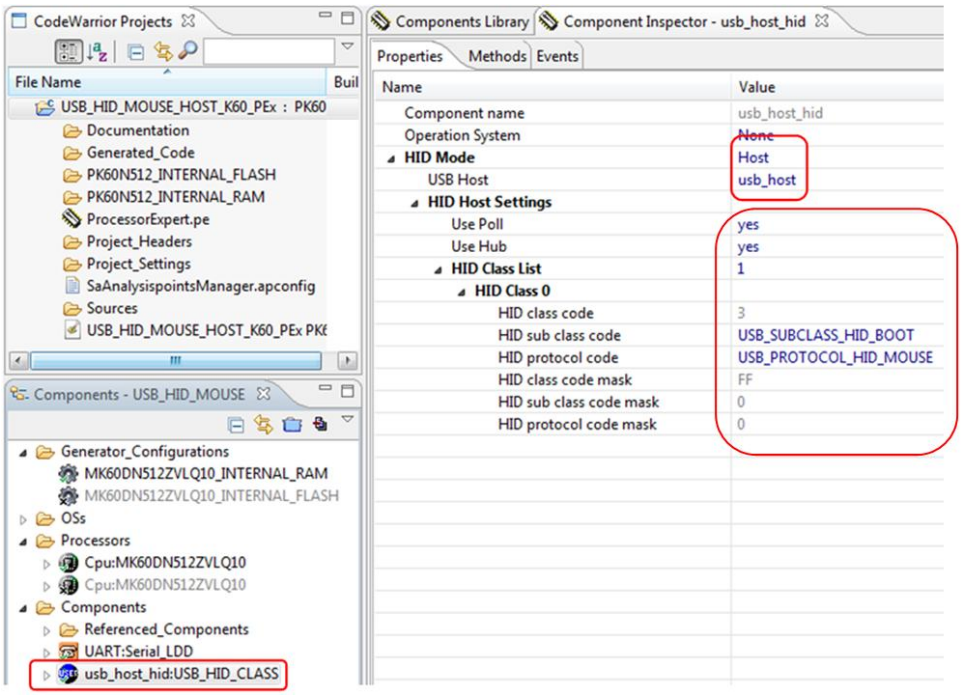


Figure 3-18 USB HID Host settings

Step4. Configure the USB_HOST_STACK to run the USB HID mouse host example. Those settings are inherited from the USB HID CLASS component.

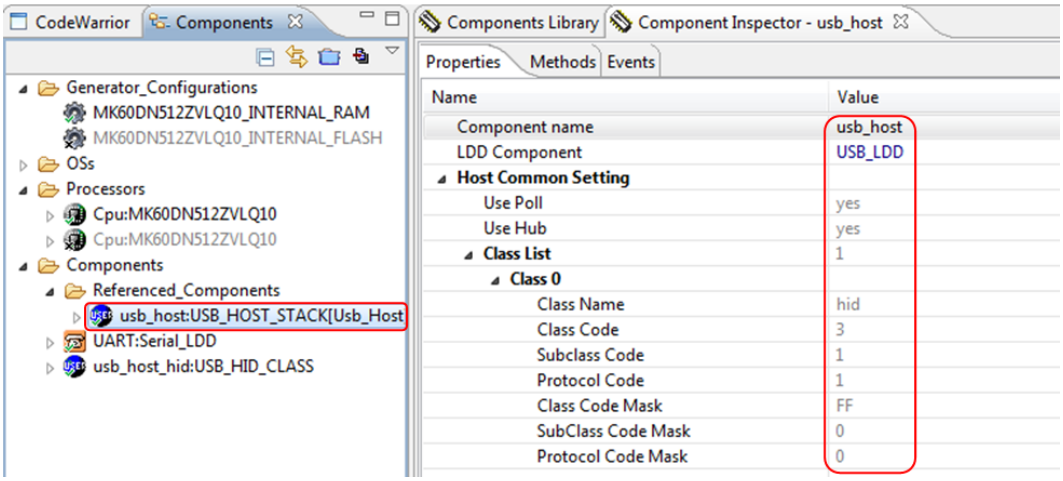


Figure 3-19 USB_HOST_STACK component settings

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

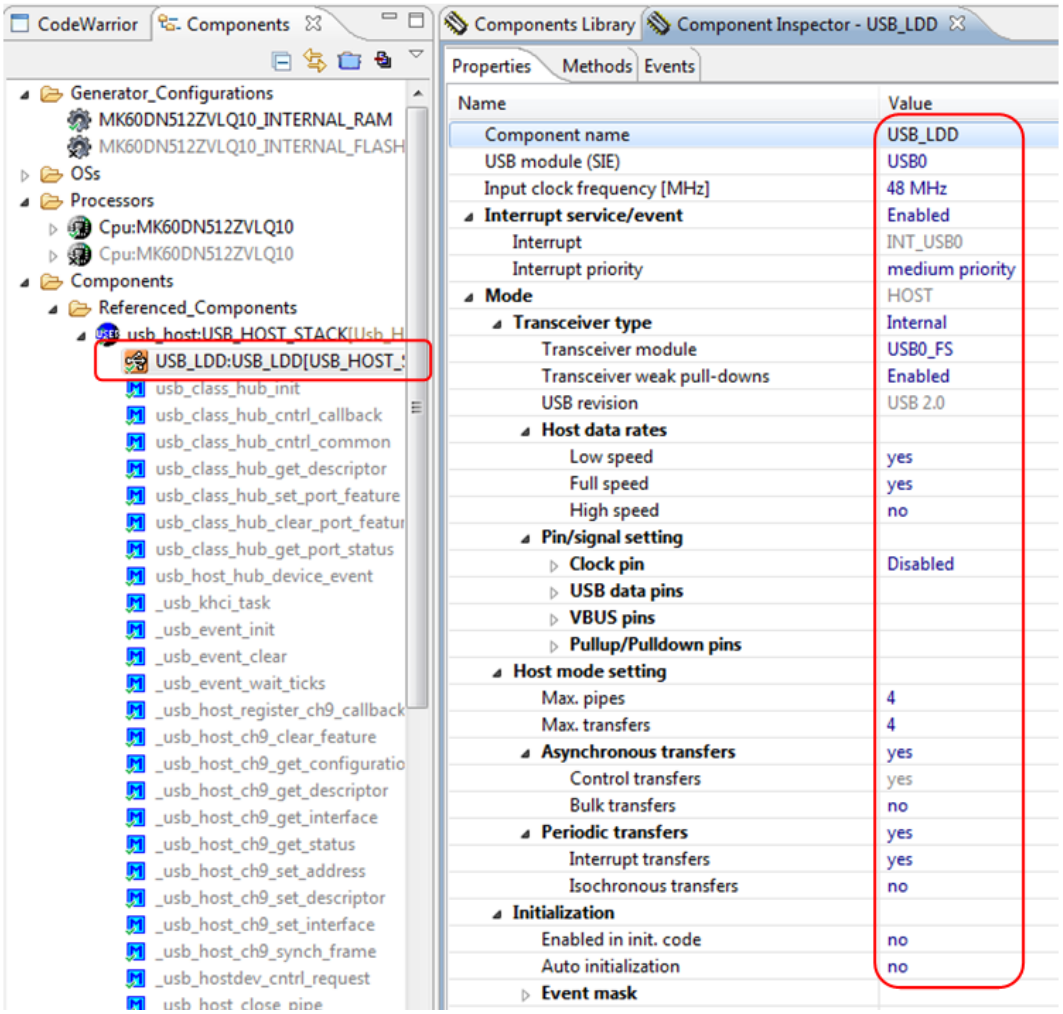


Figure 3-20 USB_LDD component host mode settings

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:
  - Data width: 8 bits
  - Parity: none
  - Stop bits: 1
  - Loop mode: Normal
  - Baud rate: 115200
  -

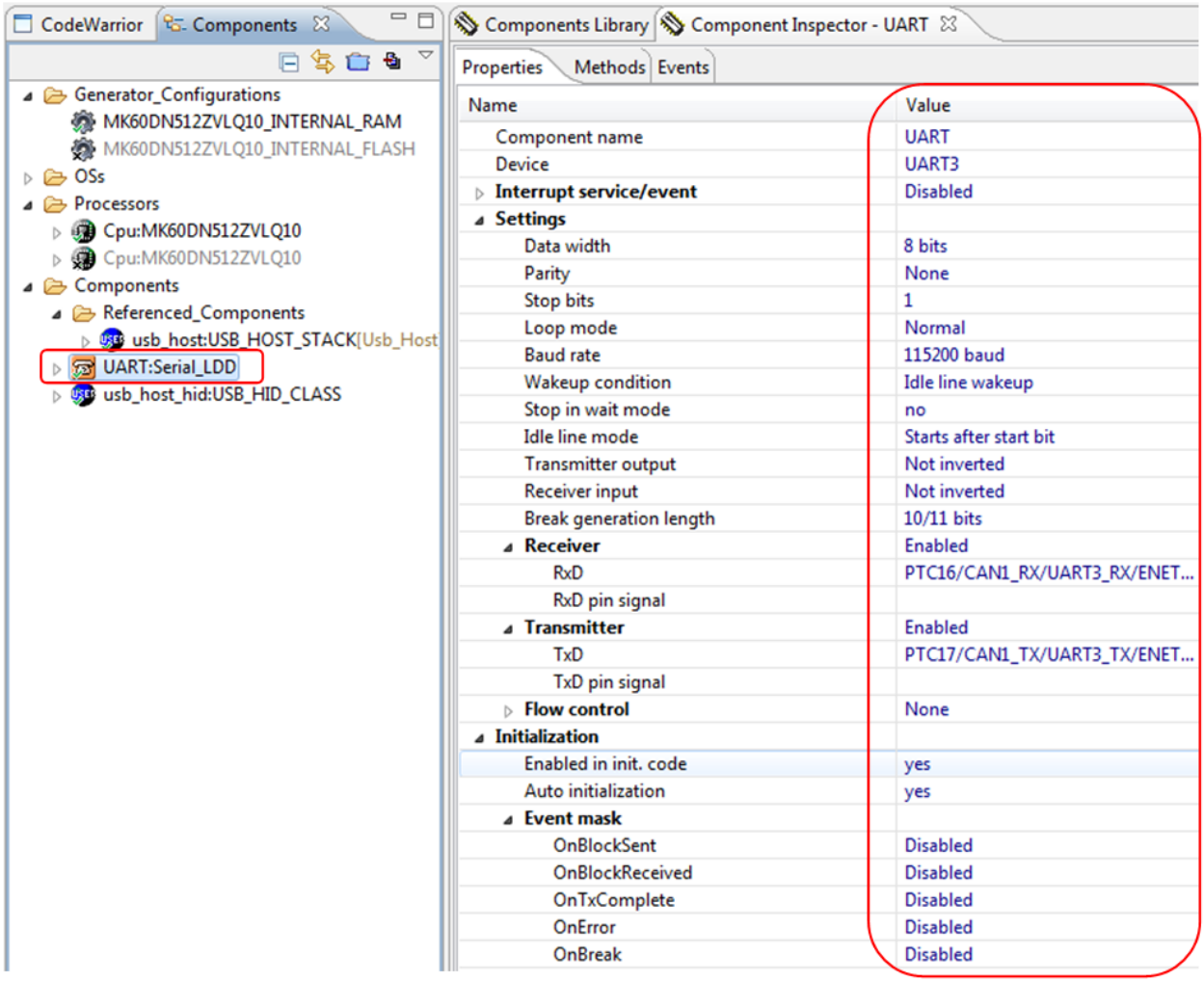


Figure 3-21 Serial_LDD component settings

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.

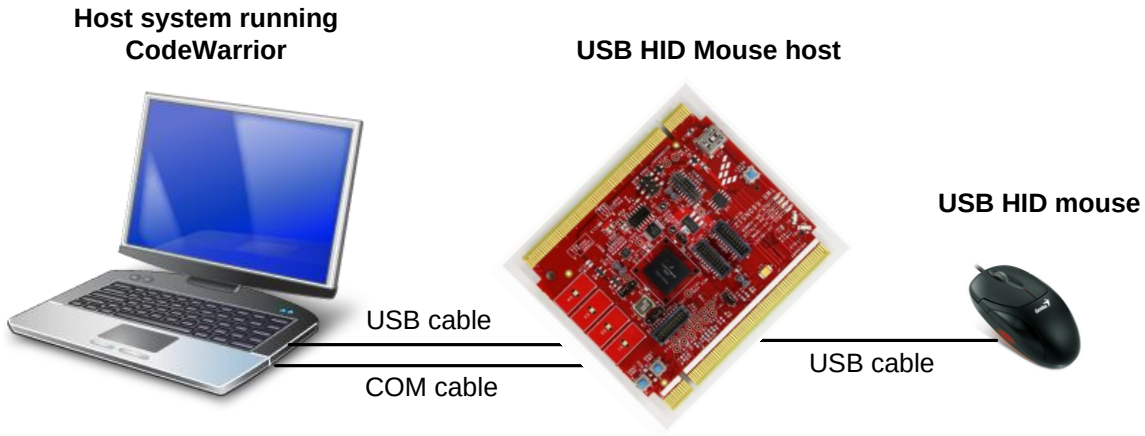


Figure 3-22 HID mouse host demo setup

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

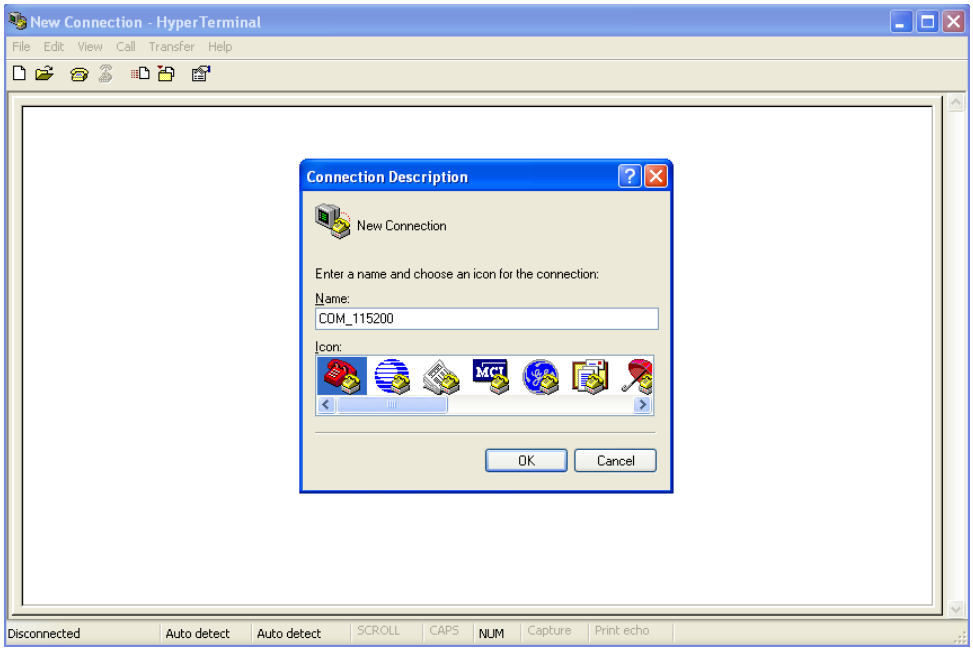


Figure 3-23 HyperTerminal startup

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

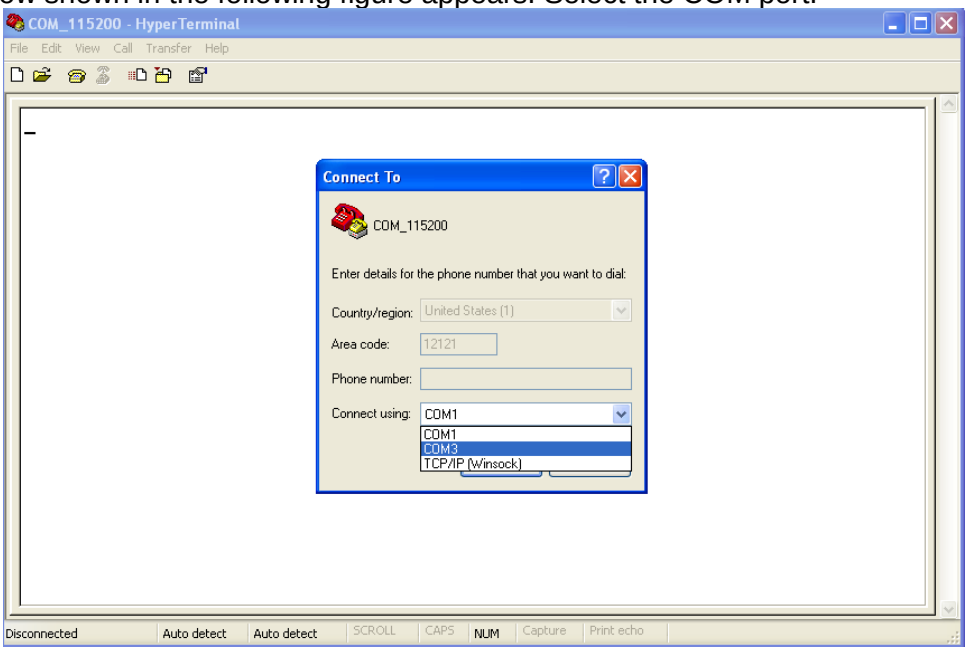


Figure 3-24 Connect using COM port

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

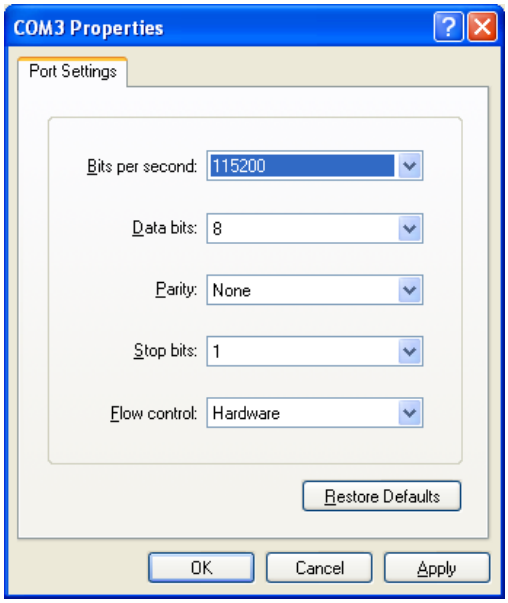


Figure 3-25 COM properties

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

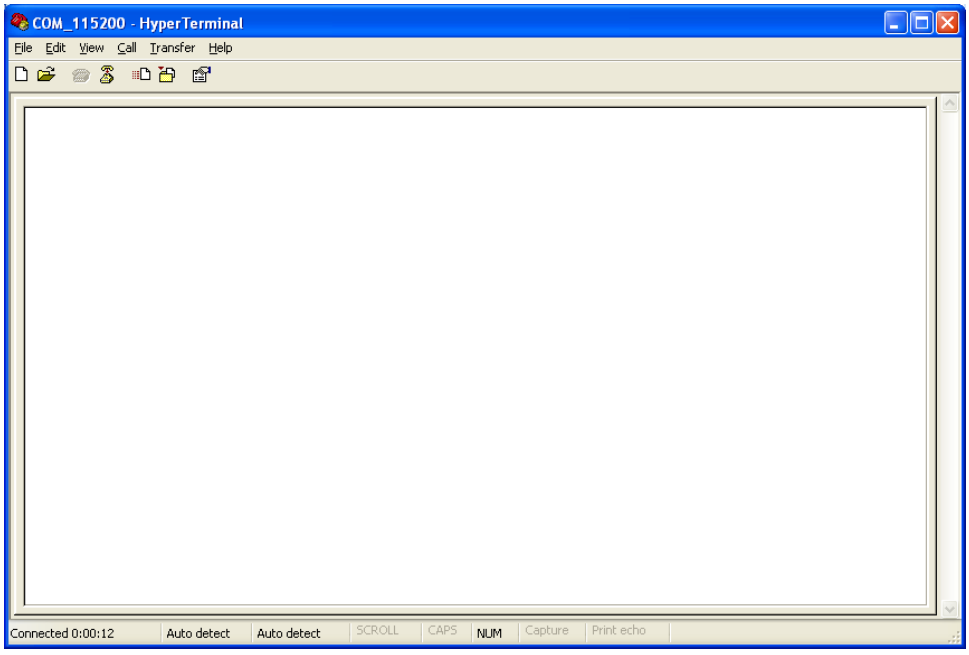


Figure 3-26 Hyper Terminal

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:

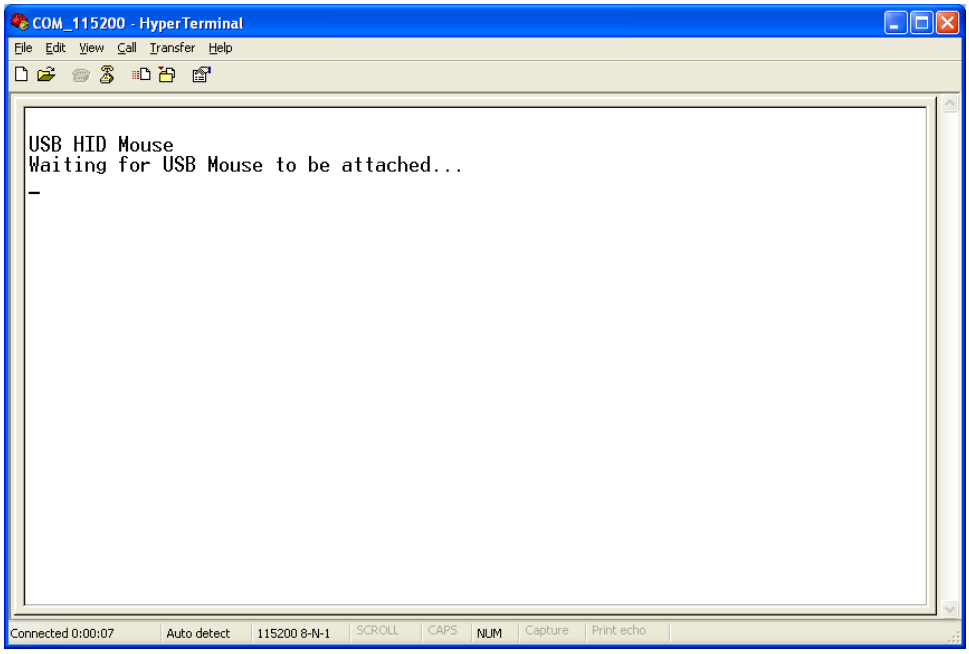


Figure 3-27 USB mouse host initialize

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

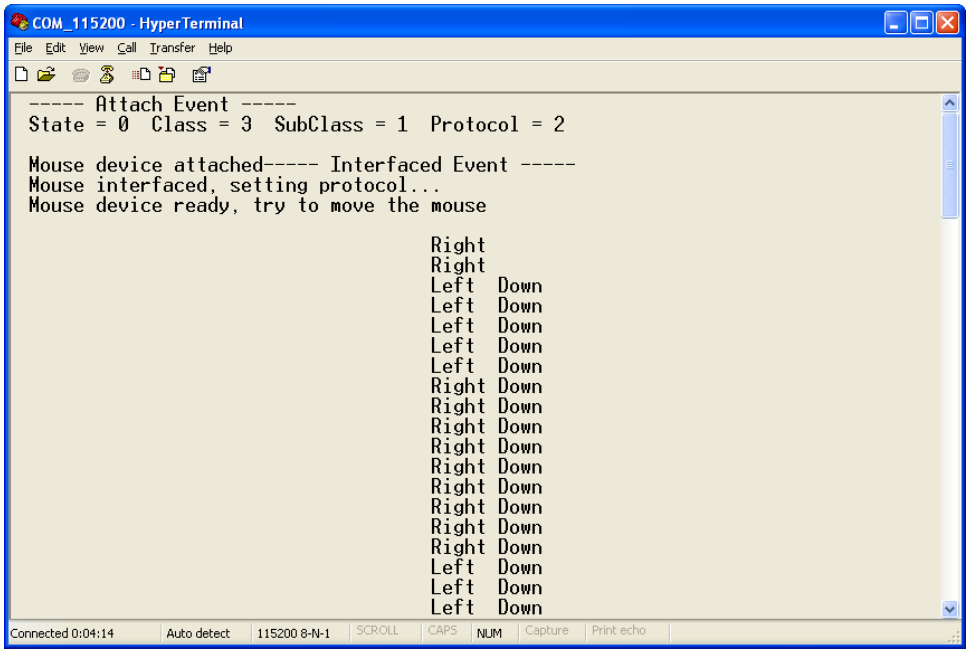


Figure 3-28 USB mouse host operation

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

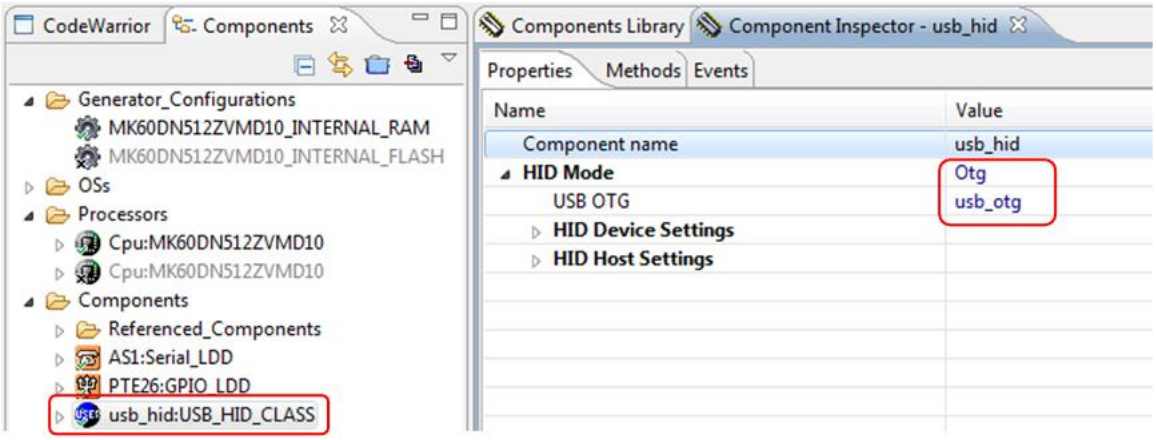


Figure 3-29 USB HID OTG mode

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.

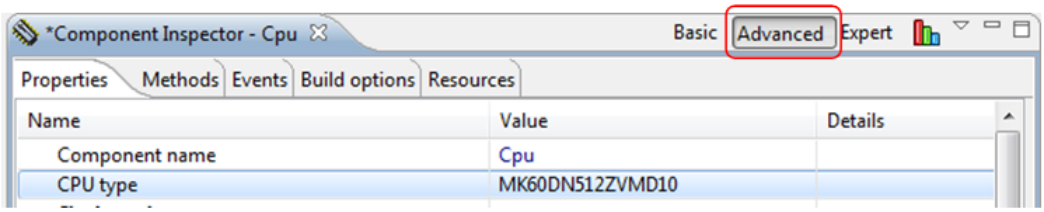


Figure 3-30 Open Advanced Tab

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

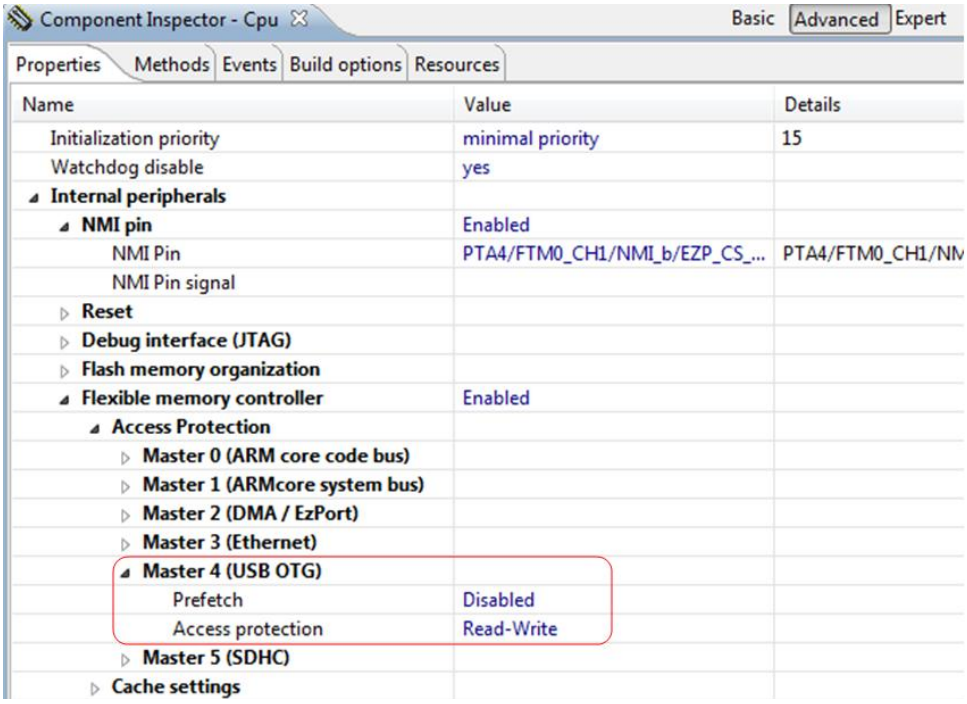


Figure 3-31 Flexible memory controller settings

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

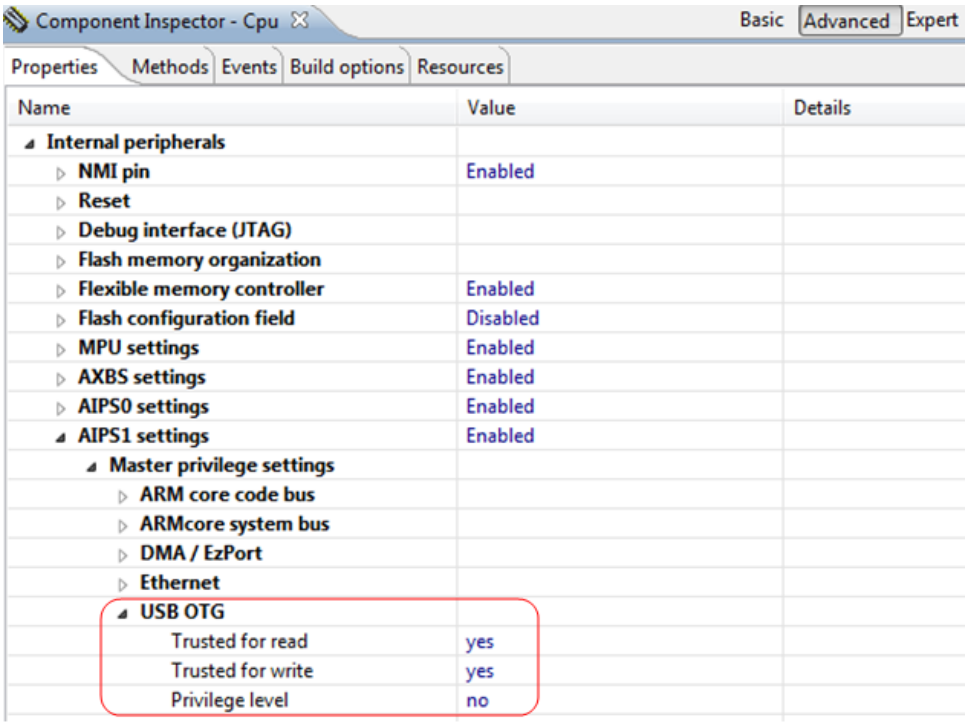


Figure 3-32 AIPS1 settings

- Enable LVD interrupt in the Power management controller field.

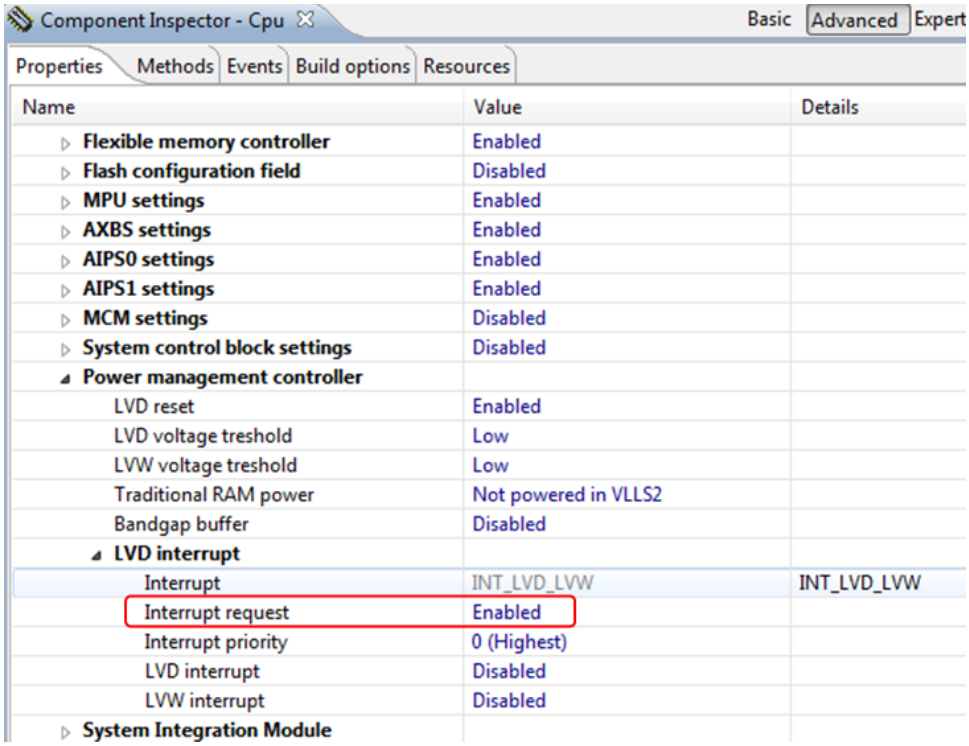


Figure 3-33 Power management controller settings

Step3. Configure the USB_HID_CLASS component:

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

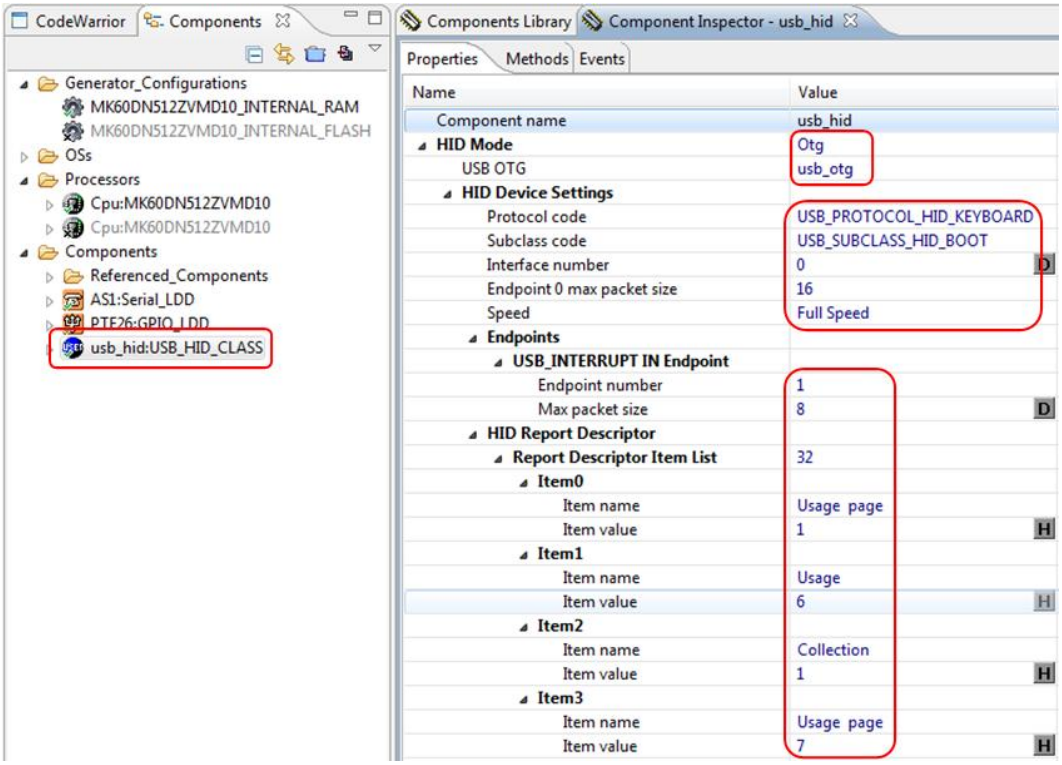


Figure 3-34 HID Device settings

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

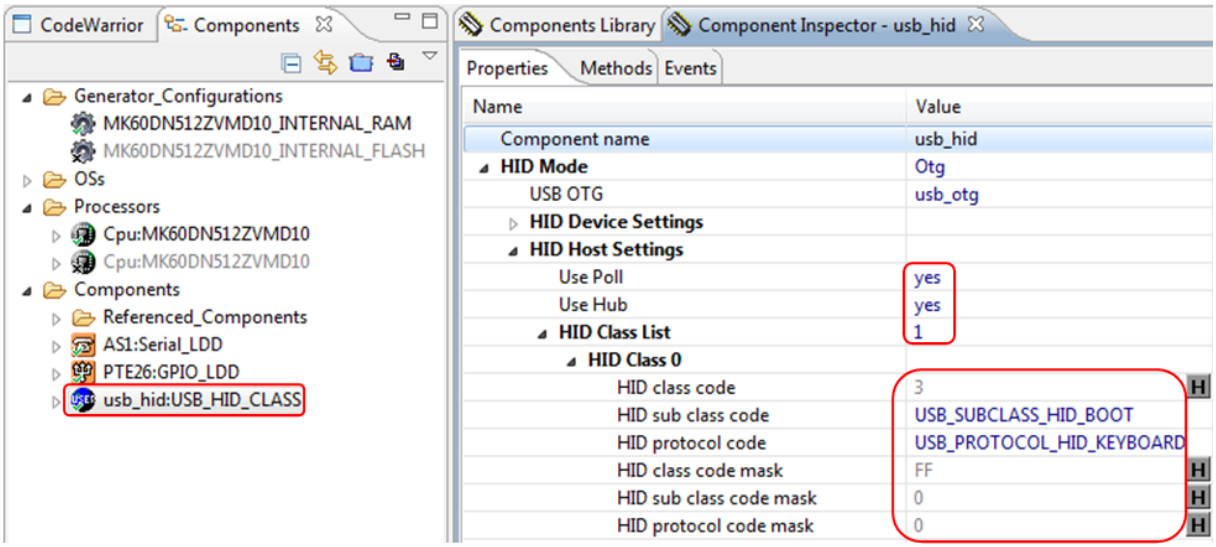


Figure 3-35 HID Host settings

Step4. Configure the USB_OTG_STACK component:

- OTG settings:
  - Chose the GIPO_LDD component to configure interrupt pin for MAX3353 as the following figure:

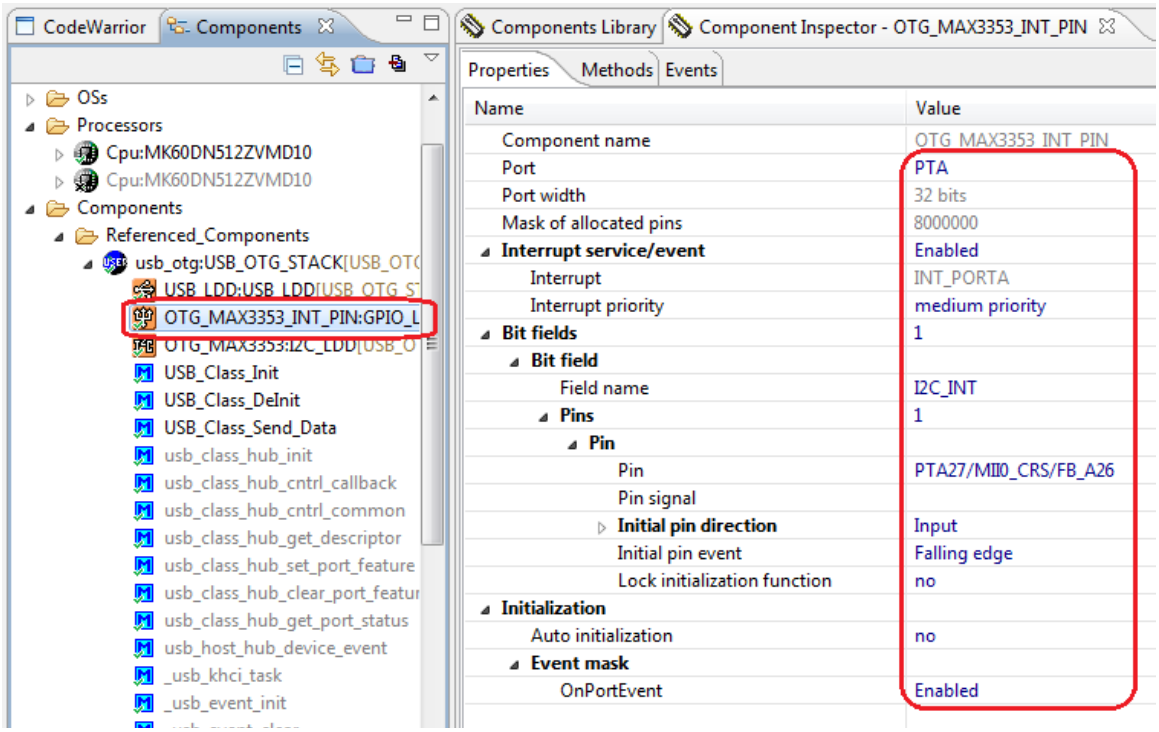


Figure 3-36 MAX3353 Interrupt pin settings

- Chose the I2C_LDD component to configure the features for MAX3353

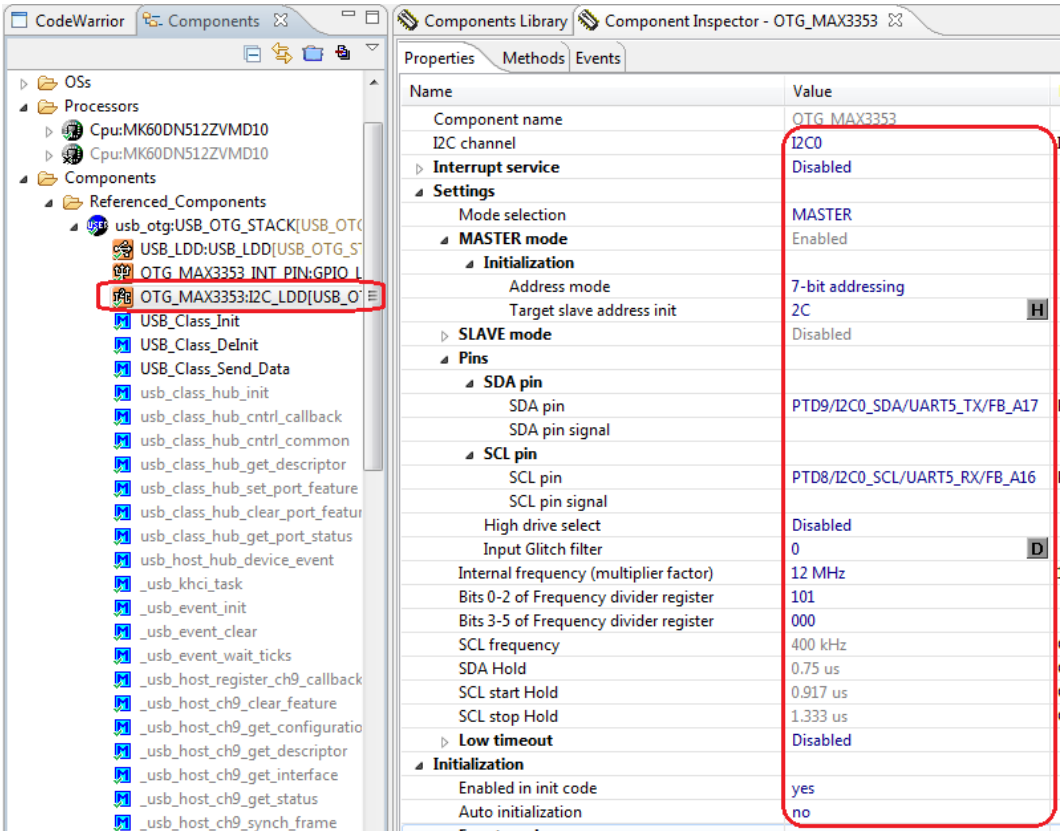


Figure 3-37 MAX3353 settings

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

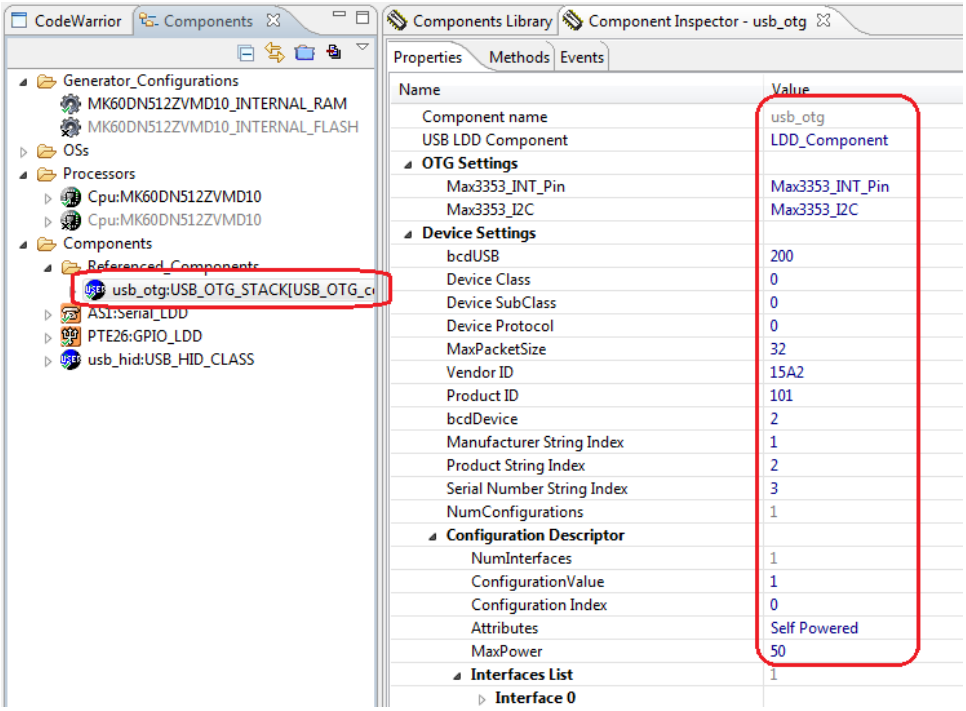


Figure 3-38 General descriptor settings

- Configure HID descriptor as shown below:

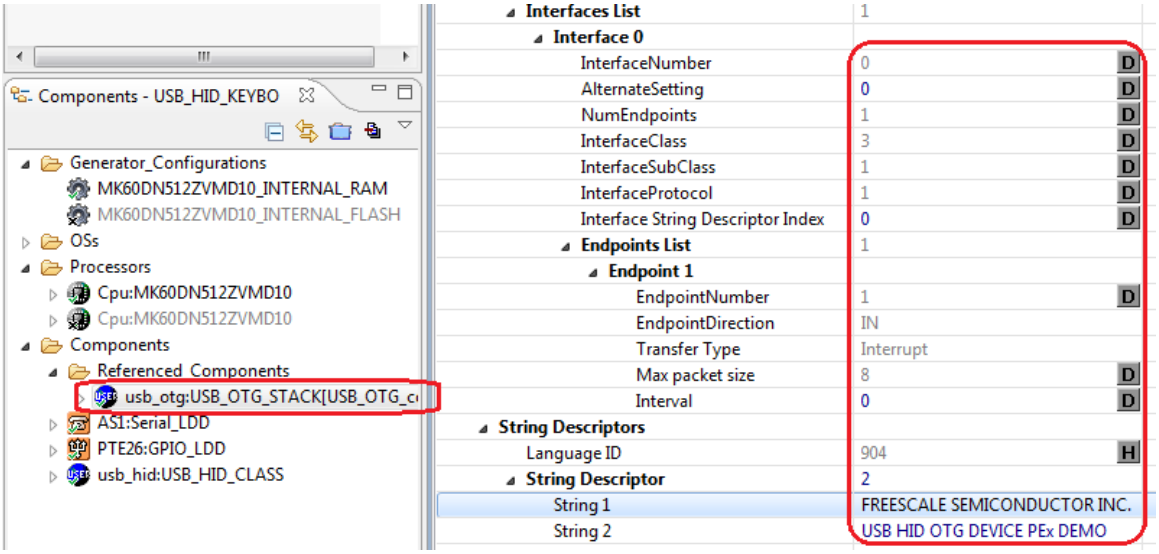


Figure 3-39 HID descriptor settings

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

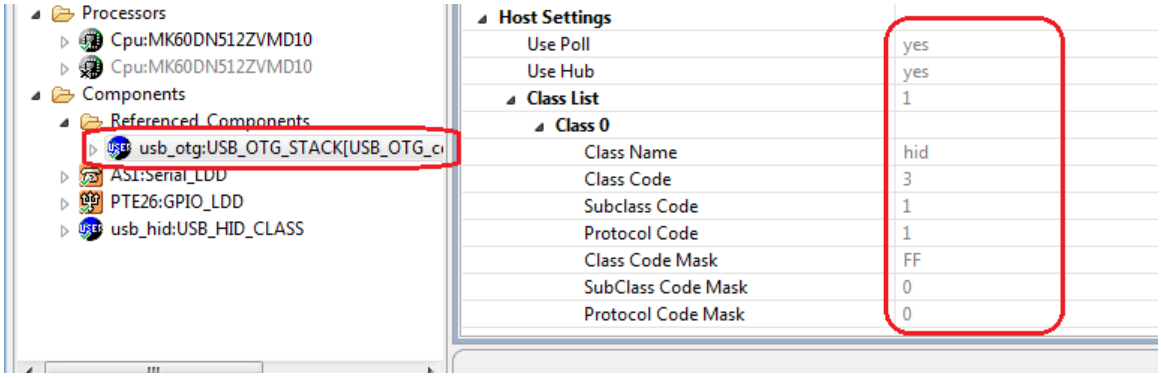


Figure 3-40 Host settings

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

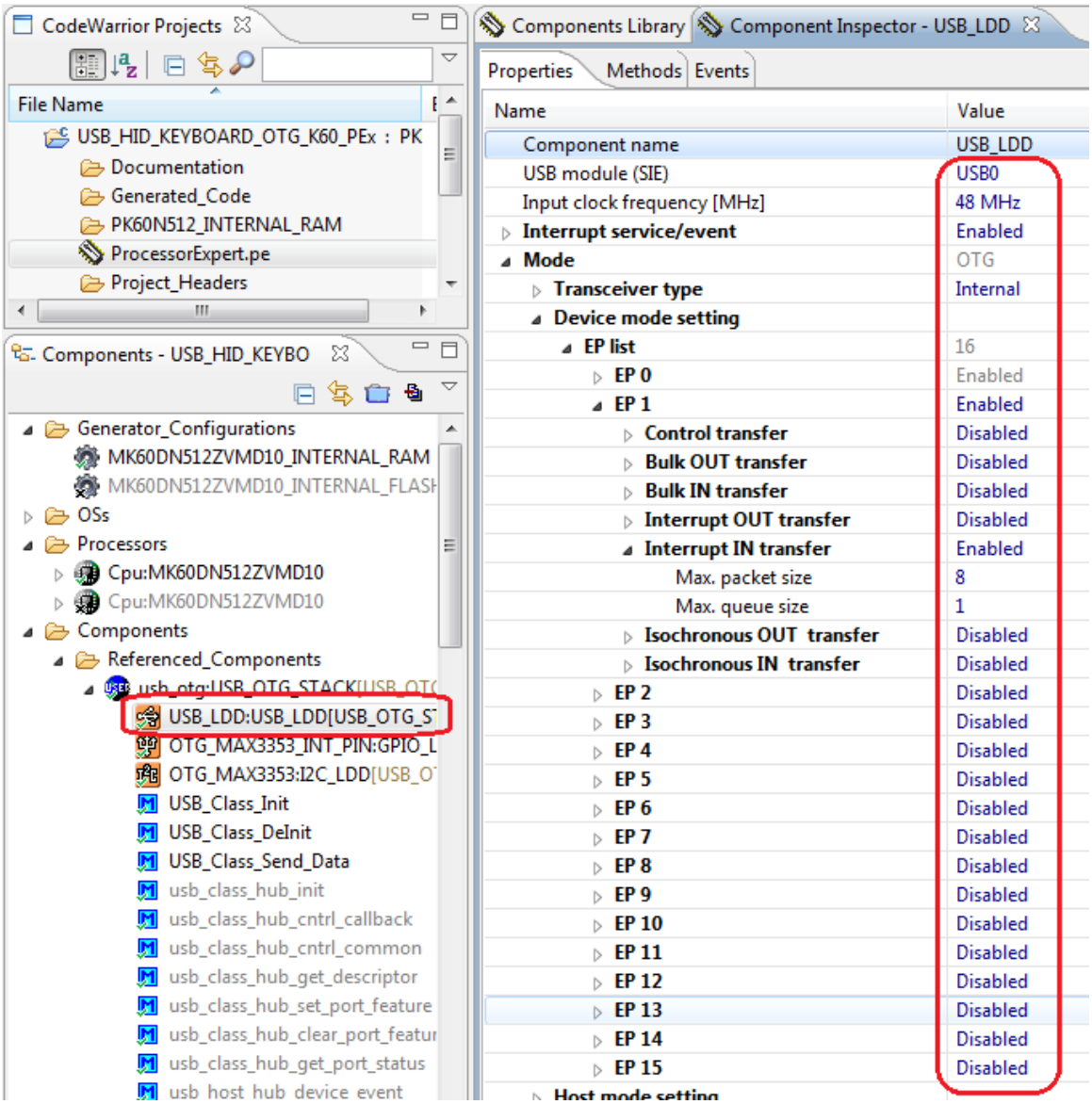


Figure 3-41 USB_LDD component OTG mode settings

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.

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

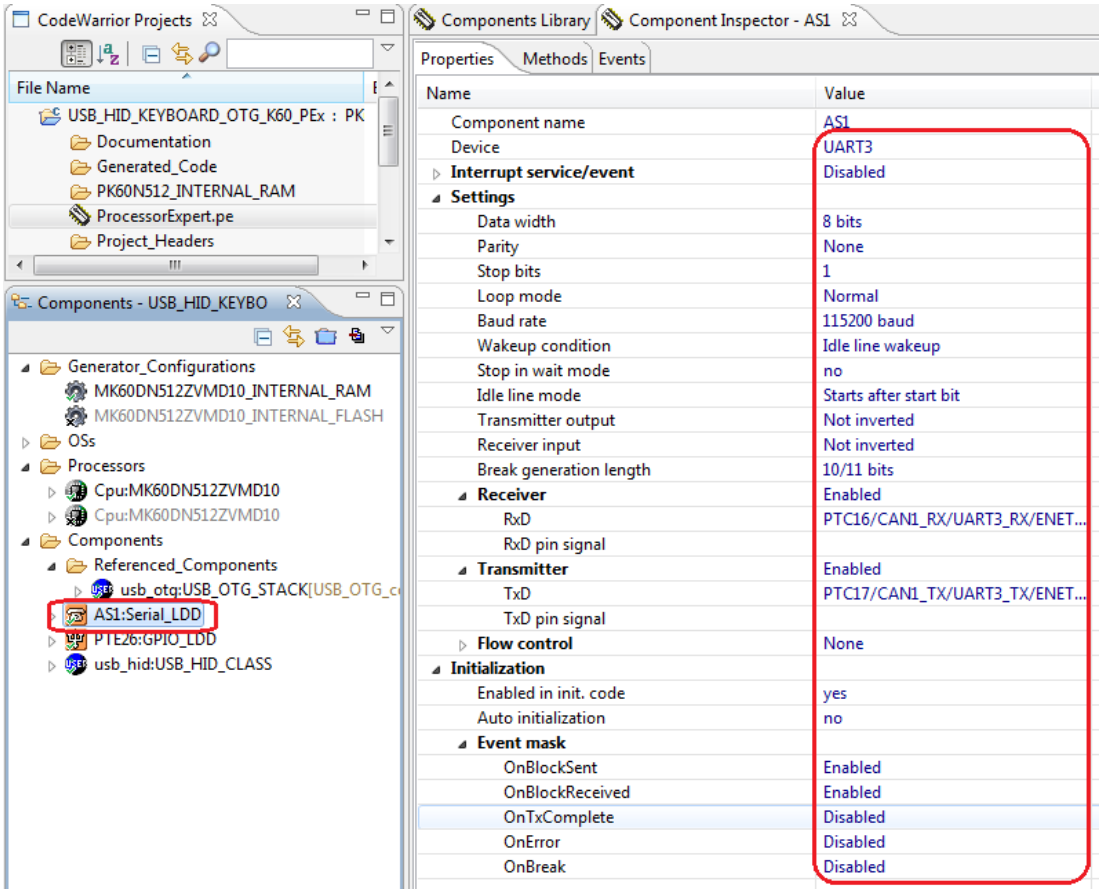


Figure 3-42 Serial_LDD component settings

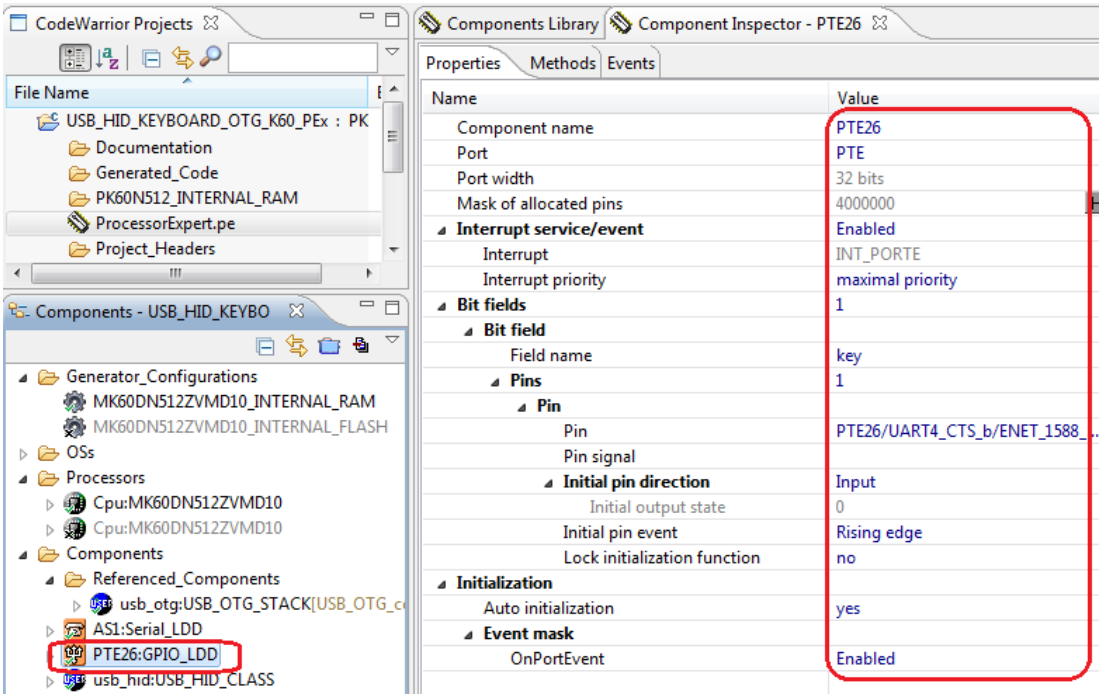


Figure 3-43 GPIO_LDD component settings

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.

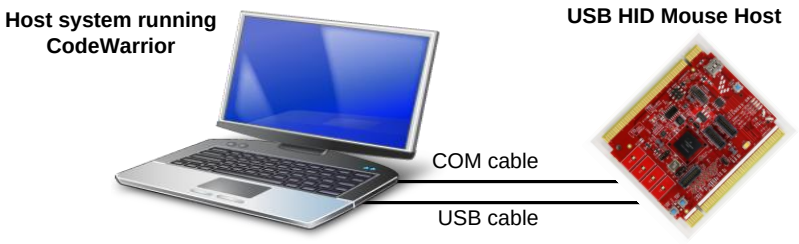


Figure 3-44 HID keyboard OTG demo setup

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:

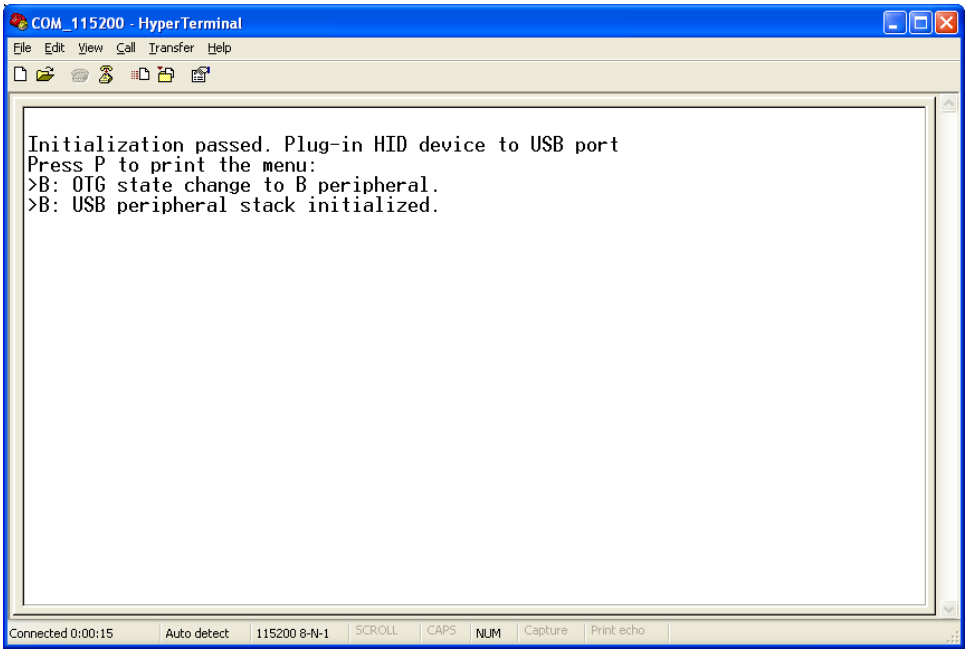
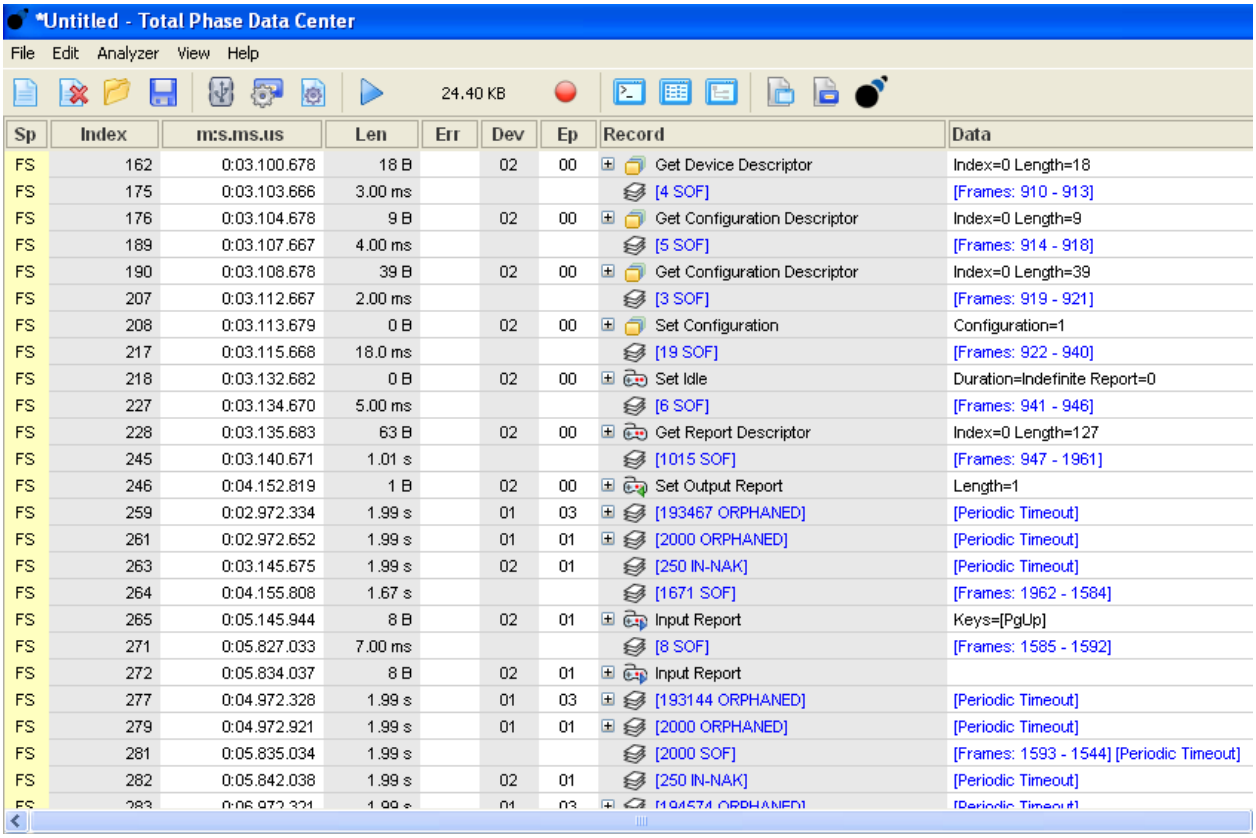


Figure 3-45 USB HID keyboard OTG initializing

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:



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				[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]

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.

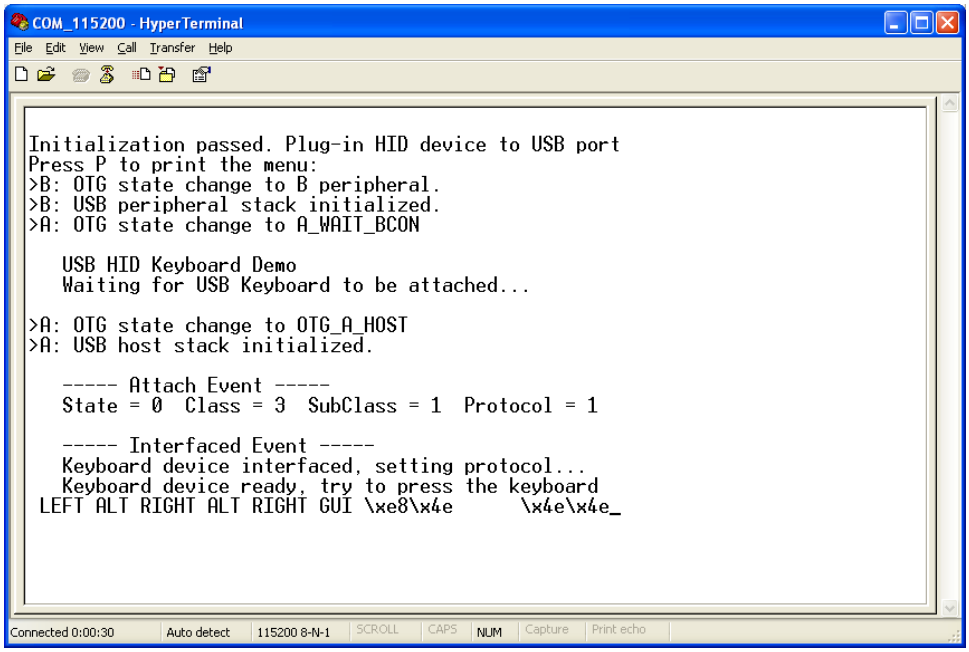


Figure 3-47 USB HID keyboard OTG operation – Device-A Host

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.

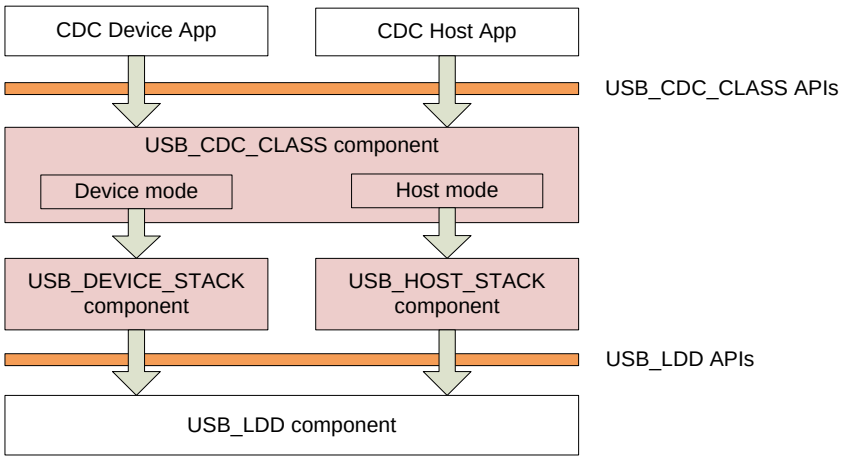


Figure 4-1 Architecture overview

4.2 Folder structure

4.2.1 USB CDC component package

The delivered package includes items as in the following figure:

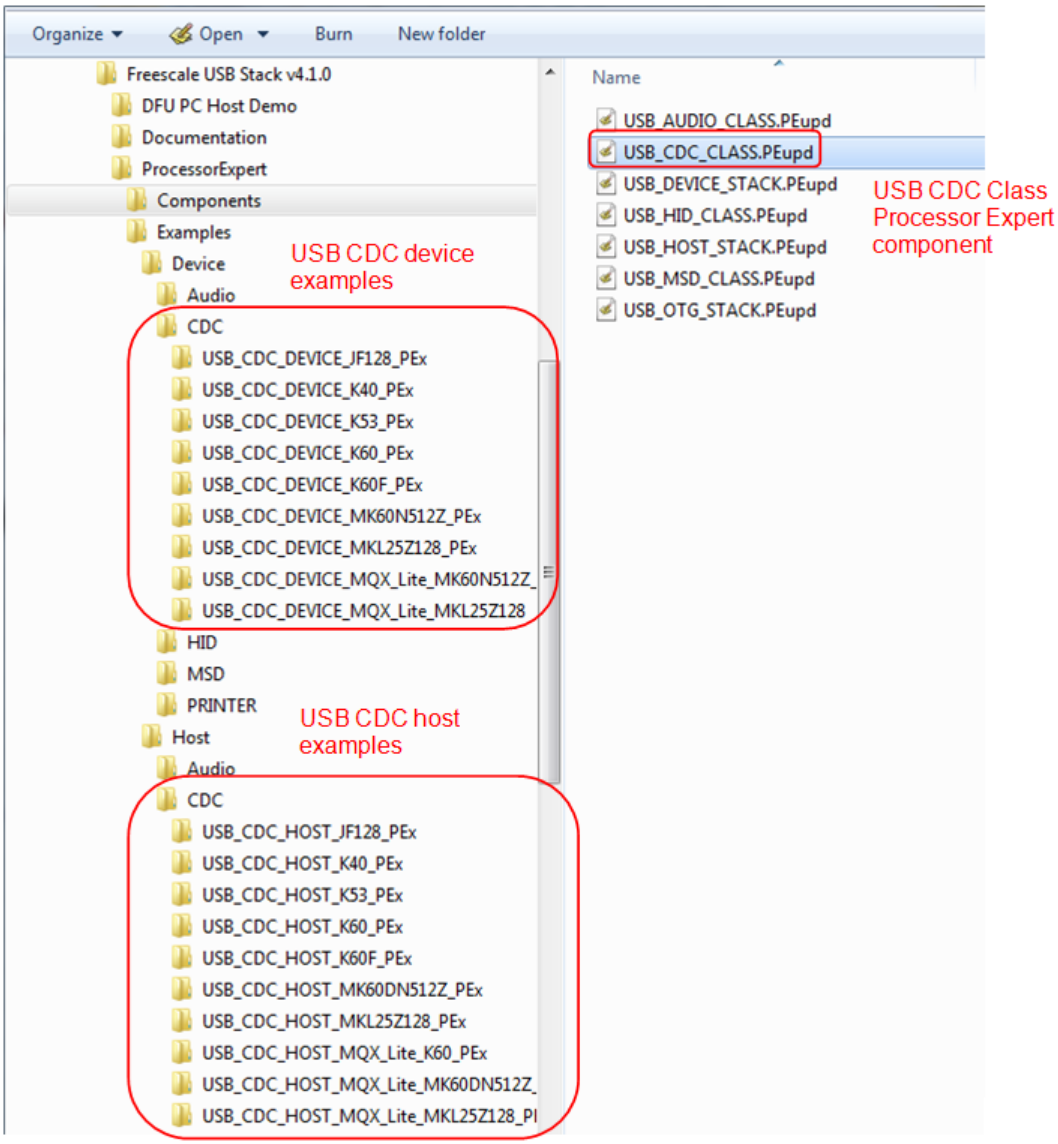


Figure 4-2 Deliver package directory structure

- 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:

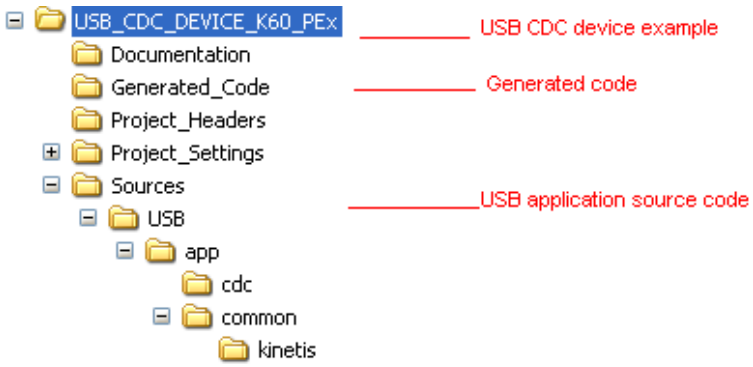


Figure 4-3 CDC device directory structure

4.2.3 CDC host example structure

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

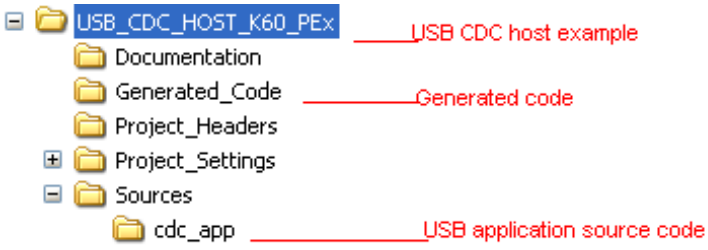


Figure 4-4 CDC host directory structure

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

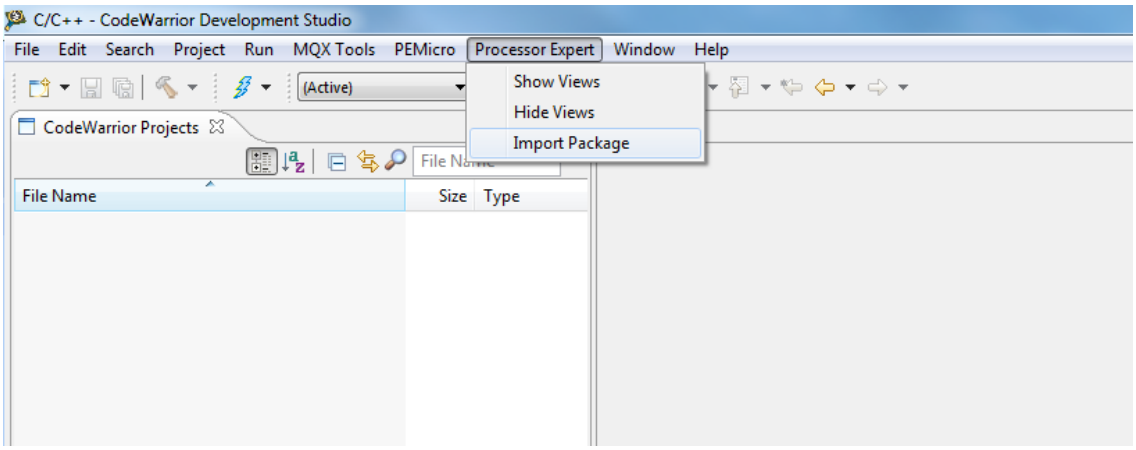


Figure 4-5 Processor Expert – Import Package

Step2. A browser window appears. Navigate to the folder that contains the USB_CDC_CLASS component, chose it and click open.

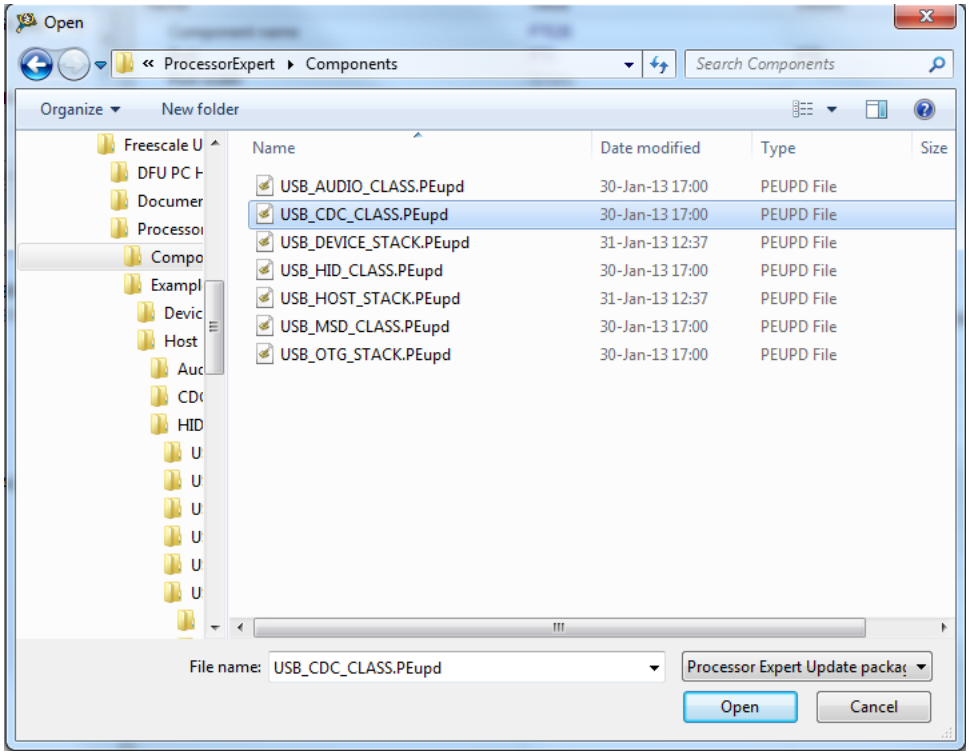


Figure 4-6 Delivered package navigation

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

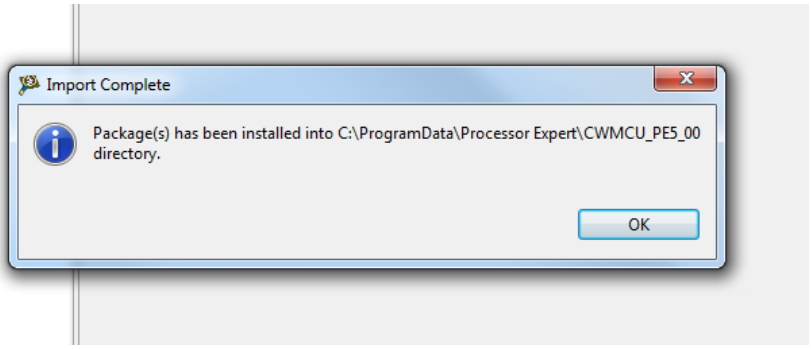


Figure 4-7 Import USB components

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.

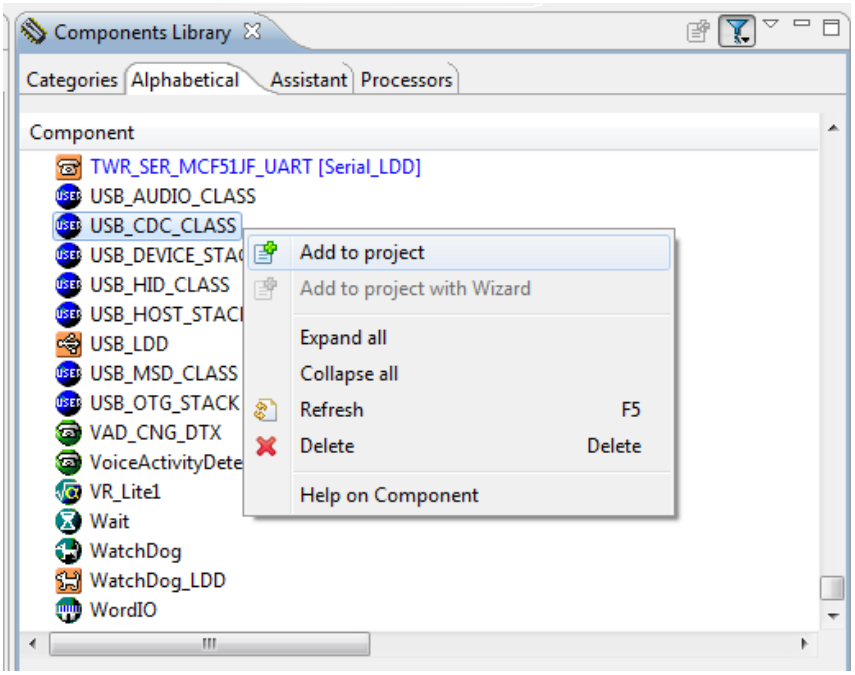


Figure 4-8 Component show views

Step2. Configure CPU clock to run USB module

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

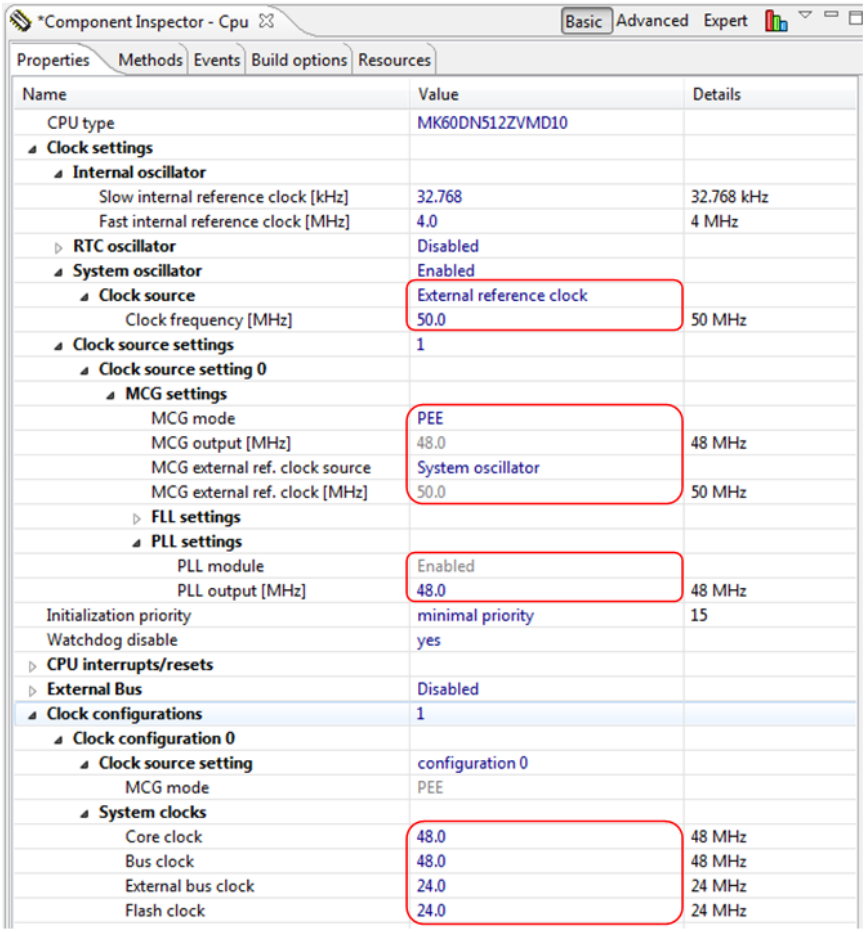


Figure 4-9 Clock source settings

- Chose advanced tab

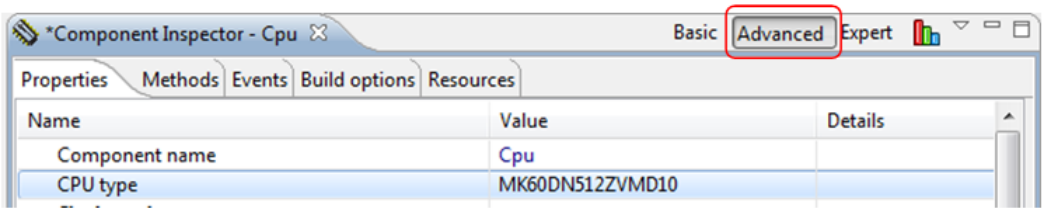


Figure 4-10 Open Advanced Tab

- PLL/FLL clock selection:
  - PLL clock

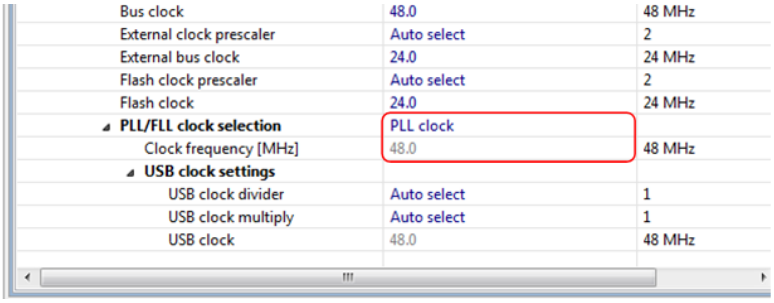


Figure 4-11 Clock configuration

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.

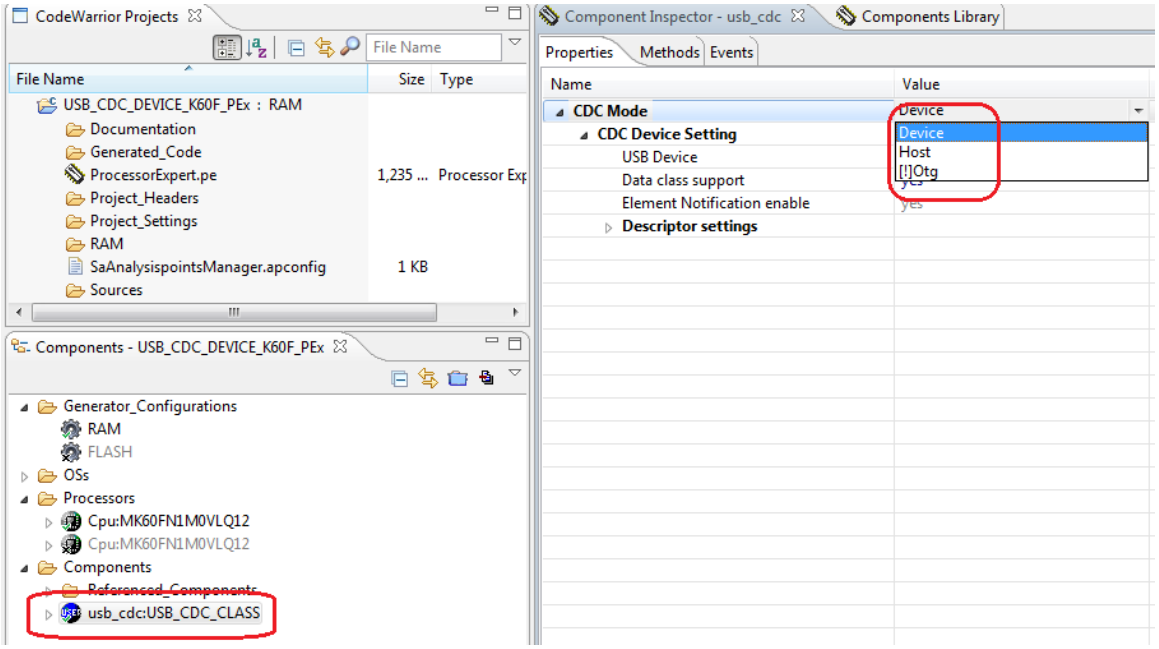


Figure 4-12 CDC device mode settings

- Fill input parameters of the USB_CDC_CLASS component as in the following figures:

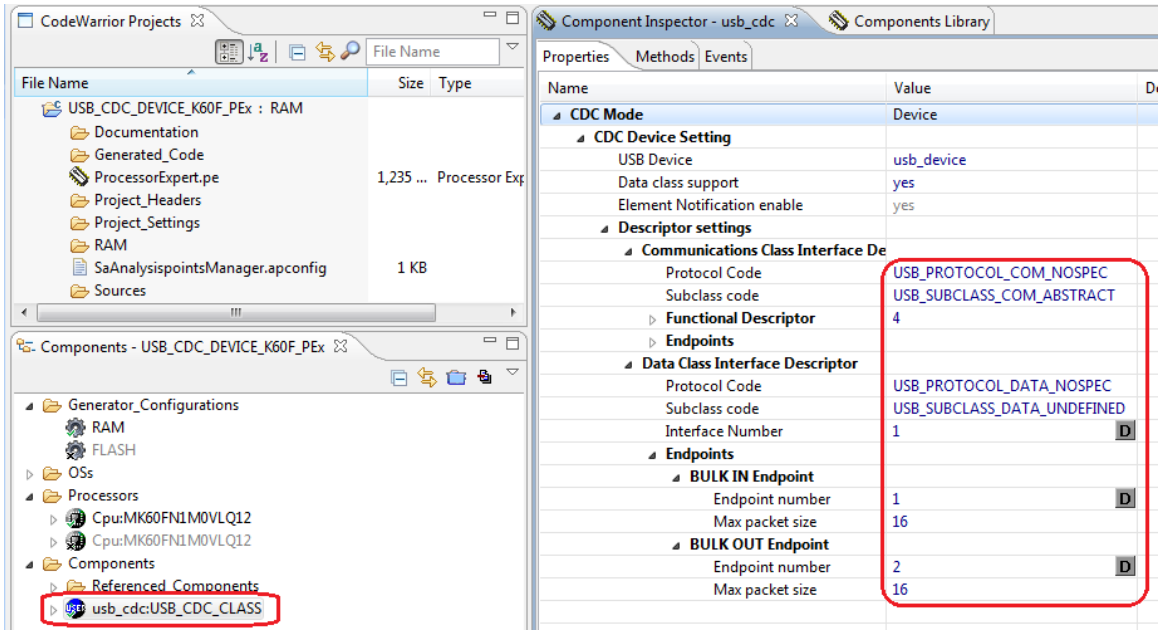


Figure 4-13 CDC Device settings

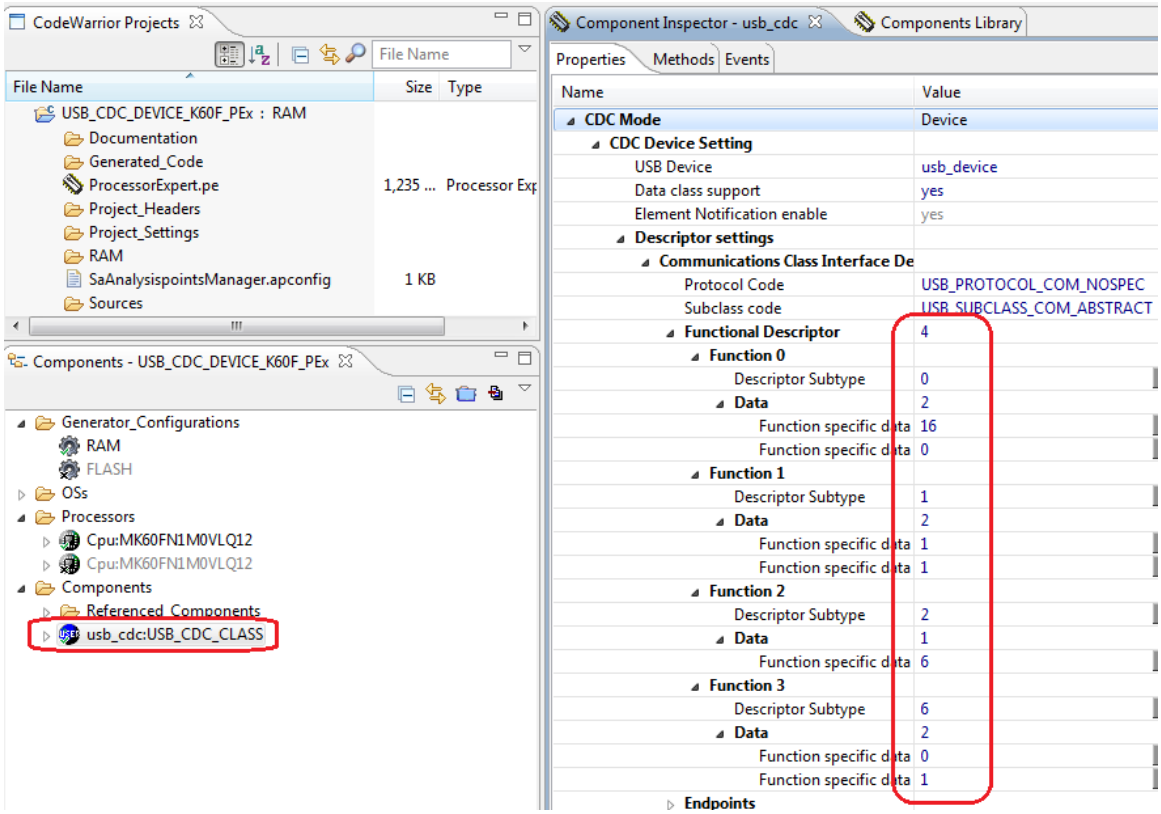


Figure 4-14 Function Descriptor settings

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:
    - Device Descriptor

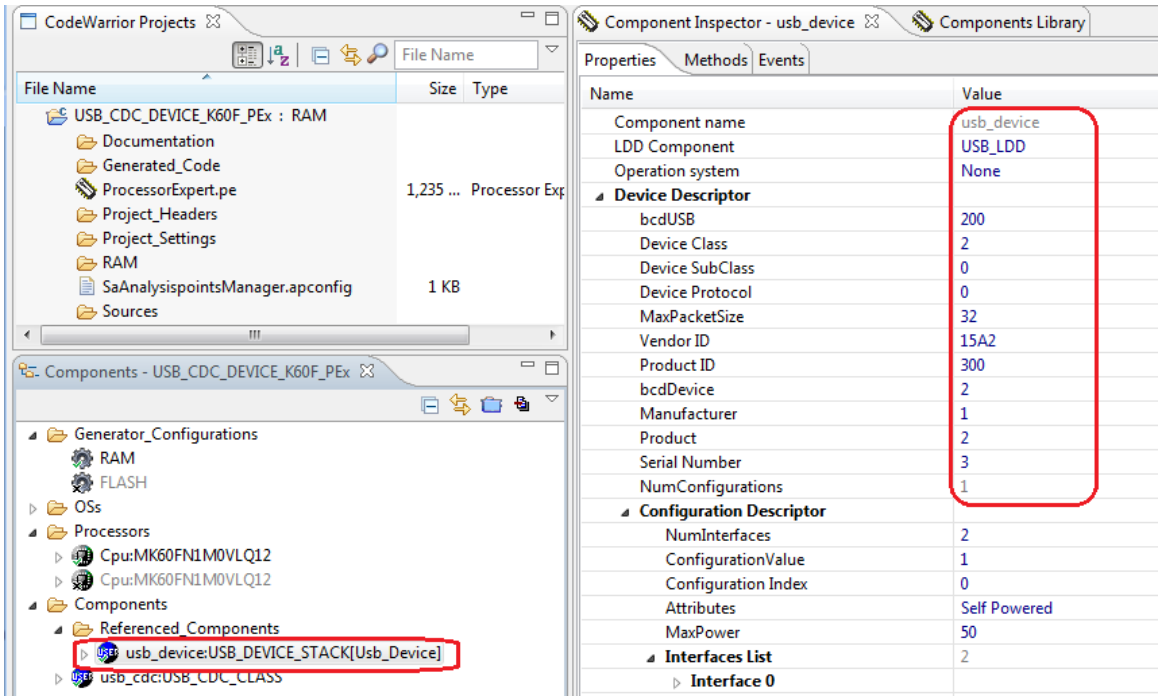


Figure 4-15 Device Descriptor settings

- Configuration Descriptor: This group inherited the USB_CDC_CLASS component.

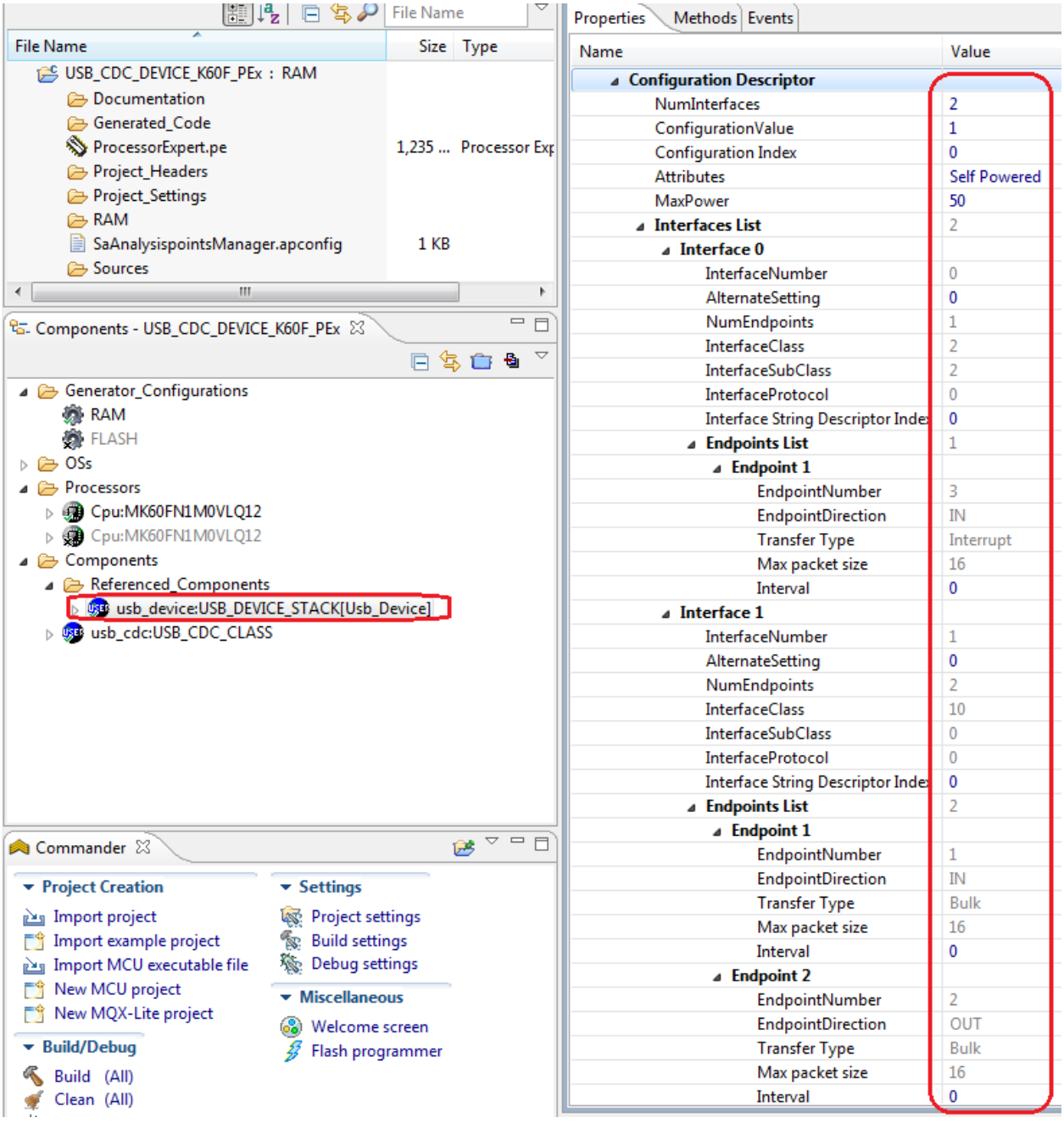


Figure 4-16 Configuration Descriptor settings

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

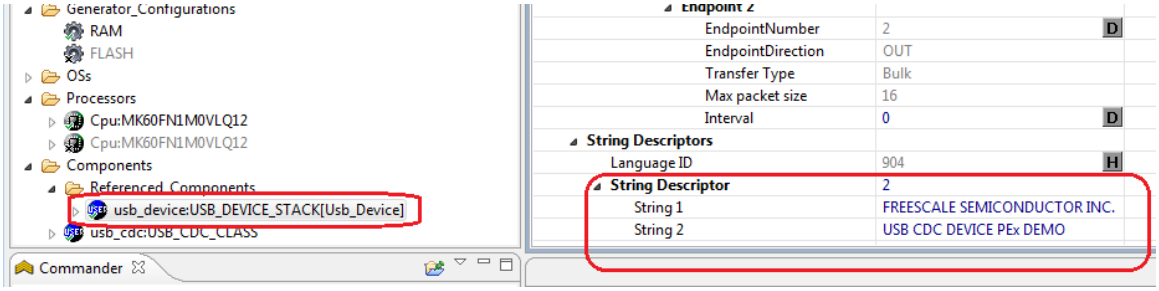


Figure 4-17 String Descriptor settings

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.

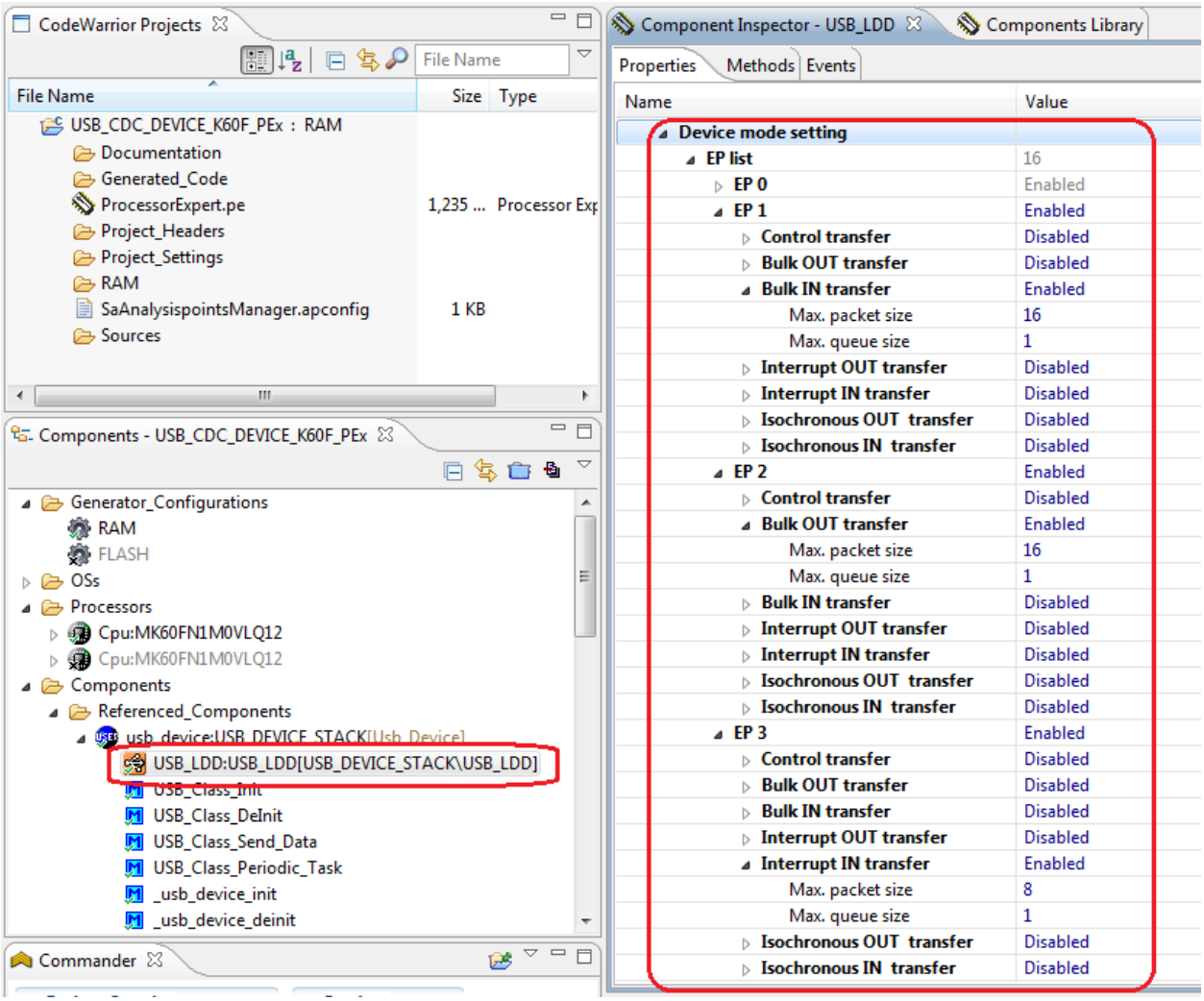


Figure 4-18 USB_LDD component settings

Step6. Generate Processor Expert code:

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

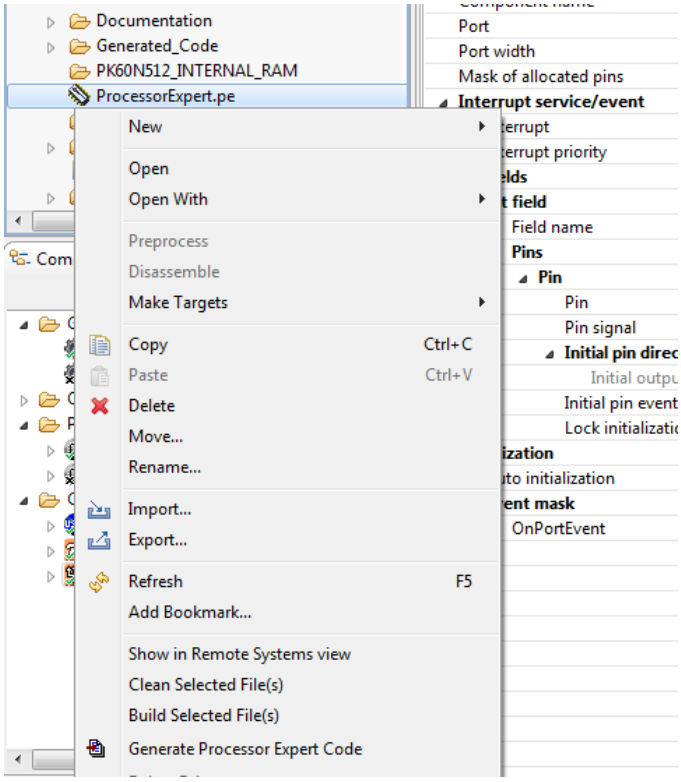


Figure 4-19 Generate Processor Expert code

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

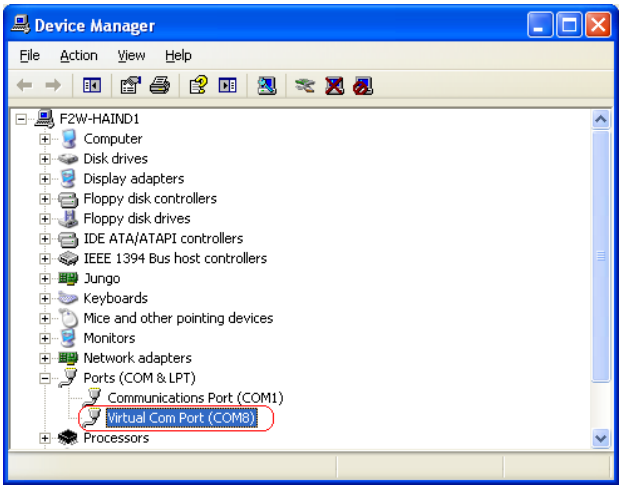


Figure 4-20 Device entry

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:

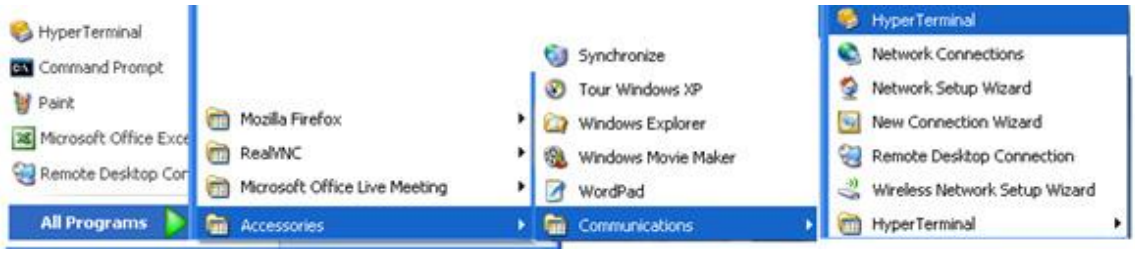


Figure 4-21 Launch HyperTerminal application

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

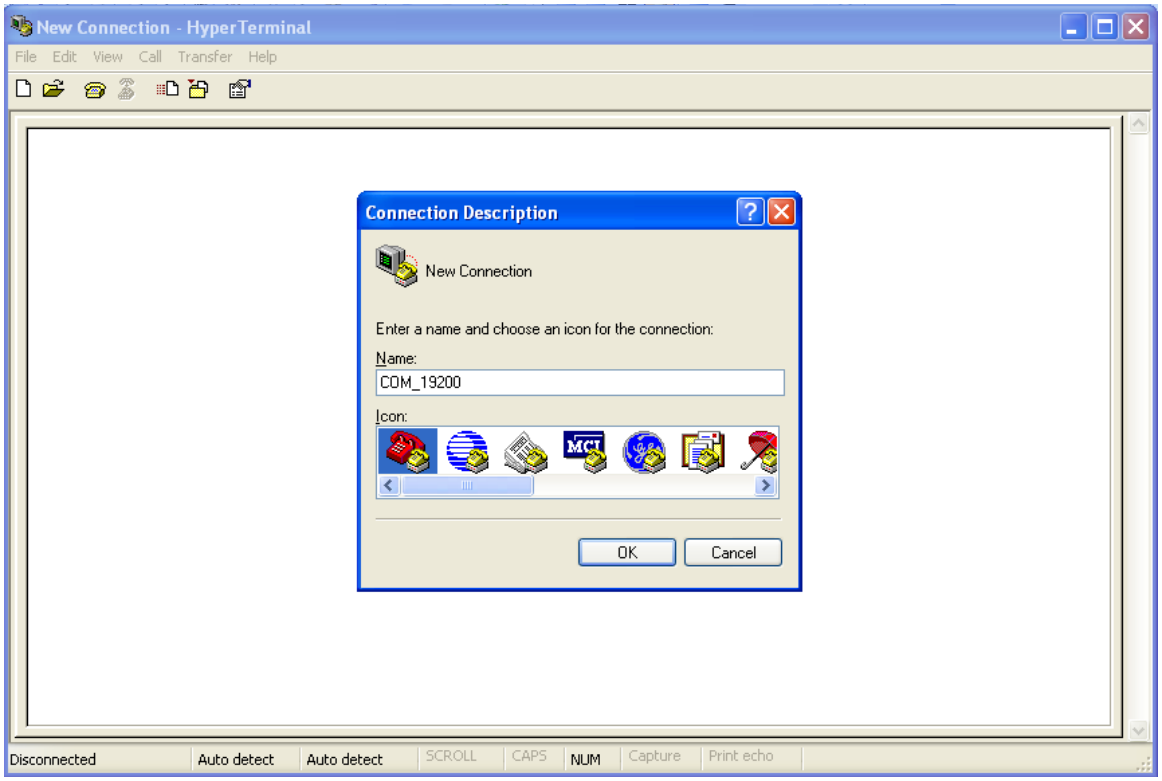


Figure 4-22 HyperTerminal startup

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

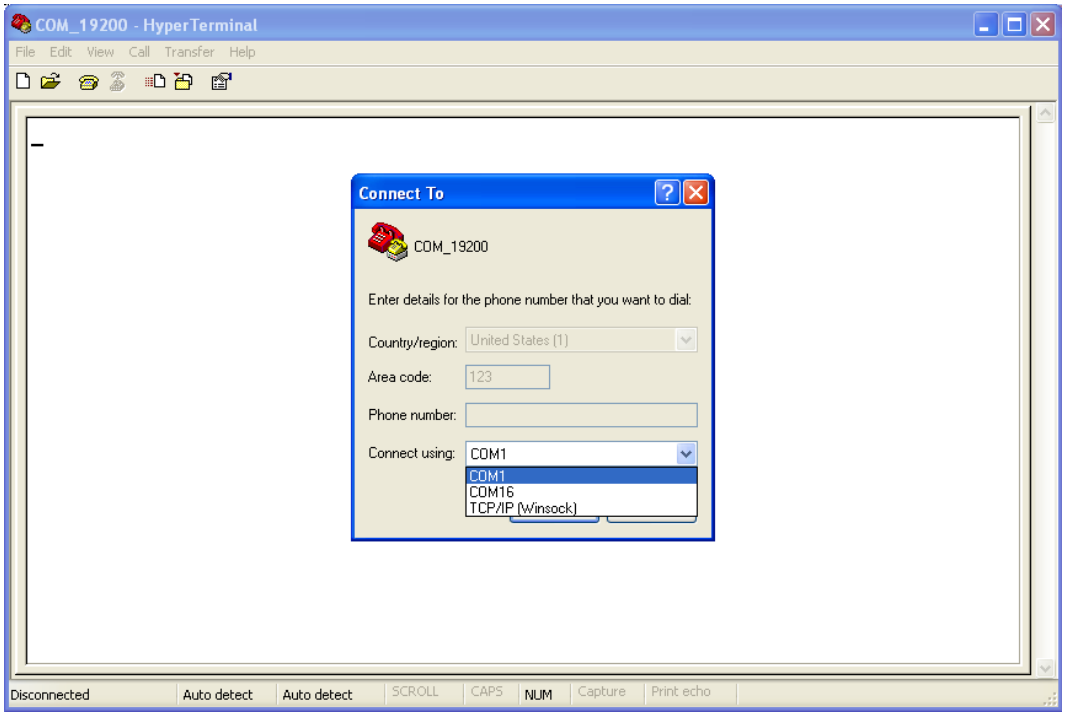


Figure 4-23 Connect using COM port

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

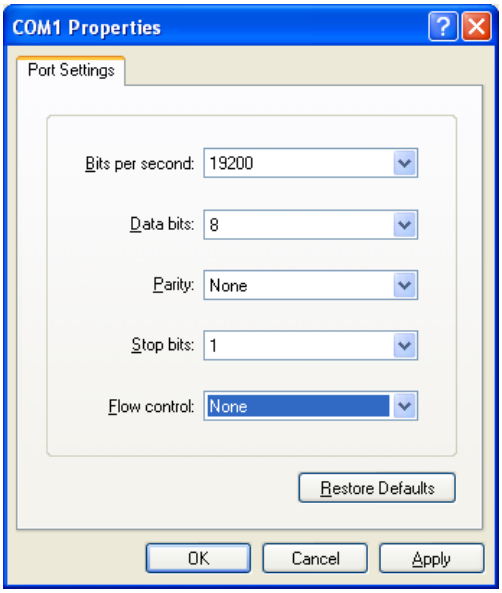


Figure 4-24 COM properties

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

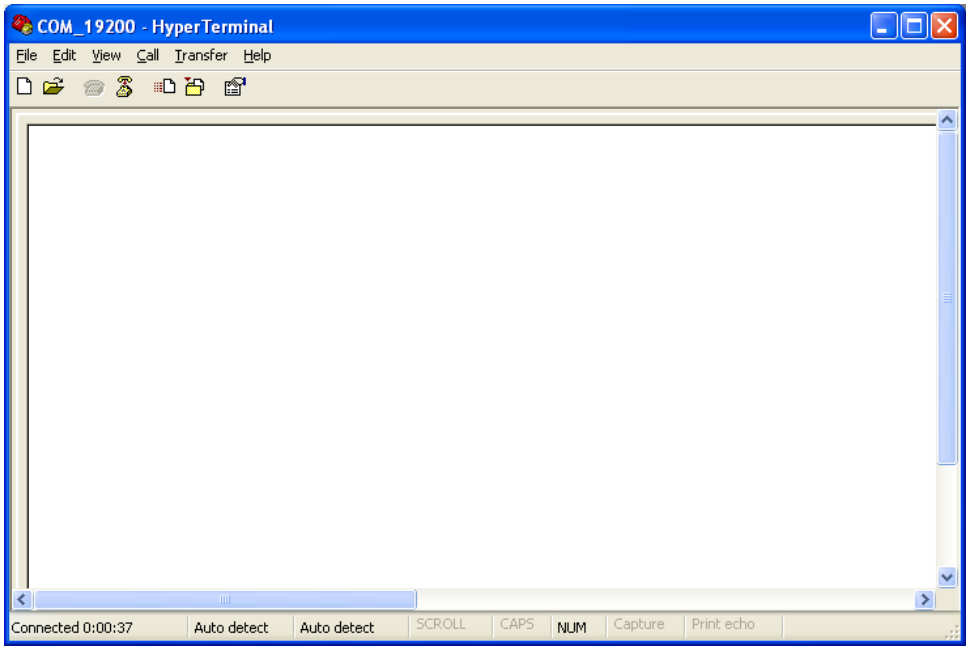


Figure 4-25 Hyper Terminal

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

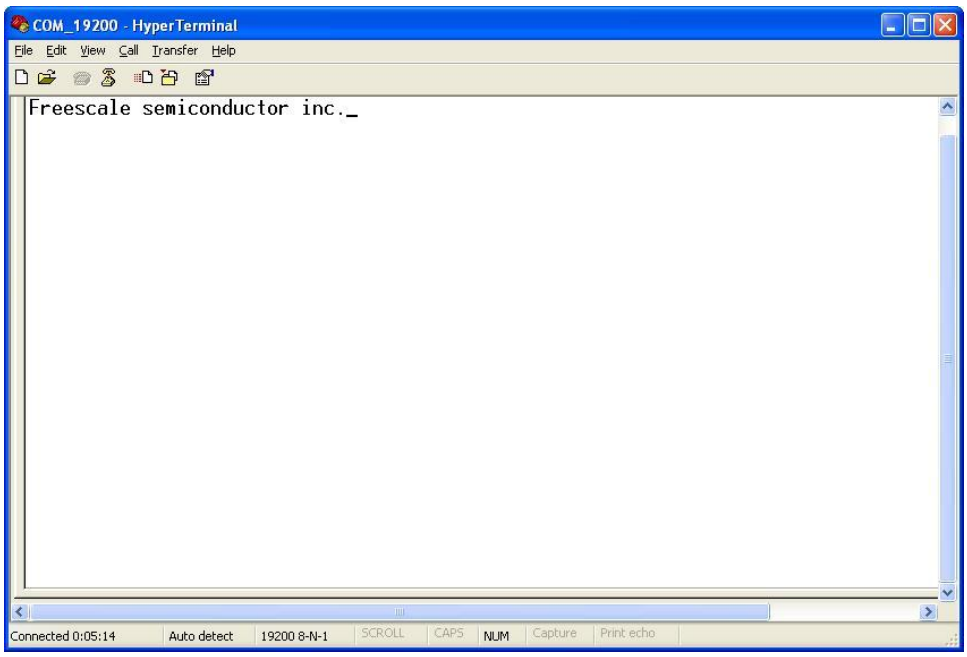


Figure 4-26 CDC device operating

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.

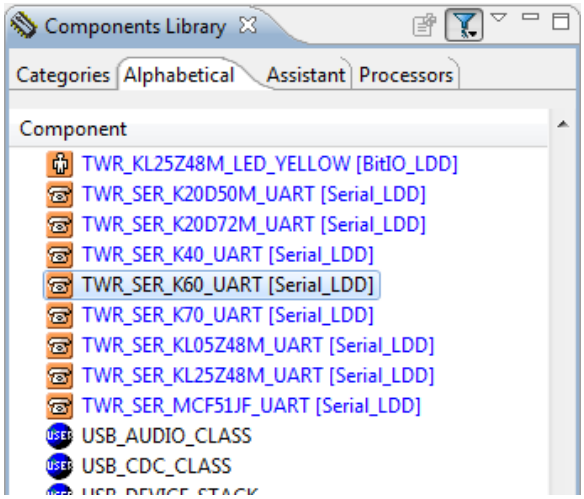


Figure 4-27 Components Library

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.

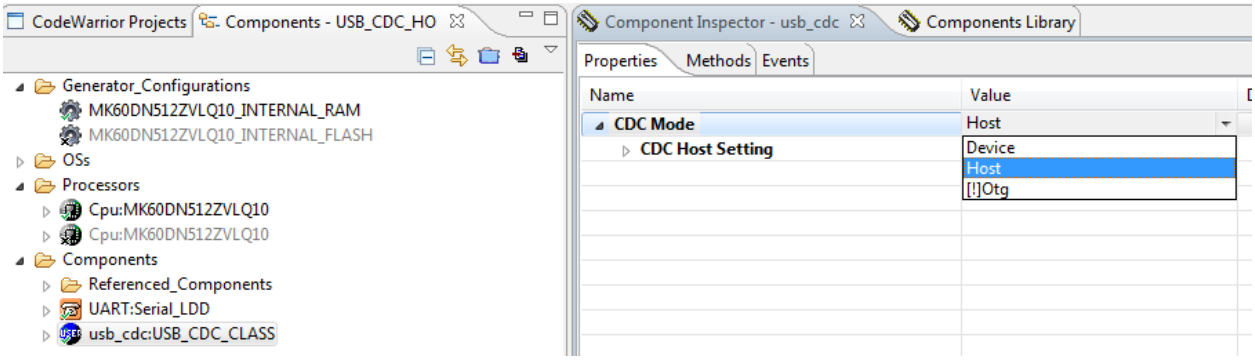


Figure 4-28 CDC host mode setting

- Fill input parameters into CDC class list settings:

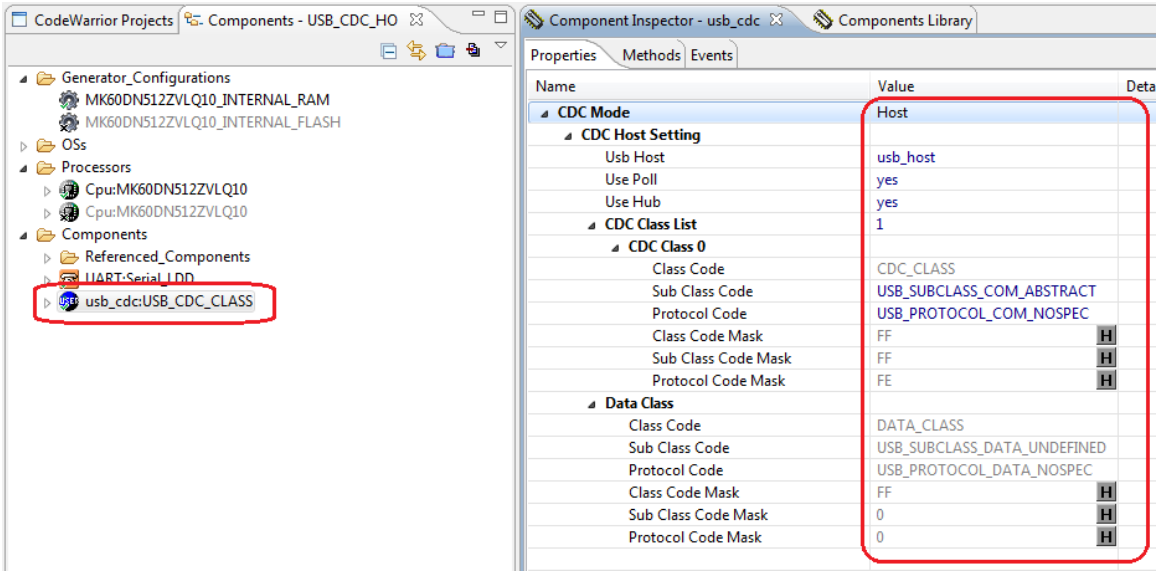


Figure 4-29 USB_CDC_CLASS component settings

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.

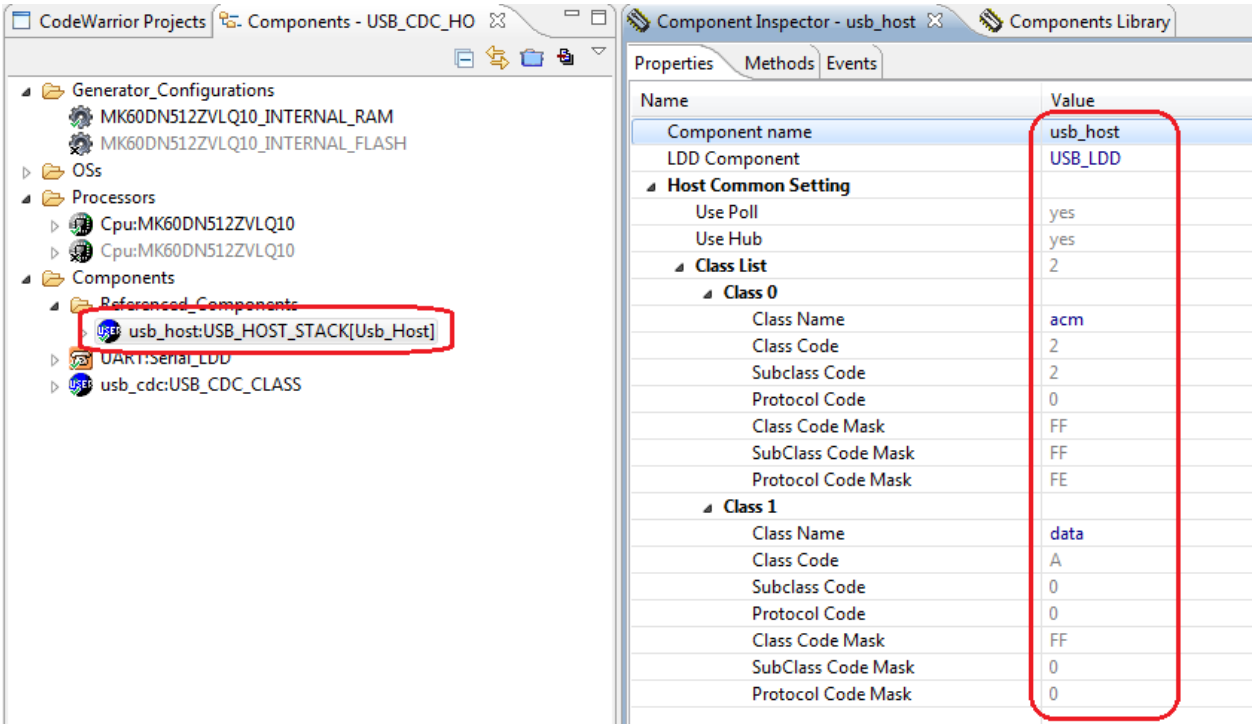


Figure 4-30 USB_HOST_STACK component settings

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.

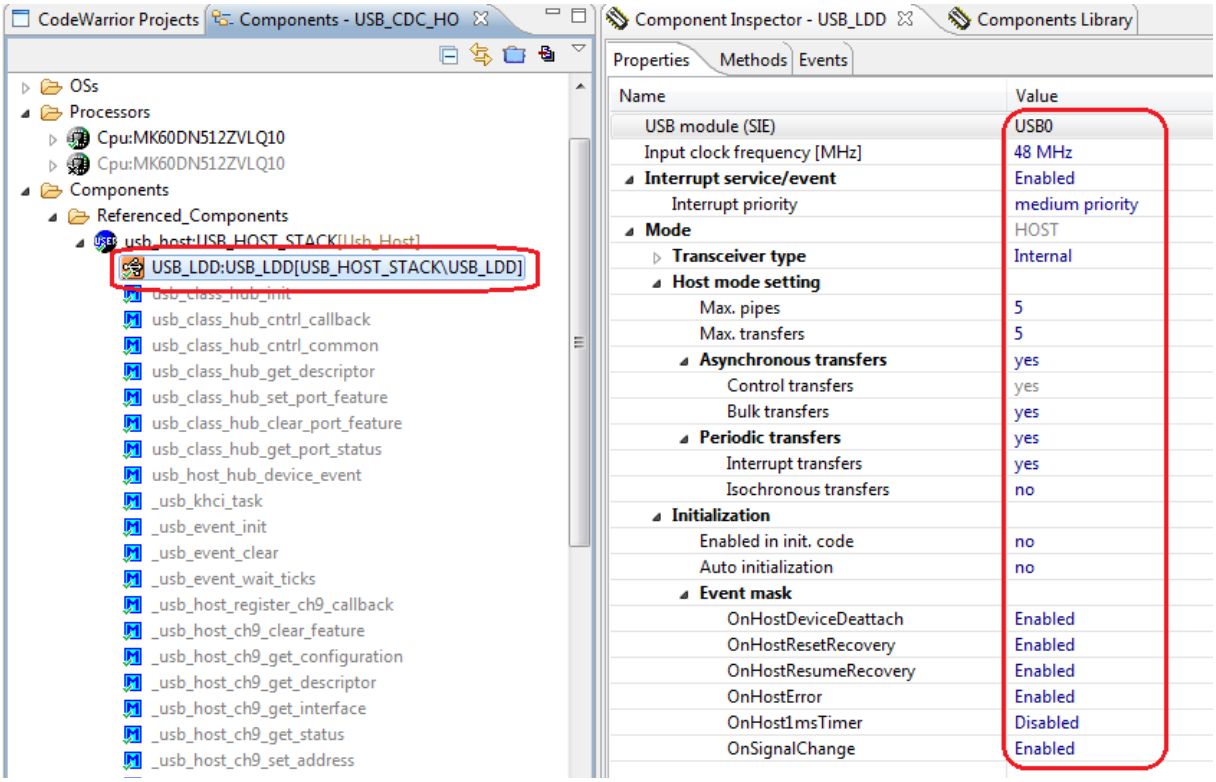


Figure 4-31 USB_LDD component settings

Step7. Setup the TWR_SER_K60_UART component as following figure:

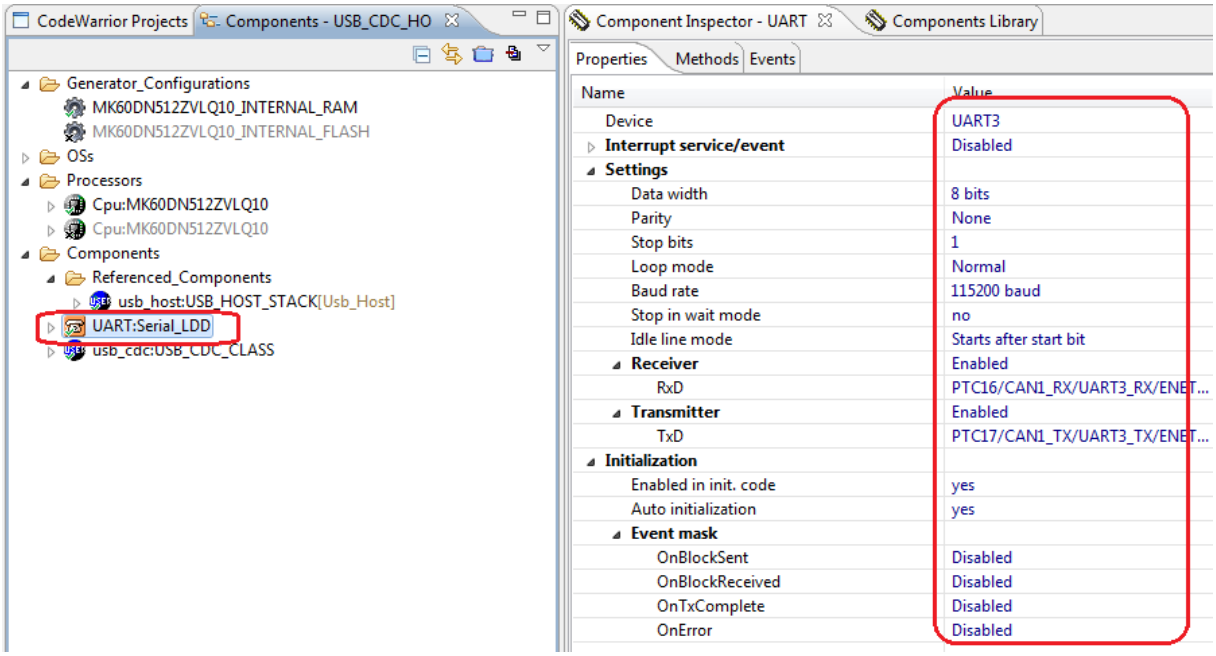


Figure 4-32 UART settings

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.

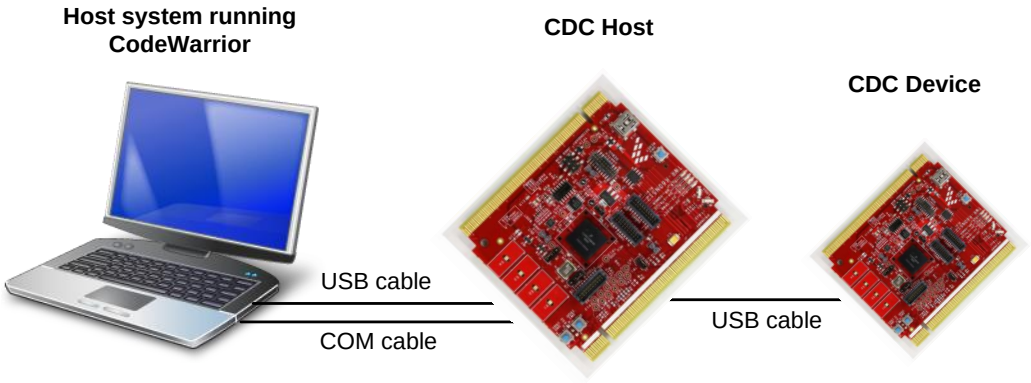


Figure 4-33 Hardware setup

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

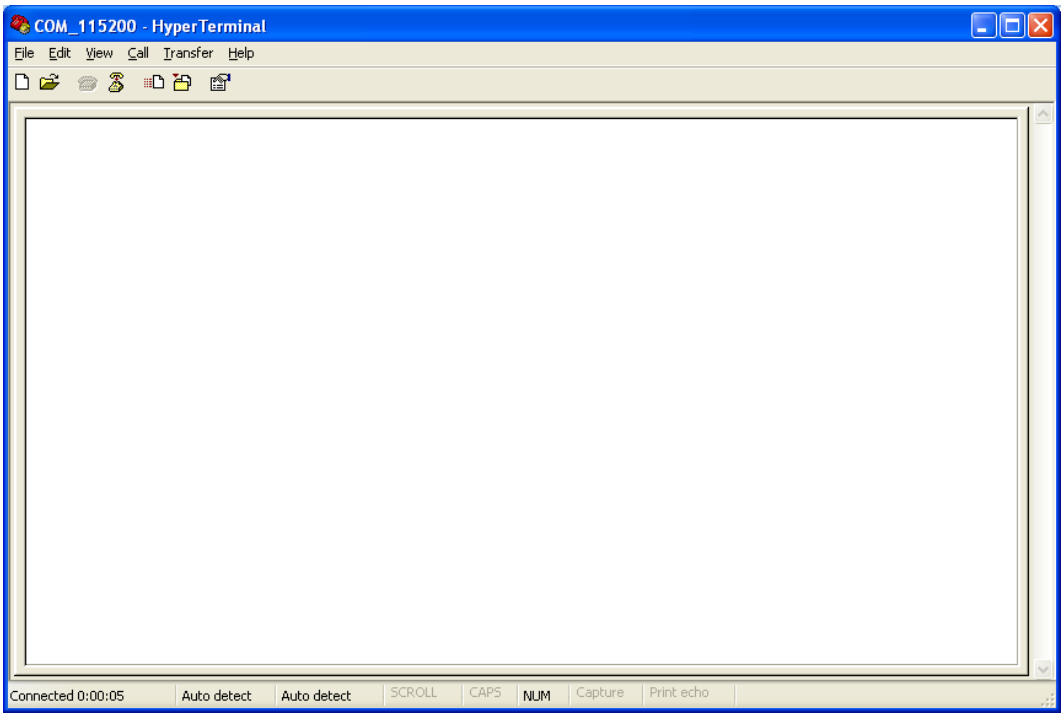


Figure 4-34 Hyper Terminal

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

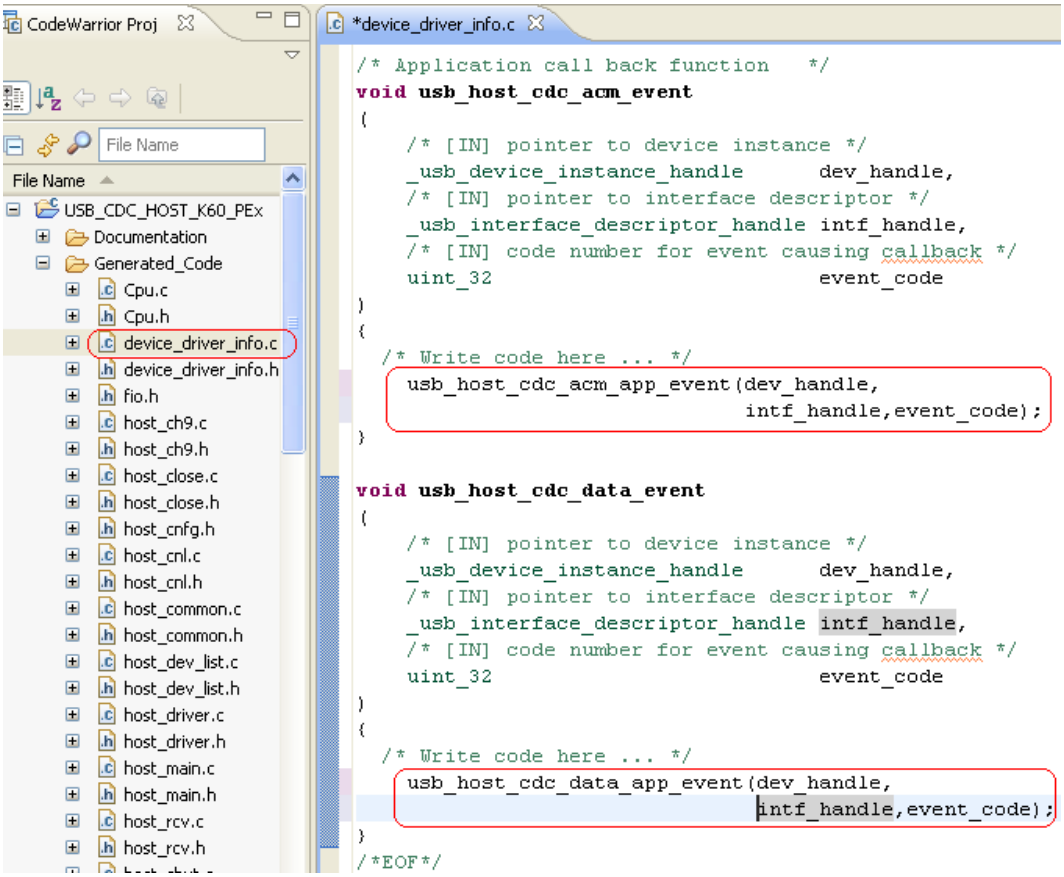


Figure 4-35 Driver Table Information

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:

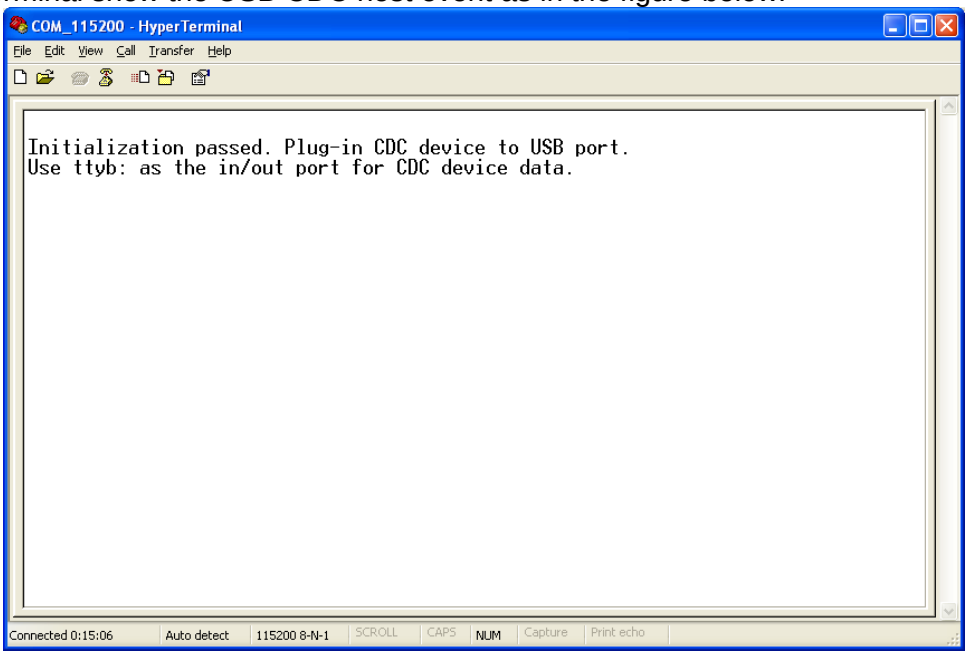


Figure 4-36 CDC host initializing

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

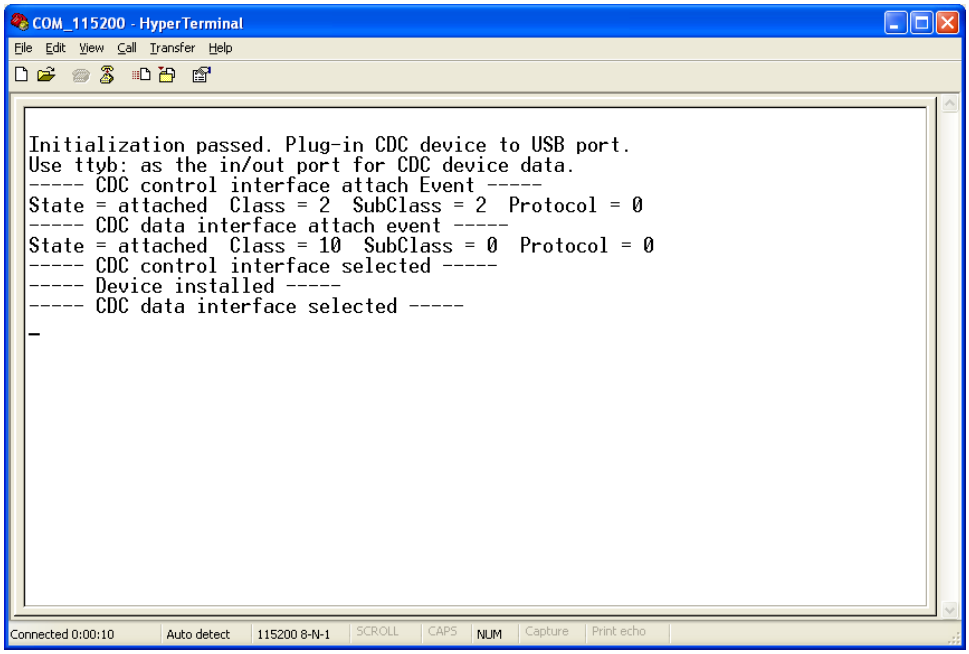


Figure 4-37 CDC host operating

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:

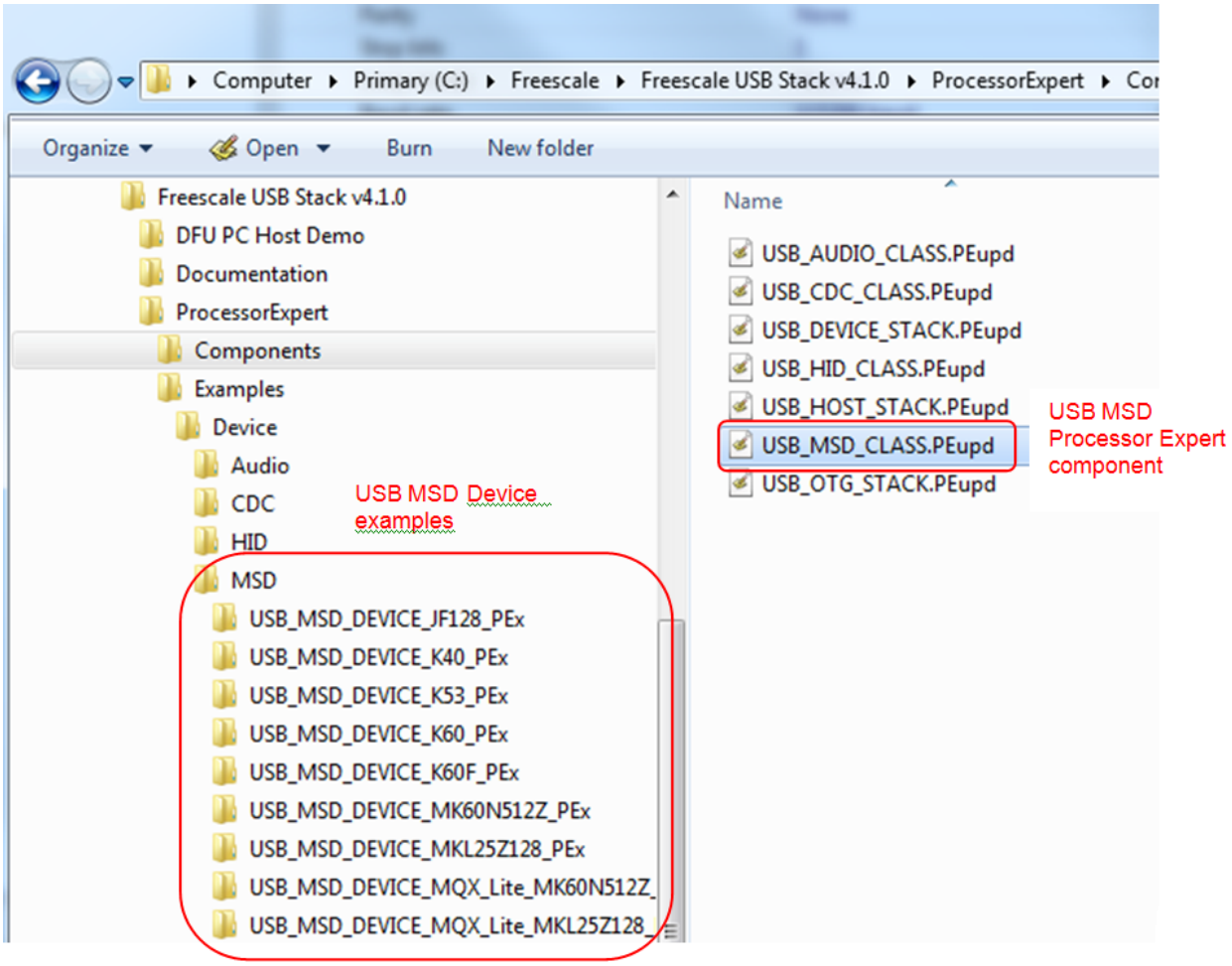


Figure 5-1: USB MSD Component packet

- 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:

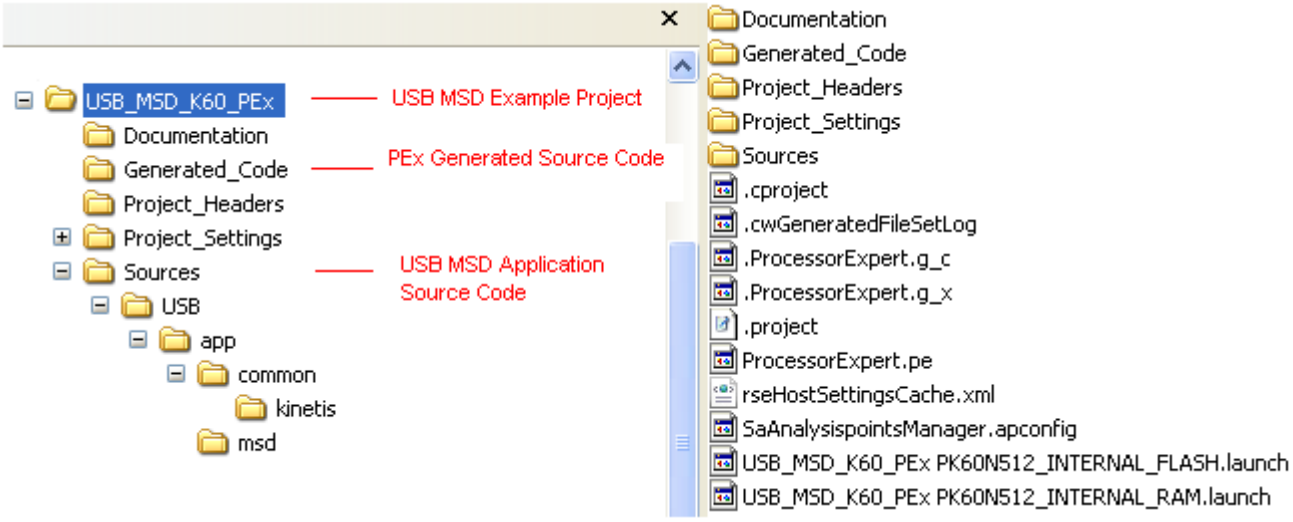


Figure 5-2: USB MSD example package

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

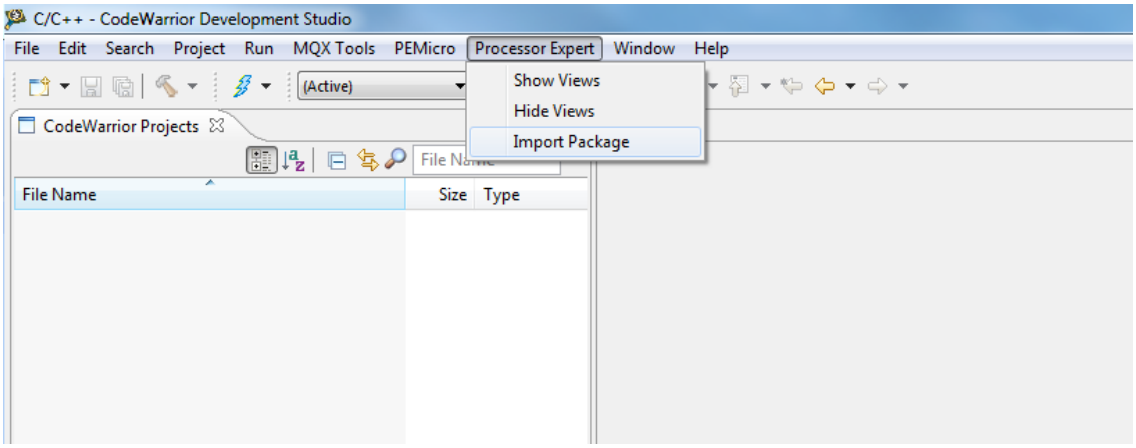


Figure 5-3 Processor Expert – Import Package

Step2. A browser window appears. Navigate to the folder that contains the USB_MSD_CLASS component, chose it and click open.

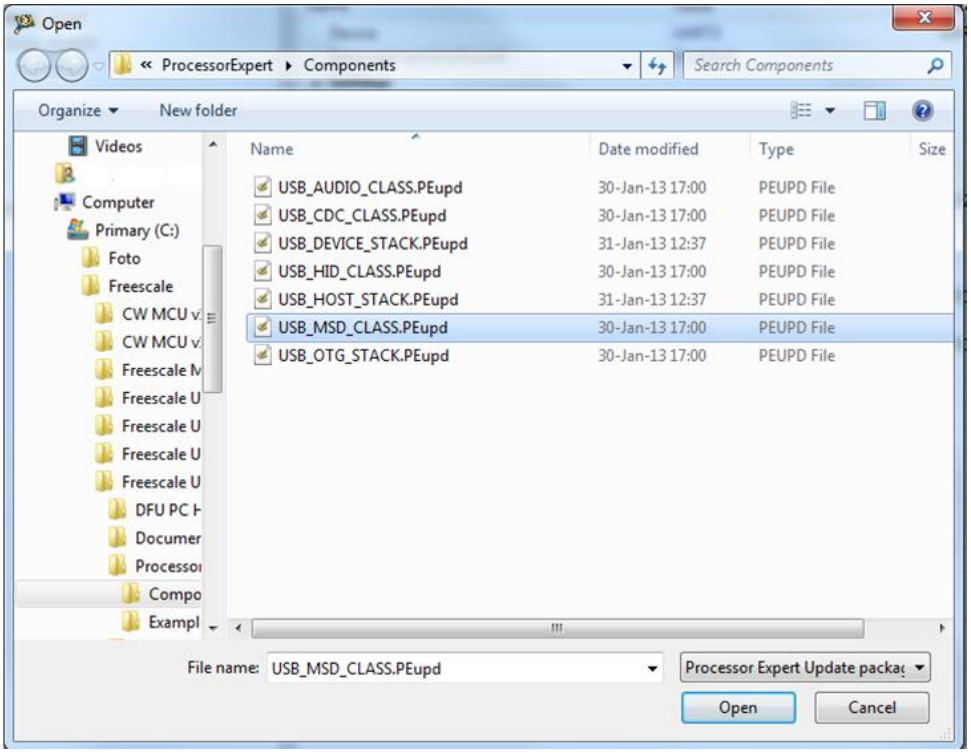


Figure 5-4: Open USB MSD package

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

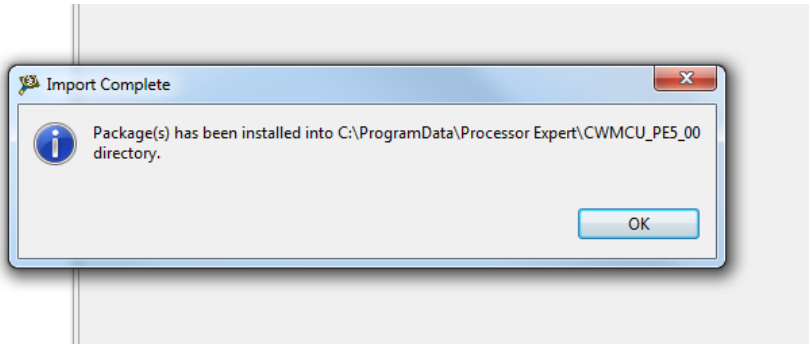


Figure 5-5: Import USB components

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.

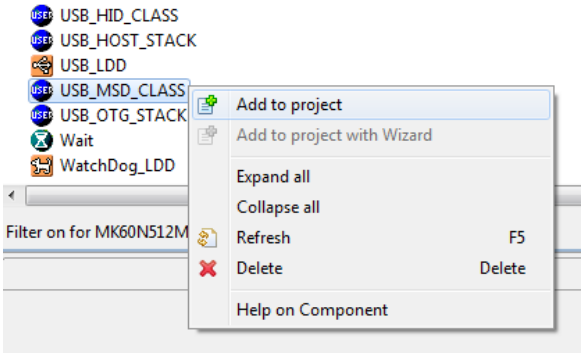


Figure 5-6: Components Library

Step2. Configure CPU clock to run USB module

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

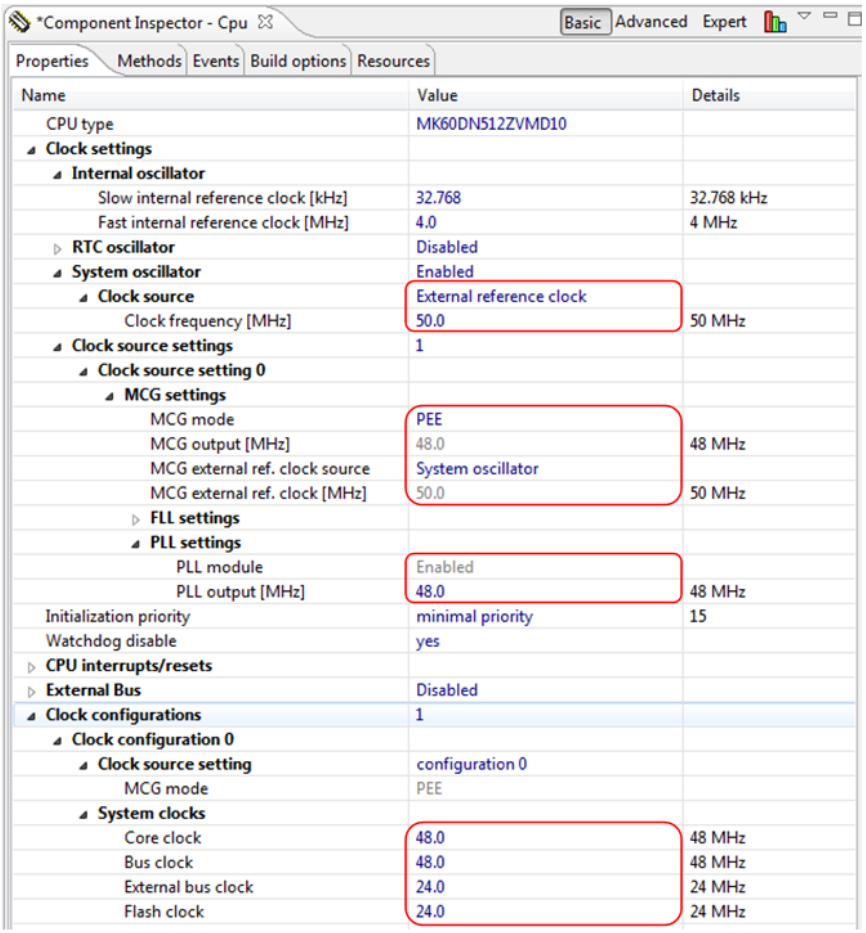


Figure 5-7 Clock source settings

- Chose advanced tab

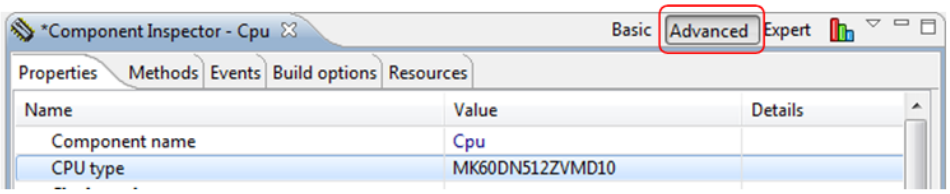


Figure 5-8: Open Advanced Tab

- PLL/FLL clock selection:
  - PLL clock

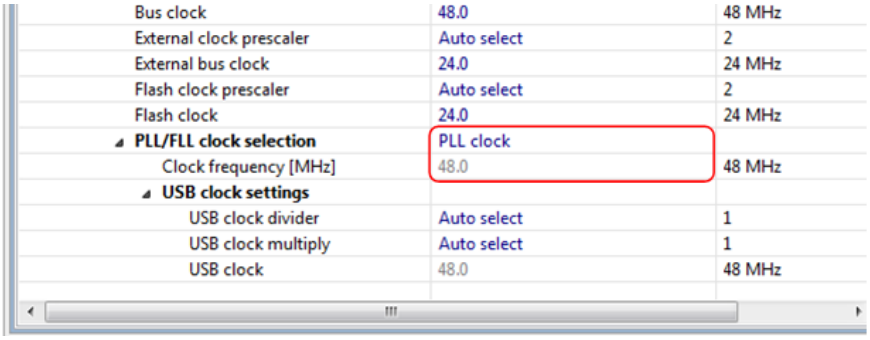


Figure 5-9: Clock configuration

Step3. Setup the USB_MSD_CLASS component

- Select USB_MSD component → Fill all values into component value fields:

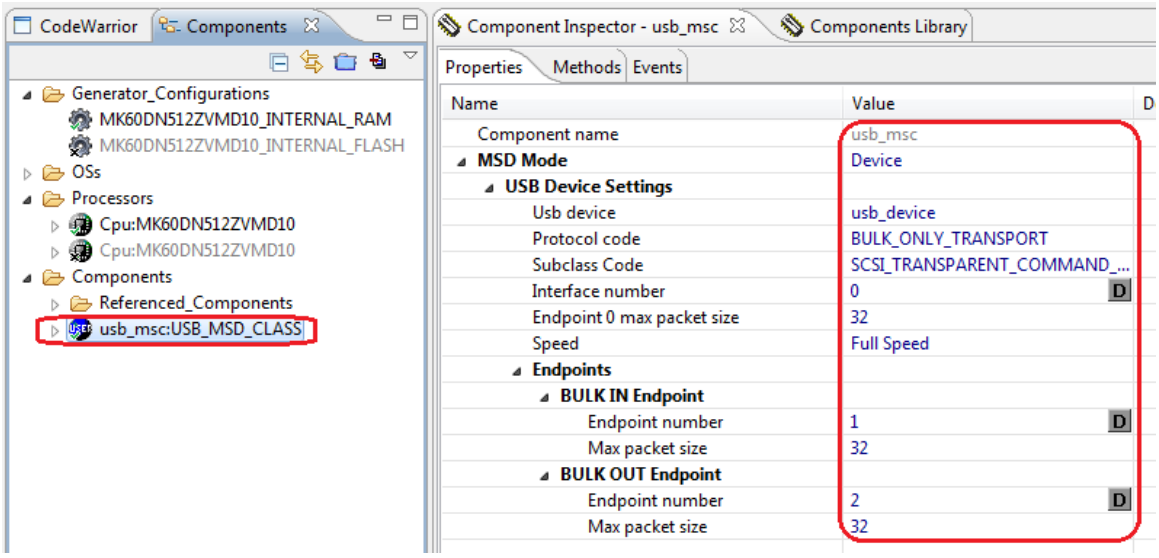


Figure 5-10: MSD device mode settings

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.
  - Device Descriptor:

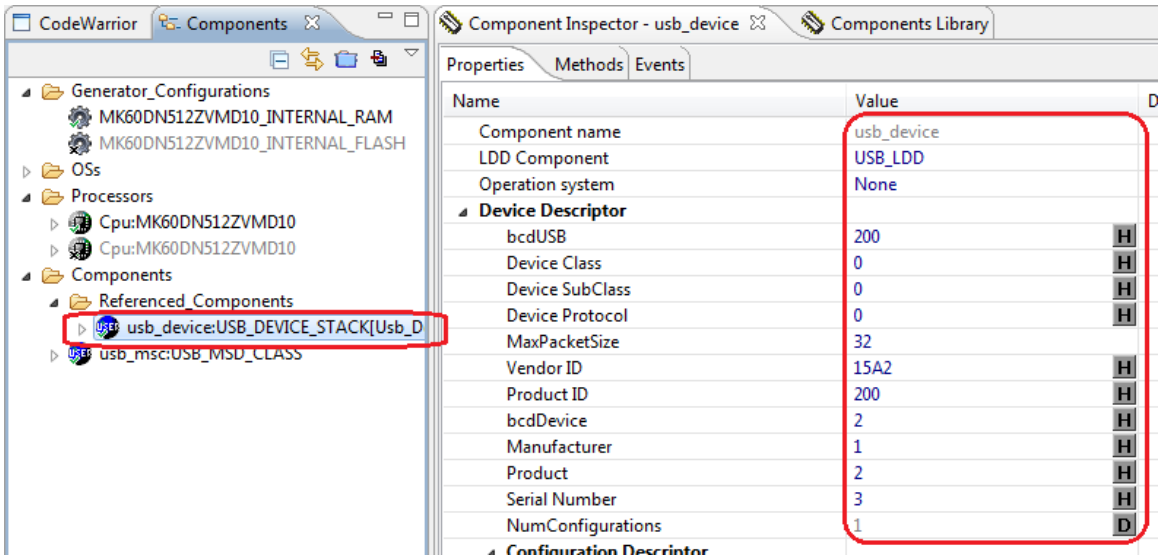


Figure 5-11: Device Descriptor settings

- Configuration Descriptor: This group is inherited in the USB_MSD_CLASS component.

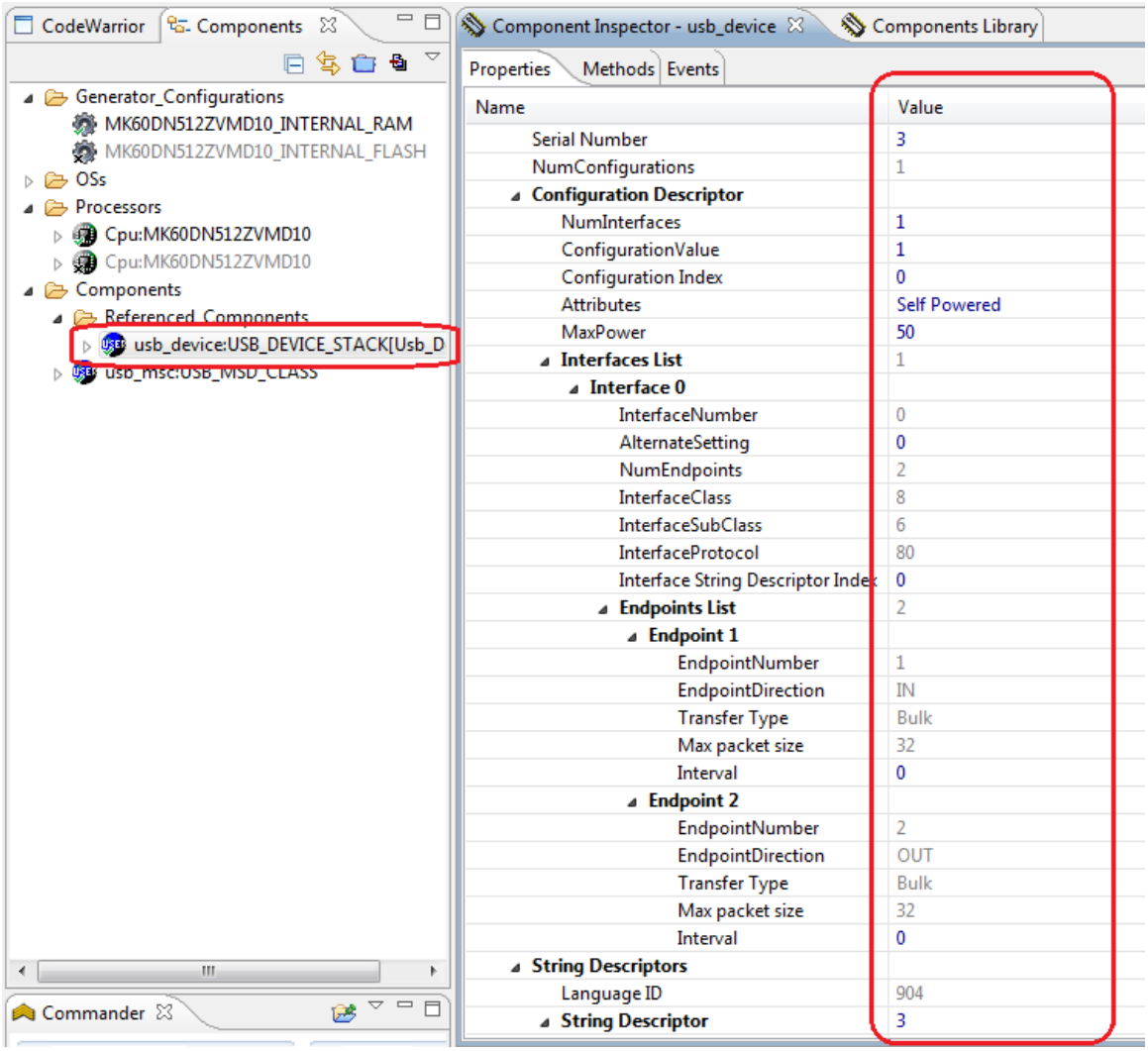


Figure 5-12: Configuration Descriptor settings

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

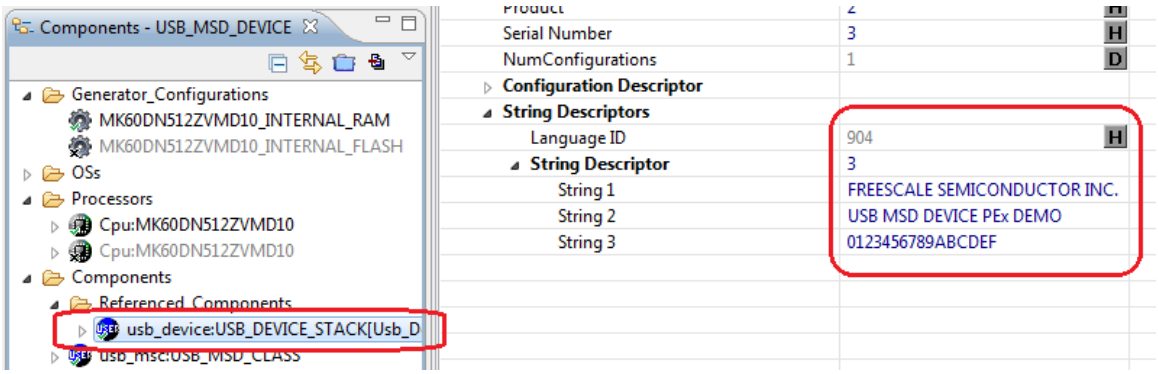


Figure 5-13: String Descriptor settings

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

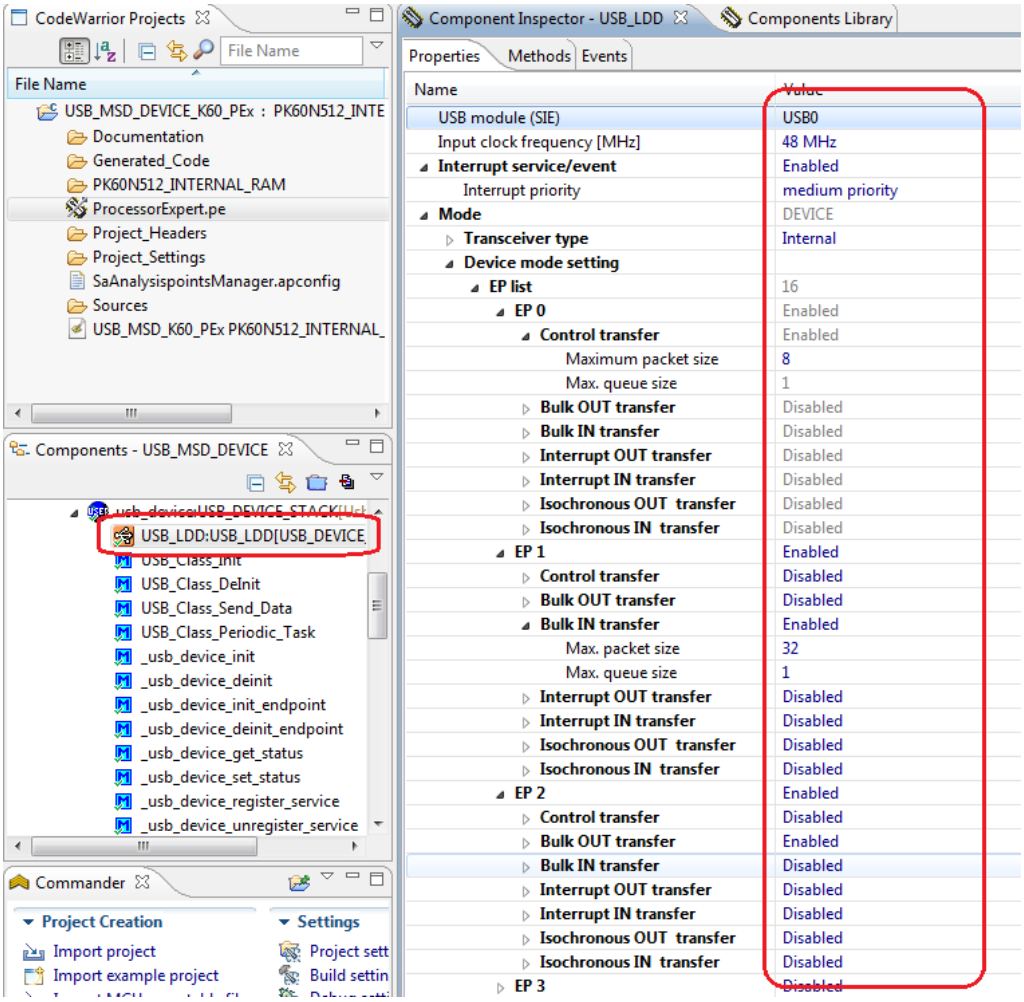


Figure 5-14: USB_LDD component

- Enable events

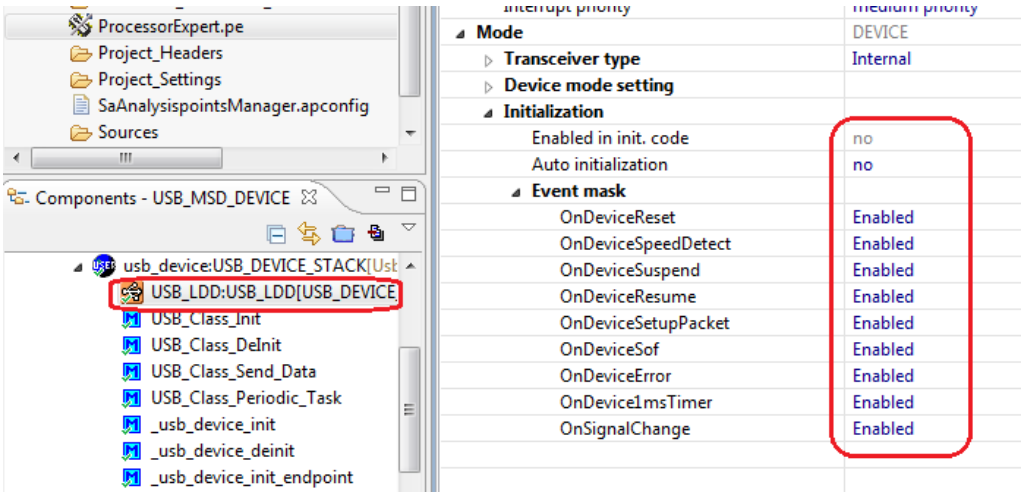


Figure 5-15: Enable events

Step6. Generate Processor Expert code:
After setting the components, select ProcessorExpert.pe in the project folder → Right click → Chose Generate Processor Expert Code field to generate source code.

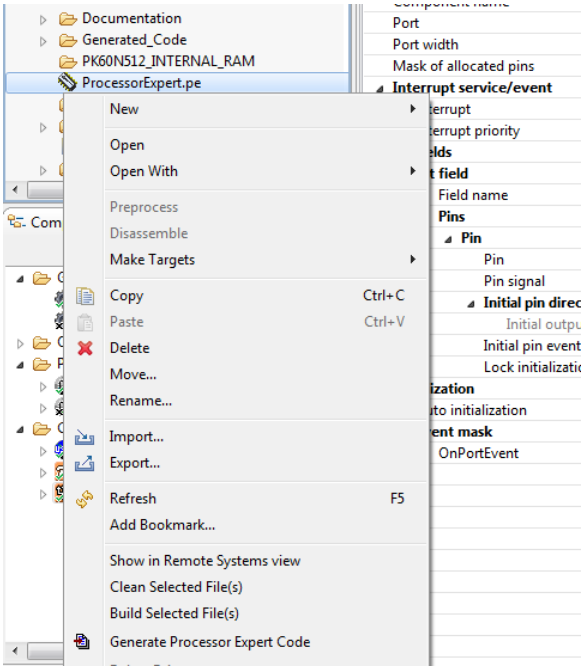


Figure 5-16: Generate Processor Expert code

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

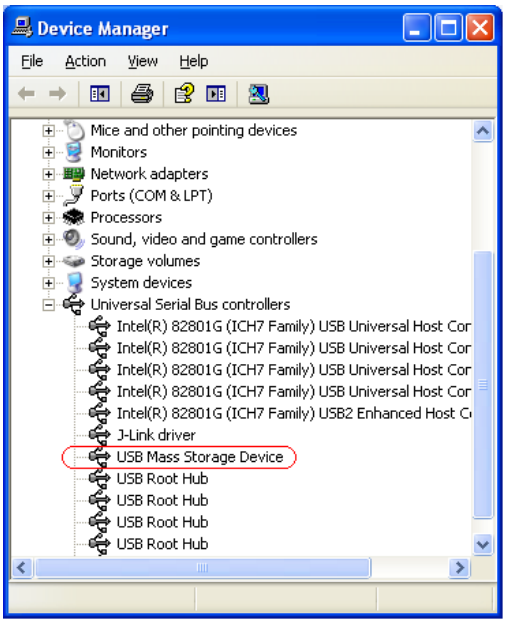


Figure 5-17: Device entry

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

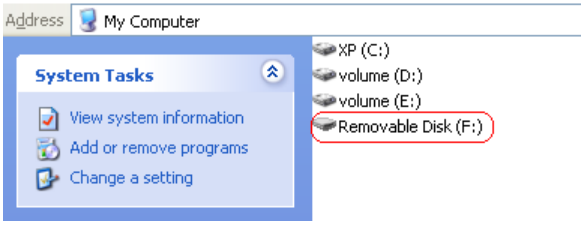
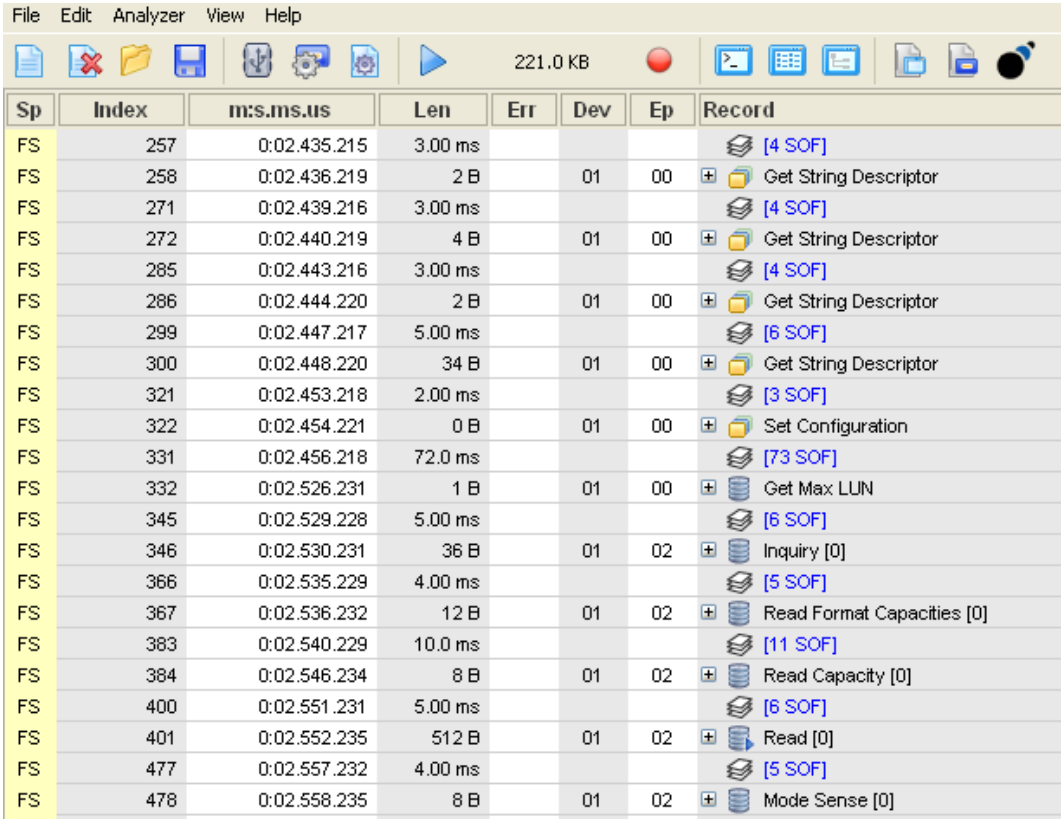


Figure 5-18: Removable disk

- 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.

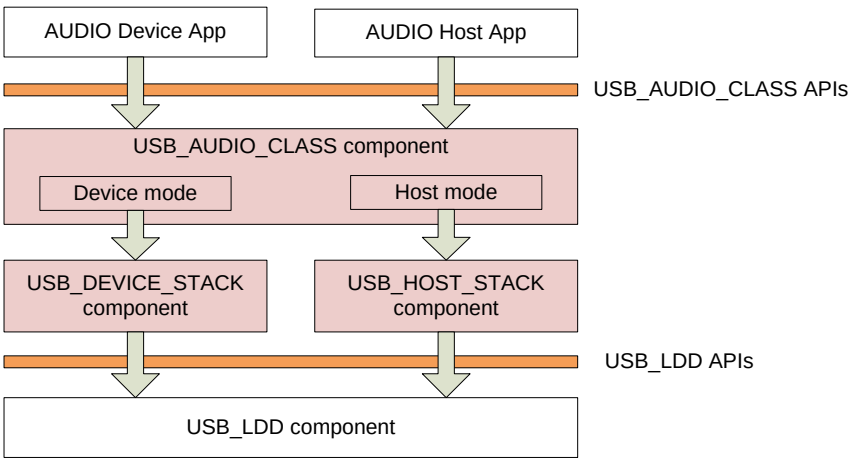


Figure 6-1 Architecture overview

6.2 Folder structure

6.2.1 USB Audio component package structure

The deliver package includes items as the following figure:

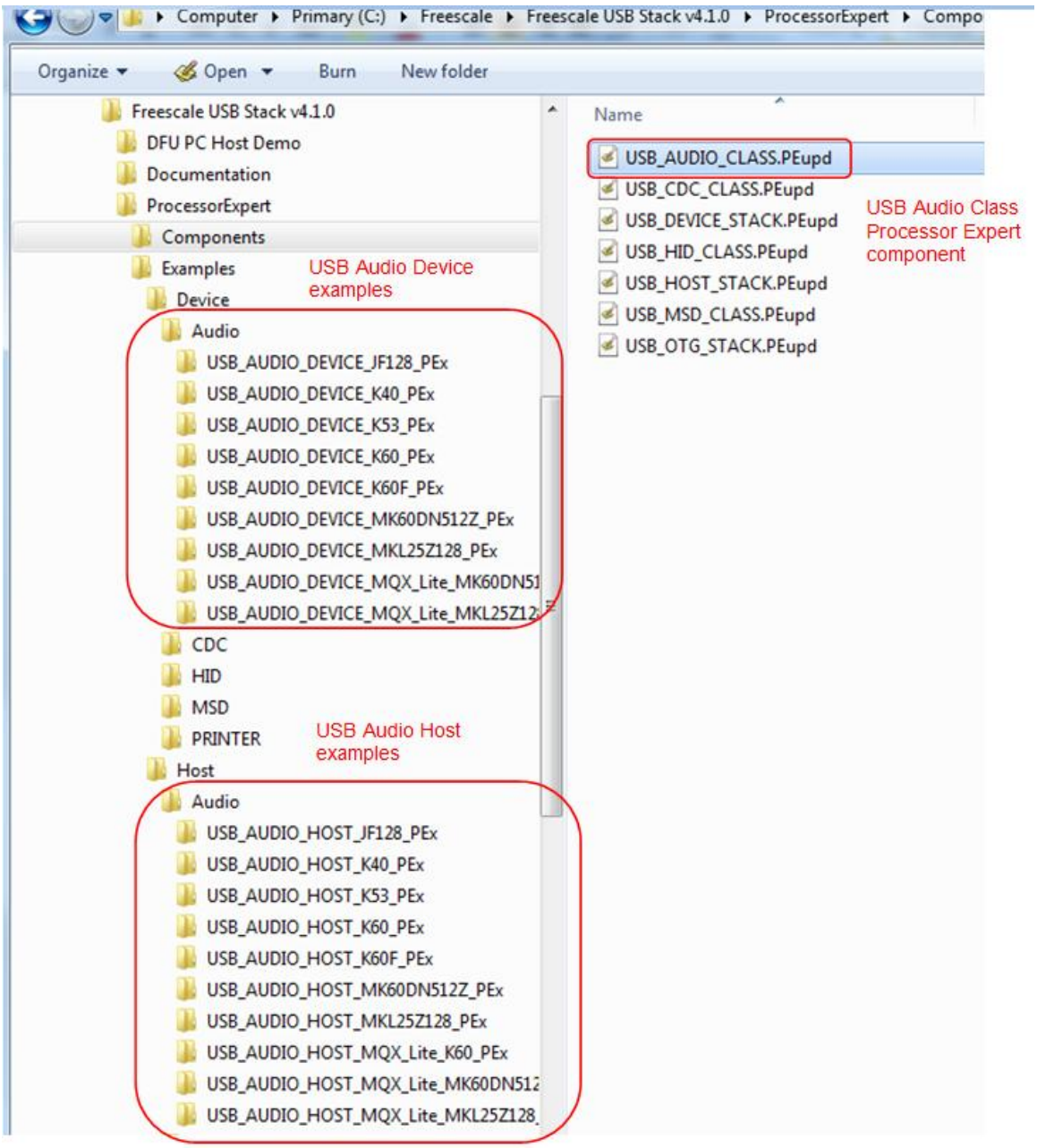


Figure 6-2 Delivered Package Directory Structure

- 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:

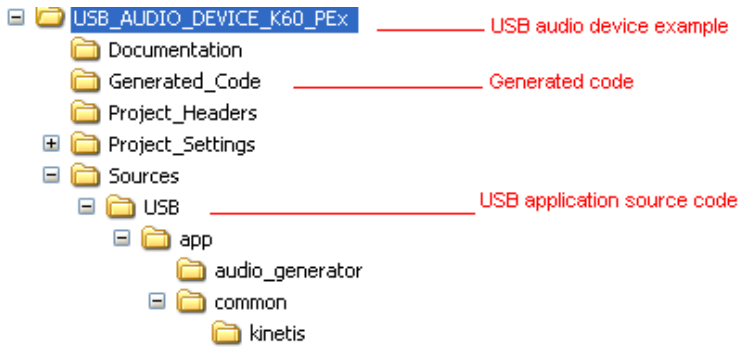


Figure 6-3 AUDIO device folder structure

6.2.3 AUDIO host example structure

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

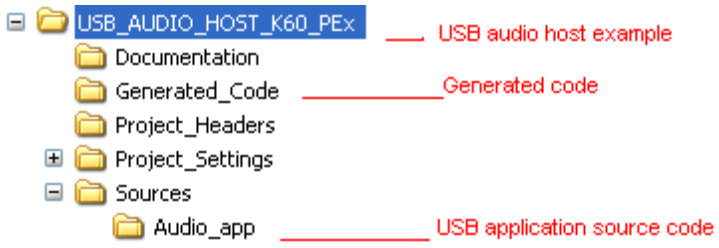


Figure 6-4 AUDIO host folder structure

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

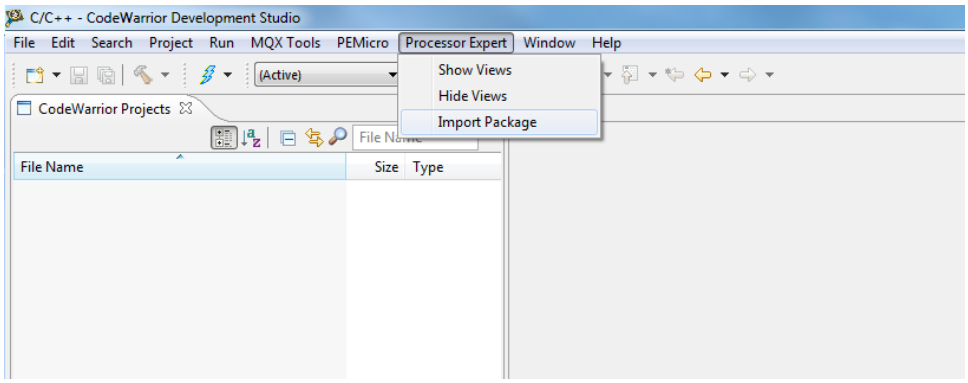


Figure 6-5 Processor Expert Import Package

Step2. A browser window appears. Navigate to the folder that contains the USB_AUDIO_CLASS component, chose it and click open.

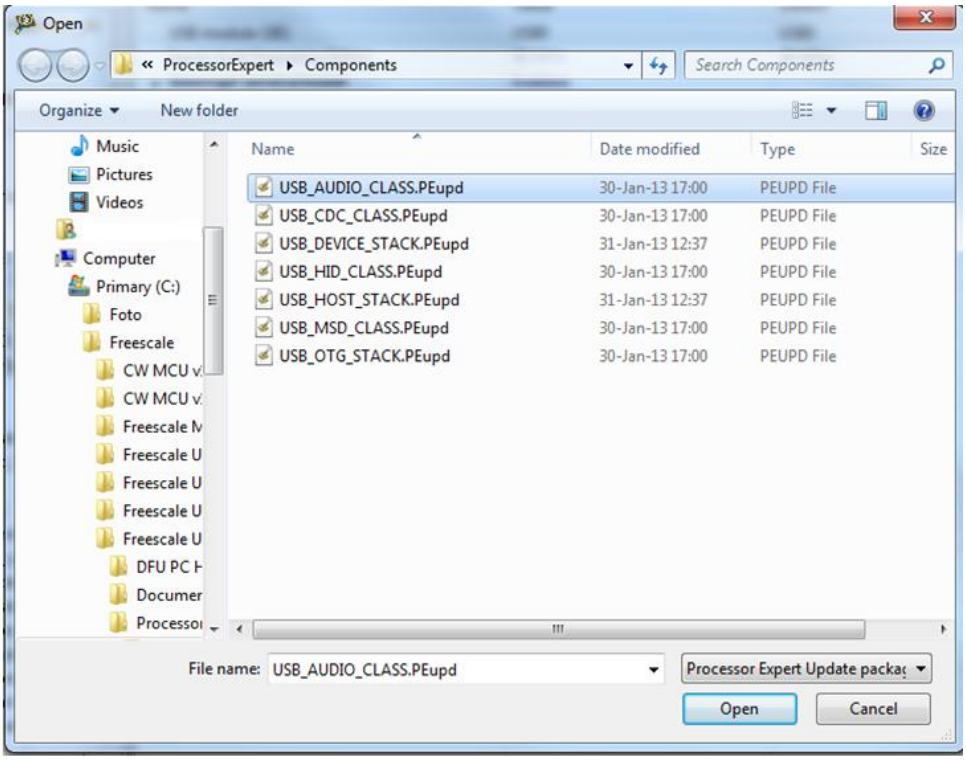


Figure 6-6 Deliver package navigation

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

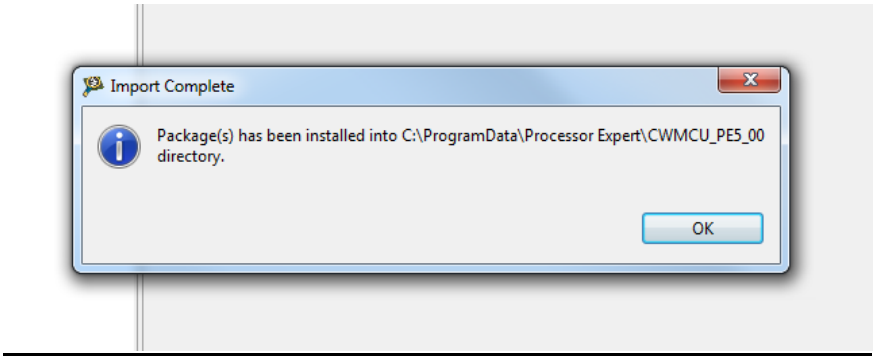


Figure 6-7 Import USB components

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.

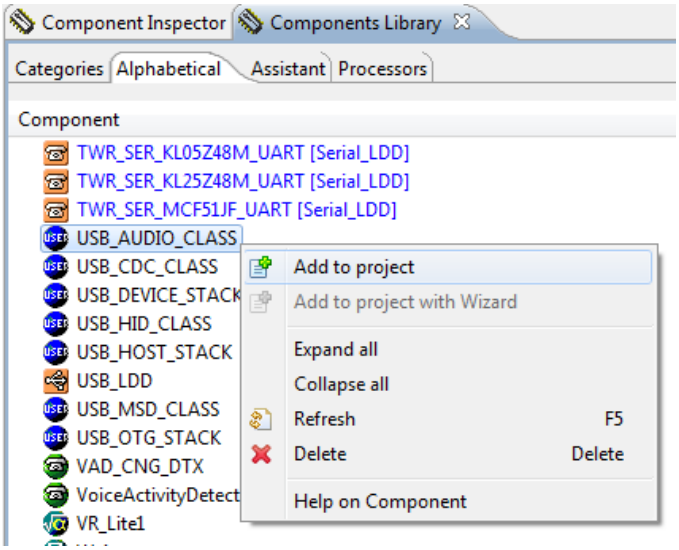


Figure 6-8 Components Library

Step2. Configure CPU clock to run USB module

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

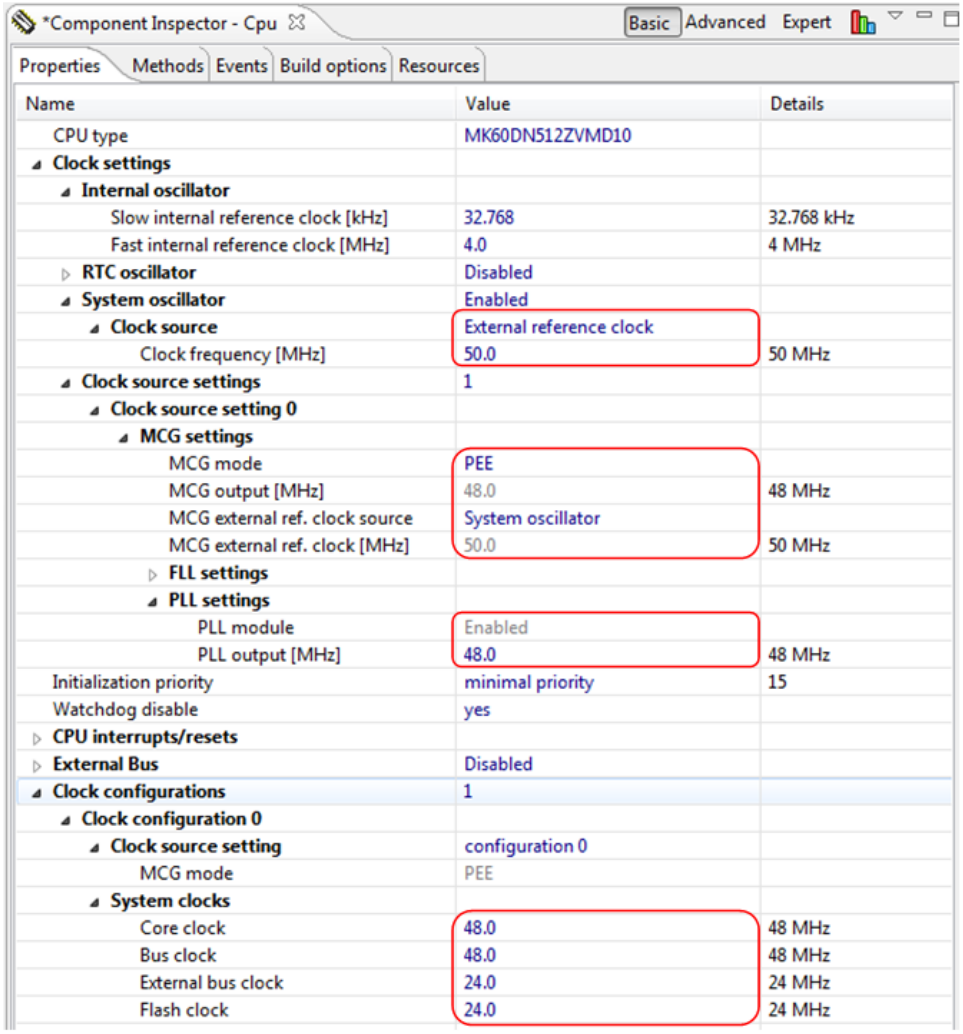


Figure 6-9 Clock source settings

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

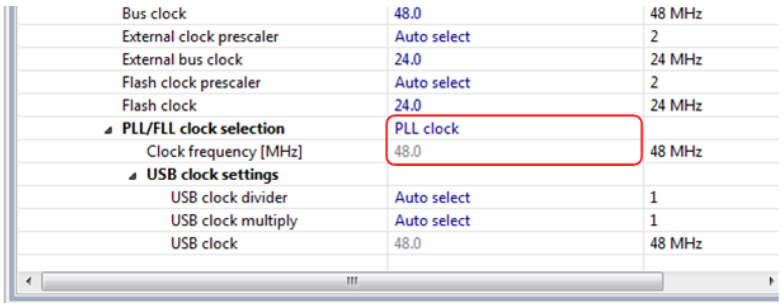


Figure 6-10 Clock configuration

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.

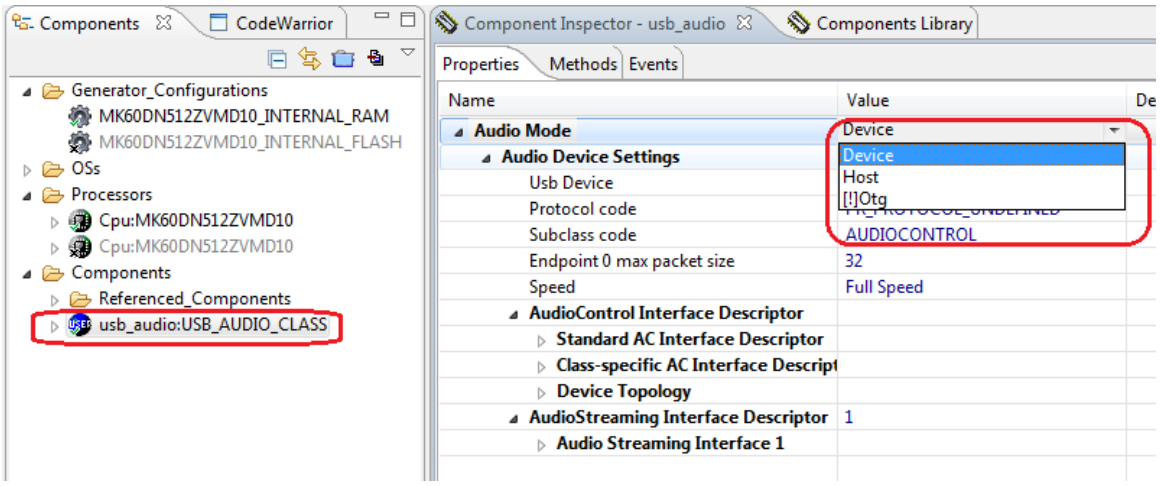


Figure 6-11 AUDIO device mode setting

- Fill input parameters of the USB_AUDIO_CLASS component as in the following figure:

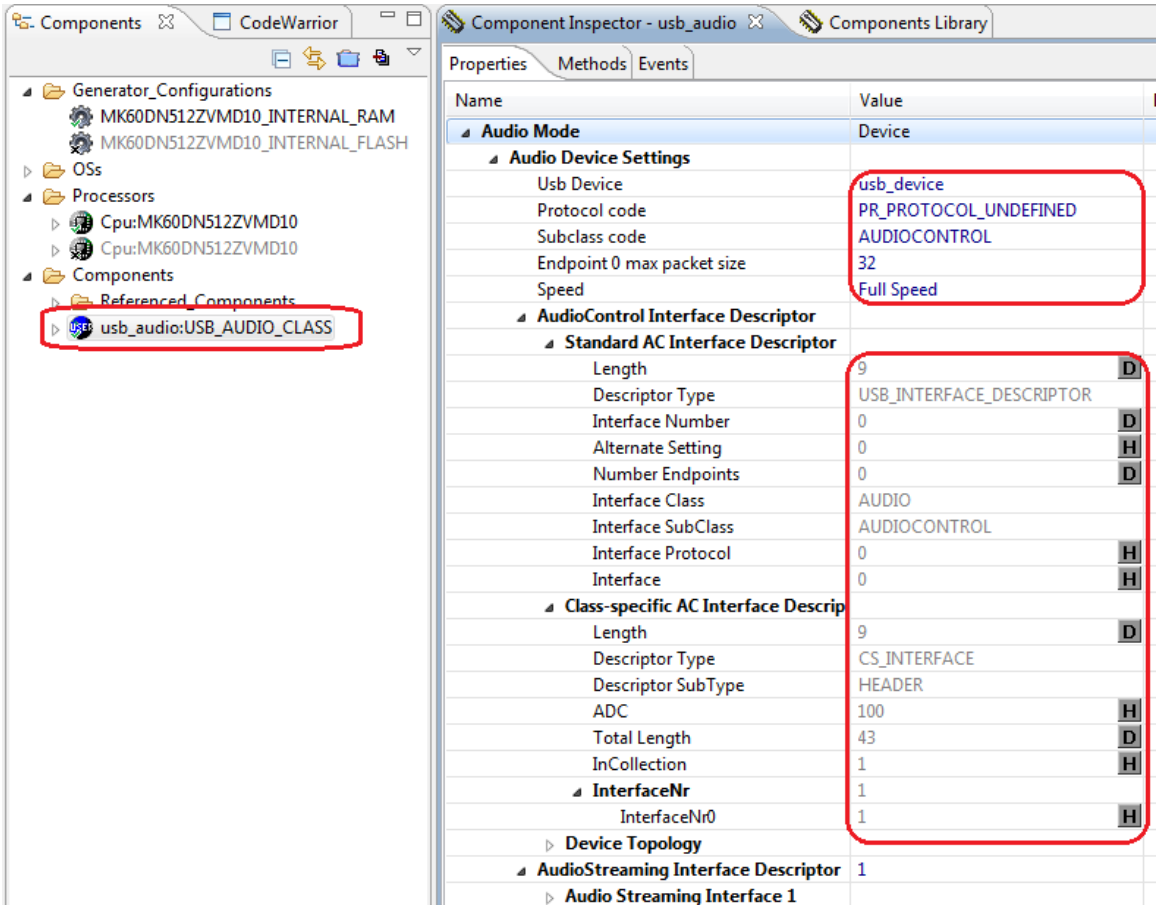


Figure 6-12 Audio Control Interface settings

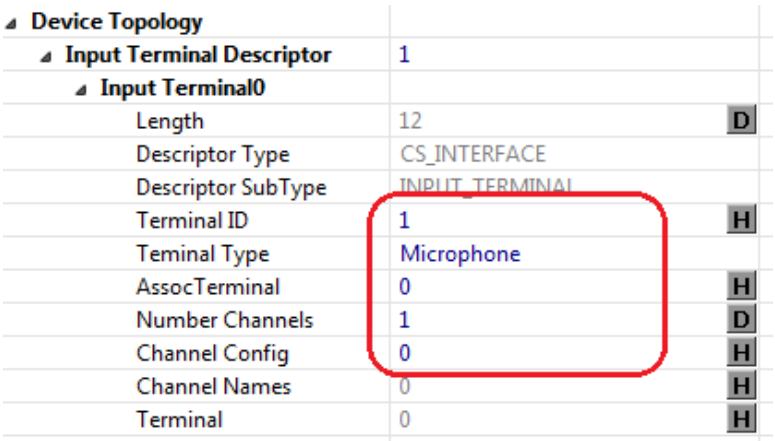


Figure 6-13 Input terminal settings

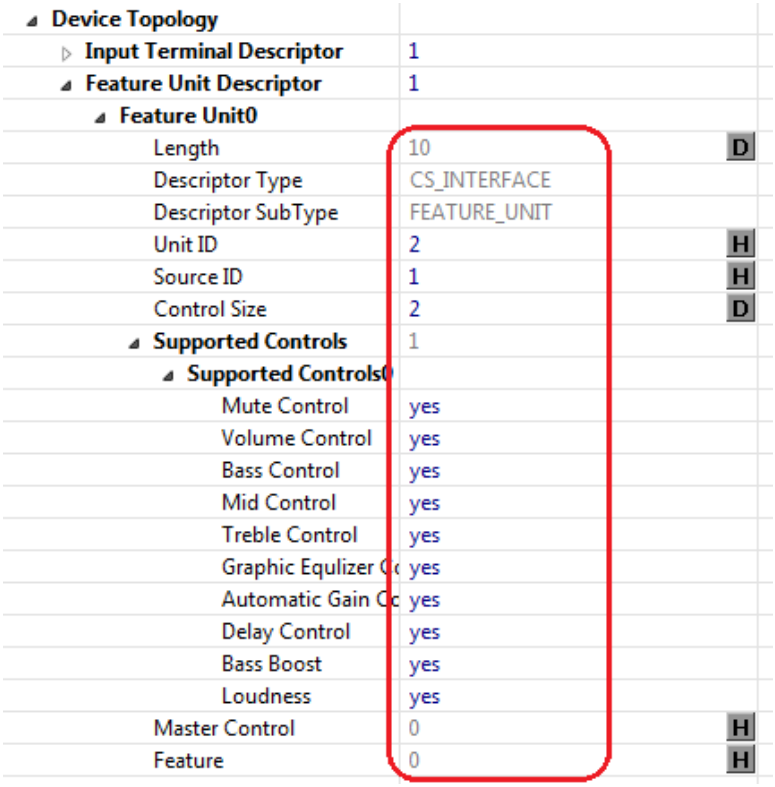


Figure 6-14 Feature Unit settings

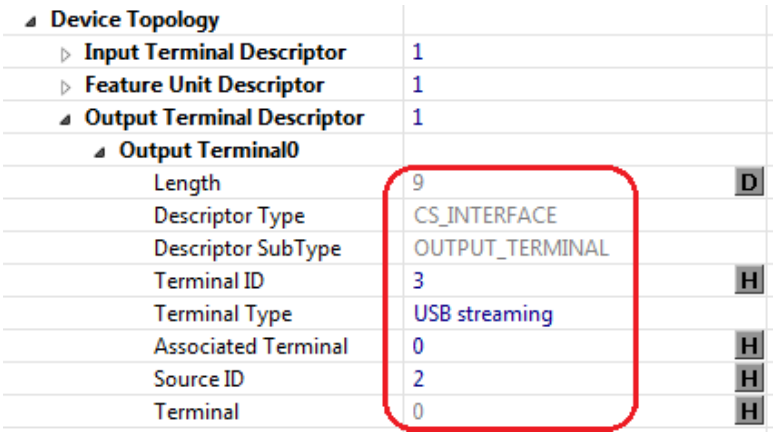


Figure 6-15 Output terminal settings

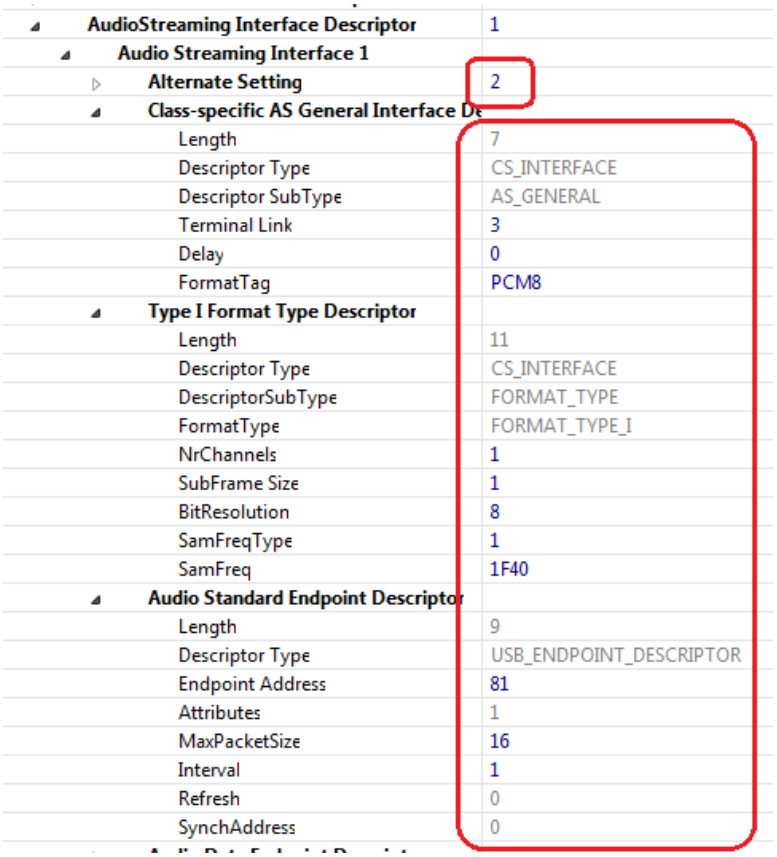


Figure 6-16 Audio Streaming Interface settings

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:
    - Device Descriptor :

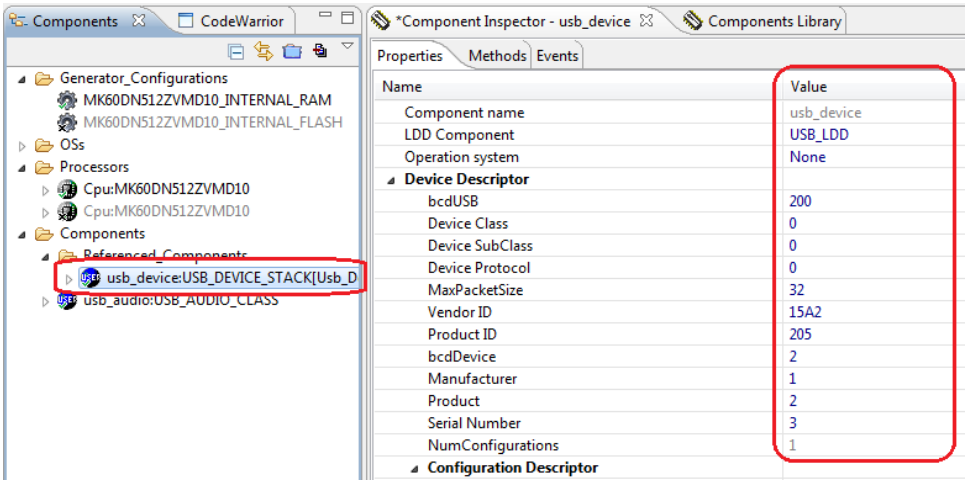


Figure 6-17 Device Descriptor settings

- Configuration Descriptor: This group is inherited the USB_AUDIO_CLASS component.

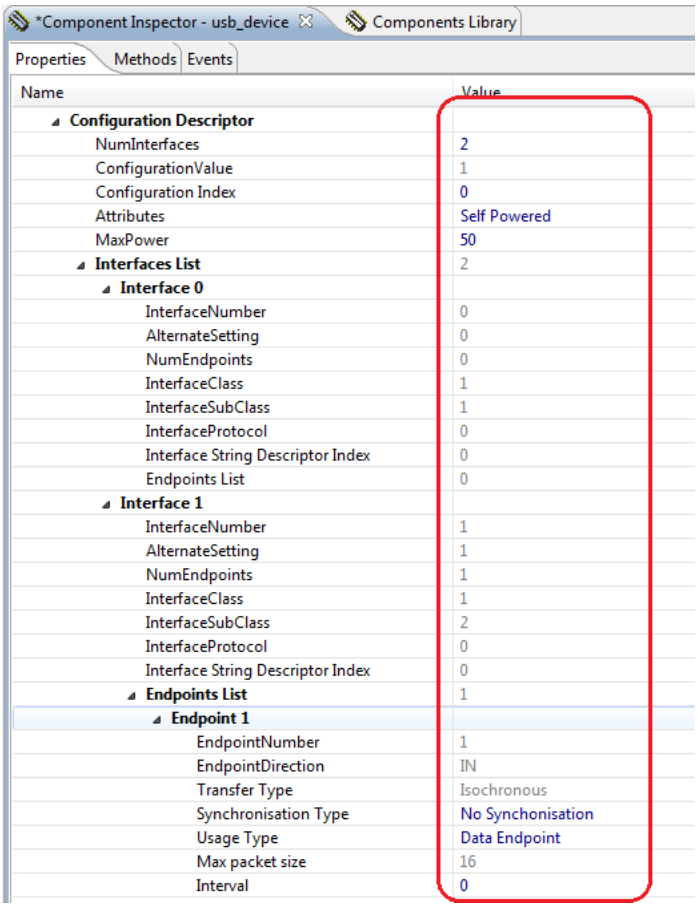


Figure 6-18 Configuration Descriptor settings

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

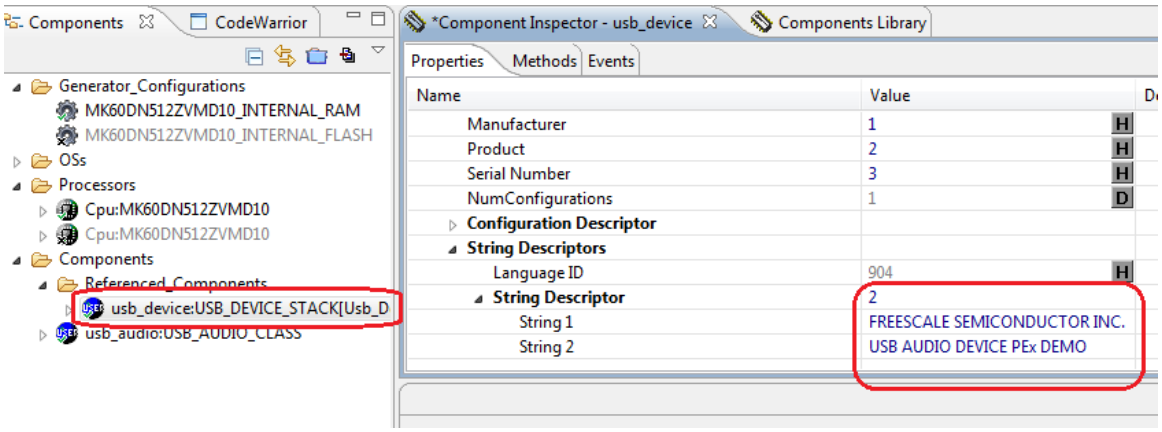


Figure 6-19 String Descriptor settings

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:

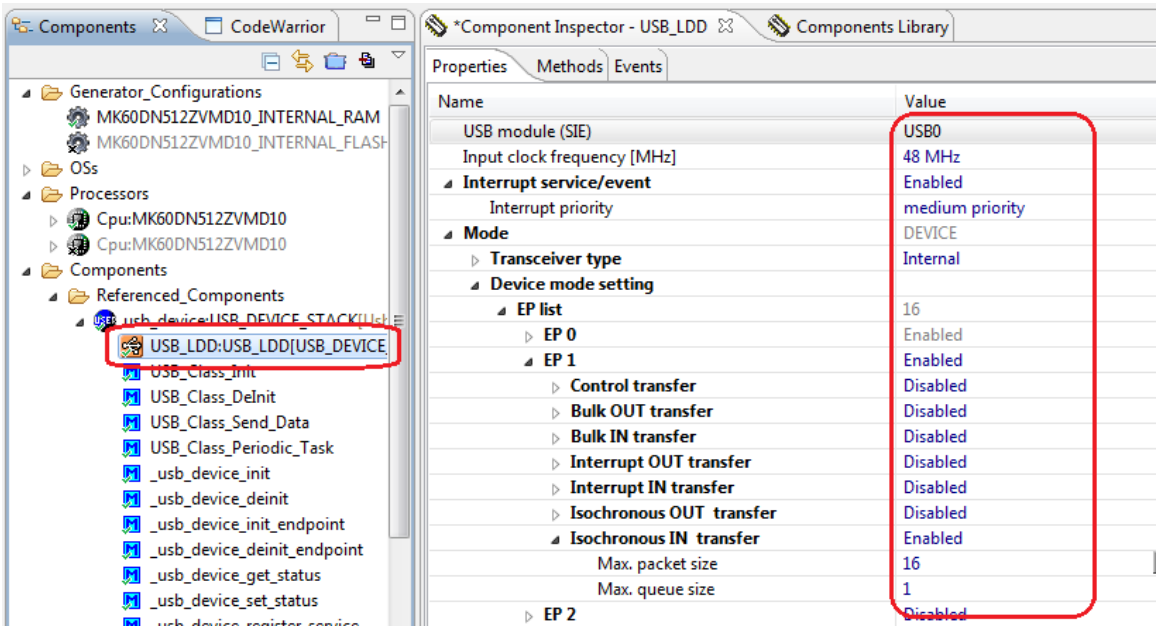


Figure 6-20 USB_LDD component settings

Step6. Generate Processor Expert code:
After setting the components, select ProcessorExpert.pe in the project folder → Right click → Chose Generate Processor Expert Code field to generate source code.

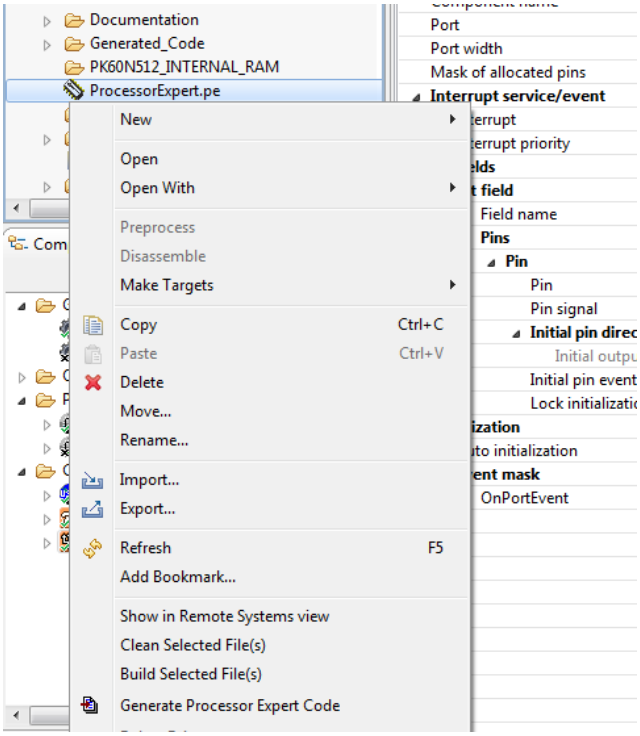


Figure 6-21 Generate Processor Expert code

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:

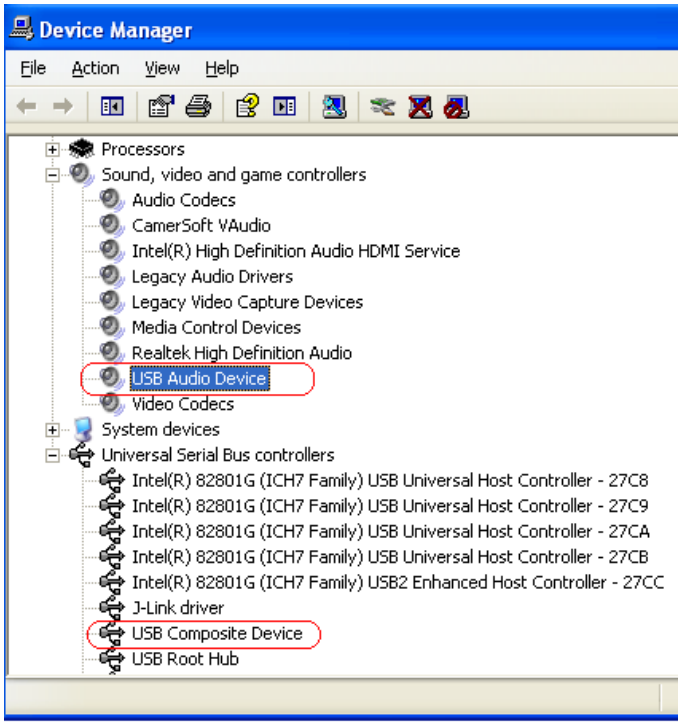


Figure 6-22 Device entry

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

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

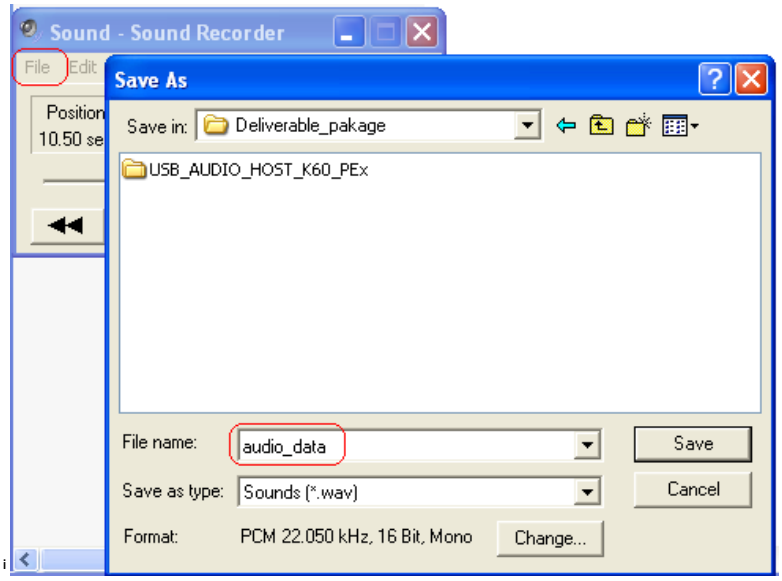


Figure 6-27 Sound Recorder Operating

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.

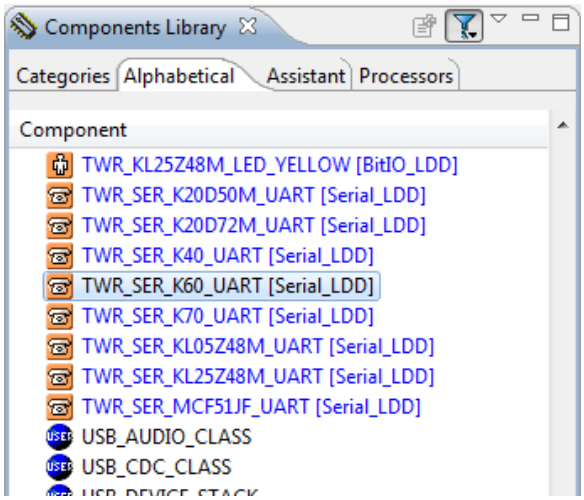


Figure 6-28 Component show views

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.

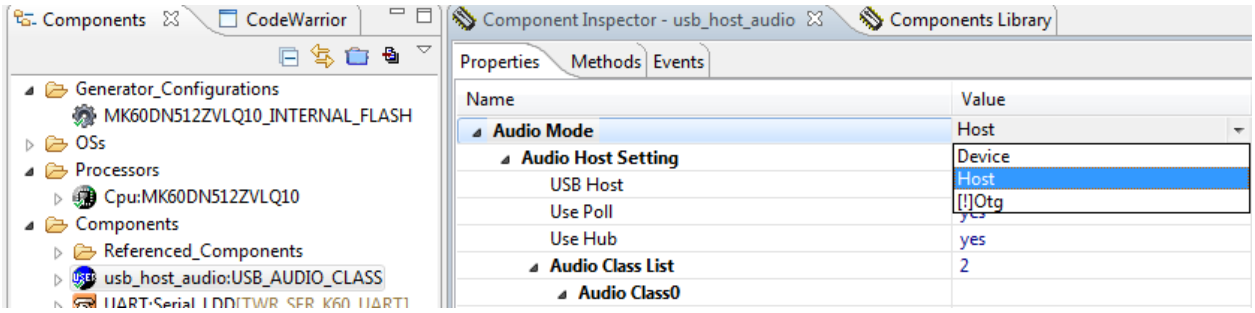


Figure 6-29 AUDIO host mode setting

- Fill input parameters into AUDIO class list settings:

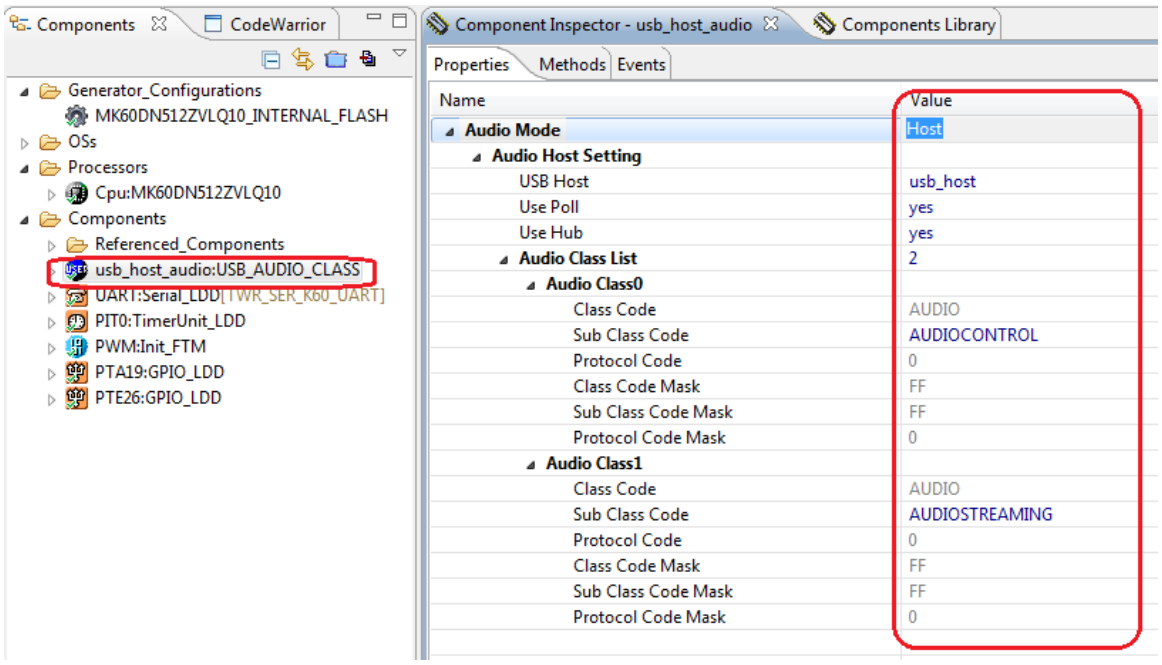


Figure 6-30 USB_AUDIO_CLASS component settings

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.

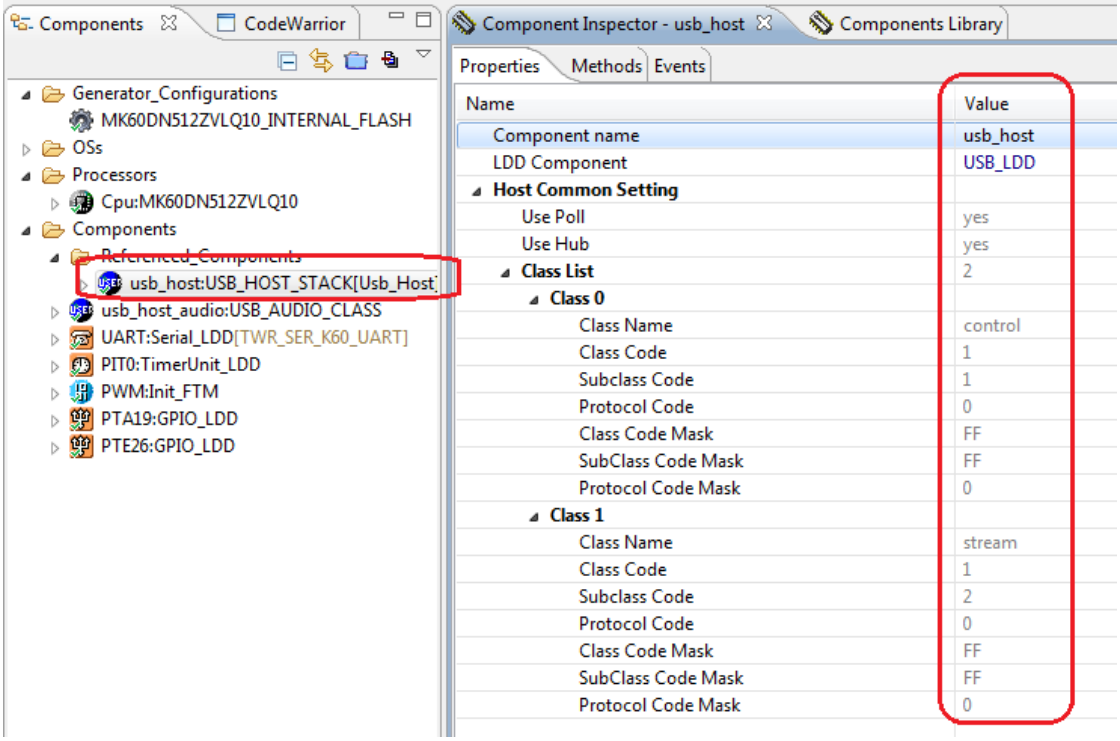


Figure 6-31 USB_HOST_STACK component settings

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:

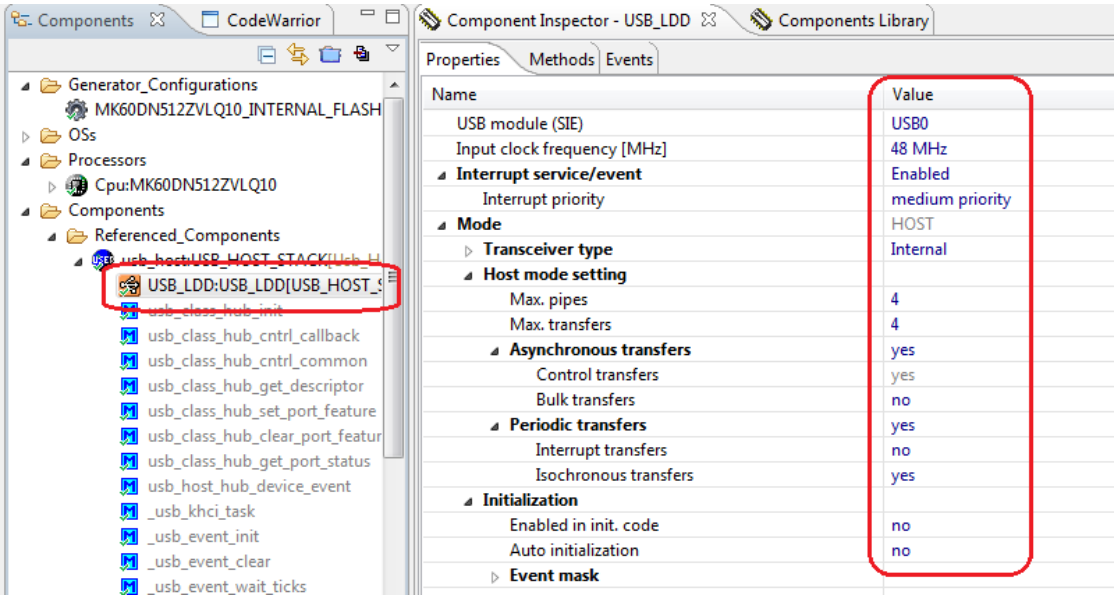


Figure 6-32 USB_LDD component settings

Step7. Setup the TWR_SER_K60_UART component as in the following figure:

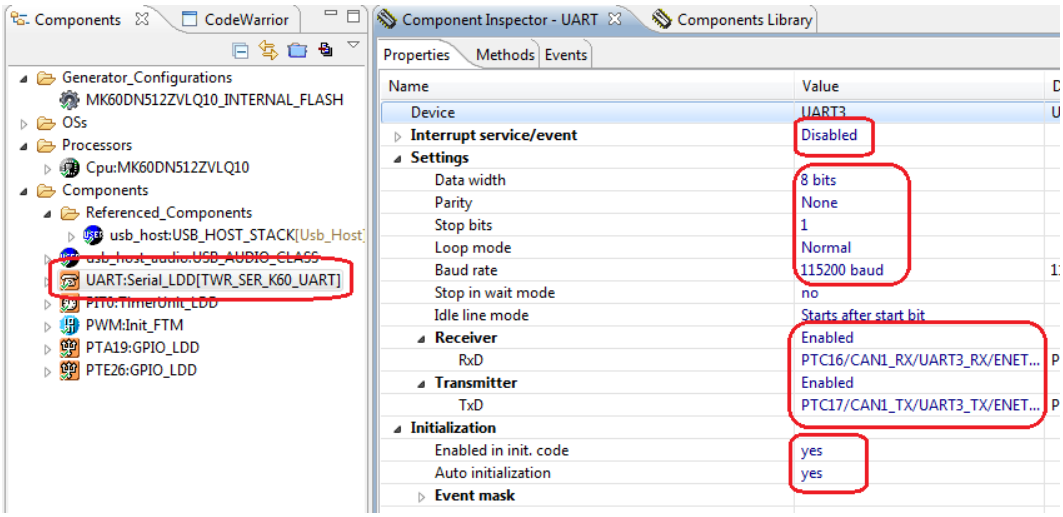


Figure 6-33 UART settings

Step7. Setup the TimerUnit_LDD component for PIT0 as in the following figures:

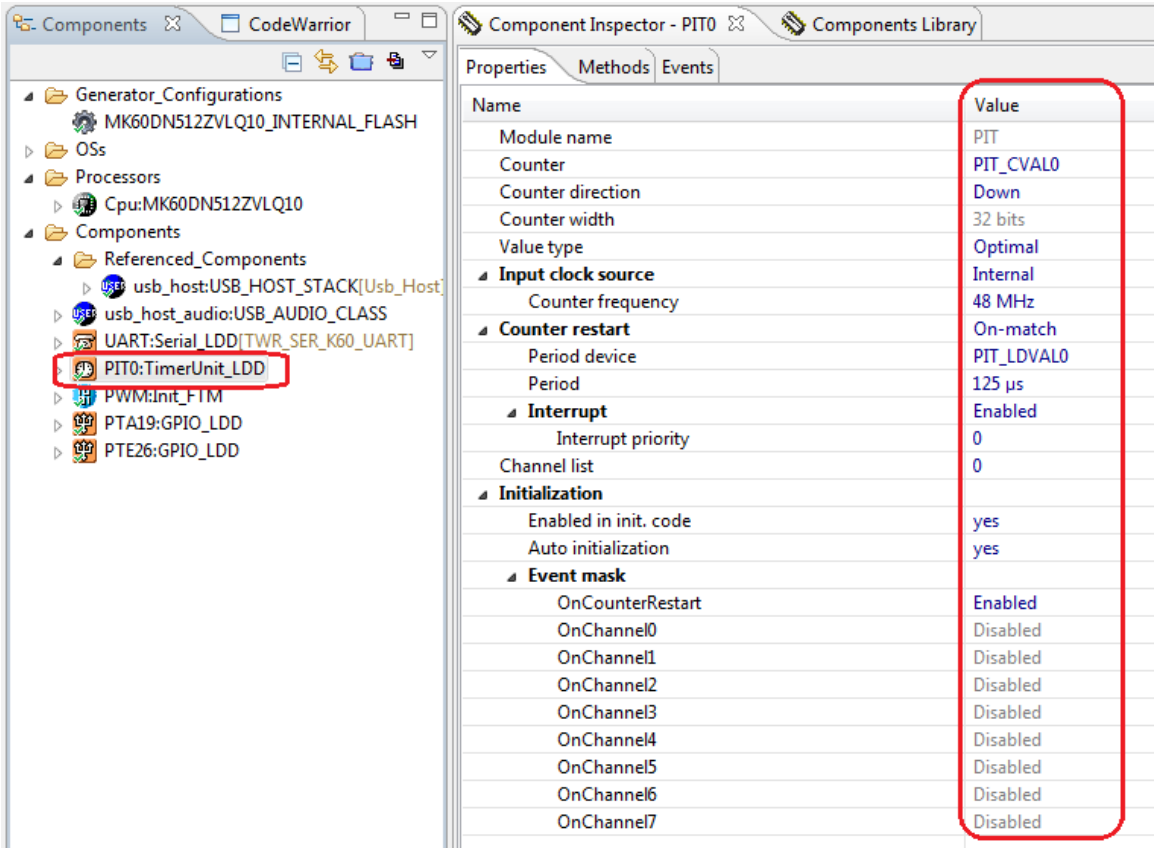


Figure 6-34 PIT 0 Properties Settings

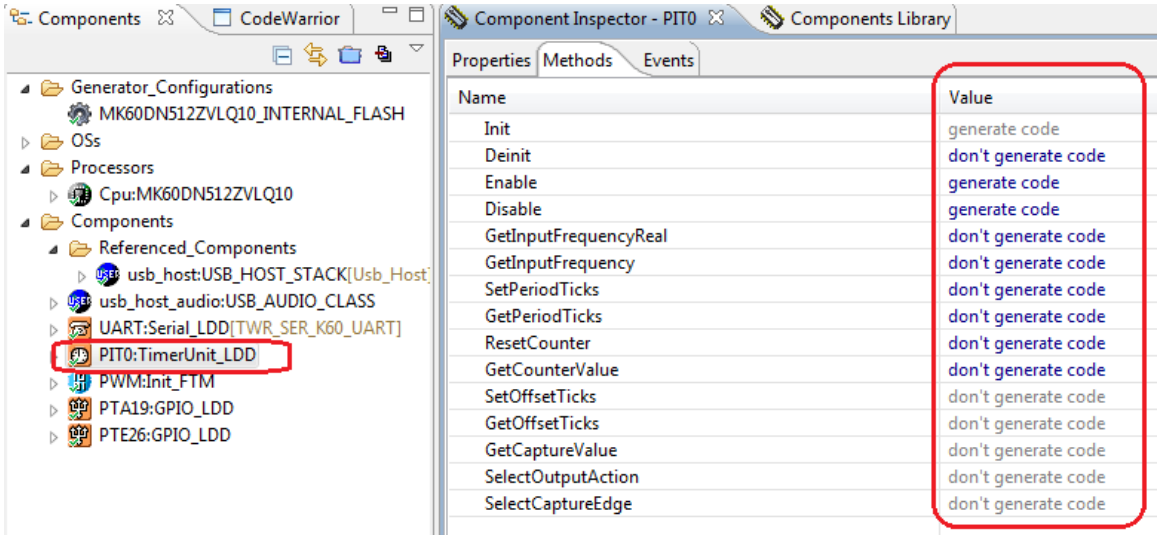


Figure 6-35 PIT 0 Methods Settings

Step8. Setup the Init_FTM component for PWM as in the following figure:

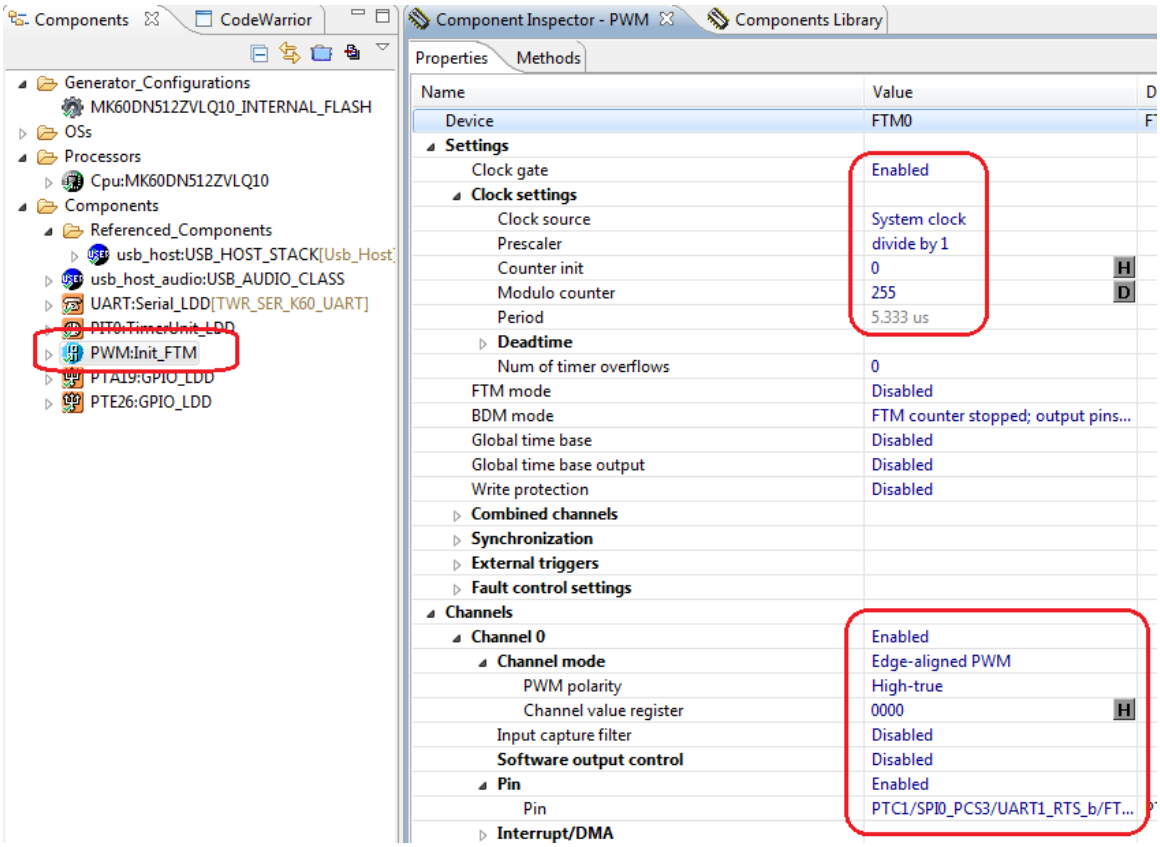


Figure 6-36 PWM settings

Step9. Setup the GPIO_LDD component for PTA19 as in the following figures:

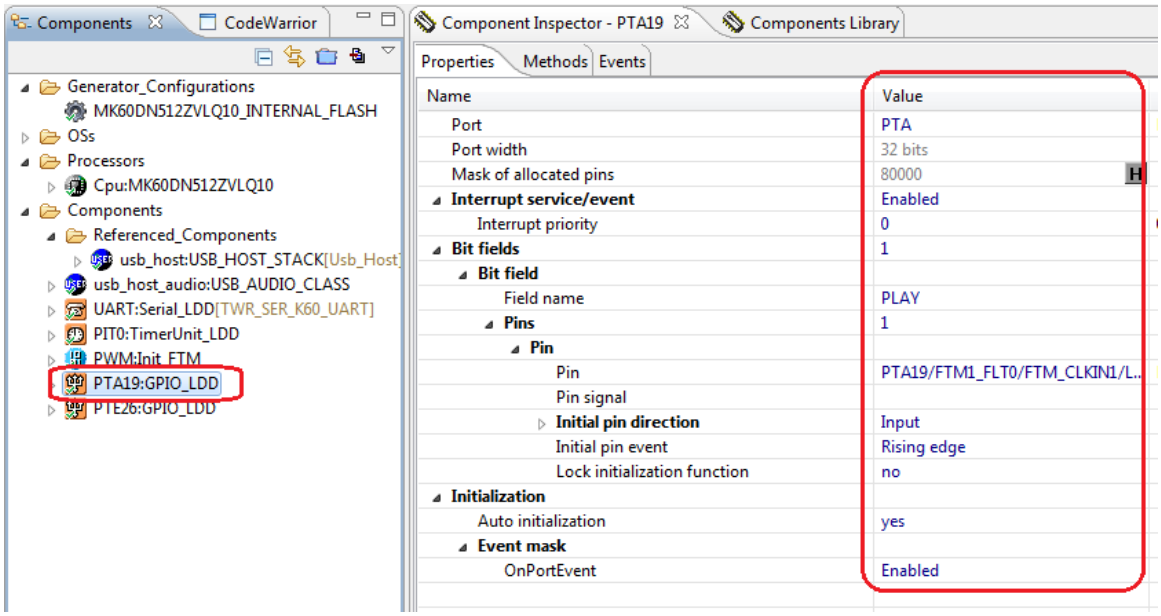


Figure 6-37 PTA19-“PLAY” Button Properties settings

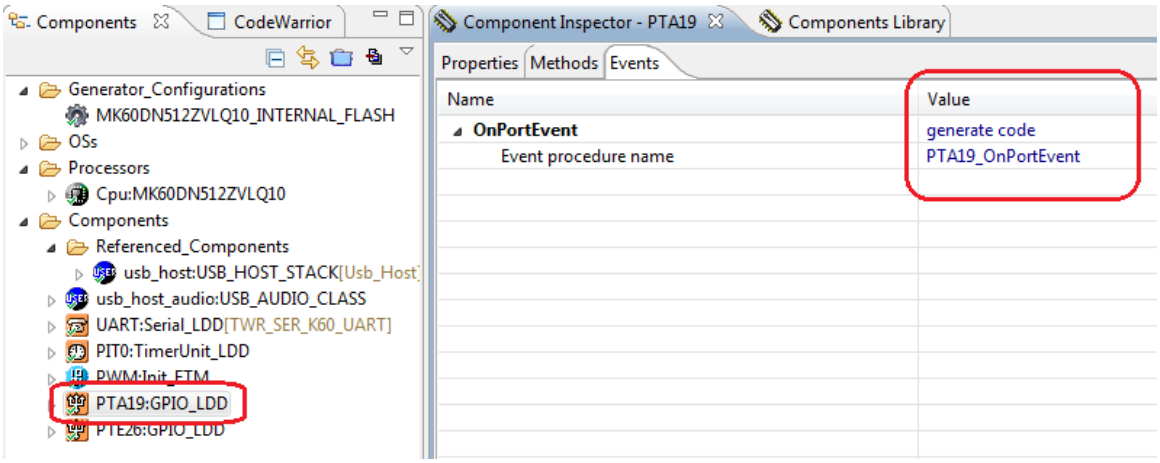


Figure 6-38 PTA19-“PLAY” Button Events settings

Step9. Setup the GPIO_LDD component for PTE26 as in the following figures:

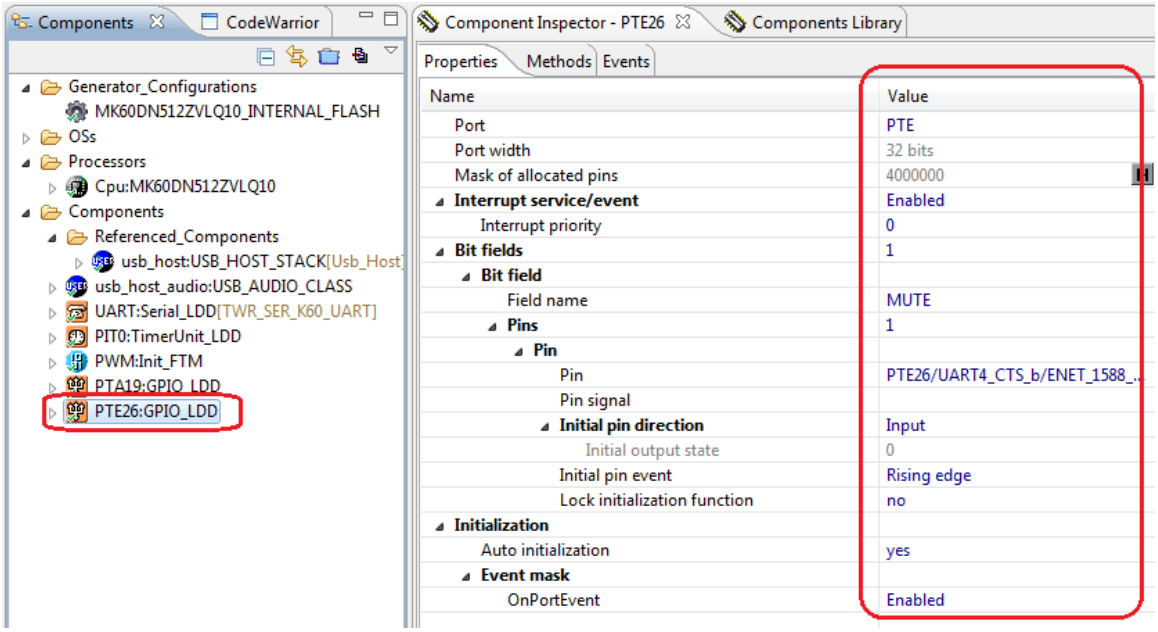


Figure 6-39 PTE26-"MUTE" button Properties settings

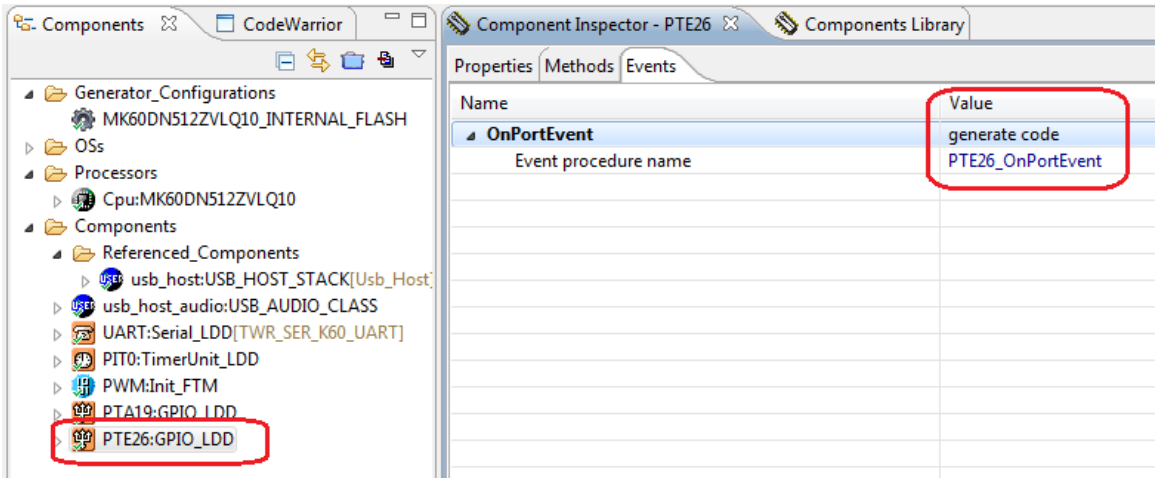


Figure 6-40 PTE26-"MUTE" button Events settings

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.

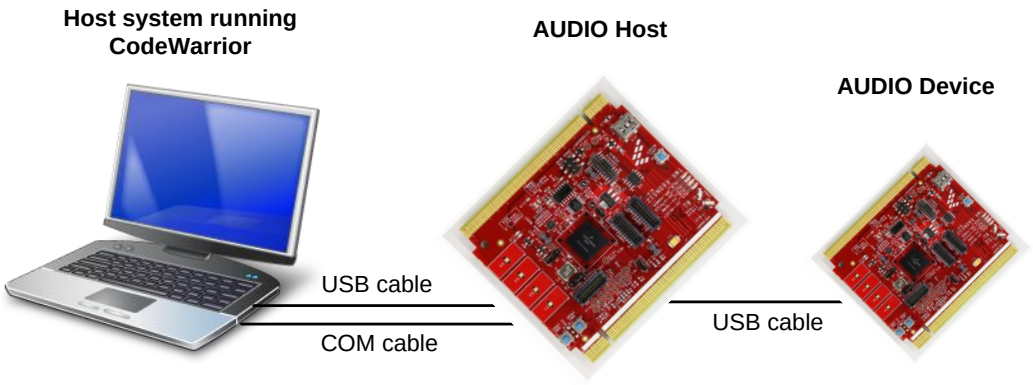


Figure 6-41 Hardware setup

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.

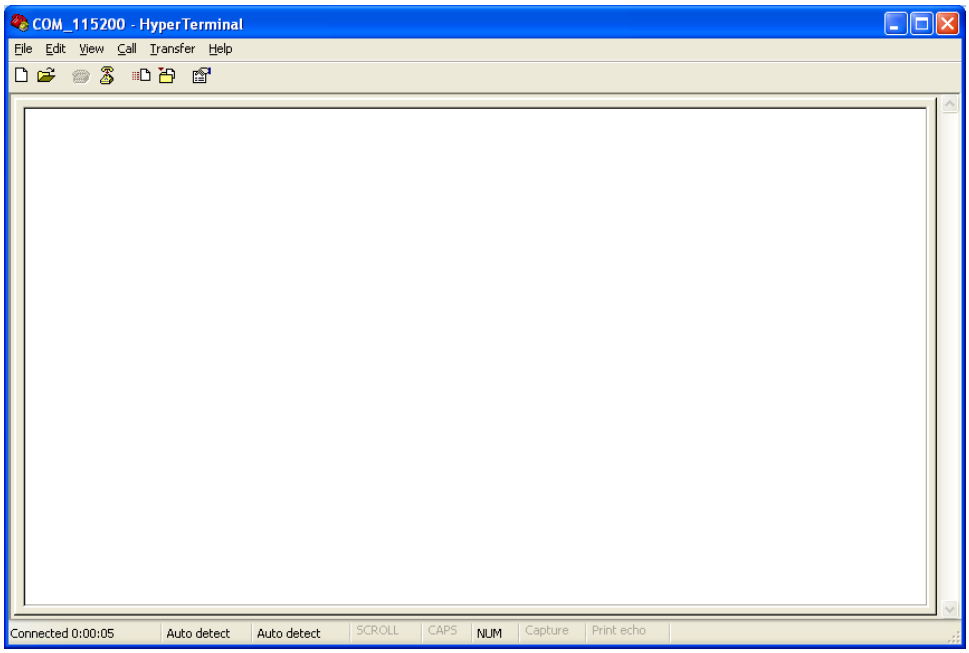


Figure 6-42 Hyper Terminal

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:

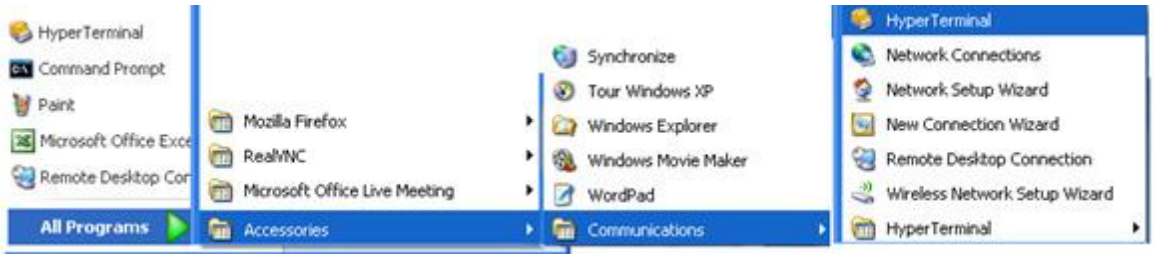


Figure 6-43 Launch HyperTerminal application

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

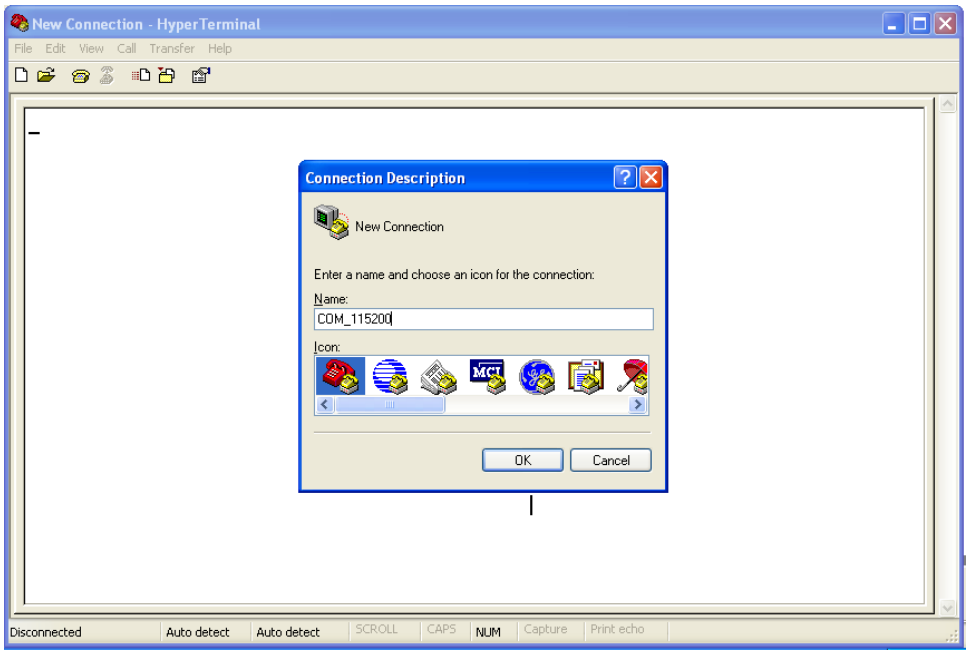


Figure 6-44 HyperTerminal startup

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

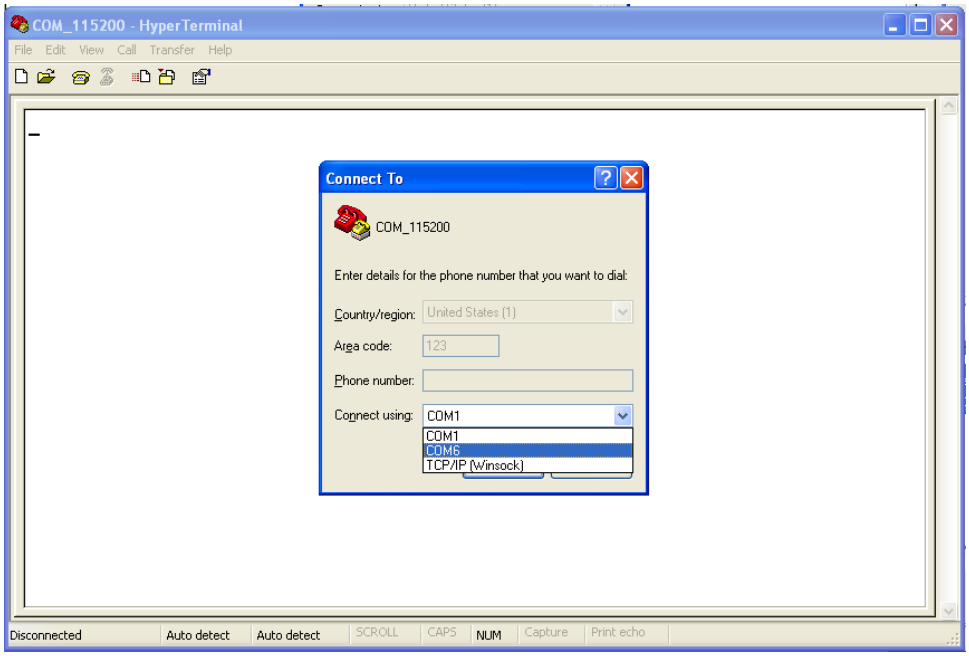


Figure 6-45 Connect using COM port

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

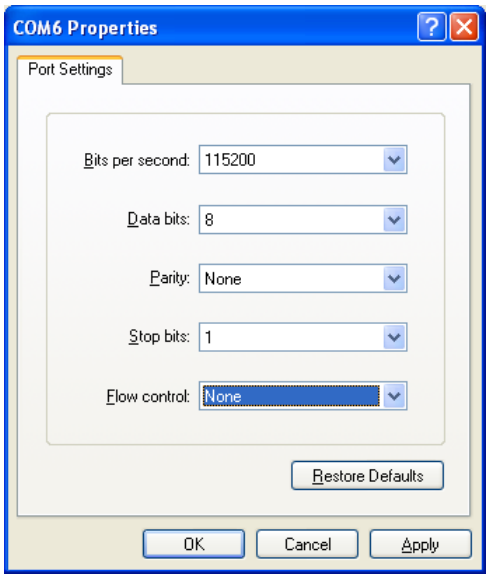


Figure 6-46 COM properties

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

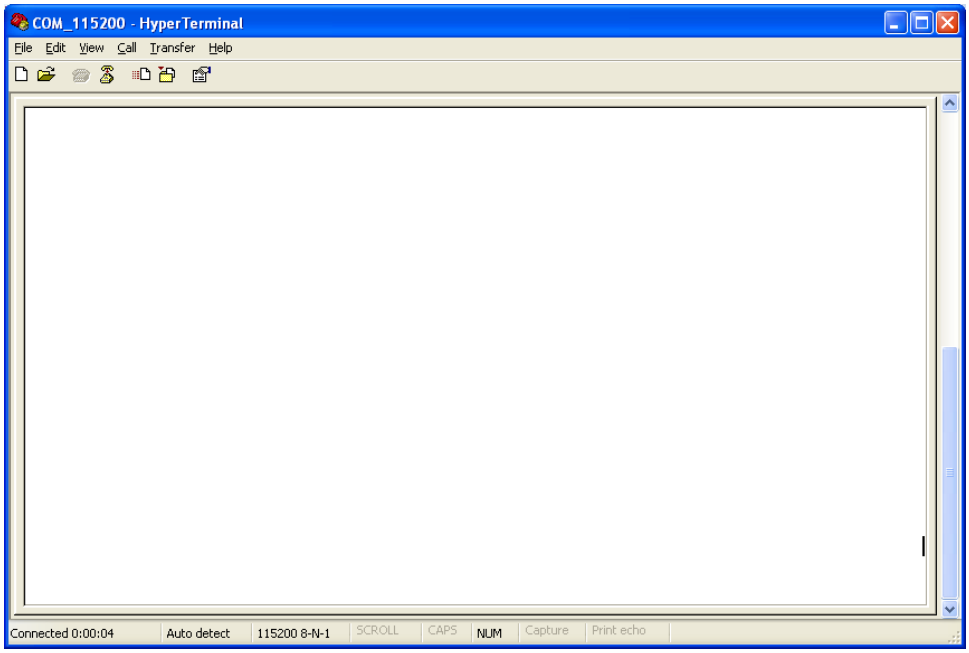


Figure 6-47 Hyper Terminal

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

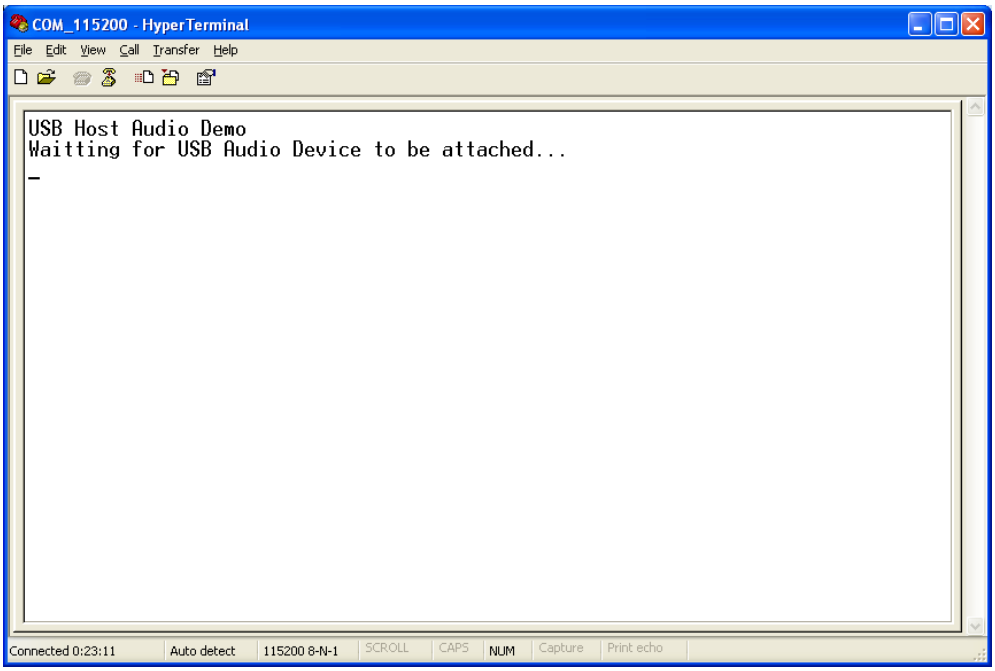


Figure 6-48 USB Host waiting for audio device attachment event

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

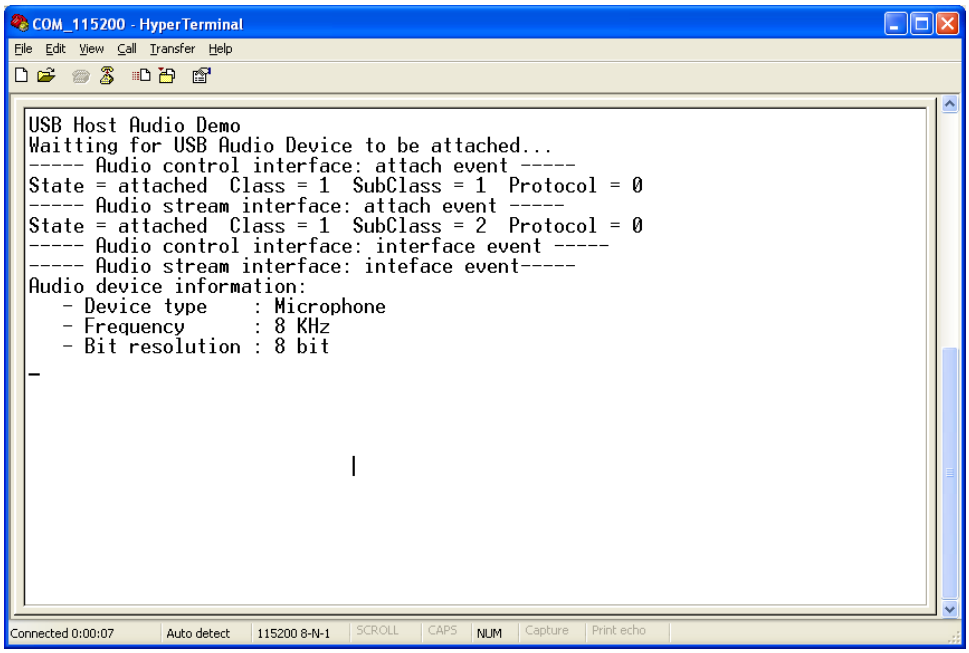


Figure 6-49 Attached device is Speaker type

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

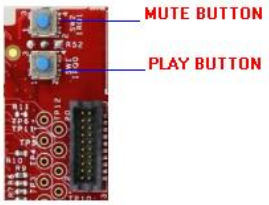


Figure 6-50 AUDIO functional button

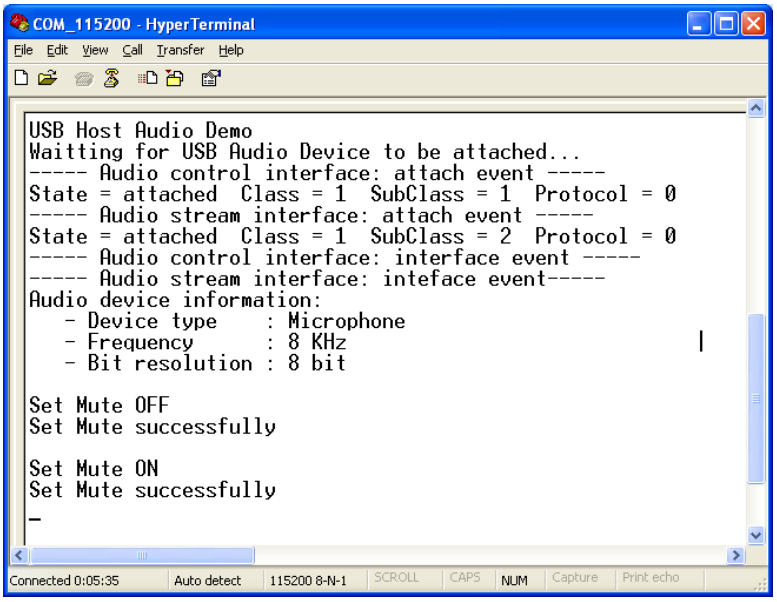


Figure 6-51 Set Mute ON/OFF

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

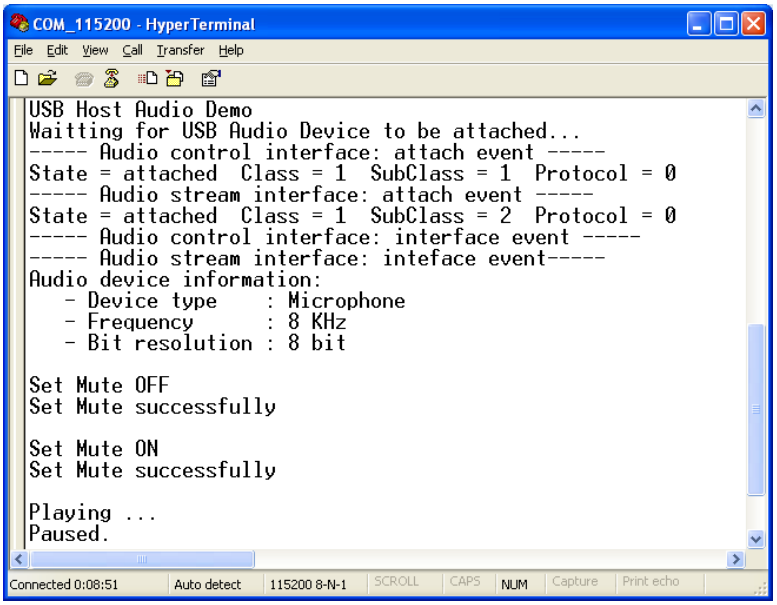


Figure 6-52 Start/Stop transferring audio data

Step8. Unplug the Audio Device.

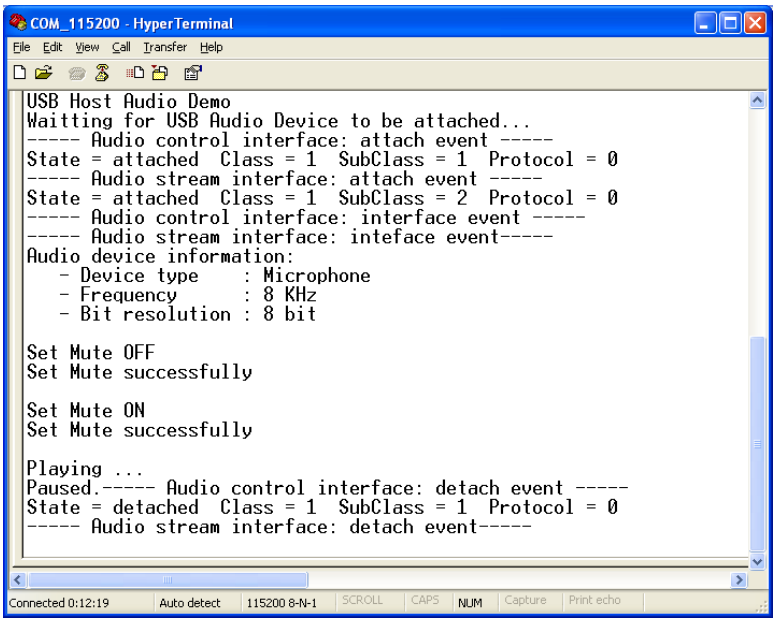


Figure 6-53 Audio Device detached

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.

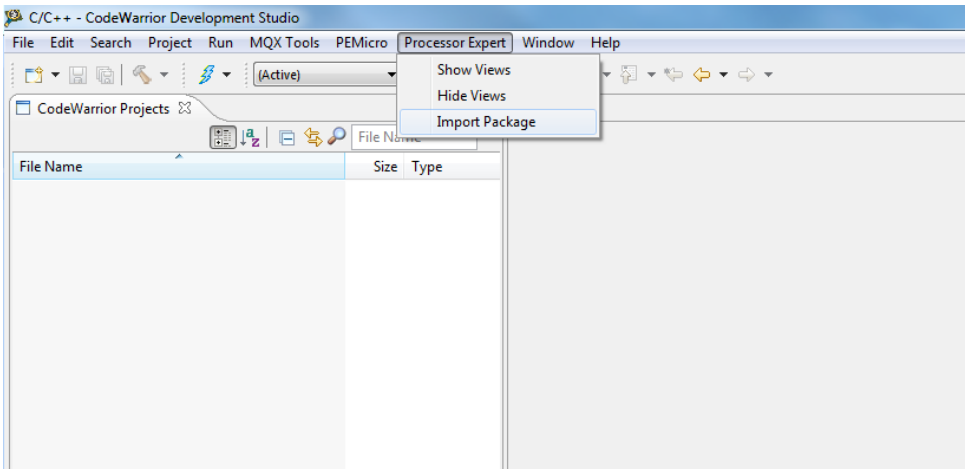


Figure 7-1 Processor Expert - Import package

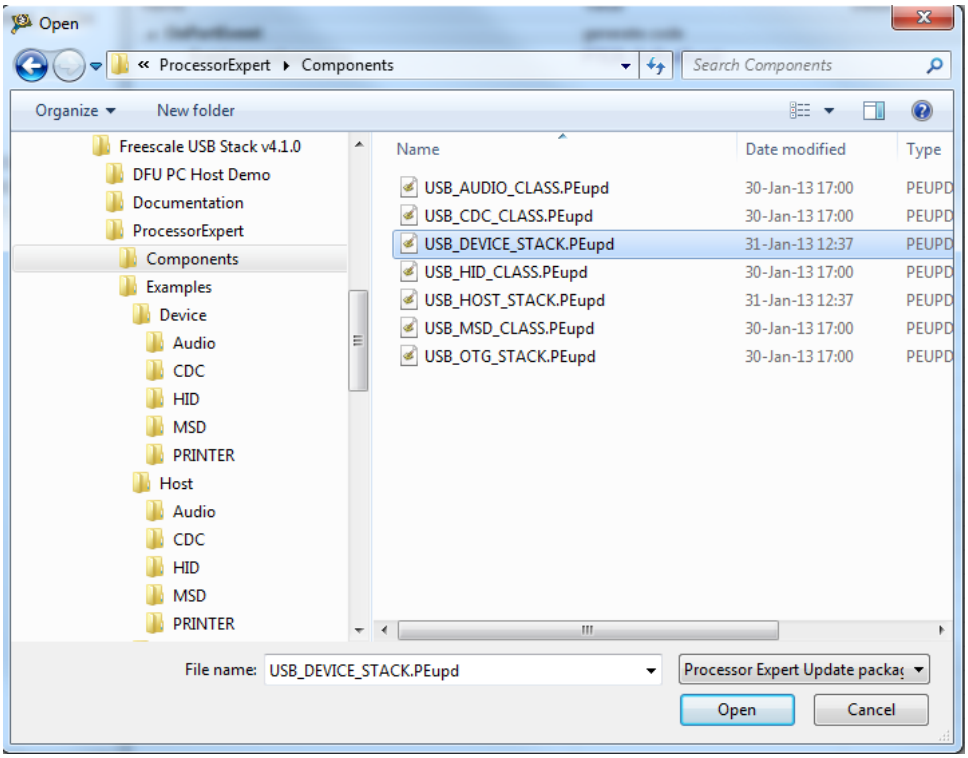


Figure 7-2 Select USB Device stack package

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.

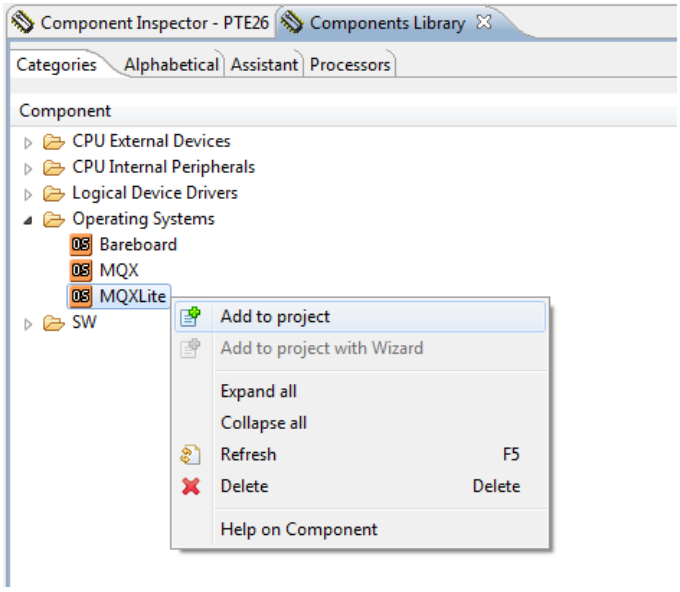


Figure 7-3 Add the MQX Lite component

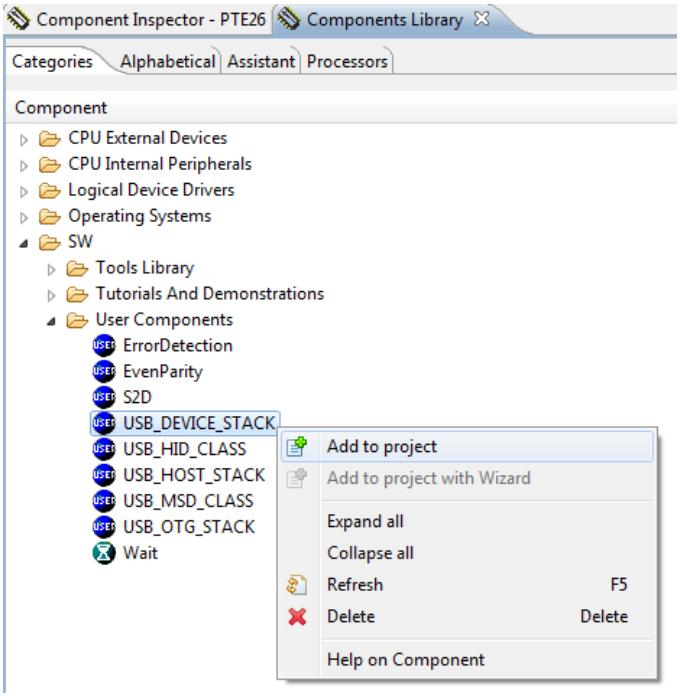


Figure 7-4 Add the USB Device stack component

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:
  - CPU type: MKL25Z128VLK4
  - Clock source: External crystal
  - Clock frequency: 8.0MHz
  - MCG mode: PEE
  - MCG output: 96MHz
  - PLL output: 96MHz

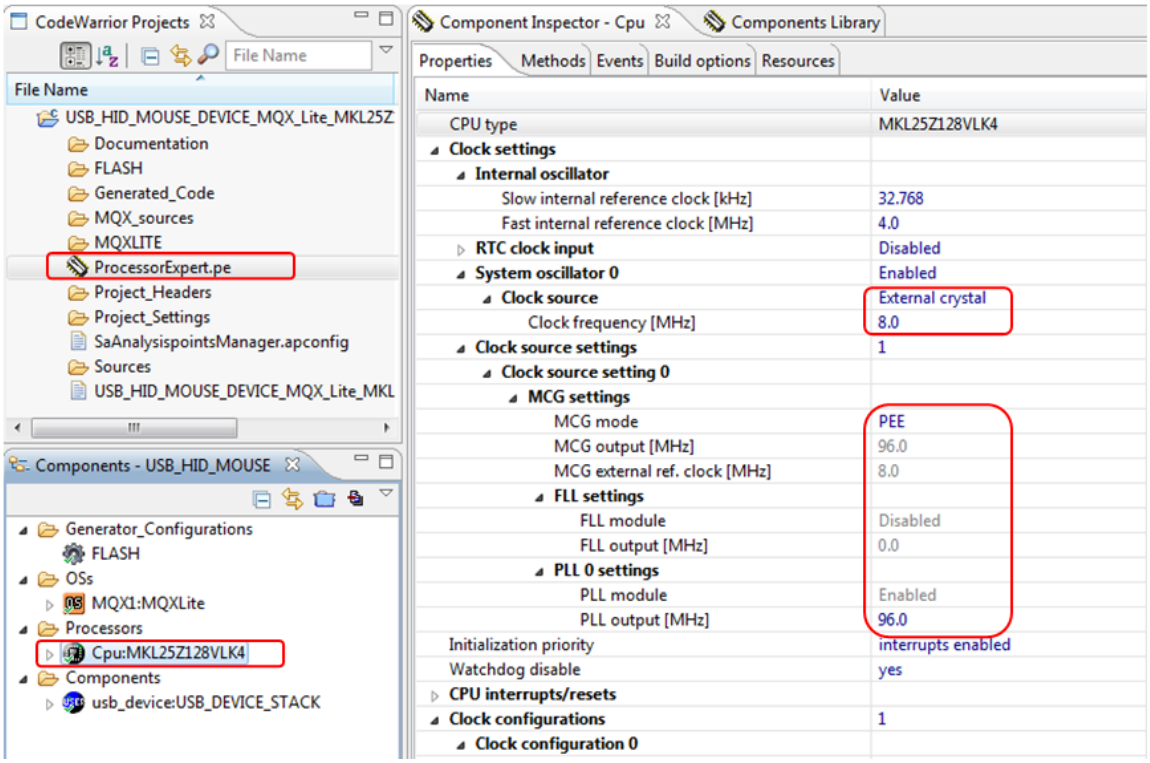


Figure 7-5 CPU Clock settings

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

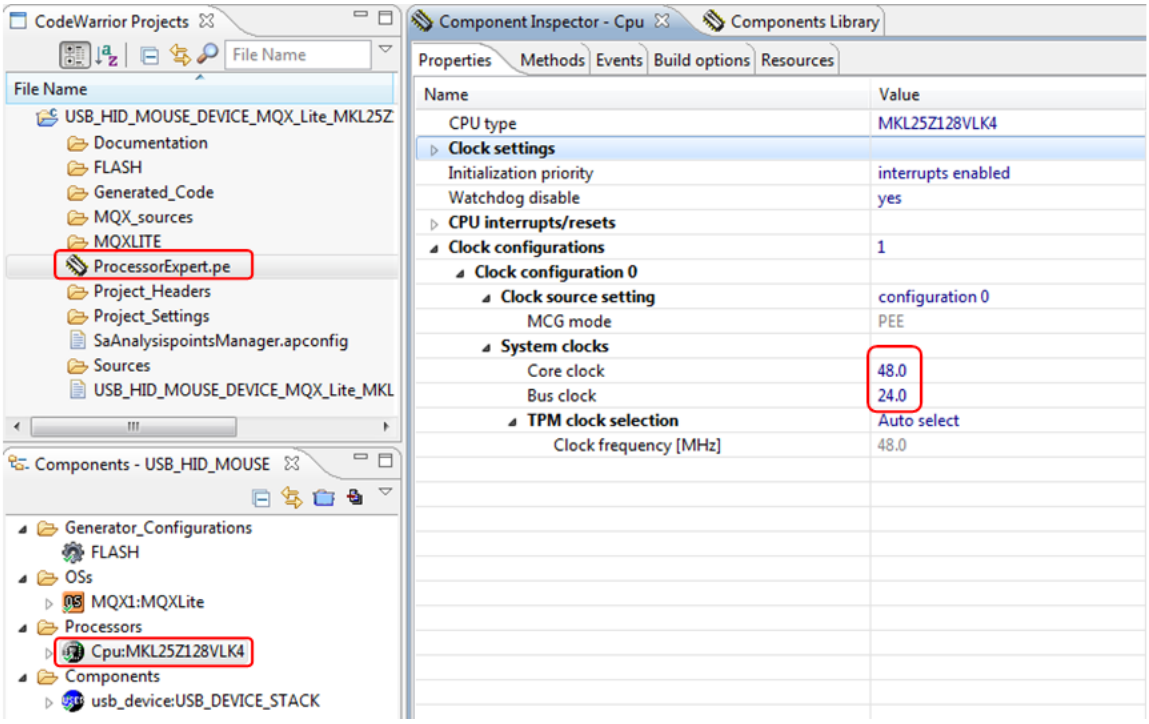


Figure 7-6 CPU Clock Configurations

Step2. Configure the MQX Lite component:

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

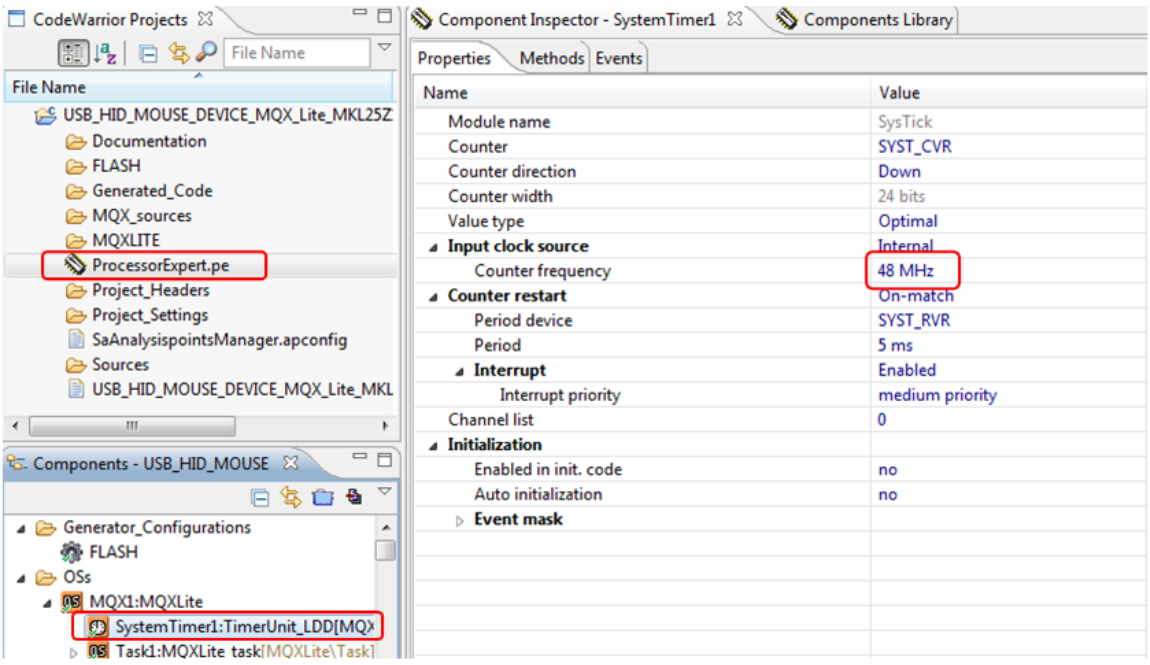


Figure 7-7 System Timer Configurations

- Configure the Tasks:

  - Name: usb_task
  - Entry point function: usb_task

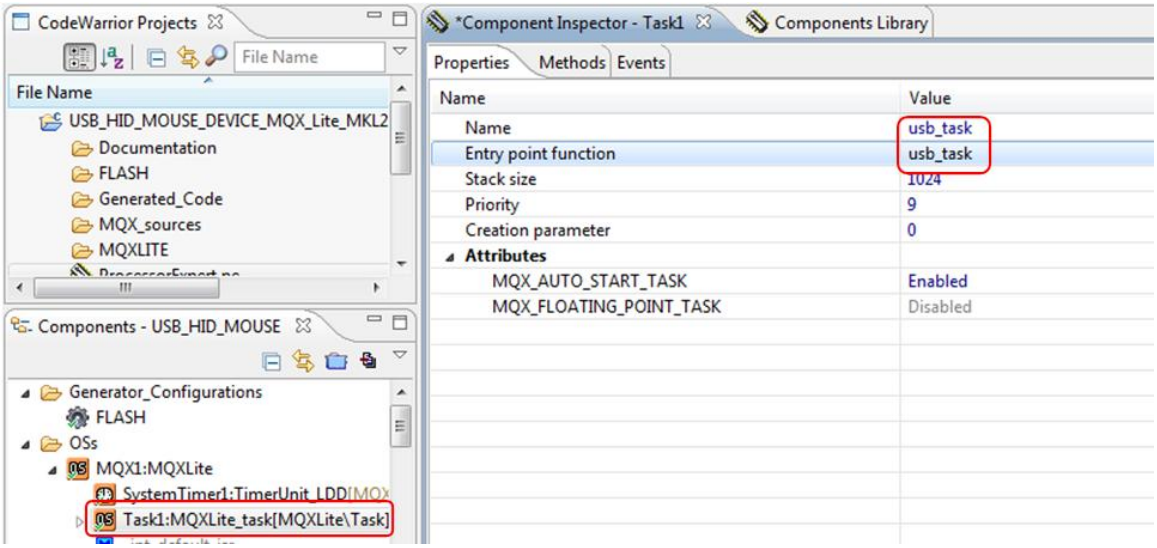


Figure 7-8 Task Configuration

Step3. Configure the USB device stack component:

- Configure the Device descriptor:

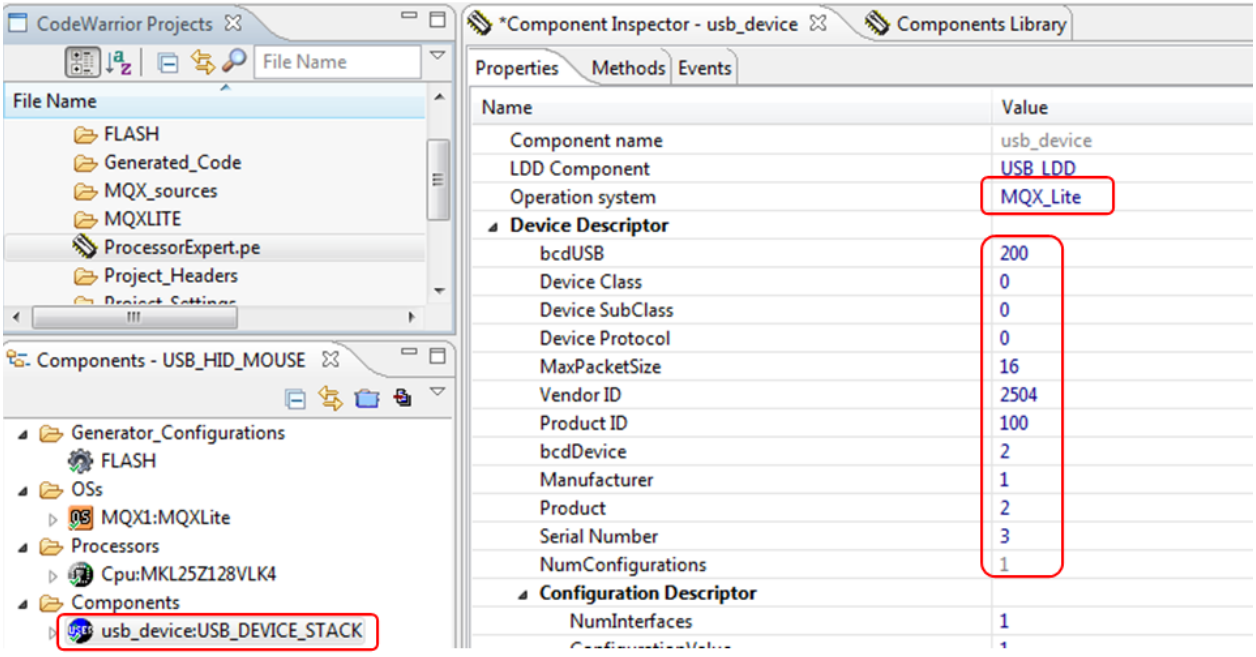


Figure 7-9 Device Descriptor configuration

- Configure the Configuration Descriptor:

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

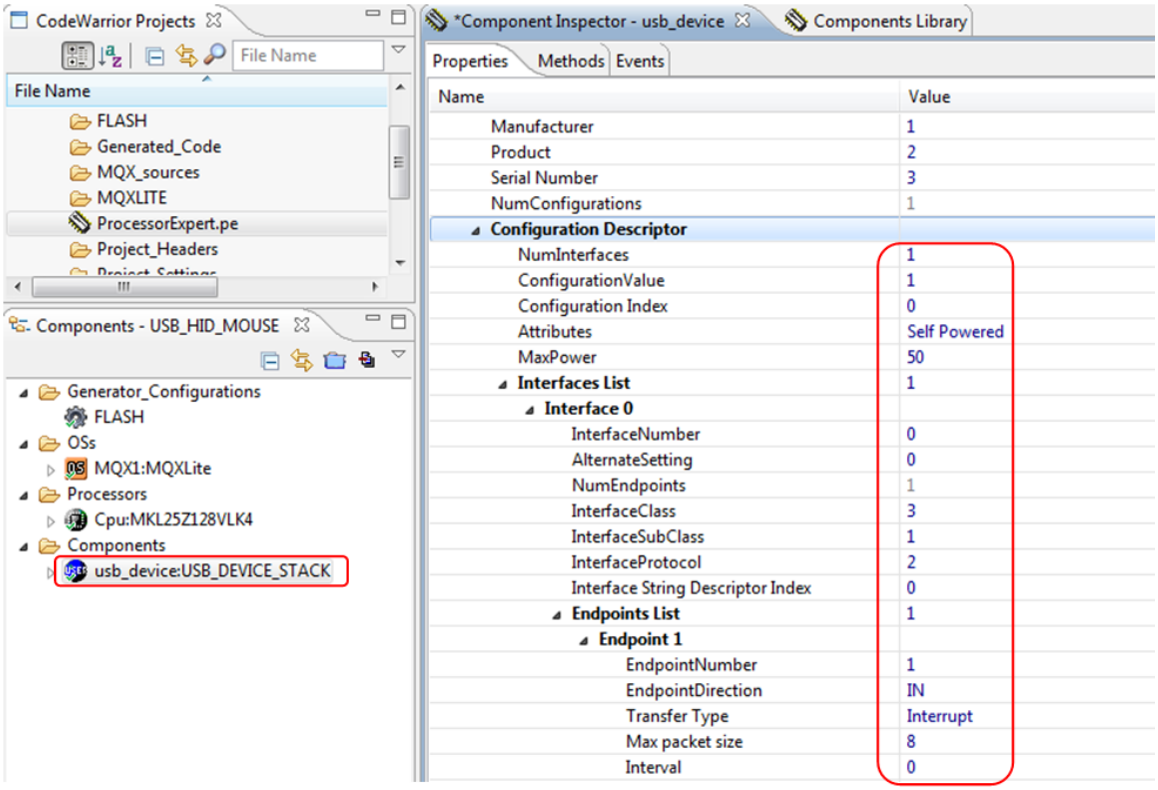


Figure 7-10 Configuration Descriptor configuration

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

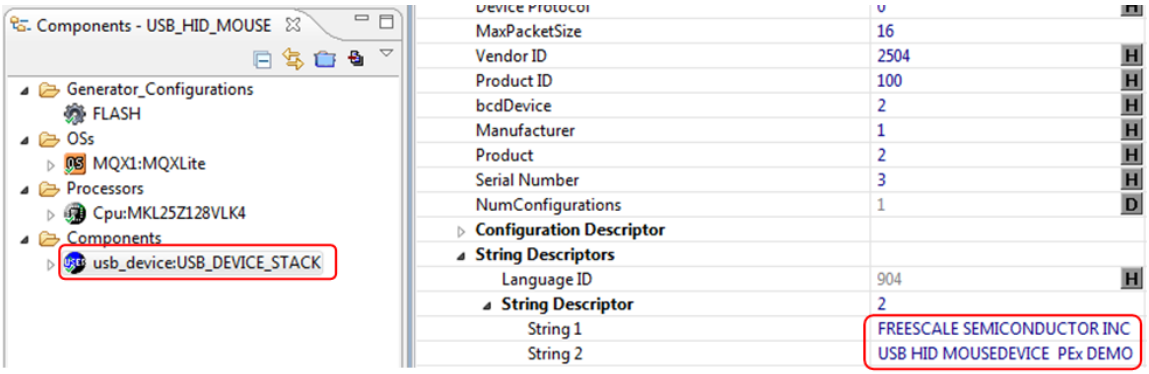


Figure 7-11 String Descriptor configuration

- Configure the USB_LDD :
    - Enable the EP1, enable Interrupt IN transfer, Max. packet size 8, Max. queue size 1.

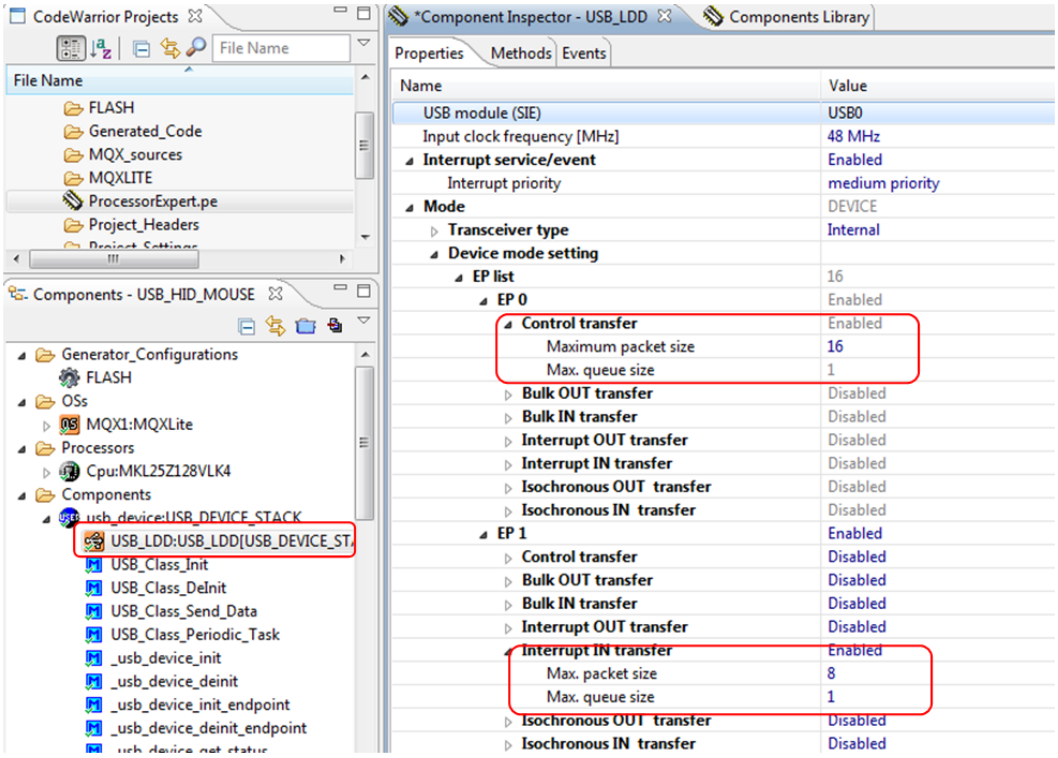


Figure 7-12 USB_LLD configuration

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.

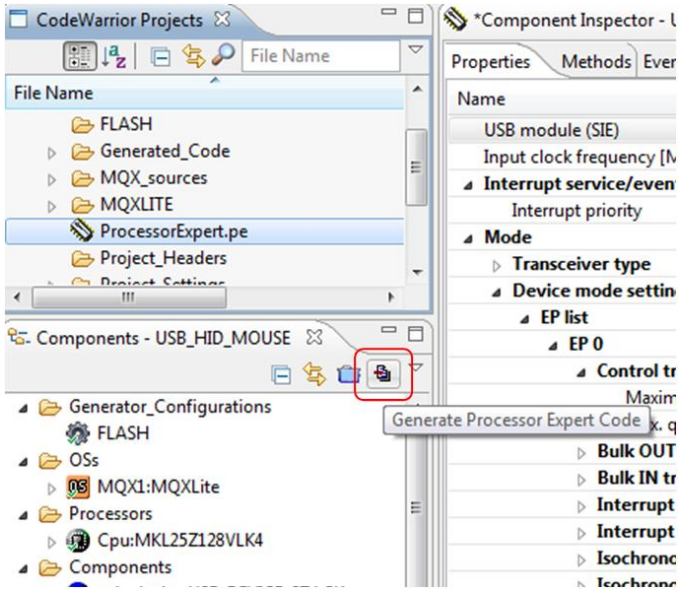


Figure 7-13 Generate Processor Expert Code

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

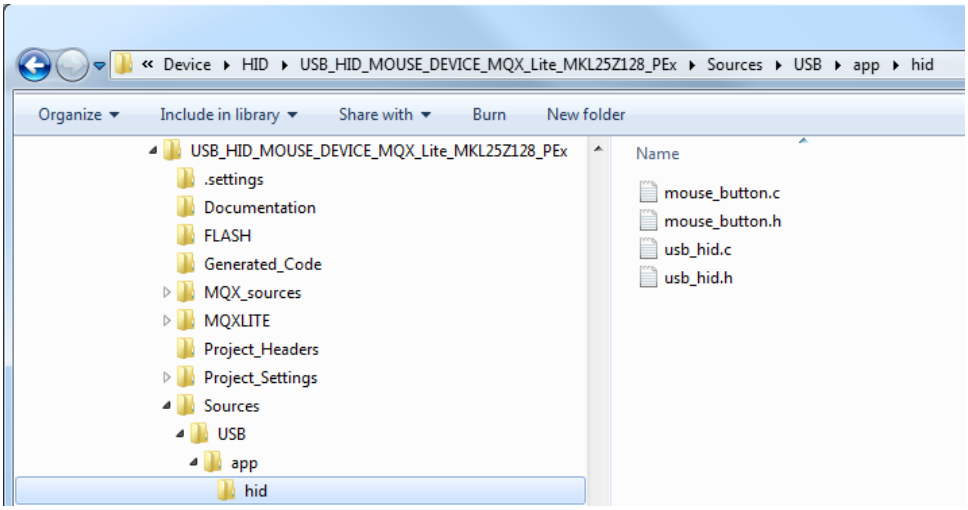


Figure 7-14 The USB Mouse Device and the HID Class code

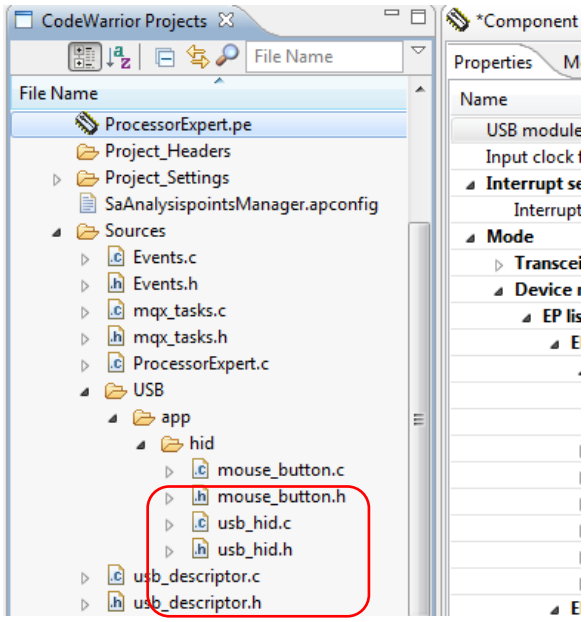


Figure 7-15 Add the USB Mouse Device and the HID Class to the project

- Edit the "***usb_descriptor.c***":
  - 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***":
    - 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*/
#undef    CONFIG_DESC_SIZE
#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:

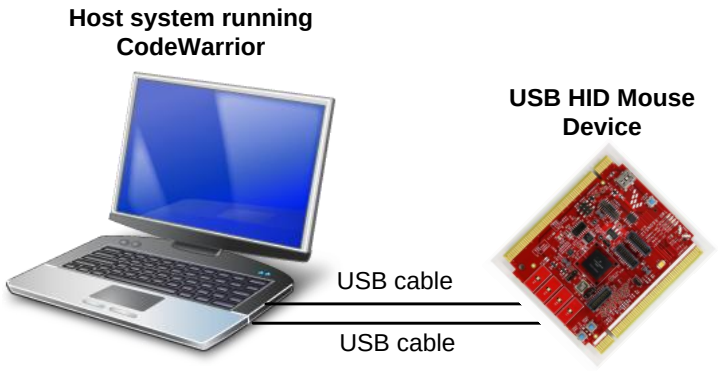


Figure 7-16 HID mouse device demo setup

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.