HANDS-ON WORKSHOP: GENERAL PURPOSE MCU S32K DEEP DIVE AND S32 SDK

MATHIEU, CLAIN AND KUSHAL, SHAH
PRODUCT MARKETING & APPLICATION ENGINEERING
AUTOMOTIVE MICROCONTROLLER AND PROCESSORS

AMF-DES-T2718 | JUNE 2017

SECURE CONNECTIONS
FOR A SMARTER WORLD

AGENDA

- GPIS PL Overview
- S32K Product Features
  - Product overview
  - Enablement
- Hands-On
  - Introduction (S32 SDK + S32 DS)
  - GPIOs Lab
  - Clocks Lab
  - Interrupts Lab

NXP AMP

Safe, Secure & Reliable Portfolio

Automotive Microcontroller and Processor

AMP Positioning

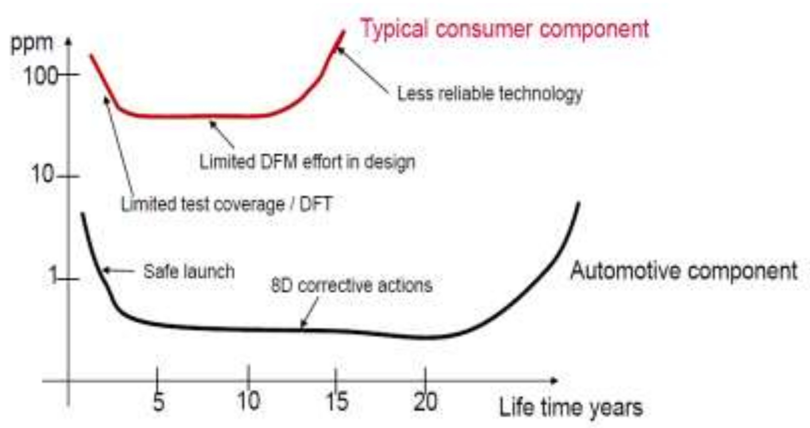
Safety

Soft error
Fault tolerant system
Functional safety

Security

Detection
Encryption-Authentication
Secure communication

Reliability



Quality
Longevity
High temperature

NXP AMP Product Line Introduction

ADAS
(Advanced Driver Assistance Systems)

Radar, LIDAR
Vision
Sensor Fusion

- #1 in Radar with strong IP and system knowledge
- High performance low power accelerators
- Scalable high performance roadmap for central processing

Products:
- S32R - Radar
- S32V - Vision
- S32A – Safe Autonomous Systems

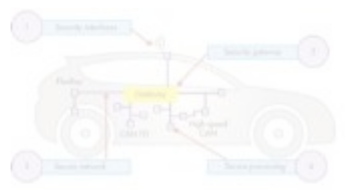
GPIS
(General Purpose & Integrated Solutions)

Body Electronics
Edge Nodes

- 500+ customers
- Broadest portfolio of integrated MCU+HV mixed-signal solutions
- Complete Tools & Software enablement

Products:
S08/S12/PPC → ARM
KEA – S32K
S12 MagniV – S32M

C&S
(Connectivity & Security)

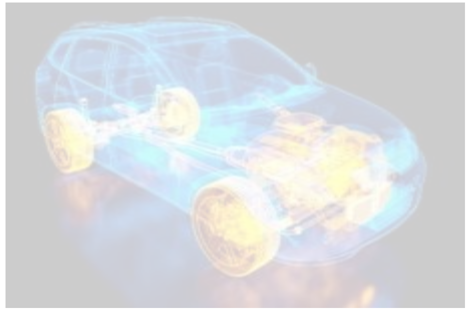


Gateway

- #1 in Vehicle Networking with leading networking and security IP
- #1 in Automotive HW Security with Strong IP and broad portfolio
- End to end portfolio of networking devices (MCU/MPU, TX/RX)

Products:
MPC564xB/C
MPC574xG
S32G

VDS
(Vehicle Dynamics & Safety)

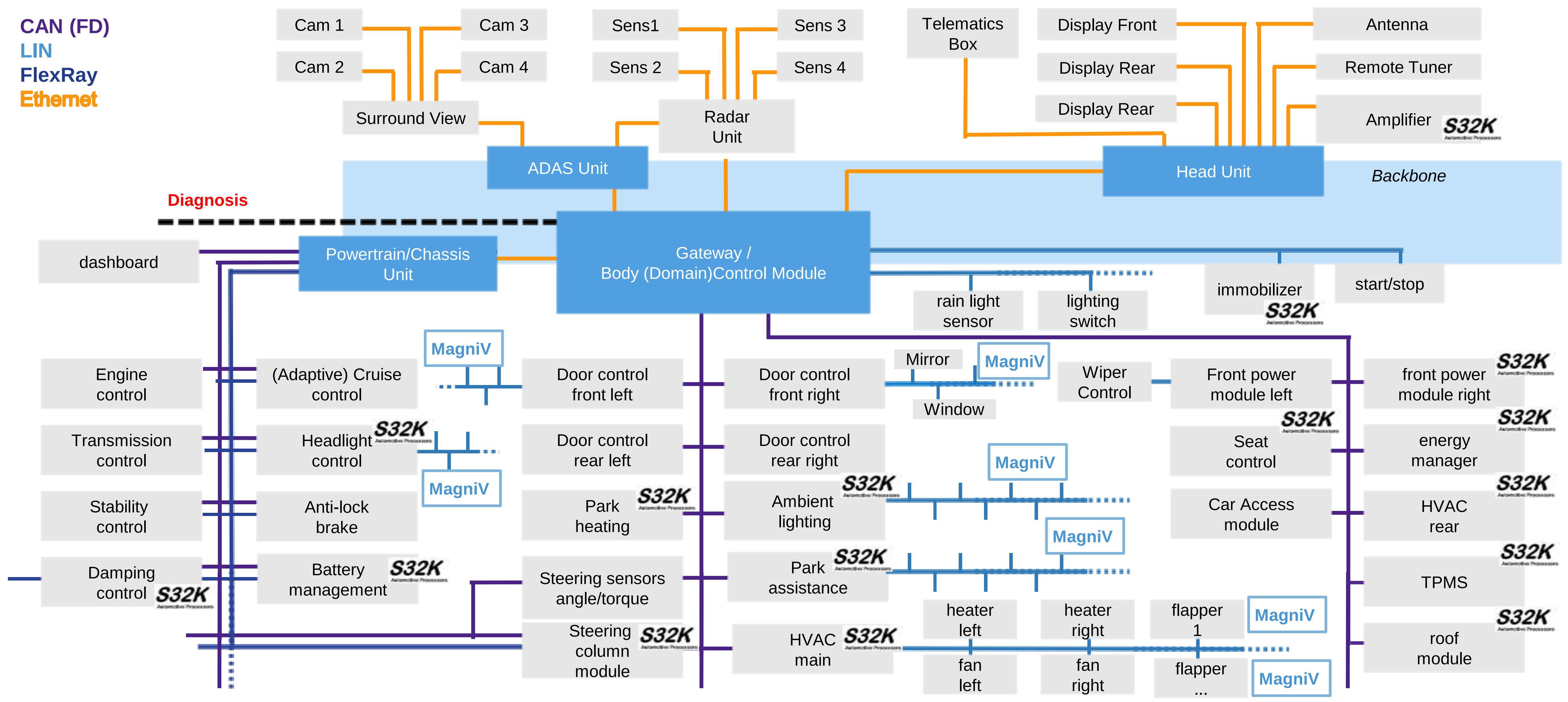


Chassis & Safety
Powertrain & Hybrid/EV

- Long term Innovator in Chassis and Powertrain Control.
- Significant Growth in Safety as Autonomous Control Drives Robust Fault Tolerant Systems

Products:
MPC56xx
MPC57xx
S32S/P/H

In-Vehicle Nodes Map

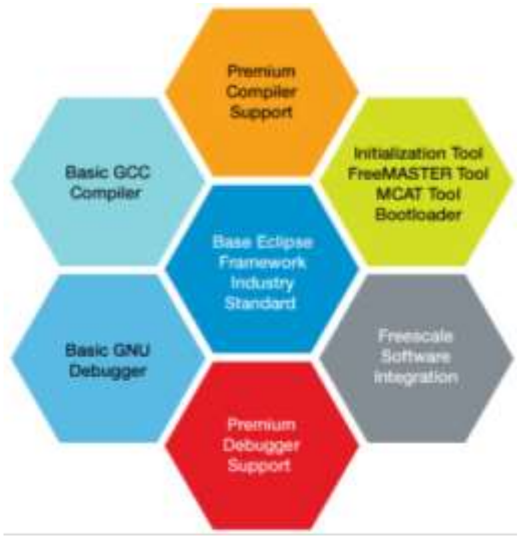


S32K Product Features

S32K - Automotive ARM Cortex MCUs

- **Hardware**
  - High performance ARM Cortex-M0+/4 cores
  - Low power operating modes & peripherals – 25µA in VLPS
  - Scalability from 8KB to 2MB, ECC, 16-176 pins
  - Hardware Security, ISO CAN-FD, Ethernet, FlexIO, Audio …
  - ISO26262 ASIL-B compliant
  - AEC Q100, 125°C, 15 years minimum longevity

- **Enablement**
  - Free S32 Design Studio IDE, low cost demo boards
  - SDK with FreeRTOS & low-level drivers
  - AUTOSAR, MCAL and OS
  - Large third-party ecosystem
  - Model-based design support


Premium Compiler Support
Basic GCC Compiler
Initialization Tool FreeMASTER Tool MCAT Tool Bootloader
Base Eclipse Framework Industry Standard
Basic GNU Debugger
Freescale Software Integration
Premium Debugger Support

General Purpose Applications

ASIL-B

Body control module

Human machine interface

Battery Management

Tire pressure monitoring receiver

Steering column lock

Wireless charging

Climate control

Door/Window/sunroof

Lighting

Secure transmission / encryption

Over the Air

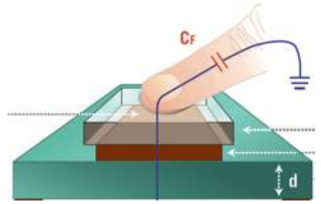
Firmware update

Near Field Communication

S32W

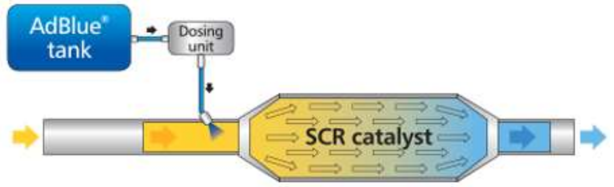
Chassis systems

PMSM/BLDC motor control



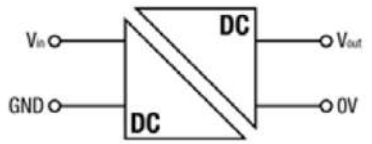
Touch sensing

Park assist



NOx reduction systems

Motorbike ECU/ABS

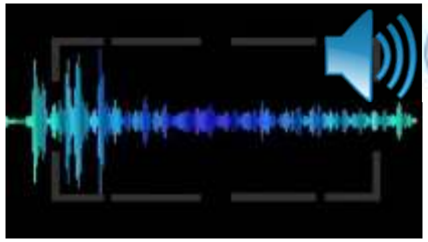


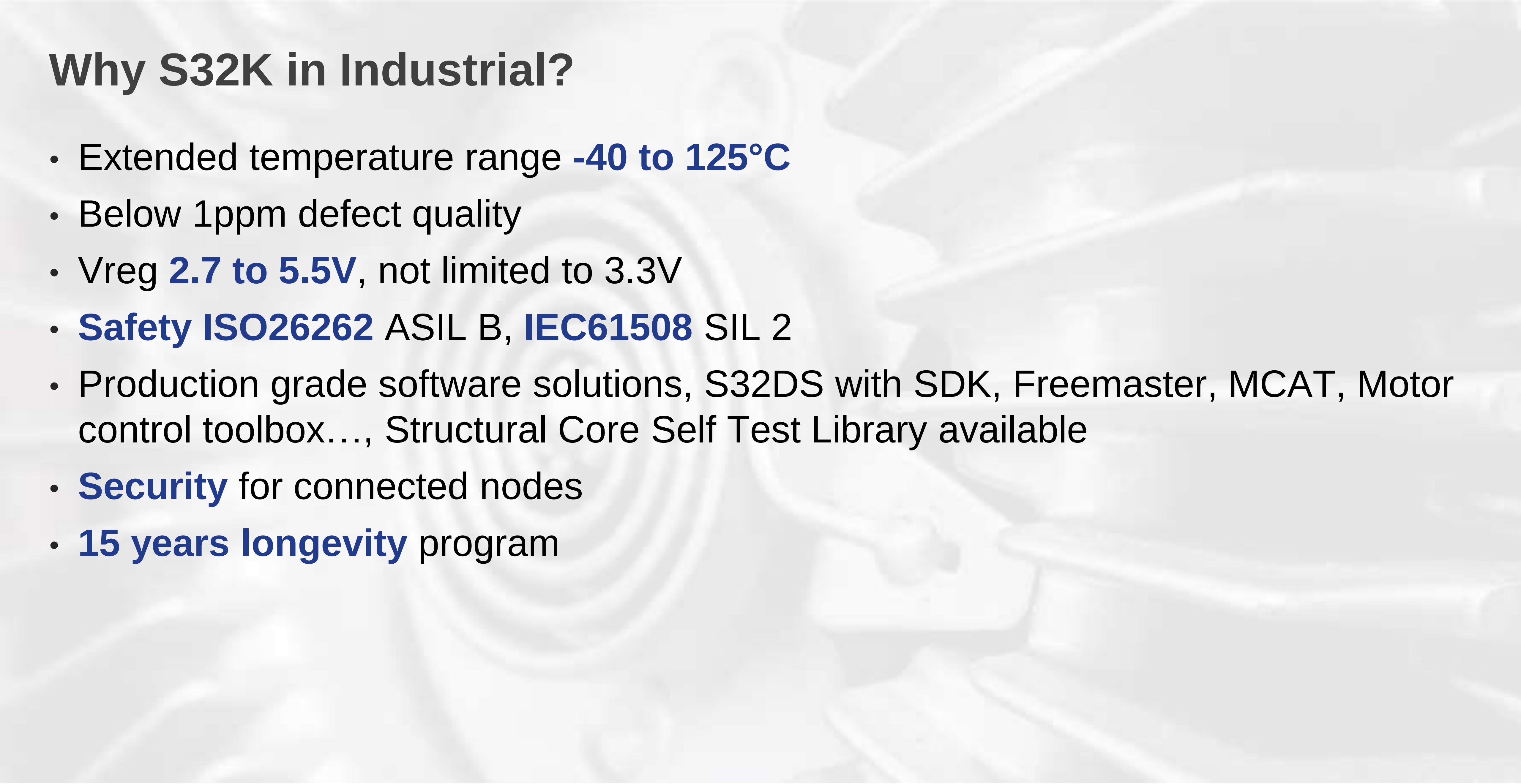
DC/DC converters

E-shifter

Rear view camera tilt

Steering wheel electronics

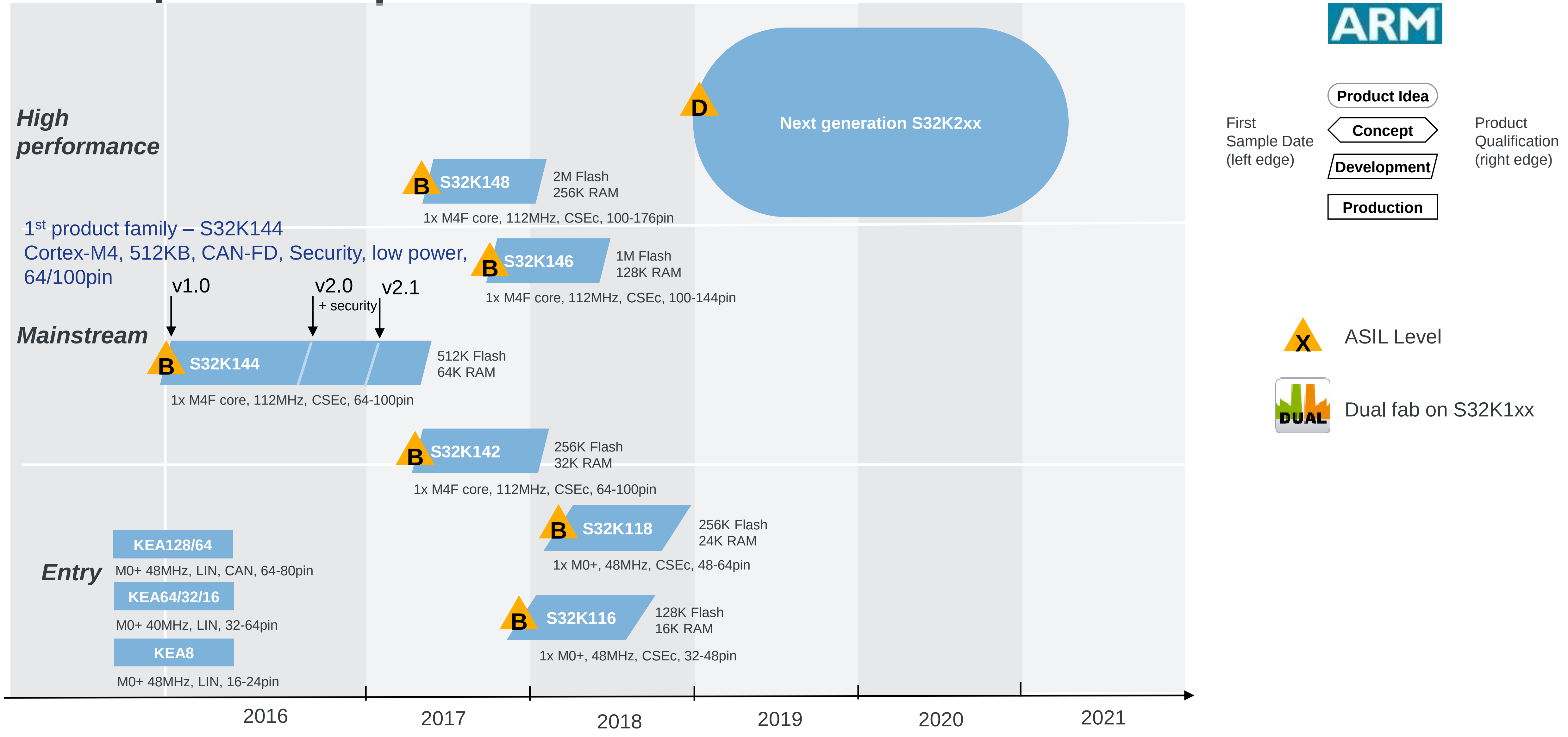

Ethernet audio amplifier



Why S32K in Industrial?

- Extended temperature range **-40 to 125°C**
- Below 1ppm defect quality
- Vreg **2.7 to 5.5V**, not limited to 3.3V
- **Safety ISO26262** ASIL B, **IEC61508** SIL 2
- Production grade software solutions, S32DS with SDK, Freemaster, MCAT, Motor control toolbox…, Structural Core Self Test Library available
- **Security** for connected nodes
- **15 years longevity** program

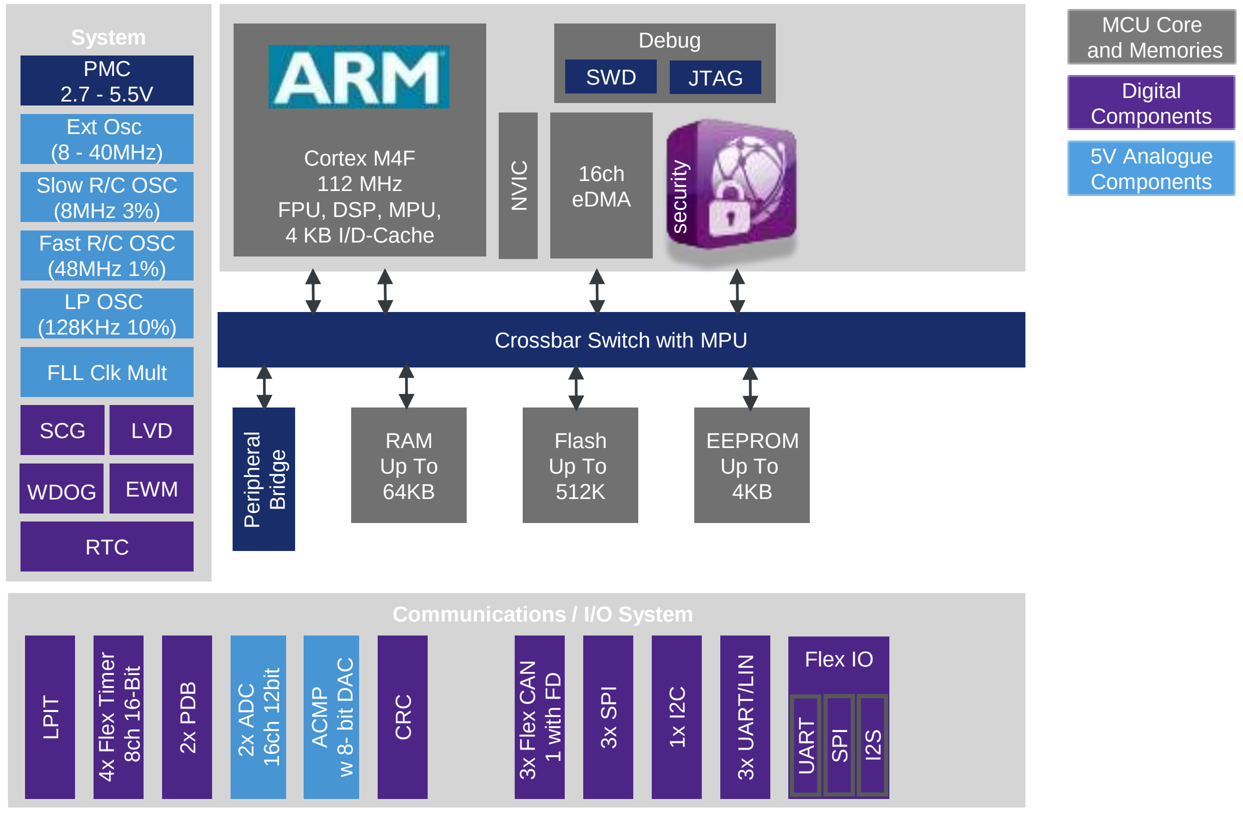
S32K | Roadmap



subject to change

S32K144 Block Diagram

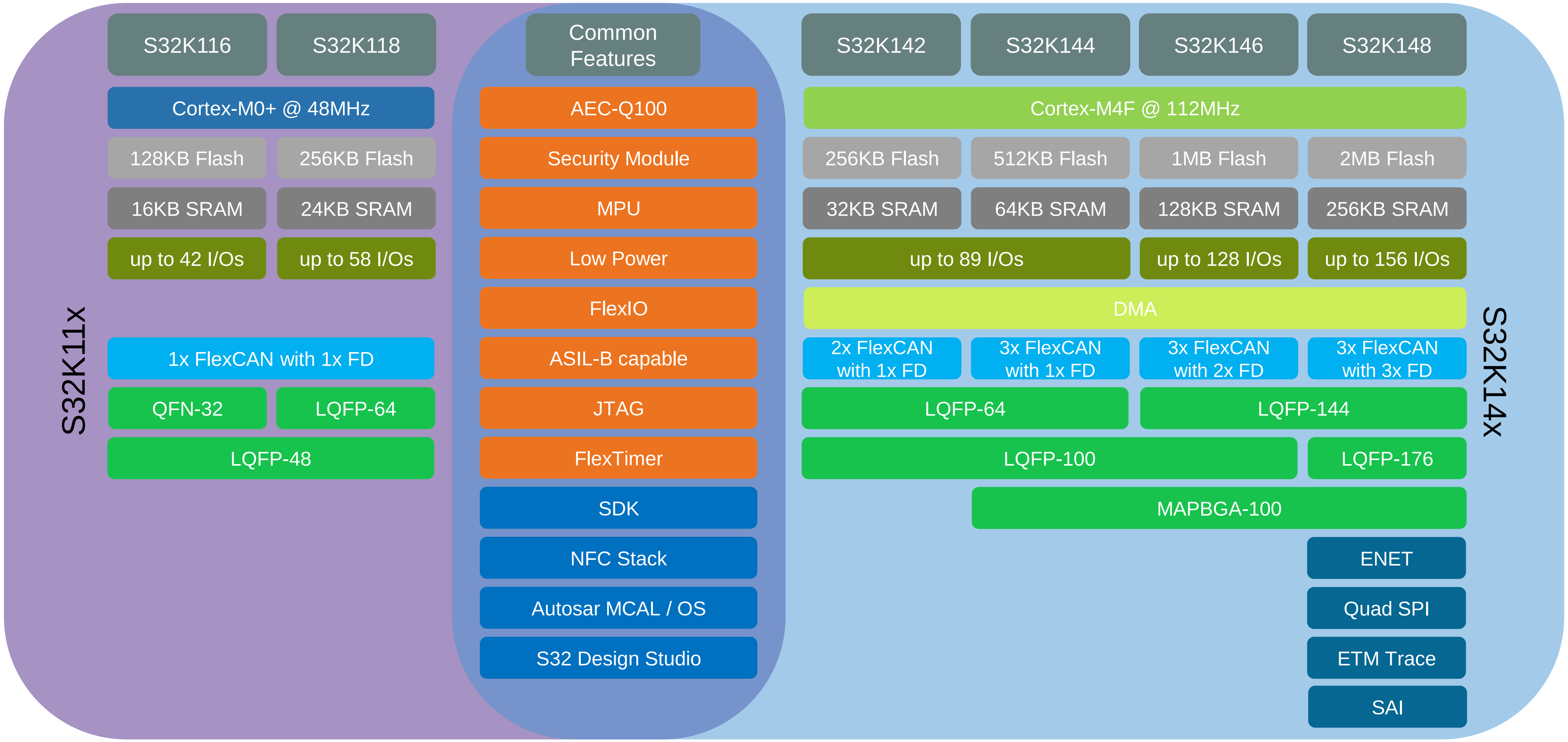
- **High performance**
  - ARM Cortex M4F up to 112MHz w FPU
  - eDMA from 57xxx family
- **Software Friendly Architecture**
  - High RAM to Flash ratio
  - Independent CPU and peripheral clocking
  - 48MHz 1% IRC – no PLL init required in LP
  - Registers maintained in all modes
  - Programmable triggers for ADC no SW delay counters or extra interrupts
- **Functional safety**
  - ISO26262 support for ASIL B or higher
  - Memory Protection Unit, ECC on Flash/Dataflash and RAM
  - Independent internal OSC for Watchdog
  - Diversity between ADC and ACMP, SPI/SCI and FlexIO
  - Core self test libraries
  - Scalable LVD protection, CRC
- **Low power**
  - Low leakage technology
  - Multiple VLP modes and IRC combos
  - Wake-up on analog thresholds
- **Security**
  - CSEc (SHE-spec)



Operating Characteristics
- Voltage range: 2.7V to 5.5V
- Temperature (ambient): -40°C to +125°C

Packages & IO
- Open-drain for 3.3 V and hi-drive pins
- Powered ESD protection
- Packages: 100 BGA, 64 LQFP, 100 LQFP

S32K1xx Overview



S32K11x

S32K116	S32K118
Cortex-M0+ @ 48MHz	Cortex-M0+ @ 48MHz
128KB Flash	256KB Flash
16KB SRAM	24KB SRAM
up to 42 I/Os	up to 58 I/Os
1x FlexCAN with 1x FD	1x FlexCAN with 1x FD
QFN-32	LQFP-64
LQFP-48	LQFP-48

Common Features
AEC-Q100
Security Module
MPU
Low Power
FlexIO
ASIL-B capable
JTAG
FlexTimer
SDK
NFC Stack
Autosar MCAL / OS
S32 Design Studio

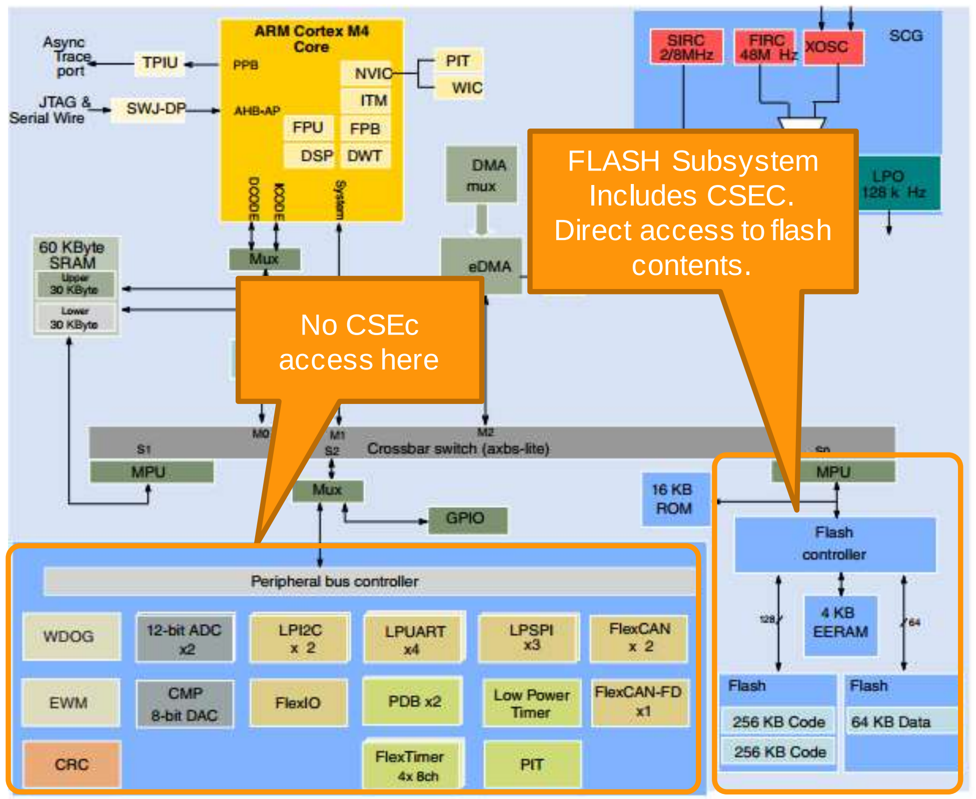
S32K14x

S32K142	S32K144	S32K146	S32K148
Cortex-M4F @ 112MHz	Cortex-M4F @ 112MHz	Cortex-M4F @ 112MHz	Cortex-M4F @ 112MHz
256KB Flash	512KB Flash	1MB Flash	2MB Flash
32KB SRAM	64KB SRAM	128KB SRAM	256KB SRAM
up to 89 I/Os	up to 89 I/Os	up to 128 I/Os	up to 156 I/Os
DMA	DMA	DMA	DMA
2x FlexCAN with 1x FD	3x FlexCAN with 1x FD	3x FlexCAN with 2x FD	3x FlexCAN with 3x FD
LQFP-64	LQFP-64	LQFP-144	LQFP-144
LQFP-100	LQFP-100	LQFP-100	LQFP-176
	MAPBGA-100	MAPBGA-100	MAPBGA-100
			ENET
			Quad SPI
			ETM Trace
			SAI

CSEc Security Block Diagram

- Supports all Global OEM Requirements for End Node Security

- Supports >SHE functionality
  - Secure key storage
  - AES-128 encryption/decryption
  - AES-128 Cypher-based Message Authentication Code (CMAC) calculation and authentication
  - True and Pseudo random number generation
  - User configurable Secure Boot Mode (Sequential, Strict, or Parallel Boot)

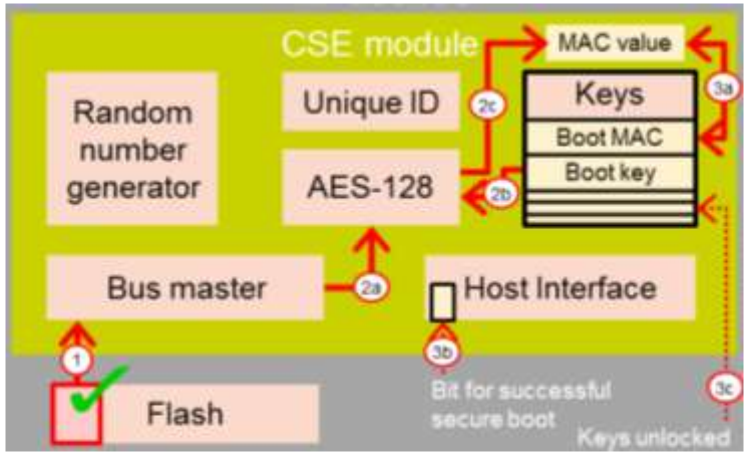


Security: CSE Security Use Cases

Secure Boot

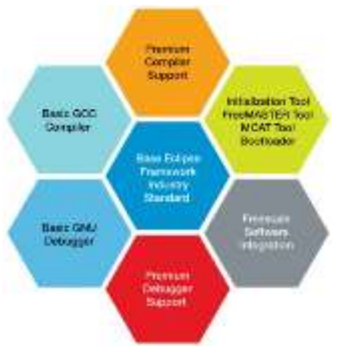
Check Boot Loader for Integrity and Authenticity

1. Check Boot Loader for Integrity and Authenticity
2. CSE module uses the boot key to calculates the MAC value of the bootloader
3. CSE module compares calculated MAC with stored boot MAC.
   a. If identical: successful secure boot → set respective bit in host interface and unlock keys
   b. MCU starts bootloader



S32K1 Enablement

Software


Premium Compiler Support
Basic GCC Compiler
Initialization Tool FreeMASTER Tool MCAT Tool Bootloader
Base Eclipse Framework Industry Standard
Basic GNU Debugger
Freescale Software Integration
Premium Debugger Support

- *Free* S32 SDK – Automotive-grade, pre-qualified, multiple low-level drivers, optional middleware (LIN, NFC, TSI),
- *Free* S32 Design Studio IDE – Eclipse based, supports multiple compiler & debugger plug-ins
- Math, Motor Control & Core Self Test Libraries, MATLAB based design tools

Hardware

- NXP & 3rd party SW tool compatible, out-of-box examples for fast start-up & prototyping
- Arduino UNO compatible with expansion “shield” support
- $49 resale

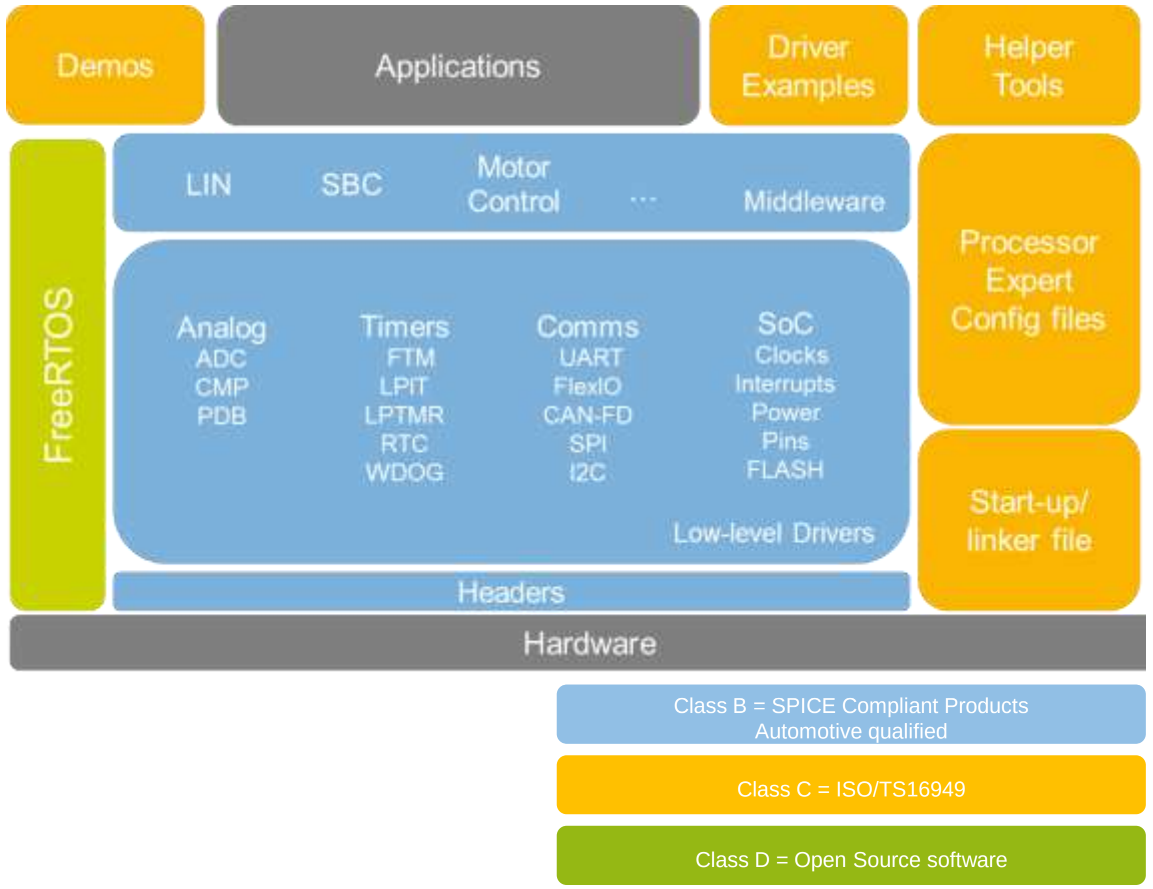
Ecosystem

- Premium level IDE, complier & debugger tools
- NXP & 3rd party MCAL / AUTOSAR + *new* ARCCORE Starter Kit
- S32K, S32DS & SDK Communities http://www.nxp.com/community

S32 SDK Overview

Features

- Integrated Non-Autosar SW package
- Graphical-based configuration
- Layered SW architecture
- Documented Source code and examples
- Integrated with S32 Design Studio and other IDEs
- Various Middleware e.g. Core Self Test, LIN Stack, Automotive Math and Motor Control Library (sold separately, demo as binary)
- FreeRTOS integration
- Multiple toolchains supported
- Several examples and demos



Summary: Why S32K?....it's SIMPLE

Security & Safety

Integration

M-Cortex

Power Consumption

Longevity

Enablement

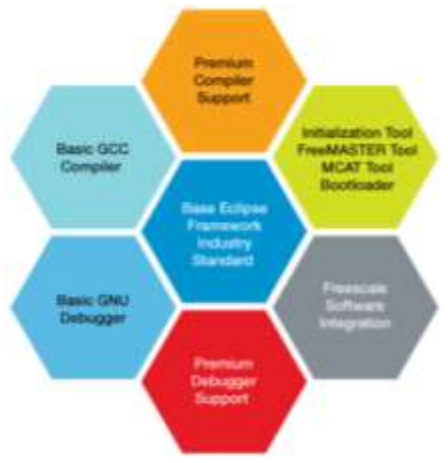
ARM

CAN-FD

SAFE ASSURE™ by NXP

Product Longevity

AUTOSAR Enabling Innovation


Premium Compiler Support
Basic GCC Compiler
Initialization Tool FreeMASTER Tool MCAT Tool Bootloader
Base Eclipse Framework Industry Standard
Basic GNU Debugger
Freescale Software Integration
Premium Debugger Support

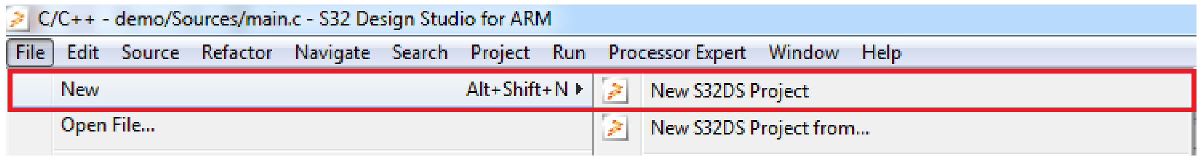
S32 SDK

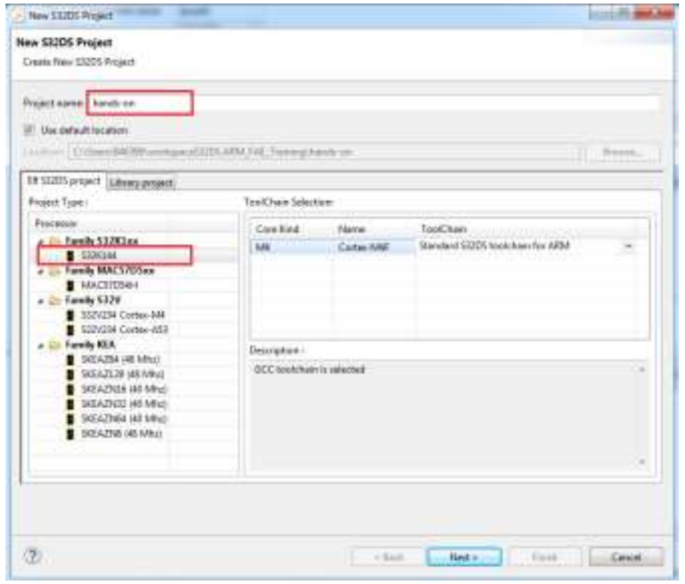
freeRTOS

Hands-on

S32 Design Studio – How to create a project

- Create a new S32DS Project

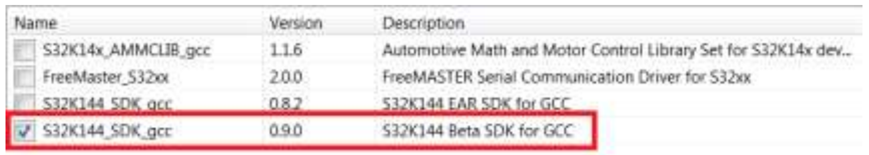




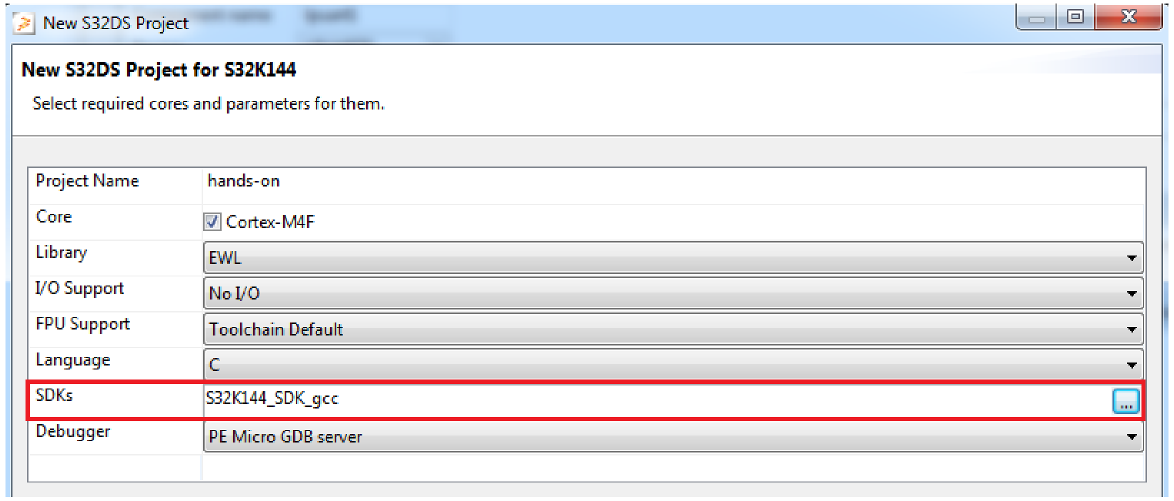
Project name: hands-on

Core Kind	Name	ToolChain
M4	Cortex-M4F	Standard S32DS toolchain for ARM

GCC toolchain is selected



	Name	Version	Description
☐	S32K14x_AMMCLIB_gcc	1.1.6	Automotive Math and Motor Control Library Set for S32K14x dev...
☐	FreeMaster_S32xx	2.0.0	FreeMASTER Serial Communication Driver for S32xx
☐	S32K144_SDK_gcc	0.8.2	S32K144 EAR SDK for GCC
✓	S32K144_SDK_gcc	0.9.0	S32K144 Beta SDK for GCC

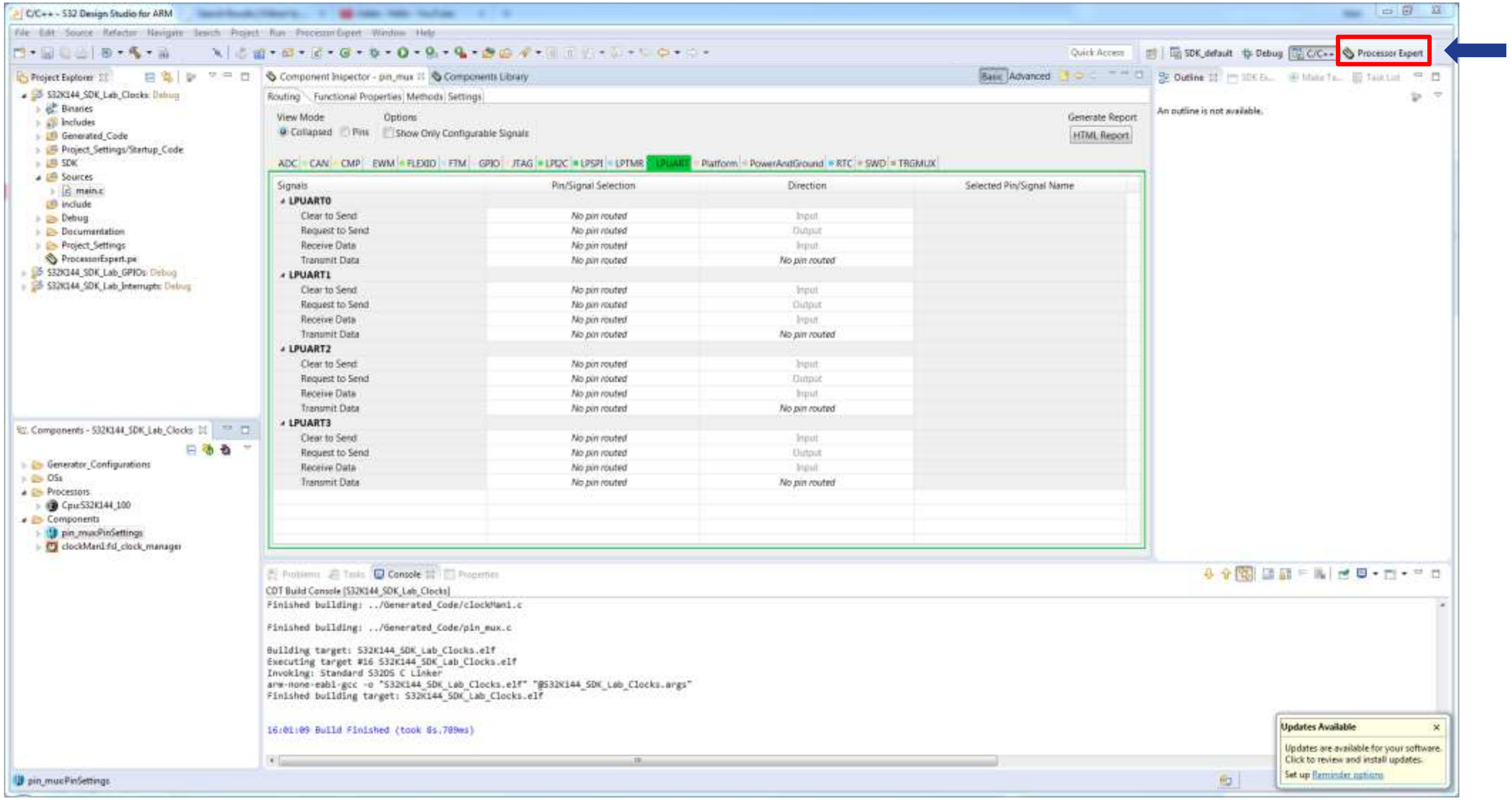


New S32DS Project for S32K144

Select required cores and parameters for them.

Project Name	hands-on
Core	✓ Cortex-M4F
Library	EWL
I/O Support	No I/O
FPU Support	Toolchain Default
Language	C
SDKs	S32K144_SDK_gcc
Debugger	PE Micro GDB server

S32 Design Studio – graphical configuration environment


C/C++ - S32 Design Studio for ARM
File Edit Source Refactor Navigate Search Project Run Processor Expert Window Help
Quick Access
SDK_default
Debug
C/C++
Processor Expert
Project Explorer
S32K144_SDK_Lab_Clocks: Debug
Binaries
Includes
Generated_Code
Project_Settings/Startup_Code
SDK
Sources
main.c
include
Debug
Documentation
Project_Settings
ProcessorExpert.pe
S32K144_SDK_Lab_GPIOs: Debug
S32K144_SDK_Lab_Interrupts: Debug
Components - S32K144_SDK_Lab_Clocks
Generator_Configurations
OSs
Processors
Cpu:S32K144_100
Components
pin_mux:PinSettings
clockMan1:fsl_clock_manager
Component Inspector - pin_mux
Components Library
Basic
Advanced
Routing
Functional Properties
Methods
Settings
View Mode
Collapsed
Pins
Options
Show Only Configurable Signals
Generate Report
HTML Report
ADC CAN CMP EWM FLEXIO FTM GPIO JTAG LPI2C LPSPI LPTMR LPUART Platform PowerAndGround RTC SWD TRGMUX
Outline
SDK E...
Make Ta...
Task List
An outline is not available.
Problems
Tasks
Console
Properties
pin_mux:PinSettings

Signals	Pin/Signal Selection	Direction	Selected Pin/Signal Name
LPUART0			
Clear to Send	*No pin routed*	Input	
Request to Send	*No pin routed*	Output	
Receive Data	*No pin routed*	Input	
Transmit Data	*No pin routed*	*No pin routed*	
LPUART1			
Clear to Send	*No pin routed*	Input	
Request to Send	*No pin routed*	Output	
Receive Data	*No pin routed*	Input	
Transmit Data	*No pin routed*	*No pin routed*	
LPUART2			
Clear to Send	*No pin routed*	Input	
Request to Send	*No pin routed*	Output	
Receive Data	*No pin routed*	Input	
Transmit Data	*No pin routed*	*No pin routed*	
LPUART3			
Clear to Send	*No pin routed*	Input	
Request to Send	*No pin routed*	Output	
Receive Data	*No pin routed*	Input	
Transmit Data	*No pin routed*	*No pin routed*	

CDT Build Console [S32K144_SDK_Lab_Clocks]

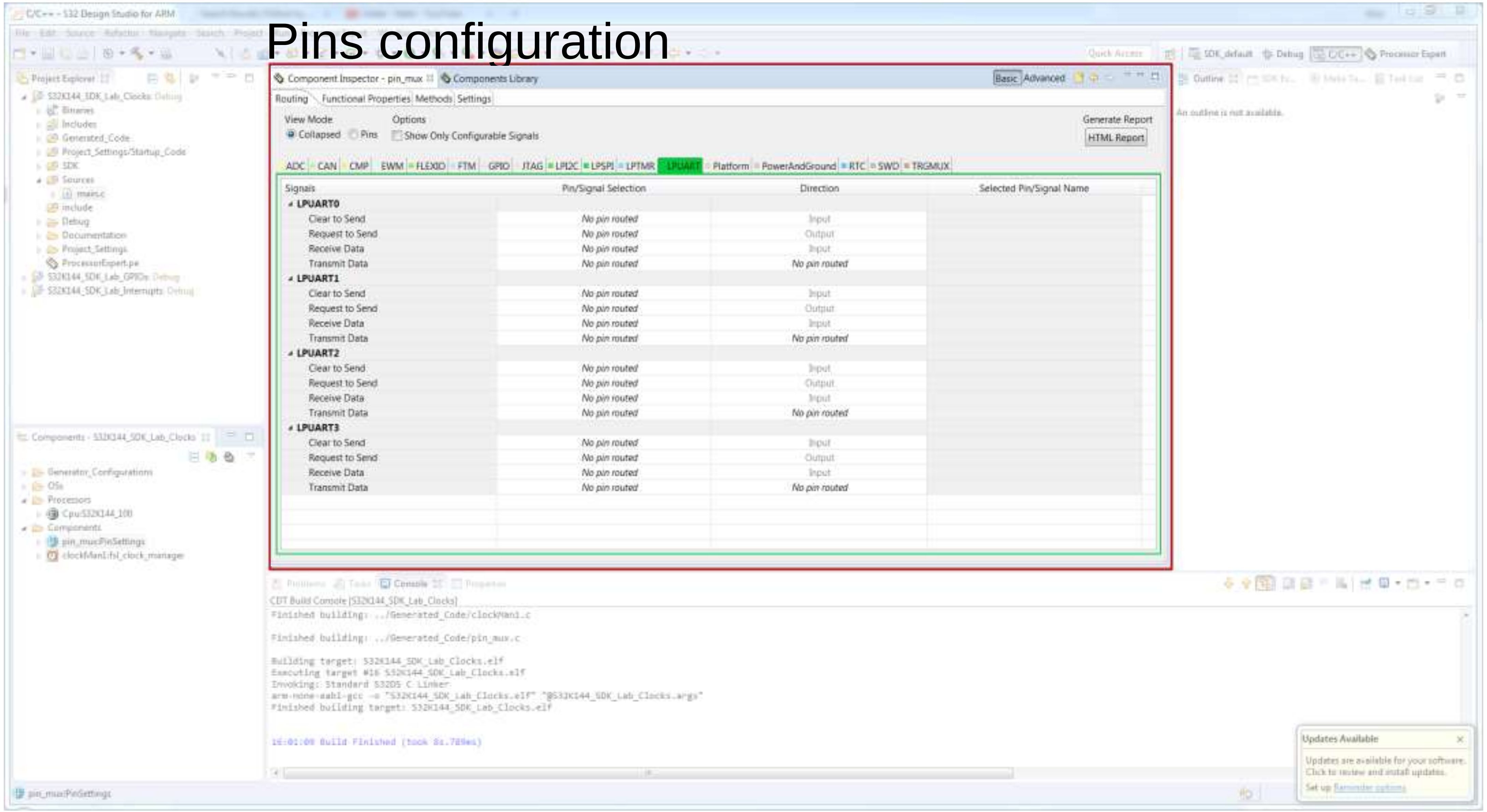
```
Finished building: ../Generated_Code/clockMan1.c

Finished building: ../Generated_Code/pin_mux.c

Building target: S32K144_SDK_Lab_Clocks.elf
Executing target #16 S32K144_SDK_Lab_Clocks.elf
Invoking: Standard S32DS C Linker
arm-none-eabi-gcc -o "S32K144_SDK_Lab_Clocks.elf" "@S32K144_SDK_Lab_Clocks.args"
Finished building target: S32K144_SDK_Lab_Clocks.elf


16:01:09 Build Finished (took 8s.789ms)
```

Updates Available

Updates are available for your software. Click to review and install updates.

Set up Reminder options

S32 Design Studio – graphical configuration environment



Pins configuration

Component Inspector - pin_mux | Components Library

Routing | Functional Properties | Methods | Settings

View Mode: [✓ Collapsed] Pins

Options: Show Only Configurable Signals

Generate Report: HTML Report

ADC | CAN | CMP | EWM | FLEXIO | FTM | GPIO | JTAG | LPI2C | LPSPI | LPTMR | LPUART | Platform | PowerAndGround | RTC | SWD | TRGMUX

Signals	Pin/Signal Selection	Direction	Selected Pin/Signal Name
LPUART0			
Clear to Send	No pin routed	Input	
Request to Send	No pin routed	Output	
Receive Data	No pin routed	Input	
Transmit Data	No pin routed	No pin routed	
LPUART1			
Clear to Send	No pin routed	Input	
Request to Send	No pin routed	Output	
Receive Data	No pin routed	Input	
Transmit Data	No pin routed	No pin routed	
LPUART2			
Clear to Send	No pin routed	Input	
Request to Send	No pin routed	Output	
Receive Data	No pin routed	Input	
Transmit Data	No pin routed	No pin routed	
LPUART3			
Clear to Send	No pin routed	Input	
Request to Send	No pin routed	Output	
Receive Data	No pin routed	Input	
Transmit Data	No pin routed	No pin routed	

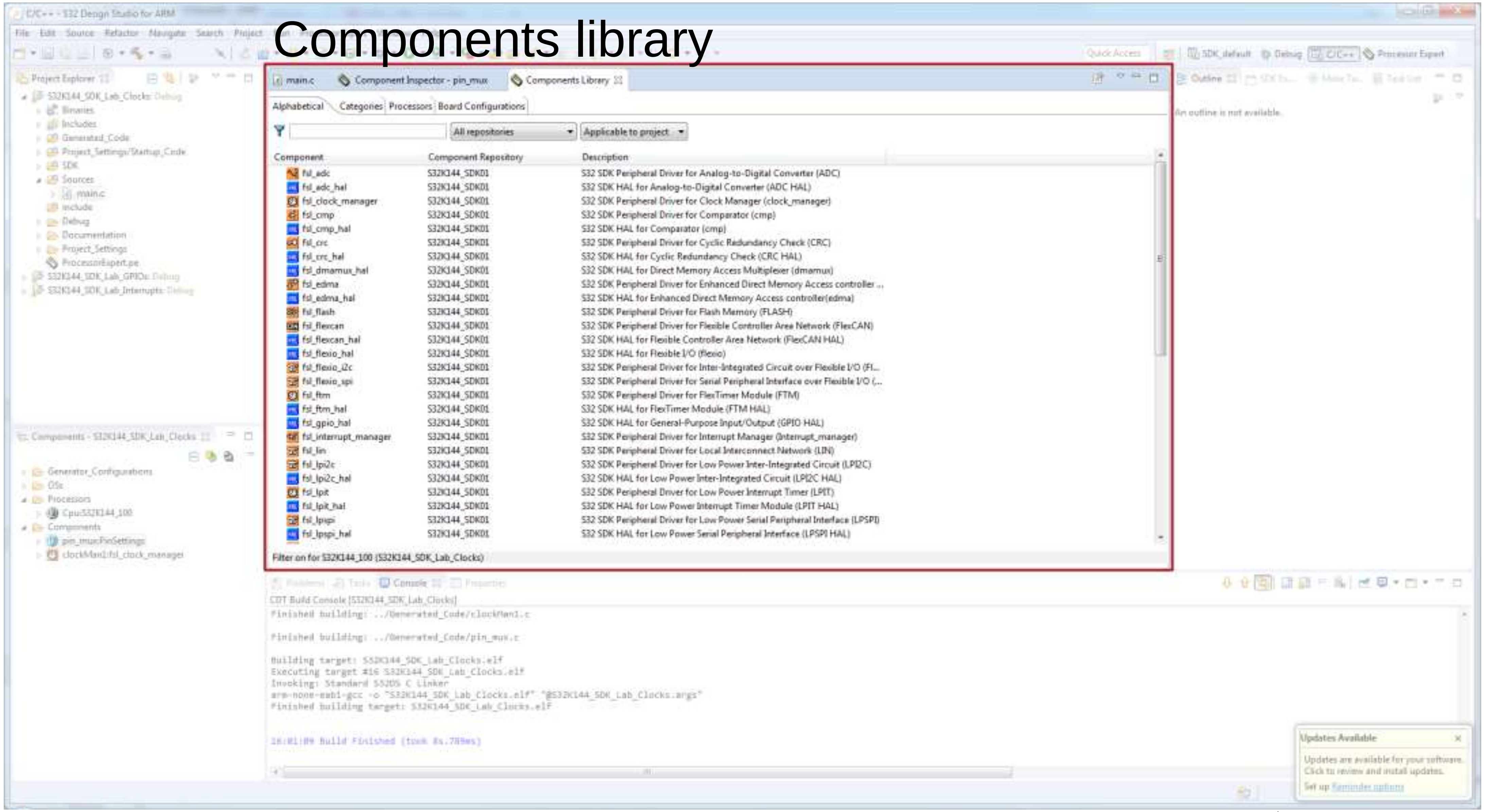
CDT Build Console [S32K144_SDK_Lab_Clocks]

```
Finished building: ../Generated_Code/clockMan1.c

Finished building: ../Generated_Code/pin_mux.c

Building target: S32K144_SDK_Lab_Clocks.elf
Executing target #16 S32K144_SDK_Lab_Clocks.elf
Invoking: Standard S32DS C Linker
arm-none-eabi-gcc -o "S32K144_SDK_Lab_Clocks.elf" "@S32K144_SDK_Lab_Clocks.args"
Finished building target: S32K144_SDK_Lab_Clocks.elf


16:01:09 Build Finished (took 8s.789ms)
```

Updates Available

Updates are available for your software. Click to review and install updates.

S32 Design Studio – graphical configuration environment



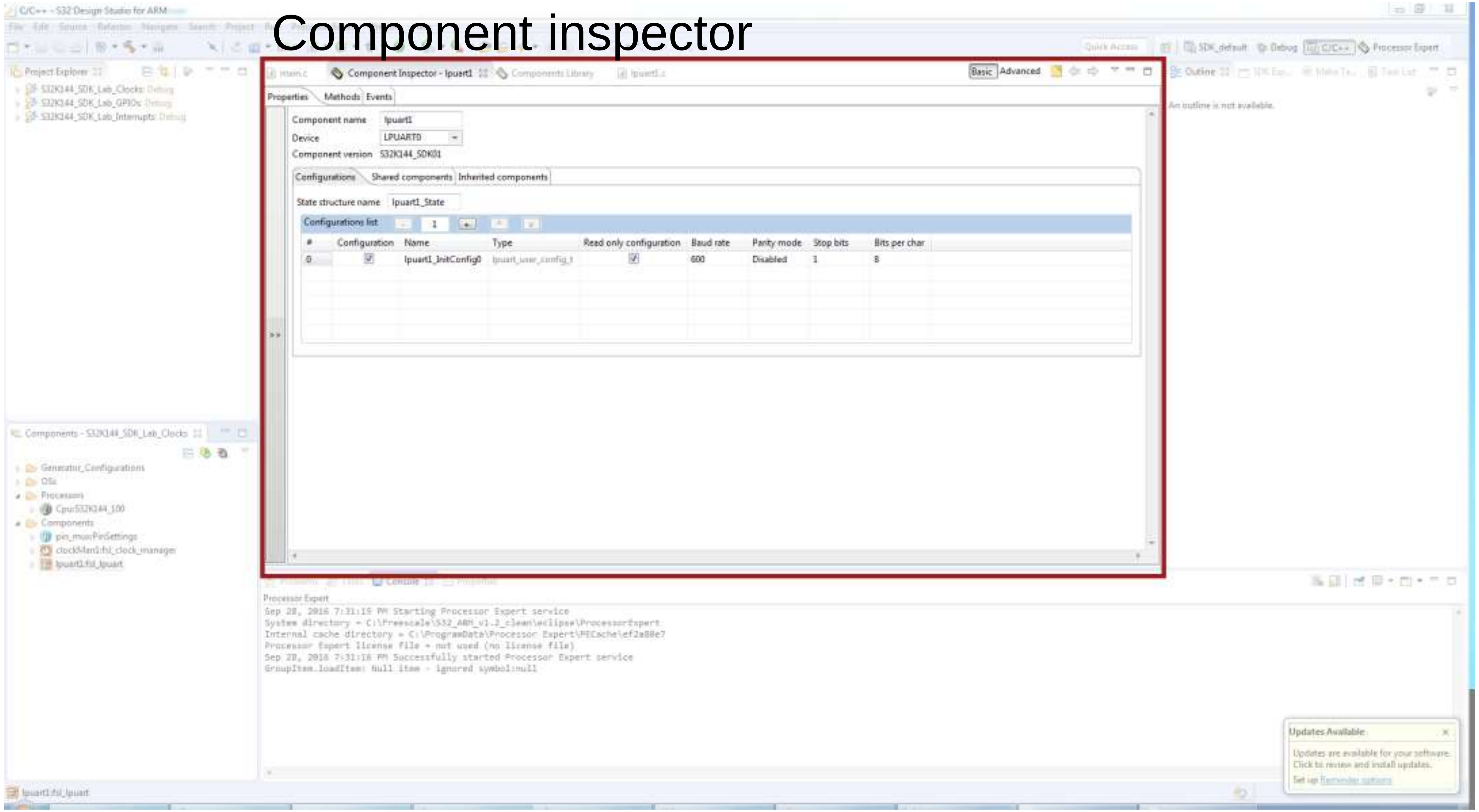
Component	Component Repository	Description
fsl_adc	S32K144_SDK01	S32 SDK Peripheral Driver for Analog-to-Digital Converter (ADC)
fsl_adc_hal	S32K144_SDK01	S32 SDK HAL for Analog-to-Digital Converter (ADC HAL)
fsl_clock_manager	S32K144_SDK01	S32 SDK Peripheral Driver for Clock Manager (clock_manager)
fsl_cmp	S32K144_SDK01	S32 SDK Peripheral Driver for Comparator (cmp)
fsl_cmp_hal	S32K144_SDK01	S32 SDK HAL for Comparator (cmp)
fsl_crc	S32K144_SDK01	S32 SDK Peripheral Driver for Cyclic Redundancy Check (CRC)
fsl_crc_hal	S32K144_SDK01	S32 SDK HAL for Cyclic Redundancy Check (CRC HAL)
fsl_dmamux_hal	S32K144_SDK01	S32 SDK HAL for Direct Memory Access Multiplexer (dmamux)
fsl_edma	S32K144_SDK01	S32 SDK Peripheral Driver for Enhanced Direct Memory Access controller ...
fsl_edma_hal	S32K144_SDK01	S32 SDK HAL for Enhanced Direct Memory Access controller(edma)
fsl_flash	S32K144_SDK01	S32 SDK Peripheral Driver for Flash Memory (FLASH)
fsl_flexcan	S32K144_SDK01	S32 SDK Peripheral Driver for Flexible Controller Area Network (FlexCAN)
fsl_flexcan_hal	S32K144_SDK01	S32 SDK HAL for Flexible Controller Area Network (FlexCAN HAL)
fsl_flexio_hal	S32K144_SDK01	S32 SDK HAL for Flexible I/O (flexio)
fsl_flexio_i2c	S32K144_SDK01	S32 SDK Peripheral Driver for Inter-Integrated Circuit over Flexible I/O (Fl...
fsl_flexio_spi	S32K144_SDK01	S32 SDK Peripheral Driver for Serial Peripheral Interface over Flexible I/O (...
fsl_ftm	S32K144_SDK01	S32 SDK Peripheral Driver for FlexTimer Module (FTM)
fsl_ftm_hal	S32K144_SDK01	S32 SDK HAL for FlexTimer Module (FTM HAL)
fsl_gpio_hal	S32K144_SDK01	S32 SDK HAL for General-Purpose Input/Output (GPIO HAL)
fsl_interrupt_manager	S32K144_SDK01	S32 SDK Peripheral Driver for Interrupt Manager (interrupt_manager)
fsl_lin	S32K144_SDK01	S32 SDK Peripheral Driver for Local Interconnect Network (LIN)
fsl_lpi2c	S32K144_SDK01	S32 SDK Peripheral Driver for Low Power Inter-Integrated Circuit (LPI2C)
fsl_lpi2c_hal	S32K144_SDK01	S32 SDK HAL for Low Power Inter-Integrated Circuit (LPI2C HAL)
fsl_lpit	S32K144_SDK01	S32 SDK Peripheral Driver for Low Power Interrupt Timer (LPIT)
fsl_lpit_hal	S32K144_SDK01	S32 SDK HAL for Low Power Interrupt Timer Module (LPIT HAL)
fsl_lpspi	S32K144_SDK01	S32 SDK Peripheral Driver for Low Power Serial Peripheral Interface (LPSPI)
fsl_lpspi_hal	S32K144_SDK01	S32 SDK HAL for Low Power Serial Peripheral Interface (LPSPI HAL)

Filter on for S32K144_100 (S32K144_SDK_Lab_Clocks)

CDT Build Console [S32K144_SDK_Lab_Clocks]

```
Finished building: ../Generated_Code/clockMan1.c

Finished building: ../Generated_Code/pin_mux.c

Building target: S32K144_SDK_Lab_Clocks.elf
Executing target #16 S32K144_SDK_Lab_Clocks.elf
Invoking: Standard S32DS C Linker
arm-none-eabi-gcc -o "S32K144_SDK_Lab_Clocks.elf" "@S32K144_SDK_Lab_Clocks.args"
Finished building target: S32K144_SDK_Lab_Clocks.elf


16:01:09 Build Finished (took 8s.789ms)
```

S32 Design Studio – graphical configuration environment



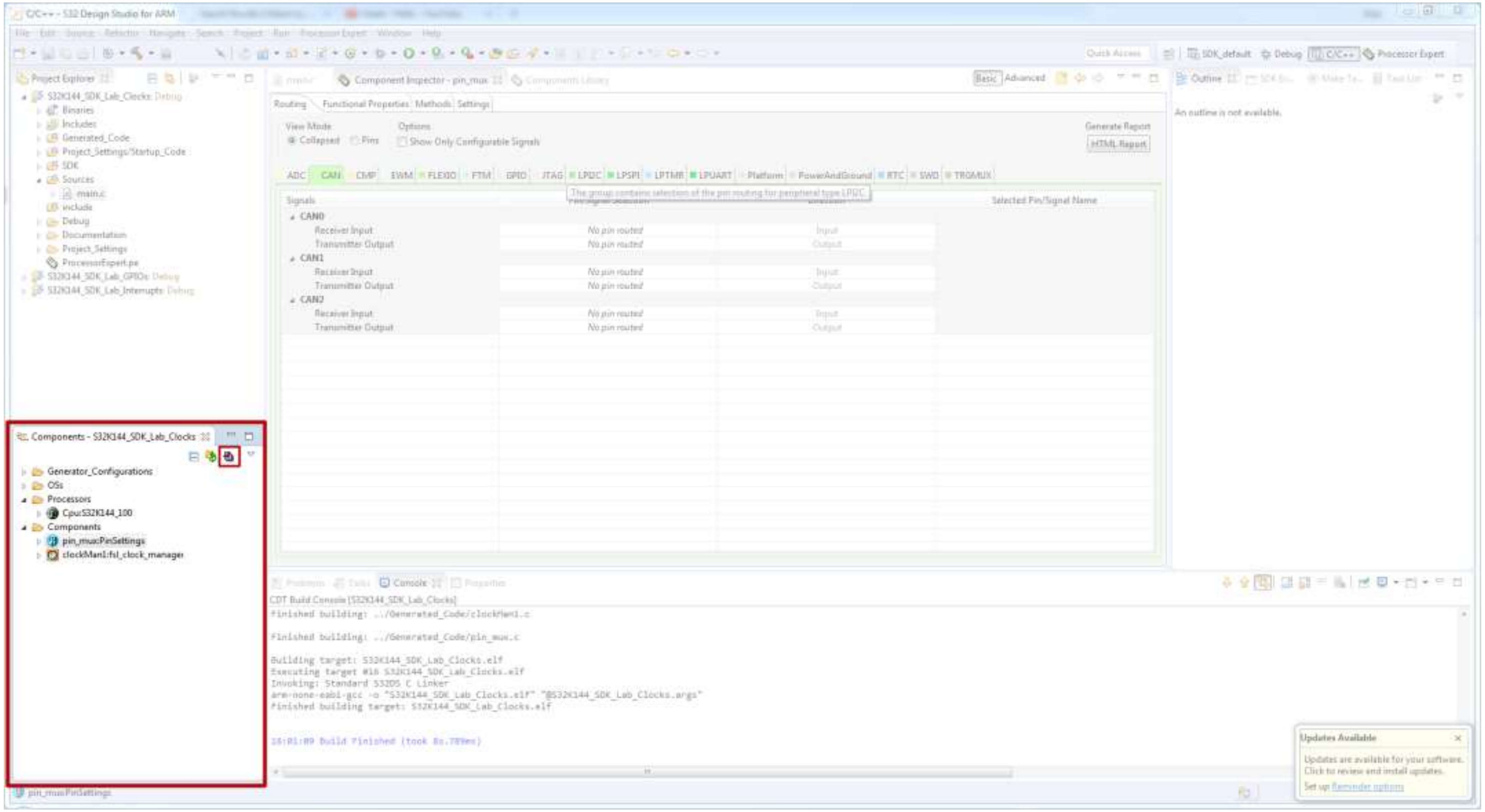
Component name: lpuart1
Device: LPUART0
Component version: S32K144_SDK01

State structure name: lpuart1_State

#	Configuration	Name	Type	Read only configuration	Baud rate	Parity mode	Stop bits	Bits per char
0	✓	lpuart1_InitConfig0	lpuart_user_config_t	✓	600	Disabled	1	8

```
Sep 28, 2016 7:31:15 PM Starting Processor Expert service
System directory = C:\Freescale\S32_ARM_v1.2_clean\eclipse\ProcessorExpert
Internal cache directory = C:\ProgramData\Processor Expert\PECache\ef2a88e7
Processor Expert license file = not used (no license file)
Sep 28, 2016 7:31:16 PM Successfully started Processor Expert service
GroupItem.loadItem: Null item - ignored symbol:null
```

S32 Design Studio – graphical configuration environment

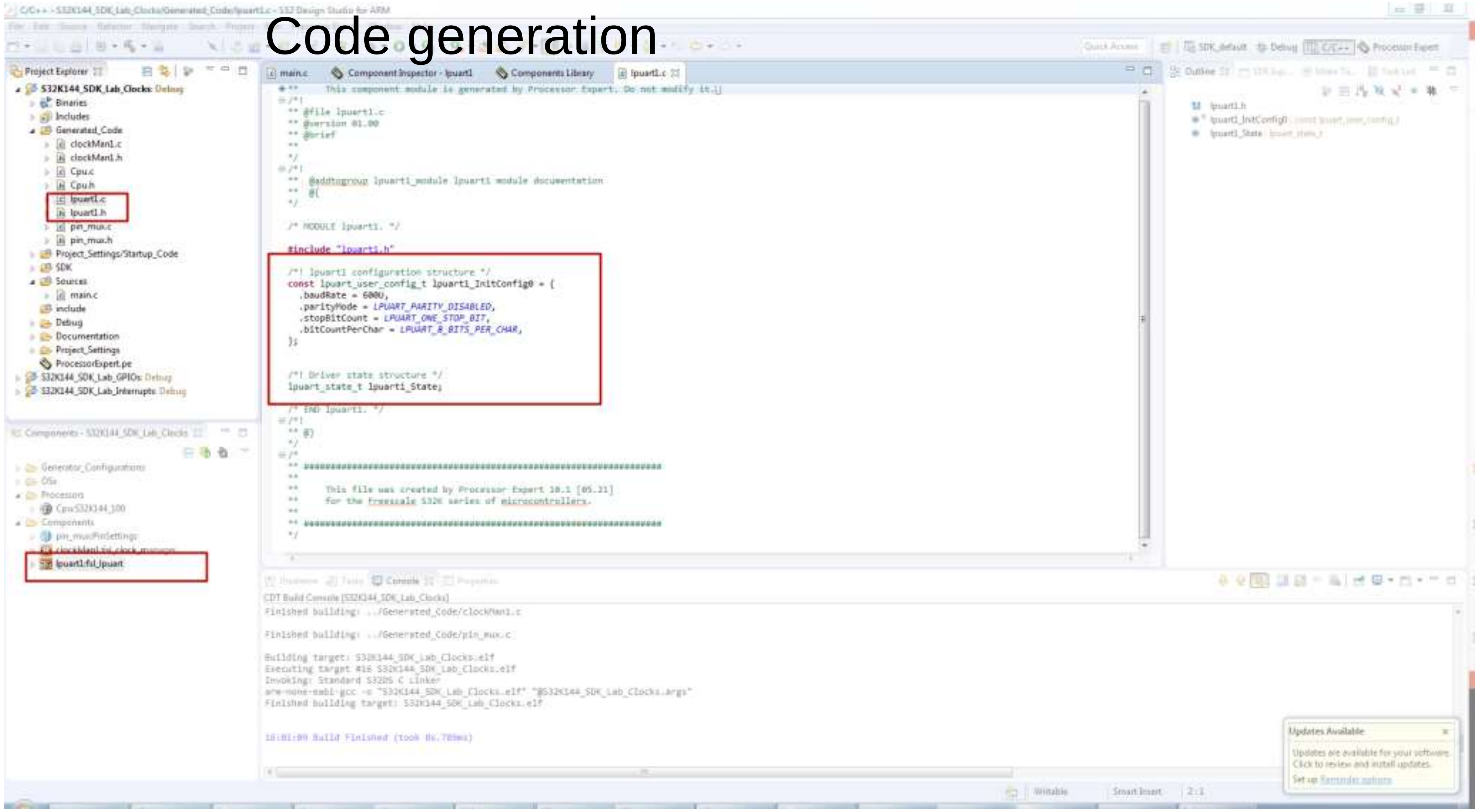


Signals	Pin/Signal Selection	Direction	Selected Pin/Signal Name
CAN0			
Receiver Input	No pin routed	Input	
Transmitter Output	No pin routed	Output	
CAN1			
Receiver Input	No pin routed	Input	
Transmitter Output	No pin routed	Output	
CAN2			
Receiver Input	No pin routed	Input	
Transmitter Output	No pin routed	Output	

```
CDT Build Console [S32K144_SDK_Lab_Clocks]
Finished building: ../Generated_Code/clockMan1.c

Finished building: ../Generated_Code/pin_mux.c

Building target: S32K144_SDK_Lab_Clocks.elf
Executing target #16 S32K144_SDK_Lab_Clocks.elf
Invoking: Standard S32DS C Linker
arm-none-eabi-gcc -o "S32K144_SDK_Lab_Clocks.elf" "@S32K144_SDK_Lab_Clocks.args"
Finished building target: S32K144_SDK_Lab_Clocks.elf


16:01:09 Build Finished (took 8s.789ms)
```

S32 Design Studio – graphical configuration environment

Code generation

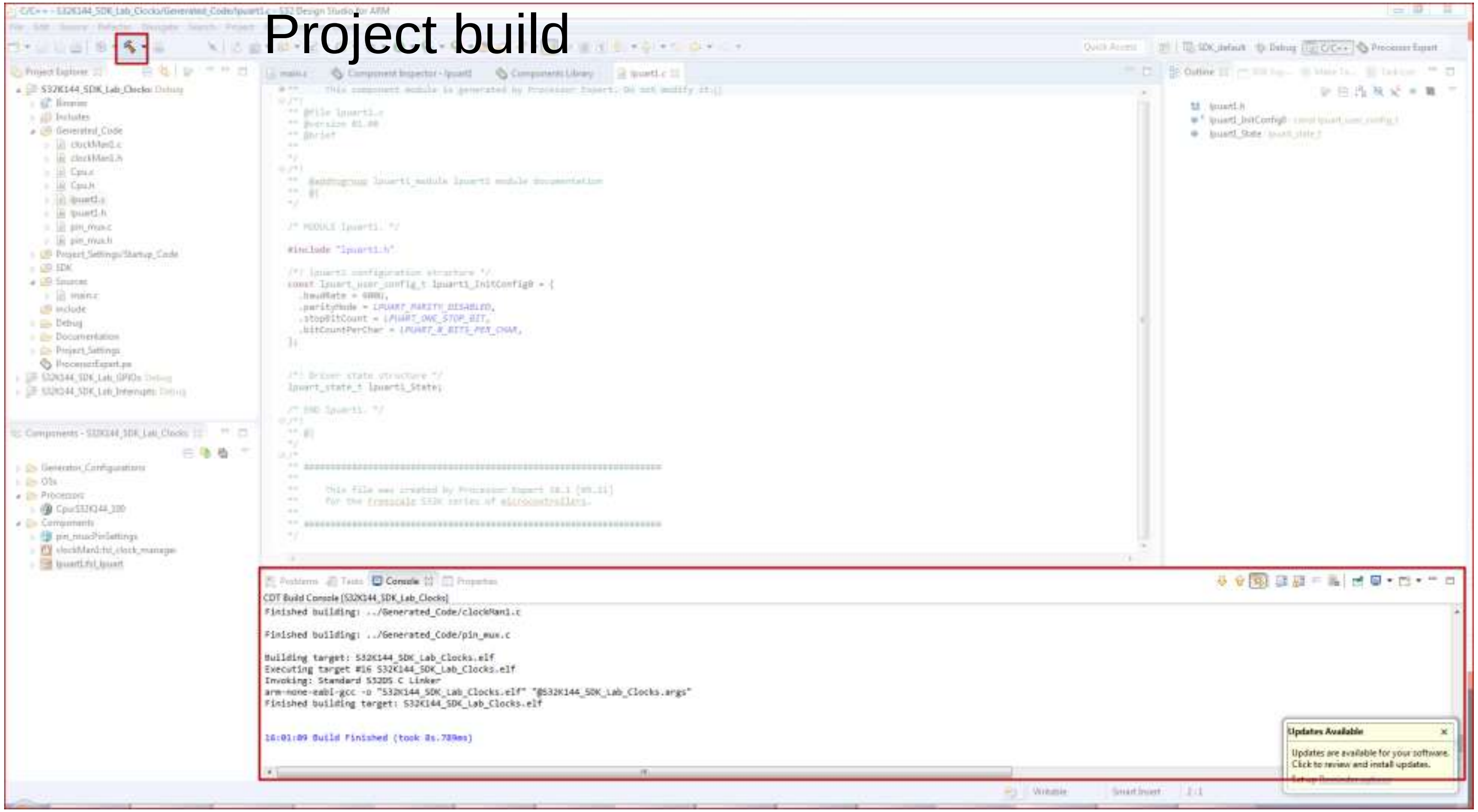


```c
 **     This component module is generated by Processor Expert. Do not modify it.
/*!
 ** @file lpuart1.c
 ** @version 01.00
 ** @brief
 **
 */
/*!
 **  @addtogroup lpuart1_module lpuart1 module documentation
 **  @{
 */

/* MODULE lpuart1. */

#include "lpuart1.h"

/*! lpuart1 configuration structure */
const lpuart_user_config_t lpuart1_InitConfig0 = {
  .baudRate = 600U,
  .parityMode = LPUART_PARITY_DISABLED,
  .stopBitCount = LPUART_ONE_STOP_BIT,
  .bitCountPerChar = LPUART_8_BITS_PER_CHAR,
};


/*! Driver state structure */
lpuart_state_t lpuart1_State;
/* END lpuart1. */
/*!
 ** @}
 */
/*
 ** ###################################################################
 **
 **     This file was created by Processor Expert 10.1 [05.21]
 **     for the Freescale S32K series of microcontrollers.
 **
 ** ###################################################################
 */
```

```
CDT Build Console [S32K144_SDK_Lab_Clocks]
Finished building: ../Generated_Code/clockMan1.c

Finished building: ../Generated_Code/pin_mux.c

Building target: S32K144_SDK_Lab_Clocks.elf
Executing target #16 S32K144_SDK_Lab_Clocks.elf
Invoking: Standard S32DS C Linker
arm-none-eabi-gcc -o "S32K144_SDK_Lab_Clocks.elf" "@S32K144_SDK_Lab_Clocks.args"
Finished building target: S32K144_SDK_Lab_Clocks.elf


10:01:09 Build Finished (took 8s.708ms)
```

Updates are available for your software. Click to review and install updates.
Set up Reminder options

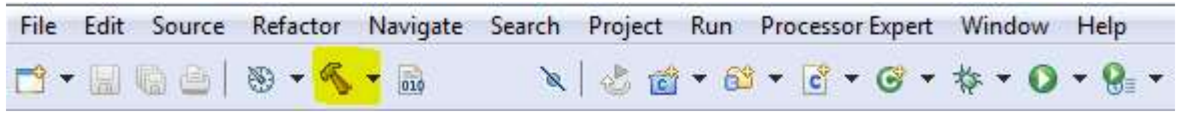
S32 Design Studio – graphical configuration environment

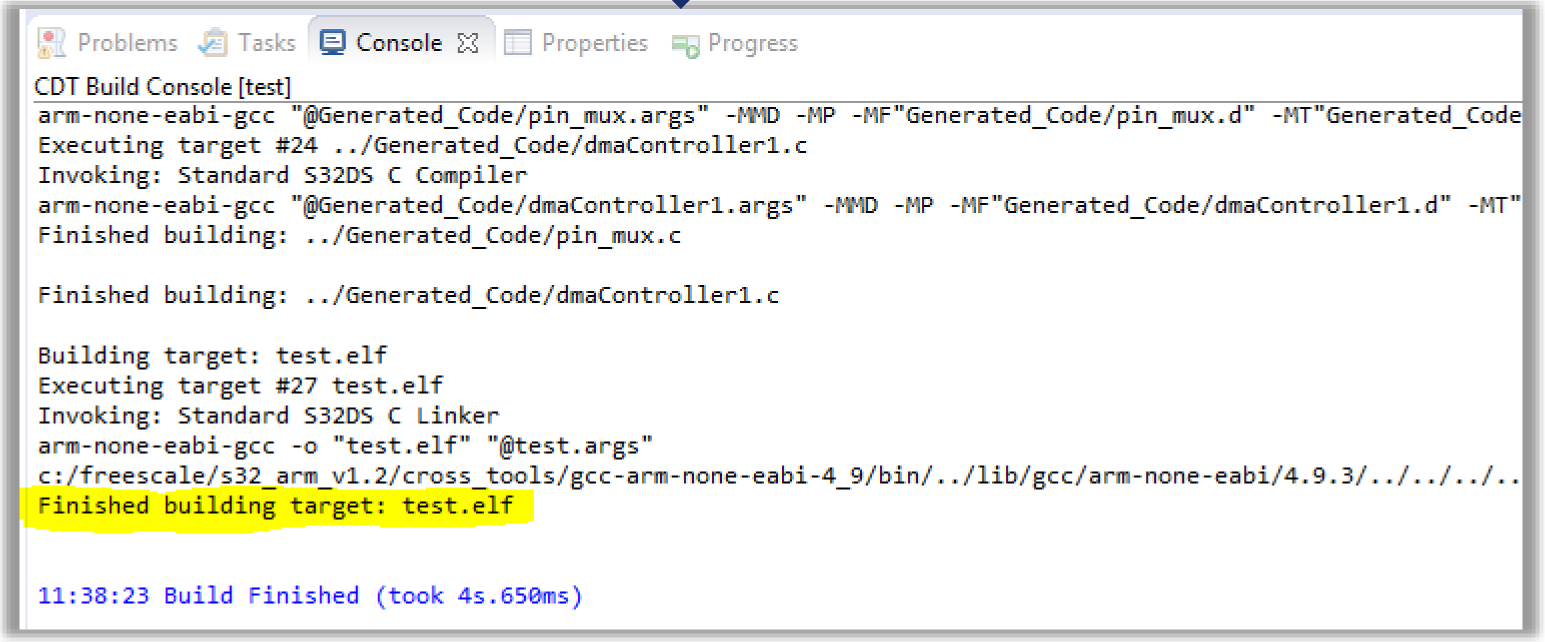

Project build
Problems
Tasks
Console
Properties
Updates Available
Updates are available for your software. Click to review and install updates.

```
CDT Build Console [S32K144_SDK_Lab_Clocks]
Finished building: ../Generated_Code/clockMan1.c

Finished building: ../Generated_Code/pin_mux.c

Building target: S32K144_SDK_Lab_Clocks.elf
Executing target #16 S32K144_SDK_Lab_Clocks.elf
Invoking: Standard S32DS C Linker
arm-none-eabi-gcc -o "S32K144_SDK_Lab_Clocks.elf" "@S32K144_SDK_Lab_Clocks.args"
Finished building target: S32K144_SDK_Lab_Clocks.elf


16:01:09 Build Finished (took 0s.789ms)
```

S32 Design Studio – deploying the application

- Code generation
- Project build

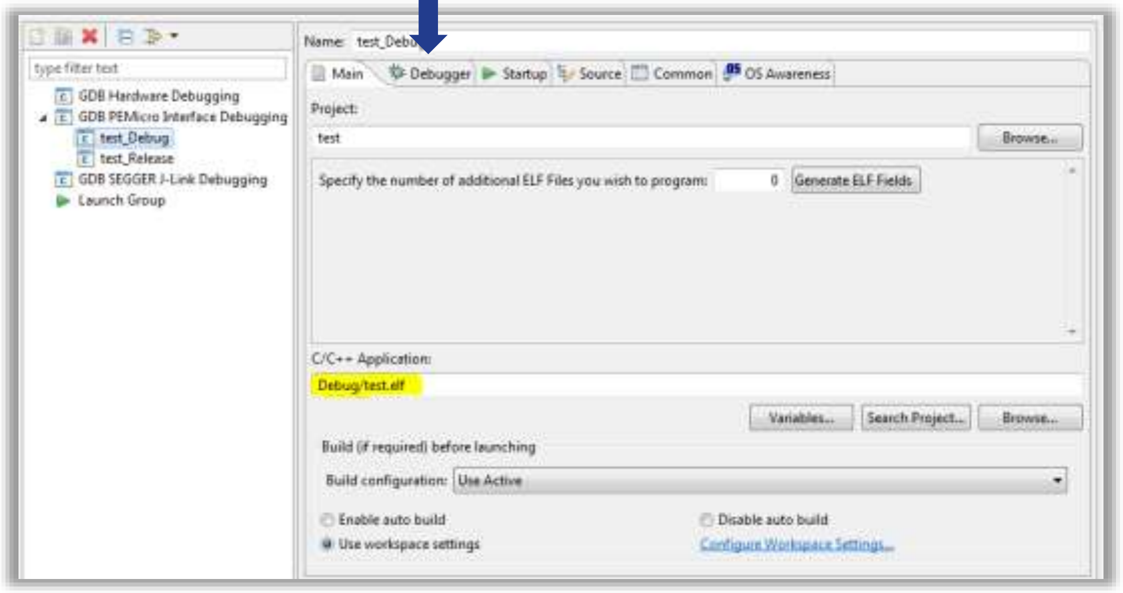




```
arm-none-eabi-gcc "@Generated_Code/pin_mux.args" -MMD -MP -MF"Generated_Code/pin_mux.d" -MT"Generated_Code
Executing target #24 ../Generated_Code/dmaController1.c
Invoking: Standard S32DS C Compiler
arm-none-eabi-gcc "@Generated_Code/dmaController1.args" -MMD -MP -MF"Generated_Code/dmaController1.d" -MT"
Finished building: ../Generated_Code/pin_mux.c

Finished building: ../Generated_Code/dmaController1.c

Building target: test.elf
Executing target #27 test.elf
Invoking: Standard S32DS C Linker
arm-none-eabi-gcc -o "test.elf" "@test.args"
c:/freescale/s32_arm_v1.2/cross_tools/gcc-arm-none-eabi-4_9/bin/../lib/gcc/arm-none-eabi/4.9.3/../../../..
Finished building target: test.elf


11:38:23 Build Finished (took 4s.650ms)
```

S32 Design Studio – deploying the application

- Code generation
- Project build
- Target debug



type filter text
- GDB Hardware Debugging
- GDB PEMicro Interface Debugging
  - test_Debug
  - test_Release
- GDB SEGGER J-Link Debugging
- Launch Group

Name: test_Debug

Main | Debugger | Startup | Source | Common | OS Awareness

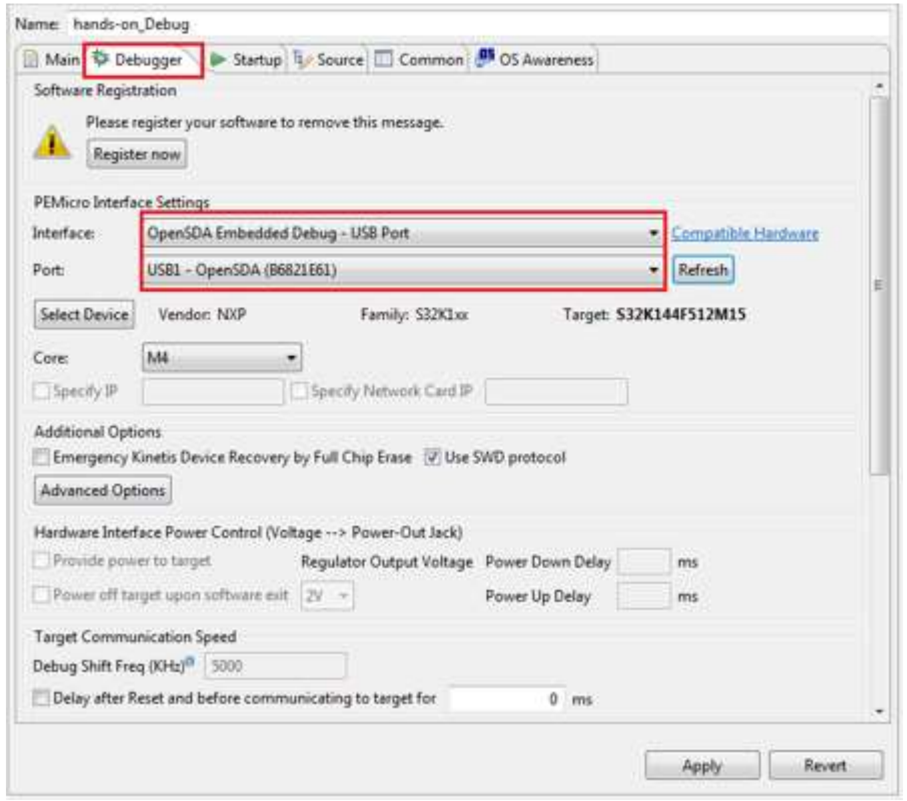
Project:
test
Browse...

Specify the number of additional ELF Files you wish to program: 0 Generate ELF Fields

C/C++ Application:
Debug/test.elf

Variables... | Search Project... | Browse...

Build (if required) before launching
Build configuration: Use Active
- Enable auto build
- Disable auto build
- Use workspace settings
Configure Workspace Settings...



Name: hands-on_Debug

Main | Debugger | Startup | Source | Common | OS Awareness

Software Registration
Please register your software to remove this message.
Register now

PEMicro Interface Settings
Interface: OpenSDA Embedded Debug - USB Port — Compatible Hardware
Port: USB1 - OpenSDA (B6821E61) — Refresh

Select Device — Vendor: NXP — Family: S32K1xx — Target: S32K144F512M15

Core: M4
Specify IP — Specify Network Card IP

Additional Options
Emergency Kinetis Device Recovery by Full Chip Erase — [✓ Use SWD protocol]
Advanced Options

Hardware Interface Power Control (Voltage --> Power-Out Jack)
Provide power to target — Regulator Output Voltage — Power Down Delay ms
Power off target upon software exit — 2V — Power Up Delay ms

Target Communication Speed
Debug Shift Freq (KHz): 5000
Delay after Reset and before communicating to target for 0 ms

Apply | Revert

S32K144 GPIOs

S32K144 GPIOs Lab: Objective

- **Task:**
  - Turn ON Green LED as long as SW2 is pressed

- **Learn:**
  - About the GPIOs structure in S32K144
  - How to create a new SDK project with S32DS.
  - How to set a pin as output/input with SDK

- **Target Modules:**
  - PCC, PORT and GPIO modules

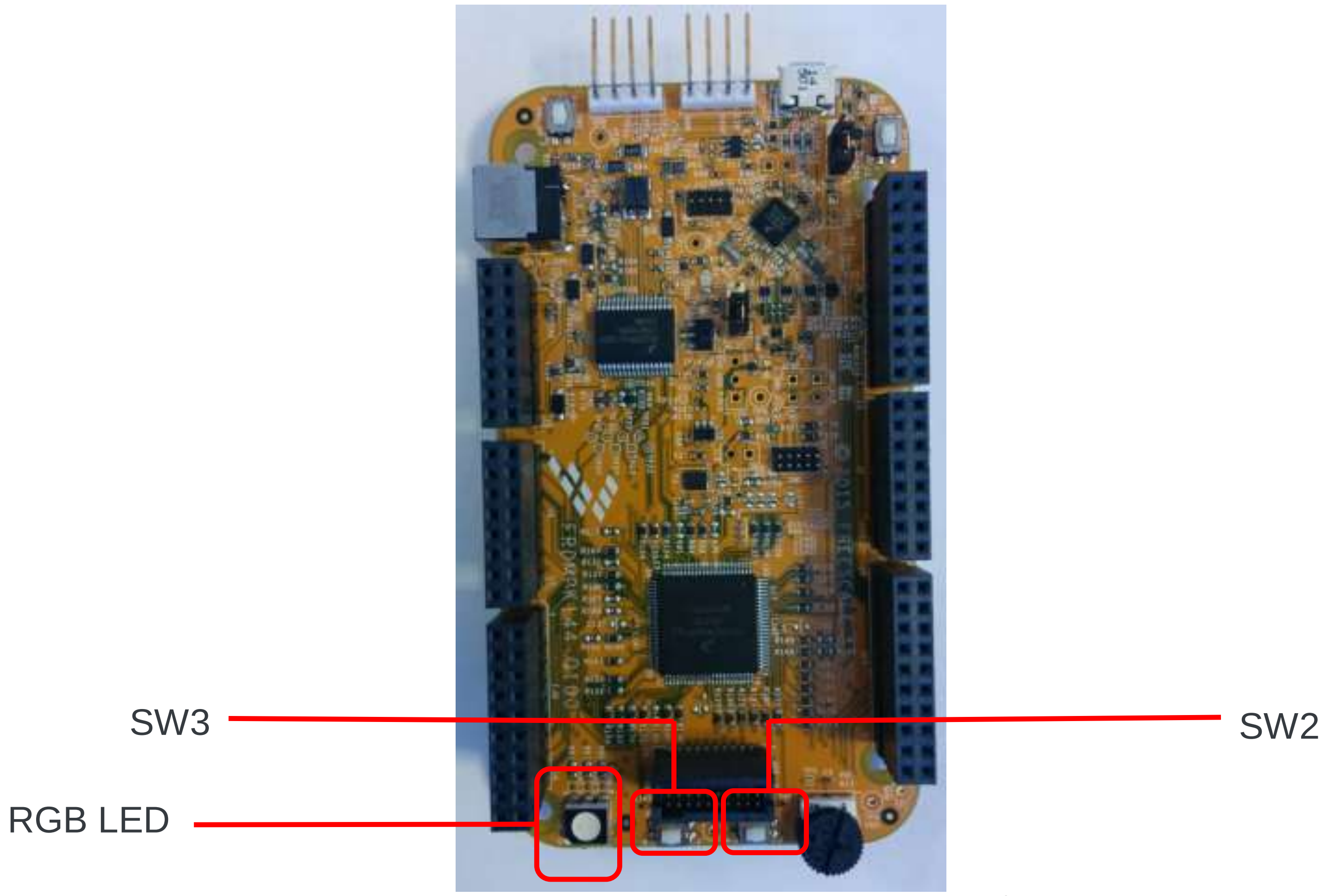
S32K144 GPIOs Lab: Resources to be used

- In this lab will use the following components of the EVB:
  - RGB LED

LED	S32K144 PIN	Pull resistor
BLUE	PTD0	Pull up
RED	PTD15	Pull up
GREEN	PTD16	Pull up

  - SW2 and SW3

Button	S32K144 PIN	Pull resistor
SW2	PTC12	Pull down
SW3	PTC13	Pull down

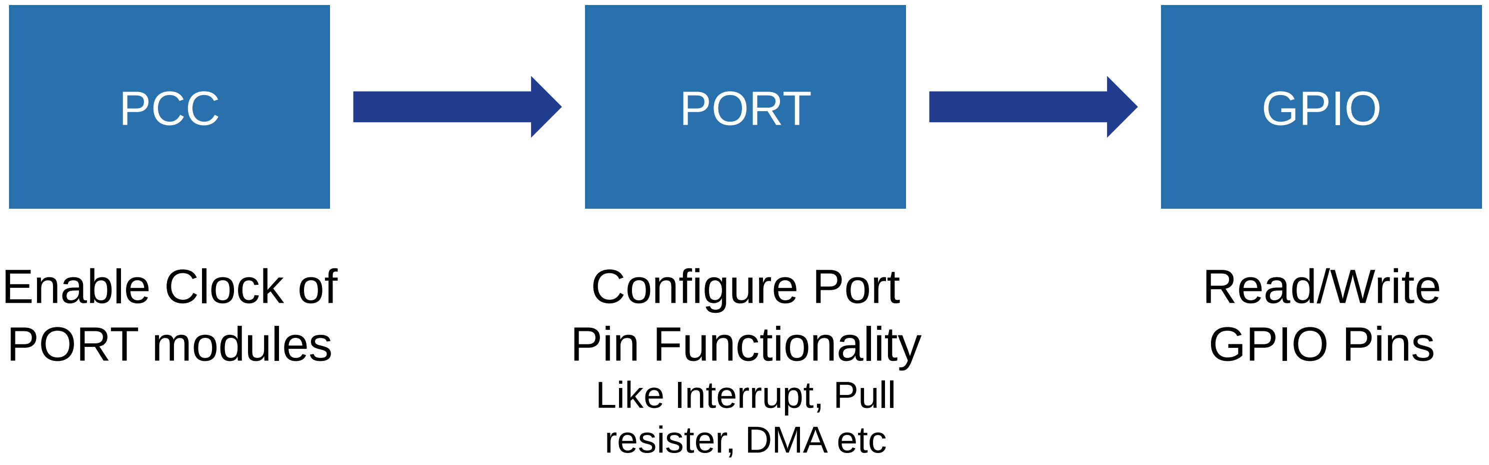


S32K144 GPIOs Lab: Theory

- There are up to 89 GPIOs in the S32K144
  - 5 PORTs (PTA, PTB, PTC, PTD, PTE)
- 8 high current pins (up to 20 mA each):
  - PTD1, PTD0, PTD16, PTD15, PTB5, PTB4,PTE1, and PTE0
- Each I/O is interrupt capable
- Each I/O is DMA capable
- Support for edge or level sensitive
- Each can wake up MCU from low power modes
- Digital filter included for each I/O

Package	GPIOs	High current pins
100 LQFP	89	8 - PTD1 - PTD0 - PTD16 - PTD15 - PTB5 - PTB4 - PTE1 - PTE0
64 LQFP	59	8 - PTD1 - PTD0 - PTD16 - PTD15 - PTB5 - PTB4 - PTE1 - PTE0

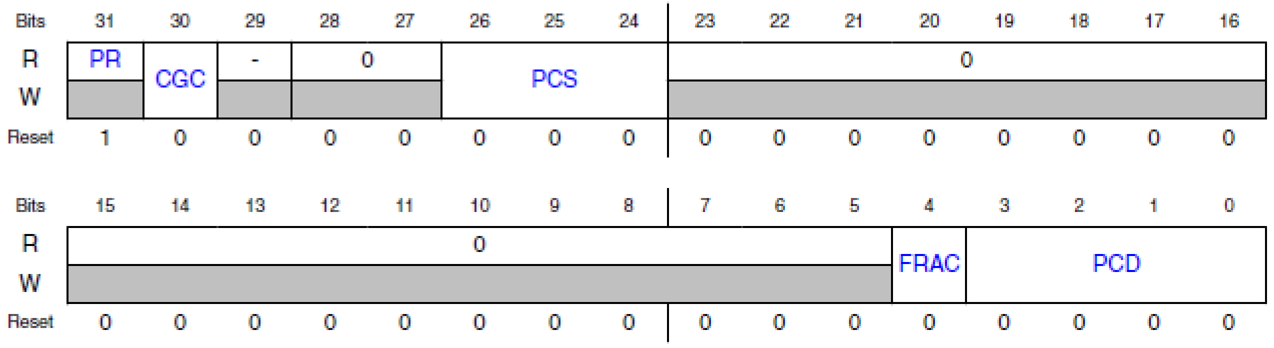
S32K144 GPIOs Lab: Theory


PCC
PORT
GPIO
Enable Clock of PORT modules
Configure Port Pin Functionality
Like Interrupt, Pull resister, DMA etc
Read/Write GPIO Pins

S32K144 GPIOs Lab: Theory

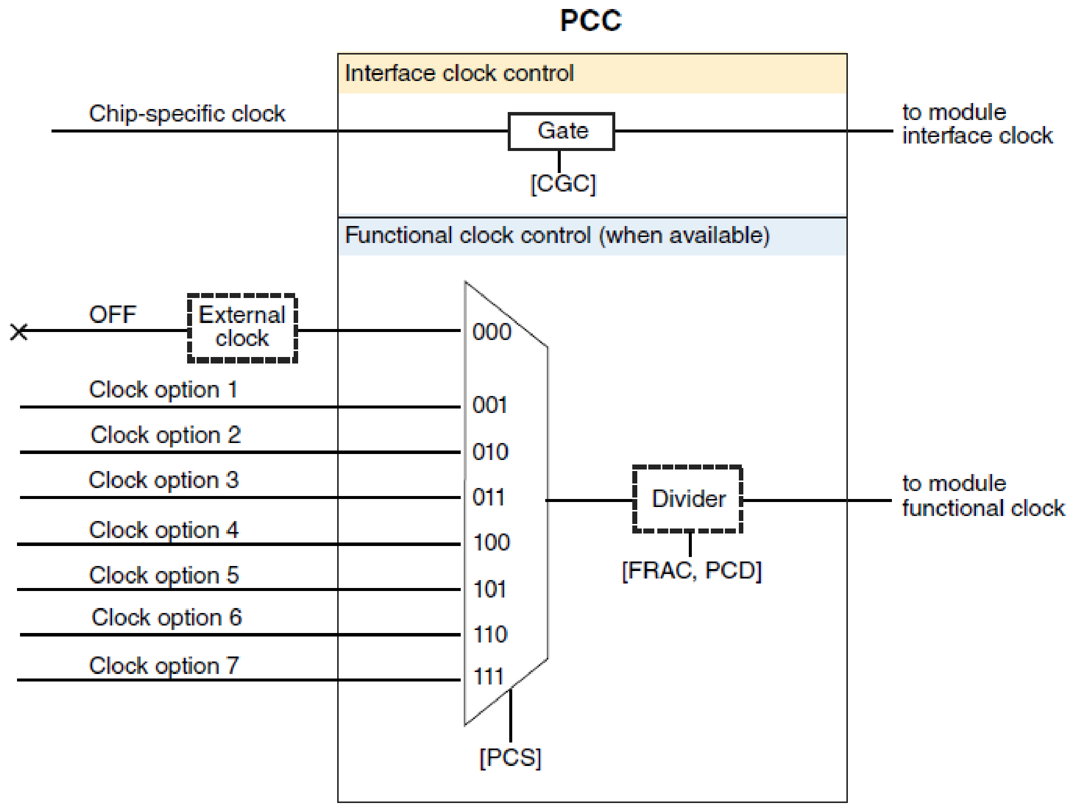
1. PCC module

- Modules can be individually turn on or off using the PCC module.
- Clock source for each peripheral can be selected from multiple sources.
- Before using a peripheral, turn on its clock.



Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PR	CGC	-	0		PCS			0							
W	▨		▨	▨	▨				▨	▨	▨	▨	▨	▨	▨	▨
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

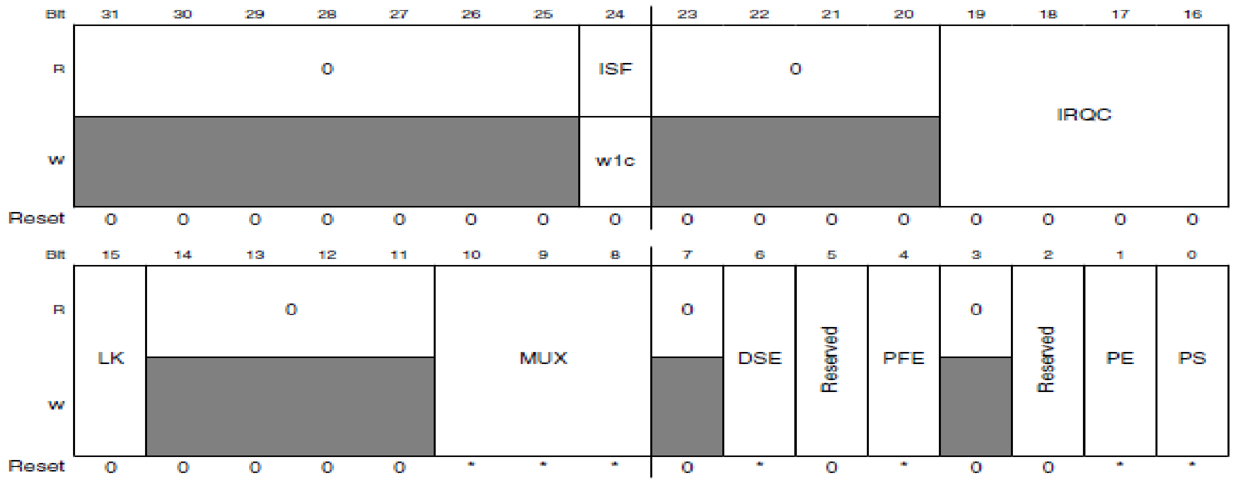
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0											FRAC	PCD			
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



S32K144 GPIOs Lab: Theory

2. PORT Module

- Each I/O is multiplexed with different functionalities
- I/O functionality is selected with PORTx->PCRn register, MUX bits.
- Alternative 1 (MUX=0b001) is GPIO functionality for all I/Os
- I/O interrupt configuration is controlled independently
- I/O Pull resistor is controlled independently



Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0							ISF	0				IRQC			
W								w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LK	0				MUX			0	DSE	Reserved	PFE	0	Reserved	PE	PS
W																
Reset	0	0	0	0	0	*	*	*	0	*	0	*	0	0	*	*

S32K144 GPIOs Lab: Theory

3. GPIO Module

- Each port pin is mapped to the following 32-bit GPIO registers, each bit represents a pin in the port x:

- GPIOx->PDOR.	Data Output
- GPIOx->PSOR.	Set Output
- GPIOx->PCOR.	Clear Output
- GPIOx->PTOR.	Toggle Output
- GPIOx->PDIR.	Input register
- GPIOx->PIDR.	Input disable register
- GPIOx-> PDDR.	Data Direction register

S32K144 GPIOs Lab: Theory

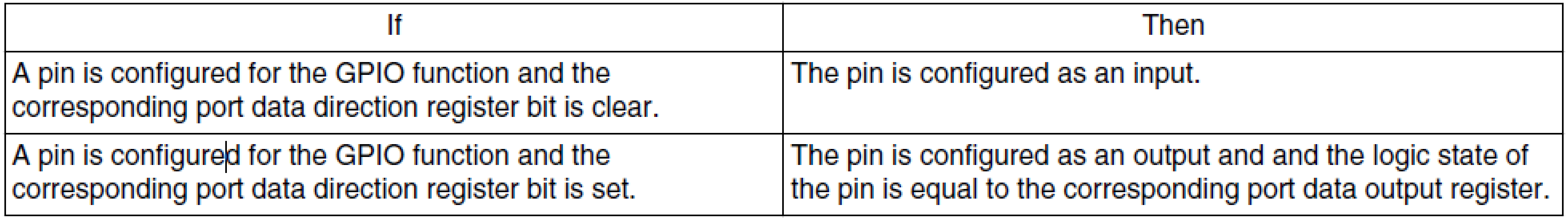
GPIO Direction selected with PDDR register.

GPIO INPUT

- Logic state available in PDIR register

GPIO OUTPUT

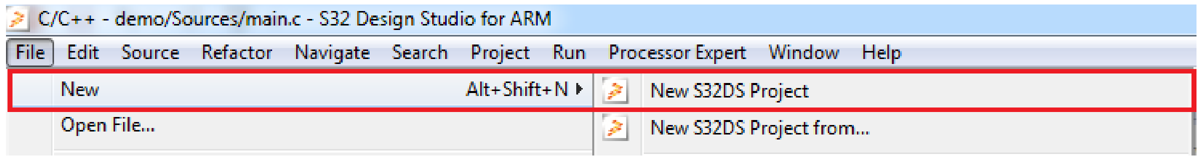
- Logic state controlled via PDOR or PCOR,PSOR and PTOR.



If	Then
A pin is configured for the GPIO function and the corresponding port data direction register bit is clear.	The pin is configured as an input.
A pin is configured for the GPIO function and the corresponding port data direction register bit is set.	The pin is configured as an output and and the logic state of the pin is equal to the corresponding port data output register.

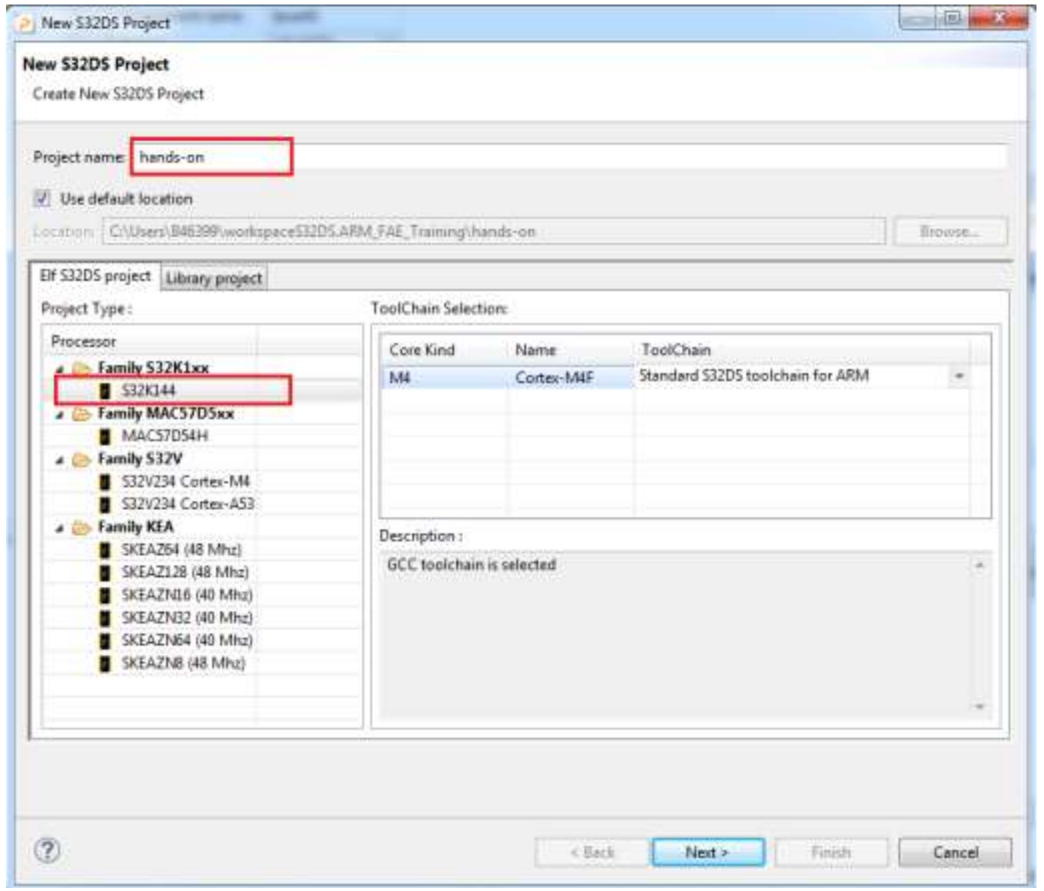
S32K144 GPIOs Lab: Initial Steps

- Create a new S32DS Project



C/C++ - demo/Sources/main.c - S32 Design Studio for ARM

File Edit Source Refactor Navigate Search Project Run Processor Expert Window Help
New Alt+Shift+N
Open File...
New S32DS Project
New S32DS Project from...



New S32DS Project

Create New S32DS Project

Project name: hands-on

☑ Use default location

Location: C:\Users\B46399\workspaceS32DS.ARM_FAE_Training\hands-on

Project Type:

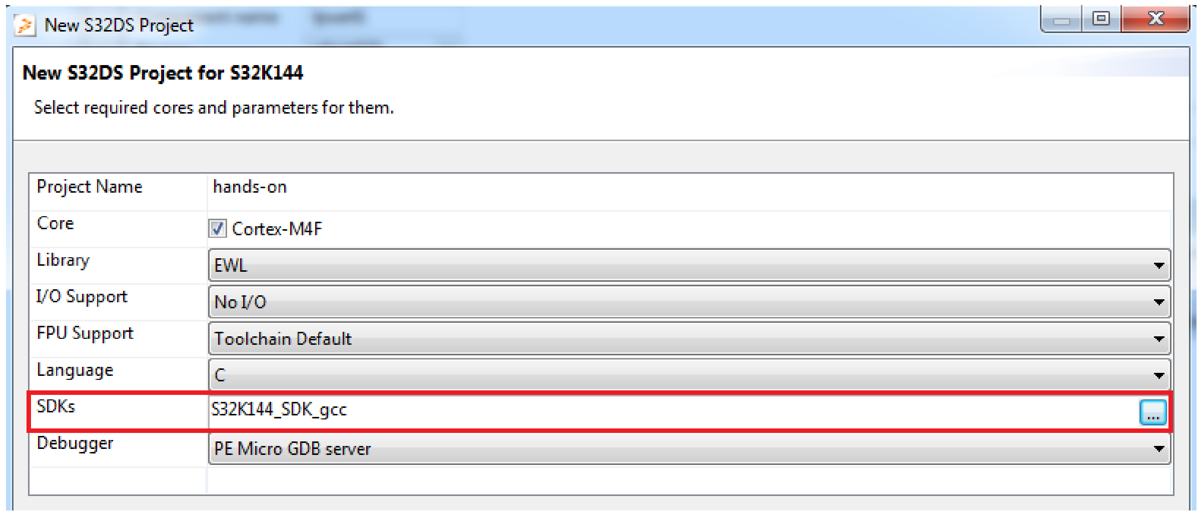
Processor
- Family S32K1xx
  - S32K144
- Family MAC57D5xx
  - MAC57D54H
- Family S32V
  - S32V234 Cortex-M4
  - S32V234 Cortex-A53
- Family KEA
  - SKEAZ64 (48 Mhz)
  - SKEAZ128 (48 Mhz)
  - SKEAZN16 (40 Mhz)
  - SKEAZN32 (40 Mhz)
  - SKEAZN64 (40 Mhz)
  - SKEAZN8 (48 Mhz)

ToolChain Selection:

Core Kind	Name	ToolChain
M4	Cortex-M4F	Standard S32DS toolchain for ARM

Description:

GCC toolchain is selected



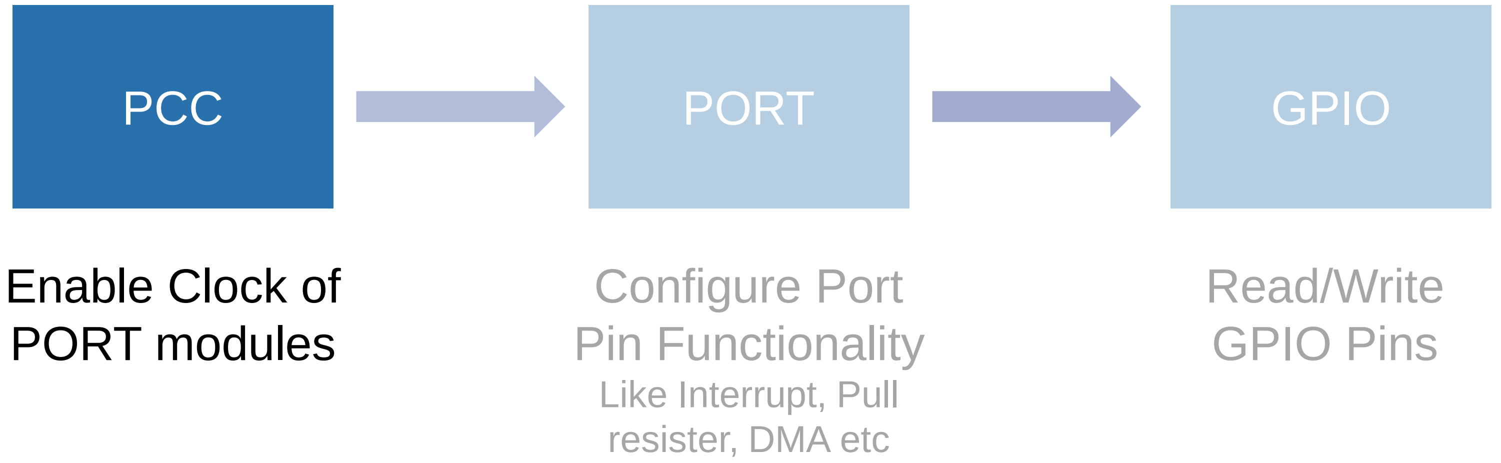
New S32DS Project

New S32DS Project for S32K144

Select required cores and parameters for them.

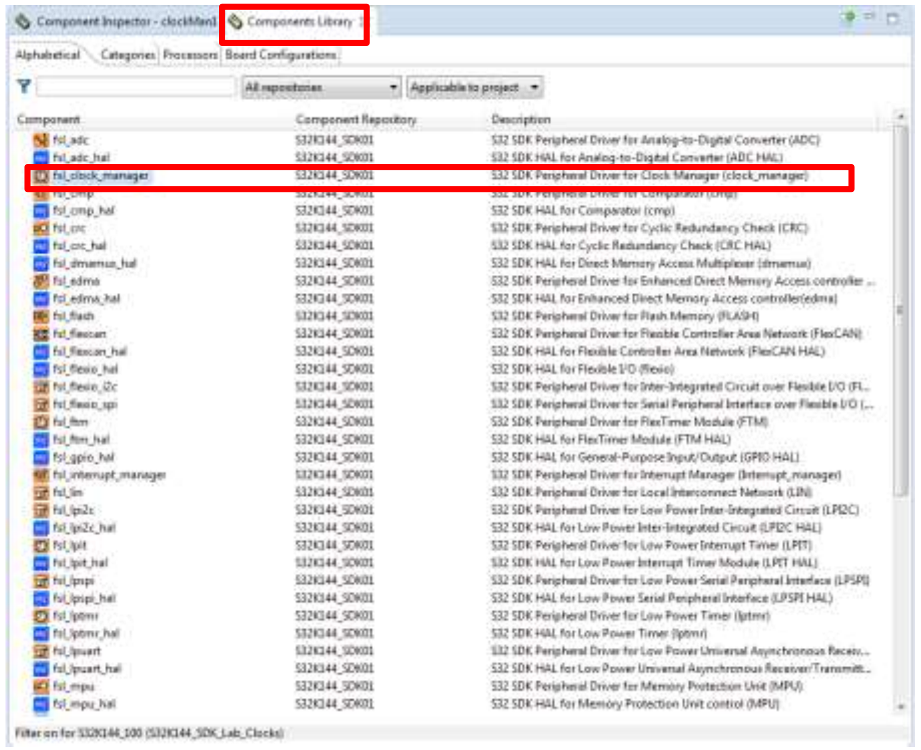
Project Name	hands-on
Core	☑ Cortex-M4F
Library	EWL
I/O Support	No I/O
FPU Support	Toolchain Default
Language	C
SDKs	S32K144_SDK_gcc
Debugger	PE Micro GDB server

S32K144 GPIOs Lab: Step-1


PCC
PORT
GPIO
Enable Clock of PORT modules
Configure Port Pin Functionality
Like Interrupt, Pull resister, DMA etc
Read/Write GPIO Pins

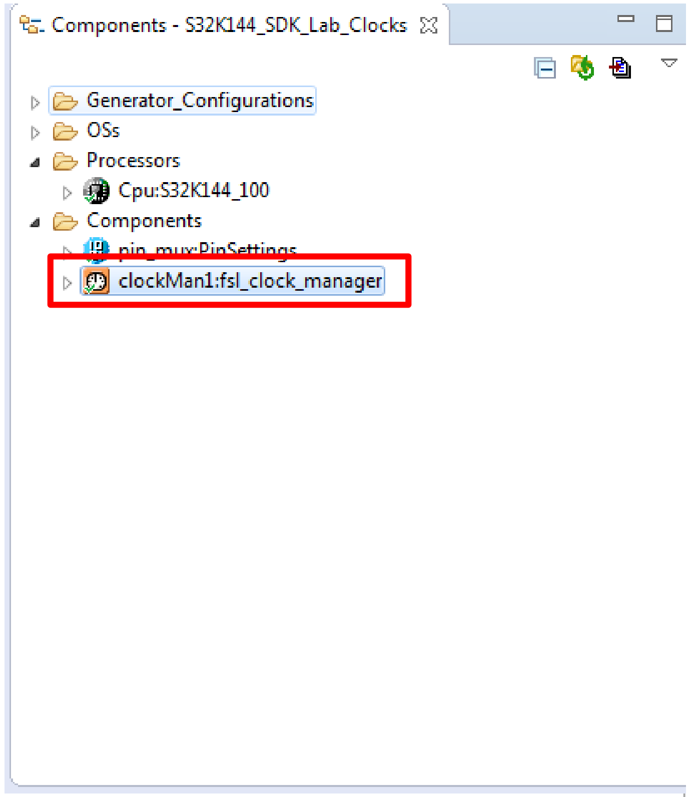
S32K144 GPIOs Lab: Add Clock manager driver

- Go to **Components Library** window.
- Select the **clock_manager** in the **Alphabetical** Tab
- Double click **clock_manager** to add to your project.
- Clock component should appear on the component window.



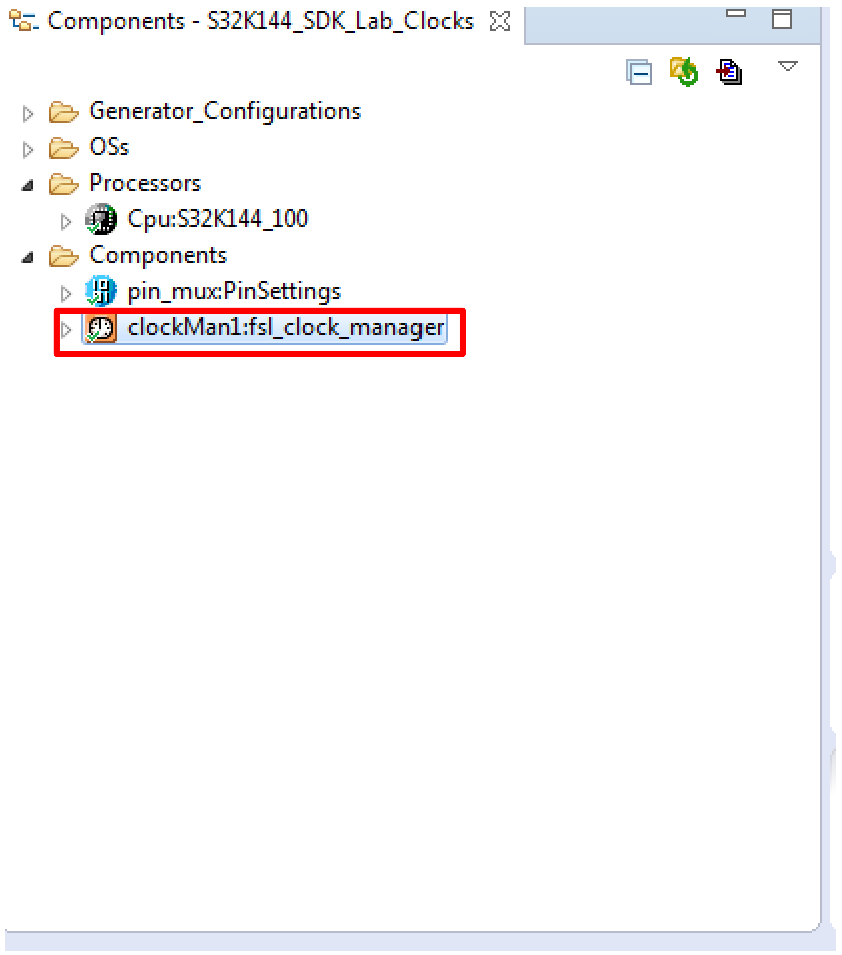
Component	Component Repository	Description
fsl_adc	S32K144_SDK01	S32 SDK Peripheral Driver for Analog-to-Digital Converter (ADC)
fsl_adc_hal	S32K144_SDK01	S32 SDK HAL for Analog-to-Digital Converter (ADC HAL)
fsl_clock_manager	S32K144_SDK01	S32 SDK Peripheral Driver for Clock Manager (clock_manager)
fsl_cmp	S32K144_SDK01	S32 SDK Peripheral Driver for Comparator (cmp)
fsl_cmp_hal	S32K144_SDK01	S32 SDK HAL for Comparator (cmp)
fsl_crc	S32K144_SDK01	S32 SDK Peripheral Driver for Cyclic Redundancy Check (CRC)
fsl_crc_hal	S32K144_SDK01	S32 SDK HAL for Cyclic Redundancy Check (CRC HAL)
fsl_dmamux_hal	S32K144_SDK01	S32 SDK HAL for Direct Memory Access Multiplexer (dmamux)
fsl_edma	S32K144_SDK01	S32 SDK Peripheral Driver for Enhanced Direct Memory Access controller ...
fsl_edma_hal	S32K144_SDK01	S32 SDK HAL for Enhanced Direct Memory Access controller(edma)
fsl_flash	S32K144_SDK01	S32 SDK Peripheral Driver for Flash Memory (FLASH)
fsl_flexcan	S32K144_SDK01	S32 SDK Peripheral Driver for Flexible Controller Area Network (FlexCAN)
fsl_flexcan_hal	S32K144_SDK01	S32 SDK HAL for Flexible Controller Area Network (FlexCAN HAL)
fsl_flexio_hal	S32K144_SDK01	S32 SDK HAL for Flexible I/O (flexio)
fsl_flexio_i2c	S32K144_SDK01	S32 SDK Peripheral Driver for Inter-Integrated Circuit over Flexible I/O (Fl...
fsl_flexio_spi	S32K144_SDK01	S32 SDK Peripheral Driver for Serial Peripheral Interface over Flexible I/O (...
fsl_ftm	S32K144_SDK01	S32 SDK Peripheral Driver for FlexTimer Module (FTM)
fsl_ftm_hal	S32K144_SDK01	S32 SDK HAL for FlexTimer Module (FTM HAL)
fsl_gpio_hal	S32K144_SDK01	S32 SDK HAL for General-Purpose Input/Output (GPIO HAL)
fsl_interrupt_manager	S32K144_SDK01	S32 SDK Peripheral Driver for Interrupt Manager (Interrupt_manager)
fsl_lin	S32K144_SDK01	S32 SDK Peripheral Driver for Local Interconnect Network (LIN)
fsl_lpi2c	S32K144_SDK01	S32 SDK Peripheral Driver for Low Power Inter-Integrated Circuit (LPI2C)
fsl_lpi2c_hal	S32K144_SDK01	S32 SDK HAL for Low Power Inter-Integrated Circuit (LPI2C HAL)
fsl_lpit	S32K144_SDK01	S32 SDK Peripheral Driver for Low Power Interrupt Timer (LPIT)
fsl_lpit_hal	S32K144_SDK01	S32 SDK HAL for Low Power Interrupt Timer Module (LPIT HAL)
fsl_lpspi	S32K144_SDK01	S32 SDK Peripheral Driver for Low Power Serial Peripheral Interface (LPSPI)
fsl_lpspi_hal	S32K144_SDK01	S32 SDK HAL for Low Power Serial Peripheral Interface (LPSPI HAL)
fsl_lptmr	S32K144_SDK01	S32 SDK Peripheral Driver for Low Power Timer (lptmr)
fsl_lptmr_hal	S32K144_SDK01	S32 SDK HAL for Low Power Timer (lptmr)
fsl_lpuart	S32K144_SDK01	S32 SDK Peripheral Driver for Low Power Universal Asynchronous Receiv...
fsl_lpuart_hal	S32K144_SDK01	S32 SDK HAL for Low Power Universal Asynchronous Receiver/Transmitt...
fsl_mpu	S32K144_SDK01	S32 SDK Peripheral Driver for Memory Protection Unit (MPU)
fsl_mpu_hal	S32K144_SDK01	S32 SDK HAL for Memory Protection Unit control (MPU)

Filter on for S32K144_100 (S32K144_SDK_Lab_Clocks)



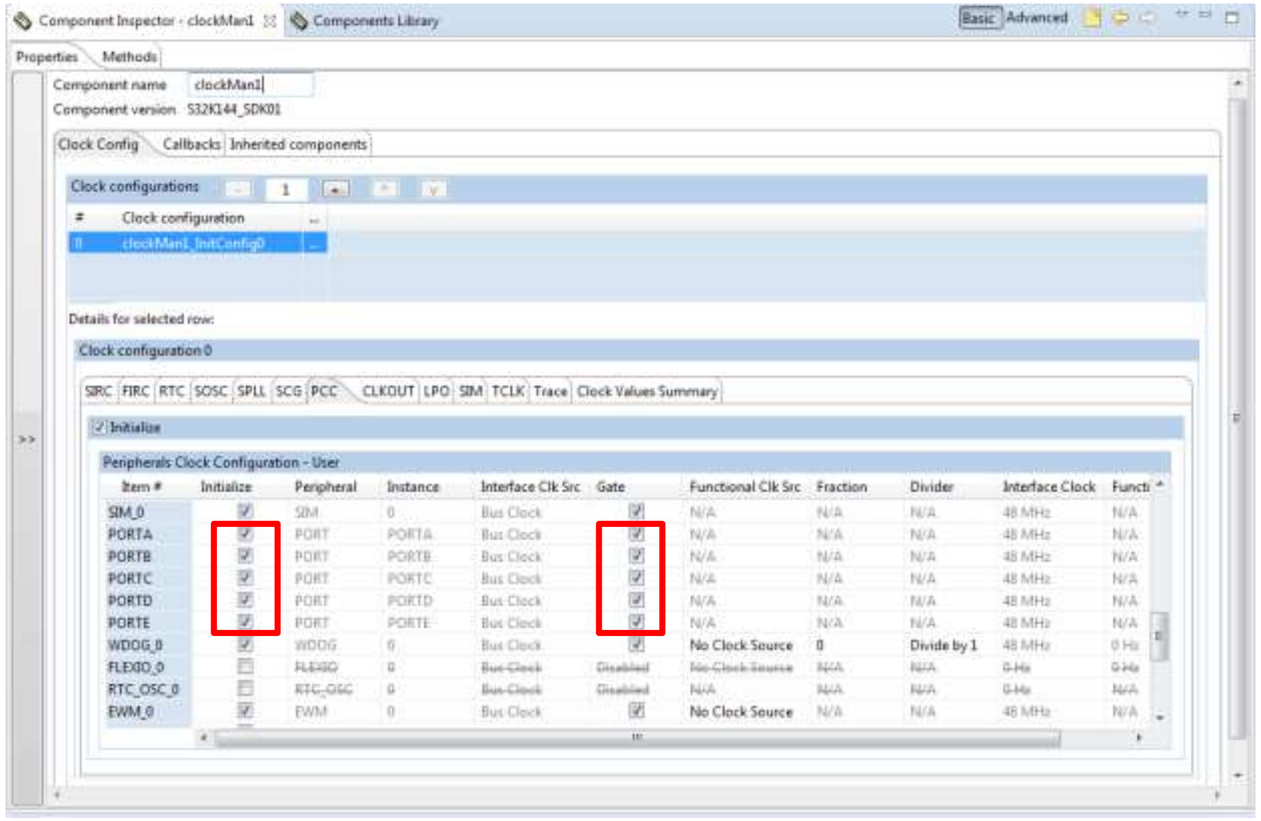
S32K144 GPIOs Lab: Enable PORTs Clock

- Select the **clock_manager** component in the **Components** window



S32K144 GPIOs Lab: Enable PORTs Clock

- Go to **Component Inspector** window of the **Clock Manager component**.
- Select the PCC tab
- Check the initialize box for PTA,PTB,PTC,PTD and PTE
- Check the gate box for PTA,PTB,PTC,PTD and PTE



Component name: clockMan1
Component version: S32K144_SDK01

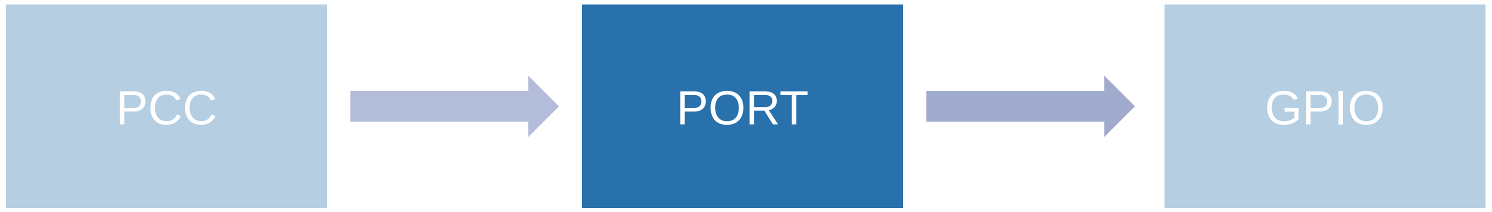
#	Clock configuration
0	clockMan1_InitConfig0

✓ Initialize

Peripherals Clock Configuration - User

Item #	Initialize	Peripheral	Instance	Interface Clk Src	Gate	Functional Clk Src	Fraction	Divider	Interface Clock	Functi
SIM_0	✓	SIM	0	Bus Clock	✓	N/A	N/A	N/A	48 MHz	N/A
PORTA	✓	PORT	PORTA	Bus Clock	✓	N/A	N/A	N/A	48 MHz	N/A
PORTB	✓	PORT	PORTB	Bus Clock	✓	N/A	N/A	N/A	48 MHz	N/A
PORTC	✓	PORT	PORTC	Bus Clock	✓	N/A	N/A	N/A	48 MHz	N/A
PORTD	✓	PORT	PORTD	Bus Clock	✓	N/A	N/A	N/A	48 MHz	N/A
PORTE	✓	PORT	PORTE	Bus Clock	✓	N/A	N/A	N/A	48 MHz	N/A
WDOG_0	✓	WDOG	0	Bus Clock	✓	No Clock Source	0	Divide by 1	48 MHz	0 Hz
FLEXIO_0	☐	~~FLEXIO~~	0	~~Bus Clock~~	~~Disabled~~	~~No Clock Source~~	~~N/A~~	~~N/A~~	~~0 Hz~~	~~0 Hz~~
RTC_OSC_0	☐	~~RTC_OSC~~	0	~~Bus Clock~~	~~Disabled~~	~~N/A~~	~~N/A~~	~~N/A~~	~~0 Hz~~	~~N/A~~
EWM_0	✓	EWM	0	Bus Clock	✓	No Clock Source	N/A	N/A	48 MHz	N/A

S32K144 GPIOs Lab: Step-2



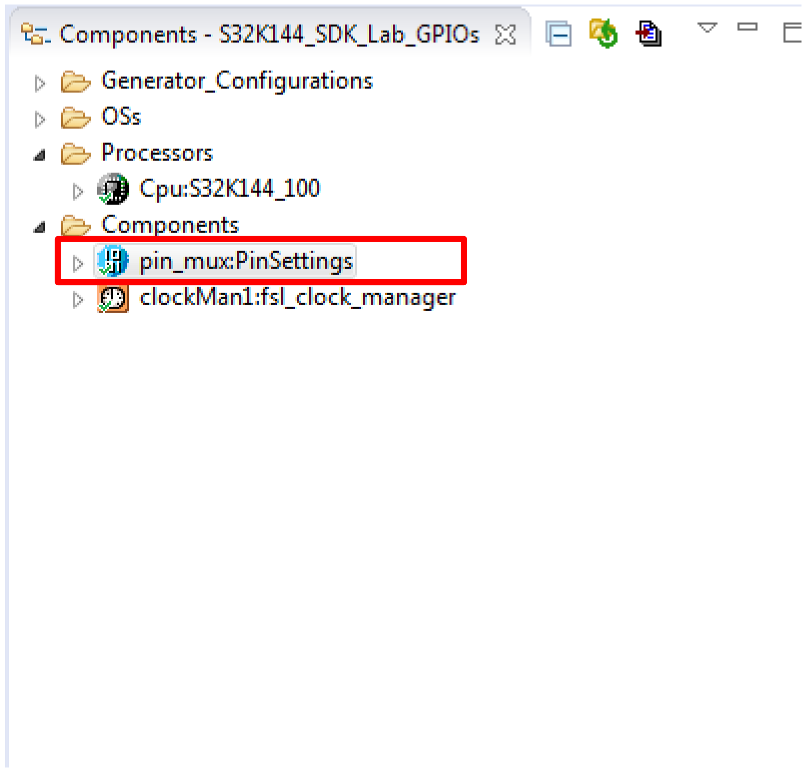
Enable Clock of PORT modules

Configure Port Pin Functionality
Like Interrupt, Pull resister, DMA etc

Read/Write GPIO Pins

S32K144 GPIOs Lab: Select I/O pins direction

- Select the **pin_mux** component in the **Components** window

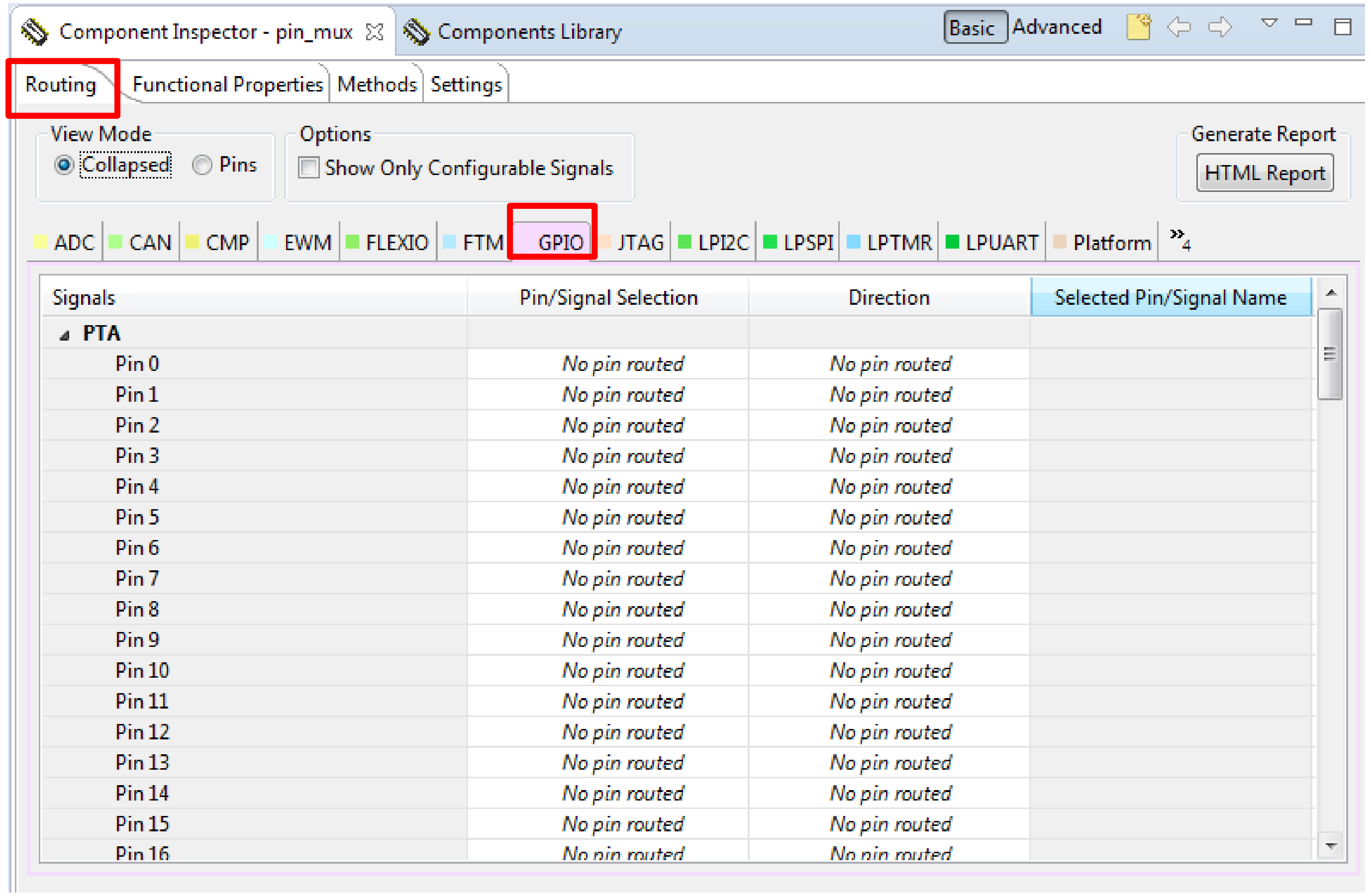


Components - S32K144_SDK_Lab_GPIOs

- Generator_Configurations
- OSs
- Processors
  - Cpu:S32K144_100
- Components
  - pin_mux:PinSettings
  - clockMan1:fsl_clock_manager

S32K144 GPIOs Lab: Select I/O pins direction

- In the **Component Inspector** window
- Select **GPIO** tab inside the **Routing** tab



View Mode: [✓ Collapsed] Pins

Options: ☐ Show Only Configurable Signals

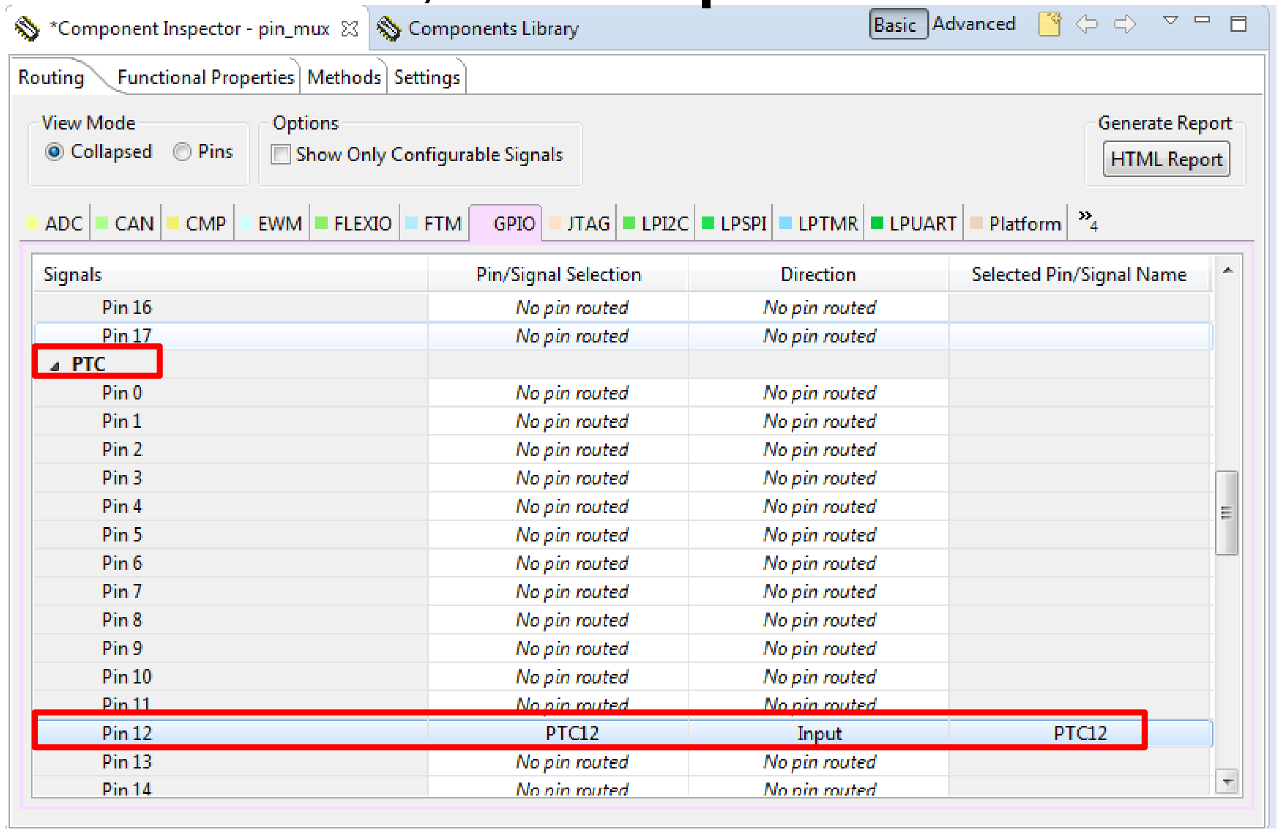
Signals	Pin/Signal Selection	Direction	Selected Pin/Signal Name
PTA			
Pin 0	No pin routed	No pin routed	
Pin 1	No pin routed	No pin routed	
Pin 2	No pin routed	No pin routed	
Pin 3	No pin routed	No pin routed	
Pin 4	No pin routed	No pin routed	
Pin 5	No pin routed	No pin routed	
Pin 6	No pin routed	No pin routed	
Pin 7	No pin routed	No pin routed	
Pin 8	No pin routed	No pin routed	
Pin 9	No pin routed	No pin routed	
Pin 10	No pin routed	No pin routed	
Pin 11	No pin routed	No pin routed	
Pin 12	No pin routed	No pin routed	
Pin 13	No pin routed	No pin routed	
Pin 14	No pin routed	No pin routed	
Pin 15	No pin routed	No pin routed	
Pin 16	No pin routed	No pin routed	

S32K144 GPIOs Lab: Select Input pin

Go to **PTC** and select pin **12**.

In the **Pin/Signal Selection** Colum, select **PTC12**.

In the **Direction** Colum, select **Input.**



View Mode: [✓ Collapsed] Pins

Options: Show Only Configurable Signals ☐

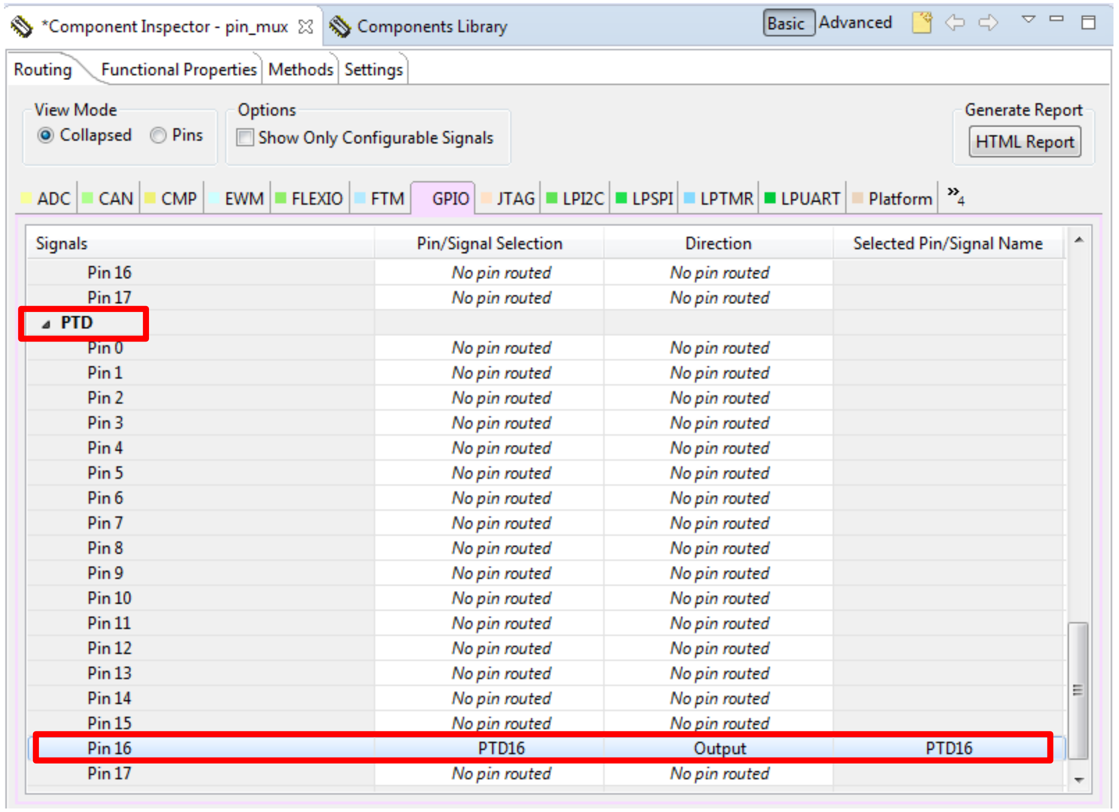
Signals	Pin/Signal Selection	Direction	Selected Pin/Signal Name
Pin 16	No pin routed	No pin routed	
Pin 17	No pin routed	No pin routed	
PTC			
Pin 0	No pin routed	No pin routed	
Pin 1	No pin routed	No pin routed	
Pin 2	No pin routed	No pin routed	
Pin 3	No pin routed	No pin routed	
Pin 4	No pin routed	No pin routed	
Pin 5	No pin routed	No pin routed	
Pin 6	No pin routed	No pin routed	
Pin 7	No pin routed	No pin routed	
Pin 8	No pin routed	No pin routed	
Pin 9	No pin routed	No pin routed	
Pin 10	No pin routed	No pin routed	
Pin 11	No pin routed	No pin routed	
Pin 12	PTC12	Input	PTC12
Pin 13	No pin routed	No pin routed	
Pin 14	No pin routed	No pin routed	

S32K144 GPIOs Lab: Select Output pin

Go to **PTD** and select pin **16**.

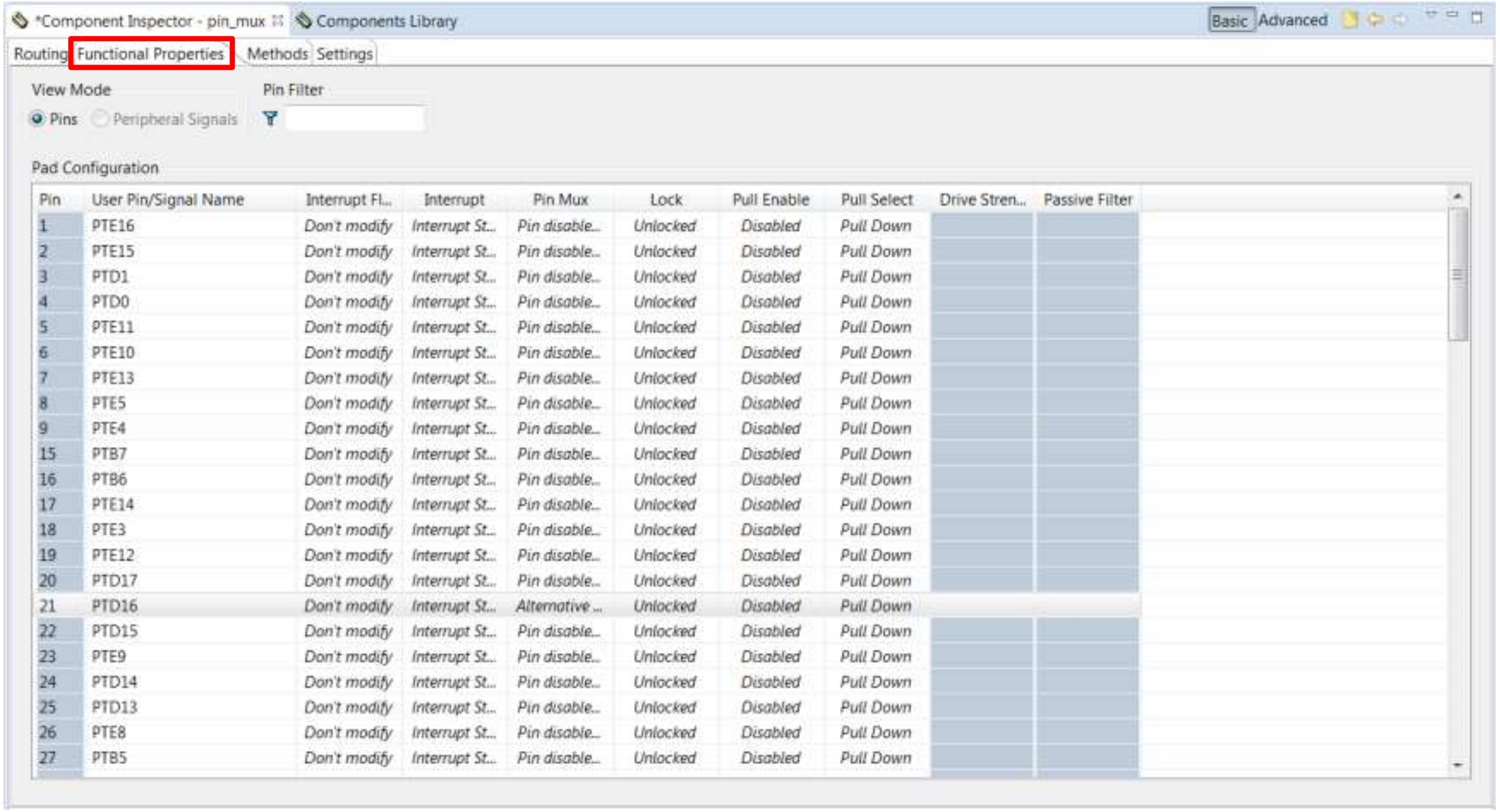
In the **Pin/Signal Selection** Colum, select **PTD16**.

In the **Direction** Colum, select **Output.**



Signals	Pin/Signal Selection	Direction	Selected Pin/Signal Name
Pin 16	*No pin routed*	*No pin routed*	
Pin 17	*No pin routed*	*No pin routed*	
PTD			
Pin 0	*No pin routed*	*No pin routed*	
Pin 1	*No pin routed*	*No pin routed*	
Pin 2	*No pin routed*	*No pin routed*	
Pin 3	*No pin routed*	*No pin routed*	
Pin 4	*No pin routed*	*No pin routed*	
Pin 5	*No pin routed*	*No pin routed*	
Pin 6	*No pin routed*	*No pin routed*	
Pin 7	*No pin routed*	*No pin routed*	
Pin 8	*No pin routed*	*No pin routed*	
Pin 9	*No pin routed*	*No pin routed*	
Pin 10	*No pin routed*	*No pin routed*	
Pin 11	*No pin routed*	*No pin routed*	
Pin 12	*No pin routed*	*No pin routed*	
Pin 13	*No pin routed*	*No pin routed*	
Pin 14	*No pin routed*	*No pin routed*	
Pin 15	*No pin routed*	*No pin routed*	
Pin 16	PTD16	Output	PTD16
Pin 17	*No pin routed*	*No pin routed*	

S32K144 GPIOs Lab: Port Functional Properties



View Mode

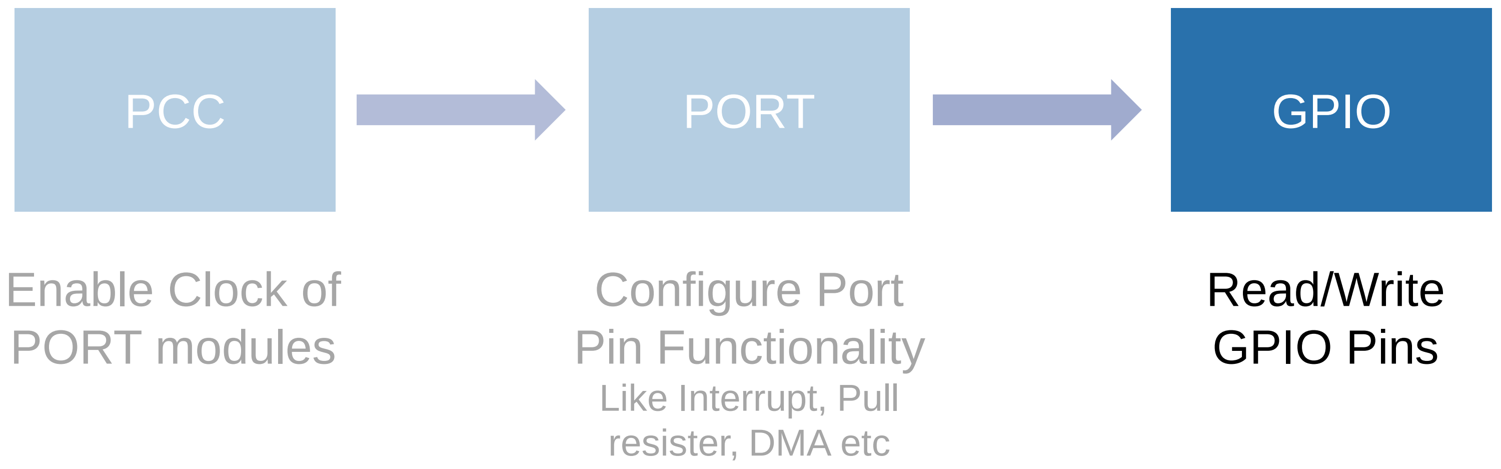
[✓ Pins] Peripheral Signals

Pin Filter

Pad Configuration

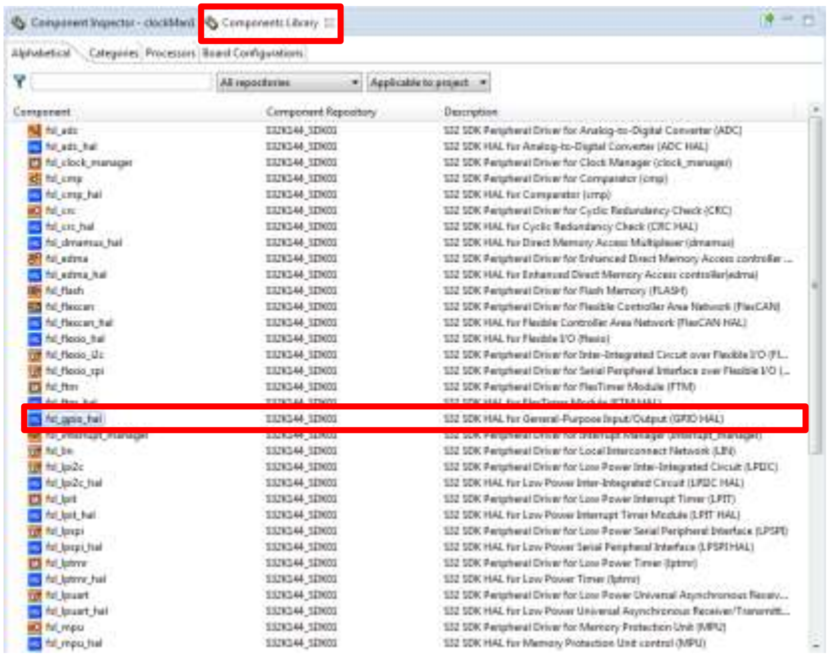
Pin	User Pin/Signal Name	Interrupt Fl...	Interrupt	Pin Mux	Lock	Pull Enable	Pull Select	Drive Stren...	Passive Filter
1	PTE16	Don't modify	Interrupt St...	Pin disable...	Unlocked	Disabled	Pull Down		
2	PTE15	Don't modify	Interrupt St...	Pin disable...	Unlocked	Disabled	Pull Down		
3	PTD1	Don't modify	Interrupt St...	Pin disable...	Unlocked	Disabled	Pull Down		
4	PTD0	Don't modify	Interrupt St...	Pin disable...	Unlocked	Disabled	Pull Down		
5	PTE11	Don't modify	Interrupt St...	Pin disable...	Unlocked	Disabled	Pull Down		
6	PTE10	Don't modify	Interrupt St...	Pin disable...	Unlocked	Disabled	Pull Down		
7	PTE13	Don't modify	Interrupt St...	Pin disable...	Unlocked	Disabled	Pull Down		
8	PTE5	Don't modify	Interrupt St...	Pin disable...	Unlocked	Disabled	Pull Down		
9	PTE4	Don't modify	Interrupt St...	Pin disable...	Unlocked	Disabled	Pull Down		
15	PTB7	Don't modify	Interrupt St...	Pin disable...	Unlocked	Disabled	Pull Down		
16	PTB6	Don't modify	Interrupt St...	Pin disable...	Unlocked	Disabled	Pull Down		
17	PTE14	Don't modify	Interrupt St...	Pin disable...	Unlocked	Disabled	Pull Down		
18	PTE3	Don't modify	Interrupt St...	Pin disable...	Unlocked	Disabled	Pull Down		
19	PTE12	Don't modify	Interrupt St...	Pin disable...	Unlocked	Disabled	Pull Down		
20	PTD17	Don't modify	Interrupt St...	Pin disable...	Unlocked	Disabled	Pull Down		
21	PTD16	Don't modify	Interrupt St...	Alternative ...	Unlocked	Disabled	Pull Down		
22	PTD15	Don't modify	Interrupt St...	Pin disable...	Unlocked	Disabled	Pull Down		
23	PTE9	Don't modify	Interrupt St...	Pin disable...	Unlocked	Disabled	Pull Down		
24	PTD14	Don't modify	Interrupt St...	Pin disable...	Unlocked	Disabled	Pull Down		
25	PTD13	Don't modify	Interrupt St...	Pin disable...	Unlocked	Disabled	Pull Down		
26	PTE8	Don't modify	Interrupt St...	Pin disable...	Unlocked	Disabled	Pull Down		
27	PTB5	Don't modify	Interrupt St...	Pin disable...	Unlocked	Disabled	Pull Down		

S32K144 GPIOs Lab: Step-3

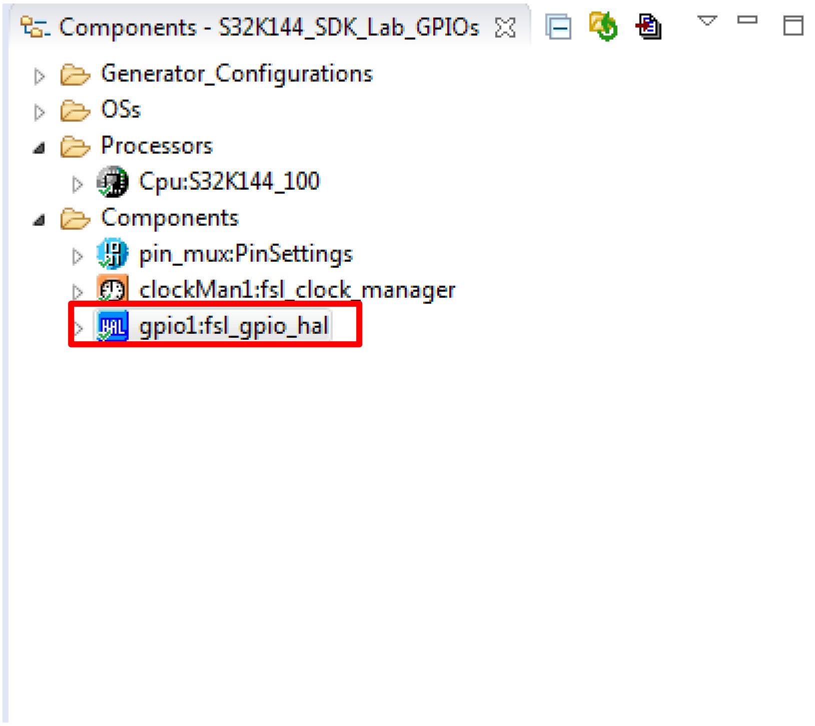

PCC
PORT
GPIO
Enable Clock of PORT modules
Configure Port Pin Functionality
Like Interrupt, Pull resister, DMA etc
Read/Write GPIO Pins

S32K144 GPIOs Lab: Add GPIO Driver

- Go to **Components Library** window.
- Select the **gpio_hal** in the Alphabetical tab.
- Double click **gpio_hal** to add to your project.
- GPIO component should appear on the component window.



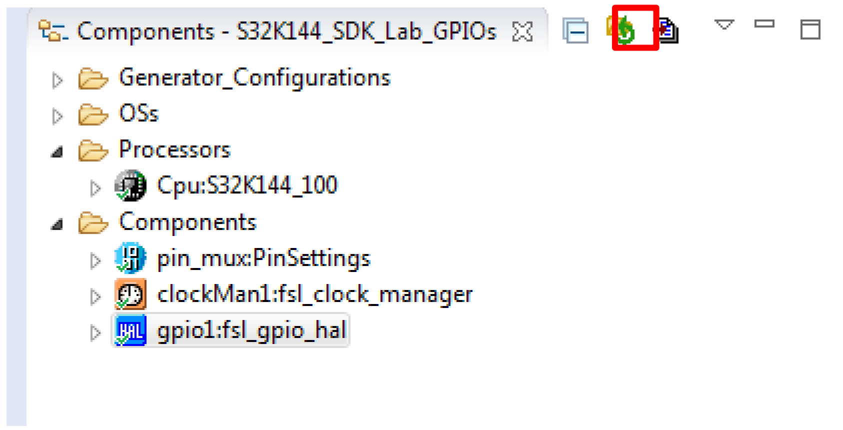
Component	Component Repository	Description
fsl_adc	S32K144_SDK01	S32 SDK Peripheral Driver for Analog-to-Digital Converter (ADC)
fsl_adc_hal	S32K144_SDK01	S32 SDK HAL for Analog-to-Digital Converter (ADC HAL)
fsl_clock_manager	S32K144_SDK01	S32 SDK Peripheral Driver for Clock Manager (clock_manager)
fsl_cmp	S32K144_SDK01	S32 SDK Peripheral Driver for Comparator (cmp)
fsl_cmp_hal	S32K144_SDK01	S32 SDK HAL for Comparator (cmp)
fsl_crc	S32K144_SDK01	S32 SDK Peripheral Driver for Cyclic Redundancy Check (CRC)
fsl_crc_hal	S32K144_SDK01	S32 SDK HAL for Cyclic Redundancy Check (CRC HAL)
fsl_dmamux_hal	S32K144_SDK01	S32 SDK HAL for Direct Memory Access Multiplexer (dmamux)
fsl_edma	S32K144_SDK01	S32 SDK Peripheral Driver for Enhanced Direct Memory Access controller ...
fsl_edma_hal	S32K144_SDK01	S32 SDK HAL for Enhanced Direct Memory Access controller(edma)
fsl_flash	S32K144_SDK01	S32 SDK Peripheral Driver for Flash Memory (FLASH)
fsl_flexcan	S32K144_SDK01	S32 SDK Peripheral Driver for Flexible Controller Area Network (FlexCAN)
fsl_flexcan_hal	S32K144_SDK01	S32 SDK HAL for Flexible Controller Area Network (FlexCAN HAL)
fsl_flexio_hal	S32K144_SDK01	S32 SDK HAL for Flexible I/O (flexio)
fsl_flexio_i2c	S32K144_SDK01	S32 SDK Peripheral Driver for Inter-Integrated Circuit over Flexible I/O (FL...
fsl_flexio_spi	S32K144_SDK01	S32 SDK Peripheral Driver for Serial Peripheral Interface over Flexible I/O (...
fsl_ftm	S32K144_SDK01	S32 SDK Peripheral Driver for FlexTimer Module (FTM)
fsl_ftm_hal	S32K144_SDK01	S32 SDK HAL for FlexTimer Module (FTM HAL)
fsl_gpio_hal	S32K144_SDK01	S32 SDK HAL for General-Purpose Input/Output (GPIO HAL)
fsl_interrupt_manager	S32K144_SDK01	S32 SDK Peripheral Driver for Interrupt Manager (interrupt_manager)
fsl_lin	S32K144_SDK01	S32 SDK Peripheral Driver for Local Interconnect Network (LIN)
fsl_lpi2c	S32K144_SDK01	S32 SDK Peripheral Driver for Low Power Inter-Integrated Circuit (LPI2C)
fsl_lpi2c_hal	S32K144_SDK01	S32 SDK HAL for Low Power Inter-Integrated Circuit (LPI2C HAL)
fsl_lpit	S32K144_SDK01	S32 SDK Peripheral Driver for Low Power Interrupt Timer (LPIT)
fsl_lpit_hal	S32K144_SDK01	S32 SDK HAL for Low Power Interrupt Timer Module (LPIT HAL)
fsl_lpspi	S32K144_SDK01	S32 SDK Peripheral Driver for Low Power Serial Peripheral Interface (LPSPI)
fsl_lpspi_hal	S32K144_SDK01	S32 SDK HAL for Low Power Serial Peripheral Interface (LPSPI HAL)
fsl_lptmr	S32K144_SDK01	S32 SDK Peripheral Driver for Low Power Timer (lptmr)
fsl_lptmr_hal	S32K144_SDK01	S32 SDK HAL for Low Power Timer (lptmr)
fsl_lpuart	S32K144_SDK01	S32 SDK Peripheral Driver for Low Power Universal Asynchronous Receiv...
fsl_lpuart_hal	S32K144_SDK01	S32 SDK HAL for Low Power Universal Asynchronous Receiver/Transmitt...
fsl_mpu	S32K144_SDK01	S32 SDK Peripheral Driver for Memory Protection Unit (MPU)
fsl_mpu_hal	S32K144_SDK01	S32 SDK HAL for Memory Protection Unit control (MPU)



- Generator_Configurations
- OSs
- Processors
  - Cpu:S32K144_100
- Components
  - pin_mux:PinSettings
  - clockMan1:fsl_clock_manager
  - gpio1:fsl_gpio_hal

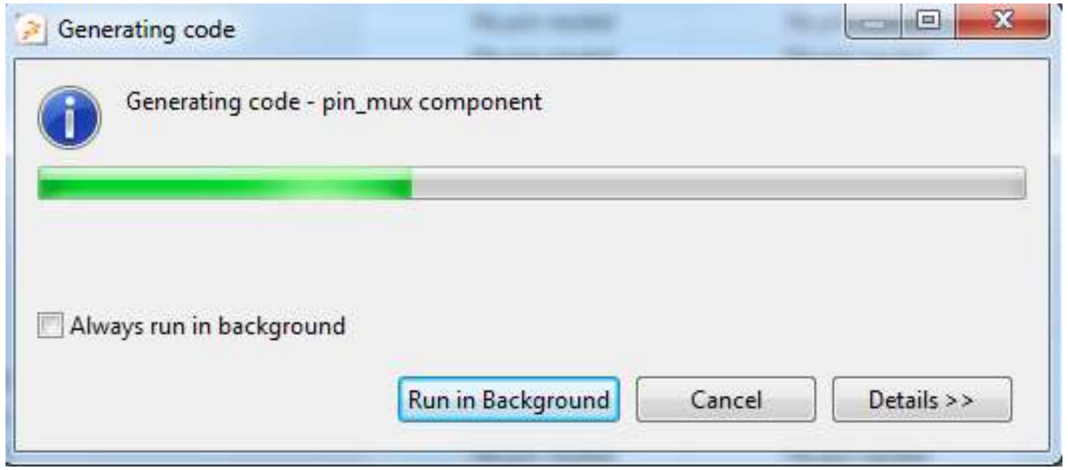
S32K144 GPIOs Lab: Generate the code

- To generate the code for the configuration select, click the **generate code** icon in the **Components** window.
- Wait for the code to be generated.



Components - S32K144_SDK_Lab_GPIOs

- Generator_Configurations
- OSs
- Processors
  - Cpu:S32K144_100
- Components
  - pin_mux:PinSettings
  - clockMan1:fsl_clock_manager
  - gpio1:fsl_gpio_hal



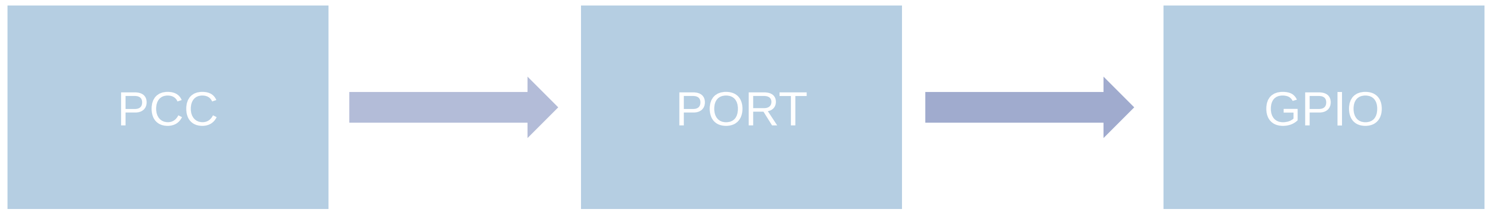
Generating code

Generating code - pin_mux component

Always run in background

S32K144 Clocks Lab: Step-4

Create an Application



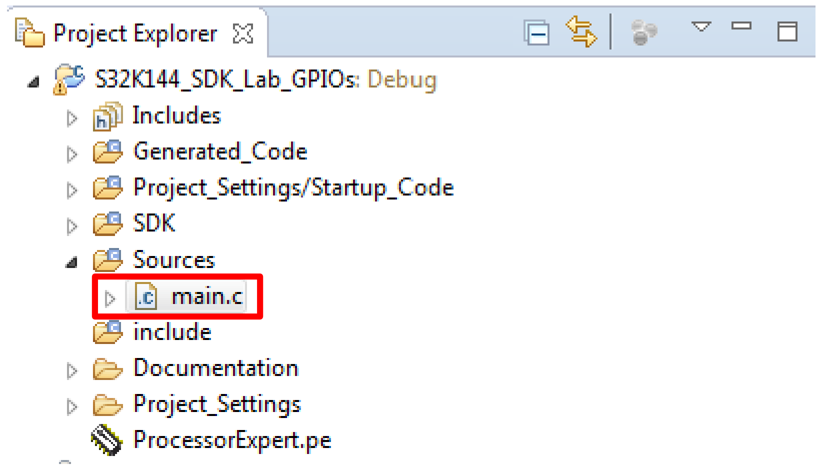
Enable Clock of PORT modules

Configure Port Pin Functionality
Like Interrupt, Pull resister, DMA etc

Read/Write GPIO Pins

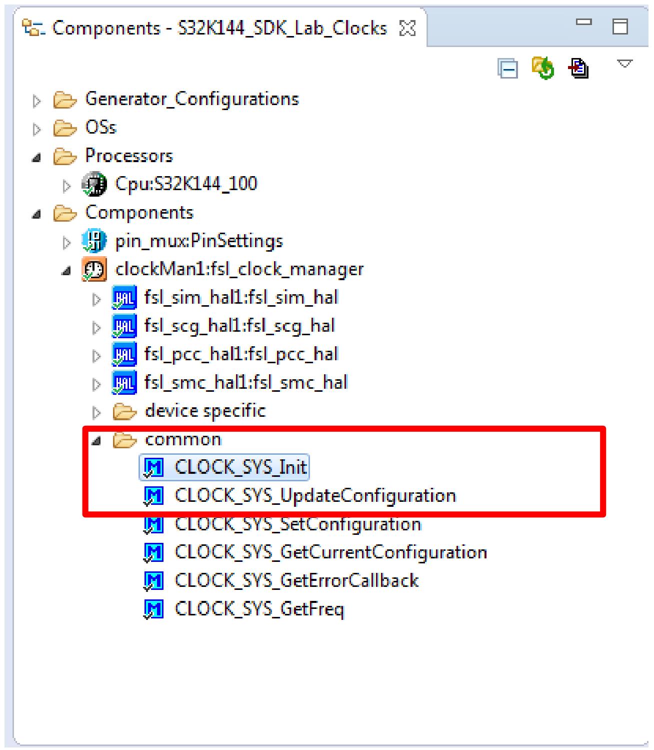
S32K144 GPIOs Lab: Open the main.c

- In the project window double click the main.c file to open it

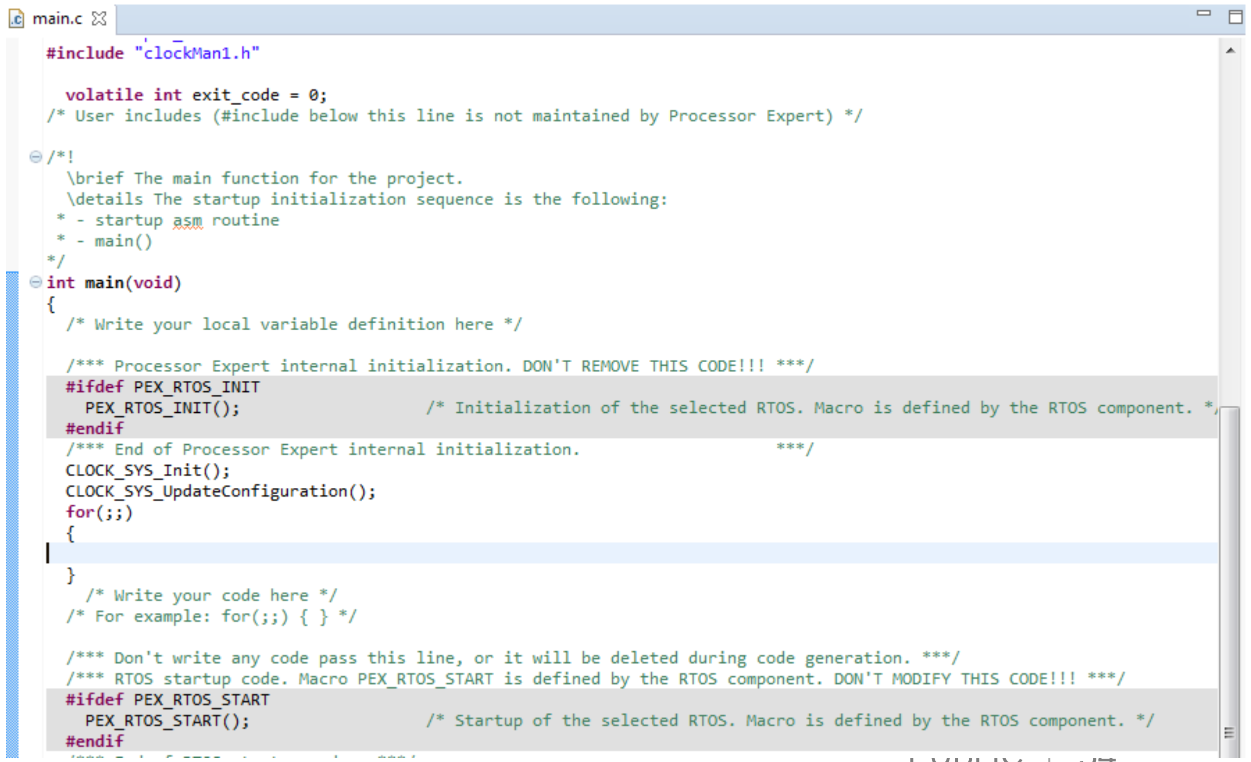


S32K144 GPIOs Lab: Add Init and Update Configuration functions

- Expand the **clock_manager** component in the **Components** Window
- Drag and drop the **CLOCK_SYS_Init** function into main.
- Drag and drop the **CLOCK_SYS_UpdateConfiguration** function into main.


Components - S32K144_SDK_Lab_Clocks

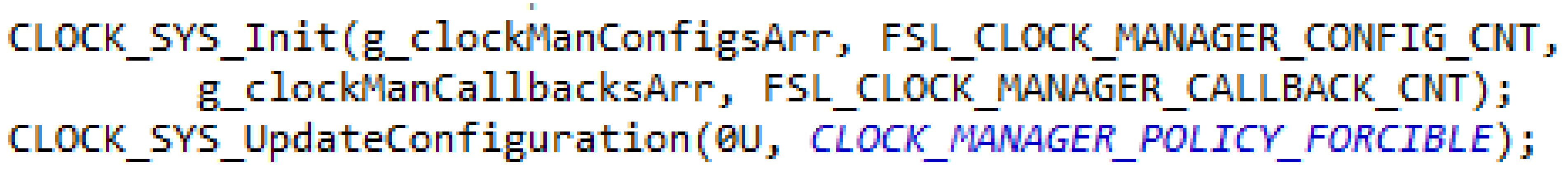
```
Generator_Configurations
OSs
Processors
  Cpu:S32K144_100
Components
  pin_mux:PinSettings
  clockMan1:fsl_clock_manager
    fsl_sim_hal1:fsl_sim_hal
    fsl_scg_hal1:fsl_scg_hal
    fsl_pcc_hal1:fsl_pcc_hal
    fsl_smc_hal1:fsl_smc_hal
    device specific
    common
      CLOCK_SYS_Init
      CLOCK_SYS_UpdateConfiguration
      CLOCK_SYS_SetConfiguration
      CLOCK_SYS_GetCurrentConfiguration
      CLOCK_SYS_GetErrorCallback
      CLOCK_SYS_GetFreq
```


main.c

```c
#include "clockMan1.h"

  volatile int exit_code = 0;
/* User includes (#include below this line is not maintained by Processor Expert) */

/*!
  \brief The main function for the project.
  \details The startup initialization sequence is the following:
 * - startup asm routine
 * - main()
*/
int main(void)
{
  /* Write your local variable definition here */

  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  #ifdef PEX_RTOS_INIT
    PEX_RTOS_INIT();                   /* Initialization of the selected RTOS. Macro is defined by the RTOS component. */
  #endif
  /*** End of Processor Expert internal initialization.                    ***/
  CLOCK_SYS_Init();
  CLOCK_SYS_UpdateConfiguration();
  for(;;)
  {

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

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

S32K144 GPIOs Lab: Add Init and Update Configuration functions

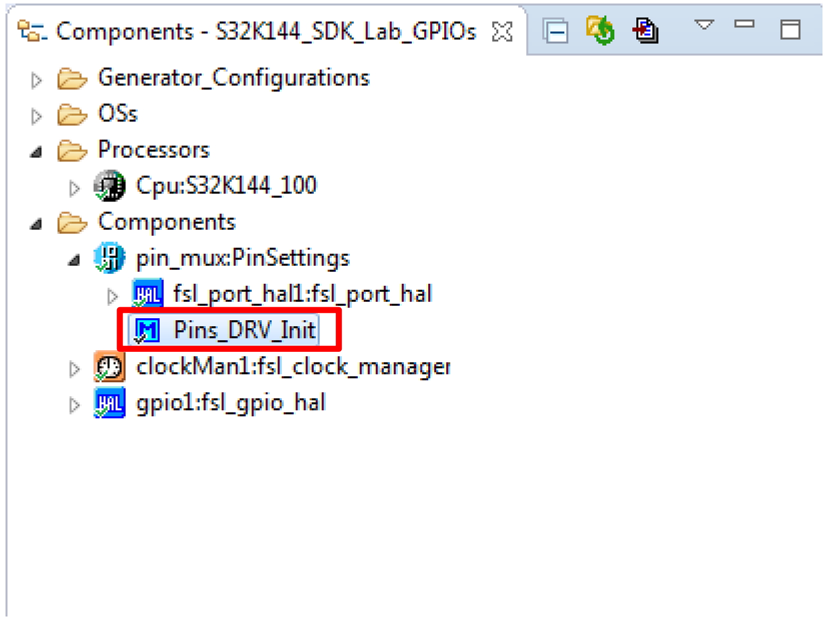
- In the **CLOCK_SYS_Init** function add the following parameters.
  - g_clockManConfigsArr,
  - CLOCK_MANAGER_CONFIG_CNT,
  - g_clockManCallbacksArr,
  - CLOCK_MANAGER_CALLBACK_CNT
- In the **CLOCK_SYS_UpdateConfiguration** add the following parameters.
  - 0U,
  - CLOCK_MANAGER_POLICY_FORCIBLE



```
CLOCK_SYS_Init(g_clockManConfigsArr, FSL_CLOCK_MANAGER_CONFIG_CNT,
        g_clockManCallbacksArr, FSL_CLOCK_MANAGER_CALLBACK_CNT);
CLOCK_SYS_UpdateConfiguration(0U, CLOCK_MANAGER_POLICY_FORCIBLE);
```

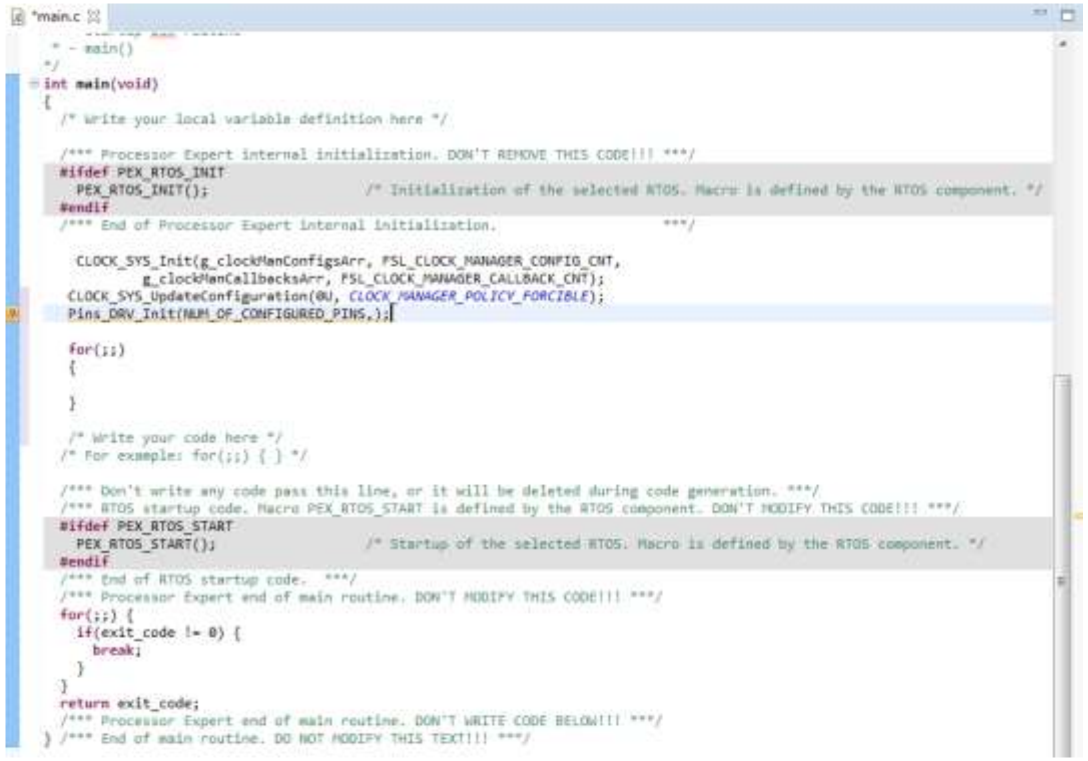
S32K144 GPIOs Lab: Initialize pins

- Expand the **pin_mux** component in the **Components** Window.
- Drag and drop the **Pins_DRV_Init** function inside the, into main, below the clock configuration



Components - S32K144_SDK_Lab_GPIOs

```
▷ Generator_Configurations
▷ OSs
◢ Processors
    ▷ Cpu:S32K144_100
◢ Components
    ◢ pin_mux:PinSettings
        ▷ fsl_port_hal1:fsl_port_hal
          Pins_DRV_Init
    ▷ clockMan1:fsl_clock_manager
    ▷ gpio1:fsl_gpio_hal
```

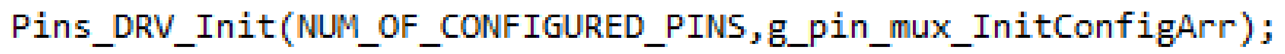


*main.c

```
 * - main()
 */
int main(void)
{
  /* Write your local variable definition here */

  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  #ifdef PEX_RTOS_INIT
    PEX_RTOS_INIT();                   /* Initialization of the selected RTOS. Macro is defined by the RTOS component. */
  #endif
  /*** End of Processor Expert internal initialization.                    ***/

    CLOCK_SYS_Init(g_clockManConfigsArr, FSL_CLOCK_MANAGER_CONFIG_CNT,
            g_clockManCallbacksArr, FSL_CLOCK_MANAGER_CALLBACK_CNT);
   CLOCK_SYS_UpdateConfiguration(0U, CLOCK_MANAGER_POLICY_FORCIBLE);
   Pins_DRV_Init(NUM_OF_CONFIGURED_PINS,);

   for(;;)
   {

   }

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

  /*** Don't write any code pass this line, or it will be deleted during code generation. ***/
  /*** RTOS startup code. Macro PEX_RTOS_START is defined by the RTOS component. DON'T MODIFY THIS CODE!!! ***/
  #ifdef PEX_RTOS_START
    PEX_RTOS_START();                  /* Startup of the selected RTOS. Macro is defined by the RTOS component. */
  #endif
  /*** End of RTOS startup code.  ***/
  /*** Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! ***/
  for(;;) {
    if(exit_code != 0) {
      break;
    }
  }
  return exit_code;
  /*** Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! ***/
} /*** End of main routine. DO NOT MODIFY THIS TEXT!!! ***/
```

S32K144 GPIOs Lab: Initialize pins

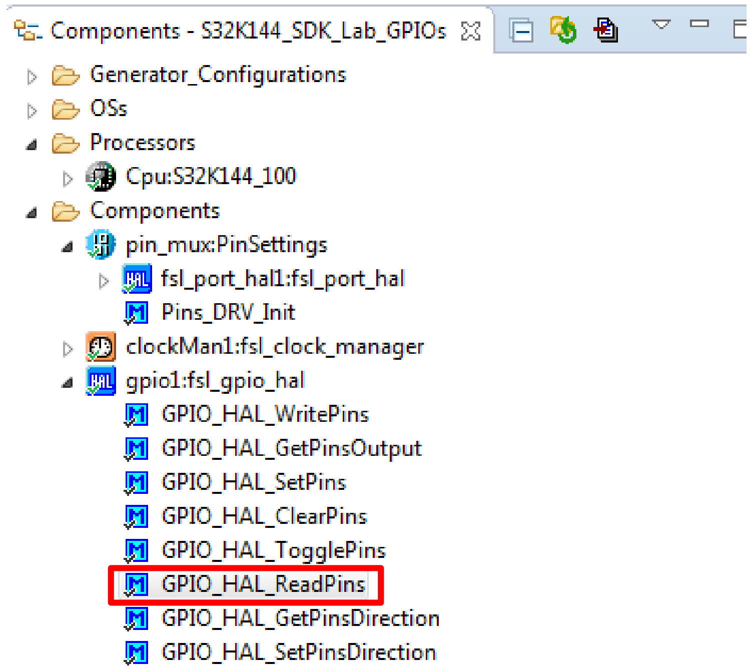
- **Pins_DRV_Init** function receives two parameters:
  - Number of pins to configure
  - Configuration structure.
- The number of pins to configure is included by default
- The configuration structure is already created, with the name **g_pin_mux_InitConfigArr**
- Add the configuration structure into the **Pins_DRV_Init** function



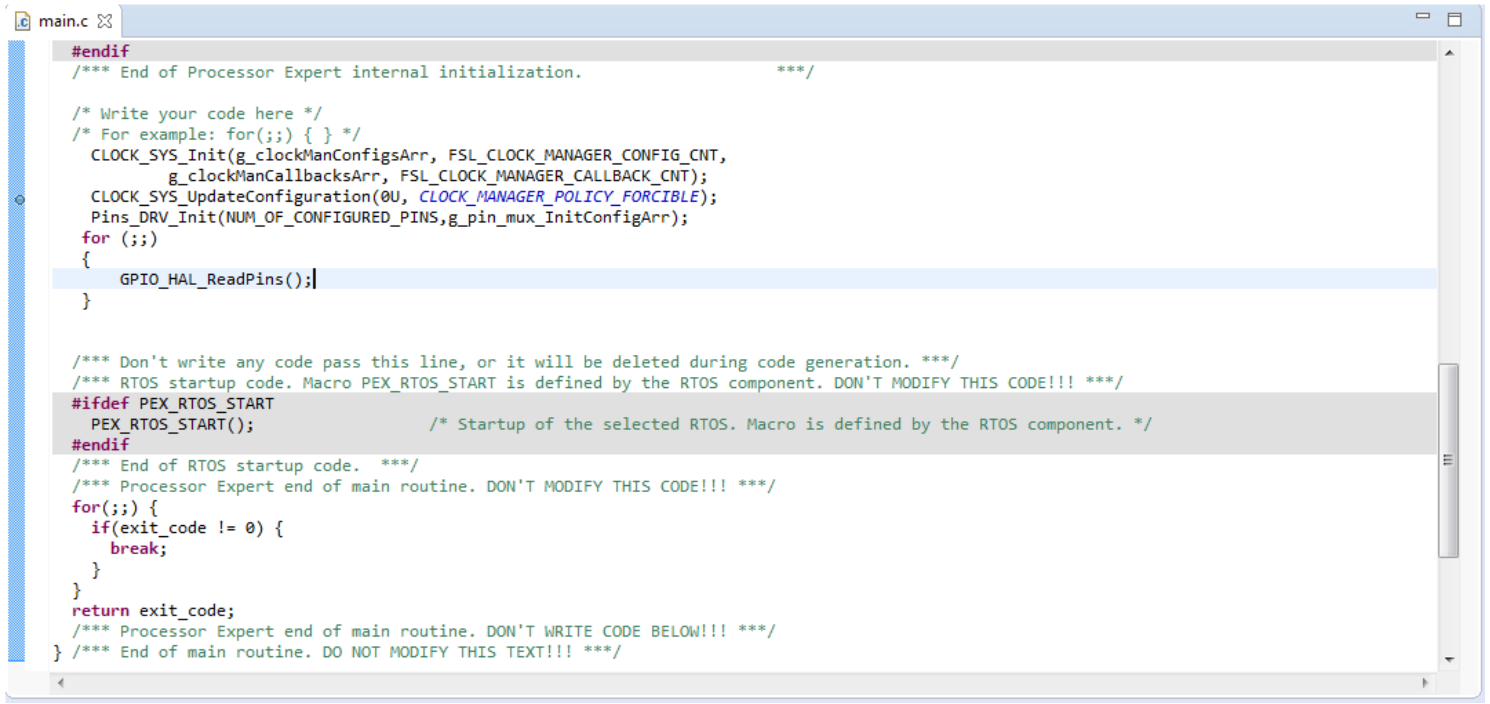
```
Pins_DRV_Init(NUM_OF_CONFIGURED_PINS,g_pin_mux_InitConfigArr);
```

S32K144 GPIOs Lab: Read SW2(PTC12) input

- Expand the **GPIO HAL** component in the **Components** Window
- Drag and drop the **GPIO_HAL_ReadPins** function in to main, an place it inside an infinite loop.


Components - S32K144_SDK_Lab_GPIOs

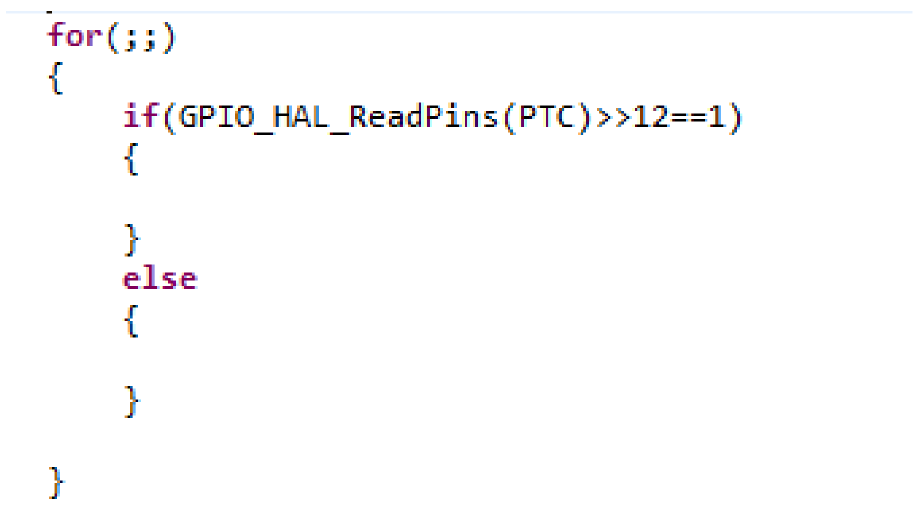
- Generator_Configurations
- OSs
- Processors
  - Cpu:S32K144_100
- Components
  - pin_mux:PinSettings
    - fsl_port_hal1:fsl_port_hal
    - Pins_DRV_Init
  - clockMan1:fsl_clock_manager
  - gpio1:fsl_gpio_hal
    - GPIO_HAL_WritePins
    - GPIO_HAL_GetPinsOutput
    - GPIO_HAL_SetPins
    - GPIO_HAL_ClearPins
    - GPIO_HAL_TogglePins
    - GPIO_HAL_ReadPins
    - GPIO_HAL_GetPinsDirection
    - GPIO_HAL_SetPinsDirection


main.c

```
#endif
/*** End of Processor Expert internal initialization.                    ***/

/* Write your code here */
/* For example: for(;;) { } */
  CLOCK_SYS_Init(g_clockManConfigsArr, FSL_CLOCK_MANAGER_CONFIG_CNT,
          g_clockManCallbacksArr, FSL_CLOCK_MANAGER_CALLBACK_CNT);
  CLOCK_SYS_UpdateConfiguration(0U, CLOCK_MANAGER_POLICY_FORCIBLE);
  Pins_DRV_Init(NUM_OF_CONFIGURED_PINS,g_pin_mux_InitConfigArr);
 for (;;)
 {
     GPIO_HAL_ReadPins();
 }


/*** Don't write any code pass this line, or it will be deleted during code generation. ***/
/*** RTOS startup code. Macro PEX_RTOS_START is defined by the RTOS component. DON'T MODIFY THIS CODE!!! ***/
#ifdef PEX_RTOS_START
  PEX_RTOS_START();                  /* Startup of the selected RTOS. Macro is defined by the RTOS component. */
#endif
/*** End of RTOS startup code.  ***/
/*** Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! ***/
for(;;) {
  if(exit_code != 0) {
    break;
  }
}
return exit_code;
/*** Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! ***/
} /*** End of main routine. DO NOT MODIFY THIS TEXT!!! ***/
```

S32K144 GPIOs Lab: Read SW2(PTC12) input

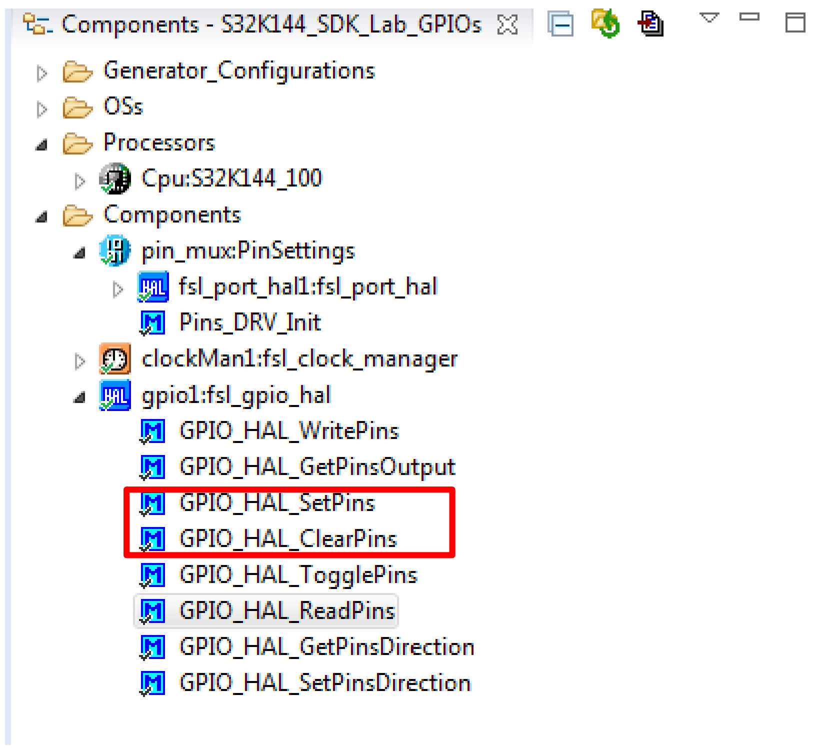
- **GPIO_HAL_ReadPins** function receives one parameter:
  - PORTx to read
- **GPIO_HAL_ReadPins** returns the value of the PIDR of PORTx
- Use an if statement as follows to read SW2(PTC12)



```
for(;;)
{
    if(GPIO_HAL_ReadPins(PTC)>>12==1)
    {

    }
    else
    {

    }

}
```

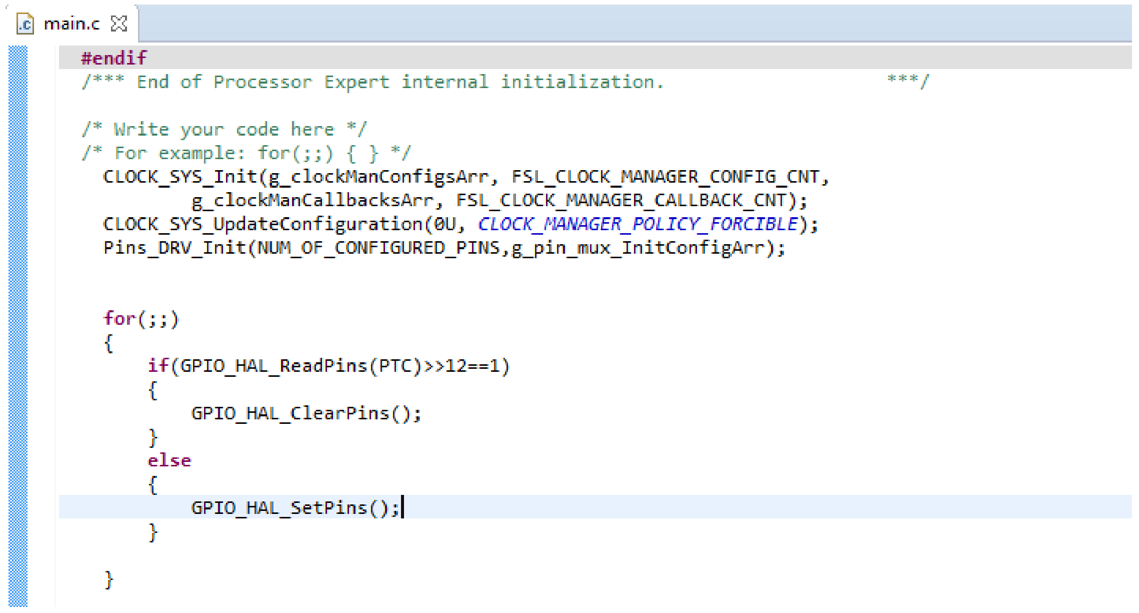
S32K144 GPIOs Lab: Turn on Green LED (PTD16)

- Expand the **GPIO HAL** component in the **Components** Window
- Drag and drop the **GPIO_HAL_ClearPins** function in to main, an place it inside the if statement
- Drag and drop the **GPIO_HAL_SetPins** function in to main, an place it inside the else statement



Components - S32K144_SDK_Lab_GPIOs

- Generator_Configurations
- OSs
- Processors
  - Cpu:S32K144_100
- Components
  - pin_mux:PinSettings
    - fsl_port_hal1:fsl_port_hal
    - Pins_DRV_Init
  - clockMan1:fsl_clock_manager
  - gpio1:fsl_gpio_hal
    - GPIO_HAL_WritePins
    - GPIO_HAL_GetPinsOutput
    - GPIO_HAL_SetPins
    - GPIO_HAL_ClearPins
    - GPIO_HAL_TogglePins
    - GPIO_HAL_ReadPins
    - GPIO_HAL_GetPinsDirection
    - GPIO_HAL_SetPinsDirection

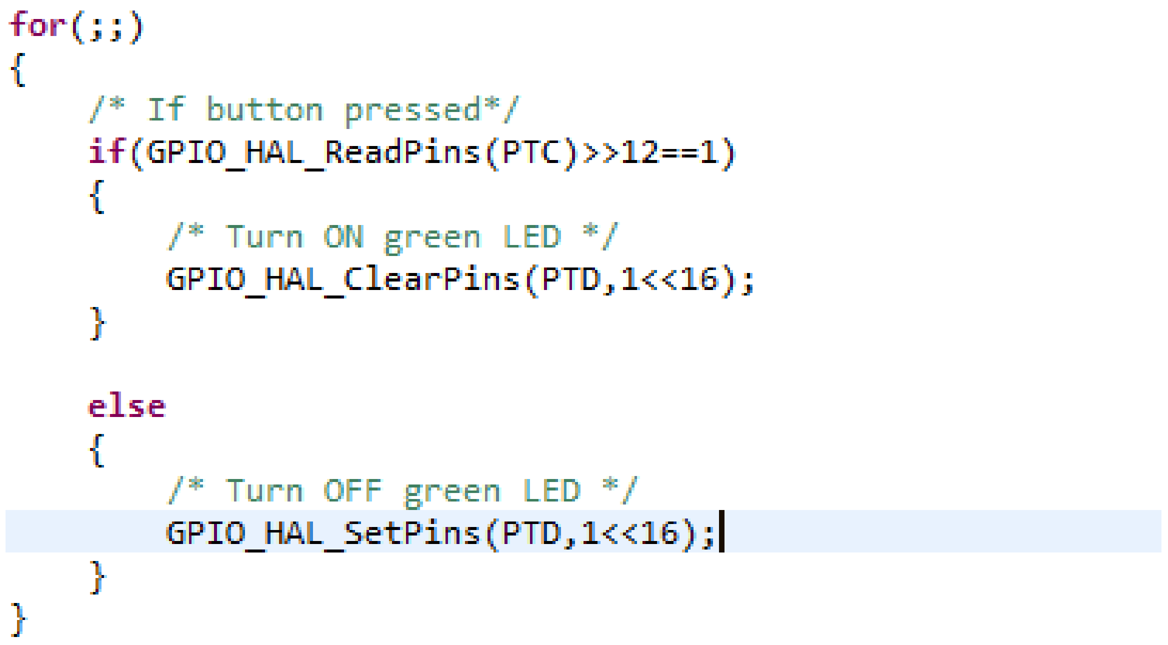


main.c

```
#endif
/*** End of Processor Expert internal initialization.                    ***/

/* Write your code here */
/* For example: for(;;) { } */
  CLOCK_SYS_Init(g_clockManConfigsArr, FSL_CLOCK_MANAGER_CONFIG_CNT,
          g_clockManCallbacksArr, FSL_CLOCK_MANAGER_CALLBACK_CNT);
  CLOCK_SYS_UpdateConfiguration(0U, CLOCK_MANAGER_POLICY_FORCIBLE);
  Pins_DRV_Init(NUM_OF_CONFIGURED_PINS,g_pin_mux_InitConfigArr);


  for(;;)
  {
      if(GPIO_HAL_ReadPins(PTC)>>12==1)
      {
          GPIO_HAL_ClearPins();
      }
      else
      {
          GPIO_HAL_SetPins();
      }

  }
```

S32K144 GPIOs Lab: Turn on Green LED (PTD16)

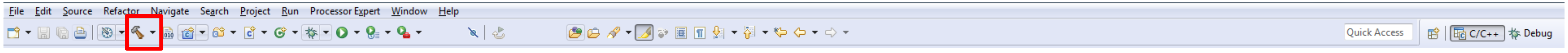
- Both functions receive two parameters:
  - PORTx to modify
  - Bit mask
- In order to turn on/off the green LED, include the following parameters into the Clear and Set functions



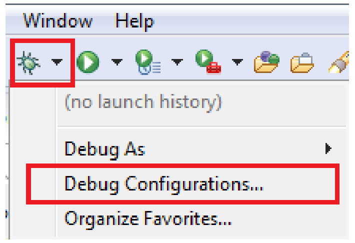
```
for(;;)
{
    /* If button pressed*/
    if(GPIO_HAL_ReadPins(PTC)>>12==1)
    {
        /* Turn ON green LED */
        GPIO_HAL_ClearPins(PTD,1<<16);
    }

    else
    {
        /* Turn OFF green LED */
        GPIO_HAL_SetPins(PTD,1<<16);
    }
}
```

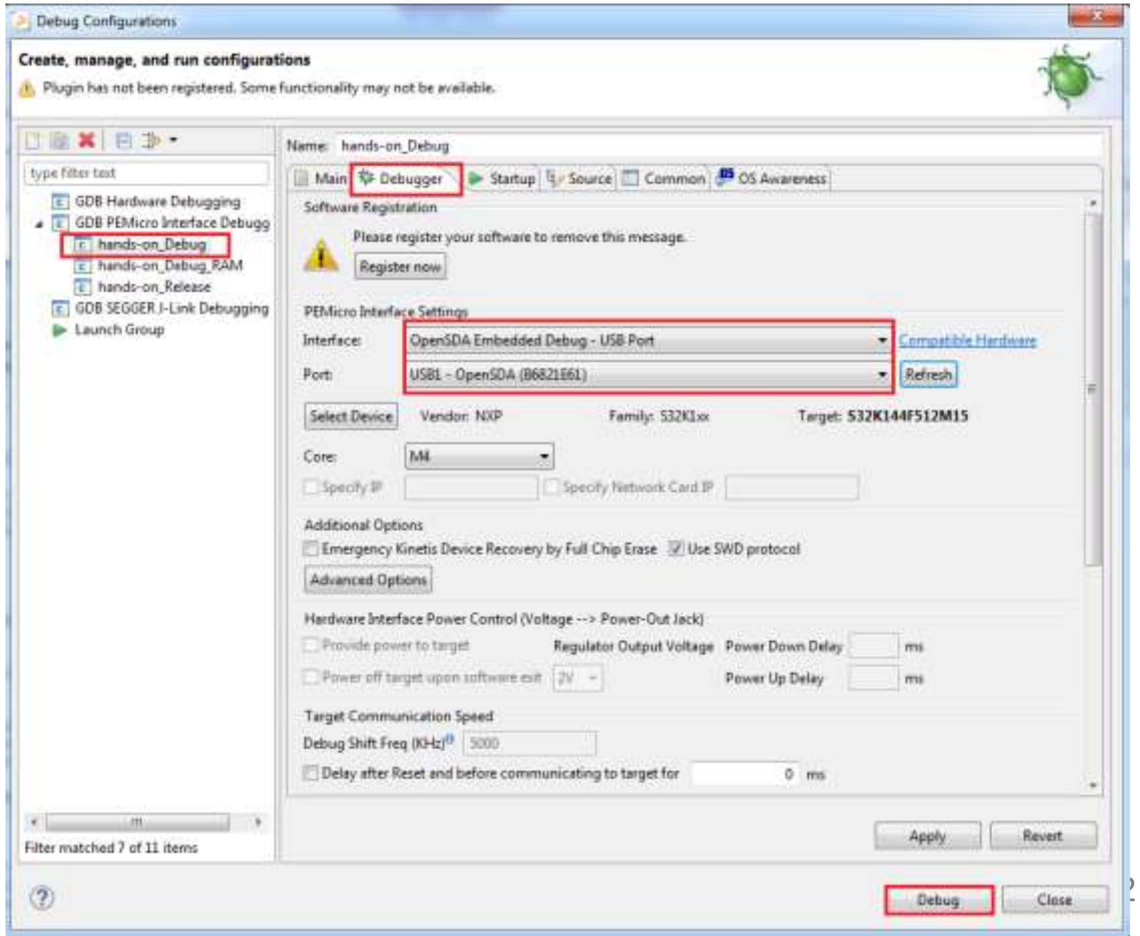
S32K144 GPIOs Lab: Build and debug the lab

- Click on the build icon to make sure that there a no compiler errors.


File Edit Source Refactor Navigate Search Project Run Processor Expert Window Help
Quick Access
C/C++
Debug

- Configure the debug configuration start a new debug session




Debug Configurations

Create, manage, and run configurations

Plugin has not been registered. Some functionality may not be available.

type filter text
- GDB Hardware Debugging
- GDB PEMicro Interface Debugg
  - hands-on_Debug
  - hands-on_Debug_RAM
  - hands-on_Release
- GDB SEGGER J-Link Debugging
- Launch Group

Filter matched 7 of 11 items

Name: hands-on_Debug

Main | Debugger | Startup | Source | Common | OS Awareness

Software Registration

Please register your software to remove this message.

Register now

PEMicro Interface Settings

Interface: OpenSDA Embedded Debug - USB Port — Compatible Hardware

Port: USB1 - OpenSDA (B6821E61) — Refresh

Select Device — Vendor: NXP — Family: S32K1xx — Target: **S32K144F512M15**

Core: M4

☐ Specify IP ☐ Specify Network Card IP

Additional Options

☐ Emergency Kinetis Device Recovery by Full Chip Erase ☑ Use SWD protocol

Advanced Options

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☐ Provide power to target — Regulator Output Voltage — Power Down Delay ms

☐ Power off target upon software exit — 2V — Power Up Delay ms

Target Communication Speed

Debug Shift Freq (KHz) 5000

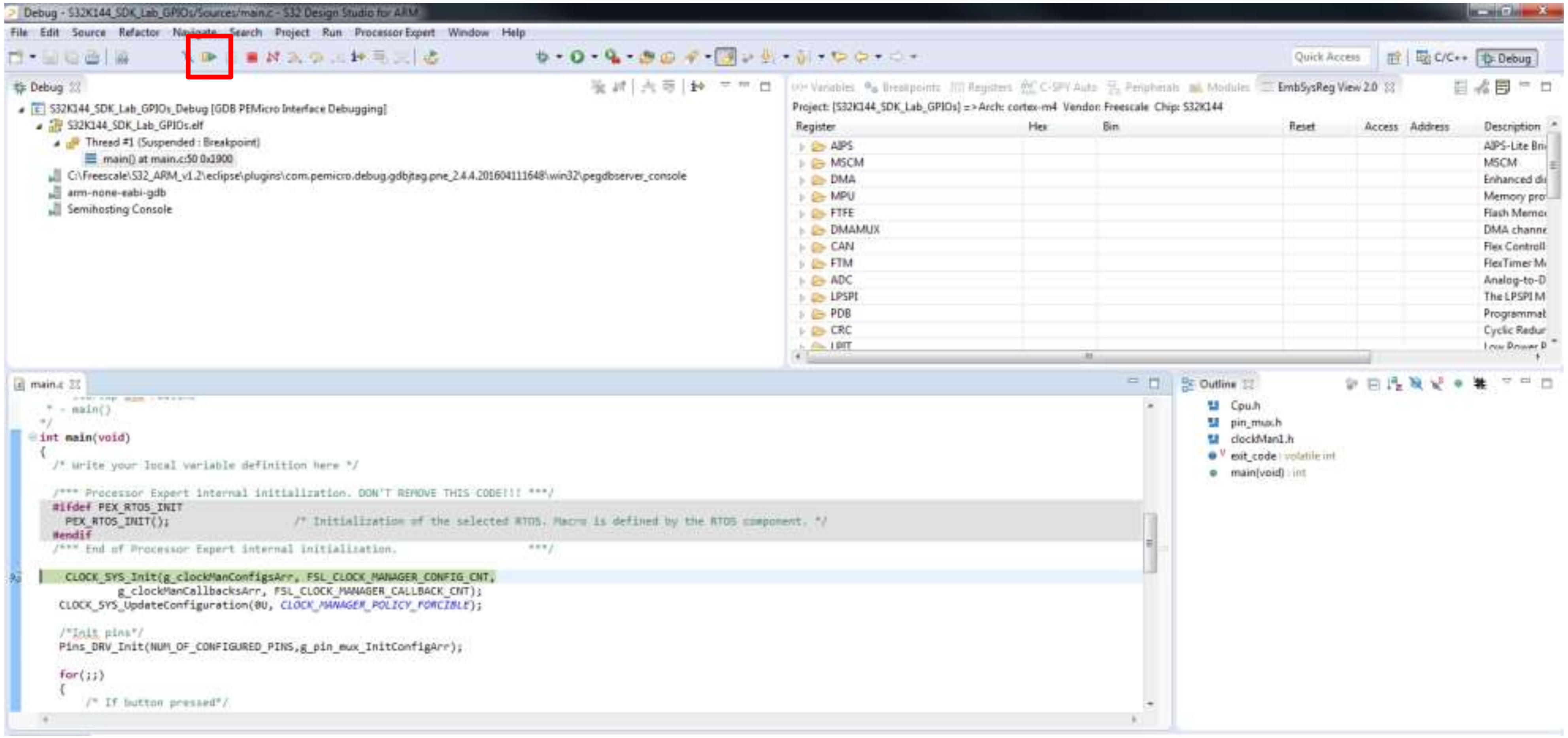
☐ Delay after Reset and before communicating to target for 0 ms

Apply Revert

Debug Close

S32K144 GPIOs Lab: Build and debug the lab

- In the debug perspective click the run icon to start the project.
- Press SW2 and the Green LED should turn on.


Debug - S32K144_SDK_Lab_GPIOs/Sources/main.c - S32 Design Studio for ARM
File Edit Source Refactor Navigate Search Project Run Processor Expert Window Help
Quick Access
C/C++
Debug
Variables Breakpoints Registers C-SPY Auto Peripherals Modules EmbSysReg View 2.0
Outline

S32K144_SDK_Lab_GPIOs_Debug [GDB PEMicro Interface Debugging]
S32K144_SDK_Lab_GPIOs.elf
Thread #1 (Suspended : Breakpoint)
main() at main.c:50 0x1900
C:\Freescale\S32_ARM_v1.2\eclipse\plugins\com.pemicro.debug.gdbjtag.pne_2.4.4.201604111648\win32\pegdbserver_console
arm-none-eabi-gdb
Semihosting Console

Project: [S32K144_SDK_Lab_GPIOs] => Arch: cortex-m4 Vendor: Freescale Chip: S32K144

Register	Hex	Bin	Reset	Access	Address	Description
AIPS						AIPS-Lite Bri
MSCM						MSCM
DMA						Enhanced di
MPU						Memory pro
FTFE						Flash Memo
DMAMUX						DMA channe
CAN						Flex Controll
FTM						FlexTimer M
ADC						Analog-to-D
LPSPI						The LPSPI M
PDB						Programmat
CRC						Cyclic Redur
LPIT						Low Power P

main.c

```c
 * - main()
 */
int main(void)
{
  /* Write your local variable definition here */

  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  #ifdef PEX_RTOS_INIT
    PEX_RTOS_INIT();                 /* Initialization of the selected RTOS. Macro is defined by the RTOS component. */
  #endif
  /*** End of Processor Expert internal initialization.                    ***/

    CLOCK_SYS_Init(g_clockManConfigsArr, FSL_CLOCK_MANAGER_CONFIG_CNT,
                   g_clockManCallbacksArr, FSL_CLOCK_MANAGER_CALLBACK_CNT);
    CLOCK_SYS_UpdateConfiguration(0U, CLOCK_MANAGER_POLICY_FORCIBLE);

    /*Init pins*/
    Pins_DRV_Init(NUM_OF_CONFIGURED_PINS,g_pin_mux_InitConfigArr);

    for(;;)
    {
        /* If button pressed*/
```

Outline:
Cpu.h
pin_mux.h
clockMan1.h
exit_code : volatile int
main(void) : int

S32K144 GPIOs Lab: Challenge

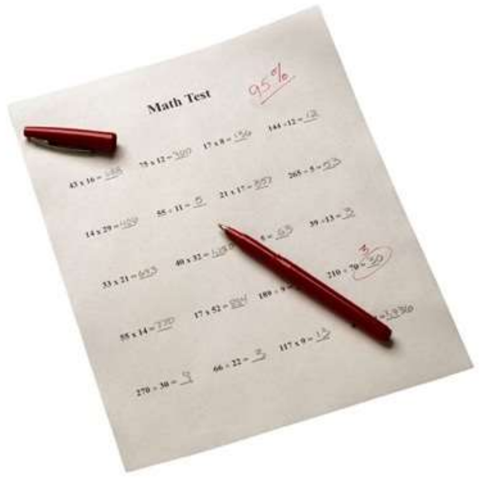
- Enable SW3(PTC13)
- Turn on RGB LED yellow when pressing SW3
- Turn on RGB LED white when pressing SW2

S32K144 GPIOs Lab: Quiz

- How many GPIO pins are in S32K144 100LQFP and 64LQFP?

- How many high current pins are in S32K144?

- How much current can a high current pin supply?



S32K144 CLOCKS

S32K144 Clocks Lab: Objective

- **Task:**
  - Configure and Use System PLL as a Clock Source

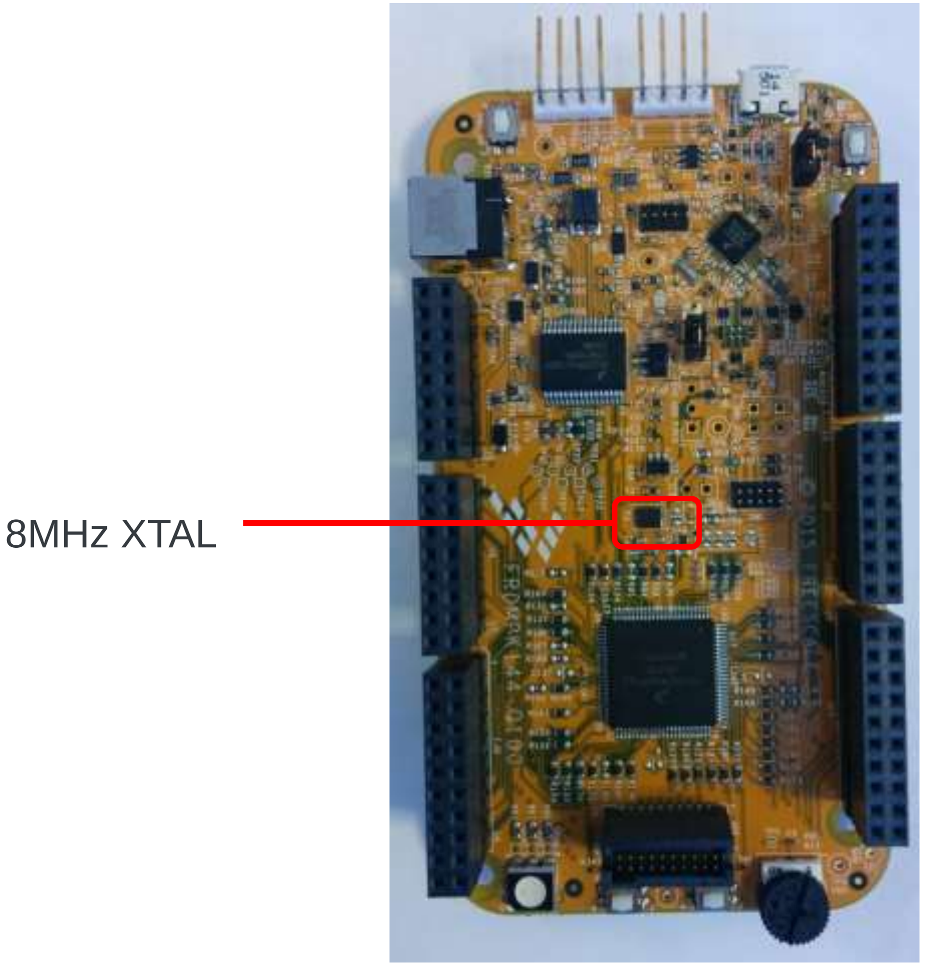
- **Learn:**
  - About the clock tree in S32K144
  - How to create a new SDK project with S32DS.
  - How to setup S32K144 in the following clock configuration
    - Clock Source: PLL
    - Core Clock: 50MHz
    - Bus Clock: 25MHz
    - Flash Clock: 25 MHz

- **Target Modules:**
  - SCG – System Clock Generator

S32K144 Clocks Lab: Resources to be used

- This lab will use the following components of the EVB:
  - 8 MHz XTAL

XTAL	S32K144 PIN
1	PTB7
2	PTB6



S32K144 Clocks Lab: Theory

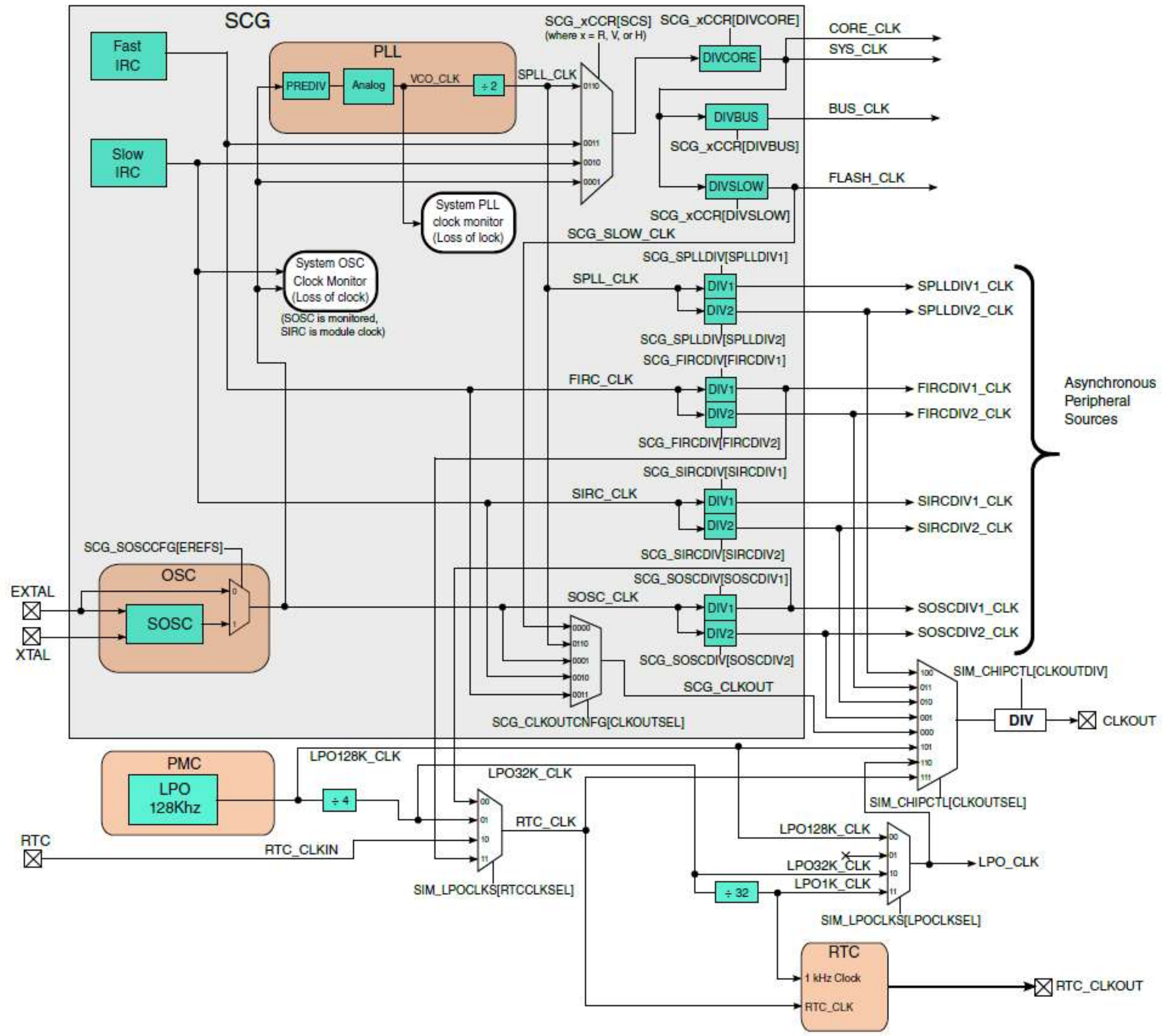
- Clock source for the core:
  - FIRC (48MHz)
  - SIRC (8 MHz)
  - PLL (up to 112 MHz)
  - SOSC (8 – 40 MHz)

- PLL Flexible multiplier (16 – 47)

- PLL source is external oscillator

- By default, device clock is FIRC
  - Core Frequency is 48 MHz

- Multiple clock source for peripherals

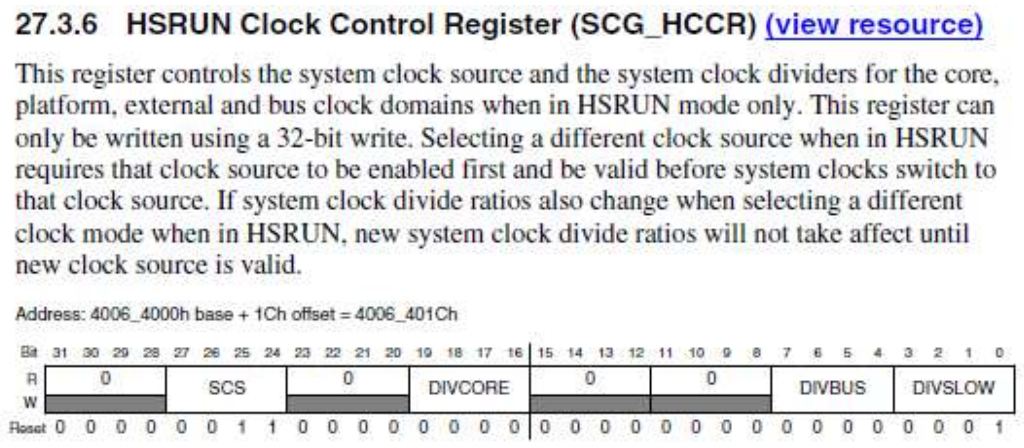


S32K144 Clocks Lab: Theory

- **Core clock**: Clocks the ARM core.
- **System clock**: Clocks the Crossbar, NVIC, Flash controller, FTM and PDB, same as Core clock.
- **Bus clock**: Clocks the Peripherals.
- **Flash clock**: Clocks the flash module.
- **SPLL DIVx clock**: Optional divided PLL source for peripherals.
- **SIRC DIVx clock**: Optional divided SIRC source for peripherals.
- **FIRC DIVx clock**: Optional divided FIRC source for peripherals.
- **OSC DIVx clock**: Optional divided System Oscillator clock for peripherals.
- **LPO clock:** Low power oscillator clock inside PMC.
- **RTC clock out:** Clock output from RTC.
- **Clock out:** Optional output clock source for external devices.

S32K144 Clocks Lab: Theory

- For each power mode (HSRUN,RUN,VLPR) there are three dividers that are controlled independently:
  - DIVCORE: Core Clock divider
  - DIVBUS: Bus Clock divider
  - DIVSLOW: Flash Clock divider
- Each of these dividers bit fields inside the Clock Control Register of each mode
- The source clock for each mode is selected with the SCS bits:
  - FIRC
  - SIRC
  - PLL
  - OSC



27.3.6 HSRUN Clock Control Register (SCG_HCCR) (view resource)

This register controls the system clock source and the system clock dividers for the core, platform, external and bus clock domains when in HSRUN mode only. This register can only be written using a 32-bit write. Selecting a different clock source when in HSRUN requires that clock source to be enabled first and be valid before system clocks switch to that clock source. If system clock divide ratios also change when selecting a different clock mode when in HSRUN, new system clock divide ratios will not take affect until new clock source is valid.

Address: 4006_4000h base + 1Ch offset = 4006_401Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				SCS				0				DIVCORE				0				0				DIVBUS				DIVSLOW			
W																																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

S32K144 Clocks Lab: Theory

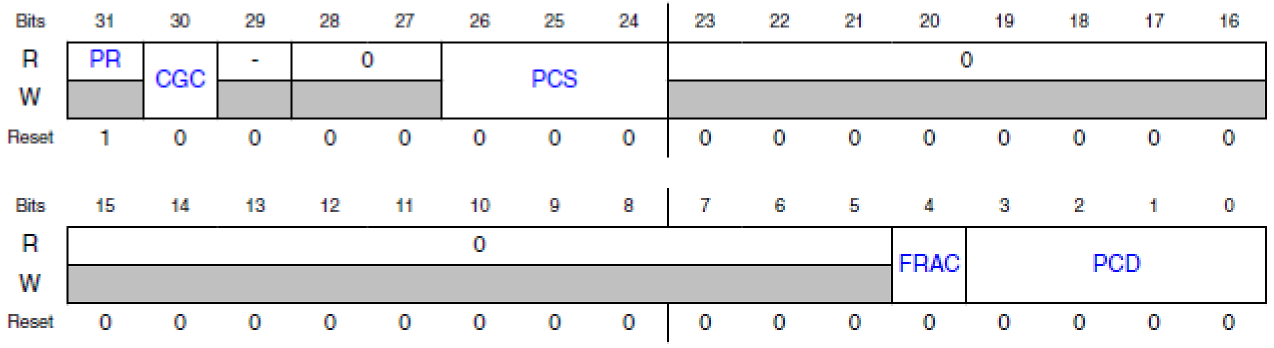
- HSRUN
  - Core clock and System clock: 112MHz or less
  - Bus clock: 56 MHz or less.
  - Flash clock: 28 MHz or less.

- RUN
  - Core clock and System clock: 80MHz or less
  - Bus clock: 40 MHz or less.
  - Flash clock: 26 MHz or less.

- VLPR:
  - Core clock and system clock: 4MHz or less.
  - Flash clock: 1MHz or less

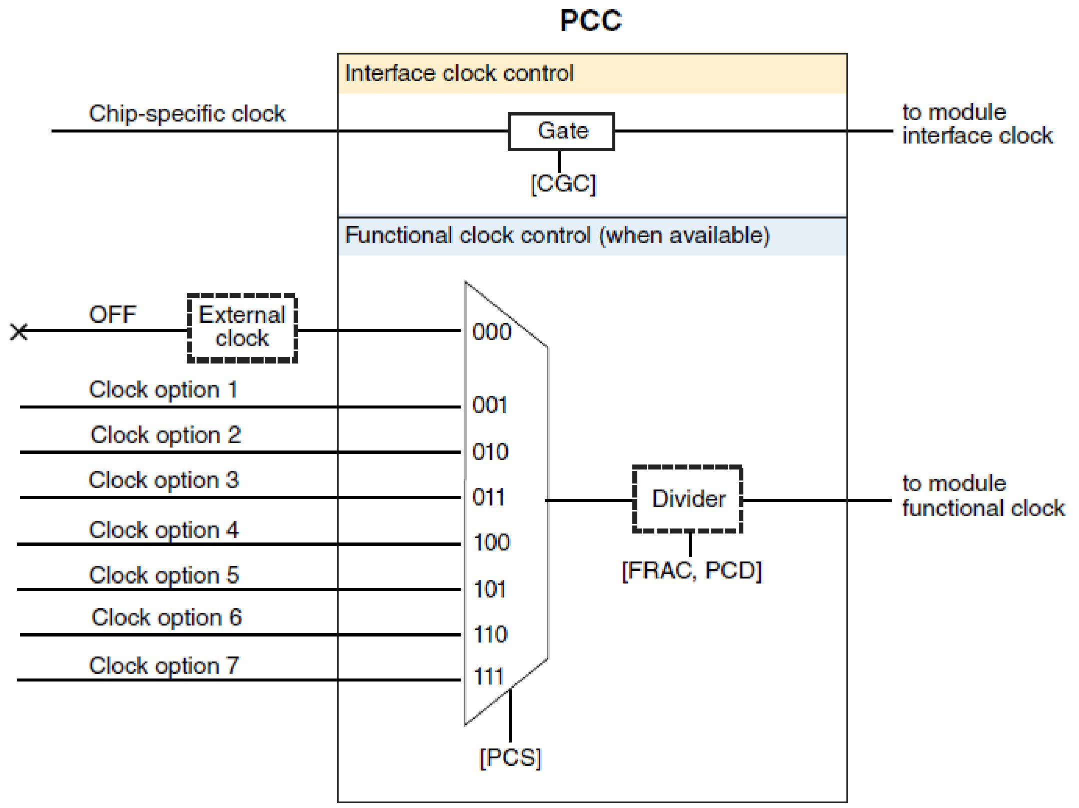
S32K144 Clocks Lab: Revisiting PCC Theory 1

- Modules can be individually turn on or off using the PCC module.
- Clock source for each peripheral can be selected from multiple sources.
- Before using a peripheral, turn on its clock.

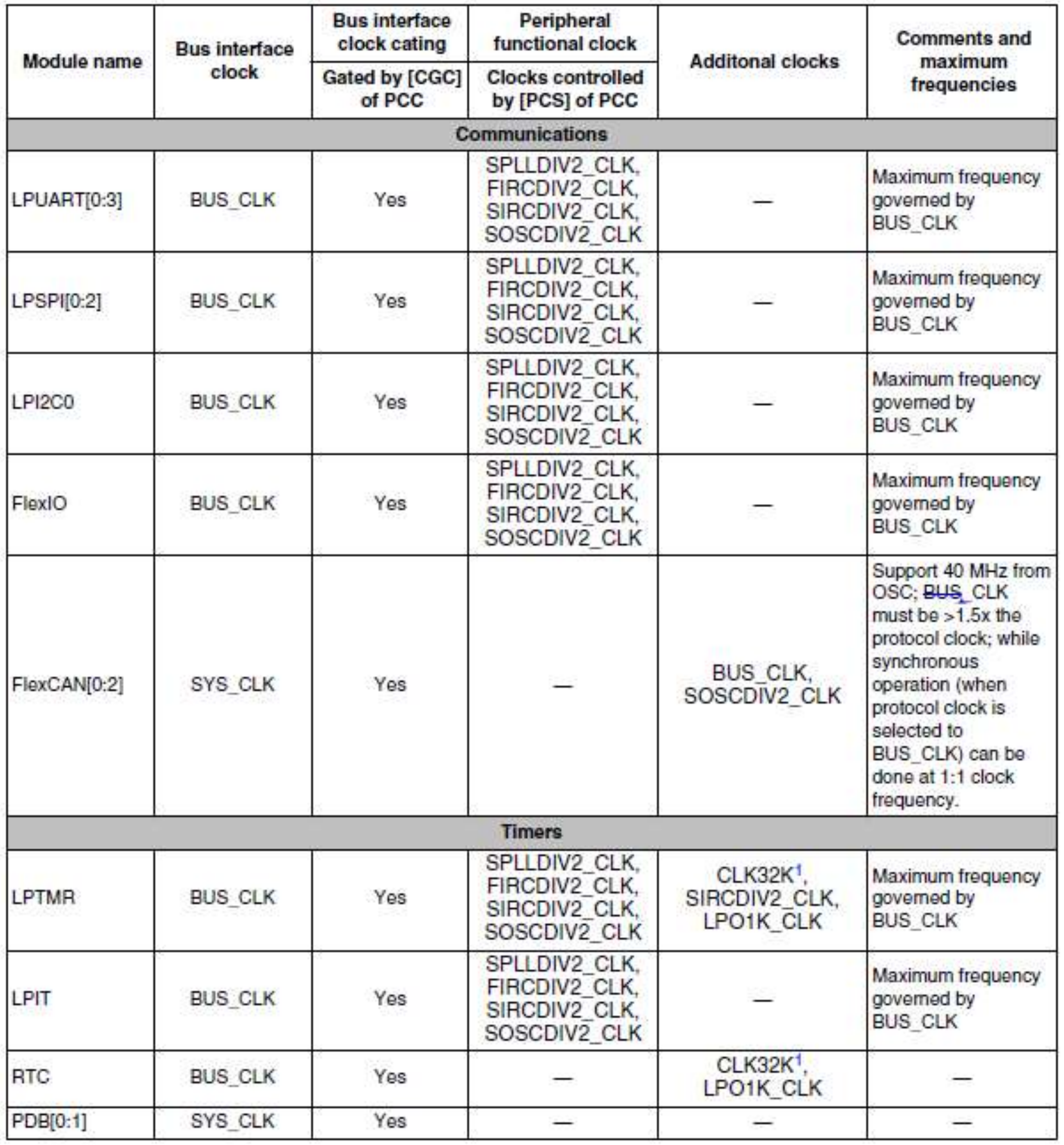


Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PR	CGC	-	0		PCS			0							
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0											FRAC	PCD			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



S32K144 Clocks Lab: Revisiting PCC Theory 2



Module name	Bus interface clock	Bus interface clock cating	Peripheral functional clock	Additonal clocks	Comments and maximum frequencies
		Gated by [CGC] of PCC	Clocks controlled by [PCS] of PCC		
Communications					
LPUART[0:3]	BUS_CLK	Yes	SPLLDIV2_CLK, FIRCDIV2_CLK, SIRCDIV2_CLK, SOSCDIV2_CLK	—	Maximum frequency governed by BUS_CLK
LPSPI[0:2]	BUS_CLK	Yes	SPLLDIV2_CLK, FIRCDIV2_CLK, SIRCDIV2_CLK, SOSCDIV2_CLK	—	Maximum frequency governed by BUS_CLK
LPI2C0	BUS_CLK	Yes	SPLLDIV2_CLK, FIRCDIV2_CLK, SIRCDIV2_CLK, SOSCDIV2_CLK	—	Maximum frequency governed by BUS_CLK
FlexIO	BUS_CLK	Yes	SPLLDIV2_CLK, FIRCDIV2_CLK, SIRCDIV2_CLK, SOSCDIV2_CLK	—	Maximum frequency governed by BUS_CLK
FlexCAN[0:2]	SYS_CLK	Yes	—	BUS_CLK, SOSCDIV2_CLK	Support 40 MHz from OSC; BUS_CLK must be >1.5x the protocol clock; while synchronous operation (when protocol clock is selected to BUS_CLK) can be done at 1:1 clock frequency.
Timers					
LPTMR	BUS_CLK	Yes	SPLLDIV2_CLK, FIRCDIV2_CLK, SIRCDIV2_CLK, SOSCDIV2_CLK	CLK32K[1], SIRCDIV2_CLK, LPO1K_CLK	Maximum frequency governed by BUS_CLK
LPIT	BUS_CLK	Yes	SPLLDIV2_CLK, FIRCDIV2_CLK, SIRCDIV2_CLK, SOSCDIV2_CLK	—	Maximum frequency governed by BUS_CLK
RTC	BUS_CLK	Yes	—	CLK32K[1], LPO1K_CLK	—
PDB[0:1]	SYS_CLK	Yes	—	—	—

S32K144 Clocks Lab: Initial Steps

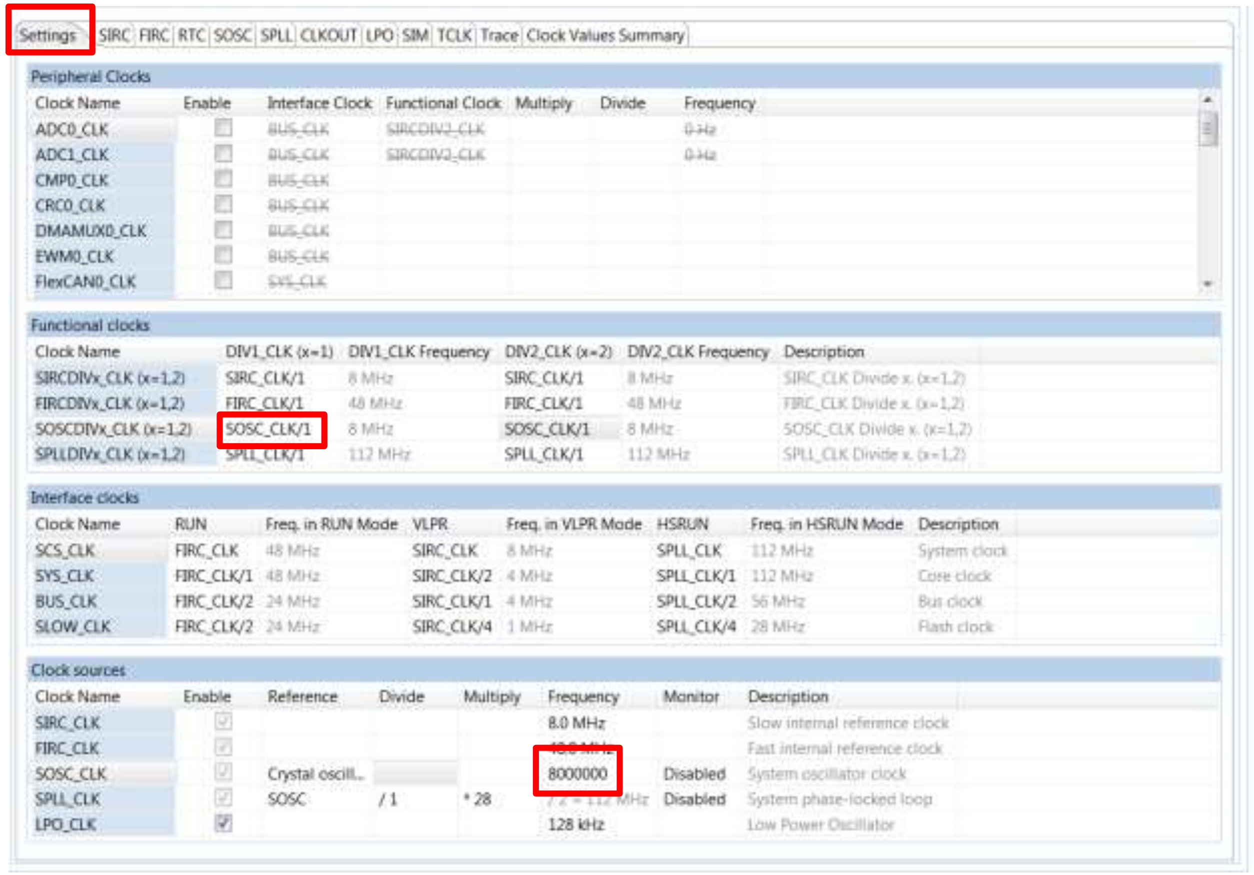
- Create a new S32DS Project
- Add **clock_manager** to your project

OR

- Just use the Previous Project

S32K144 Clocks Lab: Select SOSC configuration

- In the **Component Inspector**, select the **Settings** tab
- In the **Clock sources -> SOSC_CLK->Frequency** field write 8000000(corresponding to 8MHz)
- In the **Functional Clock-> SOSCDIVx_CLK->DIV1_CLK** select SOSC_CLK/1


Settings | SIRC | FIRC | RTC | SOSC | SPLL | CLKOUT | LPO | SIM | TCLK | Trace | Clock Values Summary

Peripheral Clocks

Clock Name	Enable	Interface Clock	Functional Clock	Multiply	Divide	Frequency
ADC0_CLK	☐	~~BUS_CLK~~	~~SIRCDIV2_CLK~~			~~0 Hz~~
ADC1_CLK	☐	~~BUS_CLK~~	~~SIRCDIV2_CLK~~			~~0 Hz~~
CMP0_CLK	☐	~~BUS_CLK~~				
CRC0_CLK	☐	~~BUS_CLK~~				
DMAMUX0_CLK	☐	~~BUS_CLK~~				
EWM0_CLK	☐	~~BUS_CLK~~				
FlexCAN0_CLK	☐	~~SYS_CLK~~				

Functional clocks

Clock Name	DIV1_CLK (x=1)	DIV1_CLK Frequency	DIV2_CLK (x=2)	DIV2_CLK Frequency	Description
SIRCDIVx_CLK (x=1,2)	SIRC_CLK/1	8 MHz	SIRC_CLK/1	8 MHz	SIRC_CLK Divide x. (x=1,2)
FIRCDIVx_CLK (x=1,2)	FIRC_CLK/1	48 MHz	FIRC_CLK/1	48 MHz	FIRC_CLK Divide x. (x=1,2)
SOSCDIVx_CLK (x=1,2)	SOSC_CLK/1	8 MHz	SOSC_CLK/1	8 MHz	SOSC_CLK Divide x. (x=1,2)
SPLLDIVx_CLK (x=1,2)	SPLL_CLK/1	112 MHz	SPLL_CLK/1	112 MHz	SPLL_CLK Divide x. (x=1,2)

Interface clocks

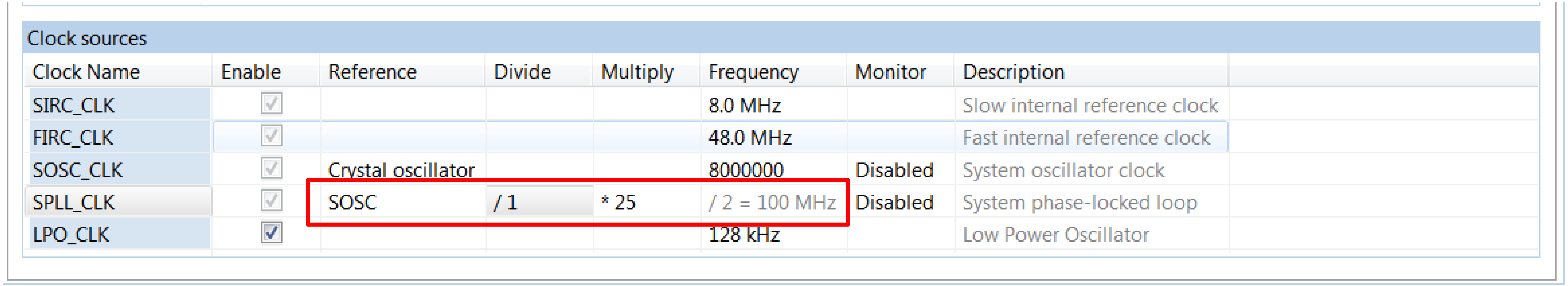
Clock Name	RUN	Freq. in RUN Mode	VLPR	Freq. in VLPR Mode	HSRUN	Freq. in HSRUN Mode	Description
SCS_CLK	FIRC_CLK	48 MHz	SIRC_CLK	8 MHz	SPLL_CLK	112 MHz	System clock
SYS_CLK	FIRC_CLK/1	48 MHz	SIRC_CLK/2	4 MHz	SPLL_CLK/1	112 MHz	Core clock
BUS_CLK	FIRC_CLK/2	24 MHz	SIRC_CLK/1	4 MHz	SPLL_CLK/2	56 MHz	Bus clock
SLOW_CLK	FIRC_CLK/2	24 MHz	SIRC_CLK/4	1 MHz	SPLL_CLK/4	28 MHz	Flash clock

Clock sources

Clock Name	Enable	Reference	Divide	Multiply	Frequency	Monitor	Description
SIRC_CLK	✓				8.0 MHz		Slow internal reference clock
FIRC_CLK	✓				48.0 MHz		Fast internal reference clock
SOSC_CLK	✓	Crystal oscill...			8000000	Disabled	System oscillator clock
SPLL_CLK	✓	SOSC	/ 1	* 28	/ 2 = 112 MHz	Disabled	System phase-locked loop
LPO_CLK	✓				128 kHz		Low Power Oscillator

S32K144 Clocks Lab: Select PLL configuration

- Go to **Settings** tab
- In the **Clock sources -> SPLL_CLK->Reference** select SOSC
- In the **Clock sources -> SPLL_CLK->Divide** select /1
- In the **Clock sources -> SPLL_CLK->Multiplicity** field select *25

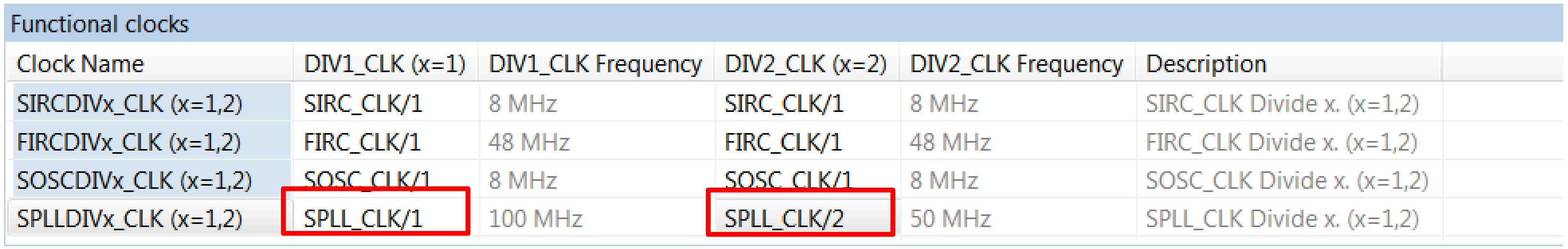


Clock sources

Clock Name	Enable	Reference	Divide	Multiply	Frequency	Monitor	Description
SIRC_CLK	✓				8.0 MHz		Slow internal reference clock
FIRC_CLK	✓				48.0 MHz		Fast internal reference clock
SOSC_CLK	✓	Crystal oscillator			8000000	Disabled	System oscillator clock
SPLL_CLK	✓	SOSC	/ 1	* 25	/ 2 = 100 MHz	Disabled	System phase-locked loop
LPO_CLK	✓				128 kHz		Low Power Oscillator

S32K144 Clocks Lab: Select PLL configuration

- Go to **Settings** tab
- In the **Functional clocks->SPLLDIVx_CLK->DIV1_CLK** select SPLL_CLK/1.
- In the **Functional clocks->SPLLDIVx_CLK->DIV2_CLK** select SPLL_CLK/2.

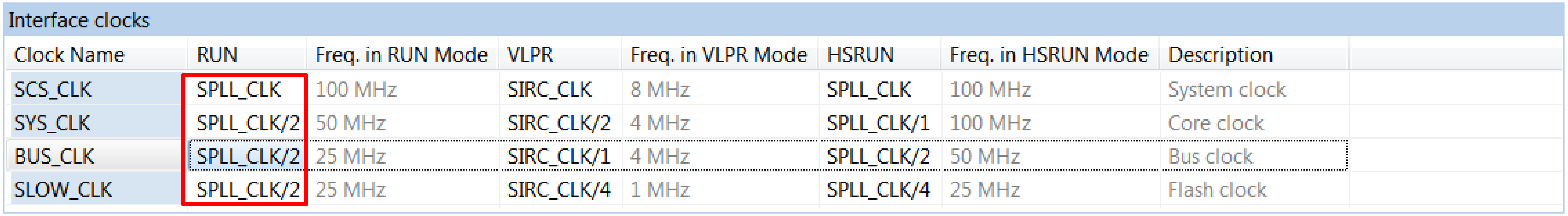


Functional clocks

Clock Name	DIV1_CLK (x=1)	DIV1_CLK Frequency	DIV2_CLK (x=2)	DIV2_CLK Frequency	Description
SIRCDIVx_CLK (x=1,2)	SIRC_CLK/1	8 MHz	SIRC_CLK/1	8 MHz	SIRC_CLK Divide x. (x=1,2)
FIRCDIVx_CLK (x=1,2)	FIRC_CLK/1	48 MHz	FIRC_CLK/1	48 MHz	FIRC_CLK Divide x. (x=1,2)
SOSCDIVx_CLK (x=1,2)	SOSC_CLK/1	8 MHz	SOSC_CLK/1	8 MHz	SOSC_CLK Divide x. (x=1,2)
SPLLDIVx_CLK (x=1,2)	SPLL_CLK/1	100 MHz	SPLL_CLK/2	50 MHz	SPLL_CLK Divide x. (x=1,2)

S32K144 Clocks Lab: Select PLL configuration

- Go to **Settings** tab
- In the **Interface clocks section** select the following RUN configuration**:**
  - **System Clock Source:** SPLL
  - **Core Clock Divide Ratio:** SPLL_CLK/2
  - **Platform Clock Divide Ratio:** SPLL_CLK/2
  - **Bus Clock Divide Ratio:** SPLL_CLK/2
  - **Slow Clock(Flash Clock) Divide Ratio:** SPLL_CLK/2

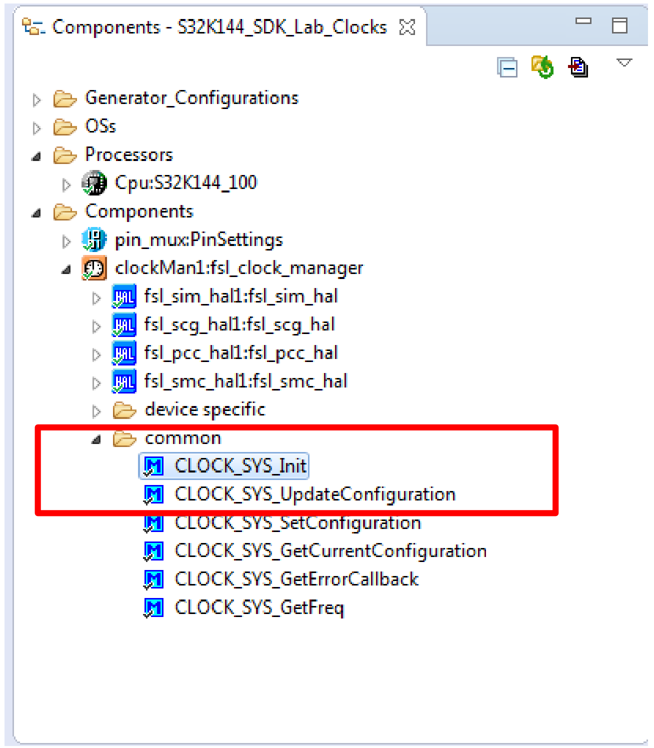


Interface clocks

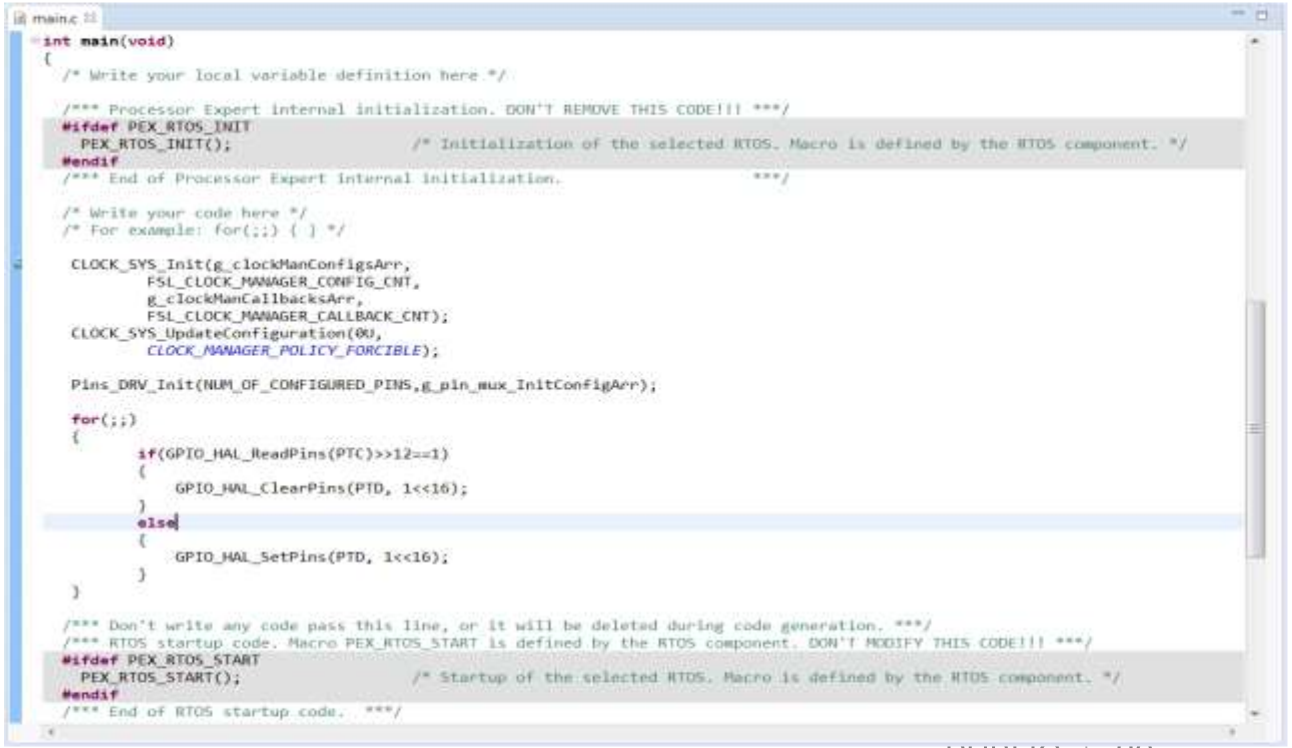
Clock Name	RUN	Freq. in RUN Mode	VLPR	Freq. in VLPR Mode	HSRUN	Freq. in HSRUN Mode	Description
SCS_CLK	SPLL_CLK	100 MHz	SIRC_CLK	8 MHz	SPLL_CLK	100 MHz	System clock
SYS_CLK	SPLL_CLK/2	50 MHz	SIRC_CLK/2	4 MHz	SPLL_CLK/1	100 MHz	Core clock
BUS_CLK	SPLL_CLK/2	25 MHz	SIRC_CLK/1	4 MHz	SPLL_CLK/2	50 MHz	Bus clock
SLOW_CLK	SPLL_CLK/2	25 MHz	SIRC_CLK/4	1 MHz	SPLL_CLK/4	25 MHz	Flash clock

S32K144 Clocks Lab: Add Init and Update Configuration functions

- Expand the **clock_manager** component in the **Components** Window
- Drag and drop the **CLOCK_SYS_Init** function into main.
- Drag and drop the **CLOCK_SYS_UpdateConfiguration** function into main.
- Include an infinite loop after these functions.


Components - S32K144_SDK_Lab_Clocks

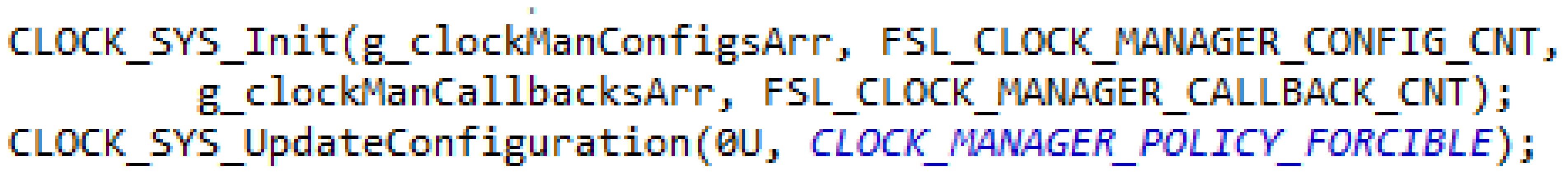
```
▷ Generator_Configurations
▷ OSs
◢ Processors
    ▷ Cpu:S32K144_100
◢ Components
    ▷ pin_mux:PinSettings
    ◢ clockMan1:fsl_clock_manager
        ▷ fsl_sim_hal1:fsl_sim_hal
        ▷ fsl_scg_hal1:fsl_scg_hal
        ▷ fsl_pcc_hal1:fsl_pcc_hal
        ▷ fsl_smc_hal1:fsl_smc_hal
        ▷ device specific
        ◢ common
            CLOCK_SYS_Init
            CLOCK_SYS_UpdateConfiguration
            CLOCK_SYS_SetConfiguration
            CLOCK_SYS_GetCurrentConfiguration
            CLOCK_SYS_GetErrorCallback
            CLOCK_SYS_GetFreq
```


main.c

```c
int main(void)
{
  /* Write your local variable definition here */

  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  #ifdef PEX_RTOS_INIT
    PEX_RTOS_INIT();                   /* Initialization of the selected RTOS. Macro is defined by the RTOS component. */
  #endif
  /*** End of Processor Expert internal initialization.                    ***/

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

   CLOCK_SYS_Init(g_clockManConfigsArr,
           FSL_CLOCK_MANAGER_CONFIG_CNT,
           g_clockManCallbacksArr,
           FSL_CLOCK_MANAGER_CALLBACK_CNT);
   CLOCK_SYS_UpdateConfiguration(0U,
           CLOCK_MANAGER_POLICY_FORCIBLE);

   Pins_DRV_Init(NUM_OF_CONFIGURED_PINS,g_pin_mux_InitConfigArr);

   for(;;)
   {
        if(GPIO_HAL_ReadPins(PTC)>>12==1)
        {
           GPIO_HAL_ClearPins(PTD, 1<<16);
        }
        else
        {
           GPIO_HAL_SetPins(PTD, 1<<16);
        }
   }

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

S32K144 Clocks Lab: Add Init and Update Configuration functions

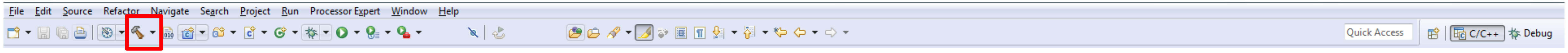
- In the **CLOCK_SYS_Init** function add the following parameters.
  - g_clockManConfigsArr,
  - CLOCK_MANAGER_CONFIG_CNT,
  - g_clockManCallbacksArr,
  - CLOCK_MANAGER_CALLBACK_CNT
- In the **CLOCK_SYS_UpdateConfiguration** add the following parameters.
  - 0U,
  - CLOCK_MANAGER_POLICY_FORCIBLE



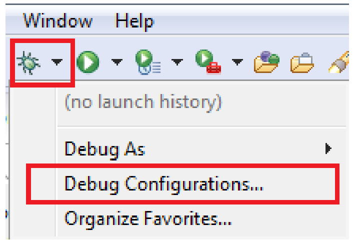
```
CLOCK_SYS_Init(g_clockManConfigsArr, FSL_CLOCK_MANAGER_CONFIG_CNT,
        g_clockManCallbacksArr, FSL_CLOCK_MANAGER_CALLBACK_CNT);
CLOCK_SYS_UpdateConfiguration(0U, CLOCK_MANAGER_POLICY_FORCIBLE);
```

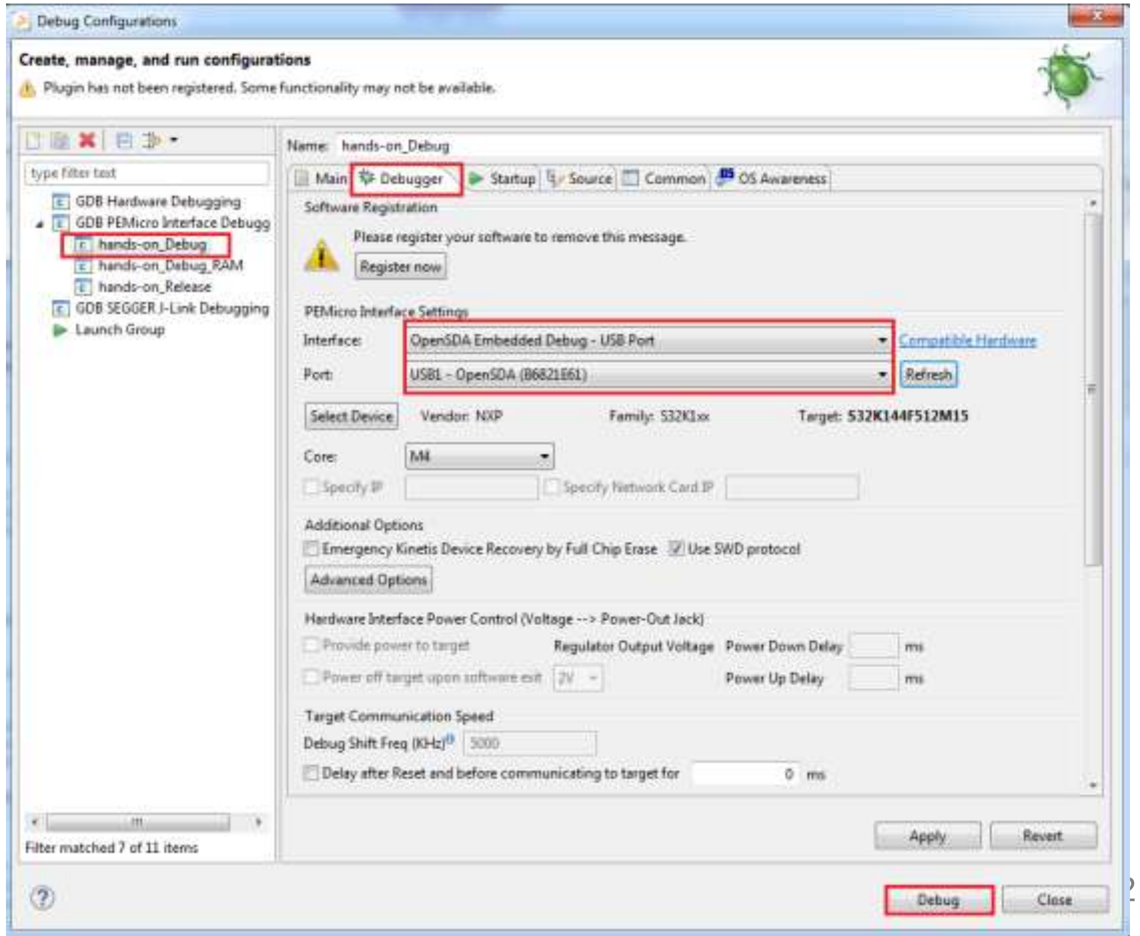
S32K144 Clocks Lab: Build and debug the lab

- Click on the build icon to make sure that there a no compiler errors.



- Configure the debug configuration start a new debug session





Debug Configurations

Create, manage, and run configurations

Plugin has not been registered. Some functionality may not be available.

type filter text
- GDB Hardware Debugging
- GDB PEMicro Interface Debugg
  - hands-on_Debug
  - hands-on_Debug_RAM
  - hands-on_Release
- GDB SEGGER J-Link Debugging
- Launch Group

Filter matched 7 of 11 items

Name: hands-on_Debug

Main | Debugger | Startup | Source | Common | OS Awareness

Software Registration

Please register your software to remove this message.

Register now

PEMicro Interface Settings

Interface: OpenSDA Embedded Debug - USB Port — Compatible Hardware

Port: USB1 - OpenSDA (B6821E61) — Refresh

Select Device — Vendor: NXP — Family: S32K1xx — Target: S32K144F512M15

Core: M4

☐ Specify IP ☐ Specify Network Card IP

Additional Options

☐ Emergency Kinetis Device Recovery by Full Chip Erase ☑ Use SWD protocol

Advanced Options

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☐ Provide power to target — Regulator Output Voltage — Power Down Delay ms

☐ Power off target upon software exit — 2V — Power Up Delay ms

Target Communication Speed

Debug Shift Freq (KHz) 5000

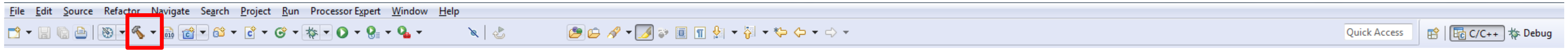
☐ Delay after Reset and before communicating to target for 0 ms

Apply Revert

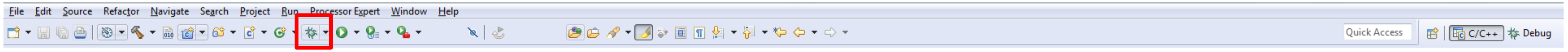
Debug Close

S32K144 Clocks Lab: Build and debug the lab

- Click on the build icon to make sure that there a no compiler errors.

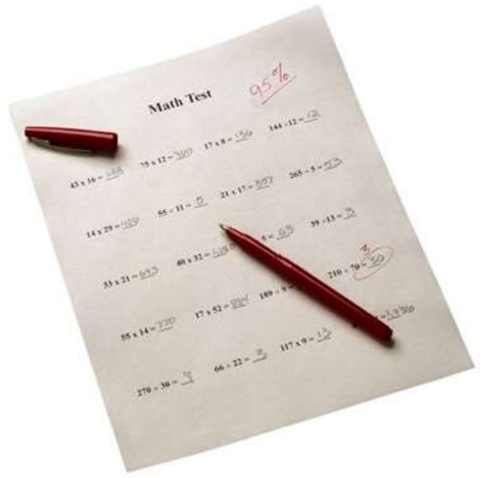

File Edit Source Refactor Navigate Search Project Run Processor Expert Window Help
Quick Access
C/C++
Debug

- Click the debug icon to start a new debug session


File Edit Source Refactor Navigate Search Project Run Processor Expert Window Help
Quick Access
C/C++
Debug

S32K144 Clocks Lab: Quiz

- Which are the sources available for the core clock on the S32K144?
- What is the maximum core speed in S32K144?
- What is the external oscillator range?



S32K144 Interrupts

S32K144 Interrupts Lab: Objective

- **Task:**
  - Use Periodic Interrupt Timer to Interrupt the Application at every 1 sec and toggle LED.

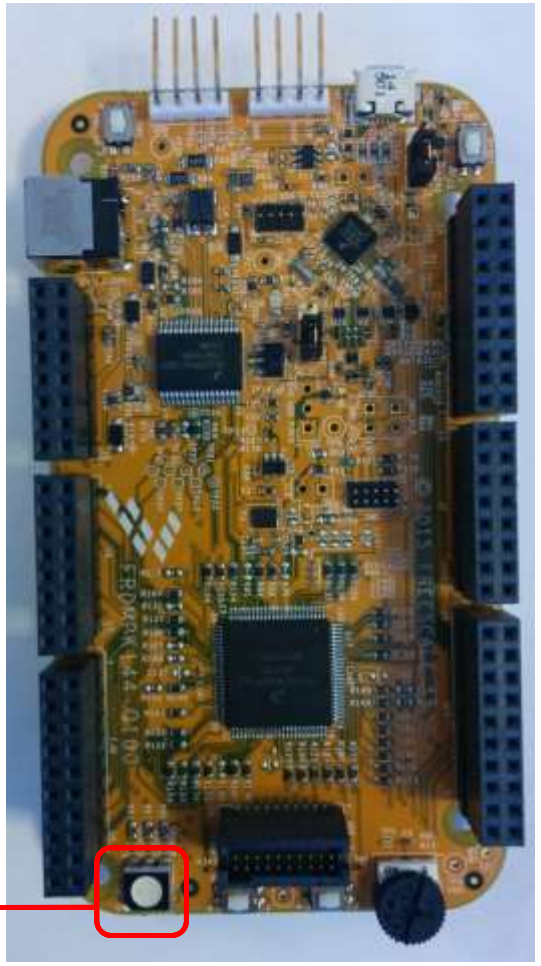
- **Learn:**
  - How interrupts works on S32K144
  - How to use the LPIT peripheral
  - Set up an interrupt in S32K144 using SDK

- **Target Modules:**
  - LPIT – Low Power Periodic Interrupt Timer
  - PCC, PORT, GPIO

S32K144 Interrupts Lab: Resources to be used

- In this lab we will be using the following components of the EVB:
  - RGB LED

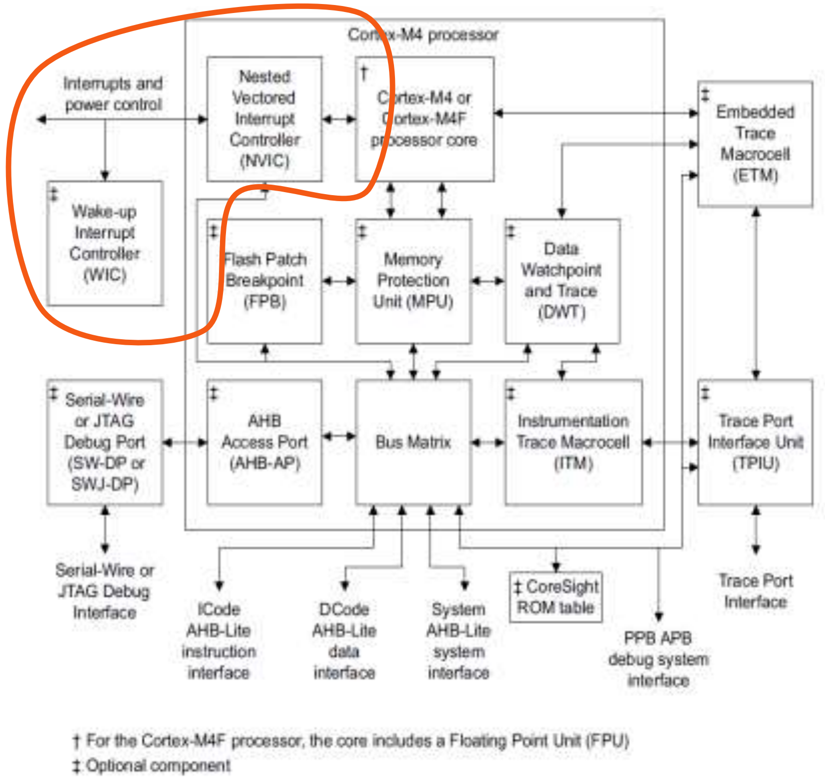
LED	S32K144 PIN
BLUE	PTD0
RED	PTD15
GREEN	PTD16



S32K144 Interrupts Lab: Theory

Nested Vector Interrupt Controller (NVIC)

- Responsible for interrupt handling
- Supports vector table relocation
- Up to 240 vectored interrupts
- 111 interrupts available in S32K144



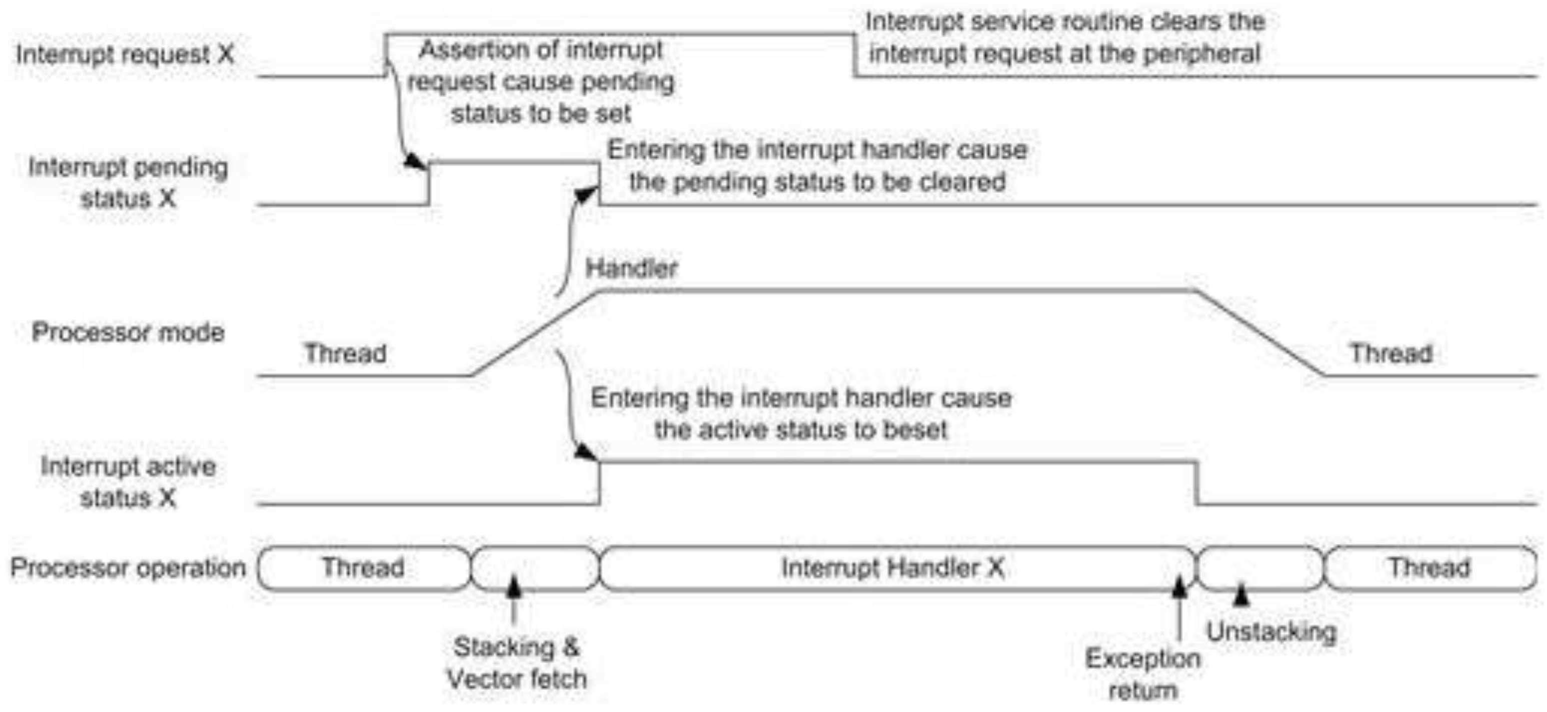
Cortex-M4 processor
Interrupts and power control
Nested Vectored Interrupt Controller (NVIC)
Cortex-M4 or Cortex-M4F processor core
Embedded Trace Macrocell (ETM)
Wake-up Interrupt Controller (WIC)
Flash Patch Breakpoint (FPB)
Memory Protection Unit (MPU)
Data Watchpoint and Trace (DWT)
Serial-Wire or JTAG Debug Port (SW-DP or SWJ-DP)
AHB Access Port (AHB-AP)
Bus Matrix
Instrumentation Trace Macrocell (ITM)
Trace Port Interface Unit (TPIU)
Serial-Wire or JTAG Debug Interface
ICode AHB-Lite instruction interface
DCode AHB-Lite data interface
System AHB-Lite system interface
‡ CoreSight ROM table
PPB APB debug system interface
Trace Port Interface
† For the Cortex-M4F processor, the core includes a Floating Point Unit (FPU)
‡ Optional component

Asynchronous Wake-up Interrupt Controller (AWIC)

- Detect asynchronous wake-up events in stop modes
- Signal to clock control logic to resume system clocking
- After clock restart, NVIC observes the pending interrupt and performs normal interrupt process
- Used during low power modes to generate an wake up signal

S32K144 Interrupts Lab: Theory

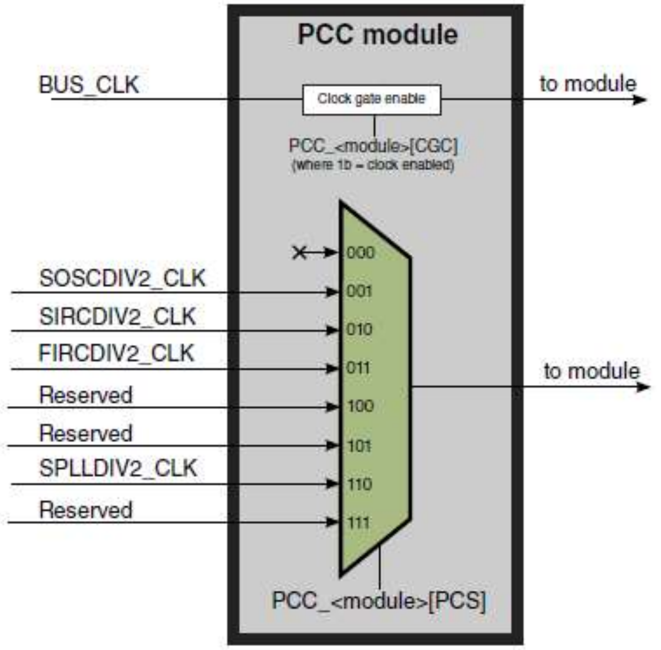
What happens when an interrupt occurs in an ARM Cortex M4?


Interrupt request X
Assertion of interrupt request cause pending status to be set
Interrupt service routine clears the interrupt request at the peripheral
Interrupt pending status X
Entering the interrupt handler cause the pending status to be cleared
Handler
Processor mode
Thread
Thread
Entering the interrupt handler cause the active status to beset
Interrupt active status X
Processor operation
Thread
Interrupt Handler X
Thread
Stacking & Vector fetch
Exception return
Unstacking

S32K144 Interrupts Lab: Theory

LPIT (Low power interrupt timer)

- 4 channels
- Individual or chained channel operation
- 32 bit counter per channel
- 4 operation modes:
  - 32-bit Periodic Counter
  - Dual 16-bit Periodic Counter
  - 32-bit Trigger Accumulator
  - 32-bit Trigger Input Capture



Module	VLPR	VLPW	Stop	VLPS
LPIT	Full functionality	Full functionality	Async operation	Async operation

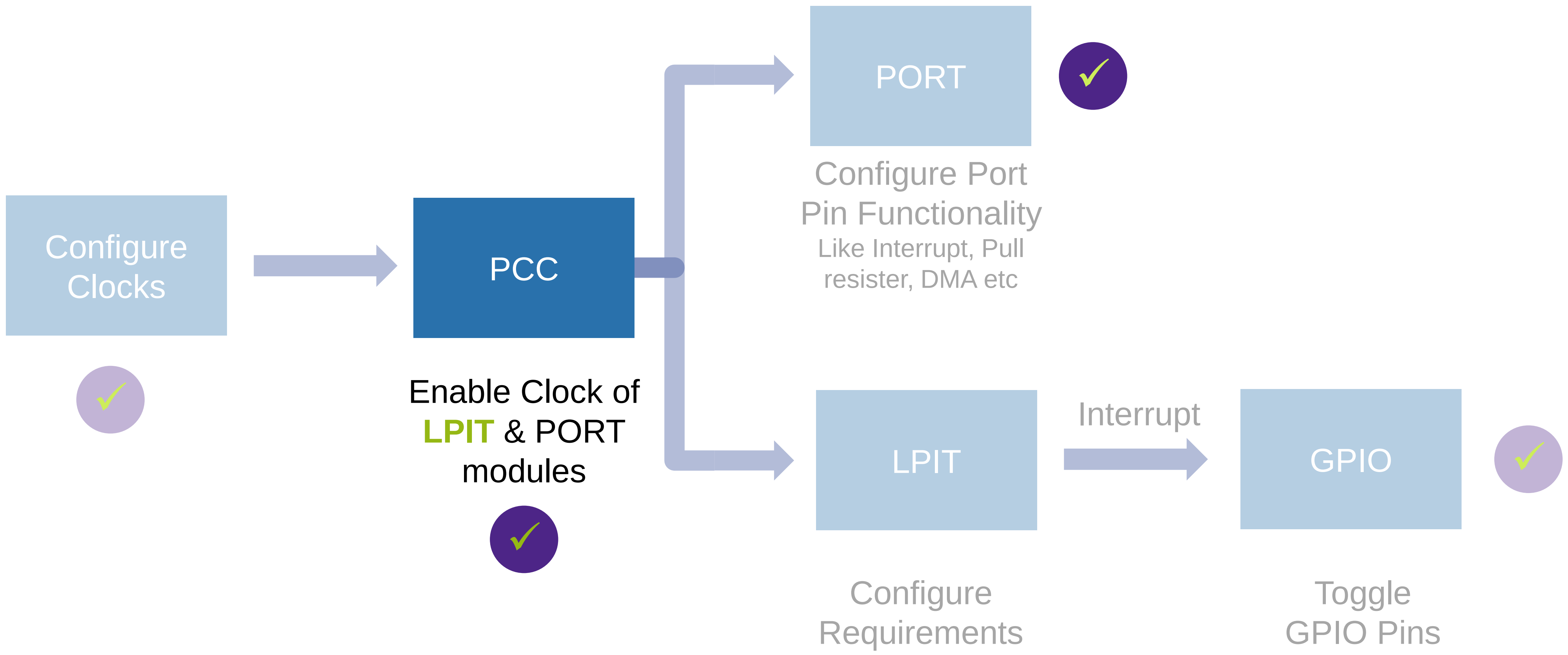
S32K144 Interrupts Lab: Previous Steps

- Create a new S32DS Project
- Setup the clocks of the S32K144
- Setup GPIOs of the S32K144 to enable EVB LEDs and buttons

OR

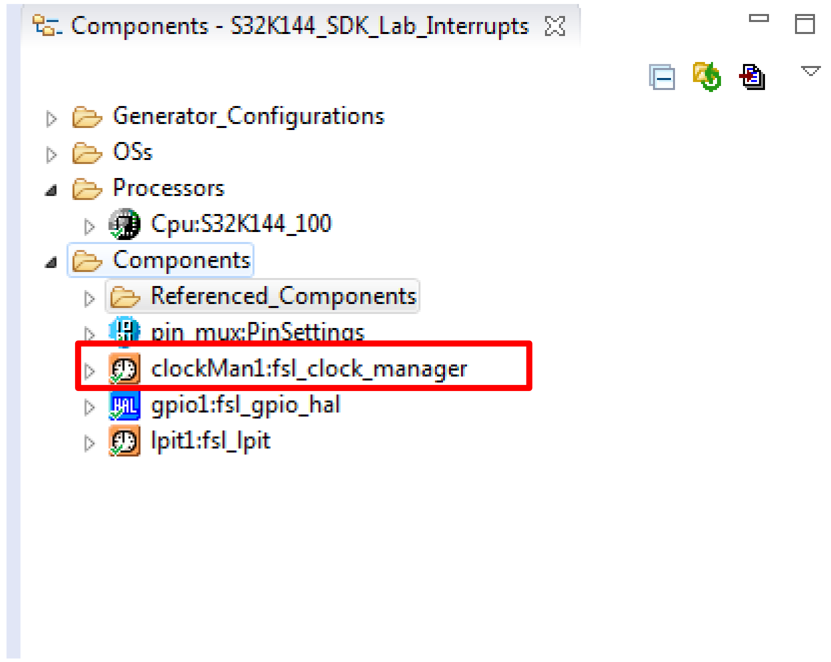
- Just use the Previous Project

S32K144 Interrupts Lab: Step - 1


Configure Clocks
✓
PCC
Enable Clock of LPIT & PORT modules
✓
PORT
✓
Configure Port Pin Functionality
Like Interrupt, Pull resister, DMA etc
LPIT
Interrupt
GPIO
✓
Configure Requirements
Toggle GPIO Pins

S32K144 Interrupts Lab: Select LPIT Clock

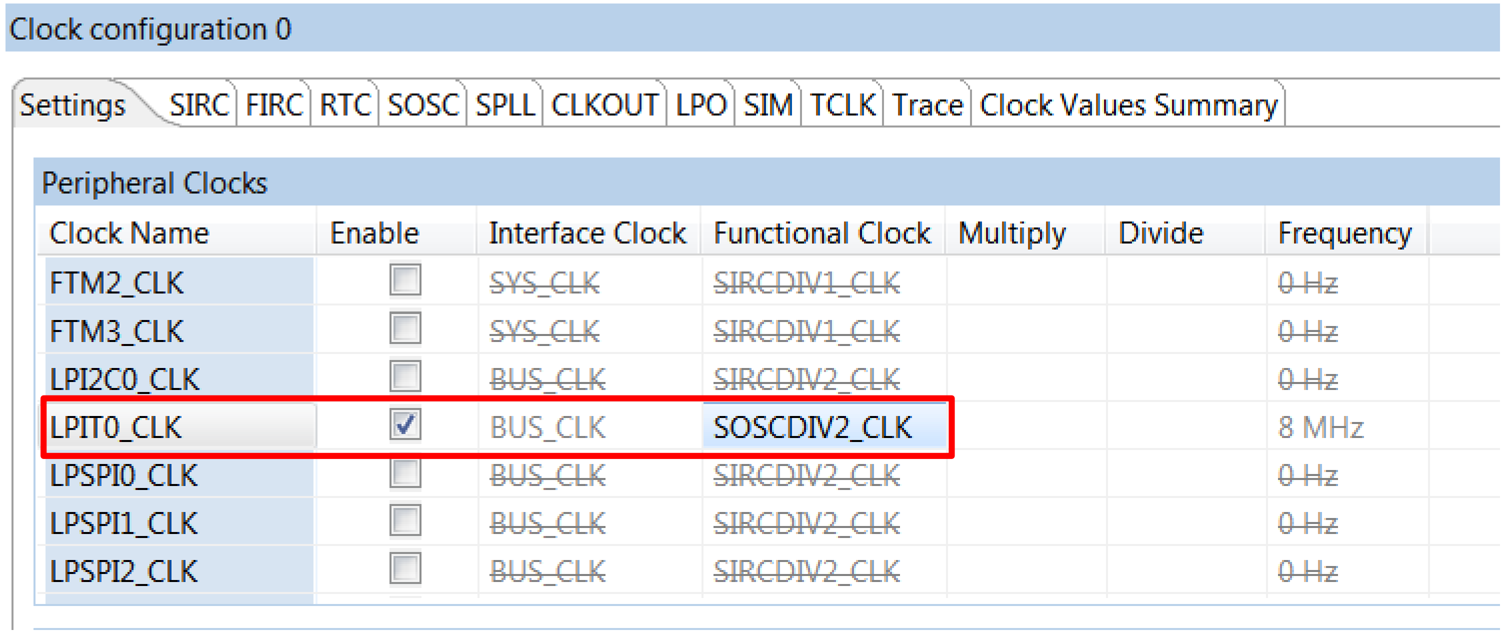
In the **Components Window** select the **Clock Manager component**.


Components - S32K144_SDK_Lab_Interrupts

- Generator_Configurations
- OSs
- Processors
  - Cpu:S32K144_100
- Components
  - Referenced_Components
  - pin_mux:PinSettings
  - clockMan1:fsl_clock_manager
  - gpio1:fsl_gpio_hal
  - lpit1:fsl_lpit

S32K144 Interrupts Lab: Select LPIT Clock

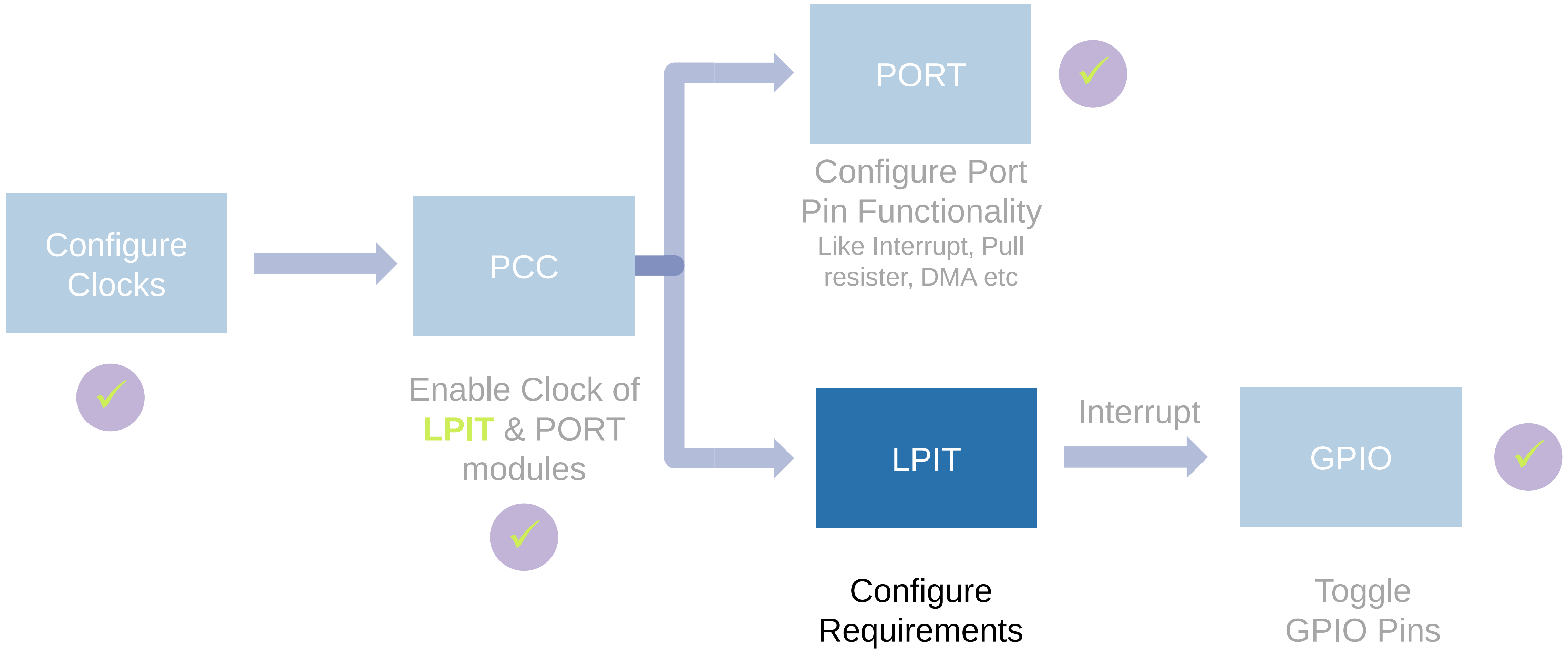
- Go to **Component Inspector**.
- Select the PCC tab
- Check the initialize box for LPIT
- Check the gate box for LPIT
- Select the **System Oscillator Clock Div2** as the clock source



Peripheral Clocks

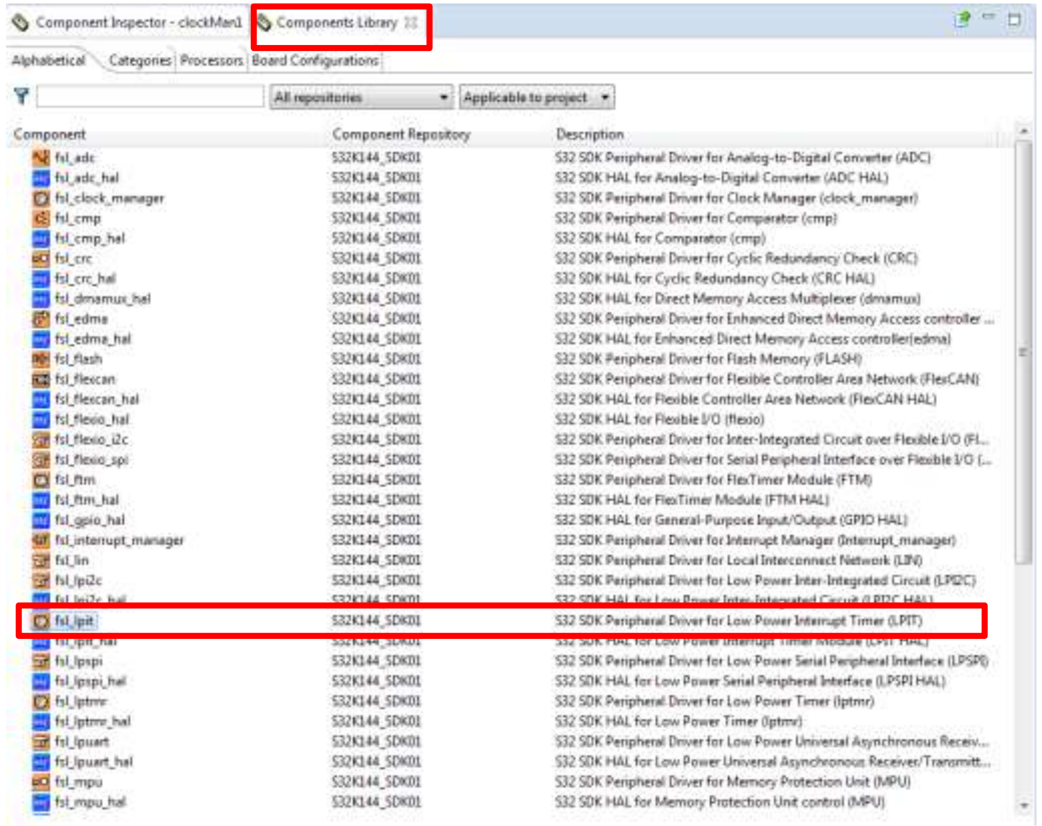
Clock Name	Enable	Interface Clock	Functional Clock	Multiply	Divide	Frequency
FTM2_CLK	☐	~~SYS_CLK~~	~~SIRCDIV1_CLK~~			~~0 Hz~~
FTM3_CLK	☐	~~SYS_CLK~~	~~SIRCDIV1_CLK~~			~~0 Hz~~
LPI2C0_CLK	☐	~~BUS_CLK~~	~~SIRCDIV2_CLK~~			~~0 Hz~~
LPIT0_CLK	✓	BUS_CLK	SOSCDIV2_CLK			8 MHz
LPSPI0_CLK	☐	~~BUS_CLK~~	~~SIRCDIV2_CLK~~			~~0 Hz~~
LPSPI1_CLK	☐	~~BUS_CLK~~	~~SIRCDIV2_CLK~~			~~0 Hz~~
LPSPI2_CLK	☐	~~BUS_CLK~~	~~SIRCDIV2_CLK~~			~~0 Hz~~

S32K144 Interrupts Lab: Step - 2

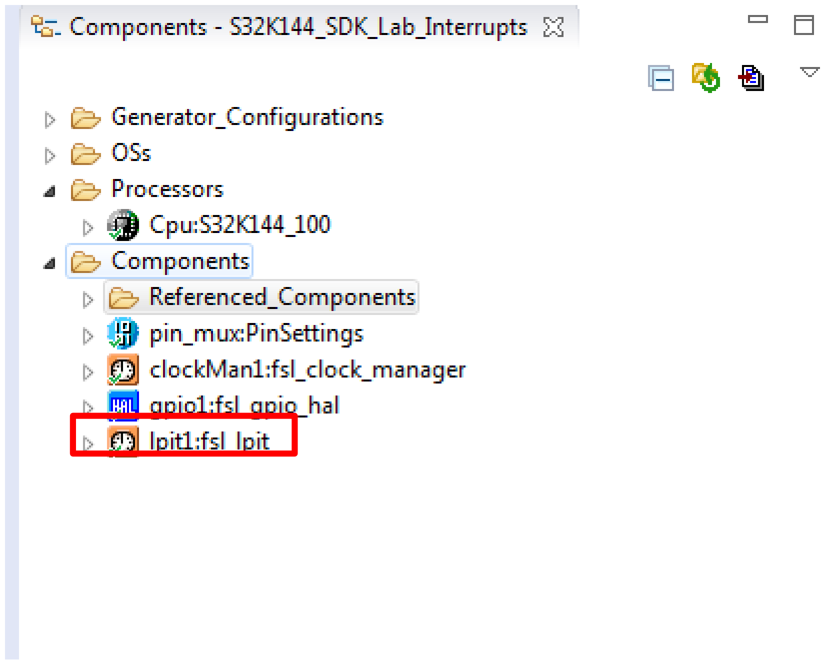

Configure Clocks
PCC
Enable Clock of LPIT & PORT modules
PORT
Configure Port Pin Functionality
Like Interrupt, Pull resister, DMA etc
LPIT
Configure Requirements
Interrupt
GPIO
Toggle GPIO Pins

S32K144 Interrupts Lab: Add LPIT Driver

- Go to **Component Library** window.
- Select the **lpit** in the Alphabetical tab.
- Double click **lpit** to add to your project.
- Lpit component should appear on the component window.



Component	Component Repository	Description
fsl_adc	S32K144_SDK01	S32 SDK Peripheral Driver for Analog-to-Digital Converter (ADC)
fsl_adc_hal	S32K144_SDK01	S32 SDK HAL for Analog-to-Digital Converter (ADC HAL)
fsl_clock_manager	S32K144_SDK01	S32 SDK Peripheral Driver for Clock Manager (clock_manager)
fsl_cmp	S32K144_SDK01	S32 SDK Peripheral Driver for Comparator (cmp)
fsl_cmp_hal	S32K144_SDK01	S32 SDK HAL for Comparator (cmp)
fsl_crc	S32K144_SDK01	S32 SDK Peripheral Driver for Cyclic Redundancy Check (CRC)
fsl_crc_hal	S32K144_SDK01	S32 SDK HAL for Cyclic Redundancy Check (CRC HAL)
fsl_dmamux_hal	S32K144_SDK01	S32 SDK HAL for Direct Memory Access Multiplexer (dmamux)
fsl_edma	S32K144_SDK01	S32 SDK Peripheral Driver for Enhanced Direct Memory Access controller ...
fsl_edma_hal	S32K144_SDK01	S32 SDK HAL for Enhanced Direct Memory Access controller(edma)
fsl_flash	S32K144_SDK01	S32 SDK Peripheral Driver for Flash Memory (FLASH)
fsl_flexcan	S32K144_SDK01	S32 SDK Peripheral Driver for Flexible Controller Area Network (FlexCAN)
fsl_flexcan_hal	S32K144_SDK01	S32 SDK HAL for Flexible Controller Area Network (FlexCAN HAL)
fsl_flexio_hal	S32K144_SDK01	S32 SDK HAL for Flexible I/O (flexio)
fsl_flexio_i2c	S32K144_SDK01	S32 SDK Peripheral Driver for Inter-Integrated Circuit over Flexible I/O (Fl...
fsl_flexio_spi	S32K144_SDK01	S32 SDK Peripheral Driver for Serial Peripheral Interface over Flexible I/O (...
fsl_ftm	S32K144_SDK01	S32 SDK Peripheral Driver for FlexTimer Module (FTM)
fsl_ftm_hal	S32K144_SDK01	S32 SDK HAL for FlexTimer Module (FTM HAL)
fsl_gpio_hal	S32K144_SDK01	S32 SDK HAL for General-Purpose Input/Output (GPIO HAL)
fsl_interrupt_manager	S32K144_SDK01	S32 SDK Peripheral Driver for Interrupt Manager (Interrupt_manager)
fsl_lin	S32K144_SDK01	S32 SDK Peripheral Driver for Local Interconnect Network (LIN)
fsl_lpi2c	S32K144_SDK01	S32 SDK Peripheral Driver for Low Power Inter-Integrated Circuit (LPI2C)
fsl_lpi2c_hal	S32K144_SDK01	S32 SDK HAL for Low Power Inter-Integrated Circuit (LPI2C HAL)
fsl_lpit	S32K144_SDK01	S32 SDK Peripheral Driver for Low Power Interrupt Timer (LPIT)
fsl_lpit_hal	S32K144_SDK01	S32 SDK HAL for Low Power Interrupt Timer Module (LPIT HAL)
fsl_lpspi	S32K144_SDK01	S32 SDK Peripheral Driver for Low Power Serial Peripheral Interface (LPSPI)
fsl_lpspi_hal	S32K144_SDK01	S32 SDK HAL for Low Power Serial Peripheral Interface (LPSPI HAL)
fsl_lptmr	S32K144_SDK01	S32 SDK Peripheral Driver for Low Power Timer (lptmr)
fsl_lptmr_hal	S32K144_SDK01	S32 SDK HAL for Low Power Timer (lptmr)
fsl_lpuart	S32K144_SDK01	S32 SDK Peripheral Driver for Low Power Universal Asynchronous Receiv...
fsl_lpuart_hal	S32K144_SDK01	S32 SDK HAL for Low Power Universal Asynchronous Receiver/Transmitt...
fsl_mpu	S32K144_SDK01	S32 SDK Peripheral Driver for Memory Protection Unit (MPU)
fsl_mpu_hal	S32K144_SDK01	S32 SDK HAL for Memory Protection Unit control (MPU)

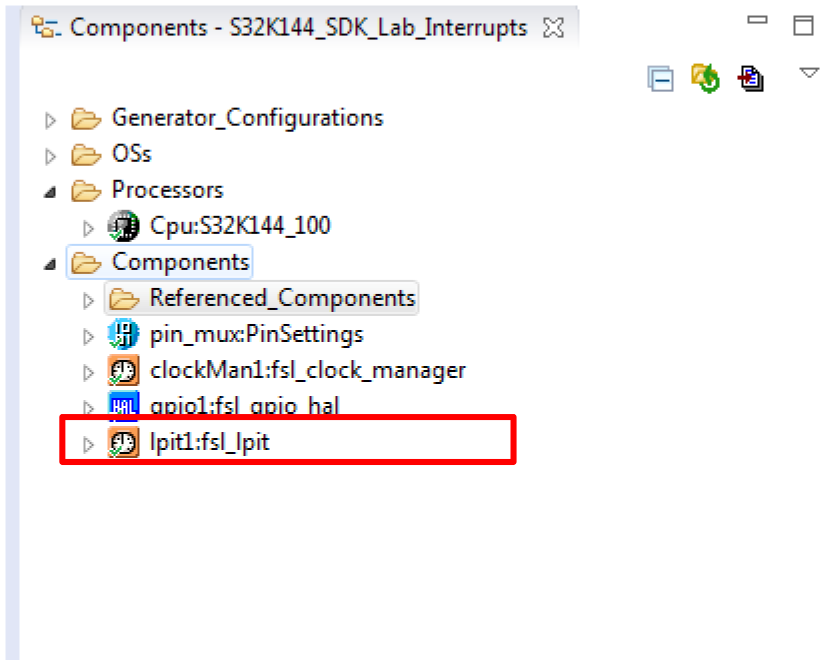


Components - S32K144_SDK_Lab_Interrupts

```
▷ Generator_Configurations
▷ OSs
◢ Processors
    ▷ Cpu:S32K144_100
◢ Components
    ▷ Referenced_Components
    ▷ pin_mux:PinSettings
    ▷ clockMan1:fsl_clock_manager
    ▷ gpio1:fsl_gpio_hal
    ▷ lpit1:fsl_lpit
```

S32K144 Interrupts Lab: Select LPIT Clock

In the **Components Window** select the **lpit**

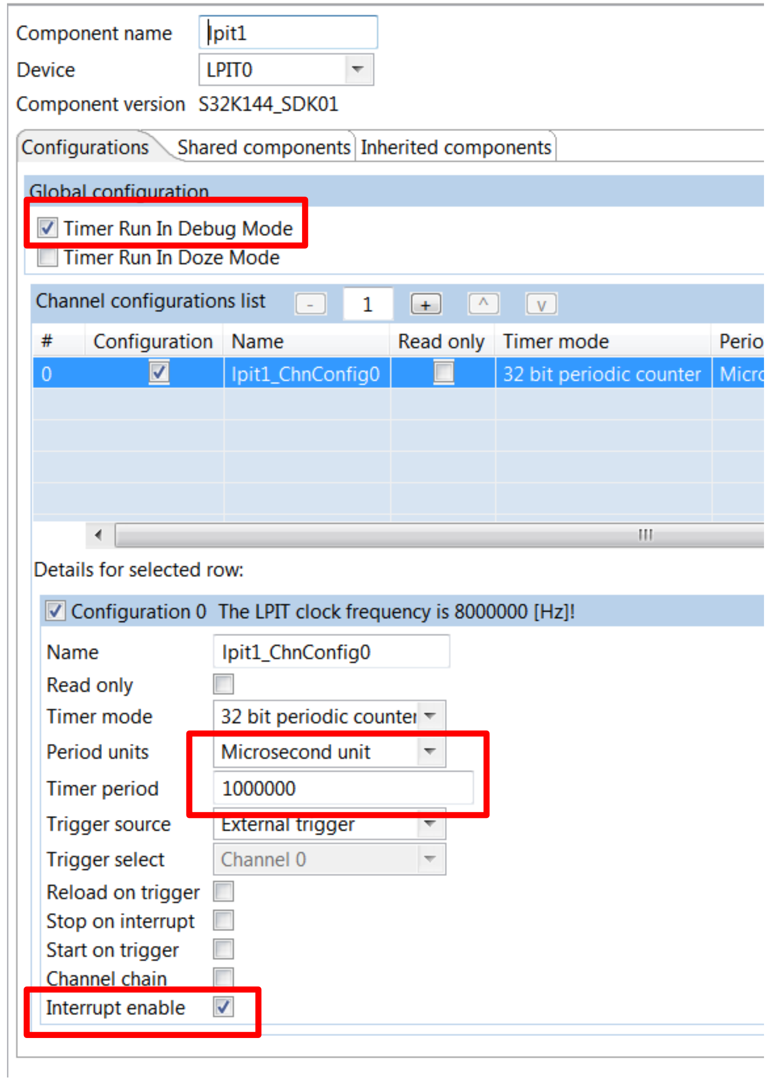


Components - S32K144_SDK_Lab_Interrupts

- Generator_Configurations
- OSs
- Processors
  - Cpu:S32K144_100
- Components
  - Referenced_Components
  - pin_mux:PinSettings
  - clockMan1:fsl_clock_manager
  - gpio1:fsl_gpio_hal
  - lpit1:fsl_lpit

S32K144 Interrupts Lab: Select LPIT Clock

- Go to **Components Inspector**.
- Check the **Timer Run In Debug Mode** box
- Check the **Interrupt enable** box
- In the **Time period [us]** field type **1000000** counts for 1 sec.



Component name	lpit1
Device	LPIT0
Component version	S32K144_SDK01

Configurations | Shared components | Inherited components

Global configuration

- [✓] Timer Run In Debug Mode
- [ ] Timer Run In Doze Mode

Channel configurations list 1

#	Configuration	Name	Read only	Timer mode	Perio
0	✓	lpit1_ChnConfig0	☐	32 bit periodic counter	Micro

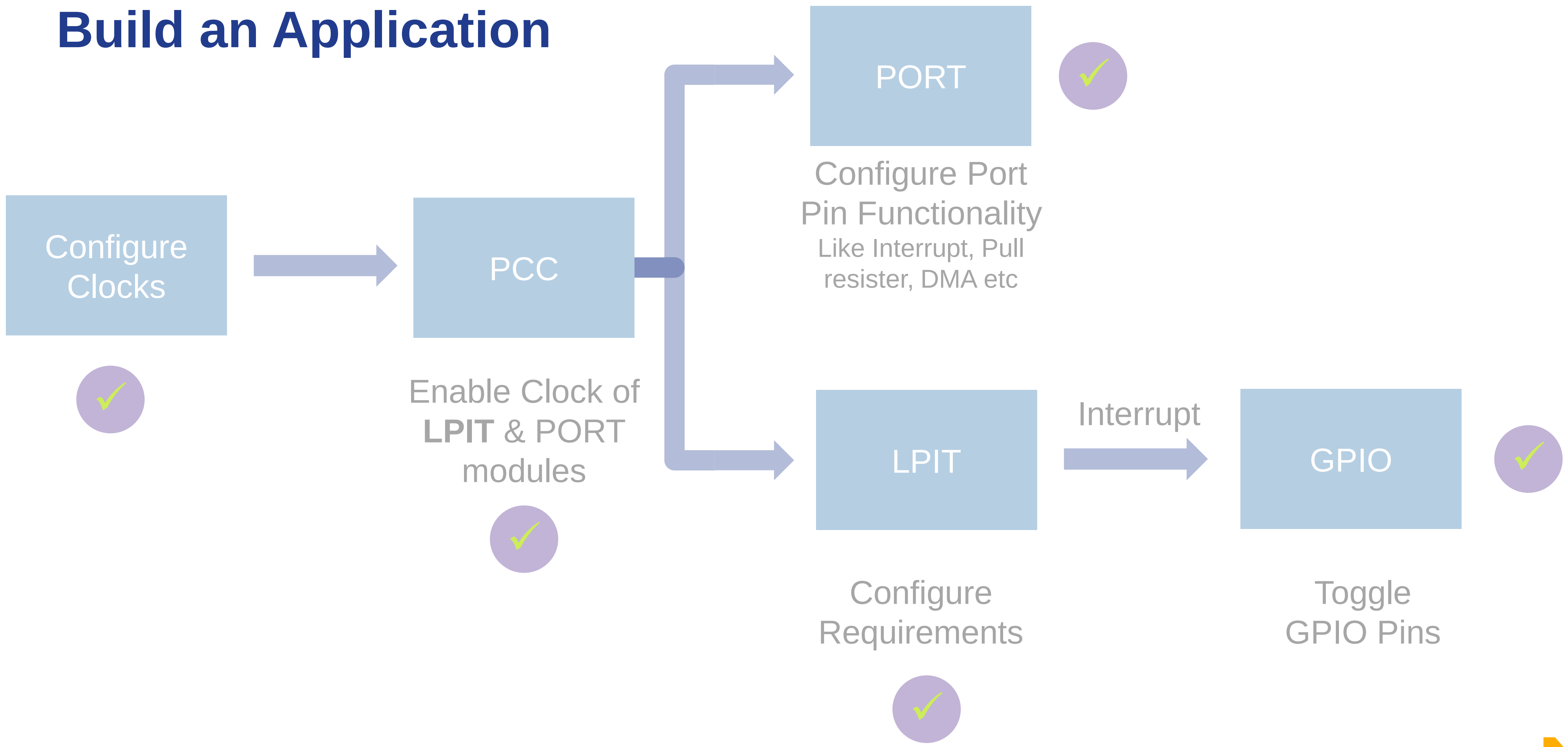
Details for selected row:

✓ Configuration 0 The LPIT clock frequency is 8000000 [Hz]!

Name	lpit1_ChnConfig0
Read only	☐
Timer mode	32 bit periodic counter
Period units	Microsecond unit
Timer period	1000000
Trigger source	External trigger
Trigger select	Channel 0
Reload on trigger	☐
Stop on interrupt	☐
Start on trigger	☐
Channel chain	☐
Interrupt enable	✓

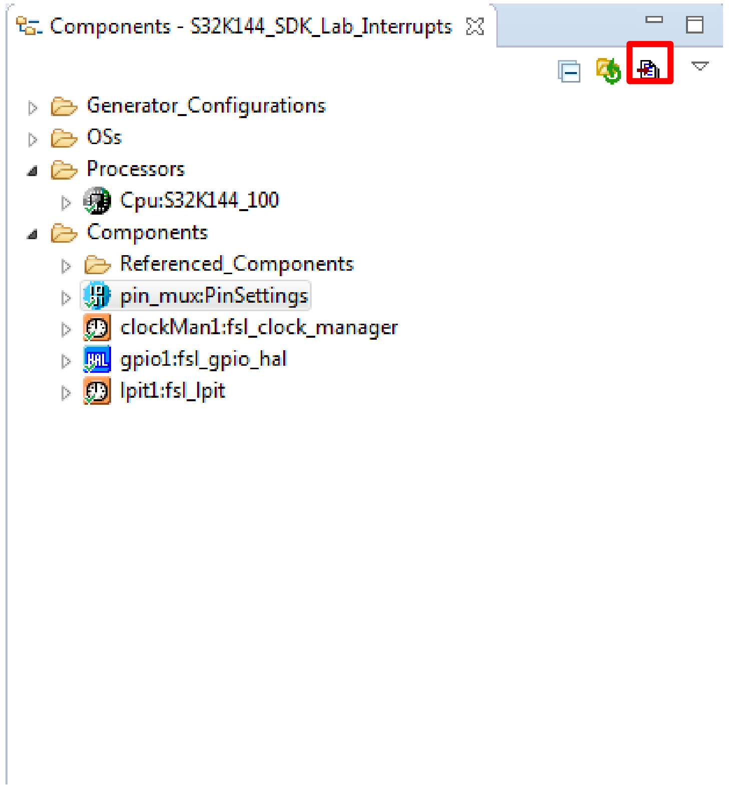
S32K144 Interrupts Lab: Step - 3

Build an Application

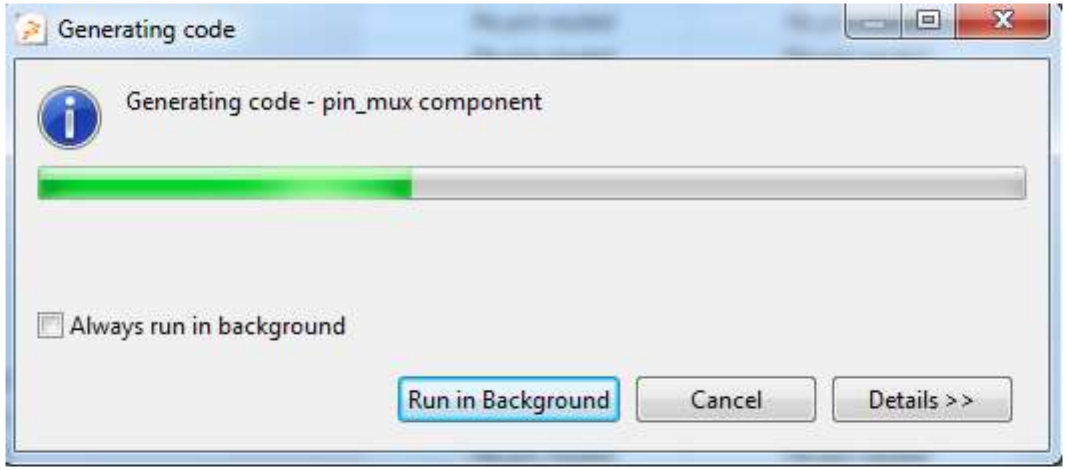

Configure Clocks
✓
PCC
Enable Clock of **LPIT** & PORT modules
✓
PORT
✓
Configure Port Pin Functionality
Like Interrupt, Pull resister, DMA etc
LPIT
Interrupt
GPIO
✓
Configure Requirements
✓
Toggle GPIO Pins

S32K144 Interrupts Lab: Generate the code

- To generate the code for the configuration select, click the **generate code** icon in the **Components** window.
- Wait for the code to be generated.


Components - S32K144_SDK_Lab_Interrupts

- Generator_Configurations
- OSs
- Processors
  - Cpu:S32K144_100
- Components
  - Referenced_Components
  - pin_mux:PinSettings
  - clockMan1:fsl_clock_manager
  - gpio1:fsl_gpio_hal
  - lpit1:fsl_lpit

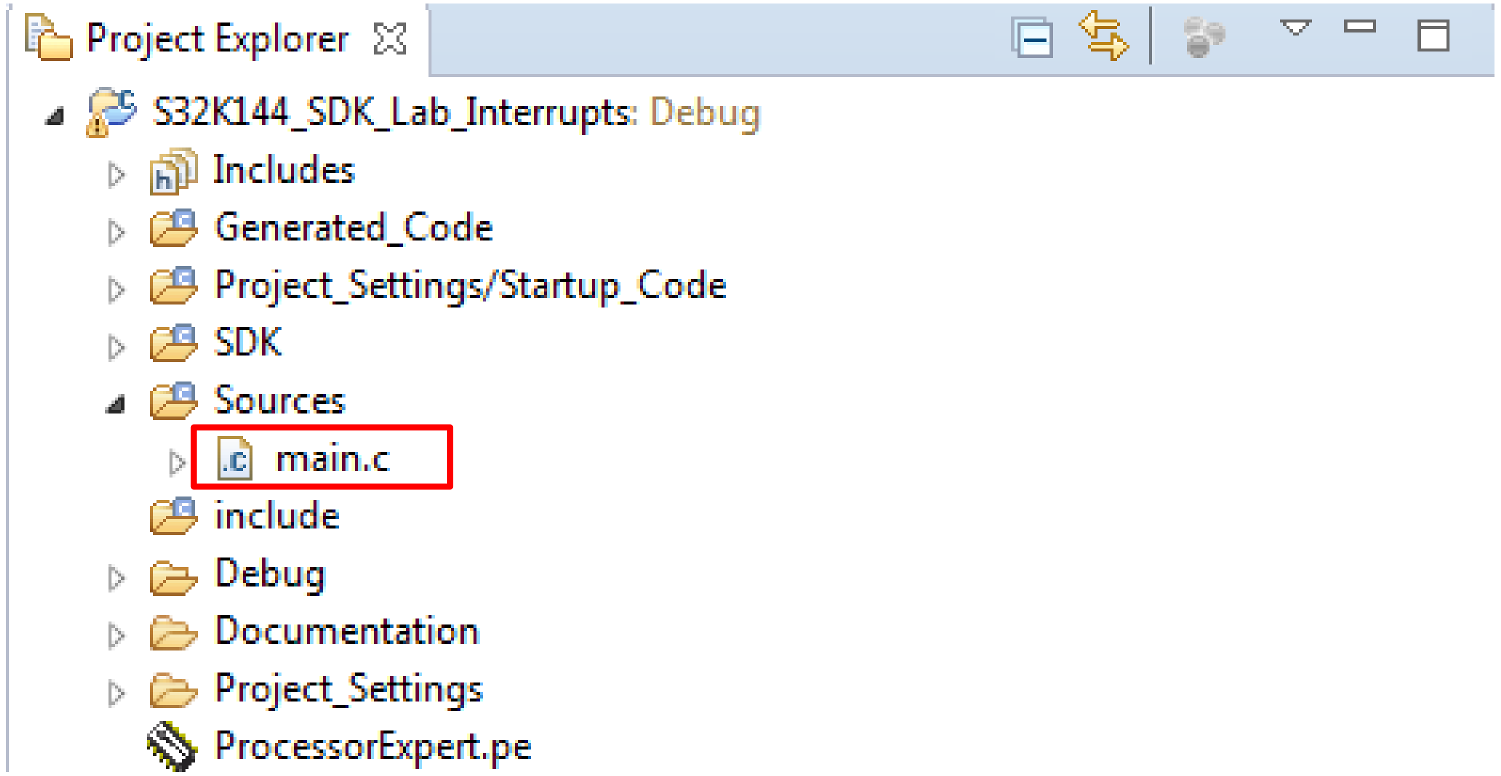

Generating code

Generating code - pin_mux component

Always run in background

S32K144 Interrupts Lab: Open the main.c

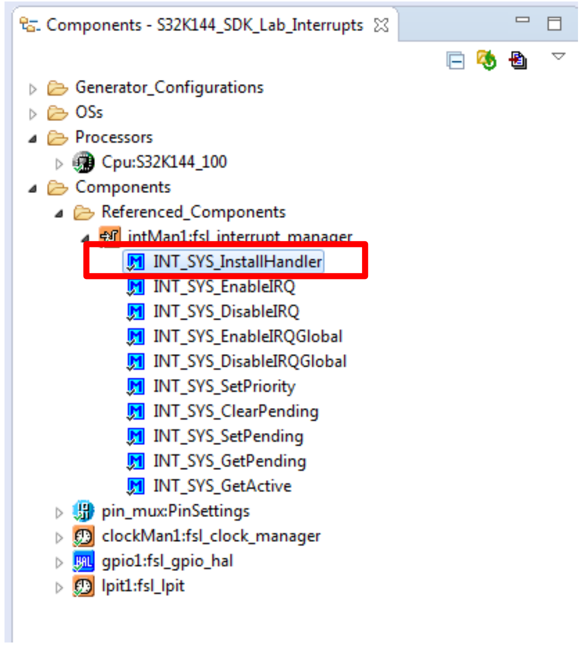
- In the project window double click the **main.c** file to open it



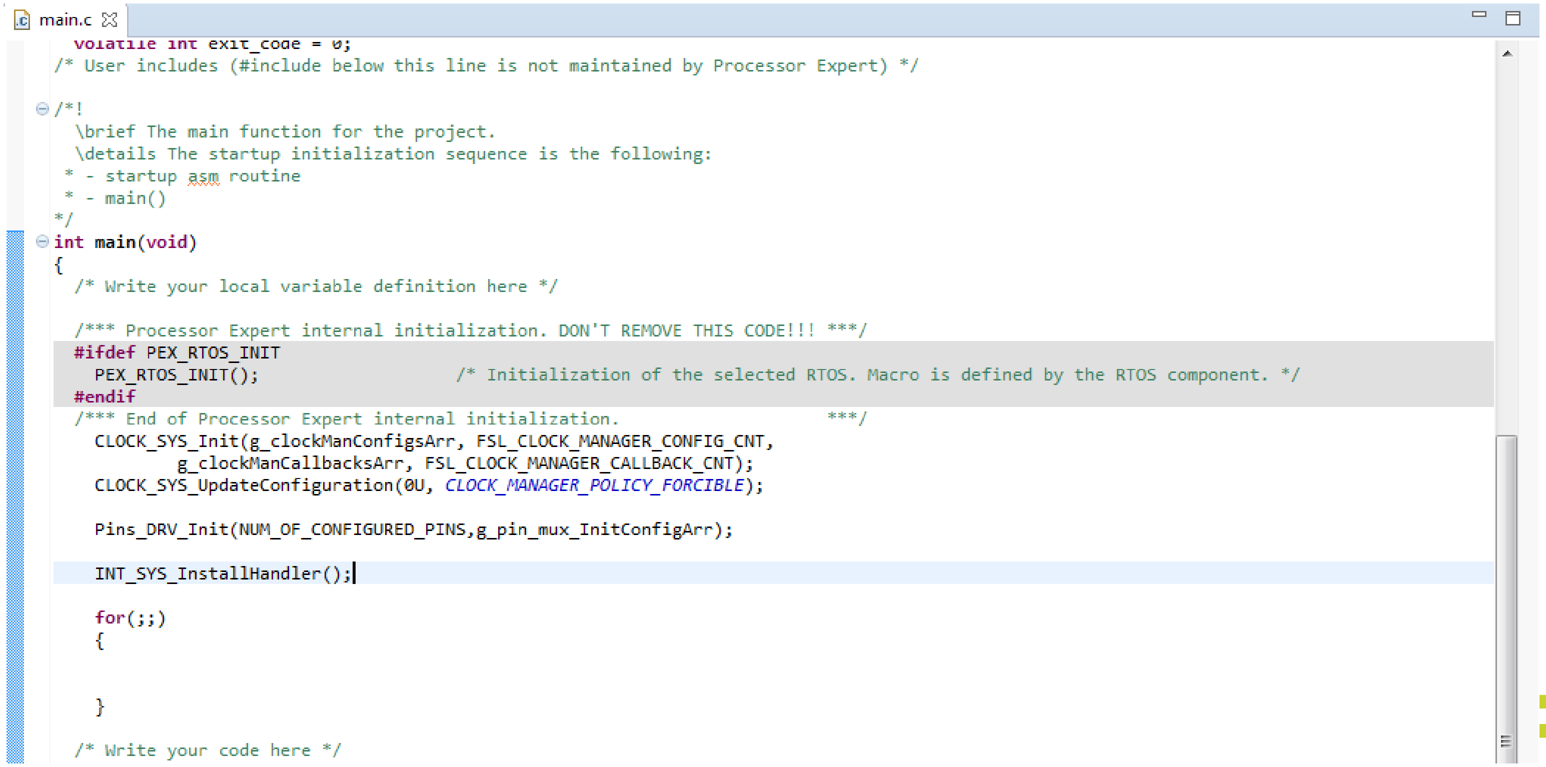
S32K144 Interrupts Lab: Install LPIT interrupt

- In the **Components** Window go to

  Components-> Referenced Components->interrupt_manager
- Exapnd the **interrupt_manager** component
- Drag and drop the **INT_SYS_InstallHandle**r function. Placed it after the **Pins_DRV_Init** function in main.c


Components - S32K144_SDK_Lab_Interrupts

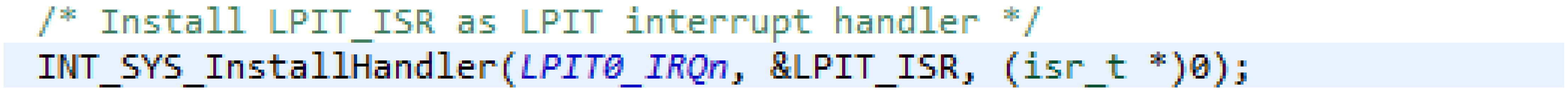
```
Generator_Configurations
OSs
Processors
  Cpu:S32K144_100
Components
  Referenced_Components
    intMan1:fsl_interrupt_manager
      INT_SYS_InstallHandler
      INT_SYS_EnableIRQ
      INT_SYS_DisableIRQ
      INT_SYS_EnableIRQGlobal
      INT_SYS_DisableIRQGlobal
      INT_SYS_SetPriority
      INT_SYS_ClearPending
      INT_SYS_SetPending
      INT_SYS_GetPending
      INT_SYS_GetActive
  pin_mux:PinSettings
  clockMan1:fsl_clock_manager
  gpio1:fsl_gpio_hal
  lpit1:fsl_lpit
```


main.c

```c
  volatile int exit_code = 0;
/* User includes (#include below this line is not maintained by Processor Expert) */

/*!
  \brief The main function for the project.
  \details The startup initialization sequence is the following:
 * - startup asm routine
 * - main()
*/
int main(void)
{
  /* Write your local variable definition here */

  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  #ifdef PEX_RTOS_INIT
    PEX_RTOS_INIT();                   /* Initialization of the selected RTOS. Macro is defined by the RTOS component. */
  #endif
  /*** End of Processor Expert internal initialization.                    ***/
    CLOCK_SYS_Init(g_clockManConfigsArr, FSL_CLOCK_MANAGER_CONFIG_CNT,
                   g_clockManCallbacksArr, FSL_CLOCK_MANAGER_CALLBACK_CNT);
    CLOCK_SYS_UpdateConfiguration(0U, CLOCK_MANAGER_POLICY_FORCIBLE);

    Pins_DRV_Init(NUM_OF_CONFIGURED_PINS,g_pin_mux_InitConfigArr);

    INT_SYS_InstallHandler();

    for(;;)
    {


    }

  /* Write your code here */
```

S32K144 Interrupts Lab: Install LPIT interrupt

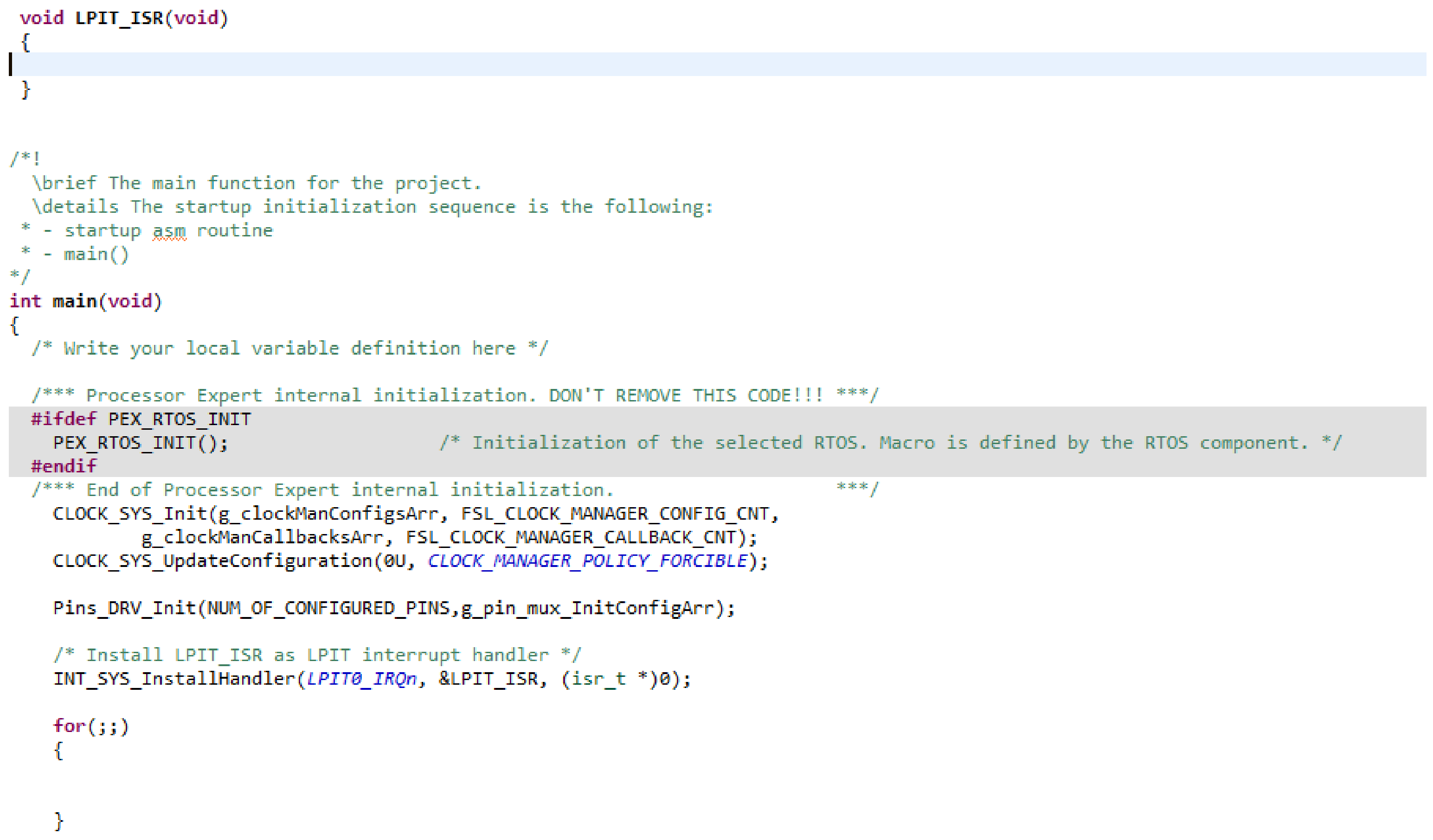
- In the **INT_SYS_InstallHandle**r function add the following parameters:
  - LPIT0_IRQn,
  - &LPIT_ISR,
  - (isr_t *)0



```
/* Install LPIT_ISR as LPIT interrupt handler */
INT_SYS_InstallHandler(LPIT0_IRQn, &LPIT_ISR, (isr_t *)0);
```

S32K144 Interrupts Lab: Install LPIT interrupt

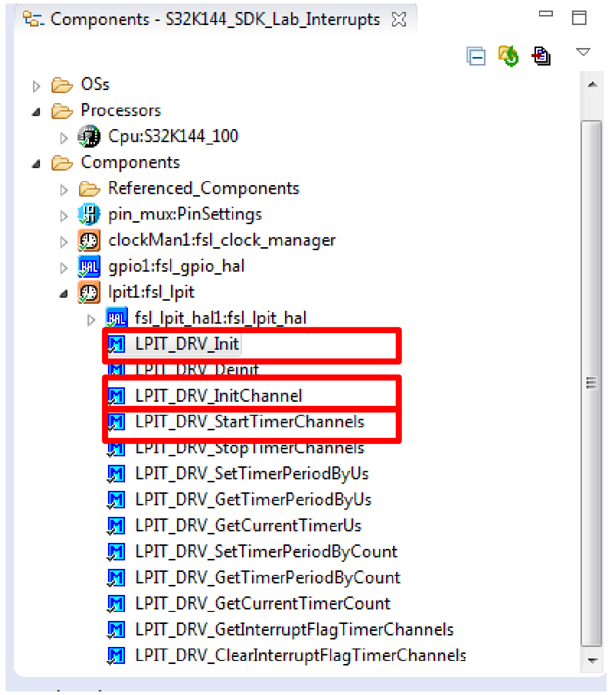
- Create a new function named LPIT_ISR and placed above main



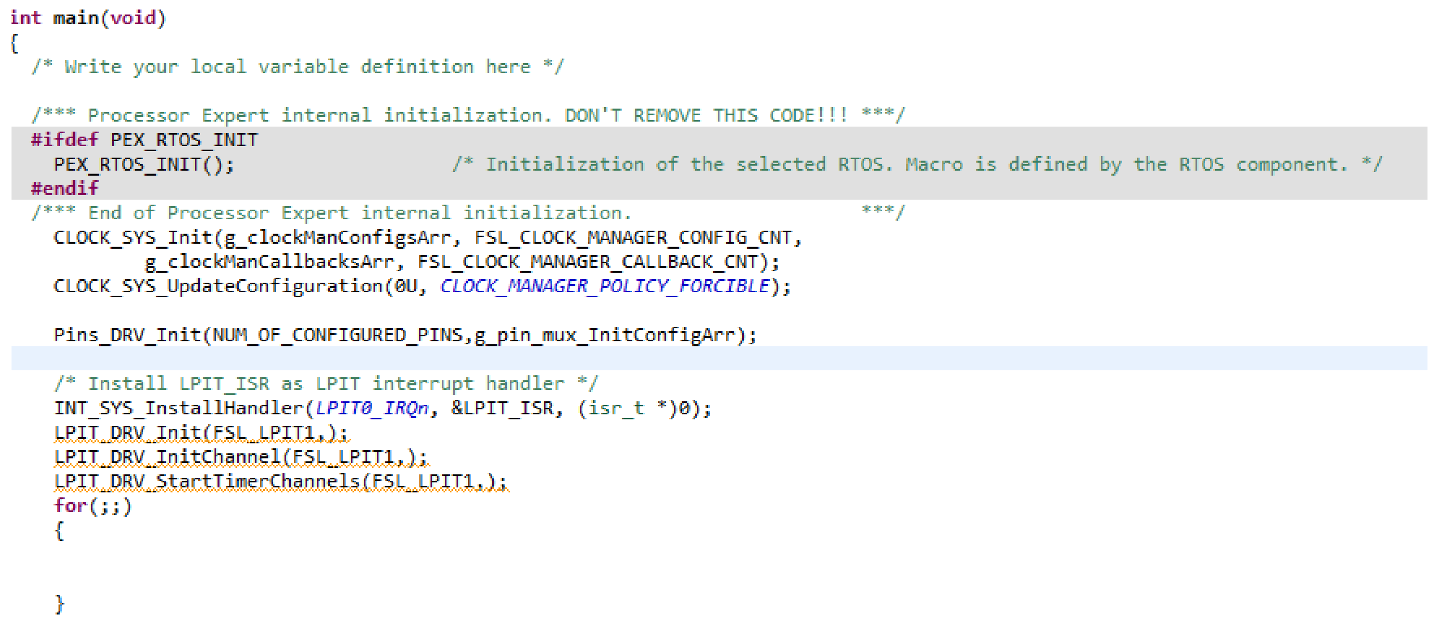
```
 void LPIT_ISR(void)
 {

 }


/*!
  \brief The main function for the project.
  \details The startup initialization sequence is the following:
 * - startup asm routine
 * - main()
*/
int main(void)
{
  /* Write your local variable definition here */

  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  #ifdef PEX_RTOS_INIT
    PEX_RTOS_INIT();                 /* Initialization of the selected RTOS. Macro is defined by the RTOS component. */
  #endif
  /*** End of Processor Expert internal initialization.                    ***/
    CLOCK_SYS_Init(g_clockManConfigsArr, FSL_CLOCK_MANAGER_CONFIG_CNT,
            g_clockManCallbacksArr, FSL_CLOCK_MANAGER_CALLBACK_CNT);
    CLOCK_SYS_UpdateConfiguration(0U, CLOCK_MANAGER_POLICY_FORCIBLE);

    Pins_DRV_Init(NUM_OF_CONFIGURED_PINS,g_pin_mux_InitConfigArr);

    /* Install LPIT_ISR as LPIT interrupt handler */
    INT_SYS_InstallHandler(LPIT0_IRQn, &LPIT_ISR, (isr_t *)0);

    for(;;)
    {


    }
```

S32K144 Interrupts Lab: Initialize LPIT

- Expand the **lpit** component in the **Components** Window
- Drag and drop the following functions in to main, place them after the **INT_SYS_InstallHandle**r function
  - **LPIT_DRV_Init**
  - **LPIT_DRV_InitChannel**
  - **LPIT_DRV_StartTimerChannels**



Components - S32K144_SDK_Lab_Interrupts
OSs
Processors
Cpu:S32K144_100
Components
Referenced_Components
pin_mux:PinSettings
clockMan1:fsl_clock_manager
gpio1:fsl_gpio_hal
lpit1:fsl_lpit
fsl_lpit_hal1:fsl_lpit_hal
LPIT_DRV_Init
LPIT_DRV_Deinit
LPIT_DRV_InitChannel
LPIT_DRV_StartTimerChannels
LPIT_DRV_StopTimerChannels
LPIT_DRV_SetTimerPeriodByUs
LPIT_DRV_GetTimerPeriodByUs
LPIT_DRV_GetCurrentTimerUs
LPIT_DRV_SetTimerPeriodByCount
LPIT_DRV_GetTimerPeriodByCount
LPIT_DRV_GetCurrentTimerCount
LPIT_DRV_GetInterruptFlagTimerChannels
LPIT_DRV_ClearInterruptFlagTimerChannels



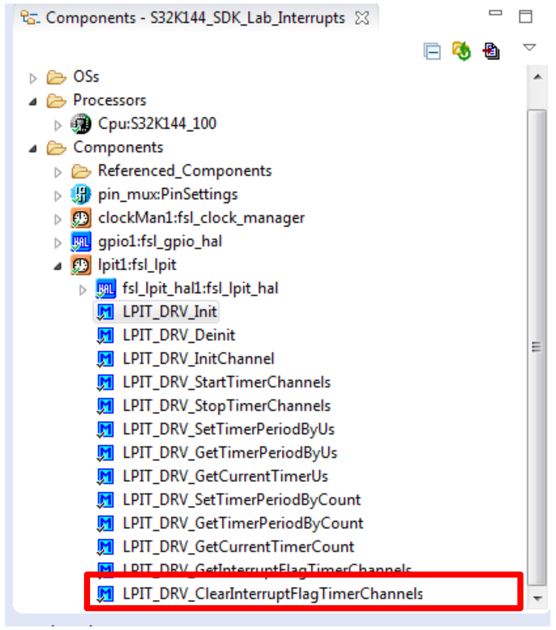
```
int main(void)
{
  /* Write your local variable definition here */

  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  #ifdef PEX_RTOS_INIT
    PEX_RTOS_INIT();                 /* Initialization of the selected RTOS. Macro is defined by the RTOS component. */
  #endif
  /*** End of Processor Expert internal initialization.                    ***/
    CLOCK_SYS_Init(g_clockManConfigsArr, FSL_CLOCK_MANAGER_CONFIG_CNT,
            g_clockManCallbacksArr, FSL_CLOCK_MANAGER_CALLBACK_CNT);
    CLOCK_SYS_UpdateConfiguration(0U, CLOCK_MANAGER_POLICY_FORCIBLE);

    Pins_DRV_Init(NUM_OF_CONFIGURED_PINS,g_pin_mux_InitConfigArr);

    /* Install LPIT_ISR as LPIT interrupt handler */
    INT_SYS_InstallHandler(LPIT0_IRQn, &LPIT_ISR, (isr_t *)0);
    LPIT_DRV_Init(FSL_LPIT1,);
    LPIT_DRV_InitChannel(FSL_LPIT1,);
    LPIT_DRV_StartTimerChannels(FSL_LPIT1,);
    for(;;)
    {


    }
```

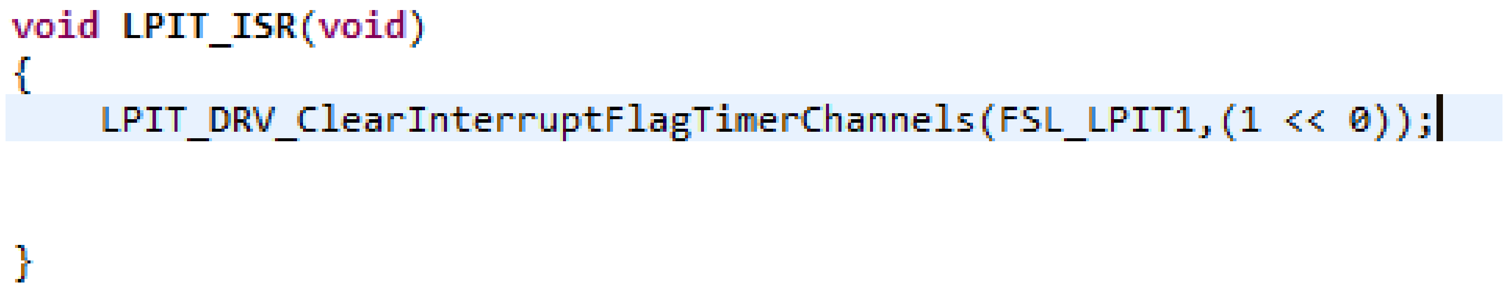
S32K144 Interrupts Lab: Initialize LPIT

- In the **LPIT_DRV_Init** function add the following parameters:.
  - INST_LPIT1,
  - &lpit1_InitConfig
- In the **LPIT_DRV_InitChannel** function add the following parameters:.
  - INST_LPIT1,
  - 0,
  - &lpit1_ChnConfig0
- In the **LPIT_DRV_StartTimerChannels** function add the following parameters:.
  - INST_LPIT1,
  - (1 << 0)

S32K144 Interrupts Lab: Clear LPIT Flag in interrupt

- Expand the **lpit** component in the **Components** Window
- Drag and drop the following function into **LPIT_ISR:**
  - **LPIT_DRV_ClearInterruptFlagTimerChannels**
- In the **LPIT_DRV_ClearInterruptFlagTimerChannels** function add the following parameters:.
  - LPIT1,
  - (1 << 0)

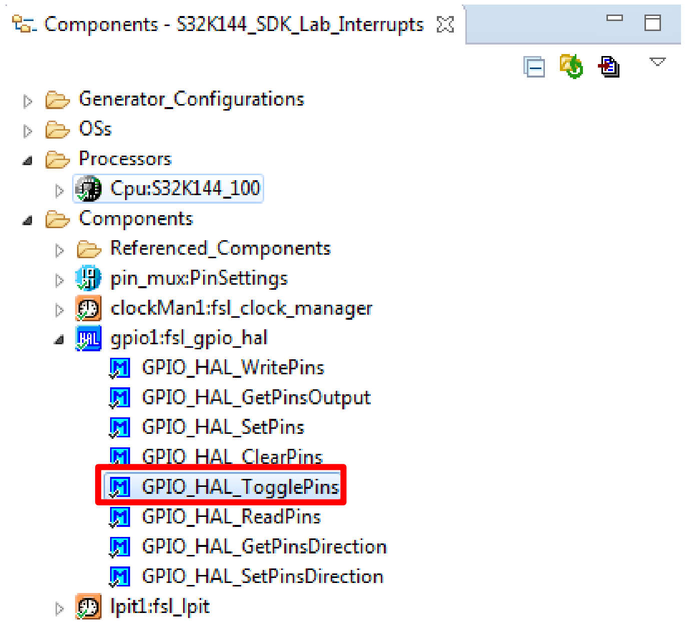

Components - S32K144_SDK_Lab_Interrupts
OSs
Processors
Cpu:S32K144_100
Components
Referenced_Components
pin_mux:PinSettings
clockMan1:fsl_clock_manager
gpio1:fsl_gpio_hal
lpit1:fsl_lpit
fsl_lpit_hal1:fsl_lpit_hal
LPIT_DRV_Init
LPIT_DRV_Deinit
LPIT_DRV_InitChannel
LPIT_DRV_StartTimerChannels
LPIT_DRV_StopTimerChannels
LPIT_DRV_SetTimerPeriodByUs
LPIT_DRV_GetTimerPeriodByUs
LPIT_DRV_GetCurrentTimerUs
LPIT_DRV_SetTimerPeriodByCount
LPIT_DRV_GetTimerPeriodByCount
LPIT_DRV_GetCurrentTimerCount
LPIT_DRV_GetInterruptFlagTimerChannels
LPIT_DRV_ClearInterruptFlagTimerChannels



```
void LPIT_ISR(void)
{
    LPIT_DRV_ClearInterruptFlagTimerChannels(FSL_LPIT1,(1 << 0));


}
```

S32K144 Interrupts Lab: Toggle Green LED (PTD16)

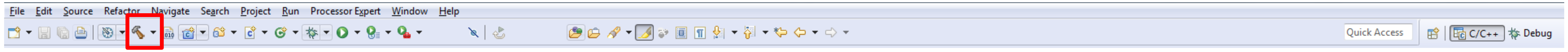
- Expand the **gpio_hal** component in the **Components** Window
- Drag and drop the **GPIO_HAL_TogglePins** function into LPIT_ISR
- Add the following parameters:
  - PTD
  - (1<<16)



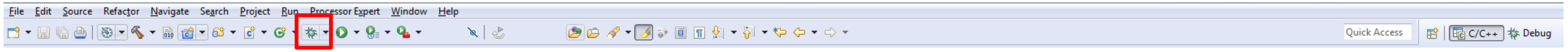
```
void LPIT_ISR(void)
{
    LPIT_DRV_ClearInterruptFlagTimerChannels(FSL_LPIT1,(1 << 0));
    GPIO_HAL_TogglePins(PTD,(1<<16));
}
```

S32K144 Interrupts Lab: Build and debug the lab

- Click on the build icon to make sure that there a no compiler errors.

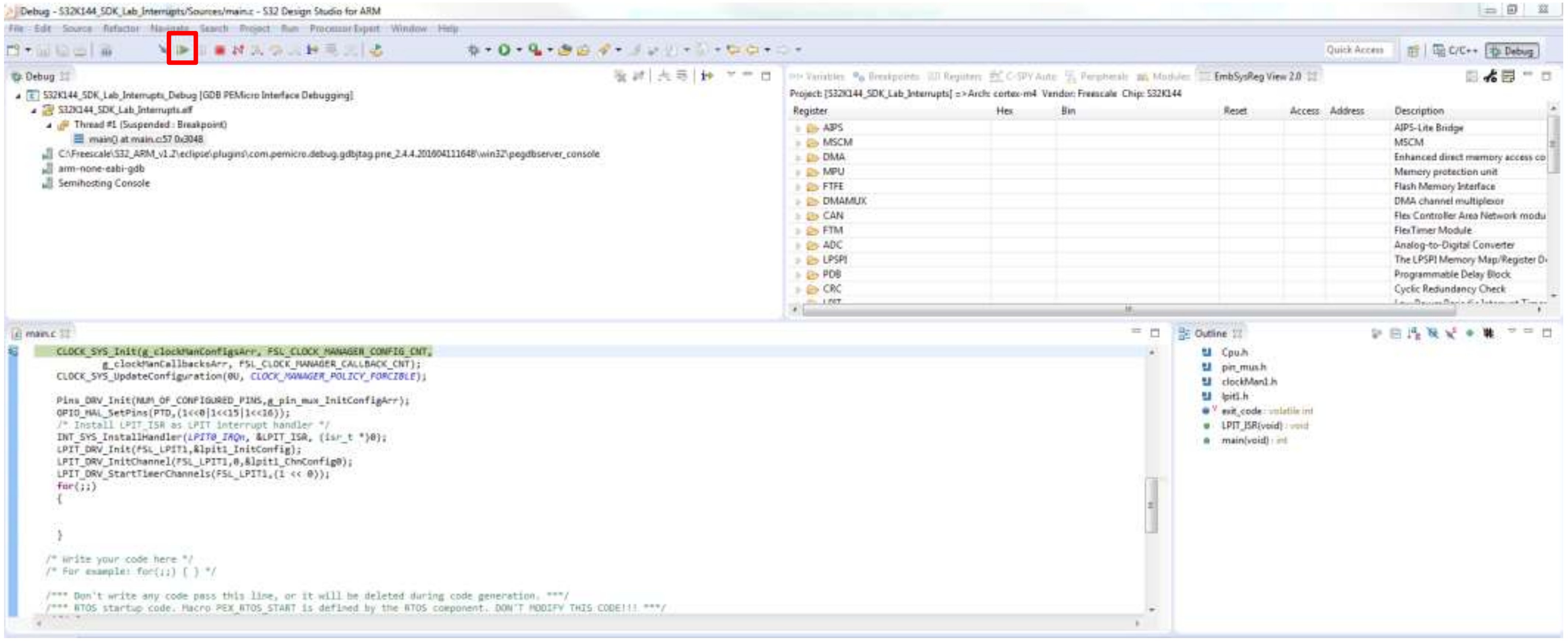

File Edit Source Refactor Navigate Search Project Run Processor Expert Window Help
Quick Access
C/C++
Debug

- Click the debug icon to start a new debug session


File Edit Source Refactor Navigate Search Project Run Processor Expert Window Help
Quick Access
C/C++
Debug

S32K144 Interrupts Lab: Build and debug the lab

- In the debug perspective click the run icon to start the project.
- Green LED should toggle every 1 sec.



Debug - S32K144_SDK_Lab_Interrupts/Sources/main.c - S32 Design Studio for ARM

```
S32K144_SDK_Lab_Interrupts_Debug [GDB PEMicro Interface Debugging]
  S32K144_SDK_Lab_Interrupts.elf
    Thread #1 (Suspended : Breakpoint)
      main() at main.c:57 0x3048
  C:\Freescale\S32_ARM_v1.2\eclipse\plugins\com.pemicro.debug.gdbjtag.pne_2.4.4.201604111648\win32\pegdbserver_console
  arm-none-eabi-gdb
  Semihosting Console
```

Project: [S32K144_SDK_Lab_Interrupts] =>Arch: cortex-m4 Vendor: Freescale Chip: S32K144

Register	Hex	Bin	Reset	Access	Address	Description
AIPS						AIPS-Lite Bridge
MSCM						MSCM
DMA						Enhanced direct memory access co
MPU						Memory protection unit
FTFE						Flash Memory Interface
DMAMUX						DMA channel multiplexor
CAN						Flex Controller Area Network modu
FTM						FlexTimer Module
ADC						Analog-to-Digital Converter
LPSPI						The LPSPI Memory Map/Register D
PDB						Programmable Delay Block
CRC						Cyclic Redundancy Check

main.c

```c
    CLOCK_SYS_Init(g_clockManConfigsArr, FSL_CLOCK_MANAGER_CONFIG_CNT,
            g_clockManCallbacksArr, FSL_CLOCK_MANAGER_CALLBACK_CNT);
    CLOCK_SYS_UpdateConfiguration(0U, CLOCK_MANAGER_POLICY_FORCIBLE);

    Pins_DRV_Init(NUM_OF_CONFIGURED_PINS,g_pin_mux_InitConfigArr);
    GPIO_HAL_SetPins(PTD,(1<<0|1<<15|1<<16));
    /* Install LPIT_ISR as LPIT interrupt handler */
    INT_SYS_InstallHandler(LPIT0_IRQn, &LPIT_ISR, (isr_t *)0);
    LPIT_DRV_Init(FSL_LPIT1,&lpit1_InitConfig);
    LPIT_DRV_InitChannel(FSL_LPIT1,0,&lpit1_ChnConfig0);
    LPIT_DRV_StartTimerChannels(FSL_LPIT1,(1 << 0));
    for(;;)
    {


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

  /*** Don't write any code pass this line, or it will be deleted during code generation. ***/
  /*** RTOS startup code. Macro PEX_RTOS_START is defined by the RTOS component. DON'T MODIFY THIS CODE!!! ***/
```

Outline

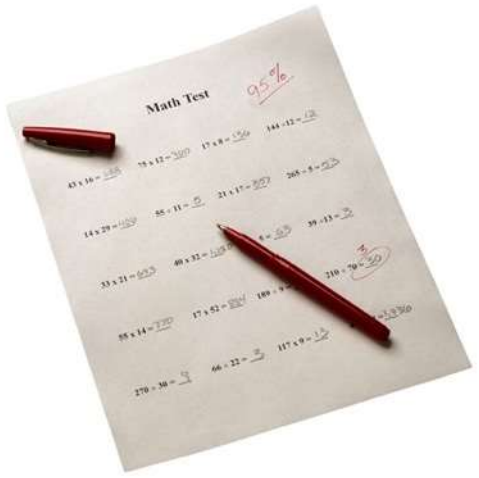
```
Cpu.h
pin_mux.h
clockMan1.h
lpit1.h
exit_code : volatile int
LPIT_ISR(void) : void
main(void) : int
```

S32K144 Interrupts Lab: Challenge

- Blink LEDs to generate White colored light every 100 ms.

S32K144 Interrupts Lab: Quiz

- Which module is responsible for MCU wakeup in stop modes?
- How many LPIT channels are available?



SECURE CONNECTIONS
FOR A SMARTER WORLD