HANDS-ON WORKSHOP: S32 SDK FOR S32K

VLAD LIONTE
EMBEDDED SW ENGINEER

NON-AUTOSAR SOFTWARE SOLUTIONS BASED ON S32 SDK

AMF-AUT-T2689 | JUNE 2017

SECURE CONNECTIONS
FOR A SMARTER WORLD

PUBLIC

AGENDA

- Introduction
  - S32 SDK
  - S32 Design Studio
- Hands-on
  - Blinking LED
  - Secured CAN Communication
- Q&A

01.

Introduction

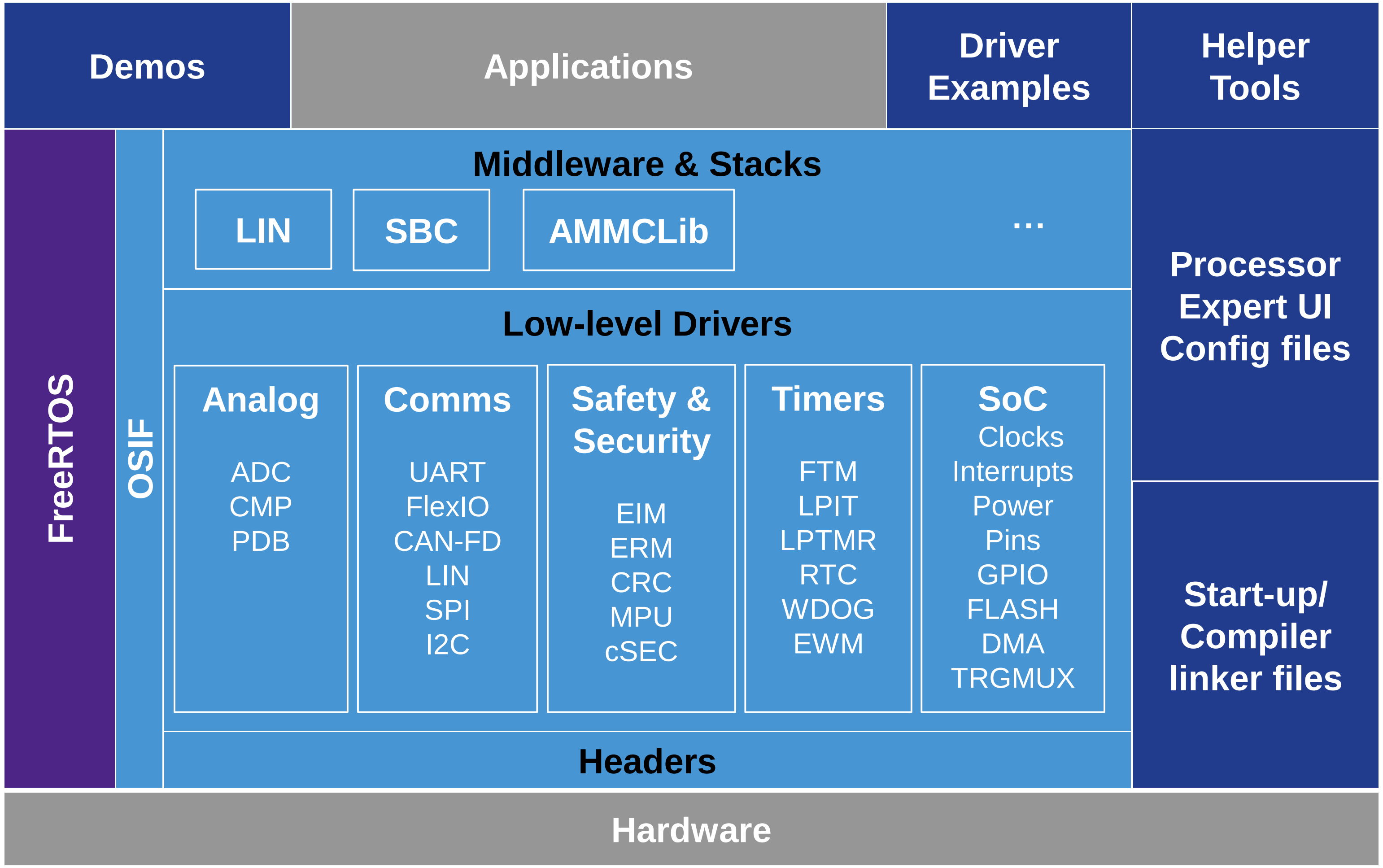
S32 Software Development Kit (SDK)

- Non-Autosar Software package
- **Automotive Grade:** SPICE/CMMI compliant, MISRA 2012
- Graphical-based configuration
- Compatible with Eclipse & other IDEs
- Supports all S32K MCU Families
- Supports multiple toolchains

S32 SDK – Architecture

SW Quality Class

- Class B
- Class C
- Class D



Demos

Applications

Driver Examples

Helper Tools

FreeRTOS

OSIF

Middleware & Stacks

LIN

SBC

AMMCLib

…

Low-level Drivers

Analog
ADC
CMP
PDB

Comms
UART
FlexIO
CAN-FD
LIN
SPI
I2C

Safety & Security
EIM
ERM
CRC
MPU
cSEC

Timers
FTM
LPIT
LPTMR
RTC
WDOG
EWM

SoC
Clocks
Interrupts
Power
Pins
GPIO
FLASH
DMA
TRGMUX

Headers

Processor Expert UI Config files

Start-up/ Compiler linker files

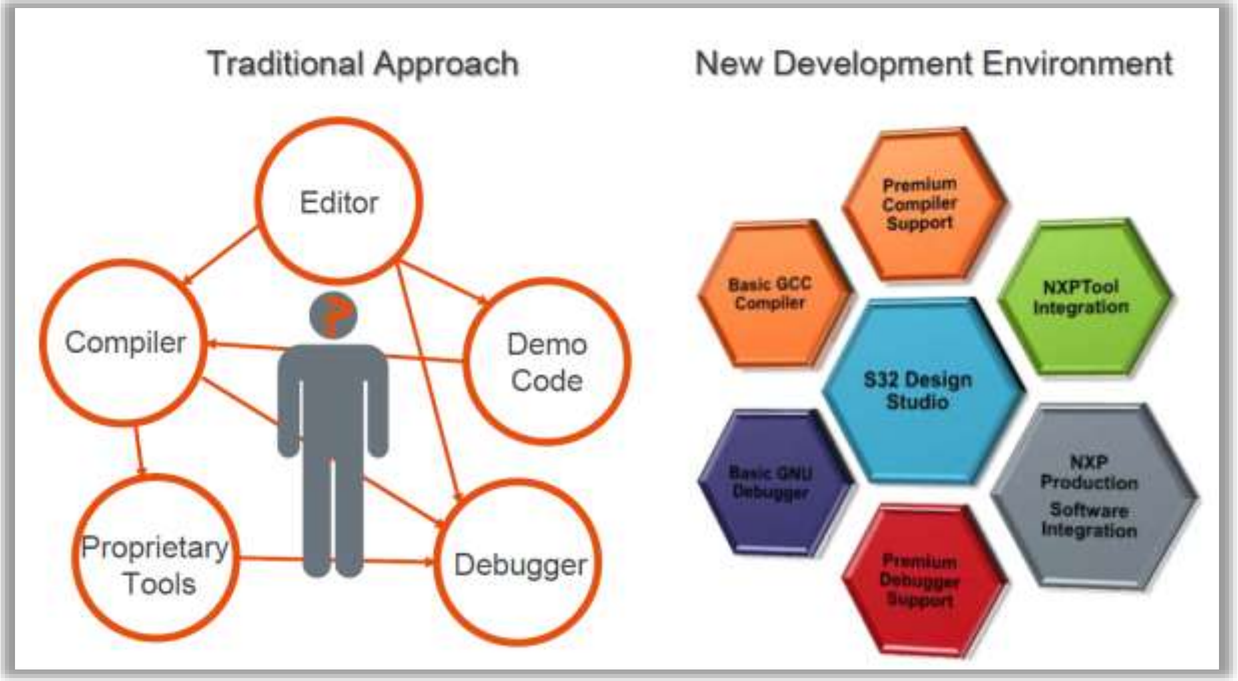
Hardware

Features

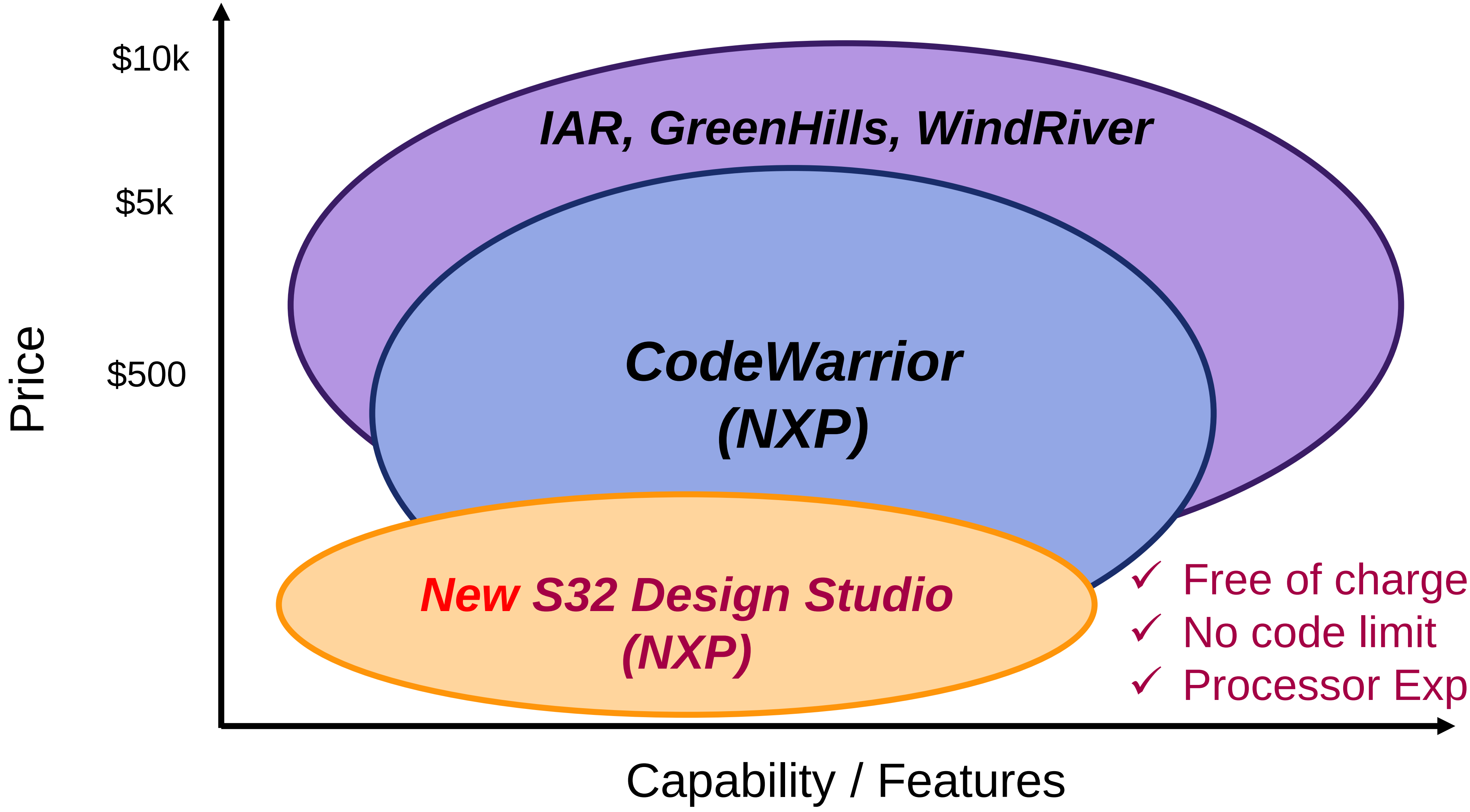
- Integrated **Non-Autosar SW Production-grade** software
- Graphical-based Configuration
- Layered Software Architecture
- Documented Source Code and Examples
- Integrated with S32 Design Studio and other IDEs
- Featuring various Middleware
- FreeRTOS integration
- Multiple toolchains supported
- Several examples and demos

NXP S32 Design Studio IDE www.nxp.com/S32DS

- Free of charge
- Unlimited code size
- Eclipse based environment
- GNU compiler & debugger integrated
- S32 SDK integrated (graphical configuration)
- Processor Expert integrated (automatic code generator)
- Can use with 3rd party compliers & debuggers (IAR) via Connection Utility
- Supports S32K and Power Architecture (MPC) products
- Not a replacement for NXP’s CodeWarrior IDE
- Not intended to compete with premium 3rd party IDEs

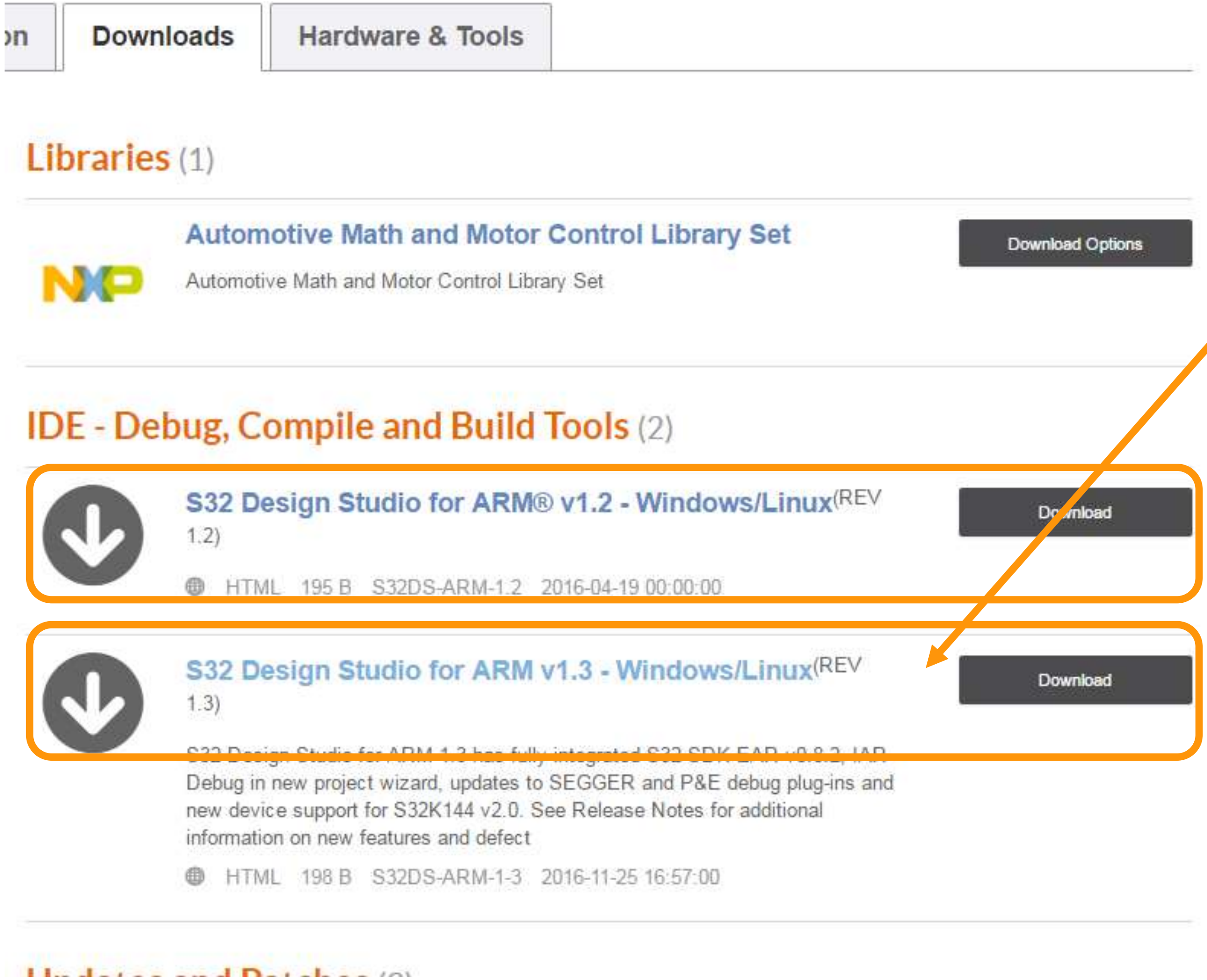


NXP & 3rd Party IDEs – Performance/Price Map



- ✓ Free of charge
- ✓ No code limit
- ✓ Processor Expert Integrated

S32 Design Studio @ www.nxp.com/S32DS



Libraries (1)

Automotive Math and Motor Control Library Set

Automotive Math and Motor Control Library Set

IDE - Debug, Compile and Build Tools (2)

S32 Design Studio for ARM® v1.2 - Windows/Linux(REV 1.2)

HTML 195 B S32DS-ARM-1.2 2016-04-19 00:00:00

S32 Design Studio for ARM v1.3 - Windows/Linux(REV 1.3)

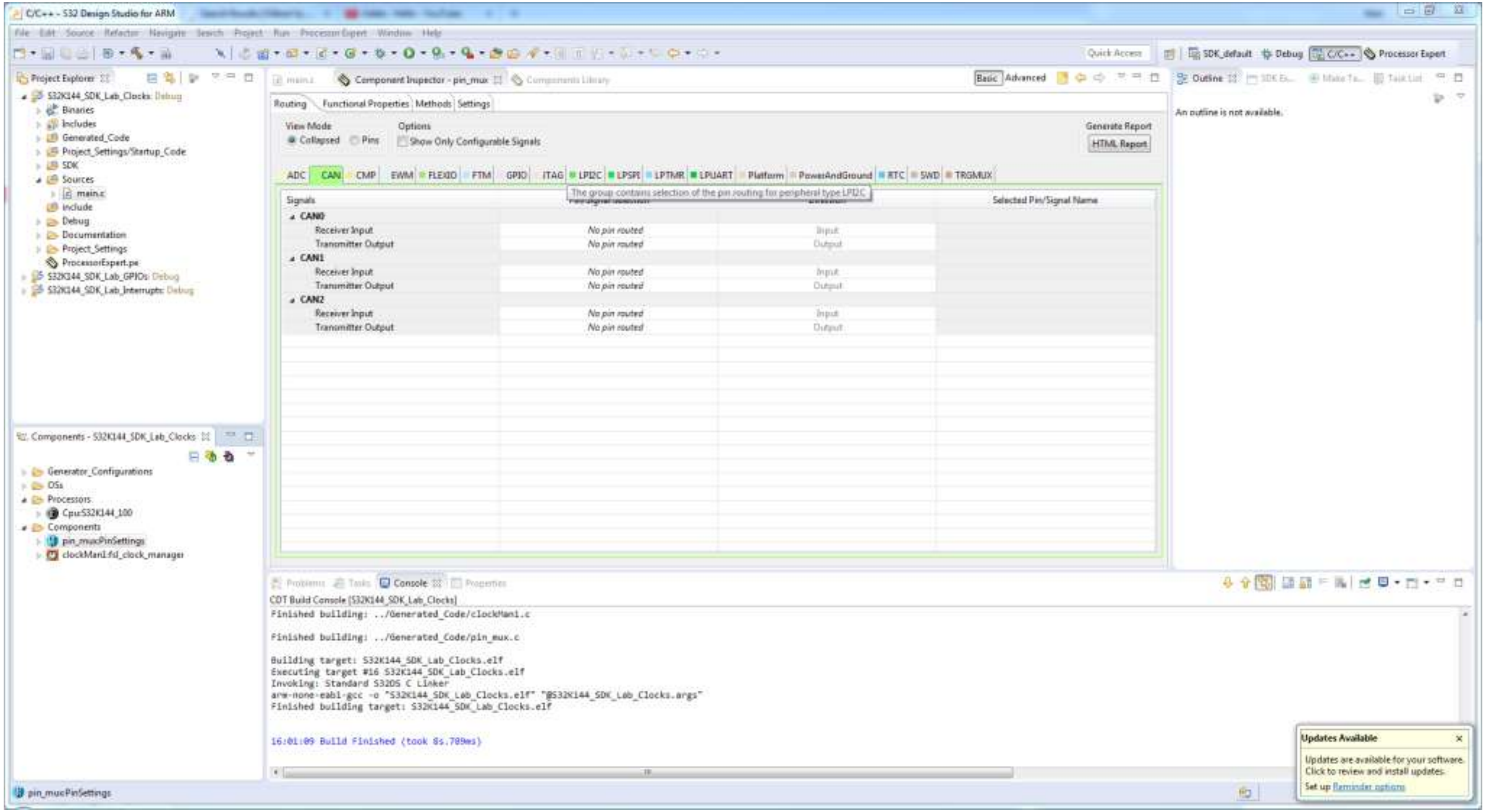
S32 Design Studio for ARM 1.3 has fully integrated S32 SDK EAR v0.8.2, IAR Debug in new project wizard, updates to SEGGER and P&E debug plug-ins and new device support for S32K144 v2.0. See Release Notes for additional information on new features and defect

HTML 198 B S32DS-ARM-1-3 2016-11-25 16:57:00

Updates and Patches (2)

- **S32DS_v1.3 (includes SDK v0.9.0)**
  - Supports S32K144 MCU, 0N47T & 0N57U mask sets
- **S32DS_v1.2 (includes SDK_v0.8.2)**
  - Supports S32K144 MCU, 0N77P mask set only (early silicon for alpha customers only – not available to mass market customers)

S32 Design Studio – graphical configuration environment



View Mode
● Collapsed ○ Pins

Options
□ Show Only Configurable Signals

The group contains selection of the pin routing for peripheral type LPI2C

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	

An outline is not available.

CDT Build Console [S32K144_SDK_Lab_Clocks]

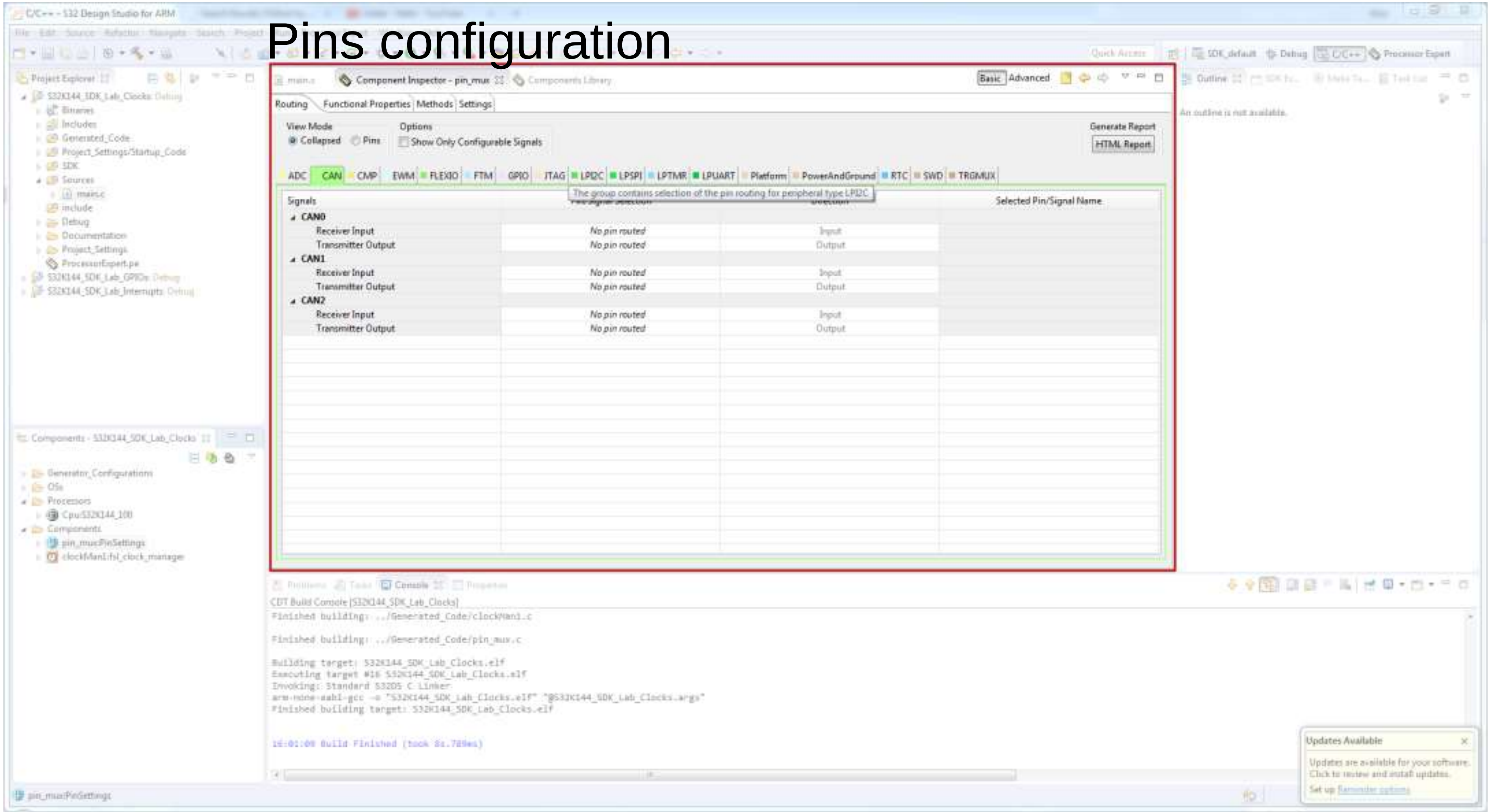
```
Finished building: ../Generated_Code/clockMan1.c

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

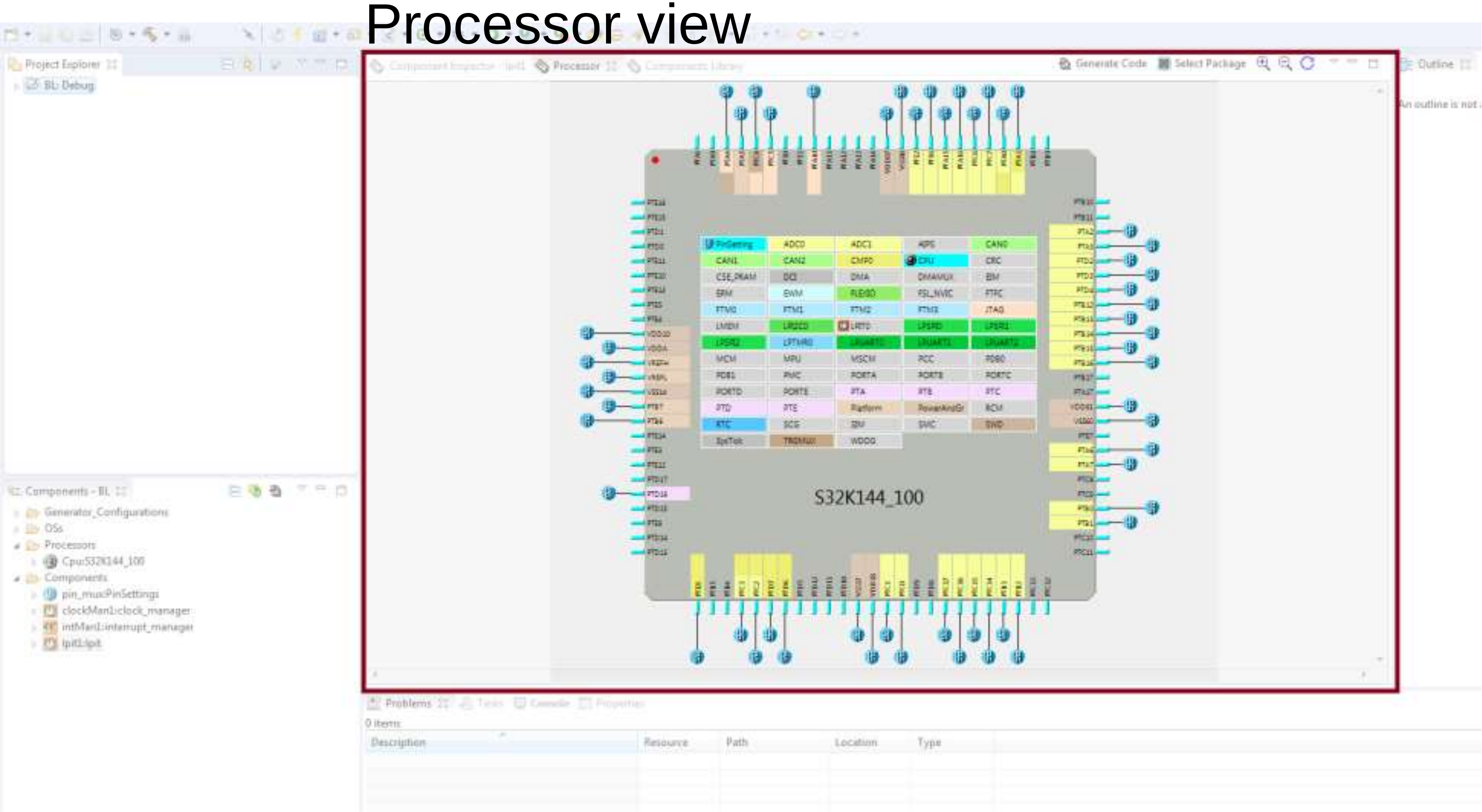
Signals	Pin/Signal Selection	Direction	Selected Pin/Signal Name
CAN0			
Receiver Input	*No pin routed*	Input	
Transmitter Output	*No pin routed*	Output	
CAN1			
Receiver Input	*No pin routed*	Input	
Transmitter Output	*No pin routed*	Output	
CAN2			
Receiver Input	*No pin routed*	Input	
Transmitter Output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

Processor view



Project Explorer

BL: Debug

Component Inspector - lpit1　Processor　Components Library

Generate Code　Select Package

Outline

An outline is not

PinSettings	ADC0	ADC1	AIPS	CAN0
CAN1	CAN2	CMP0	CPU	CRC
CSE_PRAM	DCI	DMA	DMAMUX	EIM
ERM	EWM	FLEXIO	FSL_NVIC	FTFC
FTM0	FTM1	FTM2	FTM3	JTAG
LMEM	LPI2C0	LPIT0	LPSPI0	LPSPI1
LPSPI2	LPTMR0	LPUART0	LPUART1	LPUART2
MCM	MPU	MSCM	PCC	PDB0
PDB1	PMC	PORTA	PORTB	PORTC
PORTD	PORTE	PTA	PTB	PTC
PTD	PTE	Platform	PowerAndGr	RCM
RTC	SCG	SIM	SMC	SWD
SysTick	TRGMUX	WDOG		

S32K144_100

Components - BL

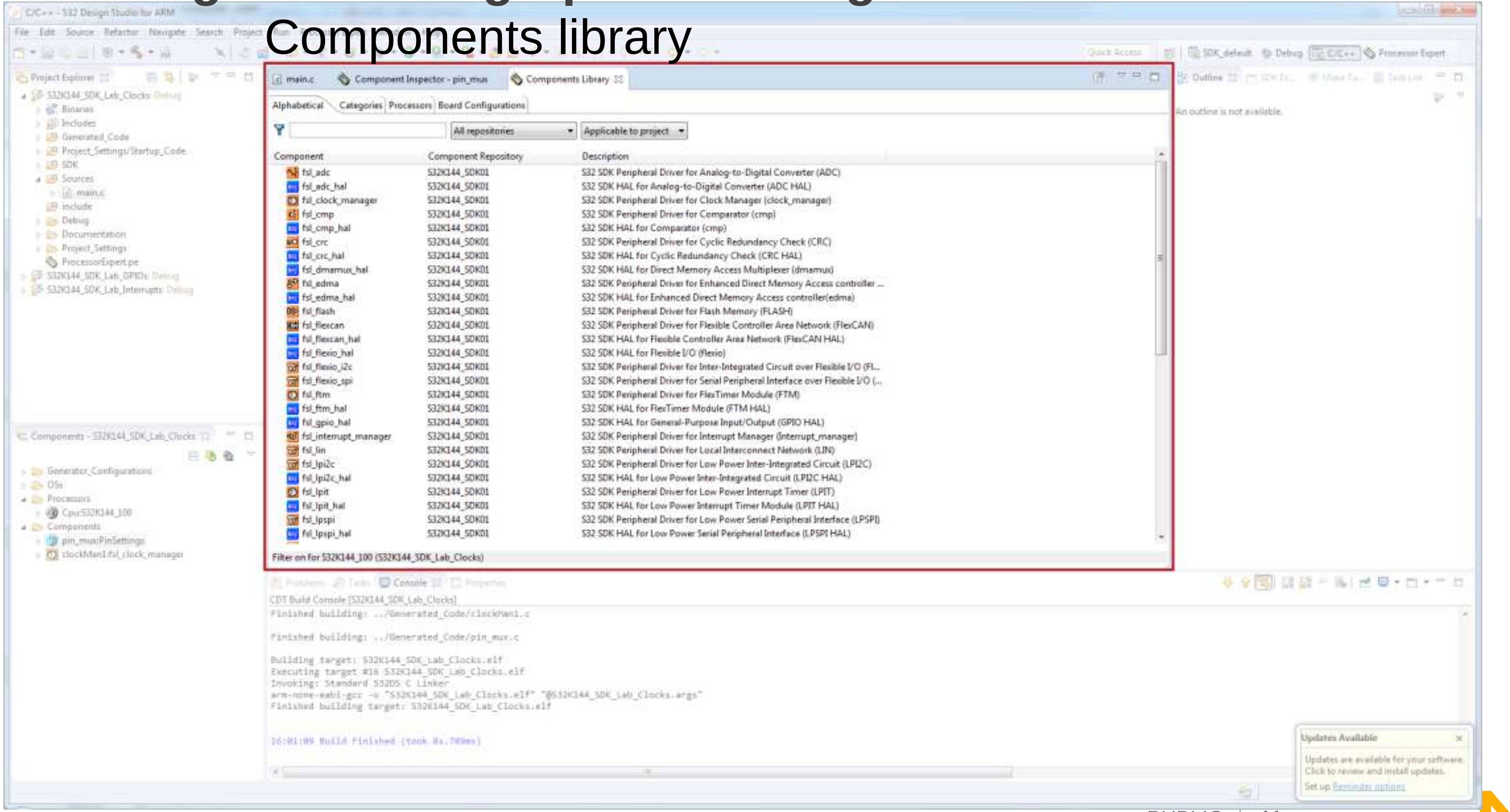
- Generator_Configurations
- OSs
- Processors
  - Cpu:S32K144_100
- Components
  - pin_mux:PinSettings
  - clockMan1:clock_manager
  - intMan1:interrupt_manager
  - lpit1:lpit

Problems　Tasks　Console　Properties

0 items

Description	Resource	Path	Location	Type

S32 Design Studio – graphical configuration environment



Components library

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]

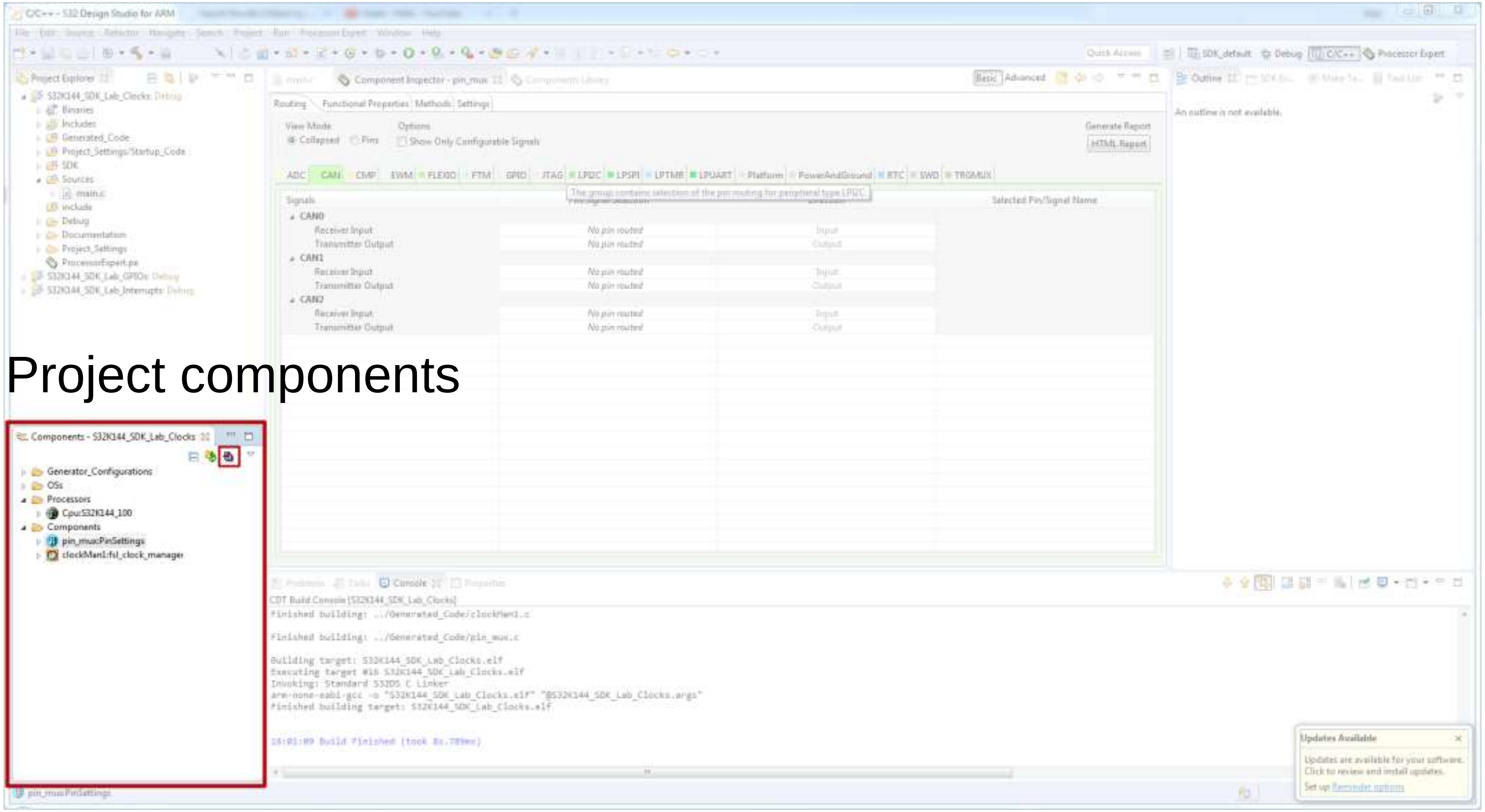
```
Finished building: ../Generated_Code/clockMan1.c

Finished building: ../Generated_Code/pin_mux.c

Building target: S32K144_SDK_Lab_Clocks.elf
Executing target #18 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



Project components

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
Component Inspector - pin_mux
Components Library
Basic
Advanced
Outline
An outline is not available.
Routing
Functional Properties
Methods
Settings
View Mode
Options
Generate Report
HTML Report
ADC CAN CMP EWM FLEXIO FTM GPIO JTAG LPI2C LPSPI LPTMR LPUART Platform PowerAndGround RTC SWD TRGMUX
The group contains selection of the pin routing for peripheral type LPI2C
Problems
Tasks
Console
Properties
Updates Available
pin_mux:PinSettings

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

View Mode: ● Collapsed ○ Pins

Options: □ Show Only Configurable Signals

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	

Components - S32K144_SDK_Lab_Clocks
- Generator_Configurations
- OSs
- Processors
  - Cpu:S32K144_100
- Components
  - pin_mux:PinSettings
  - clockMan1:fsl_clock_manager

CDT Build Console [S32K144_SDK_Lab_Clocks]

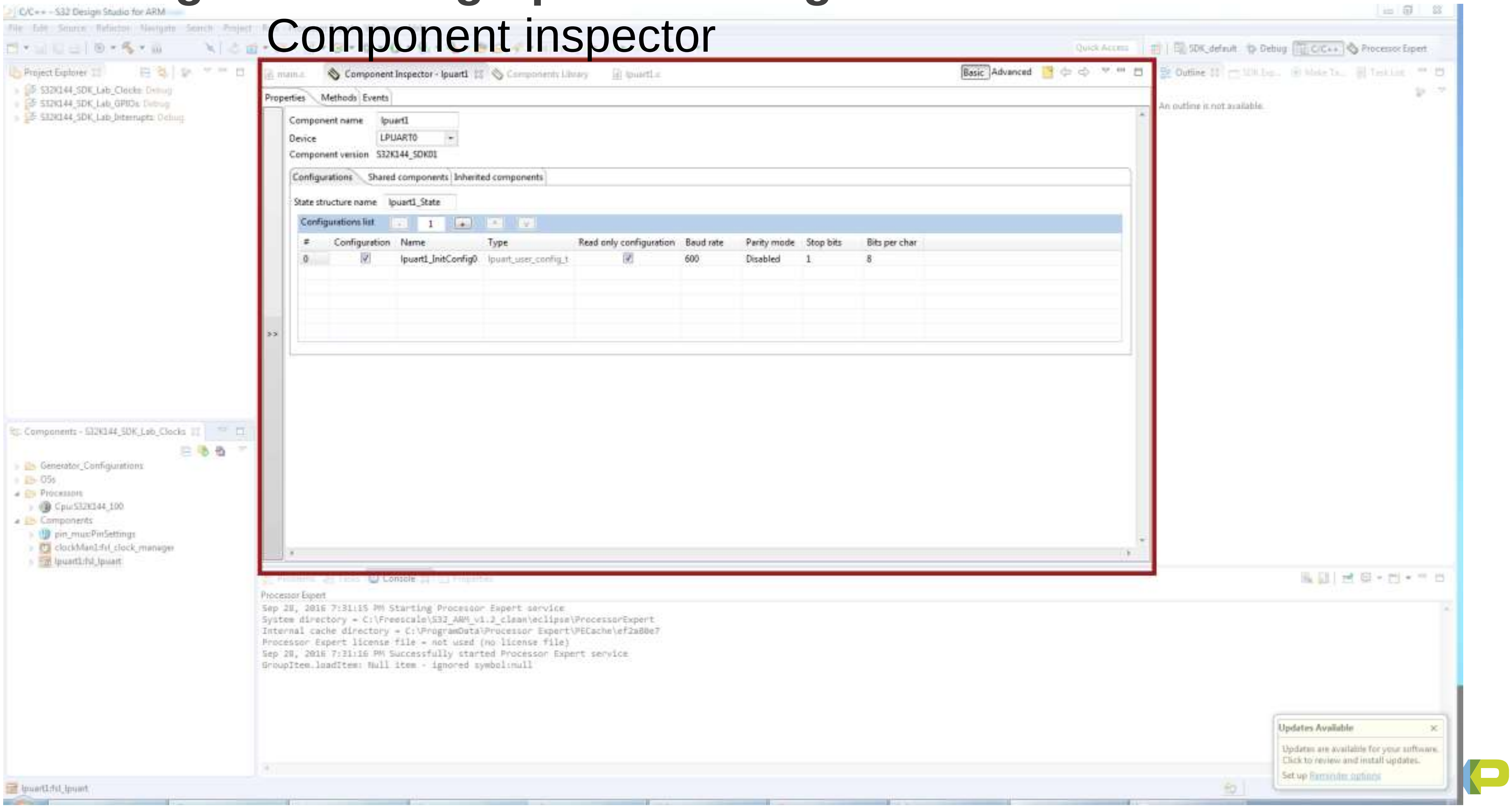
```
Finished building: ../Generated_Code/clockMan1.c

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

Set up Reminder options

S32 Design Studio – graphical configuration environment

Component inspector


C/C++ - S32 Design Studio for ARM
File Edit Source Refactor Navigate Search Project
Quick Access
SDK_default
Debug
C/C++
Processor Expert
Project Explorer
S32K144_SDK_Lab_Clocks: Debug
S32K144_SDK_Lab_GPIOs: Debug
S32K144_SDK_Lab_Interrupts: Debug
main.c
Component Inspector - lpuart1
Components Library
lpuart1.c
Basic
Advanced
Outline
An outline is not available.
Components - S32K144_SDK_Lab_Clocks
Generator_Configurations
OSs
Processors
Cpu:S32K144_100
Components
pin_mux:PinSettings
clockMan1:fsl_clock_manager
lpuart1:fsl_lpuart
Console
Updates Available
Updates are available for your software. Click to review and install updates.
Set up Reminder options
lpuart1:fsl_lpuart

Properties | Methods | Events

Component name: lpuart1

Device: LPUART0

Component version: S32K144_SDK01

Configurations | Shared components | Inherited components

State structure name: lpuart1_State

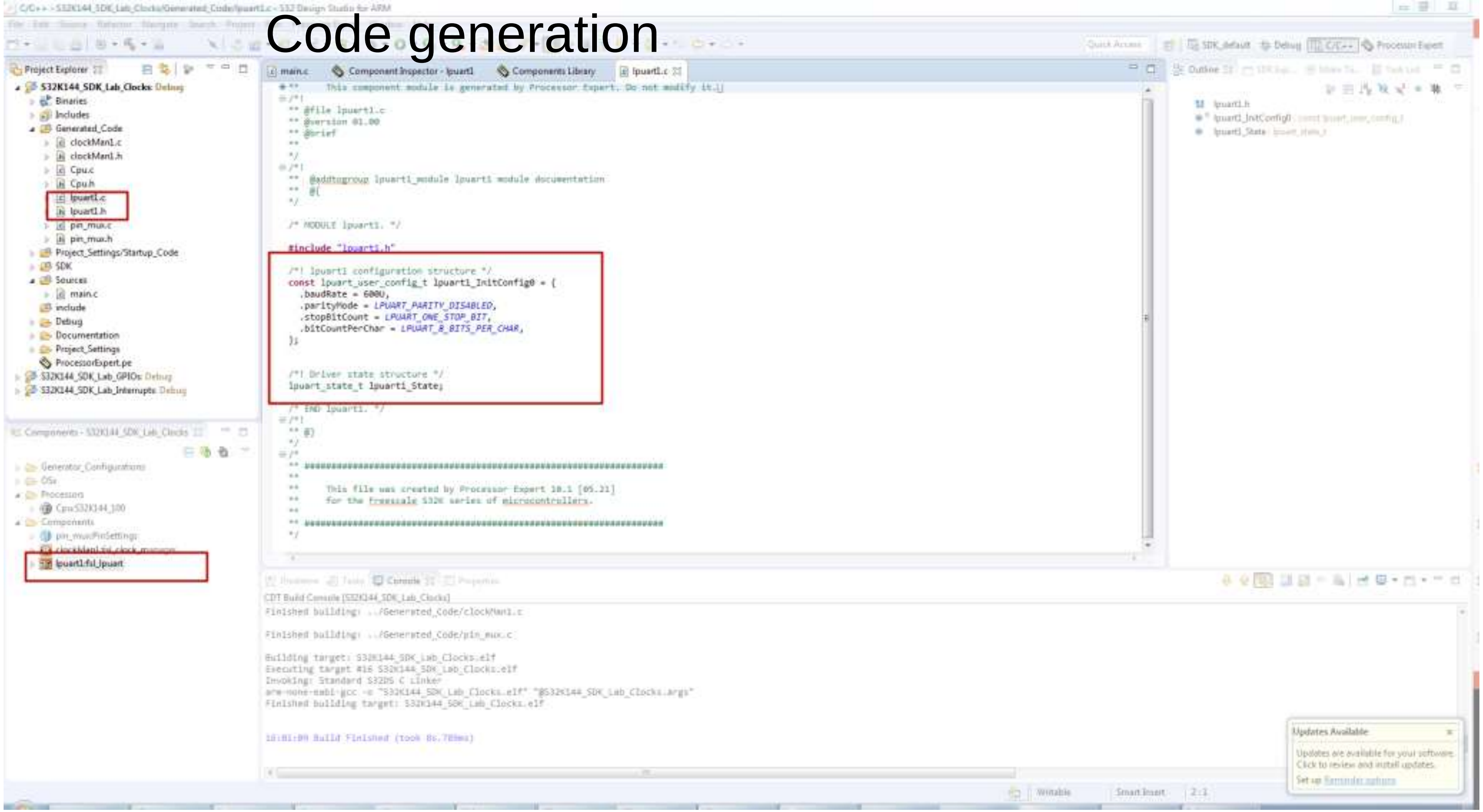
Configurations list: 1

#	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

Processor Expert

```
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\ef2a80e7
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



Code generation

```
**     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.
**
** ###################################################################
*/
```

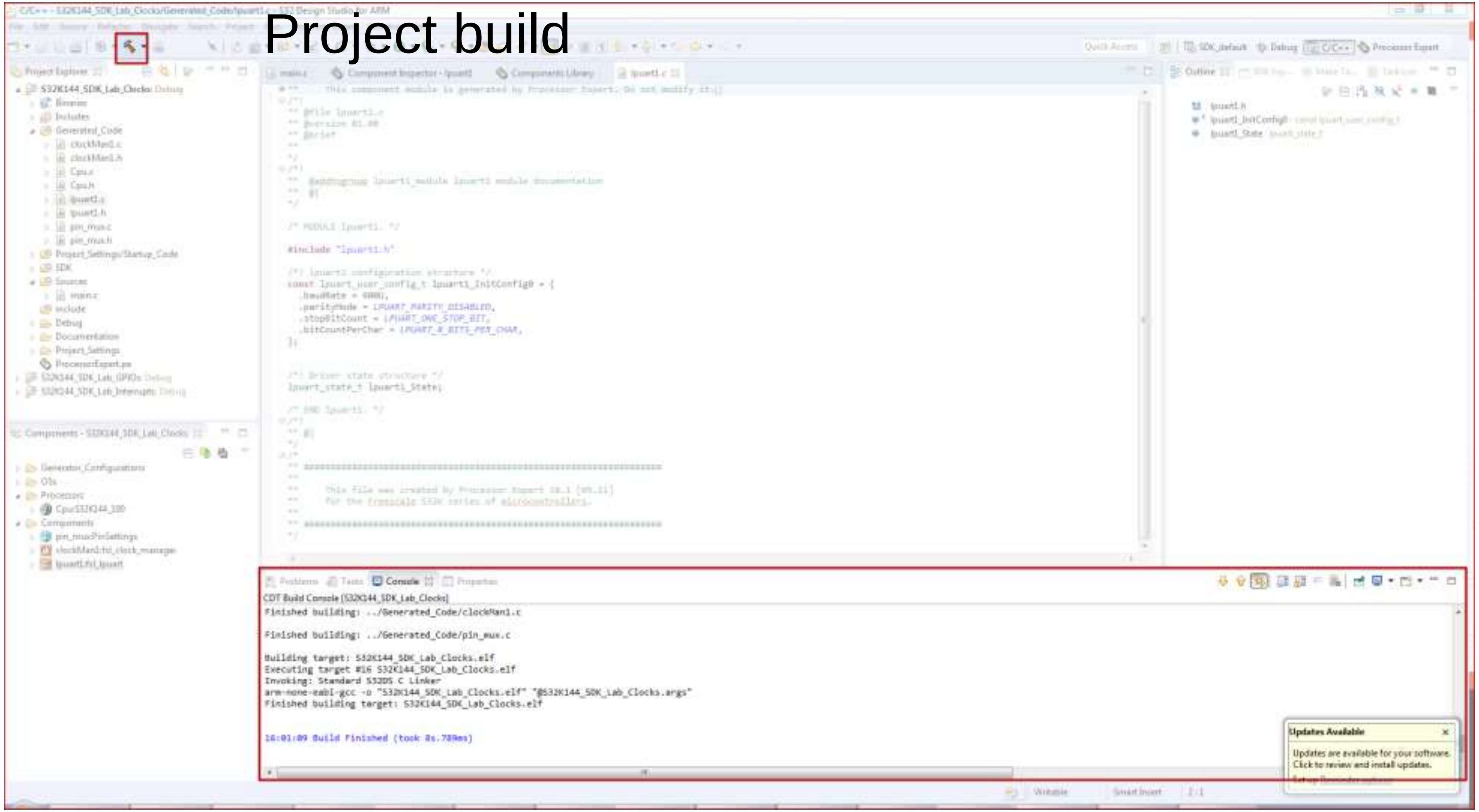
```
CDT Build Console [S32K144_SDK_Lab_Clocks]
Finished building: ../Generated_Code/clockMan1.c

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


10:01:00 Build Finished (took 8s.780ms)
```

Updates Available

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

Set up Reminder options

S32 Design Studio – graphical configuration environment



```
**     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 = 9600U,
  .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


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

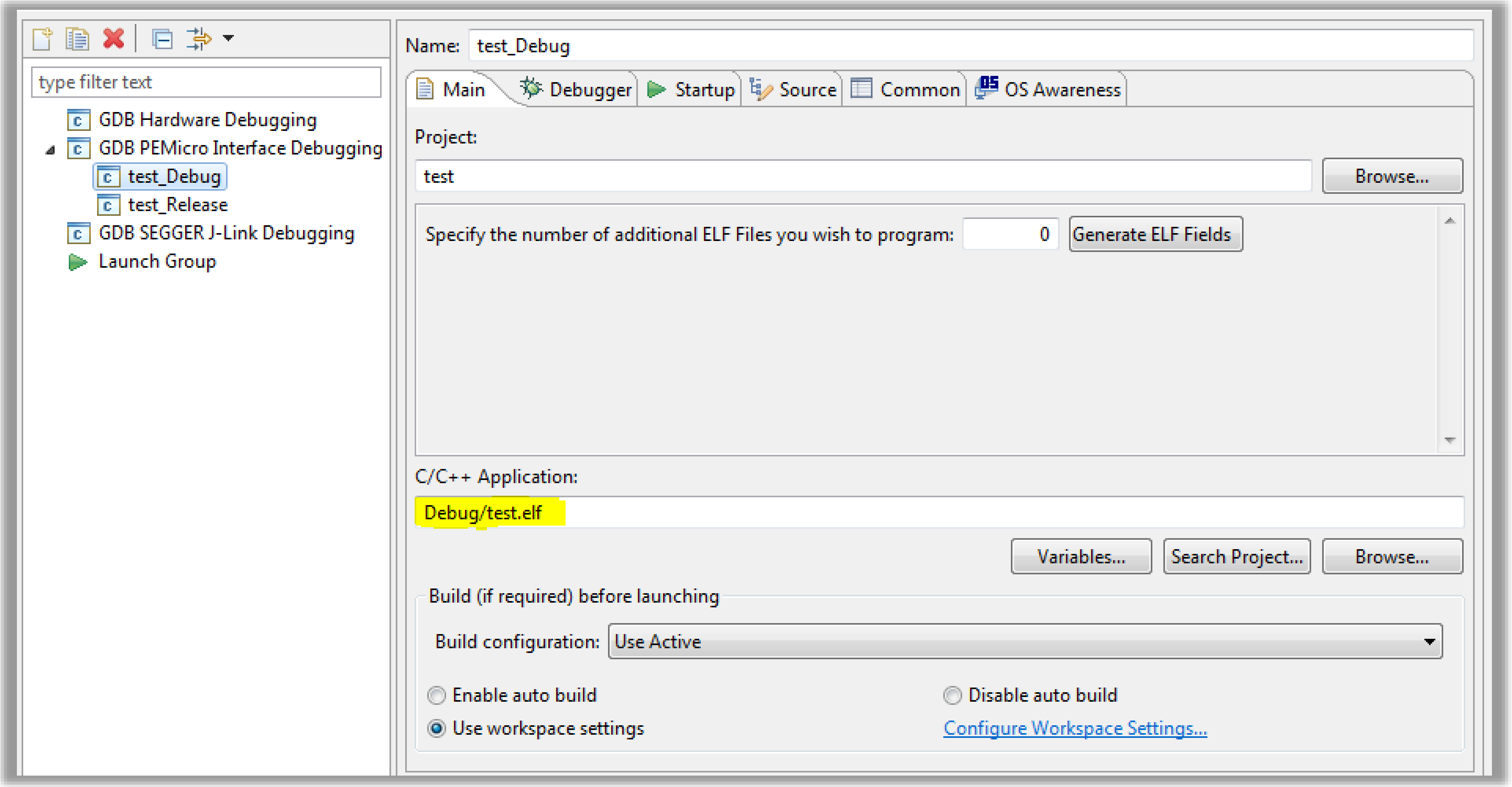
Updates Available

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

Set up Reminder options

S32 Design Studio – deploying the application

Target debug



Name: test_Debug

Project: test

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

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

Build (if required) before launching

Build configuration: Use Active

○ Enable auto build
○ Disable auto build
● Use workspace settings

02.

Hands-on – Blinking LED

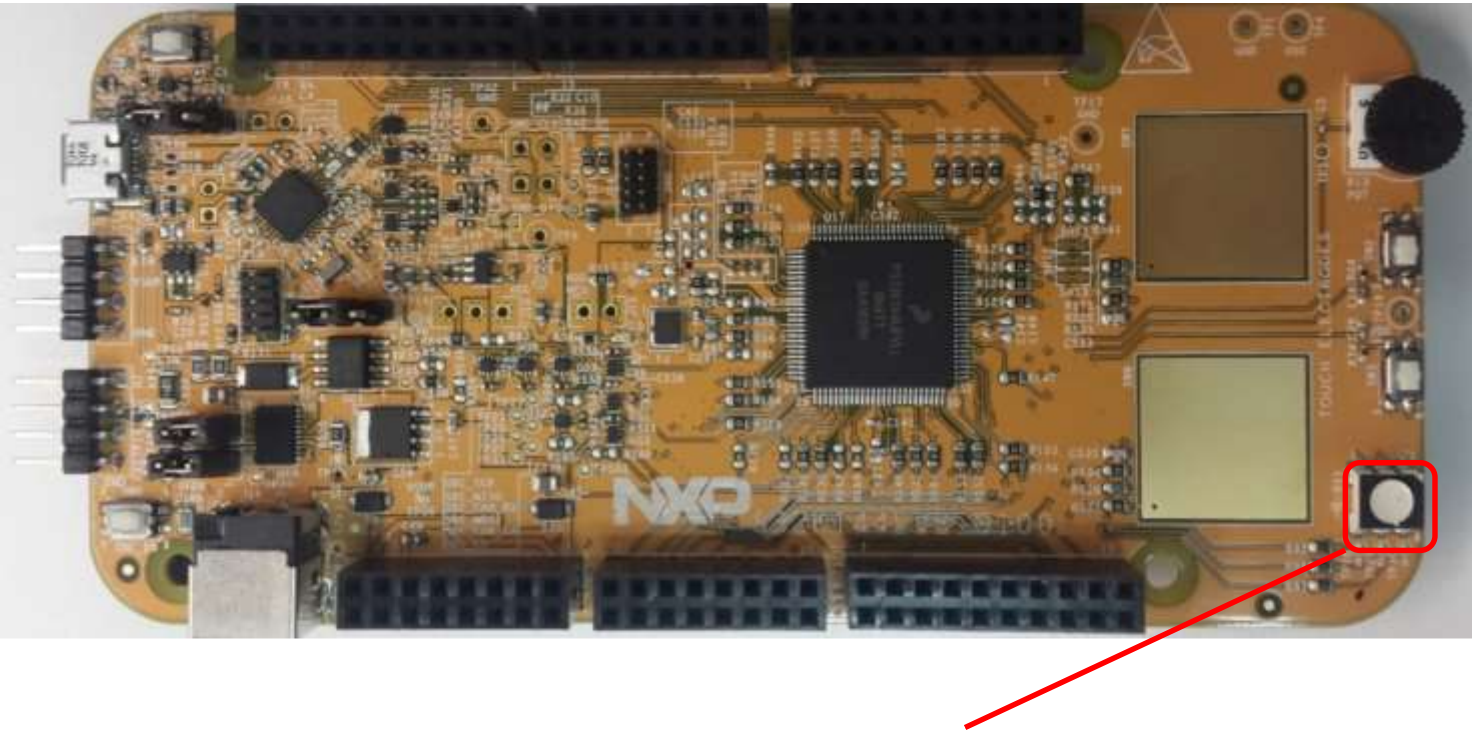
S32K144 Blinking LED: Objective

- In this lab you will learn:
  - About the GPIOs structure in S32K144
  - How interrupts works on S32K144
  - How to create a new SDK project with S32DS.
  - How to set a pin as output/input with SDK
  - How the use the LPIT peripheral
  - Set up an interrupt in S32K144 using SDK
  - Blink an LED every 0.5 sec using the LPIT interrupt

S32K144 Blinking LED: Resources to be used

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

LED	S32K144 PIN
BLUE	PTD0
RED	PTD15
GREEN	PTD16



RGB LED

S32K144 Blinking LED: Theory

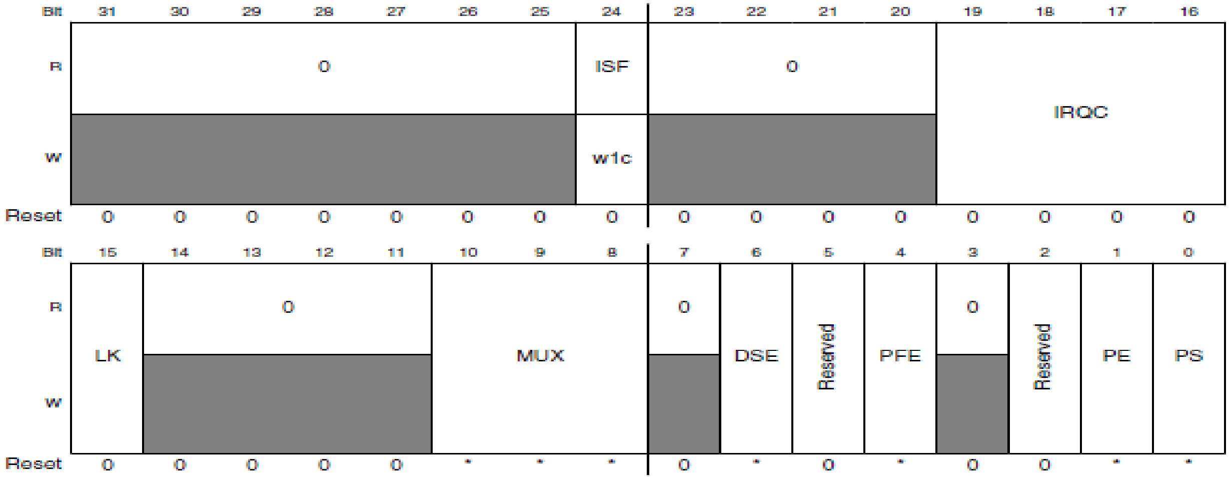
- There are up to 89 GPIOs in the S32K144
  - 5 PORTs (PTA, PTB, PTC, PTD, PTE)
- 8 high current pins (up to 20 mA each):
  - PTD1, PTD0, PTD16, PTD15, PTB5, PTB4,PTE1, and PTE0
- Each I/O is interrupt capable
- Each I/O is DMA capable
- Support for edge or level sensitive
- Each can wake up MCU from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

- Each I/O is multiplexed with different functionalities
- I/O functionality is selected with PORTx 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

Pin Control Register n (PORT_PCR*n*)



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 Blinking LED: Theory

- 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 Blinking LED: Theory

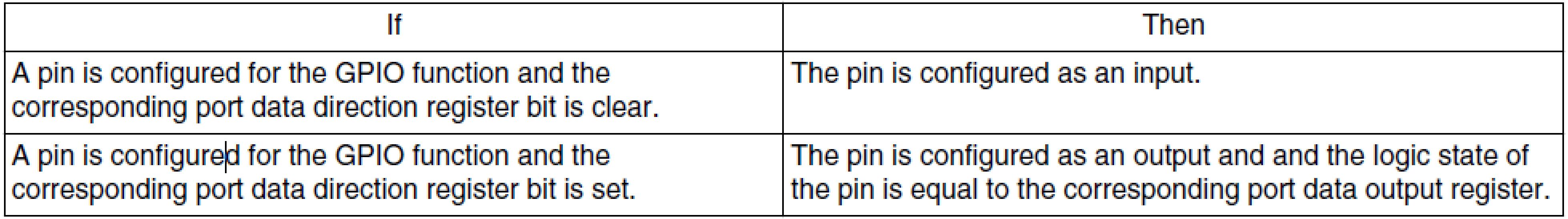
GPIO Direction selected with PDDR register.

GPIO INPUT

- Logic state available in PDIR register

GPIO OUTPUT

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

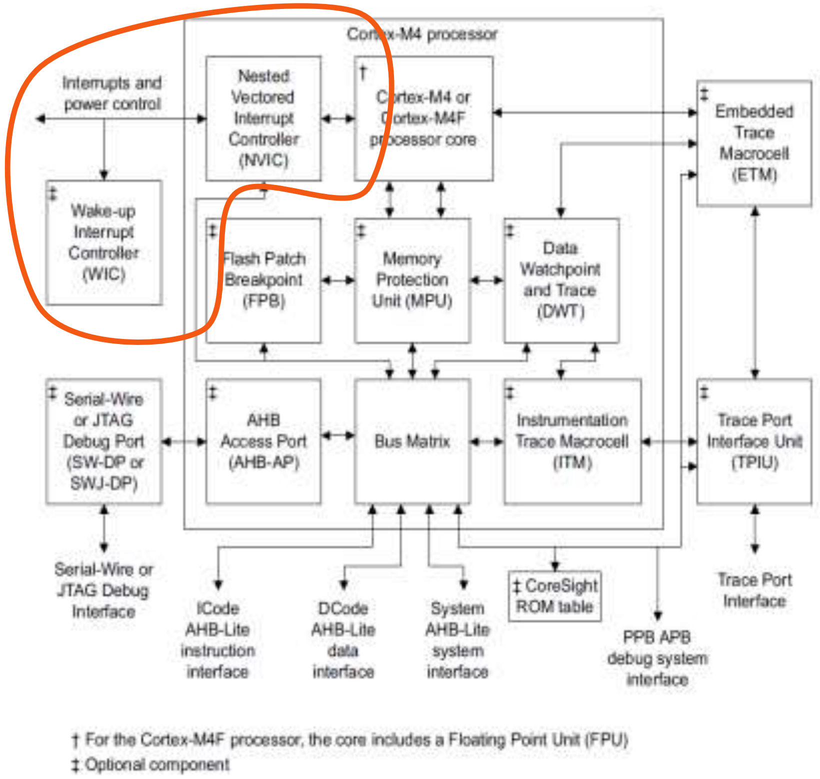


If	Then
A pin is configured for the GPIO function and the corresponding port data direction register bit is clear.	The pin is configured as an input.
A pin is configured for the GPIO function and the corresponding port data direction register bit is set.	The pin is configured as an output and and the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

S32K144 Blinking LED: Theory

NVIC (Nested Vector Interrupt Controller)

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



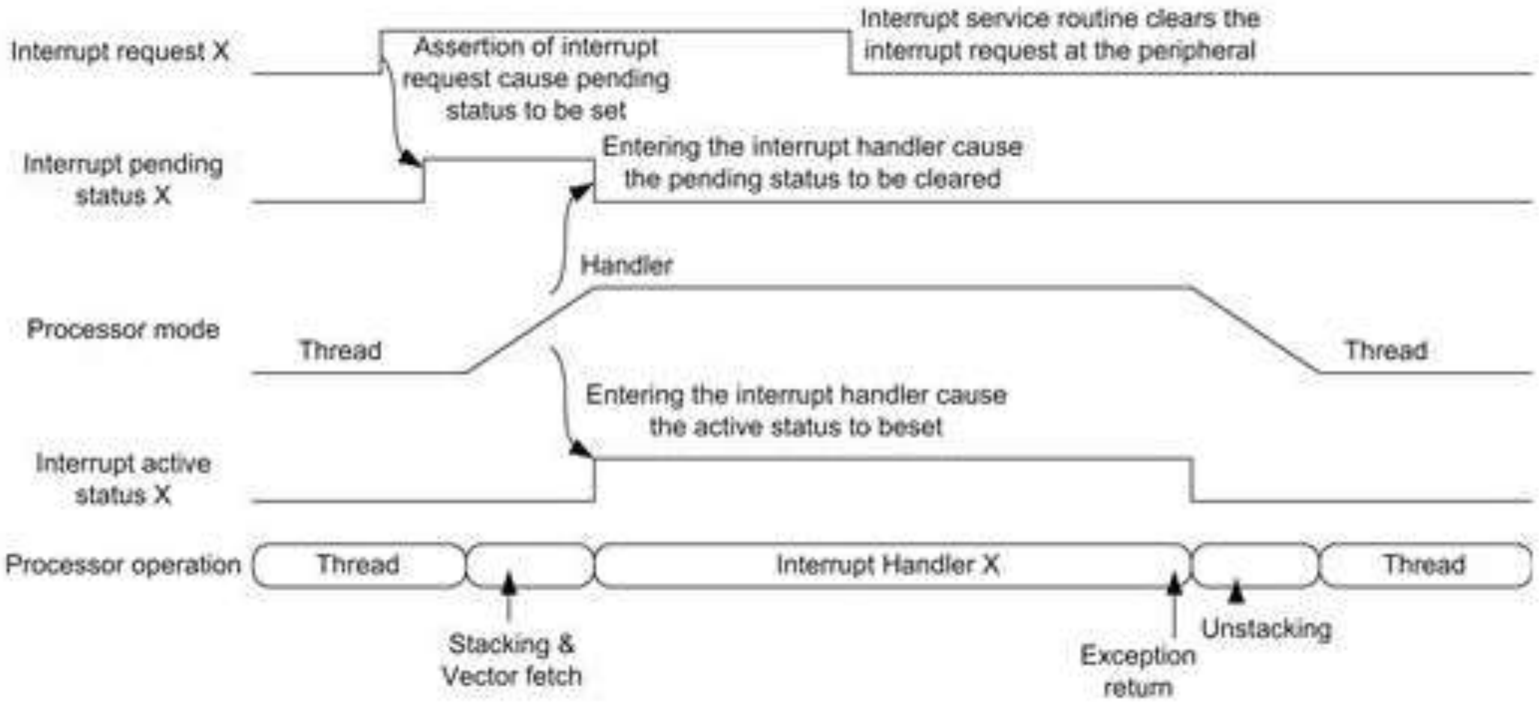
Cortex-M4 processor
Interrupts and power control
Nested Vectored Interrupt Controller (NVIC)
† Cortex-M4 or Cortex-M4F processor core
‡ Embedded Trace Macrocell (ETM)
‡ Wake-up Interrupt Controller (WIC)
‡ Flash Patch Breakpoint (FPB)
‡ Memory Protection Unit (MPU)
‡ Data Watchpoint and Trace (DWT)
‡ 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 Blinking LED: Theory

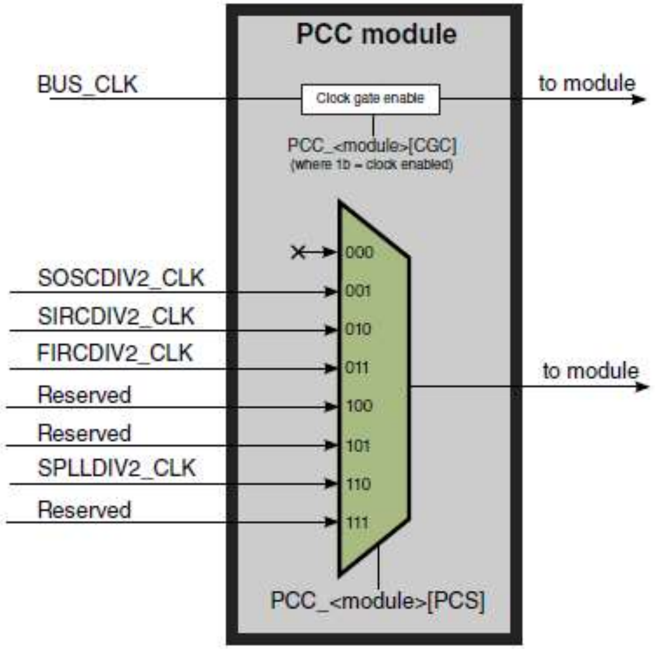
What happens when an interrupt occurs in an ARM Cortex M4?


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 Blinking LED: Theory

LPIT (Low power interrupt timer)

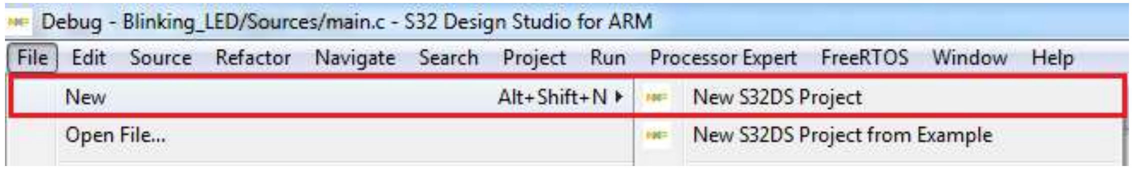
- 4 channels
- Individual or chained channel operation
- 32 bit counter per channel
- 4 operation modes:
  - 32-bit Periodic Counter
  - Dual 16-bit Periodic Counter
  - 32-bit Trigger Accumulator
  - 32-bit Trigger Input Capture

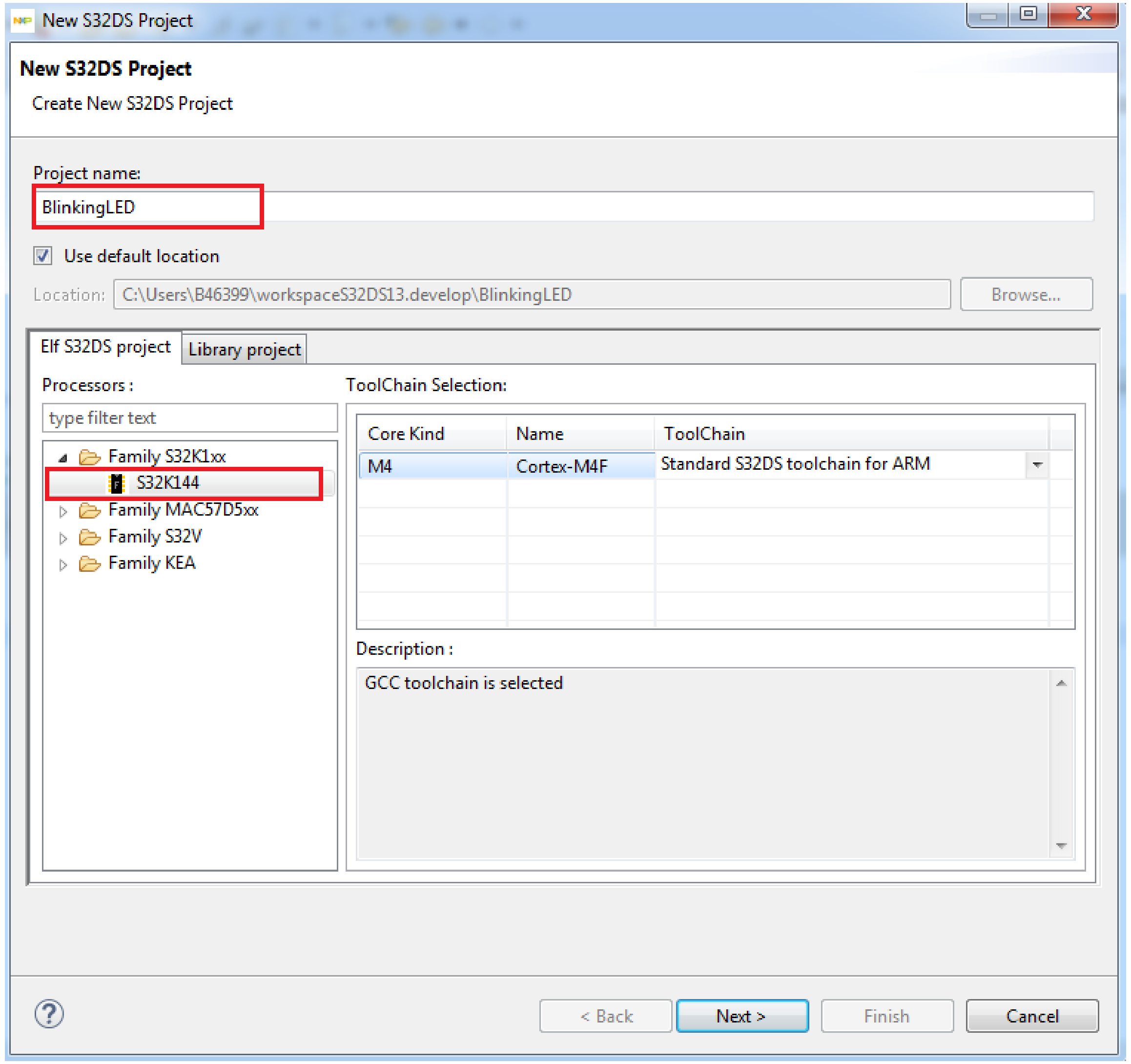


Module	VLPR	VLPW	Stop	VLPS
LPIT	Full functionality	Full functionality	Async operation	Async operation

S32K144 Blinking LED: Create New Project

- Create a new S32DS Project





New S32DS Project

Create New S32DS Project

Project name: BlinkingLED

☑ Use default location

Location: C:\Users\B46399\workspaceS32DS13.develop\BlinkingLED

Processors :

type filter text

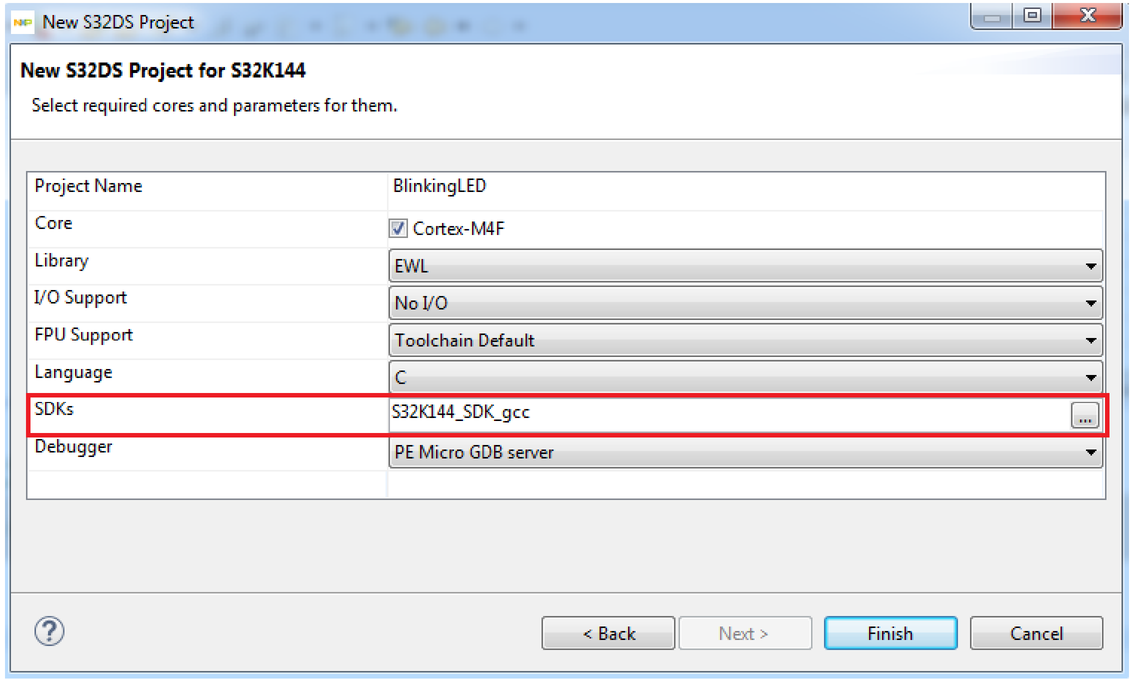
- Family S32K1xx
  - S32K144
- Family MAC57D5xx
- Family S32V
- Family KEA

ToolChain Selection:

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

Description :

GCC toolchain is selected



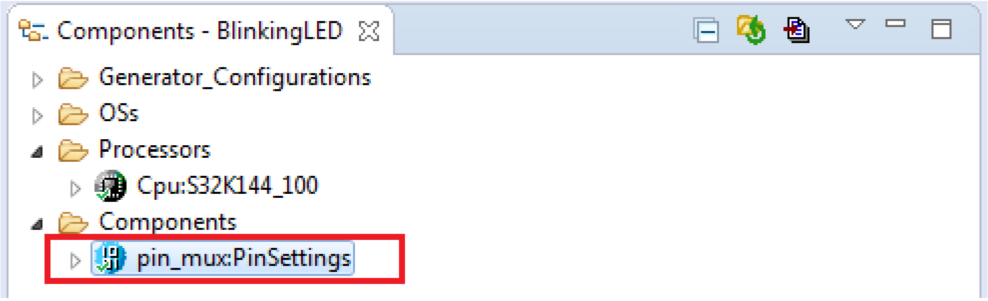
New S32DS Project for S32K144

Select required cores and parameters for them.

Project Name	BlinkingLED
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 Blinking LED: Configuring pins

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

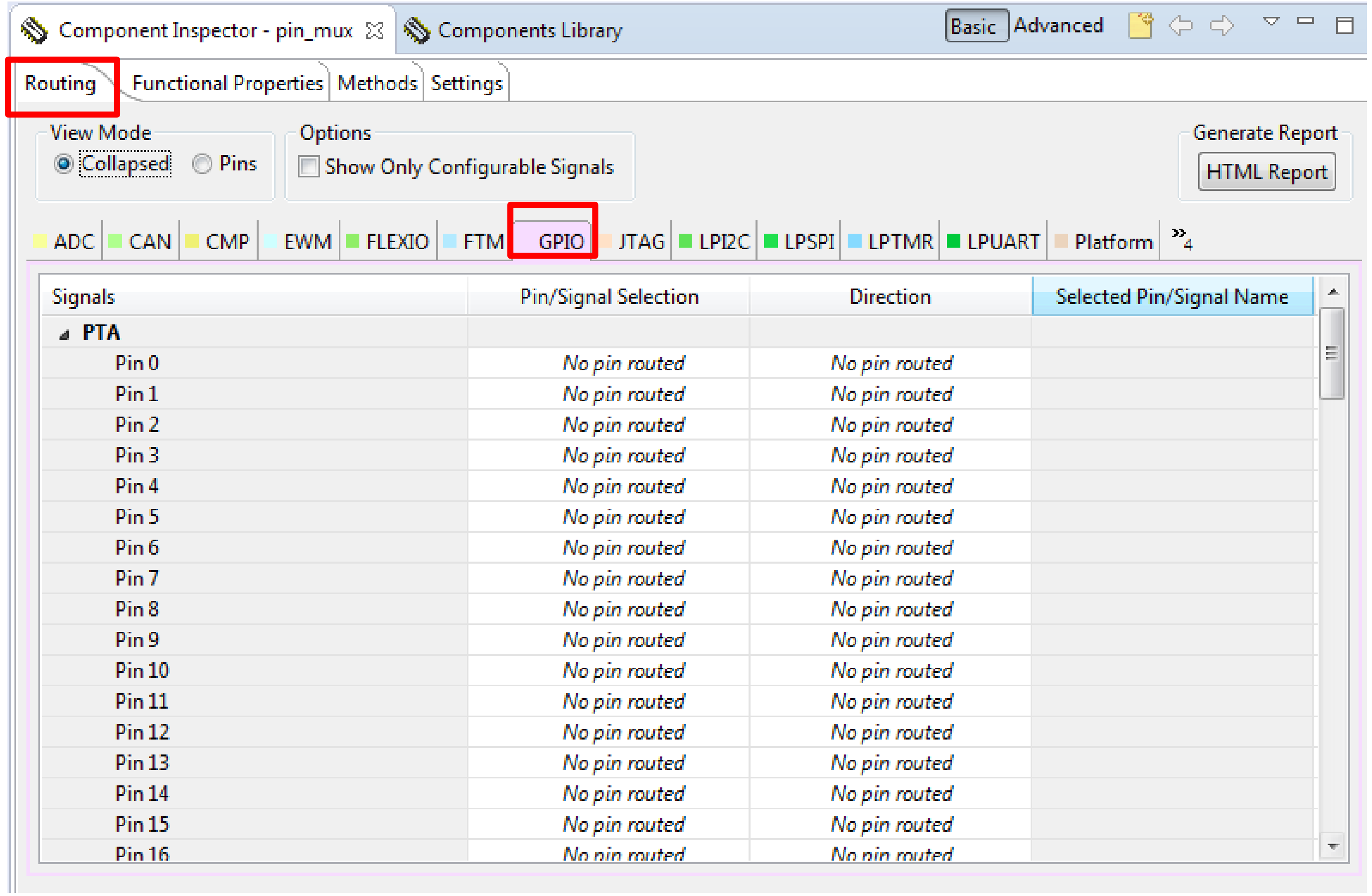


Components - BlinkingLED

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

S32K144 Blinking LED: Select I/O pins direction

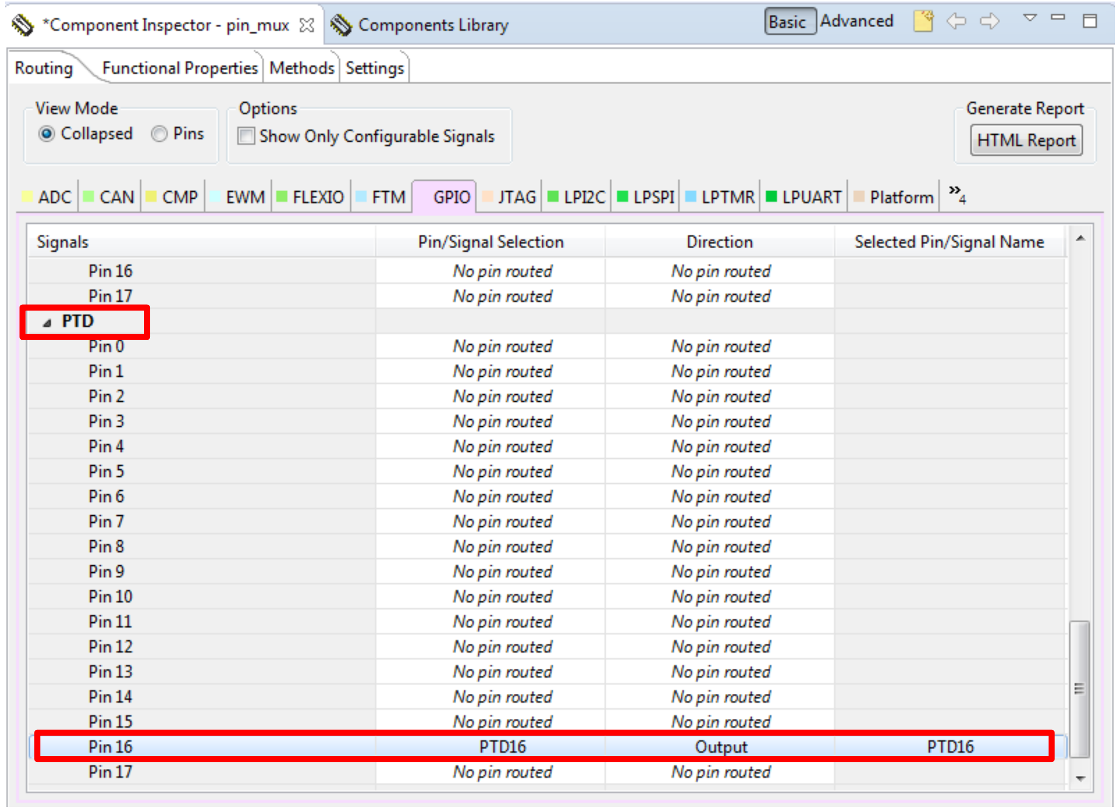
- In the **Component Inspector** window
- Select **GPIO** tab inside the **Routing** tab



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 Blinking LED: Select Output pin

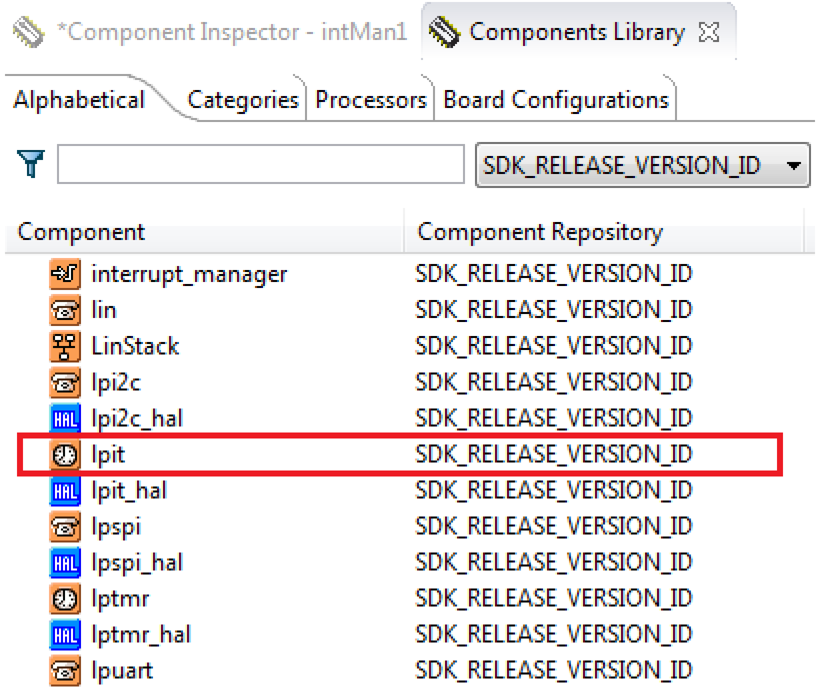
- Go to **PTD** and select pin **16**.
- In the **Pin/Signal Selection** Colum, select **PTD16**.
- In the **Direction** Colum, select **Output.**



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 Blinking LED: Add LPIT Component

- 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.
- Adding the lpit component will automatically add clock_manager and interrupt_manager components

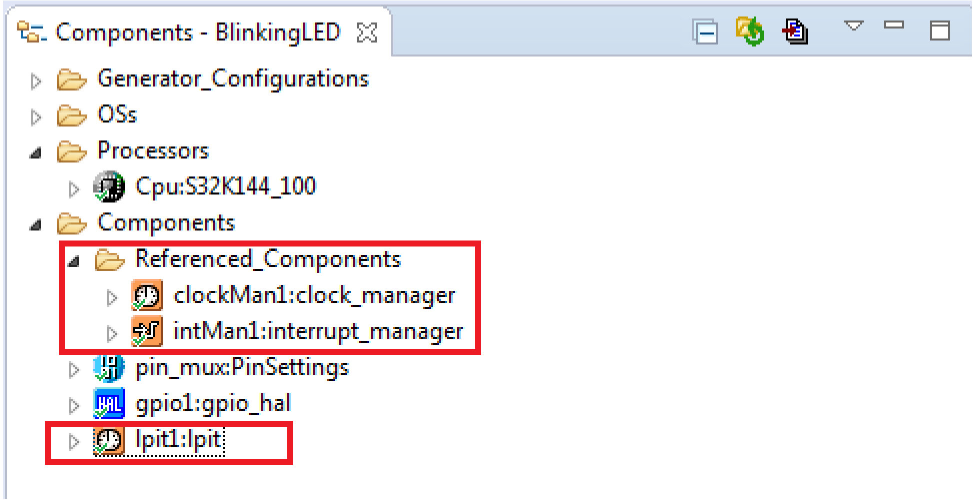


*Component Inspector - intMan1 | Components Library

Alphabetical | Categories | Processors | Board Configurations

SDK_RELEASE_VERSION_ID

Component	Component Repository
interrupt_manager	SDK_RELEASE_VERSION_ID
lin	SDK_RELEASE_VERSION_ID
LinStack	SDK_RELEASE_VERSION_ID
lpi2c	SDK_RELEASE_VERSION_ID
lpi2c_hal	SDK_RELEASE_VERSION_ID
lpit	SDK_RELEASE_VERSION_ID
lpit_hal	SDK_RELEASE_VERSION_ID
lpspi	SDK_RELEASE_VERSION_ID
lpspi_hal	SDK_RELEASE_VERSION_ID
lptmr	SDK_RELEASE_VERSION_ID
lptmr_hal	SDK_RELEASE_VERSION_ID
lpuart	SDK_RELEASE_VERSION_ID

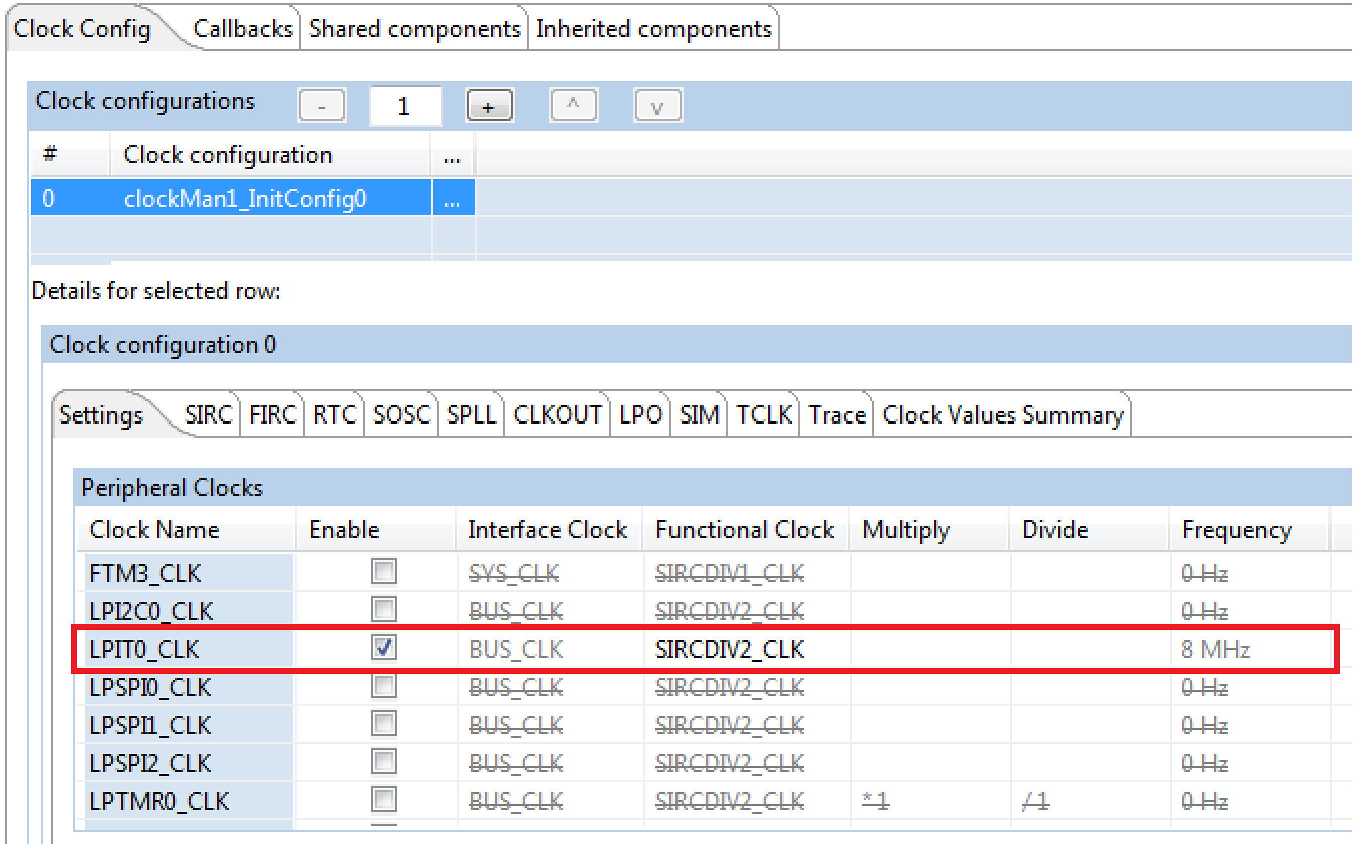


Components - BlinkingLED

- Generator_Configurations
- OSs
- Processors
  - Cpu:S32K144_100
- Components
  - Referenced_Components
    - clockMan1:clock_manager
    - intMan1:interrupt_manager
  - pin_mux:PinSettings
  - gpio1:gpio_hal
  - lpit1:lpit

S32K144 Blinking LED: Peripheral Clocks

- When adding a component to project, the clock_manger component enables the appropriate peripheral clocks.



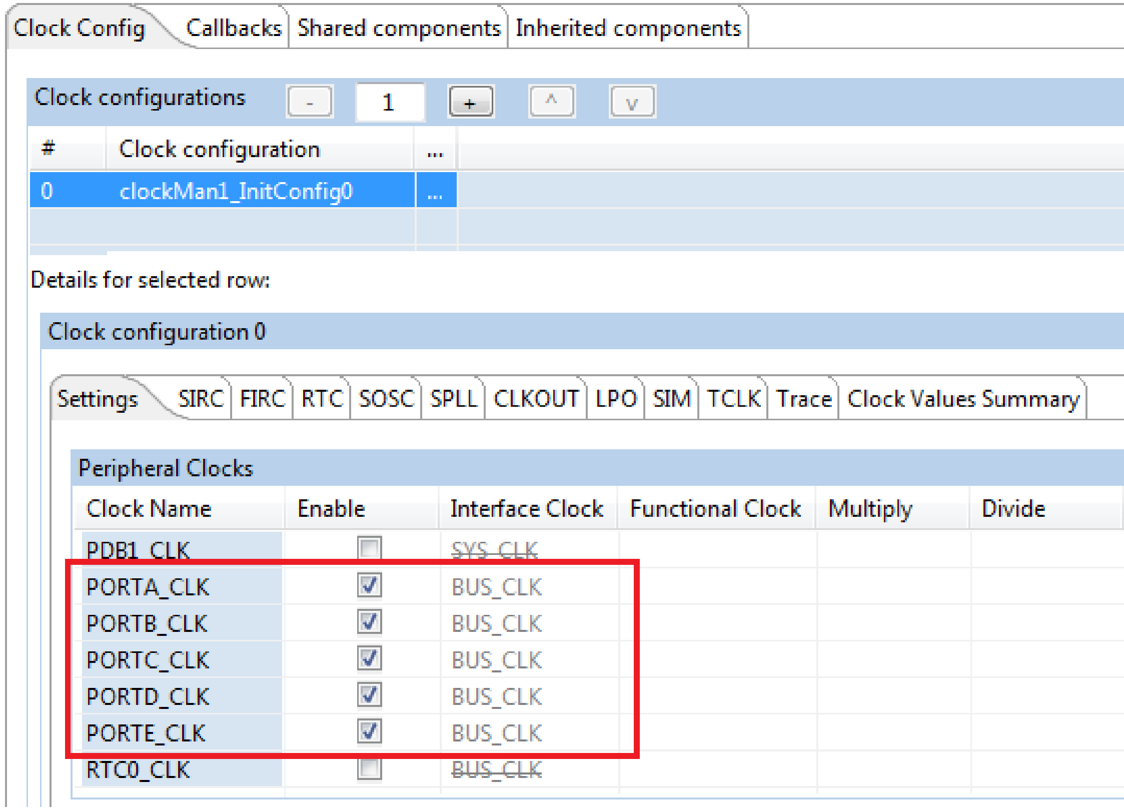
#	Clock configuration	...
0	clockMan1_InitConfig0	...

Details for selected row:

Clock configuration 0

Peripheral Clocks

Clock Name	Enable	Interface Clock	Functional Clock	Multiply	Divide	Frequency
FTM3_CLK	□	~~SYS_CLK~~	~~SIRCDIV1_CLK~~			~~0 Hz~~
LPI2C0_CLK	□	~~BUS_CLK~~	~~SIRCDIV2_CLK~~			~~0 Hz~~
LPIT0_CLK	☑	BUS_CLK	SIRCDIV2_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~~
LPTMR0_CLK	□	~~BUS_CLK~~	~~SIRCDIV2_CLK~~	~~*1~~	~~/1~~	~~0 Hz~~



#	Clock configuration	...
0	clockMan1_InitConfig0	...

Details for selected row:

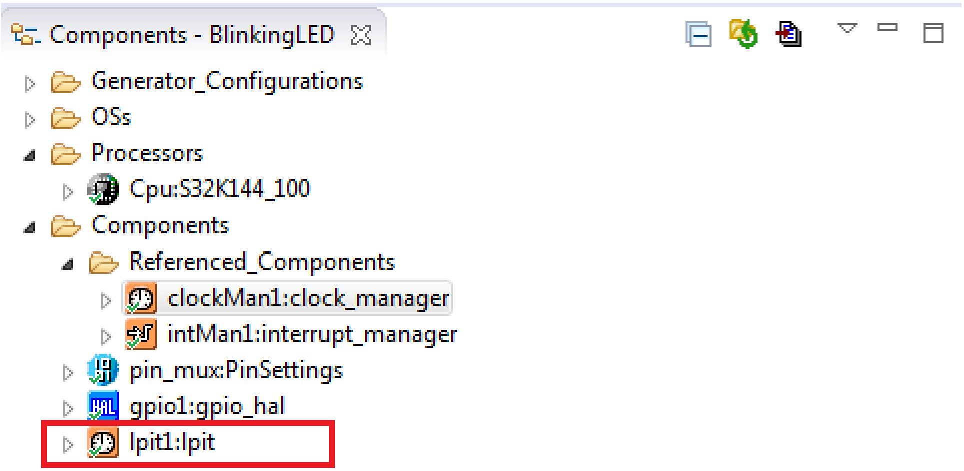
Clock configuration 0

Peripheral Clocks

Clock Name	Enable	Interface Clock	Functional Clock	Multiply	Divide
PDB1_CLK	□	~~SYS_CLK~~			
PORTA_CLK	☑	BUS_CLK			
PORTB_CLK	☑	BUS_CLK			
PORTC_CLK	☑	BUS_CLK			
PORTD_CLK	☑	BUS_CLK			
PORTE_CLK	☑	BUS_CLK			
RTC0_CLK	□	~~BUS_CLK~~			

S32K144 Blinking LED: LPIT Configuration

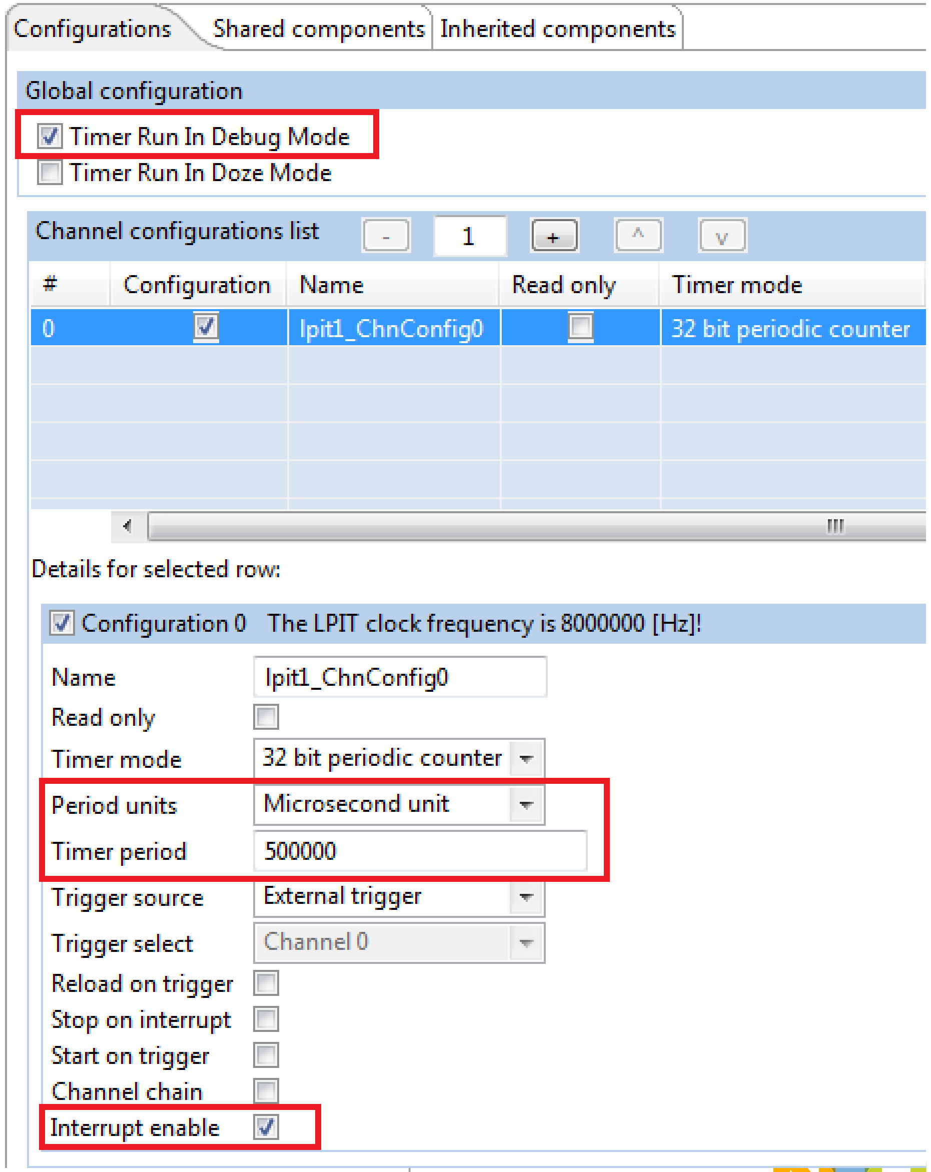
In the **Components Window** select the **lpit component**



```
Components - BlinkingLED
▷ Generator_Configurations
▷ OSs
◢ Processors
    ▷ Cpu:S32K144_100
◢ Components
    ◢ Referenced_Components
        ▷ clockMan1:clock_manager
        ▷ intMan1:interrupt_manager
    ▷ pin_mux:PinSettings
    ▷ gpio1:gpio_hal
    ▷ lpit1:lpit
```

S32K144 Blinking LED: LPIT Configuration

- Go to **Components Inspector**.
- Check the **Timer Run In Debug Mode** box
- Check the **Interrupt enable** box
- Select **Microsecond unit** as period unit
- In the **Time period** field type **500000** counts for 0.5 sec.



Global configuration

☑ Timer Run In Debug Mode
□ Timer Run In Doze Mode

Channel configurations list

#	Configuration	Name	Read only	Timer mode
0	☑	lpit1_ChnConfig0	□	32 bit periodic counter

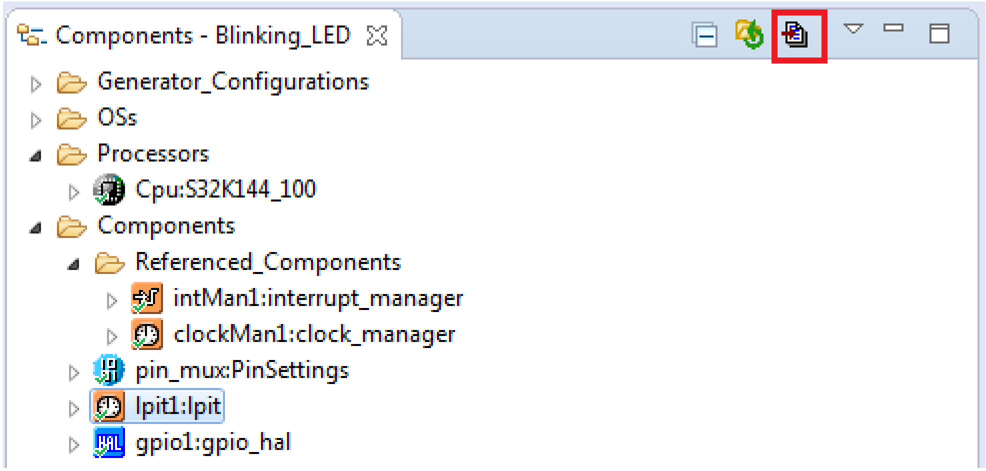
Details for selected row:

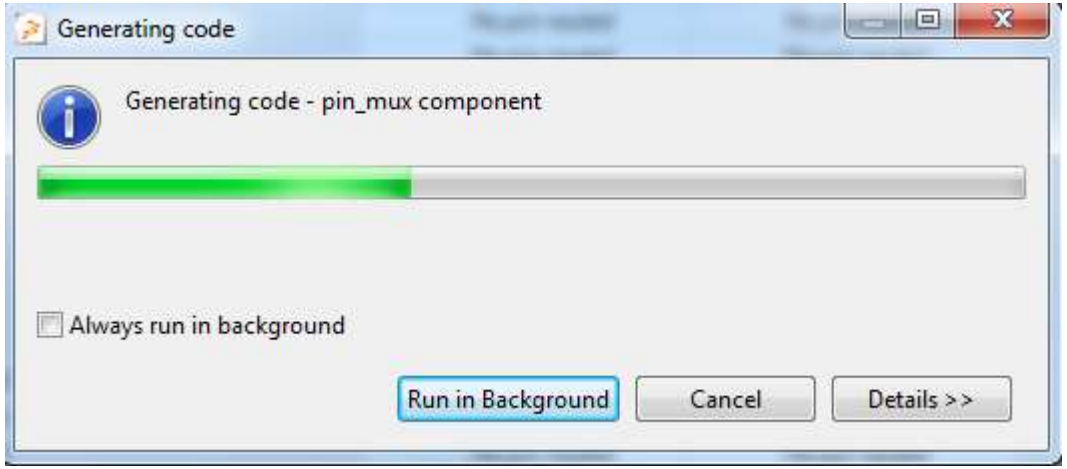
☑ Configuration 0 The LPIT clock frequency is 8000000 [Hz]!

Name	lpit1_ChnConfig0
Read only	□
Timer mode	32 bit periodic counter
Period units	Microsecond unit
Timer period	500000
Trigger source	External trigger
Trigger select	Channel 0
Reload on trigger	□
Stop on interrupt	□
Start on trigger	□
Channel chain	□
Interrupt enable	☑

S32K144 Blinking LED: 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.



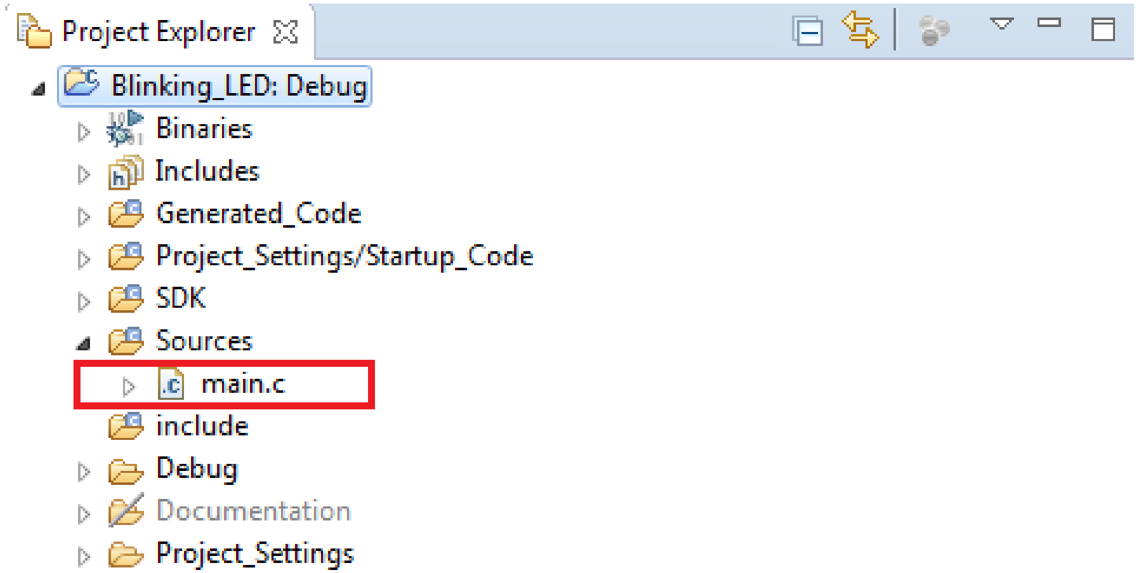


Generating code - pin_mux component

☐ Always run in background

S32K144 Blinking LED: Application Code

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

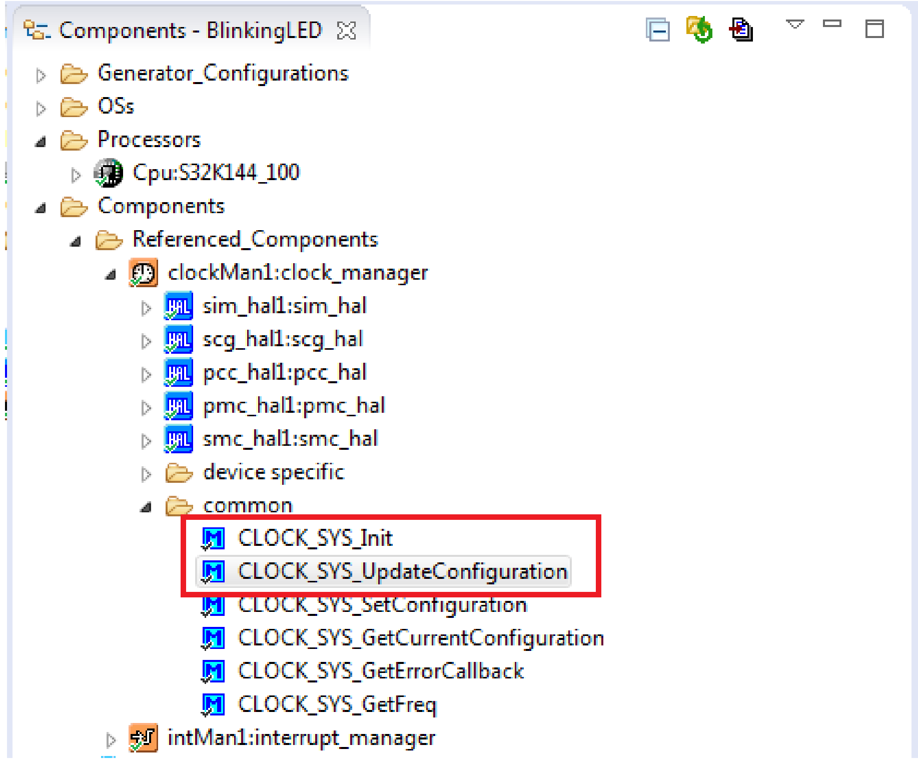


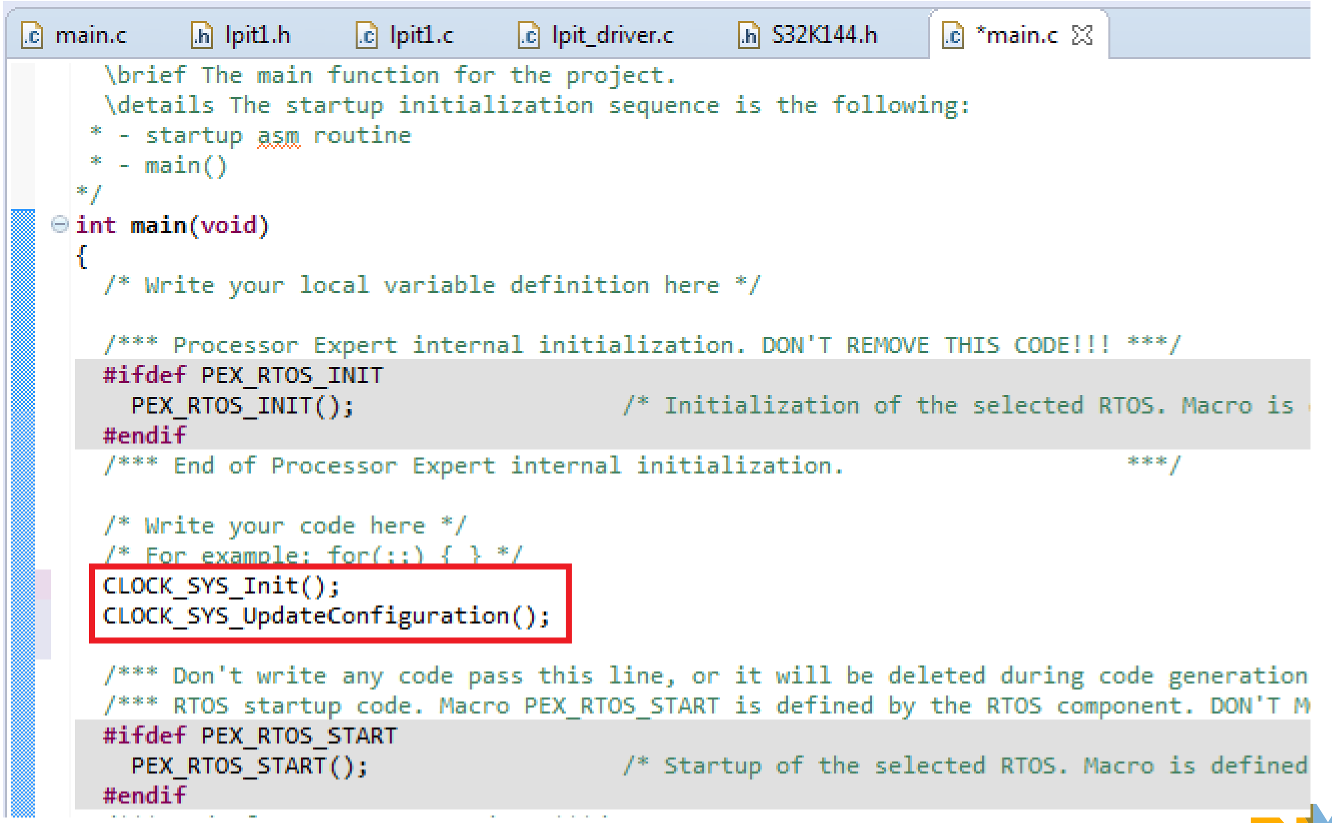
Project Explorer

Blinking_LED: Debug
- Binaries
- Includes
- Generated_Code
- Project_Settings/Startup_Code
- SDK
- Sources
  - main.c
  - include
- Debug
- Documentation
- Project_Settings

S32K144 Blinking LED: 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.

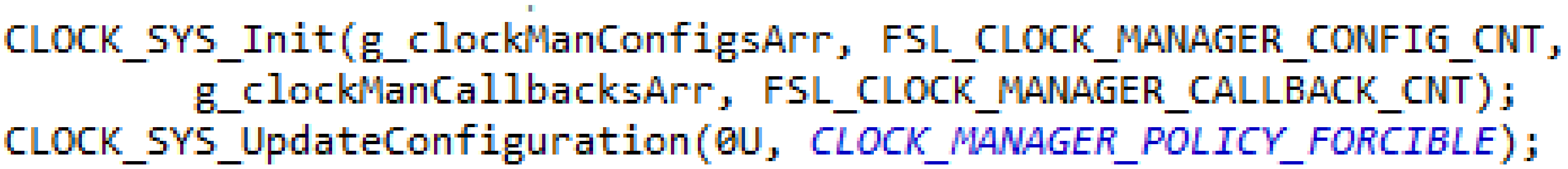




```
  \brief The main function for the project.
  \details The startup initialization sequence is the following:
 * - startup asm routine
 * - main()
*/
int main(void)
{
  /* Write your local variable definition here */

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

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

  /*** 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 M
  #ifdef PEX_RTOS_START
    PEX_RTOS_START();                  /* Startup of the selected RTOS. Macro is defined
  #endif
```

S32K144 Blinking LED: Init and Update Configuration Functions

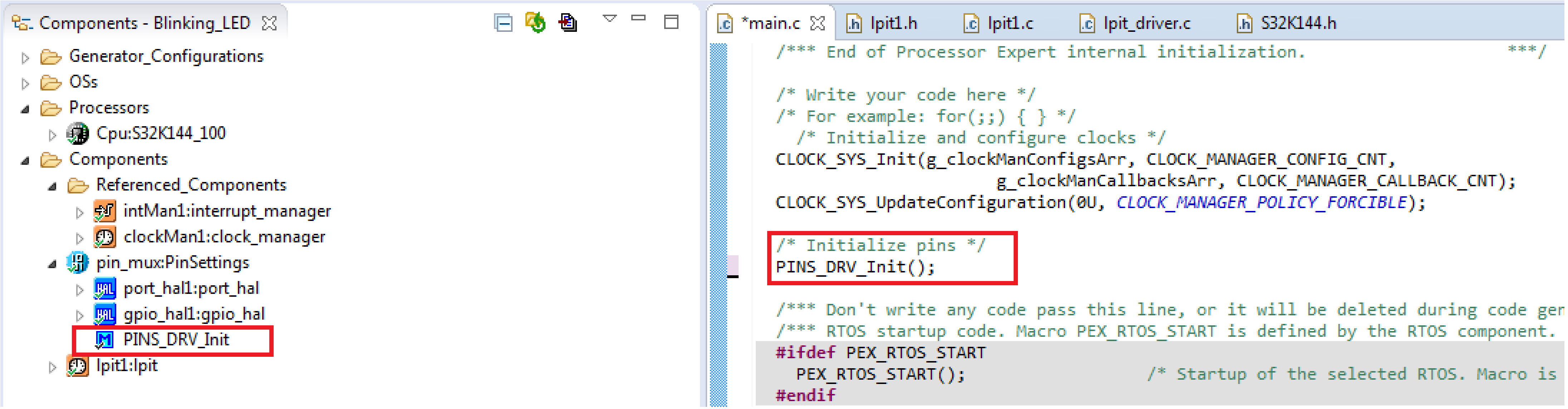
- In the **CLOCK_SYS_Init** function add the following parameters.
  - g_clockManConfigsArr,
  - CLOCK_MANAGER_CONFIG_CNT,
  - g_clockManCallbacksArr,
  - CLOCK_MANAGER_CALLBACK_CNT
- In the **CLOCK_SYS_UpdateConfiguration** add the following parameters.
  - 0U,
  - CLOCK_MANAGER_POLICY_FORCIBLE



```
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 Blinking LED: Initialize Pins

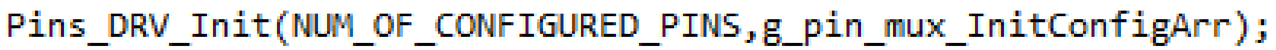
- Expand the **pin_mux** component in the **Components** Window.
- Drag and drop the **Pins_DRV_Init** function inside the, into main, below the clock configuration



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

/* Write your code here */
/* For example: for(;;) { } */
  /* Initialize and configure clocks */
CLOCK_SYS_Init(g_clockManConfigsArr, CLOCK_MANAGER_CONFIG_CNT,
                      g_clockManCallbacksArr, CLOCK_MANAGER_CALLBACK_CNT);
CLOCK_SYS_UpdateConfiguration(0U, CLOCK_MANAGER_POLICY_FORCIBLE);

/* Initialize pins */
PINS_DRV_Init();

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

S32K144 Blinking LED: 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

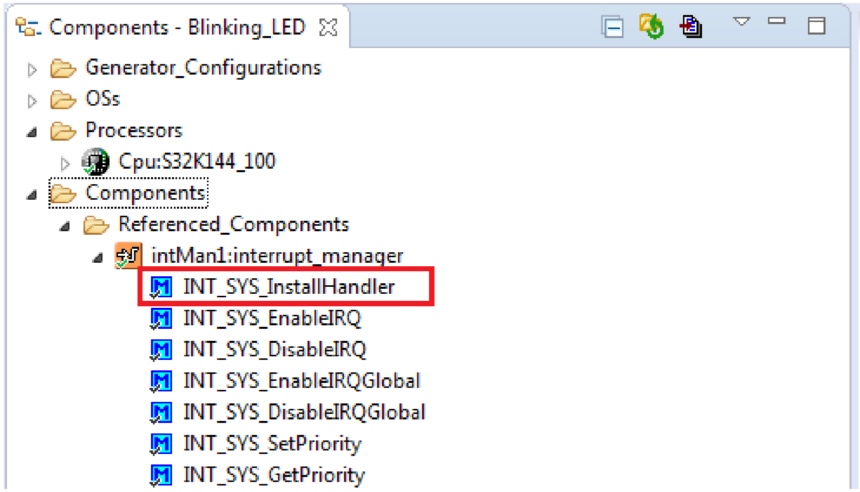


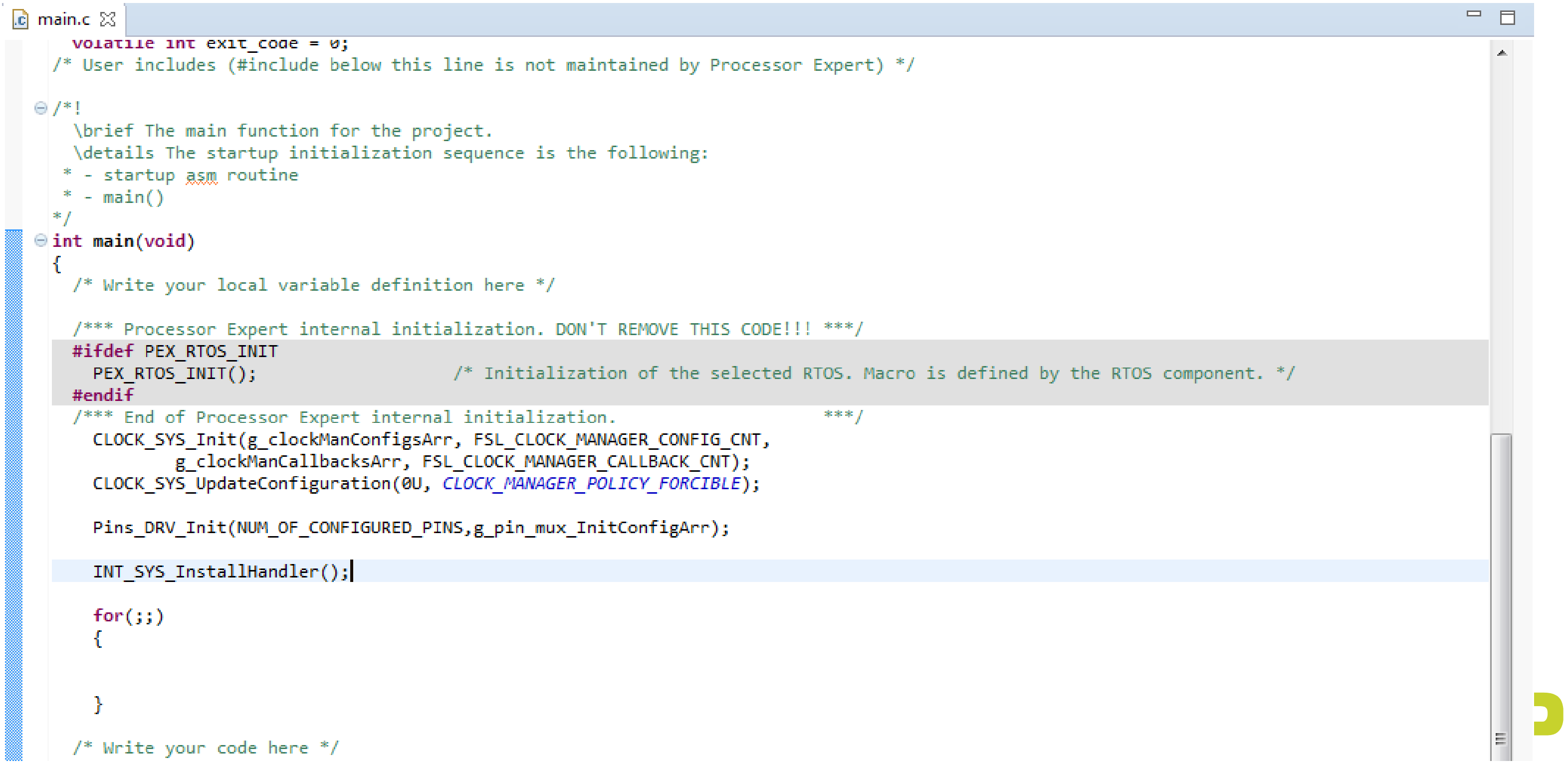
```
Pins_DRV_Init(NUM_OF_CONFIGURED_PINS,g_pin_mux_InitConfigArr);
```

S32K144 Blinking LED: 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

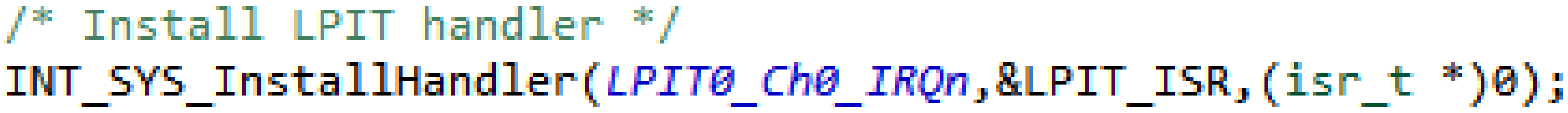




```
  volatile int exit_code = 0;
/* User includes (#include below this line is not maintained by Processor Expert) */

/*!
  \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 Blinking LED: Install LPIT Interrupt

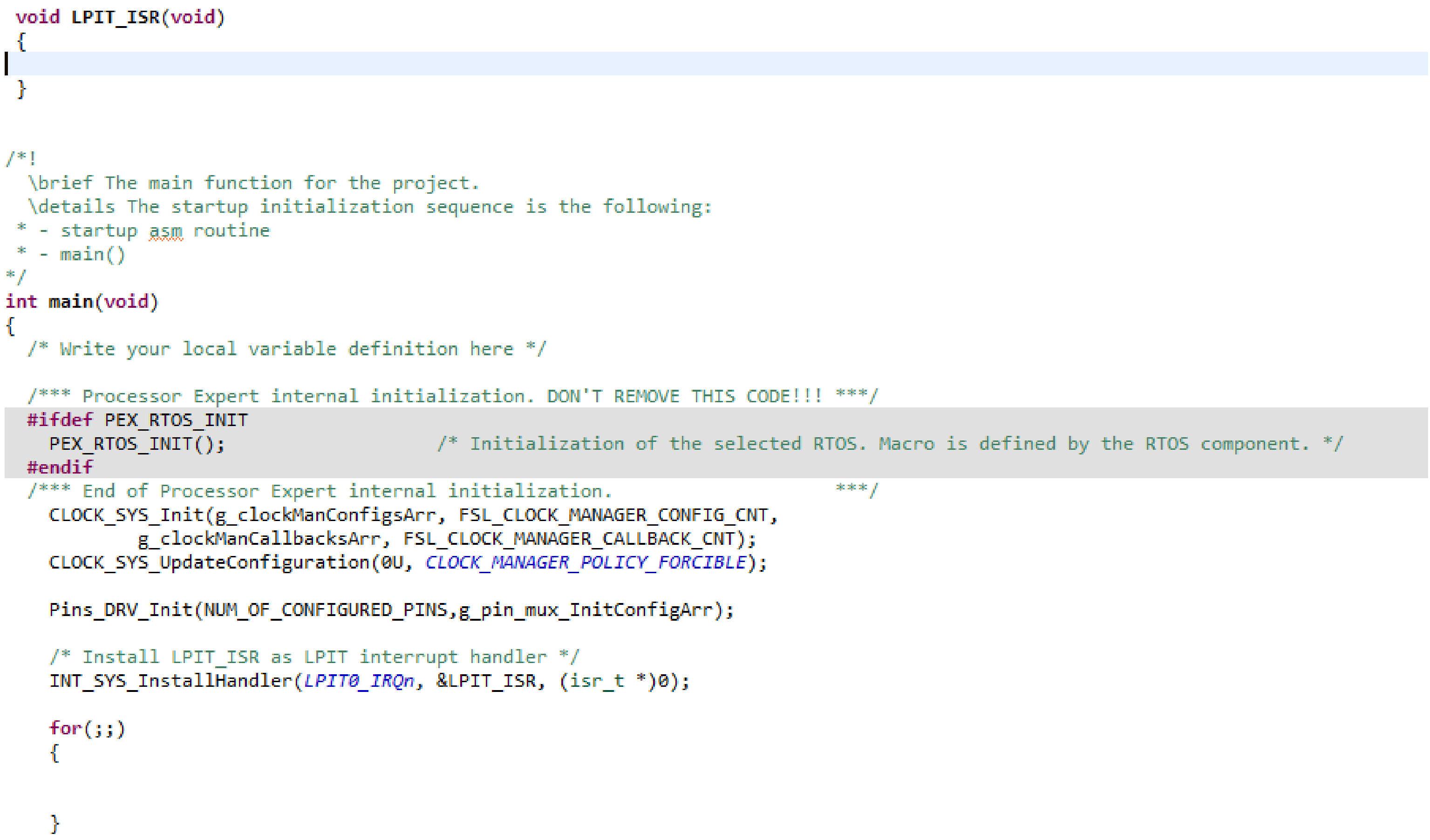
- In the **INT_SYS_InstallHandle**r function add the following parameters:
  - LPIT0_Ch0_IRQn,
  - &LPIT_ISR,
  - (isr_t *)0



```
/* Install LPIT handler */
INT_SYS_InstallHandler(LPIT0_Ch0_IRQn,&LPIT_ISR,(isr_t *)0);
```

S32K144 Blinking LED: Install LPIT Interrupt

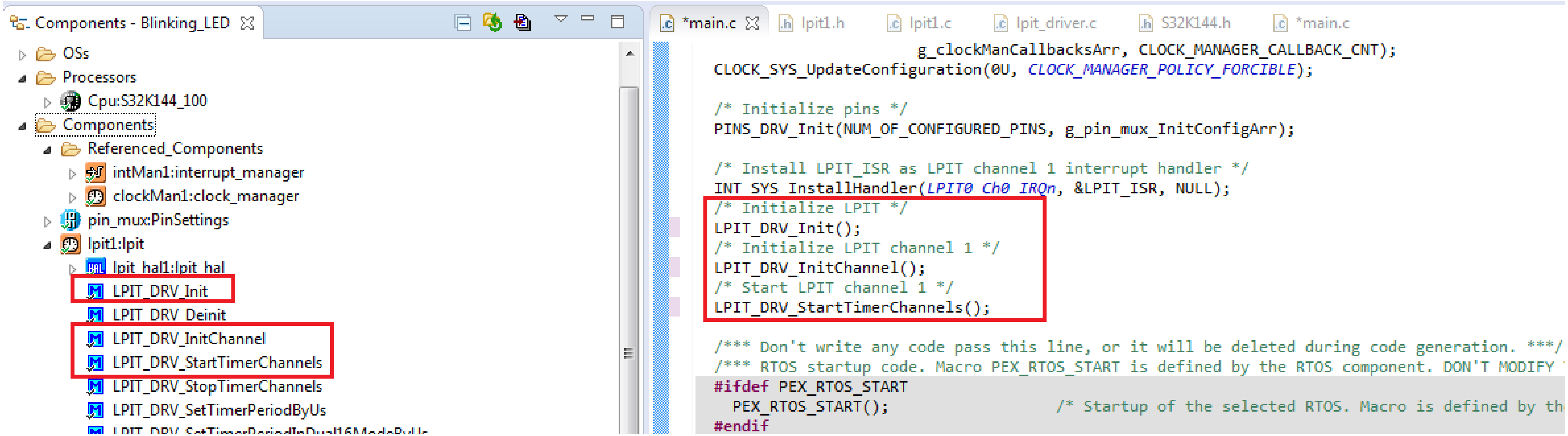
- Create a new function named LPIT_ISR and placed above main



```
 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 Blinking LED: 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**



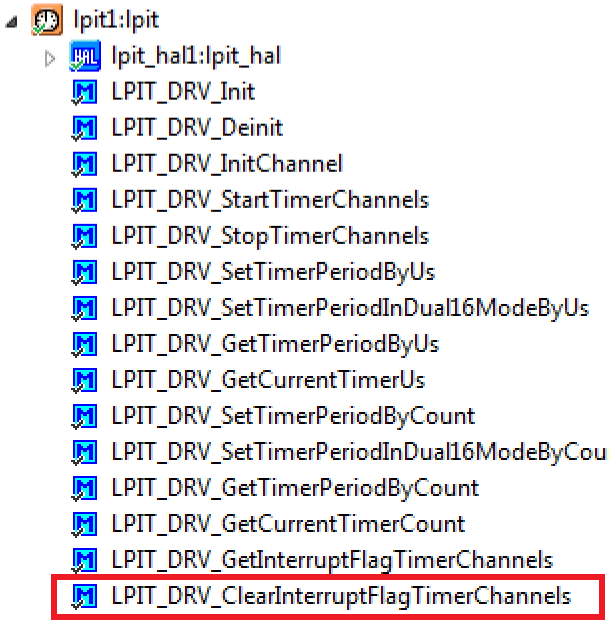
```
                         g_clockManCallbacksArr, CLOCK_MANAGER_CALLBACK_CNT);
CLOCK_SYS_UpdateConfiguration(0U, CLOCK_MANAGER_POLICY_FORCIBLE);

/* Initialize pins */
PINS_DRV_Init(NUM_OF_CONFIGURED_PINS, g_pin_mux_InitConfigArr);

/* Install LPIT_ISR as LPIT channel 1 interrupt handler */
INT_SYS_InstallHandler(LPIT0_Ch0_IRQn, &LPIT_ISR, NULL);
/* Initialize LPIT */
LPIT_DRV_Init();
/* Initialize LPIT channel 1 */
LPIT_DRV_InitChannel();
/* Start LPIT channel 1 */
LPIT_DRV_StartTimerChannels();

/*** 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
#ifdef PEX_RTOS_START
  PEX_RTOS_START();                  /* Startup of the selected RTOS. Macro is defined by th
#endif
```

S32K144 Blinking LED: Initialize LPIT

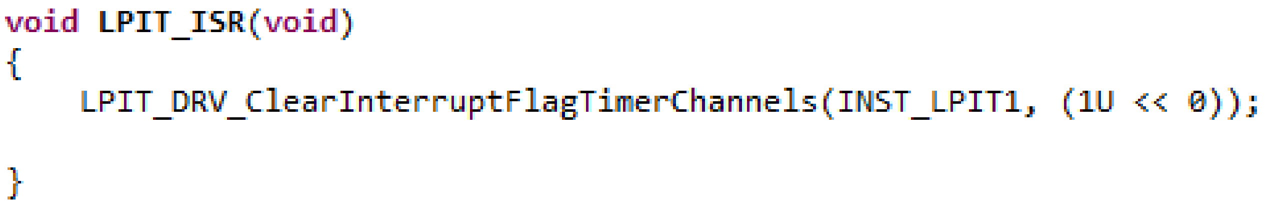
- In the **LPIT_DRV_Init** function add the following parameters:.
  - INST_LPIT1,
  - &lpit1_InitConfig
- In the **LPIT_DRV_InitChannel** function add the following parameters:.
  - INST_LPIT1,
  - 0
  - &lpit1_ChnConfig0
- In the **LPIT_DRV_StartTimerChannels** function add the following parameters:.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

S32K144 Blinking LED: 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:.
  - FSL_LPIT1
  - (1 << 0)



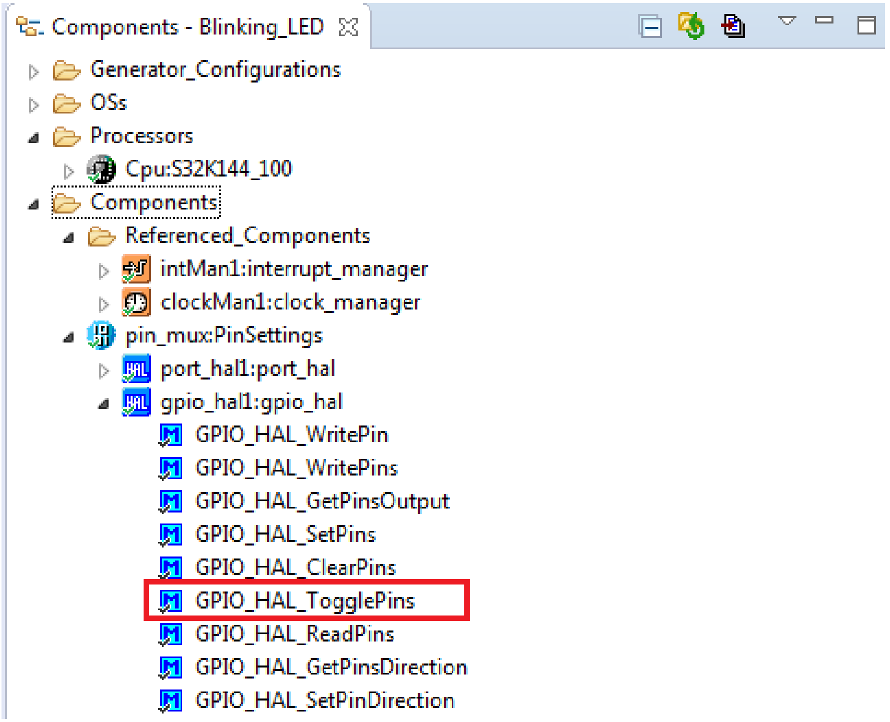
lpit1:lpit
lpit_hal1:lpit_hal
LPIT_DRV_Init
LPIT_DRV_Deinit
LPIT_DRV_InitChannel
LPIT_DRV_StartTimerChannels
LPIT_DRV_StopTimerChannels
LPIT_DRV_SetTimerPeriodByUs
LPIT_DRV_SetTimerPeriodInDual16ModeByUs
LPIT_DRV_GetTimerPeriodByUs
LPIT_DRV_GetCurrentTimerUs
LPIT_DRV_SetTimerPeriodByCount
LPIT_DRV_SetTimerPeriodInDual16ModeByCount
LPIT_DRV_GetTimerPeriodByCount
LPIT_DRV_GetCurrentTimerCount
LPIT_DRV_GetInterruptFlagTimerChannels
LPIT_DRV_ClearInterruptFlagTimerChannels



```
void LPIT_ISR(void)
{
    LPIT_DRV_ClearInterruptFlagTimerChannels(INST_LPIT1, (1U << 0));

}
```

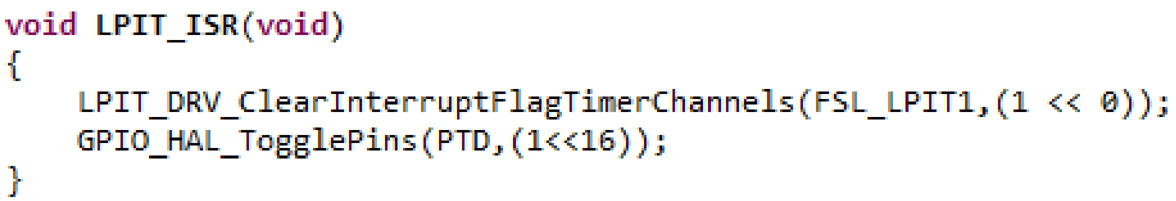
S32K144 Blinking LED: Toggle Green LED (PTD16)

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



Components - Blinking_LED

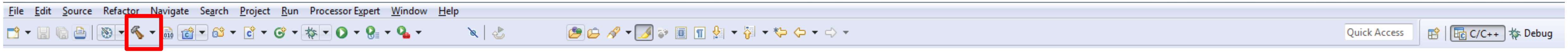
- Generator_Configurations
- OSs
- Processors
  - Cpu:S32K144_100
- Components
  - Referenced_Components
    - intMan1:interrupt_manager
    - clockMan1:clock_manager
  - pin_mux:PinSettings
    - port_hal1:port_hal
    - gpio_hal1:gpio_hal
      - GPIO_HAL_WritePin
      - GPIO_HAL_WritePins
      - GPIO_HAL_GetPinsOutput
      - GPIO_HAL_SetPins
      - GPIO_HAL_ClearPins
      - GPIO_HAL_TogglePins
      - GPIO_HAL_ReadPins
      - GPIO_HAL_GetPinsDirection
      - GPIO_HAL_SetPinDirection



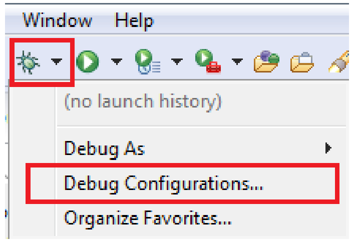
```
void LPIT_ISR(void)
{
    LPIT_DRV_ClearInterruptFlagTimerChannels(FSL_LPIT1,(1 << 0));
    GPIO_HAL_TogglePins(PTD,(1<<16));
}
```

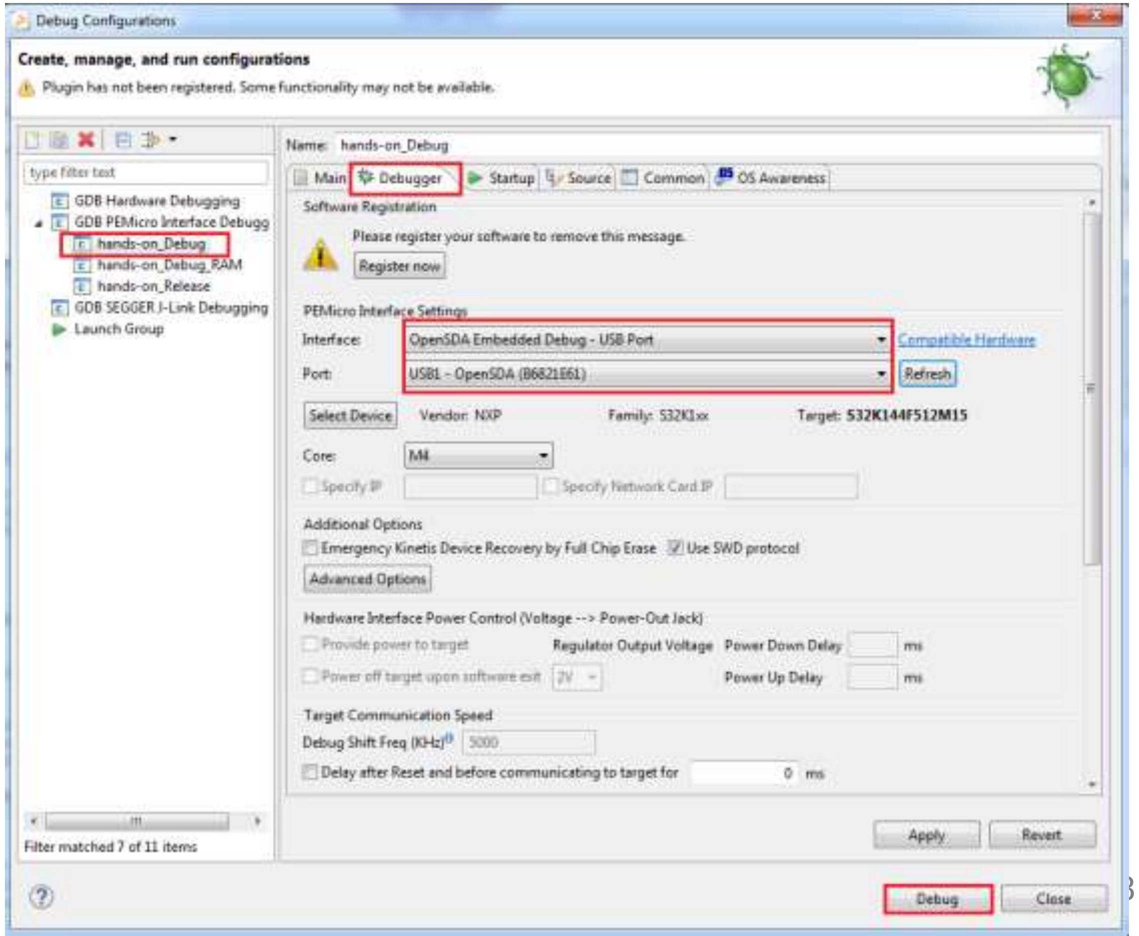
S32K144 Blinking LED: Build and debug the application

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



- Configure the debug configuration start a new debug session





Create, manage, and run configurations

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

Filter matched 7 of 11 items

Name: hands-on_Debug

Software Registration

Please register your software to remove this message.

PEMicro Interface Settings

Interface: OpenSDA Embedded Debug - USB Port

Port: USB1 - OpenSDA (B6821E61)

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

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   3V   Power Up Delay ms

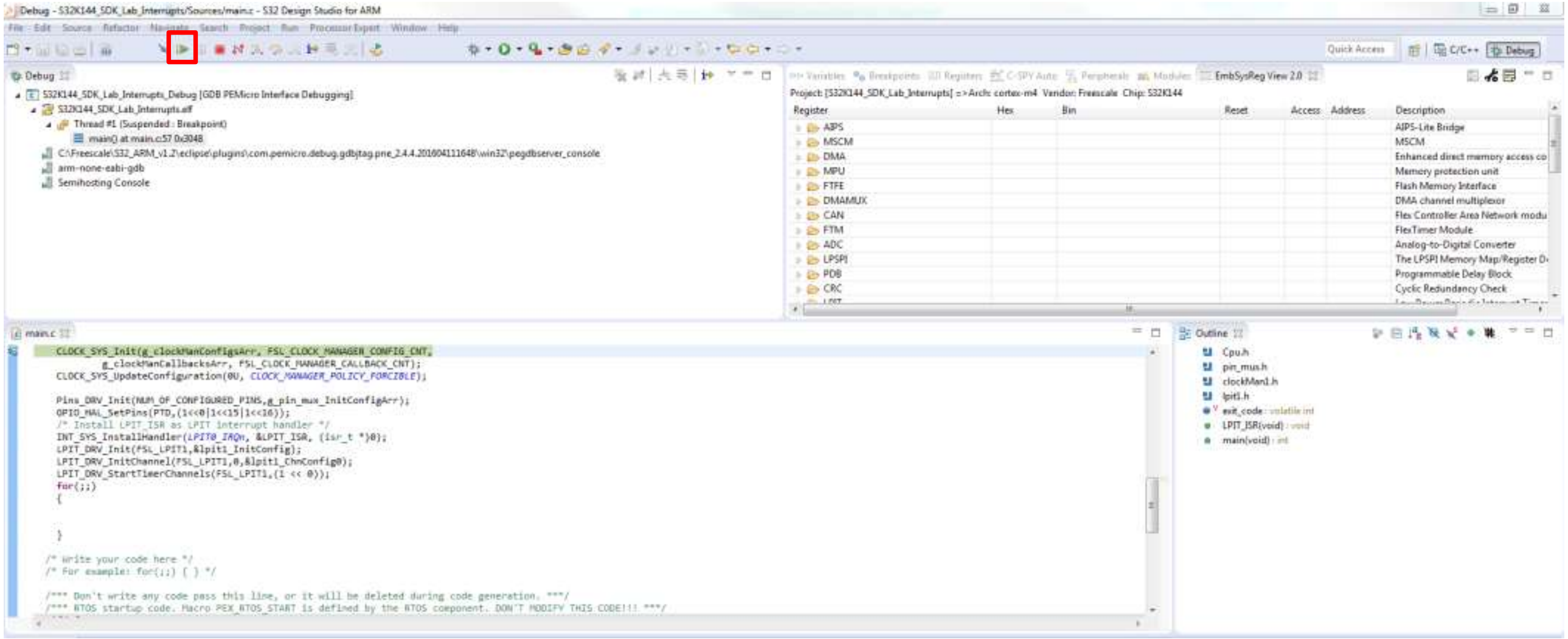
Target Communication Speed

Debug Shift Freq (KHz) 5000

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

S32K144 Blinking LED: Build and debug the application

- In the debug perspective click the run icon to start the project.
- Green LED should toggle every 0.5 sec.


Debug - S32K144_SDK_Lab_Interrupts/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
Debug
Variables
Breakpoints
Registers
C-SPY Auto
Peripherals
Modules
EmbSysReg View 2.0
main.c
Outline

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

```
    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!!! ***/
```

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

S32K144 Blinking LED: Challenge

- Toggle Green LED every 100 ms.

02.

Hands-on – Secure CAN

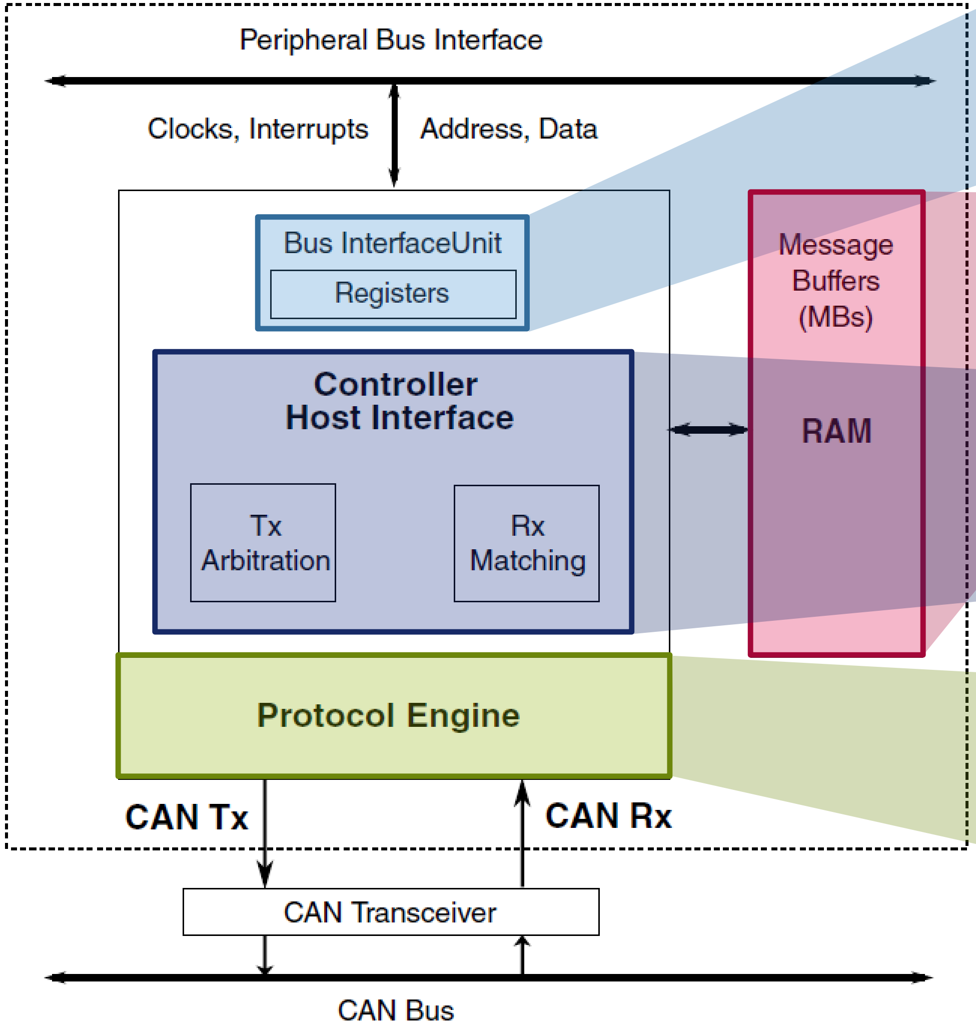
S32K144 Secured CAN: Objective

- In this lab you will learn:
  - About the features of the FlexCAN module on S32K144
  - About the features of the CSEc module on S32K144
  - How to configure FlexCAN peripheral for both Rx & Tx
  - How to initiate a CAN communication between two S32K boards
  - How to use the CSEc driver to encrypt/decrypt the messages (AES)

S32K144 Secured CAN: CAN Theory

- Full implementation of the CAN FD & CAN 2.0 B
  - data field bitrate up to 8Mbps
- Flexible mailboxes (0/8/16/32/64 bytes data length)
- Listen-Only mode capability
- Programmable Loop-Back mode supporting self-test operation
- Programmable transmission priority scheme
- Independence from the transmission medium
- CRC status for transmitted message
- Full featured Rx FIFO with storage capacity for 6 frames
- DMA request for Rx FIFO
- Programmable clock source to the CAN Protocol Interface, either bus clock or crystal oscillator
- 100% backward compatibility with previous FlexCAN version
- 3 FlexCAN instances

S32K144 Secured CAN: CAN Theory



Access to and from the internal interface bus (clocks, address and data buses, interrupts, DMA and test signals)

Embedded RAM dedicated to the FlexCAN

Message buffer selection for reception and transmission (arbitration and ID matching algorithms)

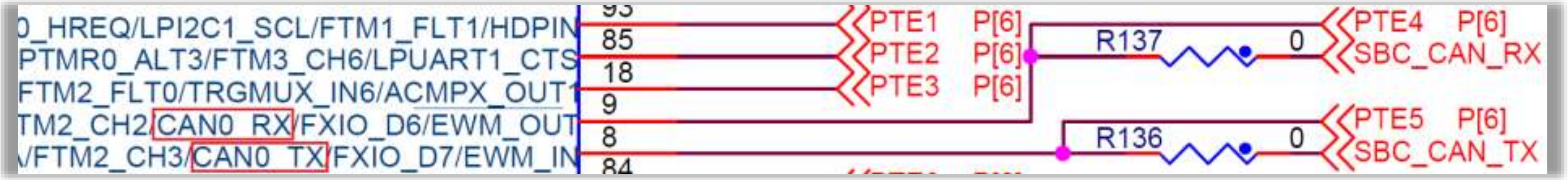
Serial communication on the CAN bus (RAM access requests for rx and tx frames, rx messages validation, error handling)

S32K144 Secured CAN: CSEc Theory

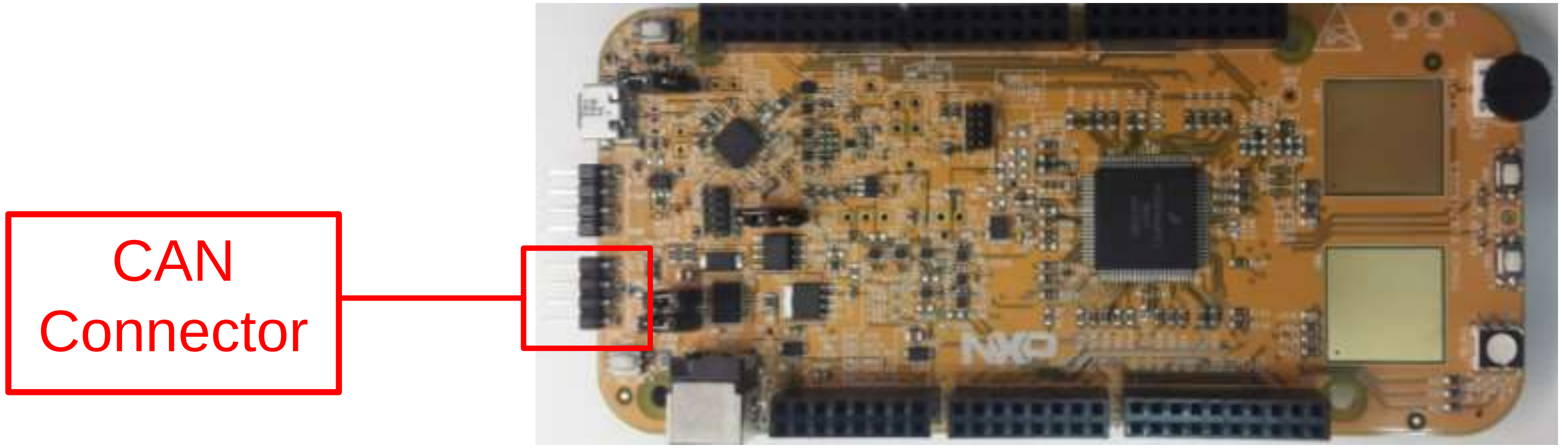
- Cryptographic Services Engine (CSEc) – comprehensive set of cryptographic functions (SHE)
  - >10 general purpose keys
  - AES-128, CBC, ECB, CMAC
  - Sequential, Parallel, and Strict Boot mode
  - AES-128 CMAC calculation and authentication
  - Pseudo random number generation (PRNG) and true random number generation (TRNG)

S32K144 Secured CAN: Resources

- S32K144 – FlexCAN signals & pins


0_HREQ/LPI2C1_SCL/FTM1_FLT1/HDPIN
PTMR0_ALT3/FTM3_CH6/LPUART1_CTS
FTM2_FLT0/TRGMUX_IN6/ACMPX_OUT1
TM2_CH2/CAN0_RX/FXIO_D6/EWM_OUT
/FTM2_CH3/CAN0_TX/FXIO_D7/EWM_IN
93
85
18
9
8
84
PTE1 P[6]
PTE2 P[6]
PTE3 P[6]
R137
0
PTE4 P[6]
SBC_CAN_RX
R136
0
PTE5 P[6]
SBC_CAN_TX

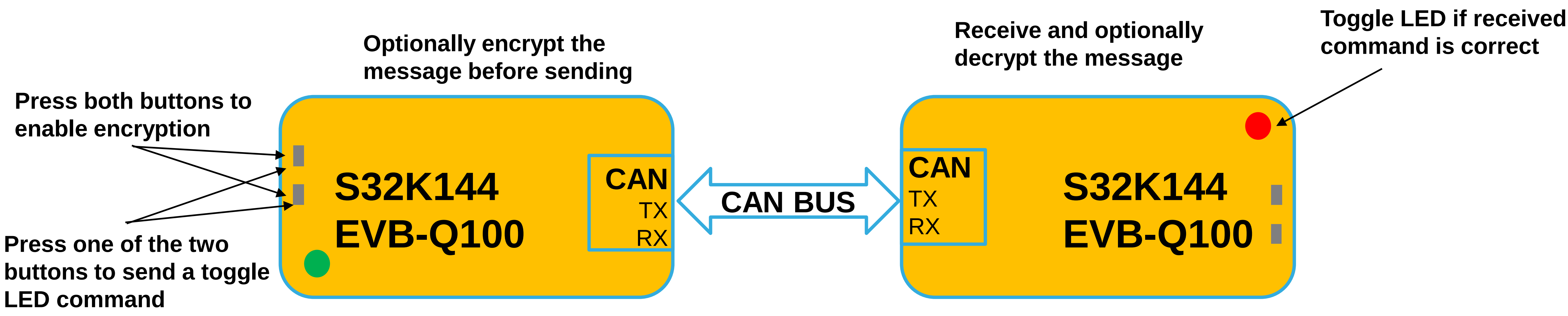
CAN0 Signal	S32K144 PIN
Tx	PTE5
Rx	PTE4



CAN Connector

S32K144 Secured CAN: Hands-on Preview

- Secured CAN communication between two S32K144 boards:
  - Message encryption at tx, decryption at rx – selectable through user buttons (blue LED on)
  - Toggle red/green LED when command successfully received (decrypted)

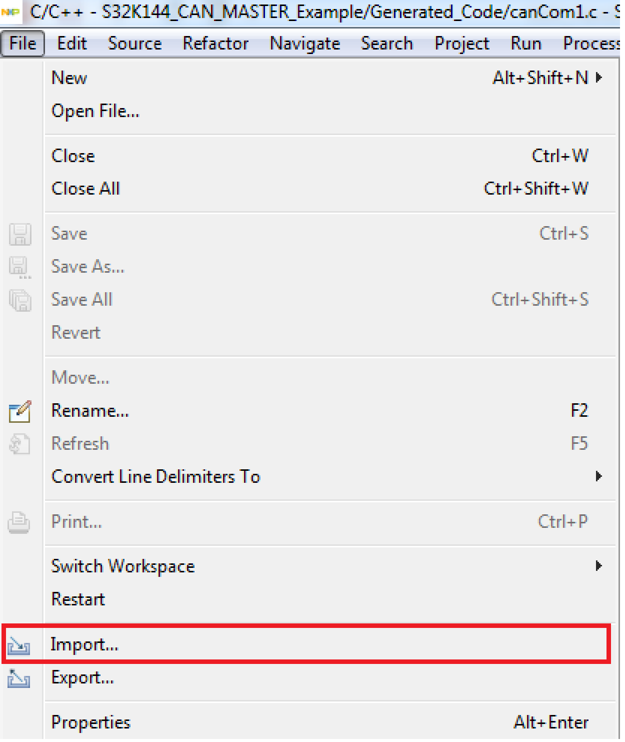


Press both buttons to enable encryption
Press one of the two buttons to send a toggle LED command
Optionally encrypt the message before sending
S32K144 EVB-Q100
CAN
TX
RX
CAN BUS
Receive and optionally decrypt the message
CAN
TX
RX
S32K144 EVB-Q100
Toggle LED if received command is correct

S32K144 Secured CAN: Importing demo applications

- Import ‘flexcan_encrypted’ example provided with the SDK:
  - File->Import->General->Existing Projects into Workspace->Select root directory
  - Select:

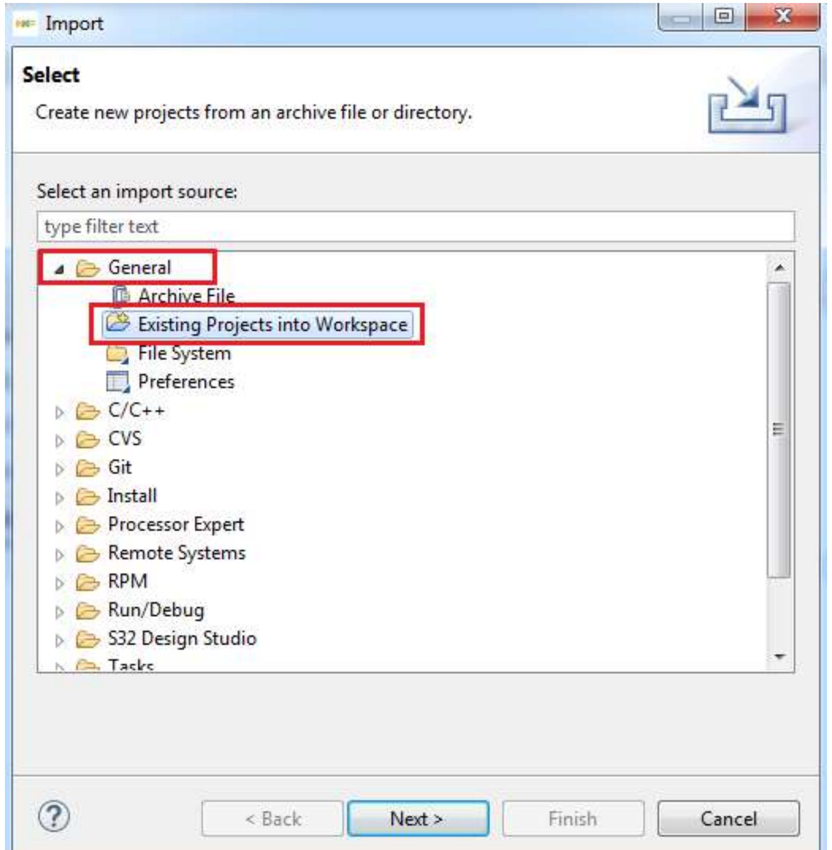
    {DS_InstallationFolder}\S32DS\S32SDK_S32K144_RTM_1.0.0\examples\S32K144\demo_apps\flexcan_encrypted
  - Make sure ‘Copy projects into workspace’ is checked, so the SDK example remains clean



C/C++ - S32K144_CAN_MASTER_Example/Generated_Code/canCom1.c - S

File Edit Source Refactor Navigate Search Project Run Process

New	Alt+Shift+N ▸
Open File...	
Close	Ctrl+W
Close All	Ctrl+Shift+W
Save	Ctrl+S
Save As...	
Save All	Ctrl+Shift+S
Revert	
Move...	
Rename...	F2
Refresh	F5
Convert Line Delimiters To	▸
Print...	Ctrl+P
Switch Workspace	▸
Restart	
Import...	
Export...	
Properties	Alt+Enter



Import

Select

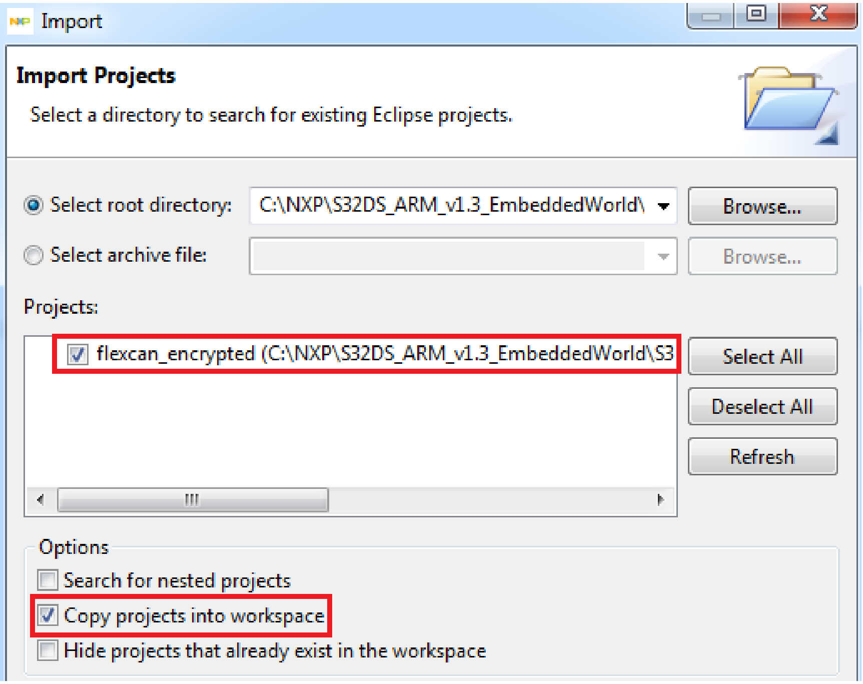
Create new projects from an archive file or directory.

Select an import source:

type filter text

- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - Preferences
- C/C++
- CVS
- Git
- Install
- Processor Expert
- Remote Systems
- RPM
- Run/Debug
- S32 Design Studio
- Tasks

< Back | Next > | Finish | Cancel



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: C:\NXP\S32DS_ARM_v1.3_EmbeddedWorld\ | Browse...

○ Select archive file: | Browse...

Projects:

☑ flexcan_encrypted (C:\NXP\S32DS_ARM_v1.3_EmbeddedWorld\S3

Select All | Deselect All | Refresh

Options

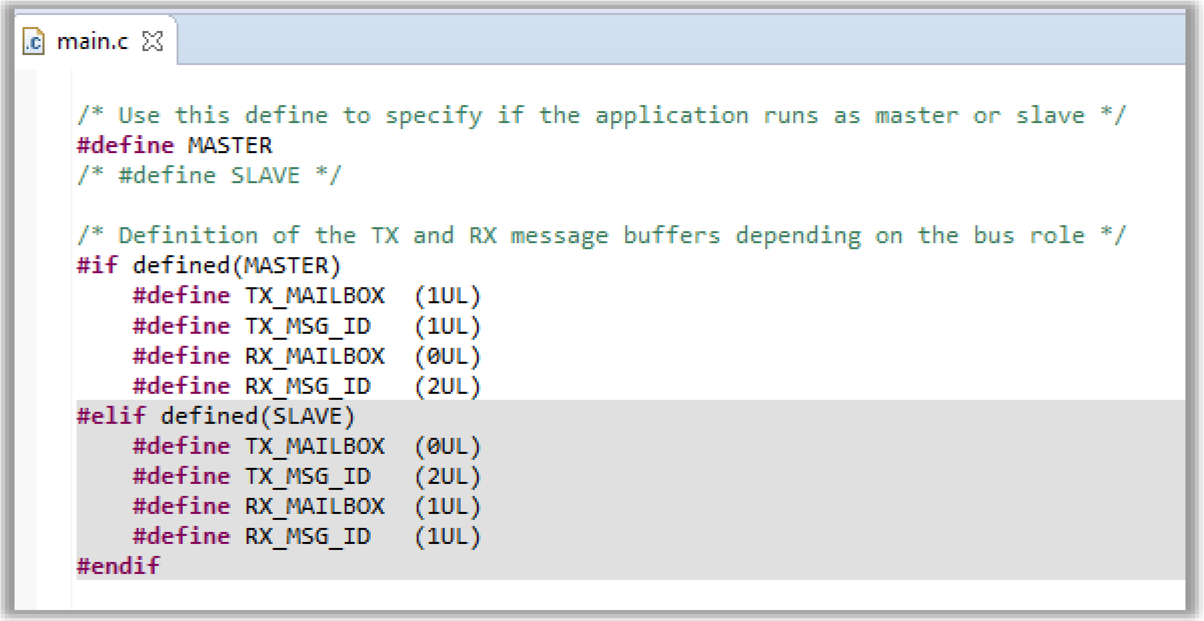
☐ Search for nested projects

☑ Copy projects into workspace

☐ Hide projects that already exist in the workspace

S32K144 Secured CAN: Master/Slave

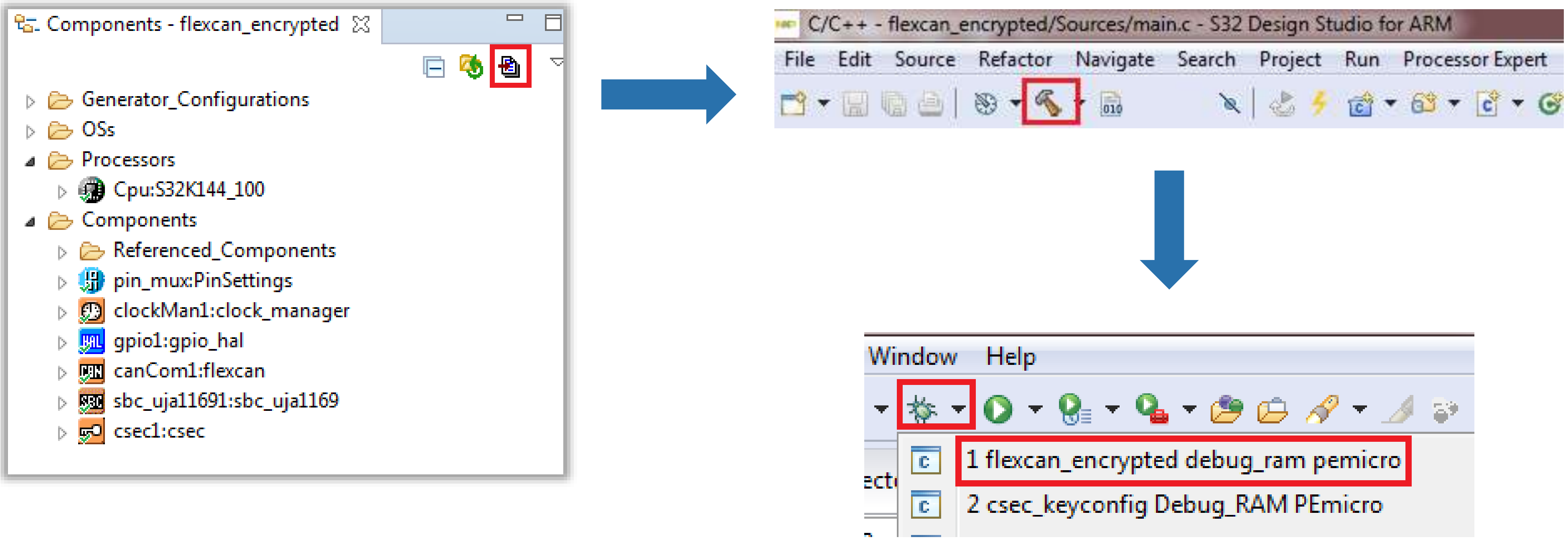
- The main.c file contains the application code
- MASTER/SLAVE macros must be defined appropriately



```
/* Use this define to specify if the application runs as master or slave */
#define MASTER
/* #define SLAVE */

/* Definition of the TX and RX message buffers depending on the bus role */
#if defined(MASTER)
    #define TX_MAILBOX  (1UL)
    #define TX_MSG_ID   (1UL)
    #define RX_MAILBOX  (0UL)
    #define RX_MSG_ID   (2UL)
#elif defined(SLAVE)
    #define TX_MAILBOX  (0UL)
    #define TX_MSG_ID   (2UL)
    #define RX_MAILBOX  (1UL)
    #define RX_MSG_ID   (1UL)
#endif
```

S32K144 Secured CAN: Build and debug

- Press the generate code button
- Build the application
- Debug on target



Components - flexcan_encrypted

- Generator_Configurations
- OSs
- Processors
  - Cpu:S32K144_100
- Components
  - Referenced_Components
  - pin_mux:PinSettings
  - clockMan1:clock_manager
  - gpio1:gpio_hal
  - canCom1:flexcan
  - sbc_uja11691:sbc_uja1169
  - csec1:csec

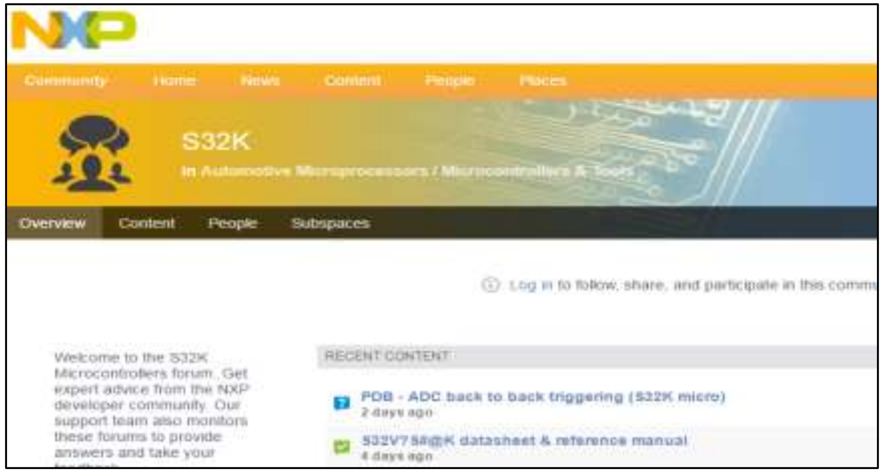
C/C++ - flexcan_encrypted/Sources/main.c - S32 Design Studio for ARM

1 flexcan_encrypted debug_ram pemicro

2 csec_keyconfig Debug_RAM PEmicro

S32K Technical Support – Communities https://community.nxp.com

- **S32K Community**
  - https://community.nxp.com/community/s32/s32k
  - Note: Includes SDK related topics



Log in to follow, share, and participate in this comm

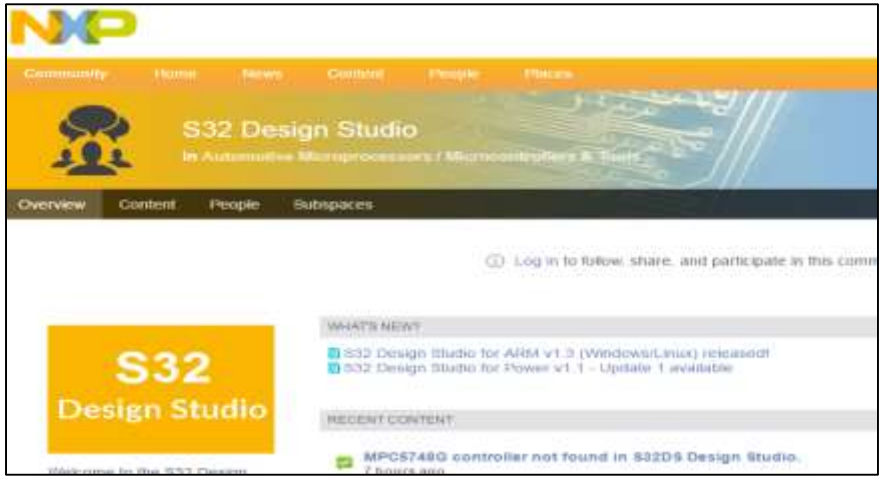
Welcome to the S32K Microcontrollers forum. Get expert advice from the NXP developer community. Our support team also monitors these forums to provide answers and take your

RECENT CONTENT

PDB - ADC back to back triggering (S32K micro)
2 days ago

S32V7 58@K datasheet & reference manual
4 days ago

- **S32_Design_Studio IDE Community**
  - https://community.nxp.com/community/s32/s32ds
  - Includes S32DS related topics



Log in to follow, share, and participate in this comm

WHAT'S NEW?

S32 Design Studio for ARM v1.3 (Windows/Linux) released!
S32 Design Studio for Power v1.1 - Update 1 available

RECENT CONTENT

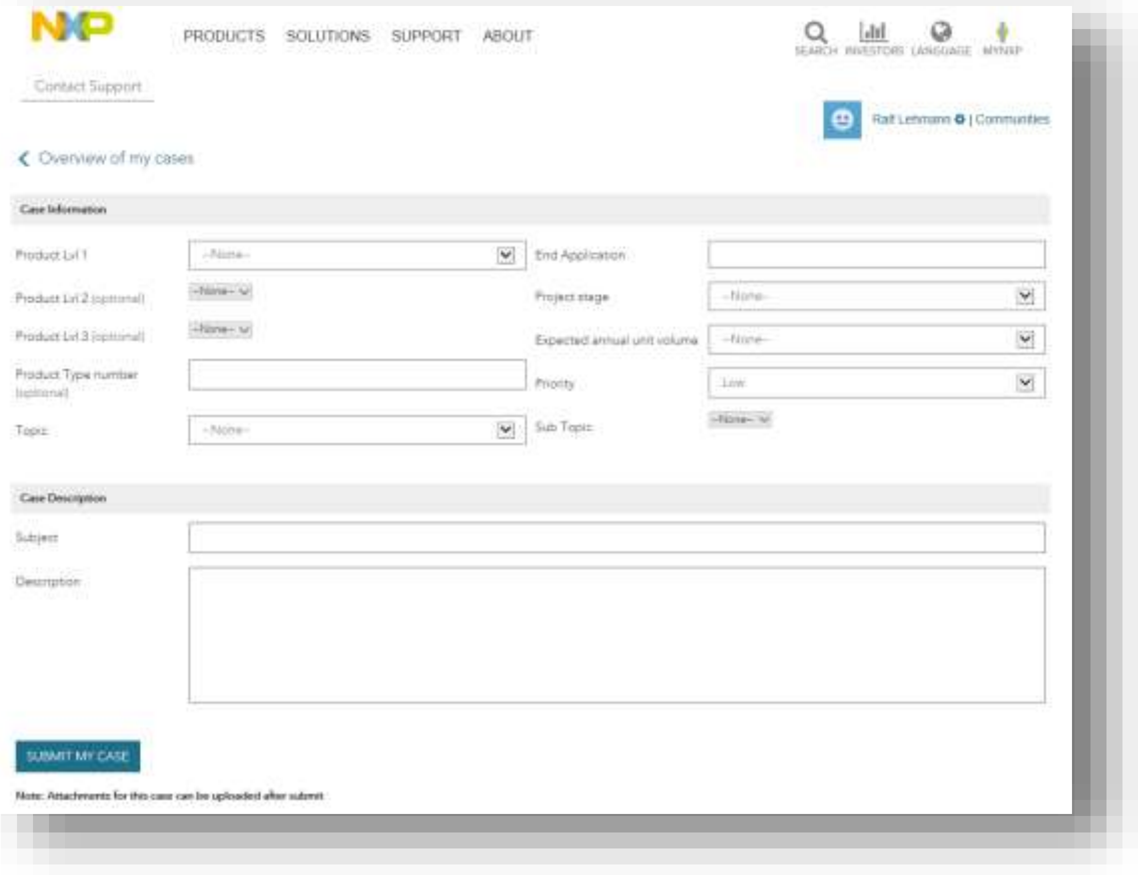
MPC5748G controller not found in S32DS Design Studio.
7 hours ago

S32K Technical Support – NXP Support Ticket / TIC (Technical Information Center)

- http://nxpcommunity.force.com/community/CommunityContactSupport

- Log-in with your NXP Communities username and password
  - If new user, please register. If no verification email is received, please check your spam folder. Email is sent from engineers.corner@nxp.com

- Enter your support CASE
  - All fields are mandatory



Case Information

Product Lvl 1	--None--	End Application	
Product Lvl 2 (optional)	--None--	Project stage	--None--
Product Lvl 3 (optional)	--None--	Expected annual unit volume	--None--
Product Type number (optional)		Priority	Low
Topic	--None--	Sub Topic	--None--

Case Description

Subject	
Description	

Note: Attachments for this case can be uploaded after submit

Thank you

nxp.com/S32K

05.

Q&A

SECURE CONNECTIONS
FOR A SMARTER WORLD