FTF FREESCALE TECHNOLOGY FORUM 2014

# New Kinetis Design Studio

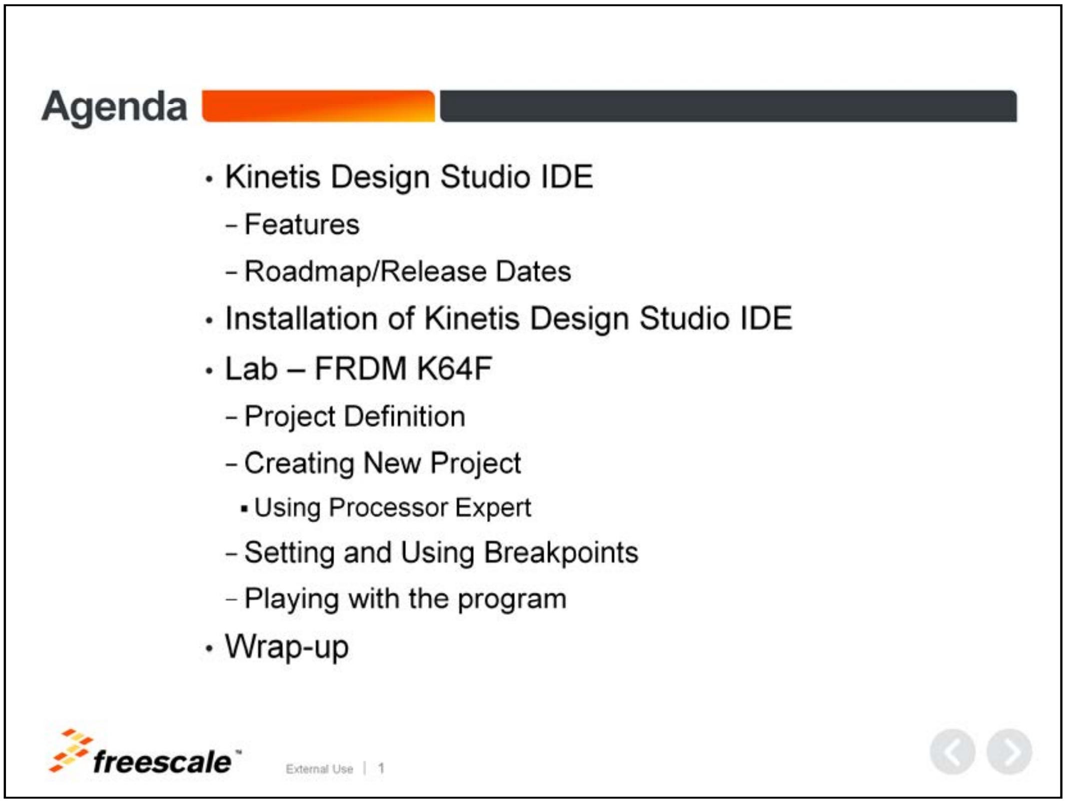
## FTF-SDS-F0250

## Hands-on Workshop

Michael Norman | MCU Software and Tools Marketing Manager
Mark Ruthenbeck | Applications Engineering

April.08/09.2014

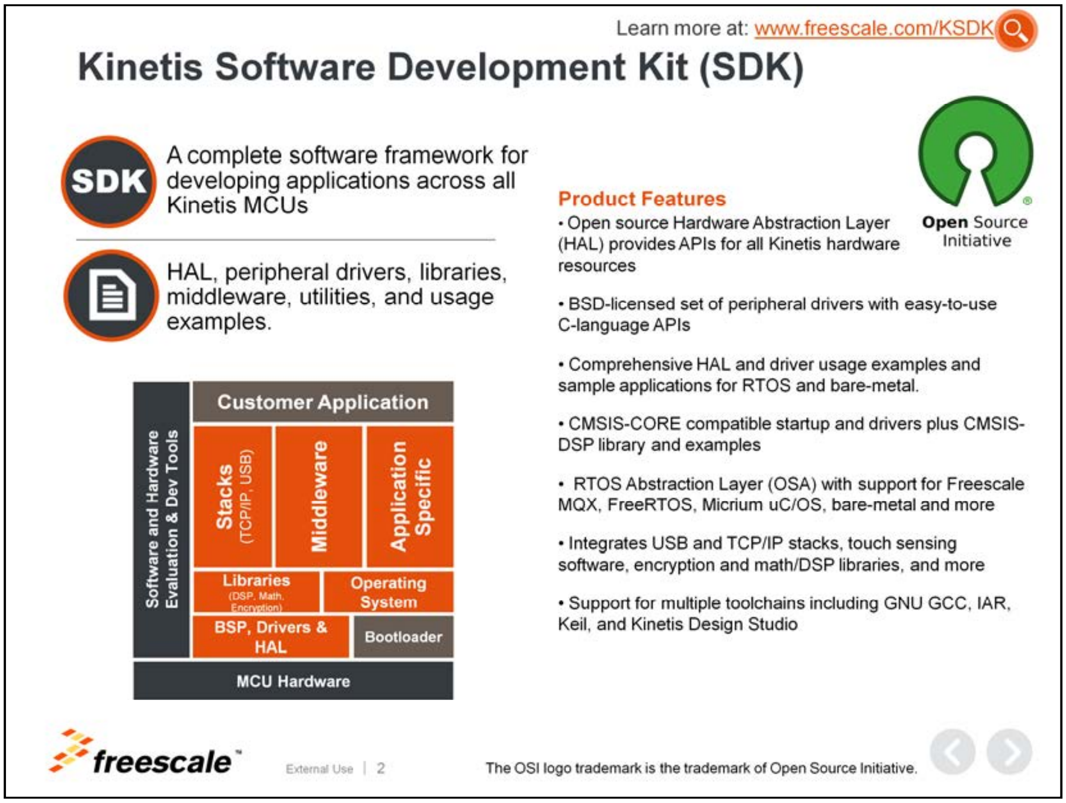
External Use



## Agenda

- Kinetis Design Studio IDE
  - Features
  - Roadmap/Release Dates
- Installation of Kinetis Design Studio IDE
- Lab – FRDM K64F
  - Project Definition
  - Creating New Project
    - Using Processor Expert
  - Setting and Using Breakpoints
  - Playing with the program
- Wrap-up

External Use | 1



Learn more at: www.freescale.com/KSDK

# Kinetis Software Development Kit (SDK)

SDK A complete software framework for developing applications across all Kinetis MCUs

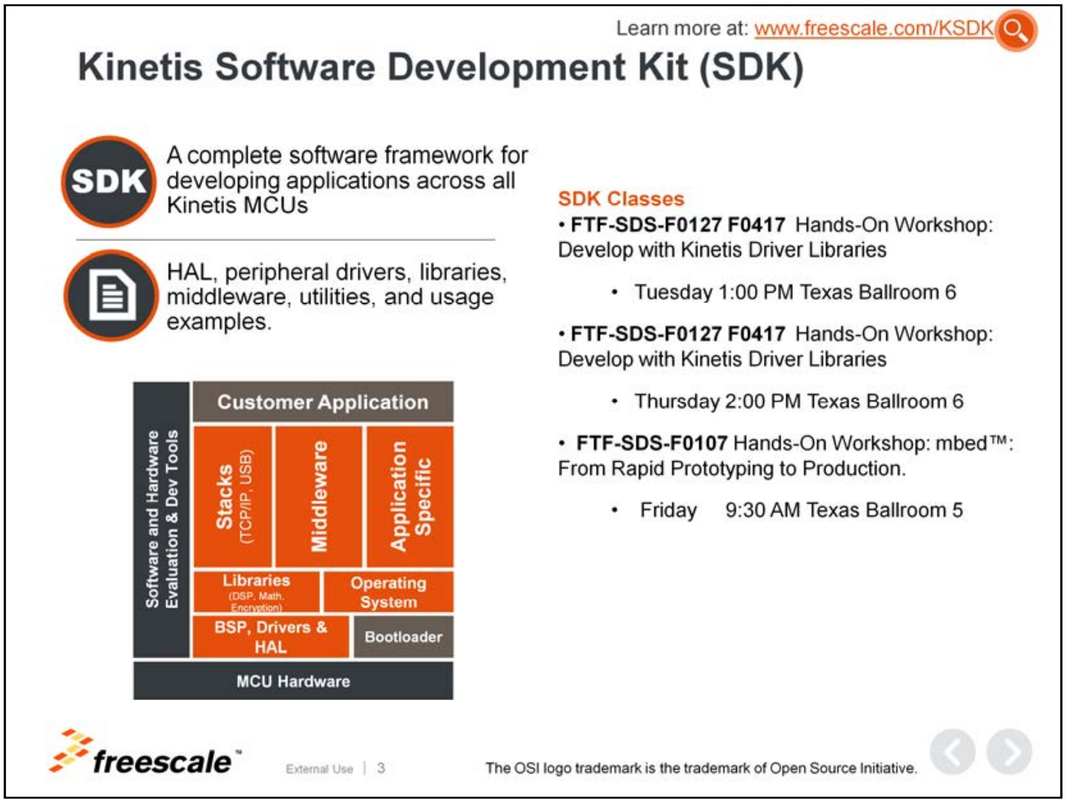
HAL, peripheral drivers, libraries, middleware, utilities, and usage examples.

**Product Features**

- Open source Hardware Abstraction Layer (HAL) provides APIs for all Kinetis hardware resources
- BSD-licensed set of peripheral drivers with easy-to-use C-language APIs
- Comprehensive HAL and driver usage examples and sample applications for RTOS and bare-metal.
- CMSIS-CORE compatible startup and drivers plus CMSIS-DSP library and examples
- RTOS Abstraction Layer (OSA) with support for Freescale MQX, FreeRTOS, Micrium uC/OS, bare-metal and more
- Integrates USB and TCP/IP stacks, touch sensing software, encryption and math/DSP libraries, and more
- Support for multiple toolchains including GNU GCC, IAR, Keil, and Kinetis Design Studio

External Use | 2

The OSI logo trademark is the trademark of Open Source Initiative.



Learn more at: www.freescale.com/KSDK

## Kinetis Software Development Kit (SDK)

SDK A complete software framework for developing applications across all Kinetis MCUs

HAL, peripheral drivers, libraries, middleware, utilities, and usage examples.

Customer Application

Stacks (TCP/IP, USB)

Middleware

Application Specific

Libraries (DSP, Math, Encryption)

Operating System

BSP, Drivers & HAL

Bootloader

MCU Hardware

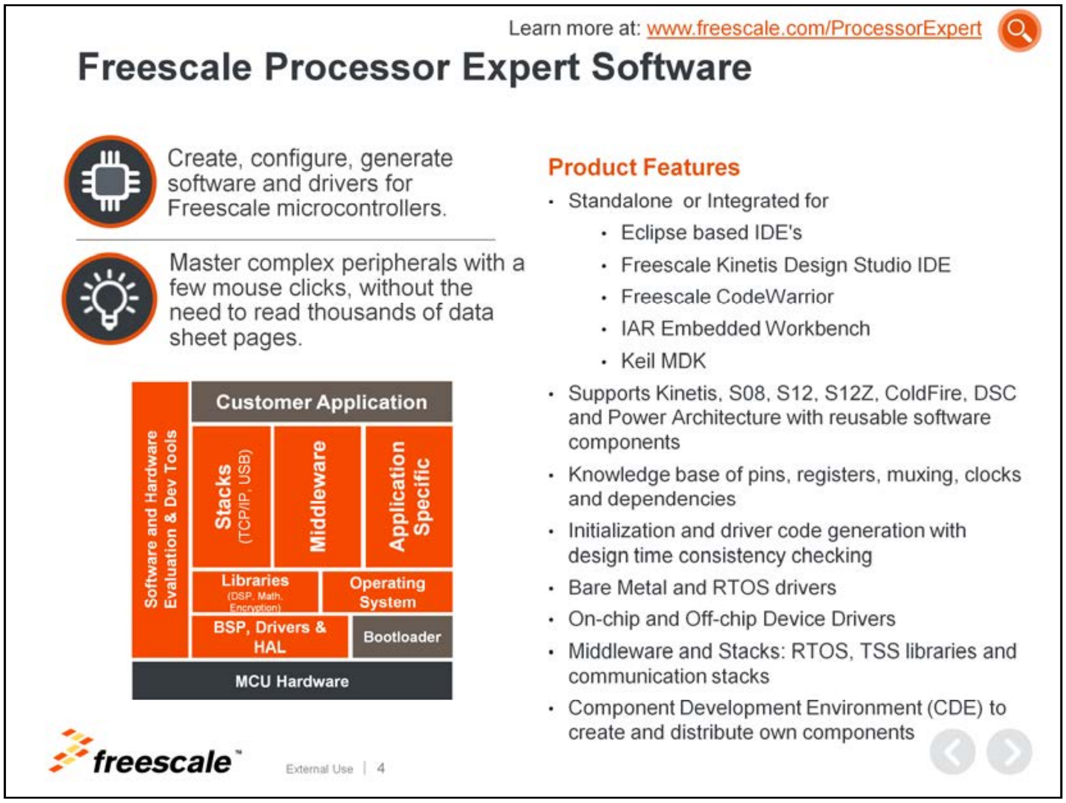
Software and Hardware Evaluation & Dev Tools

**SDK Classes**

- **FTF-SDS-F0127 F0417** Hands-On Workshop: Develop with Kinetis Driver Libraries
  - Tuesday 1:00 PM Texas Ballroom 6
- **FTF-SDS-F0127 F0417** Hands-On Workshop: Develop with Kinetis Driver Libraries
  - Thursday 2:00 PM Texas Ballroom 6
- **FTF-SDS-F0107** Hands-On Workshop: mbed™: From Rapid Prototyping to Production.
  - Friday 9:30 AM Texas Ballroom 5

freescale™ External Use | 3

The OSI logo trademark is the trademark of Open Source Initiative.



Learn more at: www.freescale.com/ProcessorExpert

## Freescale Processor Expert Software

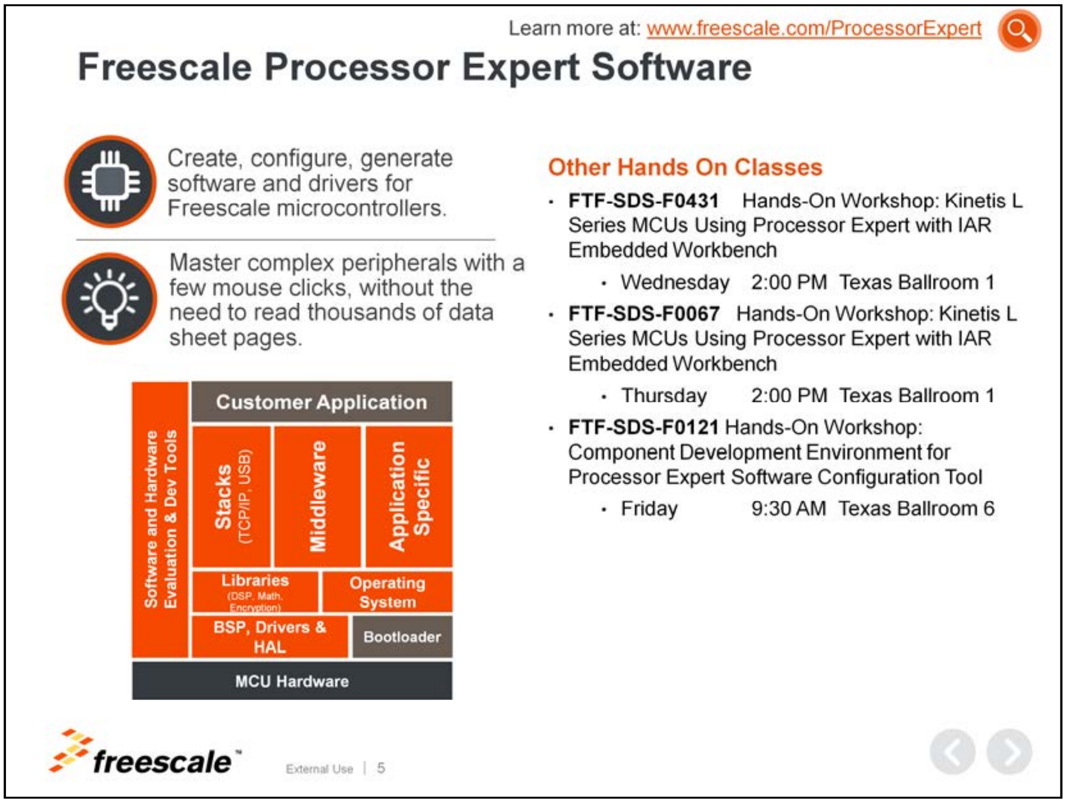
Create, configure, generate software and drivers for Freescale microcontrollers.

Master complex peripherals with a few mouse clicks, without the need to read thousands of data sheet pages.

Customer Application
Software and Hardware Evaluation & Dev Tools
Stacks (TCP/IP, USB)
Middleware
Application Specific
Libraries (DSP, Math, Encryption)
Operating System
BSP, Drivers & HAL
Bootloader
MCU Hardware

### Product Features

- Standalone or Integrated for
  - Eclipse based IDE's
  - Freescale Kinetis Design Studio IDE
  - Freescale CodeWarrior
  - IAR Embedded Workbench
  - Keil MDK
- Supports Kinetis, S08, S12, S12Z, ColdFire, DSC and Power Architecture with reusable software components
- Knowledge base of pins, registers, muxing, clocks and dependencies
- Initialization and driver code generation with design time consistency checking
- Bare Metal and RTOS drivers
- On-chip and Off-chip Device Drivers
- Middleware and Stacks: RTOS, TSS libraries and communication stacks
- Component Development Environment (CDE) to create and distribute own components



Learn more at: www.freescale.com/ProcessorExpert

## Freescale Processor Expert Software

Create, configure, generate software and drivers for Freescale microcontrollers.

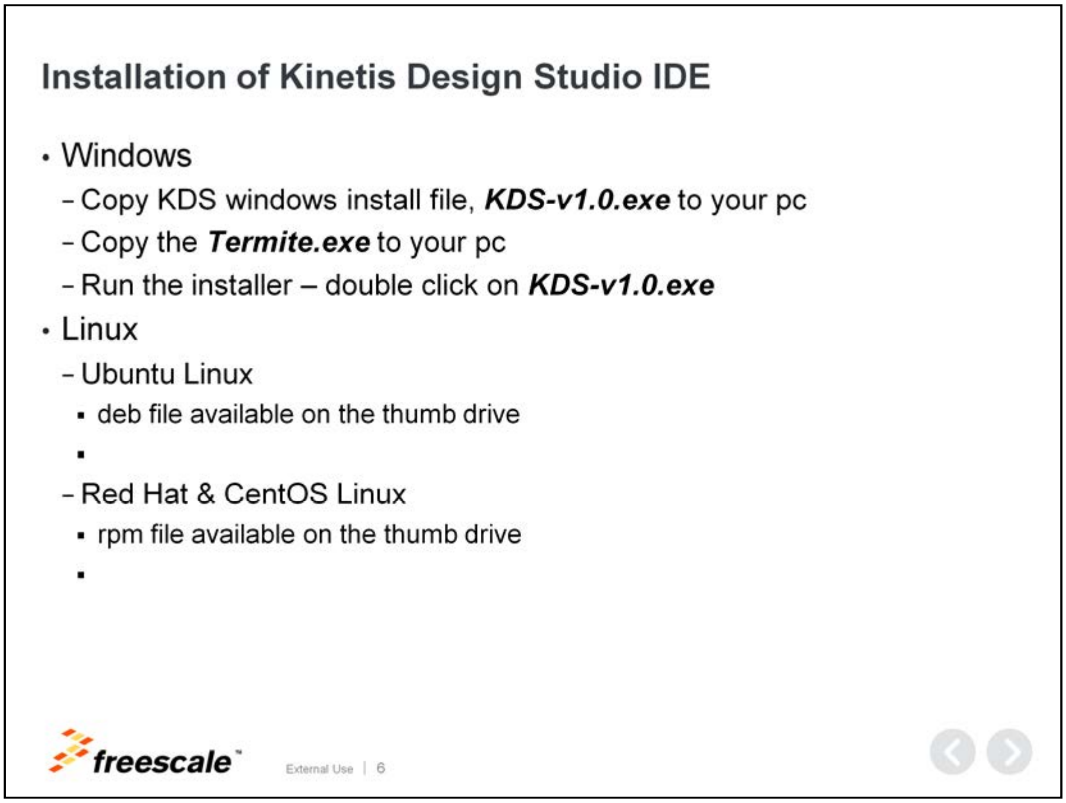
Master complex peripherals with a few mouse clicks, without the need to read thousands of data sheet pages.

Customer Application
Software and Hardware Evaluation & Dev Tools
Stacks (TCP/IP, USB)
Middleware
Application Specific
Libraries (DSP, Math, Encryption)
Operating System
BSP, Drivers & HAL
Bootloader
MCU Hardware

### Other Hands On Classes

- **FTF-SDS-F0431** Hands-On Workshop: Kinetis L Series MCUs Using Processor Expert with IAR Embedded Workbench
  - Wednesday 2:00 PM Texas Ballroom 1
- **FTF-SDS-F0067** Hands-On Workshop: Kinetis L Series MCUs Using Processor Expert with IAR Embedded Workbench
  - Thursday 2:00 PM Texas Ballroom 1
- **FTF-SDS-F0121** Hands-On Workshop: Component Development Environment for Processor Expert Software Configuration Tool
  - Friday 9:30 AM Texas Ballroom 6

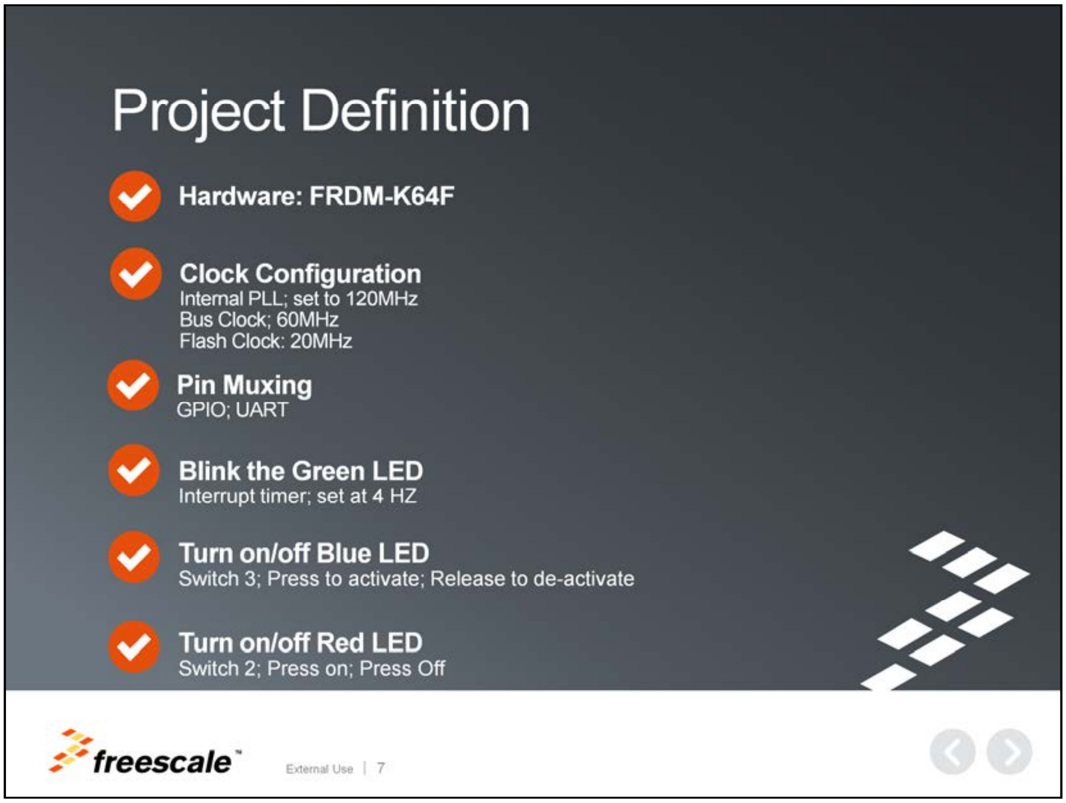
External Use | 5



## Installation of Kinetis Design Studio IDE

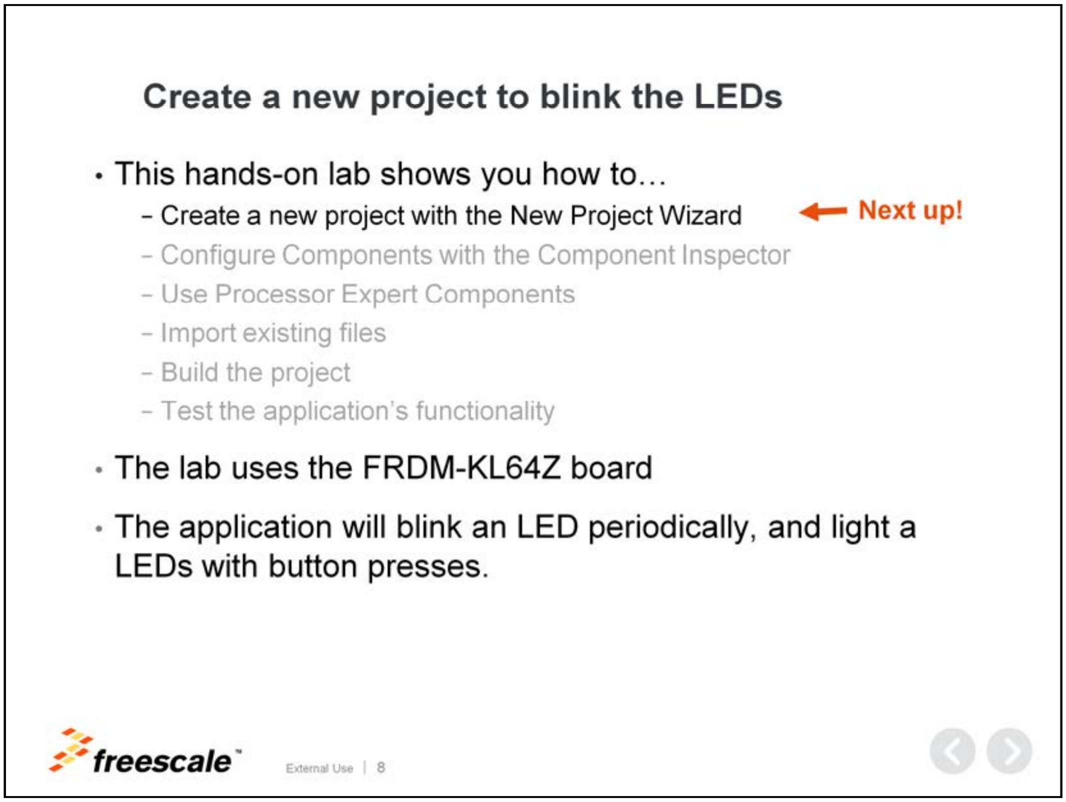
- Windows
  - Copy KDS windows install file, ***KDS-v1.0.exe*** to your pc
  - Copy the ***Termite.exe*** to your pc
  - Run the installer – double click on ***KDS-v1.0.exe***
- Linux
  - Ubuntu Linux
    - deb file available on the thumb drive
    -
  - Red Hat & CentOS Linux
    - rpm file available on the thumb drive
    -

External Use | 6



# Project Definition

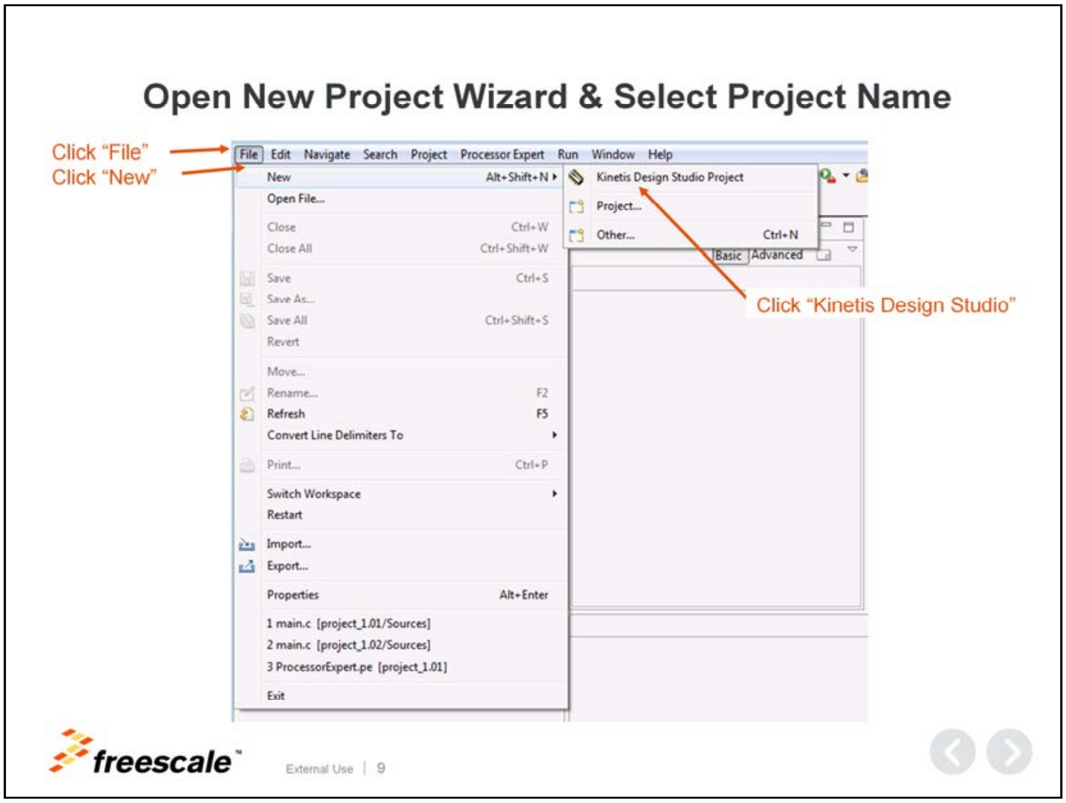
- ✓ **Hardware: FRDM-K64F**
- ✓ **Clock Configuration**
  Internal PLL; set to 120MHz
  Bus Clock; 60MHz
  Flash Clock: 20MHz
- ✓ **Pin Muxing**
  GPIO; UART
- ✓ **Blink the Green LED**
  Interrupt timer; set at 4 HZ
- ✓ **Turn on/off Blue LED**
  Switch 3; Press to activate; Release to de-activate
- ✓ **Turn on/off Red LED**
  Switch 2; Press on; Press Off



## Create a new project to blink the LEDs

- This hands-on lab shows you how to…
  - Create a new project with the New Project Wizard ← Next up!
  - Configure Components with the Component Inspector
  - Use Processor Expert Components
  - Import existing files
  - Build the project
  - Test the application's functionality
- The lab uses the FRDM-KL64Z board
- The application will blink an LED periodically, and light a LEDs with button presses.

This section will show how to use the new project wizard and to import existing files into the project.



## Open New Project Wizard & Select Project Name

Click "File"

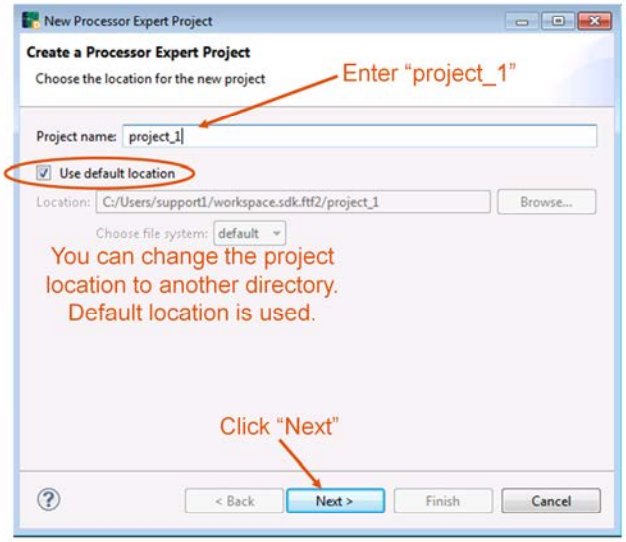
Click "New"

Click "Kinetis Design Studio"

External Use | 9

In order to use the New Project Wizard, you can only get to it from File->New->Kinetis Design Studio. If you use the "add" icon, the New Project Wizard is not a selection.

## Open New Project Wizard & Select Project Name



New Processor Expert Project

**Create a Processor Expert Project**

Choose the location for the new project

Project name: project_1

☑ Use default location

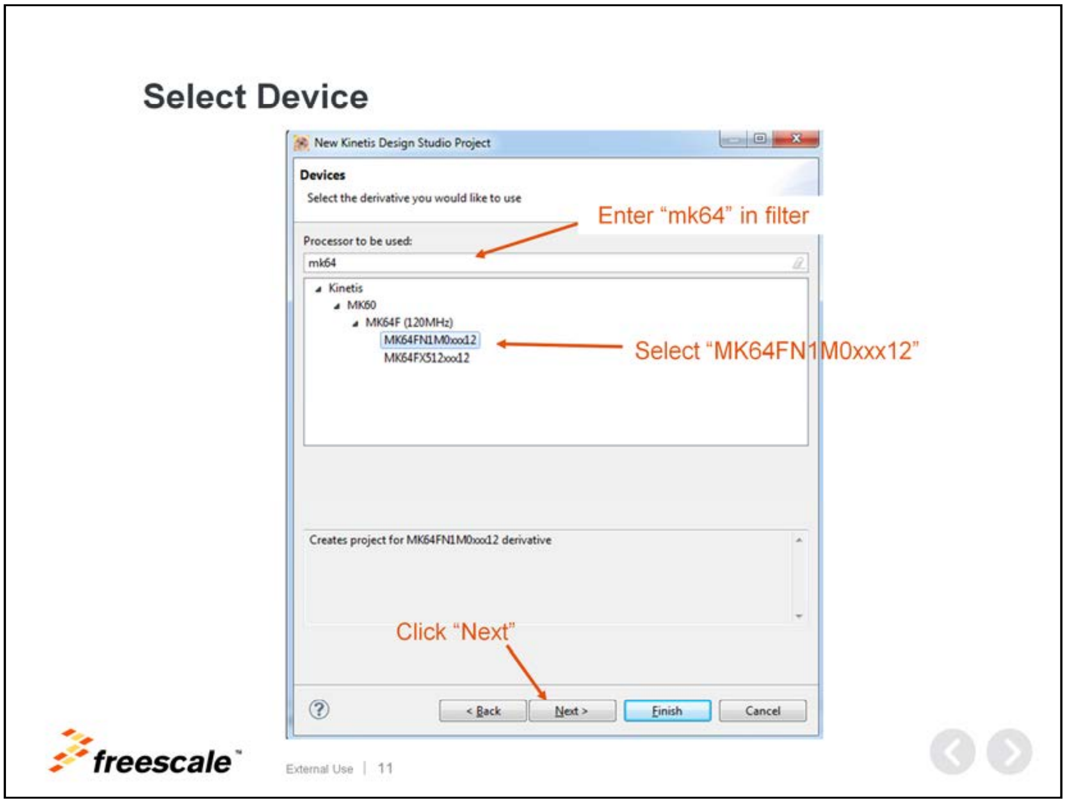
Location: C:/Users/support1/workspace.sdk.ftf2/project_1

Choose file system: default

Enter "project_1"

You can change the project location to another directory. Default location is used.

Click "Next"



## Select Device

New Kinetis Design Studio Project

**Devices**

Select the derivative you would like to use

Processor to be used:

mk64

- Kinetis
  - MK60
    - MK64F (120MHz)
      - MK64FN1M0xxx12
      - MK64FX512xxx12

Creates project for MK64FN1M0xxx12 derivative

Enter “mk64” in filter

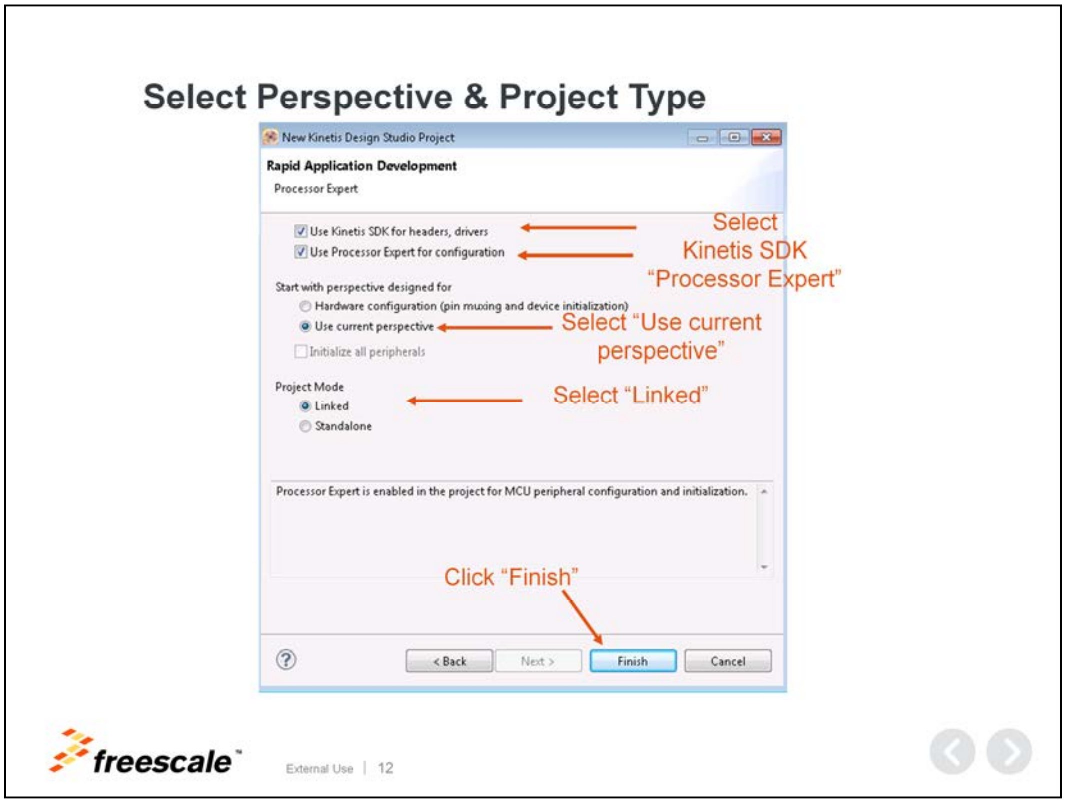
Select “MK64FN1M0xxx12”

Click “Next”

External Use | 11

The filter is a quick way to find the part you are looking for.

In the New Project Wizard, you select the base part. You will need to specify the specific part, including package & pins, you are using. This is done in the *Properties* tab of the Processor Component.



## Select Perspective & Project Type

New Kinetis Design Studio Project

**Rapid Application Development**

Processor Expert

☑ Use Kinetis SDK for headers, drivers

☑ Use Processor Expert for configuration

Start with perspective designed for

○ Hardware configuration (pin muxing and device initialization)

● Use current perspective

□ Initialize all peripherals

Project Mode

● Linked

○ Standalone

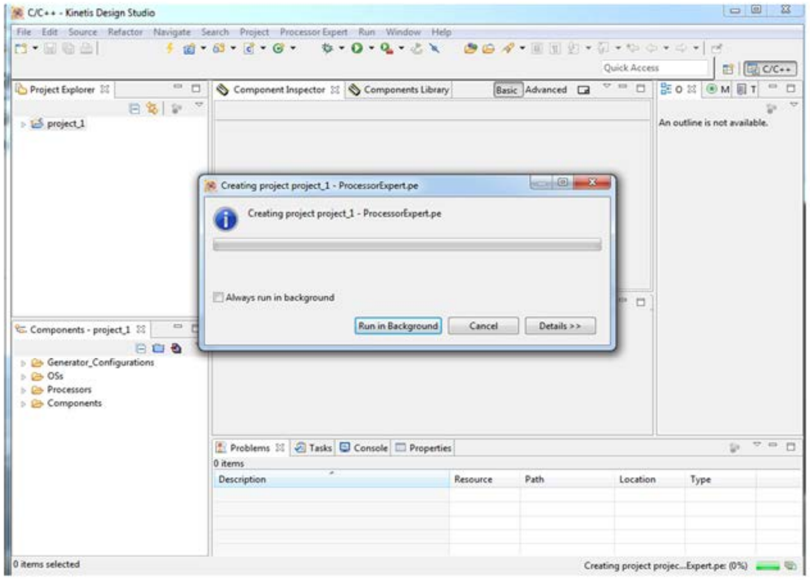
Processor Expert is enabled in the project for MCU peripheral configuration and initialization.

External Use | 12

Select Kinetis SDK (this will be selectable for processors that are supported by the SDK). Start with Perspective: Lets set this to the non-processor expert hardware configuration perspective and select Linked project mode. The Standalone project mode is not functional in the this Beta release. If you use "Linked" mode, and you modify the contents of the code, then you are modifying the reference code.

If you plan on archiving the project or sharing the project, this assures the entire project is included, all the source files, etc.

## Project displayed in C/C++ Perspective



C/C++ - Kinetis Design Studio

Creating project project_1 - ProcessorExpert.pe

Creating project project_1 - ProcessorExpert.pe

□ Always run in background

An outline is not available.

0 items

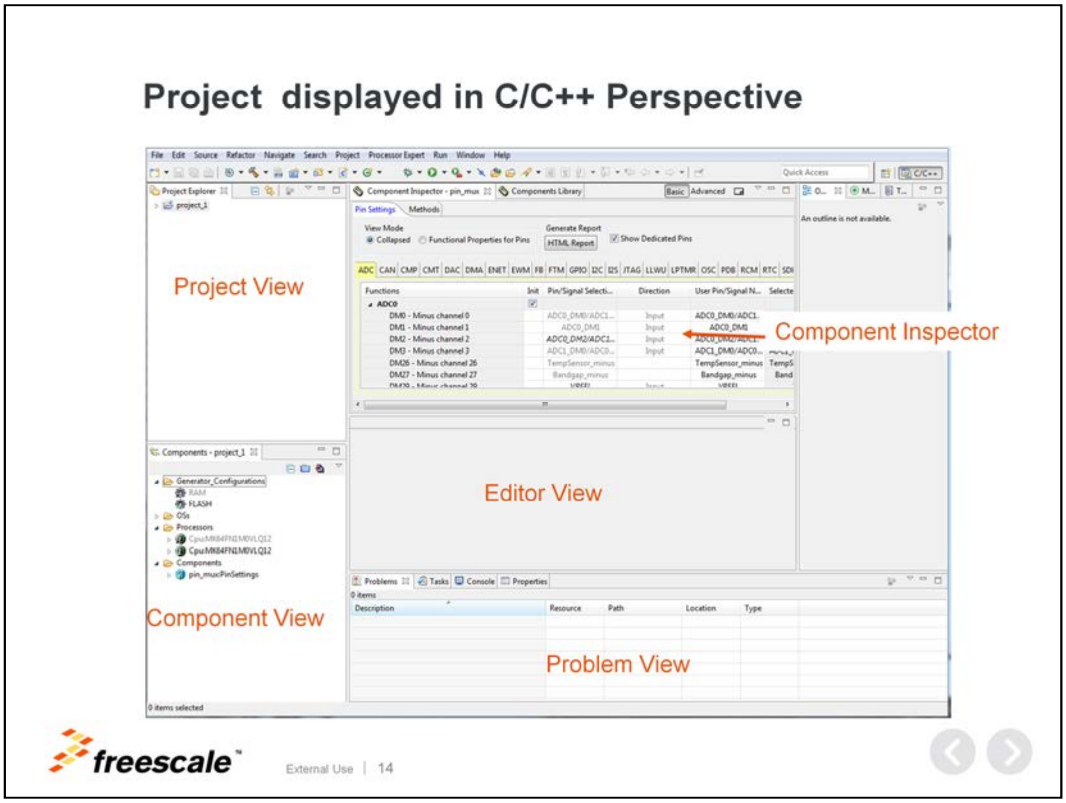
0 items selected

Creating project projec...Expert.pe: (0%)

External Use | 13

Processor Expert is working…

## Project displayed in C/C++ Perspective



| Functions | Init | Pin/Signal Selecti... | Direction | User Pin/Signal N... | Selecte |
|---|---|---|---|---|---|
| ADC0 | ☑ | | | | |
| DM0 - Minus channel 0 | | ADC0_DM0/ADC1... | Input | ADC0_DM0/ADC1. | |
| DM1 - Minus channel 1 | | ADC0_DM1 | Input | ADC0_DM1 | |
| DM2 - Minus channel 2 | | ADC0_DM2/ADC1... | Input | ADC0_DM2/ADC1. | |
| DM3 - Minus channel 3 | | ADC1_DM0/ADC0... | Input | ADC1_DM0/ADC0... | |
| DM26 - Minus channel 26 | | TempSensor_minus | | TempSensor_minus | TempS |
| DM27 - Minus channel 27 | | Bandgap_minus | | Bandgap_minus | Band |

Project View

Component Inspector

Editor View

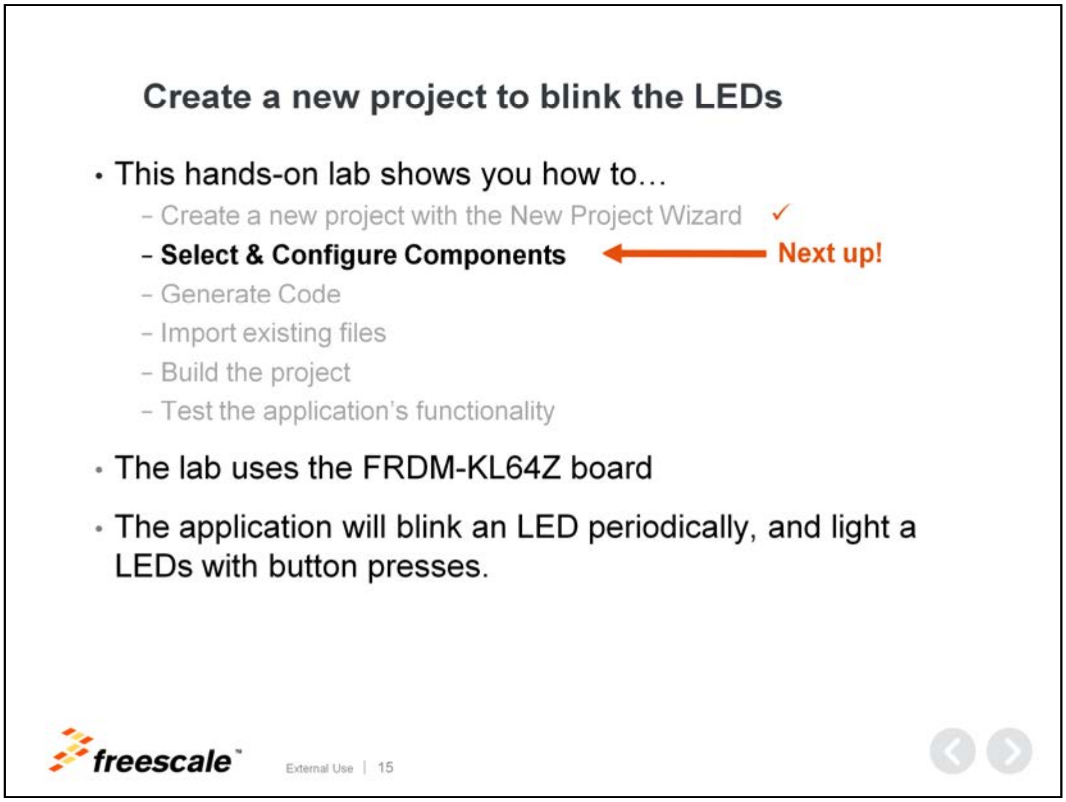
Component View

Problem View

freescale™ External Use | 14

After the New Project Wizard, you end up in the standard C/C++ perspective with the Processor Expert view also turned on. So, you see the Component Inspector and the Component View as defaults.

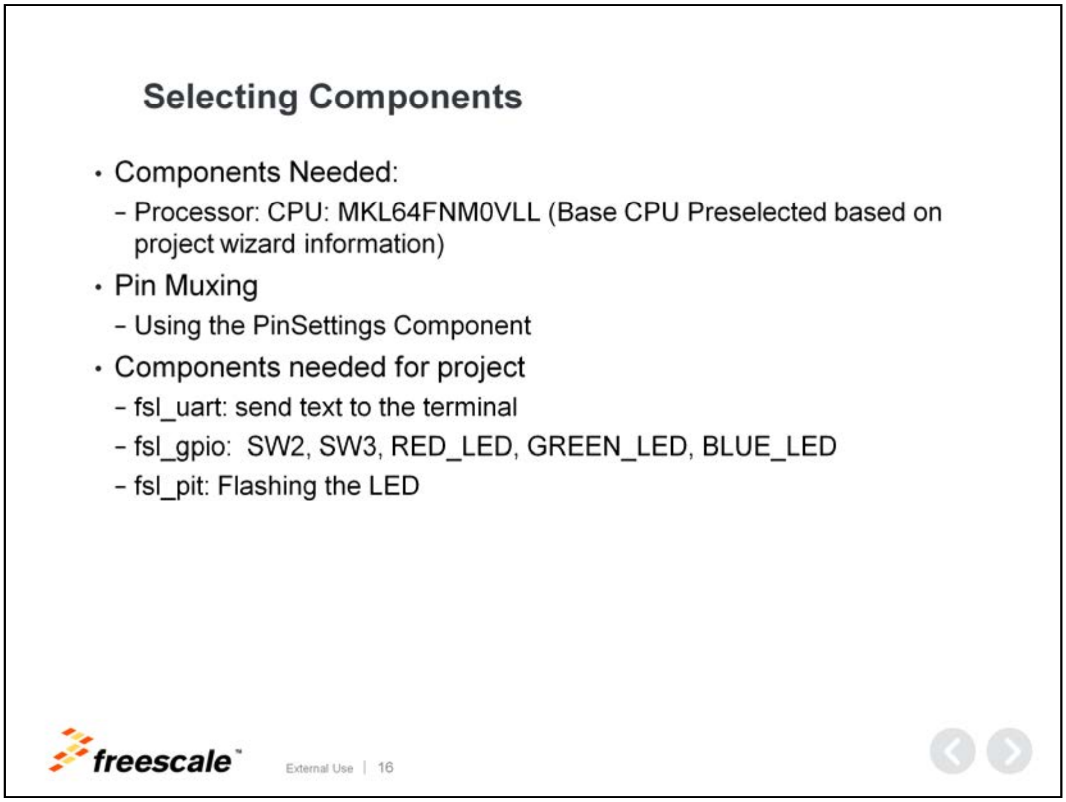
There may be errors in the “problems view”. These will be removed as we configure the part.



## Create a new project to blink the LEDs

- This hands-on lab shows you how to…
  - Create a new project with the New Project Wizard ✓
  - **Select & Configure Components** ← **Next up!**
  - Generate Code
  - Import existing files
  - Build the project
  - Test the application's functionality
- The lab uses the FRDM-KL64Z board
- The application will blink an LED periodically, and light a LEDs with button presses.

External Use | 15

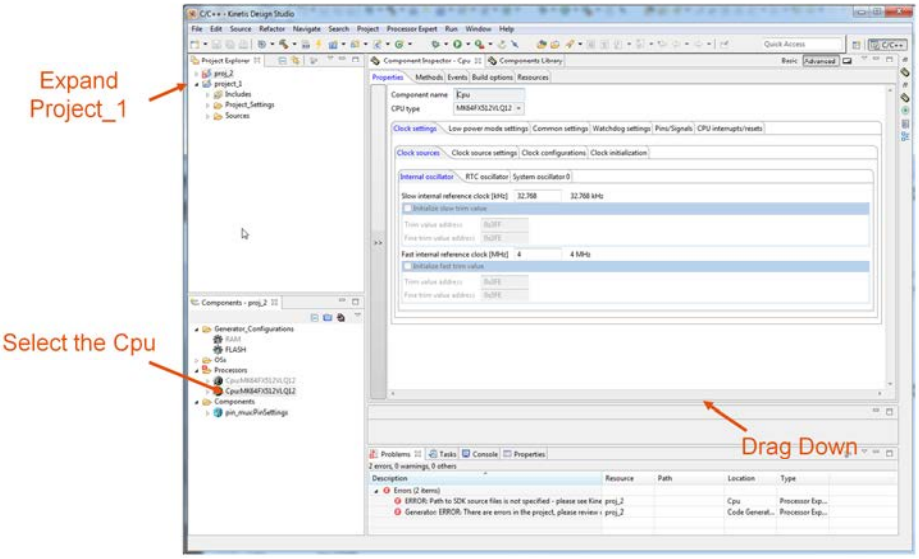


## Selecting Components

- Components Needed:
  - Processor: CPU: MKL64FNM0VLL (Base CPU Preselected based on project wizard information)
- Pin Muxing
  - Using the PinSettings Component
- Components needed for project
  - fsl_uart: send text to the terminal
  - fsl_gpio:  SW2, SW3, RED_LED, GREEN_LED, BLUE_LED
  - fsl_pit: Flashing the LED

External Use | 16

## Component Inspector



Expand Project_1

Select the Cpu

Drag Down

Component name: Cpu
CPU type: MK64FX512VLQ12

Slow internal reference clock [kHz] 32.768 32.768 kHz
□ Initialize slow trim value
Trim value address 0x3FF
Fine trim value address 0x3FE

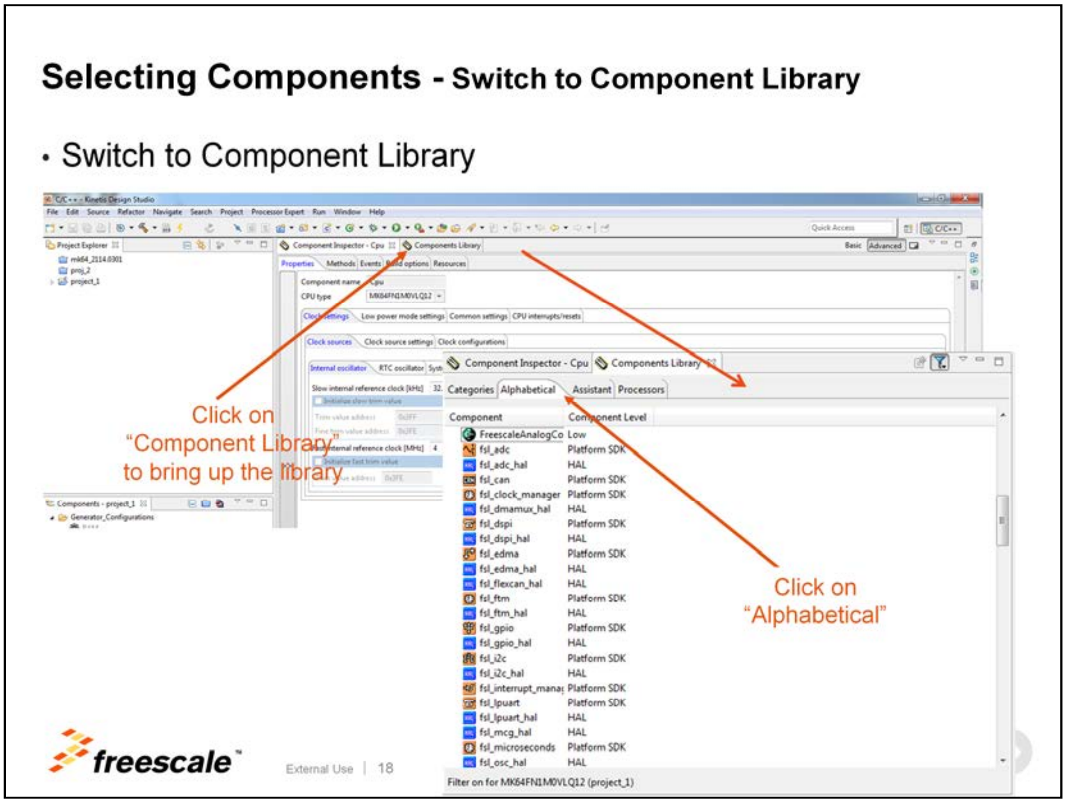
Fast internal reference clock [MHz] 4 4 MHz
□ Initialize fast trim value
Trim value address 0x3FE
Fine trim value address 0x3FE

2 errors, 0 warnings, 0 others

| Description | Resource | Path | Location | Type |
|---|---|---|---|---|
| Errors (2 items) | | | | |
| ERROR: Path to SDK source files is not specified - please see Kine | proj_2 | | Cpu | Processor Exp... |
| Generator: ERROR: There are errors in the project, please review | proj_2 | | Code Generat... | Processor Exp... |

External Use | 17

Select the CPU and Open up the window a bit for better visibility.



## Selecting Components - Switch to Component Library

- Switch to Component Library

Click on "Component Library" to bring up the library

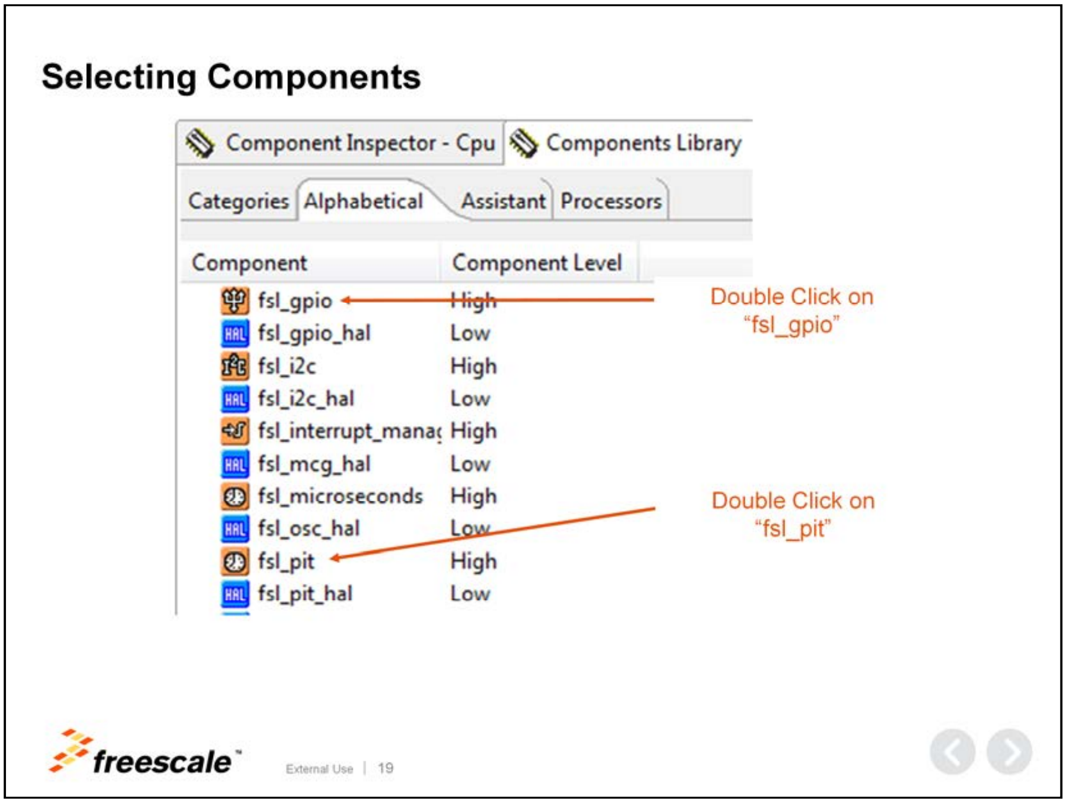
Click on "Alphabetical"

| Component | Component Level |
|---|---|
| FreescaleAnalogCo | Low |
| fsl_adc | Platform SDK |
| fsl_adc_hal | HAL |
| fsl_can | Platform SDK |
| fsl_clock_manager | Platform SDK |
| fsl_dmamux_hal | HAL |
| fsl_dspi | Platform SDK |
| fsl_dspi_hal | HAL |
| fsl_edma | Platform SDK |
| fsl_edma_hal | HAL |
| fsl_flexcan_hal | HAL |
| fsl_ftm | Platform SDK |
| fsl_ftm_hal | HAL |
| fsl_gpio | Platform SDK |
| fsl_gpio_hal | HAL |
| fsl_i2c | Platform SDK |
| fsl_i2c_hal | HAL |
| fsl_interrupt_mana | Platform SDK |
| fsl_lpuart | Platform SDK |
| fsl_lpuart_hal | HAL |
| fsl_mcg_hal | HAL |
| fsl_microseconds | Platform SDK |
| fsl_osc_hal | HAL |

Filter on for MK64FN1M0VLQ12 (project_1)

freescale External Use | 18

Here in the component library, we select the components needed for our project.



## Selecting Components

| Component | Component Level |
|---|---|
| fsl_gpio | High |
| fsl_gpio_hal | Low |
| fsl_i2c | High |
| fsl_i2c_hal | Low |
| fsl_interrupt_manag | High |
| fsl_mcg_hal | Low |
| fsl_microseconds | High |
| fsl_osc_hal | Low |
| fsl_pit | High |
| fsl_pit_hal | Low |

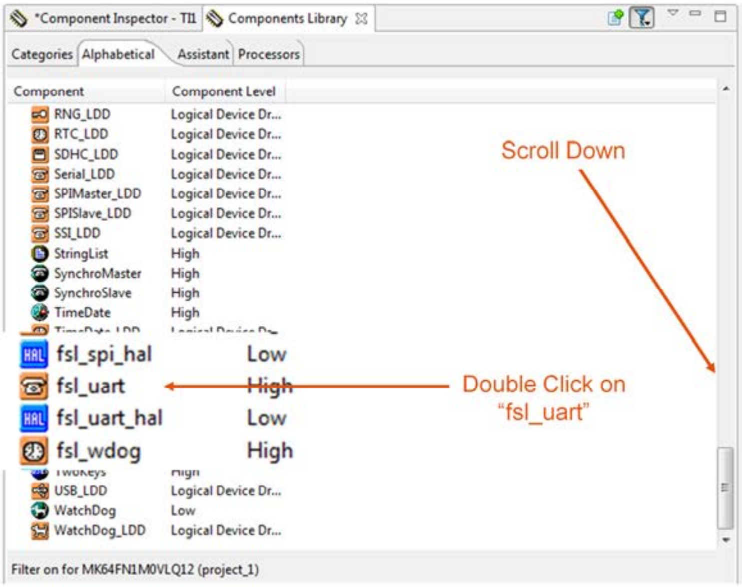
Double Click on “fsl_gpio”

Double Click on “fsl_pit”

We are pulling in the components here. One for the switches and LED’s, one for the interrupt timer, and one for the serial port.

An alternative to “double click” on the component, one could right click and select “add to project”..

## Selecting Components



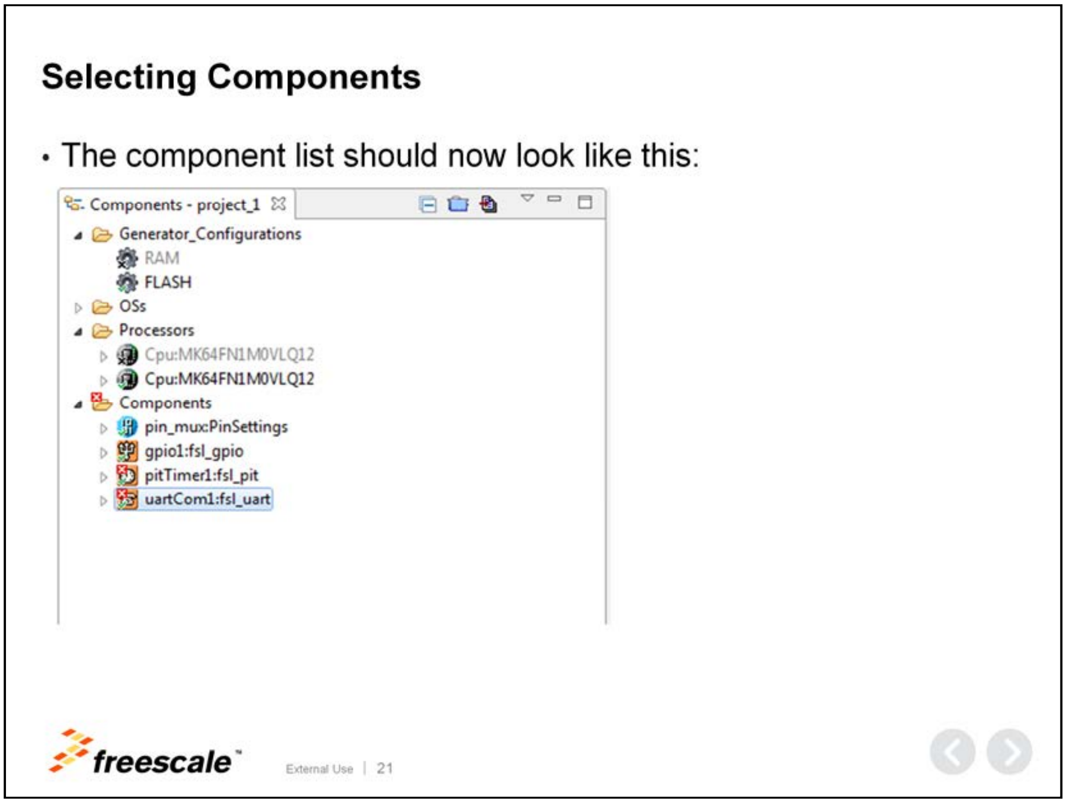
| Component | Component Level |
|---|---|
| RNG_LDD | Logical Device Dr... |
| RTC_LDD | Logical Device Dr... |
| SDHC_LDD | Logical Device Dr... |
| Serial_LDD | Logical Device Dr... |
| SPIMaster_LDD | Logical Device Dr... |
| SPISlave_LDD | Logical Device Dr... |
| SSI_LDD | Logical Device Dr... |
| StringList | High |
| SynchroMaster | High |
| SynchroSlave | High |
| TimeDate | High |
| TimeDate_LDD | Logical Device Dr... |
| fsl_spi_hal | Low |
| fsl_uart | High |
| fsl_uart_hal | Low |
| fsl_wdog | High |
| TwoKeys | High |
| USB_LDD | Logical Device Dr... |
| WatchDog | Low |
| WatchDog_LDD | Logical Device Dr... |

Filter on for MK64FN1M0VLQ12 (project_1)

Scroll Down

Double Click on "fsl_uart"

Scroll down and select the fsl_uart.



## Selecting Components

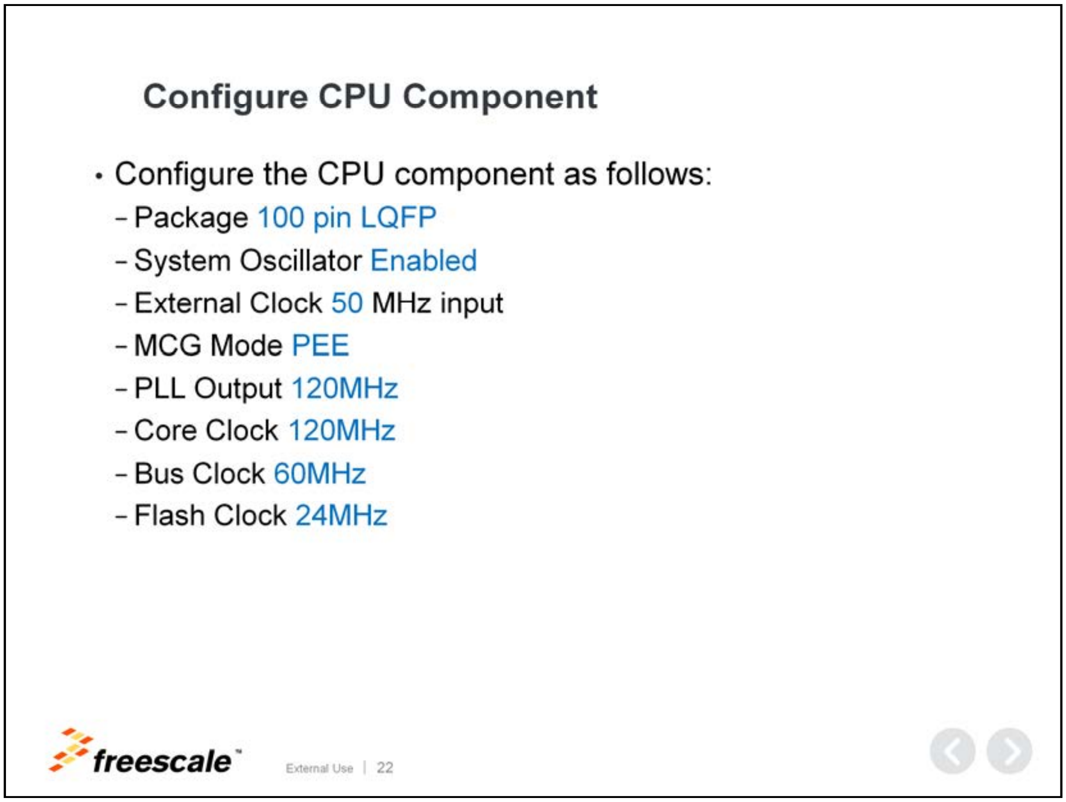
- The component list should now look like this:

Components - project_1

- Generator_Configurations
  - RAM
  - FLASH
- OSs
- Processors
  - Cpu:MK64FN1M0VLQ12
  - Cpu:MK64FN1M0VLQ12
- Components
  - pin_mux:PinSettings
  - gpio1:fsl_gpio
  - pitTimer1:fsl_pit
  - uartCom1:fsl_uart

External Use | 21

With all our components added to the project, we will move on to configuring the components. Starting with the CPU. Notice there are errors in the components. This is normal behavior, since the components have not been configured just yet.

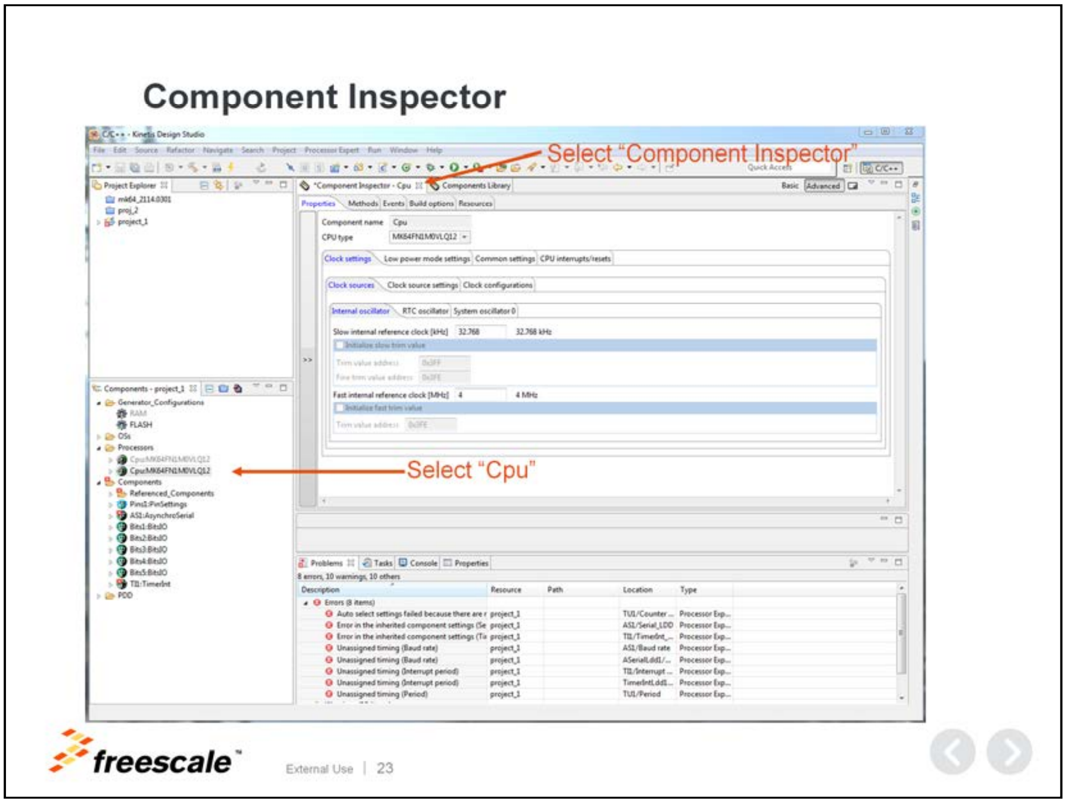


## Configure CPU Component

- Configure the CPU component as follows:
  - Package 100 pin LQFP
  - System Oscillator Enabled
  - External Clock 50 MHz input
  - MCG Mode PEE
  - PLL Output 120MHz
  - Core Clock 120MHz
  - Bus Clock 60MHz
  - Flash Clock 24MHz

External Use | 22

These are the settings we will use for the next set of slides. This is the hardware spec that we will be setting the chip up for.



## Component Inspector

Select "Component Inspector"

Select "Cpu"

*Component Inspector - Cpu

Component name: Cpu

CPU type: MK64FN1M0VLQ12

Slow internal reference clock [kHz]: 32.768 — 32.768 kHz

□ Initialize slow trim value

Trim value address: 0x3FF

Fine trim value address: 0x3FE

Fast internal reference clock [MHz]: 4 — 4 MHz

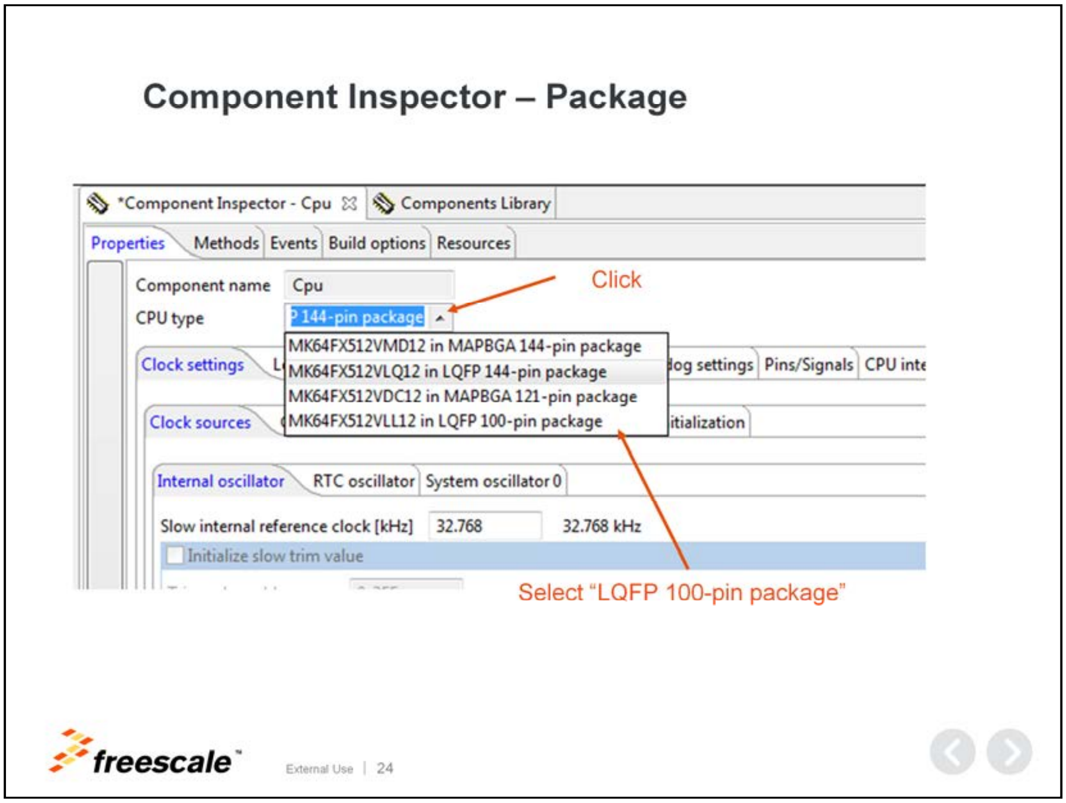
□ Initialize fast trim value

Trim value address: 0x3FE

8 errors, 10 warnings, 10 others

| Description | Resource | Path | Location | Type |
|---|---|---|---|---|
| Errors (8 items) | | | | |
| Auto select settings failed because there are r | project_1 | | TU1/Counter ... | Processor Exp... |
| Error in the inherited component settings (Se | project_1 | | AS1/Serial_LDD | Processor Exp... |
| Error in the inherited component settings (Ti | project_1 | | TI1/TimerInt_... | Processor Exp... |
| Unassigned timing (Baud rate) | project_1 | | AS1/Baud rate | Processor Exp... |
| Unassigned timing (Baud rate) | project_1 | | ASerialLdd1/... | Processor Exp... |
| Unassigned timing (Interrupt period) | project_1 | | TI1/Interrupt ... | Processor Exp... |
| Unassigned timing (Interrupt period) | project_1 | | TimerIntLdd1... | Processor Exp... |
| Unassigned timing (Period) | project_1 | | TU1/Period | Processor Exp... |

When looking at this perspective, notice the error messages in the "Problems" view. These problems will be rectified as we configure the different components.



## Component Inspector – Package

*Component Inspector - Cpu | Components Library

Properties | Methods | Events | Build options | Resources

Component name: Cpu

CPU type: P 144-pin package

- MK64FX512VMD12 in MAPBGA 144-pin package
- MK64FX512VLQ12 in LQFP 144-pin package
- MK64FX512VDC12 in MAPBGA 121-pin package
- MK64FX512VLL12 in LQFP 100-pin package

Clock settings | L… | …dog settings | Pins/Signals | CPU inte…

Clock sources | … | …itialization

Internal oscillator | RTC oscillator | System oscillator 0

Slow internal reference clock [kHz]: 32.768 — 32.768 kHz

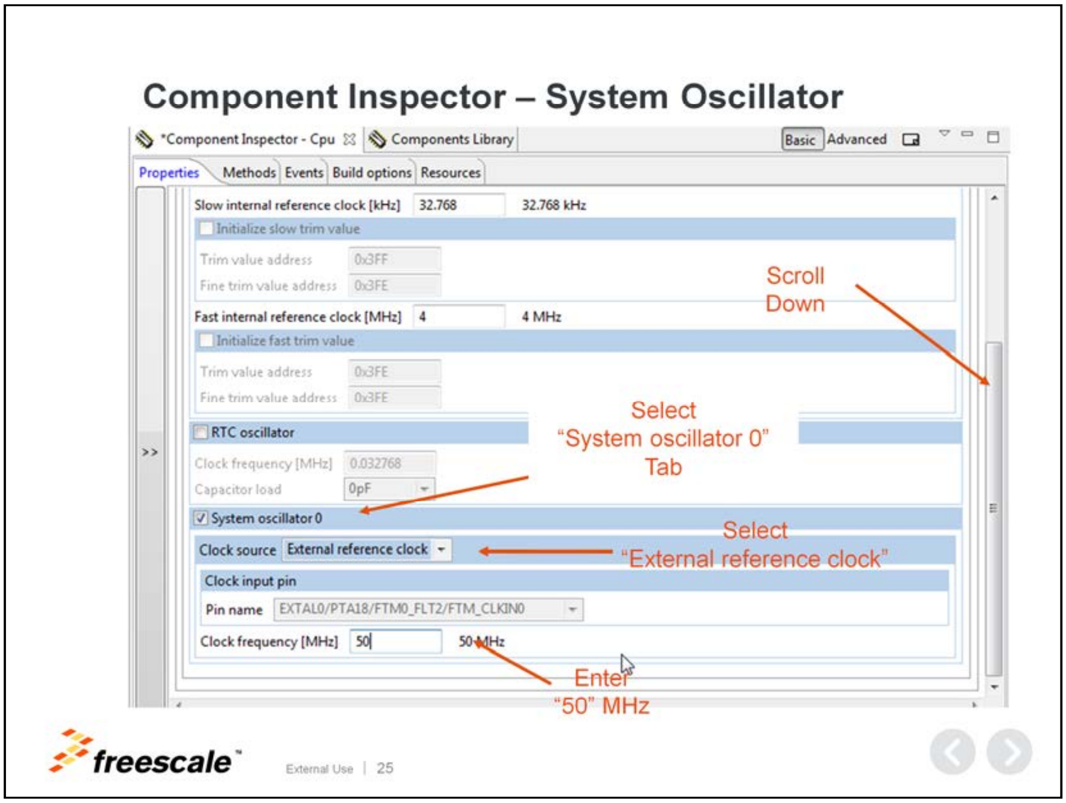
□ Initialize slow trim value

Click

Select "LQFP 100-pin package"

freescale™ External Use | 24

Selecting the "package" of the chip. Remember that each chip has several packages, selecting the proper package here is vitally important to have the proper pin muxing.



## Component Inspector – System Oscillator

Slow internal reference clock [kHz] 32.768 32.768 kHz

☐ Initialize slow trim value

Trim value address 0x3FF

Fine trim value address 0x3FE

Fast internal reference clock [MHz] 4 4 MHz

☐ Initialize fast trim value

Trim value address 0x3FE

Fine trim value address 0x3FE

☐ RTC oscillator

Clock frequency [MHz] 0.032768

Capacitor load 0pF

☑ System oscillator 0

Clock source External reference clock

Clock input pin

Pin name EXTAL0/PTA18/FTM0_FLT2/FTM_CLKIN0

Clock frequency [MHz] 50 50 MHz

Scroll Down

Select "System oscillator 0" Tab

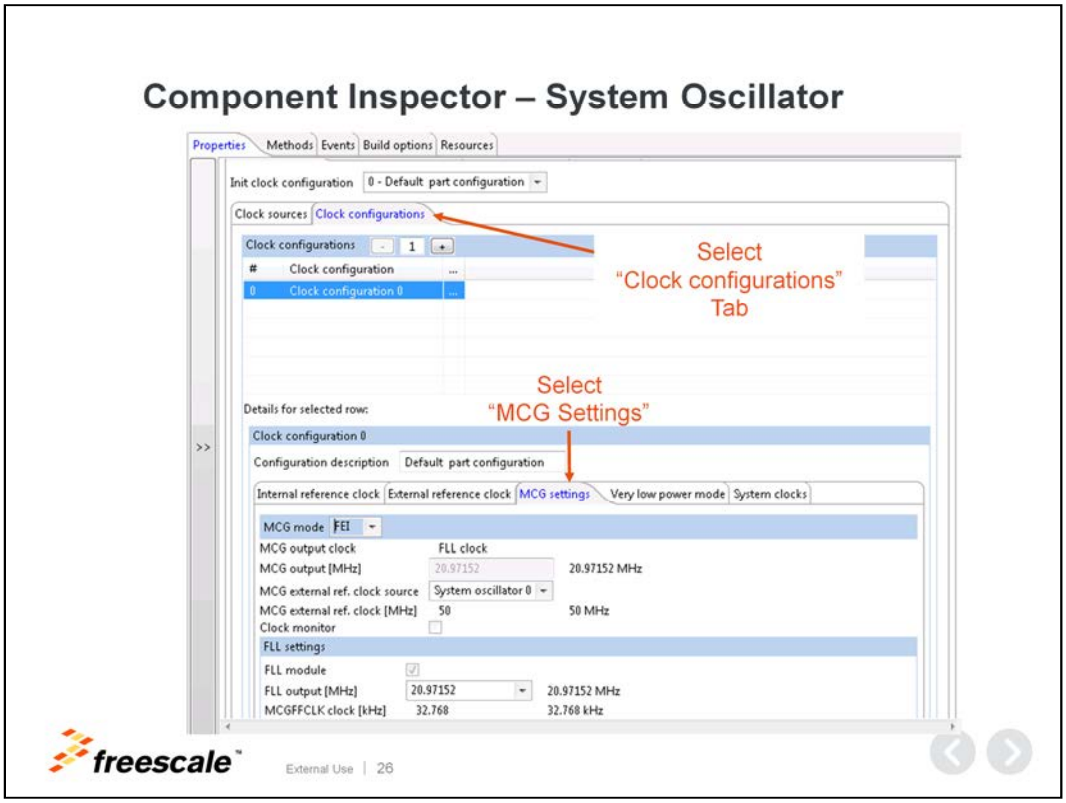
Select "External reference clock"

Enter "50" MHz

External Use | 25

Based on the hardware design, the system has a 50MHz clock source, not a crystal.

## Component Inspector – System Oscillator



Init clock configuration: 0 - Default part configuration

Clock configurations: 1

| # | Clock configuration |
|---|---|
| 0 | Clock configuration 0 |

Select "Clock configurations" Tab

Select "MCG Settings"

Details for selected row:

Clock configuration 0

Configuration description: Default part configuration

MCG mode: FEI

| | | |
|---|---|---|
| MCG output clock | FLL clock | |
| MCG output [MHz] | 20.97152 | 20.97152 MHz |
| MCG external ref. clock source | System oscillator 0 | |
| MCG external ref. clock [MHz] | 50 | 50 MHz |
| Clock monitor | □ | |

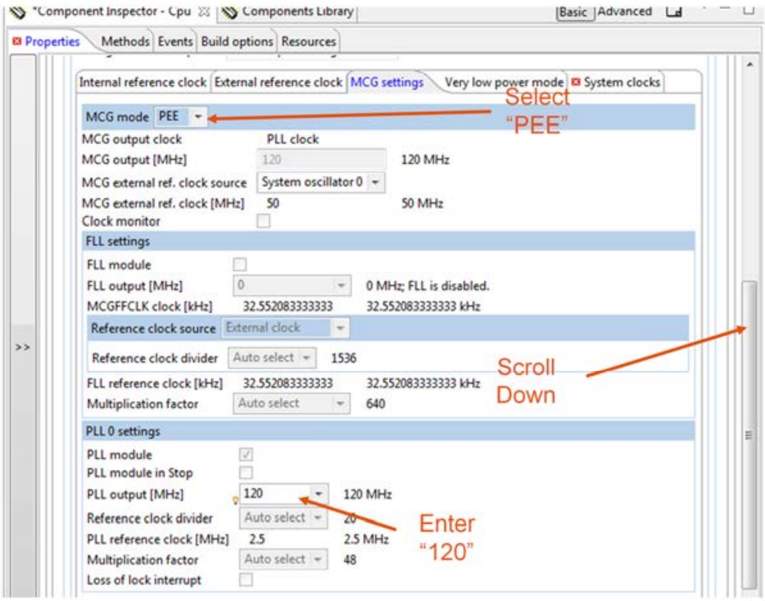
FLL settings

| | | |
|---|---|---|
| FLL module | ☑ | |
| FLL output [MHz] | 20.97152 | 20.97152 MHz |
| MCGFFCLK clock [kHz] | 32.768 | 32.768 kHz |

External Use | 26

The input clock is set for 50MHz and now we will setup the clock configurations MCG

## Component Inspector – System Oscillator



MCG mode: PEE
MCG output clock: PLL clock
MCG output [MHz]: 120 — 120 MHz
MCG external ref. clock source: System oscillator 0
MCG external ref. clock [MHz]: 50 — 50 MHz
Clock monitor: □

FLL settings
FLL module: □
FLL output [MHz]: 0 — 0 MHz; FLL is disabled.
MCGFFCLK clock [kHz]: 32.552083333333 — 32.552083333333 kHz
Reference clock source: External clock
Reference clock divider: Auto select — 1536
FLL reference clock [kHz]: 32.552083333333 — 32.552083333333 kHz
Multiplication factor: Auto select — 640

PLL 0 settings
PLL module: ☑
PLL module in Stop: □
PLL output [MHz]: 120 — 120 MHz
Reference clock divider: Auto select — 20
PLL reference clock [MHz]: 2.5 — 2.5 MHz
Multiplication factor: Auto select — 48
Loss of lock interrupt: □

Select "PEE"

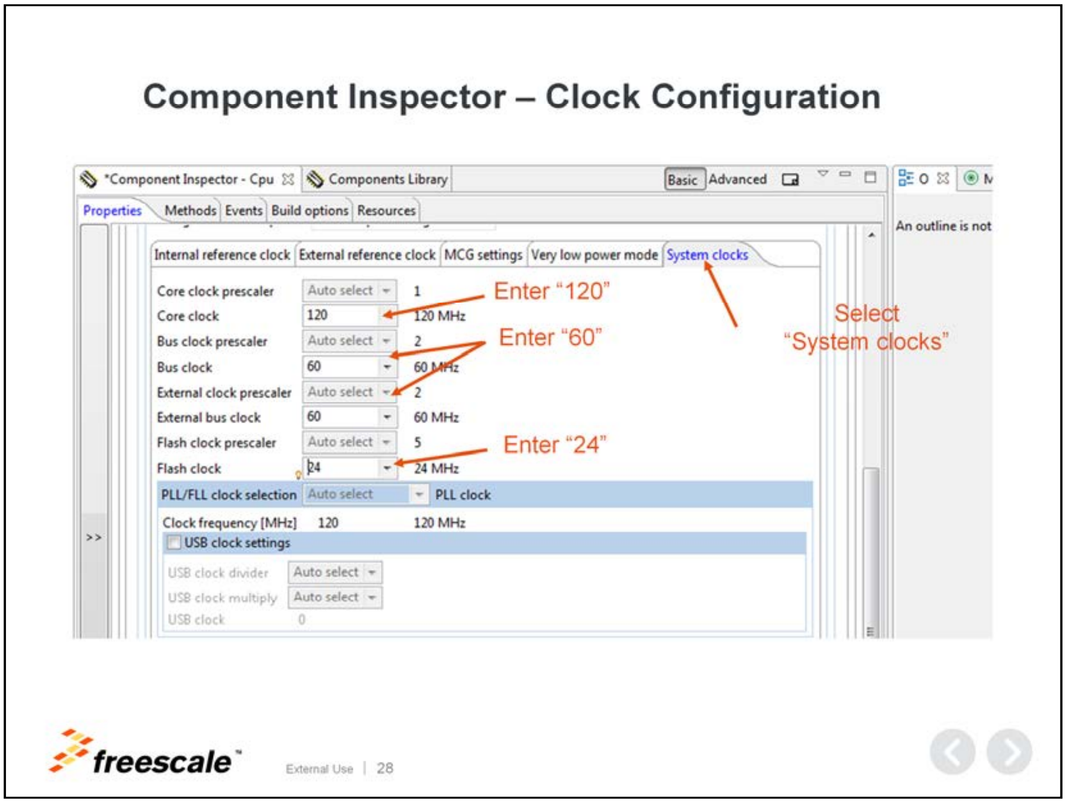
Scroll Down

Enter "120"

External Use | 27

Here we are setting up the MCG. The board has an external 50MHz signal, and so to operate at maximum speed of the chip, the us of the PLL clock is required. So, select PEE, and set the PLL output to 120MHz.

Scroll down and select “Clock configurations” tab



## Component Inspector – Clock Configuration

| | | |
|---|---|---|
| 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 | 2 |
| External bus clock | 60 | 60 MHz |
| Flash clock prescaler | Auto select | 5 |
| Flash clock | 24 | 24 MHz |
| PLL/FLL clock selection | Auto select | PLL clock |
| Clock frequency [MHz] | 120 | 120 MHz |
| □ USB clock settings | | |
| USB clock divider | Auto select | |
| USB clock multiply | Auto select | |
| USB clock | 0 | |

Enter “120”

Enter “60”

Enter “24”

Select “System clocks”

External Use | 28

With the PLL set to 120, the system clocks need to be addressed.

Core Clock = 120MHz

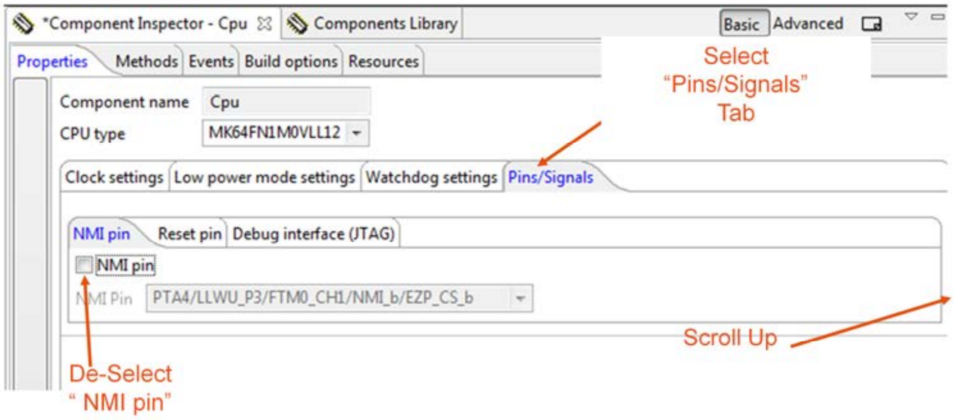
Bus Clock = 60MHz

External bus Clock = 60MHz

Flash Clock = 24MHz

This completes the clocking configuration. Next we move onto other CPU settings.

## Component Inspector – CPU Common Settings



Component name Cpu

CPU type MK64FN1M0VLL12

□ NMI pin

NMI Pin PTA4/LLWU_P3/FTM0_CH1/NMI_b/EZP_CS_b

Select "Pins/Signals" Tab

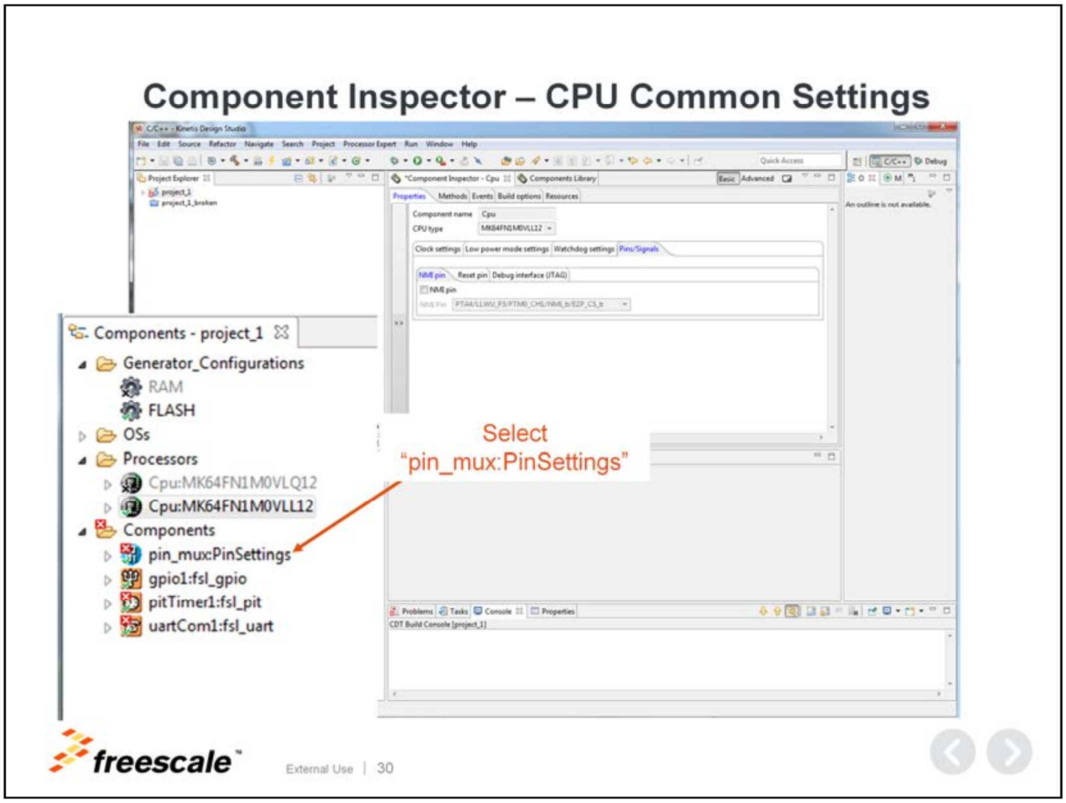
De-Select " NMI pin"

Scroll Up

External Use | 29

Since our project does not require an nmi interrupt routine, we will uncheck this box. Also, if you are using the nmi pin as a signal pin, like the FRDM board.

This completes the configuration of the cpu. Now its time to move onto the component configurations.

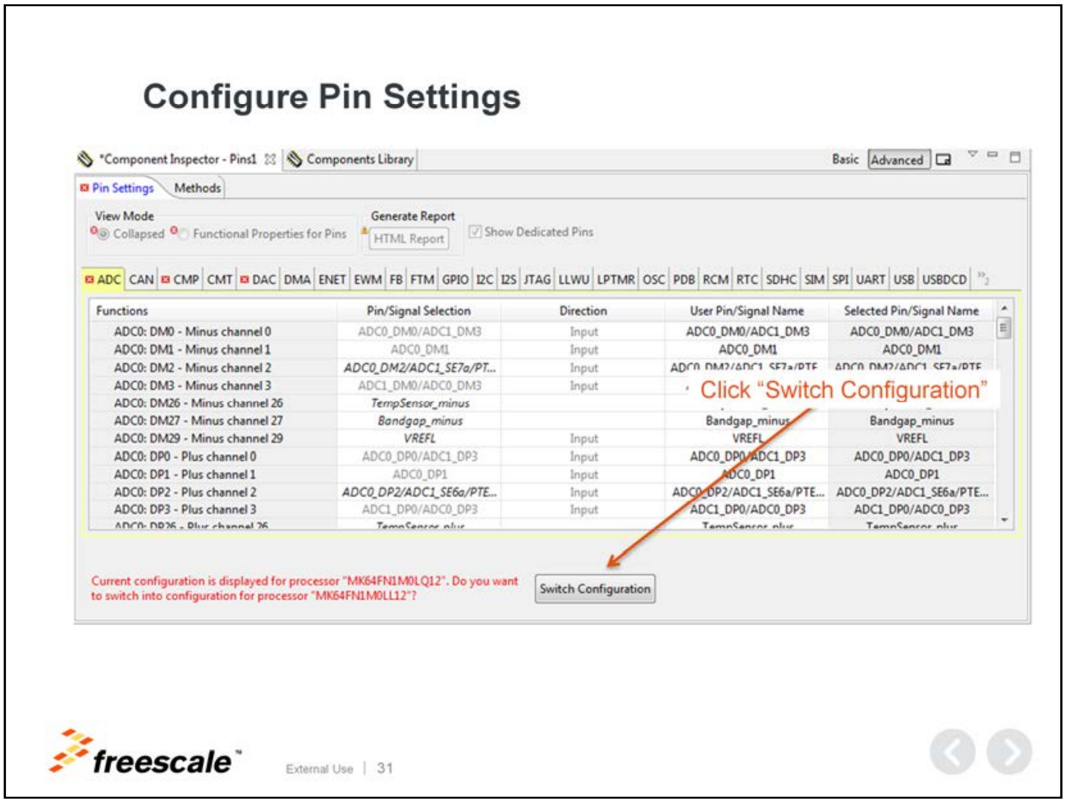


## Component Inspector – CPU Common Settings

Component name: Cpu
CPU type: MK64FN1M0VLL12

Select “pin_mux:PinSettings”

Now select the “pin_mux:PinSettings” from the component view



## Configure Pin Settings

*Component Inspector - Pins1
Components Library
Basic
Advanced
Pin Settings
Methods
View Mode
Collapsed
Functional Properties for Pins
Generate Report
HTML Report
☑ Show Dedicated Pins
ADC | CAN | CMP | CMT | DAC | DMA | ENET | EWM | FB | FTM | GPIO | I2C | I2S | JTAG | LLWU | LPTMR | OSC | PDB | RCM | RTC | SDHC | SIM | SPI | UART | USB | USBDCD

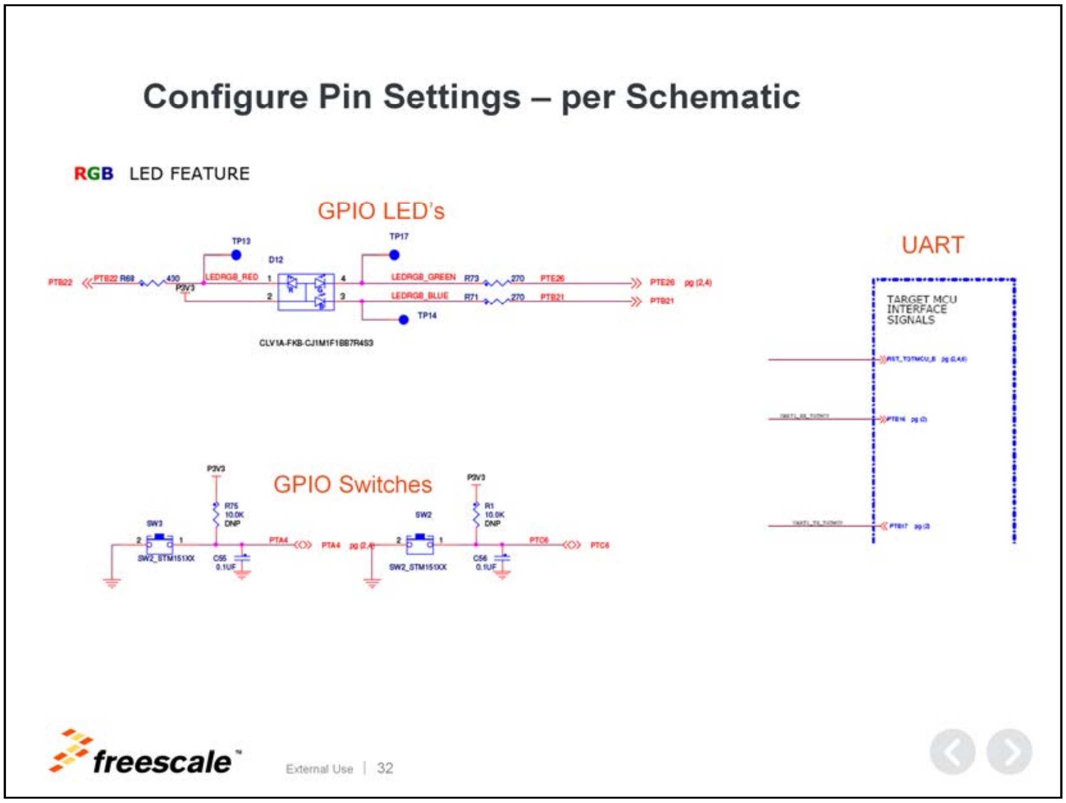
| Functions | Pin/Signal Selection | Direction | User Pin/Signal Name | Selected Pin/Signal Name |
|---|---|---|---|---|
| ADC0: DM0 - Minus channel 0 | ADC0_DM0/ADC1_DM3 | Input | ADC0_DM0/ADC1_DM3 | ADC0_DM0/ADC1_DM3 |
| ADC0: DM1 - Minus channel 1 | ADC0_DM1 | Input | ADC0_DM1 | ADC0_DM1 |
| ADC0: DM2 - Minus channel 2 | ADC0_DM2/ADC1_SE7a/PT... | Input | ADC0_DM2/ADC1_SE7a/PTE | ADC0_DM2/ADC1_SE7a/PTE |
| ADC0: DM3 - Minus channel 3 | ADC1_DM0/ADC0_DM3 | Input | | |
| ADC0: DM26 - Minus channel 26 | TempSensor_minus | | | |
| ADC0: DM27 - Minus channel 27 | Bandgap_minus | | Bandgap_minus | Bandgap_minus |
| ADC0: DM29 - Minus channel 29 | VREFL | Input | VREFL | VREFL |
| ADC0: DP0 - Plus channel 0 | ADC0_DP0/ADC1_DP3 | Input | ADC0_DP0/ADC1_DP3 | ADC0_DP0/ADC1_DP3 |
| ADC0: DP1 - Plus channel 1 | ADC0_DP1 | Input | ADC0_DP1 | ADC0_DP1 |
| ADC0: DP2 - Plus channel 2 | ADC0_DP2/ADC1_SE6a/PTE... | Input | ADC0_DP2/ADC1_SE6a/PTE... | ADC0_DP2/ADC1_SE6a/PTE... |
| ADC0: DP3 - Plus channel 3 | ADC1_DP0/ADC0_DP3 | Input | ADC1_DP0/ADC0_DP3 | ADC1_DP0/ADC0_DP3 |
| ADC0: DP26 - Plus channel 26 | TempSensor_plus | | TempSensor_plus | TempSensor_plus |

Click "Switch Configuration"

Current configuration is displayed for processor "MK64FN1M0LQ12". Do you want to switch into configuration for processor "MK64FN1M0LL12"?

freescale External Use | 31

Since the package has changed from the default, the system lets you know the configuration will change.



## Configure Pin Settings – per Schematic

RGB LED FEATURE

GPIO LED's

GPIO Switches

UART

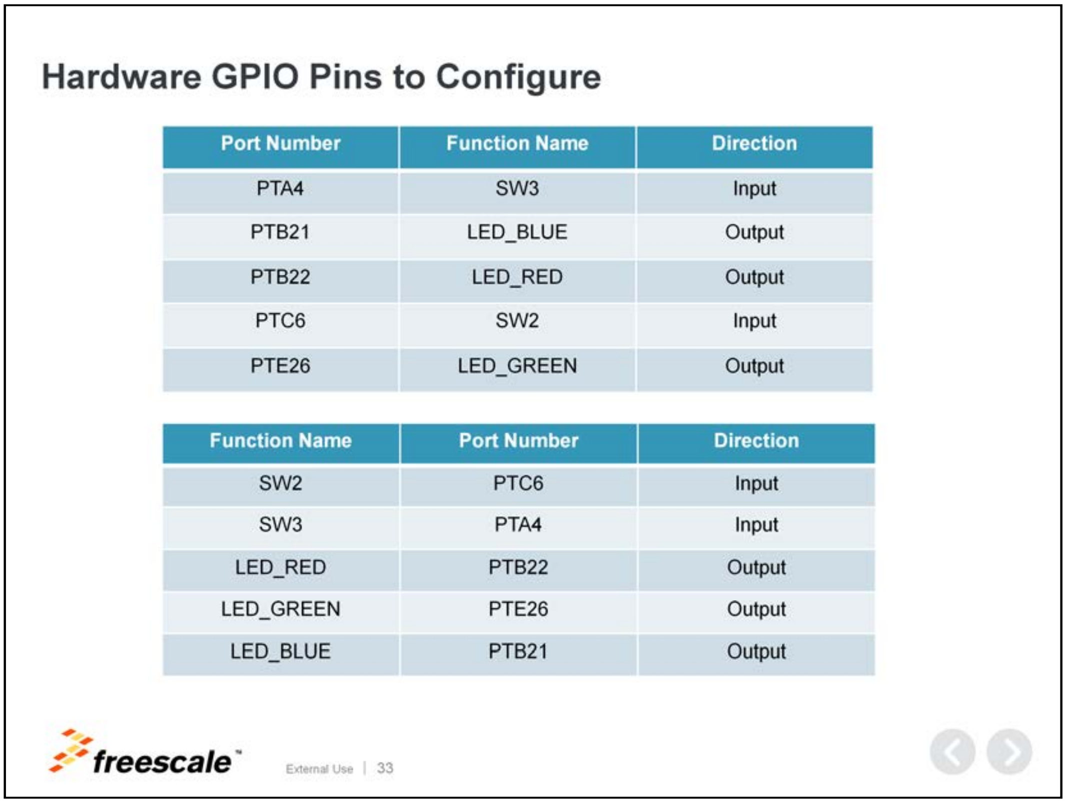
External Use | 32

Now we move into the pin settings. And our io's will be:

LED's are on ports PTB-21, PTB-22, and PTE-26

SWitches are on ports PTA-4, and PTC-6

UART is on PTB-16, and PTB-17



## Hardware GPIO Pins to Configure

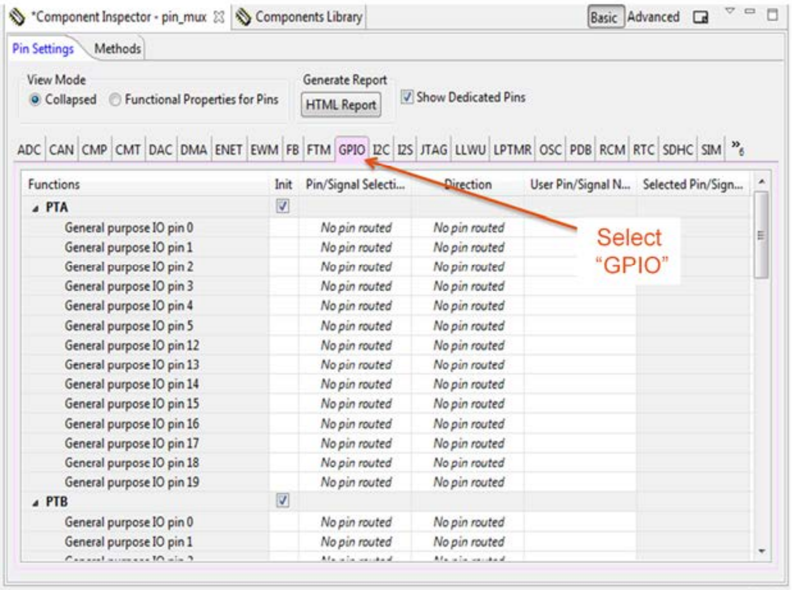
| Port Number | Function Name | Direction |
|---|---|---|
| PTA4 | SW3 | Input |
| PTB21 | LED_BLUE | Output |
| PTB22 | LED_RED | Output |
| PTC6 | SW2 | Input |
| PTE26 | LED_GREEN | Output |

| Function Name | Port Number | Direction |
|---|---|---|
| SW2 | PTC6 | Input |
| SW3 | PTA4 | Input |
| LED_RED | PTB22 | Output |
| LED_GREEN | PTE26 | Output |
| LED_BLUE | PTB21 | Output |

External Use | 33

This is a summary list of the pins we are going to use for this project.

## Configure Pin Settings



*Component Inspector - pin_mux ☒ | Components Library | Basic | Advanced

Pin Settings | Methods

View Mode: ● Collapsed ○ Functional Properties for Pins

Generate Report: HTML Report

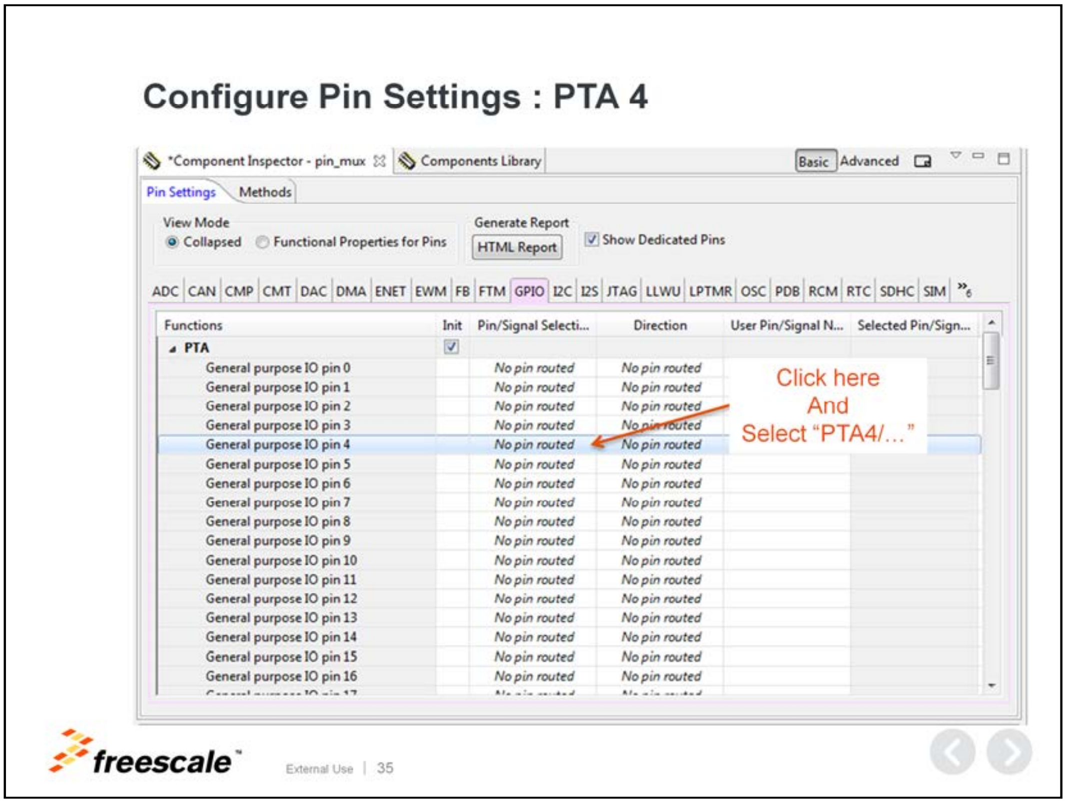
☑ Show Dedicated Pins

ADC | CAN | CMP | CMT | DAC | DMA | ENET | EWM | FB | FTM | GPIO | I2C | I2S | JTAG | LLWU | LPTMR | OSC | PDB | RCM | RTC | SDHC | SIM | »6

| Functions | Init | Pin/Signal Selecti... | Direction | User Pin/Signal N... | Selected Pin/Sign... |
|---|---|---|---|---|---|
| PTA | ☑ | | | | |
| General purpose IO pin 0 | | No pin routed | No pin routed | | |
| General purpose IO pin 1 | | No pin routed | No pin routed | | |
| General purpose IO pin 2 | | No pin routed | No pin routed | | |
| General purpose IO pin 3 | | No pin routed | No pin routed | | |
| General purpose IO pin 4 | | No pin routed | No pin routed | | |
| General purpose IO pin 5 | | No pin routed | No pin routed | | |
| General purpose IO pin 12 | | No pin routed | No pin routed | | |
| General purpose IO pin 13 | | No pin routed | No pin routed | | |
| General purpose IO pin 14 | | No pin routed | No pin routed | | |
| General purpose IO pin 15 | | No pin routed | No pin routed | | |
| General purpose IO pin 16 | | No pin routed | No pin routed | | |
| General purpose IO pin 17 | | No pin routed | No pin routed | | |
| General purpose IO pin 18 | | No pin routed | No pin routed | | |
| General purpose IO pin 19 | | No pin routed | No pin routed | | |
| PTB | ☑ | | | | |
| General purpose IO pin 0 | | No pin routed | No pin routed | | |
| General purpose IO pin 1 | | No pin routed | No pin routed | | |

Select "GPIO"

Now select the GPIO tab.



## Configure Pin Settings : PTA 4

*Component Inspector - pin_mux
Components Library
Basic
Advanced
Pin Settings
Methods
ADC CAN CMP CMT DAC DMA ENET EWM FB FTM GPIO I2C I2S JTAG LLWU LPTMR OSC PDB RCM RTC SDHC SIM

View Mode ● Collapsed ○ Functional Properties for Pins

Generate Report [HTML Report]

☑ Show Dedicated Pins

| Functions | Init | Pin/Signal Selecti... | Direction | User Pin/Signal N... | Selected Pin/Sign... |
|---|---|---|---|---|---|
| PTA | ☑ | | | | |
| General purpose IO pin 0 | | No pin routed | No pin routed | | |
| General purpose IO pin 1 | | No pin routed | No pin routed | | |
| General purpose IO pin 2 | | No pin routed | No pin routed | | |
| General purpose IO pin 3 | | No pin routed | No pin routed | | |
| General purpose IO pin 4 | | No pin routed | No pin routed | | |
| General purpose IO pin 5 | | No pin routed | No pin routed | | |
| General purpose IO pin 6 | | No pin routed | No pin routed | | |
| General purpose IO pin 7 | | No pin routed | No pin routed | | |
| General purpose IO pin 8 | | No pin routed | No pin routed | | |
| General purpose IO pin 9 | | No pin routed | No pin routed | | |
| General purpose IO pin 10 | | No pin routed | No pin routed | | |
| General purpose IO pin 11 | | No pin routed | No pin routed | | |
| General purpose IO pin 12 | | No pin routed | No pin routed | | |
| General purpose IO pin 13 | | No pin routed | No pin routed | | |
| General purpose IO pin 14 | | No pin routed | No pin routed | | |
| General purpose IO pin 15 | | No pin routed | No pin routed | | |
| General purpose IO pin 16 | | No pin routed | No pin routed | | |

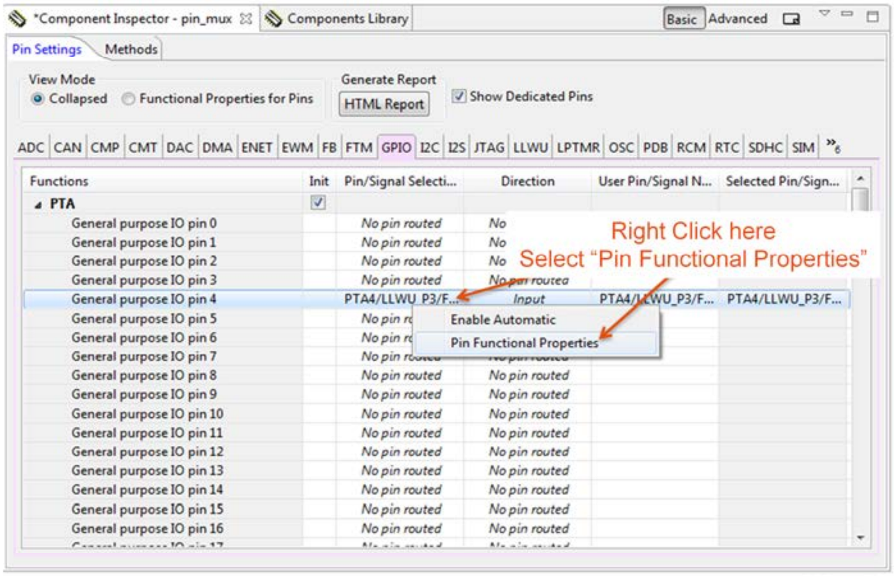
Click here
And
Select “PTA4/…”

External Use | 35

Summary of pins

| Port Number | Function Name | Direction |
|---|---|---|
| PTA4 | SW3 | Input |
| PTB21 | LED_BLUE | Output |
| PTB22 | LED_RED | Output |
| PTC6 | SW2 | Input |
| PTE26 | LED_GREEN | Output |

## Configure Pin Settings : PTA 4

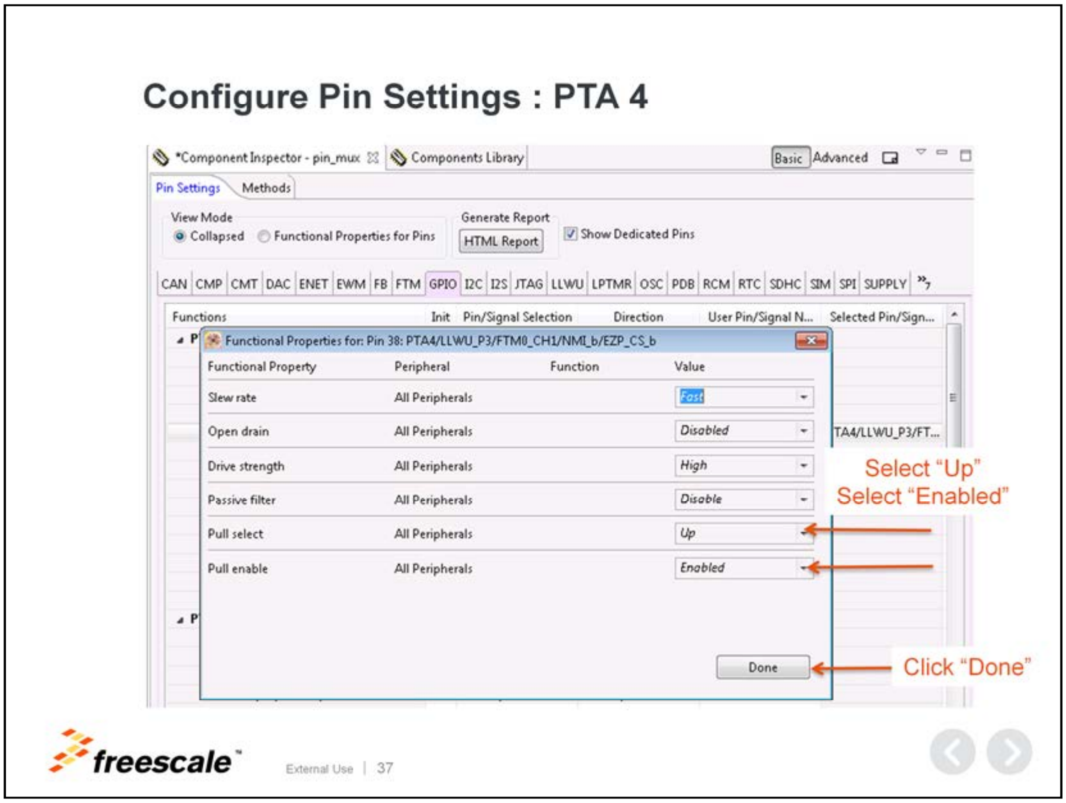

*Component Inspector - pin_mux
Components Library
Basic
Advanced
Pin Settings
Methods
View Mode
● Collapsed ○ Functional Properties for Pins
Generate Report
HTML Report
☑ Show Dedicated Pins
ADC CAN CMP CMT DAC DMA ENET EWM FB FTM GPIO I2C I2S JTAG LLWU LPTMR OSC PDB RCM RTC SDHC SIM »6
Enable Automatic
Pin Functional Properties

| Functions | Init | Pin/Signal Selecti... | Direction | User Pin/Signal N... | Selected Pin/Sign... |
|---|---|---|---|---|---|
| PTA | ☑ | | | | |
| General purpose IO pin 0 | | No pin routed | No | | |
| General purpose IO pin 1 | | No pin routed | No | | |
| General purpose IO pin 2 | | No pin routed | No | | |
| General purpose IO pin 3 | | No pin routed | No pin routed | | |
| General purpose IO pin 4 | | PTA4/LLWU_P3/F... | Input | PTA4/LLWU_P3/F... | PTA4/LLWU_P3/F... |
| General purpose IO pin 5 | | No pin r | | | |
| General purpose IO pin 6 | | No pin r | | | |
| General purpose IO pin 7 | | No pin routed | | | |
| General purpose IO pin 8 | | No pin routed | No pin routed | | |
| General purpose IO pin 9 | | No pin routed | No pin routed | | |
| General purpose IO pin 10 | | No pin routed | No pin routed | | |
| General purpose IO pin 11 | | No pin routed | No pin routed | | |
| General purpose IO pin 12 | | No pin routed | No pin routed | | |
| General purpose IO pin 13 | | No pin routed | No pin routed | | |
| General purpose IO pin 14 | | No pin routed | No pin routed | | |
| General purpose IO pin 15 | | No pin routed | No pin routed | | |
| General purpose IO pin 16 | | No pin routed | No pin routed | | |

Right Click here

Select "Pin Functional Properties"

Now we will configure the pin properties.



## Configure Pin Settings : PTA 4

*Component Inspector - pin_mux | Components Library | Basic | Advanced

Pin Settings | Methods

View Mode: ● Collapsed ○ Functional Properties for Pins

Generate Report: HTML Report

☑ Show Dedicated Pins

CAN | CMP | CMT | DAC | ENET | EWM | FB | FTM | GPIO | I2C | I2S | JTAG | LLWU | LPTMR | OSC | PDB | RCM | RTC | SDHC | SIM | SPI | SUPPLY

Functions | Init | Pin/Signal Selection | Direction | User Pin/Signal N... | Selected Pin/Sign...

TA4/LLWU_P3/FT...

Functional Properties for: Pin 38: PTA4/LLWU_P3/FTM0_CH1/NMI_b/EZP_CS_b

| Functional Property | Peripheral | Function | Value |
|---|---|---|---|
| Slew rate | All Peripherals | | Fast |
| Open drain | All Peripherals | | Disabled |
| Drive strength | All Peripherals | | High |
| Passive filter | All Peripherals | | Disable |
| Pull select | All Peripherals | | Up |
| Pull enable | All Peripherals | | Enabled |

Done

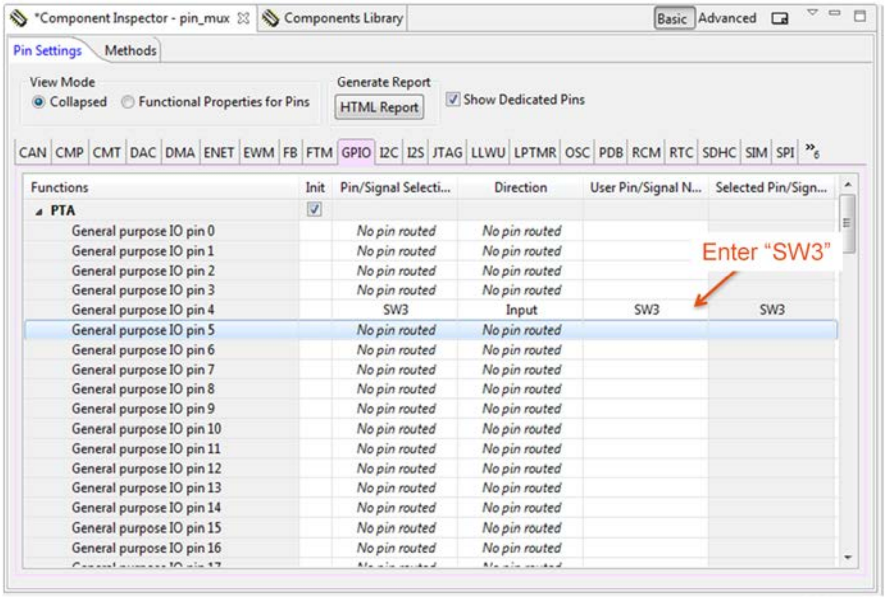
Select "Up"
Select "Enabled"

Click "Done"

External Use | 37

For the switch inputs, the switches require the use of the internal pull-up since there is no external pull-up on the schematic.

## Configure Pin Settings: PTA 4


*Component Inspector - pin_mux
Components Library
Basic
Advanced
Pin Settings
Methods
CAN CMP CMT DAC DMA ENET EWM FB FTM GPIO I2C I2S JTAG LLWU LPTMR OSC PDB RCM RTC SDHC SIM SPI »6

View Mode
● Collapsed ○ Functional Properties for Pins

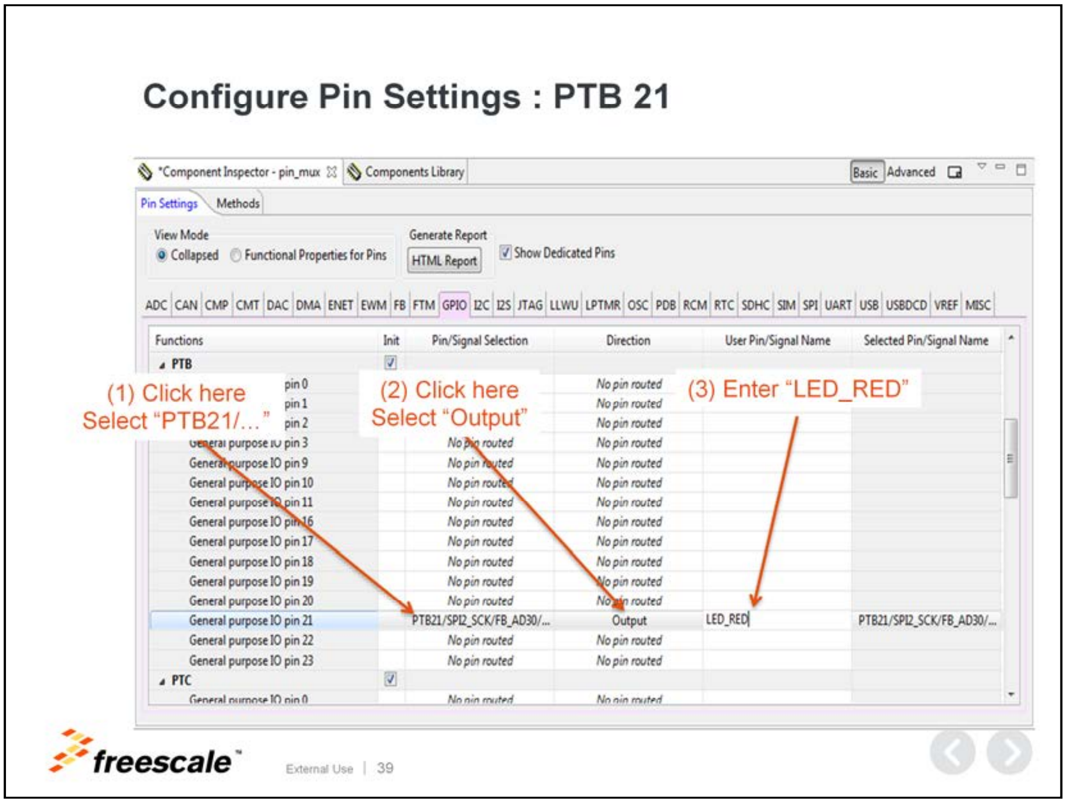
Generate Report
HTML Report

☑ Show Dedicated Pins

| Functions | Init | Pin/Signal Selecti... | Direction | User Pin/Signal N... | Selected Pin/Sign... |
|---|---|---|---|---|---|
| PTA | ☑ | | | | |
| General purpose IO pin 0 | | No pin routed | No pin routed | | |
| General purpose IO pin 1 | | No pin routed | No pin routed | | |
| General purpose IO pin 2 | | No pin routed | No pin routed | | |
| General purpose IO pin 3 | | No pin routed | No pin routed | | |
| General purpose IO pin 4 | | SW3 | Input | SW3 | SW3 |
| General purpose IO pin 5 | | No pin routed | No pin routed | | |
| General purpose IO pin 6 | | No pin routed | No pin routed | | |
| General purpose IO pin 7 | | No pin routed | No pin routed | | |
| General purpose IO pin 8 | | No pin routed | No pin routed | | |
| General purpose IO pin 9 | | No pin routed | No pin routed | | |
| General purpose IO pin 10 | | No pin routed | No pin routed | | |
| General purpose IO pin 11 | | No pin routed | No pin routed | | |
| General purpose IO pin 12 | | No pin routed | No pin routed | | |
| General purpose IO pin 13 | | No pin routed | No pin routed | | |
| General purpose IO pin 14 | | No pin routed | No pin routed | | |
| General purpose IO pin 15 | | No pin routed | No pin routed | | |
| General purpose IO pin 16 | | No pin routed | No pin routed | | |
| General purpose IO pin 17 | | No pin routed | No pin routed | | |

Enter “SW3”

In the User/Pin Signal Name column, you can enter names here that will be easier to remember or the same labels that are on your schematic.



## Configure Pin Settings : PTB 21

*Component Inspector - pin_mux
Components Library
Basic
Advanced
Pin Settings
Methods

View Mode ● Collapsed ○ Functional Properties for Pins

Generate Report: HTML Report

☑ Show Dedicated Pins

ADC CAN CMP CMT DAC DMA ENET EWM FB FTM GPIO I2C I2S JTAG LLWU LPTMR OSC PDB RCM RTC SDHC SIM SPI UART USB USBDCD VREF MISC

| Functions | Init | Pin/Signal Selection | Direction | User Pin/Signal Name | Selected Pin/Signal Name |
|---|---|---|---|---|---|
| PTB | ☑ | | | | |
| pin 0 | | | No pin routed | | |
| pin 1 | | | No pin routed | | |
| pin 2 | | | No pin routed | | |
| General purpose IO pin 3 | | No pin routed | No pin routed | | |
| General purpose IO pin 9 | | No pin routed | No pin routed | | |
| General purpose IO pin 10 | | No pin routed | No pin routed | | |
| General purpose IO pin 11 | | No pin routed | No pin routed | | |
| General purpose IO pin 16 | | No pin routed | No pin routed | | |
| General purpose IO pin 17 | | No pin routed | No pin routed | | |
| General purpose IO pin 18 | | No pin routed | No pin routed | | |
| General purpose IO pin 19 | | No pin routed | No pin routed | | |
| General purpose IO pin 20 | | No pin routed | No pin routed | | |
| General purpose IO pin 21 | | PTB21/SPI2_SCK/FB_AD30/... | Output | LED_RED | PTB21/SPI2_SCK/FB_AD30/... |
| General purpose IO pin 22 | | No pin routed | No pin routed | | |
| General purpose IO pin 23 | | No pin routed | No pin routed | | |
| PTC | ☑ | | | | |
| General purpose IO pin 0 | | No pin routed | No pin routed | | |

(1) Click here Select "PTB21/…"

(2) Click here Select "Output"

(3) Enter "LED_RED"

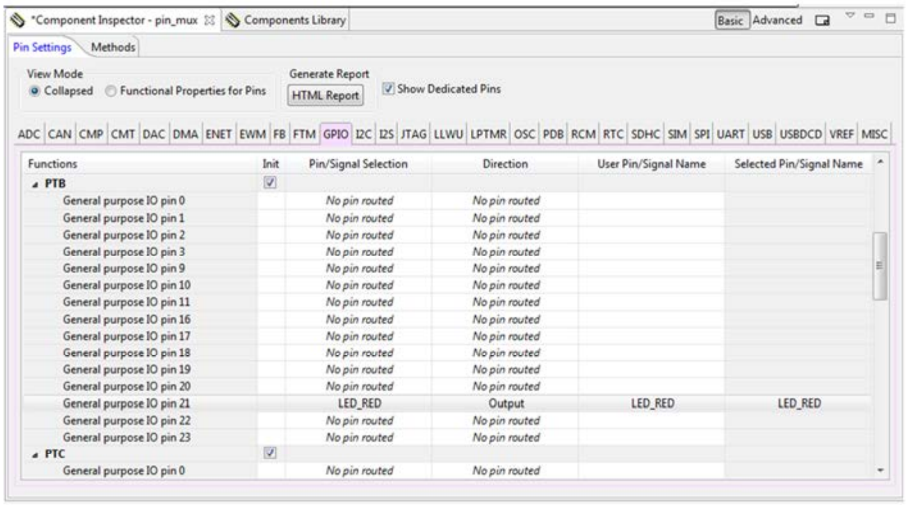
freescale External Use | 39

Setting the pin mux for LED_RED

Summary of pins

| Port Number | Function Name | Direction |
|---|---|---|
| PTA4 | SW3 | Input |
| PTB21 | LED_BLUE | Output |
| PTB22 | LED_RED | Output |
| PTC6 | SW2 | Input |
| PTE26 | LED_GREEN | Output |

## Configure Pin Settings : PTB 21



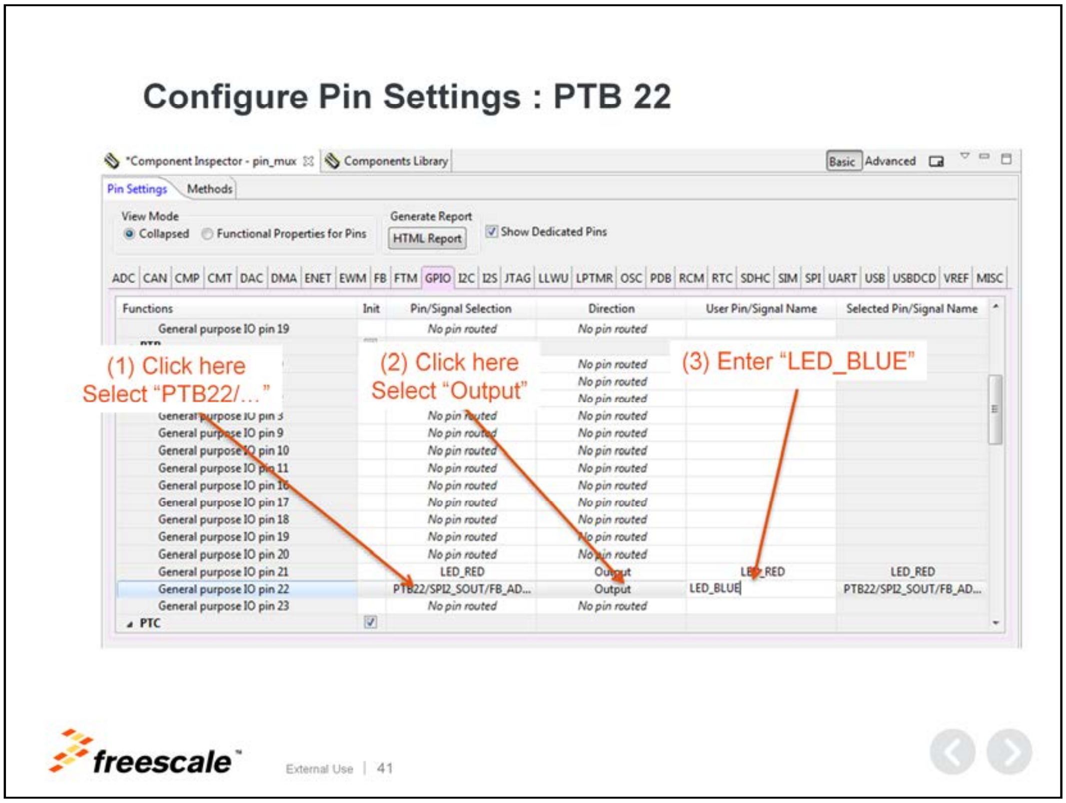
*Component Inspector - pin_mux
Components Library
Basic
Advanced
Pin Settings
Methods
View Mode
● Collapsed ○ Functional Properties for Pins
Generate Report
HTML Report
☑ Show Dedicated Pins
ADC CAN CMP CMT DAC DMA ENET EWM FB FTM GPIO I2C I2S JTAG LLWU LPTMR OSC PDB RCM RTC SDHC SIM SPI UART USB USBDCD VREF MISC

| Functions | Init | Pin/Signal Selection | Direction | User Pin/Signal Name | Selected Pin/Signal Name |
|---|---|---|---|---|---|
| PTB | ☑ | | | | |
| General purpose IO pin 0 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 1 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 2 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 3 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 9 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 10 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 11 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 16 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 17 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 18 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 19 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 20 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 21 | | LED_RED | Output | LED_RED | LED_RED |
| General purpose IO pin 22 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 23 | | *No pin routed* | *No pin routed* | | |
| PTC | ☑ | | | | |
| General purpose IO pin 0 | | *No pin routed* | *No pin routed* | | |

External Use | 40

Summary of pins

| **Port Number** | **Function Name** | **Direction** |
|---|---|---|
| PTA4 | SW3 | Input |
| PTB21 | LED_BLUE | Output |
| PTB22 | LED_RED | Output |
| PTC6 | SW2 | Input |
| PTE26 | LED_GREEN | Output |



## Configure Pin Settings : PTB 22

*Component Inspector - pin_mux
Components Library
Basic
Advanced
Pin Settings
Methods
View Mode
● Collapsed ○ Functional Properties for Pins
Generate Report
HTML Report
☑ Show Dedicated Pins
ADC CAN CMP CMT DAC DMA ENET EWM FB FTM GPIO I2C I2S JTAG LLWU LPTMR OSC PDB RCM RTC SDHC SIM SPI UART USB USBDCD VREF MISC

| Functions | Init | Pin/Signal Selection | Direction | User Pin/Signal Name | Selected Pin/Signal Name |
|---|---|---|---|---|---|
| General purpose IO pin 19 | | No pin routed | No pin routed | | |
| PTB | | | | | |
| | | | No pin routed | | |
| | | | No pin routed | | |
| | | | No pin routed | | |
| General purpose IO pin 3 | | No pin routed | No pin routed | | |
| General purpose IO pin 9 | | No pin routed | No pin routed | | |
| General purpose IO pin 10 | | No pin routed | No pin routed | | |
| General purpose IO pin 11 | | No pin routed | No pin routed | | |
| General purpose IO pin 16 | | No pin routed | No pin routed | | |
| General purpose IO pin 17 | | No pin routed | No pin routed | | |
| General purpose IO pin 18 | | No pin routed | No pin routed | | |
| General purpose IO pin 19 | | No pin routed | No pin routed | | |
| General purpose IO pin 20 | | No pin routed | No pin routed | | |
| General purpose IO pin 21 | | LED_RED | Output | LED_RED | LED_RED |
| General purpose IO pin 22 | | PTB22/SPI2_SOUT/FB_AD... | Output | LED_BLUE | PTB22/SPI2_SOUT/FB_AD... |
| General purpose IO pin 23 | | No pin routed | No pin routed | | |
| PTC | ☑ | | | | |

(1) Click here Select "PTB22/…"

(2) Click here Select "Output"

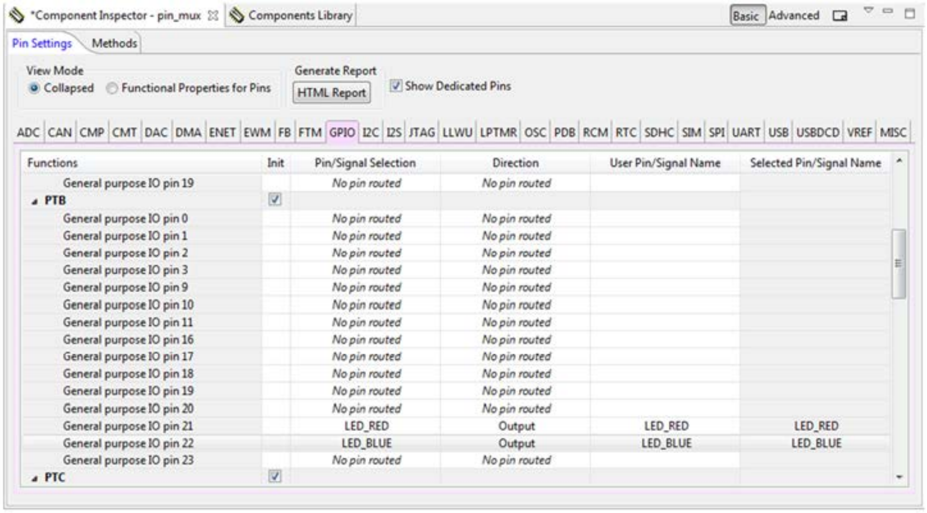
(3) Enter "LED_BLUE"

Setting the Pin Mux for LED_BLUE

Summary of pins

| Port Number | Function Name | Direction |
|---|---|---|
| PTA4 | SW3 | Input |
| PTB21 | LED_BLUE | Output |
| PTB22 | LED_RED | Output |
| PTC6 | SW2 | Input |
| PTE26 | LED_GREEN | Output |

# Configure Pin Settings: PTB 22



*Component Inspector - pin_mux | Components Library | Basic | Advanced

Pin Settings | Methods

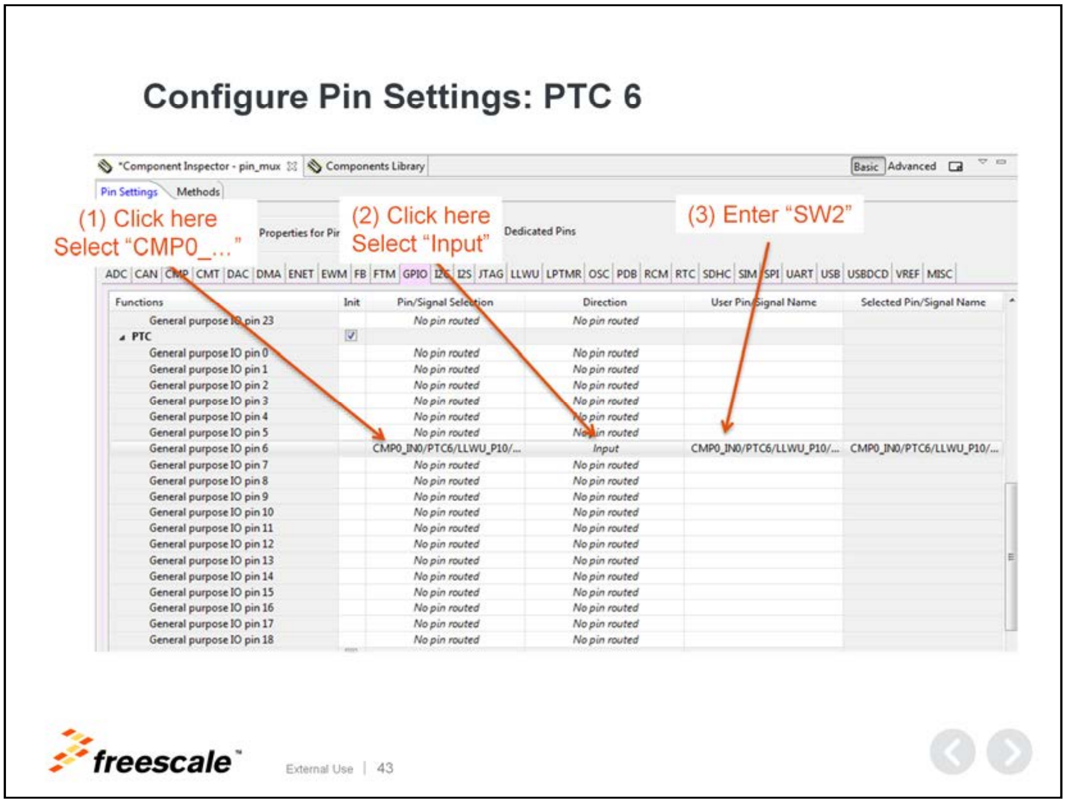
View Mode: ● Collapsed ○ Functional Properties for Pins

Generate Report: HTML Report

☑ Show Dedicated Pins

ADC | CAN | CMP | CMT | DAC | DMA | ENET | EWM | FB | FTM | GPIO | I2C | I2S | JTAG | LLWU | LPTMR | OSC | PDB | RCM | RTC | SDHC | SIM | SPI | UART | USB | USBDCD | VREF | MISC

| Functions | Init | Pin/Signal Selection | Direction | User Pin/Signal Name | Selected Pin/Signal Name |
|---|---|---|---|---|---|
| General purpose IO pin 19 | | *No pin routed* | *No pin routed* | | |
| ◢ **PTB** | ☑ | | | | |
| General purpose IO pin 0 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 1 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 2 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 3 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 9 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 10 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 11 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 16 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 17 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 18 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 19 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 20 | | *No pin routed* | *No pin routed* | | |
| General purpose IO pin 21 | | LED_RED | Output | LED_RED | LED_RED |
| General purpose IO pin 22 | | LED_BLUE | Output | LED_BLUE | LED_BLUE |
| General purpose IO pin 23 | | *No pin routed* | *No pin routed* | | |
| ◢ **PTC** | ☑ | | | | |



## Configure Pin Settings: PTC 6

(1) Click here Select "CMP0_..."

(2) Click here Select "Input"

(3) Enter "SW2"

| Functions | Init | Pin/Signal Selection | Direction | User Pin/Signal Name | Selected Pin/Signal Name |
|---|---|---|---|---|---|
| General purpose IO pin 23 | | No pin routed | No pin routed | | |
| PTC | ☑ | | | | |
| General purpose IO pin 0 | | No pin routed | No pin routed | | |
| General purpose IO pin 1 | | No pin routed | No pin routed | | |
| General purpose IO pin 2 | | No pin routed | No pin routed | | |
| General purpose IO pin 3 | | No pin routed | No pin routed | | |
| General purpose IO pin 4 | | No pin routed | No pin routed | | |
| General purpose IO pin 5 | | No pin routed | No pin routed | | |
| General purpose IO pin 6 | | CMP0_IN0/PTC6/LLWU_P10/... | Input | CMP0_IN0/PTC6/LLWU_P10/... | CMP0_IN0/PTC6/LLWU_P10/... |
| General purpose IO pin 7 | | No pin routed | No pin routed | | |
| General purpose IO pin 8 | | No pin routed | No pin routed | | |
| General purpose IO pin 9 | | No pin routed | No pin routed | | |
| General purpose IO pin 10 | | No pin routed | No pin routed | | |
| General purpose IO pin 11 | | No pin routed | No pin routed | | |
| General purpose IO pin 12 | | No pin routed | No pin routed | | |
| General purpose IO pin 13 | | No pin routed | No pin routed | | |
| General purpose IO pin 14 | | No pin routed | No pin routed | | |
| General purpose IO pin 15 | | No pin routed | No pin routed | | |
| General purpose IO pin 16 | | No pin routed | No pin routed | | |
| General purpose IO pin 17 | | No pin routed | No pin routed | | |
| General purpose IO pin 18 | | No pin routed | No pin routed | | |

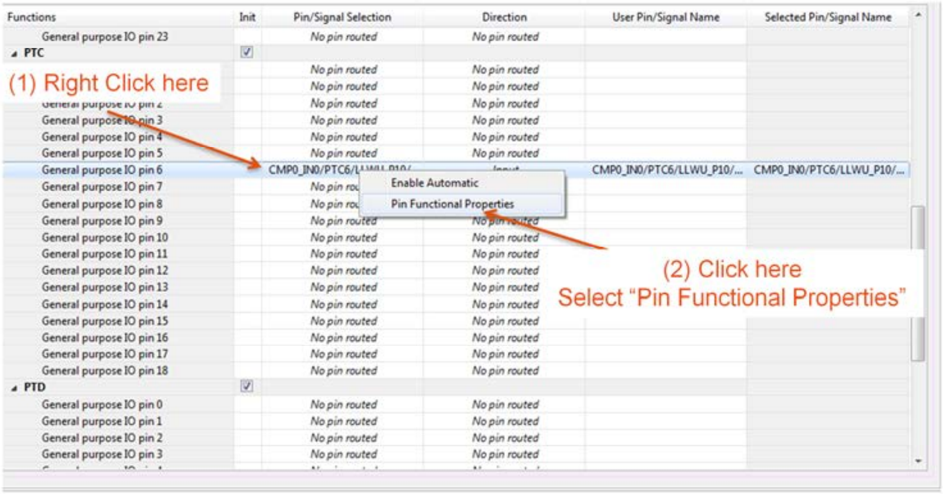
External Use | 43

Scroll down to PTC (Port C) and select the GPIO pin 6, this is for the SW2 settings.

Summary of pins

| **Port Number** | **Function Name** | **Direction** |
|---|---|---|
| PTA4 | SW3 | Input |
| PTB21 | LED_BLUE | Output |
| PTB22 | LED_RED | Output |
| PTC6 | SW2 | Input |
| PTE26 | LED_GREEN | Output |

## Configure Pin Settings: PTC 6

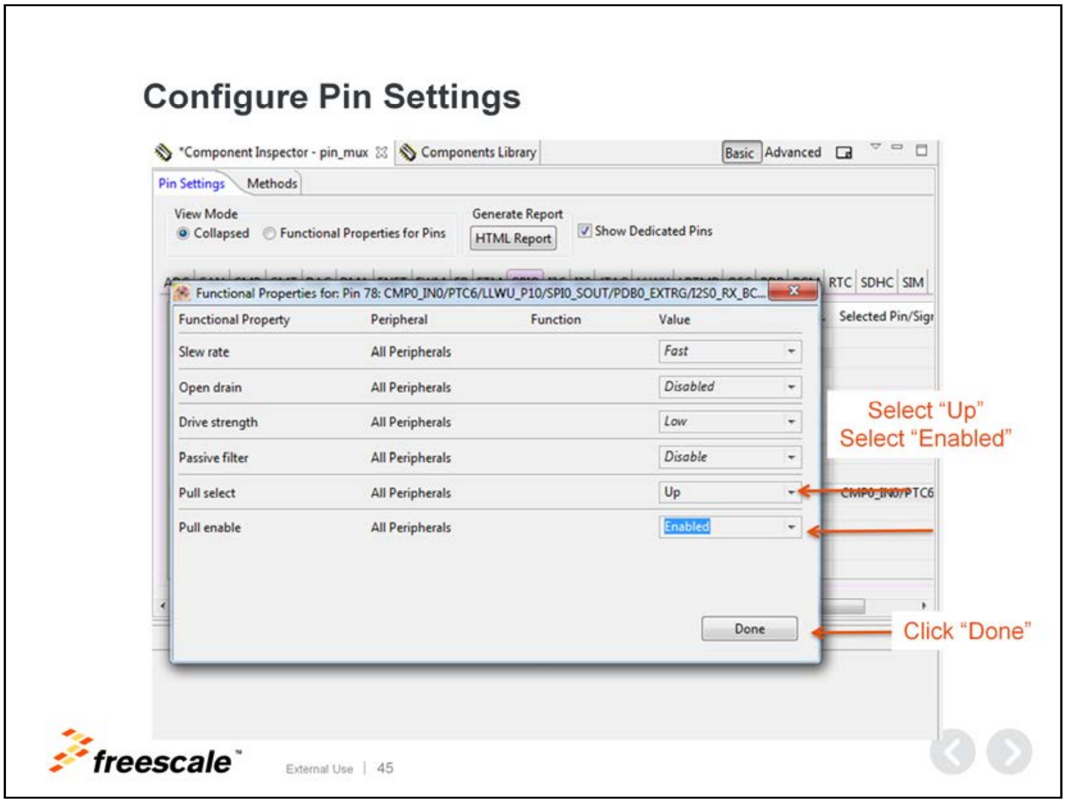


| Functions | Init | Pin/Signal Selection | Direction | User Pin/Signal Name | Selected Pin/Signal Name |
|---|---|---|---|---|---|
| General purpose IO pin 23 | | No pin routed | No pin routed | | |
| PTC | ☑ | | | | |
| | | No pin routed | No pin routed | | |
| | | No pin routed | No pin routed | | |
| General purpose IO pin 2 | | No pin routed | No pin routed | | |
| General purpose IO pin 3 | | No pin routed | No pin routed | | |
| General purpose IO pin 4 | | No pin routed | No pin routed | | |
| General purpose IO pin 5 | | No pin routed | No pin routed | | |
| General purpose IO pin 6 | | CMP0_IN0/PTC6/LLWU_P10/ | Input | CMP0_IN0/PTC6/LLWU_P10/... | CMP0_IN0/PTC6/LLWU_P10/... |
| General purpose IO pin 7 | | No pin rou | | | |
| General purpose IO pin 8 | | No pin rou | | | |
| General purpose IO pin 9 | | No pin routed | No pin routed | | |
| General purpose IO pin 10 | | No pin routed | No pin routed | | |
| General purpose IO pin 11 | | No pin routed | No pin routed | | |
| General purpose IO pin 12 | | No pin routed | No pin routed | | |
| General purpose IO pin 13 | | No pin routed | No pin routed | | |
| General purpose IO pin 14 | | No pin routed | No pin routed | | |
| General purpose IO pin 15 | | No pin routed | No pin routed | | |
| General purpose IO pin 16 | | No pin routed | No pin routed | | |
| General purpose IO pin 17 | | No pin routed | No pin routed | | |
| General purpose IO pin 18 | | No pin routed | No pin routed | | |
| PTD | ☑ | | | | |
| General purpose IO pin 0 | | No pin routed | No pin routed | | |
| General purpose IO pin 1 | | No pin routed | No pin routed | | |
| General purpose IO pin 2 | | No pin routed | No pin routed | | |
| General purpose IO pin 3 | | No pin routed | No pin routed | | |

(1) Right Click here

(2) Click here
Select "Pin Functional Properties"

Again, for the inputs, we need to select the pin functional properties



## Configure Pin Settings

View Mode
● Collapsed ○ Functional Properties for Pins

☑ Show Dedicated Pins

Functional Properties for: Pin 78: CMP0_IN0/PTC6/LLWU_P10/SPI0_SOUT/PDB0_EXTRG/I2S0_RX_BC...

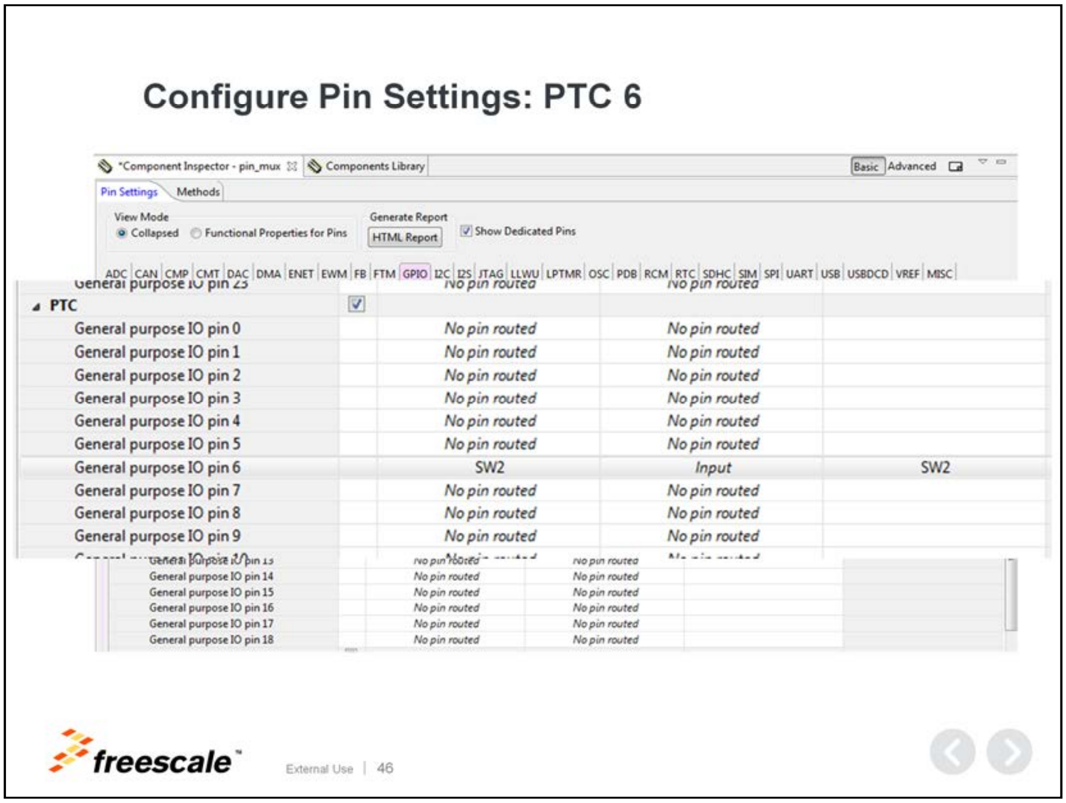
| Functional Property | Peripheral | Function | Value |
|---|---|---|---|
| Slew rate | All Peripherals | | Fast |
| Open drain | All Peripherals | | Disabled |
| Drive strength | All Peripherals | | Low |
| Passive filter | All Peripherals | | Disable |
| Pull select | All Peripherals | | Up |
| Pull enable | All Peripherals | | Enabled |

Select "Up"
Select "Enabled"

Click "Done"

External Use | 45

For the switch inputs, the switches require the use of the internal pull-up since there is no external pull-up on the schematic.

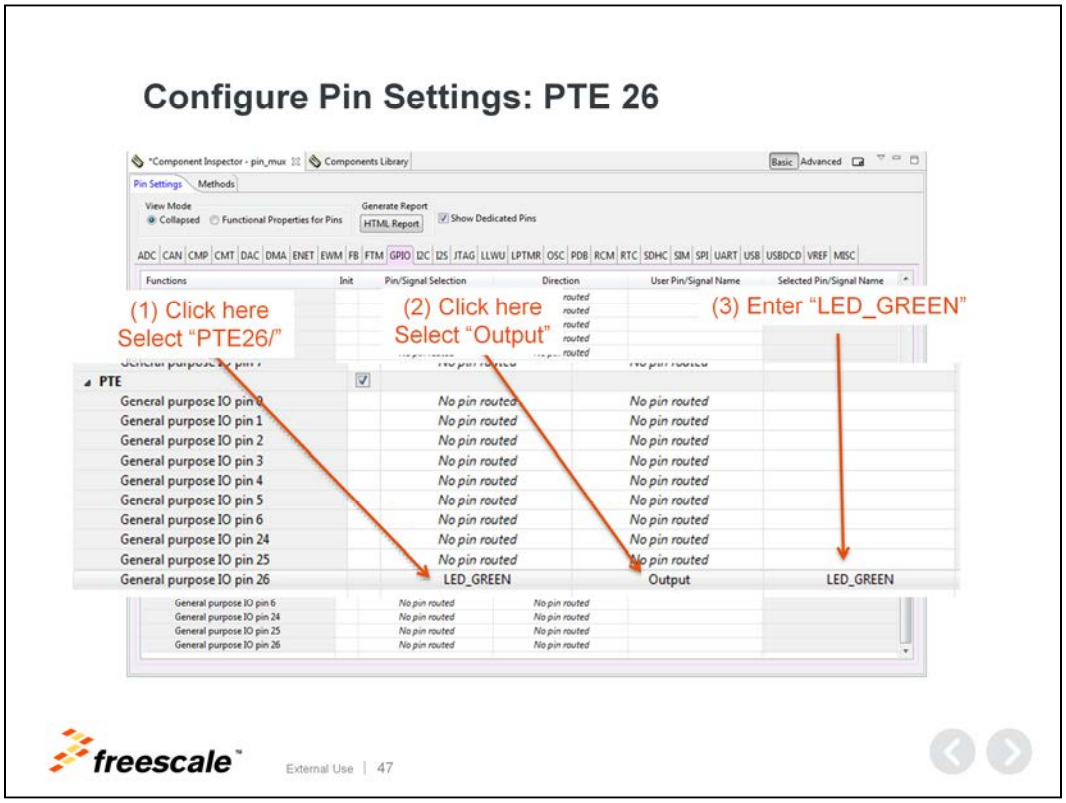


## Configure Pin Settings: PTC 6

*Component Inspector - pin_mux
Components Library
Basic
Advanced
Pin Settings
Methods
View Mode
● Collapsed
○ Functional Properties for Pins
Generate Report
HTML Report
☑ Show Dedicated Pins
ADC CAN CMP CMT DAC DMA ENET EWM FB FTM GPIO I2C I2S JTAG LLWU LPTMR OSC PDB RCM RTC SDHC SIM SPI UART USB USBDCD VREF MISC

| | | | | |
|---|---|---|---|---|
| General purpose IO pin 23 | | No pin routed | No pin routed | |
| PTC | ☑ | | | |
| General purpose IO pin 0 | | No pin routed | No pin routed | |
| General purpose IO pin 1 | | No pin routed | No pin routed | |
| General purpose IO pin 2 | | No pin routed | No pin routed | |
| General purpose IO pin 3 | | No pin routed | No pin routed | |
| General purpose IO pin 4 | | No pin routed | No pin routed | |
| General purpose IO pin 5 | | No pin routed | No pin routed | |
| General purpose IO pin 6 | | SW2 | Input | SW2 |
| General purpose IO pin 7 | | No pin routed | No pin routed | |
| General purpose IO pin 8 | | No pin routed | No pin routed | |
| General purpose IO pin 9 | | No pin routed | No pin routed | |
| General purpose IO pin 10 | | No pin routed | No pin routed | |
| General purpose IO pin 13 | | No pin routed | No pin routed | |
| General purpose IO pin 14 | | No pin routed | No pin routed | |
| General purpose IO pin 15 | | No pin routed | No pin routed | |
| General purpose IO pin 16 | | No pin routed | No pin routed | |
| General purpose IO pin 17 | | No pin routed | No pin routed | |
| General purpose IO pin 18 | | No pin routed | No pin routed | |

One more setting for GPIO, that would be **LED Green PTE26 Output**



## Configure Pin Settings: PTE 26

*Component Inspector - pin_mux | Components Library | Basic | Advanced

Pin Settings | Methods

View Mode: ● Collapsed ○ Functional Properties for Pins

Generate Report: HTML Report

☑ Show Dedicated Pins

ADC CAN CMP CMT DAC DMA ENET EWM FB FTM GPIO I2C I2S JTAG LLWU LPTMR OSC PDB RCM RTC SDHC SIM SPI UART USB USBDCD VREF MISC

| Functions | Init | Pin/Signal Selection | Direction | User Pin/Signal Name | Selected Pin/Signal Name |
|---|---|---|---|---|---|
| PTE | ☑ | | | | |
| General purpose IO pin 0 | | No pin routed | No pin routed | | |
| General purpose IO pin 1 | | No pin routed | No pin routed | | |
| General purpose IO pin 2 | | No pin routed | No pin routed | | |
| General purpose IO pin 3 | | No pin routed | No pin routed | | |
| General purpose IO pin 4 | | No pin routed | No pin routed | | |
| General purpose IO pin 5 | | No pin routed | No pin routed | | |
| General purpose IO pin 6 | | No pin routed | No pin routed | | |
| General purpose IO pin 24 | | No pin routed | No pin routed | | |
| General purpose IO pin 25 | | No pin routed | No pin routed | | |
| General purpose IO pin 26 | | LED_GREEN | Output | | LED_GREEN |

(1) Click here Select "PTE26/"

(2) Click here Select "Output"

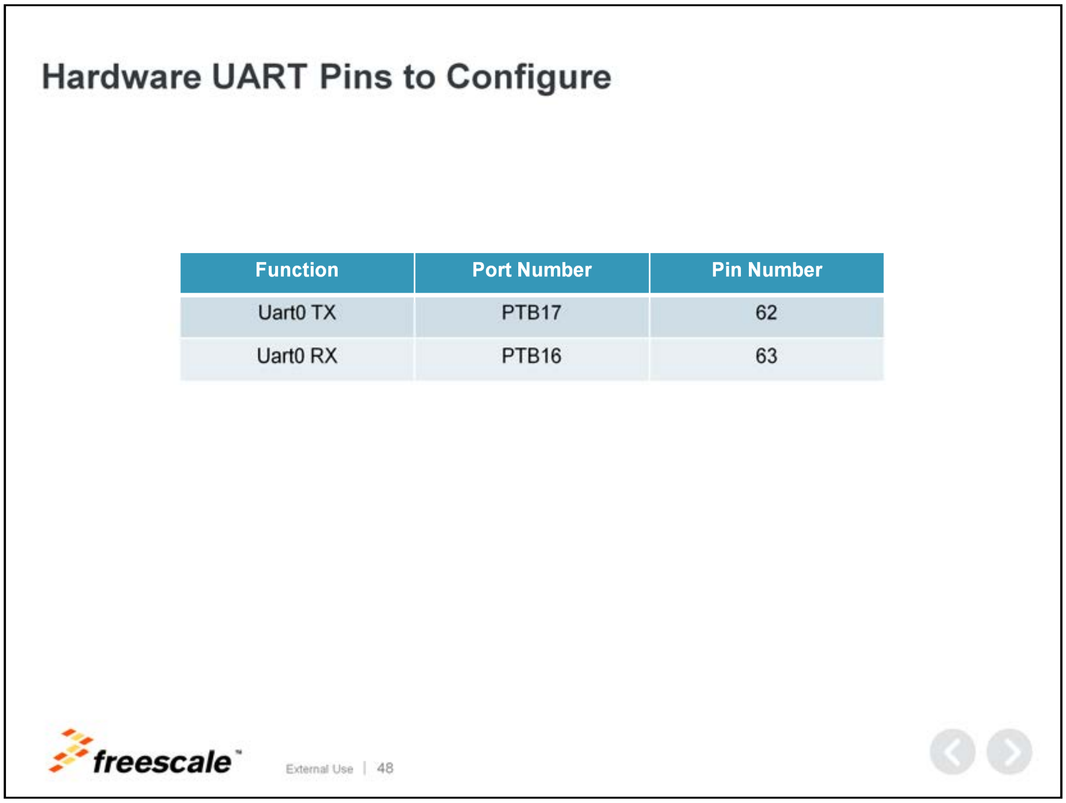
(3) Enter "LED_GREEN"

External Use | 47

Scroll down to PTE (Port E) and **LED Green PTE26 Output**

Notice that when you change the name of the User Pin/Signal Name, the "Pin/Signal Selection" also changes name. We will use these user names when we set up the drivers.

Now we move onto the UART pin muxing



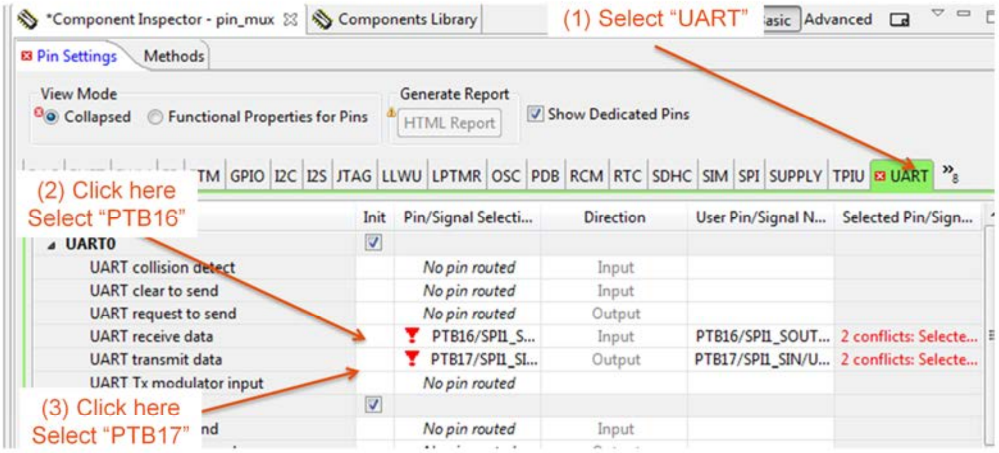
## Hardware UART Pins to Configure

| Function | Port Number | Pin Number |
|---|---|---|
| Uart0 TX | PTB17 | 62 |
| Uart0 RX | PTB16 | 63 |

freescale External Use | 48

Here are the pins we need to deal with for the UART. This uart is connected to the debug interface chip on this board, and it allows for the uart to print on the PC terminal window.

## Configure Pin Settings: UART



View Mode
● Collapsed ○ Functional Properties for Pins

☑ Show Dedicated Pins

| | Init | Pin/Signal Selecti... | Direction | User Pin/Signal N... | Selected Pin/Sign... |
|---|---|---|---|---|---|
| UART0 | ☑ | | | | |
| UART collision detect | | No pin routed | Input | | |
| UART clear to send | | No pin routed | Input | | |
| UART request to send | | No pin routed | Output | | |
| UART receive data | | PTB16/SPI1_S... | Input | PTB16/SPI1_SOUT... | 2 conflicts: Selecte... |
| UART transmit data | | PTB17/SPI1_SI... | Output | PTB17/SPI1_SIN/U... | 2 conflicts: Selecte... |
| UART Tx modulator input | | No pin routed | | | |
| | ☑ | | | | |
| nd | | No pin routed | Input | | |

(1) Select "UART"

(2) Click here Select "PTB16"

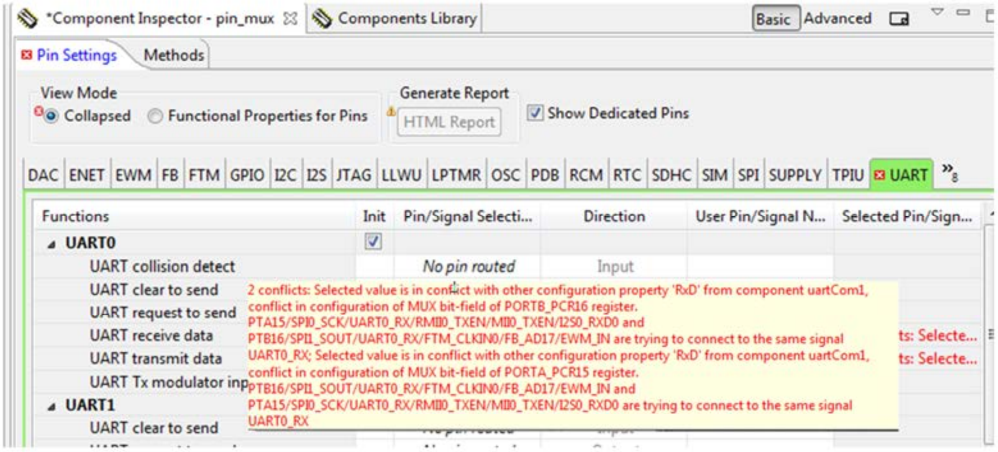
(3) Click here Select "PTB17"

External Use | 49

There is no real need to give this pin a user name, since we will use the uart as a module. we can and will leave this at the default name. For your own design, you may want to rename this to match your schematic.

Notice there are errors showing up.

## Configure Pin Settings: UART


*Component Inspector - pin_mux
Components Library
Basic
Advanced
Pin Settings
Methods
Generate Report
HTML Report
DAC ENET EWM FB FTM GPIO I2C I2S JTAG LLWU LPTMR OSC PDB RCM RTC SDHC SIM SPI SUPPLY TPIU UART

View Mode: ● Collapsed ○ Functional Properties for Pins

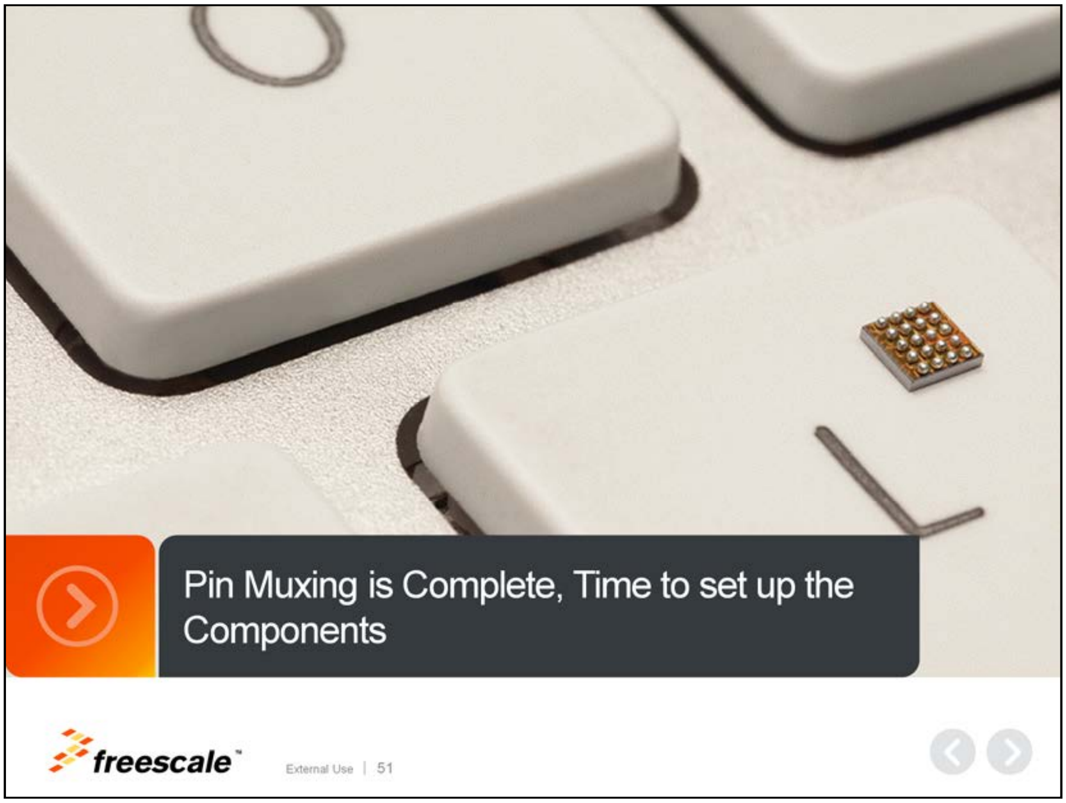
☑ Show Dedicated Pins

| Functions | Init | Pin/Signal Selecti... | Direction | User Pin/Signal N... | Selected Pin/Sign... |
|---|---|---|---|---|---|
| UART0 | ☑ | | | | |
| UART collision detect | | *No pin routed* | Input | | |
| UART clear to send | | | | | |
| UART request to send | | | | | |
| UART receive data | | | | | ts: Selecte... |
| UART transmit data | | | | | ts: Selecte... |
| UART Tx modulator inp | | | | | |
| UART1 | | | | | |
| UART clear to send | | | | | |

2 conflicts: Selected value is in conflict with other configuration property 'RxD' from component uartCom1, conflict in configuration of MUX bit-field of PORTB_PCR16 register. PTA15/SPI0_SCK/UART0_RX/RMII0_TXEN/MII0_TXEN/I2S0_RXD0 and PTB16/SPI1_SOUT/UART0_RX/FTM_CLKIN0/FB_AD17/EWM_IN are trying to connect to the same signal UART0_RX; Selected value is in conflict with other configuration property 'RxD' from component uartCom1, conflict in configuration of MUX bit-field of PORTA_PCR15 register. PTB16/SPI1_SOUT/UART0_RX/FTM_CLKIN0/FB_AD17/EWM_IN and PTA15/SPI0_SCK/UART0_RX/RMII0_TXEN/MII0_TXEN/I2S0_RXD0 are trying to connect to the same signal UART0_RX

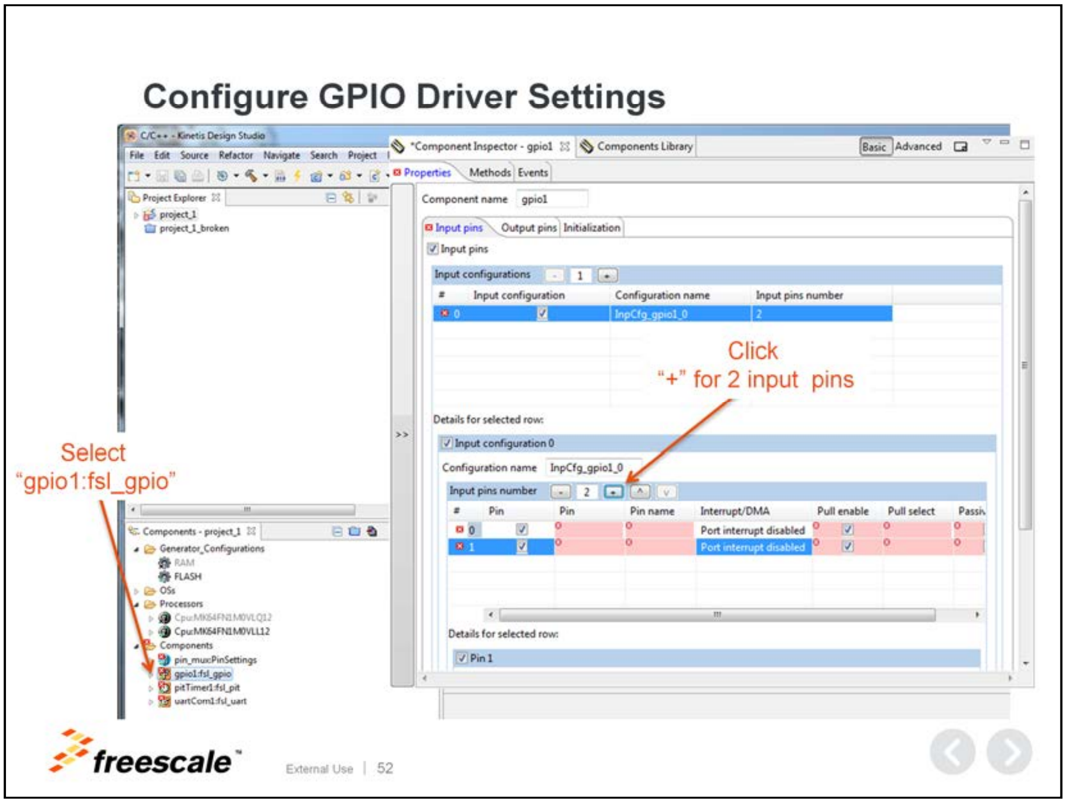
External Use | 50

The error message indicates a conflict with “uartCom1”, which we have not configured yet. These errors will be corrected when the uart component is configured.



## Pin Muxing is Complete, Time to set up the Components

External Use | 51



## Configure GPIO Driver Settings

Select "gpio1:fsl_gpio"

Click "+" for 2 input pins

Component name: gpio1

☑ Input pins

Input configurations: 1

| # | Input configuration | Configuration name | Input pins number |
|---|---|---|---|
| 0 | ☑ | InpCfg_gpio1_0 | 2 |

Details for selected row:

☑ Input configuration 0

Configuration name: InpCfg_gpio1_0

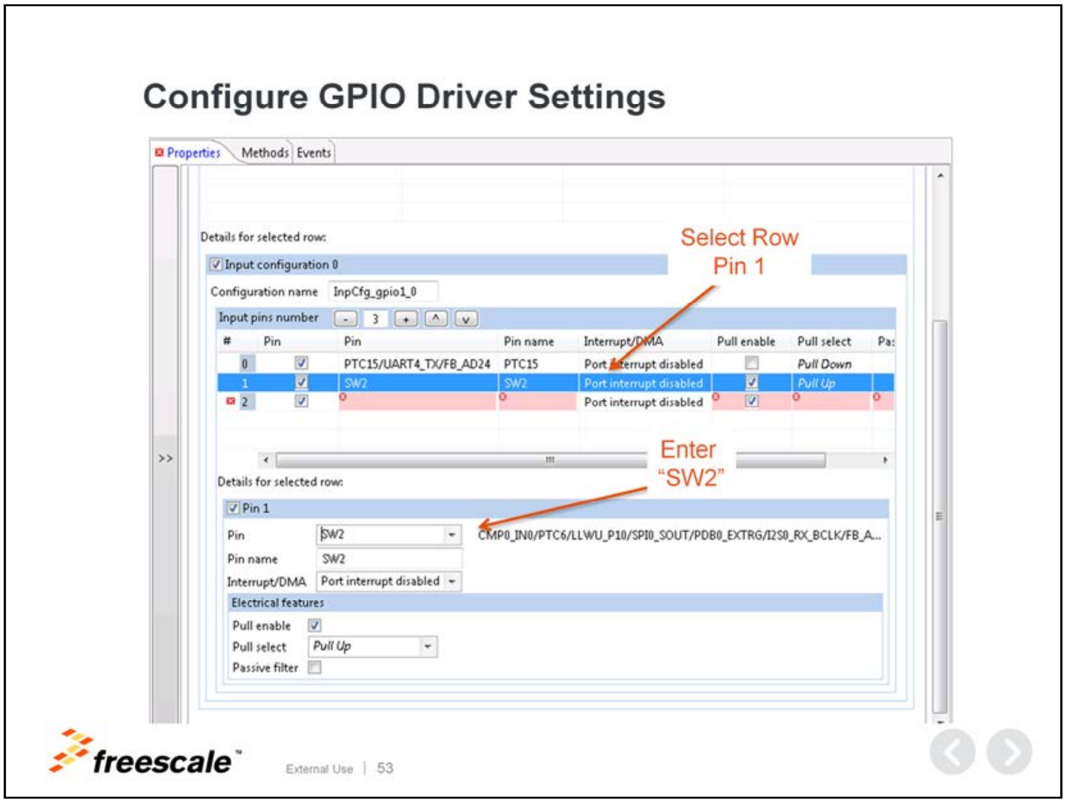
Input pins number: 2

| # | Pin | Pin | Pin name | Interrupt/DMA | Pull enable | Pull select | Passiv |
|---|---|---|---|---|---|---|---|
| 0 | ☑ | | | Port interrupt disabled | ☑ | | |
| 1 | ☑ | | | Port interrupt disabled | ☑ | | |

Details for selected row:

☑ Pin 1

At this point we can now configure the drivers. The first on our list is the gpio1:fsl_gpio. There are 2 inputs that are needed for our project, click the '+' twice. There may already be gpio pins set. If this is the case, we can delete them since our project does not need them. If you were adding gpio's to an existing project, then you would expect the pins to be listed.



## Configure GPIO Driver Settings

Properties | Methods | Events

Details for selected row:

☑ Input configuration 0

Configuration name: InpCfg_gpio1_0

Input pins number: 3

| # | Pin | Pin | Pin name | Interrupt/DMA | Pull enable | Pull select | Pa: |
|---|---|---|---|---|---|---|---|
| 0 | ☑ | PTC15/UART4_TX/FB_AD24 | PTC15 | Port interrupt disabled | ☐ | Pull Down | |
| 1 | ☑ | SW2 | SW2 | Port interrupt disabled | ☑ | Pull Up | |
| 2 | ☑ | | | Port interrupt disabled | ☑ | | |

Details for selected row:

☑ Pin 1

Pin: SW2 — CMP0_IN0/PTC6/LLWU_P10/SPI0_SOUT/PDB0_EXTRG/I2S0_RX_BCLK/FB_A...

Pin name: SW2

Interrupt/DMA: Port interrupt disabled

Electrical features

Pull enable: ☑

Pull select: Pull Up

Passive filter: ☐

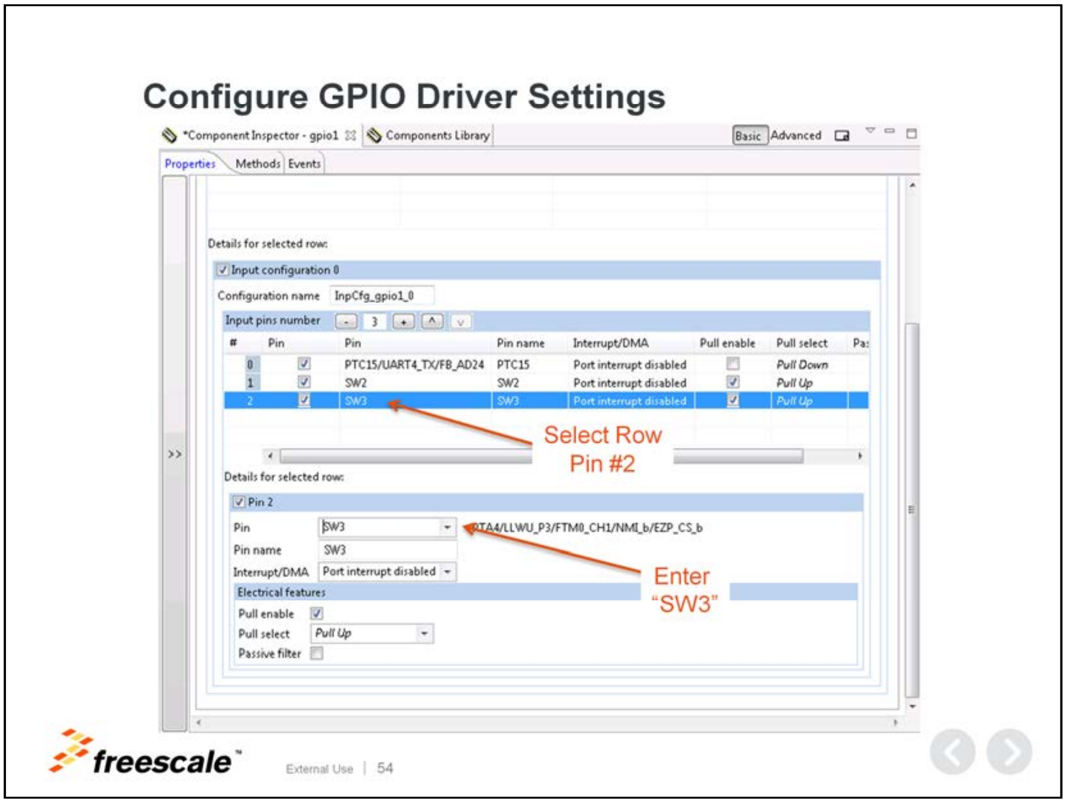
Select Row Pin 1

Enter "SW2"

freescale External Use | 53

By selecting the row, which in this case is pin #1, select the details for this pin we added. Since we assigned "SW2" as an input with pull ups enabled, that selection carries over to this panel. Now do this again for pin #3, and call I "SW3". The selection that shows row 0 as an input pin is not being used by this project. This can be removed. This is an errata.

## Configure GPIO Driver Settings



Details for selected row:

☑ Input configuration 0

Configuration name: InpCfg_gpio1_0

Input pins number: 3

| # | Pin | Pin | Pin name | Interrupt/DMA | Pull enable | Pull select | Pa: |
|---|---|---|---|---|---|---|---|
| 0 | ☑ | PTC15/UART4_TX/FB_AD24 | PTC15 | Port interrupt disabled | □ | Pull Down | |
| 1 | ☑ | SW2 | SW2 | Port interrupt disabled | ☑ | Pull Up | |
| 2 | ☑ | SW3 | SW3 | Port interrupt disabled | ☑ | Pull Up | |

Select Row
Pin #2

Details for selected row:

☑ Pin 2

Pin: SW3 PTA4/LLWU_P3/FTM0_CH1/NMI_b/EZP_CS_b

Pin name: SW3

Interrupt/DMA: Port interrupt disabled

Electrical features

Pull enable ☑

Pull select: Pull Up

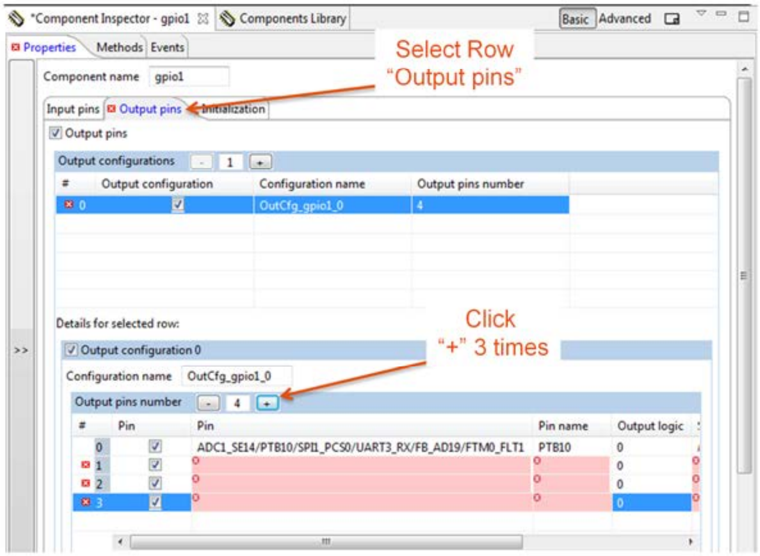
Passive filter □

Enter
“SW3”

External Use | 54

By selecting the row, which in this case is pin #1, select the details for this pin. Since we assigned “SW3” as an input with pull ups enabled, that selection carries over to this panel.

## Configure GPIO Driver Settings



Select Row "Output pins"

Component name: gpio1

☑ Output pins

Output configurations: 1

| # | Output configuration | Configuration name | Output pins number |
|---|---|---|---|
| 0 | ☑ | OutCfg_gpio1_0 | 4 |

Details for selected row:

☑ Output configuration 0

Configuration name: OutCfg_gpio1_0

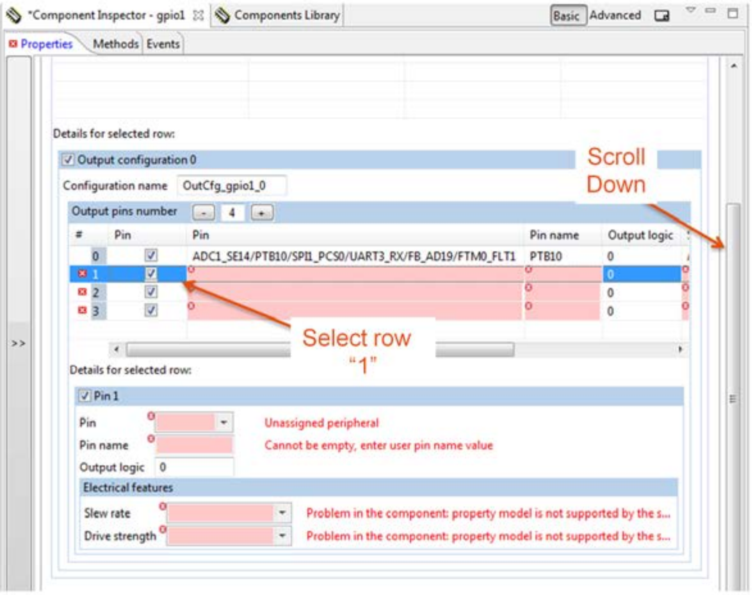
Output pins number: 4

Click "+" 3 times

| # | Pin | Pin | Pin name | Output logic |
|---|---|---|---|---|
| 0 | ☑ | ADC1_SE14/PTB10/SPI1_PCS0/UART3_RX/FB_AD19/FTM0_FLT1 | PTB10 | 0 |
| 1 | ☑ | | | 0 |
| 2 | ☑ | | | 0 |
| 3 | ☑ | | | 0 |

Its time to move to the output pins. Add 3 output pins, one each for RED, GREEN, BLUE.

# Configure GPIO Driver Settings



Details for selected row:

☑ Output configuration 0

Configuration name: OutCfg_gpio1_0

Output pins number: 4

| # | Pin | Pin | Pin name | Output logic |
|---|---|---|---|---|
| 0 | ☑ | ADC1_SE14/PTB10/SPI1_PCS0/UART3_RX/FB_AD19/FTM0_FLT1 | PTB10 | 0 |
| 1 | ☑ | | | 0 |
| 2 | ☑ | | | 0 |
| 3 | ☑ | | | 0 |

Details for selected row:

☑ Pin 1

Pin: Unassigned peripheral

Pin name: Cannot be empty, enter user pin name value

Output logic: 0

Electrical features

Slew rate: Problem in the component: property model is not supported by the s...

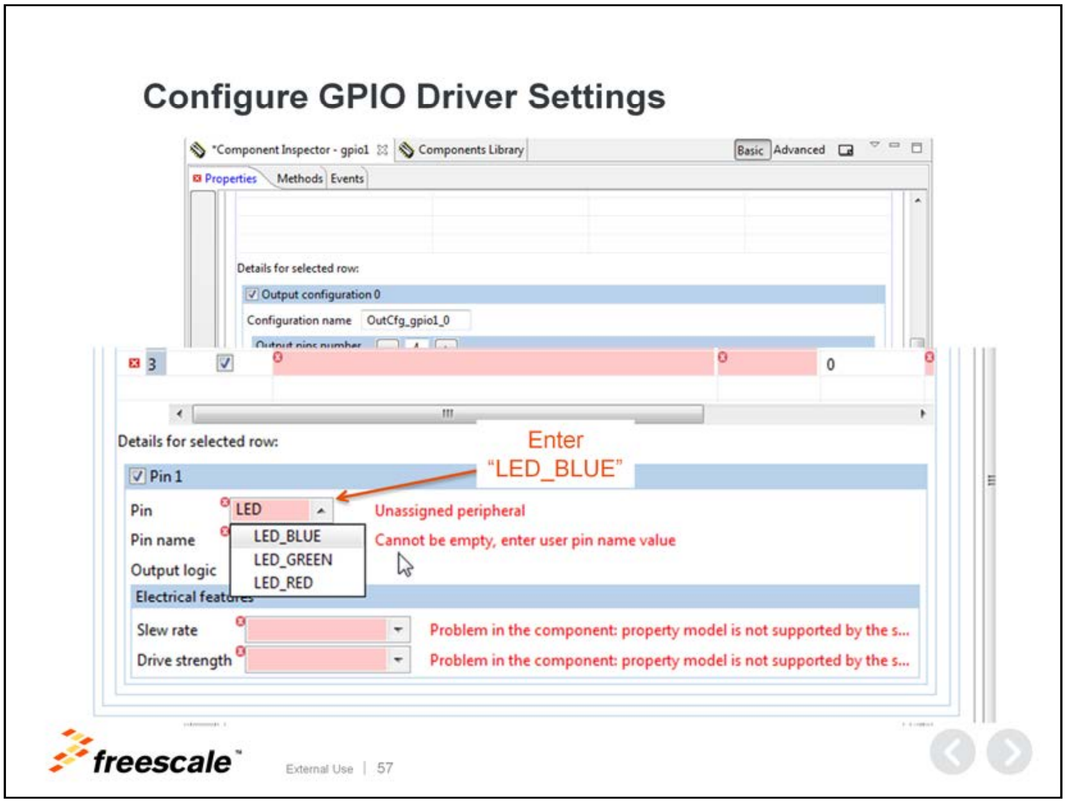
Drive strength: Problem in the component: property model is not supported by the s...

Scroll Down

Select row "1"

External Use | 56

Scroll down to see the pin settings properties. Click or select row 1, which is the first of the rows (pins) that we added.



## Configure GPIO Driver Settings

Details for selected row:

☑ Output configuration 0

Configuration name: OutCfg_gpio1_0

Output pins number

3 ☑ 0

Details for selected row:

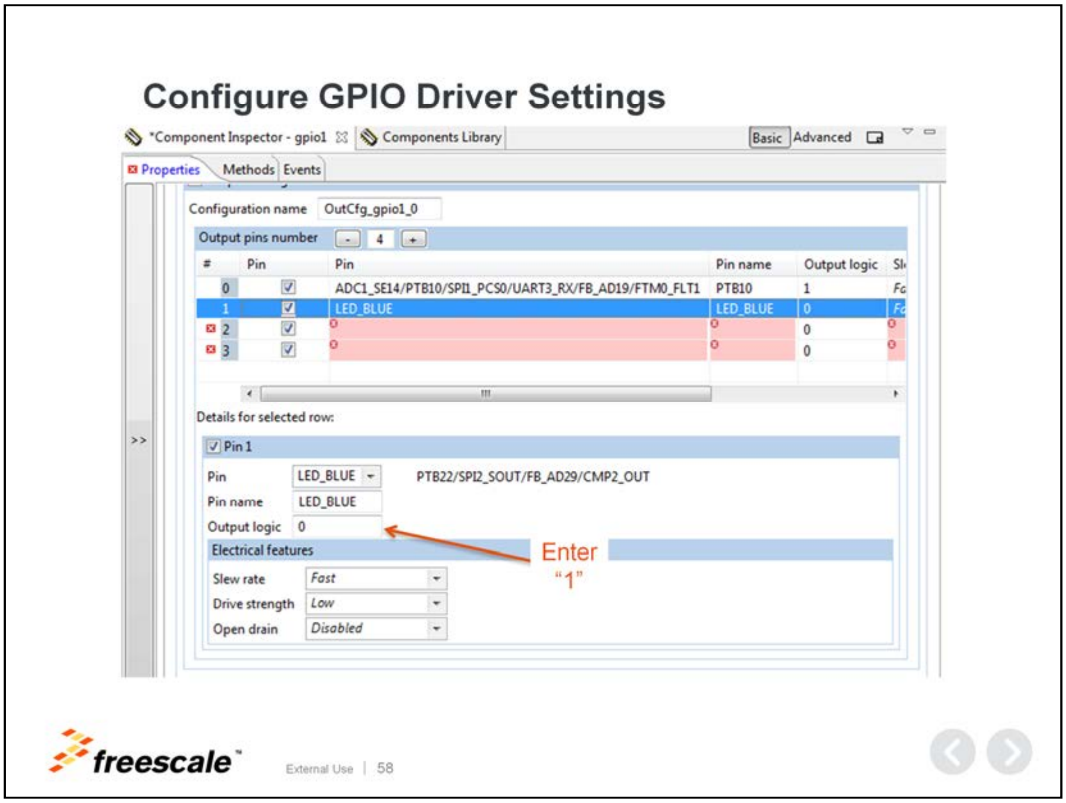
☑ Pin 1

| | | |
|---|---|---|
| Pin | LED | Unassigned peripheral |
| Pin name | LED_BLUE<br>LED_GREEN<br>LED_RED | Cannot be empty, enter user pin name value |
| Output logic | | |
| Electrical features | | |
| Slew rate | | Problem in the component: property model is not supported by the s... |
| Drive strength | | Problem in the component: property model is not supported by the s... |

Enter "LED_BLUE"

External Use | 57

At this point, the pin will be assigned based on the pin_mux names that was used in the pin_mux settings section. As you enter the pin name, the dropdown list will filter as you enter the name. So, after entering "LED" the three LED_ names pop up. You can either keep typing the name or select the name from the list.



## Configure GPIO Driver Settings

Configuration name: OutCfg_gpio1_0

Output pins number: 4

| # | Pin | Pin | Pin name | Output logic | Sl |
|---|---|---|---|---|---|
| 0 | ☑ | ADC1_SE14/PTB10/SPI1_PCS0/UART3_RX/FB_AD19/FTM0_FLT1 | PTB10 | 1 | Fa |
| 1 | ☑ | LED_BLUE | LED_BLUE | 0 | Fa |
| 2 | ☑ | | | 0 | |
| 3 | ☑ | | | 0 | |

Details for selected row:

☑ Pin 1

Pin: LED_BLUE PTB22/SPI2_SOUT/FB_AD29/CMP2_OUT

Pin name: LED_BLUE

Output logic: 0

Electrical features

Slew rate: Fast

Drive strength: Low

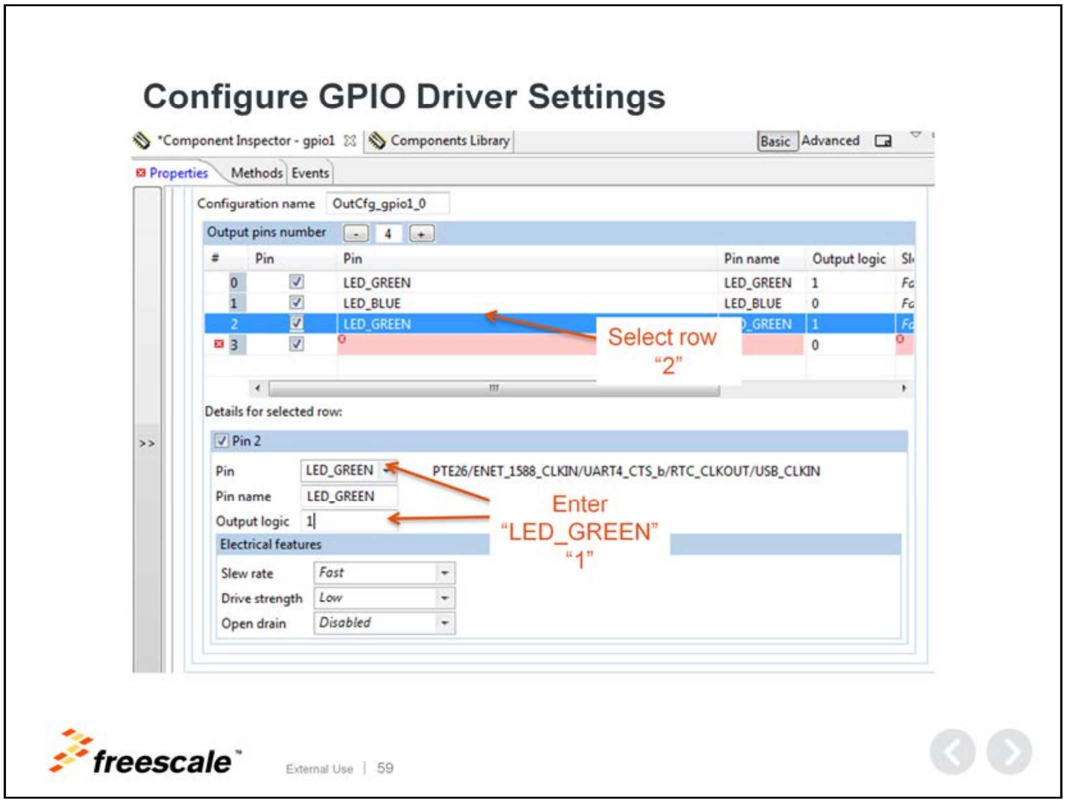
Open drain: Disabled

Enter "1"

External Use | 58

Set the output logic to 1. So the default state of the LED is OFF.

And it looks like this…



## Configure GPIO Driver Settings

Configuration name: OutCfg_gpio1_0

Output pins number: 4

| # | Pin | Pin | Pin name | Output logic | Sl |
|---|---|---|---|---|---|
| 0 | ☑ | LED_GREEN | LED_GREEN | 1 | Fa |
| 1 | ☑ | LED_BLUE | LED_BLUE | 0 | Fa |
| 2 | ☑ | LED_GREEN | _GREEN | 1 | Fa |
| 3 | ☑ | | | 0 | |

Select row "2"

Details for selected row:

☑ Pin 2

Pin: LED_GREEN PTE26/ENET_1588_CLKIN/UART4_CTS_b/RTC_CLKOUT/USB_CLKIN

Pin name: LED_GREEN

Output logic: 1

Enter "LED_GREEN" "1"

Electrical features

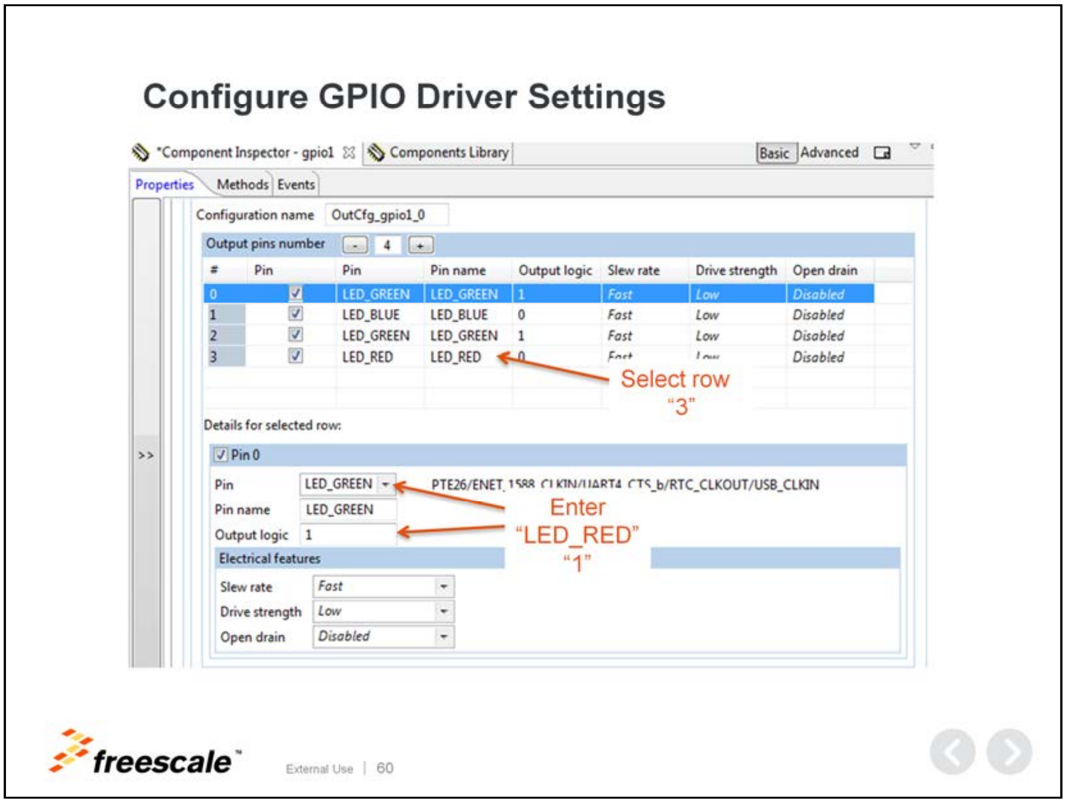
Slew rate: Fast

Drive strength: Low

Open drain: Disabled

freescale External Use | 59

Scroll down to see the pin settings properties. Click or select row 2 and enter LED_GREEN, and set output logic to 1



## Configure GPIO Driver Settings

*Component Inspector - gpio1 | Components Library | Basic | Advanced

Properties | Methods | Events

Configuration name: OutCfg_gpio1_0

Output pins number: 4

| # | Pin | Pin | Pin name | Output logic | Slew rate | Drive strength | Open drain |
|---|---|---|---|---|---|---|---|
| 0 | ☑ | LED_GREEN | LED_GREEN | 1 | Fast | Low | Disabled |
| 1 | ☑ | LED_BLUE | LED_BLUE | 0 | Fast | Low | Disabled |
| 2 | ☑ | LED_GREEN | LED_GREEN | 1 | Fast | Low | Disabled |
| 3 | ☑ | LED_RED | LED_RED | 0 | Fast | Low | Disabled |

Select row "3"

Details for selected row:

☑ Pin 0

Pin: LED_GREEN — PTE26/ENET_1588_CLKIN/UART4_CTS_b/RTC_CLKOUT/USB_CLKIN

Pin name: LED_GREEN

Output logic: 1

Enter "LED_RED" "1"

Electrical features

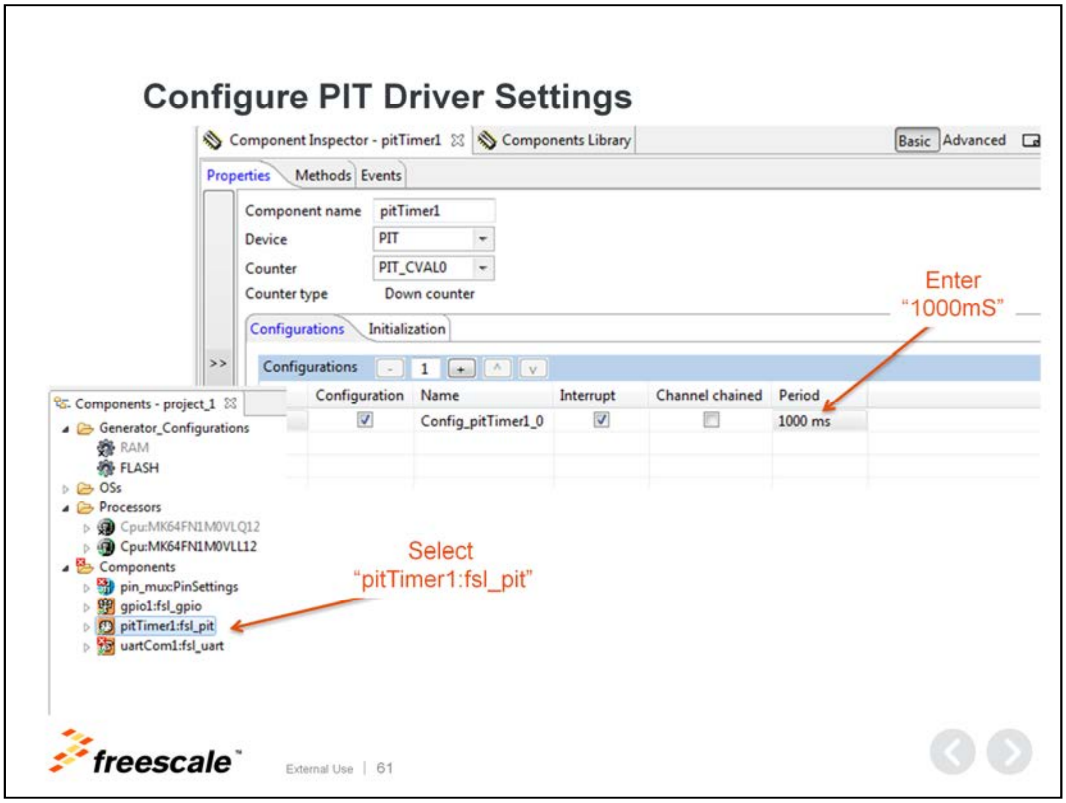
Slew rate: Fast

Drive strength: Low

Open drain: Disabled

freescale External Use | 60

Scroll down to see the pin settings properties. Click or select row 3 and enter LED_RED.



## Configure PIT Driver Settings

Component name: pitTimer1

Device: PIT

Counter: PIT_CVAL0

Counter type: Down counter

Configurations: 1

| Configuration | Name | Interrupt | Channel chained | Period |
|---|---|---|---|---|
| ☑ | Config_pitTimer1_0 | ☑ | ☐ | 1000 ms |

Components - project_1

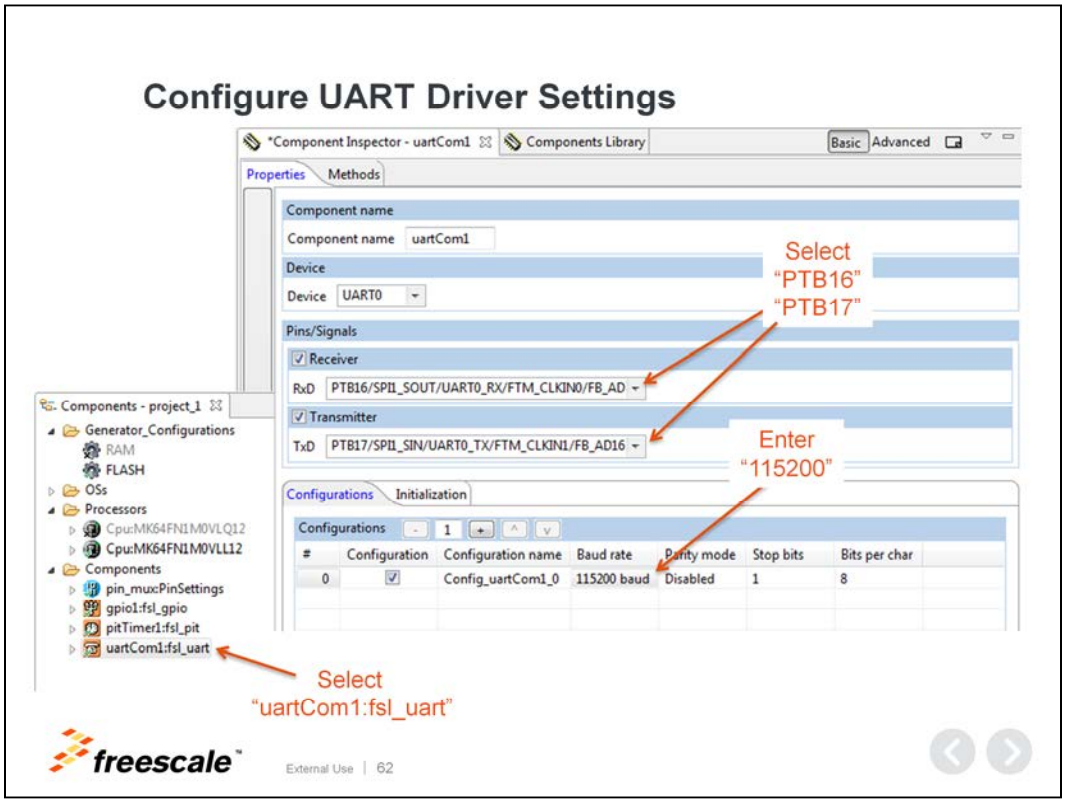
- Generator_Configurations
  - RAM
  - FLASH
- OSs
- Processors
  - Cpu:MK64FN1M0VLQ12
  - Cpu:MK64FN1M0VLL12
- Components
  - pin_mux:PinSettings
  - gpio1:fsl_gpio
  - pitTimer1:fsl_pit
  - uartCom1:fsl_uart

Enter “1000mS”

Select “pitTimer1:fsl_pit”

freescale™ External Use | 61

Now we move to the interrupt timer (PIT) and set it to 1000mS.



## Configure UART Driver Settings

Component name

Component name: uartCom1

Device

Device: UART0

Pins/Signals

☑ Receiver

RxD: PTB16/SPI1_SOUT/UART0_RX/FTM_CLKIN0/FB_AD

☑ Transmitter

TxD: PTB17/SPI1_SIN/UART0_TX/FTM_CLKIN1/FB_AD16

Configurations: 1

| # | Configuration | Configuration name | Baud rate | Parity mode | Stop bits | Bits per char |
|---|---|---|---|---|---|---|
| 0 | ☑ | Config_uartCom1_0 | 115200 baud | Disabled | 1 | 8 |

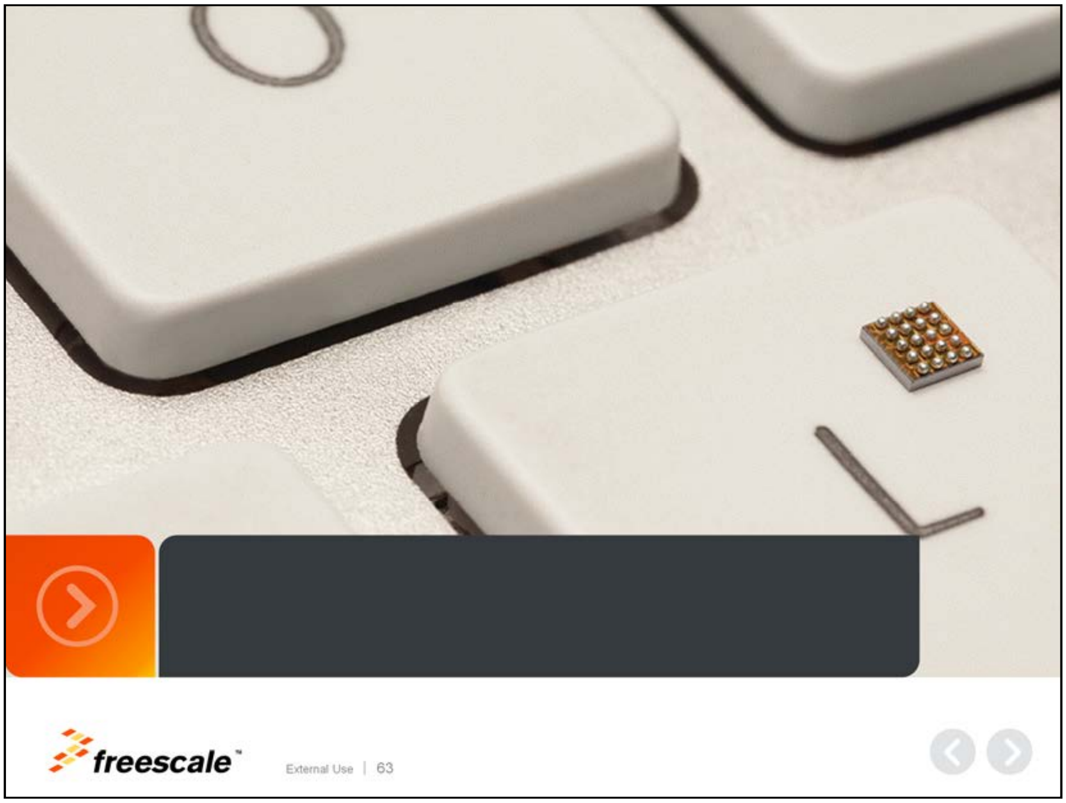
Select "PTB16" "PTB17"

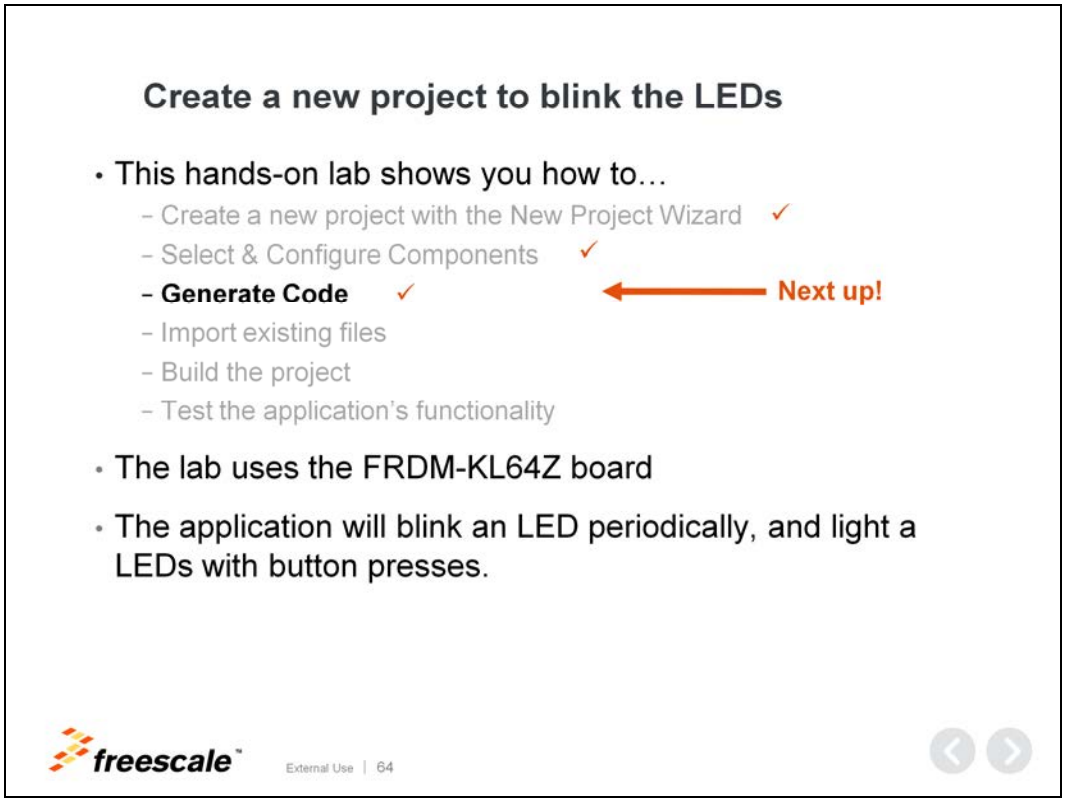
Enter "115200"

Select "uartCom1:fsl_uart"

freescale™ External Use | 62

Here the pins are selected and the baud rate is selected.

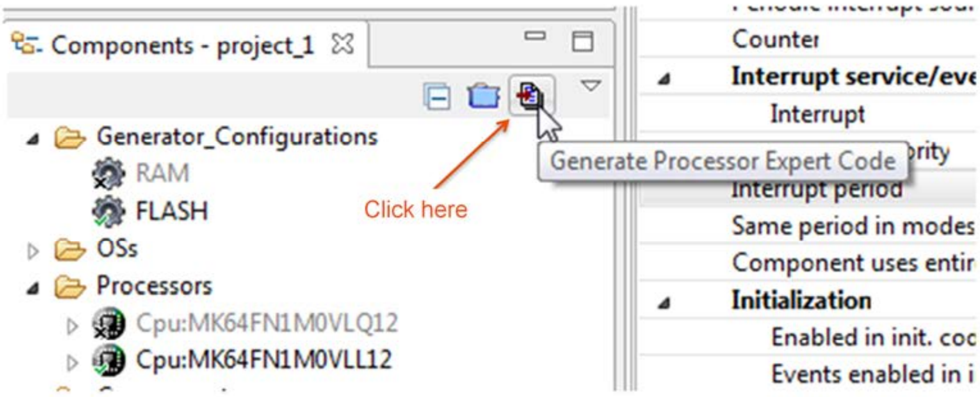




## Create a new project to blink the LEDs

- This hands-on lab shows you how to…
  - Create a new project with the New Project Wizard ✓
  - Select & Configure Components ✓
  - **Generate Code** ✓ ← **Next up!**
  - Import existing files
  - Build the project
  - Test the application's functionality
- The lab uses the FRDM-KL64Z board
- The application will blink an LED periodically, and light a LEDs with button presses.

## Generate Code with Processor Expert

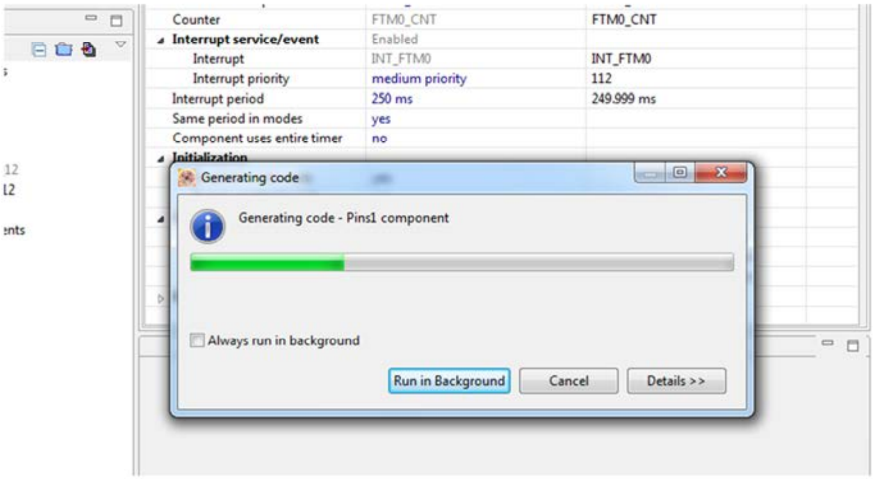


Click here

To generate Processor Expert Code, press on this icon. Also, if you hover over the icon, it gives you an indication of what it does.

As with most programs, there are always more than one way to do something, and this is no exception. To generate Processor Expert code, you can also click on Project -> Generate Processor Expert Code.

# Generate Code with Processor Expert

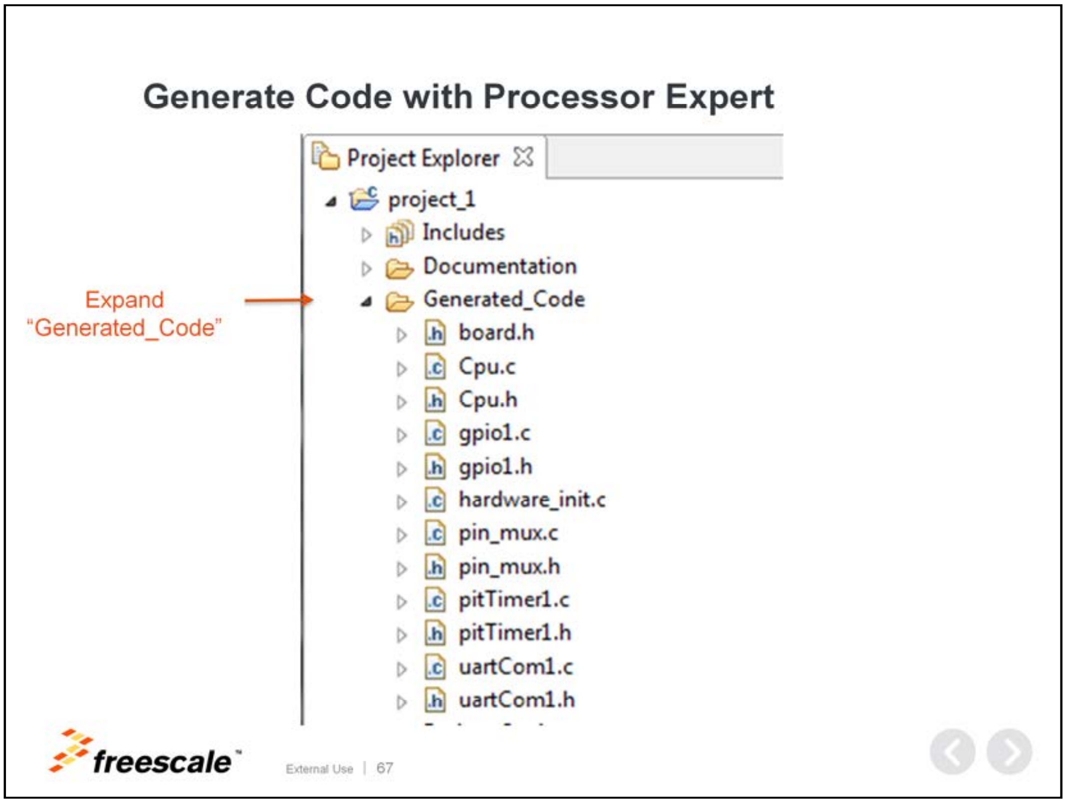


| | | |
|---|---|---|
| Counter | FTM0_CNT | FTM0_CNT |
| Interrupt service/event | Enabled | |
| Interrupt | INT_FTM0 | INT_FTM0 |
| Interrupt priority | medium priority | 112 |
| Interrupt period | 250 ms | 249.999 ms |
| Same period in modes | yes | |
| Component uses entire timer | no | |
| Initialization | | |

Generating code

Generating code - Pins1 component

□ Always run in background



## Generate Code with Processor Expert

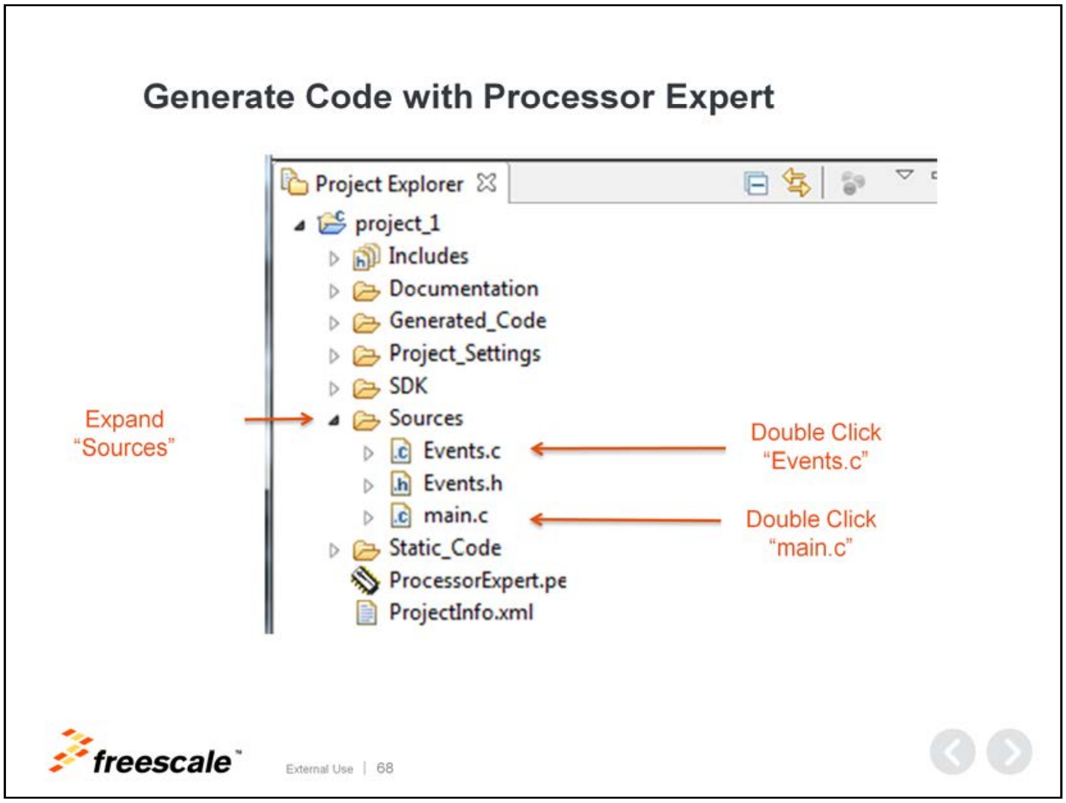
Expand "Generated_Code"

Project Explorer

- project_1
  - Includes
  - Documentation
  - Generated_Code
    - board.h
    - Cpu.c
    - Cpu.h
    - gpio1.c
    - gpio1.h
    - hardware_init.c
    - pin_mux.c
    - pin_mux.h
    - pitTimer1.c
    - pitTimer1.h
    - uartCom1.c
    - uartCom1.h

External Use | 67

A quick check of some of the generated code. When you expand the generated code section in the Project, you will see listings of header and c files.



## Generate Code with Processor Expert

Project Explorer

- project_1
  - Includes
  - Documentation
  - Generated_Code
  - Project_Settings
  - SDK
  - Sources
    - Events.c
    - Events.h
    - main.c
  - Static_Code
  - ProcessorExpert.pe
  - ProjectInfo.xml

Expand "Sources"

Double Click "Events.c"

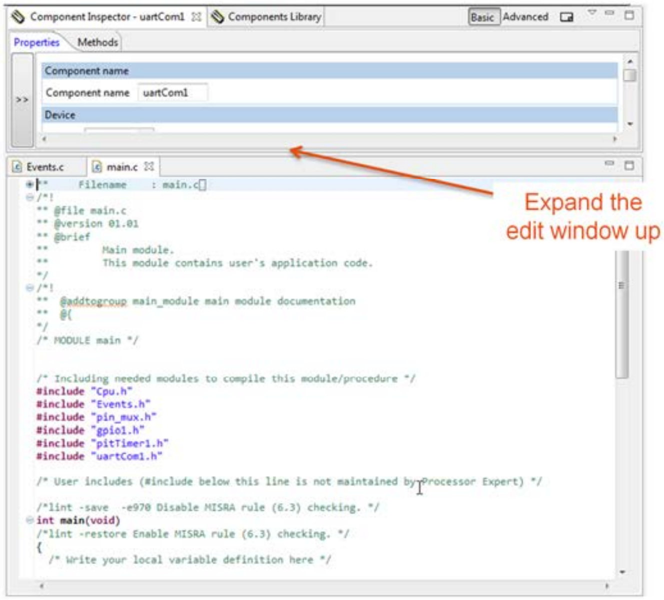
Double Click "main.c"

External Use | 68

A quick check of some of the generated code. The "Sources" folder includes other source files, including main.c and events.c.

Events.c is where the interrupt callback routines are placed. The project has the GREEN_LED blinking based on the interrupt timer, this is where the code is placed to blink the LED.

## Examine the Code



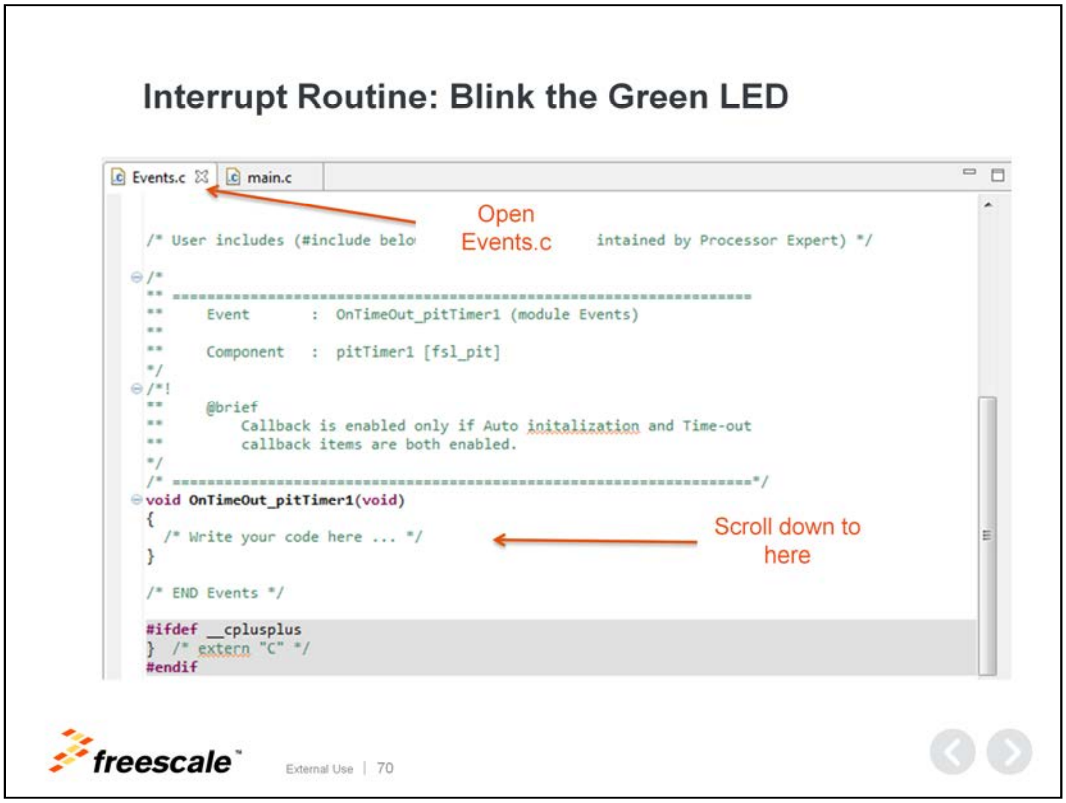
Component name

Component name: uartCom1

Device

```
/**     Filename     : main.c
/*!
** @file main.c
** @version 01.01
** @brief
**         Main module.
**         This module contains user's application code.
*/
/*!
**  @addtogroup main_module main module documentation
**  @{
*/
/* MODULE main */


/* Including needed modules to compile this module/procedure */
#include "Cpu.h"
#include "Events.h"
#include "pin_mux.h"
#include "gpio1.h"
#include "pitTimer1.h"
#include "uartCom1.h"

/* User includes (#include below this line is not maintained by Processor Expert) */

/*lint -save  -e970 Disable MISRA rule (6.3) checking. */
int main(void)
/*lint -restore Enable MISRA rule (6.3) checking. */
{
  /* Write your local variable definition here */
```

Expand the edit window up



## Interrupt Routine: Blink the Green LED

```
/* User includes (#include belo          intained by Processor Expert) */

/*
** ===================================================================
**     Event       :  OnTimeOut_pitTimer1 (module Events)
**
**     Component   :  pitTimer1 [fsl_pit]
*/
/*!
**     @brief
**         Callback is enabled only if Auto initalization and Time-out
**         callback items are both enabled.
*/
/* ===================================================================*/
void OnTimeOut_pitTimer1(void)
{
  /* Write your code here ... */
}

/* END Events */

#ifdef __cplusplus
}  /* extern "C" */
#endif
```

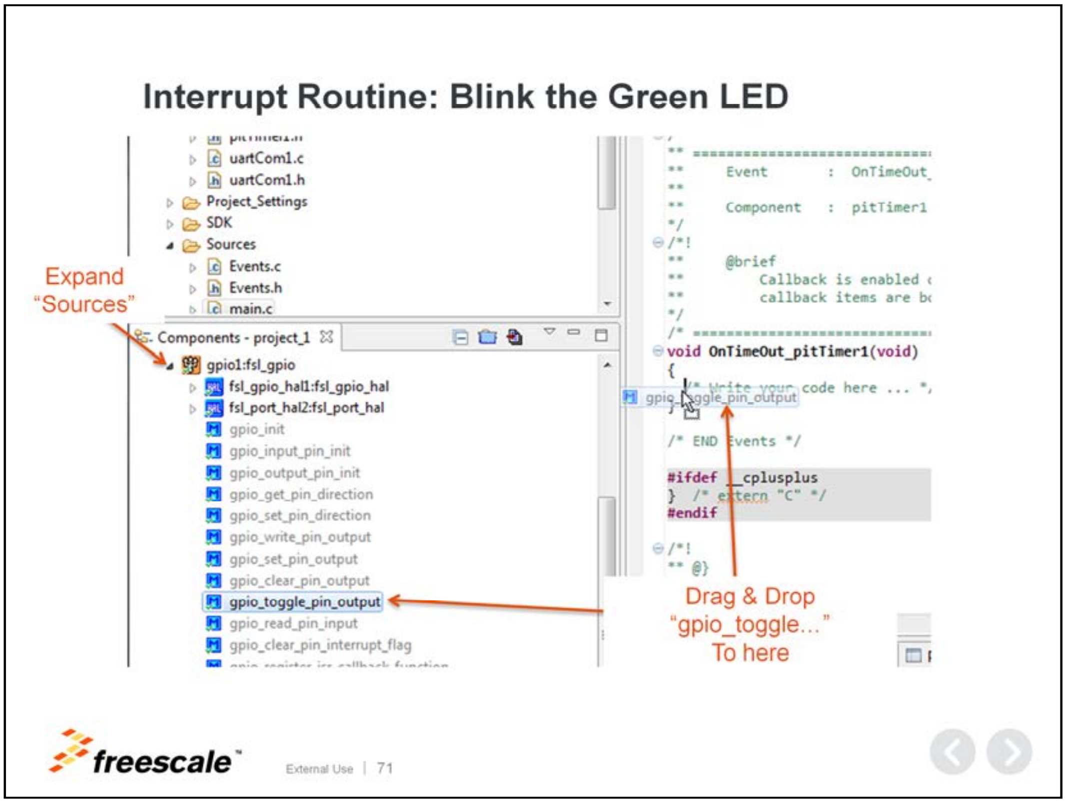
Open Events.c

Scroll down to here

freescale External Use | 70

In “events.c” the interrupt callback is installed. Here is where we will put the blink the led code. Please note the “/* Write your code here … */”

## Interrupt Routine: Blink the Green LED



```
** =============================
**     Event       :  OnTimeOut_
**
**     Component   :  pitTimer1
*/
/*!
**     @brief
**         Callback is enabled (
**         callback items are b(
*/
/* =============================
void OnTimeOut_pitTimer1(void)
{
  /* Write your code here ... */
}

/* END Events */

#ifdef __cplusplus
}  /* extern "C" */
#endif

/*!
** @}
```

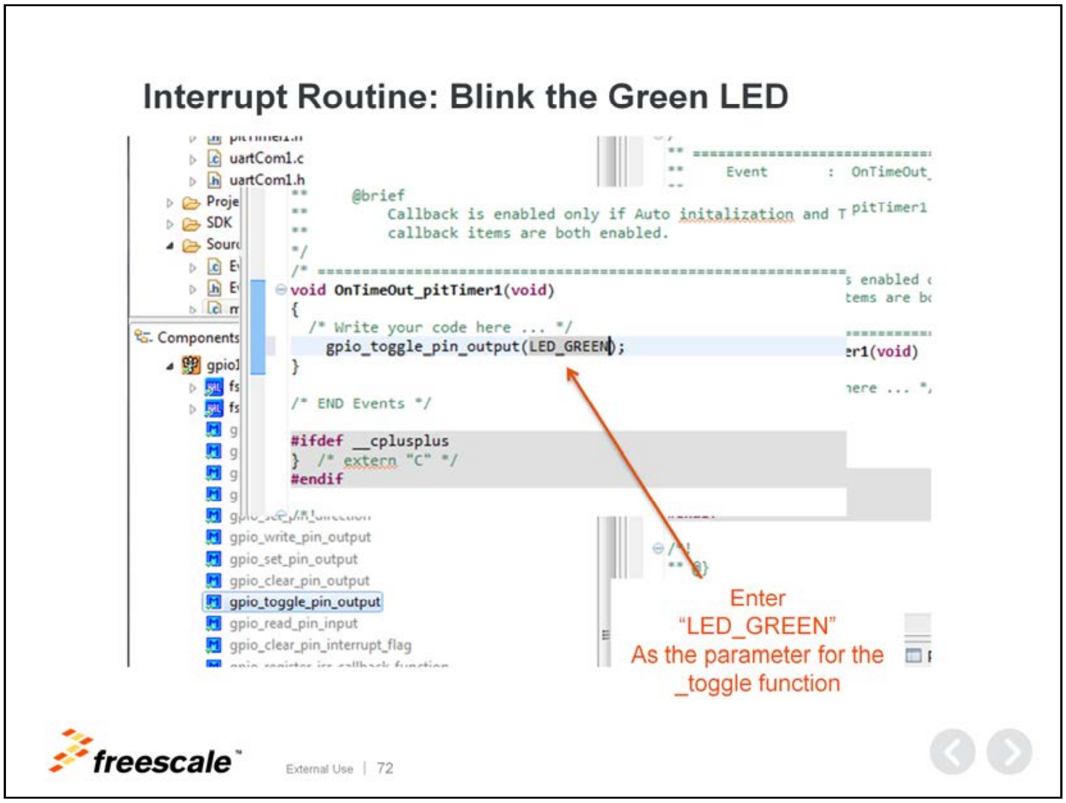
Expand "Sources"

Drag & Drop "gpio_toggle…" To here

freescale External Use | 71

Simply "drag & drop" gpio_toggle_pin_output" to the Events.c file. Make sure this is between the {}.

This sets up the call back routine to turn the LED on/off.



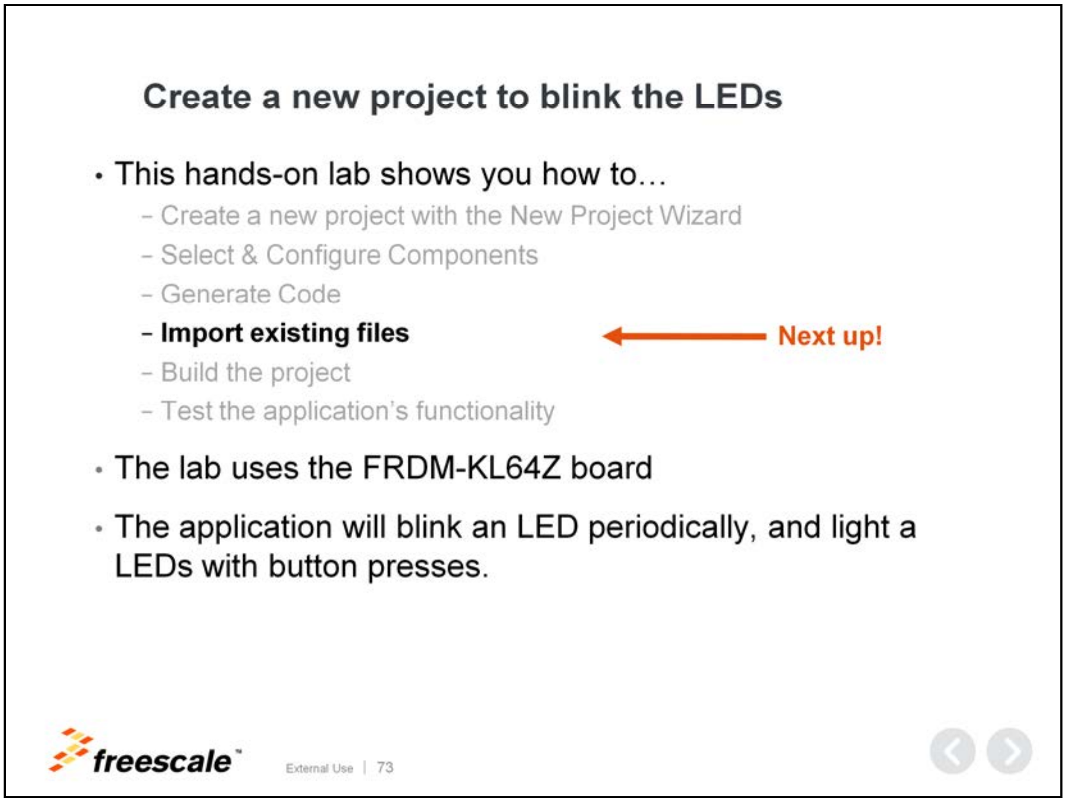
## Interrupt Routine: Blink the Green LED

```
**     Event       :  OnTimeOut_
**     @brief
**         Callback is enabled only if Auto initalization and T
**         callback items are both enabled.
*/
/* ===================================================================*/
void OnTimeOut_pitTimer1(void)
{
  /* Write your code here ... */
    gpio_toggle_pin_output(LED_GREEN);
}

/* END Events */

#ifdef __cplusplus
}  /* extern "C" */
#endif
```

Enter
"LED_GREEN"
As the parameter for the _toggle function

External Use | 72

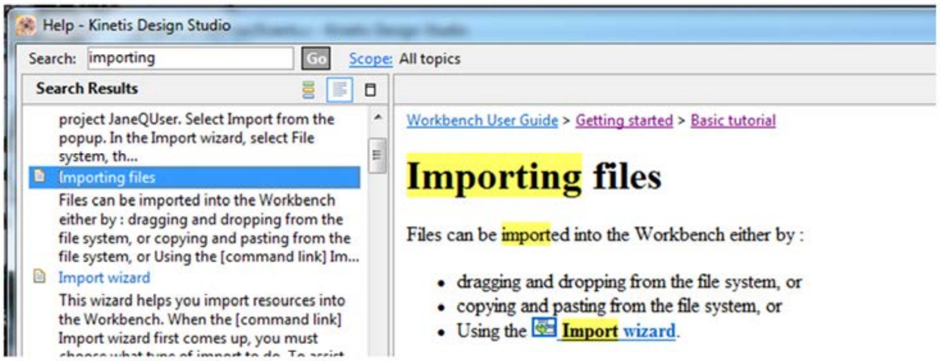
For the gpio_toggle_..., a parameter needs to be passed. Since we want the GREEN led to blink at the timer."LED_GREEN" was defined when we set the pin name.



## Create a new project to blink the LEDs

- This hands-on lab shows you how to…
  - Create a new project with the New Project Wizard
  - Select & Configure Components
  - Generate Code
  - **Import existing files** ← Next up!
  - Build the project
  - Test the application's functionality
- The lab uses the FRDM-KL64Z board
- The application will blink an LED periodically, and light a LEDs with button presses.

## Importing Files



project JaneQUser. Select Import from the popup. In the Import wizard, select File system, th...

Importing files

Files can be imported into the Workbench either by : dragging and dropping from the file system, or copying and pasting from the file system, or Using the [command link] Im...

Import wizard

This wizard helps you import resources into the Workbench. When the [command link] Import wizard first comes up, you must choose what type of import to do. To assist

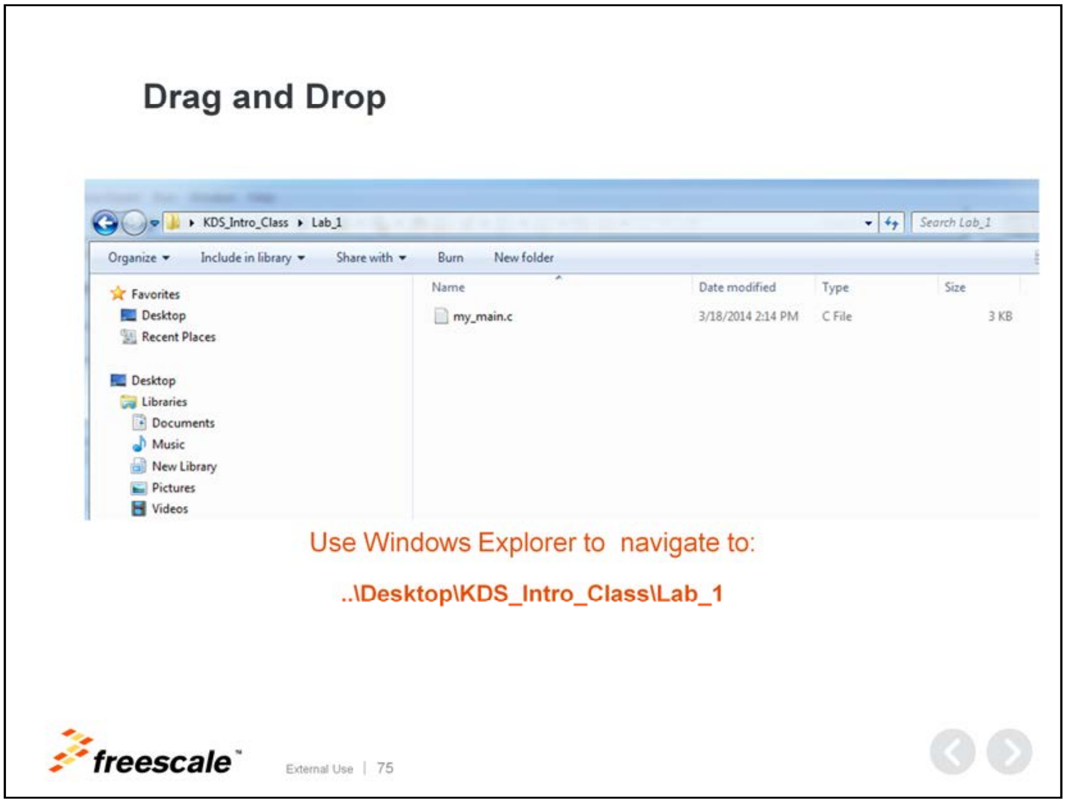
Workbench User Guide > Getting started > Basic tutorial

# Importing files

Files can be imported into the Workbench either by :

- dragging and dropping from the file system, or
- copying and pasting from the file system, or
- Using the Import wizard.

Looking at KDS Help, there are three ways to import files. We will demonstrate the “drag and drop” method in this exercise. Remember, in our OS’s, there are always more than one way to do any particular task.



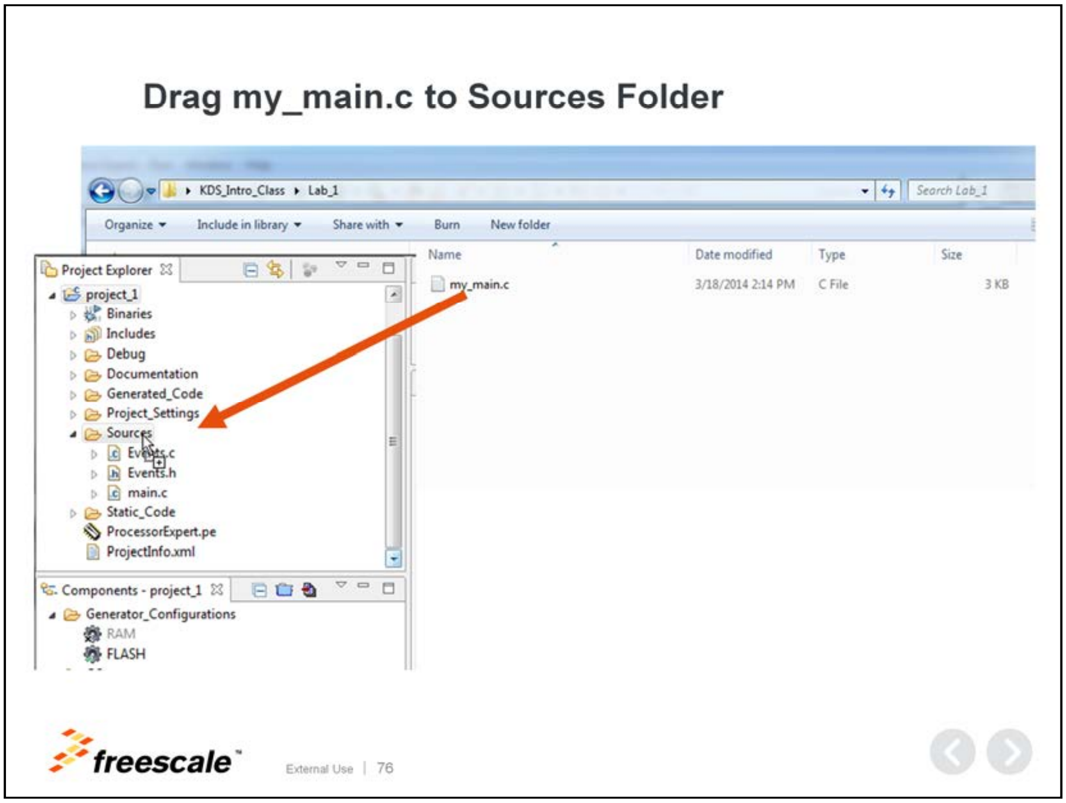
## Drag and Drop

Use Windows Explorer to navigate to:

**..\Desktop\KDS_Intro_Class\Lab_1**

External Use | 75

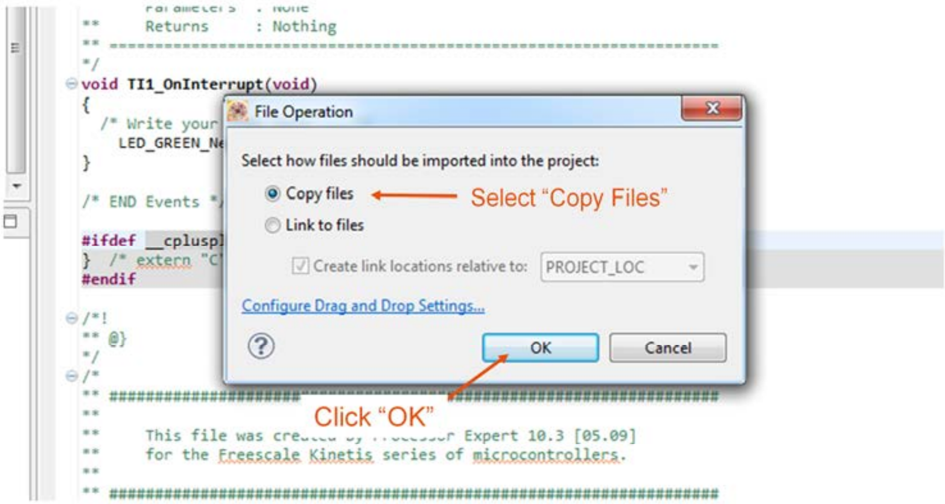
Find the file you want to use…



## Drag my_main.c to Sources Folder

And Drag & Drop into the “Sources” folder. Or copy/paste it into the “Sources” folder

## Drag my_main.c to Sources Folder



```
**     Parameters  : None
**     Returns     : Nothing
** ===================================================================
*/
void TI1_OnInterrupt(void)
{
  /* Write your
    LED_GREEN_Ne
}

/* END Events */

#ifdef __cplusp
}  /* extern "C
#endif

/*!
** @}
*/
/*
** ###################                 ###############################
**
**     This file was cre...  ...r Expert 10.3 [05.09]
**     for the Freescale Kinetis series of microcontrollers.
**
** ######################################################################
```

File Operation

Select how files should be imported into the project:

● Copy files

○ Link to files

☑ Create link locations relative to: PROJECT_LOC

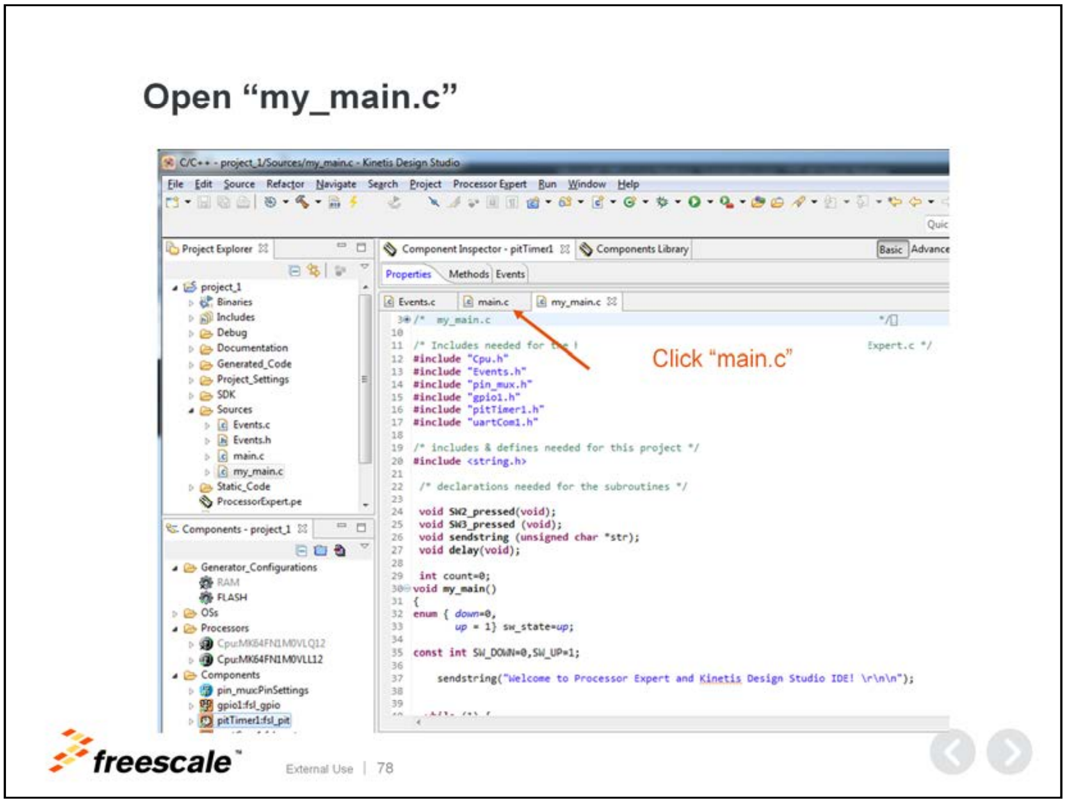
Configure Drag and Drop Settings...

Select "Copy Files"

Click "OK"

Select the appropriate import. We will use "Copy files". This allows for the user to modify the sources without effecting the original sources.

## Open “my_main.c”

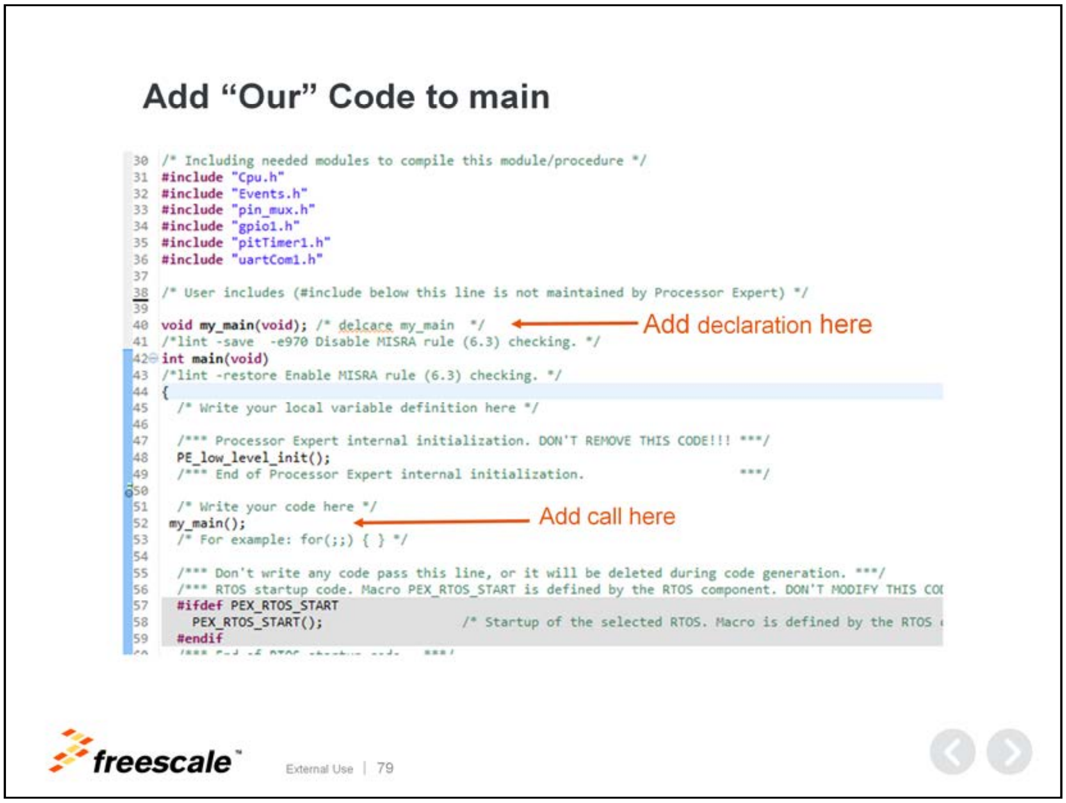


```
/*  my_main.c                                                            */

/* Includes needed for the !                                  Expert.c */
#include "Cpu.h"
#include "Events.h"
#include "pin_mux.h"
#include "gpio1.h"
#include "pitTimer1.h"
#include "uartCom1.h"

/* includes & defines needed for this project */
#include <string.h>

 /* declarations needed for the subroutines */

 void SW2_pressed(void);
 void SW3_pressed (void);
 void sendstring (unsigned char *str);
 void delay(void);

 int count=0;
void my_main()
{
enum { down=0,
       up = 1} sw_state=up;

const int SW_DOWN=0,SW_UP=1;

    sendstring("Welcome to Processor Expert and Kinetis Design Studio IDE! \r\n\n");
```

Click “main.c”

freescale™ External Use | 78

Now the file is included in the “Sources” folder.



## Add “Our” Code to main

```
30 /* Including needed modules to compile this module/procedure */
31 #include "Cpu.h"
32 #include "Events.h"
33 #include "pin_mux.h"
34 #include "gpio1.h"
35 #include "pitTimer1.h"
36 #include "uartCom1.h"
37
38 /* User includes (#include below this line is not maintained by Processor Expert) */
39
40 void my_main(void); /* delcare my_main  */
41 /*lint -save  -e970 Disable MISRA rule (6.3) checking. */
42 int main(void)
43 /*lint -restore Enable MISRA rule (6.3) checking. */
44 {
45   /* Write your local variable definition here */
46
47   /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
48   PE_low_level_init();
49   /*** End of Processor Expert internal initialization.                    ***/
50
51   /* Write your code here */
52  my_main();
53   /* For example: for(;;) { } */
54
55   /*** Don't write any code pass this line, or it will be deleted during code generation. ***/
56   /*** RTOS startup code. Macro PEX_RTOS_START is defined by the RTOS component. DON'T MODIFY THIS COD
57   #ifdef PEX_RTOS_START
58     PEX_RTOS_START();                  /* Startup of the selected RTOS. Macro is defined by the RTOS 
59   #endif
```

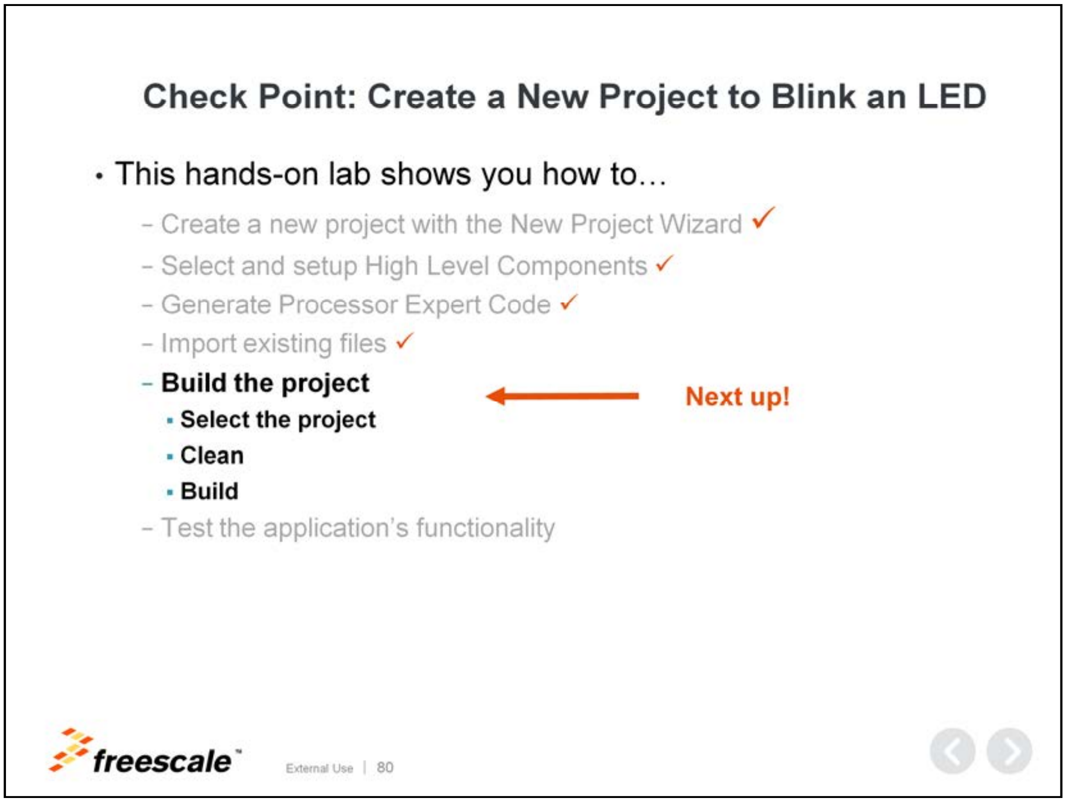
Add declaration here

Add call here

freescale™ External Use | 79

Around line 40 to add the declaration of “my_main” and the call around line 51.

Be sure the call is included after the line “/* Write your code here */

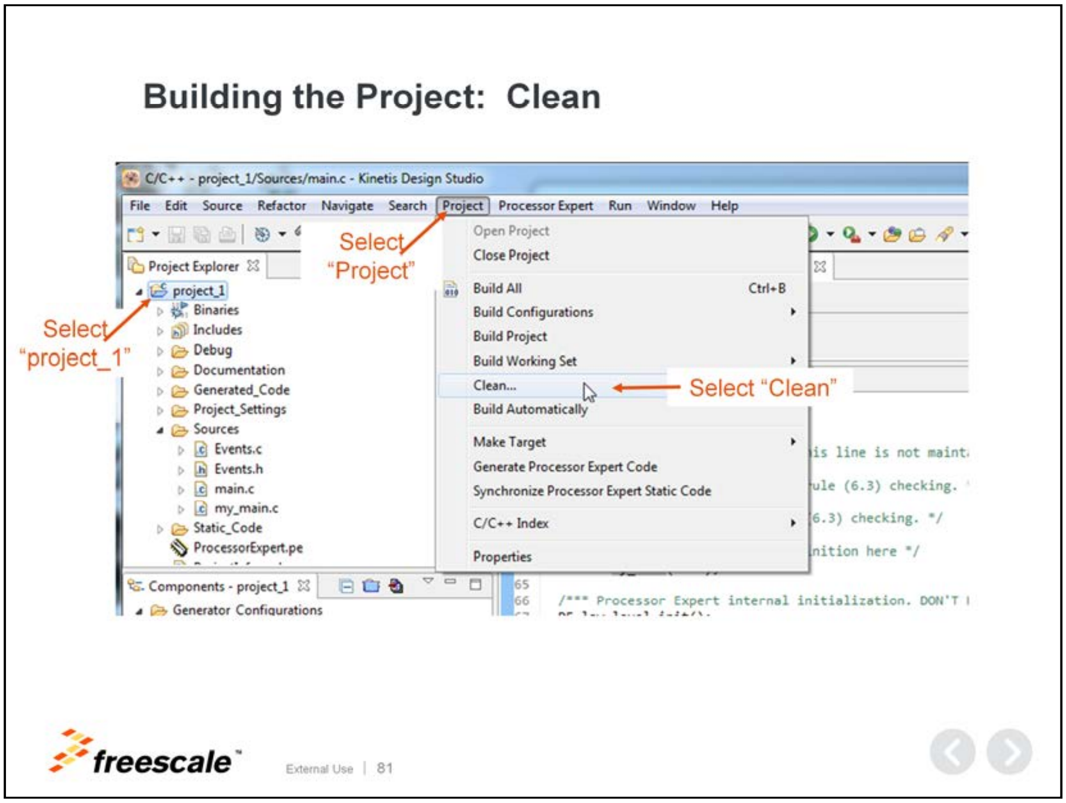


## Check Point: Create a New Project to Blink an LED

- This hands-on lab shows you how to…
  - Create a new project with the New Project Wizard ✓
  - Select and setup High Level Components ✓
  - Generate Processor Expert Code ✓
  - Import existing files ✓
  - **Build the project**
    - **Select the project**
    - **Clean**
    - **Build**
  - Test the application's functionality

**Next up!**

External Use | 80



## Building the Project: Clean

Select “project_1”

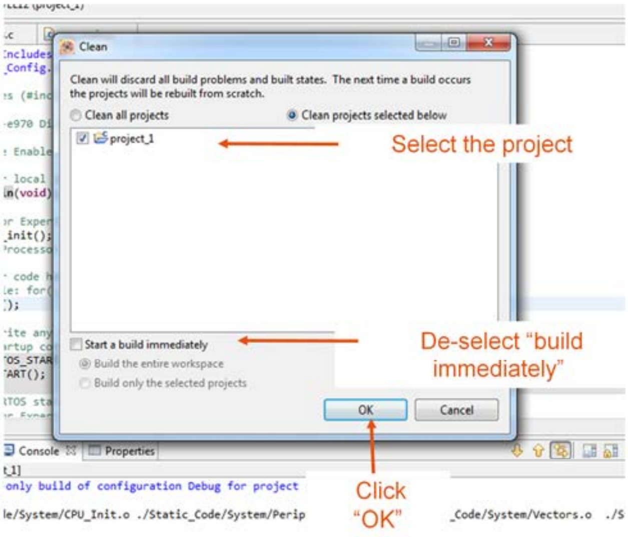
Select “Project”

Select “Clean”

freescale External Use | 81

Right now we will clean the project. I make it a habit to clean a project especially when there are files imported or if the entire project is imported. This ensures that there are no artifacts carried over from the import.

## Building the Project: Clean



Clean

Clean will discard all build problems and built states. The next time a build occurs the projects will be rebuilt from scratch.

○ Clean all projects ● Clean projects selected below

☑ project_1

□ Start a build immediately

● Build the entire workspace

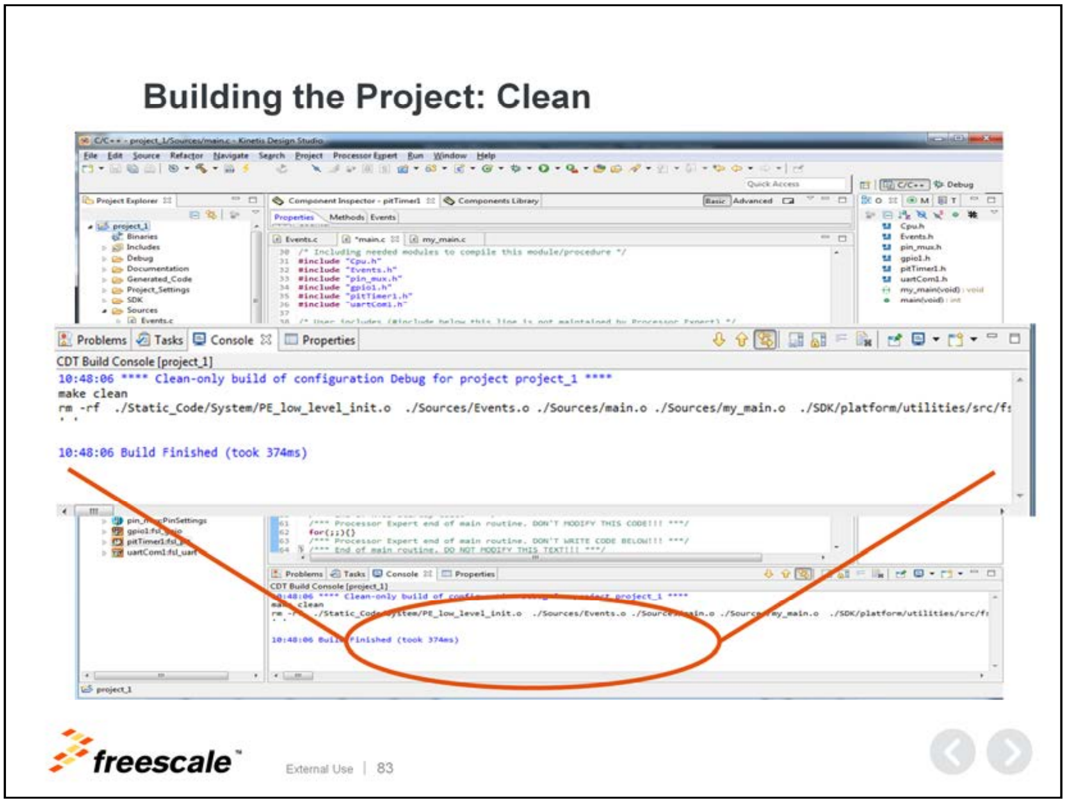
○ Build only the selected projects

```
t_1]
only build of configuration Debug for project
le/System/CPU_Init.o ./Static_Code/System/Perip        _Code/System/Vectors.o  ./S
```

Select the project

De-select "build immediately"

Click "OK"



## Building the Project: Clean

```
/* Including needed modules to compile this module/procedure */
#include "Cpu.h"
#include "Events.h"
#include "pin_mux.h"
#include "gpio1.h"
#include "pitTimer1.h"
#include "uartCom1.h"
```

CDT Build Console [project_1]

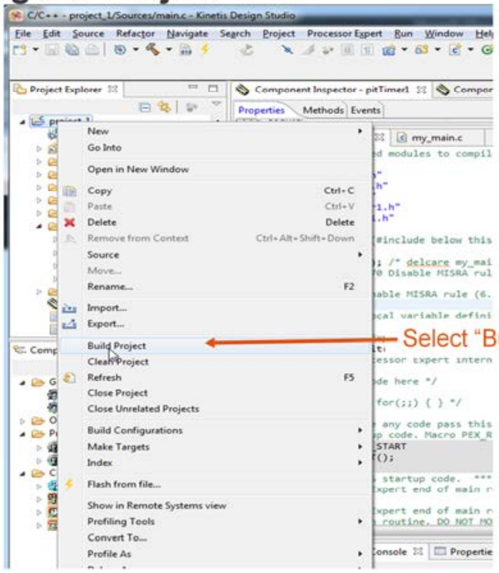
```
10:48:06 **** Clean-only build of configuration Debug for project project_1 ****
make clean
rm -rf  ./Static_Code/System/PE_low_level_init.o  ./Sources/Events.o ./Sources/main.o ./Sources/my_main.o  ./SDK/platform/utilities/src/fs
' '

10:48:06 Build Finished (took 374ms)
```

External Use | 83

The project is cleaned and ready to build…

## Building the Project

Right Click On Project_1

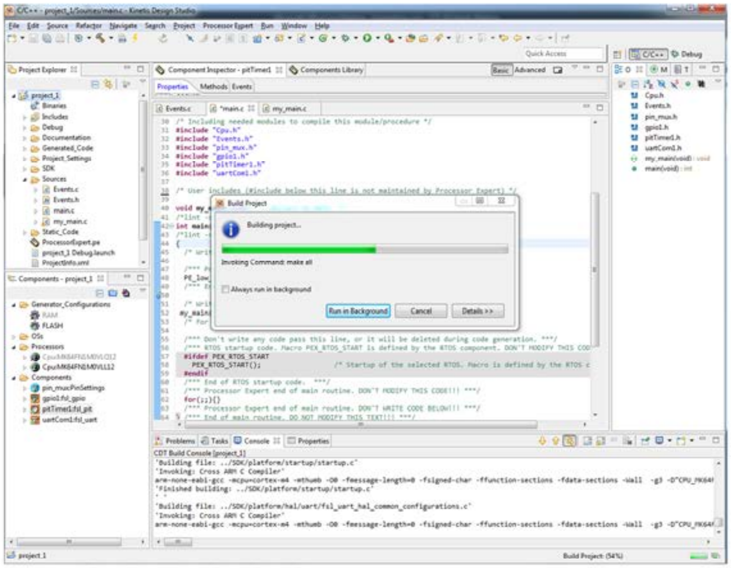


Select "Build"

External Use | 84

Right click on the project you wish to build. Then select build. One could also select the project then using the menus "Project -> Build".

## Building the Project



```
/* Including needed modules to compile this module/procedure */
#include "Cpu.h"
#include "Events.h"
#include "pin_mux.h"
#include "gpio1.h"
#include "pitTimer1.h"
#include "uartCom1.h"

/* User includes (#include below this line is not maintained by Processor Expert) */

void my_
/*lint -
int main
/*lint -
{
  /* Writ

  /*** Pr
  PE_low_
  /*** En

  /* Writ
  my_main
  /* For

  /*** Don't write any code pass this line, or it will be deleted during code generation. ***/
  /*** RTOS startup code. Macro PEX_RTOS_START is defined by the RTOS component. DON'T MODIFY THIS COD
  #ifdef PEX_RTOS_START
    PEX_RTOS_START();                  /* Startup of the selected RTOS. Macro is defined by the RTOS c
  #endif
  /*** End of RTOS startup code.  ***/
  /*** Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! ***/
  for(;;){}
  /*** Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! ***/
} /*** End of main routine. DO NOT MODIFY THIS TEXT!!! ***/
```

Building project...

Invoking Command: make all

☐ Always run in background

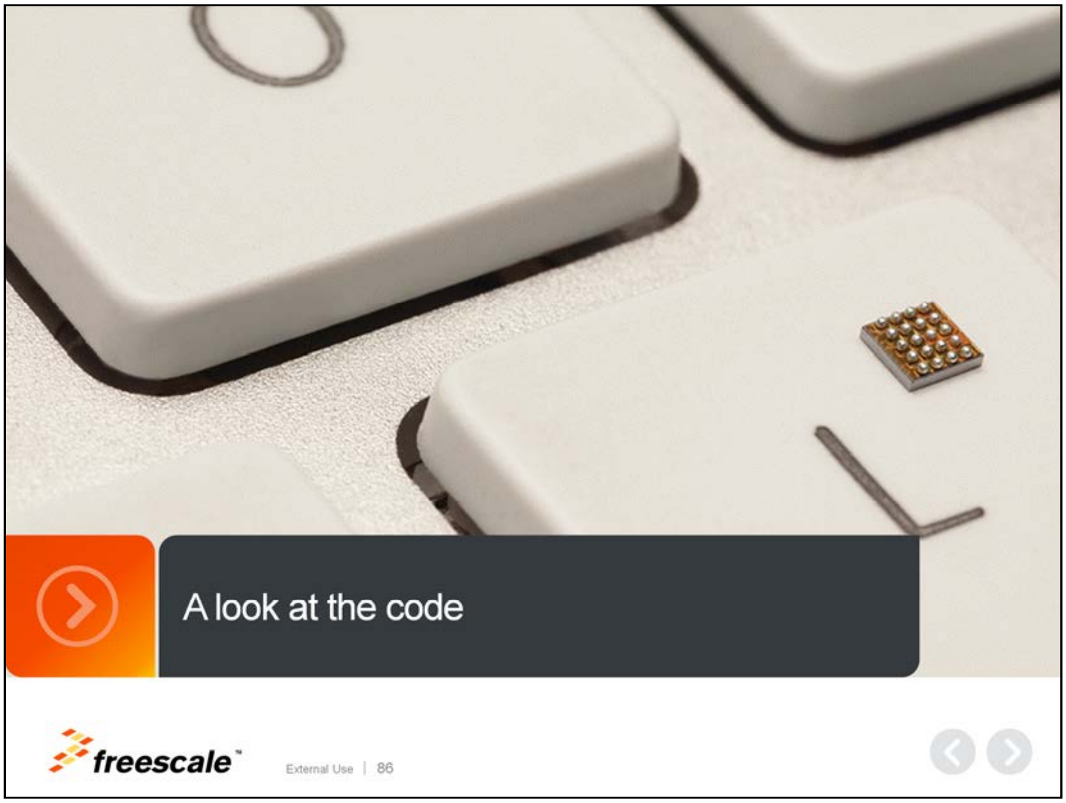
CDT Build Console [project_1]

```
'Building file: ../SDK/platform/startup/startup.c'
'Invoking: Cross ARM C Compiler'
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -O0 -fmessage-length=0 -fsigned-char -ffunction-sections -fdata-sections -Wall  -g3 -D"CPU_MK64
'Finished building: ../SDK/platform/startup/startup.c'
' '
'Building file: ../SDK/platform/hal/uart/fsl_uart_hal_common_configurations.c'
'Invoking: Cross ARM C Compiler'
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -O0 -fmessage-length=0 -fsigned-char -ffunction-sections -fdata-sections -Wall  -g3 -D"CPU_MK64
```

Build Project: (54%)

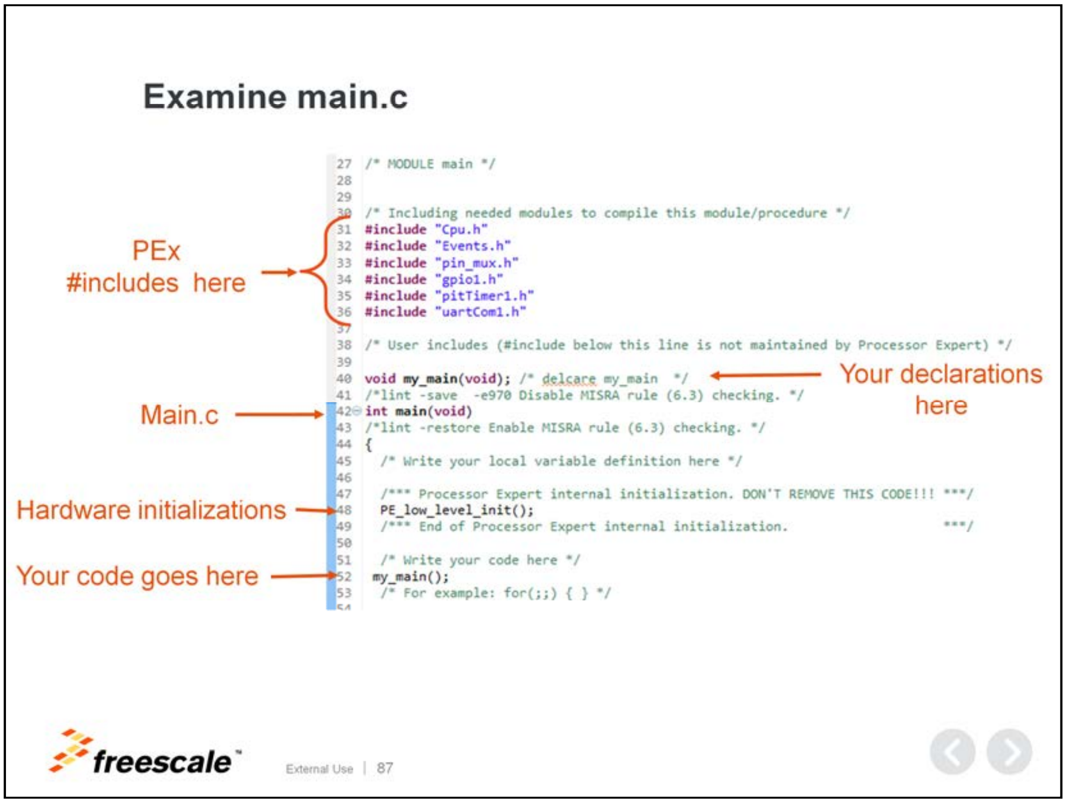
freescale™ External Use | 85

Build Progress – Just like most Eclipse based tools.



# A look at the code

External Use | 86



## Examine main.c

```
/* MODULE main */


/* Including needed modules to compile this module/procedure */
#include "Cpu.h"
#include "Events.h"
#include "pin_mux.h"
#include "gpio1.h"
#include "pitTimer1.h"
#include "uartCom1.h"

/* User includes (#include below this line is not maintained by Processor Expert) */

void my_main(void); /* delcare my_main  */
/*lint -save  -e970 Disable MISRA rule (6.3) checking. */
int main(void)
/*lint -restore Enable MISRA rule (6.3) checking. */
{
  /* 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 */
 my_main();
  /* For example: for(;;) { } */
```

PEx #includes here

Your declarations here

Main.c

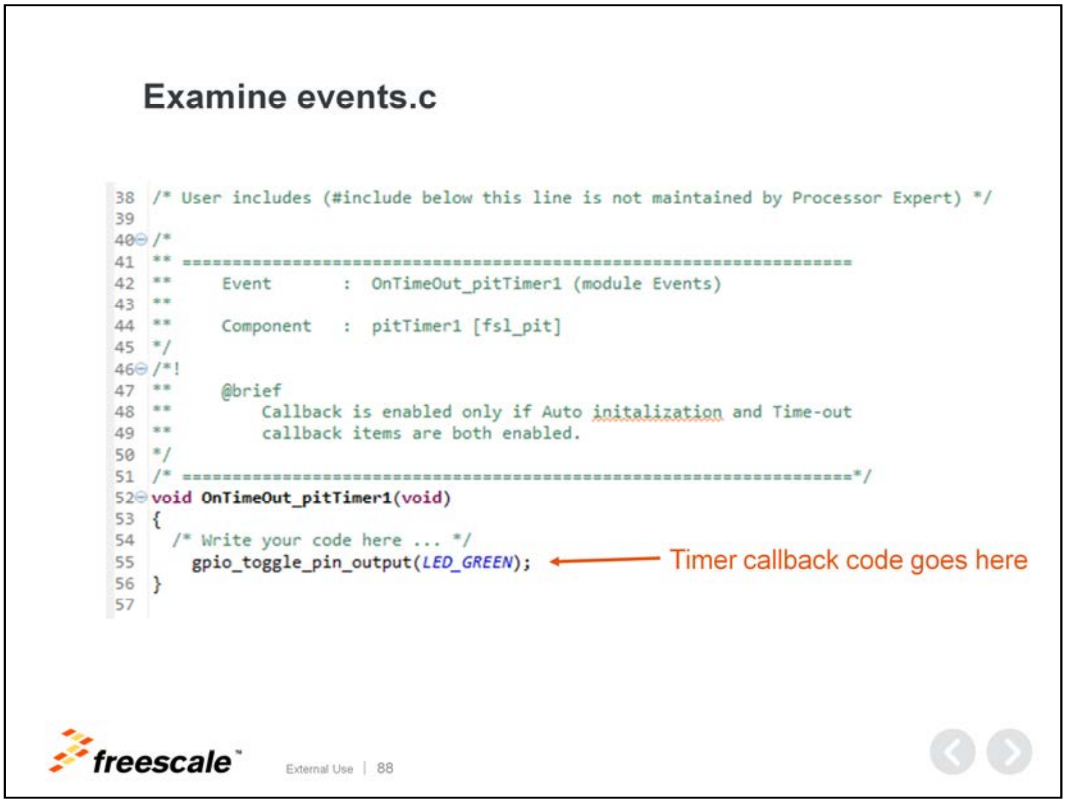
Hardware initializations

Your code goes here

External Use | 87

These includes are what is generated and needed by Processor Expert.

This is an auto generated file. All we need to add is our declarations and a call to our code.



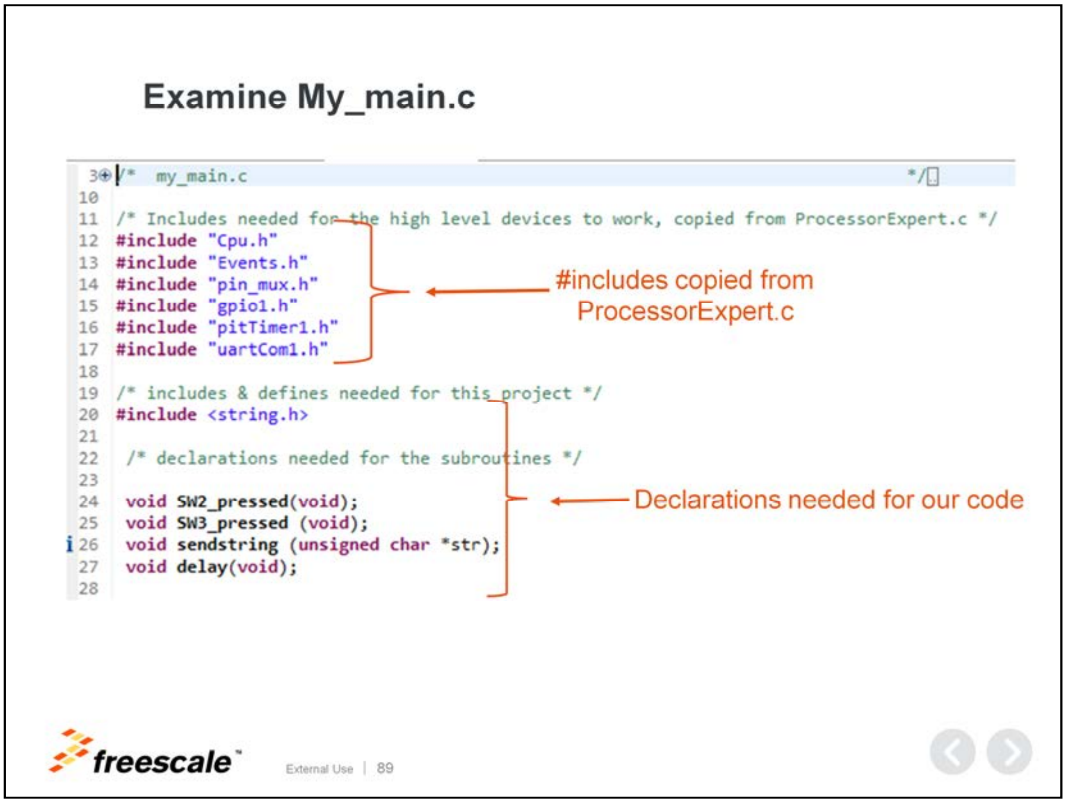
## Examine events.c

```
/* User includes (#include below this line is not maintained by Processor Expert) */

/*
** ===================================================================
**     Event       :  OnTimeOut_pitTimer1 (module Events)
**
**     Component   :  pitTimer1 [fsl_pit]
*/
/*!
**     @brief
**         Callback is enabled only if Auto initalization and Time-out
**         callback items are both enabled.
*/
/* ===================================================================*/
void OnTimeOut_pitTimer1(void)
{
  /* Write your code here ... */
    gpio_toggle_pin_output(LED_GREEN);
}

```

Timer callback code goes here

External Use | 88

This set of code is your events handler. Every where you included an interrupt for the peripheral module, PEx generates a routine in this file.



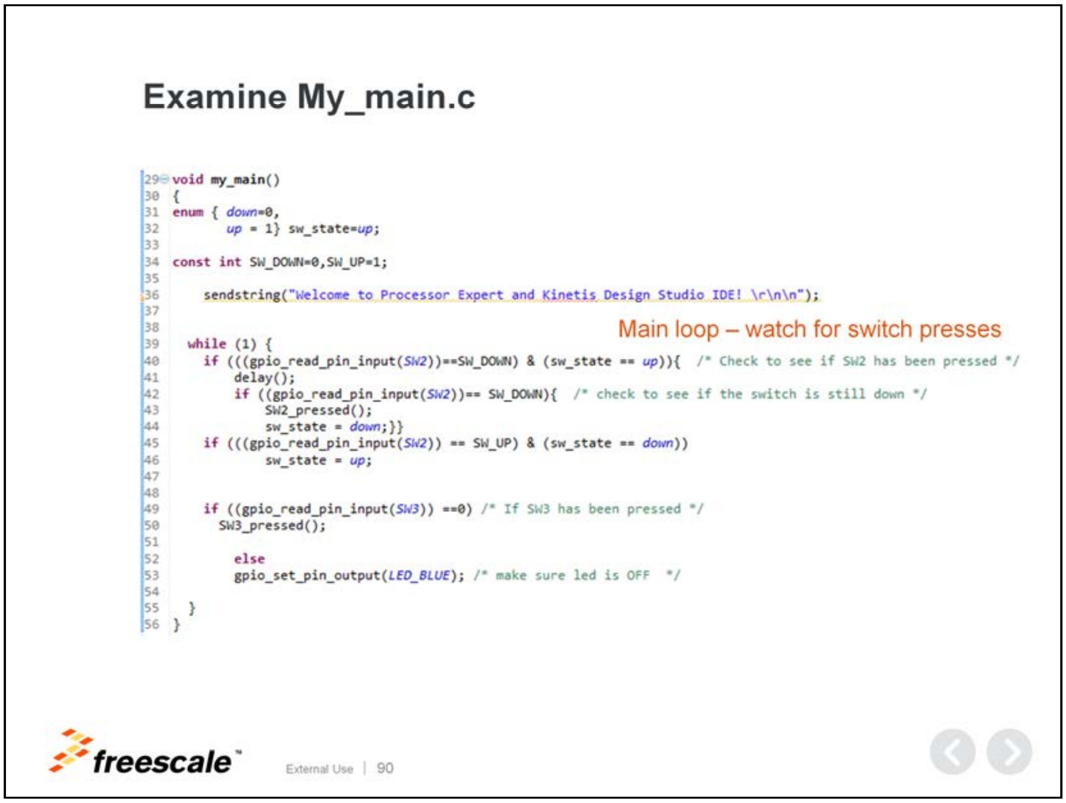
## Examine My_main.c

```
/*  my_main.c                                                    */

/* Includes needed for the high level devices to work, copied from ProcessorExpert.c */
#include "Cpu.h"
#include "Events.h"
#include "pin_mux.h"
#include "gpio1.h"
#include "pitTimer1.h"
#include "uartCom1.h"

/* includes & defines needed for this project */
#include <string.h>

 /* declarations needed for the subroutines */

 void SW2_pressed(void);
 void SW3_pressed (void);
 void sendstring (unsigned char *str);
 void delay(void);
```

#includes copied from ProcessorExpert.c

Declarations needed for our code

External Use | 89

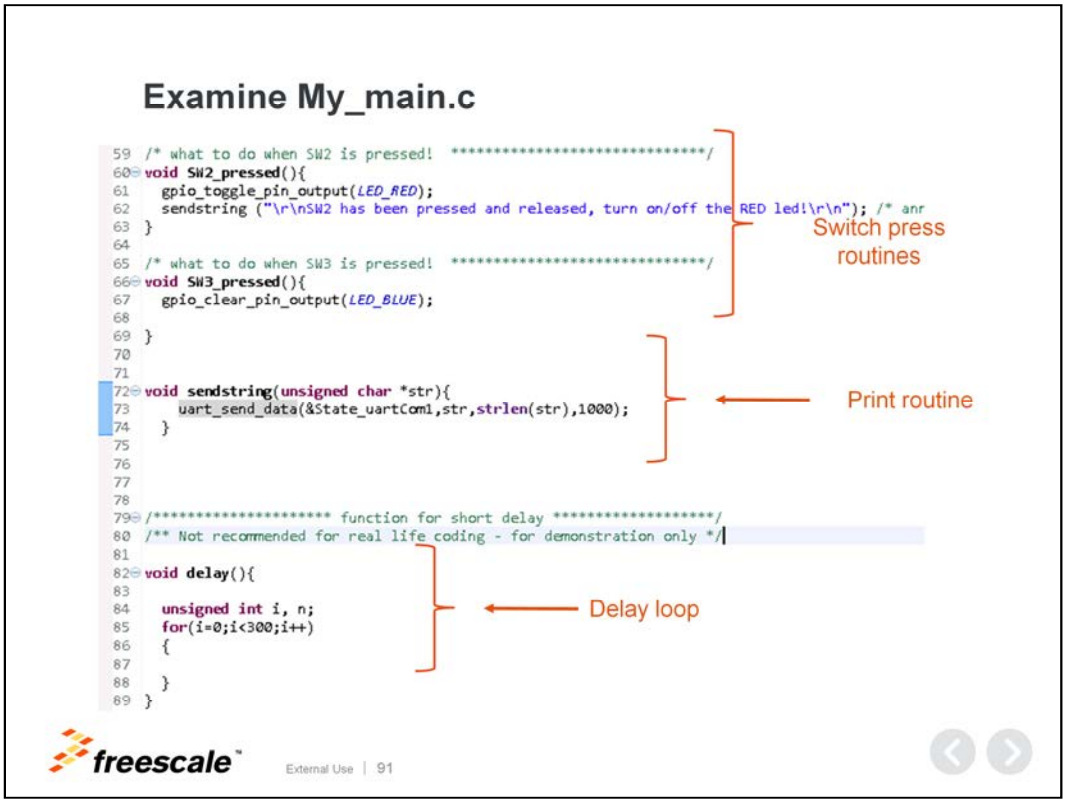


## Examine My_main.c

```
void my_main()
{
enum { down=0,
       up = 1} sw_state=up;

const int SW_DOWN=0,SW_UP=1;

    sendstring("Welcome to Processor Expert and Kinetis Design Studio IDE! \r\n\n");


  while (1) {
    if (((gpio_read_pin_input(SW2))==SW_DOWN) & (sw_state == up)){  /* Check to see if SW2 has been pressed */
        delay();
        if ((gpio_read_pin_input(SW2))== SW_DOWN){  /* check to see if the switch is still down */
            SW2_pressed();
            sw_state = down;}}
    if (((gpio_read_pin_input(SW2)) == SW_UP) & (sw_state == down))
            sw_state = up;


    if ((gpio_read_pin_input(SW3)) ==0) /* If SW3 has been pressed */
      SW3_pressed();

        else
        gpio_set_pin_output(LED_BLUE); /* make sure led is OFF  */

  }
}
```

Main loop – watch for switch presses

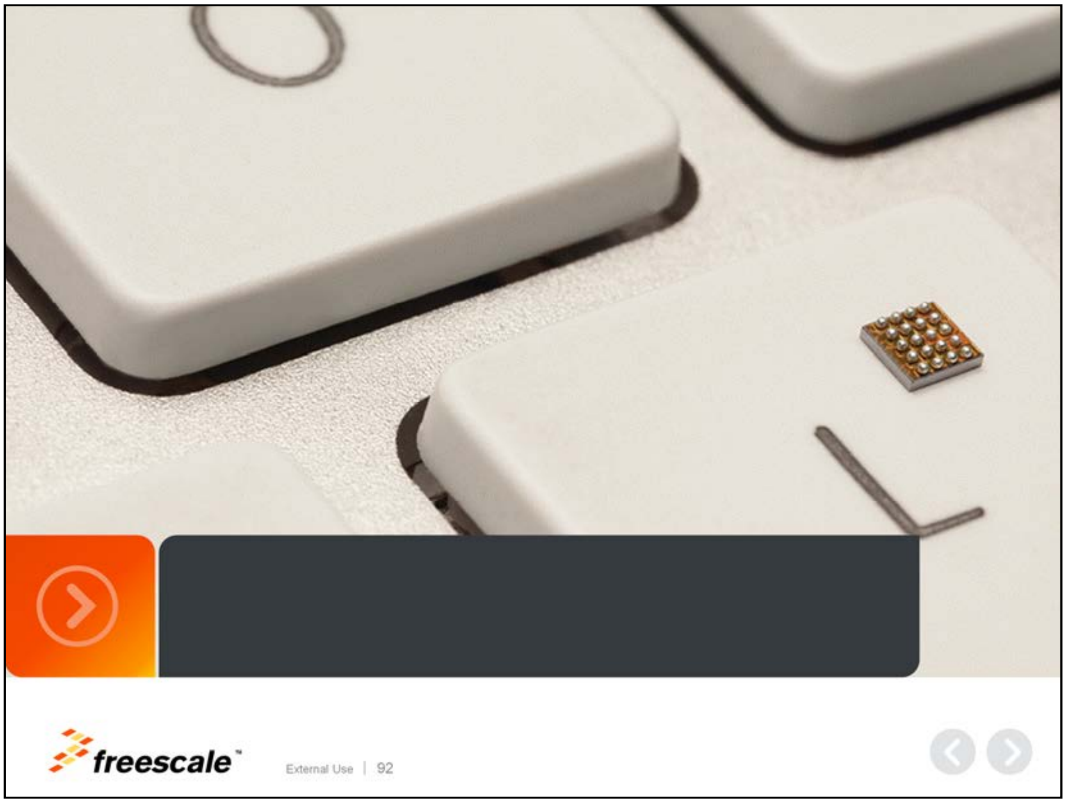
External Use | 90



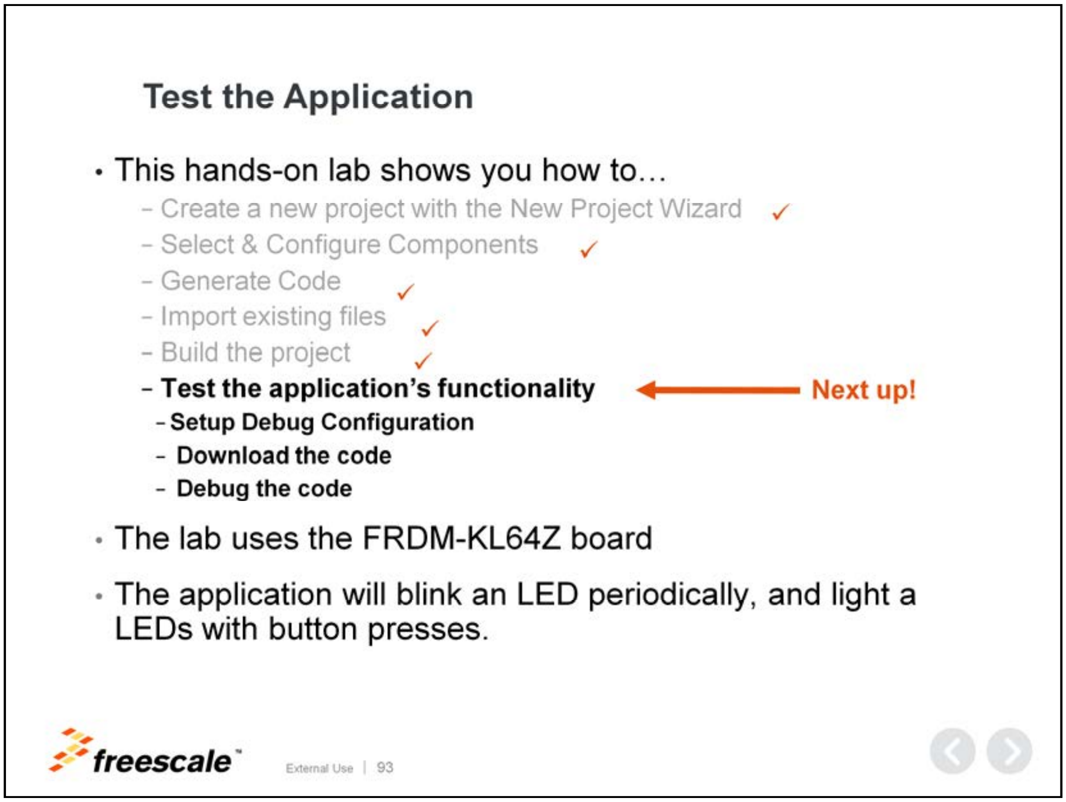
## Examine My_main.c

```
/* what to do when SW2 is pressed!  *******************************/
void SW2_pressed(){
  gpio_toggle_pin_output(LED_RED);
  sendstring ("\r\nSW2 has been pressed and released, turn on/off the RED led!\r\n"); /* anr
}

/* what to do when SW3 is pressed!  *******************************/
void SW3_pressed(){
  gpio_clear_pin_output(LED_BLUE);

}


void sendstring(unsigned char *str){
    uart_send_data(&State_uartCom1,str,strlen(str),1000);
  }




/********************* function for short delay *******************/
/** Not recommended for real life coding - for demonstration only */

void delay(){

  unsigned int i, n;
  for(i=0;i<300;i++)
  {

  }
}
```

External Use | 91



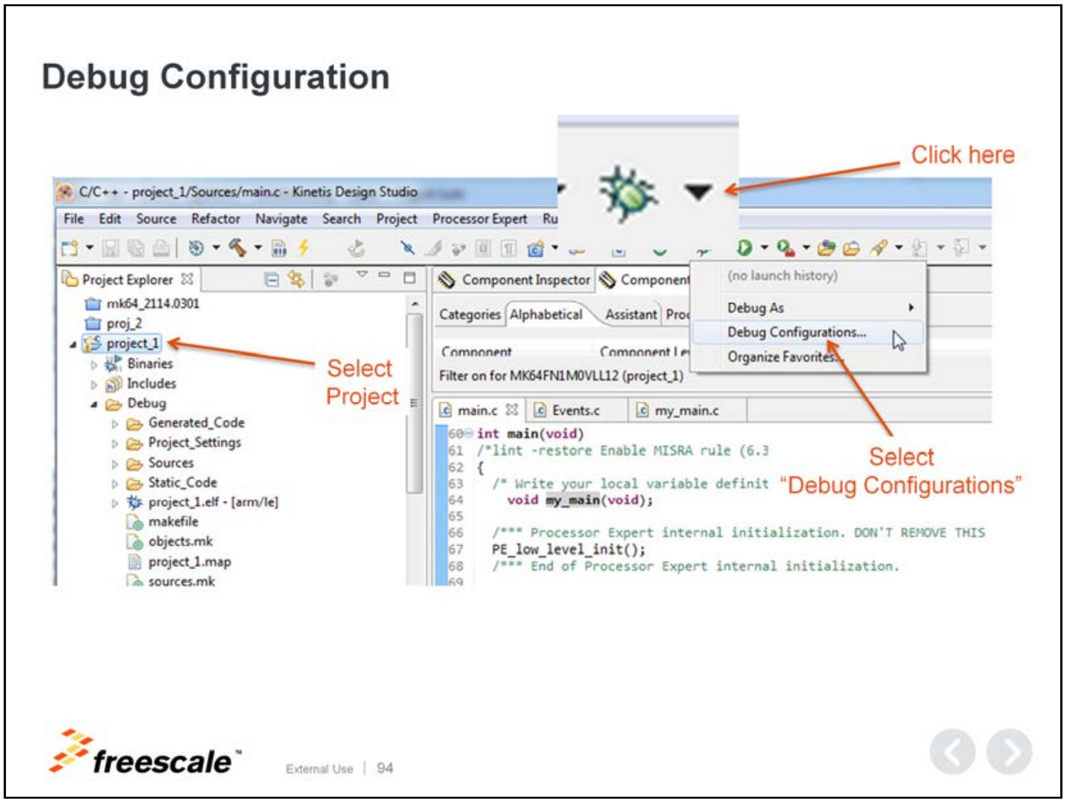
External Use | 92



## Test the Application

- This hands-on lab shows you how to…
  - Create a new project with the New Project Wizard ✓
  - Select & Configure Components ✓
  - Generate Code ✓
  - Import existing files ✓
  - Build the project ✓
  - **Test the application's functionality** ← **Next up!**
    - **Setup Debug Configuration**
    - **Download the code**
    - **Debug the code**
- The lab uses the FRDM-KL64Z board
- The application will blink an LED periodically, and light a LEDs with button presses.

External Use | 93



## Debug Configuration

Click here

Select Project

Select "Debug Configurations"

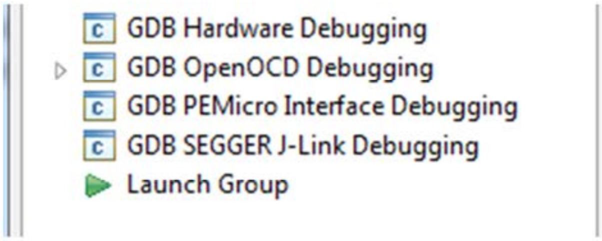
```
int main(void)
/*lint -restore Enable MISRA rule (6.3
{
  /* Write your local variable definit
    void my_main(void);

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

External Use | 94

The first thing that you need to do before debugging, is to setup the debug configuration. To get to the debug configuration for your project, highlight the project (by selecting it) and then click on the "▼", and select "Debug Configurations…"

## Debug Configuration

- Multiple GDB Debug Configurations Supported

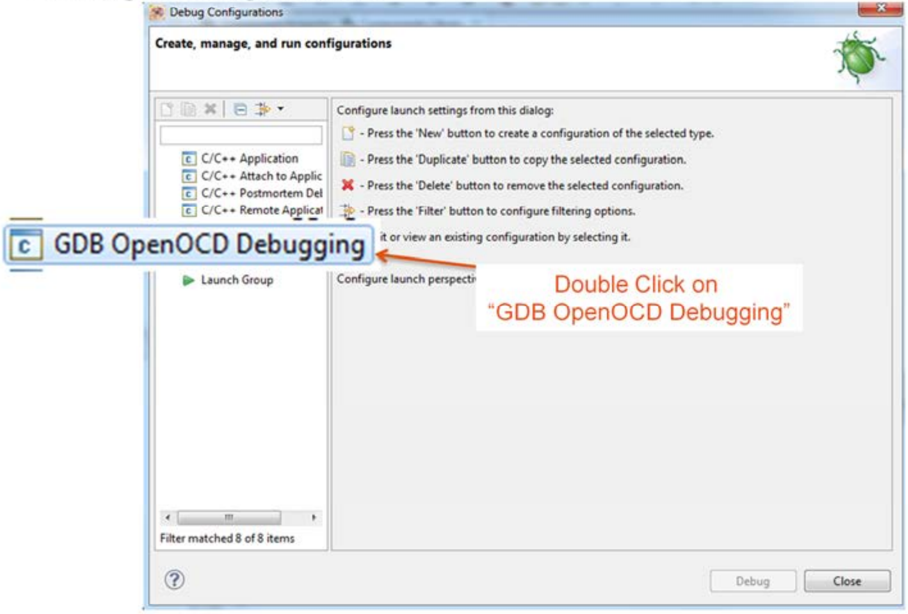


GDB Hardware Debugging
GDB OpenOCD Debugging
GDB PEMicro Interface Debugging
GDB SEGGER J-Link Debugging
Launch Group

- For documentation on how to use P&E Micro and Segger, see their respective web pages.

KDS is based on stock eclipse with our Processor Expert Plug-ins and base debug configurations. The base debug configurations are GDB. With this version the GDB includes OpenOCD, P&E Micro and Segger J-Link.

## Debug Configuration



Configure launch settings from this dialog:

- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- it or view an existing configuration by selecting it.

Configure launch perspecti

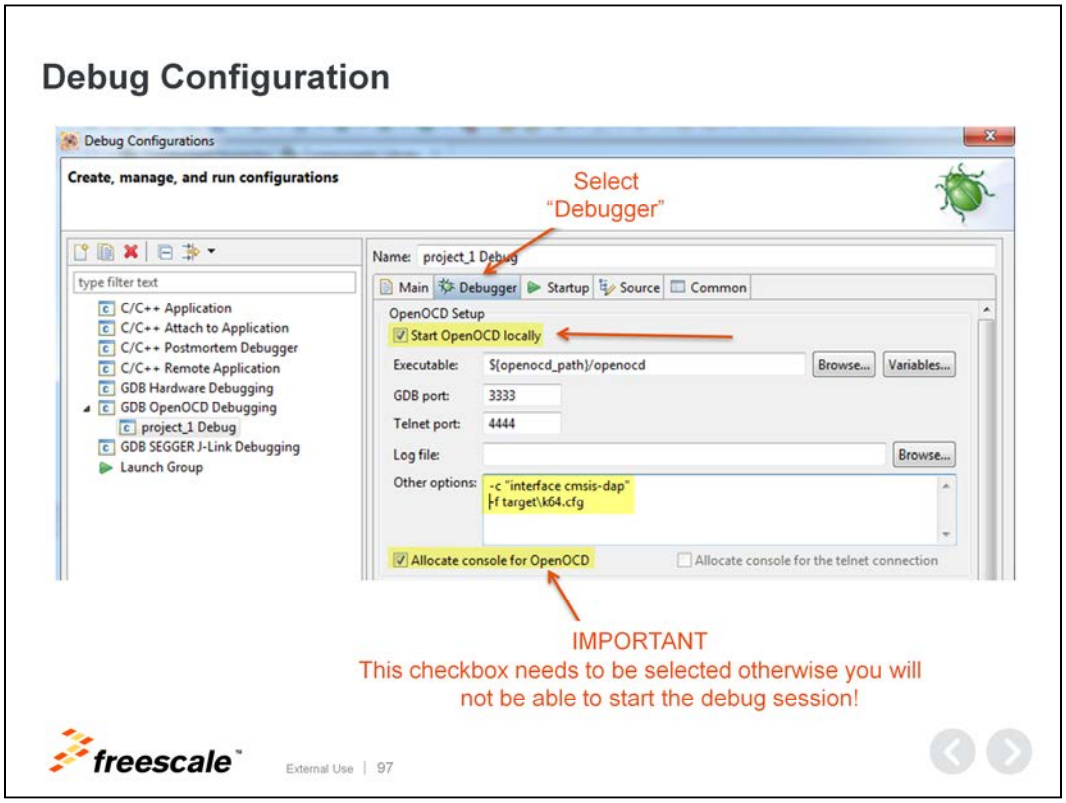
Filter matched 8 of 8 items

Double Click on
“GDB OpenOCD Debugging”

freescale™ External Use | 96

KDS is based on stock eclipse with our Processor Expert Plug-ins and base debug configurations. The base debug configurations are GDB. With this version the GDB includes OpenOCD, P&E Micro and Segger J-Link.

# Debug Configuration



Name: project_1 Debug

OpenOCD Setup

☑ Start OpenOCD locally

Executable: ${openocd_path}/openocd

GDB port: 3333

Telnet port: 4444

Log file:

Other options:
```
-c "interface cmsis-dap"
-f target\k64.cfg
```

☑ Allocate console for OpenOCD ☐ Allocate console for the telnet connection

Select "Debugger"

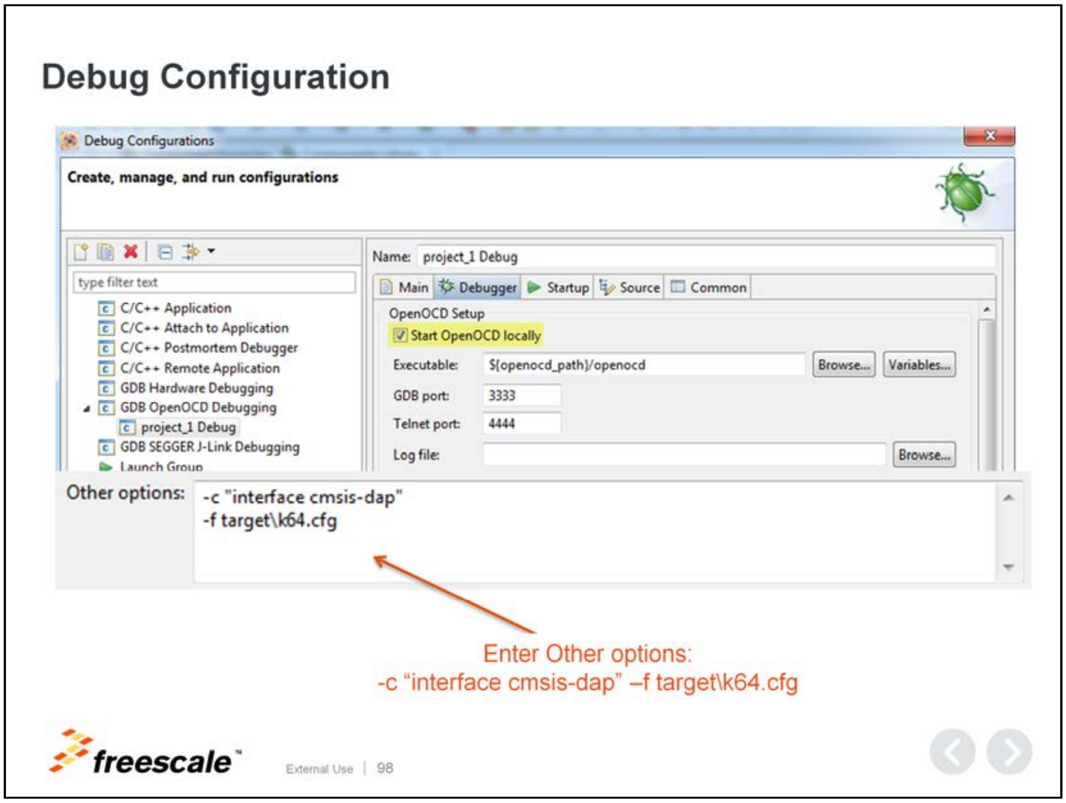
IMPORTANT
This checkbox needs to be selected otherwise you will not be able to start the debug session!

External Use | 97

The "Start OpenOCD locally" option allows for the OpenOCD executable to be started by the tools. The other option is to start the OpenOCD manually and then KDS will find it via the GDB port.

The "Allocate console…" selection MUST be selected in order to start the debug session.

With GDB as the debugger engine, this will allow for remote debugging. The tools on one machine, and the UUT and GDB server on another machine. To set that up, use the GDB Hardware Debugging and setup the server IP address, connection (JTAG Device), and port number of the remote machine.



Debug Configuration

Debug Configurations

Create, manage, and run configurations

Name: project_1 Debug

OpenOCD Setup

☑ Start OpenOCD locally

Executable: ${openocd_path}/openocd

GDB port: 3333

Telnet port: 4444

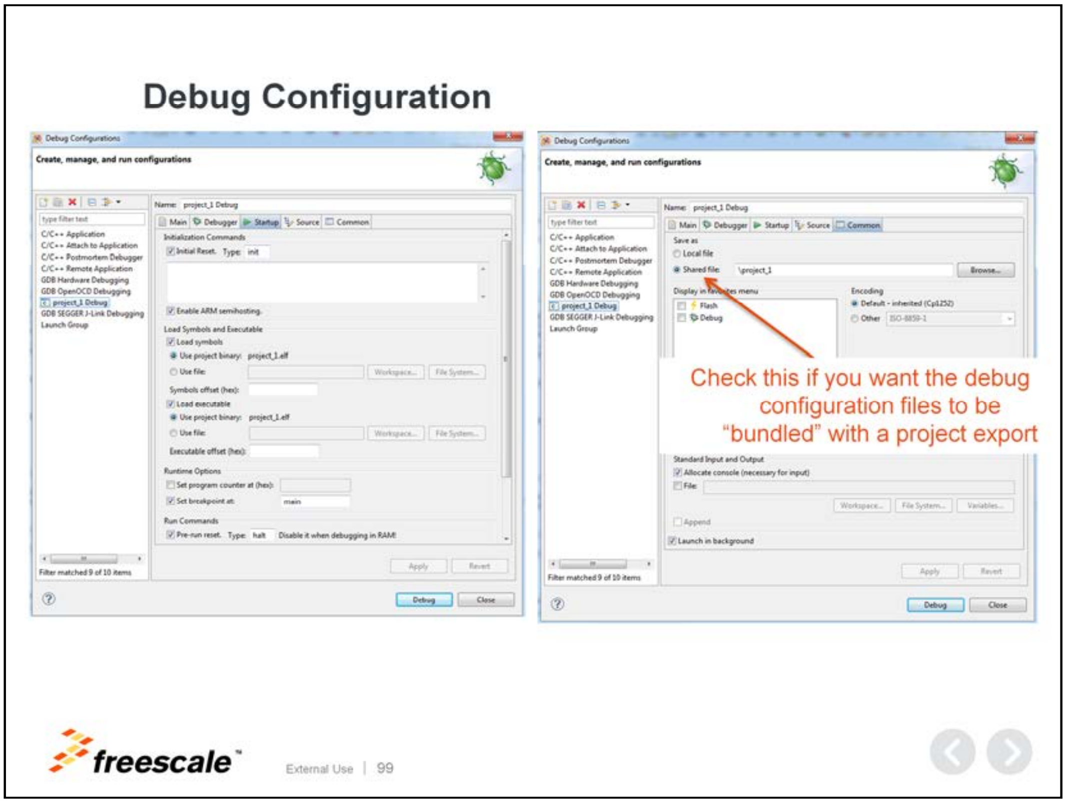
Log file:

Other options:
```
-c "interface cmsis-dap"
-f target\k64.cfg
```

Enter Other options:
-c “interface cmsis-dap” –f target\k64.cfg

External Use | 98

In Other options, you must enter the type of interface and the target configuration file location. Be sure to enter ***-c “interface cmsis-dap” –f target\k64.cfg*** into the “Other options:” box. This can be on a single line or separate lines. I chose separate lines for readability.



## Debug Configuration

Debug Configurations

Create, manage, and run configurations

Name: project_1 Debug

Startup

Initialization Commands

☑ Initial Reset. Type: init

☑ Enable ARM semihosting.

Load Symbols and Executable

☑ Load symbols

● Use project binary: project_1.elf

○ Use file:

Symbols offset (hex):

☑ Load executable

● Use project binary: project_1.elf

○ Use file:

Executable offset (hex):

Runtime Options

□ Set program counter at (hex):

☑ Set breakpoint at: main

Run Commands

☑ Pre-run reset. Type: halt Disable it when debugging in RAM!

Filter matched 9 of 10 items

Common

Save as

○ Local file

● Shared file: \project_1

Display in favorites menu

□ Flash

□ Debug

Encoding

● Default - inherited (Cp1252)

○ Other ISO-8859-1

Standard Input and Output

☑ Allocate console (necessary for input)

□ File:

□ Append

☑ Launch in background

Filter matched 9 of 10 items

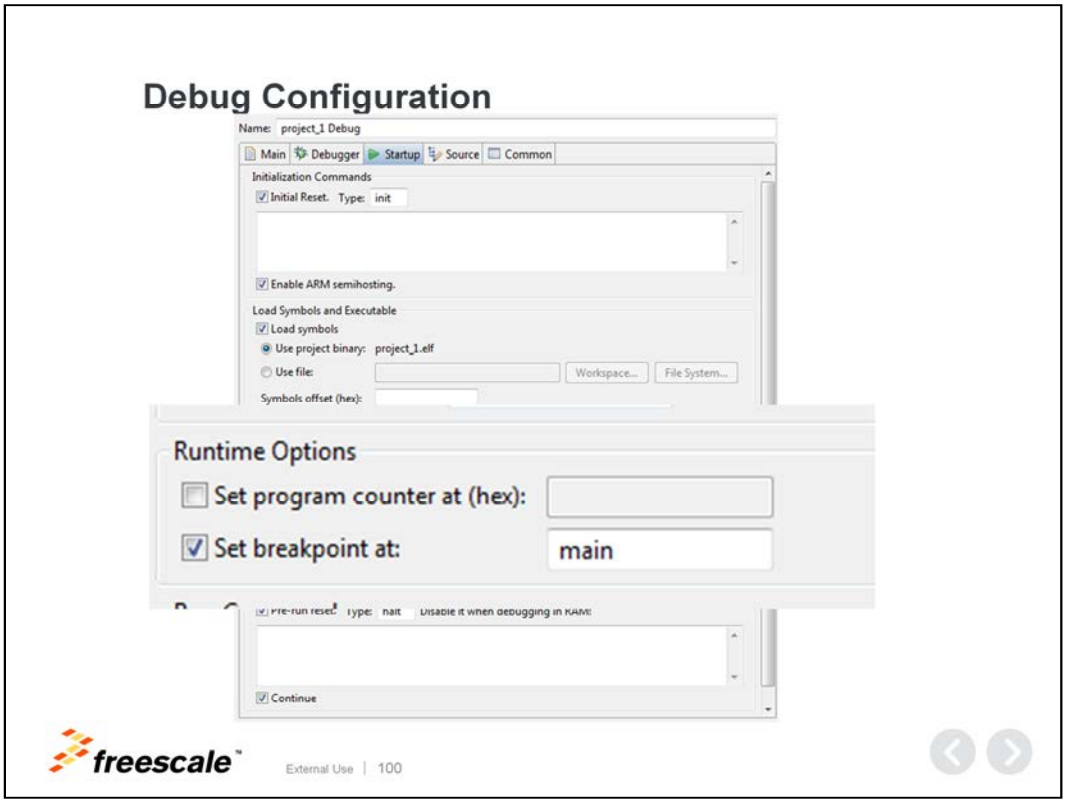
Check this if you want the debug configuration files to be “bundled” with a project export

External Use | 99

Defaults for the “Startup” tab, and recommend to check the “Shared file” for the debug configuration. This will bundle the configuration file with the project when exported.

The Startup panel is where you would set the initial breakpoint for the debugger. By default, it stops at main.

The Common tab has the “Shared file” selection. Check this if you want the debug configuration to be included in the project when you export the project.



## Debug Configuration

Name: project_1 Debug

Initialization Commands

☑ Initial Reset. Type: init

☑ Enable ARM semihosting.

Load Symbols and Executable

☑ Load symbols

● Use project binary: project_1.elf

○ Use file:

Symbols offset (hex):

Runtime Options

□ Set program counter at (hex):

☑ Set breakpoint at: main

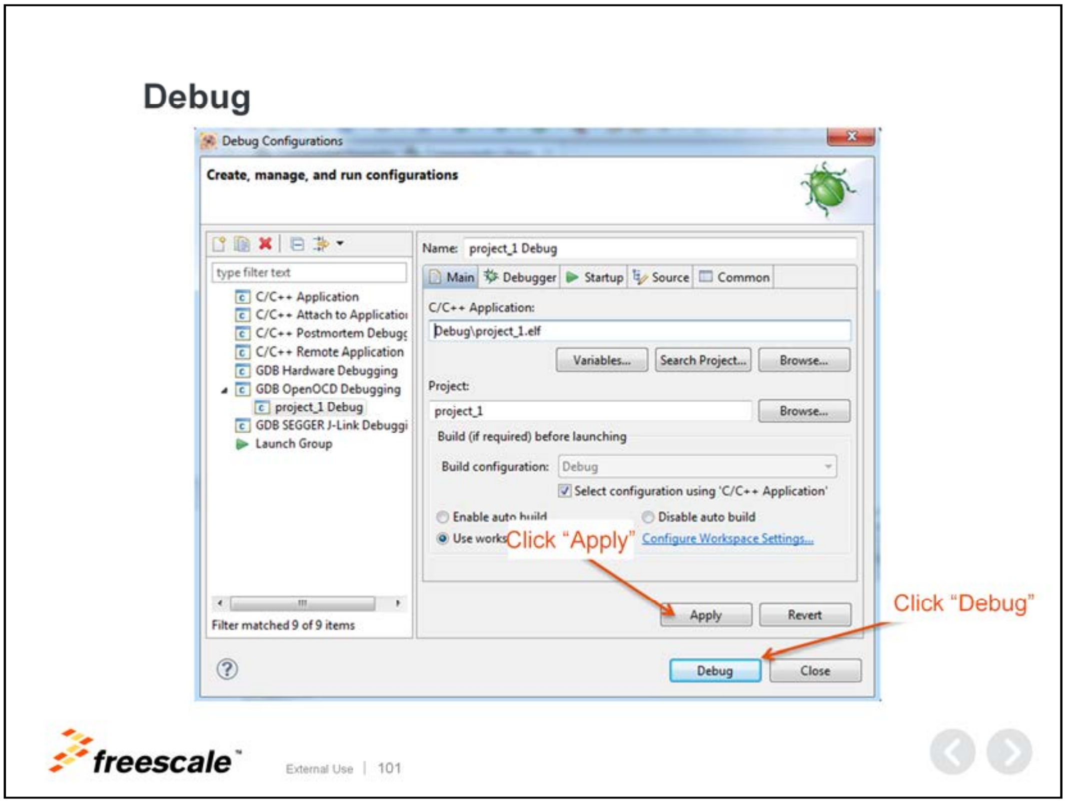
☑ Pre-run reset. Type: halt Disable it when debugging in RAM!

☑ Continue

External Use | 100

All Defaults

The Startup panel is where you would set the initial breakpoint for the debugger. By default, it stops at main. If you want to stop at a particular place, you can enter an address, or function name.



## Debug

Debug Configurations

Create, manage, and run configurations

Name: project_1 Debug

Main | Debugger | Startup | Source | Common

C/C++ Application:

Debug\project_1.elf

Project:

project_1

Build (if required) before launching

Build configuration: Debug

☑ Select configuration using 'C/C++ Application'

○ Enable auto build ○ Disable auto build

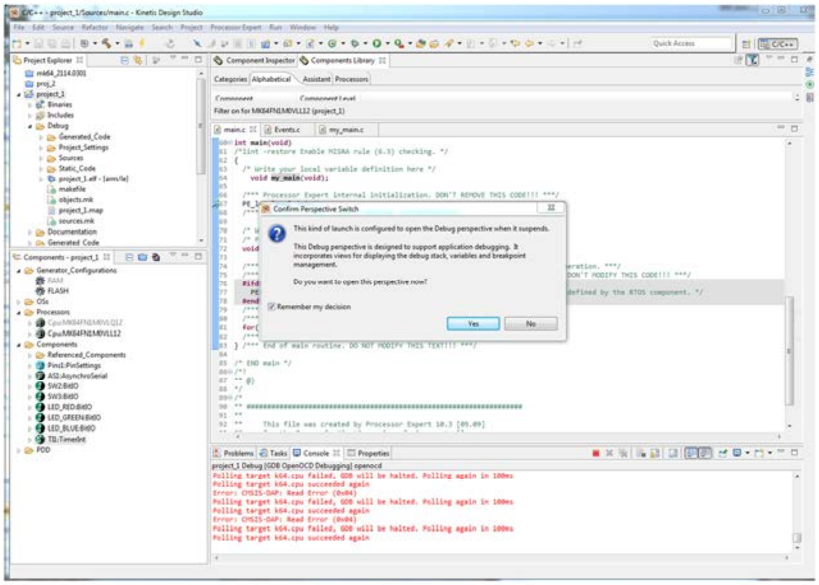
● Use workspace settings Configure Workspace Settings...

Click “Apply”

Click “Debug”

Click “Apply” then click “Debug” this will launch the debugger.

# Debug



Filter on for MK64FN1M0VLL12 (project_1)

```
int main(void)
/*lint -restore Enable MISRA rule (6.3) checking. */
{
  /* Write your local variable definition here */
    void my_main(void);

  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_l
  /***

  /* W
  /* F
  void

  /***                                                        eration. ***/
  /***                                                        DON'T MODIFY THIS CODE!!! ***/
  #ifd
    PE                                                        defined by the RTOS component. */
  #end
  /***
  /***
  for(
  /***
} /*** End of main routine. DO NOT MODIFY THIS TEXT!!! ***/

/* END main */
/*!
** @}
*/
/*
** ###################################################################
**
**     This file was created by Processor Expert 10.3 [05.09]
```

Confirm Perspective Switch

This kind of launch is configured to open the Debug perspective when it suspends.

This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

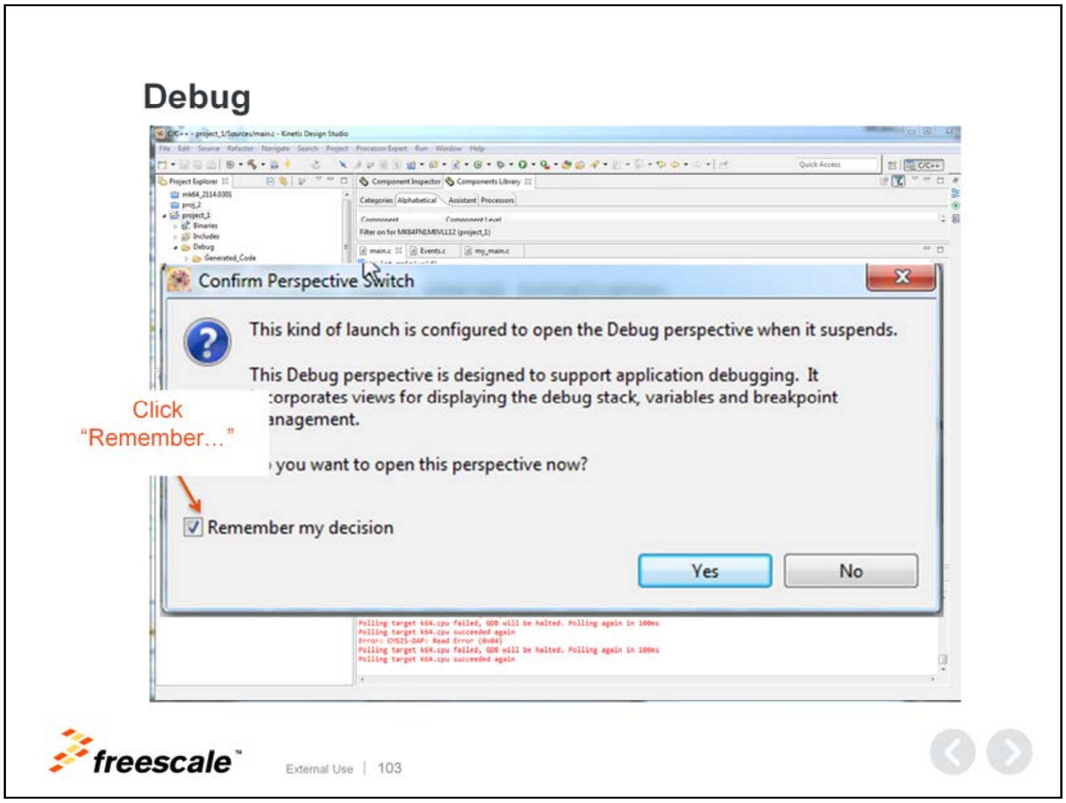
Do you want to open this perspective now?

☑ Remember my decision

Yes No

project_1 Debug [GDB OpenOCD Debugging] openocd

```
Polling target k64.cpu failed, GDB will be halted. Polling again in 100ms
Polling target k64.cpu succeeded again
Error: CMSIS-DAP: Read Error (0x04)
Polling target k64.cpu failed, GDB will be halted. Polling again in 100ms
Polling target k64.cpu succeeded again
Error: CMSIS-DAP: Read Error (0x04)
Polling target k64.cpu failed, GDB will be halted. Polling again in 100ms
Polling target k64.cpu succeeded again
```



## Debug

Confirm Perspective Switch

This kind of launch is configured to open the Debug perspective when it suspends.

This Debug perspective is designed to support application debugging. It :corporates views for displaying the debug stack, variables and breakpoint ınagement.

› you want to open this perspective now?

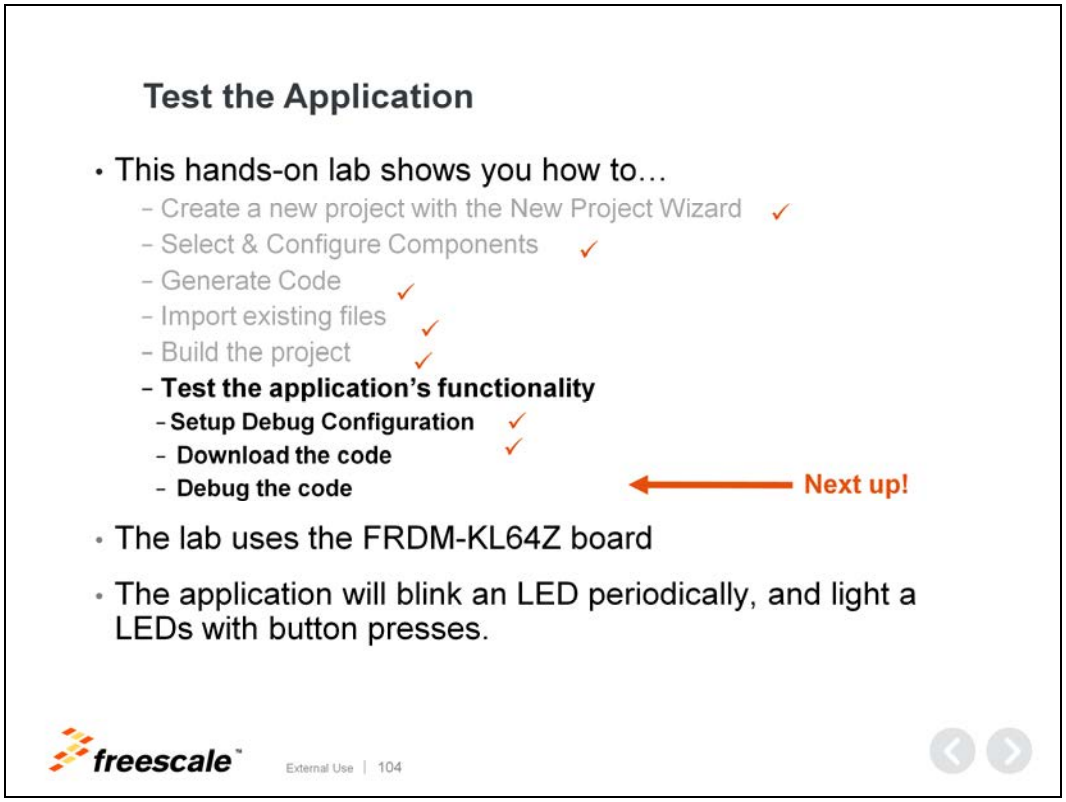
☑ Remember my decision

```
Polling target k64.cpu failed, GDB will be halted. Polling again in 100ms
Polling target k64.cpu succeeded again
Error: CMSIS-DAP: Read Error (0x04)
Polling target k64.cpu failed, GDB will be halted. Polling again in 100ms
Polling target k64.cpu succeeded again
```

Click “Remember…”

External Use | 103

Click checkbox, so you do not get this again. What is happening here, is that Eclipse wants to change to the Debug perspective, If you do not want to see this dialog again, just check this box, and never to see this again.

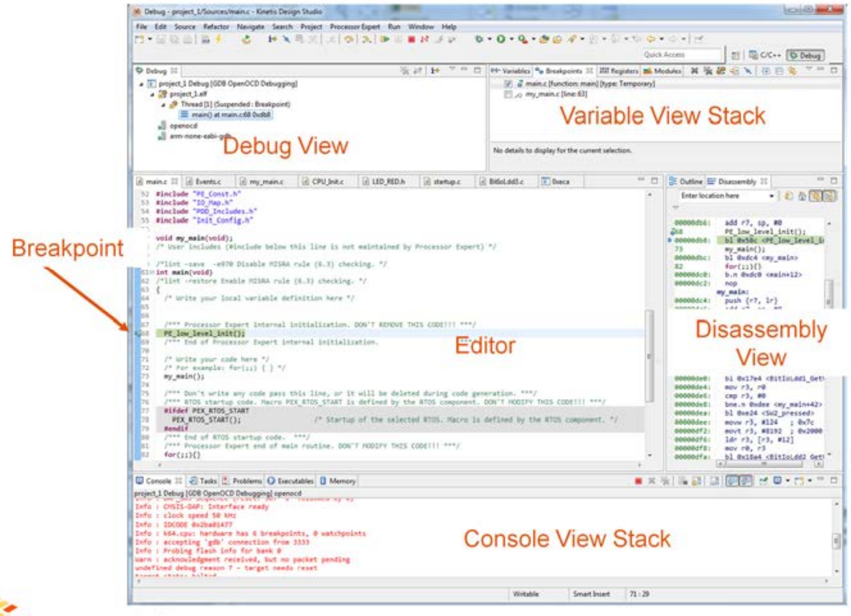


## Test the Application

- This hands-on lab shows you how to…
  - Create a new project with the New Project Wizard ✓
  - Select & Configure Components ✓
  - Generate Code ✓
  - Import existing files ✓
  - Build the project ✓
  - **Test the application's functionality**
    - **Setup Debug Configuration** ✓
    - **Download the code** ✓
    - **Debug the code** ← **Next up!**
- The lab uses the FRDM-KL64Z board
- The application will blink an LED periodically, and light a LEDs with button presses.

External Use | 104

## Application Test: Debug Perspective



```
project_1 Debug [GDB OpenOCD Debugging]
  project_1.elf
    Thread [1] (Suspended : Breakpoint)
      main() at main.c:68 0xdb8
  openocd
  arm-none-eabi-gdb
```

```
main.c [function: main] [type: Temporary]
my_main.c [line: 63]
```

No details to display for the current selection.

```
#include "PE_Const.h"
#include "IO_Map.h"
#include "PDD_Includes.h"
#include "Init_Config.h"

void my_main(void);
/* User includes (#include below this line is not maintained by Processor Expert) */

/*lint -save  -e970 Disable MISRA rule (6.3) checking. */
int main(void)
/*lint -restore Enable MISRA rule (6.3) checking. */
{
  /* 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 */
  /* For example: for(;;) { } */
  my_main();

  /*** Don't write any code pass this line, or it will be deleted during code generation. ***/
  /*** RTOS startup code. Macro PEX_RTOS_START is defined by the RTOS component. DON'T MODIFY THIS CODE!!! ***/
  #ifdef PEX_RTOS_START
    PEX_RTOS_START();                  /* Startup of the selected RTOS. Macro is defined by the RTOS component. */
  #endif
  /*** End of RTOS startup code.  ***/
  /*** Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! ***/
  for(;;){}
```

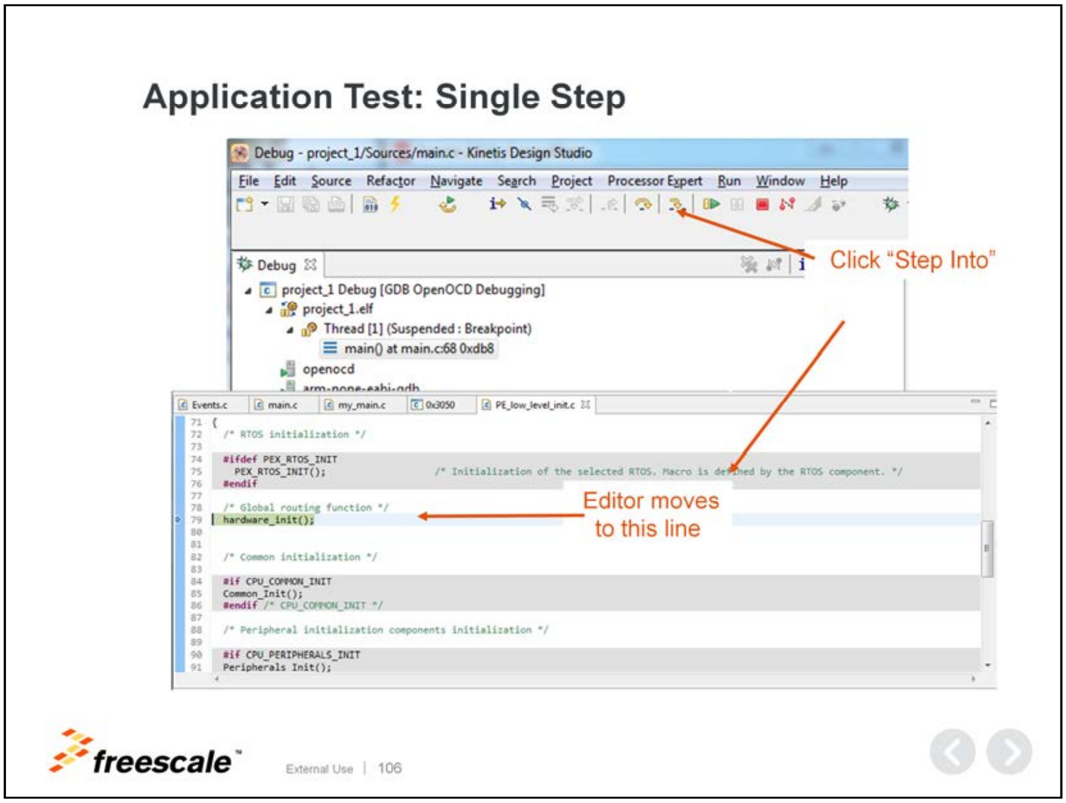
```
00000db6:   add r7, sp, #0
68            PE_low_level_init();
00000db8:   bl 0x58c <PE_low_level_i
73            my_main();
00000dbc:   bl 0xdc4 <my_main>
82            for(;;){}
00000dc0:   b.n 0xdc0 <main+12>
00000dc2:   nop
          my_main:
00000dc4:   push {r7, lr}
...
00000de0:   bl 0x17e4 <BitIoLdd1_Get
00000de4:   mov r3, r0
00000de6:   cmp r3, #0
00000de8:   bne.n 0xdee <my_main+42>
00000dea:   bl 0xe24 <SW2_pressed>
00000dee:   movw r3, #124    ; 0x7c
00000df2:   movt r3, #8192   ; 0x2000
00000df6:   ldr r3, [r3, #12]
00000df8:   mov r0, r3
00000dfa:   bl 0x18a4 <BitIoLdd2_Get
```

```
project_1 Debug [GDB OpenOCD Debugging] openocd
Info : CMSIS-DAP: Interface ready
Info : clock speed 50 kHz
Info : IDCODE 0x2ba01477
Info : k64.cpu: hardware has 6 breakpoints, 0 watchpoints
Info : accepting 'gdb' connection from 3333
Info : Probing flash info for bank 0
Warn : acknowledgment received, but no packet pending
undefined debug reason 7 - target needs reset
```

freescale™ External Use | 105

Immediately after the Download task is completed, the perspective automatically changes into the Debug Perspective. This perspective has a variety of windowing possibilities, by default this perspective gives you 5 major windows. The Debug window, the inspection window, source code window, and the console window. Within the Inspection and Console window space, there are other choices of things to view.

Another item, which is set by default, the code automatically breakpoints after main. This choice can be changed in the Debug Configuration



## Application Test: Single Step

Debug - project_1/Sources/main.c - Kinetis Design Studio
File Edit Source Refactor Navigate Search Project Processor Expert Run Window Help
Debug
Events.c
main.c
my_main.c
0x3050
PE_low_level_init.c

Click “Step Into”

```
project_1 Debug [GDB OpenOCD Debugging]
  project_1.elf
    Thread [1] (Suspended : Breakpoint)
      main() at main.c:68 0xdb8
  openocd
  arm-none-eabi-gdb
```

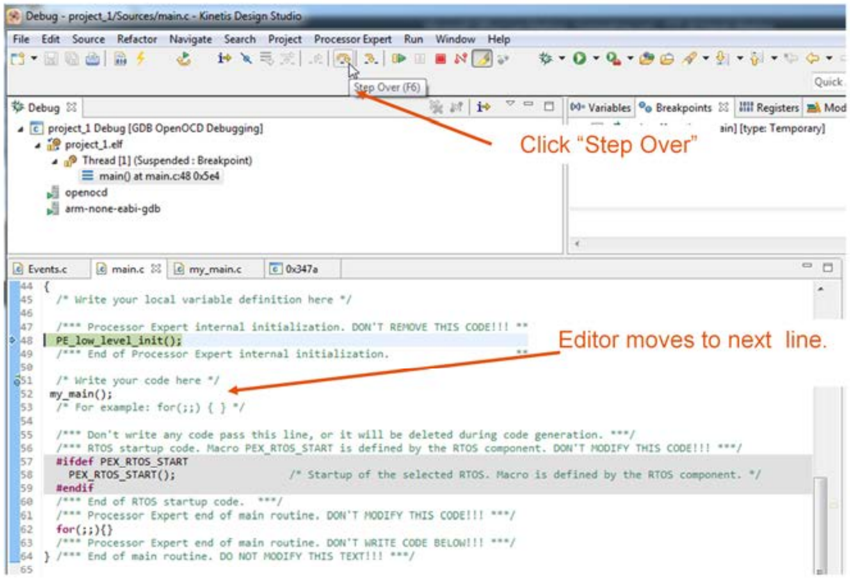
```
{
  /* RTOS initialization */

  #ifdef PEX_RTOS_INIT
    PEX_RTOS_INIT();                    /* Initialization of the selected RTOS. Macro is defined by the RTOS component. */
  #endif

  /* Global routing function */
  hardware_init();


  /* Common initialization */

  #if CPU_COMMON_INIT
  Common_Init();
  #endif /* CPU_COMMON_INIT */

  /* Peripheral initialization components initialization */

  #if CPU_PERIPHERALS_INIT
  Peripherals Init();
```

Editor moves to this line

External Use | 106

Since the Step “into” was selected on the base line of PE_low_level_init(); the editor moves to the first line of the call. Now we will do a single step, which will take us to the next function call of this routine.

## Application Test: Single Step – Step Return


Debug - project_1/Sources/main.c - Kinetis Design Studio
File Edit Source Refactor Navigate Search Project Processor Expert Run Window Help
Step Over (F6)
Quick
Debug
Variables
Breakpoints
Registers
Mod
Events.c
main.c
my_main.c
0x347a

```
project_1 Debug [GDB OpenOCD Debugging]
  project_1.elf
    Thread [1] (Suspended : Breakpoint)
      main() at main.c:48 0x5e4
  openocd
  arm-none-eabi-gdb
```

ain] [type: Temporary]

Click “Step Over”

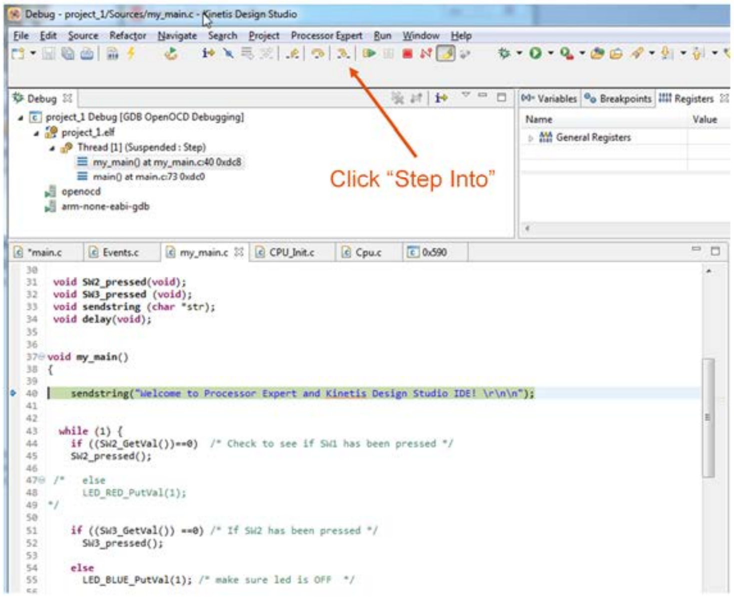
```
{
  /* Write your local variable definition here */

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

  /* Write your code here */
 my_main();
  /* For example: for(;;) { } */

  /*** Don't write any code pass this line, or it will be deleted during code generation. ***/
  /*** RTOS startup code. Macro PEX_RTOS_START is defined by the RTOS component. DON'T MODIFY THIS CODE!!! ***/
  #ifdef PEX_RTOS_START
    PEX_RTOS_START();                  /* Startup of the selected RTOS. Macro is defined by the RTOS component. */
  #endif
  /*** End of RTOS startup code.  ***/
  /*** Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! ***/
  for(;;){}
  /*** Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! ***/
} /*** End of main routine. DO NOT MODIFY THIS TEXT!!! ***/
```

Editor moves to next  line.

freescale™ External Use | 107

This will stop over the “PE_low_level_init and stop at my_main

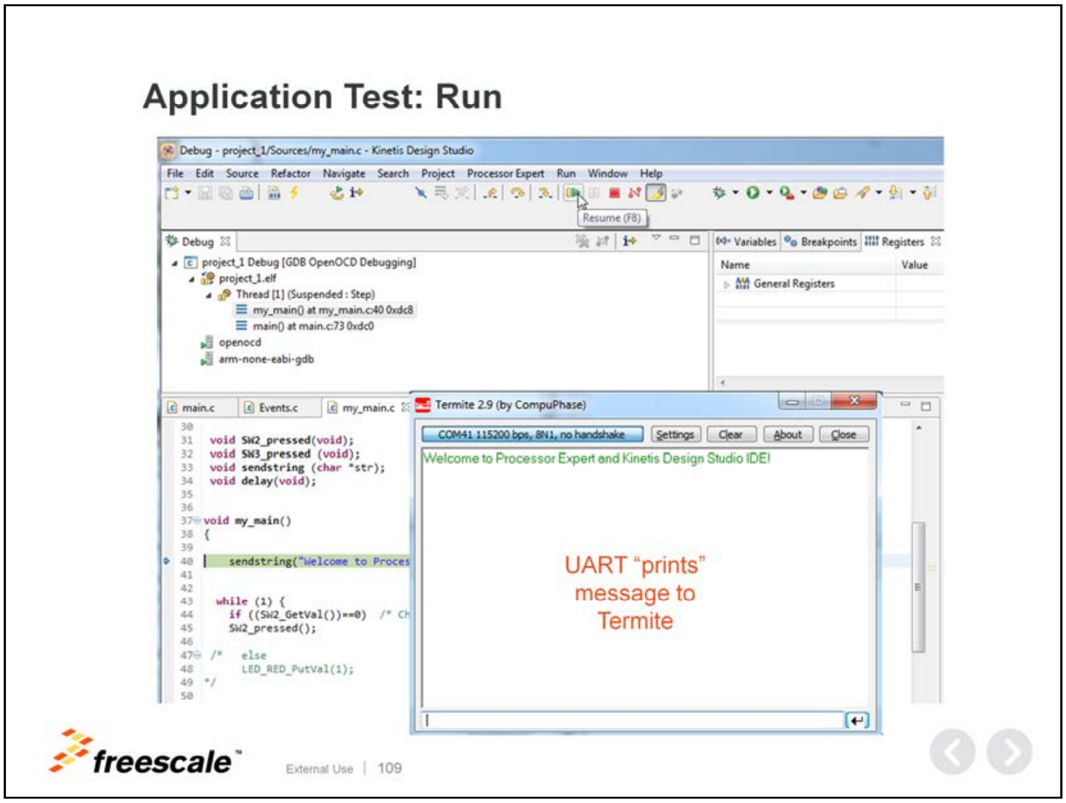
## Application Test: Single Step – Step Into



```
project_1 Debug [GDB OpenOCD Debugging]
  project_1.elf
    Thread [1] (Suspended : Step)
      my_main() at my_main.c:40 0xdc8
      main() at main.c:73 0xdc0
  openocd
  arm-none-eabi-gdb
```

```

 void SW2_pressed(void);
 void SW3_pressed (void);
 void sendstring (char *str);
 void delay(void);


void my_main()
{

   sendstring("Welcome to Processor Expert and Kinetis Design Studio IDE! \r\n\n");


  while (1) {
    if ((SW2_GetVal())==0)  /* Check to see if SW1 has been pressed */
    SW2_pressed();

 /*   else
      LED_RED_PutVal(1);
*/

    if ((SW3_GetVal()) ==0) /* If SW2 has been pressed */
      SW3_pressed();

    else
      LED_BLUE_PutVal(1); /* make sure led is OFF  */
```

Click “Step Into”



## Application Test: Run

Debug - project_1/Sources/my_main.c - Kinetis Design Studio
File Edit Source Refactor Navigate Search Project Processor Expert Run Window Help
Resume (F8)
Debug
Variables
Breakpoints
Registers
Name
Value
General Registers
main.c
Events.c
my_main.c
Termite 2.9 (by CompuPhase)
COM41 115200 bps, 8N1, no handshake
Settings
Clear
About
Close

```
project_1 Debug [GDB OpenOCD Debugging]
  project_1.elf
    Thread [1] (Suspended : Step)
      my_main() at my_main.c:40 0xdc8
      main() at main.c:73 0xdc0
  openocd
  arm-none-eabi-gdb
```

```
void SW2_pressed(void);
void SW3_pressed (void);
void sendstring (char *str);
void delay(void);


void my_main()
{

  sendstring("Welcome to Proces


 while (1) {
   if ((SW2_GetVal())==0)  /* Ch
   SW2_pressed();

/*   else
     LED_RED_PutVal(1);
*/
```

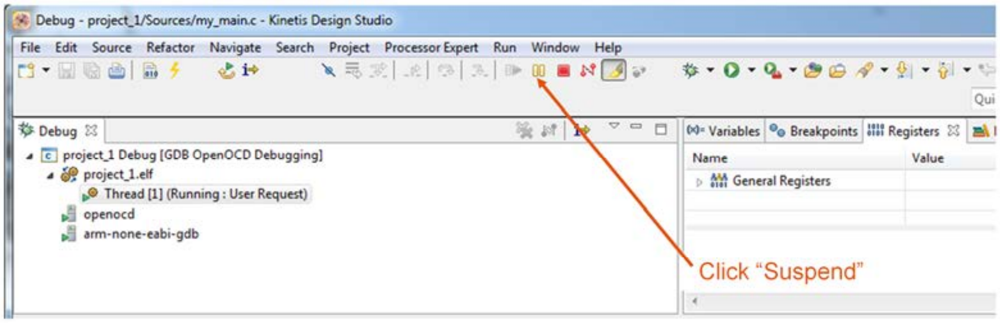
```
Welcome to Processor Expert and Kinetis Design Studio IDE!
```

UART "prints" message to Termite

External Use | 109

Notice the sendstring printed on the terminal and the green LED should be blinking.

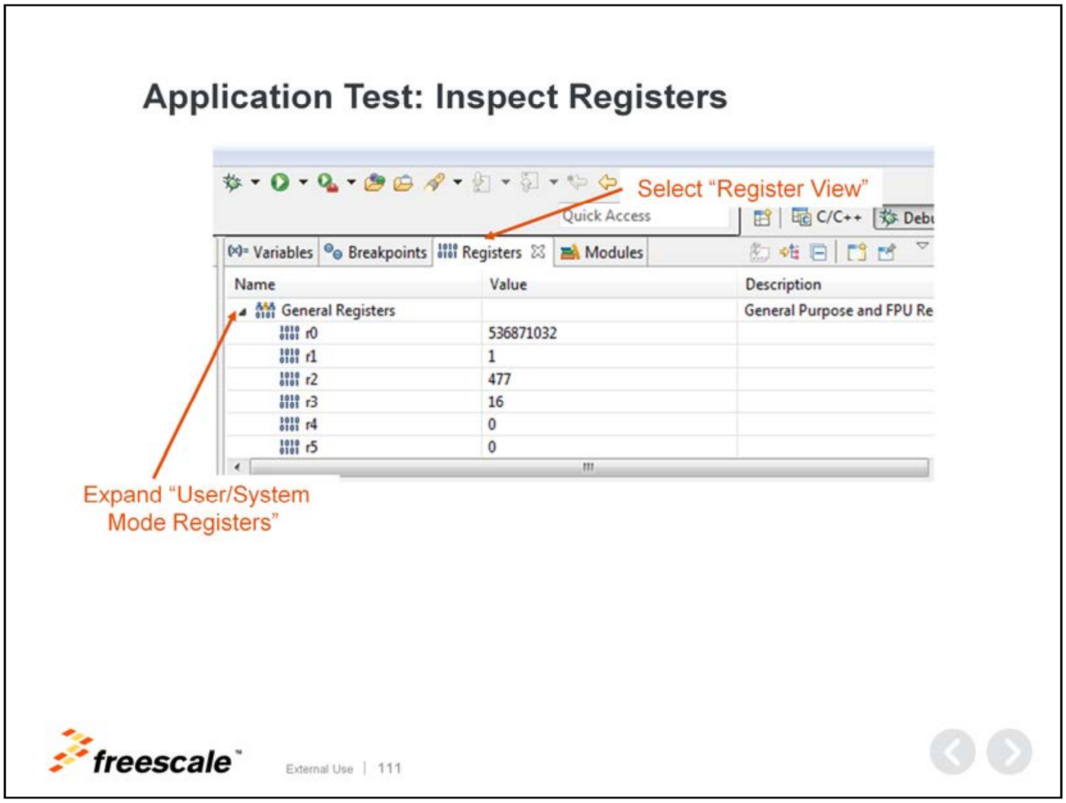
## Application Test: Inspect Registers


Debug - project_1/Sources/my_main.c - Kinetis Design Studio
File Edit Source Refactor Navigate Search Project Processor Expert Run Window Help
Debug
Variables
Breakpoints
Registers
Name
Value
General Registers

project_1 Debug [GDB OpenOCD Debugging]
project_1.elf
Thread [1] (Running : User Request)
openocd
arm-none-eabi-gdb

Click “Suspend”

With the program running, now click the “Suspend Icon”



## Application Test: Inspect Registers

Select "Register View"

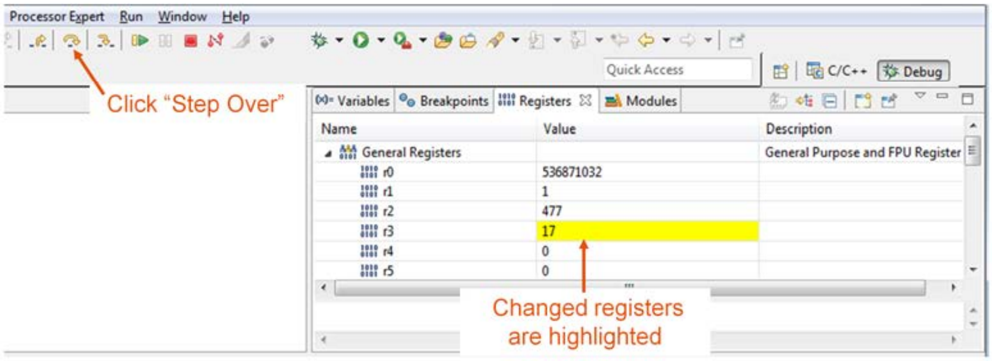
| Name | Value | Description |
|---|---|---|
| General Registers | | General Purpose and FPU Re |
| r0 | 536871032 | |
| r1 | 1 | |
| r2 | 477 | |
| r3 | 16 | |
| r4 | 0 | |
| r5 | 0 | |

Expand "User/System Mode Registers"

External Use | 111

Expand the Registers View and expand the User/System Mode Registers

## Application Test: Inspect Registers



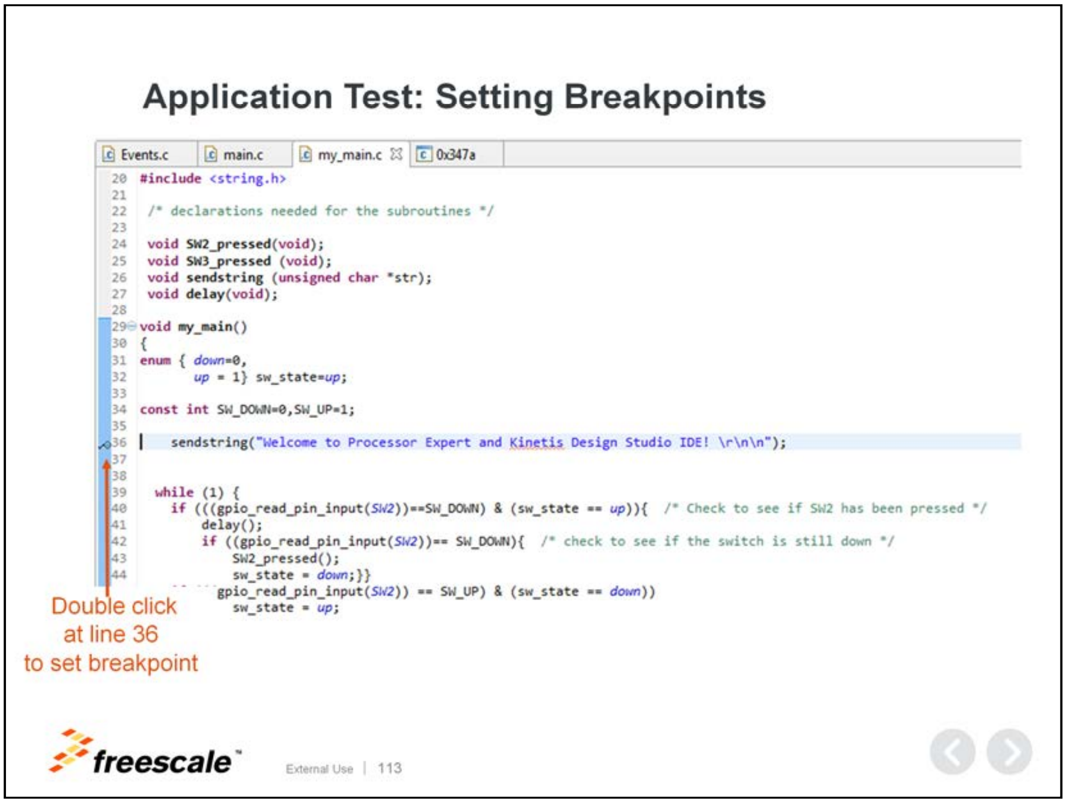
Click "Step Over"

| Name | Value | Description |
|---|---|---|
| General Registers | | General Purpose and FPU Register |
| r0 | 536871032 | |
| r1 | 1 | |
| r2 | 477 | |
| r3 | 17 | |
| r4 | 0 | |
| r5 | 0 | |

Changed registers are highlighted

With the register window open, single step over, multiple times and see the register value change as the code steps through the delay function.

Now click the suspend button.



## Application Test: Setting Breakpoints

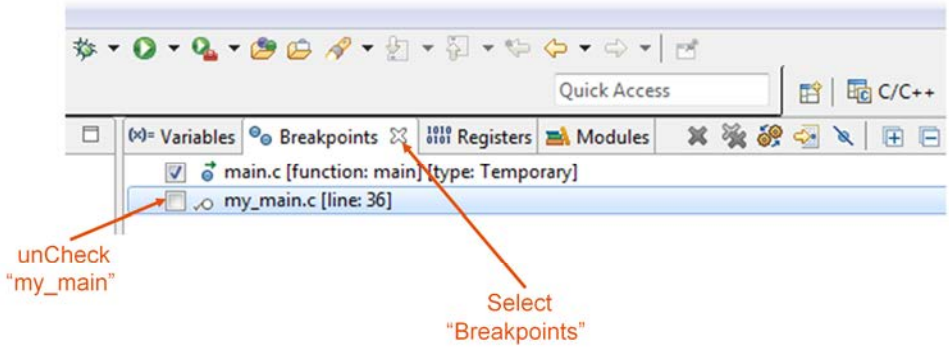
```
20 #include <string.h>
21
22  /* declarations needed for the subroutines */
23
24  void SW2_pressed(void);
25  void SW3_pressed (void);
26  void sendstring (unsigned char *str);
27  void delay(void);
28
29 void my_main()
30 {
31 enum { down=0,
32        up = 1} sw_state=up;
33
34 const int SW_DOWN=0,SW_UP=1;
35
36     sendstring("Welcome to Processor Expert and Kinetis Design Studio IDE! \r\n\n");
37
38
39   while (1) {
40     if (((gpio_read_pin_input(SW2))==SW_DOWN) & (sw_state == up)){  /* Check to see if SW2 has been pressed */
41         delay();
42         if ((gpio_read_pin_input(SW2))== SW_DOWN){  /* check to see if the switch is still down */
43             SW2_pressed();
44             sw_state = down;}}
              gpio_read_pin_input(SW2)) == SW_UP) & (sw_state == down))
             sw_state = up;
```

Double click
at line 36
to set breakpoint

External Use | 113

Set breakpoints by double clicking in the margin next to the line you wish to break point. In this case we want to break point at line 36.

## Application Test: Setting Breakpoints



☑ main.c [function: main] [type: Temporary]

□ my_main.c [line: 36]

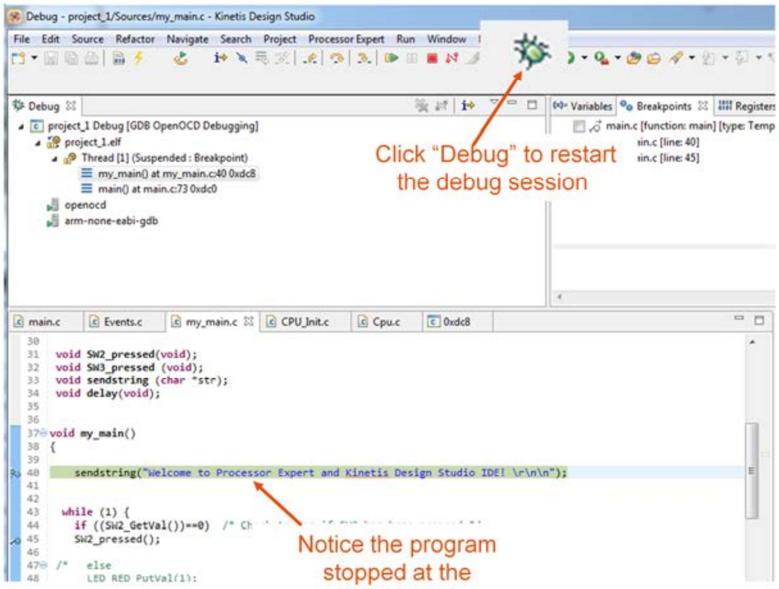
unCheck "my_main"

Select "Breakpoints"

Lets turn off a breakpoint at “my_main” and continue to run the program. Press the buttons and see the pretty lights go on an off.

Did we meet our design goals?

## Application Test: Setting Breakpoints



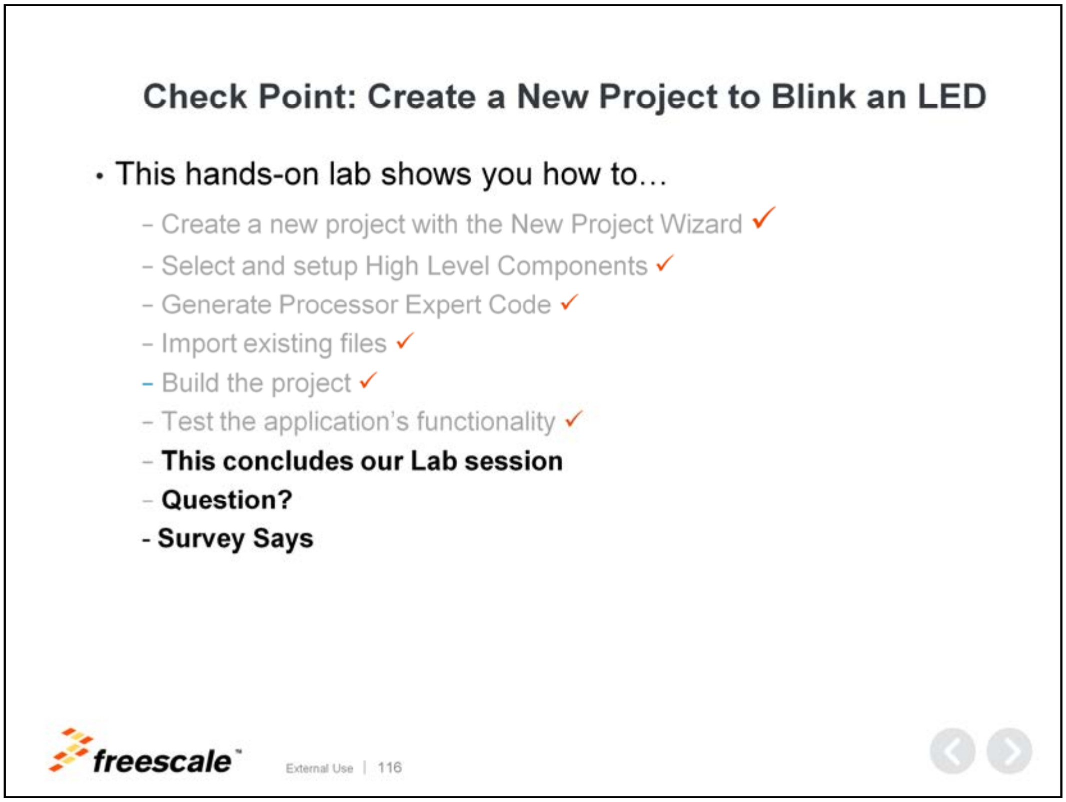
```
project_1 Debug [GDB OpenOCD Debugging]
  project_1.elf
    Thread [1] (Suspended : Breakpoint)
      my_main() at my_main.c:40 0xdc8
      main() at main.c:73 0xdc0
  openocd
  arm-none-eabi-gdb
```

```
main.c [function: main] [type: Temp
ain.c [line: 40]
ain.c [line: 45]
```

```
void SW2_pressed(void);
void SW3_pressed (void);
void sendstring (char *str);
void delay(void);


void my_main()
{

    sendstring("Welcome to Processor Expert and Kinetis Design Studio IDE! \r\n\n");


  while (1) {
    if ((SW2_GetVal())==0)  /* Ch
    SW2_pressed();

/*    else
      LED_RED_PutVal(1);
```

Click “Debug” to restart the debug session

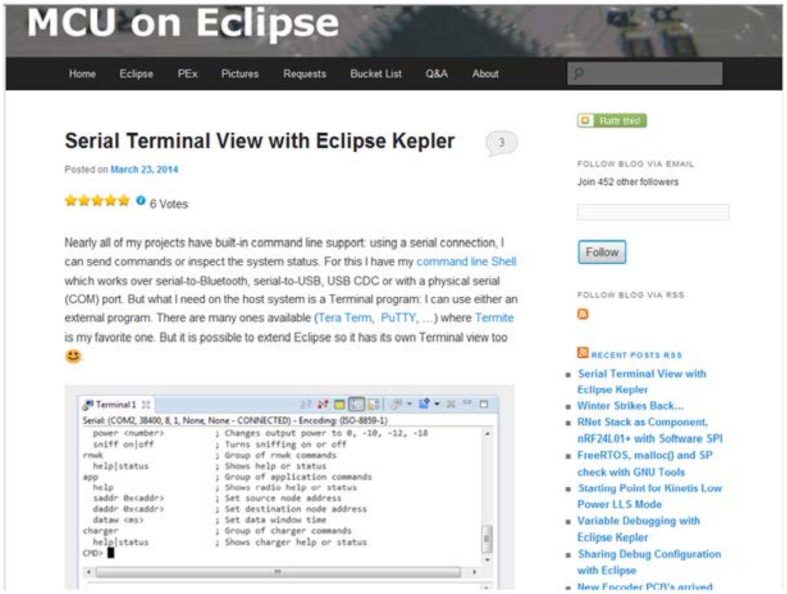
Notice the program stopped at the Breakpoint



## Check Point: Create a New Project to Blink an LED

- This hands-on lab shows you how to…
  - Create a new project with the New Project Wizard ✓
  - Select and setup High Level Components ✓
  - Generate Processor Expert Code ✓
  - Import existing files ✓
  - Build the project ✓
  - Test the application's functionality ✓
  - **This concludes our Lab session**
  - **Question?**
  - **Survey Says**

## Useful References: www.mcuoneclipse.com



### Serial Terminal View with Eclipse Kepler

3

Posted on March 23, 2014

★★★★★ 6 Votes

Nearly all of my projects have built-in command line support: using a serial connection, I can send commands or inspect the system status. For this I have my command line Shell which works over serial-to-Bluetooth, serial-to-USB, USB CDC or with a physical serial (COM) port. But what I need on the host system is a Terminal program: I can use either an external program. There are many ones available (Tera Term, PuTTY, …) where Termite is my favorite one. But it is possible to extend Eclipse so it has its own Terminal view too 😊.

Terminal 1

Serial: (COM2, 38400, 8, 1, None, None - CONNECTED) - Encoding: (ISO-8859-1)

```
  power <number>          ; Changes output power to 0, -10, -12, -18
  sniff on|off            ; Turns sniffing on or off
rnwk                      ; Group of rnwk commands
  help|status             ; Shows help or status
app                       ; Group of application commands
  help                    ; Shows radio help or status
  saddr 0x<addr>          ; Set source node address
  daddr 0x<addr>          ; Set destination node address
  dataw <ms>              ; Set data window time
charger                   ; Group of charger commands
  help|status             ; Shows charger help or status
CMD> █
```

FOLLOW BLOG VIA EMAIL

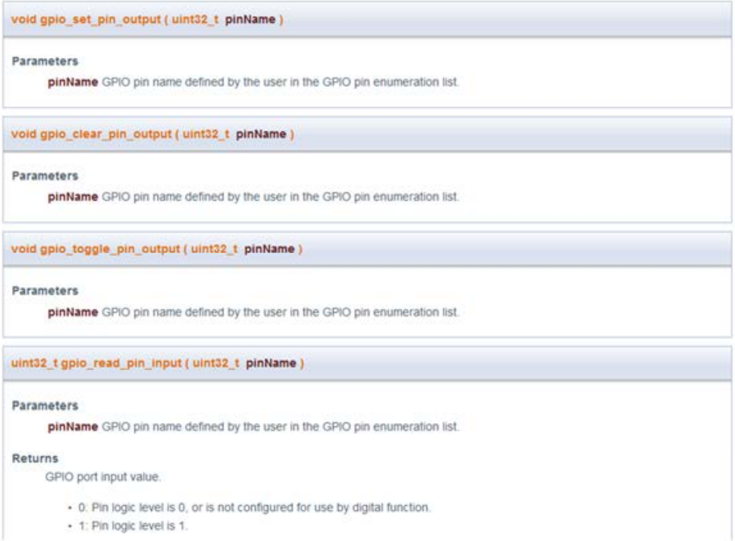
Join 452 other followers

FOLLOW BLOG VIA RSS

RECENT POSTS RSS

- Serial Terminal View with Eclipse Kepler
- Winter Strikes Back…
- RNet Stack as Component, nRF24L01+ with Software SPI
- FreeRTOS, malloc() and SP check with GNU Tools
- Starting Point for Kinetis Low Power LLS Mode
- Variable Debugging with Eclipse Kepler
- Sharing Debug Configuration with Eclipse
- New Encoder PCB's arrived

# SDK - GPIO



void gpio_set_pin_output ( uint32_t pinName )

Parameters

 **pinName** GPIO pin name defined by the user in the GPIO pin enumeration list.

void gpio_clear_pin_output ( uint32_t pinName )

Parameters

 **pinName** GPIO pin name defined by the user in the GPIO pin enumeration list.

void gpio_toggle_pin_output ( uint32_t pinName )

Parameters

 **pinName** GPIO pin name defined by the user in the GPIO pin enumeration list.

uint32_t gpio_read_pin_input ( uint32_t pinName )

Parameters

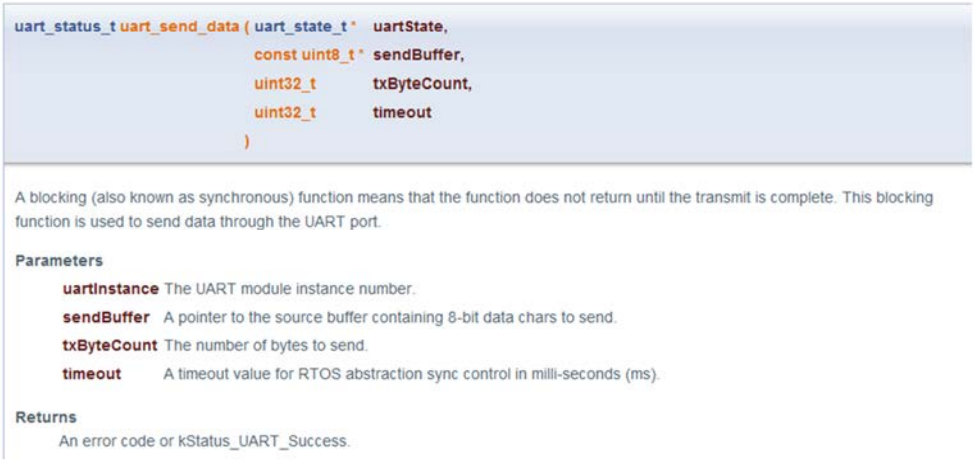
 **pinName** GPIO pin name defined by the user in the GPIO pin enumeration list.

Returns

GPIO port input value.

- 0: Pin logic level is 0, or is not configured for use by digital function.
- 1: Pin logic level is 1.

## SDK - UART



```
uart_status_t uart_send_data ( uart_state_t *     uartState,
                               const uint8_t *    sendBuffer,
                               uint32_t           txByteCount,
                               uint32_t           timeout
                             )
```

A blocking (also known as synchronous) function means that the function does not return until the transmit is complete. This blocking function is used to send data through the UART port.

**Parameters**

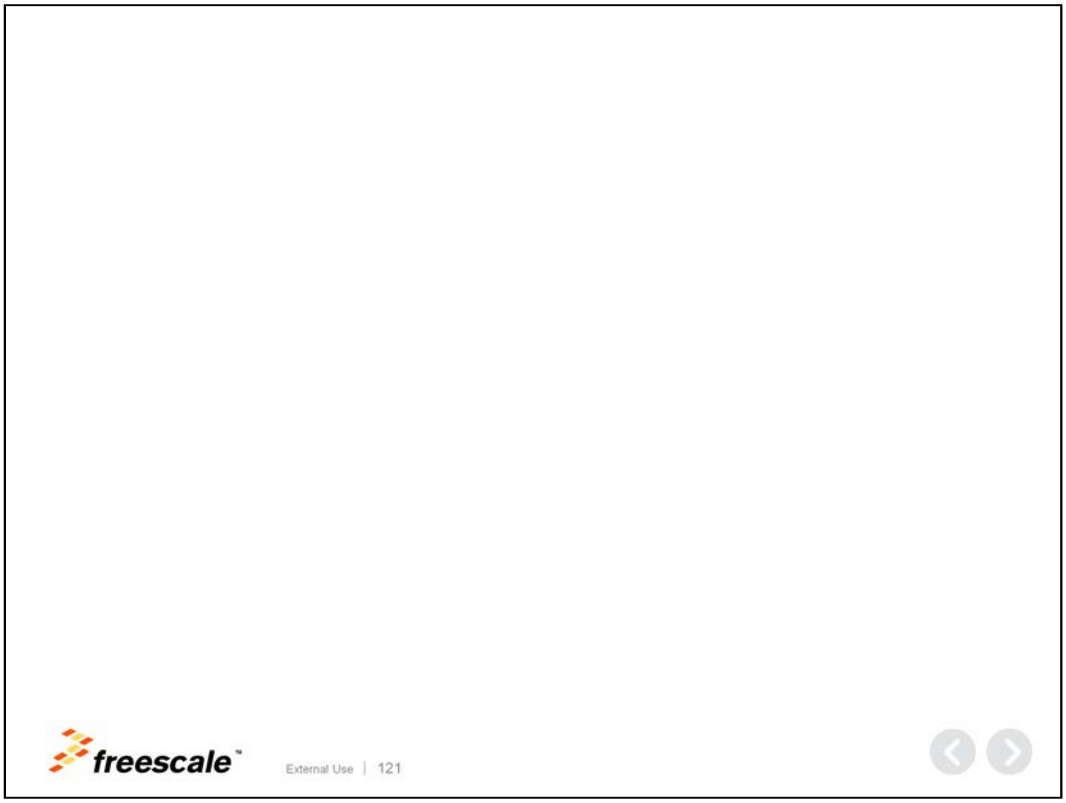
| | |
|---|---|
| **uartInstance** | The UART module instance number. |
| **sendBuffer** | A pointer to the source buffer containing 8-bit data chars to send. |
| **txByteCount** | The number of bytes to send. |
| **timeout** | A timeout value for RTOS abstraction sync control in milli-seconds (ms). |

**Returns**

An error code or kStatus_UART_Success.

freescale™

www.Freescale.com



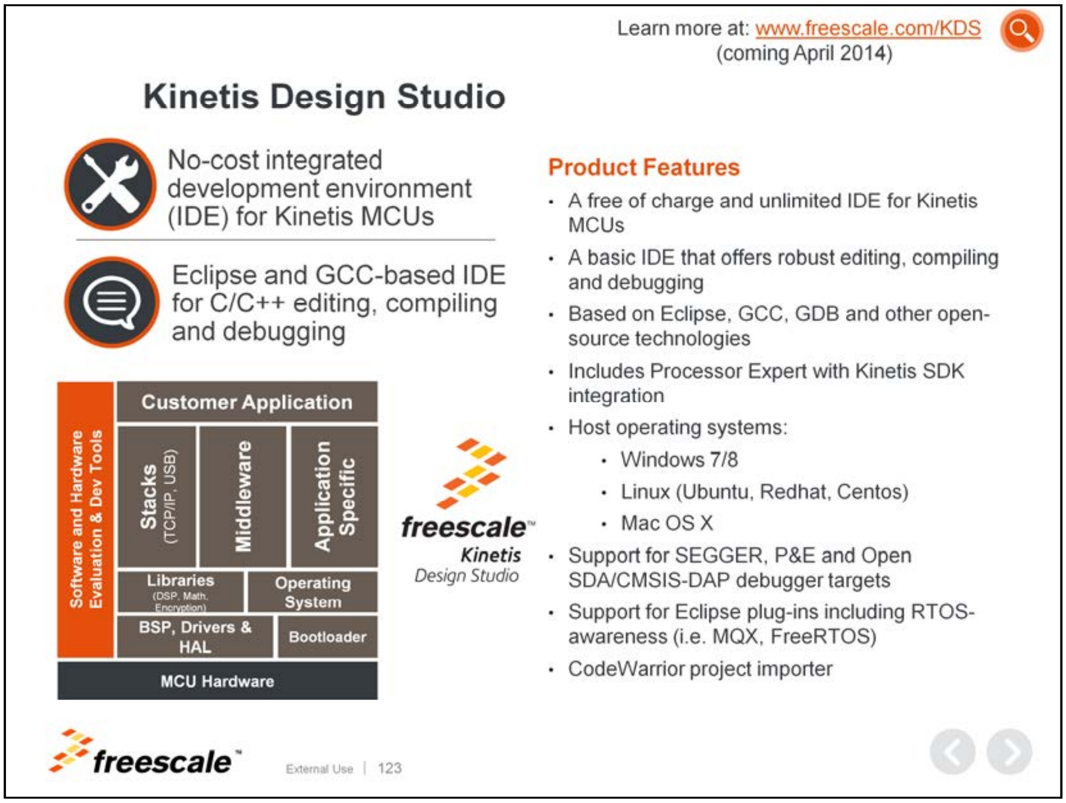
# Kinetis Design Studio

www.freescale.com

Microcontrollers Software and Services

Feb.24.2013

External Use



Learn more at: www.freescale.com/KDS
(coming April 2014)

## Kinetis Design Studio

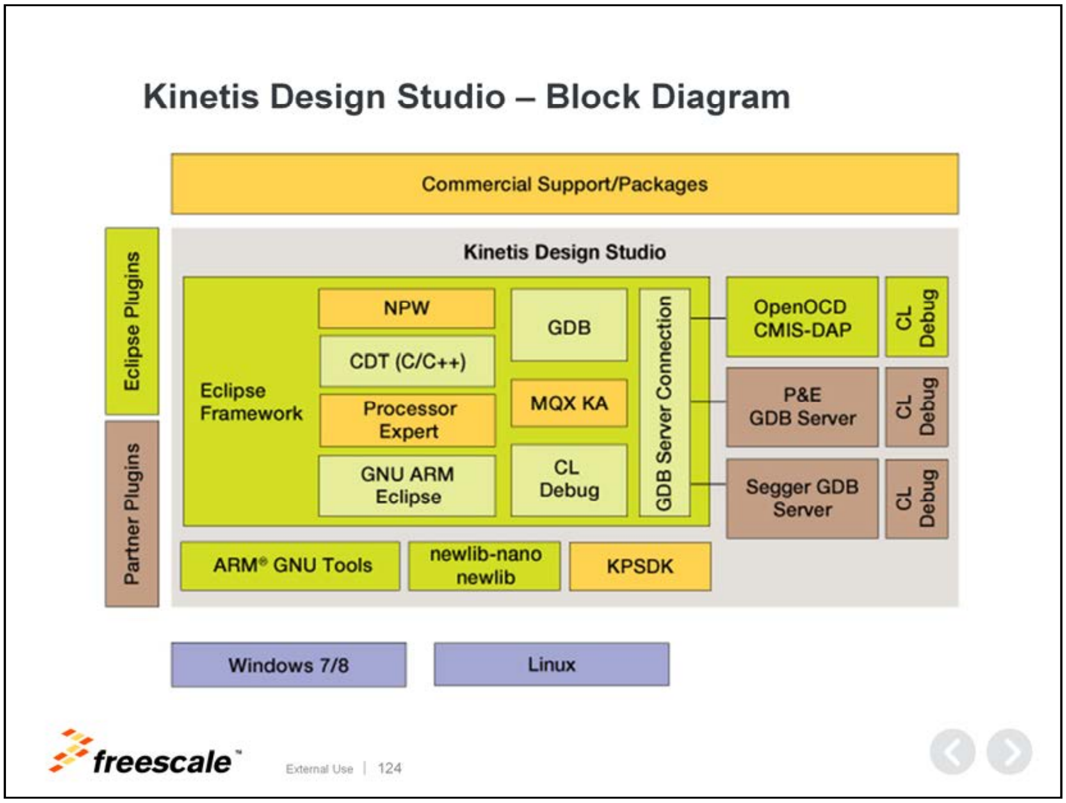
No-cost integrated development environment (IDE) for Kinetis MCUs

Eclipse and GCC-based IDE for C/C++ editing, compiling and debugging

### Product Features

- A free of charge and unlimited IDE for Kinetis MCUs
- A basic IDE that offers robust editing, compiling and debugging
- Based on Eclipse, GCC, GDB and other open-source technologies
- Includes Processor Expert with Kinetis SDK integration
- Host operating systems:
  - Windows 7/8
  - Linux (Ubuntu, Redhat, Centos)
  - Mac OS X
- Support for SEGGER, P&E and Open SDA/CMSIS-DAP debugger targets
- Support for Eclipse plug-ins including RTOS-awareness (i.e. MQX, FreeRTOS)
- CodeWarrior project importer

External Use | 123



## Kinetis Design Studio – Block Diagram

Commercial Support/Packages

Eclipse Plugins

Partner Plugins

Kinetis Design Studio

Eclipse Framework

NPW

CDT (C/C++)

Processor Expert

GNU ARM Eclipse

GDB

MQX KA

CL Debug

GDB Server Connection

OpenOCD CMIS-DAP

CL Debug

P&E GDB Server

CL Debug

Segger GDB Server

CL Debug

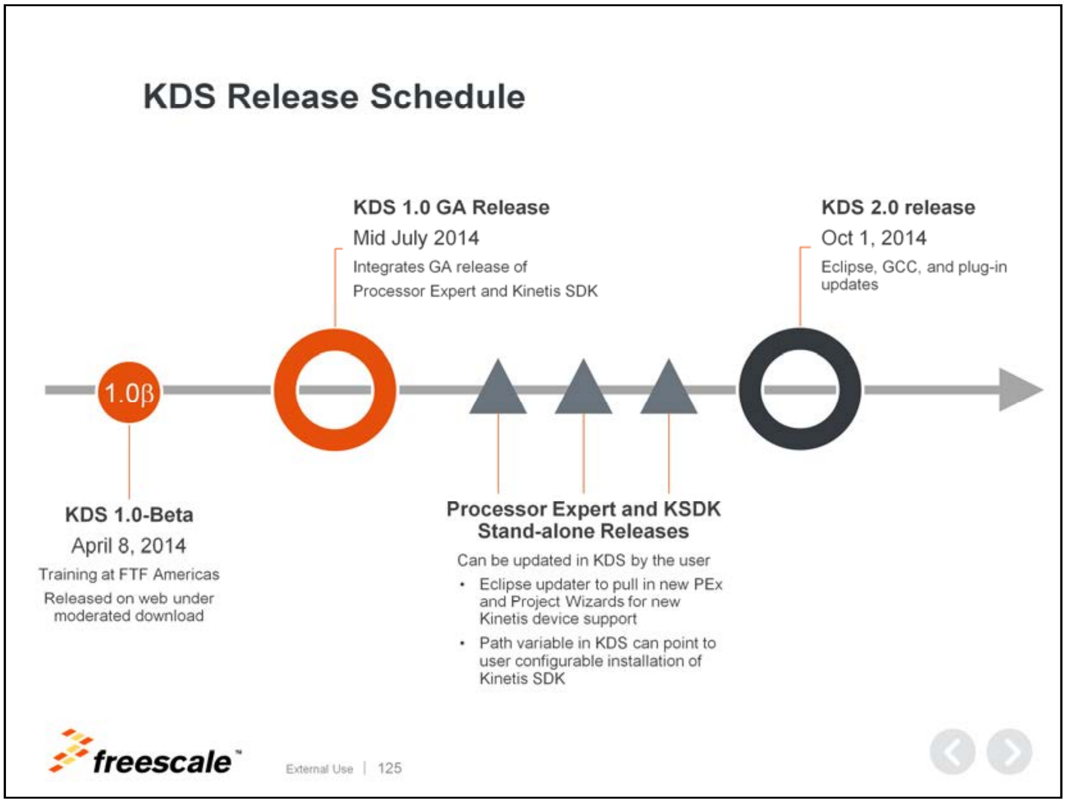
ARM® GNU Tools

newlib-nano newlib

KPSDK

Windows 7/8

Linux



## KDS Release Schedule

**KDS 1.0-Beta**

April 8, 2014

Training at FTF Americas

Released on web under moderated download

**KDS 1.0 GA Release**

Mid July 2014

Integrates GA release of Processor Expert and Kinetis SDK

**Processor Expert and KSDK Stand-alone Releases**

Can be updated in KDS by the user

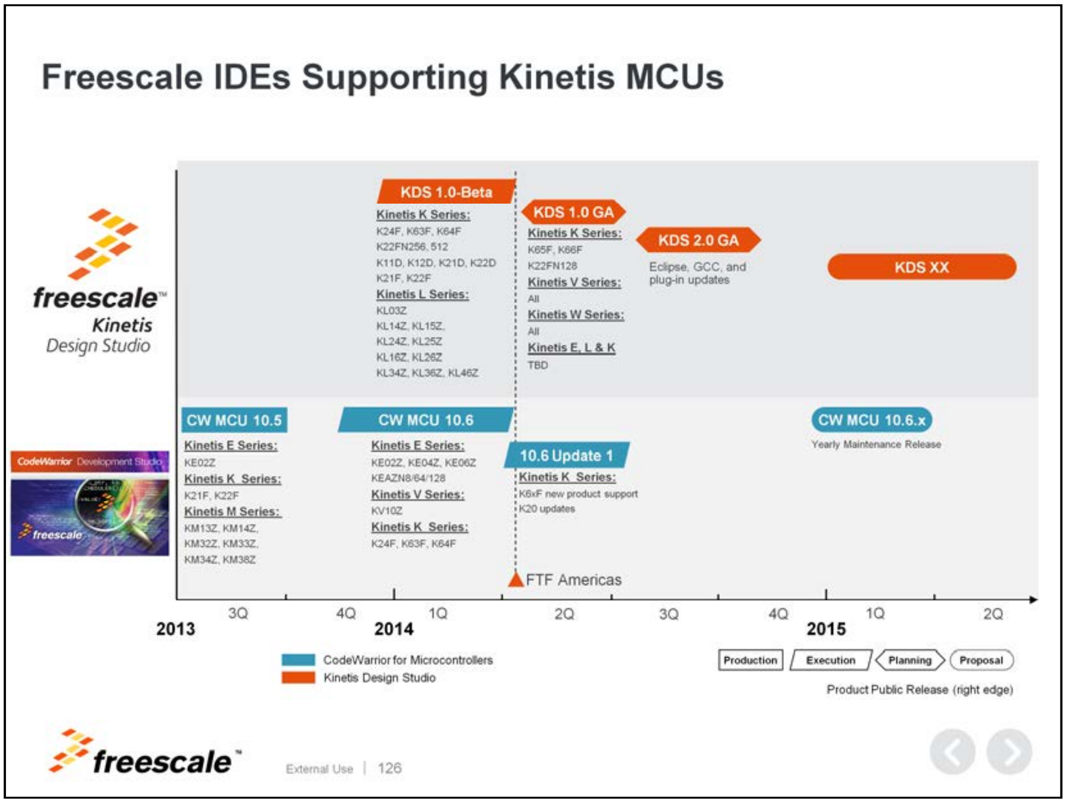
- Eclipse updater to pull in new PEx and Project Wizards for new Kinetis device support
- Path variable in KDS can point to user configurable installation of Kinetis SDK

**KDS 2.0 release**

Oct 1, 2014

Eclipse, GCC, and plug-in updates

External Use | 125



# Freescale IDEs Supporting Kinetis MCUs

Kinetis Design Studio

KDS 1.0-Beta

Kinetis K Series:
K24F, K63F, K64F
K22FN256, 512
K11D, K12D, K21D, K22D
K21F, K22F

Kinetis L Series:
KL03Z
KL14Z, KL15Z,
KL24Z, KL25Z
KL16Z, KL26Z
KL34Z, KL36Z, KL46Z

KDS 1.0 GA

Kinetis K Series:
K65F, K66F
K22FN128

Kinetis V Series:
All

Kinetis W Series:
All

Kinetis E, L & K
TBD

KDS 2.0 GA

Eclipse, GCC, and plug-in updates

KDS XX

CodeWarrior Development Studio

CW MCU 10.5

Kinetis E Series:
KE02Z

Kinetis K Series:
K21F, K22F

Kinetis M Series:
KM13Z, KM14Z,
KM32Z, KM33Z,
KM34Z, KM38Z

CW MCU 10.6

Kinetis E Series:
KE02Z, KE04Z, KE06Z
KEAZN8/64/128

Kinetis V Series:
KV10Z

Kinetis K Series:
K24F, K63F, K64F

10.6 Update 1

Kinetis K Series:
K6xF new product support
K20 updates

CW MCU 10.6.x

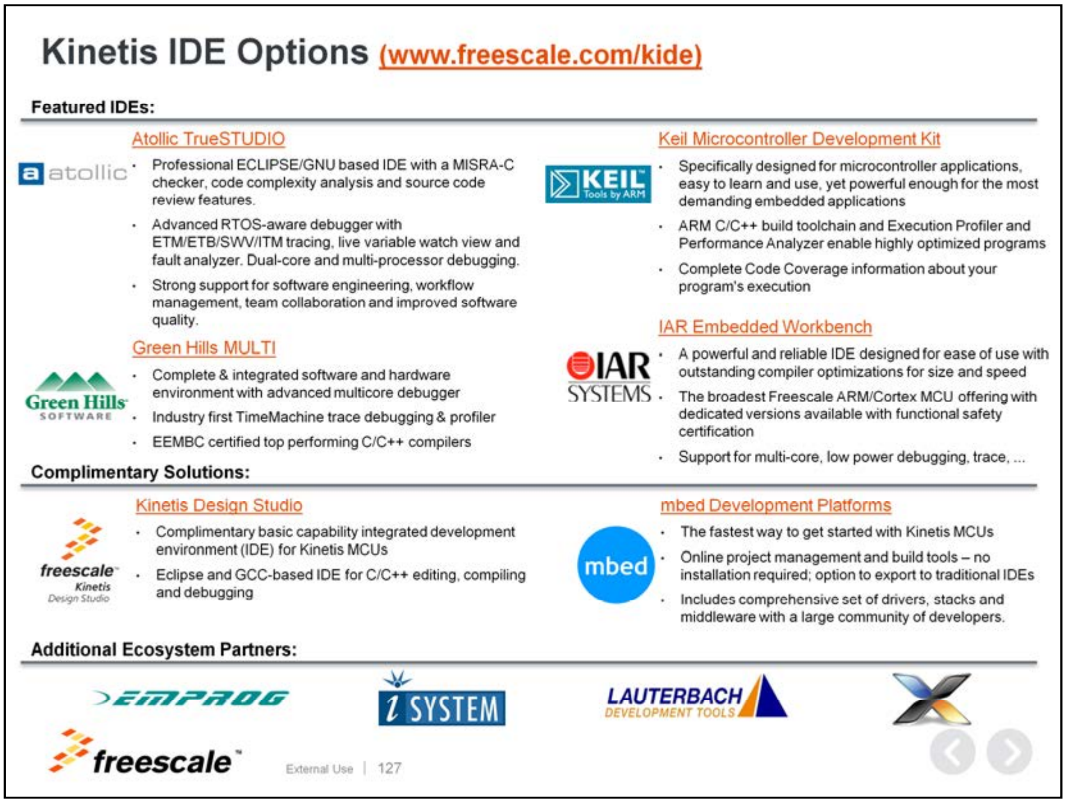
Yearly Maintenance Release

FTF Americas

🟦 CodeWarrior for Microcontrollers
🟥 Kinetis Design Studio

Production | Execution | Planning | Proposal

Product Public Release (right edge)



## Kinetis IDE Options (www.freescale.com/kide)

**Featured IDEs:**

Atollic TrueSTUDIO

- Professional ECLIPSE/GNU based IDE with a MISRA-C checker, code complexity analysis and source code review features.
- Advanced RTOS-aware debugger with ETM/ETB/SWV/ITM tracing, live variable watch view and fault analyzer. Dual-core and multi-processor debugging.
- Strong support for software engineering, workflow management, team collaboration and improved software quality.

Green Hills MULTI

- Complete & integrated software and hardware environment with advanced multicore debugger
- Industry first TimeMachine trace debugging & profiler
- EEMBC certified top performing C/C++ compilers

Keil Microcontroller Development Kit

- Specifically designed for microcontroller applications, easy to learn and use, yet powerful enough for the most demanding embedded applications
- ARM C/C++ build toolchain and Execution Profiler and Performance Analyzer enable highly optimized programs
- Complete Code Coverage information about your program's execution

IAR Embedded Workbench

- A powerful and reliable IDE designed for ease of use with outstanding compiler optimizations for size and speed
- The broadest Freescale ARM/Cortex MCU offering with dedicated versions available with functional safety certification
- Support for multi-core, low power debugging, trace, ...

**Complimentary Solutions:**

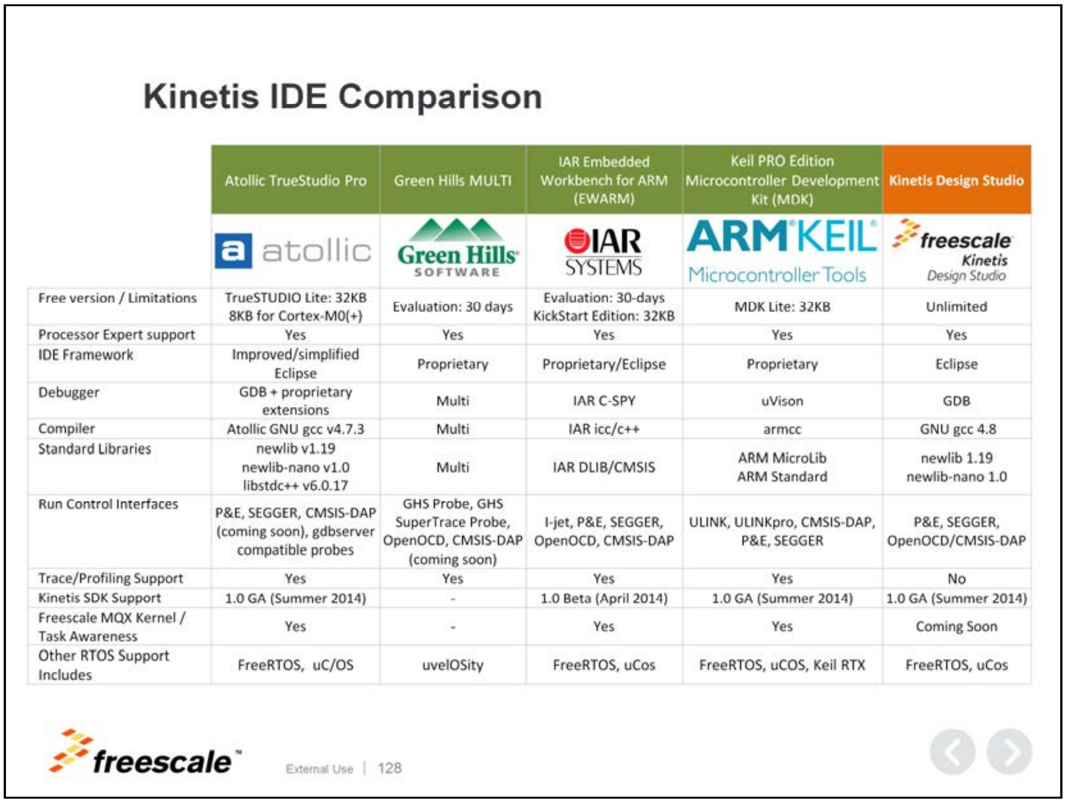
Kinetis Design Studio

- Complimentary basic capability integrated development environment (IDE) for Kinetis MCUs
- Eclipse and GCC-based IDE for C/C++ editing, compiling and debugging

mbed Development Platforms

- The fastest way to get started with Kinetis MCUs
- Online project management and build tools – no installation required; option to export to traditional IDEs
- Includes comprehensive set of drivers, stacks and middleware with a large community of developers.

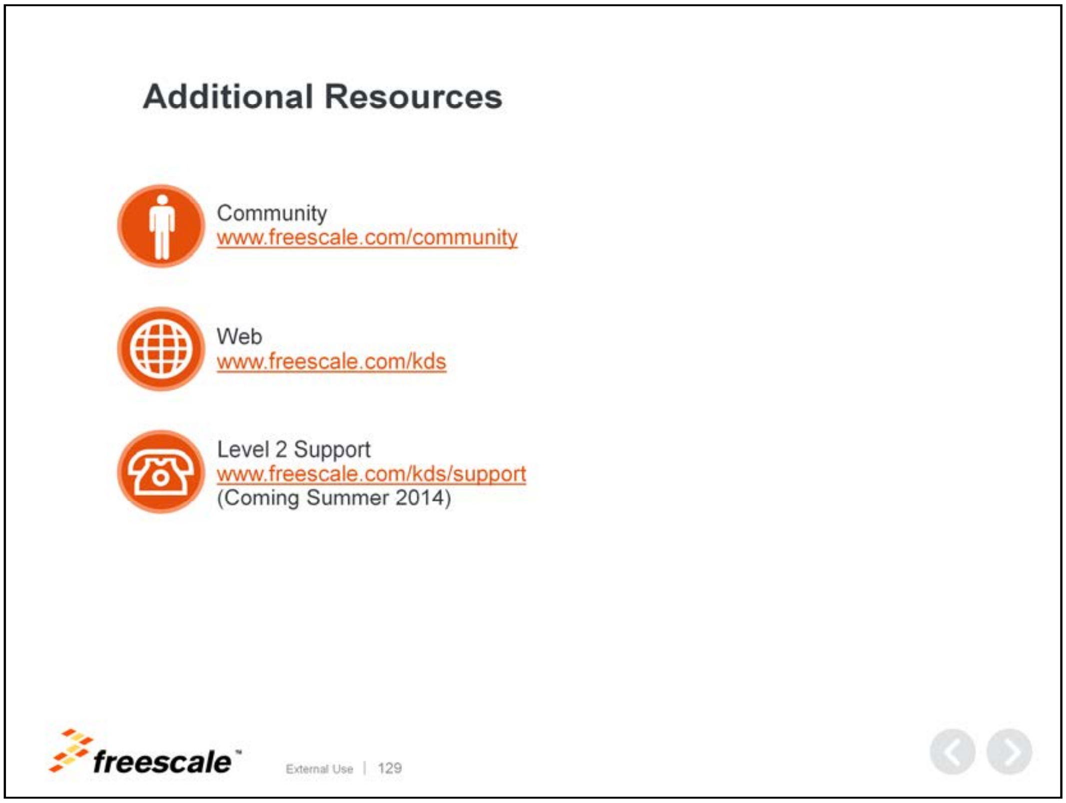
**Additional Ecosystem Partners:**



# Kinetis IDE Comparison

| | Atollic TrueStudio Pro | Green Hills MULTI | IAR Embedded Workbench for ARM (EWARM) | Keil PRO Edition Microcontroller Development Kit (MDK) | Kinetis Design Studio |
|---|---|---|---|---|---|
| | atollic | Green Hills SOFTWARE | IAR SYSTEMS | ARM KEIL Microcontroller Tools | freescale Kinetis Design Studio |
| Free version / Limitations | TrueSTUDIO Lite: 32KB<br>8KB for Cortex-M0(+) | Evaluation: 30 days | Evaluation: 30-days<br>KickStart Edition: 32KB | MDK Lite: 32KB | Unlimited |
| Processor Expert support | Yes | Yes | Yes | Yes | Yes |
| IDE Framework | Improved/simplified Eclipse | Proprietary | Proprietary/Eclipse | Proprietary | Eclipse |
| Debugger | GDB + proprietary extensions | Multi | IAR C-SPY | uVison | GDB |
| Compiler | Atollic GNU gcc v4.7.3 | Multi | IAR icc/c++ | armcc | GNU gcc 4.8 |
| Standard Libraries | newlib v1.19<br>newlib-nano v1.0<br>libstdc++ v6.0.17 | Multi | IAR DLIB/CMSIS | ARM MicroLib<br>ARM Standard | newlib 1.19<br>newlib-nano 1.0 |
| Run Control Interfaces | P&E, SEGGER, CMSIS-DAP (coming soon), gdbserver compatible probes | GHS Probe, GHS SuperTrace Probe, OpenOCD, CMSIS-DAP (coming soon) | I-jet, P&E, SEGGER, OpenOCD, CMSIS-DAP | ULINK, ULINKpro, CMSIS-DAP, P&E, SEGGER | P&E, SEGGER, OpenOCD/CMSIS-DAP |
| Trace/Profiling Support | Yes | Yes | Yes | Yes | No |
| Kinetis SDK Support | 1.0 GA (Summer 2014) | - | 1.0 Beta (April 2014) | 1.0 GA (Summer 2014) | 1.0 GA (Summer 2014) |
| Freescale MQX Kernel / Task Awareness | Yes | - | Yes | Yes | Coming Soon |
| Other RTOS Support Includes | FreeRTOS, uC/OS | uvelOSity | FreeRTOS, uCos | FreeRTOS, uCOS, Keil RTX | FreeRTOS, uCos |

freescale™ External Use | 128



## Additional Resources

☎ Community
www.freescale.com/community

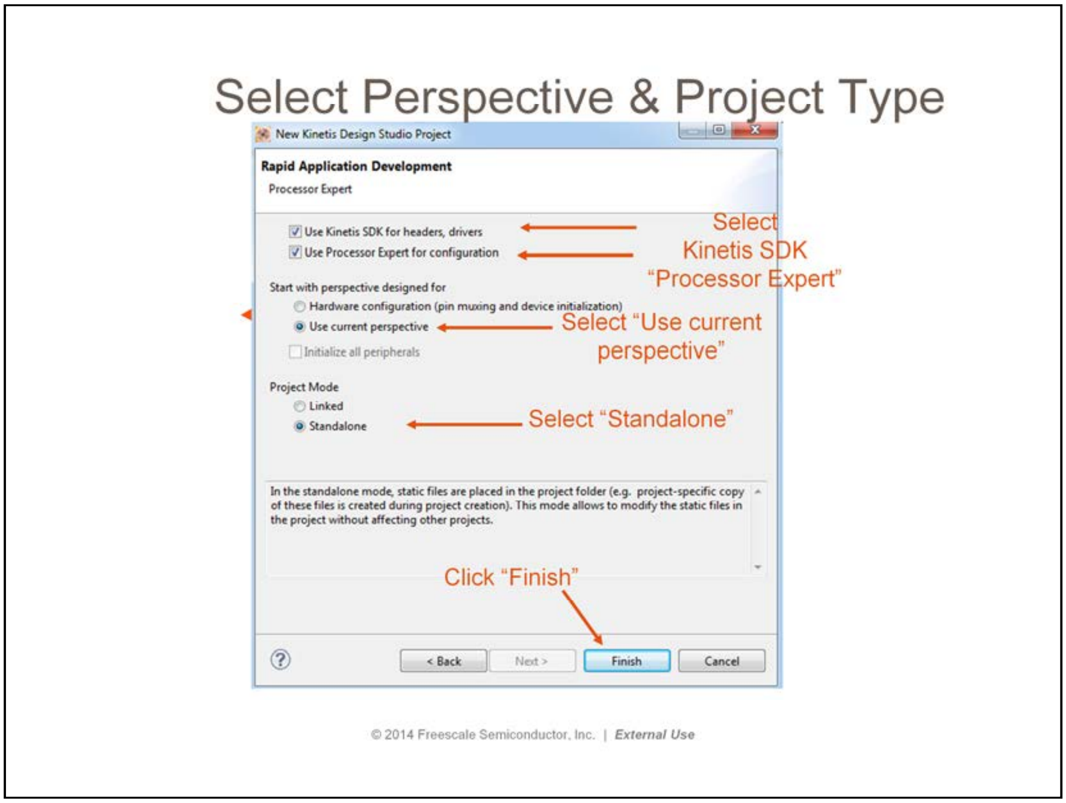
Web
www.freescale.com/kds

☎ Level 2 Support
www.freescale.com/kds/support
(Coming Summer 2014)

freescale™ External Use | 129

freescale™

www.Freescale.com



## Select Perspective & Project Type

New Kinetis Design Studio Project

**Rapid Application Development**

Processor Expert

☑ Use Kinetis SDK for headers, drivers

☑ Use Processor Expert for configuration

Select Kinetis SDK "Processor Expert"

Start with perspective designed for

○ Hardware configuration (pin muxing and device initialization)

● Use current perspective

Select "Use current perspective"

□ Initialize all peripherals

Project Mode

○ Linked

● Standalone

Select "Standalone"

In the standalone mode, static files are placed in the project folder (e.g. project-specific copy of these files is created during project creation). This mode allows to modify the static files in the project without affecting other projects.

Click "Finish"

© 2014 Freescale Semiconductor, Inc. | *External Use*

Start with Perspective: Lets set this to the non-processor expert hardware configuration perspective. Select Kinetis SDK (this will be selectable for processors that are supported by the SDK), and select Standalone project mode. This allows for you to modify the files and settings to suite your needs without effecting the files stored on disk. If you use "Linked" mode, and you modify the contents of the code, then you are modifying the reference code.

If you plan on archiving the project or sharing the project, this assures the entire project is included, all the source files, etc.