

Hands-on Workshop: Using CodeWarrior IDE and Processor Expert Software Modeling Tool – Part 1

AMF-ENT-T0711

Mark Ruthenbeck
Ruth Rhoades

Product Manager



September 2013

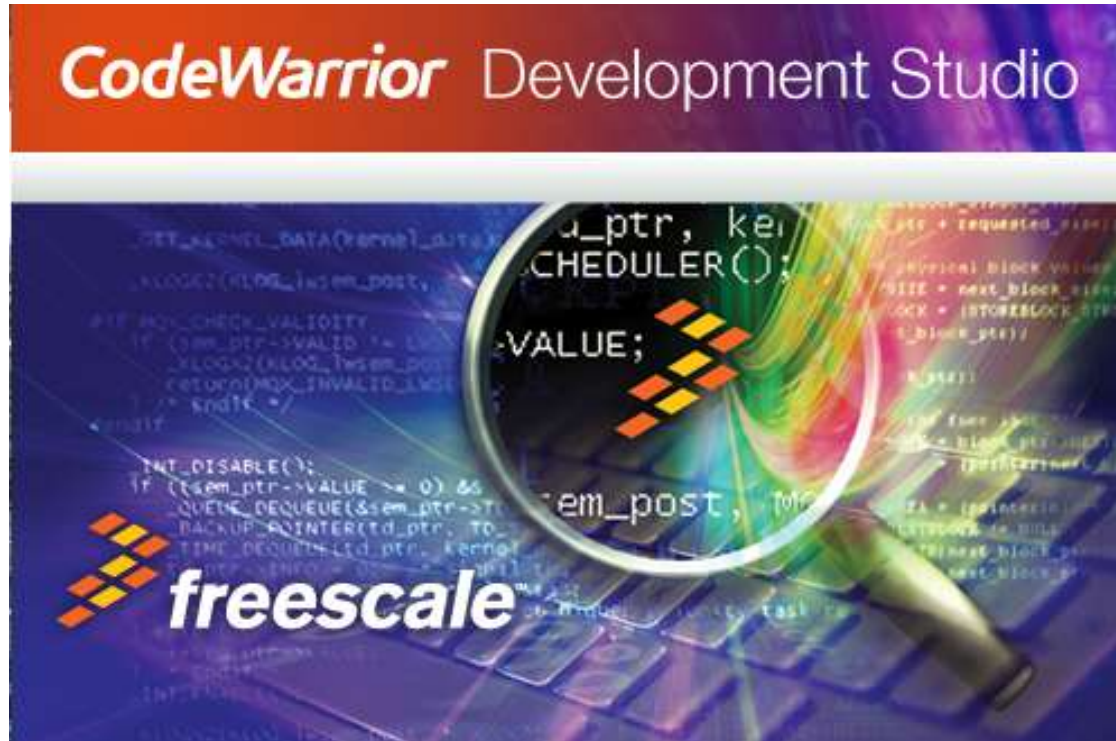
Freescale, the Freescale logo, AllWin, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Wire, the Energy Efficient Solutions logo, iM801, iM801GT, P50, PowerQAC, Processor Expert, QorIQ, QorIQ+, SafeAssure, the SafeAssure logo, StarCore, Symphony and VortiQa are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. AirStar, Beolite, BeeStack, ClearNet, Flexio, LayerScope, MagiK, M6C, Platform in a Package, QorIQ Converge, QorIQ Engine, ReadyPlay, SMARTMOS, Tower, TurboLink, Vybrid and Xtrinsic are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © 2013 Freescale Semiconductor, Inc.



- CodeWarrior Development Studio
 - Features, Suites, Pricing, Downloads, Upgrade Policy
- Basic Processor Expert Concepts & Terms
- Lab 1: FRDM-KL46Z Board Project
 - Create a new project with the New Project Wizard
 - Import existing components and files
 - Build the project
 - Test the application's functionality

CodeWarrior Development Studio

- CodeWarrior Development Studio for Microcontrollers v10.4 integrates the development tools for ColdFire, ColdFire+, DSC, Kinetis L Series, Kinetis K Series, Qorivva, PX Series, RS08, S08 and S12Z architectures into a single product based on the Eclipse open development platform.



CodeWarrior Differentiating Features

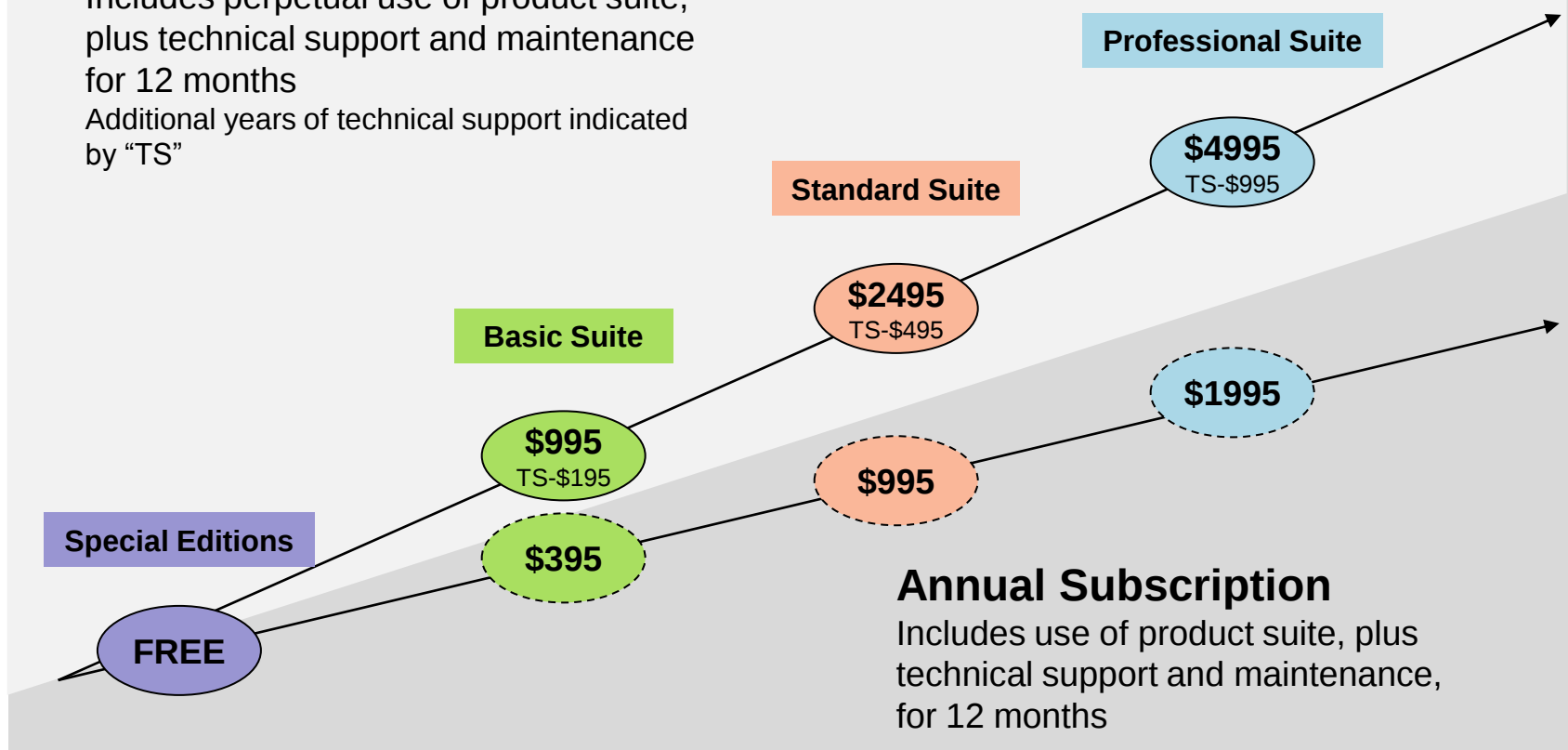
- **Special Edition** – Available at no charge and allows projects to be developed with code size limitations
- **New Project Wizard** – Allows a project to be created in as few as six clicks
- **LiveView** – Allows registers, memory and global variables to be monitored without stopping the processor
- **Processor Expert** – Creates tested, optimized initialization code and low-level drivers tuned to application needs and selected Freescale derivative
 - Built-in knowledge base immediately flags resource conflicts and incorrect settings, so errors are caught early in design cycle
 - Processor Expert for ColdFire+ and Kinetis works with and extends the capabilities of MQX
- **Trace and Profile** support for on-chip trace buffer – provides sophisticated emulator-like debug capability without additional trace capture hardware

CodeWarrior Pricing and Packaging Model

Perpetual

Includes perpetual use of product suite,
plus technical support and maintenance
for 12 months

Additional years of technical support indicated by “TS”



Feature Summary & Comparison

	Special Edition	Basic Edition	Standard Edition	Professional Edition
Key Features				
Unlimited Project Files	X	X	X	X
Unlimited Assembler	X	X	X	X
Processor Expert Software	X	X	X	X
Component Development Environment	Non-commercial	Non-commercial	Non-commercial	Commercial
Task Aware Debug				X
C++				X
Licensing: Annual or Permanent		X	X	X
1 Year Support		X	X	X
Code/Debug Size Limits				
RS08, S08	64K	128K	None	None
S12Z	64K	128K	None	None
Coldfire V1, Coldfire+	64K	128K	None	None
Coldfire V2-V4	128K	512K	None	None
DSC	64K	256K	None	None
Kinetis L	64K	128K	None	None
Kinetis K	128K	512K	None	None
Qorriova, PX	512K	1M	None	None
Pricing				
Pricing: Annual License/Support	Free	\$395	\$995	\$1995
Pricing: Permanent License	Free	\$995	\$2495	\$4995
Pricing: Annual Support Permanent License	Free	\$195	\$495	\$995

CodeWarrior Upgrade Policy

Annual Subscription License

- If you have an active annual subscription license, you can download CodeWarrior MCU v10.4 at no cost
 - If your annual subscription license has expired, you must purchase another license before downloading CodeWarrior MCU v10.4
 - Basic Suite: \$395
 - Standard Suite: \$995
 - Professional Suite: \$1,995
- Valid technical support agreement equals
NO cost to upgrade**

**Valid technical support agreement equals
NO cost to upgrade**

Perpetual License

- If you purchased a perpetual license within the last 12 months, you can download CodeWarrior MCU v10.4 at no cost
- If you have an active technical support agreement (after the first 12 months), you can download CodeWarrior MCU v10.4 at no cost
- If your technical support agreement has expired, you must purchase another technical support agreement before downloading CodeWarrior MCU v10.4
 - Basic Suite: \$195
 - Standard Suite: \$495
 - Professional Suite: \$995

CodeWarrior License Registration/Renewal

A new license is required for CW MCU v10.4. Licenses for previous versions will NOT work.

- **New purchase:** If you just purchased a CodeWarrior Suite, the process to activate and license your product is simple:
 - Log in to “My Freescale”
 - Select “ CodeWarrior Licensing”
 - Navigate to your product in “Products you are entitled to license”
 - Click the “Get License” button to retrieve your license
- **Existing customer:** If you already own a CodeWarrior suite and have obtained previous versions of CodeWarrior for Microcontrollers (e.g., 10.0, 10.1, 10.2 or 10.3), the process is a little different:
 - Log in to “My Freescale”
 - Select “ CodeWarrior Licensing”
 - Navigate to your product in “Products that you have licensed”
 - Click the “Version Renewal” button next to your existing CodeWarrior for Microcontrollers product to retrieve your license



FreeScale, the Freescale logo, AllWin, e3, CodeTEST, CodeWarrior, ColdFire, ColdFire+, e2Ware, the Energy Efficient Solution logo, iMx6, iMx67, iMx67L, PPG, PowerQICC, ProFusion Express, QorIQ, QorIQcore, SafeSense, the SafeSense logo, StarCore, Symphony and Vortex are trademarks of Freescale Semiconductor, Inc. in the U.S. and/or in other countries. All other trademarks, including the Freescale Semiconductor logo, are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © 2013 Freescale Semiconductor, Inc.

Embedded Components

- Embedded Components encapsulate the functionality of basic elements of embedded systems.
 - CPU core
 - CPU on-chip peripherals
 - Standalone peripherals
 - Virtual devices
 - Software – an RTOS, a library (e.g. FSL's touch sensing library)
- Processor components include support for the following architectures
 - ColdFire/ColdFire+
 - 56800/E (DSC)
 - Kinetis
 - RS08/S08
 - S12Z

Processor Expert Project Design

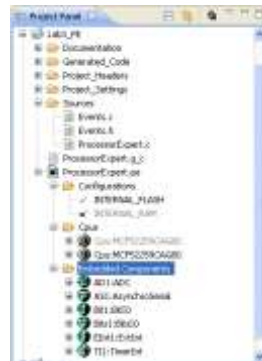
Create Project

- Create a new project with Processor Expert



Configure Components

- Use Inspector to set all component settings



Generate Code

- Let Processor Expert generate all components drivers code



Build and Debug

- Build the application common way
- Debug the application with CodeWarriror



Processor Expert Project

Design Timeline

Add Components

- Select components required in the project from Components Library



Verify Settings

- Make sure there are no design-time errors in the project



Write Code

- Write application code using code generated for components



Processor Expert Views – C/C++ Perspective

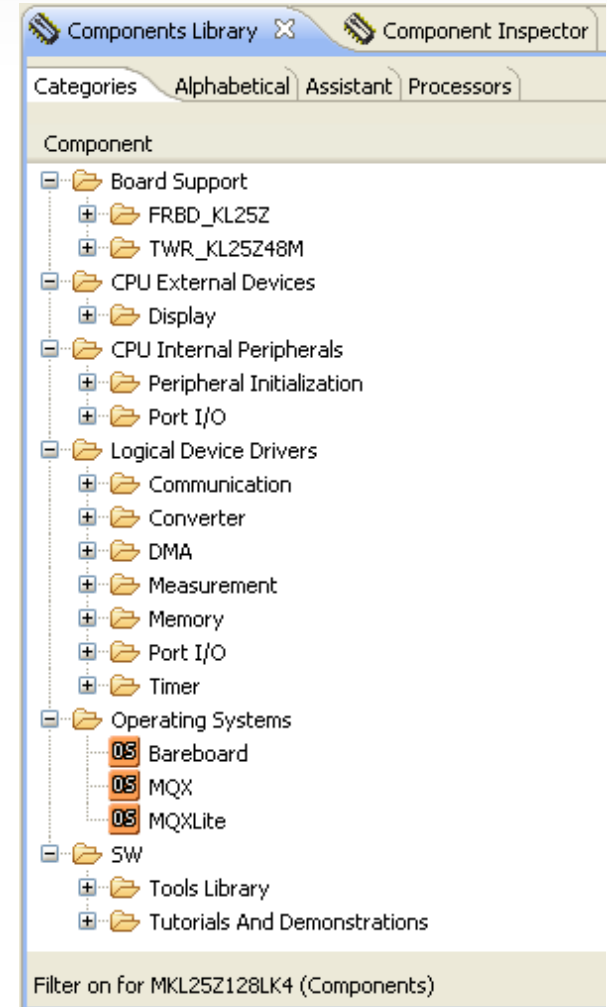
The screenshot displays the CodeWarrior Studio IDE with several callouts highlighting specific views and components:

- Projects View:** Points to the 'CodeWarrior Projects' sidebar on the left, which lists the project 'LED : MC9S08QE128' and its associated files like 'Binaries', 'Documentation', and 'Generated_Code'.
- Components Library:** Points to the 'Components Library' window at the top, which shows a table of available components and their properties.
- Component Inspector:** Points to the 'Component Inspector - Cpu' window, which provides detailed configuration for the selected component, such as 'Clock settings' and 'CPU interrupts'.
- Components View:** Points to the 'Components - LED' sidebar, which shows the hierarchy of components added to the project, including 'Release_508QE128CLK', 'Debug_508QE128CLK', and 'Processors'.
- Problems View:** Points to the 'Problems' window at the bottom, which displays any compilation or runtime errors.

The main editor window shows the source code for 'Events.c', which includes a description of a timer interrupt event and a function definition for 'TI1_OnInterrupt'.

Components Library

- Available components are displayed in Components Library
- Select appropriate TAB to change how components are organized and displayed
 - Categories
 - Alphabetical
 - Assistant
 - Processors
- Select filter to toggle displayed components
 - Show only components available for selected processor.
 - Show all components available .



Component Inspector

***Component Inspector - UART0**

Basic Advanced Expert

Properties Methods

Name	Value	Details
Device	UART0	UART0
Settings		
Clock gate	Do not initialize	
Clock settings		
Baud rate divisor	40	
Baud rate fine adjust	0	D
Baud rate	32768.0000 baud	
Transfer settings		
Data format	8bit	
Bits ordering	LSB first	
Parity	Off	
Parity placement	Parity in last data bit	
Idle character counting	After start bit	
Break character generation	Short	
LIN Break detection	Disabled	
Stop in mode	Disabled	
Receiver wakeup setting		
Pin settings		
Pins/Signals		
Receiver pin	Enabled	
Pin	ADC0_SE7b/PTD6/LLWU_P15...	ADC0_SE7b/PTD6/LLWU_P15...
Transmitter pin	Enabled	
Pin		Unassigned peripheral
Transmitter module	Disabled	
CTS pin	Disabled	
RTS pin	Disabled	
Interrupts/DMA		

Properties

Property Value

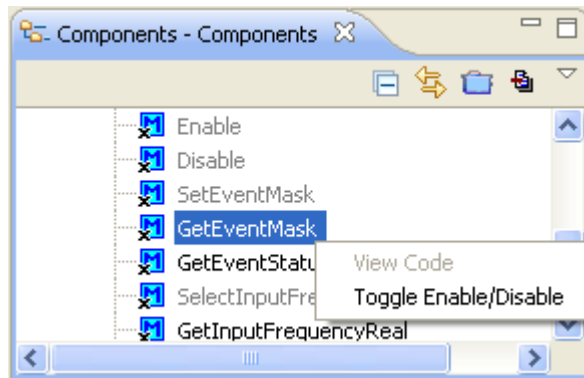
Select level of information displayed

Recently changed items are highlighted

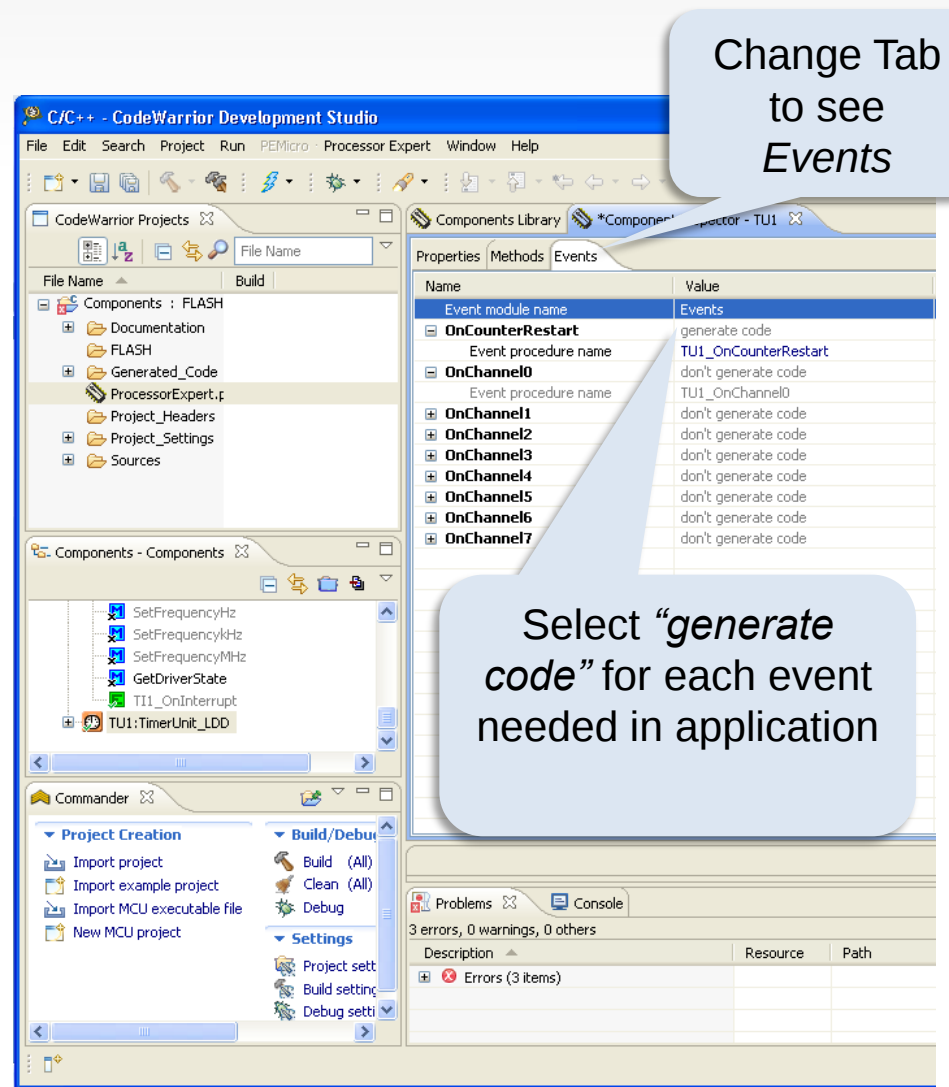
Details

Component Inspector – Events

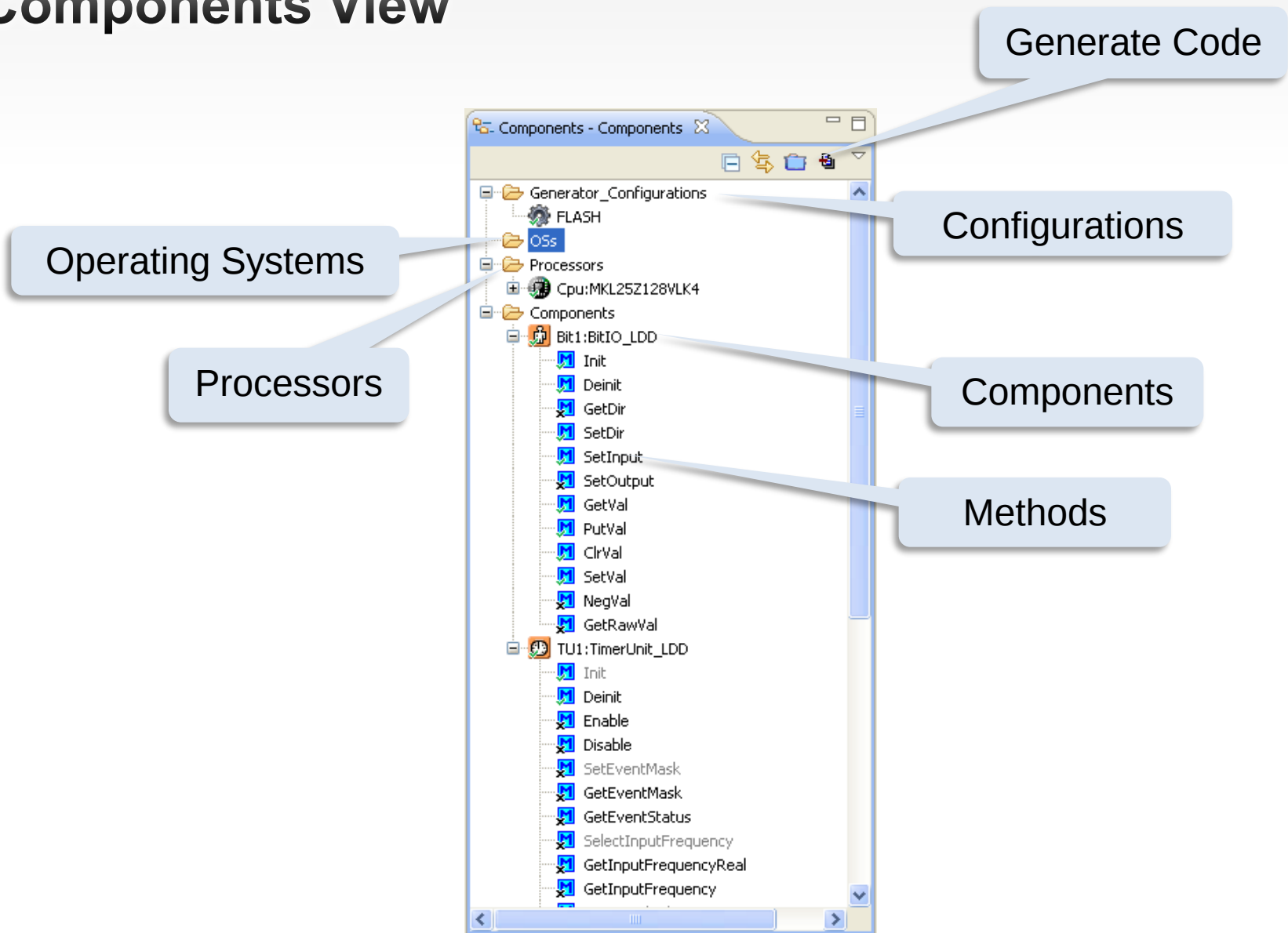
- Select the *Events TAB*
- Select “generate code” for each event required in the application
- Code will only be generated for the events selected



- To enable code generation for an event in the Components View, right-click on the event.
- Select “*Toggle Enable/Disable*”

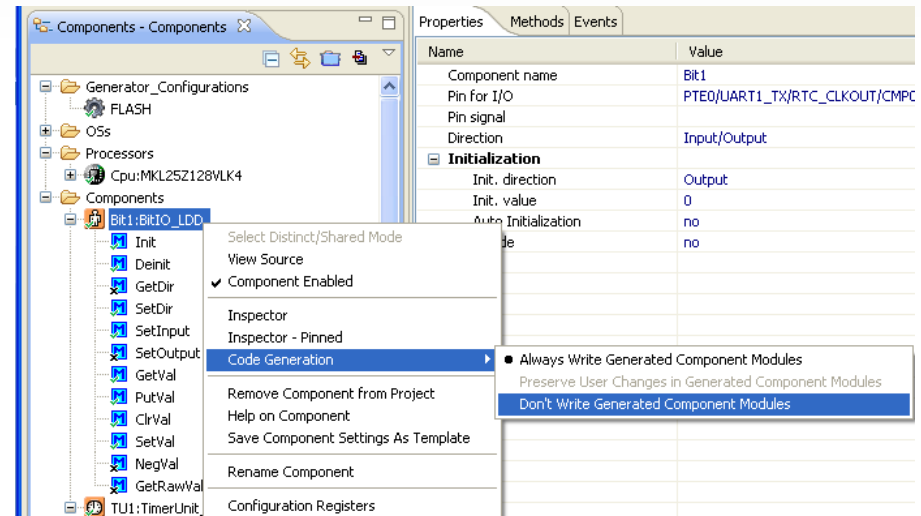


Components View



Components View – Code Generation

- To generate code
 - Select *Generate Code* icon in the *Components View*
- To turn off code generation for a component
 - Right-click on component
 - Select *Code Generation*
 - Select *Don't write Generated Component Modules*





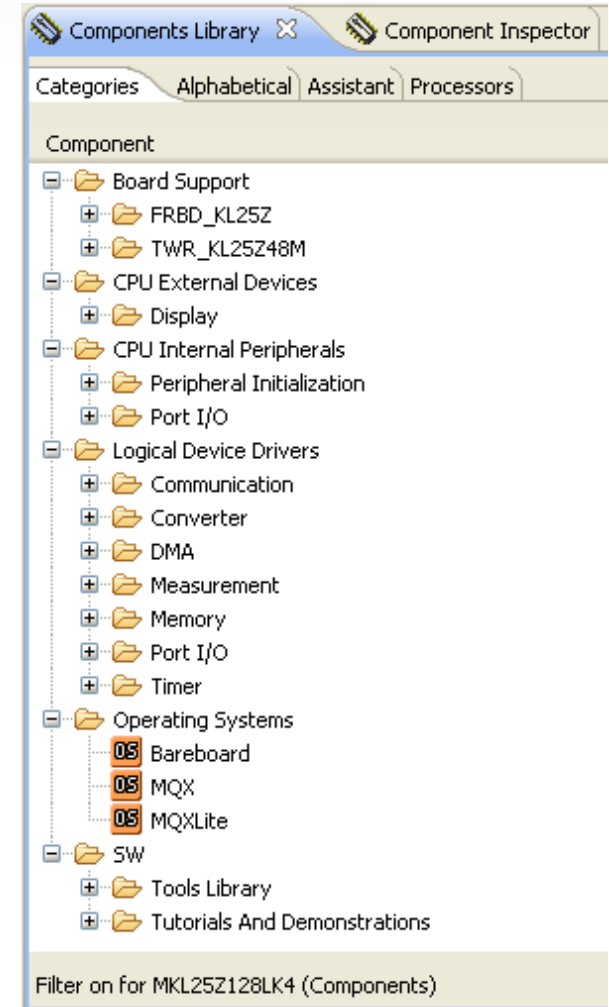
Processor Expert Software Embedded Components



Freescale, the Freescale logo, AllWin, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Wire, the Energy Efficient Solutions logo, i.MX, i.MX6, i.MX6GT, PGG, PowerQUICC, Processor Expert, QorIQ, QorIQ+, SafeAssure, the SafeAssure logo, StarCore, Symphony and Vybrid are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. AirStar, BeeBit, BeeStack, ClearNet, Flexio, LayerScope, MagiK, M6C, Platform in a Package, QorIQ Converge, QUICC Engine, ReadyPlay, SMARTMOS, Tower, TurboLink, Vybrid and Vybrid are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © 2013 Freescale Semiconductor, Inc.

Component Categories

- **Board Support**
 - Configuration file (template) including
 - CPU component
 - One or more pre-configured peripheral components
- **CPU External Devices**
 - Embedded Component support for Console I/O, LED displays, etc.
- **CPU Internal Peripherals (High level)**
 - Embedded Component support for internal peripherals
 - Standard API ensures generated code is portable
 - Generates initialization code and low level device drivers.
- **Peripheral Initialization**
 - Hardware specific support, which is not portable
 - Generates initialization code only



Component Categories

- **Logical Device Drivers**

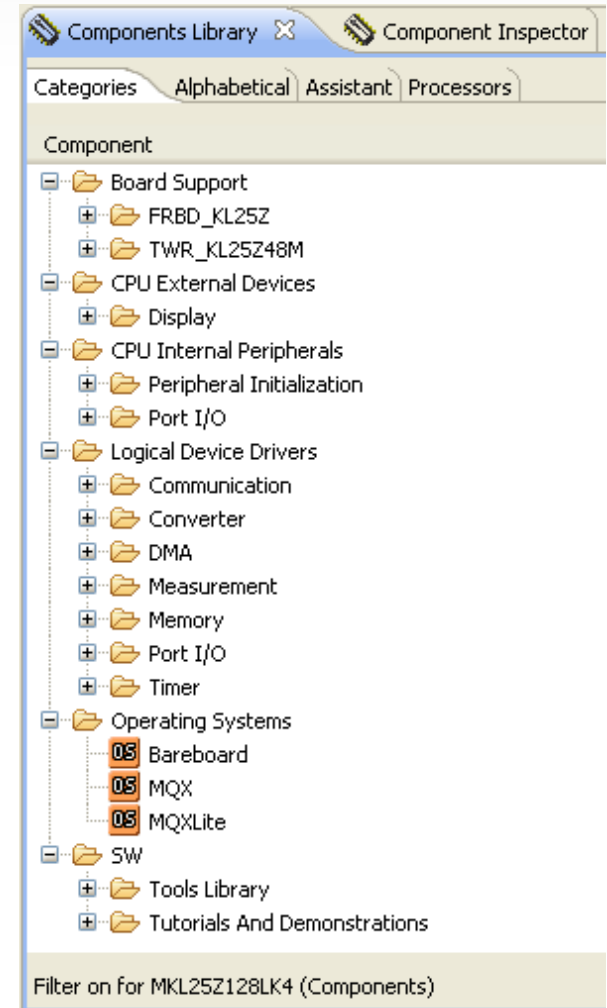
- Embedded Component support for internal peripherals
- Uses a standard API, which allows generated device drivers to be adapted for used with bare-metal applications as well as RTOS applications.

- **Operating Systems**

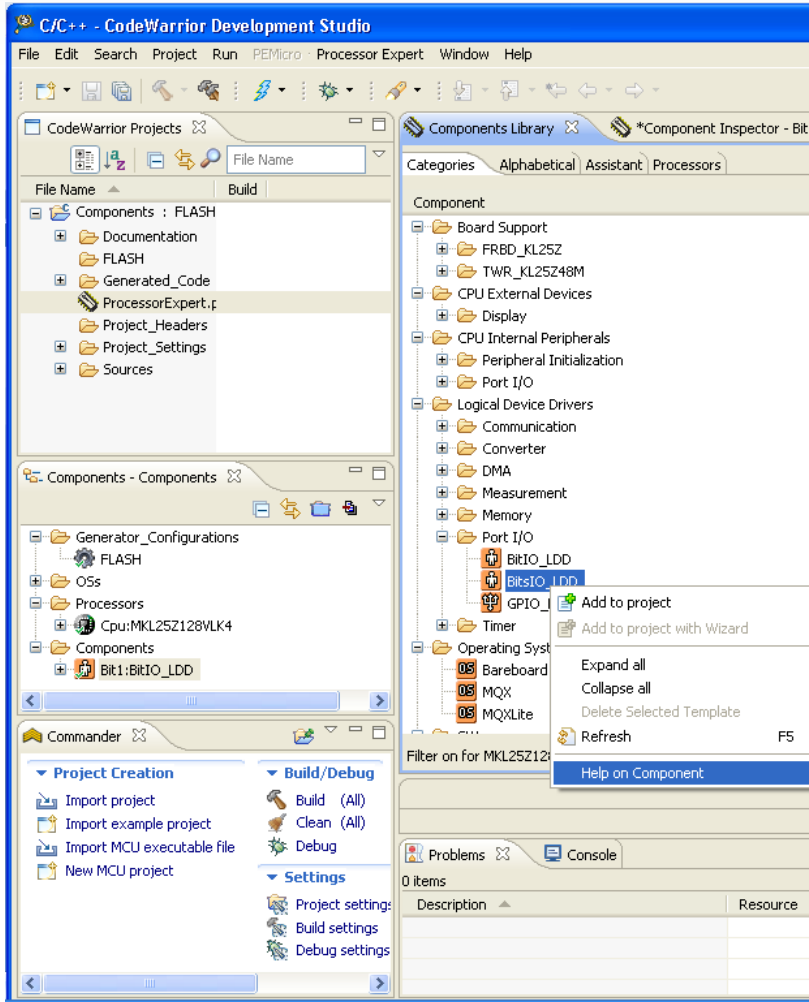
- RTOS adapter components
- Adapter defines the RTOS API
 - How memory is allocated
 - How a critical section is created
 - How interrupt vectors are allocated
- Generated device drivers are adapted to work with the RTOS API

- **Software**

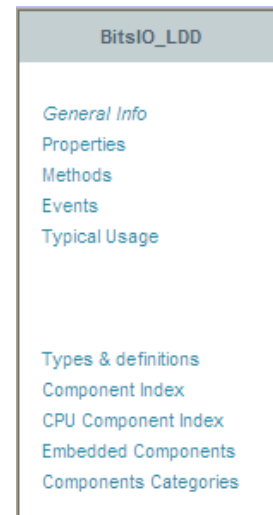
- Encapsulate software algorithms
- Add functionality to an existing component



Help on Component



- Right-click on the Component in the *Components Library*
- Select *Help on Component* in the pop-up menu
- Component documentation will be displayed with the following topics
 - General Info
 - Properties
 - Methods
 - Events
 - Typical Usage



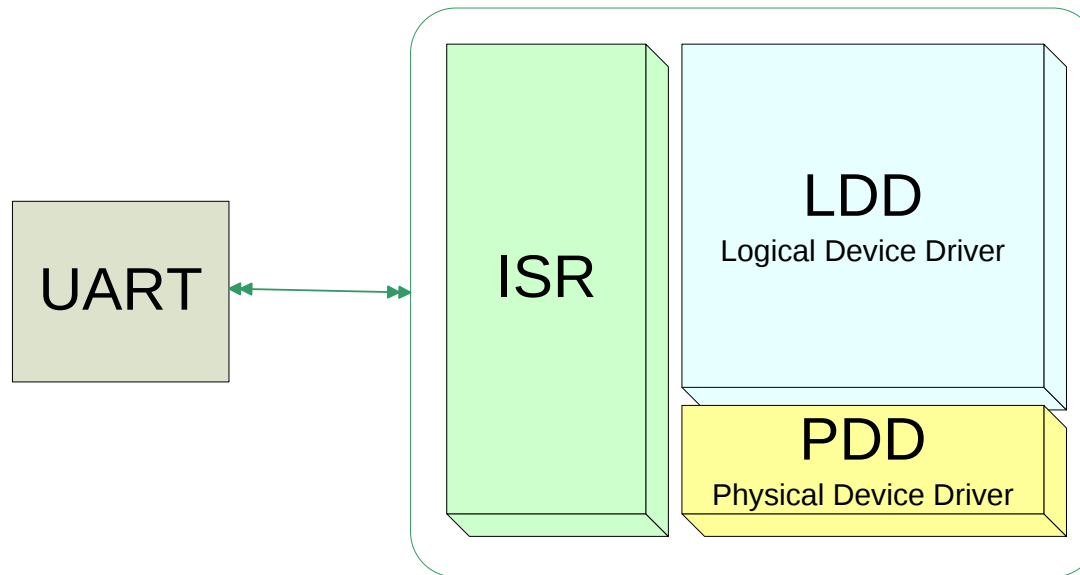


High Level and Logical Device Driver Components

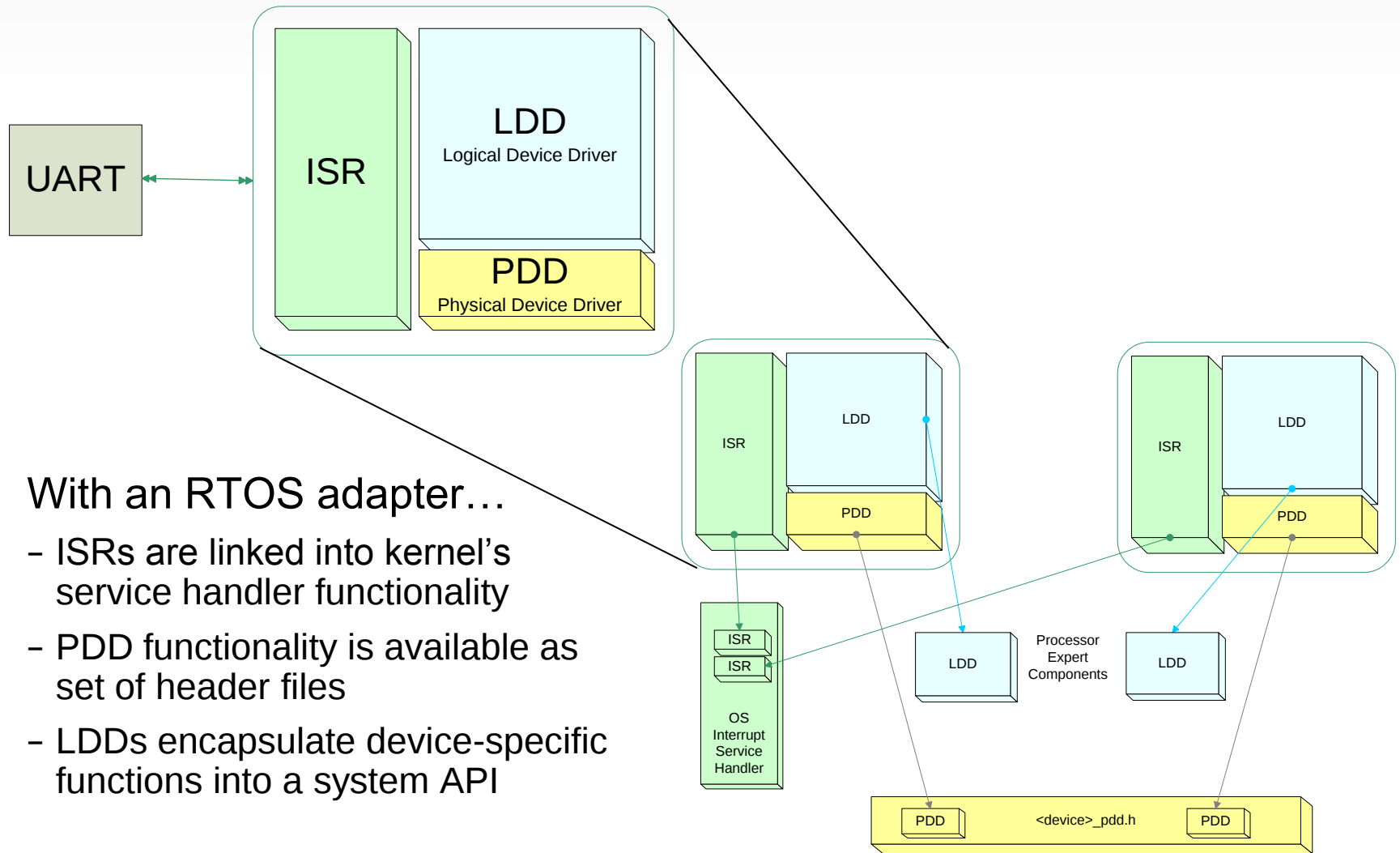
[illegible]

Peripheral Driver Anatomy

- Peripheral drivers...
 - Always include Physical Device Driver (PDD) layer. PDD uses base address and control registers to interface with the device.
 - May include Logical Device Driver (LDD) component. LDD handles buffering and state machine type logic.
 - May include interrupt or deferred service (ISR) routine. ISR provides callback functionality.



Peripheral Driver Anatomy



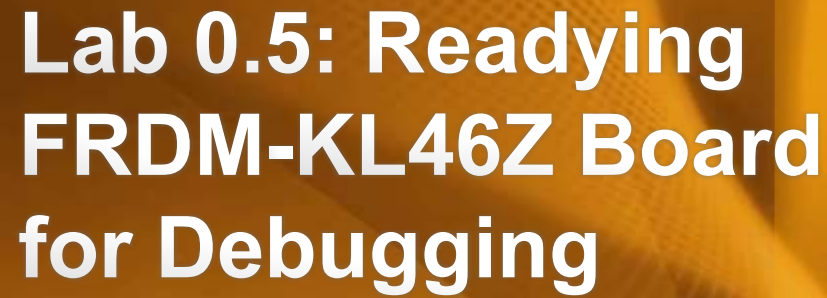
LDD Components

- Every component has Init() method
 - Only one parameter - UserDataPtr.
 - Initializes appropriate peripheral and driver.
- Every component has Deinit() method
 - De-initializes appropriate peripheral and driver.
- First parameter for every method is pointer to device structure returned from Init() method.
- Code generation is based on information in RTOS adapter
- Events can be enabled/disabled at runtime.
- Components can be disabled in Low Power Modes.
- CPU component
 - Does not automatically initialize components by default.
 - Auto-initialization can be enabled/disabled.

Migrate HLC to LDD

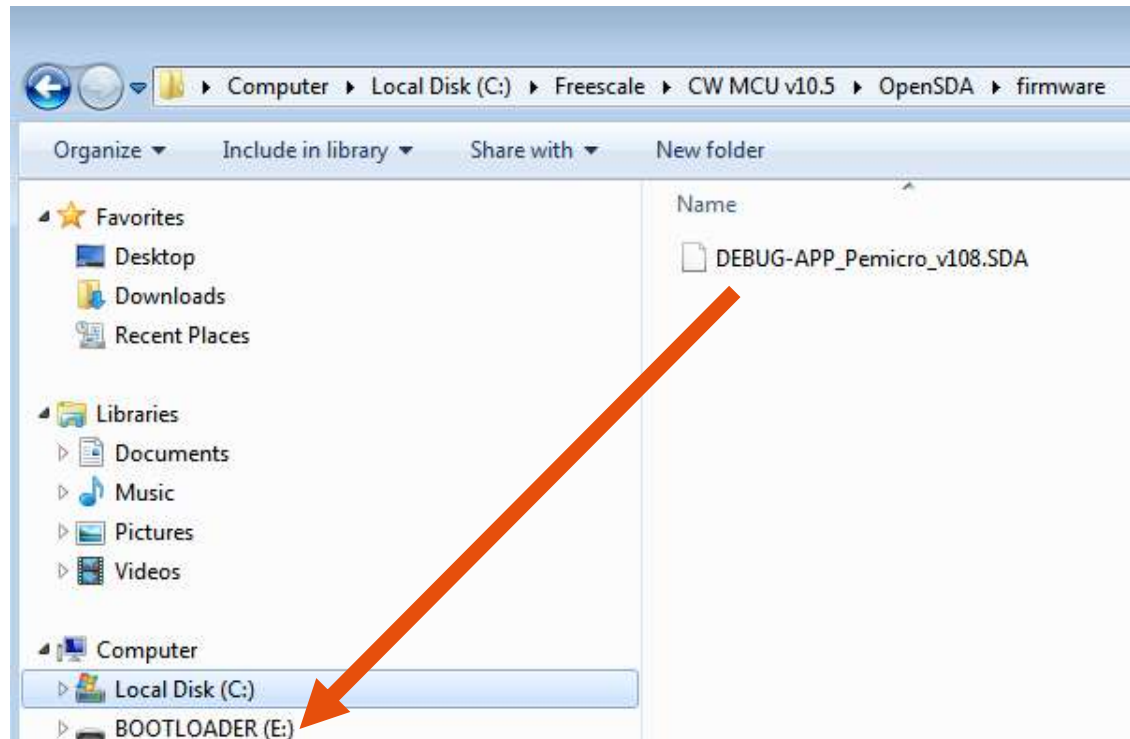
- Developed high level software components, which inherit from PDD or LDD, to migrate S08 projects to Kinetis L Series devices
 - Provides same application program interface (API)
 - Software wrapper at the level required to provide the necessary functionality
- High level components available for Kinetis

- AsynchroSerial - SynchroMaster - SynchroSlave - ADC - ExtInt - FreescaleAnalogComp	- IntFlash - BitIO - BitsIO - PWM - TimerInt
---	--

[illegible]

Readying FRDM-KL46Z Board

- Drag & Drop the DEBUG-APP-Pemicro_v10x.SDA to the BOOTLOADER
- C:\Freescale\CW MCU v10.5\OpenSDA\firmware



Readying FRDM-KL46Z Board

- Check for “solid” Green LED
- You are ready to go!

Green LED ON



Lab 1: Use New Project Wizard to Create Project



Create a new project to blink the LEDs

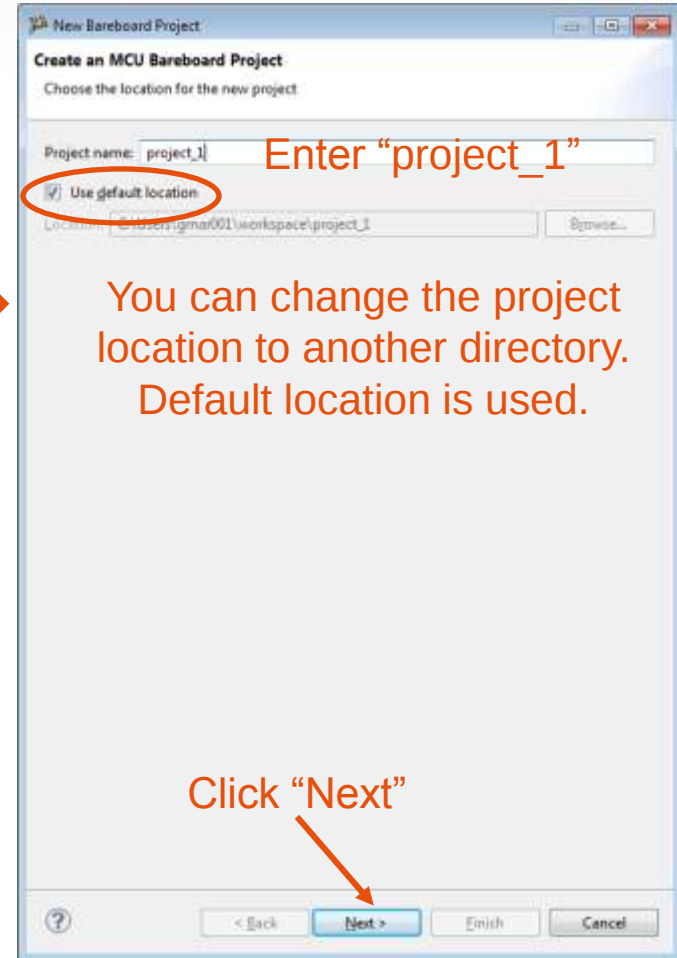
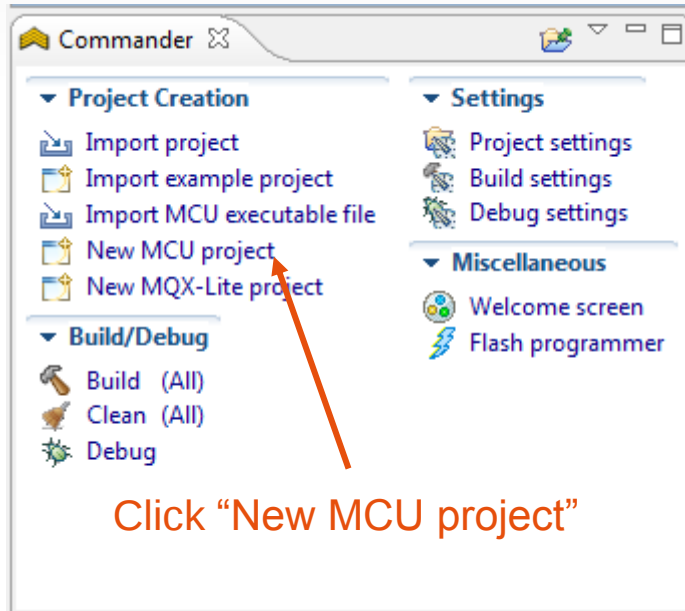
- This hands-on lab shows you how to...
 - Create a new project with the New Project Wizard
 - Configure Components with the Component Inspector
 - Use High Level Device functions
 - Import existing files
 - Build the project
 - Test the application's functionality
 - Trace the Code
- The lab uses the FRDM-KL46Z board
- The application will blink an LED periodically, and light a LED with button presses.

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 & Configure Components
 - Configure Components with the Component Inspector
 - Use High Level Device functions
 - Import existing components
 - Build the project
 - Test the application's functionality
 - Trace the Code

← Next up!

Open New Project Wizard & Select Project Name



Select Device and Connection

New Bareboard Project

Devices
Select the derivative or board y

Device or board to be used:
kl46

Enter "kl46" in filter

- Kinetis L Series
 - KL4x Family
 - KL46Z (48 MHz) Family
 - MKL46Z128
 - MKL46Z256

Select "MKL46Z256"

Project Type / Output:
☒ Application
☐ Library

Creates project for MKL46Z256 (48 Mhz) derivative

Click "Next"

< Back Next > Finish Cancel

New Bareboard Project

Connections
Choose the connection to use for this project

Connection to be used:

- ☐ P&E USB MultiLink Universal [FX] / USB MultiLink
- ☐ P&E Cyclone MAX
- ☐ P&E TraceLink
- ☐ Open Source JTAG
- ☒ OpenSDA
- ☐ Segger J-Link / J-Trace / SWO (SWD based)

De - Select:
"P&E USB MultiLink Universal..."

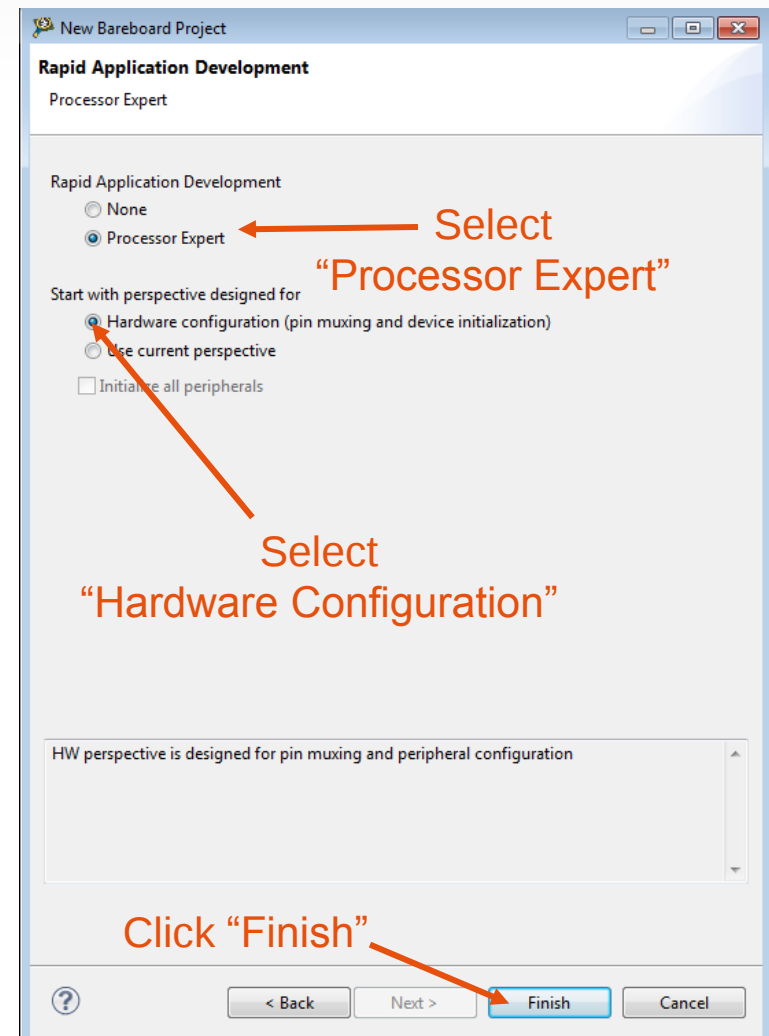
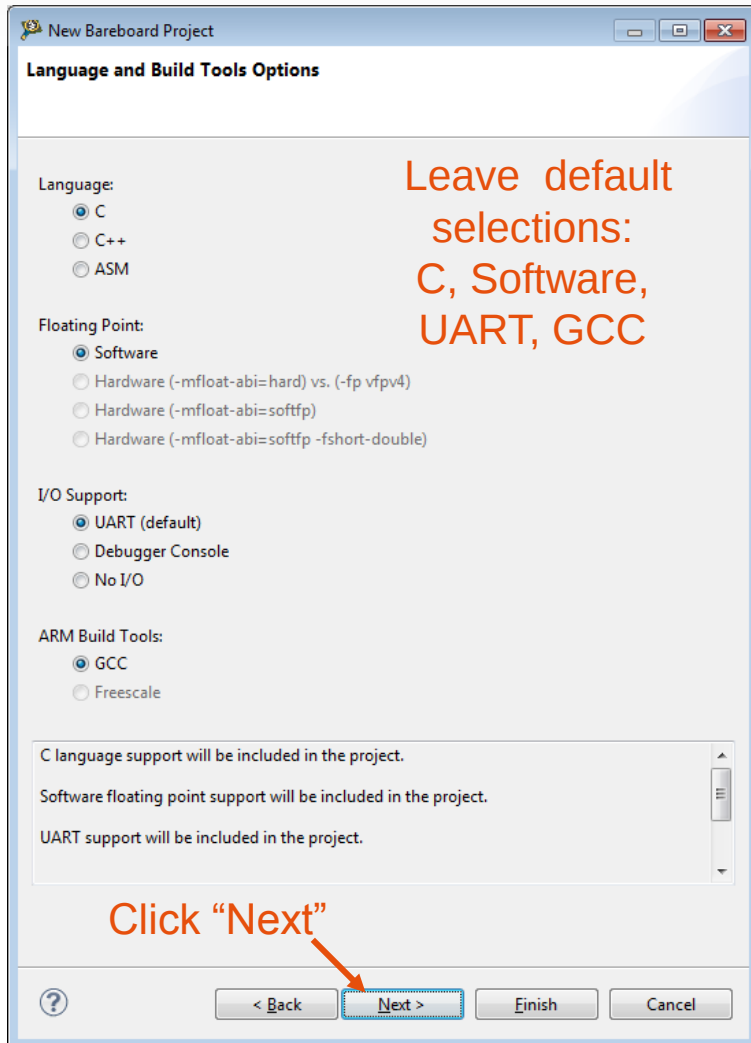
Select "Open Source SDA"

Connect to Segger J-Link / J-Trace / SWO (SWD based).

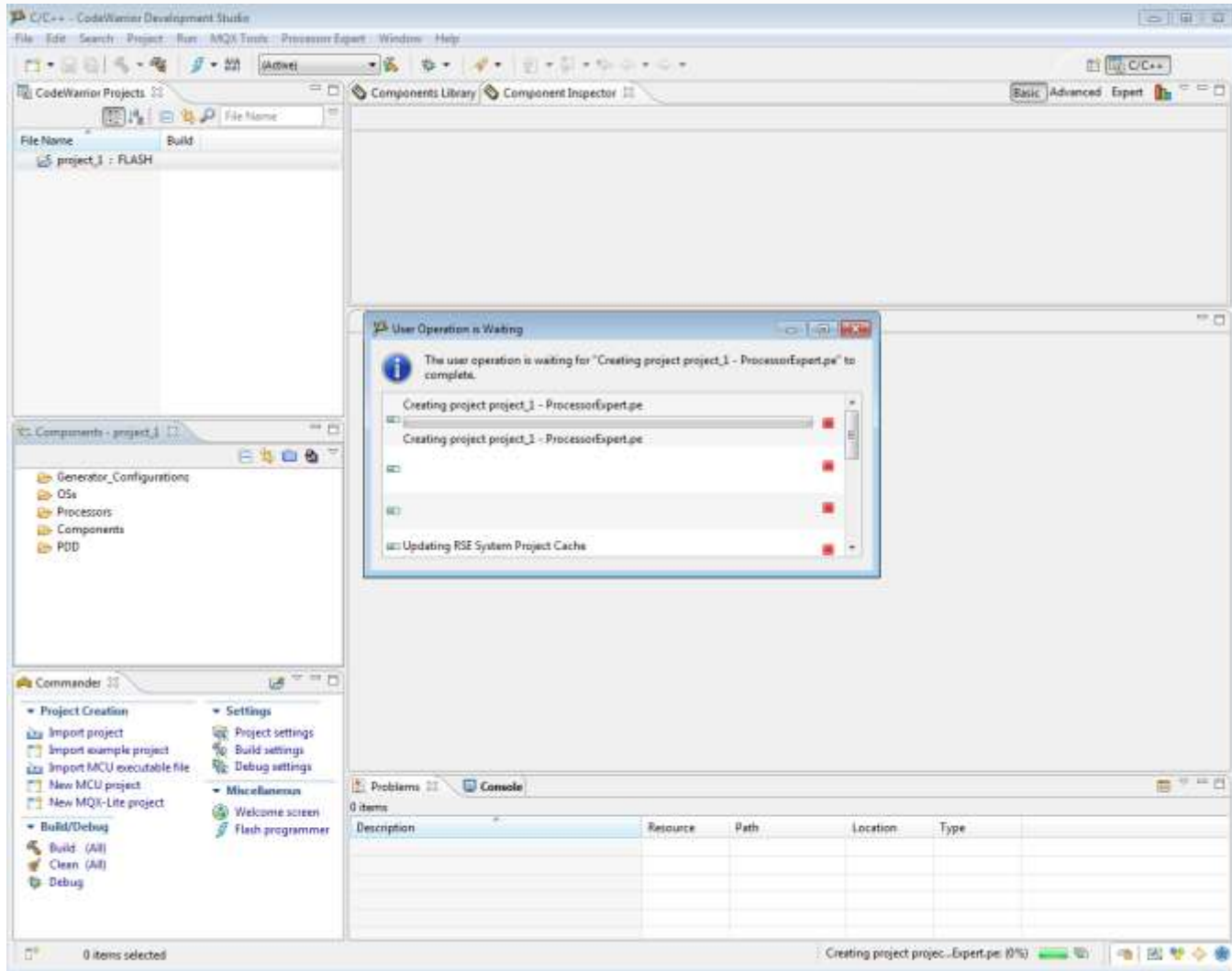
Click "Next"

< Back Next > Finish Cancel

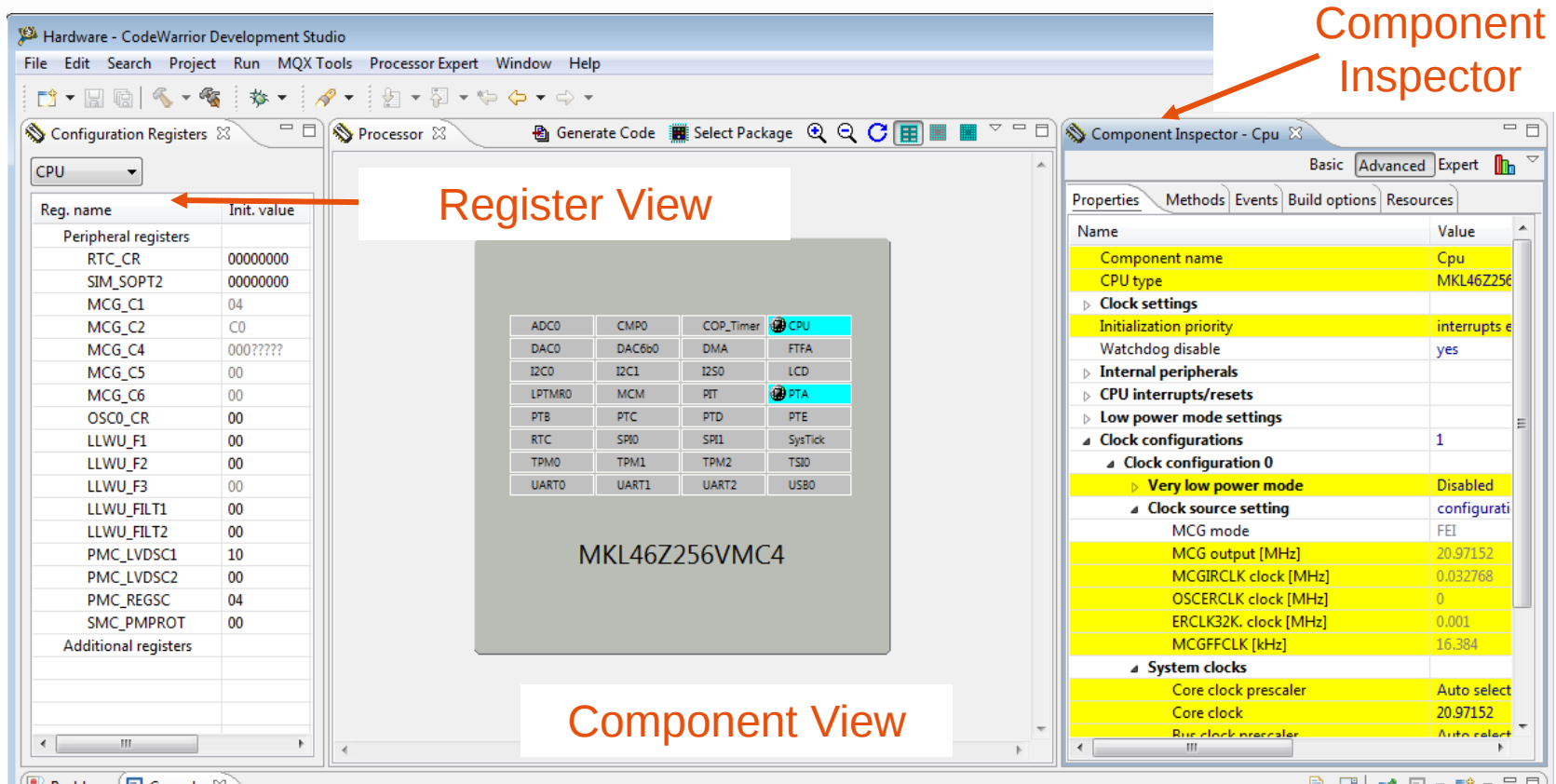
Select Language and Processor Expert



Project Creation in Progress

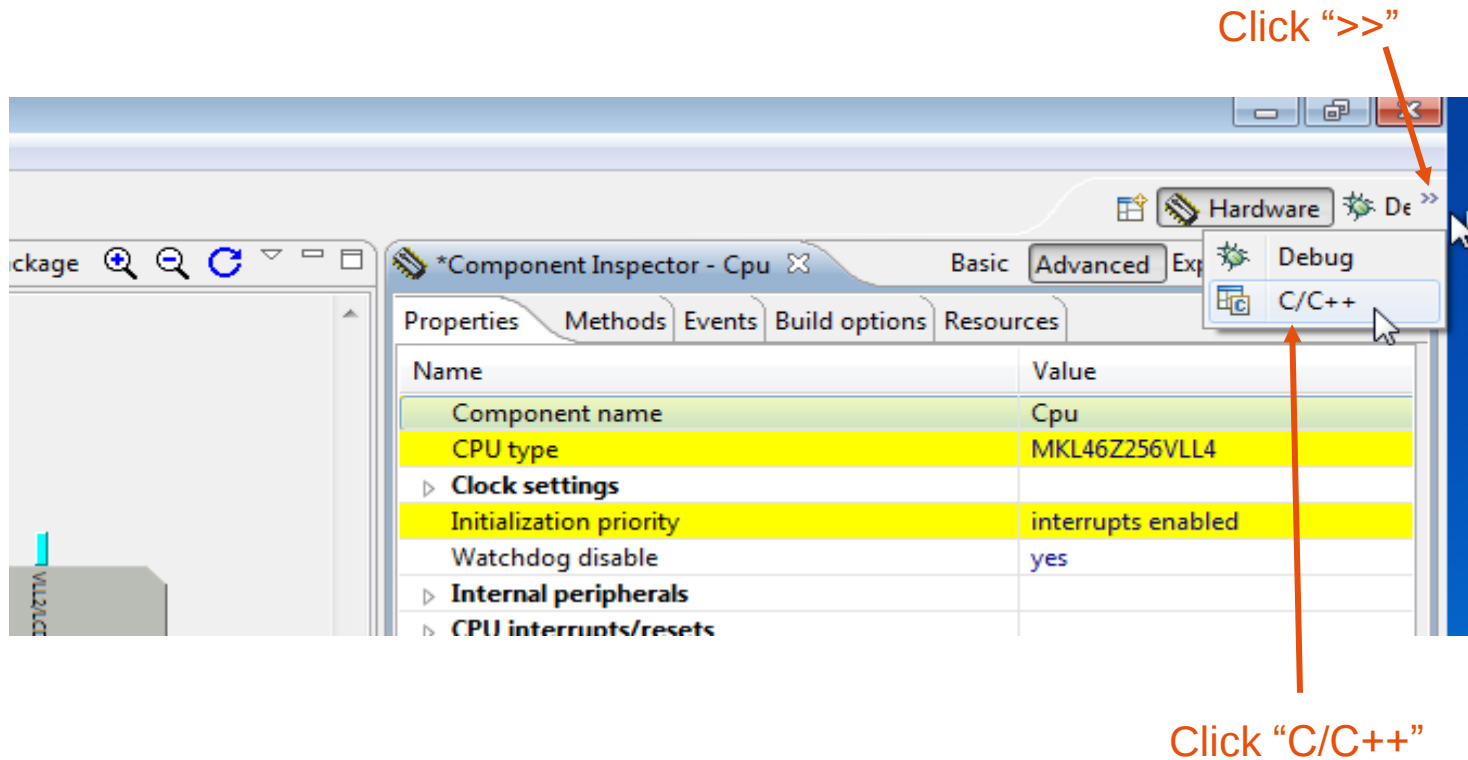


Project displayed in Hardware Perspective

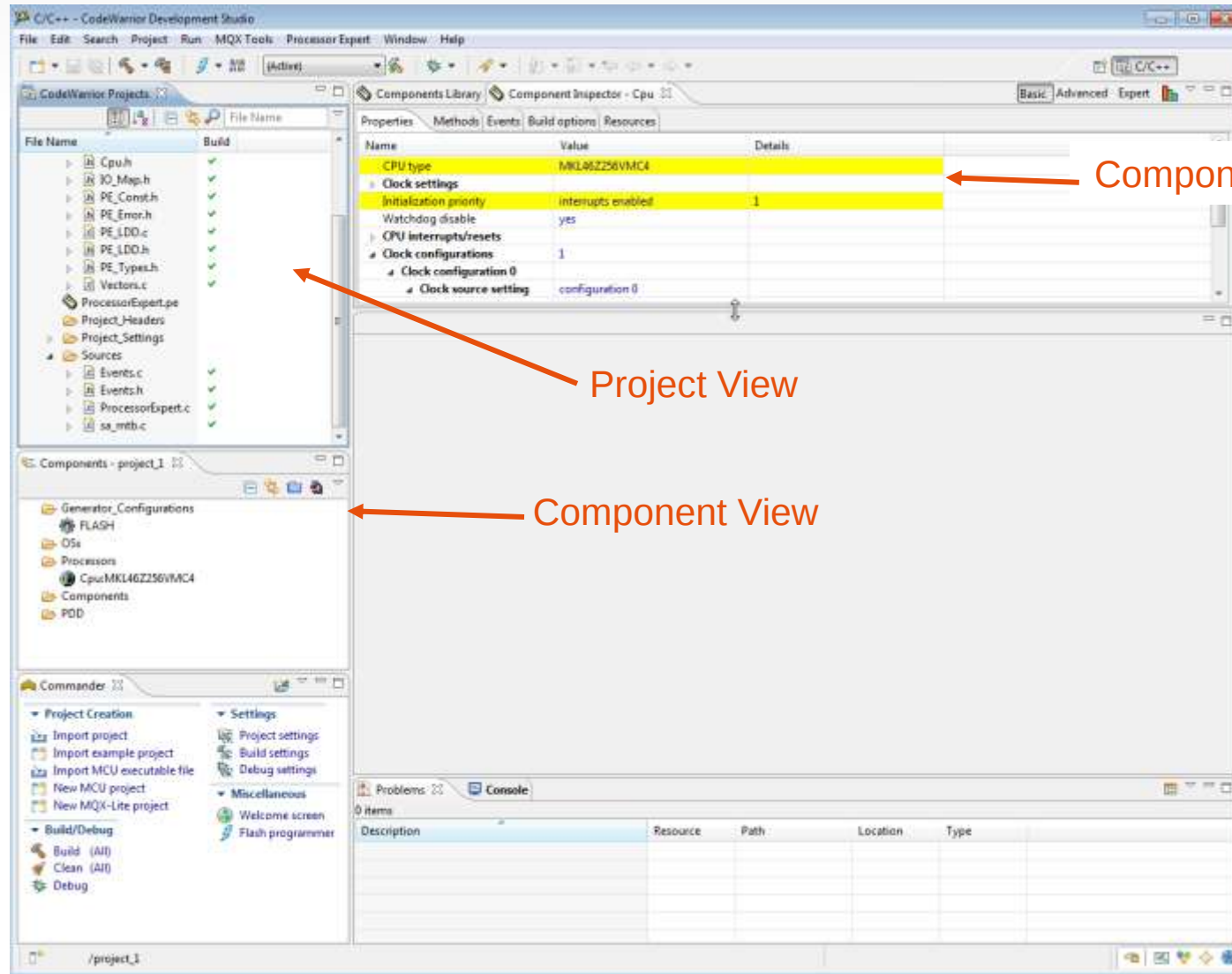


Project displayed in Hardware Perspective

- Select C++ Perspective



Project displayed in C/C++ Perspective



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 & Configure Components** ← **Next up!**
 - Configure Components with the Component Inspector
 - Use High Level Device functions
 - Import existing components
 - Build the project
 - Test the application's functionality
 - Trace the Code

Selecting Components

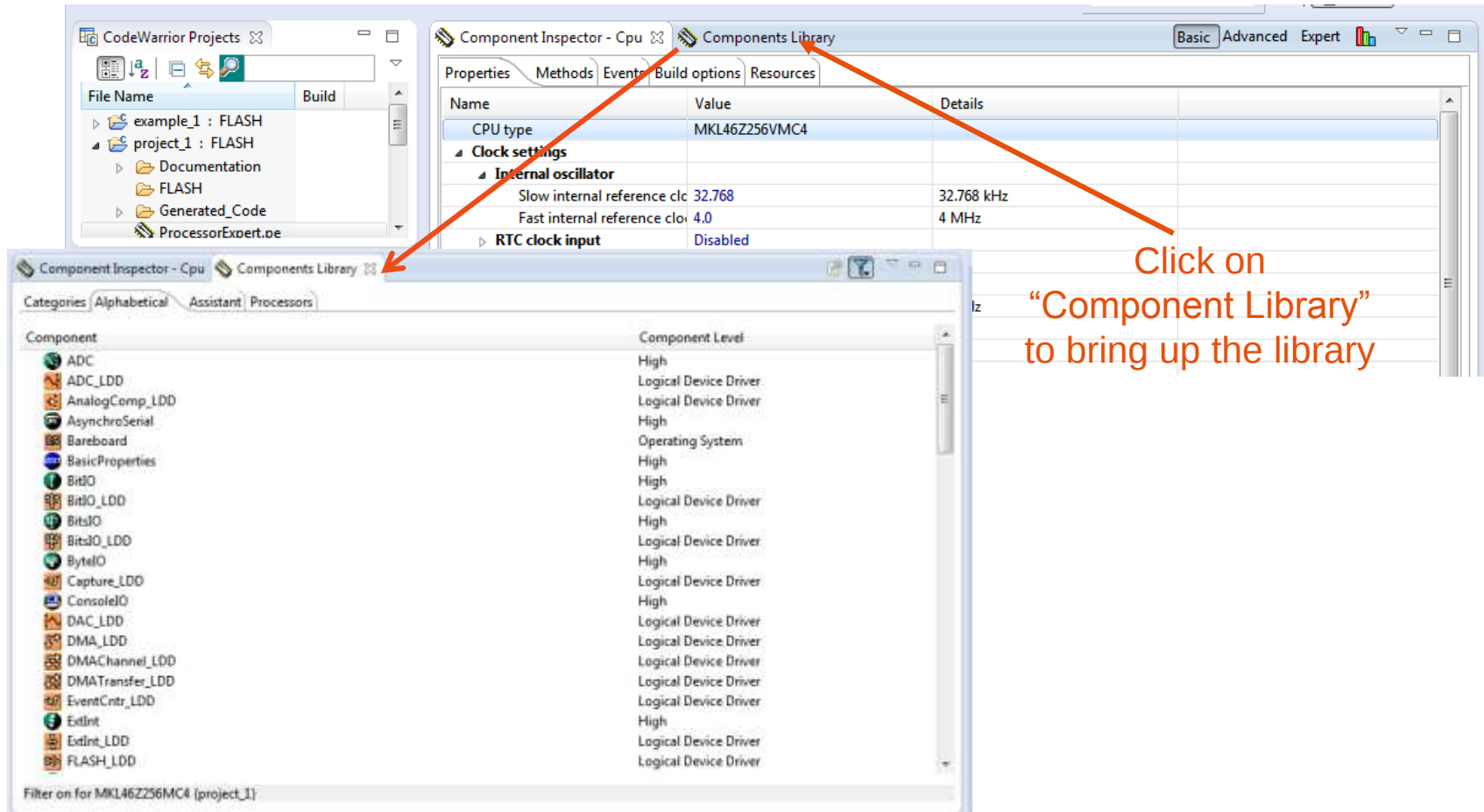
- Components Needed:
 - Processor: CPU: MKL46Z256VMC4 (Preselected based on project wizard information)
- Peripheral Initialization:
 - GPIO: Init_GPIO
- Other components needed for project (High Level Components)
 - GPIO: BitIO: for SW1
 - GPIO: BitIO: for SW2
 - GPIO: BitIO: for Red LED
 - GPIO: BitIO: for Green LED
 - Timer: TimerInt: Flashing the LED
 - AsynchroSerial: Serial UART for debug output

Component Inspector

The screenshot shows the CodeWarrior Development Studio interface. The 'Component Inspector' for the 'Cpu' component is open, displaying the 'Clock settings' section. The 'Initialization priority' is set to 'interrupts enabled' with a value of '1'. An orange arrow points to the 'Clock source setting' field, which is currently set to 'configuration 0'. The text 'Drag Down' is written next to the arrow, indicating the next step in the configuration process.

Selecting Components

- Switch to Component Library



Selecting Components

- Double Click on “Init_GPIO”

Component Inspector - Cpu Components Library

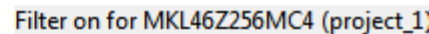
Categories Alphabetical Assistant Processors

Component	Component Level
Init_ADC	Peripheral Initialization
Init_COP	Peripheral Initialization
Init_DAC	Peripheral Initialization
Init_DMA	Peripheral Initialization
Init_FTL	Peripheral Initialization
Init_GPIO	Peripheral Initialization
Init_HSCMP	Peripheral Initialization

Component Inspector - Cpu Components Library

Categories Alphabetical Assistant Processors

Component	Component Level
Bareboard	Operating System
BasicProperties	High
BitIO	High
BitIO_LDD	Logical Device Driver
BitsIO	High
BitsIO_LDD	Logical Device Driver
ByteIO	High
Capture_LDD	Logical Device Driver
ConsoleIO	High
DAC_LDD	Logical Device Driver

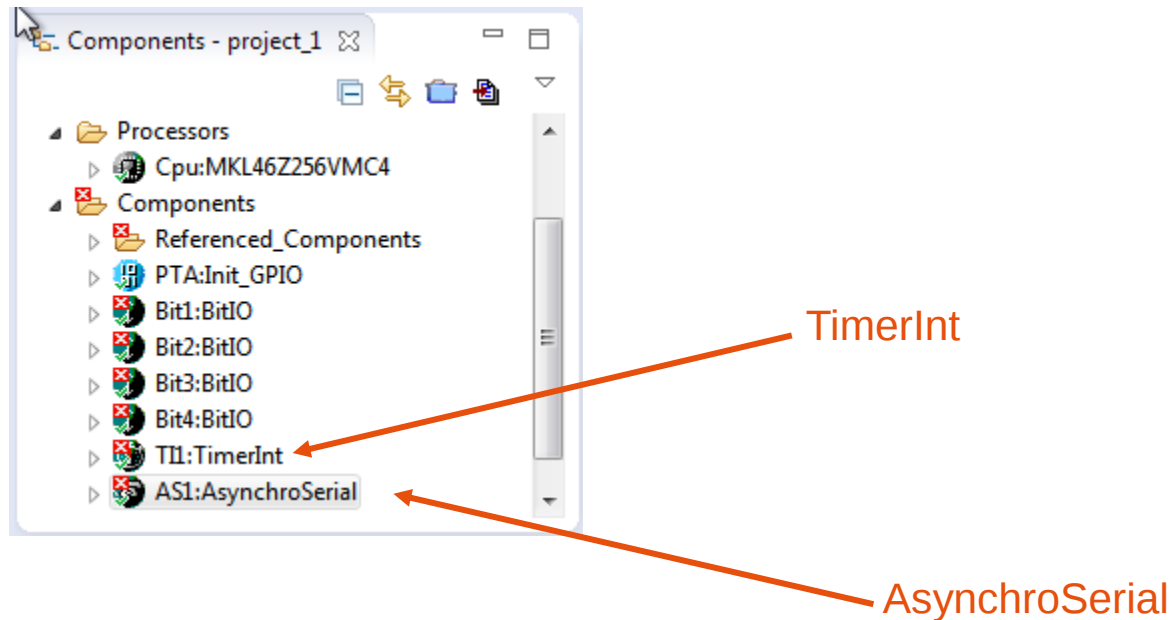


Double Click on TimerInt

Double Click on AsynchroSerial

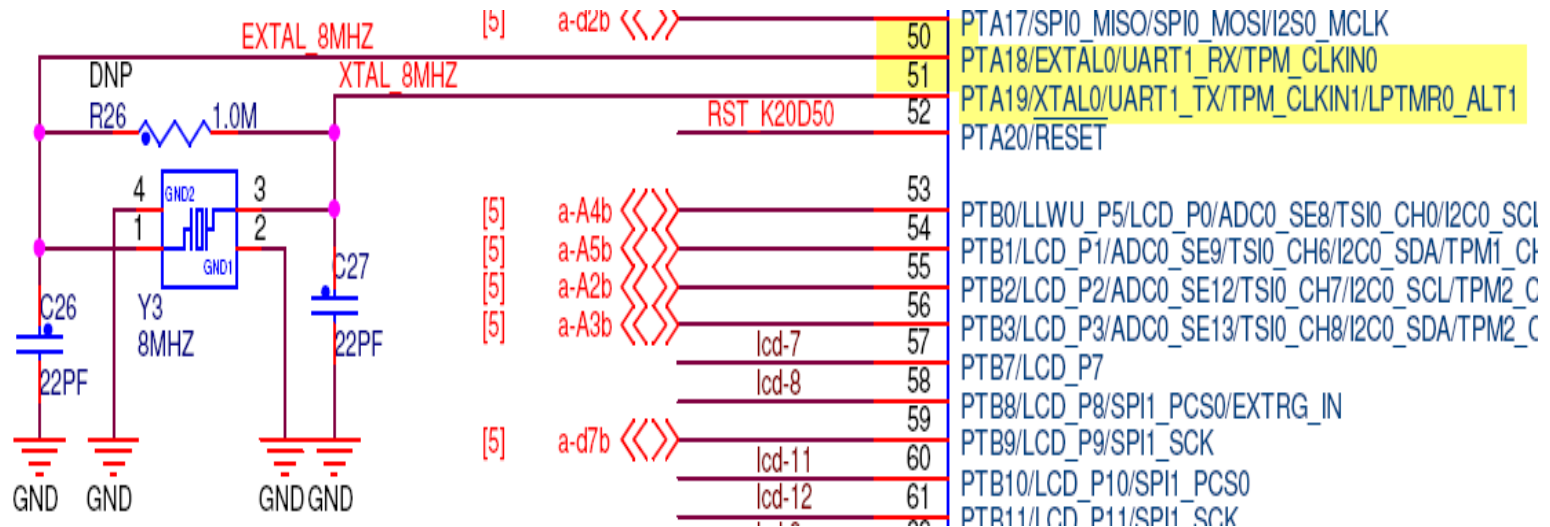
Selecting Components

- The component list should look like this:



FRDM-KL25Z Schematic

- 8.0 MHz External Clock Input
 - EXTAL0 – pin 50
 - XTAL0 – pin 51



Configure CPU Component

- Configure the CPU component as follows:
 - Package 100 pin LQFP
 - System Oscillator Enabled
 - External Clock 8MHz input
 - MCG Mode PEE
 - PLL Output 96MHz
 - Core Clock 48MHz
 - Bus Clock 24MHz

Component Inspector

The screenshot shows the CodeWarrior Development Studio interface. The 'Component Inspector - Cpu' window is open, displaying various configuration settings for the MKL46Z256VMC4. The 'CpuMKL46Z256VMC4' component is selected in the 'Components - project_1' pane. The 'Component Inspector' pane shows settings like CPU type, clock settings, and system clocks. The 'Console' pane at the bottom is empty.

Select "Component Inspector"

Select "Cpu"

Name	Value	Details
CPU type	MKL46Z256VMC4	
▶ Clock settings		
Initialization priority	interrupts enabled	1
Watchdog disable	yes	
▶ CPU interrupts/resets		
▶ Clock configurations	1	
▶ Clock configuration 0		
▶ Clock source setting	configuration 0	
MCG mode	FS	
▶ System clocks		
Core clock	20.97152	20.97152 MHz
Bus clock	20.97152	20.97152 MHz
▶ TPM clock selection	Auto select	PLL/FLL clock
Clock frequency [A]	20.97152	20.97152 MHz

Component Inspector – Package

The screenshot shows the 'Component Inspector - Cpu' window. The 'Properties' tab is selected. The 'Name' column lists various CPU properties, and the 'Value' column shows their current settings. The 'CPU type' property is expanded, showing a list of available packages. The 'MKL46Z256VLL4 in LQFP 100-pin package' is highlighted in blue. An orange arrow points to the far left edge of the 'Value' column header, with the text 'Click on far left edge' next to it. Another orange arrow points to the highlighted package name, with the text 'Select "LQFP 100-pin package"' below it.

Name	Value	Details
CPU type	MKL46Z256VLL4 in LQFP 100-pin	
▶ Clock settings	MKL46Z256VMC4 in MAPBGA 121-pin package	
Initialization priority	MKL46Z256VLL4 in LQFP 100-pin package	
Watchdog disable	MKL46Z256VLH4 in LQFP 64-pin package	
▶ CPU interrupts/resets		
▲ Clock configurations	1	
▲ Clock configuration 0		

Component Inspector – Clock Settings

The screenshot shows the CodeWarrior Development Studio interface. The main window is titled "C/C++ - CodeWarrior Development Studio". The menu bar includes File, Edit, Search, Project, Run, MQX Tools, Processor Expert, Window, and Help. The toolbar contains various icons for file operations, project management, and execution. The left sidebar shows the "CodeWarrior Projects" pane with a project named "project_1 : FLASH". Below it is the "Components - project_1" pane, which lists the components of the project: Generator_Configurations, FLASH, OSs, Processors, Cpu/MKL46Z256VMC4, Components, and PDD. The "Commander" pane at the bottom left shows a tree view of project creation and build/debug options. The main workspace is divided into two panes: "Components Library" and "Component Inspector - Cpu". The "Component Inspector" pane is active, showing a tree view of the CPU components and a table of their properties and values. The table has columns for Name, Value, and Details. The components listed include CPU type, Clock settings, Internal oscillator, RTC clock input, System oscillator 0, Clock source settings, MCG settings, FLL settings, PLL 0 settings, Initialization priority, Watchdog disable, CPU interrupts/resets, Clock configurations, and TPM clock selection. The values and details are displayed in the corresponding columns.

Name	Value	Details
CPU type	MKL46Z256VMC4	
Clock settings		
Internal oscillator		
Slow internal reference clk	32.768	32.768 kHz
Fast internal reference clk	4.0	4 MHz
RTC clock input	Disabled	
System oscillator 0	Disabled	
Clock source settings	1	
Clock source setting 0		
MCG settings		
MCG mode	FEI	
MCG output (MHz)	20.97152	20.97152 MHz
MCG external ref. c	0.0	0 MHz; System Oscillator is disabled.
FLL settings		
FLL module	Enabled	
FLL output (MHz)	20.97152	20.97152 MHz
PLL 0 settings		
PLL module	Disabled	
PLL output (MHz)	0.0	0 MHz; PLL is disabled
Initialization priority	interrupts enabled	1
Watchdog disable	yes	
CPU interrupts/resets		
Clock configurations	1	
Clock configuration 0		
Clock source setting	configuration 0	
MCG mode	FEI	
System clocks		
Core clock	20.97152	20.97152 MHz
Bus clock	20.97152	20.97152 MHz
TPM clock selection	Auto select	PLL/FLL clock
Clock frequency (Hz)	20.97152	20.97152 MHz

Component Inspector – Clock Settings

The screenshot shows the CodeWarrior Development Studio interface. The 'Component Inspector - Cpu' window is open, displaying the 'Clock settings' section. A red arrow points to the 'Clock settings' section in the 'Components Library' pane on the left, with the text 'Expand "Clock Settings"' overlaid. The 'Clock settings' section is expanded, showing a table of properties and values.

Name	Value	Details
CPU type	MKL46Z256VMC4	
Clock settings		
Initialization priority	interrupts enabled	1
Watchdog disable	yes	
CPU interrupts/resets		
Clock configurations	1	
Clock configuration 0		
Clock source setting	configuration 0	
MCG mode	FEI	
System clocks		
Core clock	20.97152	20.97152 MHz
Bus clock	20.97152	20.97152 MHz
TPM clock selection	Auto select	PLL/FLL clock
Clock frequency [A]	20.97152	20.97152 MHz

Component Inspector – Clock Settings

Component Inspector - Cpu		
Properties	Methods	Events
Name	Value	Details
CPU type	MKL46Z256VMC4	
Clock settings		
Internal oscillator		
Slow internal reference clock [kHz]	32.768	32.768 kHz
Fast internal reference clock [MHz]	4.0	4 MHz
RTC clock input	Disabled	
System oscillator 0	Disabled	
Clock source settings	1	
Clock source setting 0		
MCG settings		
MCG mode	FEI	
MCG output [MHz]	20.97152	20.97152 MHz
MCG external ref. clock [MHz]	0.0	0 MHz; System Oscillator is disabled.
FLL settings		
FLL module	Enabled	
FLL output [MHz]	20.97152	20.97152 MHz
PLL 0 settings		
PLL module	Disabled	
PLL output [MHz]	0.0	0 MHz; PLL is disabled
Initialization priority	interrupts enabled	1
Watchdog disable	yes	
CPU interrupts/resets		
Clock configurations	1	
Clock configuration 0		
Clock source setting	configuration 0	
MCG mode	FEI	
System clocks		
Core clock	20.97152	20.97152 MHz
Bus clock	20.97152	20.97152 MHz
TPM clock selection	Auto select	PLL/FLL clock
Clock frequency [MHz]	20.97152	20.97152 MHz

Select
"Enable"

Component Inspector – Clock Settings

System oscillator 0	Enabled	
Clock source	External	This property selects the MCG mode.
Clock frequency [MHz]	8.0	
Clock source settings	1	
Clock source setting 0		
MCG settings		
MCG mode	FEI	-FLL Engaged Internal (FEI) In FEI mode, MCG output clock is derived from the FLL output clock that is controlled by the 32 kHz Slow internal reference clock. The FLL loop will lock the DCO frequency to the FLL factor, as selected by the FLL Multiplication factor, times the internal reference frequency.
MCG output [MHz]	20.97	In FEI mode, the PLL is disabled in a low-power state unless enabled by PLL module property.
MCG external ref. clock [MHz]	8.0	-FLL Engaged External (FEE) In FEE mode, MCG output clock is derived from the FLL output clock that is controlled by the external reference clock. The source of the external reference clock is controlled by MCG ref. clock source property. The FLL loop will lock the DCO frequency to the FLL factor, as selected by the FLL Multiplication factor, times the times the FLL reference clock, as specified by the Reference clock divider.
FLL settings		In FEE mode, the PLL is disabled in a low-power state unless enabled by PLL module property.
FLL module	Enabled	-FLL Bypassed Internal (FBI) In FBI mode, the MCG output clock is derived either from the slow Slow internal reference clock or Fast internal reference clock, as selected by the IRCLK source property. The FLL is operational but its output is not used. The FLL output clock is controlled by the Slow internal reference clock. The FLL loop will lock the DCO frequency to the FLL factor, as selected by the FLL Multiplication factor, times the internal reference frequency.
FLL output [MHz]	20.97	In FBI mode, the PLL is disabled in a low-power state unless enabled by PLL module property.
PLL 0 settings		-FLL Bypassed External (FBE) In FBE mode, the MCG output clock is derived from the external reference clock. Source of the external reference clock is controlled by MCG ref. clock source property. The FLL is operational but its output is not used. The FLL output clock is controlled by the external reference clock. The FLL loop will lock the DCO frequency to the FLL factor, as selected by the FLL Multiplication factor, times the times the FLL reference clock, as specified by the Reference clock divider.
PLL module	Disabled	In FBE mode, the PLL is disabled in a low-power state unless enabled by PLL module property.
PLL output [MHz]	0.0	-PLL Engaged External (PEE) In PEE mode, the MCG output [MHz] clock is derived from the PLL output clock, which is controlled by the external reference clock. Source of the external reference clock is controlled by MCG ref. clock source property. The PLL clock frequency locks to a multiplication factor, as specified by PLL Multiplication factor, times the PLL reference clockexternal reference frequency, as specified by PLL Reference clock divider.
Initialization priority	internal	The FLL is disabled in a low-power state.
Watchdog disable	yes	- PLL Bypassed External (PBE)In PBE mode, MCG outputclock is derived from the external reference clock. The PLL is operational, but its output clock is not used. The PLL clock frequency locks to a multiplication factor, as specified by PLL Multiplication factor, times the PLL reference clockexternal reference frequency, as specified by PLL Reference clock divider.
CPU interrupts/resets		The FLL is disabled in a low-power state.
Clock configurations	1	-Bypassed Low Power Internal (BLPI) In BLPI mode, MCG output clock is derived either from the slow Slow internal reference clock or Fast internal reference clock, as selected by the IRCLK source property.
Clock configuration 0	configuration	Both the FLL and PLL are disabled.
Clock source setting	FEI	-Bypassed Low Power External (BLPE) In BLPE mode, MCG outputclock is derived from the external reference clock. The source of the external reference clock is controlled by MCG ref. clock source property.
MCG mode		Both the FLL and PLL are disabled.
System clocks		
Core clock	20.97	
Bus clock	20.97	
TPM clock selection	Auto	
Clock frequency [MHz]	20.97	

Select “PEE”

Component Inspector – Clock Settings

Name	Value	Details
Clock settings		
Internal oscillator		
Slow internal reference clock [kHz]	32.768	32.768 kHz
Fast internal reference clock [MHz]	4.0	4 MHz
RTC clock input	Disabled	
System oscillator 0	Enabled	
Clock source	External crystal	
Clock frequency [MHz]	8.0	8 MHz
Clock source settings	1	
Clock source setting 0		
MCG settings		
MCG mode	PEE	
MCG output [MHz]	100.0	100 MHz
MCG external ref. clock [MHz]	8.0	8 MHz
FLL settings		
FLL module	Disabled	
FLL output [MHz]	0.0	0 MHz; FLL is disabled.
PLL 0 settings		
PLL module	Enabled	
PLL output [MHz]	100.0	100 MHz
Initialization priority	interrupts enabled	1
Watchdog disable	yes	
CPU interrupts/resets		
⚠ Clock configurations	1	
⚠ Clock configuration 0		
Clock source setting	configuration 0	
MCG mode	PEE	
⚠ System clocks		
⚠ Core clock	20.97152	Not possible to set requested valu...
Bus clock	20.97152	20.97152 MHz
TPM clock selection	Auto select	PLL/FLL clock
Clock frequency [MHz]	50.0	50 MHz

Enter "96"

Component Inspector – Clock Configurations

Name	Value	Details
▲ Clock settings		
▲ Internal oscillator		
Slow internal reference clock [kHz]	32.768	32.768 kHz
Fast internal reference clock [MHz]	4.0	4 MHz
▶ RTC clock input	Disabled	
▲ System oscillator 0	Enabled	
▲ Clock source	External crystal	
Clock frequency [MHz]	8.0	8 MHz
▲ Clock source settings	1	
▲ Clock source setting 0		
▲ MCG settings		
MCG mode	PEE	
MCG output [MHz]	96.0	96 MHz
MCG external ref. clock [MHz]	8.0	8 MHz
▲ FLL settings		
FLL module	Disabled	
FLL output [MHz]	0.0	0 MHz; FLL is disabled.
▲ PLL 0 settings		
PLL module	Enabled	
PLL output [MHz]	96.0	96 MHz
Initialization priority	interrupts enabled	1
Watchdog disable	yes	
▶ CPU interrupts/resets		
▲ ⚠ Clock configurations	1	
▲ ⚠ Clock configuration 0		
▲ Clock source setting	configuration 0	
MCG mode	PEE	
▲ ⚠ System clocks		
⚠ Core clock	20.97152	Not possible to set requested value
Bus clock	20.97152	20.97152 MHz
▲ TPM clock selection	Auto select	PLL/FLL clock
Clock frequency [MHz]	48.0	48 MHz

Enter "48"

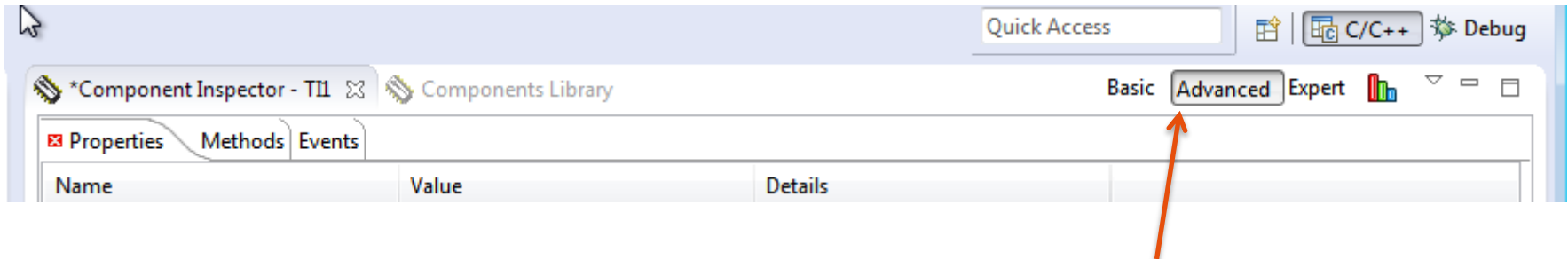
Enter "24"

Component Inspector: CPU Summary

Name	Value	Details
Clock settings		
Internal oscillator		
Slow internal reference clock [kHz]	32.768	32.768 kHz
Fast internal reference clock [MHz]	4.0	4 MHz
▶ RTC clock input	Disabled	
System oscillator 0	Enabled	
Clock source	External crystal	
Clock frequency [MHz]	8.0	8 MHz
Clock source settings	1	
Clock source setting 0		
MCG settings		
MCG mode	PEE	
MCG output [MHz]	96.0	96 MHz
MCG external ref. clock [MHz]	8.0	8 MHz
FLL settings		
FLL module	Disabled	
FLL output [MHz]	0.0	0 MHz; FLL is disabled.
PLL 0 settings		
PLL module	Enabled	
PLL output [MHz]	96.0	96 MHz
Initialization priority	interrupts enabled	1
Watchdog disable	yes	
▶ CPU interrupts/resets		
Clock configurations	1	
Clock configuration 0		
Clock source setting	configuration 0	
MCG mode	PEE	
System clocks		
Core clock	48.0	48 MHz
Bus clock	24.0	24 MHz
TPM clock selection	Auto select	PLL/FLL clock
Clock frequency [MHz]	48.0	48 MHz

Component Inspector

- Change to “Advanced” Mode



Click on “Advanced”

Configure AsynchroSerial Component

- Configure the AsynchroSerial component as follows:
 - Baud rate 115200
 - Receiver RxD UART0
 - Transmitter TxD UART0
 - Leave all other settings at their default settings
 - Channel: UART0
 - Parity: none
 - Width: 8 bits
 - Stop bit: 1
 - Receiver: Enabled
 - RxD: TSIO_CH2/...
 - Transmitter: Enabled
 - TxD: TSIO_CH3/...
 - Stop in wait mode: no
 - Idle line mode: starts after start bit
 - Enable in init code: yes

Peripheral Initialization – AsynchroSerial

The screenshot shows the CodeWarrior IDE interface. On the left, the 'CodeWarrior Projects' pane displays the project structure for 'project_1'. The 'Sources' folder contains files like 'Events.c', 'Events.h', 'ProcessorExpert.c', and 'sa_mtb.c'. Below this, the 'Components - project_1' pane shows a tree view of components, including 'OSs', 'Processors', and 'Components'. The 'Components' folder is expanded, showing 'Referenced_Components', 'PTA:Init_GPIO', and several 'BitX:BitIO' components. At the bottom, 'AS1:AsynchroSerial' is highlighted with a red arrow.

On the right, the 'Component Inspector - AS1' pane is open, showing the properties of the selected component. The 'Properties' tab is active, displaying a table of properties:

Name	Value	Details
Component name	AS1	
Channel	UART0	
Serial_LDD	Serial_LDD	Error in the inherited component s...
Interrupt service/event	Disabled	
Settings		
Parity	none	none
Width	8 bits	8 bits
Stop bit	1	1
Receiver	Enabled	
RxD	TSIO_CH2/PTA1/UART0_RX/TPM2...	TSIO_CH2/PTA1/UART0_RX/TPM2...
RxD pin signal		
Transmitter	Enabled	
TxD	TSIO_CH3/PTA2/UART0_TX/TPM2...	TSIO_CH3/PTA2/UART0_TX/TPM2...
TxD pin signal		
Baud rate		Unassigned timing
Break signal	Disabled	
Wakeup condition	Idle line wakeup	
Transmitter output	Not inverted	
Receiver input	Not inverted	
Stop in wait mode	no	
Idle line mode	starts after start bit	
Break generation length	Short	
Initialization		
Enabled in init. code	yes	
Events enabled in init.	yes	

A red arrow points from the 'AS1:AsynchroSerial' component in the 'Components' pane to the 'Select "AsynchroSerial"' text at the bottom of the image.

Peripheral Initialization – AsynchroSerial

The screenshot shows the CodeWarrior IDE with the Component Inspector for AS1. The 'Properties' tab is selected, showing the configuration for the UART0 component. The 'Baud rate' is set to 115200 baud, and the 'Interrupt service/event' is set to 'Disabled'. The 'Settings' section shows 'Parity' as 'none', 'Width' as '8 bits', and 'Stop bit' as '1'. The 'Receiver' and 'Transmitter' sections are both 'Enabled'. The 'Initialization' section shows 'Enabled in init. code' as 'yes' and 'Events enabled in init.' as 'yes'. Red arrows point to the 'Baud rate' and 'Interrupt service/event' settings.

Peripheral Initialization – TimerInt

CodeWarrior Projects

File Name Build

- Documentation
- FLASH
- Generated_Code
- ProcessorExpert.pe
- Project_Headers
- Project_Settings
- Sources
 - Events.c ✓
 - Events.h ✓
 - ProcessorExpert.c ✓
 - sa_mtb.c ✓

Components - project_1

- OSs
- Processors
 - Cpu:MKL46Z256VMC4
- Components
 - Referenced_Components
 - PTA:Init_GPIO
 - Bit1:BitIO
 - Bit2:BitIO
 - Bit3:BitIO
 - Bit4:BitIO
 - TI1:TimerInt
 - AS1:AsynchroSerial

*Component Inspector - TI1

Properties Methods Events

Name	Value	Details
Component name	TI1	
Periodic interrupt source	LPTMR0_CMR	LPTMR0_CMR
Counter	LPTMR0_CNR	LPTMR0_CNR
Interrupt service/event	Enabled	
Interrupt	INT_LPTimer	INT_LPTimer
Interrupt priority	medium priority	2
! Interrupt period		Unassigned timing
Component uses entire timer	no	
Initialization		
Enabled in init. code	yes	
Events enabled in init.	yes	
! Referenced components		

Select "TI1:TimerInt"

Peripheral Initialization – TimerInt

The screenshot shows the CodeWarrior IDE interface. On the left, the 'CodeWarrior Projects' pane displays a file tree with folders like 'Documentation', 'FLASH', 'Generated_Code', 'Project_Headers', 'Project_Settings', and 'Sources'. The 'Sources' folder is expanded, showing files like 'Events.c', 'Events.h', 'ProcessorExpert.c', and 'sa_mtb.c'. Below this, the 'Components - project_1' pane shows a tree structure with 'OSs', 'Processors', and 'Components'. The 'Components' folder is expanded, showing 'Referenced_Components' and a list of components including 'PTA:Init_GPIO', 'Bit1:BitIO', 'Bit2:BitIO', 'Bit3:BitIO', 'Bit4:BitIO', 'LED_TMR:TimerInt', and 'AS1:AsynchroSerial'.

On the right, the '*Component Inspector - LED_TMR' pane is active, showing the 'Properties' tab. It displays a table of properties for the 'LED_TMR' component:

Name	Value	Details
Component name	LED_TMR	
Periodic interrupt source	LPTMR0_CMR	LPTMR0_CMR
Counter	LPTMR0_CNR	LPTMR0_CNR
Interrupt service/event	Enabled	
Interrupt	INT_LPTimer	INT_LPTimer
Interrupt priority	medium priority	2
Interrupt period	.25 Hz	0.250 Hz
Component uses entire timer	no	
Initialization		
Enabled in init. code	yes	
Events enabled in init.	yes	
Referenced components		

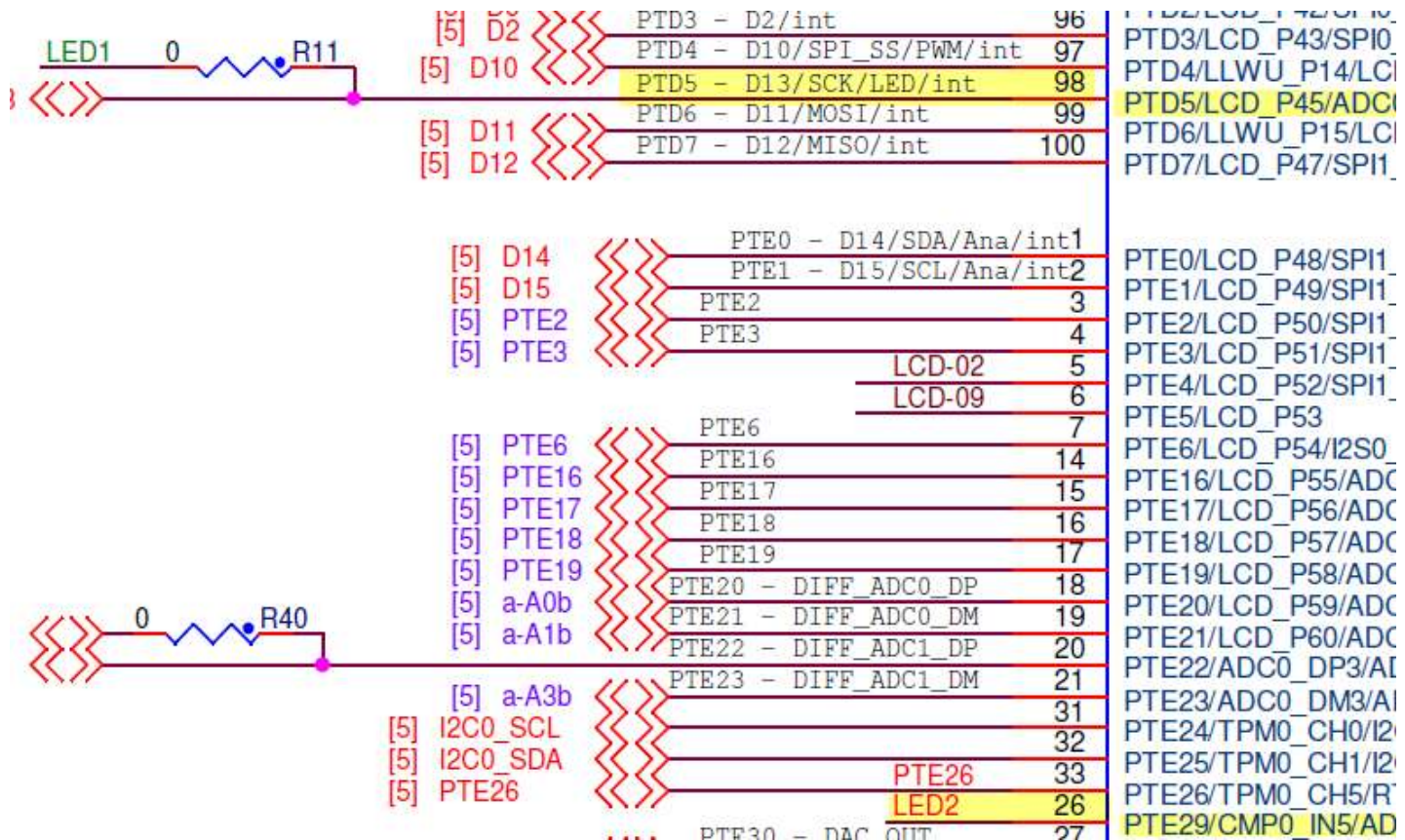
Red arrows point from the 'LED_TMR' value in the 'Component name' row to the 'LED_TMR' component in the 'Components' pane, and from the '.25 Hz' value in the 'Interrupt period' row to the 'LED_TMR:TimerInt' component in the 'Components' pane.

Select
"LED_TMR"

Enter "25Hz"

Hardware Pin Details

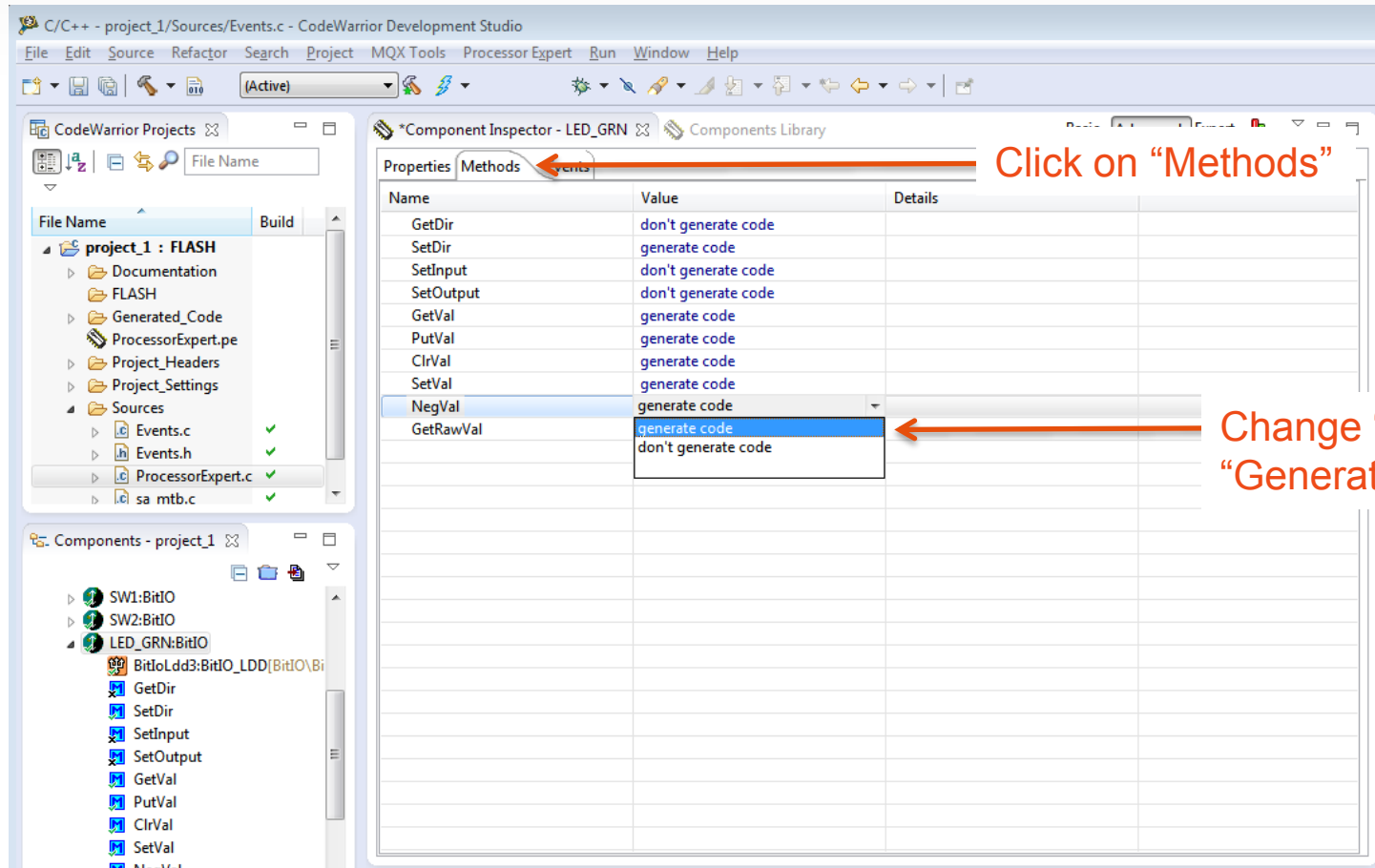
- LED1(green) on PTD5
- LED2 (red) on PTE29



Peripheral Initialization – Bit4:BitIO

The screenshot displays the CodeWarrior IDE interface. On the left, the 'CodeWarrior Projects' pane shows a project structure with folders like 'Documentation', 'FLASH', 'Generated_Code', and 'Sources'. The 'Sources' folder contains files like 'Events.c', 'Events.h', 'ProcessorExpert.pe', and 'sa_mtb.c'. Below this, the 'Components - project_1' pane shows a tree view of components. The 'Components' folder is expanded, showing a list of components including 'Referenced_Components', 'PTA:Init_GPIO', 'Bit1:BitIO', 'Bit2:BitIO', 'Bit3:BitIO', 'Bit4:BitIO' (which is selected and highlighted), 'LED_TMR:TimerInt', and 'AS1:AsynchroSerial'. An orange arrow points from the text 'Select "Bit4:BitIO"' to the 'Bit4:BitIO' component in the list. On the right, the '*Component Inspector - Bit4' pane shows the properties of the selected component. The 'Properties' tab is active, displaying a table with columns 'Name', 'Value', and 'Details'. The table lists various properties such as 'Component name' (Bit4), 'Pin for I/O' (Unassigned peripheral), 'Pin signal' (BitIO_LDD), 'Direction' (Input/Output), 'Initialization' (Output), 'Init. direction' (0), 'Init. value' (yes), 'Safe mode' (yes), and 'Optimization for' (speed).

Peripheral Initialization – Bit4:BitIO



Peripheral Initialization – Bit3:BitIO

CodeWarrior Projects

- Documentation
- FLASH
- Generated_Code
- ProcessorExpert.pe
- Project_Headers
- Project_Settings
- Sources
 - Events.c ✓
 - Events.h ✓
 - ProcessorExpert.c ✓
 - sa_mtb.c ✓

Components - project_1

- OSs
- Processors
 - Cpu:MKL46Z256VMC4
- Components
 - Referenced_Components
 - PTA:Init_GPIO
 - Bit1:BitIO
 - Bit2:BitIO
 - Bit3:BitIO** ←
 - LED_RED:BitIO
 - LED_TMR:TimerInt
 - AS1:AsynchroSerial

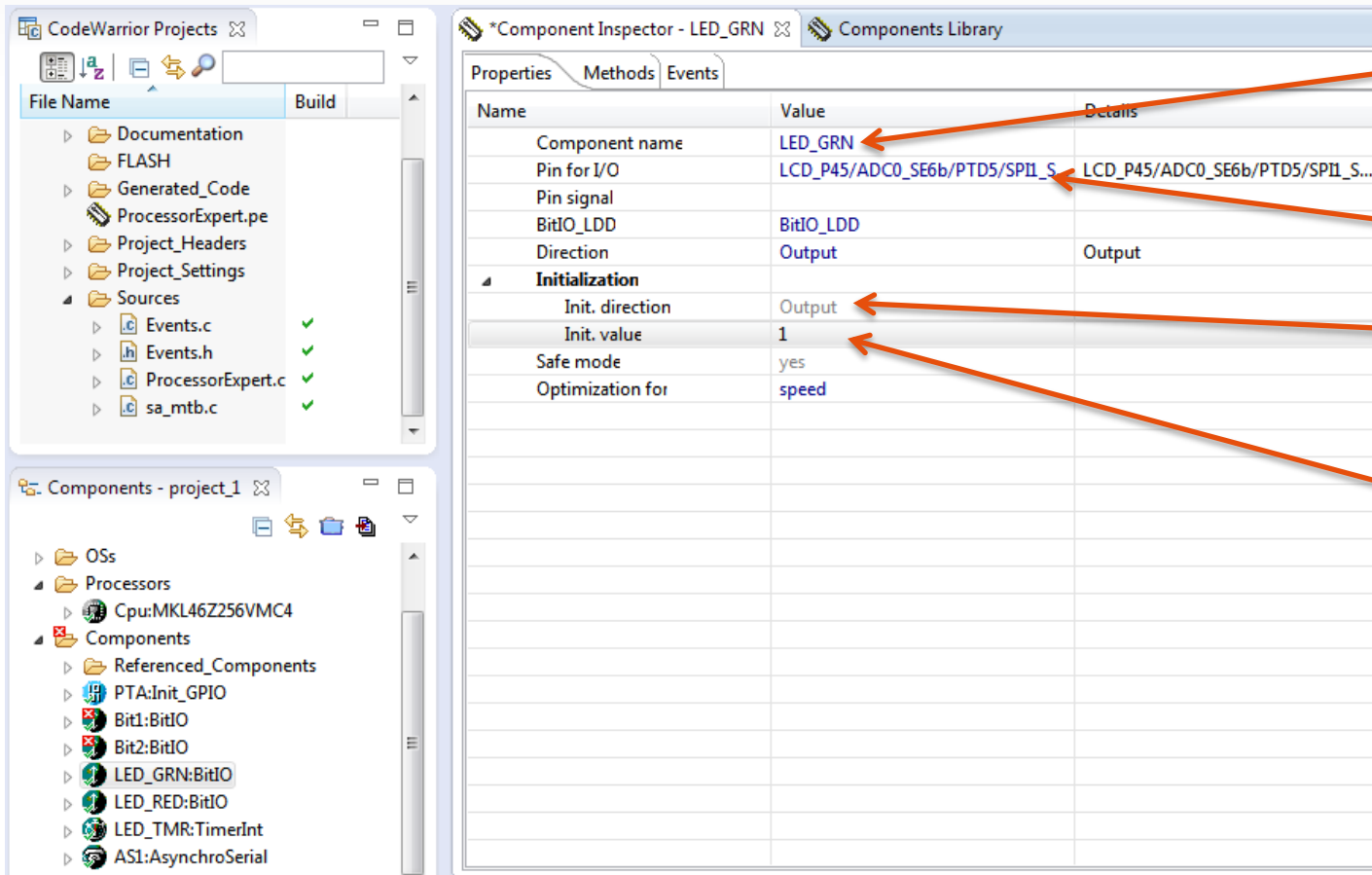
***Component Inspector - Bit3**

Properties | Methods | Events

Name	Value	Details
Component name	Bit3	
Pin for I/O		Unassigned peripheral
Pin signal		
BitIO_LDD	BitIO_LDD	Error in the inherited component s...
Direction	Input/Output	Input/Output
Initialization		
Init. direction	Output	
Init. value	0	
Safe mode	yes	
Optimization for	speed	

Select "Bit3:BitIO"

Peripheral Initialization – Bit3:Bit10



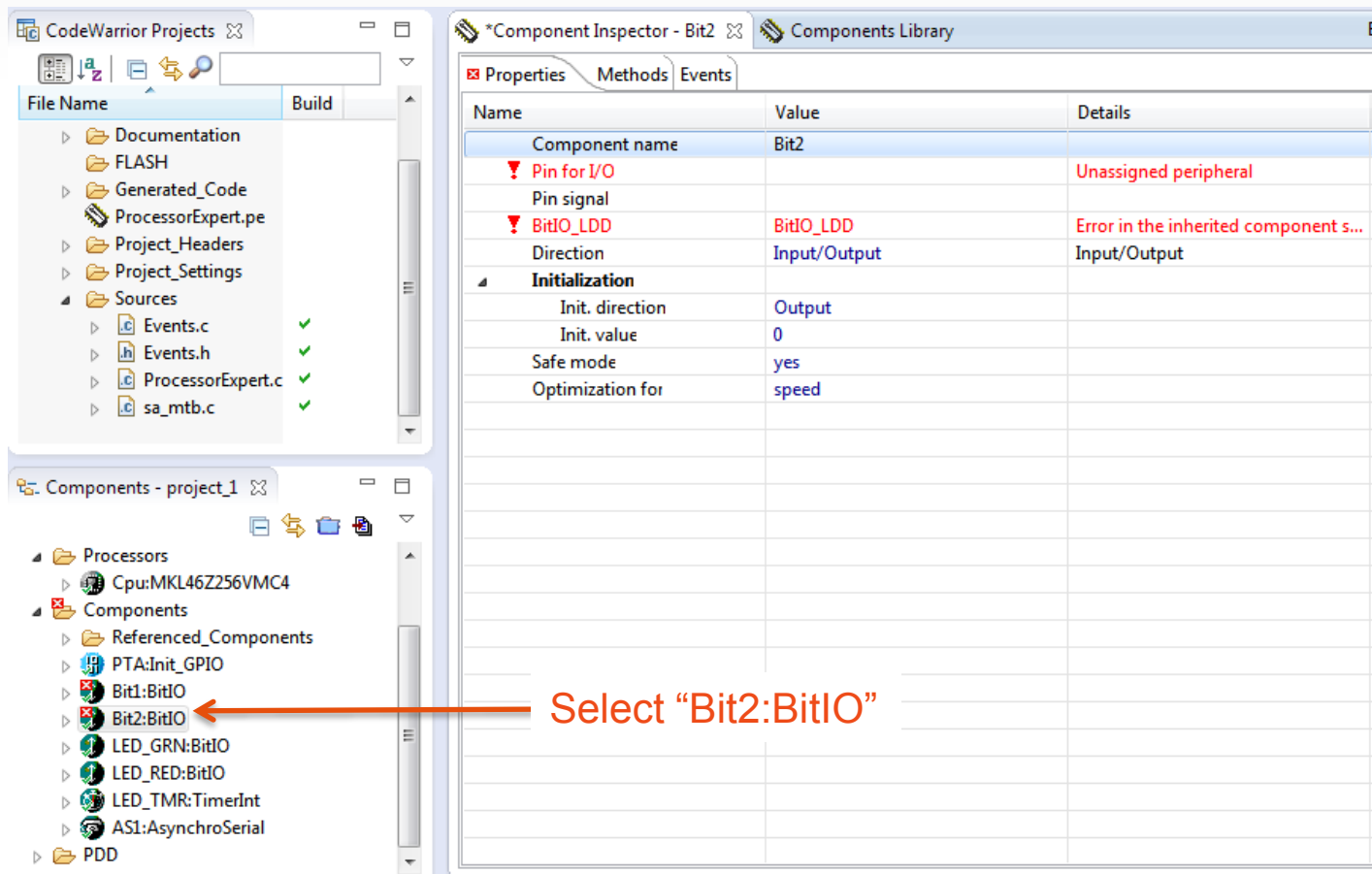
Enter
"LED_GRN"

Select
"LCD_P45/..."

Select "Output"

Enter "1"

Peripheral Initialization – Bit2:BitIO



Peripheral Initialization – Bit2:BitIO

File Name

- Documentation
- FLASH
- Generated_Code
- ProcessorExpert.pe
- Project_Headers
- Project_Settings
- Sources
 - Events.c ✓
 - Events.h ✓
 - ProcessorExpert.c ✓
 - sa_mtb.c ✓

Components - project_1

- Processors
 - Cpu:MKL46Z256VMC4
- Components
 - Referenced_Components
 - PTA:Init_GPIO
 - Bit1:BitIO
 - SW2:BitIO**
 - LED_GRN:BitIO
 - LED_RED:BitIO
 - LED_TMR:TimerInt
 - AS1:AsynchroSerial

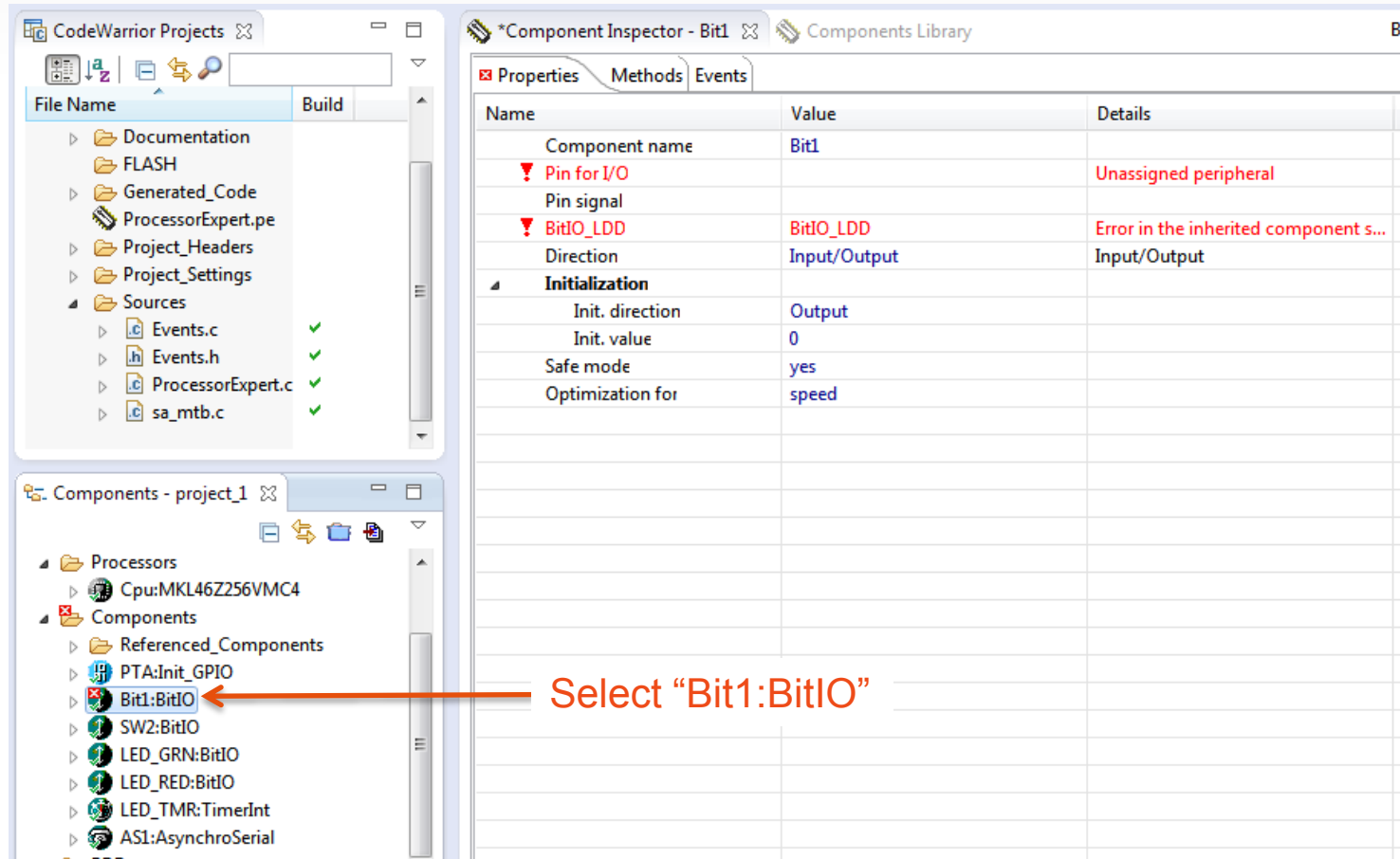
***Component Inspector - SW2**

Name	Value	Details
Component name	SW2	
Pin for I/O	LCD_P32/PTC12/TPM_CLKIN0	LCD_P32/PTC12/TPM_CLKIN0
Pin signal		
BitIO_LDD	BitIO_LDD	
Direction	Input	Input
Initialization		
Init. direction	Input	
Init. value	0	
Safe mode	yes	
Optimization for	speed	

Annotations:

- Enter "SW2"
- Select "LCD_P32/..."
- Select "Input"

Peripheral Initialization – Bit1:BitIO



Peripheral Initialization – Bit1:BitIO

The screenshot displays the CodeWarrior IDE interface. On the left, the 'CodeWarrior Projects' pane shows a project structure with folders like 'Documentation', 'FLASH', 'Generated_Code', and 'Sources'. The 'Sources' folder contains files: 'Events.c', 'Events.h', 'ProcessorExpert.c', and 'sa_mtb.c'. Below this, the 'Components - project_1' pane shows a tree view with 'Processors' (Cpu:MKL46Z256VMC4) and 'Components' (Referenced_Components, PTA:Init_GPIO, SW1:BitIO, SW2:BitIO, LED_GRN:BitIO, LED_RED:BitIO, LED_TMR:TimerInt, AS1:AsynchroSerial). The main area on the right is the '*Component Inspector - SW1' window, which has tabs for 'Properties', 'Methods', and 'Events'. The 'Properties' tab is active, showing a table of component properties:

Name	Value	Details
Component name	SW1	
Pin for I/O	LCD_P23/PTC3/LLWU_P7/UART1_...	LCD_P23/PTC3/LLWU_P7/UART1_...
Pin signal		
BitIO_LDD	BitIO_LDD	
Direction	Input/Output	Input/Output
Initialization		
Init. direction	Input	
Init. value	0	
Safe mode	yes	Warning: This setting is useless for...
Optimization for	speed	

Enter "SW1"

Select
“LCD_P23/...”

Select "Input"

Peripheral Initialization – Init_GPIO

The screenshot displays the CodeWarrior IDE interface. On the left, the 'CodeWarrior Projects' pane shows a project structure with folders like 'Documentation', 'FLASH', 'Generated_Code', 'ProcessorExpert.pe', 'Project_Headers', 'Project_Settings', and 'Sources'. The 'Sources' folder contains files 'Events.c', 'Events.h', 'ProcessorExpert.c', and 'sa_mtb.c'. Below this, the 'Components - project_1' pane shows a tree view of components, including 'Processors' (Cpu:MKL46Z256VMC4) and 'Components' (Referenced_Components, PTA:Init_GPIO, SW1:BitIO, SW2:BitIO, LED_GRN:BitIO, LED_RED:BitIO, LED_TMR:TimerInt, AS1:AsynchroSerial). A red arrow points to 'PTA:Init_GPIO' with the text 'Select "PTA:Init_GPIO"'. On the right, the '*Component Inspector - PTA' pane shows the 'Properties' tab with a table of component details:

Name	Value	Details
Component name	PTA	
Device	PTA	PTA
Settings		
Pins		
Interrupts		
Initialization		
Call Init method	yes	

Peripheral Initialization – Init_GPIO

Expand "Pins"

Name	Value	Details
Component name	PTC	
Device	PTC	PTC
Settings		
Pins		
Pin 0	Disabled	LCD_P20/ADC0_SE14/TS10_CH13/P...
Pin 1	Disabled	LCD_P21/ADC0_SE15/TS10_CH14/P...
Pin 2	Disabled	LCD_P22/ADC0_SE11/TS10_CH15/P...
Pin 3	Disabled	LCD_P23/PTC3/LLWU_P7/UART1_...
Pin 4	Disabled	LCD_P24/PTC4/LLWU_P8/SPI0_PC...
Pin 5	Disabled	LCD_P25/PTC5/LLWU_P9/SPI0_SC...
Pin 6	Disabled	LCD_P26/CMP0_IN0/PTC6/LLWU_...
Pin 7	Disabled	LCD_P27/CMP0_IN1/PTC7/SPI0_M...
Pin 8	Disabled	LCD_P28/CMP0_IN2/PTC8/I2C0_S...
Pin 9	Disabled	LCD_P29/CMP0_IN3/PTC9/I2C0_S...
Pin 10	Disabled	LCD_P30/PTC10/I2C1_SCL/I2S0_RX...
Pin 11	Disabled	LCD_P31/PTC11/I2C1_SDA/I2S0_R...
Pin 12	Disabled	LCD_P32/PTC12/TPM_CLKIN0
Pin 13	Disabled	LCD_P33/PTC13/TPM_CLKIN1
Pin 16	Disabled	LCD_P36/PTC16
Pin 17	Disabled	LCD_P37/PTC17
Pin 18	Disabled	LCD_P38/PTC18
Pin 20	Disabled	VLL2/LCD_P4/PTC20
Pin 21	Disabled	VLL1/LCD_P5/PTC21
Pin 22	Disabled	VCAP2/LCD_P6/PTC22
Pin 23	Disabled	VCAP1/LCD_P39/PTC23
Interrupts		

Select "PTC"

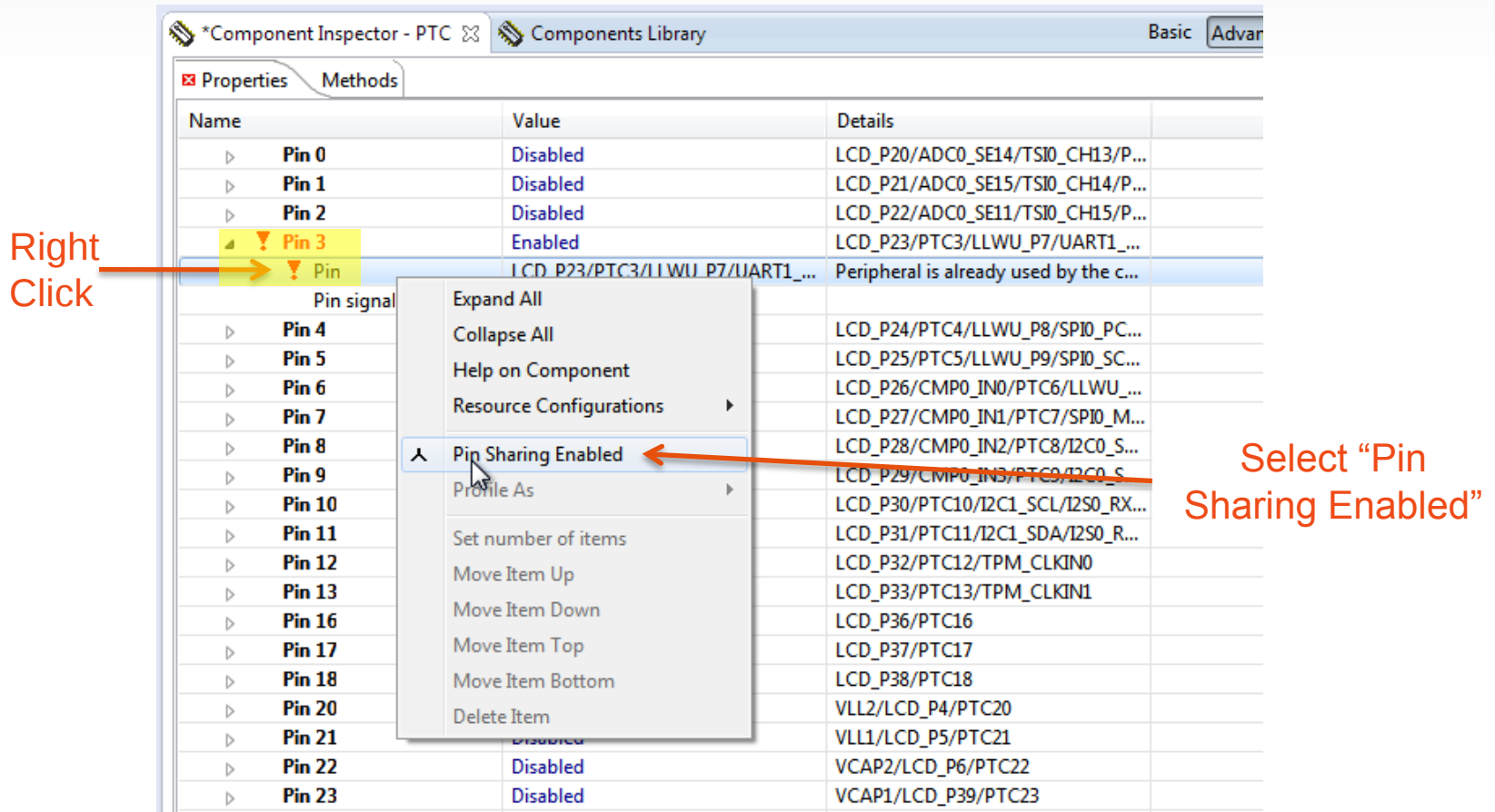
Peripheral Initialization – PTC:Init_GPIO

The screenshot shows the PTC Component Inspector window. The 'Properties' tab is active, displaying a table of pins. Pin 3 is selected and its value is 'Enabled'. An orange arrow points to Pin 3 with the text 'Select "Pin 3"'. Another orange arrow points to the 'Enabled' value with the text 'Select "Enabled"'. A yellow tooltip is visible over Pin 3, providing details about the pin's function and a warning that the peripheral is already used by another component.

Name	Value	Details
Pin 0	Disabled	LCD_P20/ADC0_SE14/TS10_CH13/P...
Pin 1	Disabled	LCD_P21/ADC0_SE15/TS10_CH14/P...
Pin 2	Disabled	LCD_P22/ADC0_SE11/TS10_CH15/P...
Pin 3	Enabled	LCD_P23/PTC3/LLWU_P7/UART1_...
Pin 4	Disabled	LCD_P23/PTC3/LLWU_P7/UART1_...
Pin 5	Disabled	LCD_P23/PTC3/LLWU_P7/UART1_...
Pin 6	Disabled	LCD_P23/PTC3/LLWU_P7/UART1_...
Pin 7	Disabled	LCD_P23/PTC3/LLWU_P7/UART1_...
Pin 8	Disabled	LCD_P23/PTC3/LLWU_P7/UART1_...
Pin 9	Disabled	LCD_P23/PTC3/LLWU_P7/UART1_...
Pin 10	Disabled	LCD_P30/PTC10/I2C1_SCL/I2S0_RX...
Pin 11	Disabled	LCD_P31/PTC11/I2C1_SDA/I2S0_R...
Pin 12	Disabled	LCD_P32/PTC12/TPM_CLKIN0
Pin 13	Disabled	LCD_P33/PTC13/TPM_CLKIN1
Pin 16	Disabled	LCD_P36/PTC16
Pin 17	Disabled	LCD_P37/PTC17
Pin 18	Disabled	LCD_P38/PTC18
Pin 20	Disabled	VLL2/LCD_P4/PTC20
Pin 21	Disabled	VLL1/LCD_P5/PTC21
Pin 22	Disabled	VCAP2/LCD_P6/PTC22
Pin 23	Disabled	VCAP1/LCD_P39/PTC23
Interrupts		
Initialization		
Call Init method	yes	

Pin of the GPIO port (for information only).
This item modifies MUX[10:8] bit field in the PORTx_PCR3 register.
Peripheral description: LCD driver pin 23; General purpose IO, Port C, bit 3; Low-leakage wake-up pin; UART1 receive data; TPM0 channel 2; FlexBus Clock Output; I2S0 TX Bit Clock; LCD fault pin 23
Pin number on the processor package: B8.
Version specific item: Settings supported only if the selected port contains the pin.
ERROR: Peripheral is already used by the component BitIoLdd1 [see item: Pin for I/O]

Peripheral Initialization – PTC:Init_GPIO



Peripheral Initialization – PTC:Init_GPIO

Pin sharing
is enabled

*Component Inspector - PTC		Components Library	Basic
Properties		Methods	
Name	Value	Details	
Pin 0	Disabled	LCD_P20/ADC0_SE14/TS10_CH13/P...	
Pin 1	Disabled	LCD_P21/ADC0_SE15/TS10_CH14/P...	
Pin 2	Disabled	LCD_P22/ADC0_SE11/TS10_CH15/P...	
Pin 3	Enabled	LCD_P23/PTC3/LLWU_P7/UART1_...	
Pin	LCD_P23/PTC3/LLWU_P7/UART1_...	Warning: The Pin 3 settings (Settin...	
Pin signal			
Pin 4	Disabled	LCD_P24/PTC4/LLWU_P8/SPI0_PC...	
Pin 5	Disabled	LCD_P25/PTC5/LLWU_P9/SPI0_SC...	
Pin 6	Disabled	LCD_P26/CMP0_IN0/PTC6/LLWU_...	
Pin 7	Disabled	LCD_P27/CMP0_IN1/PTC7/LLWU_...	

Peripheral Initialization – PTC:Init_GPIO

CodeWarrior Projects

File Name Build

- Documentation
- FLASH
- Generated_Code
- ProcessorExpert.pe
- Project_Headers
- Project_Settings
- Sources
 - Events.c
 - Events.h
 - ProcessorExpert.c
 - sa_mtb.c

Components - project_1

- Processors
 - Cpu:MKL43250M100
- Components
 - Reference
 - PTC:Init
 - SW1:BitIO
 - SW2:BitIO
 - LED_GRN:BitIO
 - LED_RED:BitIO
 - LED_TMR:TimerInt
 - AS1:AsynchroSerial
 - PDD

Expand "Pin 12"

*Component Inspector - PTC

Properties Methods

Name	Value	Details
Component name	PTC	
Device	PTC	PTC
Settings		
Pins		
Pin 0	Disabled	LCD_P20/ADC0_SE14/TSIO_CH13/P...
Pin 1	Disabled	LCD_P21/ADC0_SE15/TSIO_CH14/P...
Pin 2	Disabled	LCD_P22/ADC0_SE11/TSIO_CH15/P...
Pin 3	Enabled	LCD_P23/PTC3/LLWU_P7/UART1_...
Pin	LCD_P23/PTC3/LLWU_P7/UART1_...	Warning: The Pin 3 settings (Settin...
Pin signal		
Pin 4	Disabled	LCD_P24/PTC4/LLWU_P8/SPIO_PC...
Pin 5	Disabled	LCD_P25/PTC5/LLWU_P9/SPIO_SC...
Pin 6	Disabled	LCD_P26/CMP0_IN0/PTC6/LLWU_...
Pin 7	Disabled	LCD_P27/CMP0_IN1/PTC7/SPIO_M...
Pin 8	Disabled	LCD_P28/CMP0_IN2/PTC8/I2C0_S...
Pin 9	Disabled	LCD_P29/CMP0_IN3/PTC9/I2C0_S...
Pin 10	Disabled	LCD_P30/PTC10/I2C1_SCL/I2S0_RX...
Pin 11	Disabled	LCD_P31/PTC11/I2C1_SDA/I2S0_R...
Pin 12	Disabled	LCD_P32/PTC12/TPM_CLKIN0
Pin	LCD_P32/PTC12/TPM_CLKIN0	Peripheral is already used by the c...
Pin signal		
Pin 13	Disabled	LCD_P33/PTC13/TPM_CLKIN1
Pin 16	Disabled	LCD_P36/PTC16
Pin 17	Disabled	LCD_P37/PTC17
Pin 18	Disabled	LCD_P38/PTC18
Pin 20	Disabled	VLL2/LCD_P4/PTC20
Pin 21	Disabled	VLL1/LCD_P5/PTC21

Peripheral Initialization – PTC:Init_GPIO

The screenshot shows the 'Component Inspector - PTC' window with the 'Properties' tab selected. The table below lists the pins and their settings. Pin 12 is highlighted, and a red arrow points to a yellow warning row.

Name	Value	Details
Component name	PTC	
Device	PTC	PTC
Settings		
Pins		
Pin 0	Disabled	LCD_P20/ADC0_SE14/TS10_CH13/P...
Pin 1	Disabled	LCD_P21/ADC0_SE15/TS10_CH14/P...
Pin 2	Disabled	LCD_P22/ADC0_SE11/TS10_CH15/P...
Pin 3	Enabled	LCD_P23/PTC3/LLWU_P7/UART1_...
Pin	LCD_P23/PTC3/LLWU_P7/UART1_...	Warning: The Pin 3 settings (Settin...
Pin signal		
Pin 4	Disabled	LCD_P24/PTC4/LLWU_P8/SPIO_PC...
Pin 5	Disabled	LCD_P25/PTC5/LLWU_P9/SPIO_SC...
Pin 6	Disabled	LCD_P26/CMP0_IN0/PTC6/LLWU_...
Pin 7	Disabled	LCD_P27/CMP0_IN1/PTC7/SPIO_M...
Pin 8	Disabled	LCD_P28/CMP0_IN2/PTC8/I2C0_S...
Pin 9	Disabled	LCD_P29/CMP0_IN3/PTC9/I2C0_S...
Pin 10	Disabled	LCD_P30/PTC10/I2C1_SCL/I2S0_RX...
Pin 11	Disabled	LCD_P31/PTC11/I2C1_SDA/I2S0_R...
Pin 12	Enabled	LCD_P32/PTC12/TPM_CLKIN0
Pin	LCD_P32/PTC12/TPM_CLKIN0	Peripheral is already used by the c...
Pin signal		
Pin 13	Disat	Pin of the GPIO port (for information only).
Pin 16	Disat	This item modifies MUX[10:8] bit field in the PORTx_PCR12 register.
Pin 17	Disat	Peripheral description: LCD driver pin 32;General purpose IO, Port C, bit 12;TPM External Clock Input 0;LCD faul
Pin 18	Disat	pin 32
Pin 20	Disat	Pin number on the processor package: B6.
Pin 21	Disat	Version specific item: Settings supported only if the selected port contains the pin.
		ERROR: Peripheral is already used by the component BitIoLdd5 [see item: Pin for I/O]

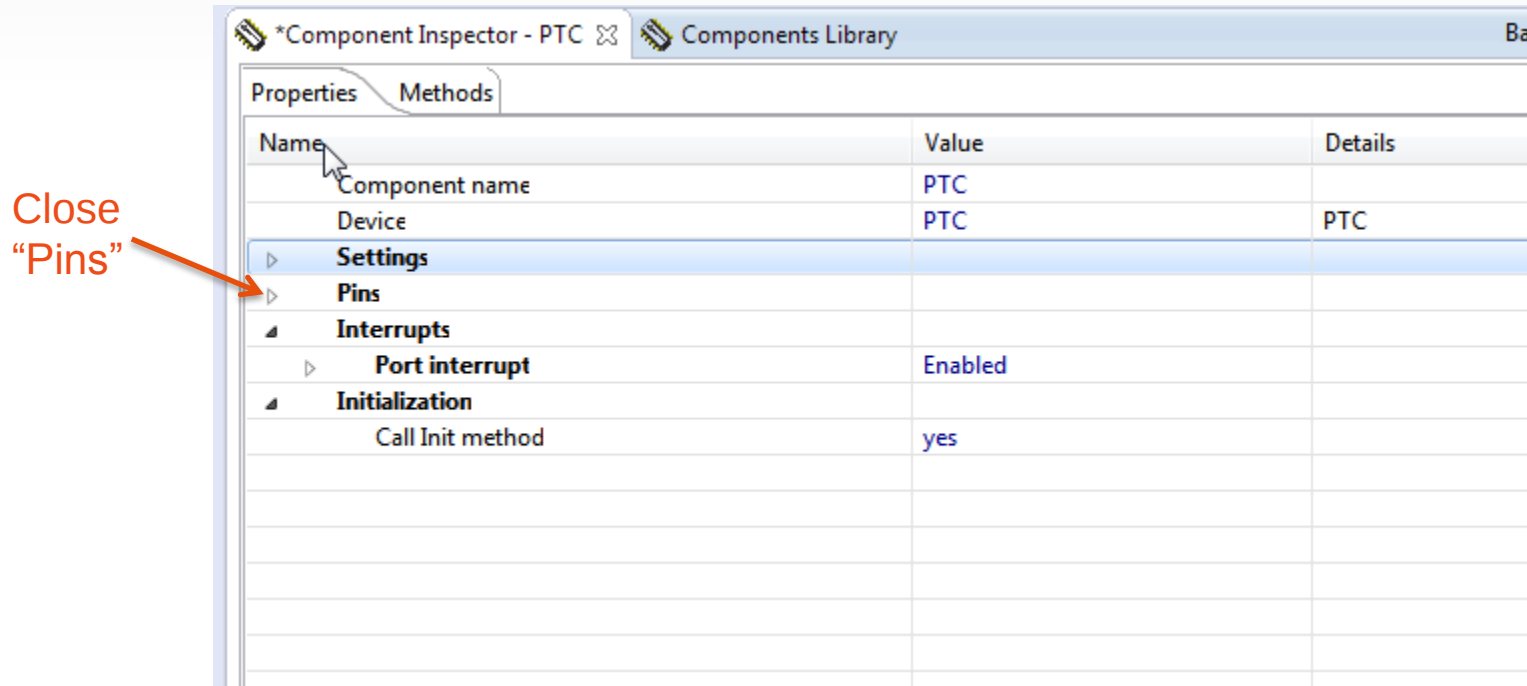
Notice error message. This pin must be “shared”

Peripheral Initialization – PTC:Init_GPIO

The screenshot shows the 'Component Inspector - PTC' window. The 'Properties' tab is active, displaying a table of pin configurations. The table has three columns: 'Name', 'Value', and 'Details'. The 'Pin 12' row is highlighted, and a context menu is open over it. The menu options are: 'Expand All', 'Collapse All', 'Help on Component', 'Resource Configurations', 'Pin Sharing Enabled', and 'Profile As...'. The 'Pin Sharing Enabled' option is selected, indicated by a red arrow. Another red arrow points to the 'Pin 12' row in the table.

Name	Value	Details
Component name	PTC	
Device	PTC	PTC
Settings		
Pin 0	Disabled	LCD_P20/ADC0_SE14/TS10_CH13/P...
Pin 1	Disabled	LCD_P21/ADC0_SE15/TS10_CH14/P...
Pin 2	Disabled	LCD_P22/ADC0_SE11/TS10_CH15/P...
Pin 3	Enabled	LCD_P23/PTC3/LLWU_P7/UART1_...
Pin 4	Disabled	LCD_P24/PTC4/LLWU_P8/SPI0_PC...
Pin 5	Disabled	LCD_P25/PTC5/LLWU_P9/SPI0_SC...
Pin 6	Disabled	LCD_P26/CMP0_IN0/PTC6/LLWU_...
Pin 7	Disabled	LCD_P27/CMP0_IN1/PTC7/SPI0_M...
Pin 8	Disabled	LCD_P28/CMP0_IN2/PTC8/I2C0_S...
Pin 9	Disabled	LCD_P29/CMP0_IN3/PTC9/I2C0_S...
Pin 10	Disabled	LCD_P30/PTC10/I2C1_SCL/I2S0_RX...
Pin 11	Disabled	LCD_P31/PTC11/I2C1_SDA/I2S0_R...
Pin 12	Enabled	LCD_P32/PTC12/TPM_CLKIN0
Pin 13		Peripheral is already used by the c...
Pin 16		LCD_P33/PTC13/TPM_CLKIN1
Pin 17		LCD_P36/PTC16
Pin 18		LCD_P37/PTC17
Pin 20		LCD_P38/PTC18
Pin 21		VLL2/LCD_P4/PTC20
		VLL1/LCD_P5/PTC21

Peripheral Initialization – PTC:Init_GPIO



Peripheral Initialization – PTC:Init_GPIO

Expand "Setting"

Expand "Pin 3"

Name	Value	Details
Component name	PTC	
Device	PTC	PTC
Settings		
Pin 0	Do not initialize	LCD_P20/ADC0_SE1
Pin 1	Do not initialize	LCD_P21/ADC0_SE1
Pin 2	Do not initialize	LCD_P22/ADC0_SE1
Pin 3	Do not initialize	LCD_P23/PTC3/LLV
Pin direction	Input	
Output value	No initialization	
Pull resistor	Enabled	
Pull selection	Pull Up	
Slew rate	No initialization	
Interrupt/DMA request	No initialization	
Pin 4	Do not initialize	LCD_P24/PTC4/LLV
Pin 5	Do not initialize	LCD_P25/PTC5/LLV
Pin 6	Do not initialize	LCD_P26/CMP0_INC
Pin 7	Do not initialize	LCD_P27/CMP0_IN1
Pin 8	Do not initialize	LCD_P28/CMP0_IN1

Peripheral Initialization – PTC:Init_GPIO

The screenshot shows the Proteus Component Inspector window. The 'Properties' tab is active, displaying a table of component settings. The 'Pin 3' row is selected, and its dropdown menu is open, showing 'Initialize' and 'Do not initialize' options. An orange arrow points to the dropdown arrow with the text 'Click Down Arrow'.

Name	Value	Details
Component name	PTC	
Device	PTC	PTC
Settings		
Pin 0	Do not initialize	20/ADC0_SE1
Pin 1	Do not initialize	21/ADC0_SE1
Pin 2	Do not initialize	LCD_P22/ADC0_SE1
Pin 3	Do not initialize	LCD_P23/PTC3/LLW
Pin direction	Initialize	
Output value	Do not initialize	
Pull resistor		
Pull selection	Pull Up	
Slew rate	No initialization	
Interrupt/DMA request	No initialization	

Peripheral Initialization – PTC:Init_GPIO

*Component Inspector - PTC Components Library

Properties Methods

Name	Value	Details
Component name	PTC	
Device	PTC	PTC
Settings		
Pin 0	Do not initialize	_P20/ADC0_SE14,
Pin 1	Do not initialize	LCD_P21/ADC0_SE15,
Pin 2	Do not initialize	LCD_P22/ADC0_SE11,
Pin 3	Initialize	LCD_P23/PTC3/LLWU
Pin direction	Initialize	
Output value	Do not initialize	
Pull resistor		
Pull selection	Pull Up	
Slew rate	No initialization	
Interrupt/DMA request	No initialization	

Select "Initialize"

Peripheral Initialization – PTC:Init_GPIO

The screenshot shows the PTC Component Inspector with the 'Settings' section expanded. The 'Pin 3' row is selected, and its configuration is displayed in the table below. Red arrows point to the 'Input', 'Enabled', and 'Pull Up' settings.

Name	Value	Details
Component name	PTC	
Device	PTC	PTC
Settings		
Pin 0	Do not initialize	LCD_P20/ADC0_SE14
Pin 1	Do not initialize	LCD_P21/ADC0_SE15
Pin 2	Do not initialize	LCD_P22/ADC0_SE16
Pin 3	Initialize	LCD_P23/PTC3/LLW
Pin direction	Input	
Output value	No initialization	
Pull resistor	Enabled	
Pull selection	Pull Up	
Slew rate	No initialization	
Interrupt/DMA request	No initialization	
Pin 4	Do not initialize	LCD_P24/PTC4/LLW
Pin 5	Do not initialize	LCD_P25/PTC5/LLW

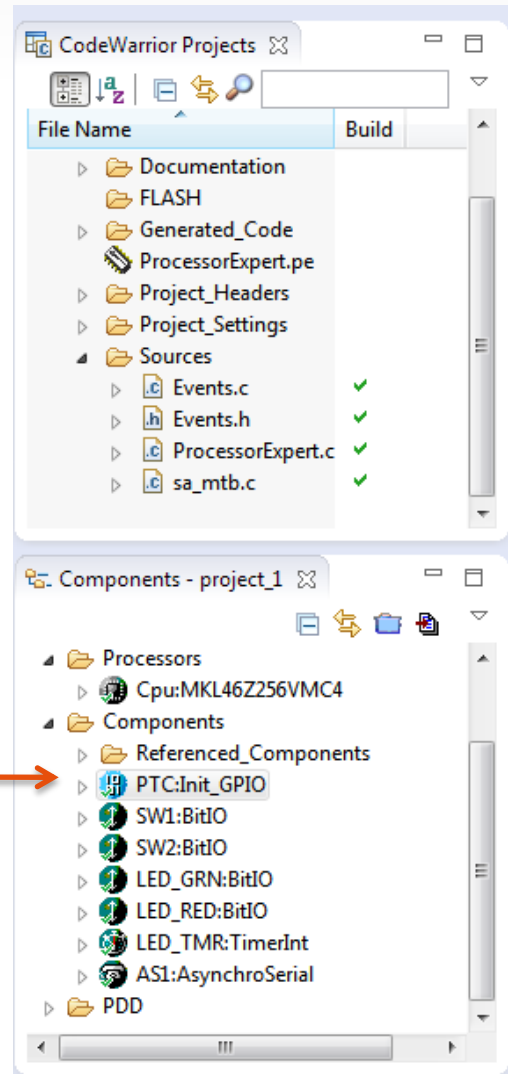
Select "Input"

Select “Enabled”

Select “Pull Up”

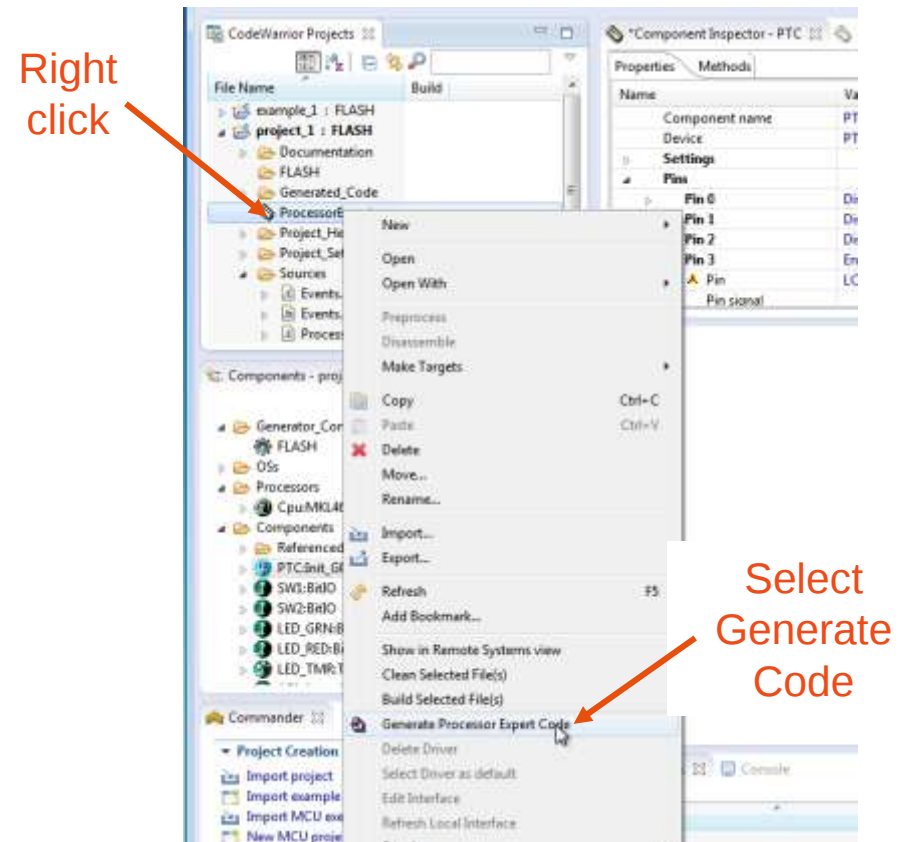
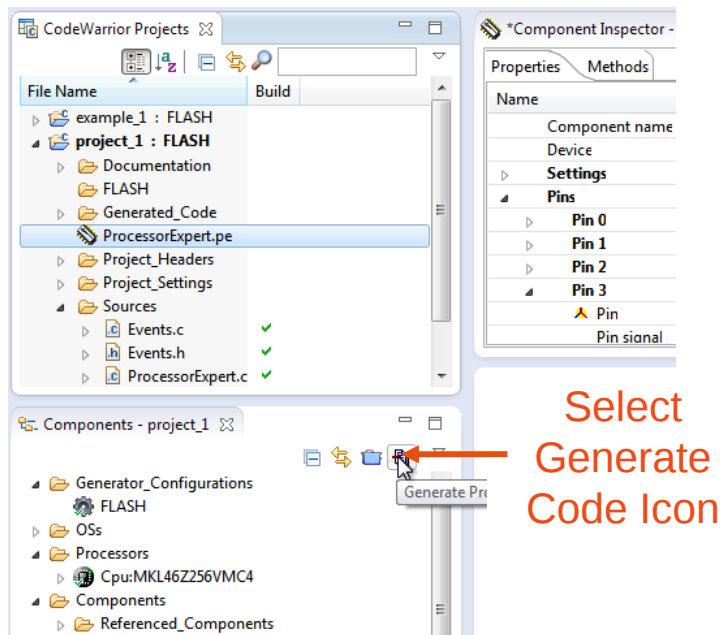
Peripheral Initialization

No Errors



Generate Code

- Now that the components have been setup, its time to generate code. This can be done in one of two ways...



Code is being generated

The screenshot displays the CodeWarrior IDE interface. In the foreground, a 'Generating code' dialog box is open, showing a progress bar and the text 'Generating files - preparing for generation'. Below the progress bar is a checkbox labeled 'Always run in background'. At the bottom of the dialog are three buttons: 'Run in Background', 'Cancel', and 'Details >>'. In the background, the 'Component Inspector - Cpu' window is visible, showing the 'Properties' tab. This tab contains a table with the following data:

Name	Value	Details
Component name	Cpu	
CPU type	MKL46Z256VMC4	
▲ Clock settings		
▲ Internal oscillator		
Slow internal reference clock	32.768	32.768 kHz
Initialize slow trim value	no	
Fast internal reference clock	4.0	4 MHz
Initialize fast trim value	no	
▶ RTC clock input	Disabled	

Using PEx Capability

Minimize Processor Expert View

Expand Project_1

The screenshot displays the CodeWarrior Development Studio environment. The top menu bar includes File, Source, Refactor, Search, Project, MQX Tools, Processor Expert, Run, Window, and Help. The main workspace is divided into several panes. On the left, the 'ModelWarrior Projects' pane shows a project named 'project_1 : FLASH'. Below it, the 'Components - project_1' pane lists various components like SW1-BidIO, SW2-BidIO, LED_GRN-BidIO, LED_RED-BidIO, LED_TMR-TimerInt, AS1-AsynchroSerial, and PDD. The 'Commander' pane on the left contains sections for Project Creation, Build/Debug, and Settings. The central 'Component Inspector - PTC' pane shows the 'Properties' tab for a component, listing various settings like Pin direction, Output value, Pull resistor, Pull selection, Slew rate, Interrupt/DMA request, and Pin 4 through Pin 13. The bottom pane shows the 'Problems' and 'Console' tabs, both of which are empty.

Using PEx Capability

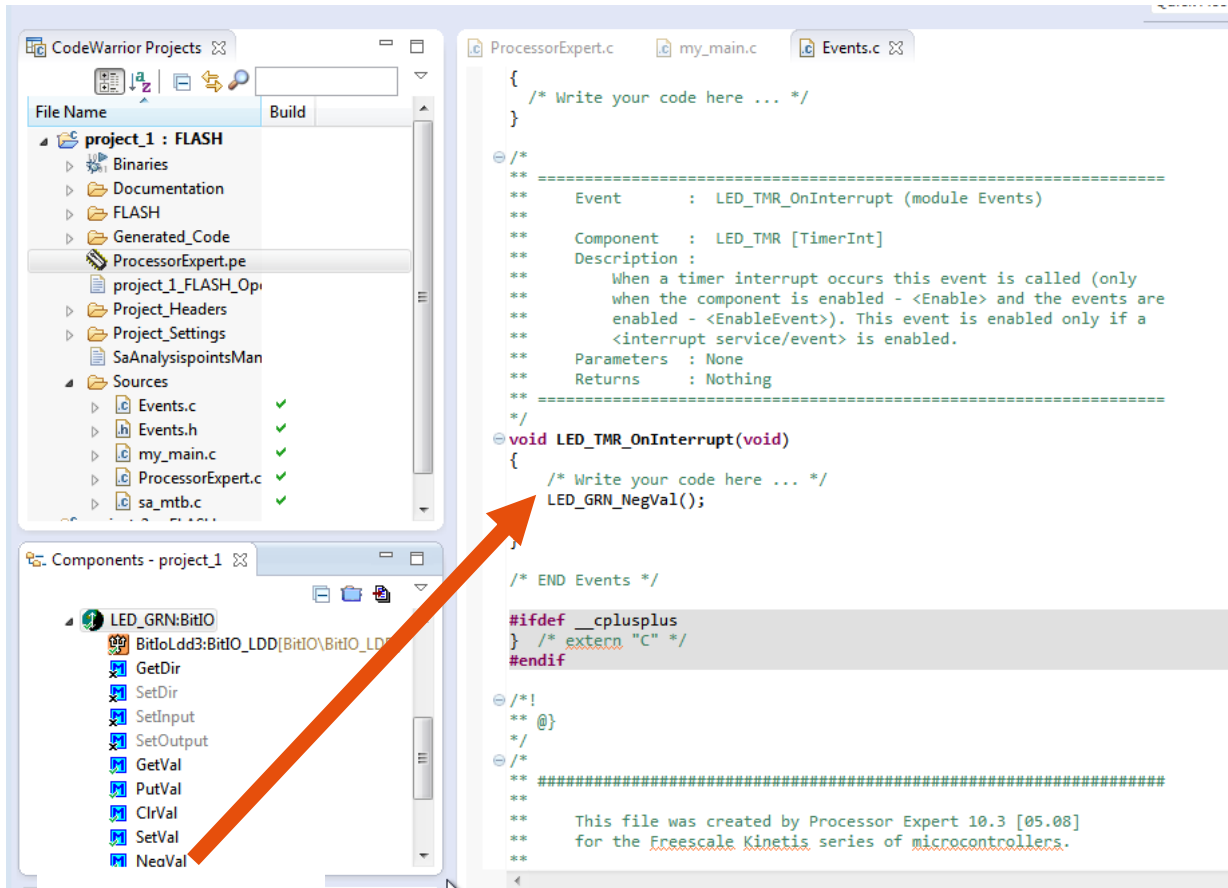
Double Click "Events.c"

Opens "Events.c"

Scroll down to "LED_TMR_OnInterrupt"

Using PEx Capability

- Add code to the Timer Interrupt - LED_TMR_OnInterrupt()



Drag & Drop “NegVal”

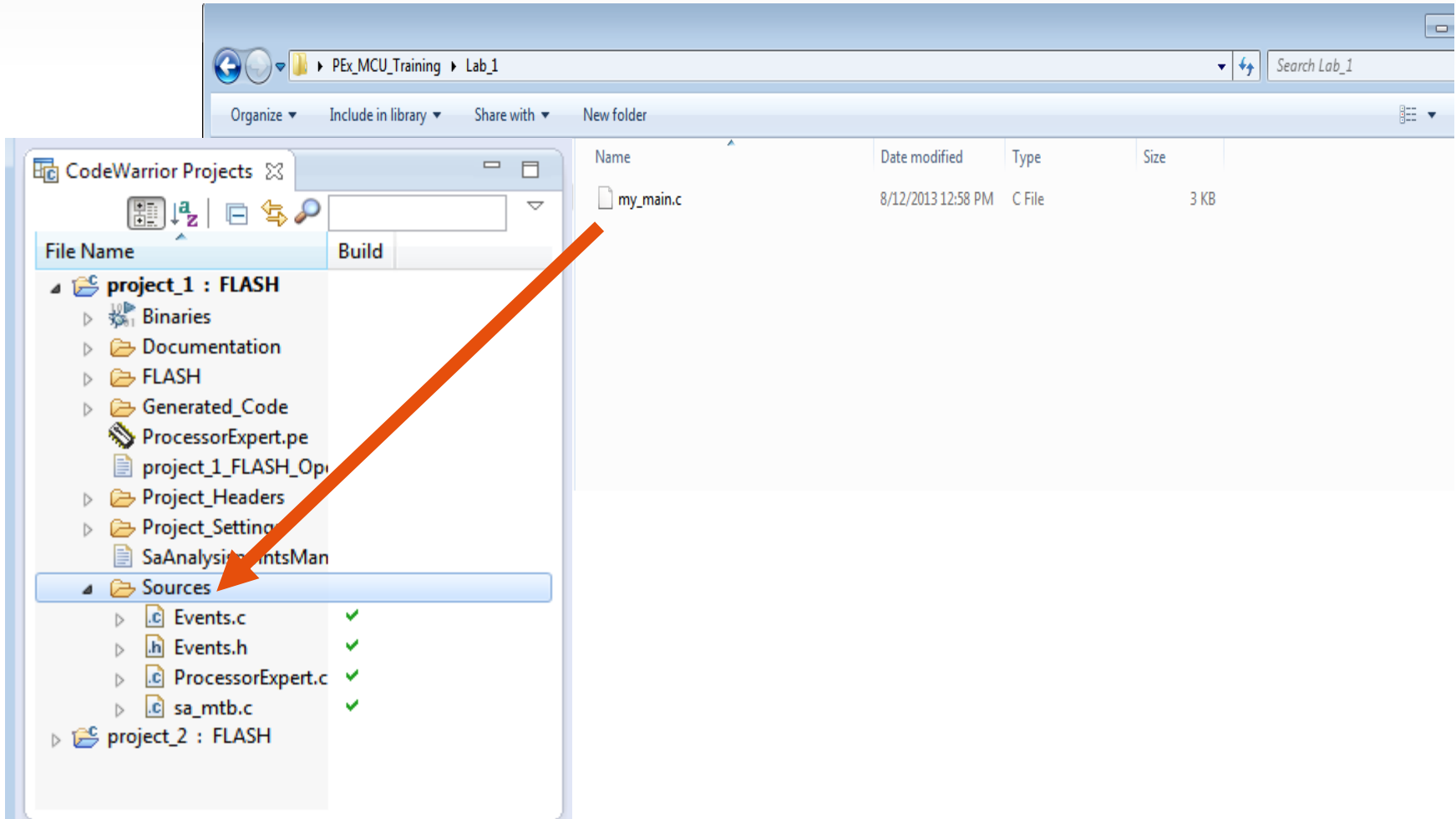
Check Point: Create a New Project to Blink an LED

- 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
- ← **Next up!**

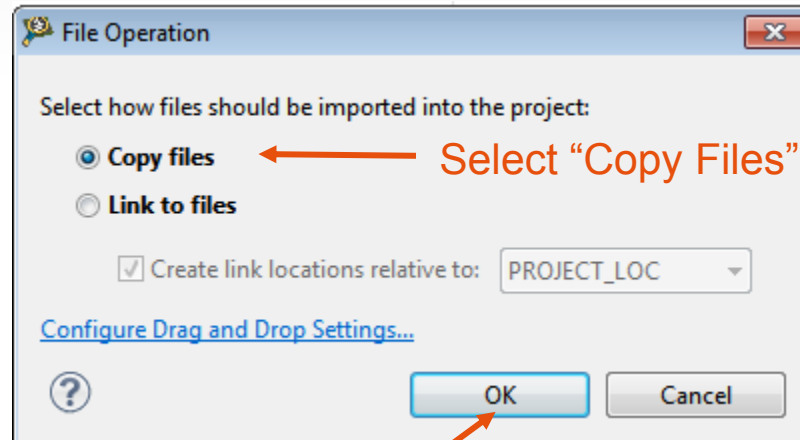
Next up!



Drag my_main.c to Sources Folder



Drag & Drop



Click “OK”

Open “my_main.c”

The screenshot shows the CodeWarrior IDE interface. On the left, the 'CodeWarrior Projects' pane displays a tree view of 'project_1 : FLASH'. Under the 'Sources' folder, 'my_main.c' is listed with a green checkmark. An orange arrow points from the text 'Double click “my_main.c”' to this file. Below the project pane is the 'Components - project_1' pane, showing a tree view of 'Processors' and 'Components'. At the bottom left is the 'Commander' pane with options for 'Project Creation' and 'Settings'. On the right, the 'Component Inspector - LED_TMR' pane shows a table of properties for the 'LED_TMR' component. Below this, the 'ProcessorExpert.c' and 'my_main.c' tabs are visible, with 'my_main.c' selected, showing the source code for the file.

Double click “my_main.c”

Name	Value	Details
Periodic interrupt source	LPTMR0_CMRR	LPTMR0_CMRR
Interrupt service/event	Enabled	
Interrupt priority	medium priority	2
Interrupt period	250 ms	250 ms
Initialization		
Enabled in init. code	yes	

```

/* my_main.c */

/* Includes needed for the high level devices to work */
#include "SW1.h"
#include "BitIoLdd1.h"
#include "SW2.h"
#include "BitIoLdd5.h"
#include "LED_GRN.h"
#include "BitIoLdd3.h"
#include "LED_RED.h"
#include "BitIoLdd4.h"
#include "LED_TMR.h"
#include "TimerIntLdd1.h"
#include "TU1.h"
#include "AS1.h"
#include "ASerialLdd1.h"
  
```

Minimize Component Inspector

The screenshot shows the leWarrior Development Studio interface. The top menu bar includes 'Is', 'Processor Expert', 'Window', and 'Help'. Below the menu is a toolbar with various icons. The main window is titled 'Component Inspector - LED_TMR'. It has tabs for 'Properties', 'Methods', and 'Events'. The 'Properties' tab is active, displaying a table with the following data:

Name	Value	Details
Component name	LED_TMR	
Periodic interrupt source	LPTMR0_CMRR	LPTMR0_CMRR
Counter	LPTMR0_CNRR	LPTMR0_CNRR
Interrupt service/event	Enabled	
Interrupt	INT_LPTimer	INT_LPTimer
Interrupt priority	medium priority	2
Interrupt period	250 ms	250 ms
Component uses entire timer	no	

An orange arrow points to the 'Minimize' button in the top right corner of the Component Inspector window. The text 'Click "Minimize"' is written in orange next to the arrow.

Minimize Component Inspector

The screenshot displays the CodeWarrior Development Studio environment. The main window shows the source file `my_main.c` with the following content:

```

1 /* ===== */
2 /* */
3 /* my_main.c */
4 /* */
5 /* */
6 /* Description */
7 /* This file is the companion file for the Example_1 This is to */
8 /* demonstrate the ability to drag and drop a file into a project */
9 /* ===== */
10
11 /* Includes needed for the high level devices to work */
12 #include "SW1.h"
13 #include "BitIoLdd1.h"
14 #include "SW2.h"
15 #include "BitIoLdd5.h"
16 #include "LED_GRN.h"
17 #include "BitIoLdd3.h"
18 #include "LED_RED.h"
19 #include "BitIoLdd4.h"
20 #include "LED_TMR.h"
21 #include "TimerIntLdd1.h"
22 #include "TU1.h"
23 #include "AS1.h"
24 #include "ASerialLdd1.h"
25
26
27 /* declarations needed for the subroutines */
28
29 void SW1_pressed(void);
30 void SW2_pressed(void);
31

```

The left sidebar contains three panels:

- CodeWarrior Projects:** Shows a tree view of the project structure. The 'Sources' folder contains `Events.c`, `Events.h`, `my_main.c`, `ProcessorExp`, and `sa_mtb.c`.
- Components - project_1:** Shows the component model. The 'Components' folder contains `Referenced_Components`, `PTC:Init_GPIO`, and `SW1:BitIO`.
- Commander:** Contains 'Project Creation' and 'Build/Debug' sections. 'Project Creation' includes options like 'Import project', 'Import example project', 'Import MCU executable file', 'New MCU project', and 'New MQX-Lite project'. 'Build/Debug' includes 'Build (All)', 'Clean (All)', 'Debug', and 'Settings' (Project settings, Build settings, Debug settings).

The bottom panel shows the 'Problems' and 'Console' tabs. The 'Problems' tab indicates '0 errors, 2 warnings, 0 others'.

Add “Our” Code

- Open “Processor Expert.c to add our code

```

52
53 /* User includes (#include below this line is not maintained by Processor Expert)
54 void my_main(void);
55 |
56
57 /*lint -save -e970 Disable MISRA rule (6.3) checking. */
58 int main(void)
59 /*lint -restore Enable MISRA rule (6.3) checking. */
60 {
61     /* Write your local variable definition here */
62
63     /** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! */
64     PE_low_level_init();
65     /** End of Processor Expert internal initialization. */
66
67     /* Write your code here */
68     /* For example: for(;;) { } */
69
70     my_main();
71
72

```

54 void my_main(void); ← Add declaration here

70 my_main(); ← Add call here

Check Point: Create a New Project to Blink an LED

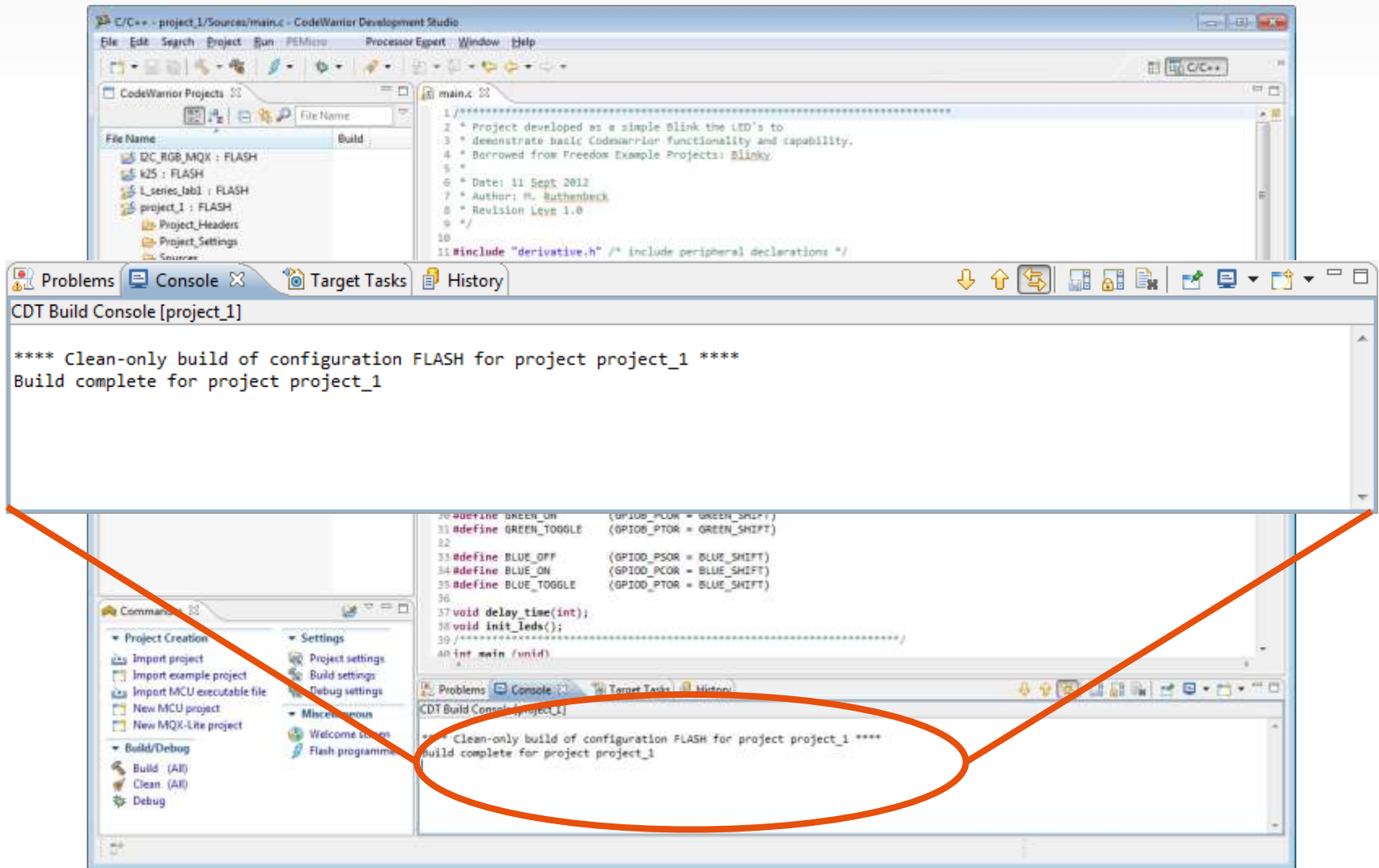
- This hands-on lab shows you how to...
 - Create a new project with the New Project Wizard ✓
 - Select and setup High Level Components ✓
 - Generate Processor Expert Code ✓
 - Import existing files ✓
 - **Build the project**
 - **Select the project**
 - **Clean**
 - **Build**
 - Test the application's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

Next up!

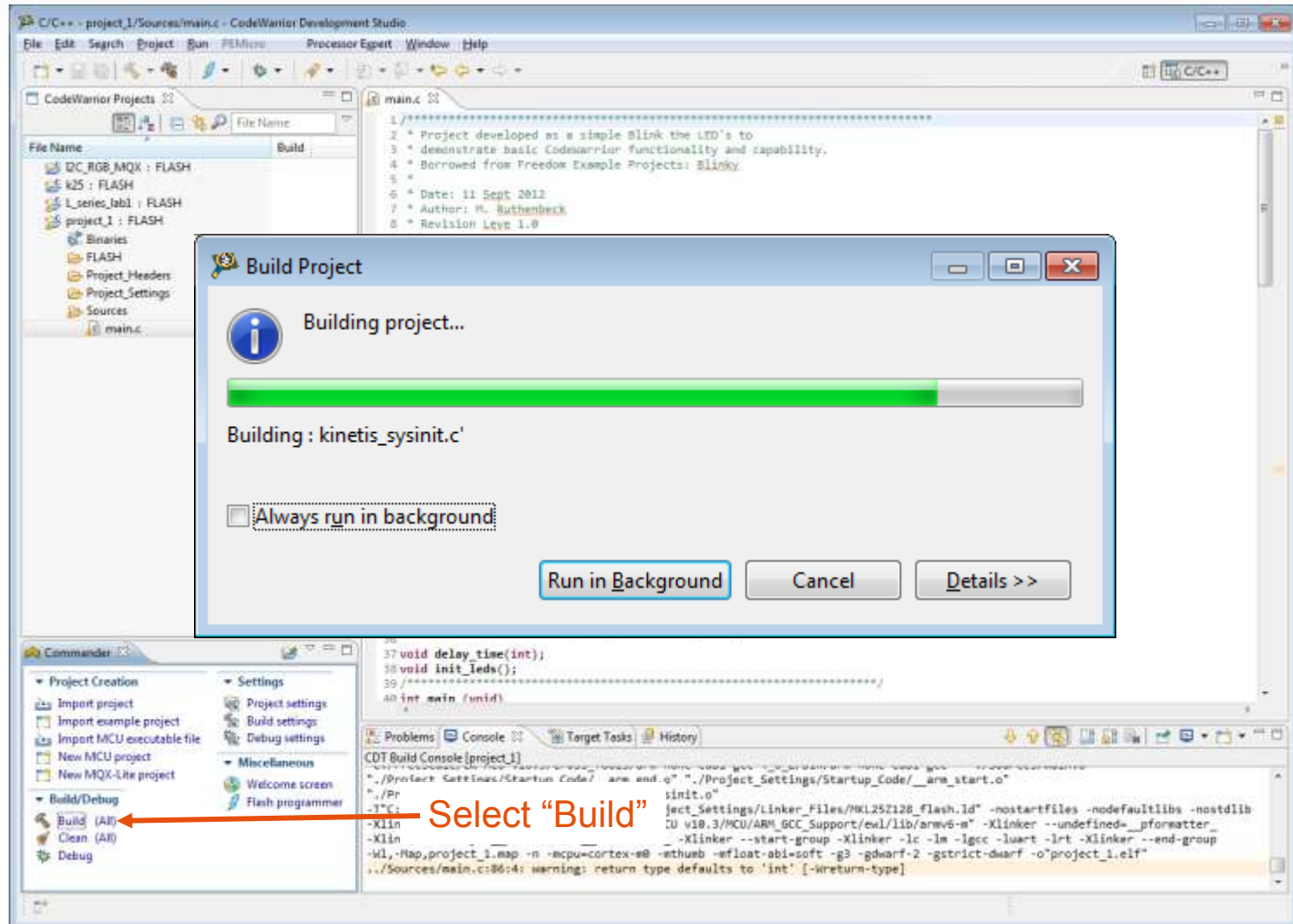
Building the Project: Clean

The screenshot shows the CodeWarrior Development Studio interface. The 'CodeWarrior Projects' pane on the left lists several projects, with 'project_1 : FLASH' selected. The 'Commander' pane on the right shows the 'Build/Debug' section, where the 'Clean (All)' option is selected. A red arrow points from the text 'Select "project_1"' to the 'project_1 : FLASH' entry in the projects list. Another red arrow points from the text 'Select "Clean"' to the 'Clean (All)' button in the 'Build/Debug' section.

Building the Project: Clean



Building the Project



Examine Processor Expert .c

PEx #includes here

```

27 /* MODULE ProcessorExpert */
28
29
30 /* Including needed modules to compile this m
31 #include "Cpu.h"
32 #include "Events.h"
33 #include "PTC.h"
34 #include "SW1.h"
35 #include "BitIoLdd1.h"
36 #include "SW2.h"
37 #include "BitIoLdd5.h"
38 #include "LED_GRN.h"
39 #include "BitIoLdd3.h"
40 #include "LED_RED.h"
41 #include "BitIoLdd4.h"
42 #include "LED_TMR.h"
43 #include "TimerIntLdd1.h"
44 #include "TU1.h"
45 #include "AS1.h"
46 #include "ASerialLdd1.h"
47 /* Including shared modules, which are used f
48 #include "PE_Types.h"
49 #include "PE_Error.h"
50 #include "PE_Const.h"
51 #include "IO_Map.h"
52

```

Examine Processor Expert .c

Your declarations here

Main.c

Hardware initializations

Your code goes here

```

53 /* User includes (#include below this line is
54 void my_main(void);
55
56
57 /*lint -save -e970 Disable MISRA rule (6.3)
58 int main(void)
59 /*lint -restore Enable MISRA rule (6.3) check
60 {
61     /* Write your local variable definition her
62
63     /** Processor Expert internal initializati
64     PE_low_level_init();
65     /** End of Processor Expert internal initi
66
67     /* Write your code here */
68     /* For example: for(;;) { } */
69
70     my_main();

```

Examine events.c

Timer Interrupt code goes here →

```

1 my_main.c 2 ProcessExpert.c 3 Events.c
22 **      Put your event handler code here.
23 */
24 /*
25 ** @addtogroup Events_module Events module documentation
26 ** @{
27 */
28 /* MODULE Events */
29
30 #include "Cpu.h"
31 #include "Events.h"
32
33 #ifdef __cplusplus
34 extern "C" {
35 #endif
36
37
38 /* User Includes (#include below this line is not maintained by Processor Expert) */
39 #include "LED_GRN.h"
40
41 /**
42 **      Event      : Cpu_OnNMIINT (module Events)
43 **
44 **      Component   : GRN [MKL46Z250NC4]
45 **
46 **/
47 /**
48 **      @brief
49 **      This event is called when the Non maskable interrupt had
50 **      occurred. This event is automatically enabled when the [NMI
51 **      interrupt] property is set to 'Enabled'.
52 **/
53 /** ===== */
54 void Cpu_OnNMIINT(void)
55 {
56     /* Write your code here ... */
57 }
58
59 /**
60 **      Event      : LED_TMR_OnInterrupt (module Events)
61 **
62 **      Component   : LED_TMR [TimerInt]
63 **      Description :
64 **      When a timer interrupt occurs this event is called (only
65 **      when the component is enabled - <Enable> and the events are
66 **      enabled - <EnableEvent>). This event is enabled only if a
67 **      <Interrupt service/event> is enabled.
68 **      Parameters : None
69 **      Returns    : Nothing
70 **
71 ** =====
72 **/
73 void LED_TMR_OnInterrupt(void)
74 {
75     /* Write your code here ... */
76     LED_GRN_NegVal();
77 }
78
79
80 /* END Events */
81
82 #ifdef __cplusplus
83 } /* extern "C" */
84 #endif
85
86 /**
87 ** @}
88 **/
89 /**
90 ** =====
91 **

```

Examine My_main.c

```
11 /* Includes needed for the high level devices to work, copied from ProcessorExpert.c */
12 #include "SW1.h"
13 #include "BitIoLdd1.h"
14 #include "SW2.h"
15 #include "BitIoLdd5.h"
16 #include "LED_GRN.h"
17 #include "BitIoLdd3.h"
18 #include "LED_RED.h"
19 #include "BitIoLdd4.h"
20 #include "LED_TMR.h"
21 #include "TimerIntLdd1.h"
22 #include "TU1.h"
23 #include "AS1.h"
24 #include "ASerialLdd1.h"
25
26
27 /* declarations needed for the subroutines */
28
29 void SW1_pressed(void);
30 void SW2_pressed (void);
31 void sendstring (char *str);
32 void delay(void);
```

Examine My_main.c

```

void my_main()
{
    sendstring("Welcome to Processor Expert and CodeWarrior!\n\n\n");

    while (1) {
        if ((SW1_GetVal()==0) /* Check to see if SW1 has been pressed */)
            SW1_pressed();
        else
            LED_RED_PutVal(1);

        if ((SW2_GetVal()) ==0) /* If SW2 has been pressed */)
            SW2_pressed();
        else
            LED_RED_PutVal(1); /* make sure led is OFF */

        delay(); /* for demo purposes only, not recommended in real life :) */
    }

    /* what to do when SW1 is pressed! *****/
    void SW1_pressed(){
        LED_RED_PutVal(0);
        sendstring ("!\nSW1 has been pressed!\n\n"); /* announce to the world switch has been pressed */
    }

    /* what to do when SW2 is pressed! *****/
    void SW2_pressed(){
        LED_RED_PutVal(0);
    }

    /*void sendstring (char *str){
        int n=0;
        while (str[n] != '\0'){
            while (ASL_SendChar(str[n])==ERR_TXFULL); /* wait for buffer to empty before moving on */
            n++;
        }
    }*/

    /****** function for short delay *****/
    /* Not recommended for real life coding - for demonstration only */
    void delay(){
        unsigned int i, m;
        for(i=0; i<3000; i++)
        {
            for(n=0; n<20; n++)
            {
                asm("nop");
            }
        }
    }
}

```

Main loop – watch for switch presses

Switch press routine

Delay loop

Check Point: Create a New Project to Blink an LED

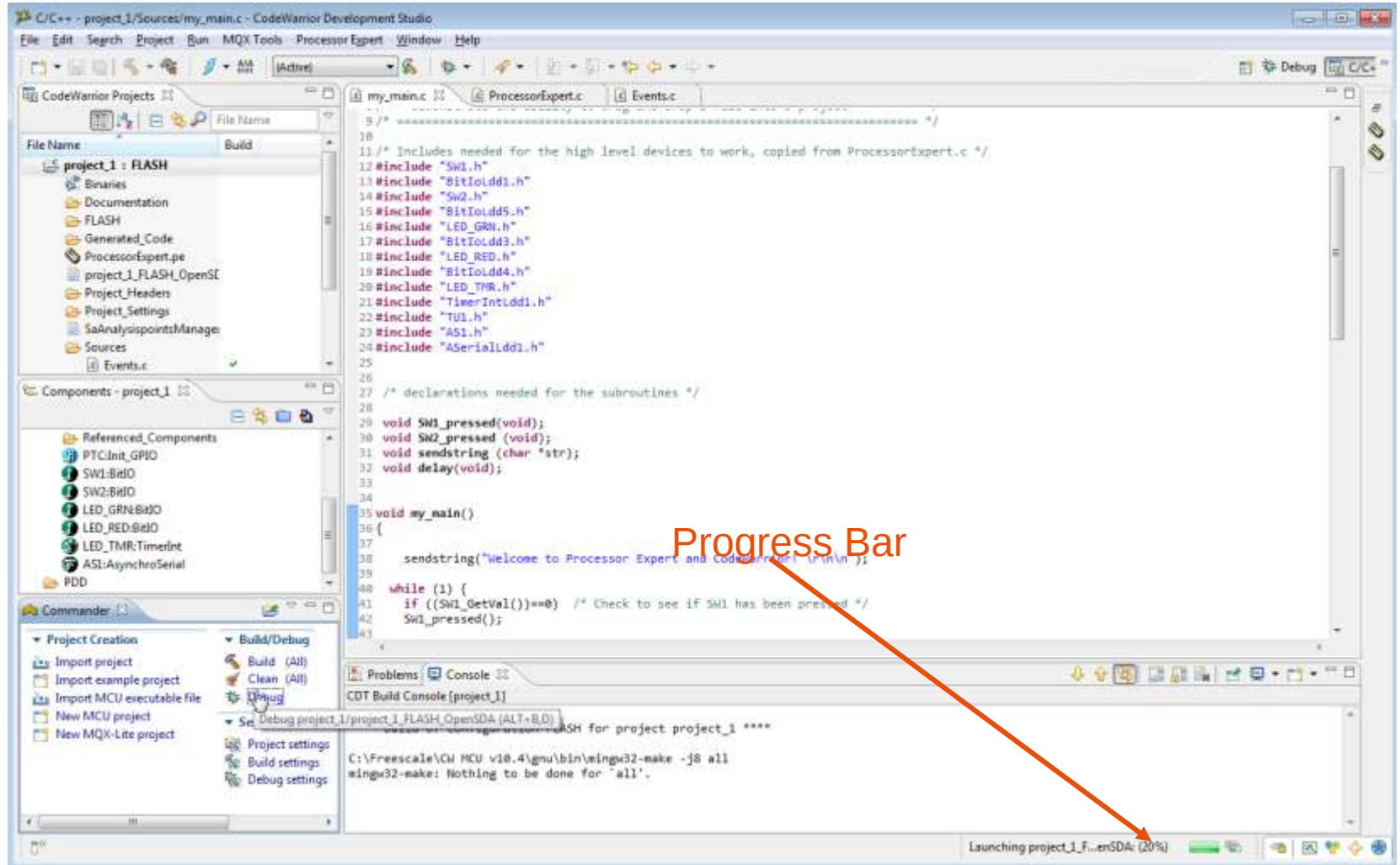
- This hands-on lab shows you how to...
 - Create a new project with the New Project Wizard ✓
 - Select and setup High Level Components ✓
 - Generate Processor Expert Code ✓
 - Import existing files ✓
 - Build the project ✓
 - **Test the application's functionality**
 - Download and debug
 - Inspect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

Next up!

Application Test: Download/Debug

The screenshot displays the CodeWarrior Development Studio interface. The 'CodeWarrior Projects' pane on the left shows a project named 'project_1 : FLASH'. The 'Components - project_1' pane lists various components like PTCInit_GPIO, SW1-BtIO, SW2-BtIO, LED_GRN-BtIO, LED_RED-BtIO, LED_TMR:TimerInt, and ASI:AsynchroSerial. The 'Commander' pane at the bottom left shows project creation options. The main editor window displays the source code for 'my_main.c', which includes various headers and defines functions like SW1_pressed, SW2_pressed, sendstring, delay, and my_main. The 'Problems' and 'Console' panes at the bottom right show the build process output, including the command: `/bin/arm-none-eabi-gcc @project_1.args -o"project_1.elf"`. Two red arrows point to the 'Build' button in the 'CodeWarrior Projects' pane and the 'Debug' button in the 'Commander' pane, with labels 'Select "project_1"' and 'Select "Debug"' respectively.

Application Test: Download/Debug

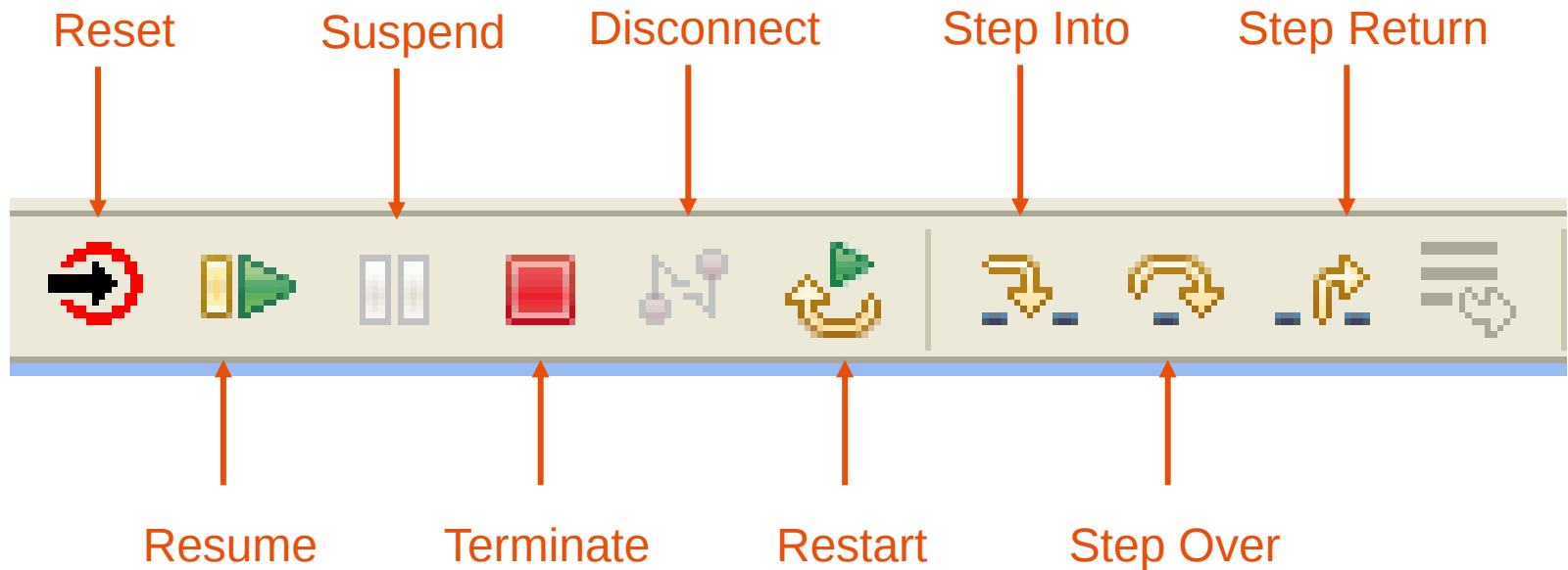


Application Test: Debug Perspective

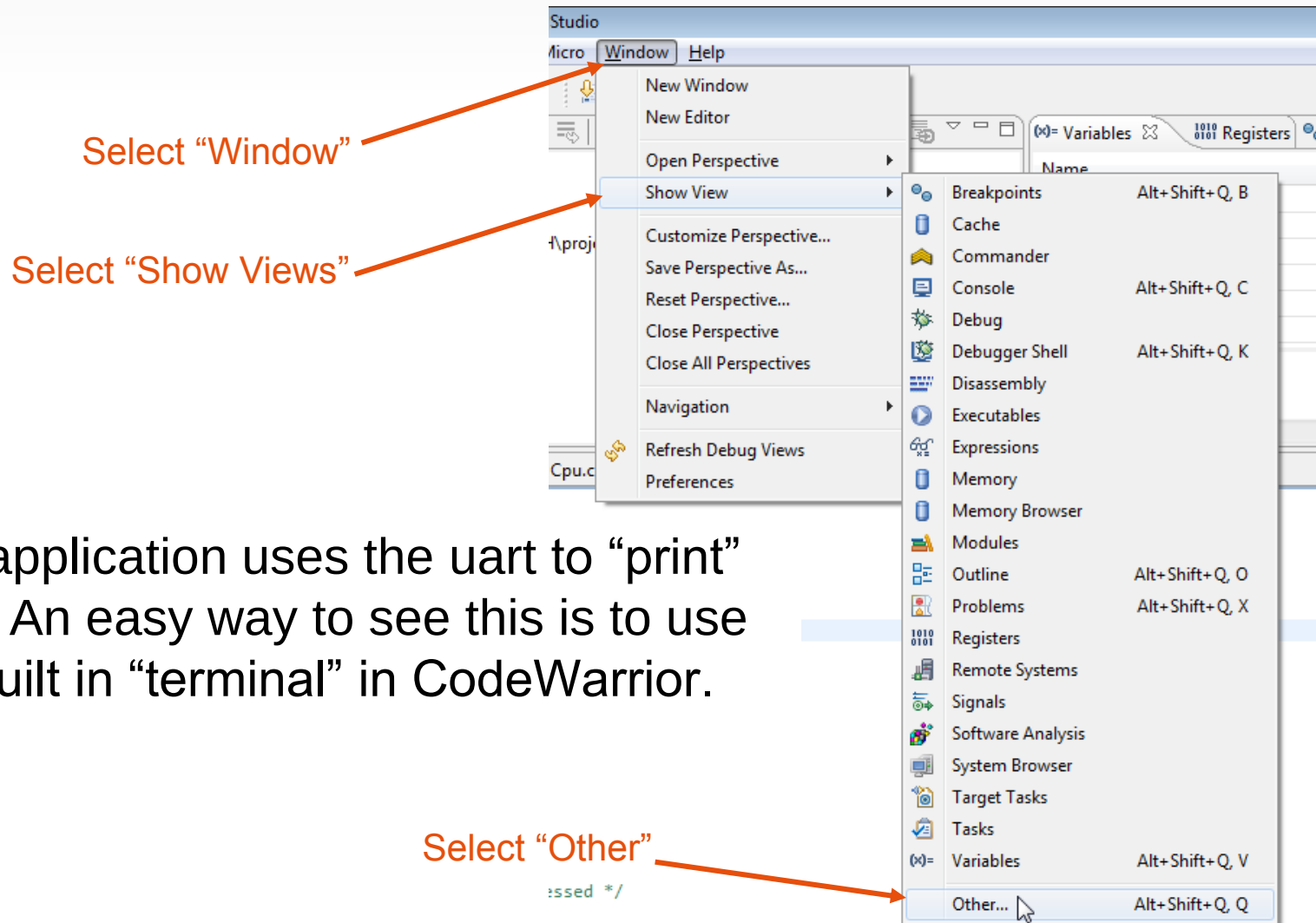
The screenshot displays the CodeWarrior Development Studio interface with several views open:

- Debug View:** Located at the top left, it shows the project hierarchy and the current thread (Thread [ID: 0x0] (Suspended: Signal 'Halt' received. Description: User halted thread.)).
- Variable View Stack:** Located at the top right, it displays a table with columns for Name, Value, and Location.
- Editor:** The central area showing the source code of `ProcessorExpert.c`. A red arrow points to a breakpoint set on line 50, which is the start of the `main` function.
- Disassembly View:** Located on the right side, it shows the assembly code corresponding to the selected line in the editor.
- Console View Stack:** Located at the bottom right, it displays the output of the program, including the text "ARM Processors, project 1.elf".
- Commander View:** Located at the bottom left, it provides a sidebar for project management, including options like "Import project", "Import example project", "Import MCU executable file", "New MCU project", and "New MQX-Lite project".

Application Test: Debug Perspective

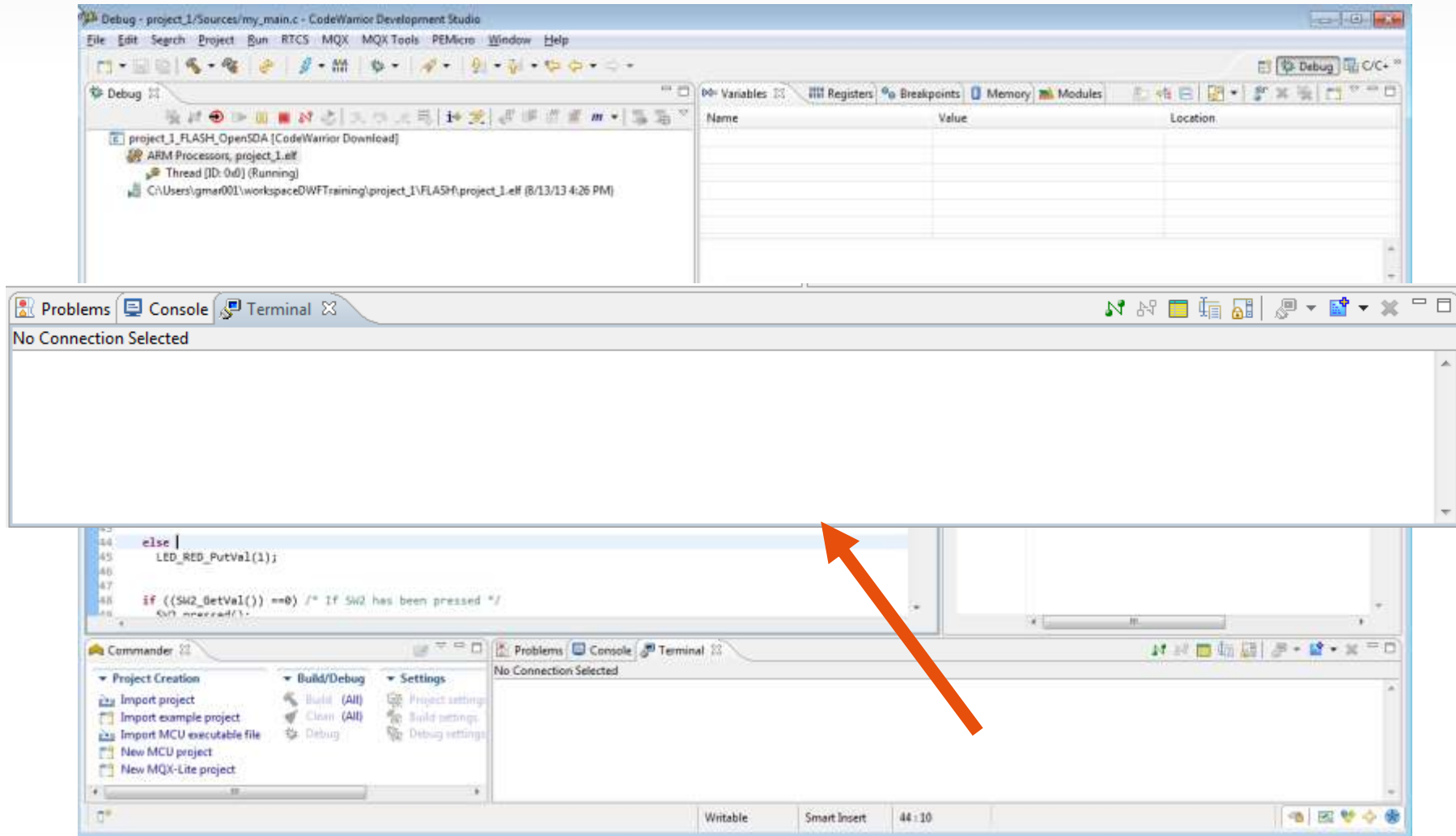


Application Test: Setting up Terminal

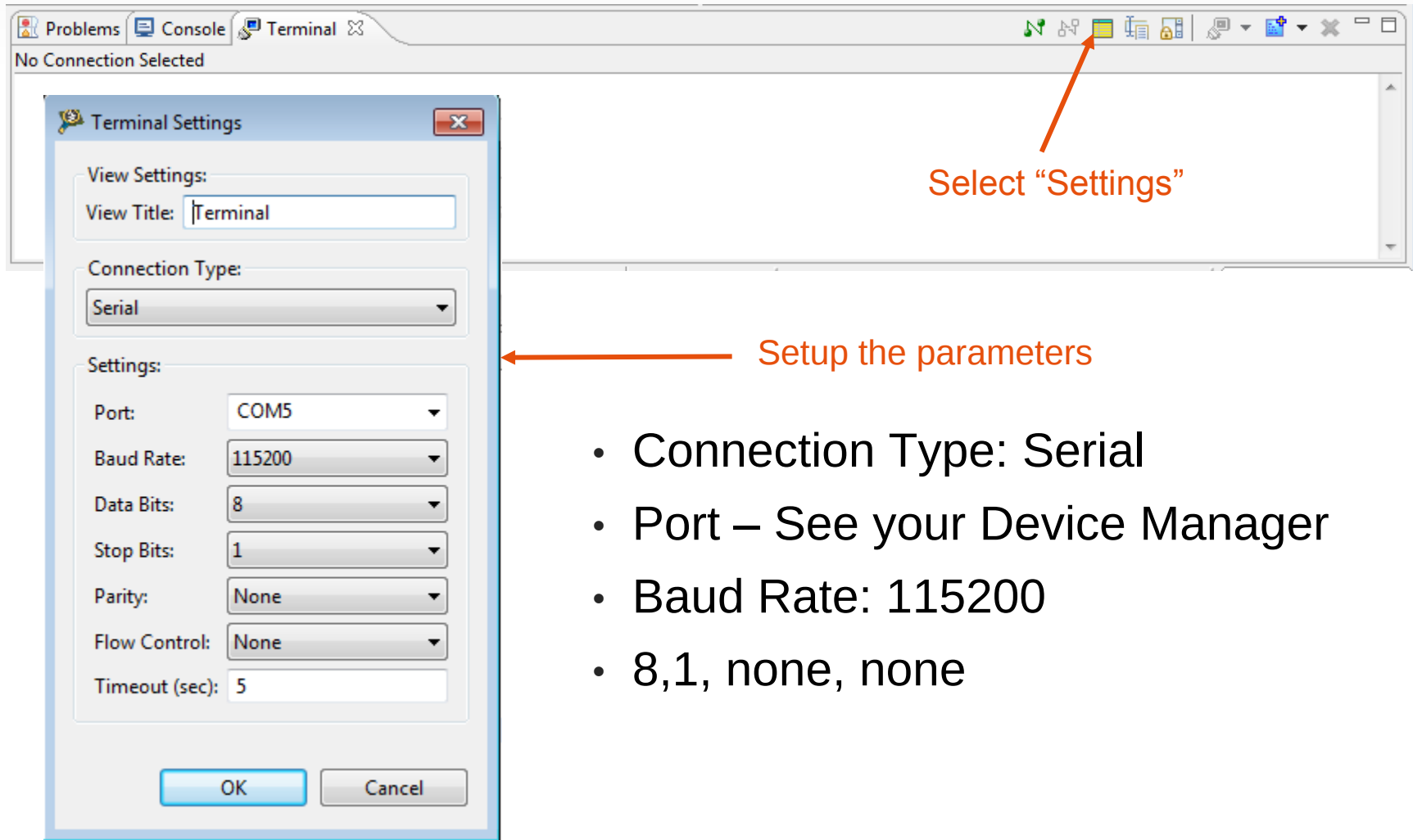


The application uses the uart to “print” data. An easy way to see this is to use the built in “terminal” in CodeWarrior.

Application Test: Setting up Terminal

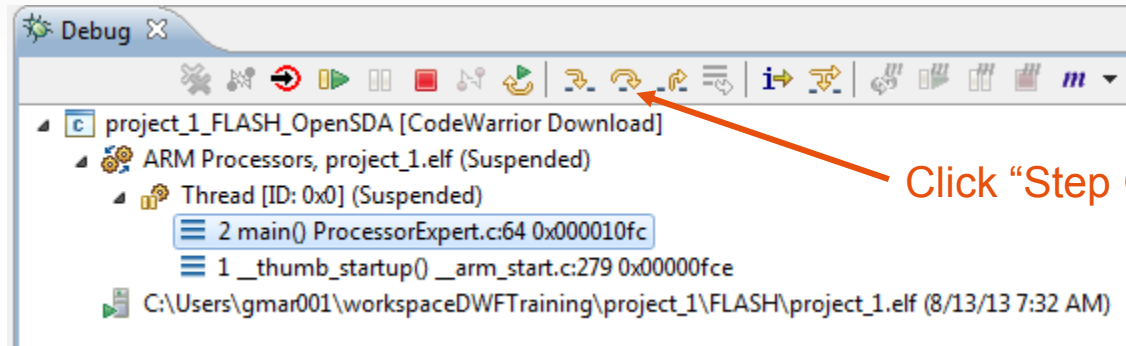


Application Test: Setting up Terminal

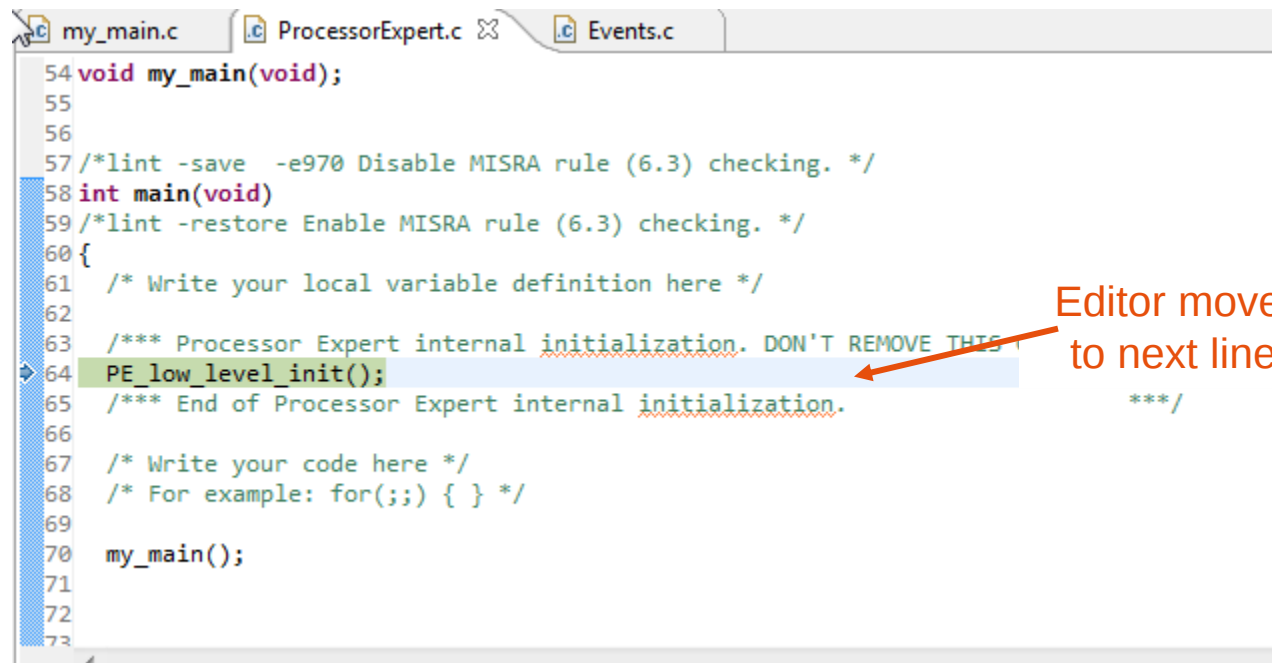


- Connection Type: Serial
- Port – See your Device Manager
- Baud Rate: 115200
- 8,1, none, none

Application Test: Single Step



- Click “Step Over”



Editor moves
to next line

Application Test: Single Step/Step Into

The screenshot shows the CodeWarrior IDE interface. At the top, the 'Debug' menu is open. The 'Step Into' button (represented by a magnifying glass over a right-pointing arrow) is highlighted in the toolbar. An orange arrow points from the text 'Click "Step Into"' to this button. Below the toolbar, the 'Project Explorer' on the left shows the project structure: 'project_1_FLASH_OpenSDA [CodeWarrior Download]' containing 'ARM Processors, project_1.elf (Suspended)' and a 'Thread [ID: 0x0] (Suspended)'. The thread's call stack lists: '3 PE_low_level_init() Cpu.c:195 0x000007e8', '2 main() ProcessorExpert.c:70 0x00001100', and '1 __thumb_startup() __arm_start.c:279 0x00000fce'. The main window displays the source code for 'Cpu.c'. The function 'void PE_low_level_init(void)' is selected, and the editor has moved to the line following the function definition. An orange arrow points from the text 'Editor moves to next line' to the line below the function definition. The code in the editor includes comments about RTOS and SIM module initialization, and defines PORTA_PCR4.

Click "Step Into"

Editor moves to next line

Application Test: Single Step

Click "Step Into"

Editor moves to next line

Application Test: Run

The screenshot shows the CodeWarrior Development Studio interface. The top menu bar includes File, Edit, Search, Project, Run, RTCS, MQX, MQX Tools, PEMicro, Window, and Help. The main workspace is divided into several panes:

- Debug Console:** Shows the execution of the program. The output indicates that the program is running on an ARM processor and has successfully printed the message "Welcome to Processor Expert and CodeWarrior!" to the terminal.
- Disassembly:** Shows the assembly code for the program, with the instruction "sendstring" highlighted.
- Terminal View:** Displays the output of the program, showing the message "Welcome to Processor Expert and CodeWarrior!" printed to the terminal.

An orange arrow points from the text "UART 'prints' message to Terminal View" to the terminal output, indicating that the message is being sent to the terminal via the UART interface.

Application Test: Inspect Registers

The screenshot shows the ARM DS-5 IDE interface. The top window is the 'Registers' view, which is expanded to show 'User/System Mode Registers'. The registers listed are R0, R1, and R2, with their current values and memory locations (SR0, SR1, SR2). The bottom window is the 'Disassembly' view, showing assembly code for a function named 'for'. The code includes instructions for loading, adding, storing, and comparing registers, along with branch instructions. The instruction 'cmp r3, #19' is highlighted in blue.

Select "Register View"

Expand "User/System Mode Registers"

Name	Value	Location
User/System Mode Registers		
R0	0x1ffe038	SR0
R1	0x00000001	SR1
R2	0x00000275	SR2

```

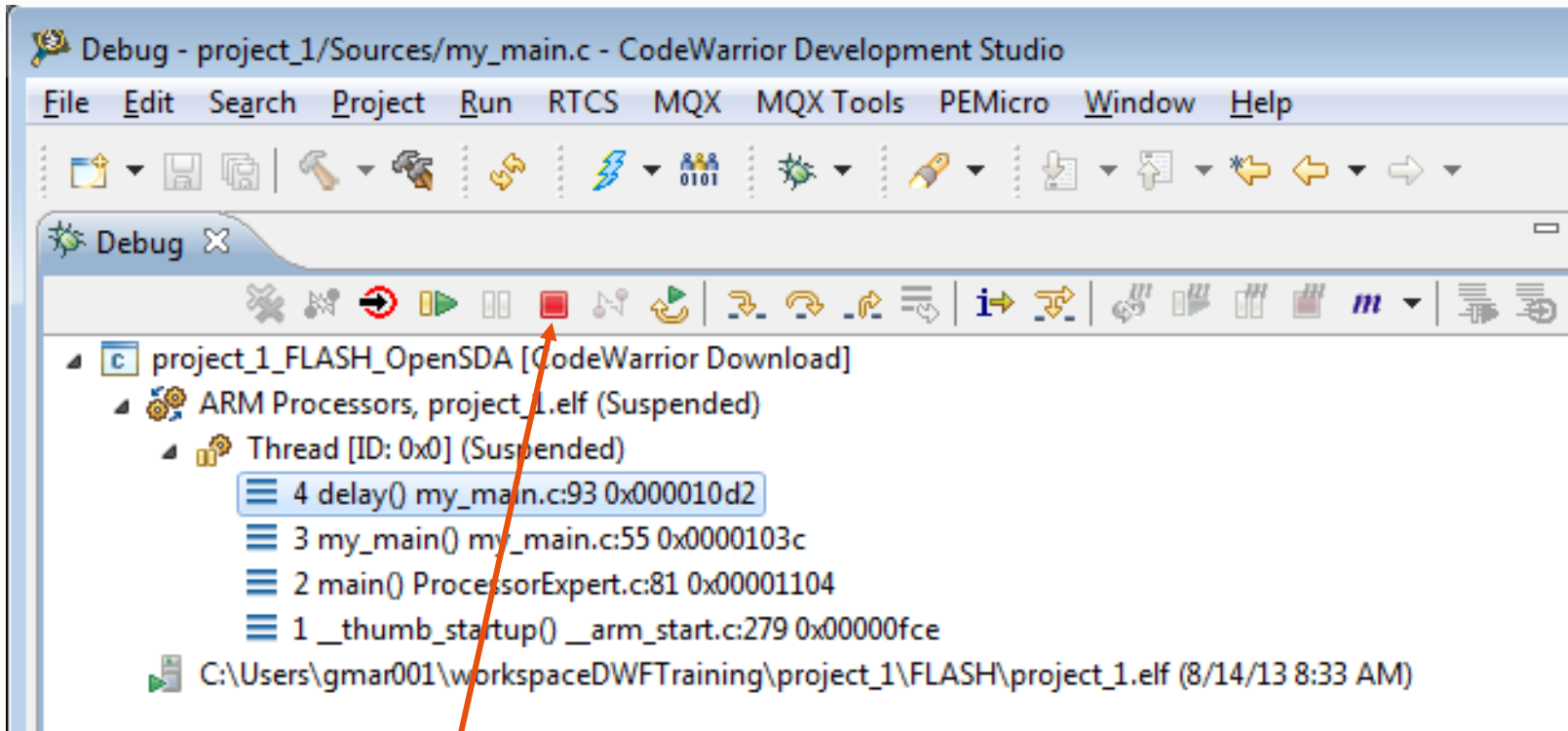
91 91      for(n=0;n<20;n++)
0) 000010d4: ldr r3,[r7,#0]
0) 000010d6: adds r3,#1
0) 000010d8: str r3,[r7,#0]
0) 000010da: ldr r3,[r7,#0]
0) 000010dc: cmp r3,#19
0) 000010de: bls delay+0x12 (0x10d2) ; 0x000010d2
8) 89      for(i=0;i<3000;i++)
0) 000010e0: ldr r3,[r7,#4]
0) 000010e2: adds r3,#1
0) 000010e4: str r3,[r7,#4]
0) 000010e6: ldr r2,[r7,#4]
0) 000010e8: ldr r3,[pc,#8]
0) 000010ea: cmp r2,r3
  
```

Application Test: Inspect Registers

The screenshot shows the CodeWarrior Studio interface. The 'Registers' window is open, displaying a table of registers. Registers R0, R1, R2, and R3 are highlighted in yellow. An arrow points to the 'Step Over' button in the toolbar, and another arrow points to the highlighted registers with the text 'Changed registers are highlighted'.

Name	Value	Location
User/System Mode Registers		
R0	0x1fffe038	\$R0
R1	0x00000001	\$R1
R2	0x00000275	\$R2
R3	0x0000000d	\$R3

Application Test: Inspect Registers



Click “Terminate” to stop debug session

Application Test: Setting Breakpoints

Double click
at line 38
and line 41
to set breakpoints

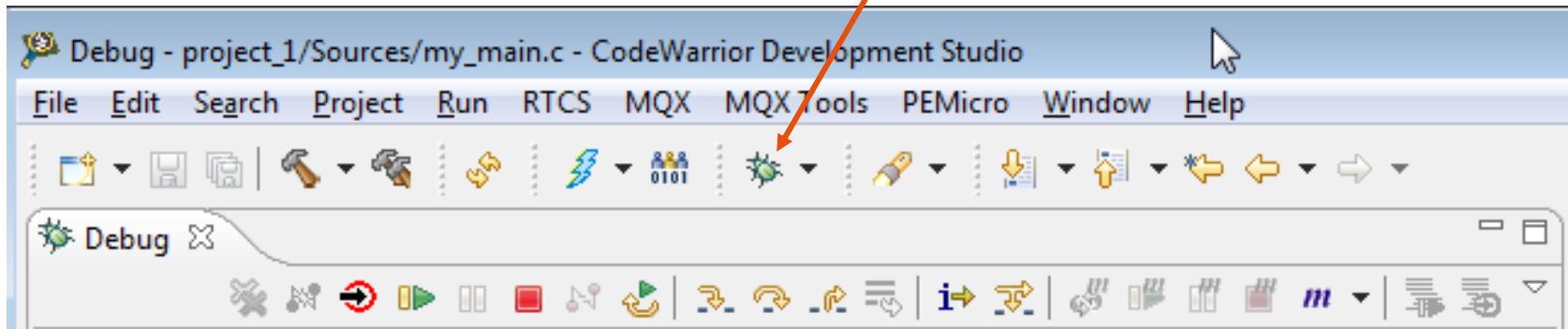
```

34
35 void my_main()
36 {
37
38     sendstring("Welcome to Processor Expert and CodeWarrior! \r\n\n");
39
40     while (1) {
41         if ((SW1_GetVal())==0) /* Check to see if SW1 has been pressed */
42             SW1_pressed();
43
44         else
45             LED_RED_PutVal(1);
46
47
48         if ((SW2_GetVal()) ==0) /* If SW2 has been pressed */
49             SW2_pressed();
50
51         else
52             LED_RED_PutVal(1); /* make sure led is OFF */
53
54         delay(); /* for demo purposes only, not recommended in real life :) */
55     }
56 }
57

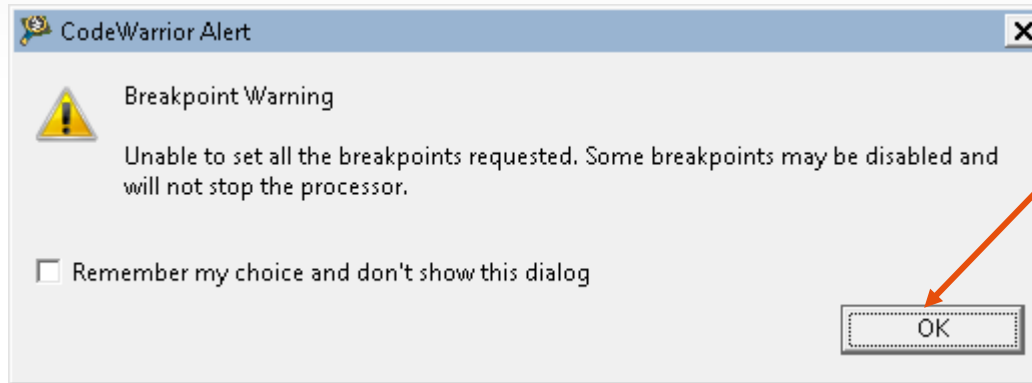
```

Application Test: Setting Breakpoints

Click “Debug” to restart
the debug session



Application Test: Setting Breakpoints



Click “OK”
to continue
debug session

- This error occurs because the number of break points set exceeds the number of breakpoints available on the chip.
- The KL46 only has 2 breakpoints available.
- There are 3 breakpoints set
 - At main
 - At line 38
 - At line 41

Application Test: Setting Breakpoints

- Code stopped at first user breakpoint and not at the system set breakpoint at “main”.

```

34
35 void my_main()
36 {
37
38     sendstring("Welcome to Processor Expert and CodeWarrior! \r\n\r\n");
39
40     while (1) {
41         if ((SW1_GetVal())==0) /* Check to see if SW1 has been pressed */
42             SW1_pressed();
43
44         else
45             LED_RED_PutVal(1);
46
47
48         if ((SW2_GetVal()) ==0) /* If SW2 has been pressed */
49             SW2_pressed();
50

```

- Click “Resume”



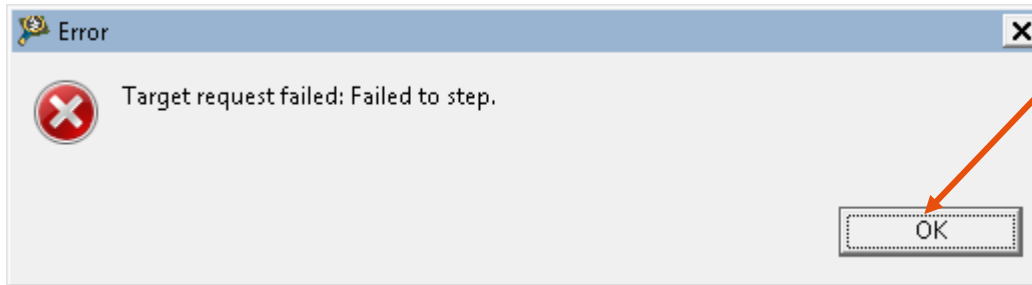
Application Test: Setting Breakpoints

- Click “Step Return”

```
my_main.c ProcessorExpert.c Events.c Cpu.c
34
35 void my_main()
36 {
37
38     sendstring("Welcome to Processor Expert and CodeWarrior! \r\n\r\n");
39
40     while (1) {
41         if ((SW1_GetVal())==0) /* Check to see if SW1 has been pressed */
42             SW1_pressed();
43
44         else
45             LED_RED_PutVal(1);
46
47
48         if ((SW2_GetVal()) ==0) /* If SW2 has been pressed */
49             SW2_pressed();
50
51         else
52             LED_RED_PutVal(1); /* make sure led is OFF */
53
54         delay(); /* for demo purposes only, not recommended in real life :) */
55     }
```

Stopped at the next Break Point

Application Test: Setting Breakpoints



Click “OK”
to continue
debug session

- This error message is displayed since there are too many break points.

Application Test: Setting Breakpoints

- Click “Resume”

```

34
35 void my_main()
36 {
37
38     sendstring("Welcome to Processor Expert and CodeWarrior! \r\n\n");
39
40     while (1) {
41         if ((SW1_GetVal())==0) /* Check to see if SW1 has been pressed */
42             SW1_pressed();
43
44         else
45             LED_RED_PutVal(1);
46
47
48         if ((SW2_GetVal()) ==0) /* If SW2 has been pressed */
49             SW2_pressed();
50
51         else
52             LED_RED_PutVal(1); /* make sure led is OFF */
53
54         delay(); /* for demo purposes only, not recommended in real life :) */
55     }

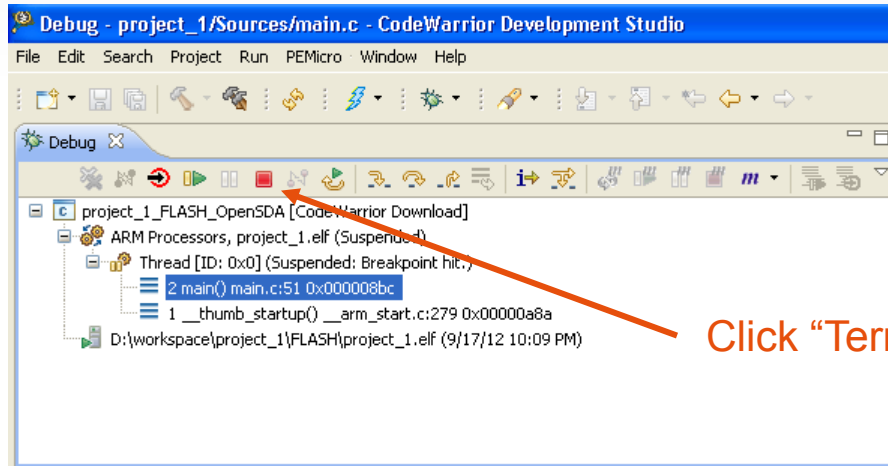
```



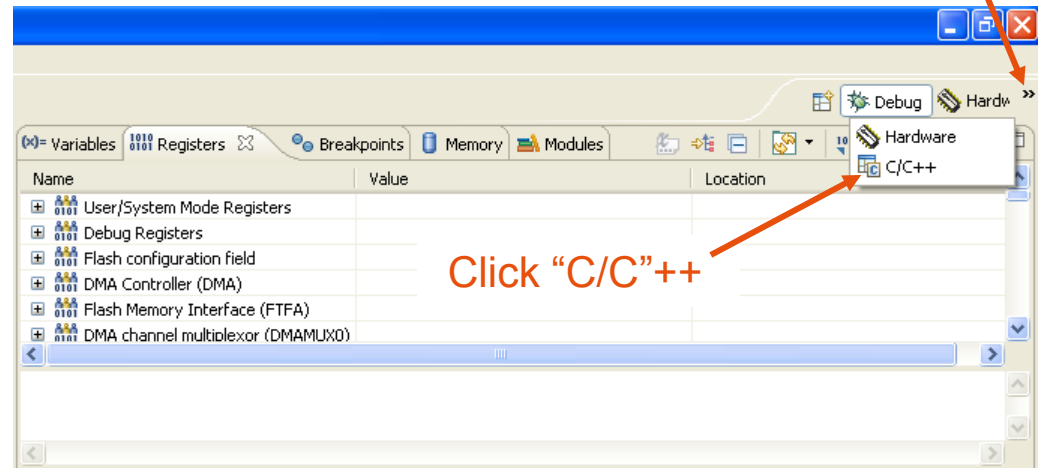
Application Test: Setting Breakpoints

```
34
35 void my_main()
36 {
37
38     sendstring("Welcome to Processor Expert and CodeWarrior! \r\n\r\n");
39
40     while (1) {
41         if ((SW1_GetVal())==0) /* Check to see if SW1 has been pressed */
42             SW1_pressed();
43
44         else
45             LED_RED_PutVal(1);
46
47
48         if ((SW2_GetVal()) ==0) /* If SW2 has been pressed */
49             SW2_pressed();
50
51         else
52             LED_RED_PutVal(1); /* make sure led is OFF */
53
54         delay(); /* for demo purposes only, not recommended in real life :) */
55     }
```

Application Test: Setting Breakpoints



Click “Terminate”



Click “>>”

Click “C/C”++

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?**



Appendix

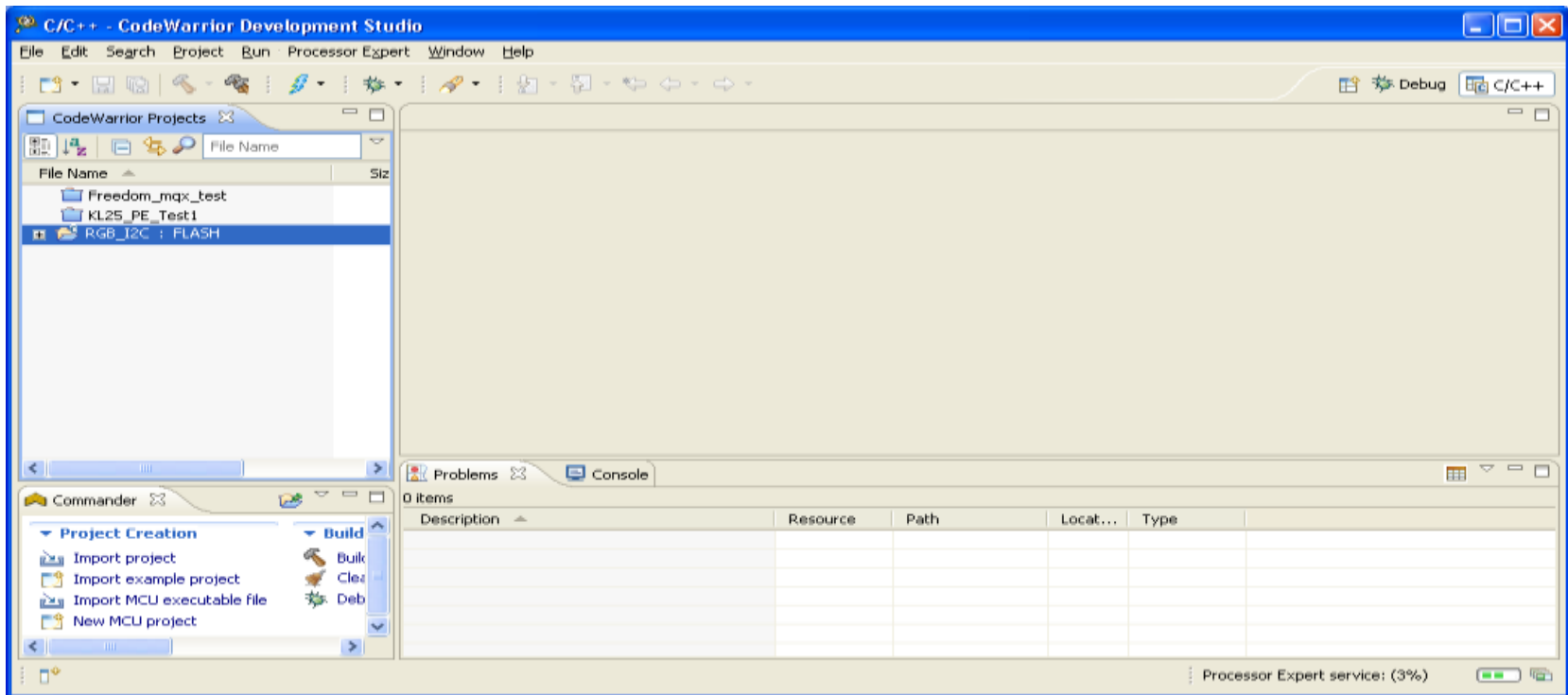




FreeScale, the FreeScale logo, i.MX, iMX6, iMX7, C-SPY, C-TEST, CodeWarrior, ColdFire, Colibri, e-Work, the Energy Efficient Subatomic Logic, Kinetis, mbedGT, PPG, PowerQUICC, Freescale Expertise, QoQo, QorIQ, SafeWatch, the SafeWatch logo, StarCore, Symphony and ViorQ are trademarks of Freescale Semiconductor, Inc. Reg. U.S. Pat & Tm. Off. AirBot, Beagle, iKerSkat, SmartRef, XScale, LayerAcad, MagPiC, M080, Platform in a Package, QoT, Goeyond, QoQo, DeepFit, Ready Plug, SMARTMOS, Tower, Turbulaid, Vybrid and Vybrid are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © 2013 Freescale Semiconductor, Inc.

Workbench

- **Workbench** = desktop development environment
 - Contains all C/C++ development-related tools
 - Shows different perspectives of the working environment
 - Main window that appears when CodeWarrior Eclipse starts:



Workspace

- **Workspace** = directory that stores the source code, files and settings related to your work
 - Specify the workspace on startup
 - More than one project can be in a workspace
 - To switch workspaces select **File > Switch Workspace**



Project

- **Project** = container for organizing files and folders
 - All C/C++ work is done in the context of projects
 - CodeWarrior creates projects in new folders by default
 - New files and folders can be added to a project
 - **File > New > File:** Creates a new file in project directory within the workspace directory
 - **File > New > Folder:** Creates a new directory in project directory within the workspace directory.
 - Files dragged into a project are physically copied into the project directory
 - Files outside the project folder (or outside the workspace) can be linked to a project
 - **File > New > File > Advanced > Link to File System:** Creates new link in project directory within the workspace, which refers to a file in the file system
 - **File > New > Folder > Advanced > Link to File System:** Creates new link to a directory in project folder within the workspace. The link points to a directory in the user's file system. Creating this link will pull the directory and all sub-components into the CodeWarrior project.

View

- **View** = a “visual” panel in the Workbench
 - Displays information about the contents of your workbench
 - Different views are provided for different tasks
 - Problems – Compile or other problems
 - Console – Standard run window console
 - C/C++ Projects – View of projects in workspace
 - Breakpoints – Debugging breakpoints in open projects
 - Editors are special “views” for editing files

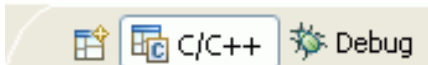
Problems Console Properties C/C++ Index

2 errors, 2 warnings, 0 infos

Description	Resource	Path	Location
Errors (2 items)			
Error: expression syntax error	mymain.c	DebugHello	line 17
ERROR: Non-zero return status from "C:\DebugHello			Unknown
Warnings (2 items)			
Warning: variable / argument 'argc' is not	main.c	HelloAgain	line 4
Warning: variable / argument 'argv' is not	main.c	HelloAgain	line 4

Perspectives

- **Perspective** = collection of Eclipse views and action sets organized into a layout that suits an assigned task
 - Current perspective is highlighted at the top of the perspective window
 - Perspectives are shown on perspective shortcut bar (top right)



- Click the perspective icon to switch among open perspectives
- To open a perspective:
 - Select **Window > Open Perspective > [select -or- Other]**
 - On the perspective view, choose **Open Perspective > [select -or- Other]**

Open Perspective Icon



- To close a perspective:
 - Select **Window > Close Perspective**



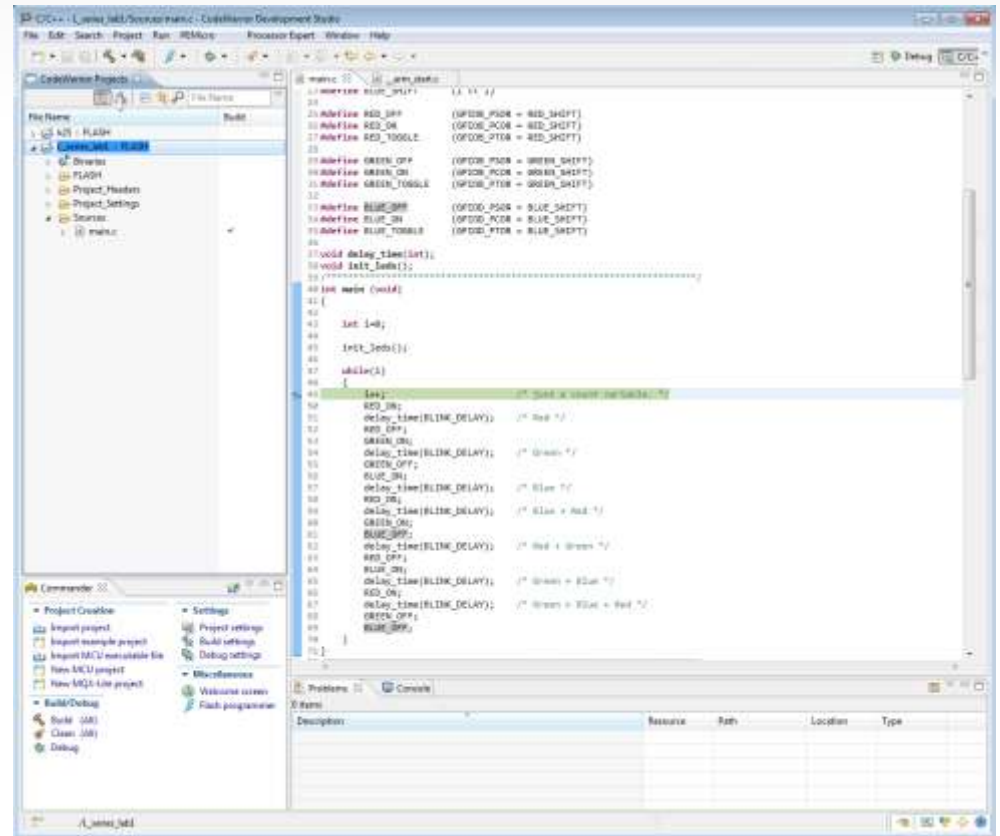
C/C++ Perspective



Freemake, the Freemake logo, AllCode, C#, CodeTEST, CodeWarrior, CodePins, CodeLines, C-Sharp, the Brings Efficient Solution! logo, Kinect, moolitEST, PEG, PowerQUICC, Processor Expert, QeIQ, QuickView, SafeShare, the SafeShare logo, StartCore, Symphony and WinQ are trademarks of Freemake Software Corporation, Inc. U.S. Pat & Tm. Off AirNet, Beekit, BeerStack, Coasting, Rites, Lysencore, Magick, Moko, Platform in a Package, QoD GoVoyage, QoD Desktop, Really Fast!, SMARTMOS, Tunes, TurboLink, Vybrid and Xtreme are trademarks of Freemake Software Corporation, Inc. All other product or service names are the property of their respective owners. © 2013 Freemake Software Corporation, Inc.

C/C++ Perspective

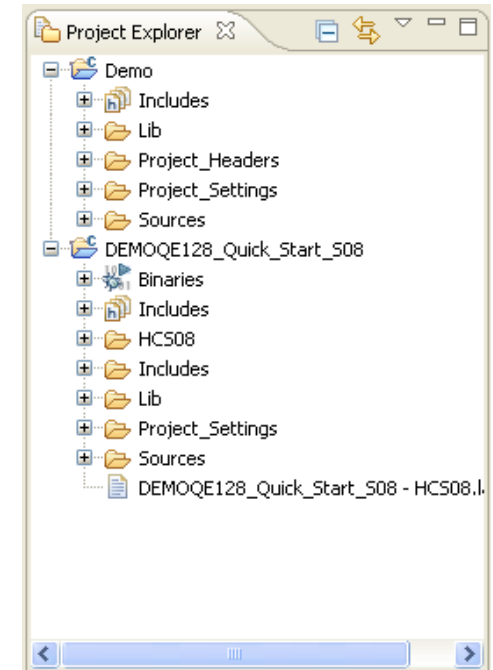
- Default views:
 - CodeWarrior Projects View
 - Editor (Area)
 - Problems View
 - Console View
 - Commander View



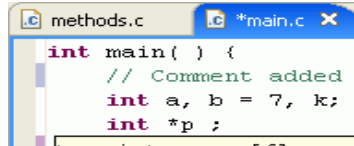
- Each view has an optional toolbar and menu that is separate from those on the main workbench window.

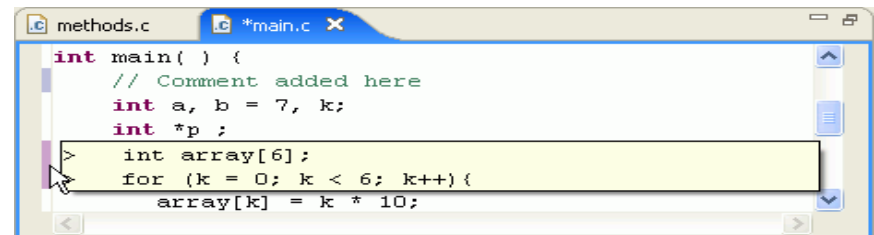
CodeWarrior Project View

- File space (folder) organization for source and object files
- Project folders are at the top level, but you can drill down:
Right-click folder > Go Into
- Filters types/status of resources to show (Toolbar menu > Filters)
- Other folders include:
 - Debug
 - Includes (compiler dependent) - contains makefiles
 - Binaries (generated by the compiler)



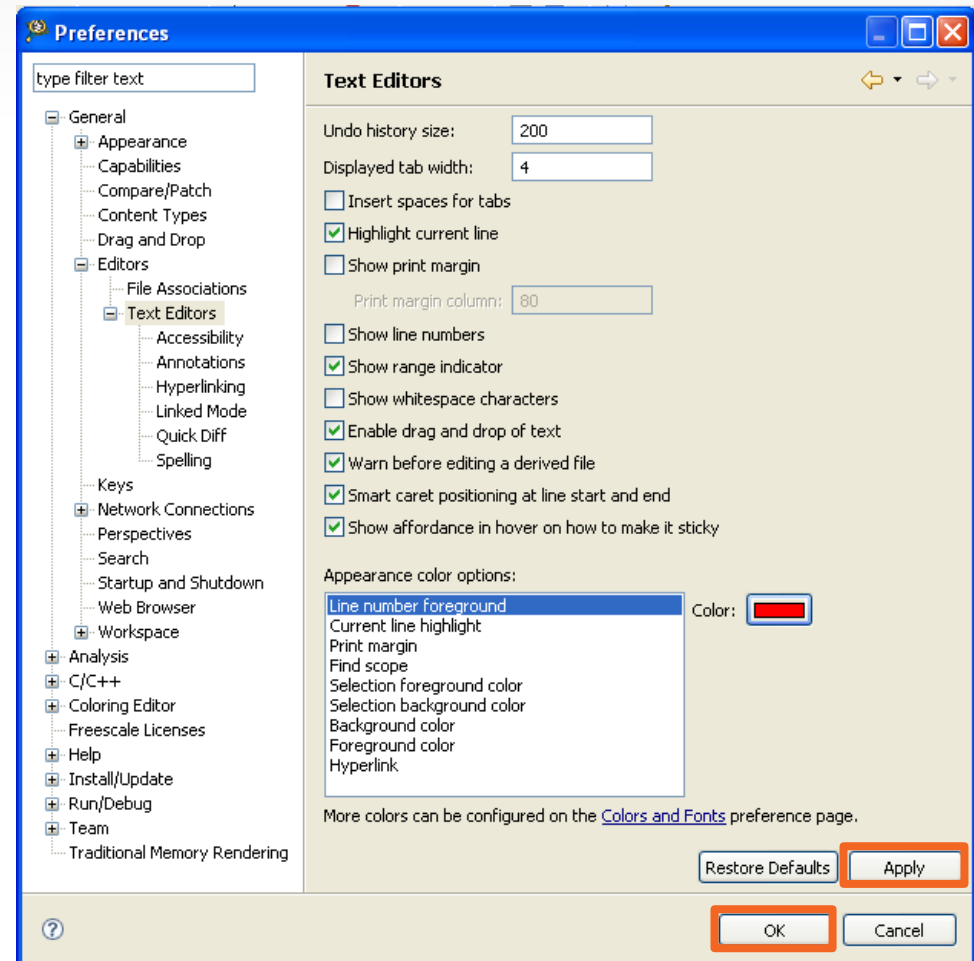
Editor Area

- There are multiple editors for multiple file types
 - C/C++ editor for source code
 - Text editor for text files
 - To open a file:
 - Double click on the file in CodeWarrior Projects or Navigator view
 - or -
 - Select File > Open File
 - Each file opens up in a new tab in editor area
 - Editors show changes in files since last save
 - See the change bar (left side of text)
 - Additions and modifications are color coded
 - Hover cursor over change bar to see previous text
 - Dirty file – indicated by * in editor tab.
 - Marker area on left side of editor indicates task, bookmarks or breakpoint
- 



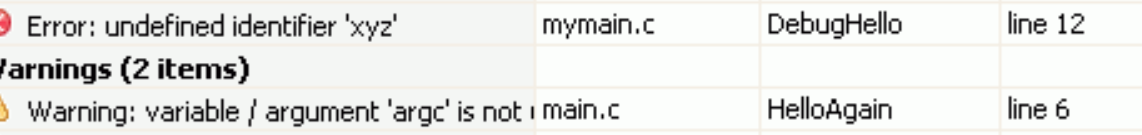
Configure Editors

- To configure editors:
 - Select **Window > Preferences**
 - Expand **General** folder
 - Select **Editors > Text Editors**
 - Configure options:
 - Show line numbers
 - Highlight current line
 - Color options
 - Etc.



Problems View

- Shows errors and warnings
 - Filters on problems to display:
 - Location of defective resource
 - Type of defect
 - Type of problem
 - Double click an error/warning message to navigate to offending source code

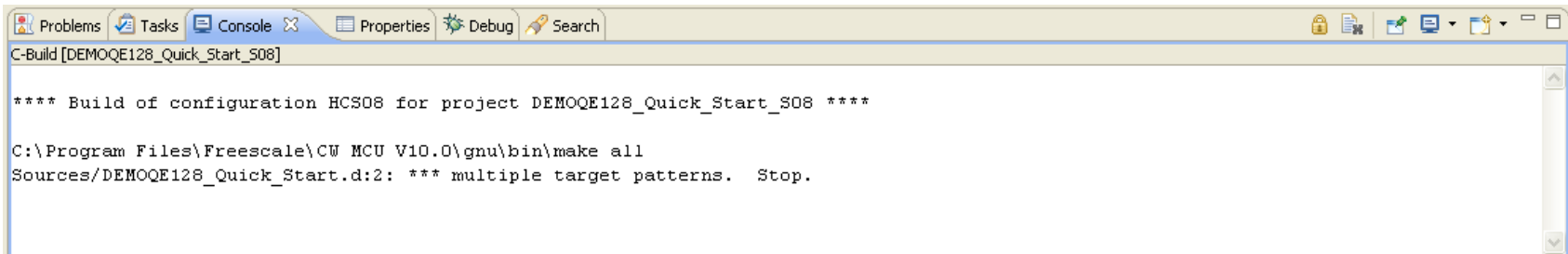


The screenshot shows the 'Problems' window in Visual Studio, which displays a list of errors and warnings. The window has tabs for 'Problems', 'Console', and 'Properties'. The 'Problems' tab is active, showing a summary of '3 errors, 2 warnings, 0 infos'. Below this, a table lists the individual issues:

Description	Resource	Path	Location
Errors (3 items)			
Error: declaration syntax error	sample.h	DebugHello	line 7
ERROR: Non-zero return status from "C:\Program Files\Microsoft Visual Studio\2017\Community\VC\Tools\MSVC\14.16.26324\bin\Hostx64\x64\cl.exe"		DebugHello	Unknown
Error: undefined identifier 'xyz'	mymain.c	DebugHello	line 12
Warnings (2 items)			
Warning: variable / argument 'argc' is not declared	main.c	HelloAgain	line 6
Warning: variable / argument 'argv' is not declared	main.c	HelloAgain	line 6

Console View

- Displays standard I/O
- Echoes Make files
- Displays program output
- Console toolbar:
 - Scroll Lock
 - Clear Console
 - Pin Console (save this console in separate view)
 - Display Selected Console
 - Open Console



```

C-Build [DEMOQE128_Quick_Start_S08]

**** Build of configuration HCS08 for project DEMOQE128_Quick_Start_S08 ****

C:\Program Files\Freescale\CW MCU V10.0\gnu\bin\make all
Sources\DEMOQE128_Quick_Start.d:2: *** multiple target patterns.  Stop.
  
```



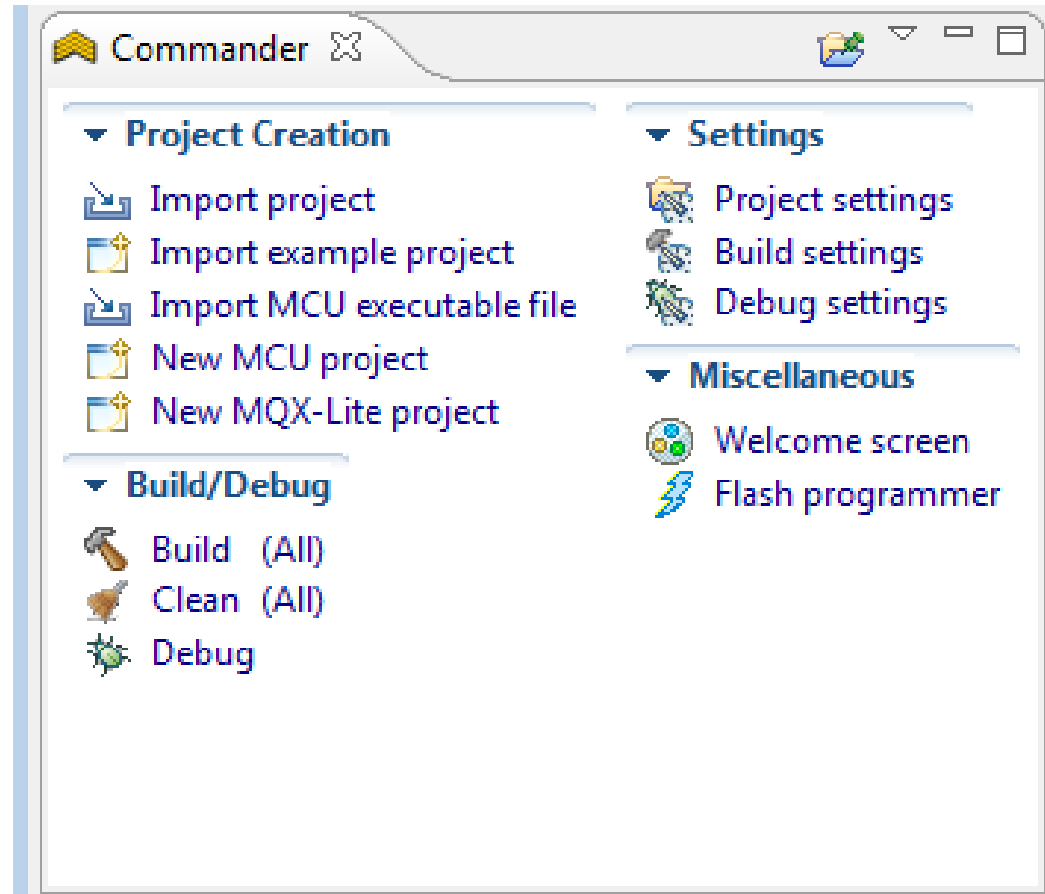
Commander View



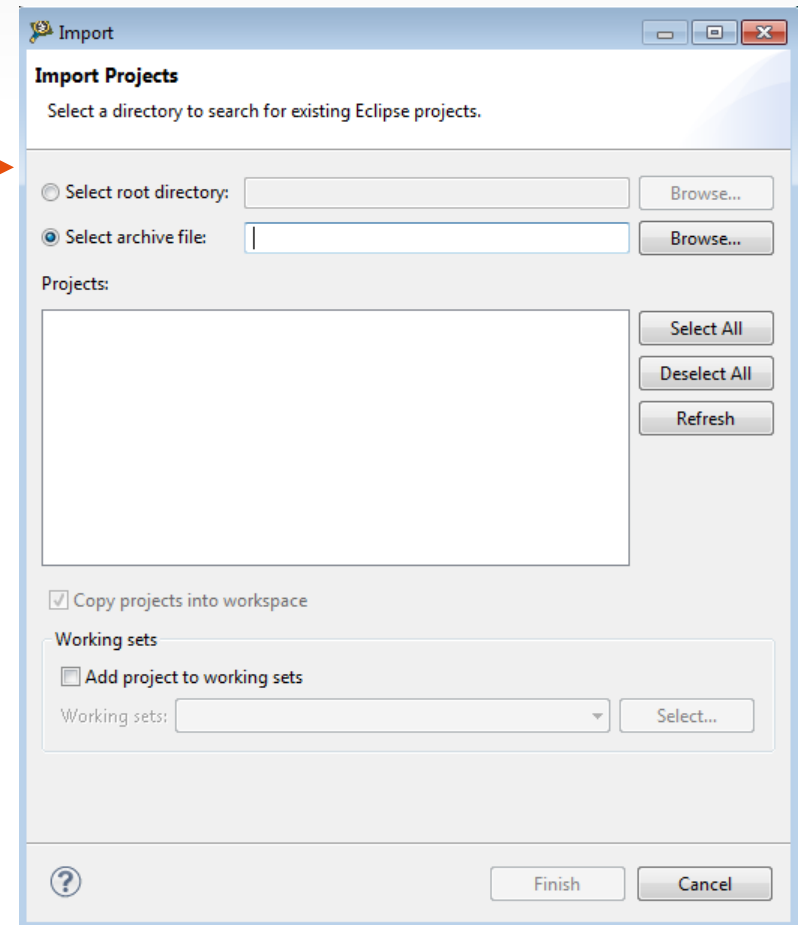
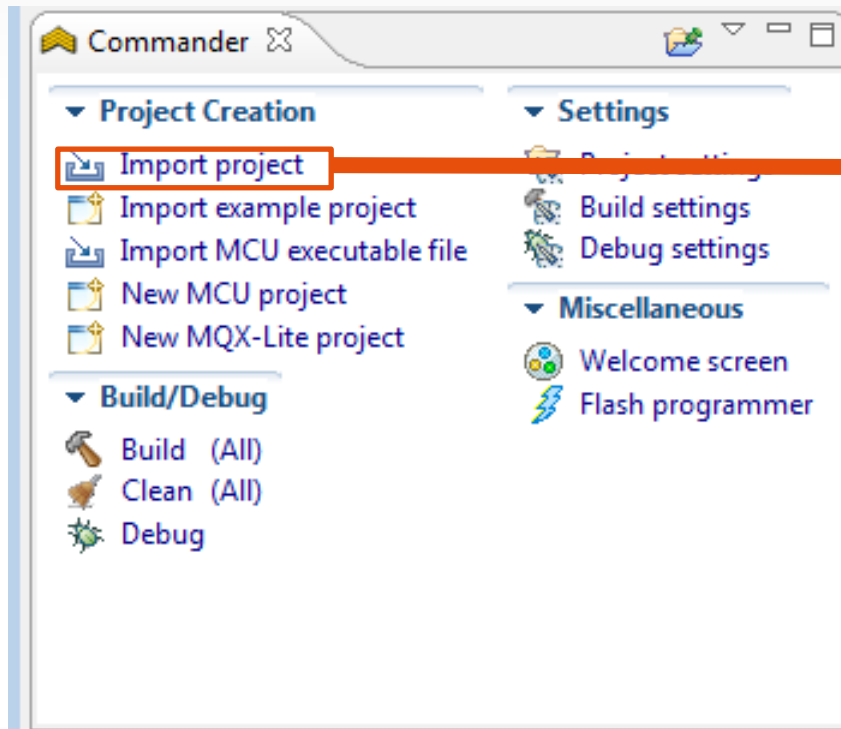
FreeScale, the Freescale logo, i.MX, eZ, C-SPYTEST, CodeWarrior, ColdFire, ColdFire+, C-WARE, the Energy Efficient Solution logo, iMxNet, mxc1010T, PEG, PowerQICC, ProFusion Express, QorIQ, QorIQcore, SafeSense, the SafeSense logo, StarCore, Symphony and Viterbo are trademarks of Freescale Semiconductor, Inc. in the U.S. Pat & Tm, Off-Arbit, Beekit, BeekStar, Clearing, Risc, Lynceus, MagiQ, M6M, Platform in a Package, QorIQ GoVersity, QorIQ Desktop, Really IP, SMARTWIS, Toss, Turbula, Vybrid and Vybrid are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © 2013 Freescale Semiconductor, Inc.

Commander View

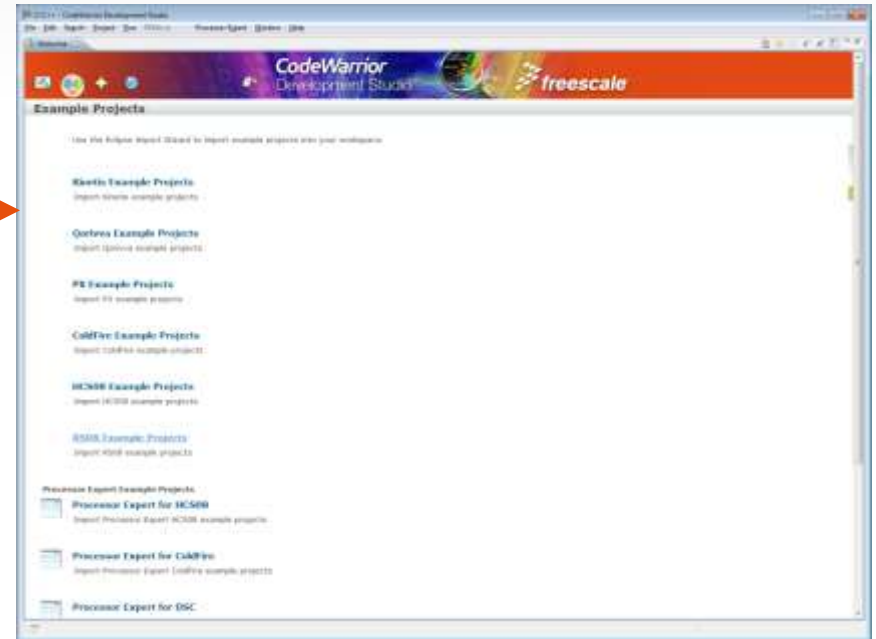
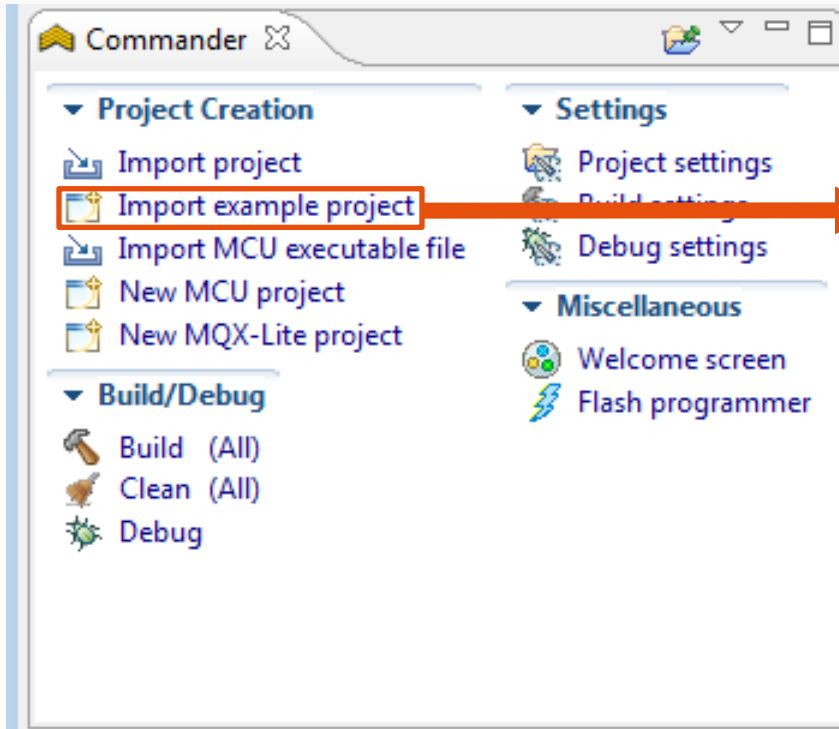
- One click access to most common operations
 - Project Creation
 - Build/Debug
 - Settings
 - Miscellaneous



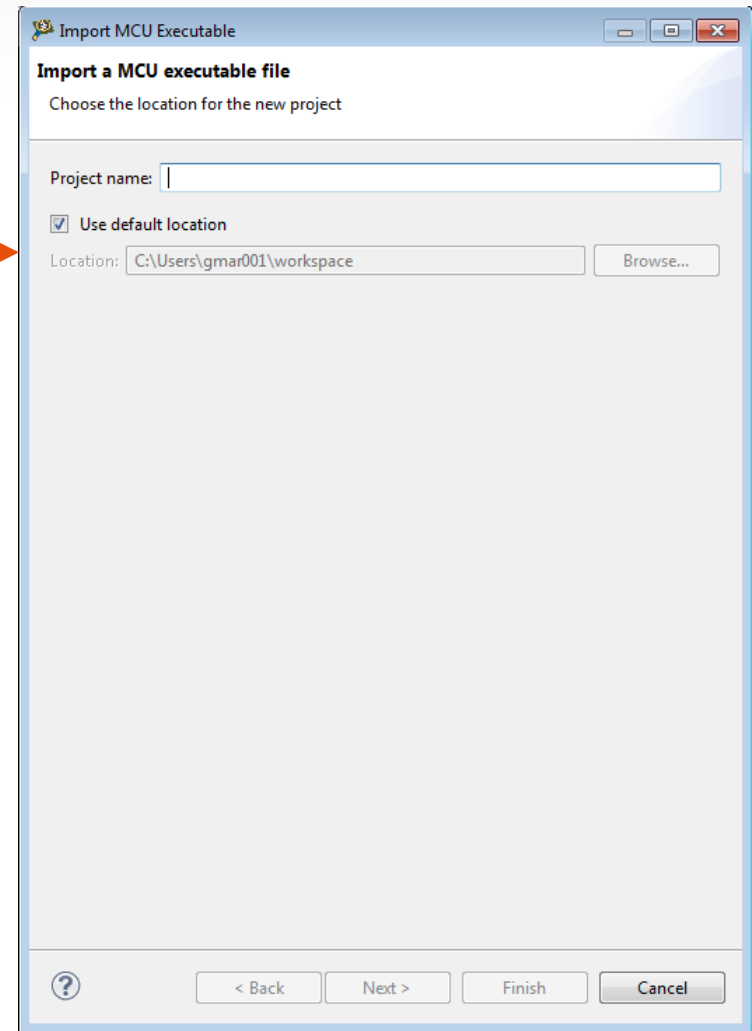
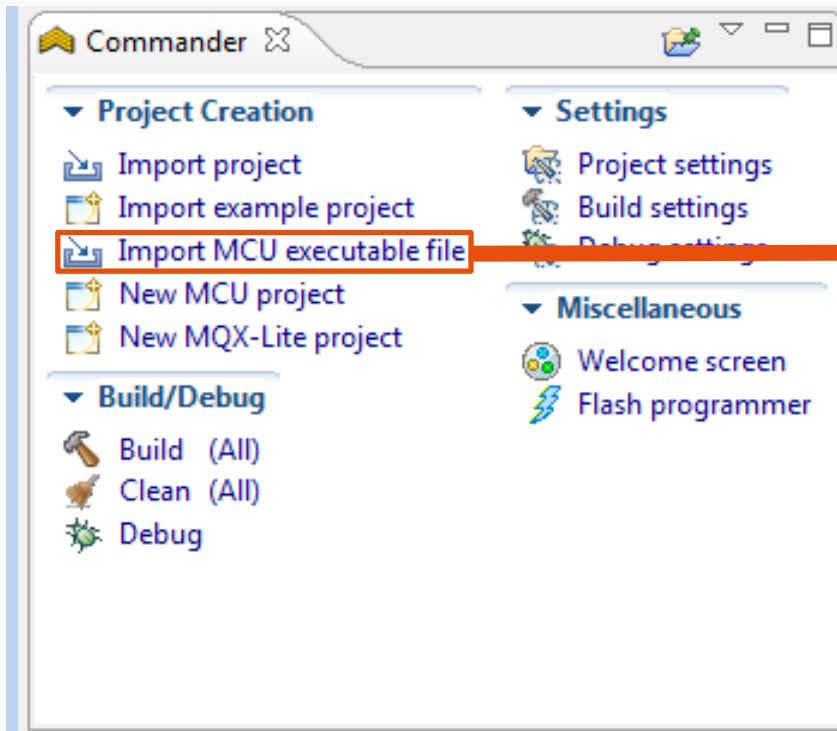
Commander View – Import Project



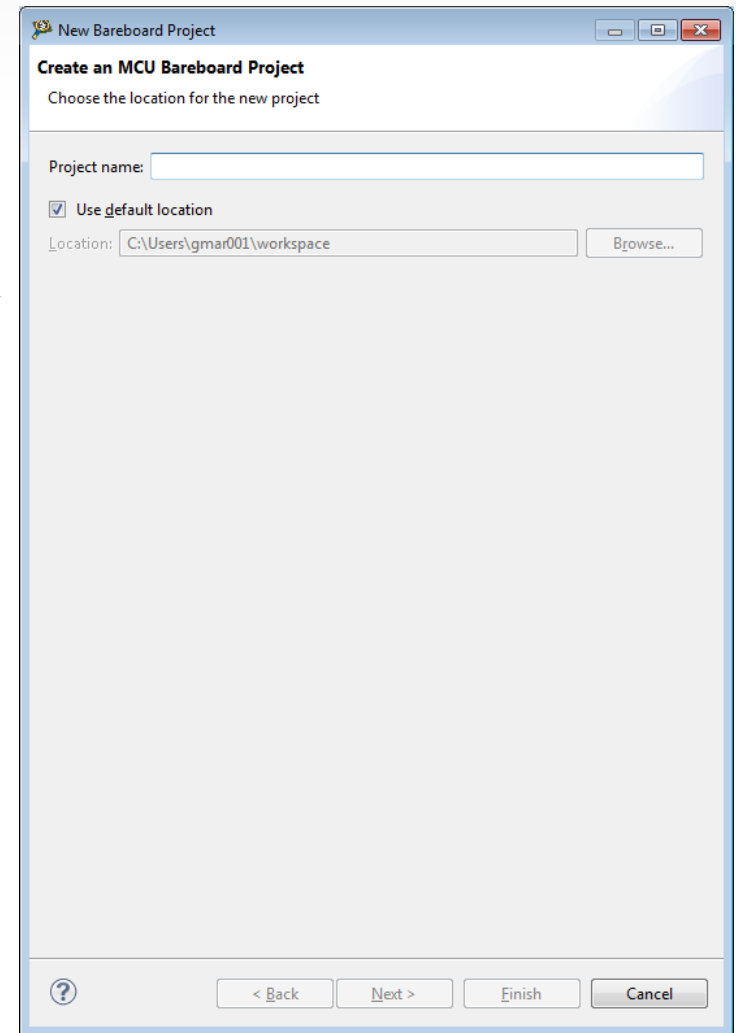
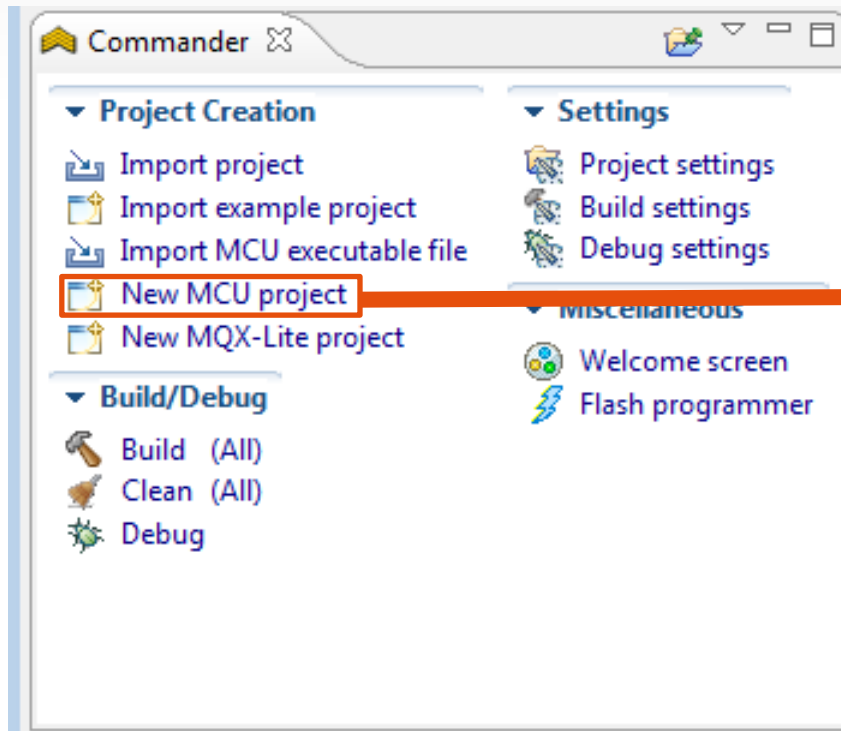
Commander View – Import Example Project



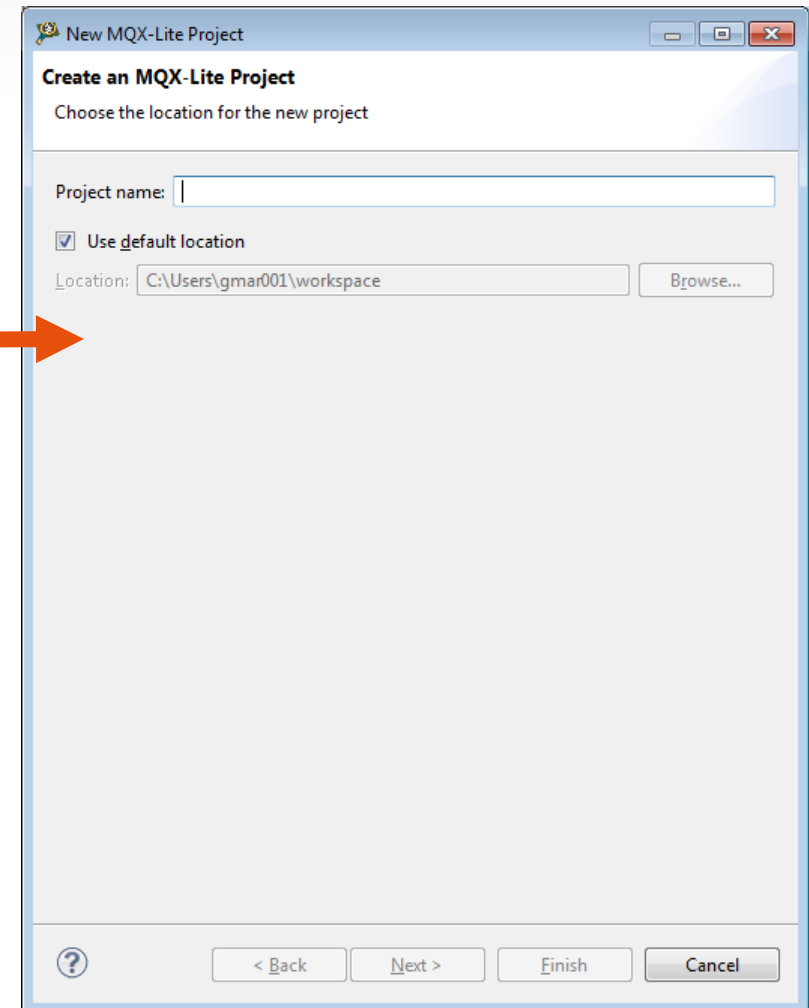
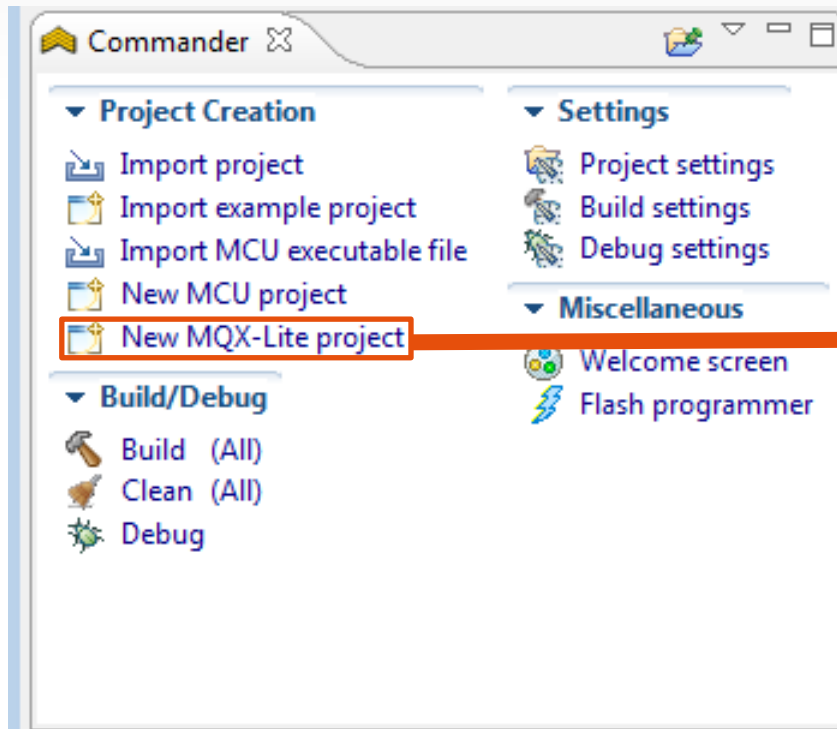
Commander View – Import MCU Executable File



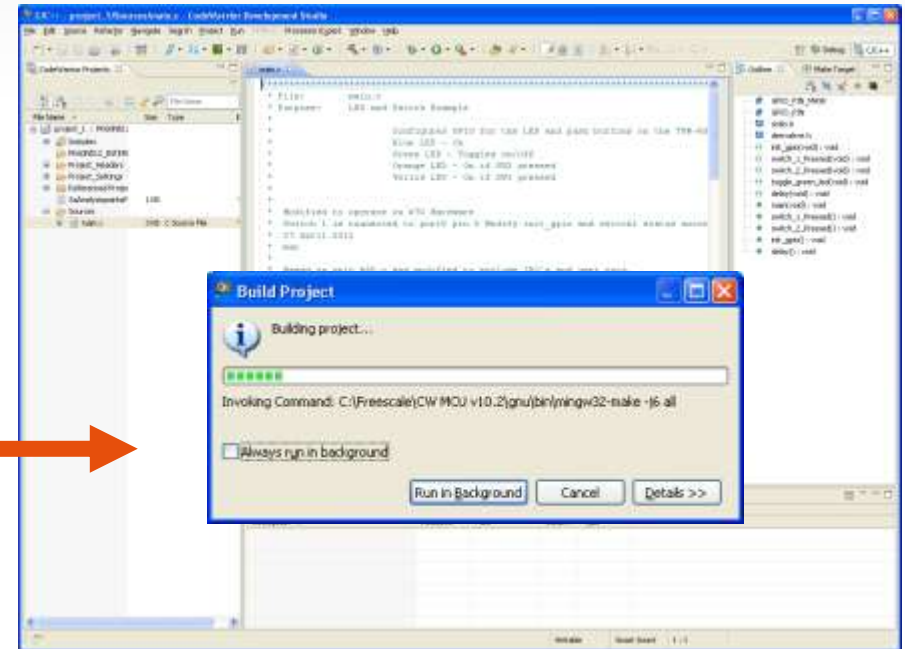
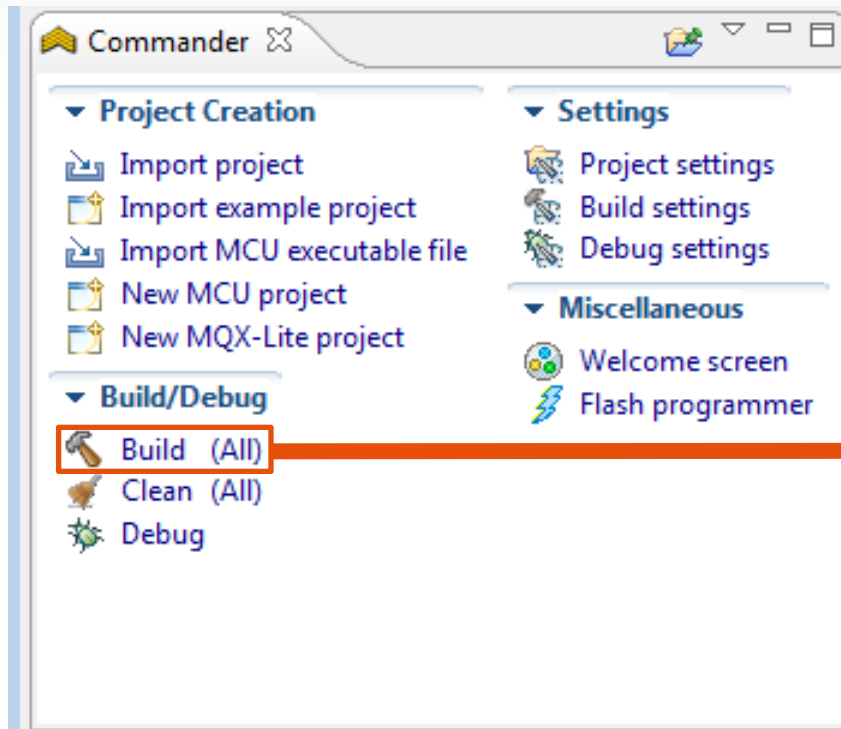
Commander View – New MCU Project



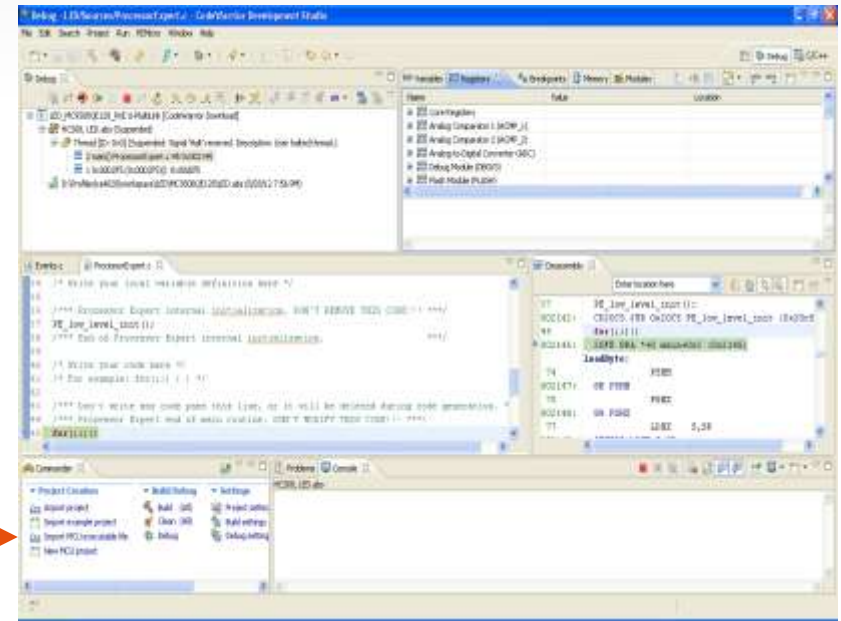
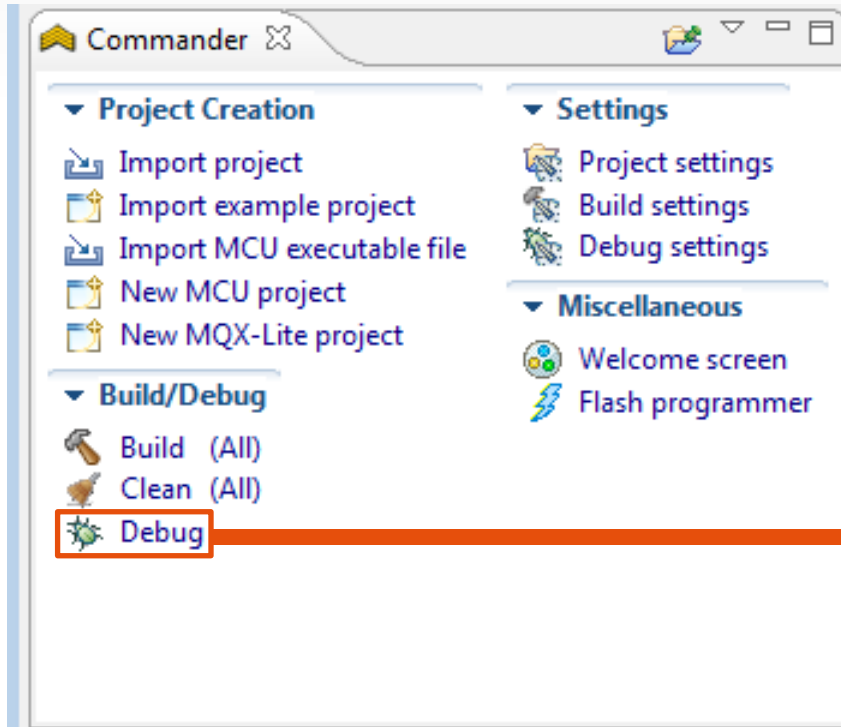
Commander View – New MQX-Lite Project



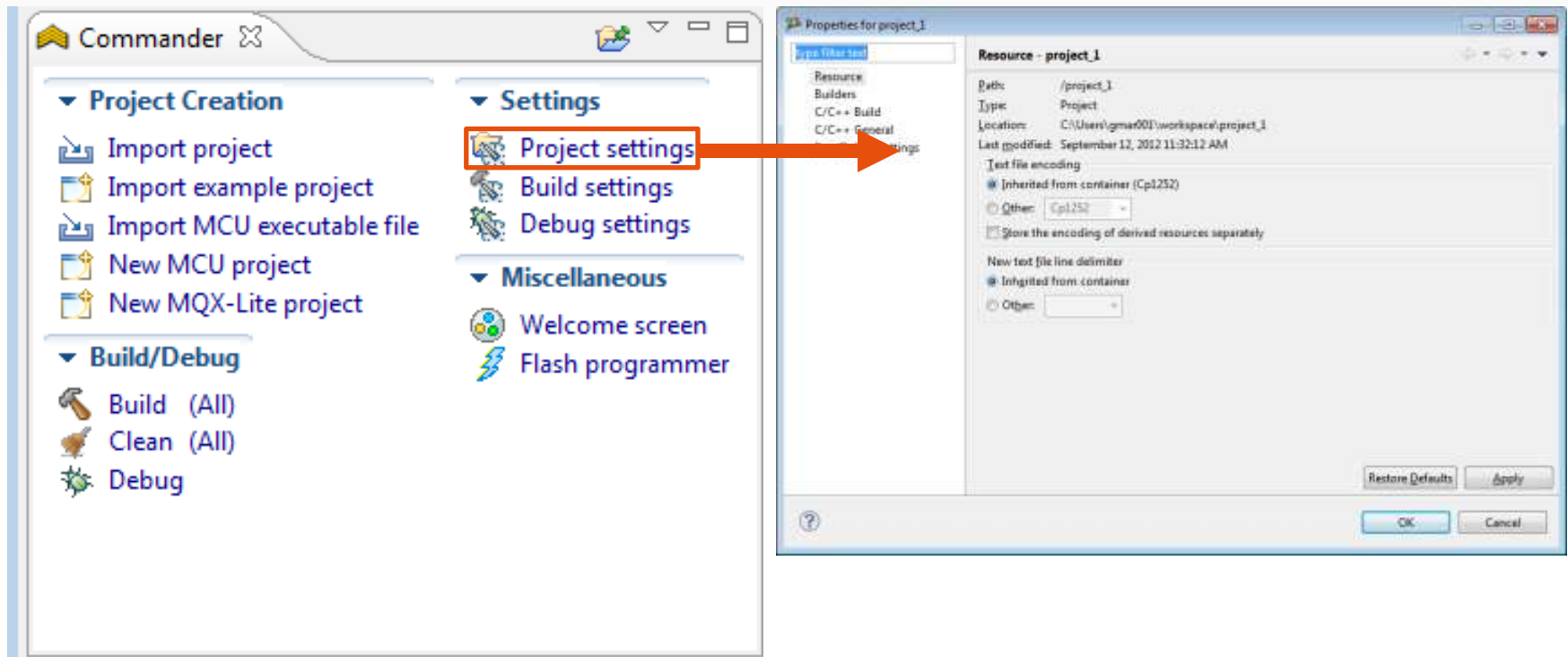
Commander View – Build



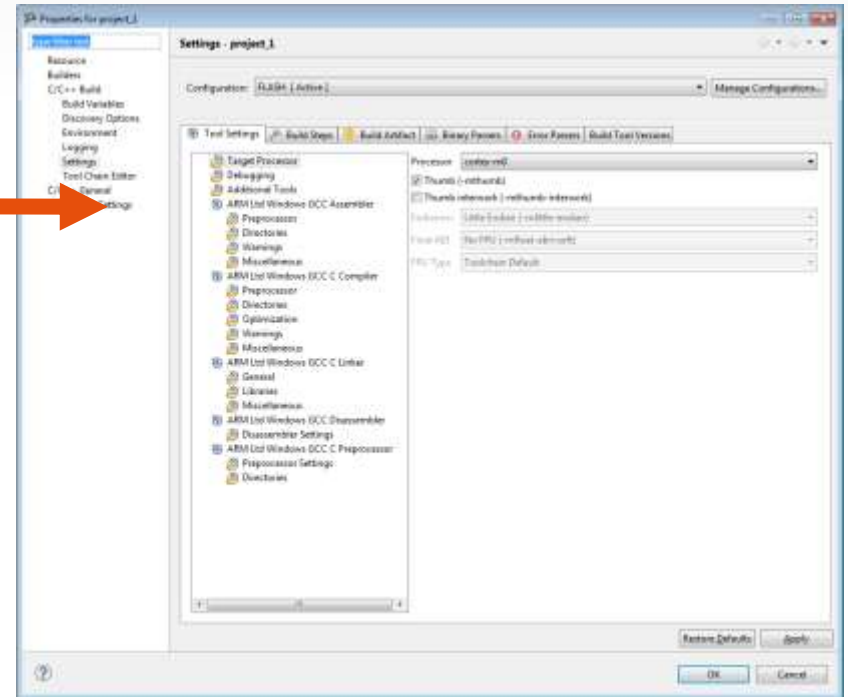
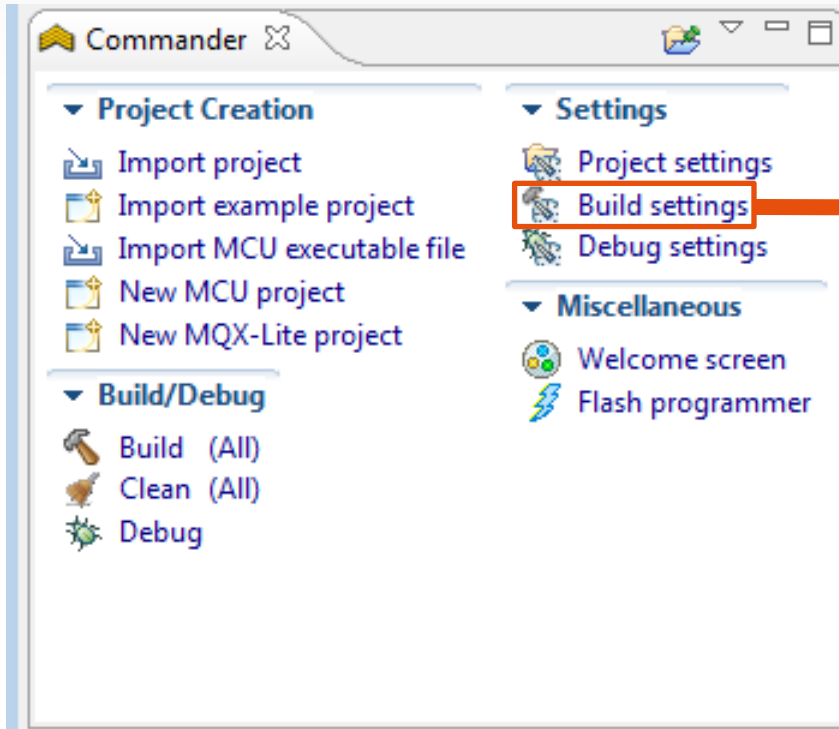
Commander View – Debug



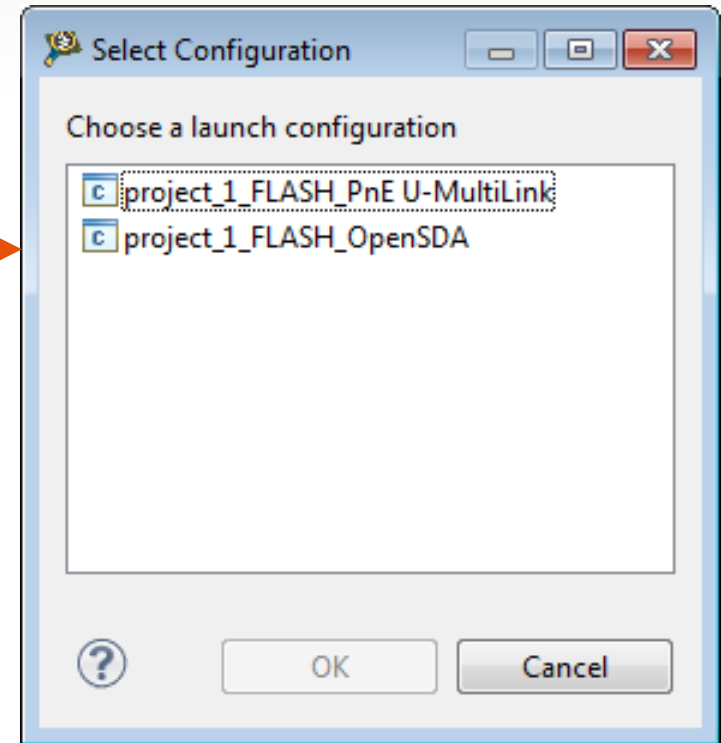
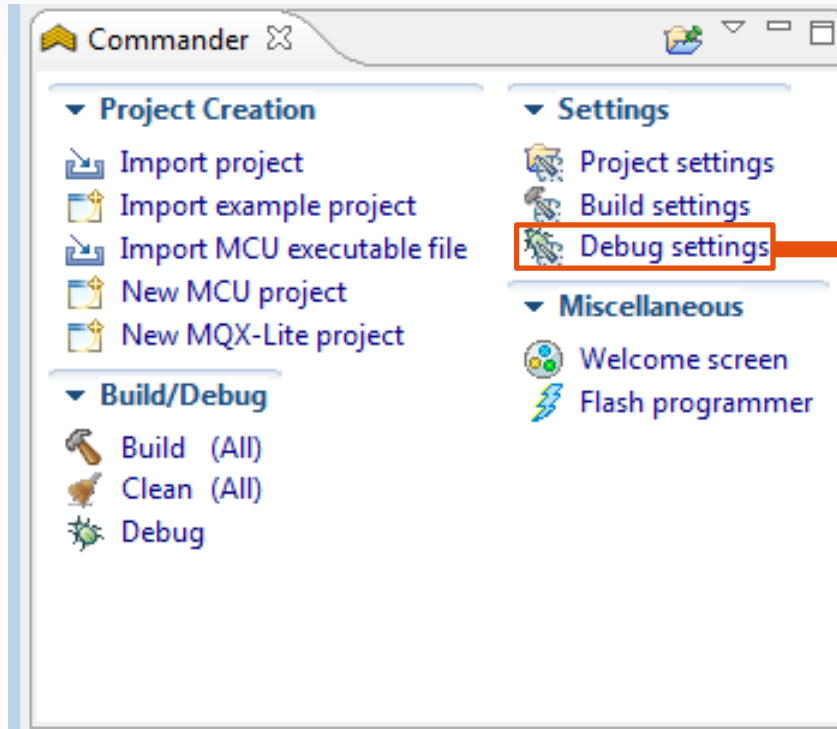
Commander View – Project Settings



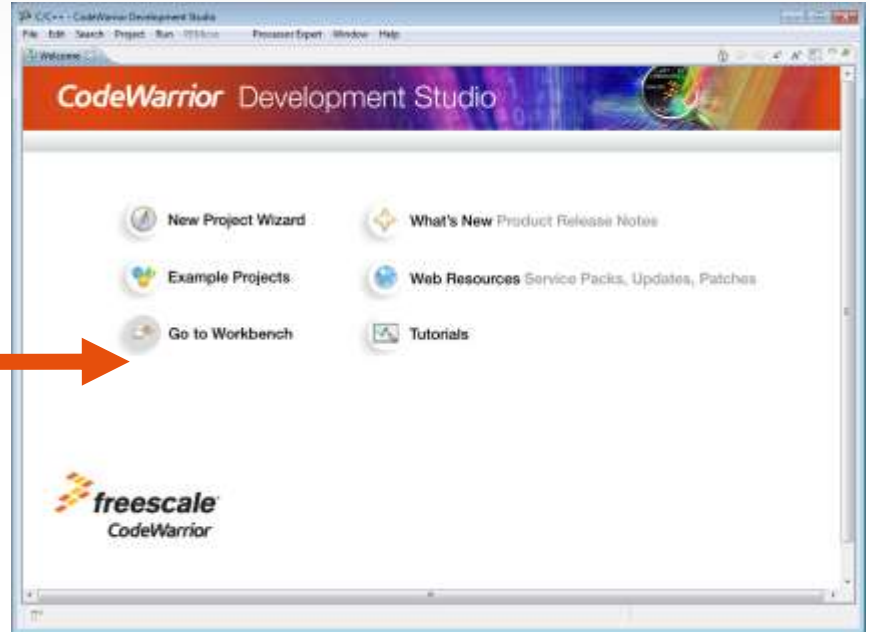
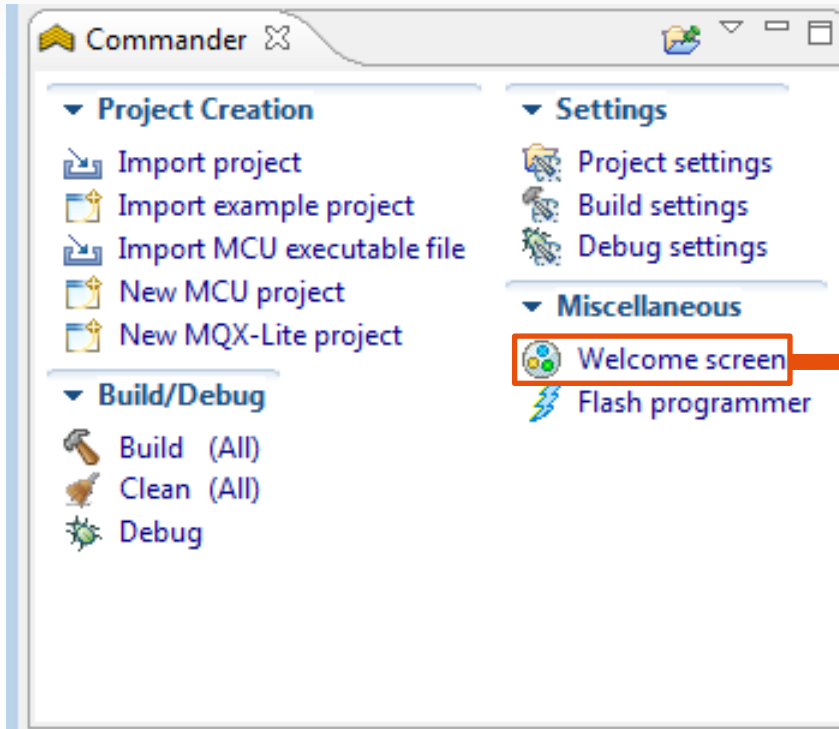
Commander View – Build Settings



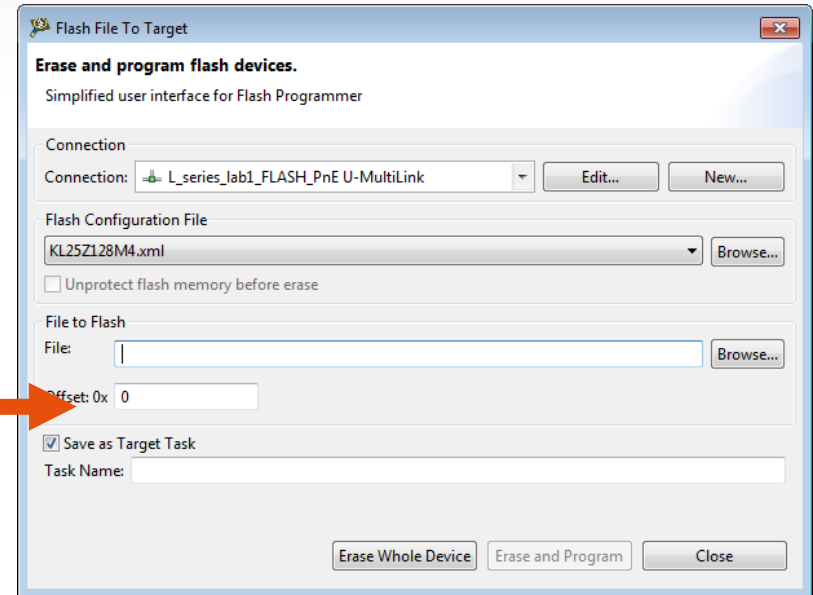
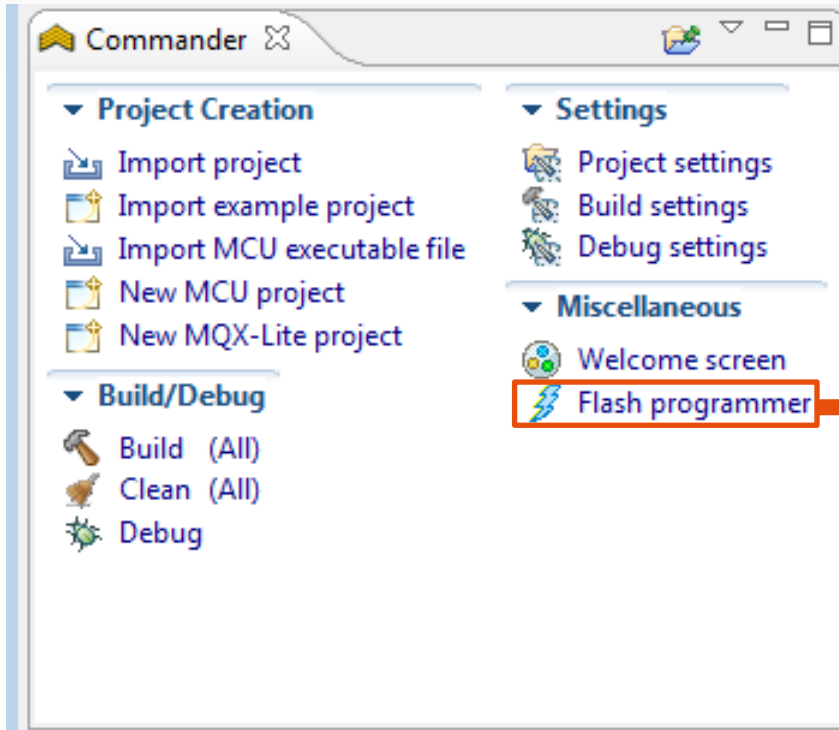
Commander View – Debug Settings



Commander View – Welcome Screen



Commander View – Flash Programmer

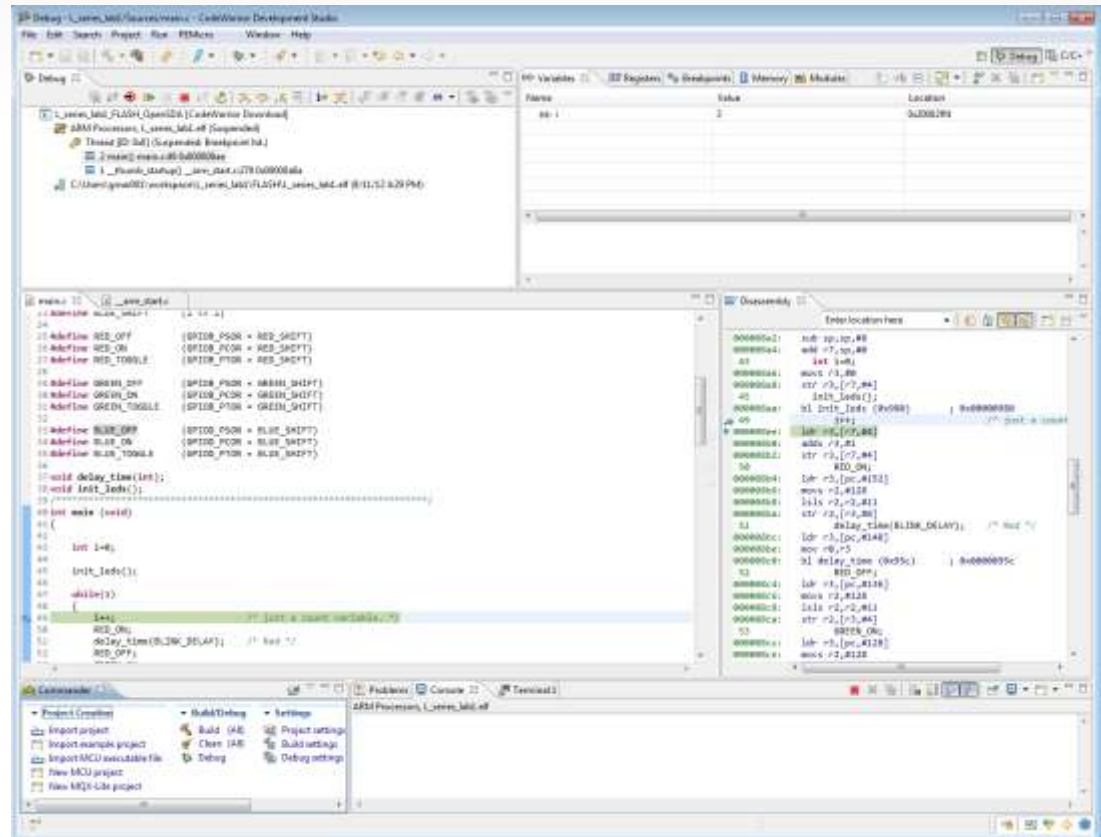


Debug Perspective



Debug Perspective

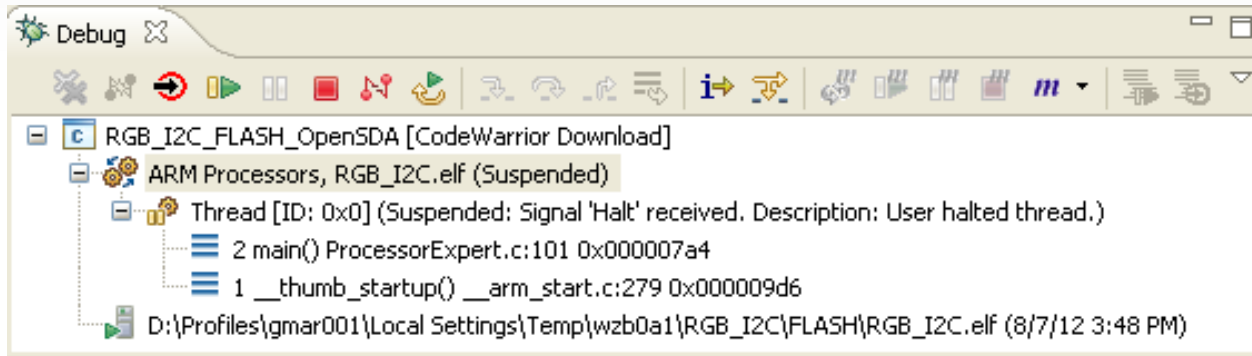
- Default views:
 - Debug View
 - Variables View
 - Breakpoints View
 - Memory View
 - Modules View
 - Editor (Area)
 - Disassembly View
 - Problems View
 - Console View
 - Commander View



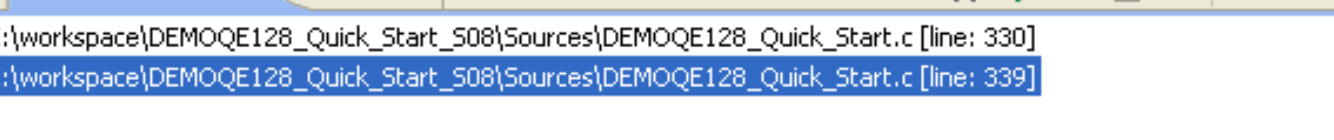
- Each view has an optional toolbar and menu that is separate from those on the main workbench window.

Debug View

- Shows the target debugging information in a tree hierarchy

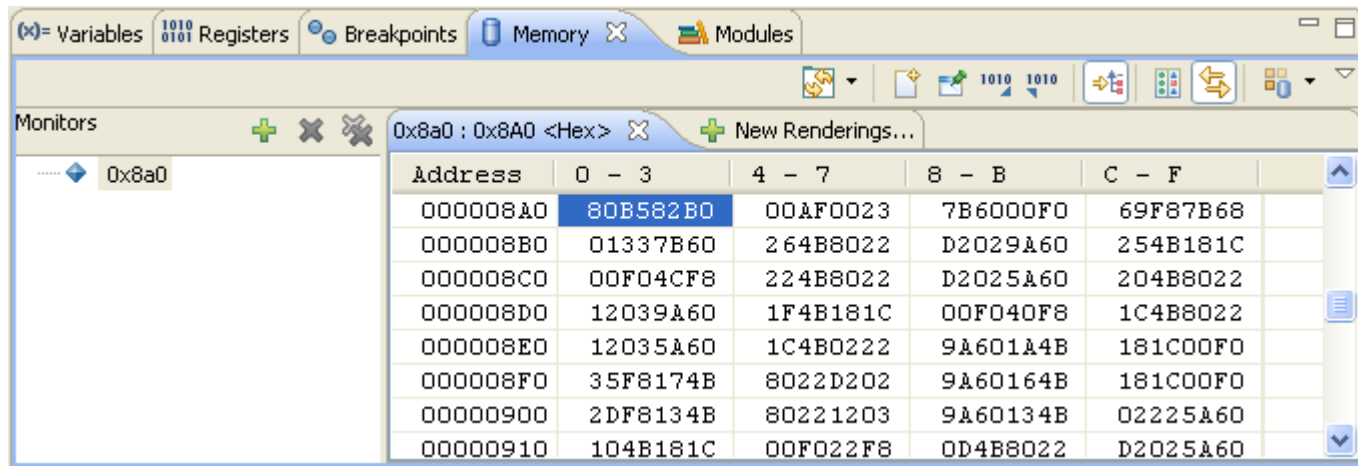


- Use this view to perform the following tasks:
 - Clear all terminated processes
 - Start a new debug session for the selected process
 - Resume execution of the currently suspended debug target
 - Halt execution of the currently selected thread in a debug target
 - Terminate the selected debug session and/or process
 - Detach the debugger from the selected process
 - Execute the current line, including any routines, and proceed to the next statement
 - Execute the current line, following execution inside a routine
 - Re-enter the selected stack frame
 - Examine a program as it steps into disassembled code

- 
- The screenshot shows the 'Breakpoints' window in Visual Studio. The window has three tabs: 'Variables', 'Breakpoints', and 'Modules'. The 'Breakpoints' tab is active. The main area displays two breakpoints for the file 'C:\workspace\DEMOQE128_Quick_Start_S08\Sources\DEMOQE128_Quick_Start.c'. The first breakpoint is at line 330 and is disabled (indicated by an unchecked checkbox). The second breakpoint is at line 339 and is enabled (indicated by a checked checkbox). The second breakpoint is highlighted with a blue selection bar.
- | Breakpoint | File | Line | Enabled |
|------------|--|------|---------|
| 1 | C:\workspace\DEMOQE128_Quick_Start_S08\Sources\DEMOQE128_Quick_Start.c | 330 | False |
| 2 | C:\workspace\DEMOQE128_Quick_Start_S08\Sources\DEMOQE128_Quick_Start.c | 339 | True |

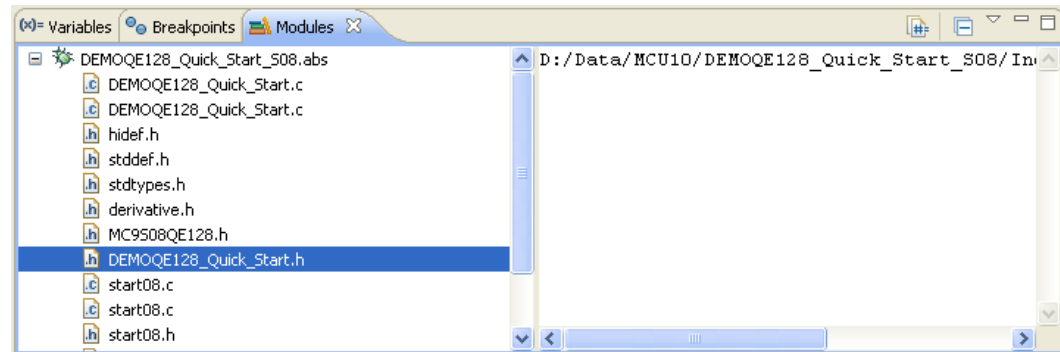
Memory View

- Allows memory to be monitored and modified
- Contains two panes:
 - Monitors panel - Displays the list of memory monitors added to the debug session currently selected in the Debug view
 - Renderings panel - Displays memory renderings.
 - Content is controlled by the selection in the Monitors panel.
 - Can be configured to display two renderings simultaneously.



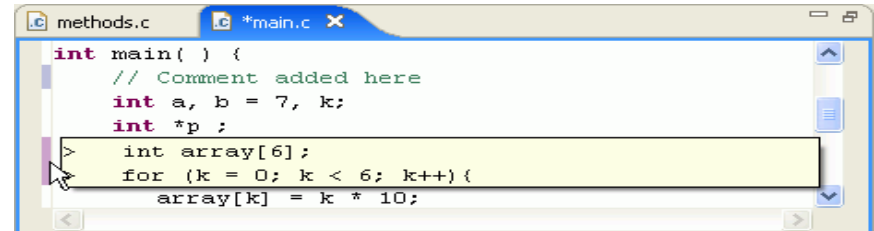
Modules View

- Displays information about the modules loaded in the current debug session
 - Executables
 - Shared libraries
- View consists of two areas:
 - Modules tree
 - Detail pane
- Detail pane displays the detail information for the module selected in the modules tree
- Expanding a module enables users to view the module's internals
 - Functions
 - Global variables
 - Associated source files



Editor Area

- There are multiple editors for multiple file types
 - C/C++ editor for source code
 - Text editor for text files
- To open a file:
 - Double click on the file in CodeWarrior Projects or Navigator view
 - or -
 - Select File > Open File
- Each file opens up in a new tab in editor area
- Editors show changes in files since last save
 - See the change bar (left side of text)
 - Additions and modifications color coded.
 - Hover cursor over change bar to see previous text
 - Dirty file – indicated by * in editor tab
- Marker area on left side of editor indicates task, bookmarks, or breakpoint



Disassembly View

- Shows the loaded program as assembler instructions mixed with source code for comparison
 - The currently executing line is indicated by an arrow marker and highlighted in the view
 - The following tasks can be performed:
 - Set breakpoints at the start of any assembler instruction
 - Enable and disable breakpoints and set their properties
 - Step through the disassembly instructions of the program
 - Jump to specific instructions in the program
- The screenshot shows a disassembly window with two tabs: 'Outline' and 'Disassembly'. The 'Disassembly' tab is active, displaying a list of assembly instructions with their addresses and hex values. The instruction at address 0x00002432 is highlighted with a blue arrow marker. Below the assembly instructions, there is a section of source code in C, which includes comments and variable declarations. The source code is partially visible, showing a function 'if (some_key_pressed && key_press_debounced) {' and a block of code that sets 'tempA = PTAD;' and 'tempD = PTDD;'.

```

if (some_key_pressed && key_press_debounced) {
0x0000242f <main+48>: C60080 LDA 0x0080
* 0x00002432 <main+51>: 2603 BNE *+5 main+0x30 (0x2437)
0x00002434 <main+53>: CC2522 JMP 0x2522 main+0x123 (0x2522)
0x00002437 <main+56>: C60081 LDA 0x0081
0x0000243a <main+59>: 2603 BNE *+5 main+0x40 (0x243f)
0x0000243c <main+61>: CC2522 JMP 0x2522 main+0x123 (0x2522)

tempA = PTAD; // get the key pressed
0x0000243f <main+64>: B600 LDA 0x00
0x00002441 <main+66>: 95 TSX
0x00002442 <main+67>: E701 STA 1,X
tempD = PTDD; // get the key pressed
0x00002444 <main+69>: BE06 LDX 0x06
0x00002446 <main+71>: 9EEF01 STX 1,SP

```

The screenshot shows the Disassembly window of a debugger. The title bar includes tabs for 'Outline' and 'Disassembly'. The assembly code is displayed in a list with addresses, disassembly, and comments. Comments are in blue text. The code implements a key press debouncing routine.

```

if (some_key_pressed&&key_press_debounced) {
0x0000242f <main+48>: C60080 LDA 0x0080
0x00002432 <main+51>: 2603 BNE *+5 main+0x30 (0x2437)
0x00002434 <main+53>: CC2522 JMP 0x2522 main+0x123 (0x2522)
0x00002437 <main+56>: C60081 LDA 0x0081
0x0000243a <main+59>: 2603 BNE *+5 main+0x40 (0x243f)
0x0000243c <main+61>: CC2522 JMP 0x2522 main+0x123 (0x2522)

    tempA = PTAD; // get the key pressed
0x0000243f <main+64>: B600 LDA 0x00
0x00002441 <main+66>: 95 TSX
0x00002442 <main+67>: E701 STA 1,X
    tempD = PTDD; // get the key pressed
0x00002444 <main+69>: BE06 LDX 0x06
0x00002446 <main+71>: 9EEF01 STX 1,SP

```

Problems View

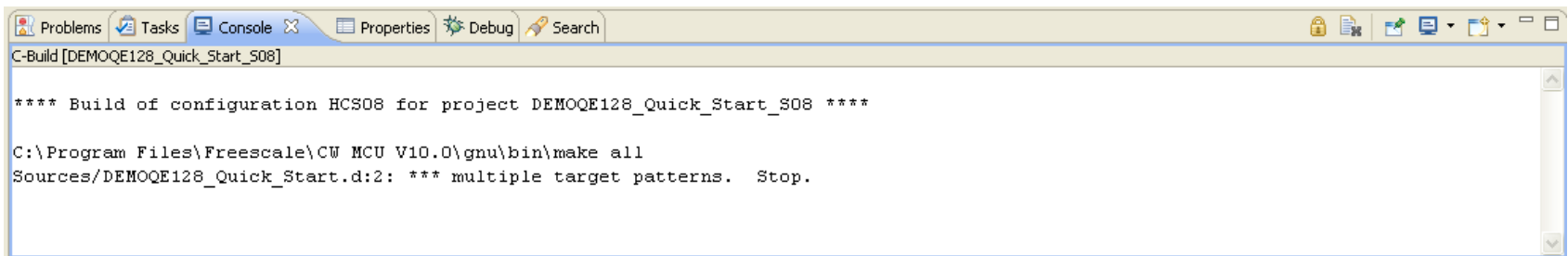
- Shows errors and warnings
 - Filters on problems to display:
 - Location of defective resource
 - Type of defect
 - Type of problem
 - Double click an error/warning message to navigate to offending source code

3 errors, 2 warnings, 0 infos

Description	Resource	Path	Location
Errors (3 items)			
Error: declaration syntax error	sample.h	DebugHello	line 7
ERROR: Non-zero return status from "C:\(DebugHello			Unknown
Error: undefined identifier 'xyz'	mymain.c	DebugHello	line 12
Warnings (2 items)			
Warning: variable / argument 'argc' is not	main.c	HelloAgain	line 6
Warning: variable / argument 'argv' is not	main.c	HelloAgain	line 6

Console View

- Displays standard I/O
- Echoes Make files
- Displays program output
- Console toolbar:
 - Scroll Lock
 - Clear Console
 - Pin Console (save this console in separate view)
 - Display Selected Console
 - Open Console



```

C-Build [DEMOQE128_Quick_Start_S08]

**** Build of configuration HCS08 for project DEMOQE128_Quick_Start_S08 ****

C:\Program Files\Freescale\CW MCU V10.0\gnu\bin\make all
Sources\DEMOQE128_Quick_Start.d:2: *** multiple target patterns. Stop.
    
```

