Processor Expert Kinetis SDK USB Stack Integration

1 2

3

4

1 Introduction

Processor Expert helps in simplifying the complex task of writing USB applications. Processor Expert supports set of components that provides properties to configure USB and layered access corresponding to the USB specifications including the class components. Each components handles the corresponding static files from Kinetis SDK, requested compiler settings, and generates code according to the selected properties. This document is lists the steps to quickly start writing USB applications in Processor Expert.

Processor Expert Software is a development system to create, configure, optimize, migrate, and deliver software components that generate source code for Freescale silicon.

For more information on Processor Expert, see www.freescale.com/processorexpert.

The Kinetis software development kit (SDK) is an extensive suite of robust peripheral drivers, stacks, middleware and example applications designed to simplify and accelerate application development on any Kinetis MCU.

For more information on Kinetis SDK, see www.freescale.com/ksdk.

The Kinetis SDK source structure contains the complete API to access Freescale microcontrollers and the USB stack (HOST/DEVICE/OTG modes) implementation source files. The Kinetis SDK USB stack is divided in layers and for these

© 2015 Freescale Semiconductor, Inc.

Contents

Introd	luction		1
PEx U	JSB stacl	k structure	2
PEx S	SDK USE	B layers description	3
3.1		_device_msd_class nent	3
3.2		_device_hid_class nent	6
3.3	fsl_usb_	_descriptors component	10
3.4	fsl_usb_	_framework component	14
3.5	fsl_usb_	_ehci_hal component	19
3.6	fsl_usb_	_khci_hal component	19
Creat	ing comn	non PEx USB project	20
4.1	Creating	g PEx USB project	27
	4.1.1	USB mass storage project	
	4.1.2	USB HID project.	30



PEx USB stack structure

layers the Processor Expert (PEx) components are created. The main function of each PEx USB component is to add source file code to the project (linked or standalone mode) and create the USB stack configuration files.

Processor Expert is fully integrated into Kinetis Design Studio (KDS). The Kinetis Design Studio IDE is a complimentary integrated development environment for Kinetis MCUs that enables robust editing, compiling, and debugging of your designs. Based on a free, open-source software including Eclipse, GNU Compiler Collection (GCC), GNU Debugger (GDB), and others.

For more information on Kinetis Design Studio, see www.freescale.com/kds.

Kinetis Design Studio is released only as a base product and does not contain SDK support by default. It is necessary to add SDK support by installing corresponding service pack (Eclipse update). The Eclipse udpate is available in each SDK in tools directory.

2 PEx USB stack structure

The following figure illustrates the Processor Expert USB stack structure.

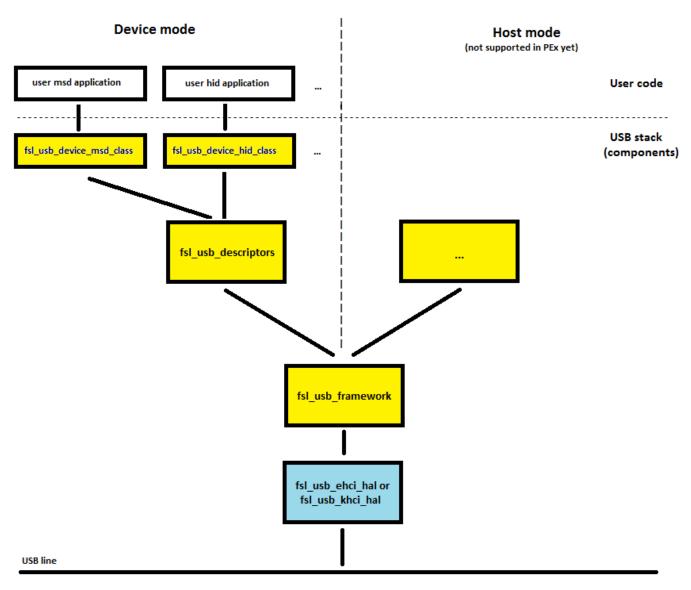


Figure 1. USB stack structure

3 PEx SDK USB layers description

3.1 fsl_usb_device_msd_class component

The component allows:

• USB device mass storage. **Subclass** and **Protocol** code configuration. Mass storage device class driver supports only SCSI transparent command subclas code and BBB (bulk only transport) protocol code.

Processor Expert Kinetis SDK USB Stack Integration, Rev. 0, 07/2015

📎 *Component Inspector - msd1 🙁 🚫 Cor	mponents Library Basic 🛛 Advanced 🎽 🎽 🖤	
Properties Methods Events		
type filter text All MSD Class SDK specific settings Initialization USB RAM disk demo Callback user parameters	Component name msd1 fsl_usb_descriptor component link usbDsc1 + > Update fsl_usb_descriptor component MSD Class SDK specific settings Subclass code 0x06 SCSI transparent command set + Protocol code Protocol code 0x50 BBB (bulk only transport) +	*

Figure 2. MSD Subclass and Protocol code configuration

• Creating MSD USB stack configuration structures and variables.

📎 *Component Inspector - msd1 🛛 🚫	Comp	onents Library Basic Advanced 🎽 🎽 🗖
Properties Methods Events		
Properties Methods Events type filter text All MSD Class SDK specific settings Initialization USB RAM disk demo Callback user parameters		Component name msd1 fsl_usb_descriptor component link usbDsc1 • > Update fsl_usb_descriptor component MSD Class SDK specific settings Initialization MSD device handler name msd1_MsdHandle MSD configuration structure name msd1_MsdConfigStructure USB RAM disk demo Sectors count 64 RAM disk sector size [B] 512 Callback user parameters
		application_callback arg NULL vendor_req_callback_arg NULL class_specific_callback arg NULL
		Auto initialization

Figure 3. Properties for MSD class initialization

• Creating C module with USB MSD callbacks. The default name of C module is *msd1_msd.c* and is stored in Sources folder. This module contains USB RAM disk demo code which is activated by the **USB RAM disk demo** property. You can set the USB RAM disk parameters such as sector size and count in the component properties.

Processor Expert Kinetis SDK USB Stack Integration, Rev. 0, 07/2015

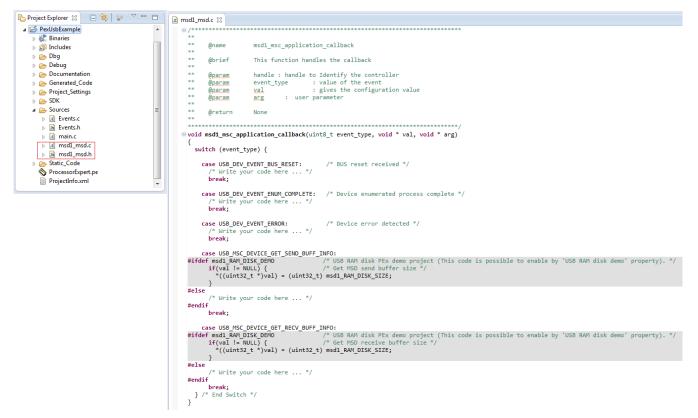


Figure 4. MSD class callbacks API (with RAM disk demo code) generated by Processor Expert

• Accessing to the MSD API functions (API functions are taken over by the usb_class_msc.h file).

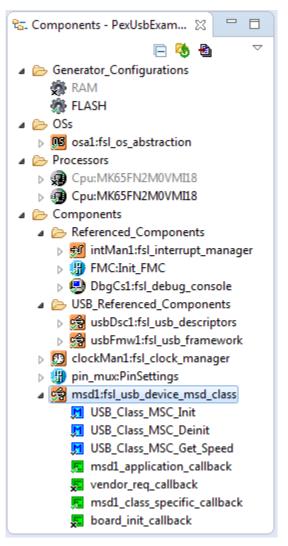


Figure 5. MSD class API and callback functions

- Adding MSD class requirements to *fsl_usb_descriptor* component such as class type, desired endpoints, and so on.
- Adding Kinetis SDK USB stack MSD driver files to the project.

3.2 fsl_usb_device_hid_class component

The component allows:

• USB device HID. Subclass and Protocol code configuration.

📎 *Component Inspector - hid1 🛛 🚫 Com	: Library	Basic [Advanced 🎽 🎽 🗖
Properties Methods Events			
type filter text All Generated code common settings HID class setting HID descriptor Class descriptor list Report descriptor templates Report descriptor Item List SDK specific settings Initialization	omponent name hid I_usb_descriptor component link usb pdate fsl_usb_descriptor component Generated code common HID class setting Subclass code 0x00 No Subclass + Protocol code 0x02 Mouse +	bDscl + >	SDK specific settings
· · · · · · · · · · · · · · · · · · ·			Þ

Figure 6. HID Subclass and Protocol code configuration

• HID report descriptor definition. *fsl_usb_device_hid_class* component contains 3 predefined report descriptor template: Standard mouse, keyboard, or thermometer.

📎 *Component Inspector - hid1 🛛 🚫 Con	ponents Library B	asic 🛛 Advanced 🎽 🌣 🗖 🗖
Properties Methods Events		
type filter text All Generated code common settings HID class setting HID descriptor Class descriptor list Report descriptor templates Report descriptor Item List SDK specific settings Initialization	Component name hidi fsl_usb_descriptor component link usbDscl + > Update fsl_usb_descriptor component Generated code common HID class setting HID descriptor Report descriptor Optional class de Subclass code 0x00 No Subclass + Protocol code 0x02 Mouse +	escri SDK specific settings
۲		4

Figure 7. HID report descriptor definition

• Creating HID USB stack configuration structures and variables.

PEx SDK USB layers description Basic Advanced 🎬 🗘 🔿 👘 $\nabla \Box$ 🗞 *Component Inspector - hid1 🔀 🗞 Components Library Properties Methods Events type filter text Initialization ⊿ All hid1_HidHandle HID device handler name Generated code common settings hid1_HidConfigStructure HID configuration structure name HID class setting Auto initialization ▲ HID descriptor Class descriptor list USB HID demo mode Callback user parameters Report descriptor Report descriptor templates USB HID demo mode Mouse $\overline{\mathbf{v}}$ Report descriptor Item List ▲ Initialization < < Callback user parameters Application callback Vendor request callback Class specific callback Board initialization callback

Figure 8. Properties for HID class initialization

• Creating C module. The default name of C module is *hid1_hid.c* and is specified by the **Event module name** property in the **Events** tab. The module is stored in the Sources folder. This module contains USB HID callbacks and USB HID demo code, which can be actovated by the **USB HID demo mode** property. The USB HID component contains **Mouse,Keyboard**, and **Thermometer** demos type. Selecting a particular demo type (mouse/keyboard/thermometer) automatically changes the HID report descriptor and the HID protocol code (mouse/keyboard).

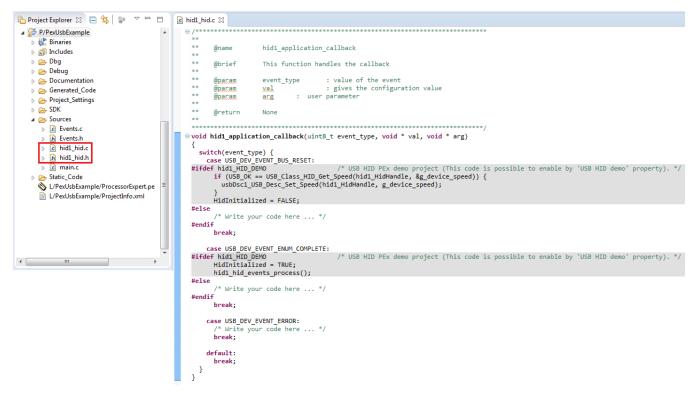


Figure 9. HID class callbacks API (with HID demo code) generated by Processor Expert

• Accessing the HID class API functions (API functions are taken over usb_class_hid.h file).

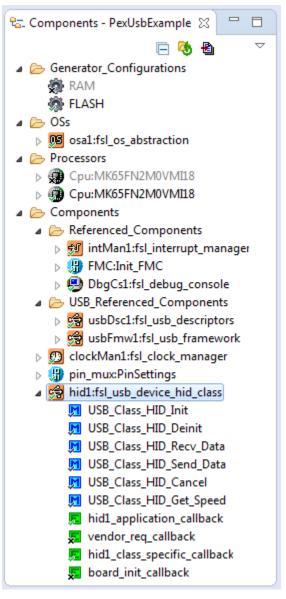


Figure 10. HID class API and callback functions

- Adding HID class requirements to *fsl_usb_descriptor* component such as class type, desired endpoint, and so on. *fsl_usb_device_hid_class* component requires only one input-endpoint (inputput-endpoint by default). If application requires receive data from HOST device (output-endpoint), you must manually add the output-endpoint in *fsl_usb_descriptors* component.
- Adding Kinetis SDK USB stack HID driver files to the project.

3.3 fsl_usb_descriptors component

fsl_usb_descriptors component is used for DEVICE USB mode.

The basic function of the *fsl_usb_descriptors* component is to collect requirements. For example, USB device class type, pipes numbers and type of all *fsl_usb_device_XXX_class* components, define enpoint sizes and configure using SDK USB class drivers. On the basis of the collected information, *fsl_usb_descriptors* component creates the USB standard and class descriptors structures and creates the standard USB function. For example, GetDescriptors, SetInterface, and SetConfiguration.

operties Methods Events		
ame	Value	Details
Component name	usbDsc1	
Lower level component link	usbFmw1	
Generated code common settings		
Mark all descriptors as const	yes	
Supported languages	1	
Language 0	0x0409 English (United States)	
External power source	yes	
Common device settings		
USB revision	USB 2.0	
Vendor ID	0000 H	
Product ID	0000 H	
Device release number	01.00	
> Manufacturer description	Enabled	Freescale Processor Expert
Product description	Enabled	Processor Expert USB Device comp
> Device's serial number	Enabled	123456789ABCDEF
✓ Device spend manufact	Full speed	
Device description	P	
Class code	0x00 Class information at interface	
Subclass code	0 D	
Protocol code	0 D	
▲ EPO settings		
Max packet size	64	
	1	
Configuration 1	-	
Configuration 1	Full_Speed_Configuration_1	
Total length	34 D	
Configuration description	Enabled	Configuration 1
Power characteristics		Configuration 1
	self powered	
Maximum power consumption	0 mA	
Remote wake-up	yes	
▲ Class list	1	
▲ Class 0	L.U.M	
Class component name	hid1	
Class user name		
Implementation specific settings	SDK	
▲ Interface list	1	
⊿ Interface 0		
⊿ Alternate setting list	1	
⊿ Alternate setting 0		
Interface user name		
Default request handler name		
Class code	0x03 HID	
Subclass code	0x00 No Subclass	
Protocol code	0x02 Mouse	
Alternate setting description	Disabled	
▲ Class descriptors	0x03 HID	
▲ HID descriptor		
Hid descriptor name	FS_Cfg_1_Int_0_AltSet_0_HidDescri	
HID Class specification relea		
Country code	0x00 Not Supported	
⊿ Class descriptor list	1	
⊿ HID class descriptor 0		
Descriptor type	HID_REPORT	
Descriptor size	50 D	
Descriptor name	hid1_MouseReportDescriptor	
⊿ Pipe list	1	
⊿ Pipe 0	Interrupt IN	FS Interrupt EP1 IN, 8 KB/s
Pipe user name	hid1_PipeIn	
Default request handler nar		
Endpoint number	1	
Maximum packet size	8 D	
Polling interval	1 ms	
ZLT	yes	
SDK specific settings		
▲ Class drivers configuration		
▲ HID class driver configuration	Enabled	
Max. human interface device number	1 D	
Max. class endpoint number	2 D	
> Data transfer queuing	Disabled	
MSD class driver configuration	Disabled	
Composite driver configuration	Disabled	

Figure 11. fsl_usb_descriptors component

Component allows:

• Creating C module. The default name of C module is *usbDsc1.c*, which is stored in the Generated_Code folder. The *usbDsc1.c* file contains description of all USB device class component such as Device/Configuration/Strings descriptors, structures which describe used endpoints and class types, and standard USB device function like GetDescriptors, SetInterface, SetConfiguration, and so on.

🎦 Project Explorer 🙁 🖃 🔄 🐨 🗖 🗖	🖻 usbDsc1.c 🔀	
⊿ 2 PexUsbExample		
Binaries	const uint8_t Full_Speed_Configurat	tion_1[]={
) 🔊 Includes	/**************************************	***************************************
Dbg	/* Configuration 1 Descriptor	*/
Debug	0x09.	/* Descriptor size: 9 bytes */
Decoup Documentation	USB CONFIGURATION DESCRIPTOR,	/* Descriptor type: Configuration descriptor */
	0x22,0x00,	/* Total length of data for this configuration: 34 bytes */
▲ Generated_Code	0x01,	/* No of interfaces supported by this configuration */
D DK	0x01,	/* Designator value for this configuration */
⊳ h board.h	0x04,	/* Configuration string descriptor index */
⊳ 💽 Cpu.c	0×E0,	/* Power source: self powered, remote wake-up: yes */
⊳ 🖻 Cpu.h	0x00,	/* Max. power consumption: 0 mA */
DbgCs1.c	/* hid1: Interface 0 Alternate se	,
b DbgCs1.h	/*************************************	**************************************
FMC_Config.h	0x09.	/* Descriptor size: 9 bytes */
b FMC.h	USB INTERFACE DESCRIPTOR,	/* Descriptor type: INTERFACE descriptor */
hardware_init.c	0x00,	/* Interface number: 0 */
hid1.c	0x00,	/* Alternative setting number: 0 */
hid1.h	0×01,	/* Number of EPs(excluding EP0): 1 */
▶ h Init_Config.h	0×03, 0×00.	/* Class code: 0x03 HID */ /* Subclass code: 0x00 No Subclass */
▷ ic osal.c	0x00, 0x02,	/* Subclass code: 0x00 No Subclass */ /* Protocol code: 0x02 Mouse */
b h osal.h	0x02, 0x00,	/* String descriptor index */
Fin Osar.n Fin Osar.n Fin Osar.n Fin Osar.n Fin Osar.n	/**************************************	***************************************
PE_ProjectInfo.h	/* HID Descriptor	*/
	/**************************************	***************************************
▶ i pin_init.c	0x09,	/* Descriptor size: 9 bytes */
▶ h pin_init.h	USB_HID_DESCRIPTOR,	/* Descriptor type: HID descriptor */
▷ is usbDsc1.c	0×11,0×01, 0×00,	<pre>/* HID specification release number: 1.11 */ /* Country code: 0x00 Not Supported bytes */</pre>
▶ h usbDsc1.h	0x00, 0x01,	/* Number of class descriptors : 1 */
▷ usbFmw1.c	USB_HID_REPORT_DESCRIPTOR,	/* Descriptor type: HID_REPORT descriptor */
b usbFmw1.h	0x32,0x00,	/* Descriptor size: 0x32 */
misra_chck.bat	/**************************************	***************************************
pe-misra.Int	/* hid1: Endpoint FS Interrupt E	
PexUsbExample.Int		***************************************
Project_Settings	0x07, USB ENDPOINT DESCRIPTOR,	/* Descriptor size: 7 bytes */ /* Descriptor type: ENDPOINT descriptor */
D 🕞 SDK	0x81,	/* Address: 1 IN */
Sources	0x03,	/* Transfer type: Interrupt */
Static Code	0x08,0x00,	/* Max. packet size: 8 byte(s) */
N ProcessorExpert.pe	0x01	/* Polling interval: 1 ms */
ProjectInfo.xml	};	
E rigecanosali		

Figure 12. Content of usbDsc1.c file, description of USB device

• Accessing the Composite class API functions (API functions are taken over *usb_class_composite.h* file). The Composite class API is enabled automatically when more that one fsl_usb_device_XXX_class components are available in project (composite USB device mode).

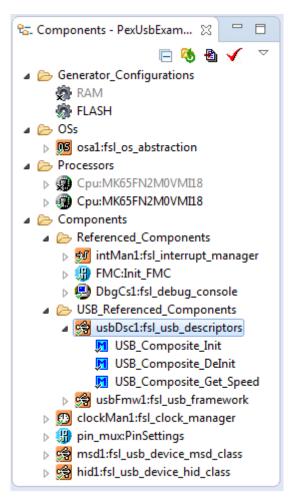


Figure 13. Composite class API functions

- Adding requirements to *fsl_usb_framework* component such as USB mode endpoint type, and count.
- Adding Kinetis SDK USB stack "Composite" driver files to the project (for USB composite mode), create kinetis SDK class configuration files.

3.4 fsl_usb_framework component

fsl_usb_framework component covers bottom layer of the USB interface that transmits and receives packets. *fsl_usb_framework* component USB low/full speed (KHCI) and USB low/full/high speed (EHCI) module.

fsl_usb_framework component supports three modes: DEVICE, HOST and OTG (USB OTG mode is not fully supported yet, OTG mode allows currently use DEVICE or HOST mode. HOST or DEVICE mode can be selected in runtime, according USB ID pin signal value, by usb_device/host_init() methods. According the selected USB mode, *fsl_usb_framework* component adds SDK USB stack source files to the project and generates USB stack configuration and BSP files. BSP files contains code for USB module timing, interrupt settings and PHY configuration.

fsl_usb_framework component is possible to use as standalone component in a project to create user defined USB stack (upper USB stack layers can be created by user).

Figure 14. USB DEVICE mode configuration

• Creating configuration structures and variables for USB DEVICE or HOST mode for standalone component use.

🗞 *Component Inspector - usbFmw1 🛛 🗞 Co	mpone	nts Library						Basic Advanced 🎬 🧲	3
Properties Methods Events									
	mponei	 ✓ Auto i USB hand ✓ Device User part External ☐ Host Driver in 	nitialization dler name usl rameter declaration of p mode information table on device driver Vendor ID 0x0000	Ni	JLL	Sub-Class code	Protocol 0xEE	Basic Advanced C	
		4							+

Figure 15. Properties for device mode initialization

• Accessing to usb_framework API functions (API functions are taken over "usb_device_stack_interface.h" file for DEVICE mode, for HOST mode over "usb_host_stack_interface.h" file)

⇒ OSs		℃ Components - PexUsbExample ※
Oss 0sa1:fsl_os_abstraction	<u></u>	RAM
Processors		FLASH
© Cpu:MK65FN2M0VMI18		
Cpu:MK65FN2M0VMI18		Oss Oss osal:fsl_os_abstraction Oss Os
		Processors
Components Feferenced_Components		Cpu:MK65FN2M0VMI18
		Cpu:MK65FN2M0VMI18
		Components
b B DbgCs1:fsl_debug_console Solution Sol		Referenced_Components
GockMan1:fsl_clock_manager Sector Secto		▷ IntMan1:fsl_interrupt_manager
B pin_mux:PinSettings		FMC:Init_FMC Desc1:fet_debug_engels
usbFmw1:fsl_usb_framework		DbgCs1:fs1_debug_console
July State Stat		D clockMan1:fsl_clock_manager
A being the second s		b in_mux:PinSettings
🚺 usb_device_init		✓ usbFmw1:fsl_usb_framework
🗾 usb_device_postinit		Just
🕅 usb_device_deinit		device methods
🗾 usb_device_recv_data		host methods
🚺 usb_device_send_data		🗾 usb_host_init
usb_device_cancel_transfer		🕅 usb_host_deinit
usb_device_register_service		usb_host_register_driver_info
🕅 usb_device_unregister_service	=	usb_host_register_unsupported_device_notify
👥 usb_device_assert_resume		💹 usb_host_open_dev_interface
🕅 usb_device_init_endpoint		💹 usb_host_close_dev_interface
💹 usb_device_deinit_endpoint		💹 usb_host_open_pipe
🕅 usb_device_stall_endpoint		💹 usb_host_close_pipe
📕 usb_device_unstall_endpoint		💹 usb_host_get_tr
📕 usb_device_register_application_notify		🕅 usb_host_release_tr
usb_device_register_vendor_class_request_notify		🕅 usb_host_send_data
💹 usb_device_register_desc_request_notify		💹 usb_host_send_setup
🕅 usb_device_get_status		🔀 usb_host_recv_data
🕅 usb_device_set_status		🔀 usb_host_cancel
> 🗁 host methods		💹 usb_host_bus_control
device callbacks		💹 usb_host_register_service
🁼 usbFmw1_usb_device_board_init		💹 usb_host_unregister_service
👼 usbFmw1_get_desc		📜 usb_host_register_ch9_callback
👼 usbFmw1_get_desc_interface		👧 usb_host_unregister_ch9_callback
👼 usbFmw1_set_desc_interface		💹 usb_host_open_dev_alternate_interface
👼 usbFmw1_set_configuration		💹 usb_host_get_dev_descriptor
👼 usbFmw1_get_desc_entity		💹 usb_host_dev_remove
b bost callbacks	*	💹 usb_host_get_host_handle
		device callbacks
		a 🗁 host callbacks
		嬺 usbFmw1_usb_host_board_init
		usbFmw1_ClassDriver0
		👳 class_driver_callback1
		class_driver_callback2
		class_driver_callback3
		class_driver_callback4
		class driver callback5

Figure 16. USB DEVICE / HOST mode API and callback functions

• MPU (Memory Protect Unit) and FMC (Flash Memory Controller) configuration (if MCU contains these devices). USB OTG controller (BUS master) can also access Flash memory (BUS slave) is through "crossbar switch" (see RM for selected CPU). The crossbar switch connects bus masters and bus slaves using a crossbar switch structure.

Processor Expert Kinetis SDK USB Stack Integration, Rev. 0, 07/2015

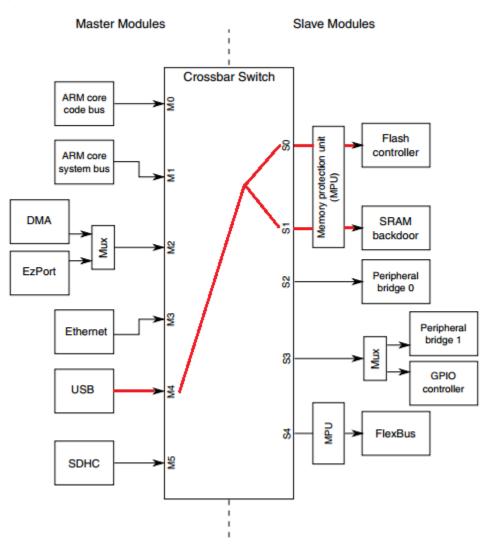


Figure 17. Crossbar switch integration

USB controller can access to variables which can are stored in the Flash memory. or example, the USB device/configuration descriptors. By default, the USB controller has no access to flash memory. Therefore, access must be permitted in Flash Memory Controller (FMC) and Memory Protect Unit (MPU):

- If the **FMC settings** option is enabled, *fsl_usb_framework* component links and configures the *Init_FMC* component for USB module Flash memory read access.
- If the **MPU settings** option is enabled, PEx generates MPU configuration code of the MPU module to the *usb_dev/ host_bsp.c* module. Enable/disable state of MPU module is driven by the *MPU module* property.

Properties Methods Events type filter text Input clock frequency [MHz] 48 MHz 48 MHz All Component name Interrupt INT_US80 Device MPU settings MPU settings 112 MPU settings MPU module MPU module HOST Auto initialization Auto initialization Int_FMC component FMC Device mode Asto initialization Device mode Events Mode Auto initialization Mode Tansceiver type Host mode Shared components fsl_debug_console Inherited components Transceiver type Host mode Mode Inherited components Fil_debug_console Interrupt Interrupt Interrupt Inherited components fsl_debug_console Inherited components Transceiver type Host mode USB tack setting Clock pin USB revision USB tack setting Transceiver type Interrupt Interrupt Clock pin Pin Clock pin Pin Clock frequency [MHz] 48	🗞 *Component Inspector - usbFmw1 🛛	s 🏈	omponents Library Basic Advanced 🎽 🗘 🗢 🗖 🗖
All Component name Device Settings MPU settings FMC settings MPU settings FMC settings MPU module Initialization Device mode Host mode Shared components fsl_debug_console Inherited components << Note MOde Mode M	Properties Methods Events		
	Properties Methods Events type filter text Image: All Component name Device Settings MPU settings MPU settings FMC settings Mode Initialization Auto initialization Device mode Host mode Shared components fsl_debug_console Stard		Input clock frequency [MHz] 48 MHz 48 MHz Interrupt INT_USB0 Interrupt priority medium priority 112 MPU settings MPU module FMC settings Init_FMC component FMC -> HOST mode request Device mode request Mode Mod
			▼ ▼

Figure 18. Properties for device mode initialization

3.5 fsl_usb_ehci_hal component

Used for high speed USB module, provides basic I/O macros for access to the USB high speed OTG controller (USB periphery read/write register access operation).

3.6 fsl_usb_khci_hal component

Used for full speed USB module, provides basic I/O macros for acces to the USB full speed OTG periphery (USB periphery read/write register access operation).

Processor Expert new project wizard helps you to create project based on the selected processor or board type.

- **Boards** project type: CPU and fsl_clock_manager components are configured according selected board, such as Pin signals, XTAL, and MCU timing. In this project type, fsl_clock_manager component contains predefined clock configurations. The **USB clock setup** clock configuration is intended for a USB stack project.
- **Processor** project type: CPU and fsl_clock_manager components are configured to default (after reset) values. In this project type, you must configure the fsl_clock_manager component. The configuration includes properties such as System oscilator-XTAL, MCG mode, and MCU timing.

🏽 New Kinetis Project	_ 0	x
Devices Select the derivative you would like to use		
Processor to be used:		
type filter text Boards Kinetis		
FRDM-K22F FRDM-K64F FRDM-K66F FRDM-K82F FRDM-KL02Z		
FRDM-KL03Z Processors Kinetis E Kinetis K Kinetis L Kinetis M Kinetis V Kinetis W Vybrid		
Contains board support		*
(?) < <u>Back</u> <u>N</u> ext > <u>Finish</u>	Cance	-

Figure 19. Processor Expert new project wizard

Project configuration for "Processor" project type:

1. Create Processor Expert SDK project with selected "Processor" - MK65F180M.

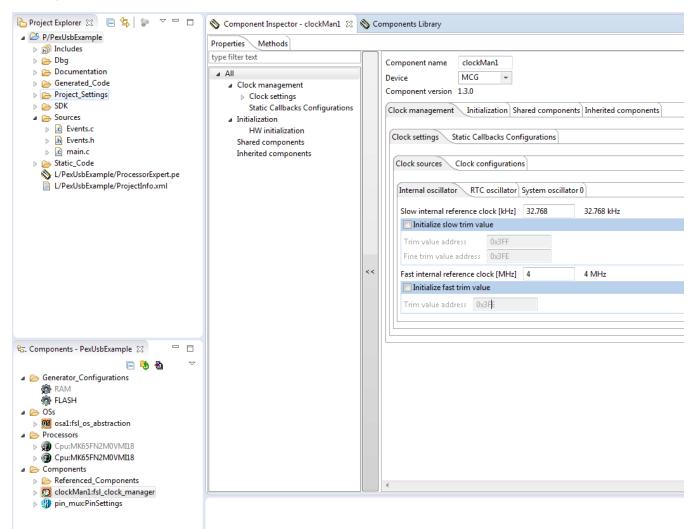


Figure 20. Created PEx project - default settings

- USB full speed (FS) OTG controller requires 48MHz input clock source which depends on fsl_clock_manager component setting. For case if USB module is cloked from internal MCU clock source. The Default (after MCU reset) Cpu clock settings is from internal oscillator and with using FLL module > Default Cpu core clock value is about 20.97152 MHz. This frequency value is not possible to use for USB controller => default Cpu clock settings must be changed.
- 3. In fsl_clock_manager component enables the **System oscilator 0** property. Select the **Clock source = External reference clock**Set ,,Clock frequency [MHz] to 16 (TWR-MK65F180M board contains 16MHz external crystal rezonator see scheme for TWR- MK65F180M board.



Properties Methods		
type filter text All Clock management Clock settings Clock sources Internal oscillator RTC oscillator Clock source Clock configurations Static Callbacks Configurations Initialization HW initialization Shared components Inherited components		✓ System oscillator 0 Clock source External crystal Pin name EXTAL0/PTA18/FTM0_FLT2/FTM_CLKIN0/TPM_CL ▼ Clock output pin Pin name XTAL0/PTA19/FTM1_FLT0/FTM_CLKIN1/LPTMR0_▼ Clock frequency [MHz] 16 16 MHz Capacitor load 0pF Oscillator operating mode Low power ▼
	<<	

Figure 21. "System oscillator 0" configuration4. In the Clock configuration menu click on the MCG settings tab and the MCG mode property. Select the MCG mode as *PEE* and set the **PLL output clock [MHz]** to 120.

a 🛚 Clock management		
☑ All ▲ ☑ Clock management		
a 🛚 Clock management		Clock configuration 0
 E3 All E3 Clock management E3 Clock settings Clock sources E3 Clock configurations Static Callbacks Configurations Initialization HW initialization Shared components Inherited components 	S	Name clockMan1_InitConfig0 Description Default part configuration Read only Internal reference clock External reference clock MCG settings Very low power mode System clock MCG mode PEE PLL select PLL 0 PLL select PLL select PLL 0 PLL select PLL select PLL 0 PLL select PLL select
	<<	MCG output clock PLL clock MCG output [MHz] 120 MCG external ref. clock source System oscillator 0 ▼ MCG external ref. clock [MHz] 16 ICock monitor 16 FLL settings 1 FLL nodule 0 FLL output [MHz] 0 0 MCGFFCLK clock [kHz] 31.25 31.25 kHz Reference clock divider Auto select ▼ 512 FLL reference clock [kHz] 31.25 31.25 kHz Multiplication factor Auto select ▼ 640 PLL osettings 120 120 MHz PLL module 120 120 MHz PLL nodule 120 120 MHz PLL output [MHz] 120 120 MHz
		Reference clock divider Auto select 2 PLL reference clock [MHz] 8 8 MHz Multiplication factor Auto select 30 Loss of lock interrupt Image: Clock interrupt External PLL (USB PHY) PLL module USB PHY PLL output [MHz] 0 0 MHz PFD block Enabled PFD clock [MHz] 0 0 MHz Fractional divider Auto select Divider

Figure 22. MCG module configuration

- 5. On System clocks set Cpu clocking, set
 - **Core clock** = 120 MHz
 - **Bus clock =** 60 MHz
 - External bus clock = 30MHz
 - Flash clock = 15MHz
 - Internal clock source for USB module is from PLL/FLL clock output
 - For USB controller applies: 120MHz / (5 / 2) = 48MHz

🗞 *Component Inspector - clockMan1 🙁 🗞 Components Library							
Properties Methods							
type filter text		Clock configurations - 1 + ^ v					
⊿ All	1						
 Clock management 		# Clock configuration					
⊿ Clock settings b Clock sources		0 Clock configuration 0					
Clock configurations							
Static Callbacks Configurations							
▲ Initialization							
HW initialization Shared components							
Inherited components		Details for selected row:					
		Clock configuration 0					
		Name clockMan1_InitConfig0					
		Description Default part configuration					
		Read only 📝					
		Internal reference clock External reference clock MCG settings Very low power mode System clocks					
		Core clock prescaler Auto select 👻 1					
		Core clock 120 🔫 120 MHz					
		Bus clock prescaler Auto select 👻 2					
		Bus clock 60 🔫 60 MHz					
	<<	External clock prescaler Auto select 👻 4					
		External bus clock 30 👻 30 MHz					
		Flash clock prescaler Auto select 👻 8					
		Flash clock 15 👻 15 MHz					
		PLL/FLL clock selection Auto select 🔻 PLL clock					
		PLL/FLL clock divider 1					
		PLL/FLL clock multiply 1					
		Clock frequency [MHz] 120 120 MHz					
		✓ USB clock settings					
		USB clock divider Auto select 👻 5					
		USB clock multiply Auto select 👻 2					
		USB clock 48 48 MHz					
		TPM clock selection Auto select PLL/FLL clock					
		Clock frequency [MHz] 120 120 MHz					

Figure 23. CPU system clocks configuration

6. USB high speed (HS) OTG controller requires 480MHz (clocked from External PLL USB PHY) input clock source which depends on fsl_clock_manager component setting. External PLL USB PHY is enabled by the **PLL module** property.

NOTE

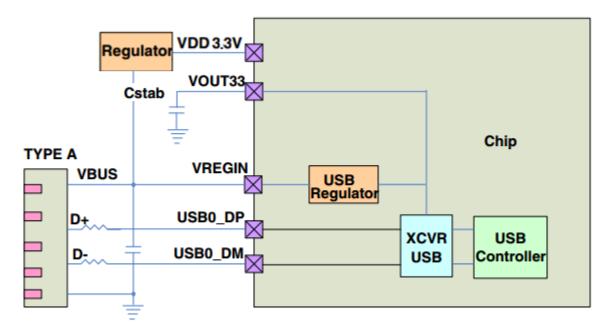
For External PLL USB PHY = 480MHz must have the System oscillator 0 enabled, see the figure above.

🖠 *Component Inspector - clockMan1 🕴 🖠	Compo	nents Library
Properties Methods		
type filter text		Clock configuration 0
a All		Name clockMan1_InitConfig0
 Clock management 		Description Default part configuration
 Clock settings Clock sources 		Read only
Clock configurations		
Static Callbacks Configurations		Internal reference clock External reference clock MCG settings Very low power mode System clocks
 Initialization HW initialization 		MCG mode PEE 👻
Shared components		PLL select PLL 0 -
Inherited components		MCG output clock PLL clock
		MCG output [MHz] 120 120 MHz
		MCG external ref. clock source System oscillator 0 👻
		MCG external ref. clock [MHz] 16 16 MHz Clock monitor
		FLL settings
		FLL module
		FLL output [MHz] 0 0 MHz; FLL is disabled.
		MCGFFCLK clock [kHz] 31.25 31.25 kHz
		Reference clock source External clock 👻
		Reference clock divider Auto select 👻 512
	<<	FLL reference clock [kHz] 31.25 31.25 kHz
		Multiplication factor Auto select 👻
		PLL 0 settings
		PLL module
		PLL module in Stop
		PLL output [MHz] 120 - 120 MHz
		Reference clock divider Auto select • 2 PLL reference clock [MHz] 8 8 MHz
		Multiplication factor Auto select - 30
		Loss of lock interrupt
		External PLL (USB PHY)
		PLL module
		USB PHY PLL output [MHz] 480 480 MHz
		PFD block Enabled 👻
		PFD clock [MHz] 180 - 180 MHz
		Fractional divider Auto select - 18/24
		Divider Auto select 👻 2
		Clock monitor Disabled 🔻

Figure 24. External PLL USB PHY" configuration

7. USB HS/FS/LS transceiver module requires voltage source (3.3V) provided by the USB regulator output. Input of the USB regulator is connected to the Vregin pin on the MCU package. This PIN can be connected to USB_VBUS line (USB transceiver is powered from USB host device, for example: PC, USB device BUS power mode) or to 5V board power supply (USB device self power mode).

Figure 25. USB regulator, BUS power use case



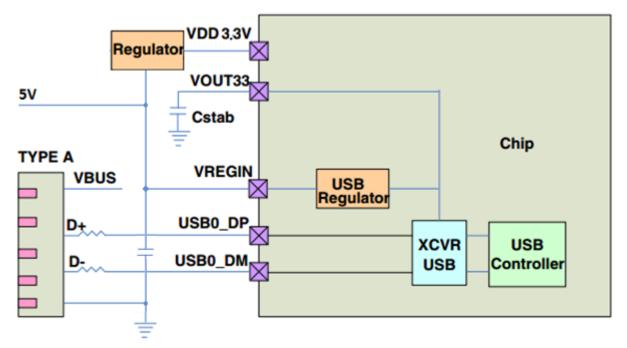


Figure 26. USB self power use case

TWR-K65F180M_Rev B: J31 selector on TWR- MK65F180M board specifies which USB module is connected J15 connector (USB micro):

- 1-2: J15 goes to USB HS and Elevator goes to USB FS
- 2-3: J15 goes to USB FS and Elevator goes to USB HS

TWR-K65F180M_Rev C:

- USBHS connected to J15 connector (USB micro on TWR-K65F128 board)
- USBFS connected to elevator (USB connector on TWR-SER board).

Processor Expert Kinetis SDK USB Stack Integration, Rev. 0, 07/2015

4.1 Creating PEx USB project

The simples way to create PEx USB project is to insert the USB class component (*fsl_usb_device_hid/msd_class*). In this case, all the required lower USB stack layer components are automatically added to the project by Processor Expert.

For better demonstration and function, the *fsl_usb_device_hid/msd_class* components contain properties for activating USB demo project. In this mode the class component is configured according to the selected demo mode type and the code in the "user" file is activated.

4.1.1 USB mass storage project

1. For USB mass storage project add the *fsl_usb_device_msd_class* component from **Components Library** window to the created project (see chapter Creating common PEx USB Project)

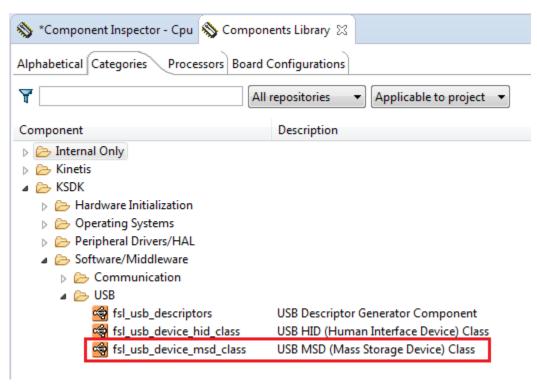


Figure 27. USB components in "Component Library"

Function of fsl_usb_device_msd_class driver requires presence of other USB layers: fsl_usb_descriptors, fsl_usb_framework (see PEx USB stack structure) and others SDK drivers (fsl_clock and interrupt_manager, etc). All these components are automatically added to the project.

2. Configure "Subclass code" and "Protocol code" on "MSD Class" tab. Only SCSI transparent command and BBB (bulk only transport) are currently supported.

🔇 *Component Inspector - ms	d1 2	3 🗞 Components Library	Basic Advanced 🎽 🕺	~ - 8
Properties Methods Events				
type filter text All MSD Class SDK specific settings Initialization	<<	Component name msdl fsl_usb_descriptor component link usbDsc1 + > Update fsl_usb_descriptor component MSD Class SDK specific settings Subclass code px06 SCSI transparent command set + Protocol code 0x50 BBB (bulk only transport) +		
				•

Figure 28. MSD Subclass and Protocol code configuration

- 3. On "SDK specific settings" tab:
 - a. Select "Initialization" and "Auto initialization" checkboxes (They should be selected by default)
 - b. Select "USB RAM disk demo" to use predefined demo code for USB RAM disk demo. Configure Sectors count and Sector size parameters of RAM disk (if default values are not suitable).

🔇 *Component Inspector - m	sd1 🛛	S 💊 Components Library	Basic Advanced 🎽 🎽 🗖
Properties Methods Events	s		
type filter text All MSD Class SDK specific settings Initialization		Component name msd1 fsl_usb_descriptor component link usbDsc1 + > Update fsl_usb_descriptor component MSD Class SDK specific settings	
		✓ Initialization MSD device handler name msd1_MsdHandle MSD configuration structure name msd1_MsdConfigStructure ✓ USB RAM disk demo	
	~~	Sectors count 64 RAM disk sector size [B] 512 Calculated RAM disk size [kB] 32	
		Callback user parameters application_callback arg vendor_req_callback_arg NULL class_specific_callback arg NULL ✓ Auto initialization	
		•	

Figure 29. Properties for MSD class initialization

Processor Expert Kinetis SDK USB Stack Integration, Rev. 0, 07/2015

4. Press code generation button. As soon as PEx generation is done, all required files should be created /added in the project.

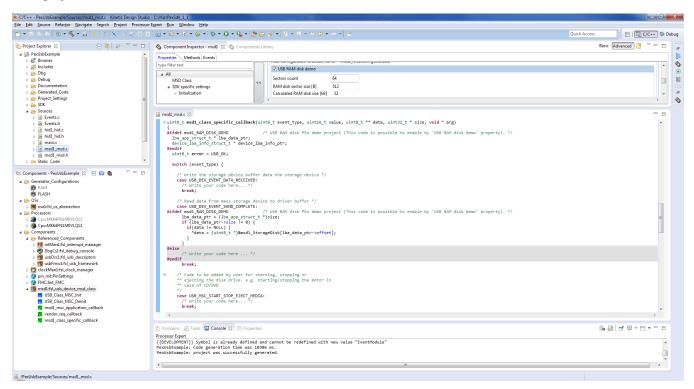


Figure 30. MSD class callbacks API (with RAM disk demo code) generated by Processor Expert

- 5. Build and load USB project into TWR-MK65F180M board.
- 6. Plug-in the USB connector of MK65F180M to PC.
- 7. The windows will prompt you to format the disk.



Figure 31. OS Windows requires to format disk

8. When the format is completed, the computer will display the capacity of removable disk.

4.1.2 USB HID project

1. For USB HID project add the *fsl_usb_device_hid_class* component from "Components Library" window to the created project (see chapter Creating common PEx USB Project).

Processor Expert Kinetis SDK USB Stack Integration, Rev. 0, 07/2015

📎 *Component Inspector - Cpu 🚫 Com	ponents Library 🔀						
Alphabetical Categories Processors Board Configurations							
¥	All repositories Applicable to project						
Component	Description						
Internal Only							
Kinetis							
🔺 🗁 KSDK							
b 🗁 Hardware Initialization							
Derating Systems							
Peripheral Drivers/HAL							
a 🗁 Software/Middleware							
Communication							
🔺 🗁 USB							
🗠 fsl_usb_descriptors	USB Descriptor Generator Component						
📸 fsl_usb_device_hid_class	USB HID (Human Interface Device) Class						
👒 fsl_usb_device_msd_class	USB MSD (Mass Storage Device) Class						

Figure 32. USB components in "Component Library"

Function of *fsl_usb_device_hid_class* driver requires presence of other USB layers: *fsl_usb_descriptors*, *fsl_usb_framework* (see PEx USB stack structure) and others SDK drivers (fsl_clock and interrupt_manager, etc). All these components are automatically added to the project.

After adding the *fsl_usb_device_hid_class* component to the project, *fsl_usb_device_hid_class* component must be configured to specify HID type. HID type is specified by Subclass, Protocol code and Report descriptor properties.

Figure 33. HID type configuration

Subclass, Protocol code and HID Report descriptor properties should be configurated according USB HID specification. For better fsl_usb_device_hid_class component function demonstration, fsl_usb_device_hid_class component contains HID type configuration properties:

• **Report descriptor template buttons** – configure report descriptor according to the one predefined templates: mouse, keyboard, thermometer.

🗞 *Component Inspector - hid1 🙁 🗞 Components Library							
Properties Methods Events							
type filter text		Report descriptor templates					
 All Generated code common settings HID class setting HID descriptor Class descriptor list Report descriptor templates Initialization Callback user parameters 	<<	Configure as mouse Configure as keyboard Configure as thermometer Confi					

Figure 34. HID template buttons

• USB HID demo mode – configure report descriptor according according selected demo type and enable demo code.

Properties Methods Events		
type filter text	Initialization	
Generated code common settings HID class setting HID descriptor Class descriptor list Report descriptor	HID device handler name HID configuration structure name Auto initialization USB HID demo mode Callback of USB HID demo mode Disabled Disabled Mouse Keyboard Thermomet	hid1_HidHandle hid1_HidConfigStructure

Figure 35. USB HID demo mode selection

HID template/demo types description:

• **Mouse** – Report descriptor is configured to standard three-button mouse detectable by BIOS. In mouse demo mode is mouse pointer left-right moved. Speed of mouse pointer move is defined by mouse PIPE "Polling interval" (set to 128ms after demo mode activation) property in *fsl_usb_descriptors* component.

• **Keyboard** - Report descriptor is configured to standard keyboard detectable by BIOS. In keyboard demo mode are repeatedly PageUp/Down keys pressed. Speed of PageUp/Down keys pressed is defined by keyboard PIPE "Polling interval" (set to 128ms after demo mode activation) property in *fsl_usb_descriptors* component.

• **Thermometer** - Report descriptor is configured to thermometer device, temperature unit = Kelvin, resolution = 1K, minimum temperature value = 218K, maximum temperature value = 393K. In thermometer demo mode are repeatedly 0K and 123K values sent to PC. For reading temperature value (on PC side) from HID t hermometer is need user application. Access to t

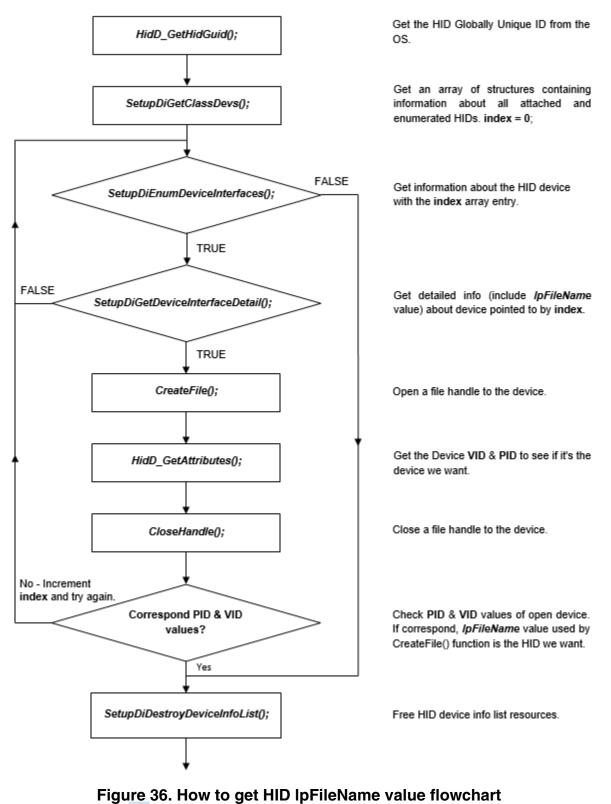
- hermometer in Windows OS is through standard windows functions:
 CreateFile() Creates or opens a file or I/O device,
 - *ReadFile()* Reads data from the specified file or input/output (I/O) device,
 - WriteFile() Writes data to the specified file or input/output (I/O) device,
 - *CloseHandle()* Closes an open object handle.

For open a HID device (get handle) by *CreateFile()* function is need know *lpFileName* parameter value. *lpFileName* parameter specifies device name to be opened. *lpFileName* parameter value is possible to get by functions defined in "hid.dll" library according USB device Product ID & Vendor ID values (PID & VID values of USB device are defined in *fsl_usb_descriptors* component). "hid.dll" library is available in Windows \System32 folder.

List of functions used to get HID *lpFileName*:

- *HidD_GetHidGuid()* returns the device interface GUID for HIDClass devices.
- *SetupDiGetClassDevs()* returns a handle to a device information set that contains requested device information elements for a local computer.
- *SetupDiEnumDeviceInterfaces()* numerates the device interfaces that are contained in a device information set.
- SetupDiDestroyDeviceInfoList() deletes a device information set and frees all associated memory.
- *SetupDiGetDeviceInterfaceDetail()* returns details about a device interface.
- *HidD_GetAttributes()* returns the attributes of a specified top-level collection.

More information about HID access functions is available here: https://msdn.microsoft.com.



2. Press the code generation 🗟 button. As soon as PEx generation is done, all required files should be created /added in the project.

Processor Expert Kinetis SDK USB Stack Integration, Rev. 0, 07/2015

Processor Expert Kinetis SDK USB Stack Integration, Rev. 0, 07/2015

_ D _ X

Creating common PEx USB project

🛞 C/C++ - PexUsbExample/Sources/hid1_hid.c - Kinetis Design Studio

Eile Edit Source Refactor Navigate Search Pro												
🔁 • 🖩 🕼 🗠 🏵 • 🗞 • 🗟 🕅 🖉 🤣	i ☆ • ◎ • i * • • • • • • • • • • • • • • • • •	6 🖉 🗁 🛷 🔹	🌙 📚 🗉 🖬	图 - 图 - 40 4	$\Leftrightarrow \bullet \bullet \bullet$		Quick Access	🔂 C/C++ 🕸 🛙	Debug			
Project Explorer 🛛 📄 😫 🔝 🗢 🗆	🗖 💊 Component Inspector - hid1 🙁 📎 Compone	nts Library					Basic Advanced 🎬 🔶		ě			
▲ ➢ PexUsbExample	Properties Methods Events	Properties Methods Events										
Includes	type filter text											
Documentation Generated_Code			V Initializati	ion								
Development Settings	HID class setting	^	HID device h	nandler name	hid1_HidHa	andle						
Des SDK	 HID descriptor Class descriptor list 		HID configu	ration structure nam	hid1 HidCo	onfigStructure						
A 🔁 Sources	▲ Report descriptor		Auto init	tialization	_	5		E	\$			
Events.c	Report descriptor templates		<< Auto init									
Events.h	Report descriptor Item List	E	USB HID de	mo mode Callba	ick user paramet	ters						
b id1_hid.c	▲ Initialization		USB HID de	mo mode Mouse	-							
hid1_hid.h	Callback user parameters							•	-			
main.c		-	•					F.				
Static_Code												
ProcessorExpert.pe	id1_hid.c ≥3											
ProjectInfo.xml	⊖uint8_t hid1_class_specific_callbac	k(uint8_t requ	est, uint16_t v	/alue, uint8_t '	** data, uint	t32_t * size, void * arg)		*				
	i uint8 t error = USB OK;											
😪 Components - PexUsbExample 🔀 👘 🗎	- if ((request -= 050_bev_event_set											
🚍 🧐 🐴	<pre>#ifdef hid1_HID_DEMO if (HidInitialized) {</pre>	/* USB HID	PEx demo proje	ect (This code :	is possible t	to enable by 'USB HID demo' propert	y). */					
▲ Generator_Configurations	hid1_hid_events_process();											
RAM -	}											
🛞 FLASH	return error;											
a 🗁 OSs	<pre>#else /* Write your code here */</pre>											
osa1:fsl_os_abstraction	#endif											
Processors	}											
Cpu:MK65FN2M0VMI18	<pre>*size = 0; /* Handle the class request */</pre>											
Gpu:MK65FN2M0VMI18	switch (request) {											
Components Efferenced_Components	case USB_HID_GET_REPORT_REQUEST											
DSB_Referenced_Components	<pre>#ifdef hid1_HID_DEMO</pre>				is possible t	to enable by 'USB HID demo' propert	y). */					
Ø clockMan1:fsl_clock_manager	*size = hid1_InputReportBuffe			bre to send '7								
b in_mux:PinSettings	#else											
A 🙀 hid1:fsl_usb_device_hid_class	/* Write your code here *	/										
USB_Class_HID_Init	<pre>#endif break;</pre>											
USB_Class_HID_Deinit	bi cury							E				
USB_Class_HID_Recv_Data	case USB_HID_SET_REPORT_REQUEST											
USB_Class_HID_Send_Data	<pre>#ifdef hid1_HID_DEMO for(int index = 0; index < hiv</pre>					to enable by 'USB HID demo' propert	sy). */					
USB_Class_HID_Cancel	hid1 InputReportBuffer[inde			(cop)	y the report	sence by the host of						
USB_Class_HID_Get_Speed	}											
hid1_application_callback	<pre>#else /* Write your code here *</pre>											
👼 vendor_req_callback 👼 hid1_class_specific_callback	#endif	1										
board init callback	break;							-				
Jourd_Init_Canouck	<							Þ				
	💽 Problems 🔀 🔎 Tasks 📃 Console 🔲 Prop	ortier						5 V P D				
		Contraction (1997)						e- L				
	0 items		D		-				-			
	Description	Resource	Path	Location	Туре							
/PexUsbExample/Sources/hid1_hid.c							1					
							•		_			

Figure 37. HID class callbacks API with Mouse demo code generated by Processor Expert

- 3. Build and load USB project into TWR-MK65F180M board.
- 4. Plug-in the USB connector of MK65F180M to PC.

How to Reach Us:

Home Page: freescale.com

Web Support: freescale.com/support Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. Freescale reserves the right to make changes without further notice to any products herein.

Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Freescale, the Freescale logo, CodeWarrior, Kinetis, and Processor Expert are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. All other product or service names are the property of their respective owners. All rights reserved.

© 2015 Freescale Semiconductor, Inc.

Document Number PEXKSDKUSBSI Revision 0, 07/2015

