S32K148 EVB

QUICK START GUIDE

REV1

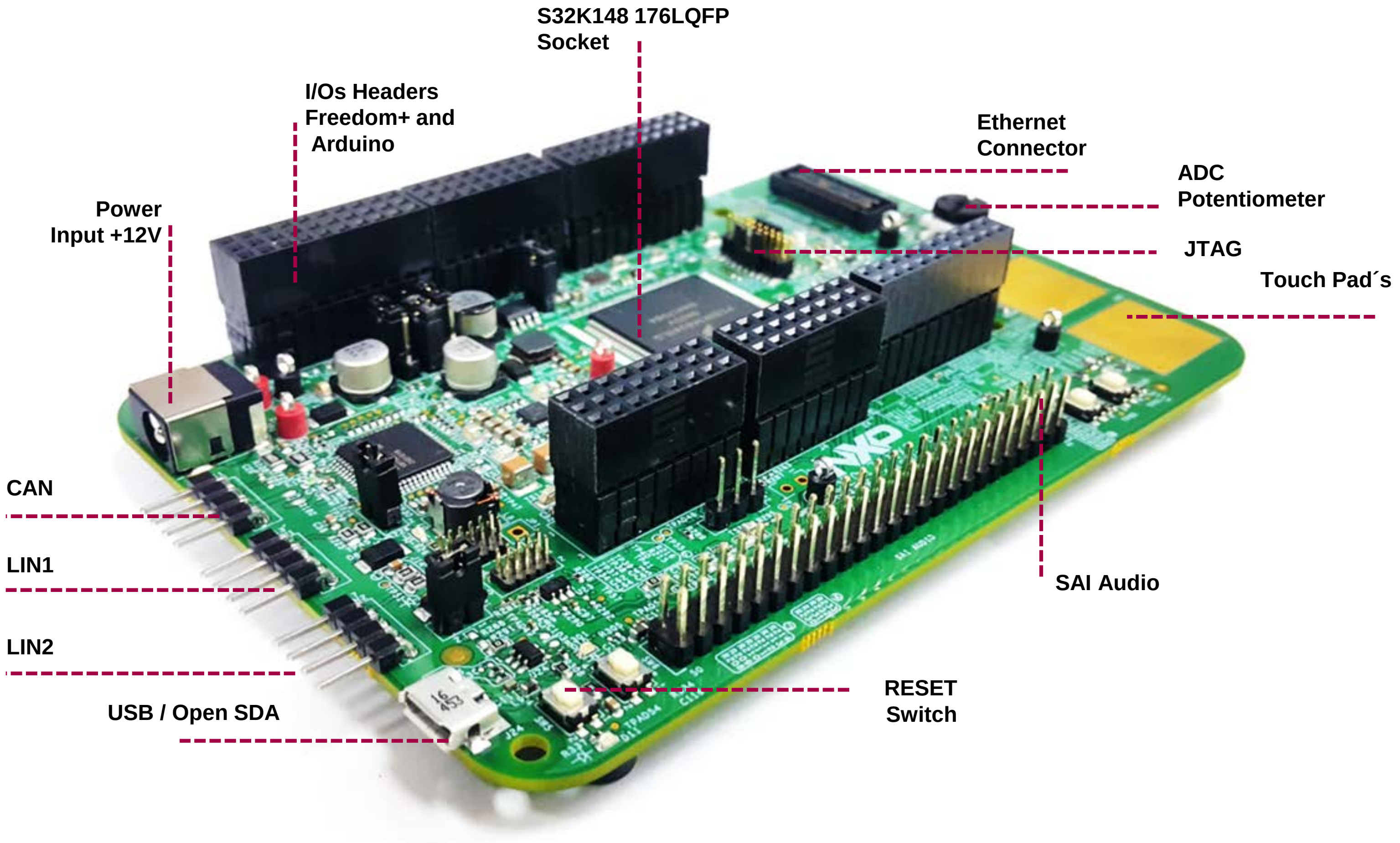
APPLIES FOR: S32K148 EVB (SCH-29644 REV A/B)

EXTERNAL USE

SECURE CONNECTIONS FOR A SMARTER WORLD

Contents:

Get to know the S32K148EVB

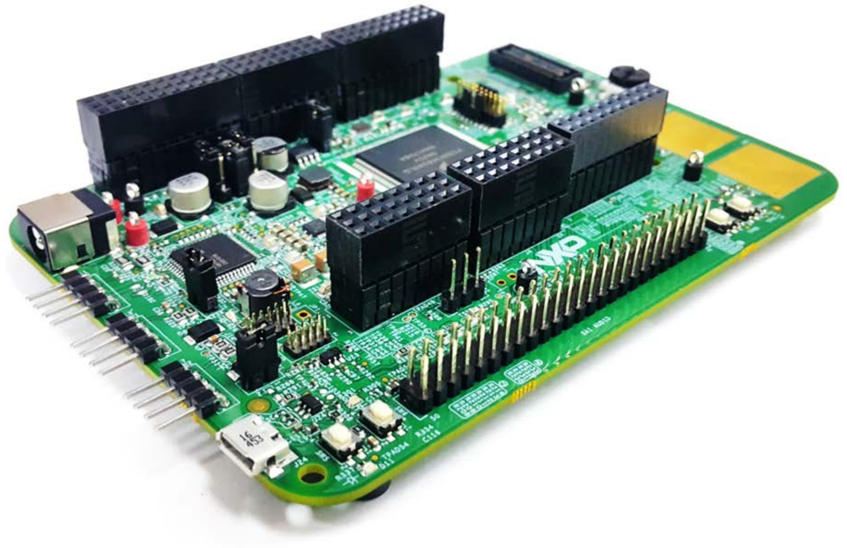


The **S32K148EVB** is a development platform for S32K Microcontrollers.

Features include easy access to all MCU I/O´s, a standard-based form factor compatible with the Arduino™ pin layout, providing a broad range of expansion board options, and an USB serial port interface for connection to the IDE, the board has option to be powered via USB or an external power supply.

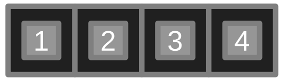
S32K148 EVB Features:

- Supports **S32K148 176LQFP**
- Arduino™ UNO footprint-compatible with expansion “shield” support
- Integrated open-standard serial and debug adapter (OpenSDA) with support for several industry-standard debug interfaces
- Easy access to the MCU I/O header pins for prototyping
- On-chip connectivity for CAN, LIN, UART/SCI.
- SBC UJA1132 with 2 LIN physical layers and 1 CAN physical layer
- Potentiometer for precise voltage and analog measurement
- RGB LED
- Two push-button switches (SW2 and SW3) and two touch electrodes
- External flash memory MX25L6433F on board
- Ethernet connector compatible with different ethernet daughter carts
- Voltage supply options for 3.3v or 5v.
- Flexible power supply options
  - microUSB or
  - external 12V power supply



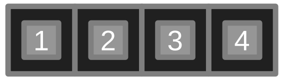
S32K148 EVB Features: CAN and LIN connectors

J11



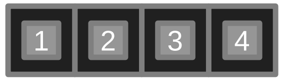
1. CANH
2 .CANL
3. VBAT [by 0 RESISTOR - DNP]
4. GND

J13



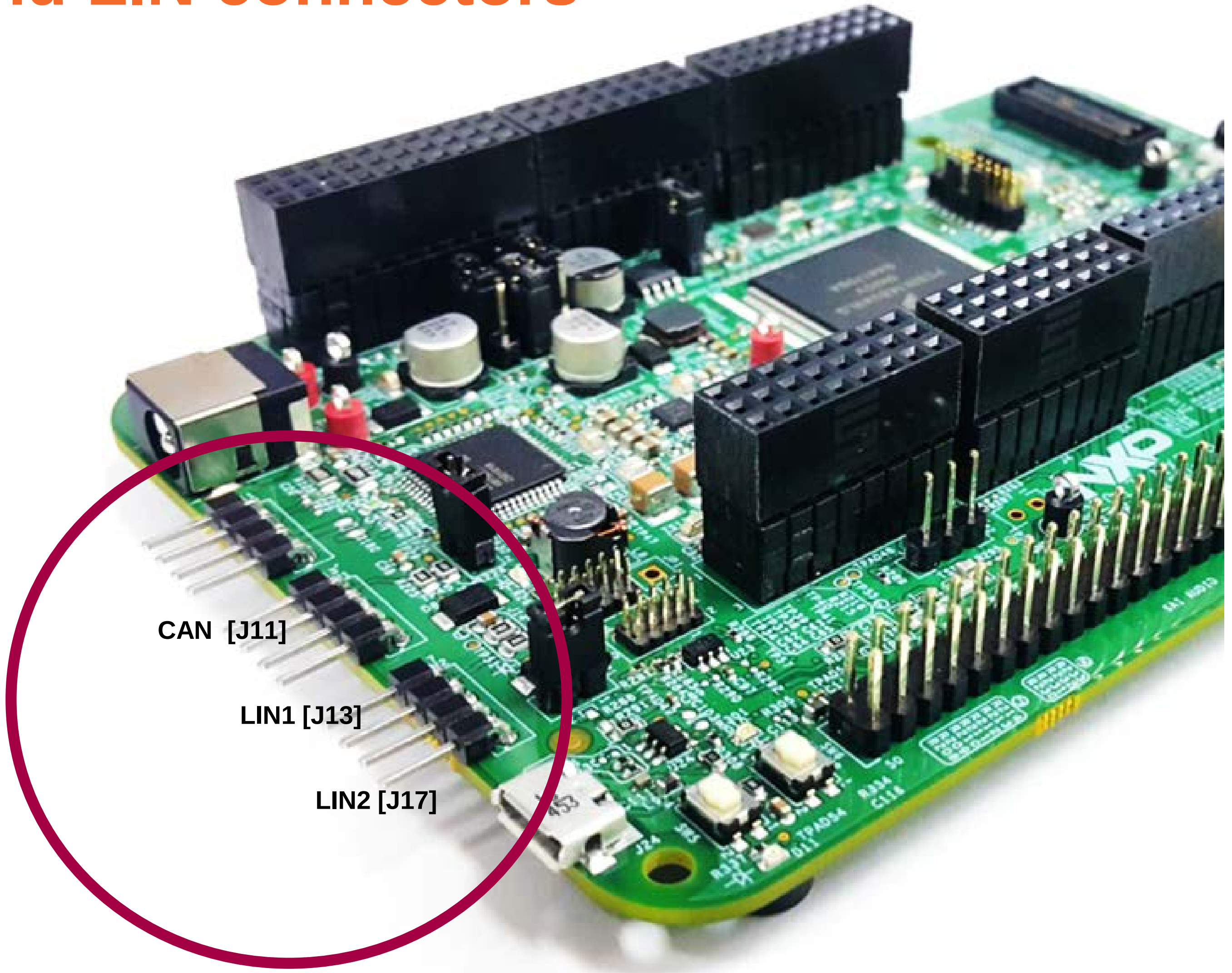
1. LIN1
2 .VBAT
3. NC
4. GND

J17



1. LIN2
2 .VBAT
3. NC
4. GND

Front view



S32K148 EVB Features: USB/OpenSDA

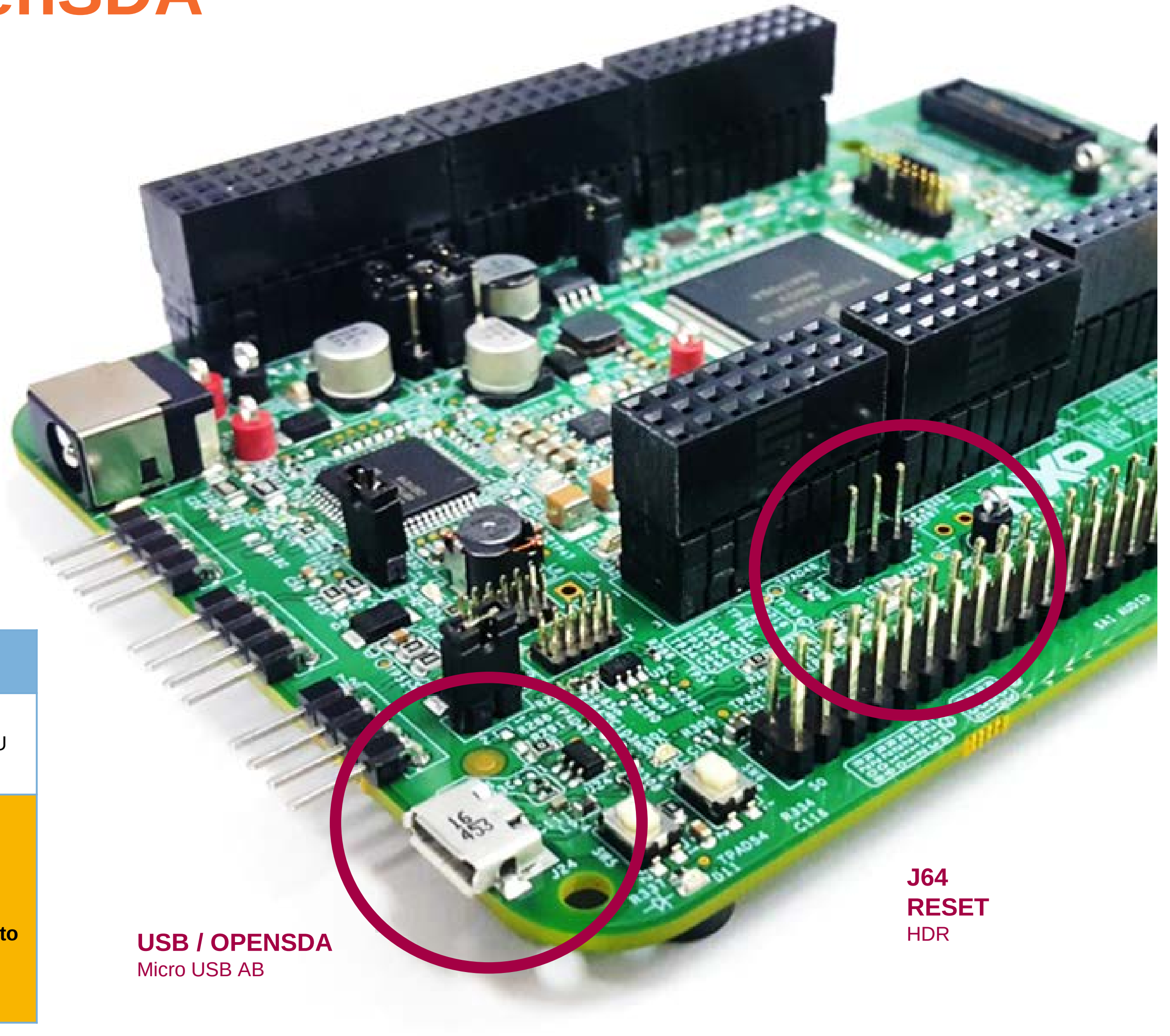
OpenSDA is a serial and debug adapter that is built into several NXP® evaluation boards. It provides a bridge between your computer (or other USB host) and the embedded target processor, which can be used for debugging, flash programming, and serial communication, all over a simple USB cable.

The OpenSDA hardware consists of a circuit featuring a Kinetis® K2x microcontroller with an integrated USB controller. On the software side, it implements a mass storage device bootloader which offers a quick and easy way to load OpenSDA applications such as flash programmers, run-control debug interfaces, serial to USB converters, and more.

REFERENCE	POSITION		DESCRIPTION
J64	1-2	1 2 3	**RESET** switch is routed RST MCU

ONLY for RevA

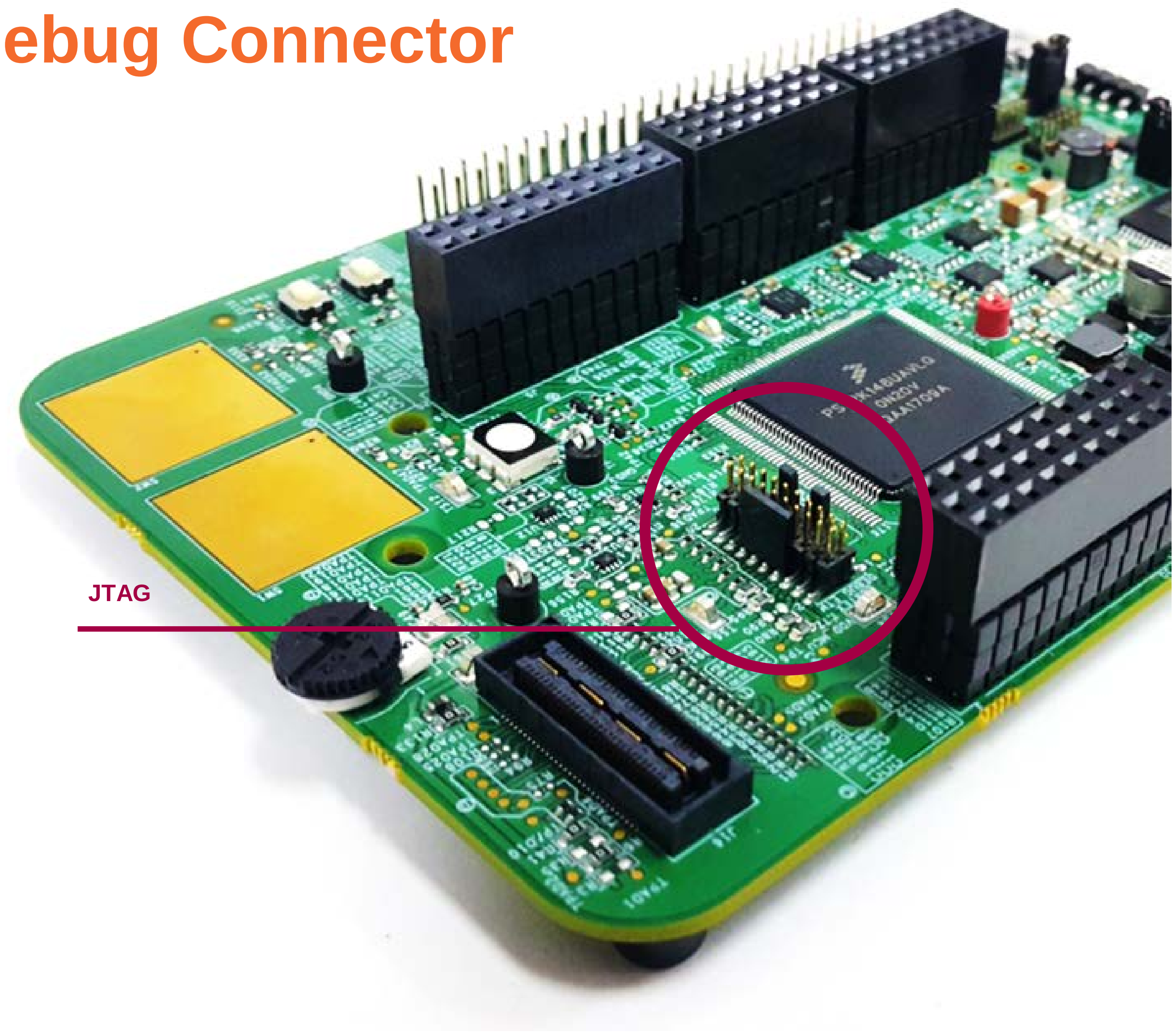
- **R537 must be removed when the S32K148 EVB is powered only via USB/OPEN SDA**
- **In order to enable SPI communication with the SBC UJA1132. R177 and R154 must be removed and to swpie [PTA29/FTM5_CH4/LPUART2_TX/LPSPI1_SIN_LS] and [PTA27/FTM5_CH2/LPSPI1_SOUT/LPUART0_TX_LS] by external wires.**



S32K148 EVB Features: JTAG Debug Connector

The following table shows the pinout of the debug connector used on the **S32K148EVB-Q144/Q176**

Signal	Pin	Pin	Signal
JTG_PWR	1	2	JTAG_TMS
GND	3	4	JTAG_TCLK
GND	5	6	JTAG_TDO
NC	7	8	JTAG_TDI
NC	9	10	JTAG_RESET
JTG_PWR	11	12	TRACE_CLK
JTG_PWR	13	14	TRACE_D0
GND	15	16	TRACE_D1
GND	17	18	TRACE_D2
GND	19	20	TRACE_D3



Jumper Settings

Jumper	Configuration	Description
J18	1-2 (Default) 2-3	VBAT(+12V) is routed to the input of the 3V3 switching power supply USB power (+5v) is routed to the input of the 3V3 switching power supply
J12	1-2 (Default)	LIN master option enabled for LIN1
J21	1-2 (Default)	LIN master option enabled for LIN2
J7	1-2 2-3 (Default)	MCU VDD domain is connected to 3.3v MCU VDD domain is connected to 5v
J8	1-2 (Default) 2-3	5V domain powered by 12V power source 5V domain powered by USB micro connector.
J22	1-2 (Default) 2-3	Reset switch is routed to MCU reset line Reset switch is routed to openSDA reset line.
J19	1-2 (Default)	VDD is routed to VDD_MCU domain (remove in order to measure the MCU current)

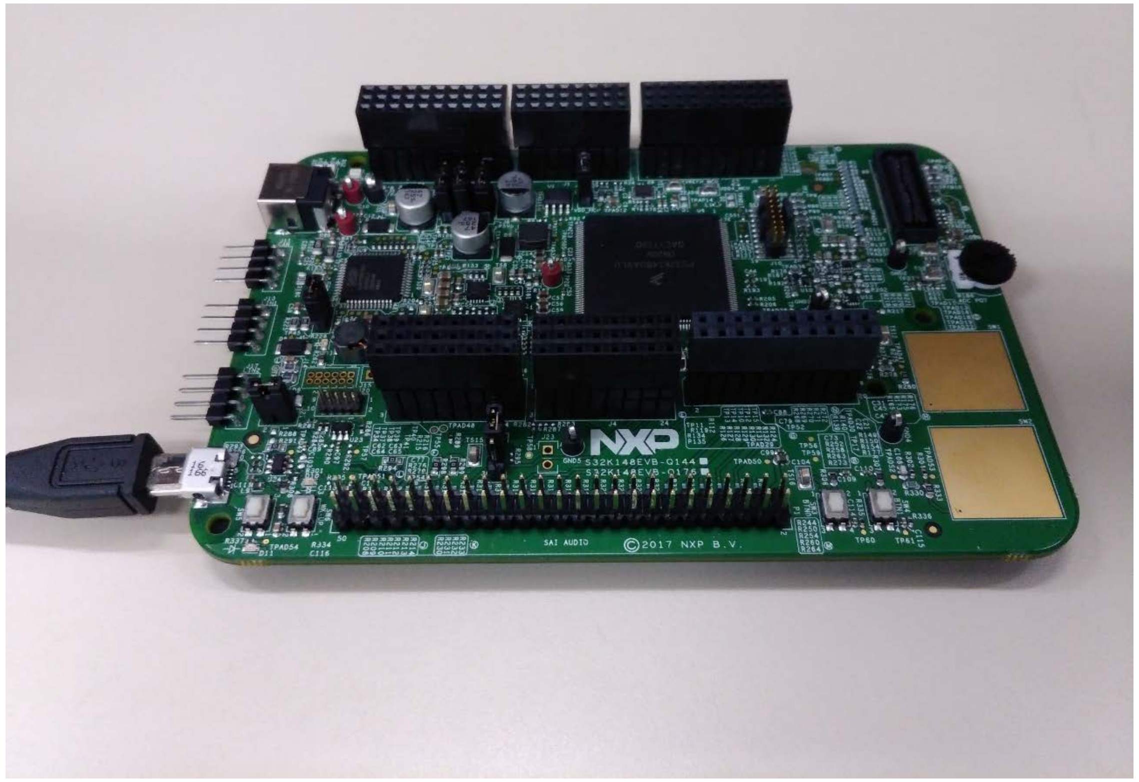
HMI mapping

Component	S32K148
Red LED	PTE21
Blue LED	PTE23
Green LED	PTE22
Potentiometer	PTC28
SW3	PTC12
SW4	PTC13
OpenSDA UART TX	PTC7 (LPUART1_TX)
OpenSDA UART RX	PTC6(LPUART1_RX)
CAN TX	PTE5(CAN0_TX)
CAN RX	PTE4 (CAN0_RX)
LIN1 TX	PTA3(LPUART0_TX)
LIN1 RX	PTA2 (LPUART0_RX)
LIN2 TX	PTA9(LPUART2_TX)
LIN2 RX	PTA8 (LPUART2_RX)
SBC_SCK	PTA28 (LPSPI1_SCK)
SBC_MISO	PTA29(LPSPI1_SIN)
SBC_MOSI	PTA27(LPSPI1_SOUT)
SBC_CS	PTA26(LPSPI1_PCS0)

S32K148 EVB
OUT OF THE BOX

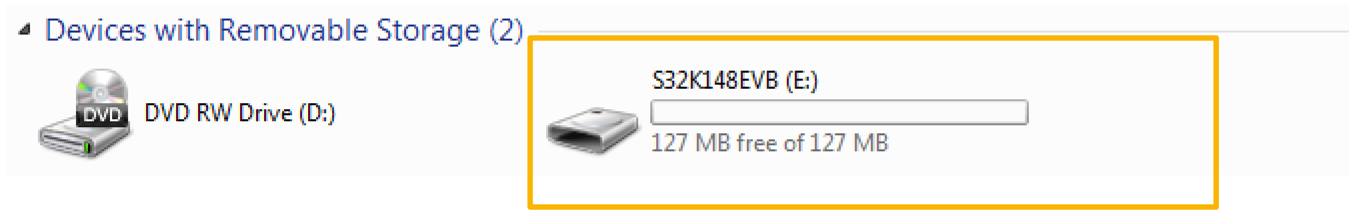
Step 1: Power up the Board – EVB Power Supplies

- The S32K148-EVB evaluation board powers from a USB or external 12V power supply. By default USB power is enabled with J8 (check slide 8)
- Connect the USB cable to a PC using supplied USB cable .
- Connect other end of USB cable (microUSB) to mini-B port on S32K148EVB at J24
- Allow the PC to automatically configure the USB drivers if needed
- Debug is done using OpenSDA through J24



Step 1: Power up the Board – Is it powered on correctly?

- When powered through USB, LEDs D2 and D3 should light green
- Once the board is recognized, it should appear as a mass storage device in your PC with the name S32K148EVB

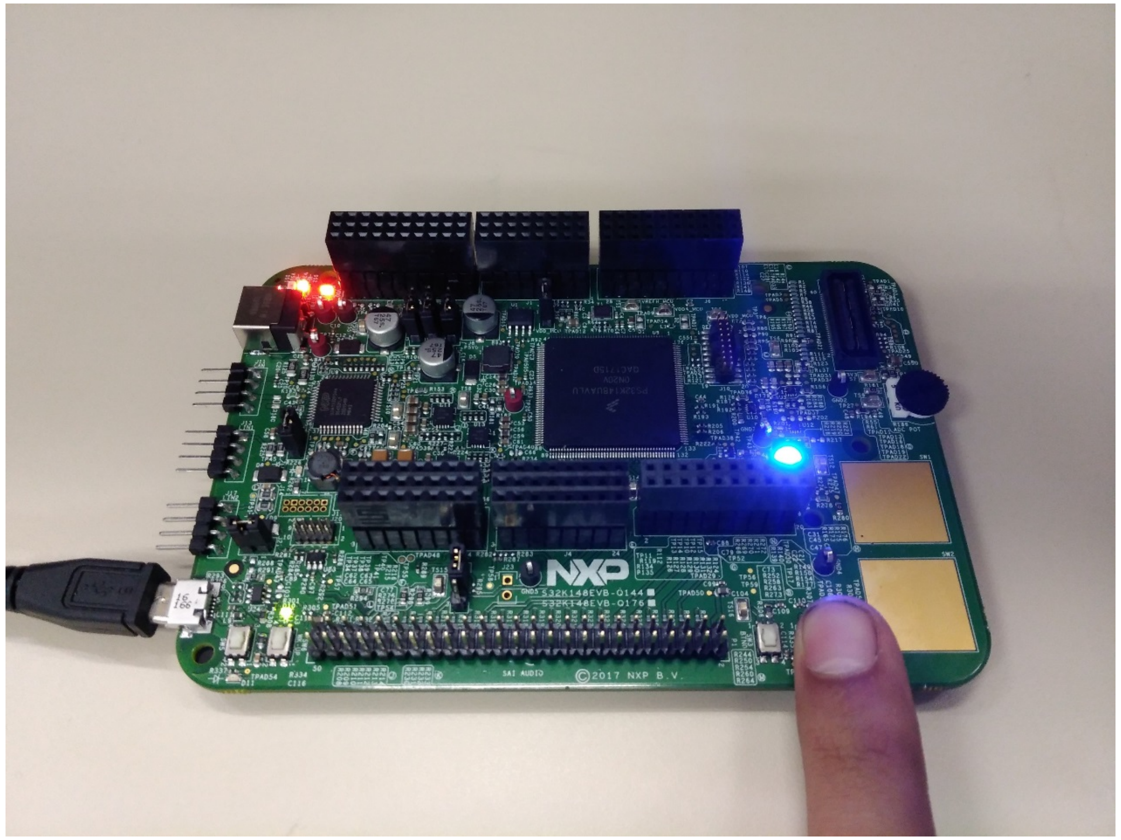


S32K148EVB (E:)

127 MB free of 127 MB

Step 1: Power up the Board – Is it powered on correctly?

- Board is preloaded with a software, in which if either of the SW3 or SW4 buttons is pressed the LED will turn on with different color.



INTRODUCTION TO OPENSDA

Introduction to OpenSDA: 1 of 2

OpenSDA is an open-standard serial and debug adapter. It bridges serial and debug communications between a USB host and an embedded target processor. OpenSDA software includes a flash-resident USB mass-storage device (MSD) bootloader and a collection of OpenSDA Applications. S32K144 EVB comes with the MSD Flash Programmer OpenSDA Application preinstalled. Follow these instructions to run the OpenSDA Bootloader and update or change the installed OpenSDA Application.

Enter OpenSDA Bootloader Mode

1. Unplug the USB cable if attached
2. Set J104 on position 1-2.
3. Press and hold the Reset button (SW5)
4. Plug in a USB cable (not included) between a USB host and the OpenSDA USB connector (labeled “SDA”)
5. Release the Reset button

A removable drive should now be visible in the host file system with a volume label of BOOTLOADER. You are now in OpenSDA Bootloader mode.

IMPORTANT NOTE: Follow the “Load an OpenSDA Application” instructions to update the MSD Flash Programmer on your S32K144 EVB to the latest version.

Load an OpenSDA Application

1. While in OpenSDA Bootloader mode, double-click ***SDA_INFO.HTML*** in the ***BOOTLOADER*** drive. A web browser will open the OpenSDA homepage containing the name and version of the installed Application. This information can also be read as text directly from ***SDA_INFO.HTML***
2. Locate the ***OpenSDA Applications***
3. Copy & paste or drag & drop the MSD Flash Programmer Application *to the* ***BOOTLOADER*** *drive*
4. Unplug the USB cable and plug it in again. The new OpenSDA Application should now be running and a ***S32K144 EVB*** drive should be visible in the host file system

You are now running the latest version of the MSD Flash Programmer. Use this same procedure to load other OpenSDA Applications.

Introduction to OpenSDA: 2 of 2

The MSD Flash Programmer is a composite USB application that provides a virtual serial port and an easy and convenient way to program applications into the KEA MCU. It emulates a FAT16 file system, appearing as a removable drive in the host file system with a volume label of EVB-S32K144. Raw binary and Motorola S-record files that are copied to the drive are programmed directly into the flash of the KEA and executed automatically. The virtual serial port enumerates as a standard serial port device that can be opened with standard serial terminal applications.

Using the MSD Flash Programmer

1. Locate the .srec file of your project , file is under the Debug folder of the S32DS project.
2. Copy & paste or drag & drop one of the .srec files to the EVB-S32K144 drive

The new application should now be running on the S32K144 EVB. Starting with v1.03 of the MSD Flash Programmer, you can program repeatedly without the need to unplug and reattach the USB cable before reprogramming.

Drag one of the .srec code for the S32K144 the S32K144 EVB board over USB to reprogram the preloaded code example to another example.

NOTE: Flash programming with the MSD Flash Programmer is currently only supported on Windows operating systems. However, the virtual serial port has been successfully tested on Windows, Linux and Mac operating systems.

Using the Virtual Serial Port

1. Determine the symbolic name assigned to the EVB-S32K144 virtual serial port. In Windows open Device Manager and look for the COM port named “PEMicro/Freescale – CDC Serial Port”.
2. Open the serial terminal emulation program of your choice. Examples for Windows include Tera Term, PuTTY, and HyperTerminal
3. Press and release the Reset button (SW0) at anytime to restart the example application. Resetting the embedded application will not affect the connection of the virtual serial port to the terminal program.
4. It is possible to debug and communicate with the serial port at the same time, no need to stop the debug.

NOTE: Refer to the OpenSDA User’s Guide for a description of a known Windows issue when disconnecting a virtual serial port while the COM port is in use.

INSTALLING S32DS

Download S32DS

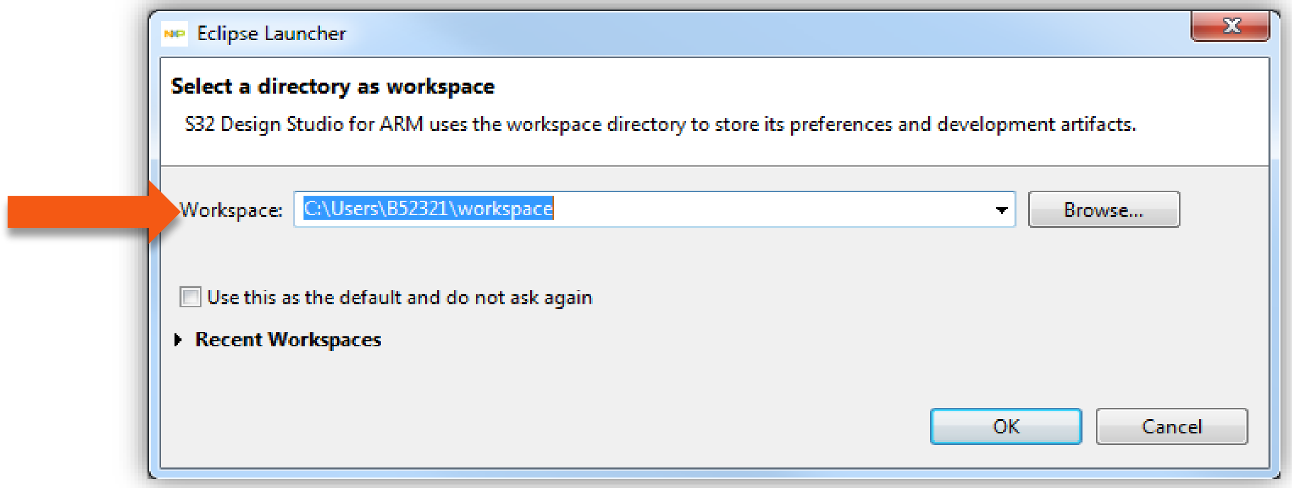
Download S32DS from ARM based MCUs from:

S32DS for ARM

CREATE A NEW PROJECT IN S32 DESIGN STUDIO

Create New Project: First Time – Select a Workspace

- Start program: Click on “S32 Design Studio for ARM” icon
- Select workspace:
  - Choose default (see below example) or specify new one
  - Suggestion: Uncheck the box “Use this as the default and do not ask again”
  - Click OK


Eclipse Launcher

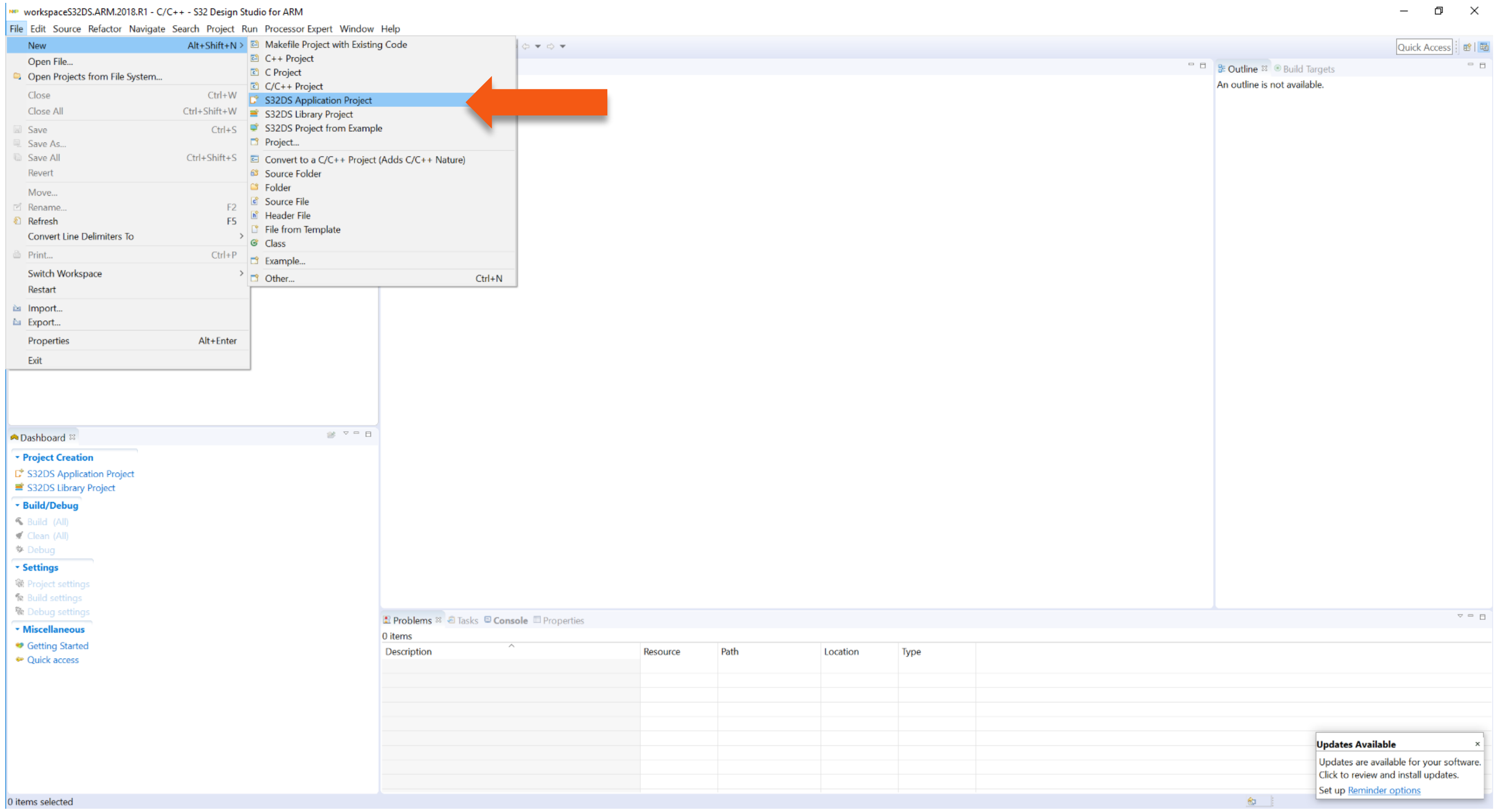
Select a directory as workspace

S32 Design Studio for ARM uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\Users\B52321\workspace

Create New Project: Top Menu Selection

- File – New –Project



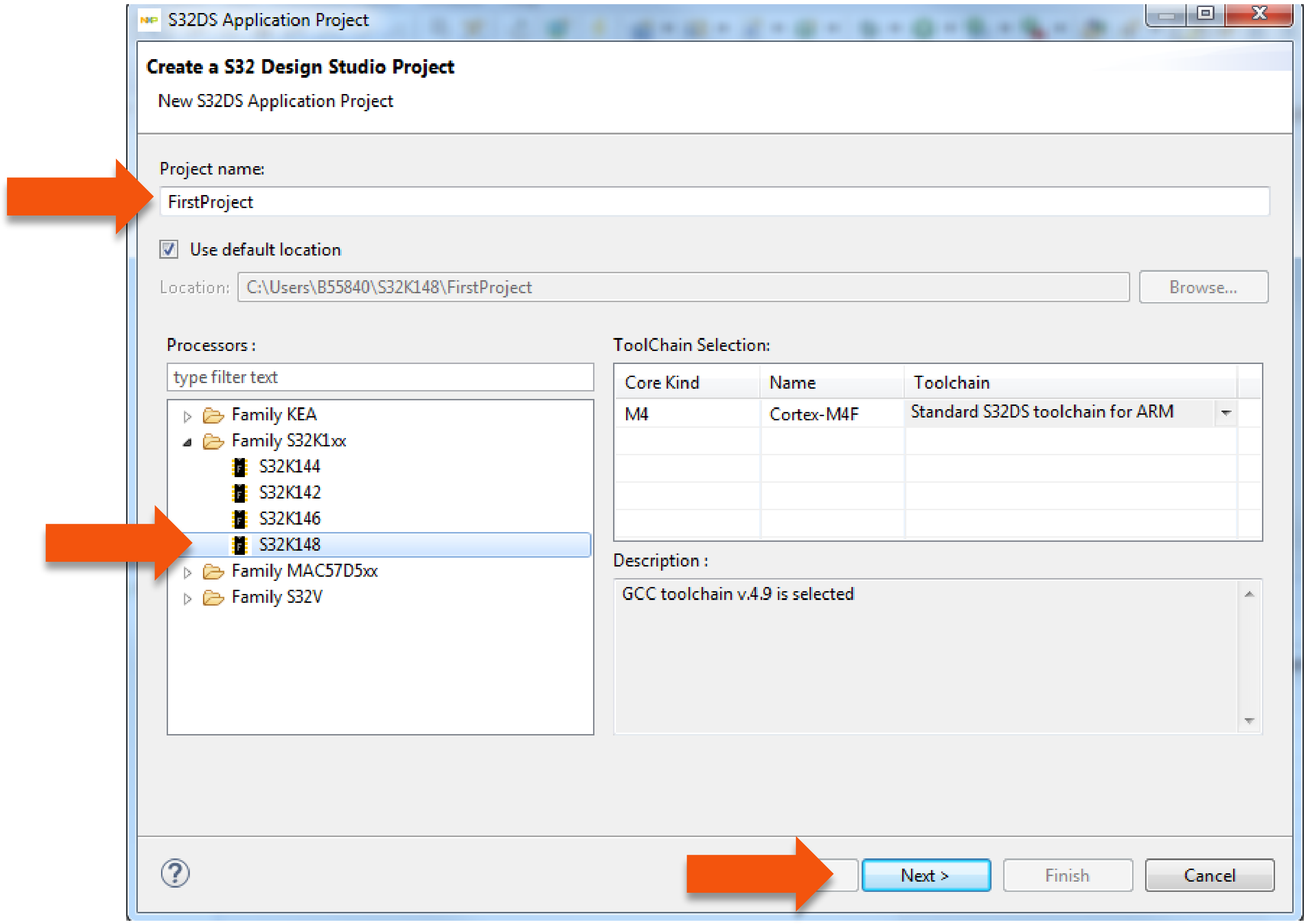
Updates Available

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

Set up Reminder options

Create New Project: S32DS Project

- Project Name:
  - Example: FirstProject
- Project Type:
  - Select from inside executable or library folder
- Next


S32DS Application Project

Create a S32 Design Studio Project

New S32DS Application Project

Project name: FirstProject

[✓] Use default location

Location: C:\Users\B55840\S32K148\FirstProject

Processors :

type filter text

- Family KEA
- Family S32K1xx
  - S32K144
  - S32K142
  - S32K146
  - S32K148
- Family MAC57D5xx
- Family S32V

ToolChain Selection:

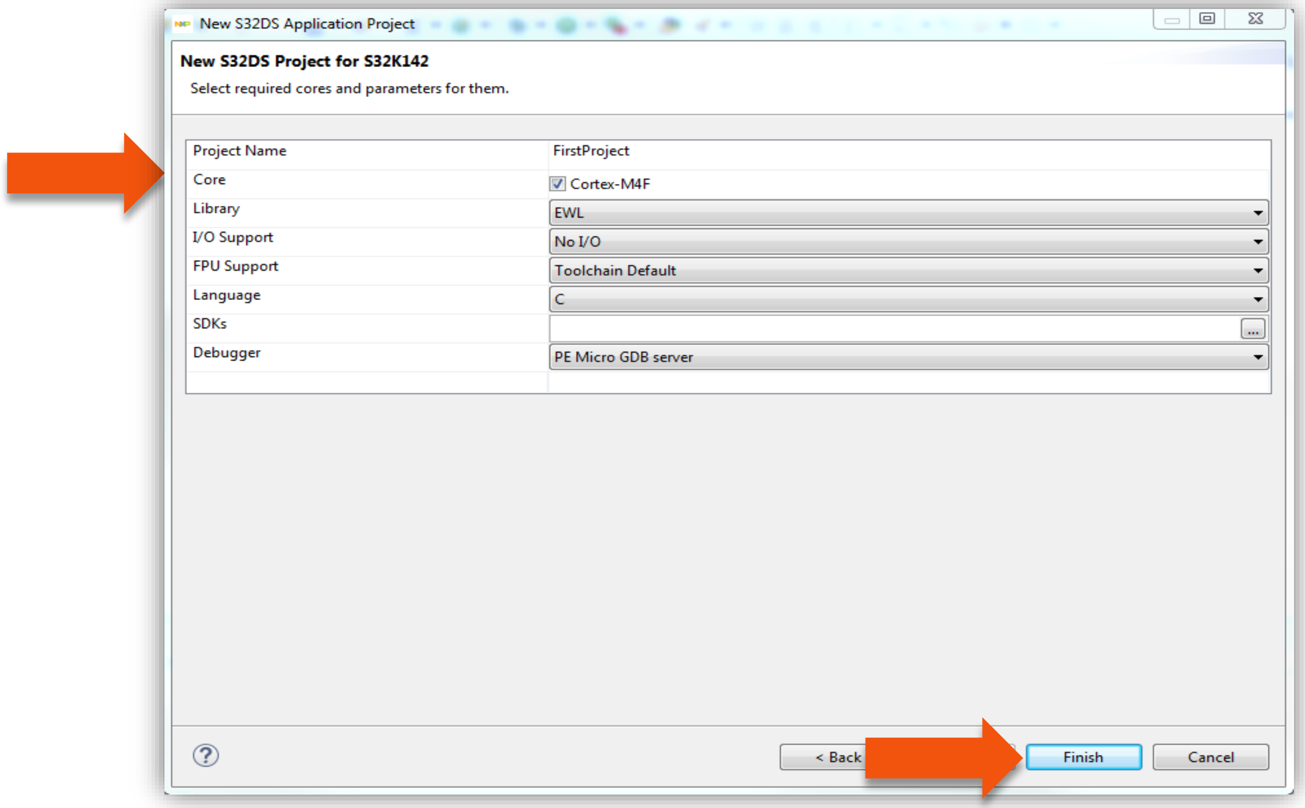
Core Kind	Name	Toolchain
M4	Cortex-M4F	Standard S32DS toolchain for ARM

Description :

GCC toolchain v.4.9 is selected

Create New Project: S32DS Project

- Select Debugger Support and Library Support
- Click Finish



New S32DS Application Project

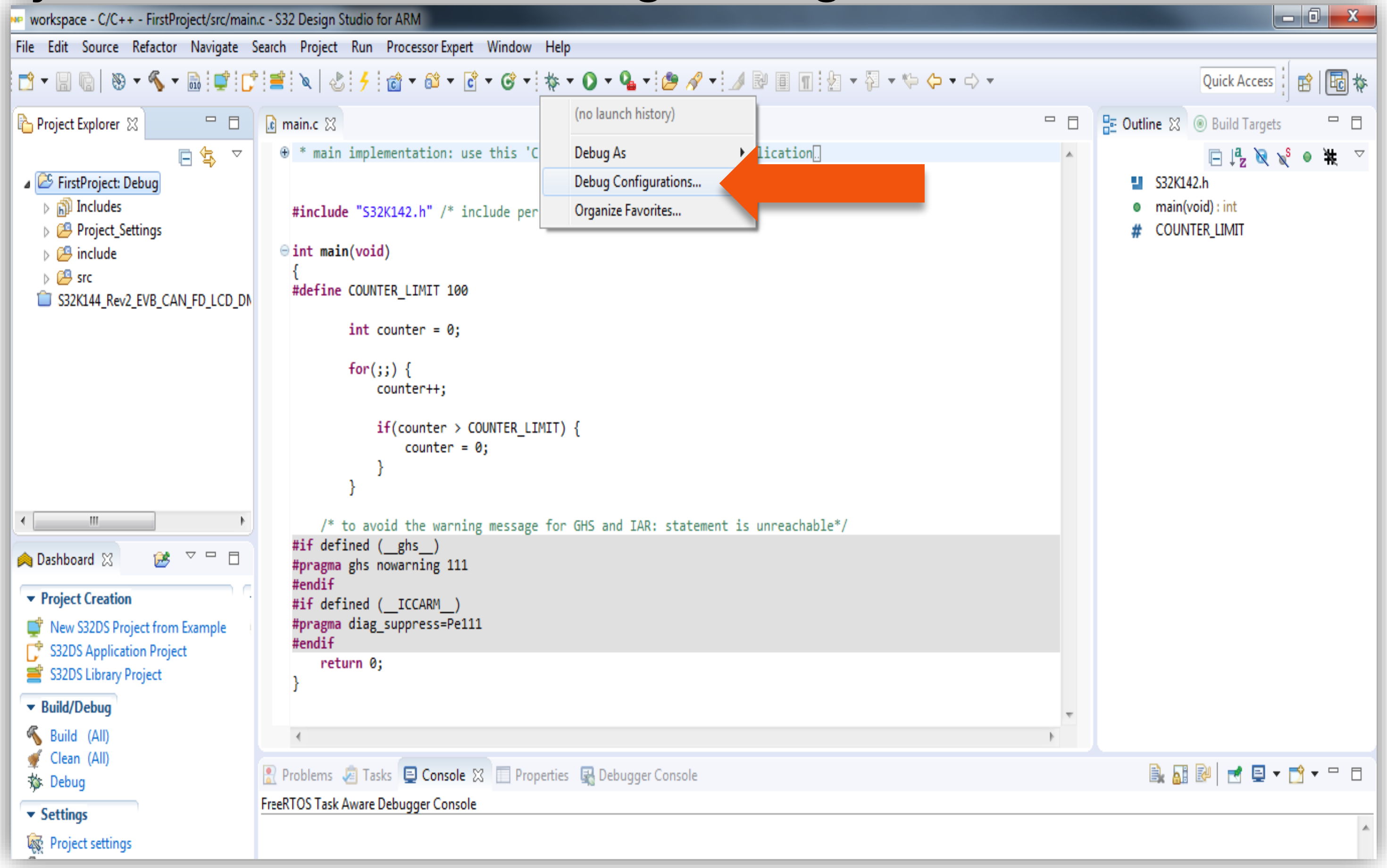
New S32DS Project for S32K142

Select required cores and parameters for them.

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

OpenSDA Configuration

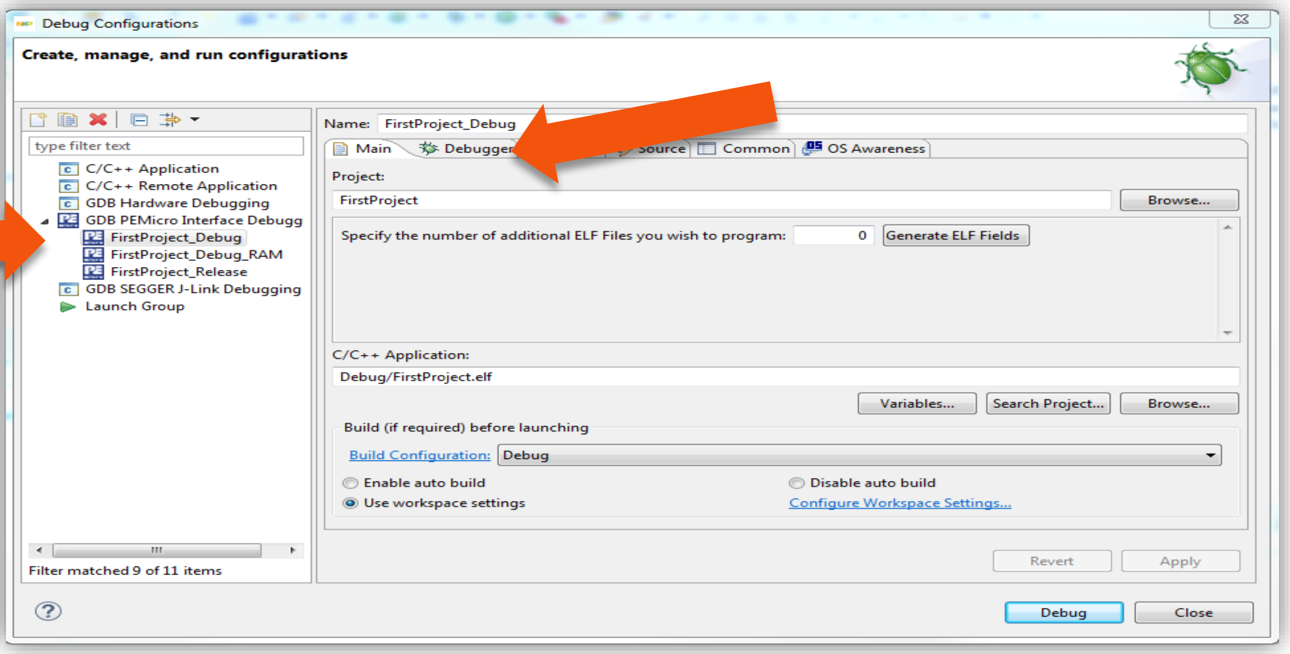
- To Debug your project with OpenSDA, it is necessary to select the OpenSDA in the Debug Configuration.
- Select your project, and click on debug configuration



```
* main implementation: use this 'C                          lication
#include "S32K142.h" /* include per

int main(void)
{
#define COUNTER_LIMIT 100

        int counter = 0;

        for(;;) {
            counter++;

            if(counter > COUNTER_LIMIT) {
                counter = 0;
            }
        }

    /* to avoid the warning message for GHS and IAR: statement is unreachable*/
#if defined (__ghs__)
#pragma ghs nowarning 111
#endif
#if defined (__ICCARM__)
#pragma diag_suppress=Pe111
#endif
    return 0;
}
```

OpenSDA Configuration

- Select the Debug configuration under GDB PEMicro Interface Debugging
- Click on Debugger tab



Name: FirstProject_Debug

Project: FirstProject

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

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

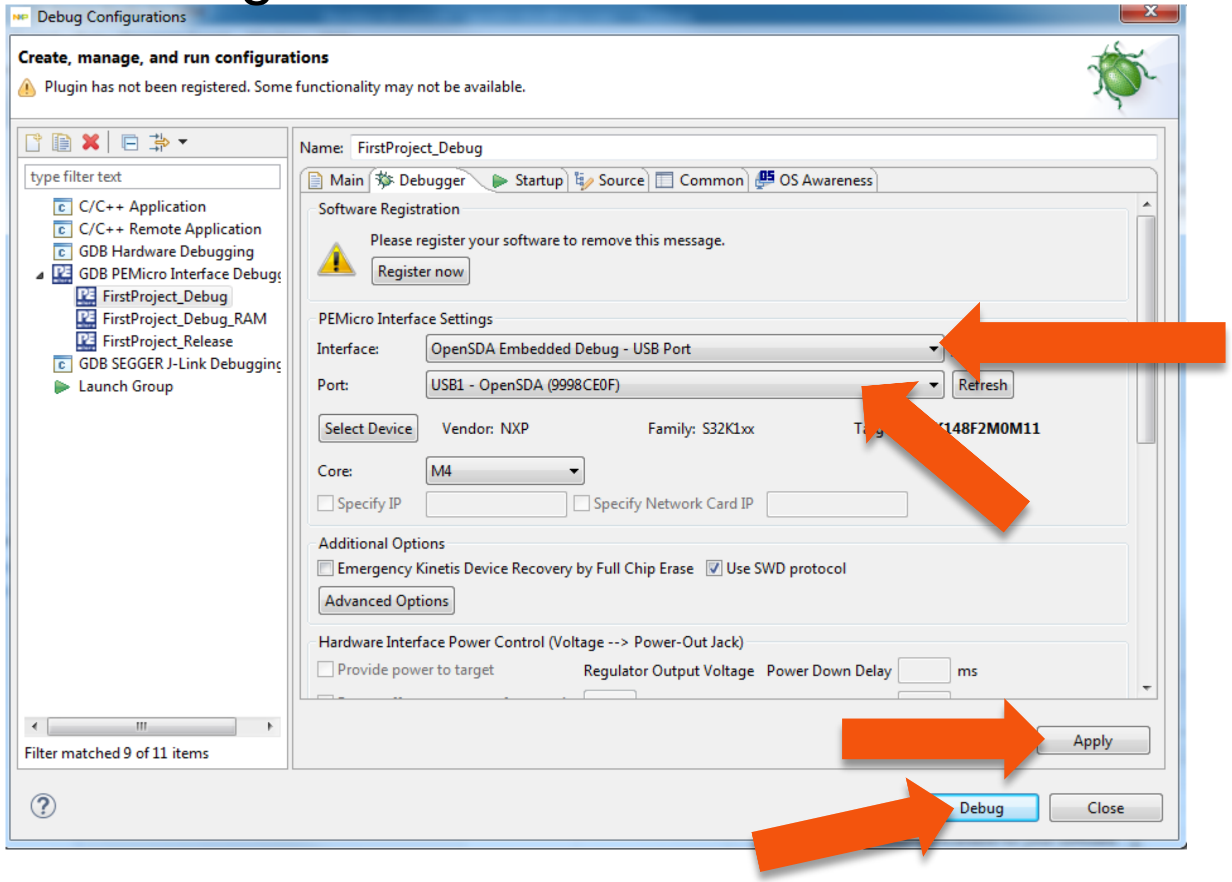
Build (if required) before launching

Build Configuration: Debug

( ) Enable auto build
(●) Use workspace settings
( ) Disable auto build
Configure Workspace Settings...

OpenSDA Configuration

- Select OpenSDA as the interface, if your board is plugged should appear in the Port field.
- Click Apply and debug to finish.



Debug Configurations

Create, manage, and run configurations

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

type filter text

- C/C++ Application
- C/C++ Remote Application
- GDB Hardware Debugging
- GDB PEMicro Interface Debugg
  - FirstProject_Debug
  - FirstProject_Debug_RAM
  - FirstProject_Release
- GDB SEGGER J-Link Debugging
- Launch Group

Filter matched 9 of 11 items

Name: FirstProject_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

Port: USB1 - OpenSDA (9998CE0F) Refresh

Select Device Vendor: NXP Family: S32K1xx Ta[illegible]148F2M0M11

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

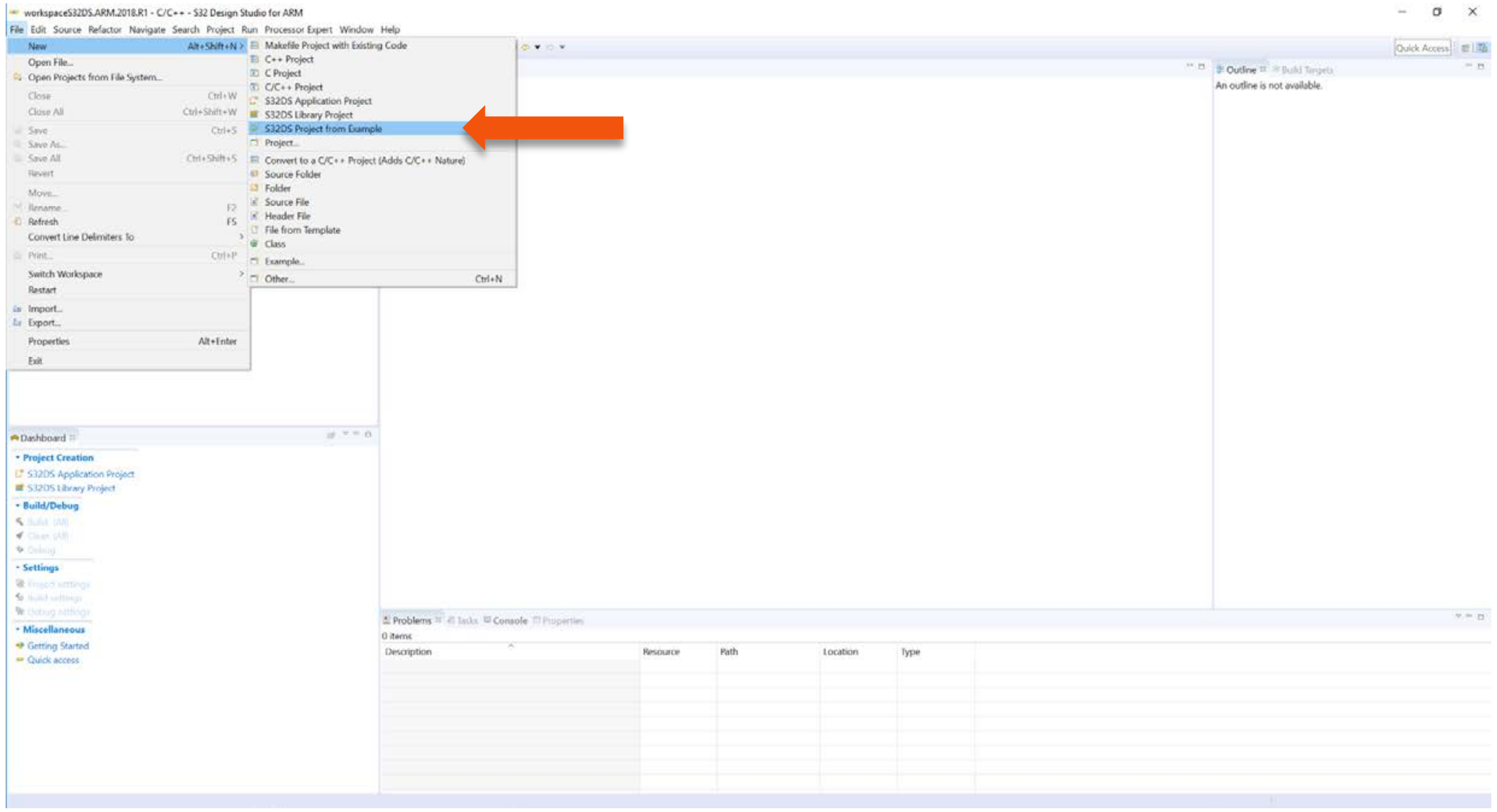
Apply

Debug Close

CREATE AN EXAMPLE FROM SDK

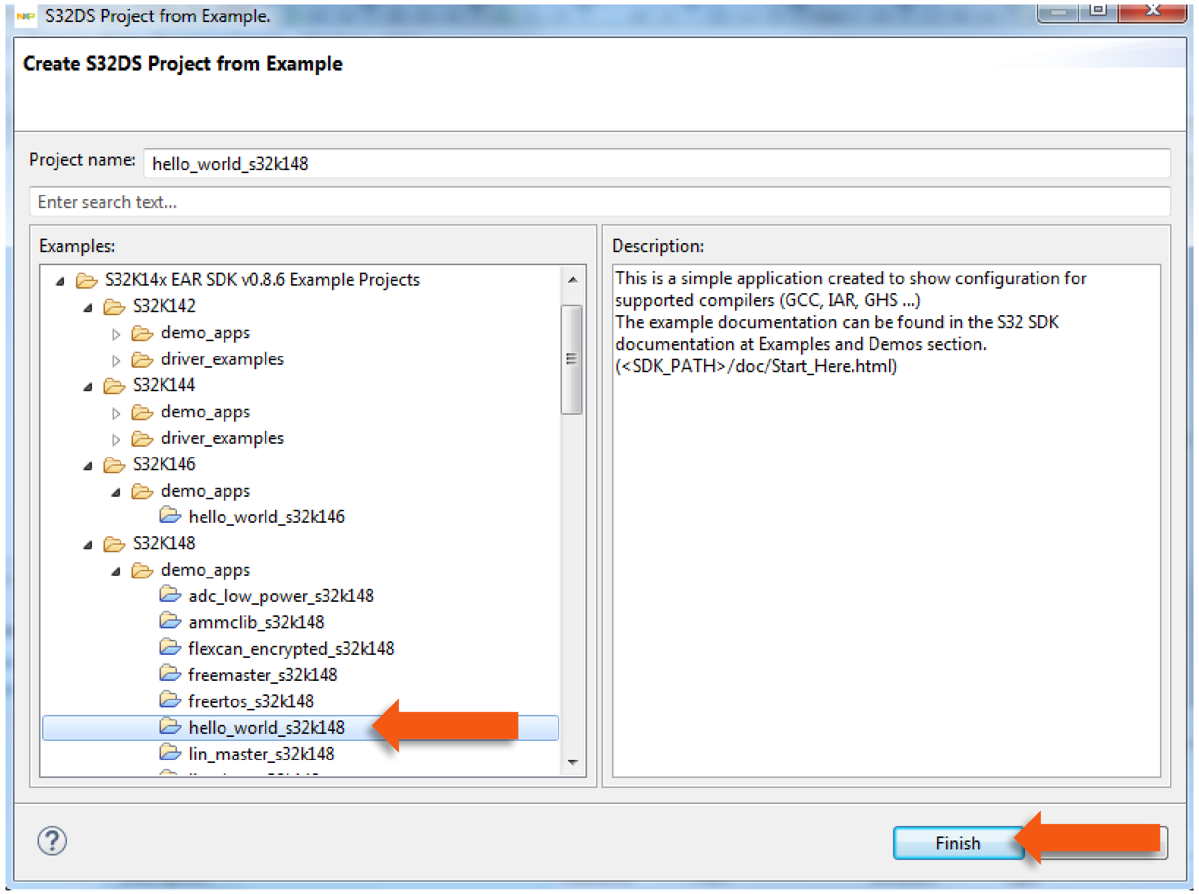
Creating example from SDK

- The S32 Design Studio IDE already includes the Software Development Kit for quickly develop applications on S32K1xx devices.
- To create a project using an example go to File – New – S32DS Project from Example



Creating example from SDK

- Go to the S32K14x EAR SDK v0.8.6 Example Projects section and select the example that wants to be used.
- In this example the hello_world is selected


S32DS Project from Example.

Create S32DS Project from Example

Project name: hello_world_s32k148

Enter search text...

Examples:

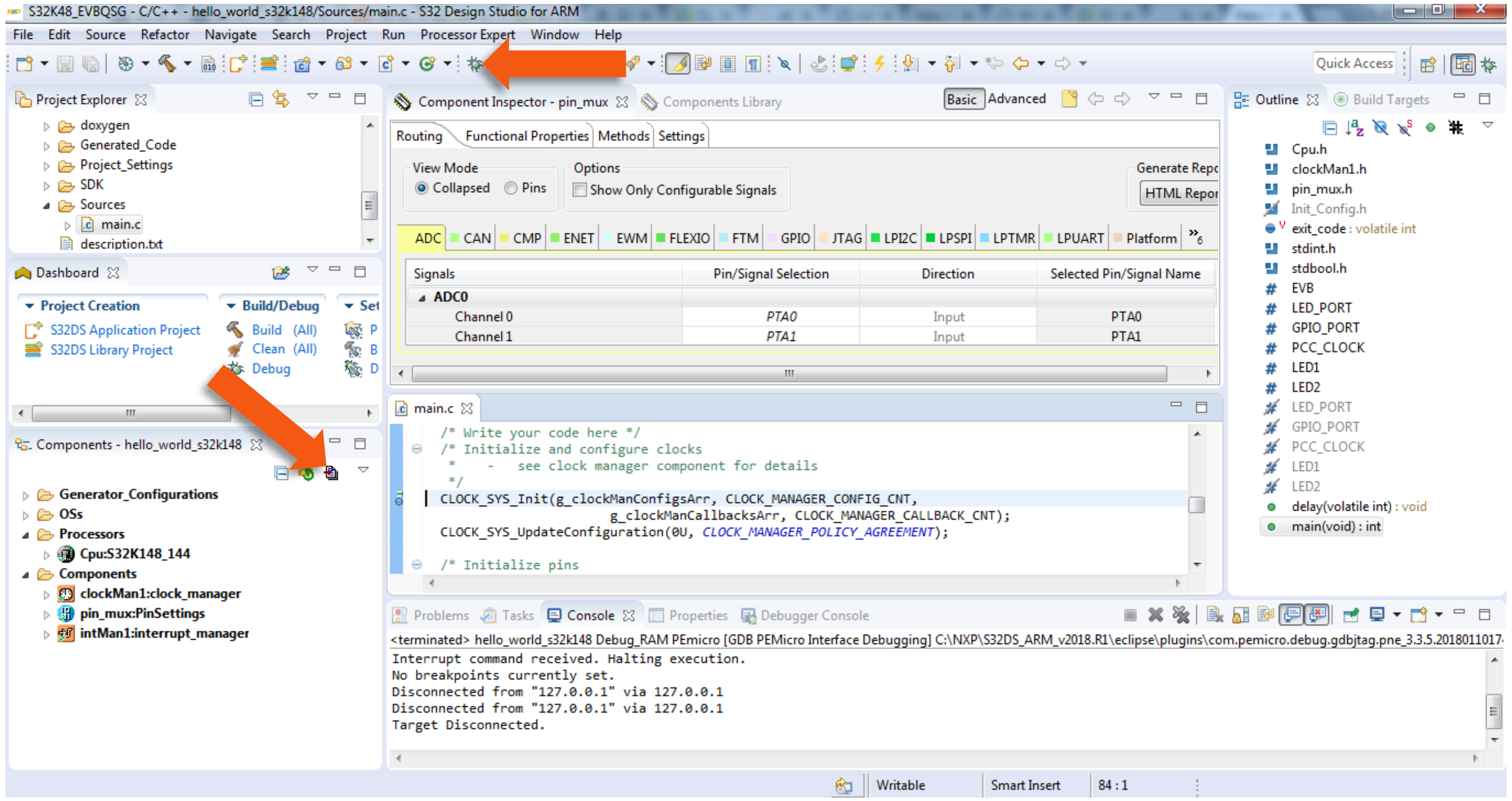
- S32K14x EAR SDK v0.8.6 Example Projects
  - S32K142
    - demo_apps
    - driver_examples
  - S32K144
    - demo_apps
    - driver_examples
  - S32K146
    - demo_apps
      - hello_world_s32k146
  - S32K148
    - demo_apps
      - adc_low_power_s32k148
      - ammclib_s32k148
      - flexcan_encrypted_s32k148
      - freemaster_s32k148
      - freertos_s32k148
      - hello_world_s32k148
      - lin_master_s32k148

Description:

This is a simple application created to show configuration for supported compilers (GCC, IAR, GHS ...)
The example documentation can be found in the S32 SDK documentation at Examples and Demos section.
(<SDK_PATH>/doc/Start_Here.html)

Creating example from SDK

- A new project would be created in the workspace. Then click on generate code icon and then on debug, as indicated.



Signals	Pin/Signal Selection	Direction	Selected Pin/Signal Name
ADC0			
Channel 0	PTA0	Input	PTA0
Channel 1	PTA1	Input	PTA1

main.c

```
/* Write your code here */
/* Initialize and configure clocks
 *    -   see clock manager component for details
 */
CLOCK_SYS_Init(g_clockManConfigsArr, CLOCK_MANAGER_CONFIG_CNT,
                    g_clockManCallbacksArr, CLOCK_MANAGER_CALLBACK_CNT);
CLOCK_SYS_UpdateConfiguration(0U, CLOCK_MANAGER_POLICY_AGREEMENT);

/* Initialize pins
```

```
<terminated> hello_world_s32k148 Debug_RAM PEmicro [GDB PEMicro Interface Debugging] C:\NXP\S32DS_ARM_v2018.R1\eclipse\plugins\com.pemicro.debug.gdbjtag.pne_3.3.5.2018011017
Interrupt command received. Halting execution.
No breakpoints currently set.
Disconnected from "127.0.0.1" via 127.0.0.1
Disconnected from "127.0.0.1" via 127.0.0.1
Target Disconnected.
```

- If run correctly, the LED should start blinking red and green.

Creating example from SDK

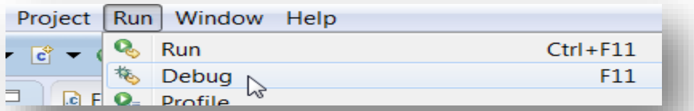
- The complete documentation of the SDK can be found in: C:\NXP\S32DS_ARM_v2018.R1\S32DS\S32SDK_S32K14x_EAR_0.8.6\doc\Start_here.html

- For more information about the use of the SDK go click on the following link for an SDK training (add hyperlink once if is online)

DEBUG BASICS

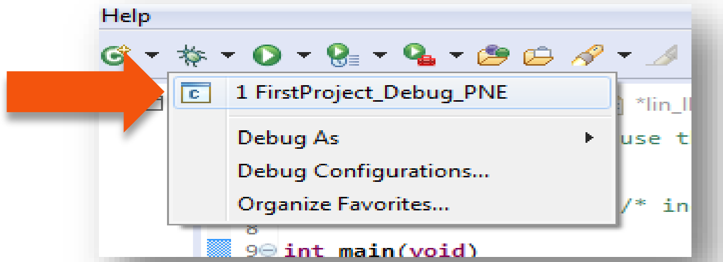
Debug Basics: Starting the Debugger

- Debug configuration is only required once. Subsequent starting of debugger does not require those steps.
- Three options to start debugger:
  - If the “Debug Configuration” has not been closed, click on “Debug” button on bottom right
  - Select Run – Debug (or hit F11)



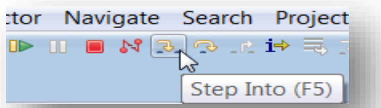
    Note: This method currently selects the desktop target (*project.*elf) and gives an error. Do not use until this is changed.

  - *Recommended Method*: Click on pull down arrow for bug icon and select …_debug.elf target

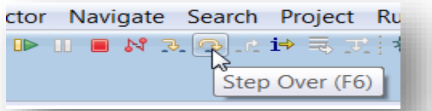


Debug Basics: Step, Run, Suspend, Resume

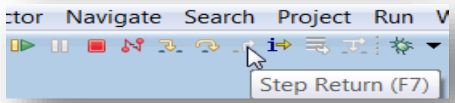
- Step Into (F5)



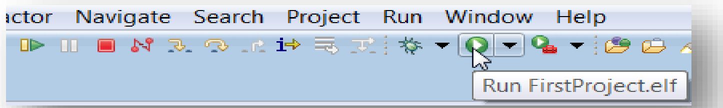
- Step Over (F6)



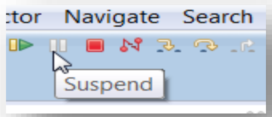
- Step Return (F7)



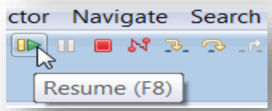
- Run



- Suspend

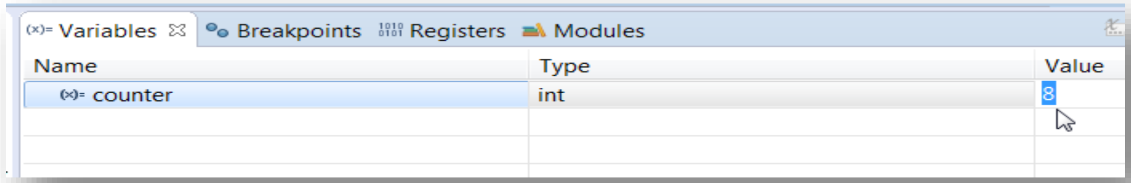


- Resume (F8)



Debug Basics: View & Alter Variables

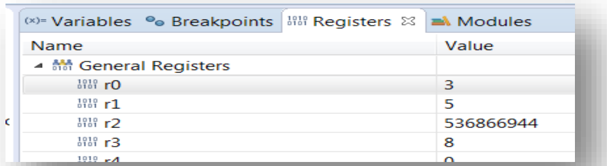
- View variables in “Variables” tab.
- Click on a value to allow typing in a different value.



Name	Type	Value
counter	int	8

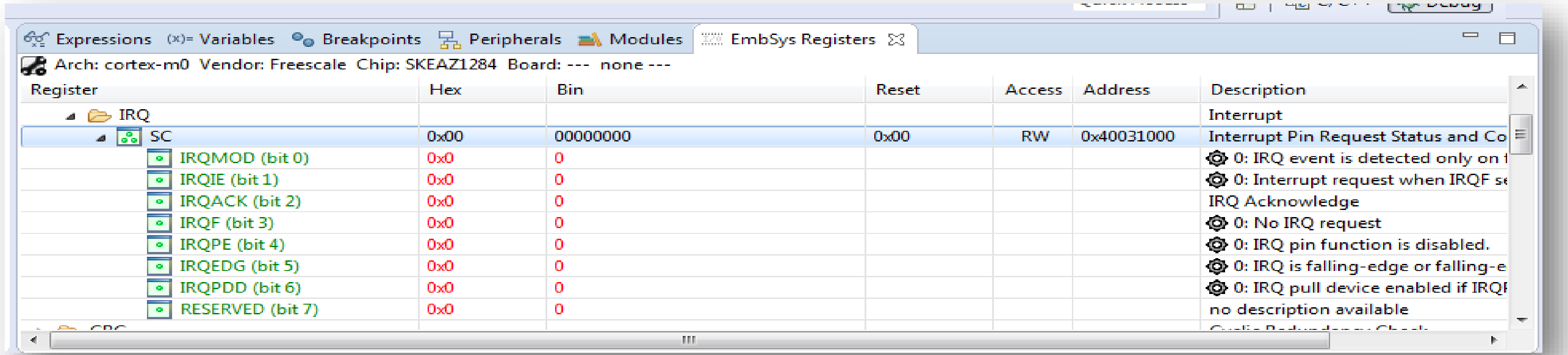
Debug Basics: View & Alter Registers

- View CPU registers in the “Registers” tab
- Click on a value to allow typing in a different value
- View peripheral registers in the EmbSys Registers tab



(x)= Variables | Breakpoints | Registers | Modules

Name	Value
General Registers	
r0	3
r1	5
r2	536866944
r3	8



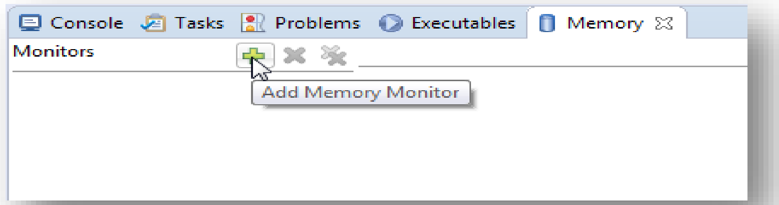
Expressions | (x)= Variables | Breakpoints | Peripherals | Modules | EmbSys Registers

Arch: cortex-m0 Vendor: Freescale Chip: SKEAZ1284 Board: --- none ---

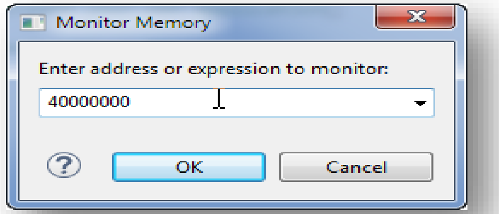
Register	Hex	Bin	Reset	Access	Address	Description
IRQ						Interrupt
SC	0x00	00000000	0x00	RW	0x40031000	Interrupt Pin Request Status and Co
IRQMOD (bit 0)	0x0	0				0: IRQ event is detected only on f
IRQIE (bit 1)	0x0	0				0: Interrupt request when IRQF se
IRQACK (bit 2)	0x0	0				IRQ Acknowledge
IRQF (bit 3)	0x0	0				0: No IRQ request
IRQPE (bit 4)	0x0	0				0: IRQ pin function is disabled.
IRQEDG (bit 5)	0x0	0				0: IRQ is falling-edge or falling-e
IRQPDD (bit 6)	0x0	0				0: IRQ pull device enabled if IRQF
RESERVED (bit 7)	0x0	0				no description available

Debug Basics: View & Alter Memory

- Add Memory Monitor

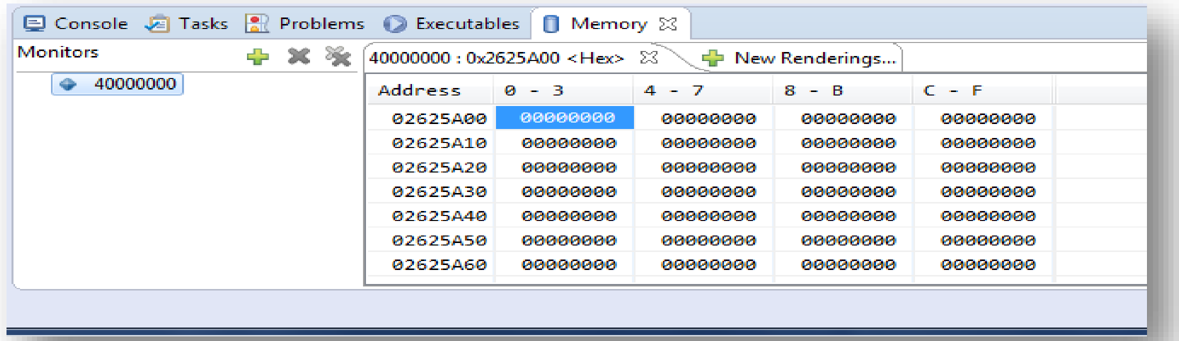


- Select Base Address to Start at : 40000000



Enter address or expression to monitor: 40000000

- View Memory



40000000

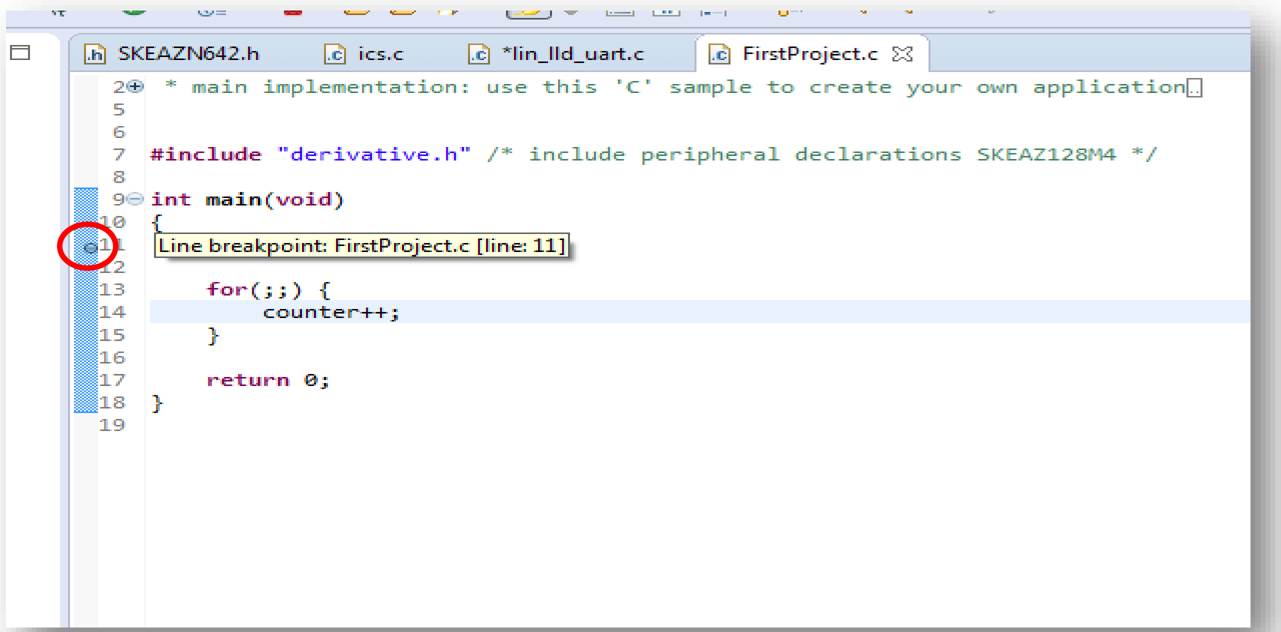
40000000 : 0x2625A00 <Hex>

Address	0 - 3	4 - 7	8 - B	C - F
02625A00	00000000	00000000	00000000	00000000
02625A10	00000000	00000000	00000000	00000000
02625A20	00000000	00000000	00000000	00000000
02625A30	00000000	00000000	00000000	00000000
02625A40	00000000	00000000	00000000	00000000
02625A50	00000000	00000000	00000000	00000000
02625A60	00000000	00000000	00000000	00000000

Debug Basics: Breakpoints

Add Breakpoint: Point and Click

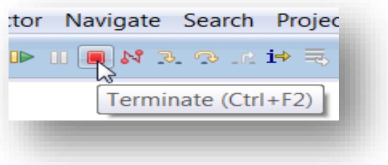
- light blue dot represents debugger breakpoint



```
 * main implementation: use this 'C' sample to create your own application

#include "derivative.h" /* include peripheral declarations SKEAZ128M4 */

int main(void)
{
Line breakpoint: FirstProject.c [line: 11]

    for(;;) {
        counter++;
    }

    return 0;
}
```

Debug Basics: Reset & Terminate Debug Session

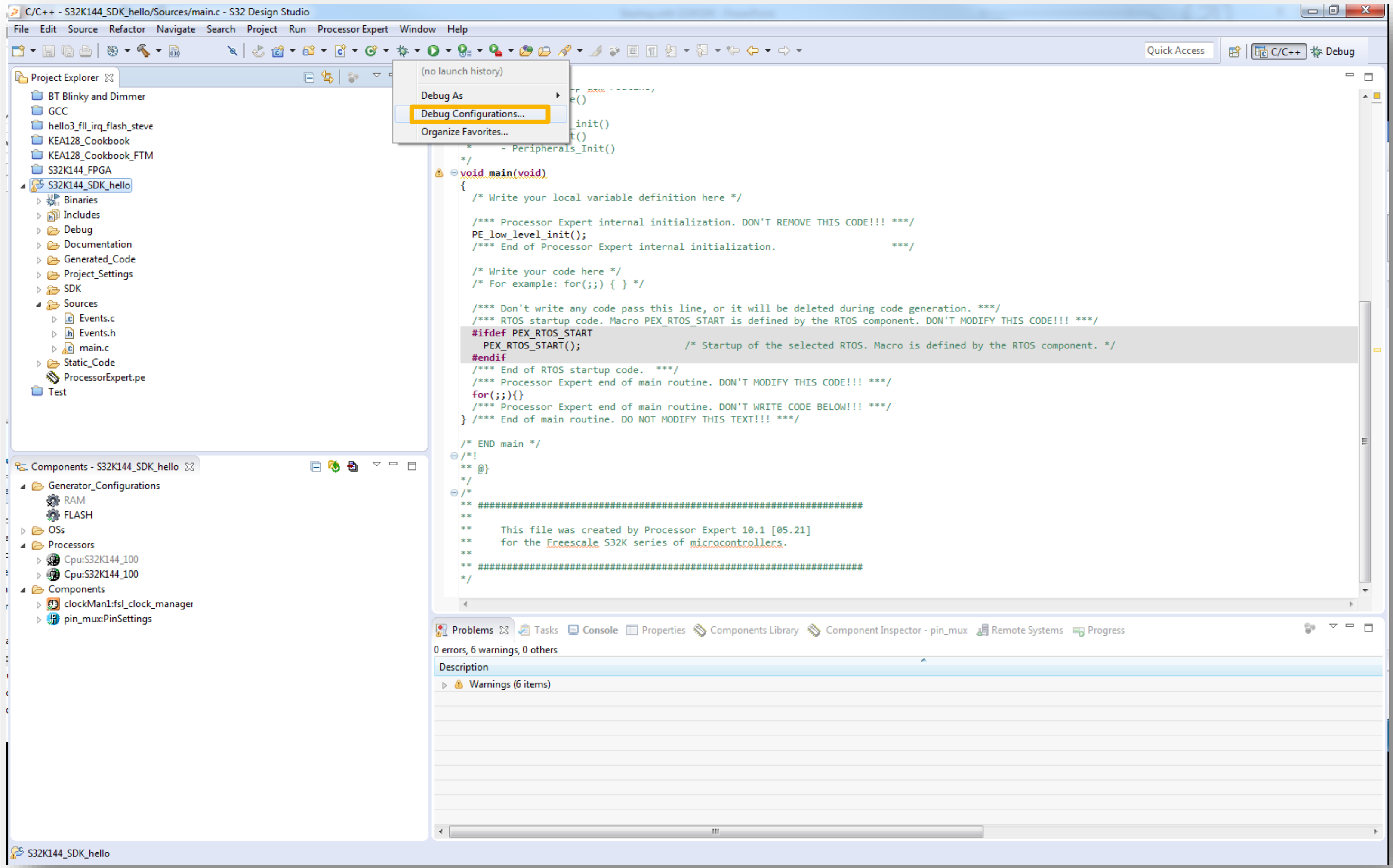
- Reset program counter
- Terminate Ctl+F2()



CREATE A P&E DEBUG CONFIGURATION (OPTIONAL)

New P&E debug configuration

- Click in debug configurations



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

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

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

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

/* END main */
/*!
** @}
*/
/*
** ###################################################################
**
**     This file was created by Processor Expert 10.1 [05.21]
**     for the Freescale S32K series of microcontrollers.
**
** ###################################################################
*/
```

0 errors, 6 warnings, 0 others

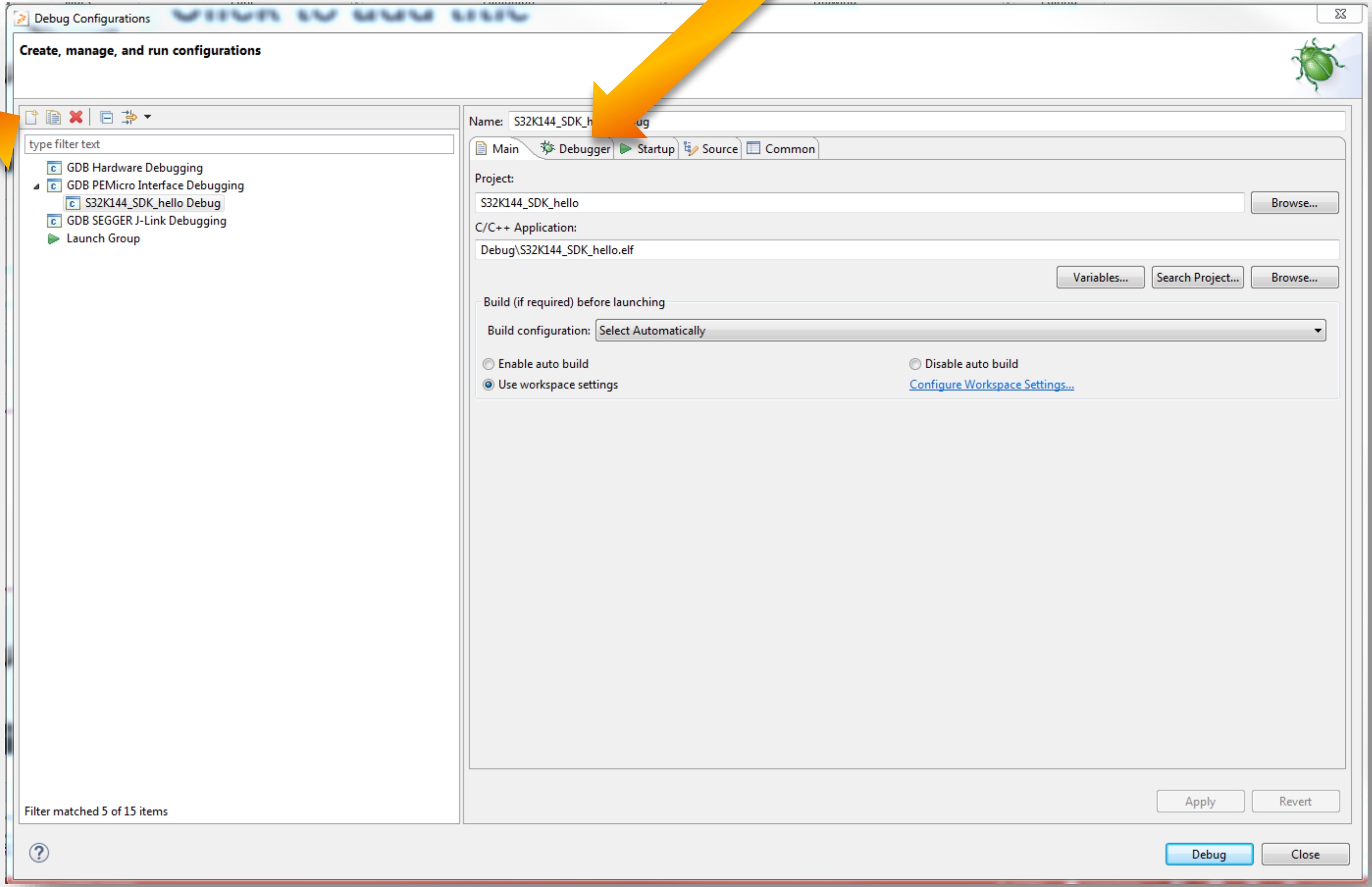
Description
Warnings (6 items)

New P&E debug configuration

- Create a new P&E launch configuration

Click on the debugger tab.

Click to create a new P&E launch



Name: S32K144_SDK_h ug

Project: S32K144_SDK_hello

C/C++ Application: Debug\S32K144_SDK_hello.elf

Build (if required) before launching

Build configuration: Select Automatically

Enable auto build

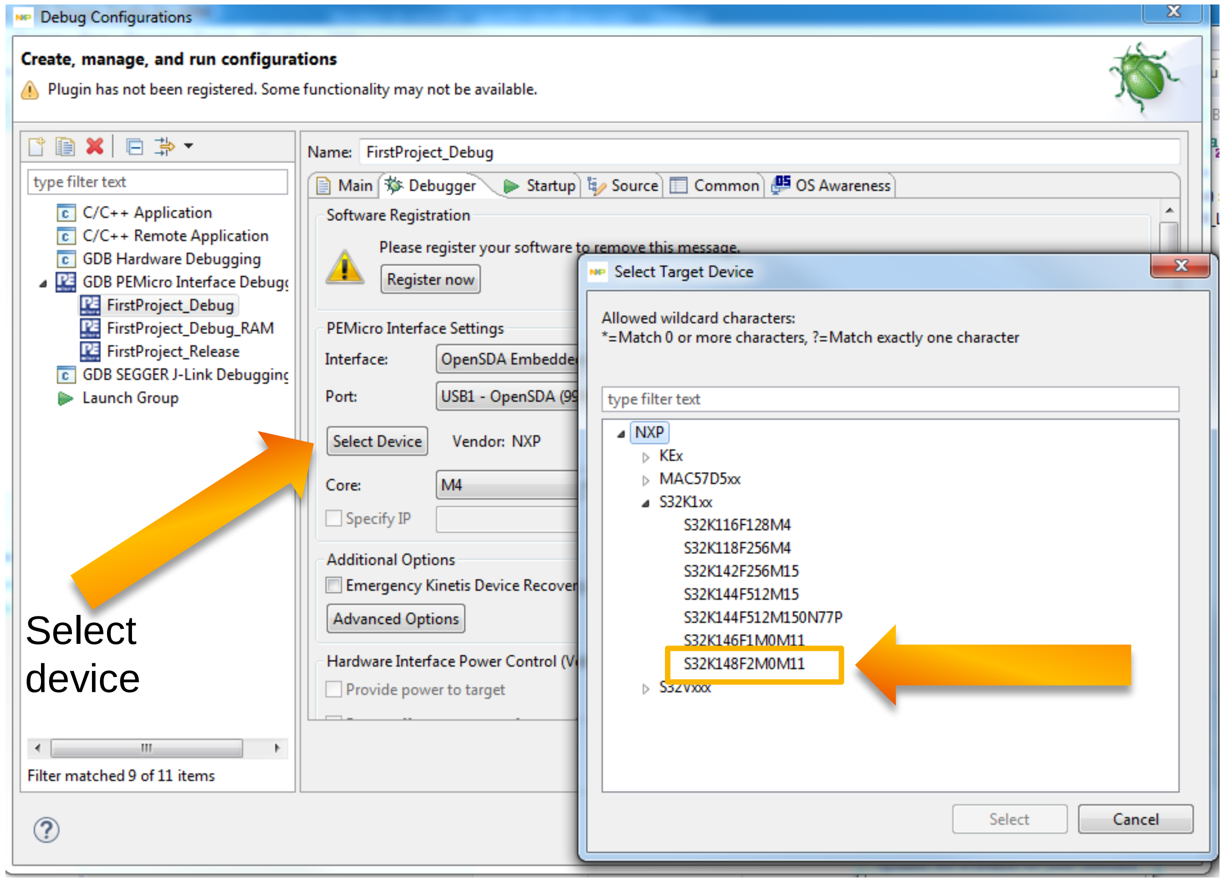
[✓ Use workspace settings]

Disable auto build

Configure Workspace Settings...

New P&E debug configuration

- Select the device



Debug Configurations

Create, manage, and run configurations

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

type filter text

- C/C++ Application
- C/C++ Remote Application
- GDB Hardware Debugging
- GDB PEMicro Interface Debugg
  - FirstProject_Debug
  - FirstProject_Debug_RAM
  - FirstProject_Release
- GDB SEGGER J-Link Debugging
- Launch Group

Filter matched 9 of 11 items

Name: FirstProject_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 Embedde

Port: USB1 - OpenSDA (9

Select Device   Vendor: NXP

Core: M4

Specify IP

Additional Options

Emergency Kinetis Device Recover

Advanced Options

Hardware Interface Power Control (V

Provide power to target

Select Target Device

Allowed wildcard characters:
*=Match 0 or more characters, ?=Match exactly one character

type filter text

- NXP
  - KEx
  - MAC57D5xx
  - S32K1xx
    - S32K116F128M4
    - S32K118F256M4
    - S32K142F256M15
    - S32K144F512M15
    - S32K144F512M150N77P
    - S32K146F1M0M11
    - S32K148F2M0M11
  - S32Vxxx

Select   Cancel

- Click Apply and debug your application

USING ETHERNET AND QSPI

IMPORTANT OBSERVATION

The S32K148 is the only member of the family able to use ethernet and QuadSPI. However, these interfaces are mutually exclusive so only one of them can be used at the same time. In order to use either Ethernet or QuadSPI user must follow an specific resistor configuration. The default configuration of the board is to be used for ethernet communication.

ETHERNET

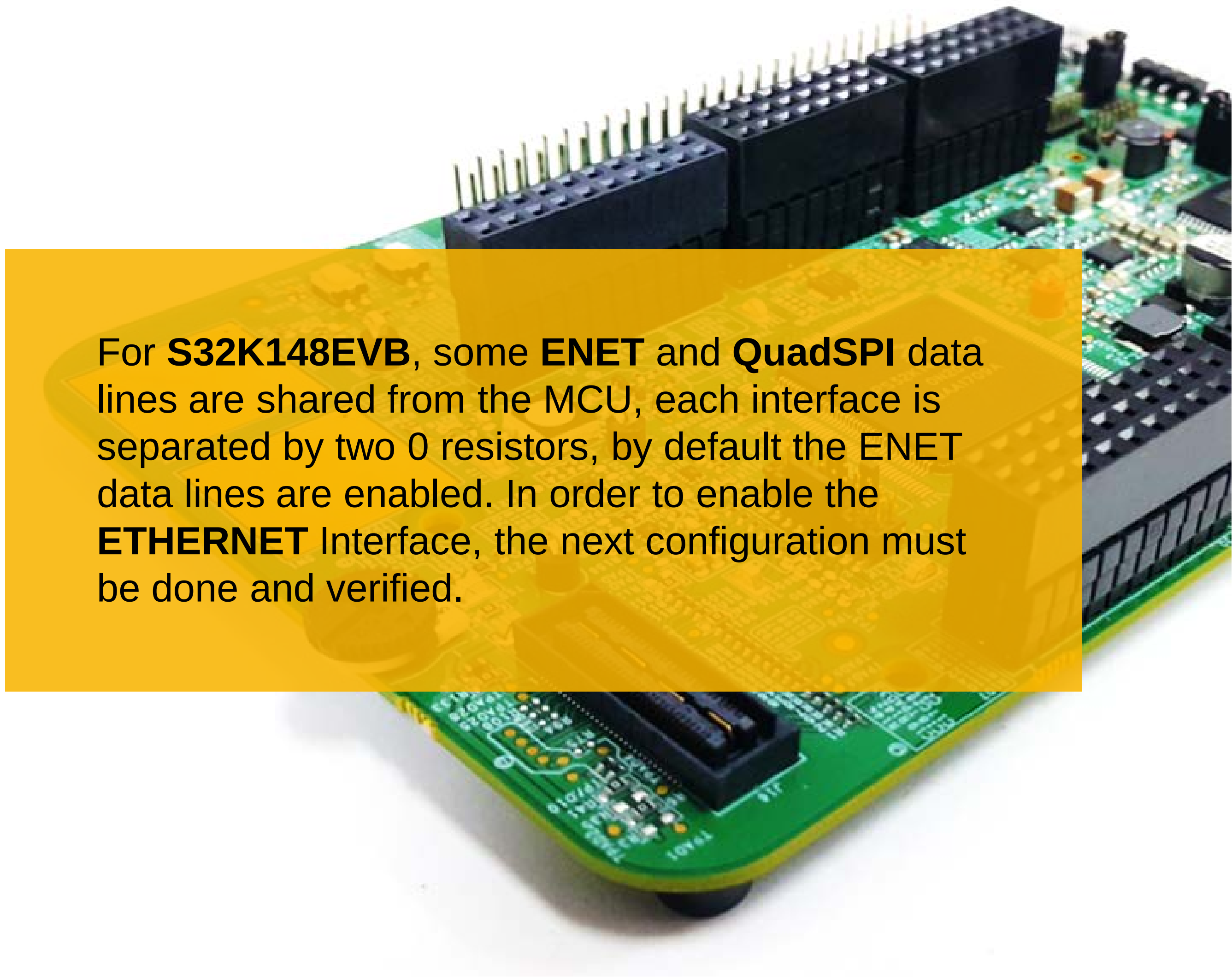
Using Ethernet on the S32K148EVB

- Different from the rest of the devices on the S32K1xx family the S32K148 has the ENET module which offers the possibility to use Ethernet communication protocol. This enables this device for applications such as:
  - Small Gateway (LIN-CAN-ETHERNET)
  - Audio Amplifier

- The Software Development Kit (SDK) for the S32K1xx devices already offers a middleware ethernet stack (LwIP), that allow the user to develop applications faster.

Using Ethernet on the S32K148EVB: Configuration (Default)

CIRCUIT	PART REFERENCE	SIGNAL NAME	DESCRIPTION
ENET	**R198**	PTD9/MII_RXD2	**Populate 0 Ohms/0402 Resistors**
	R197	PTD8/MII_RXD3	
	R196	PTC17/MII_RMII_RX_DV	
	R195	PTC16/MII_RMII_RX_ER	
	R268	PTB4/MII_RMII_MDIO	
	R269	PTD6/MII_TXD2	
	R270	PTD5/MII_TXD3	
	R272	PTD7/MII_RMII_TXD1	
	R277	PTC0/MII_RMII_RXD1	
	R264	PTC1/MII_RMII_RXD0	
	R260	PTC2/MII_RMII_TXD0	
	R250	PTD10/MII_RX_CLK	
	R208	PTD12/MII_RMII_TX_EN	
	R210	PTD11/MII_RMII_TX_CLK	
	R212	PTC3/MII_TX_ER	
External Memory	**R271**	PTD7/QSPI_A_IO1	**Depopulate 0 Ohms/0402 Resistors**
	R254	PTC2/QSPI_A_IO3	
	R209	PTD12/QSPI_A_IO2	
	R211	PTD11/QSPI_A_IO0	
	R213	PTC3/QSPI_A_CS	
	R244	PTD10/QSPI_A_SCK	



For **S32K148EVB**, some **ENET** and **QuadSPI** data lines are shared from the MCU, each interface is separated by two 0 resistors, by default the ENET data lines are enabled. In order to enable the **ETHERNET** Interface, the next configuration must be done and verified.

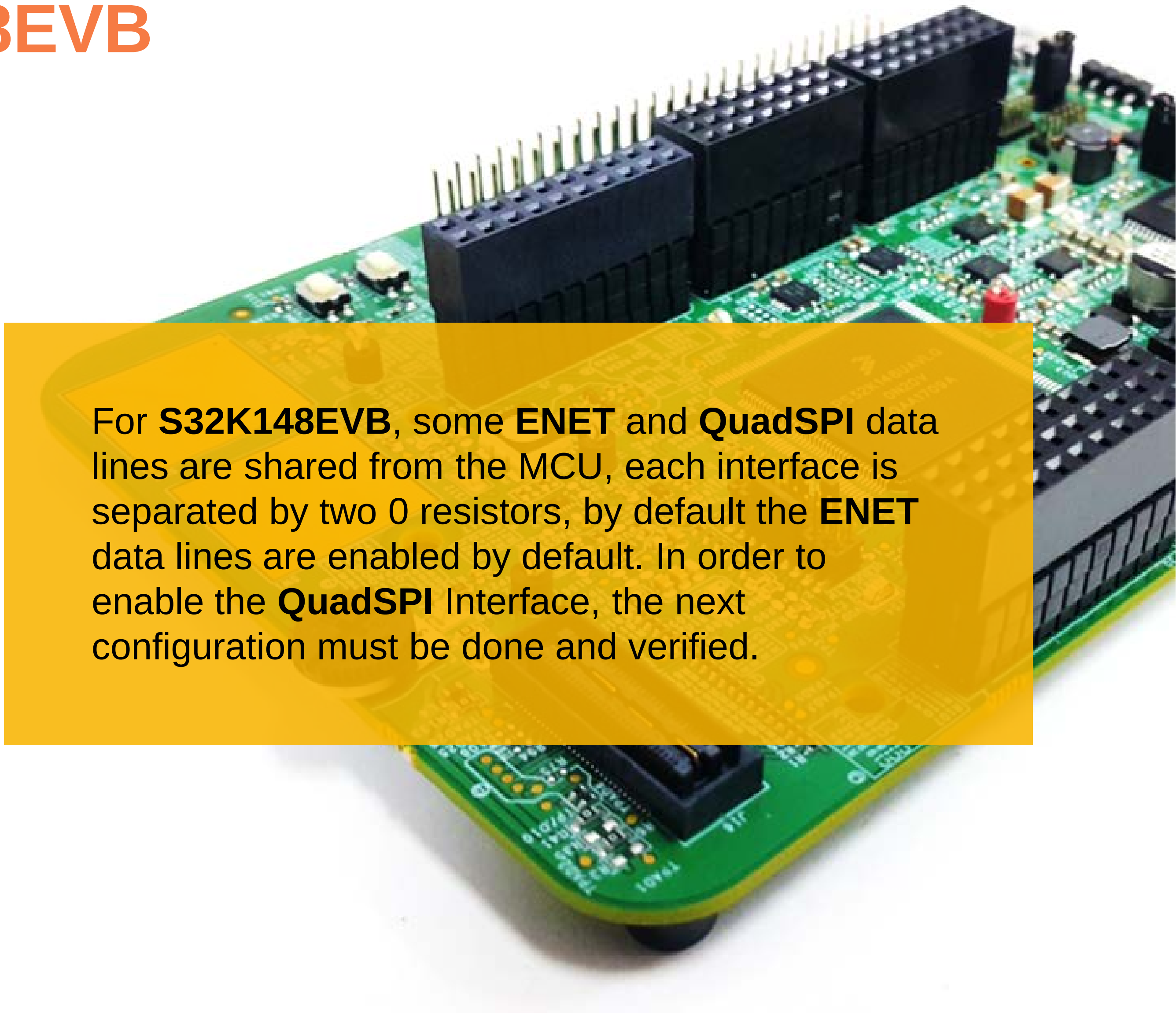
USING QUADSPI

Using QuadSPI on the S32K148EVB

- Different from the rest of the devices on the S32K1xx family the S32K148 has the QuadSPI module which offers the possibility to communicate with external devices (mostly memories) that allow QuadSPI protocol.
- The S32K148 EVB has a MX25L6433F external memory mounted on the board.
- The Software Development Kit (SDK) for the S32K1xx devices already offers an example for communicating with the external memory mounted on the board.

Using QuadSPI on the S32K148EVB

CIRCUIT	PART REFERENCE	SIGNAL NAME	DESCRIPTION
ENET	R198	PTD9/MII_RXD2	Deopulate 0 Ohms/0402 Resistors
	R197	PTD8/MII_RXD3	
	R196	PTC17/MII_RMII_RX_DV	
	R195	PTC16/MII_RMII_RX_ER	
	R268	PTB4/MII_RMII_MDIO	
	R269	PTD6/MII_TXD2	
	R270	PTD5/MII_TXD3	
	R272	PTD7/MII_RMII_TXD1	
	R277	PTC0/MII_RMII_RXD1	
	R264	PTC1/MII_RMII_RXD0	
	R260	PTC2/MII_RMII_TXD0	
	R250	PTD10/MII_RX_CLK	
	R208	PTD12/MII_RMII_TX_EN	
	R210	PTD11/MII_RMII_TX_CLK	
	R212	PTC3/MII_TX_ER	
External Memory	R271	PTD7/QSPI_A_IO1	Populate 0 Ohms/0402 Resistors
	R254	PTC2/QSPI_A_IO3	
	R209	PTD12/QSPI_A_IO2	
	R211	PTD11/QSPI_A_IO0	
	R213	PTC3/QSPI_A_CS	
	R244	PTD10/QSPI_A_SCK	



For **S32K148EVB**, some **ENET** and **QuadSPI** data lines are shared from the MCU, each interface is separated by two 0 resistors, by default the **ENET** data lines are enabled by default. In order to enable the **QuadSPI** Interface, the next configuration must be done and verified.

USEFUL LINKS

Useful Links

- Cookbook application note. This application note contains a bunch of simple examples of how to use different peripherals.

- S32K1xx community. Visit this site for request support on the S32K1xx products, you can also look for threads that may contain the answer that you are looking for.

SECURE CONNECTIONS
FOR A SMARTER WORLD