S32K148 EVB

QUICK START GUIDE

REV2.3

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

EXTERNAL USE

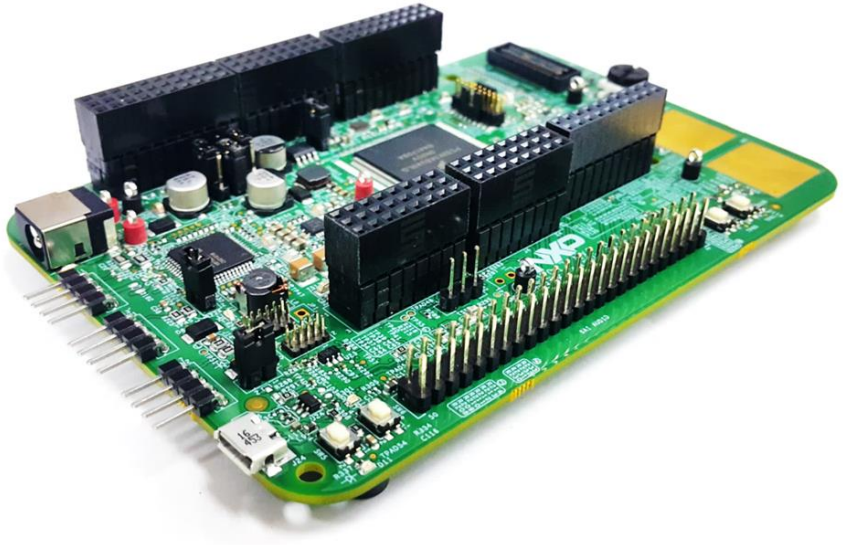
SECURE CONNECTIONS
FOR A SMARTER WORLD

Contents:

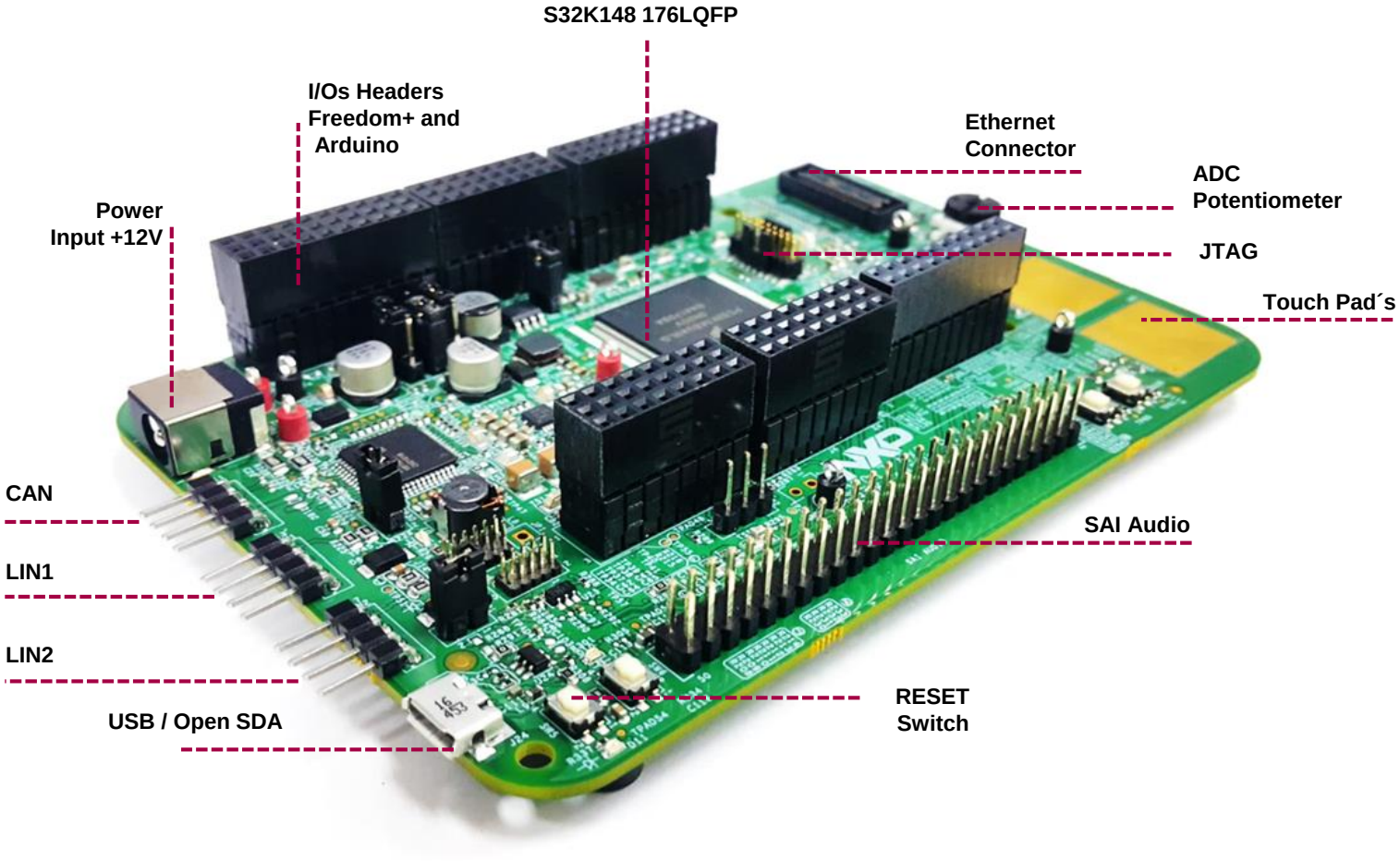
- Get to Know S32K148 EVB
- JumpStart Setup
- JumpStart based on the FreeMASTER tool
- Introduction to OpenSDA
- S32DS IDE basics:
  - Download
  - Create a project
  - Create a project from SDK example
- S32DS Debug basics
- Create a P&E debug configuration
- Using Ethernet and QuadSPI on the S32K148EVB

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 cards
- Voltage supply options for 3.3v or 5v.
- Flexible power supply options
  - microUSB or
  - external 12V power supply



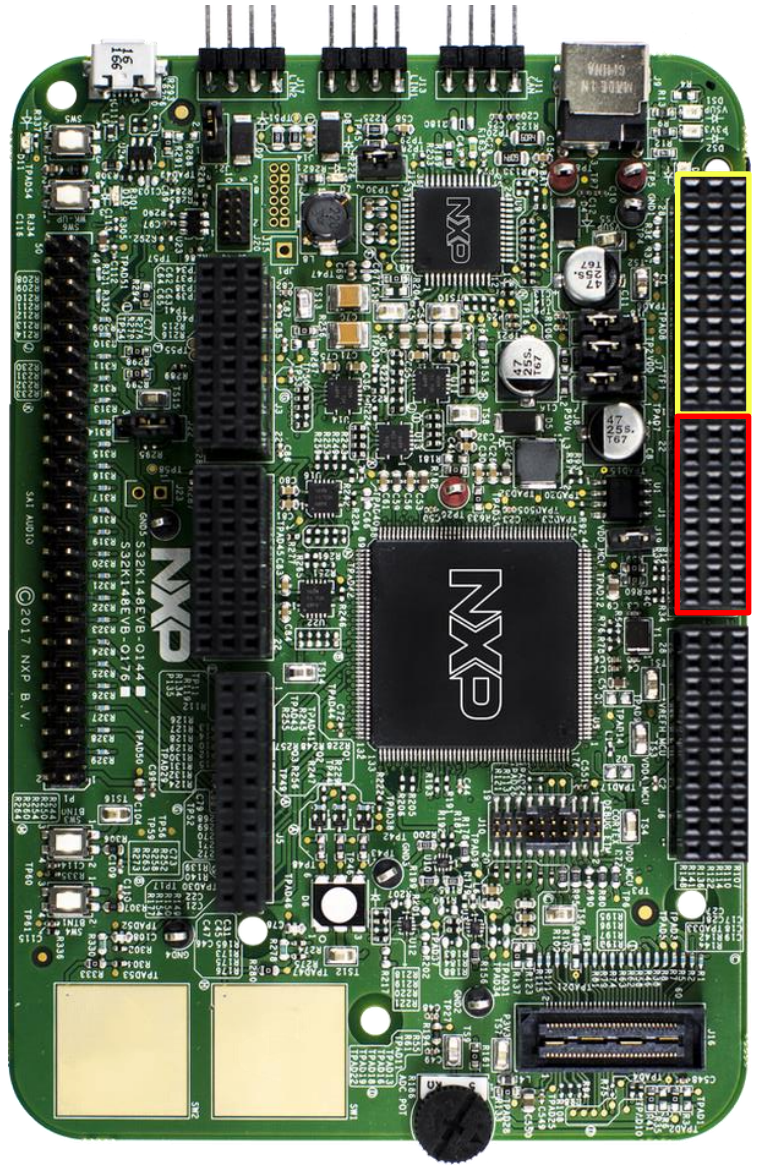
Get to know the S32K148EVB



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

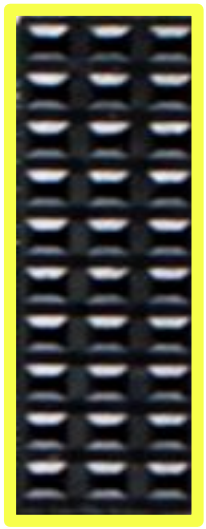
Features include easy access to all MCU I/O´s, a standard-based form factor compatible with the Arduino™ pin layout, providing a broad range of expansion board options, and an USB serial port interface for connection to the IDE, the board has option to be powered via USB or an external power supply.

PINOUT (J2 AND J1)



J2

PIN	PORT
J2-28	PTE15
J2-25	PTE16
J2-22	VREFH
J2-19	GND
J2-16	PTB2
J2-13	PTB3
J2-10	PTB1
J2-7	PTB0
J2-4	PTA30
J2-1	PTA31

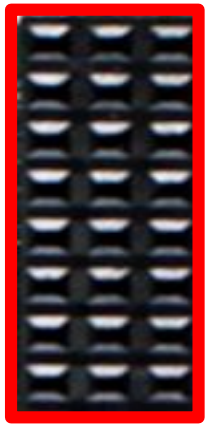


PIN	PORT
J2-29	PTB12
J2-26	PTB13
J2-23	PTE11
J2-20	PTE10
J2-17	PTB11
J2-14	PTB17
J2-11	PTB18
J2-8	PTA6
J2-5	PTA7
J2-2	PTA25

PIN	PORT
J2-30	PTD31
J2-27	PTD26
J2-24	PTD25
J2-21	PTC26
J2-18	PTC25
J2-15	PTC24
J2-12	PTC22
J2-9	PTC21
J2-6	PTC20
J2-3	PTC18

J1

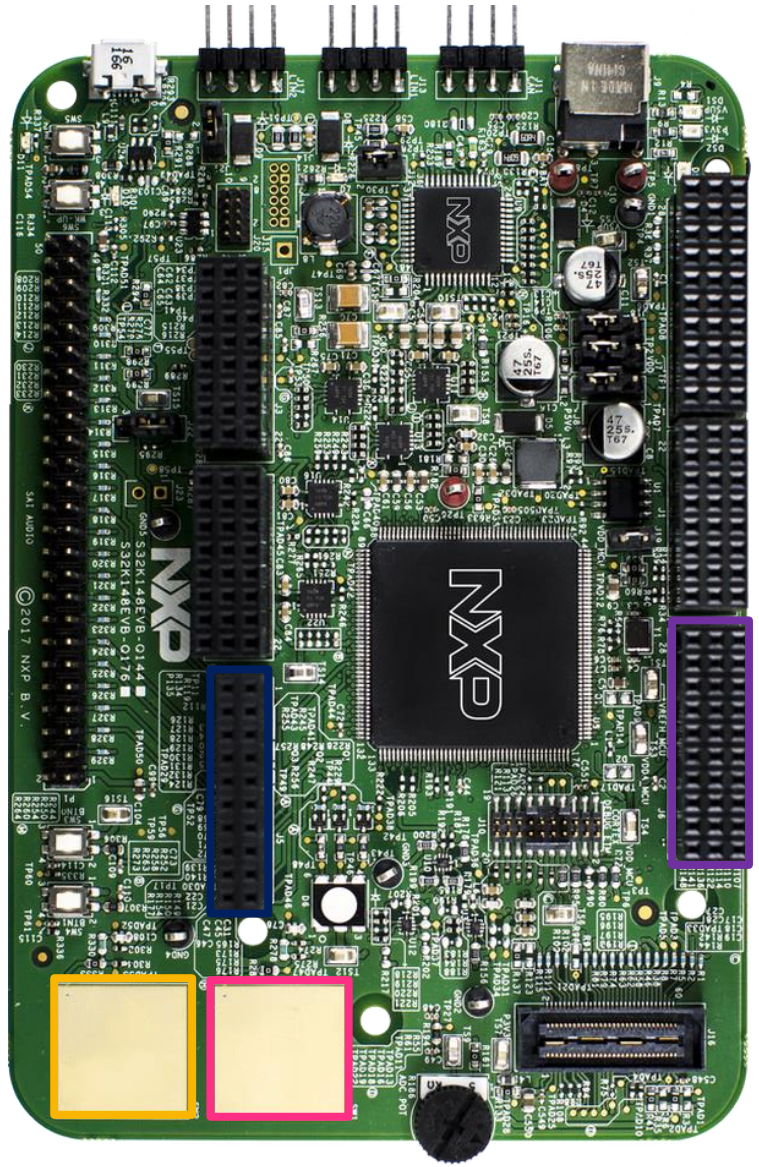
PIN	PORT
J1-22	PTD14
J1-19	PTD15
J1-16	PTD16
J1-13	PTD17
J1-10	PTC10
J1-7	PTC11
J1-4	PTA3
J1-1	PTA2



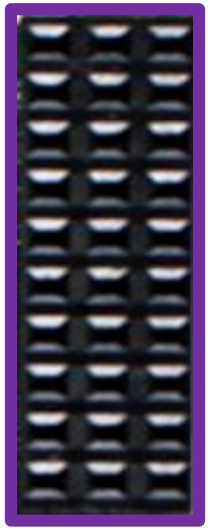
PIN	PORT
J1-23	PTA13
J1-20	PTA14
J1-17	PTE2
J1-14	PTE3
J1-11	PTE6
J1-8	PTB7
J1-5	PTE8
J1-2	PTE9

PIN	PORT
J1-24	PTD20
J1-21	PTD21
J1-18	PTB24
J1-15	PTB26
J1-12	GND
J1-9	GND
J1-6	PTB30
J1-3	PTB31

PINOUT (J6, J5 AND ELECTRODES)



J6

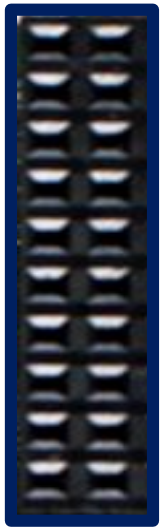


PIN	PORT
J6-28	PTD3
J6-25	PTD2
J6-22	PTD19
J6-19	PTD18
J6-16	PTA18
J6-13	PTA19
J6-10	PTB9
J6-7	PTB10
J6-4	PTB21
J6-1	PTB20

PIN	PORT
J6-29	PTD0
J6-26	PTE14
J6-23	PTE13
J6-20	PTE12
J6-17	GND
J6-14	VDD
J6-11	PTC6
J6-8	PTC7
J6-5	PTC12
J6-2	PTC13

PIN	PORT
J6-30	PTE17
J6-27	PTE18
J6-24	PTB19
J6-21	PTE27
J6-18	PTE26
J6-15	PTA20
J6-12	PTA21
J6-9	PTA22
J6-6	PTA23
J6-3	PTA24

J5



PIN	PORT
J5-2	PTC15
J5-4	PTB8
J5-6	PTA11
J5-8	PTA12
J5-10	VDD
J5-12	GND
J5-14	PTE1
J5-16	PTE0
J5-18	PTD1
J5-20	PTD13

PIN	PORT
J5-1	PTE21
J5-3	PTE22
J5-5	PTE23
J5-7	PTE24
J5-9	PTE25
J5-11	PTC19
J5-13	PTC14
J5-15	PTB14
J5-17	PTB15
J5-19	PTB16

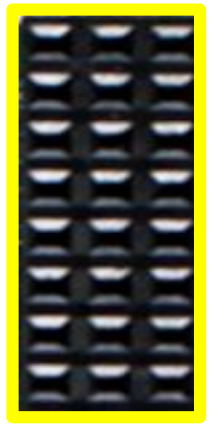
Electrode A	
PIN	PORT
TOUCH0_0	PTA0
TOUCH0_1	PTA15

Electrode B	
PIN	PORT
TOUCH1_0	PTA1
TOUCH1_1	PTA16

PINOUT (J3 AND J4)

J3

PIN	PORT
J3-3	VBAT
J3-6	VBAT
J3-9	LIN1
J3-12	GND
J3-15	LIN2
J3-18	GND
J3-21	CANH
J3-24	CANL

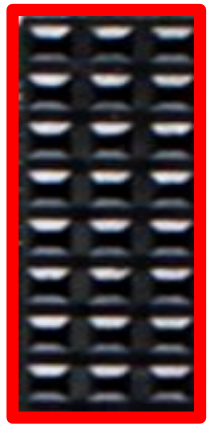


PIN	PORT
J3-2	PTB23
J3-5	PTB22
J3-8	PTB29
J3-11	PTB27
J3-14	PTB28
J3-17	PTB25
J3-20	PTA8
J3-23	PTA9

PIN	PORT
J3-1	VBAT
J3-4	VDD
J3-7	PTA5
J3-10	V3_3
J3-13	V5_0
J3-16	GND
J3-19	GDN
J3-22	VBAT

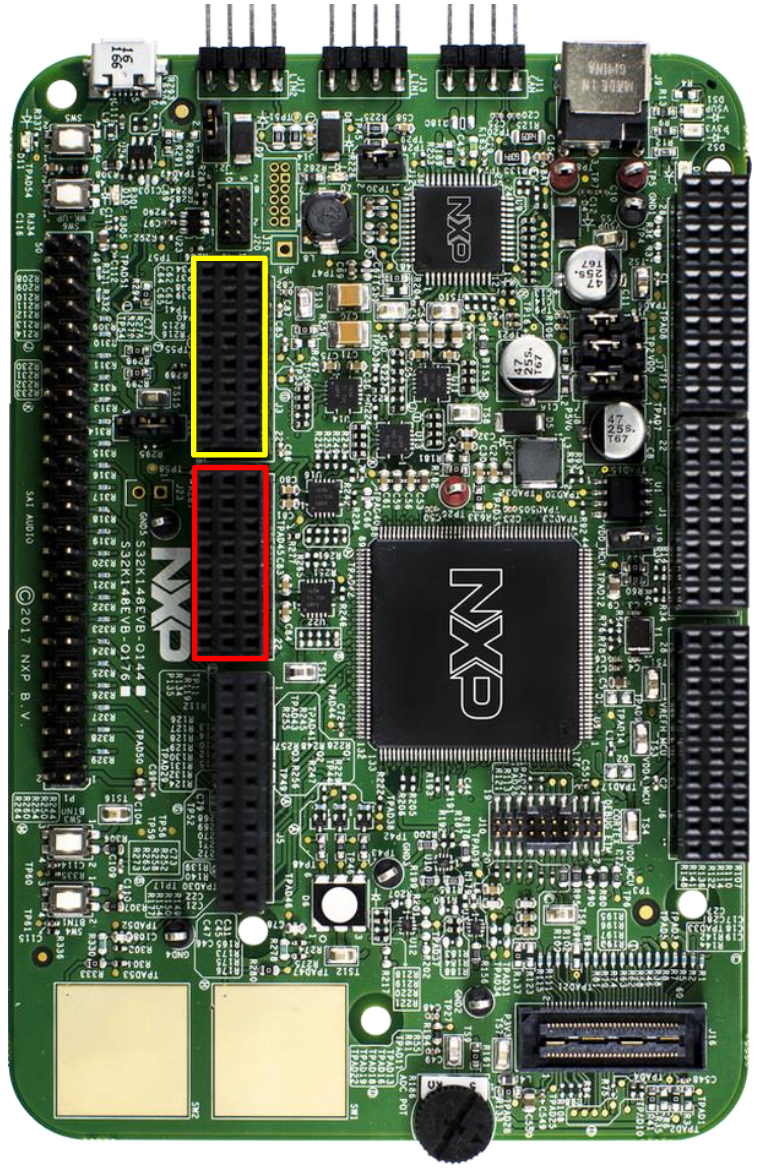
J4

PIN	PORT
J4-3	PTB17
J4-6	PTA27
J4-9	PTA28
J4-12	PTA29
J4-15	PTA0
J4-18	PTA1
J4-21	PTA15
J4-24	PTA16



PIN	PORT
J4-2	PTD4
J4-5	PTD22
J4-8	PTD23
J4-11	PTD24
J4-14	PTD27
J4-17	PTD28
J4-20	PTD29
J4-23	PTD30

PIN	PORT
J4-1	PTC23
J4-4	PTC27
J4-7	PTC28
J4-10	PTC29
J4-13	PTC30
J4-16	PTC31
J4-19	PTE19
J4-22	PTE20



S32K148 EVB Features: CAN and LIN connectors

J11

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

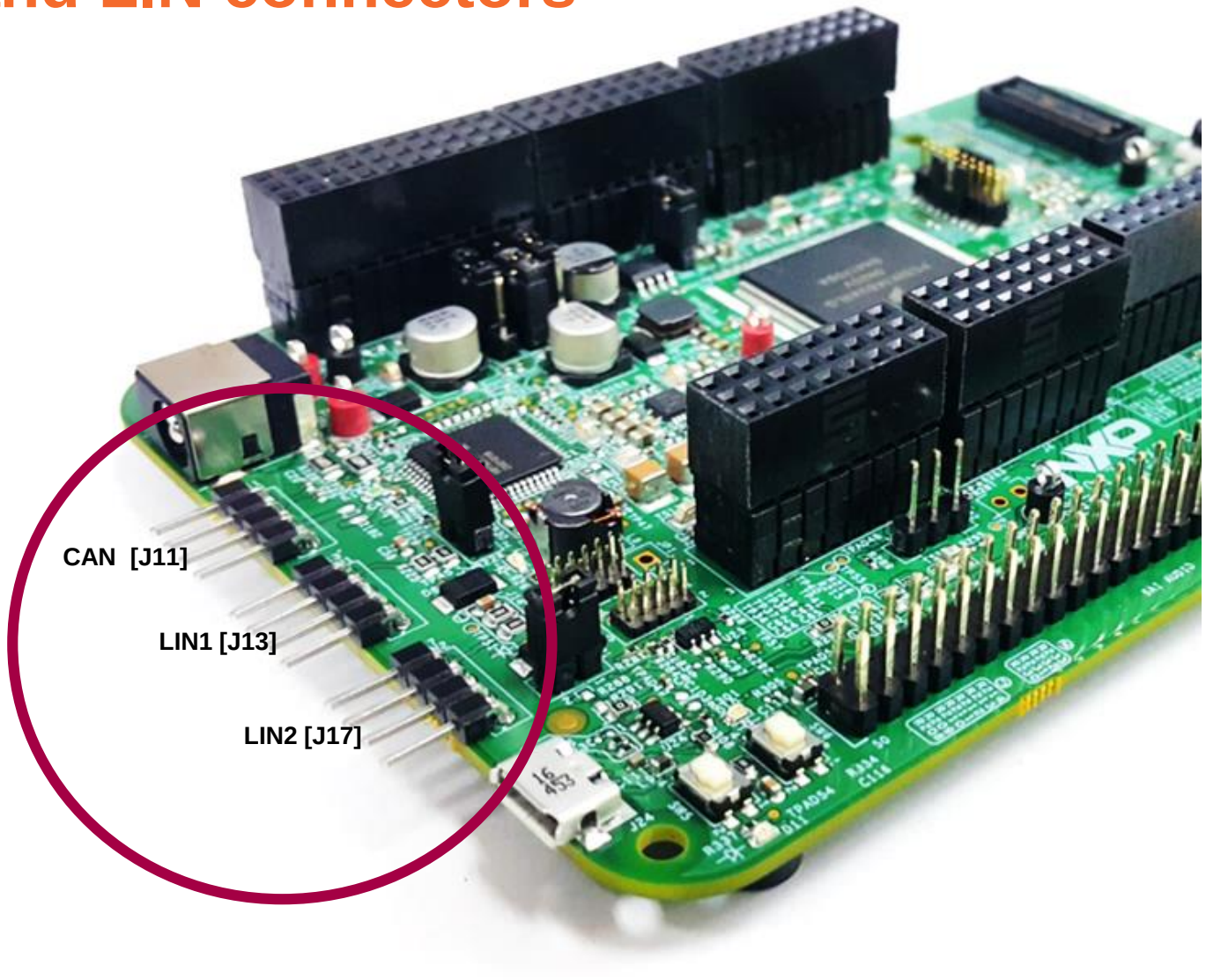
J13

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

J17

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

Front view



S32K148 EVB Features: USB/OpenSDA

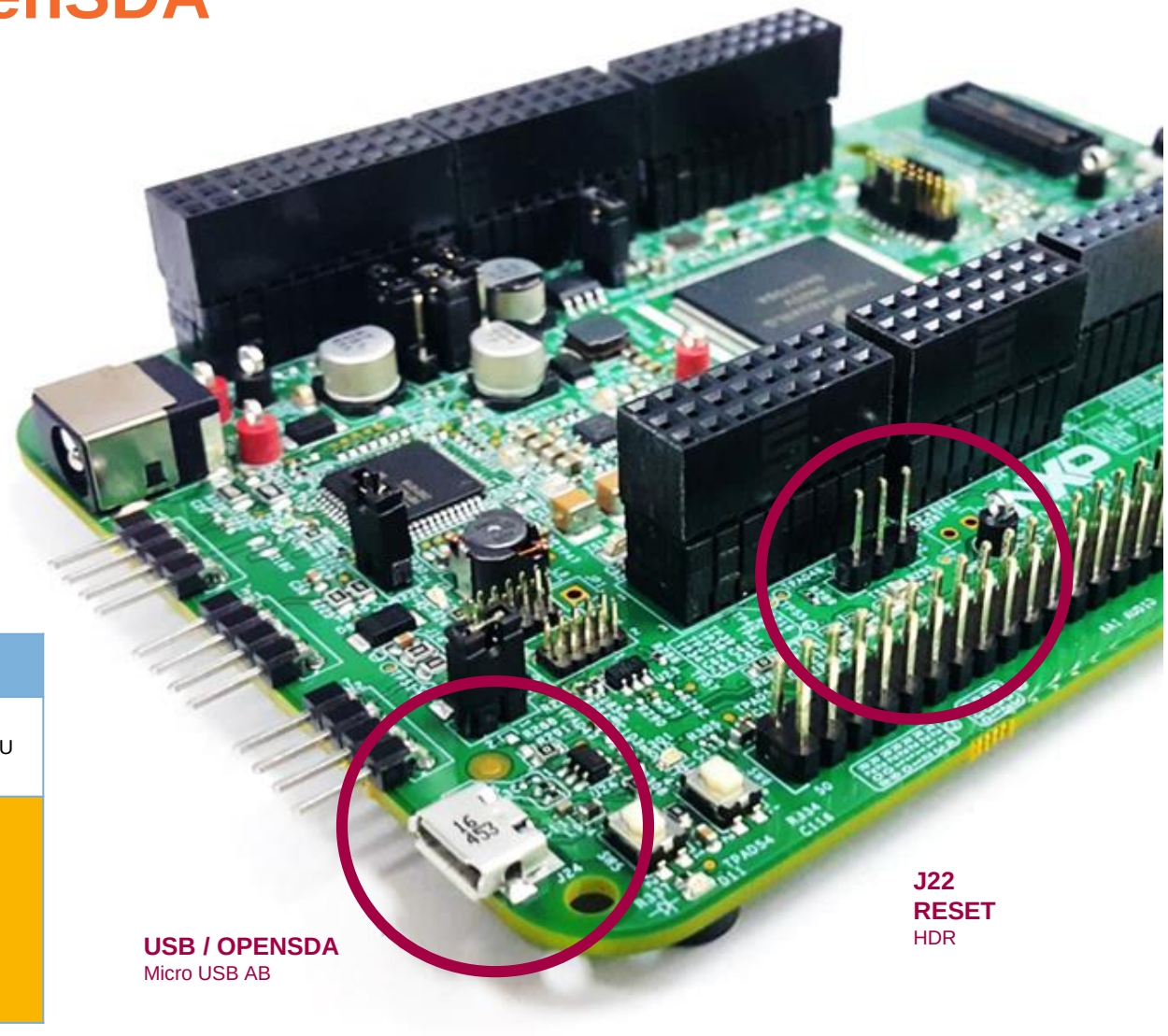
OpenSDA is a serial and debug adapter that is built into several NXP® evaluation boards. It provides a bridge between your computer (or other USB host) and the embedded target processor, which can be used for debugging, flash programming, and serial communication, all over a simple USB cable.

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
J22	1-2		**RESET** switch is routed RST MCU

ONLY for RevA

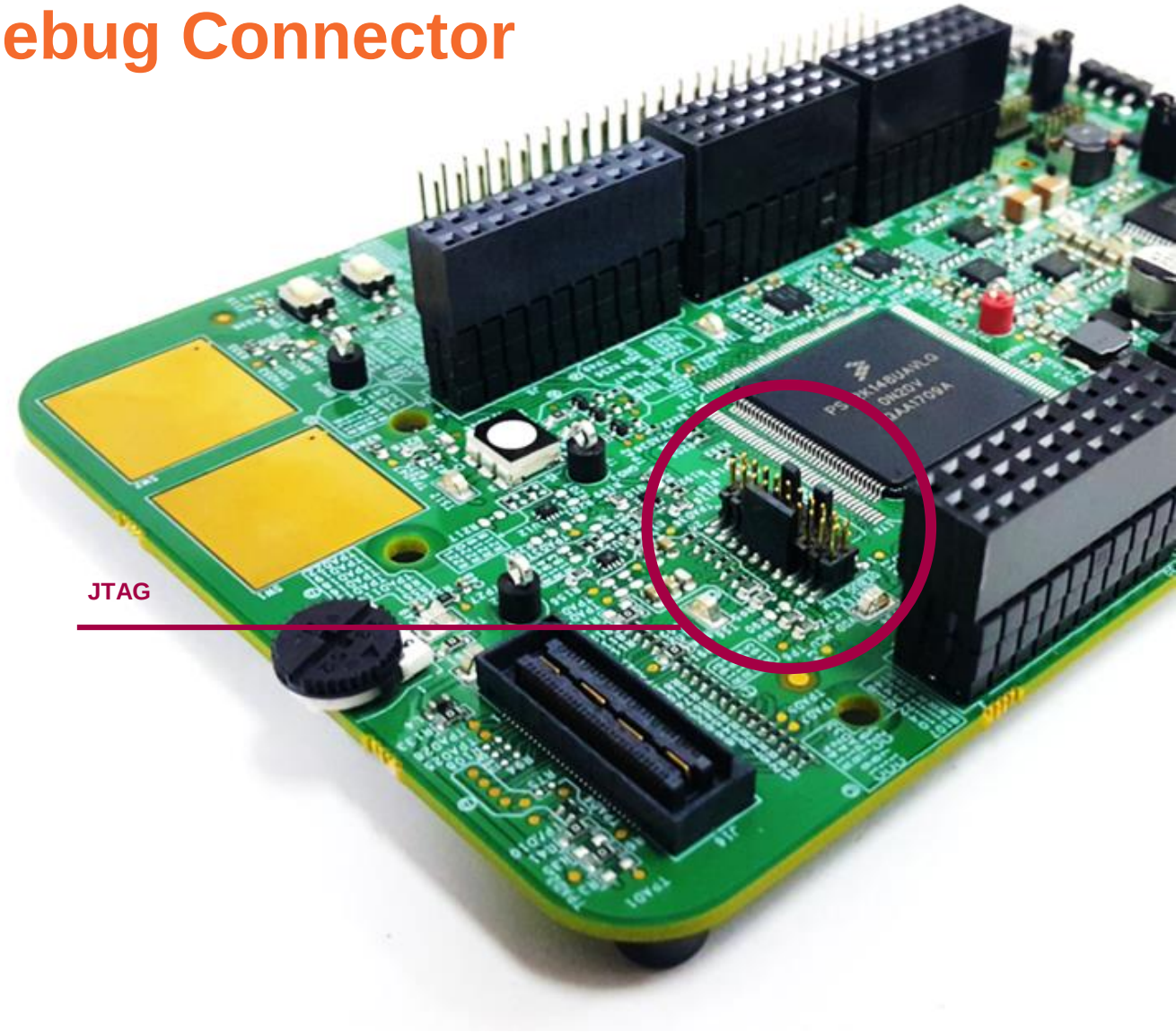
- **R537 must be removed when the S32K148 EVB is powered only via USB/OPEN SDA**
- **In order to enable SPI communication with the SBC UJA1132. R177 and R154 must be removed to swap signals [PTA29/FTM5_CH4/LPUART2_TX/LPSPI1_SIN_LS] and [PTA27/FTM5_CH2/LPSPI1_SOUT/LPUART0_TX_LS] by external wires.**



S32K148 EVB Features: JTAG Debug Connector

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

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)

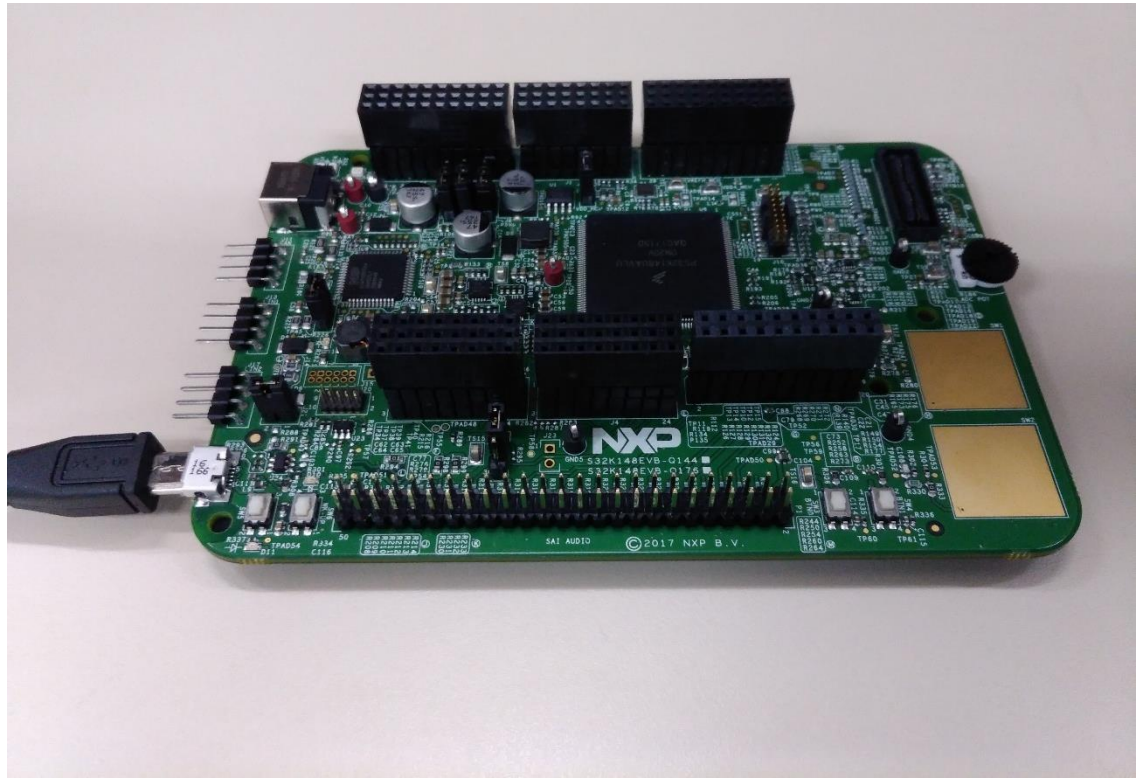
HMI mapping

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
JUMPSTART

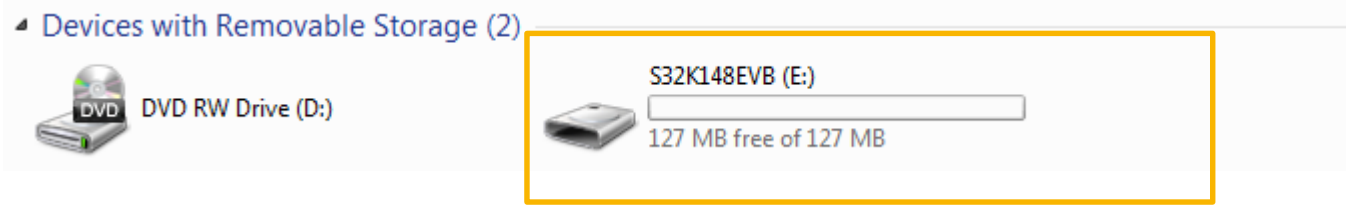
Step 1: Power up the Board – EVB Power Supplies

- The S32K148-EVB evaluation board powers from a USB or external 12V power supply. USB power can be enabled with J8 (2-3) and J18 (2-3) (check slide 10).
- Connect the USB cable to a PC using supplied USB cable .
- Connect other end of USB cable (microUSB) to micro-B port on S32K148EVB at J24
- Allow the PC to automatically configure the USB drivers if needed
- Debug is done using OpenSDA through J24



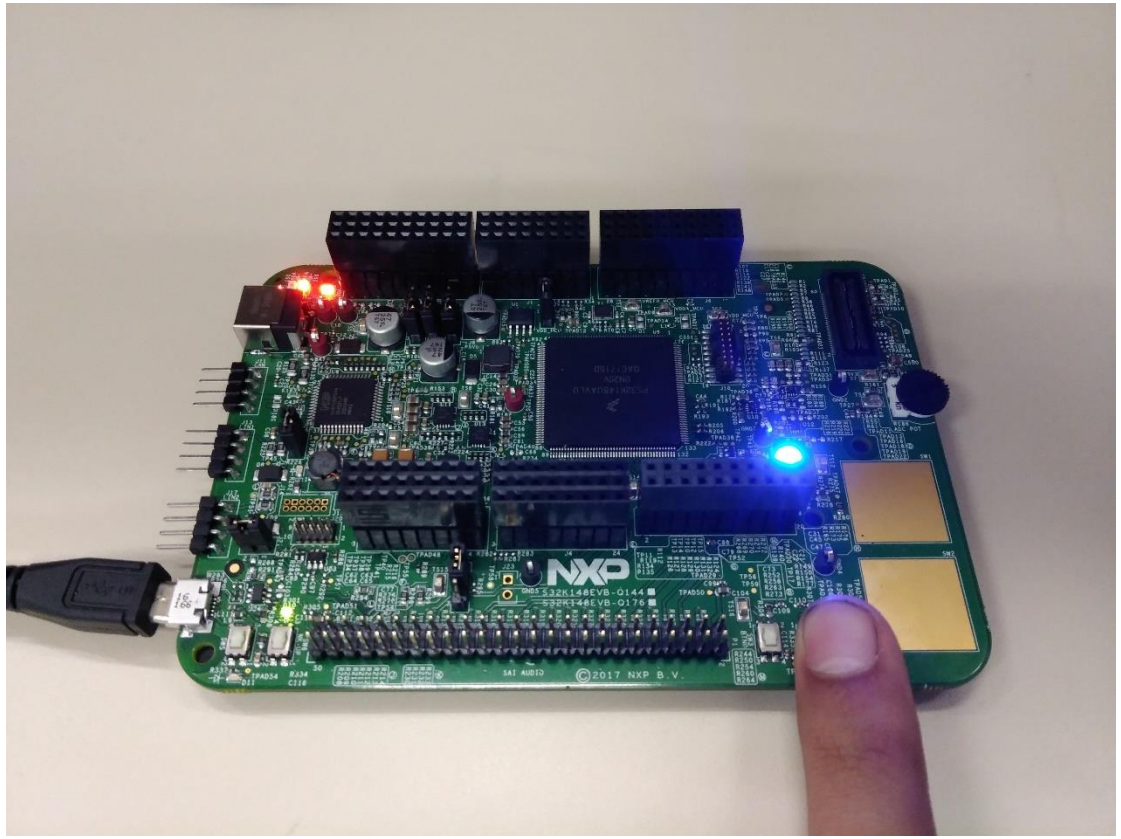
Step 1: Power up the Board – Is it powered on correctly?

- 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



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

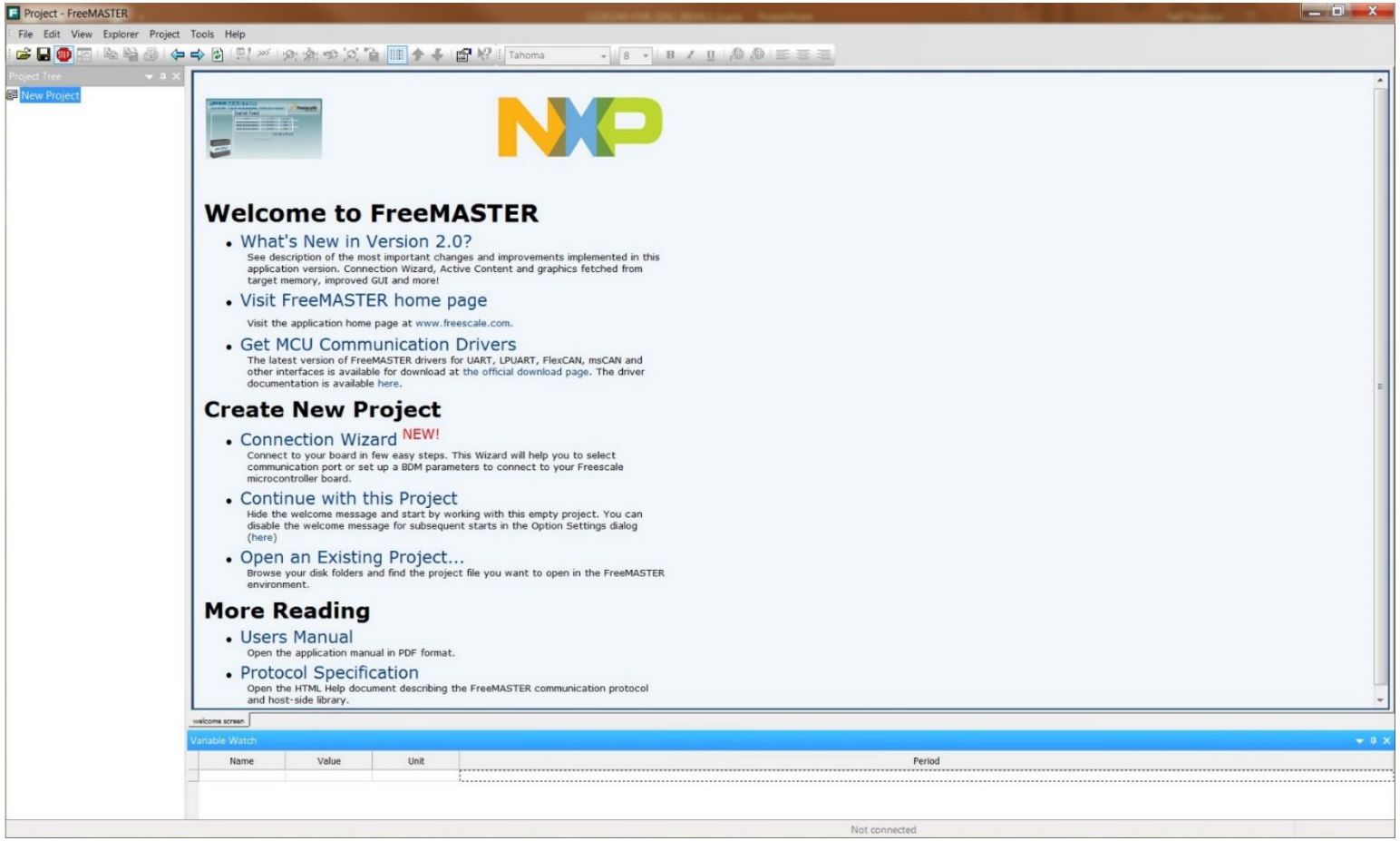
- Board is preloaded with a software toggling the RGB LED colors periodically between RED-GREEN-BLUE.



S32K148 EVB JUMPSTART BASED ON THE FREEMASTER TOOL

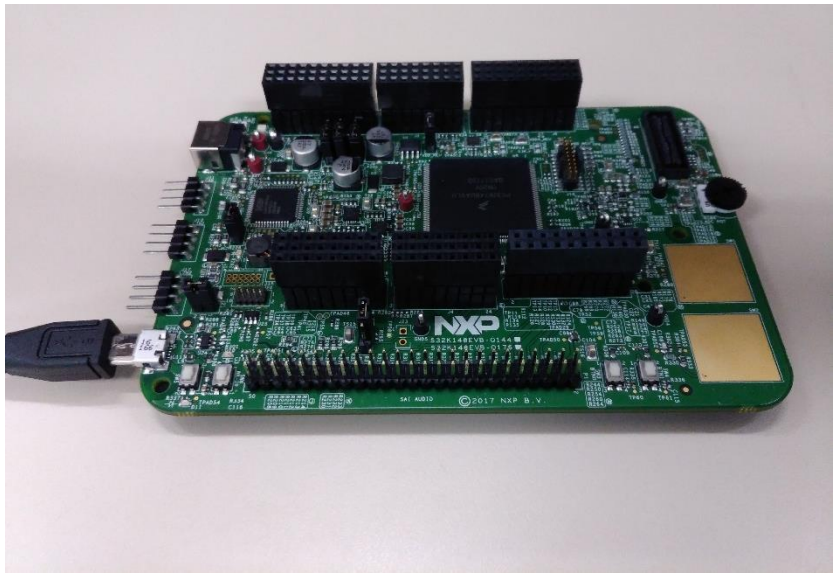
Install the FreeMASTER tool

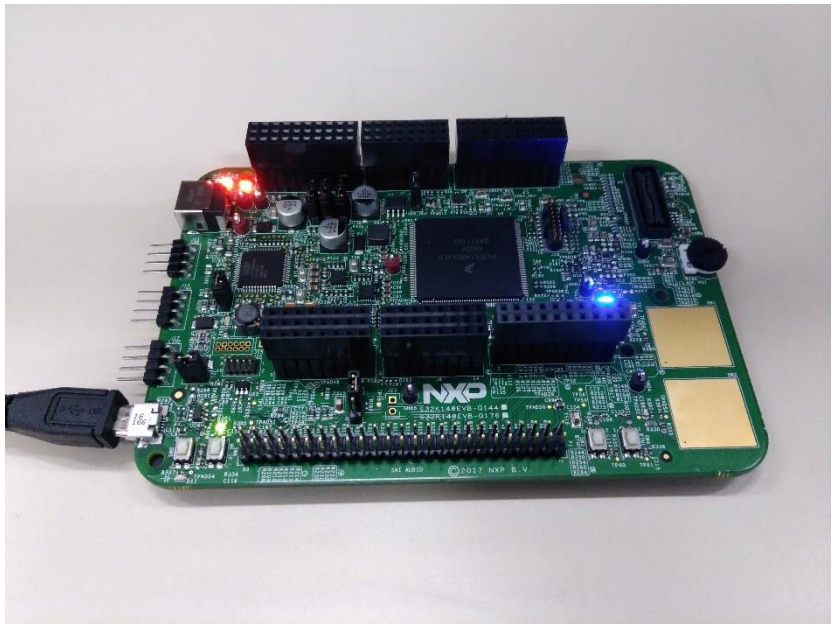
- Download and install the FreeMASTER PC application www.nxp.com/FreeMASTER .
- Open the FreeMASTER application on your PC. You should see Welcome page:



Power up the EVB board

- Power the S32K148EVB evaluation board from a USB port. USB power can be enabled with J8 (2-3) and J18 (2-3) jumpers.
- Connect the USB cable to a PC and connect micro USB connector of the USB cable to micro-B port J24 on the S32K148EVB.
- Allow the PC to automatically configure the USB drivers if needed.
- When EVB is powered from USB, LED D10 should light green, while LEDs DS2 and DS3 light orange.





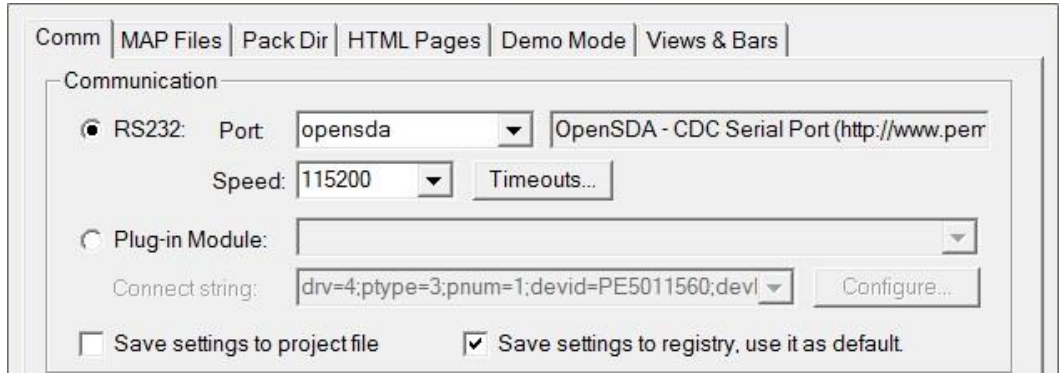
Setup serial connection in the FreeMASTER tool

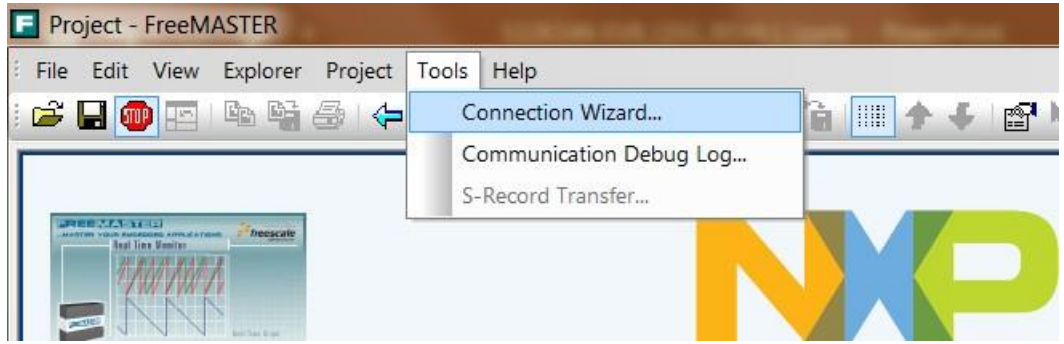
Setup communication port to “opensda” and speed to 115200 b/s:

- Setup communication manualy:
  „Project > Options > Comm“

OR

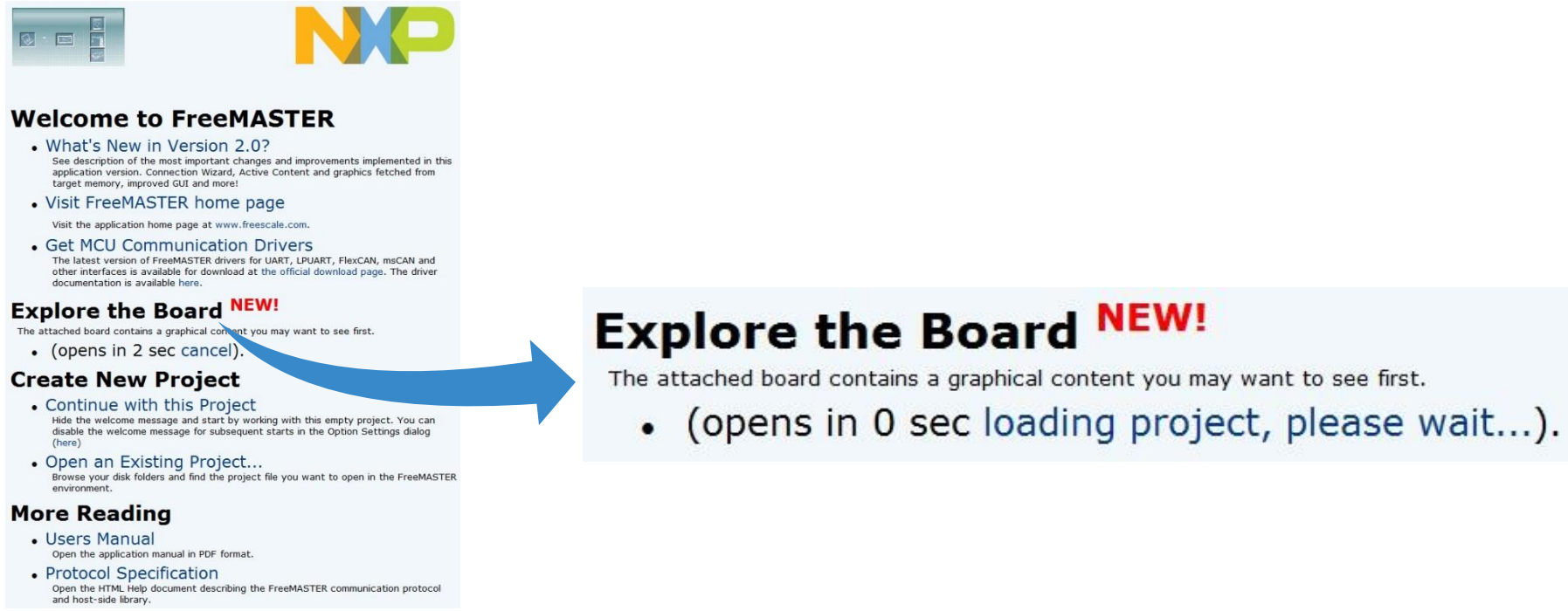
- Setup communication automatically:
  „Tools > Connection Wizard“





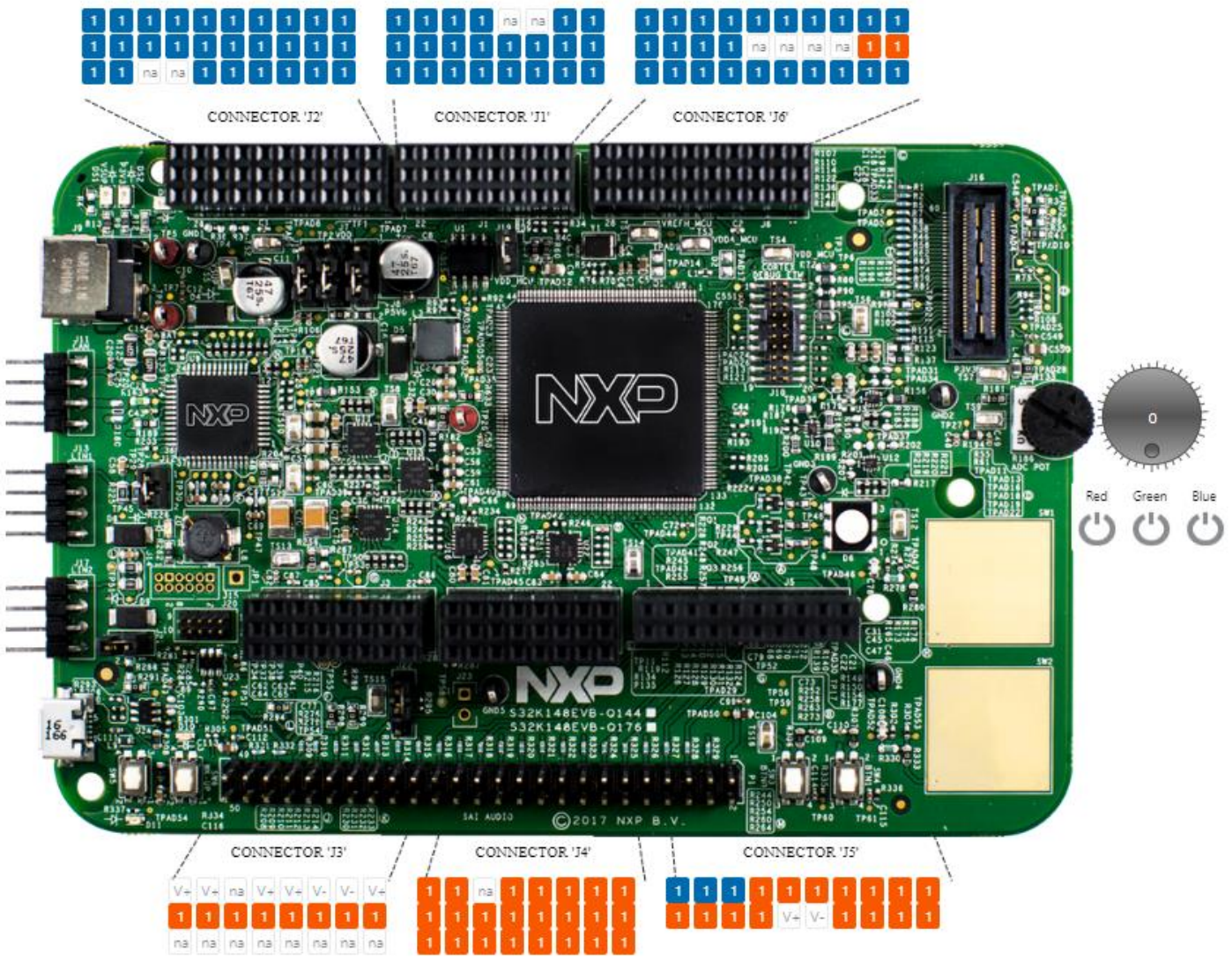
The JumpStart project will be automaticaly downloaded from www.nxp.com

Once the FreeMASTER application detects the web address stored as an TSA active content in the flash memory of the S32K148 MCU, the download of the FreeMASTER project from www.nxp.com will be initiated.



The FreeMASTER JumpStart project is loaded

Pins of the J2, J1 and J6 connectors are configured as outputs (except pins 2 and 5 on the J6 connector).



Pins of the J3, J4 and J5 connectors are configured as inputs (except pins 1,3 and 5 on the J5 connector).

S32K1xx Web Links:

>> S32K Overview

>> S32K148 Evaluation Board:

> Getting Started

> S32K148EVB Quick Start Guide

> S32K148EVB-Q176 Schematic

> S32K1xx Fact Sheet

> S32K1xx Data Sheet

> S32K1xx Reference Manual

> S32K1xx Product Brief

>> SW Tools:

> FreeMASTER Run-Time Debugging Tool

> S32 Design Studio IDE

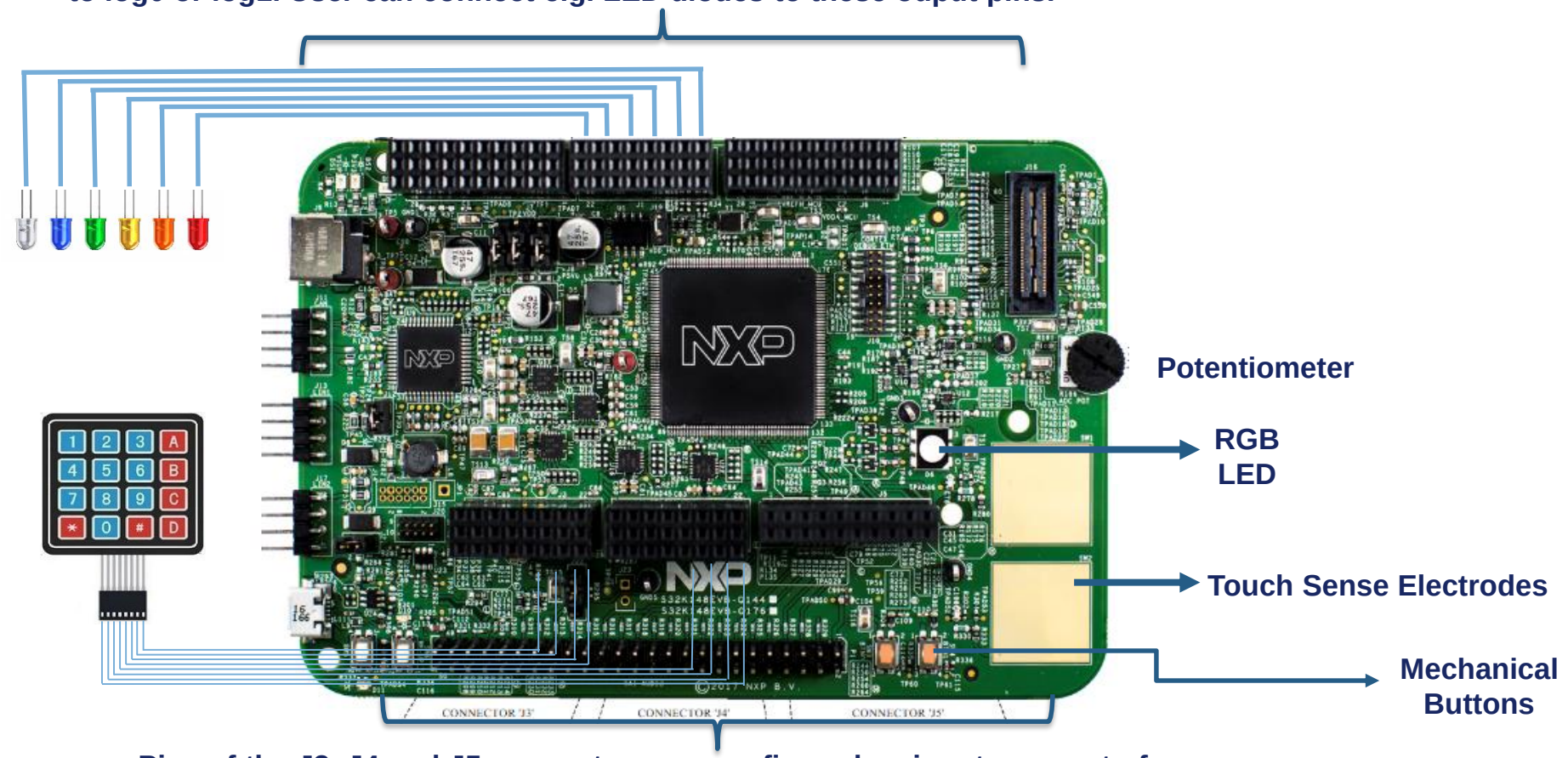
>> S32K148 EVB JumpStart Sources:

> S32K148 EVB JumpStart PC Host Project

> S32K148 EVB JumpStart Firmware

The FreeMASTER JumpStart project description

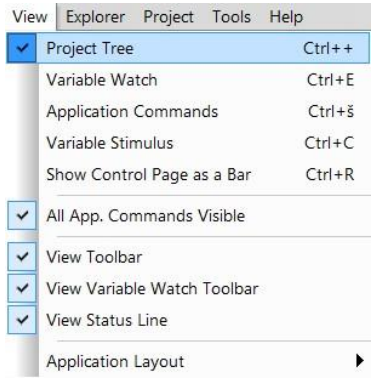
Pins of the J2, J1 and J6 connectors are configured as outputs, except for specified pins. By single click on each pin you can change their logical level to log0 or log1. User can connect e.g. LED diodes to these ouput pins.



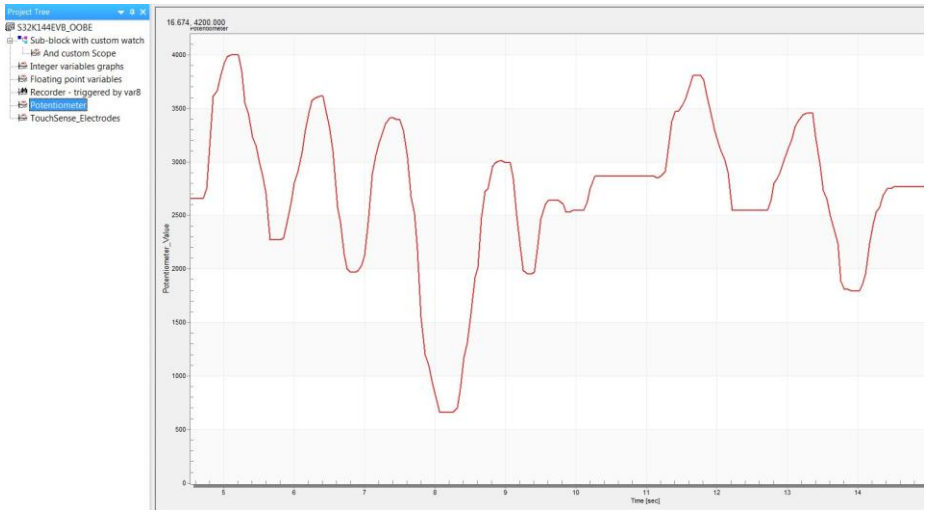
Pins of the J3, J4 and J5 connectors are configured as inputs, except of specified pins. Logical level (log0/log1) is visualised for all connector pins. User can connect e.g. push-button keyboard to these input pins.

The FreeMASTER JumpStart oscilloscope feature examples

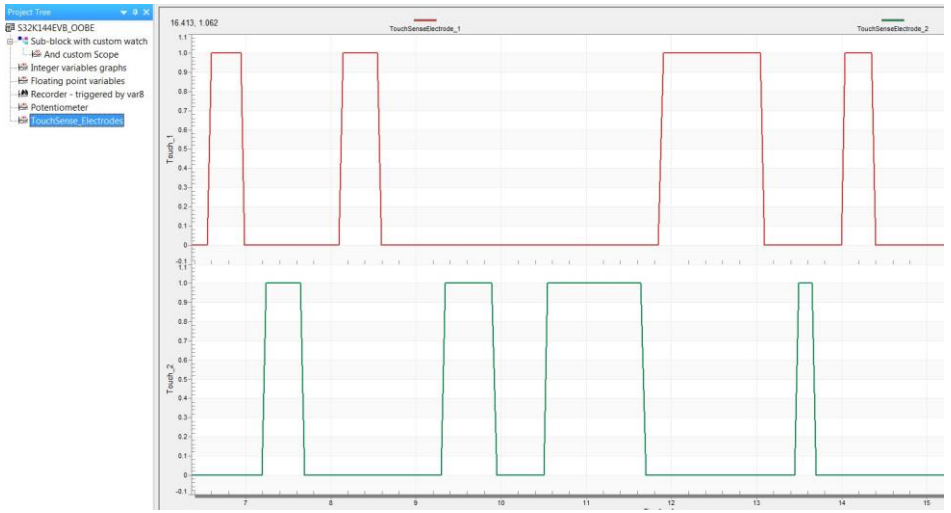
- Display main project panel „Project > View > Project Tree“.



- Display real-time oscilloscope graph examples such as „Potentiometer“ or „Touch Sense Electrodes“.



Analog values from potentiometer.



Responses from touch sense electrodes.

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. S32K148 EVB comes with the MSD Flash Programmer + Debug 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 J22 on position 2-3.
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 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 S32K148 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 + Debug 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 ***S32K148 EVB*** drive should be visible in the host file system

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

Introduction to OpenSDA: 2 of 2

The MSD Flash Programmer + Debug is a composite USB application that provides a virtual serial port, a debug interface and an easy and convenient way to program applications into the S32K MCU. It emulates a FAT16 file system, appearing as a removable drive in the host file system with a volume label of S32K148EVB. Raw binary and Motorola S-record files that are copied to the drive are programmed directly into the flash of the S32K 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 S32K148EVB drive.

The new application should now be running on the S32K148 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 files for the S32K148 to the S32K148EVB board drive 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 S32K148EVB virtual serial port. In Windows open Device Manager and look for the COM port named “OpenSDA – 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 (SW5) 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

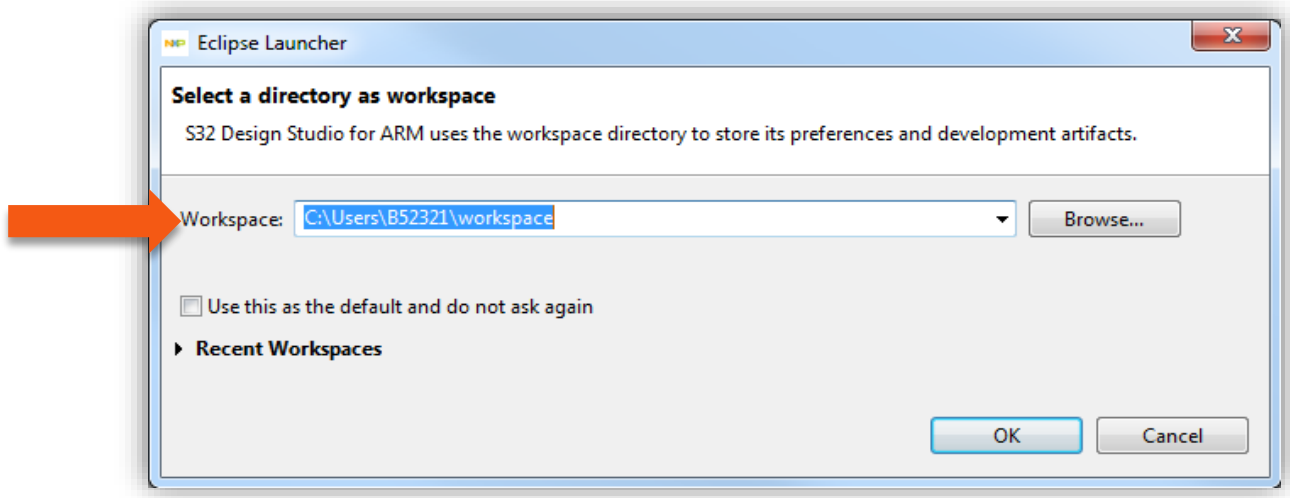
Download S32DS from ARM based MCUs from:

S32DS for ARM

CREATE A NEW PROJECT IN S32 DESIGN STUDIO

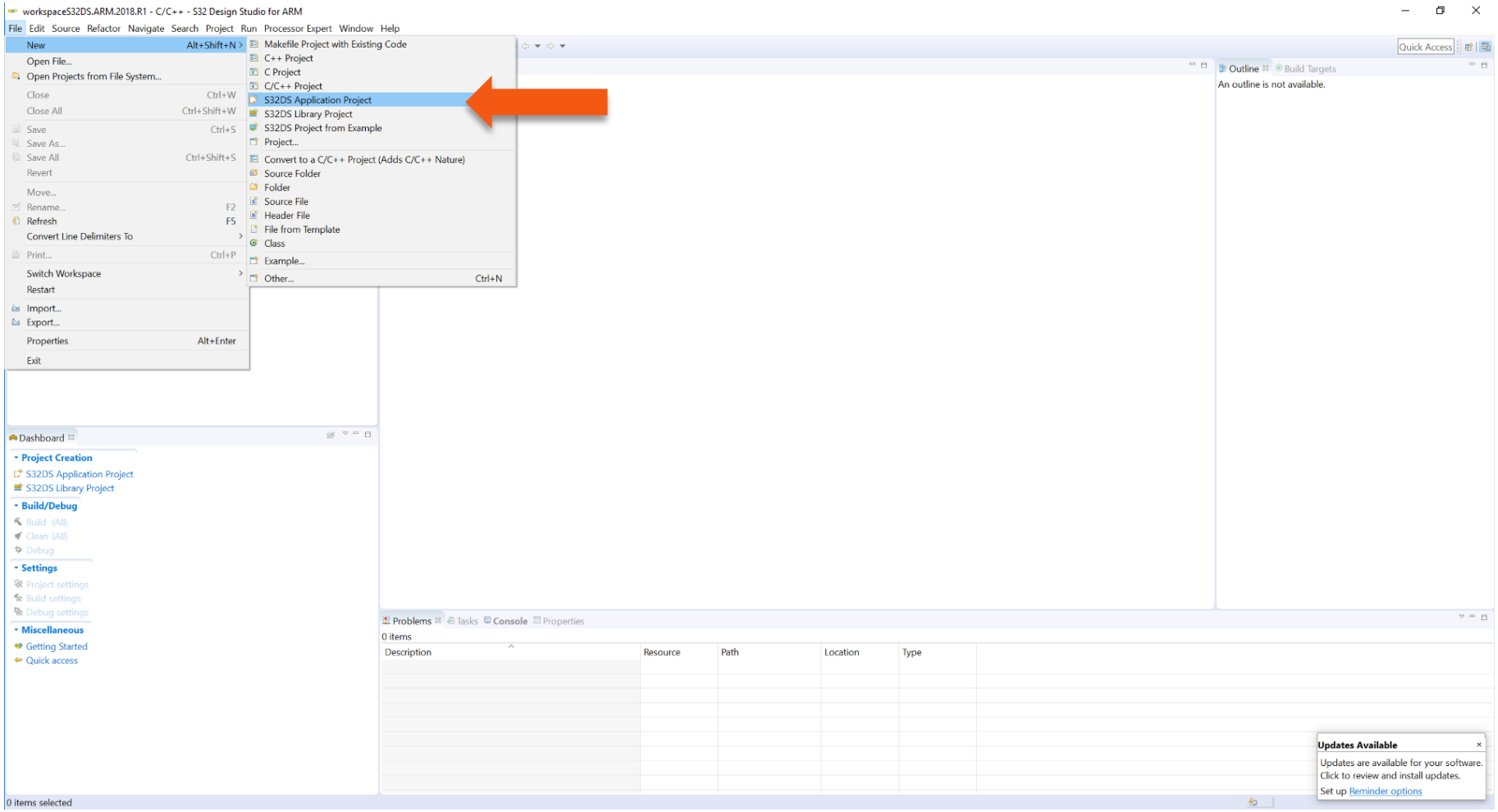
Create New Project: First Time – Select a Workspace

- 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



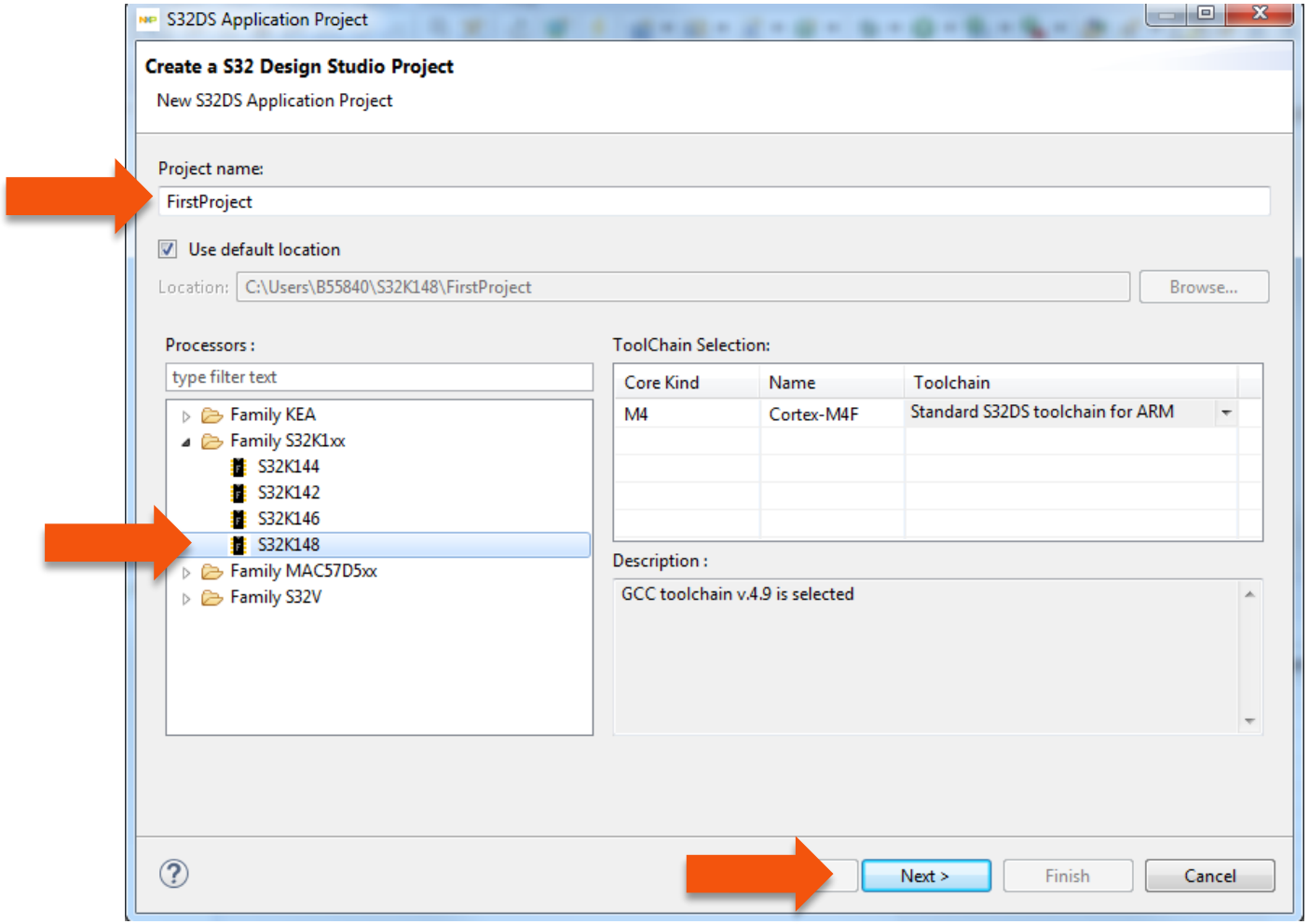
Create New Project: Top Menu Selection

- File – New – S32DS Application Project



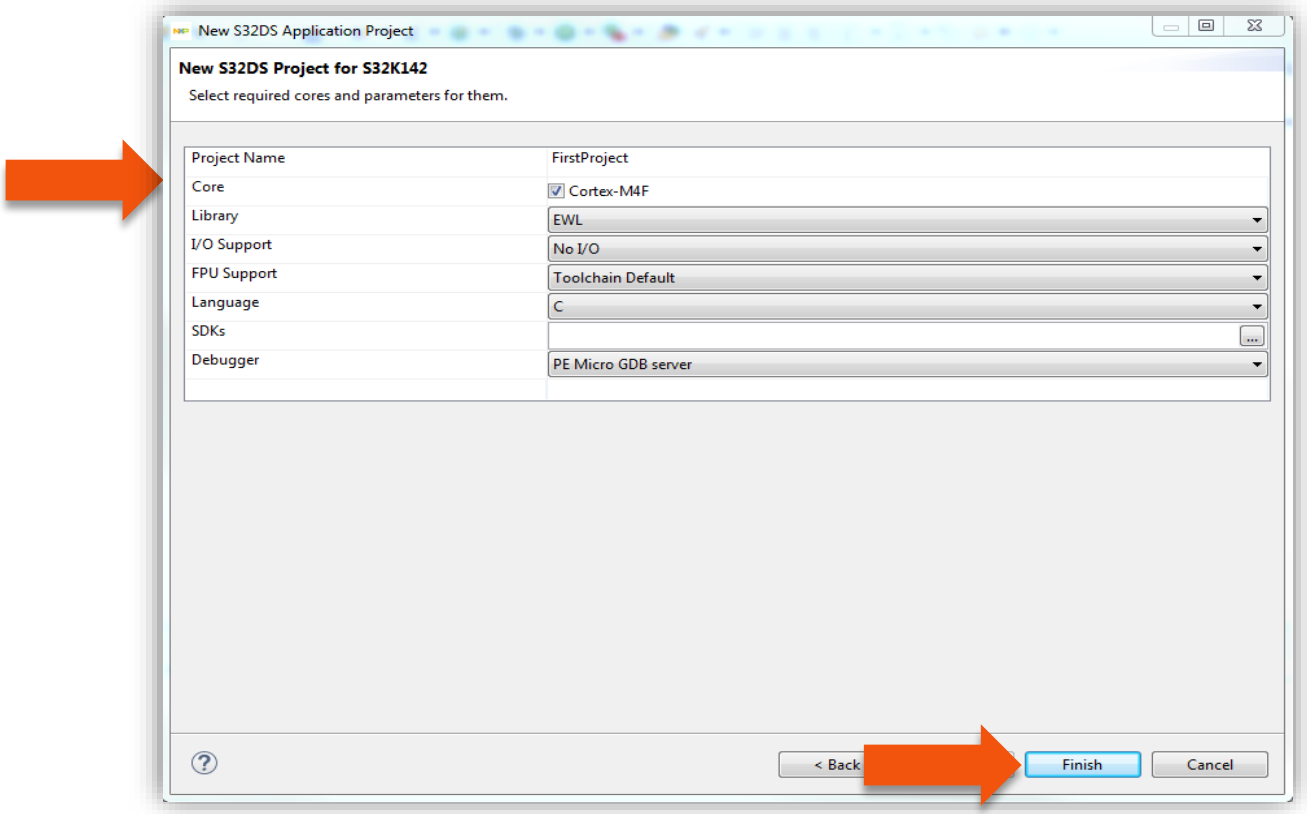
Create New Project: S32DS Project

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



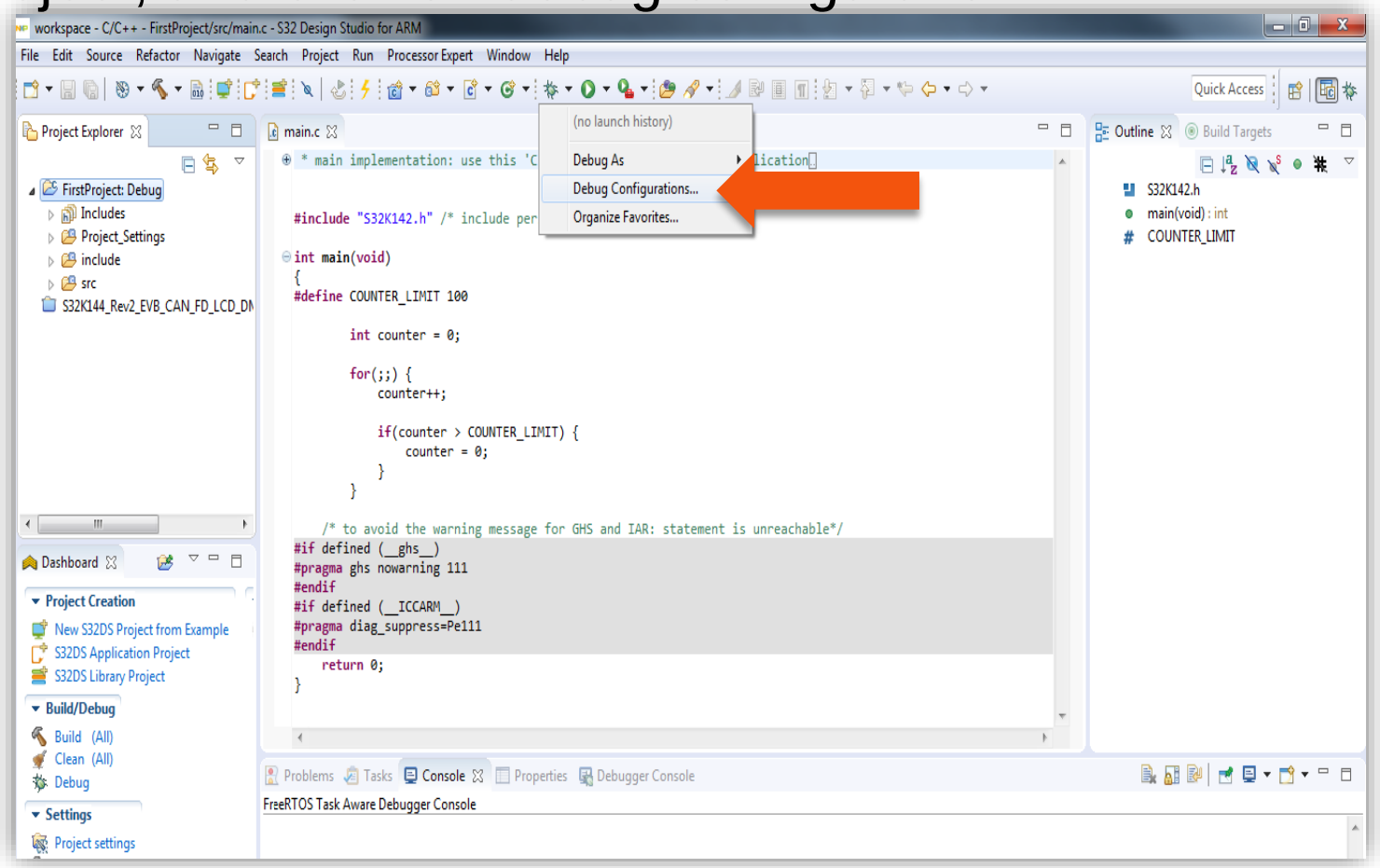
Create New Project: S32DS Project

- Select Debugger Support and Library Support
- Click Finish



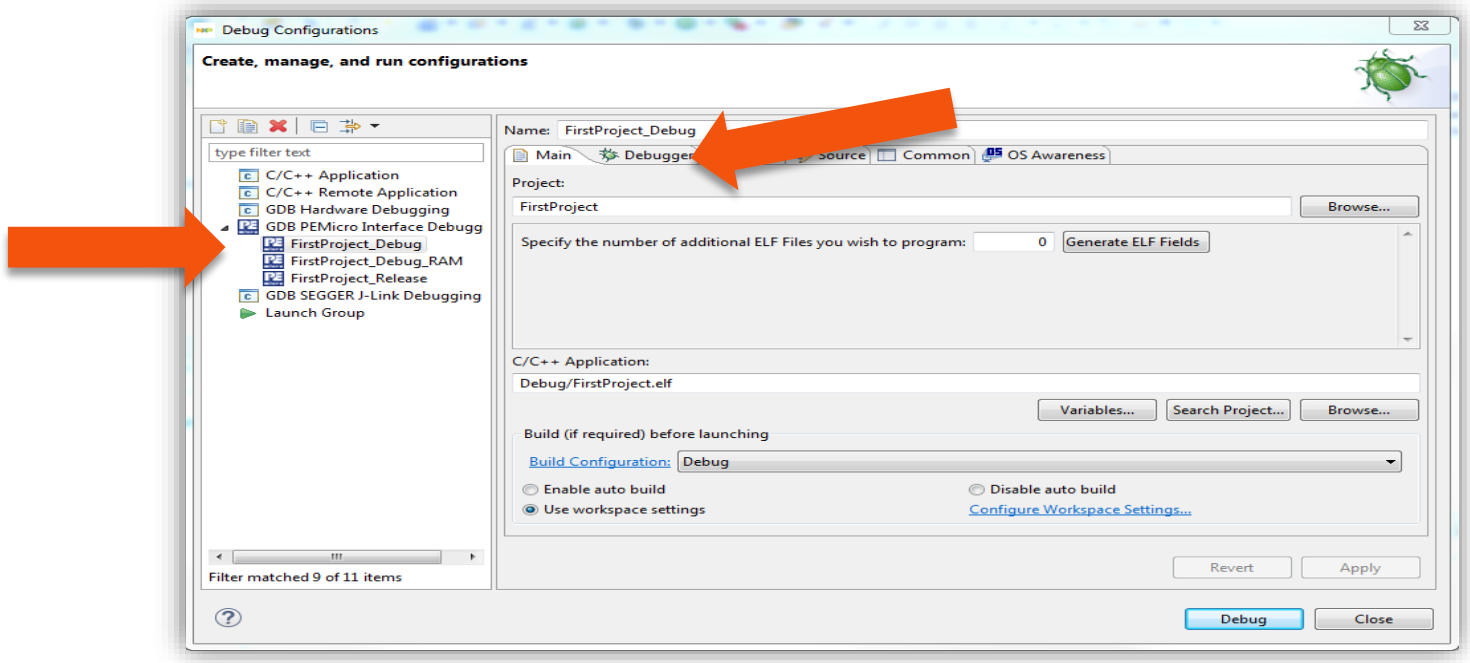
OpenSDA Configuration

- 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



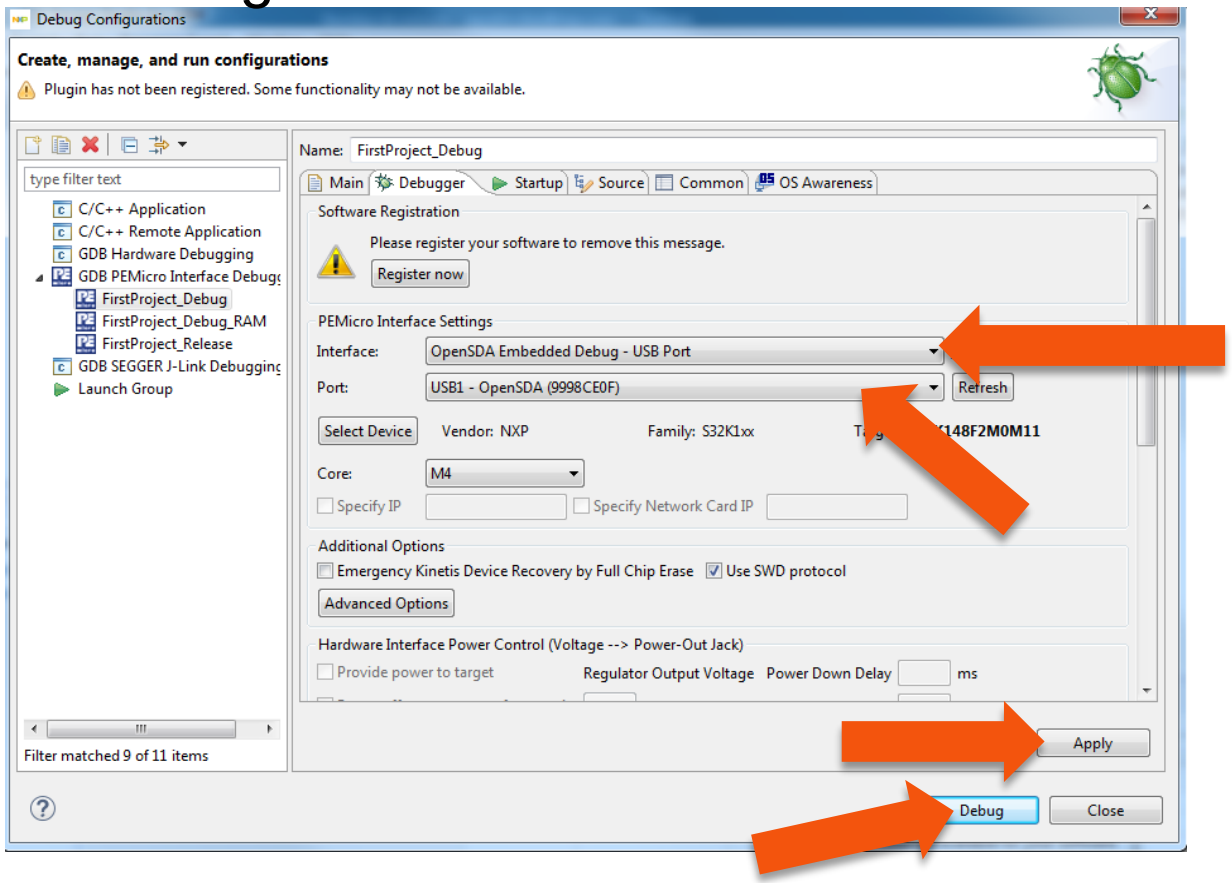
OpenSDA Configuration

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



OpenSDA Configuration

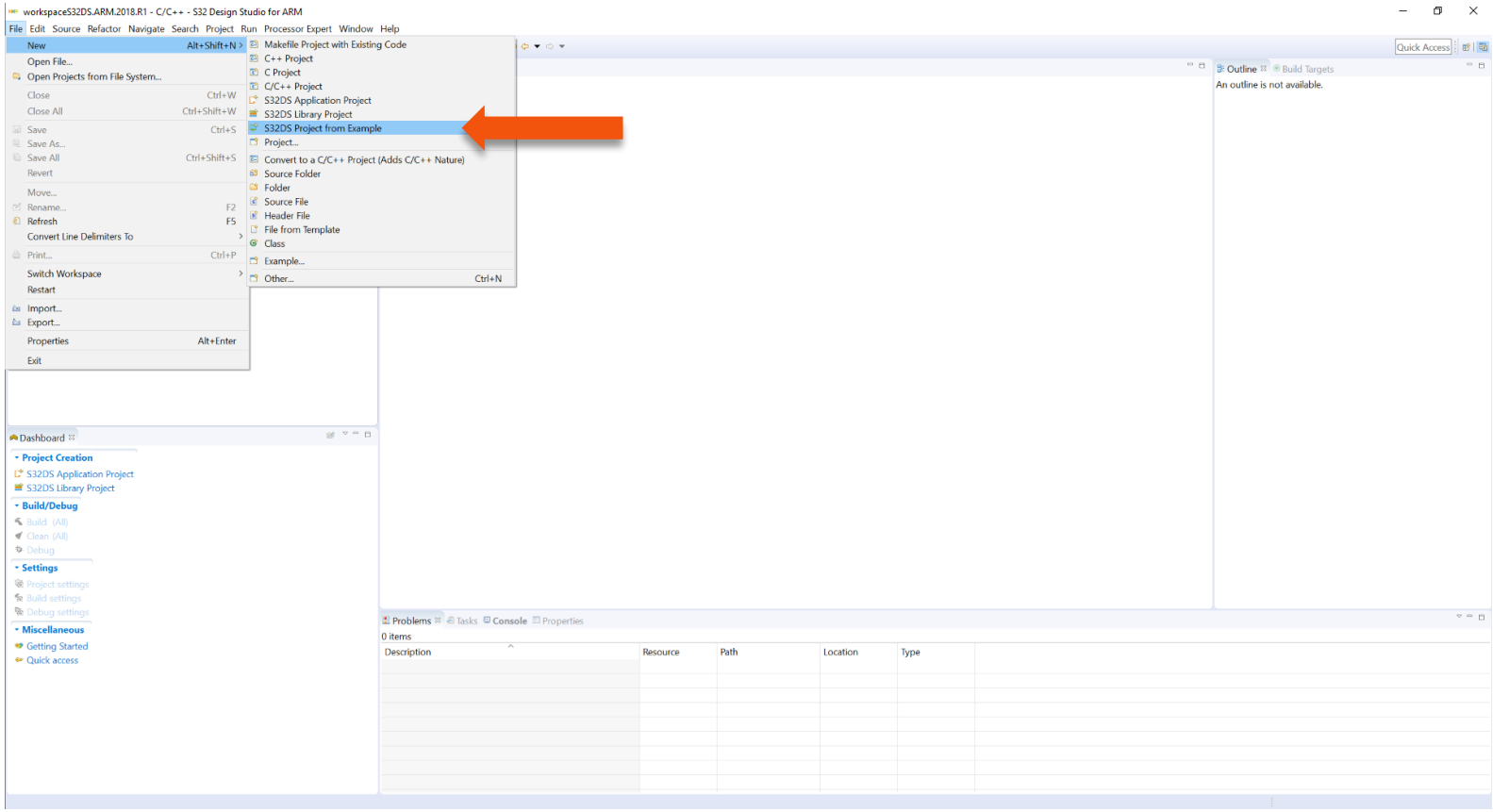
- Select OpenSDA as the interface, if your board is plugged should appear in the Port field.
- Click Apply and debug to finish.



CREATE AN EXAMPLE FROM SDK

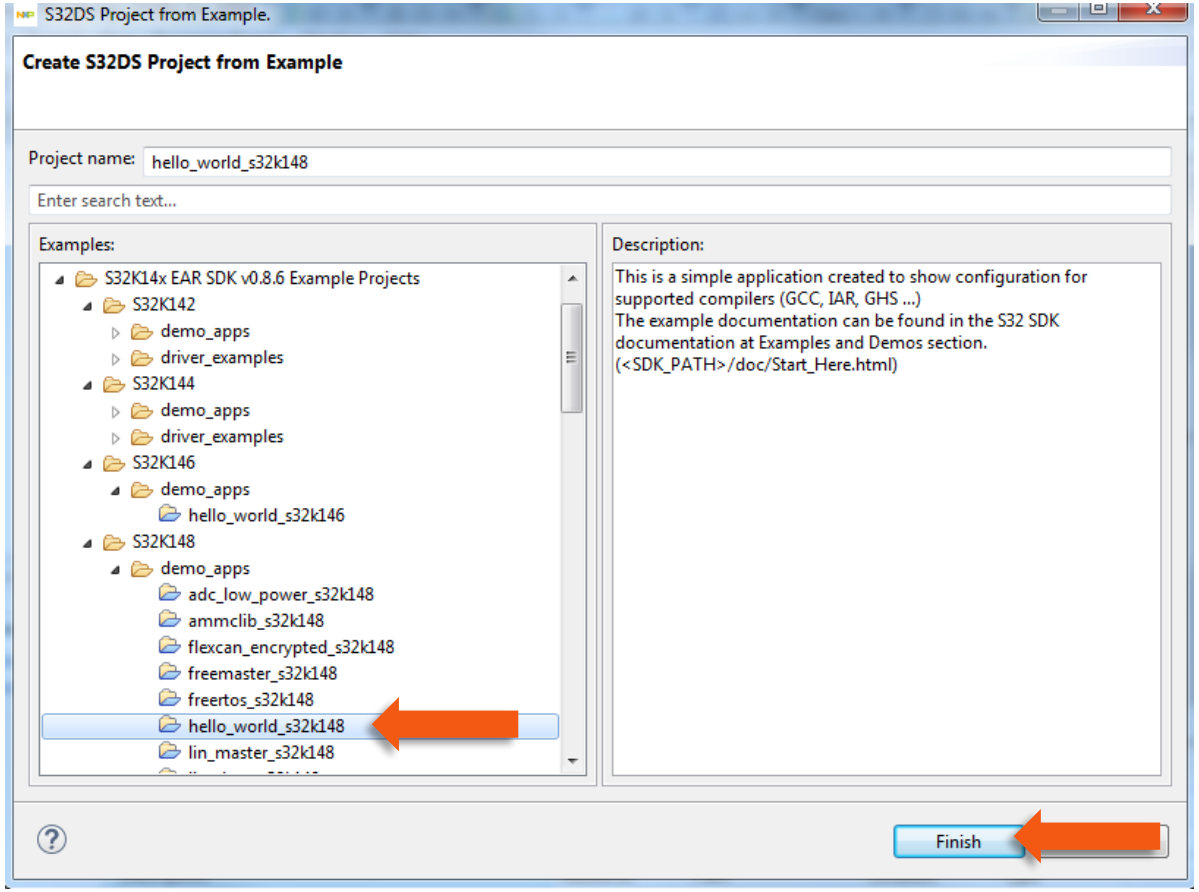
Creating example from SDK

- 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



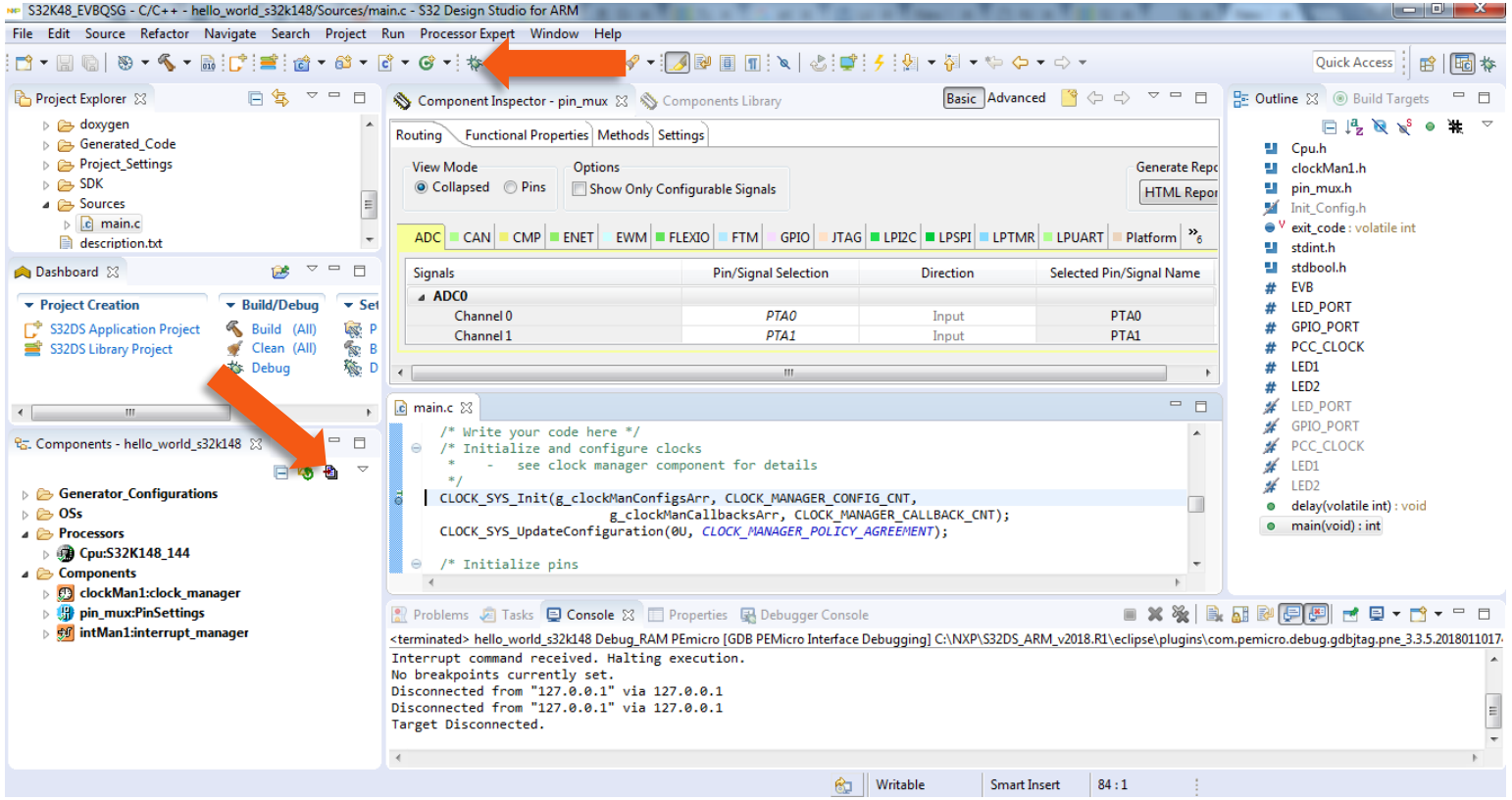
Creating example from SDK

- 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



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.



- 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

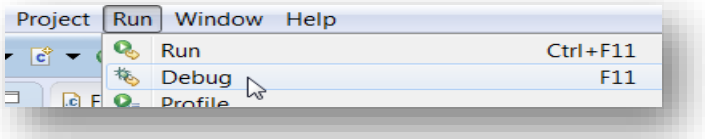
- For more information about the use of the SDK go click on the following link for an SDK training:

  https://community.nxp.com/docs/DOC-335153

DEBUG BASICS

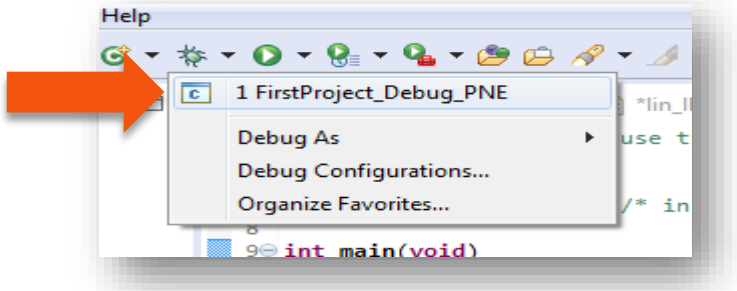
Debug Basics: Starting the Debugger

- 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)



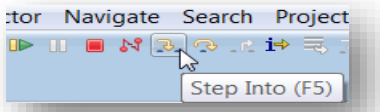
Note: This method currently selects the desktop target (*project.*elf) and gives an error. Do not use until this is changed.

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

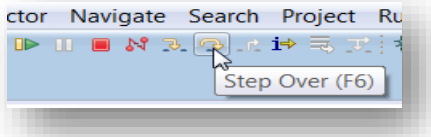


Debug Basics: Step, Run, Suspend, Resume

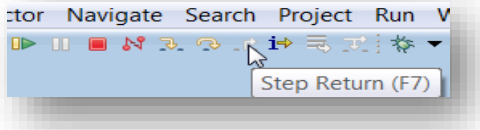
- Step Into (F5)



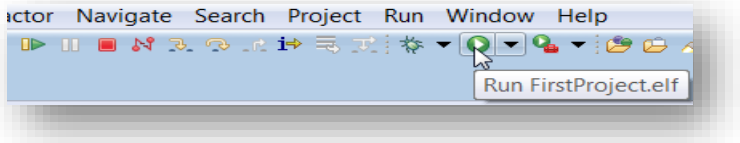
- Step Over (F6)



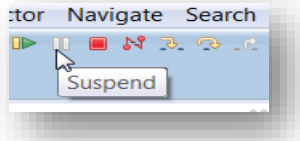
- Step Return (F7)



- Run



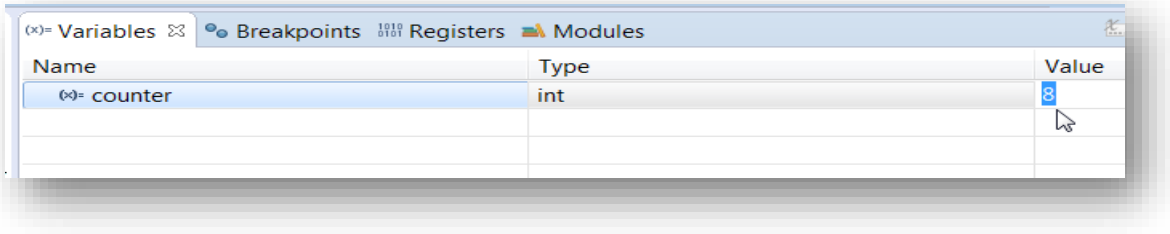
- Suspend



- Resume (F8)

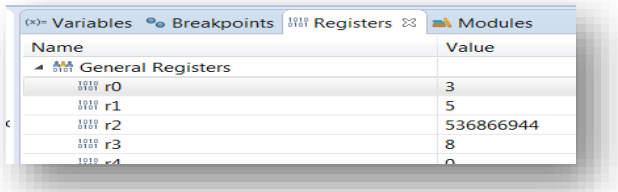
Debug Basics: View & Alter Variables

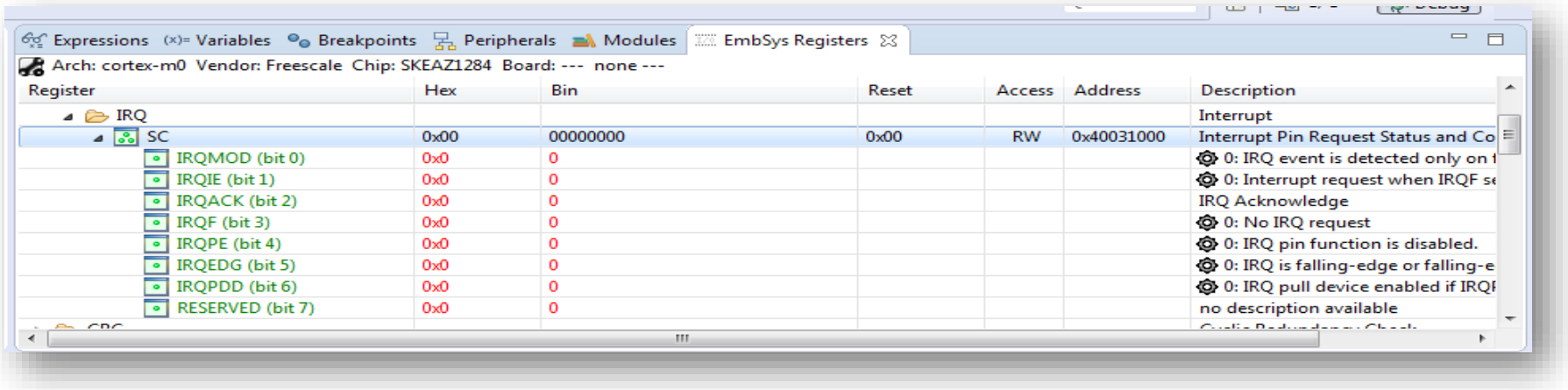
- View variables in “Variables” tab.
- Click on a value to allow typing in a different value.



Debug Basics: View & Alter Registers

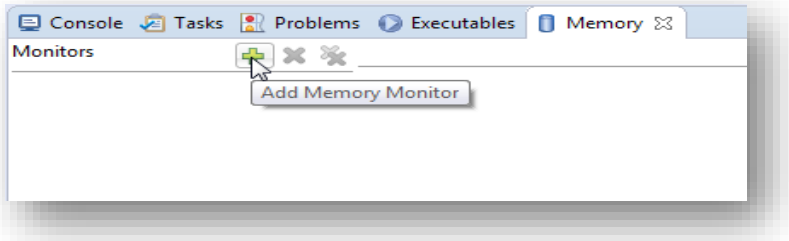
- View CPU registers in the “Registers” tab
- Click on a value to allow typing in a different value
- View peripheral registers in the EmbSys Registers tab



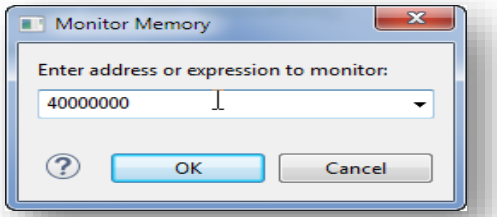


Debug Basics: View & Alter Memory

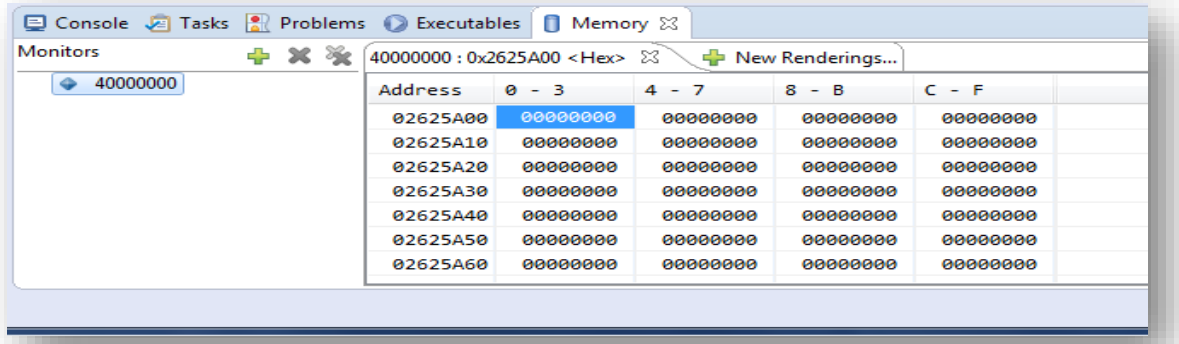
- Add Memory Monitor



- Select Base Address to Start at : 40000000



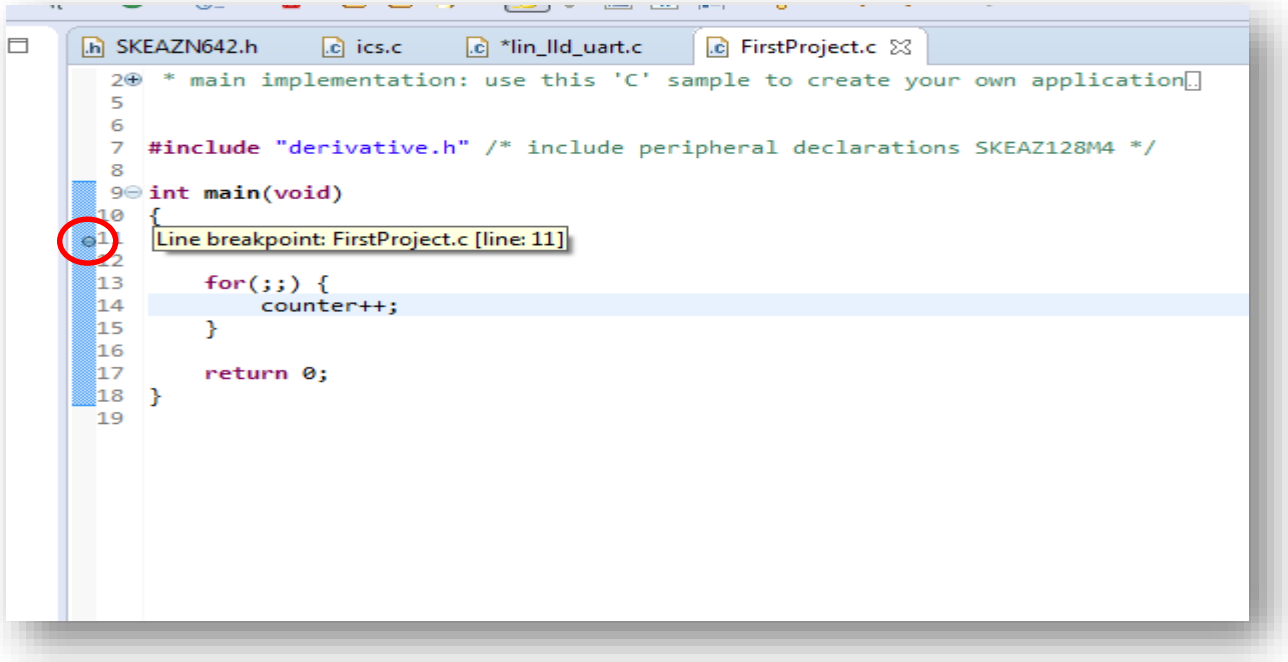
- View Memory



Debug Basics: Breakpoints

Add Breakpoint: Point and Click

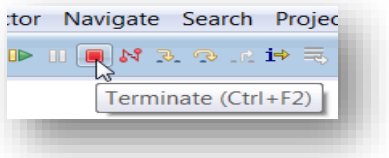
- light blue dot represents debugger breakpoint



Debug Basics: Reset & Terminate Debug Session

- Reset program counter

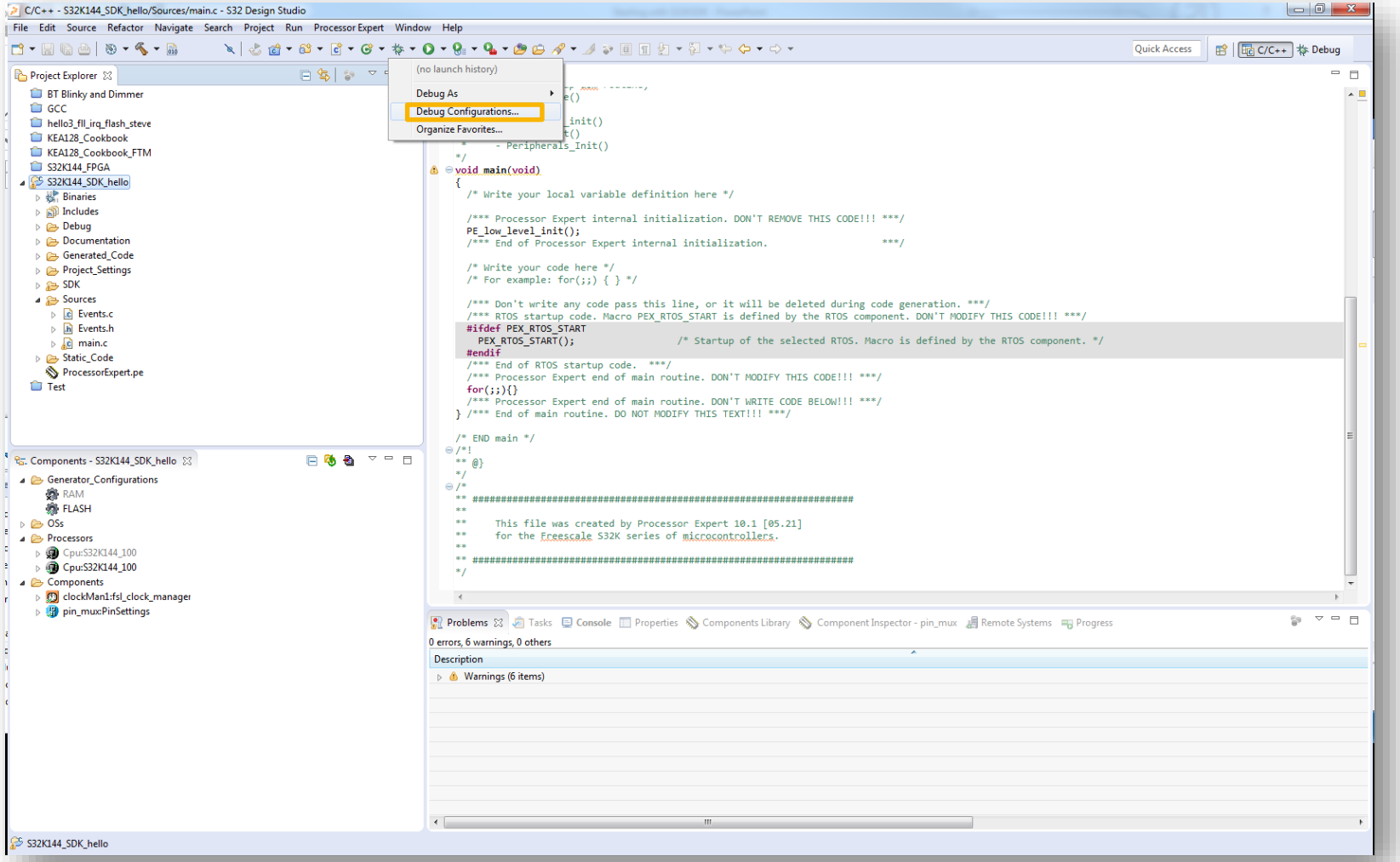
- Terminate Ctl+F2()



CREATE A P&E DEBUG CONFIGURATION (OPTIONAL)

New P&E debug configuration

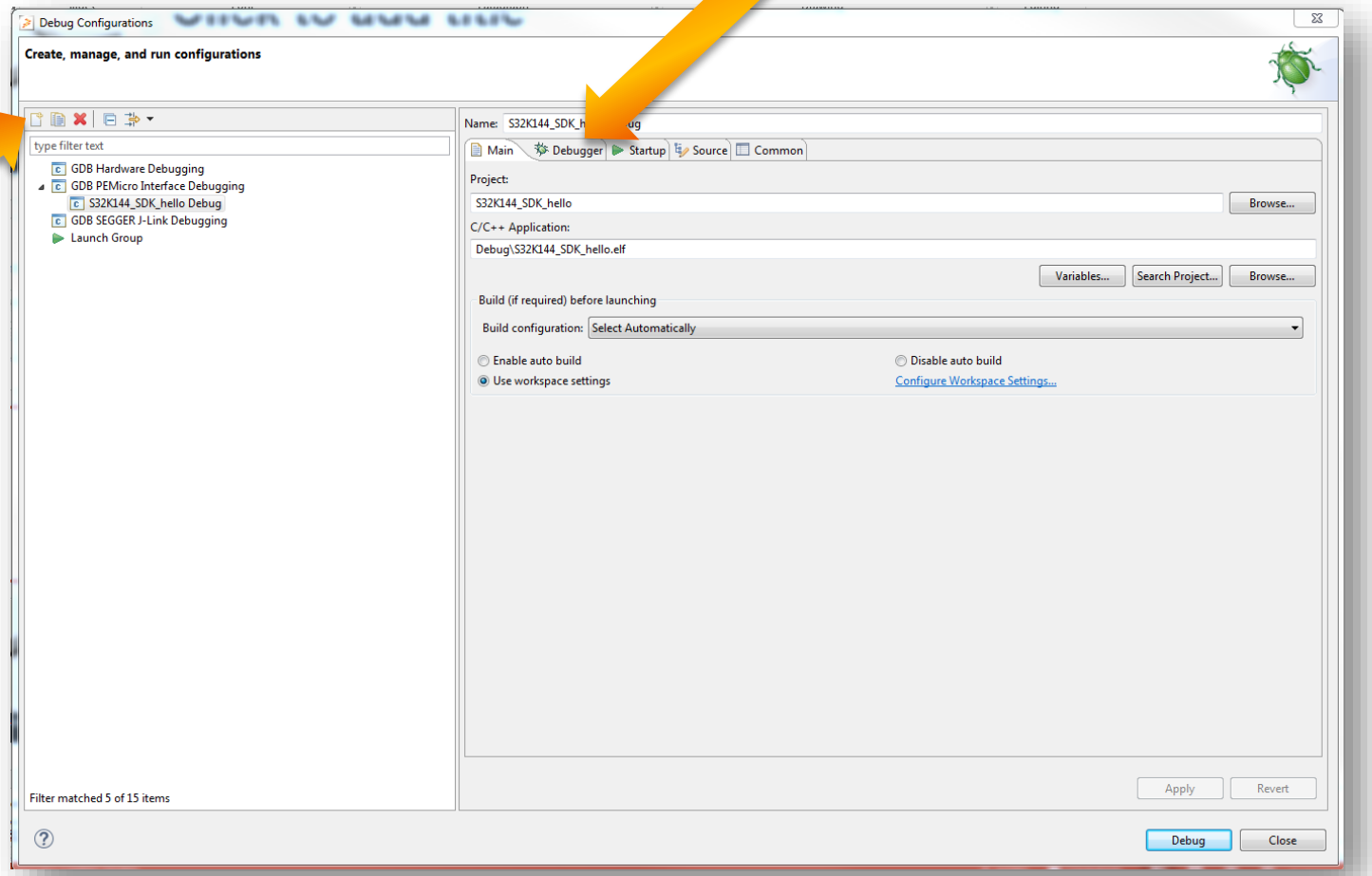
- Click in debug configurations



New P&E debug configuration

- Create a new P&E launch configuration

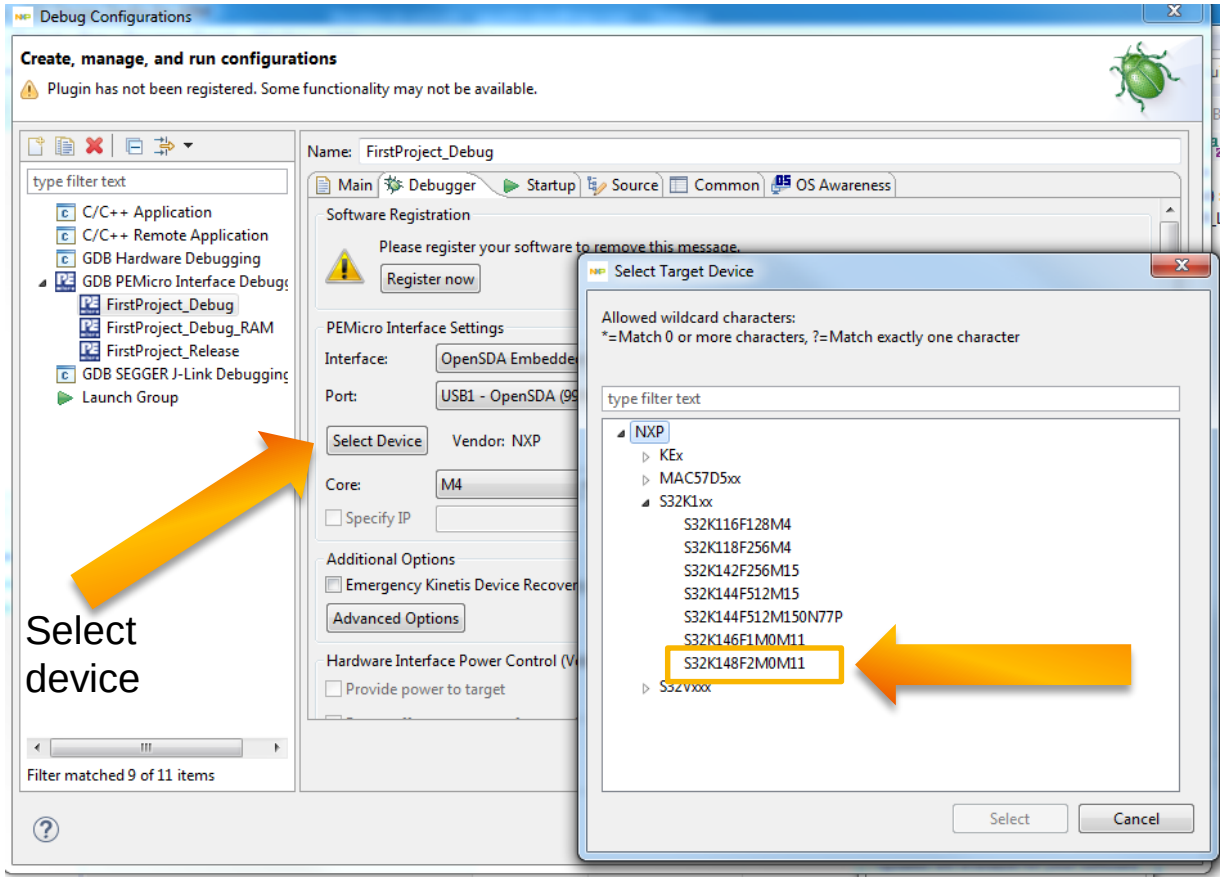
Click on the debugger tab.



Click to create a new P&E launch

New P&E debug configuration

- Select the device



- 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

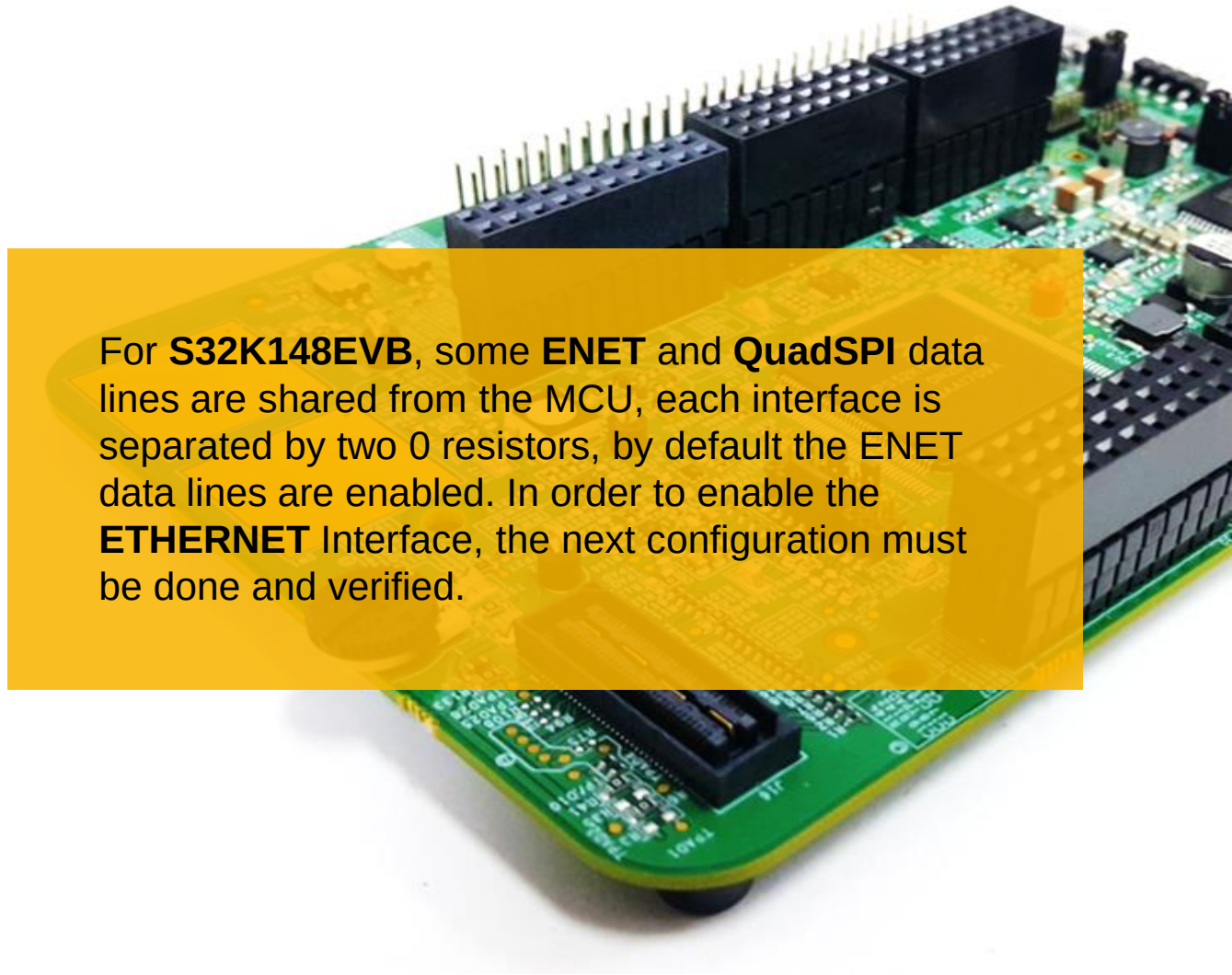
Using Ethernet on the S32K148EVB

- 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.

The TJA1101 Daughter Card

- The user need to acquire the ADTJA1101-RMII Daughter Card(TJA1101) in order to use ENET, as there is no Ethernet PHY on the board.

- The daughter card is connected into **J16** of S32K148EVB.

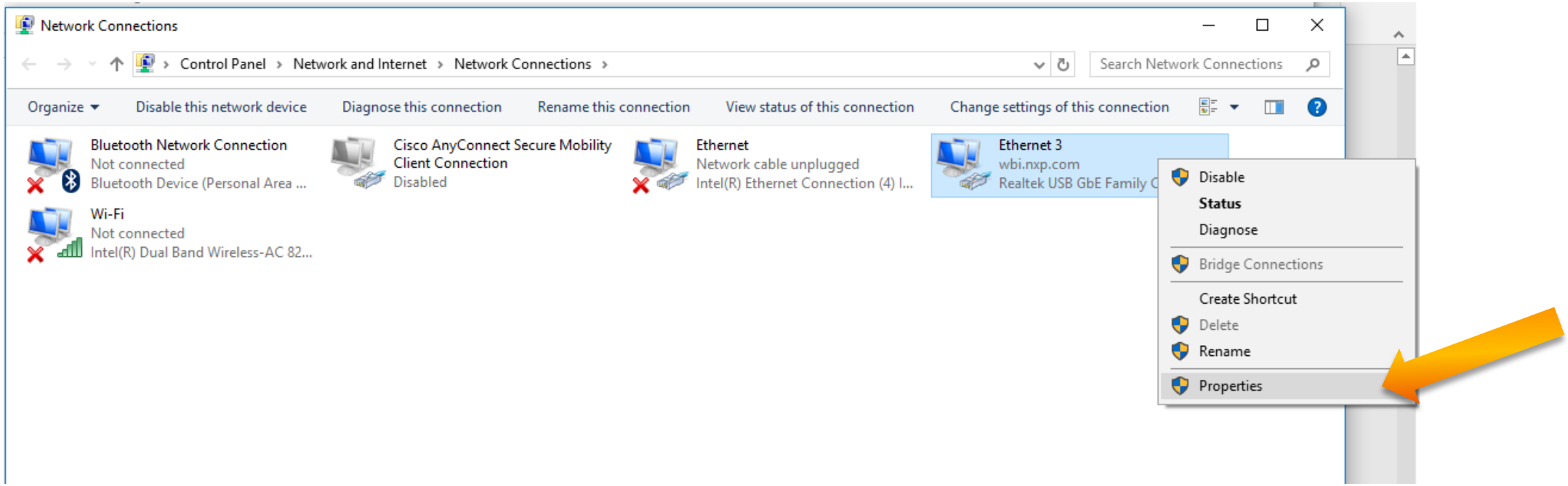
Running ENET on the S32K148 EVB

- The following slides will guide you through the tcp/ip ENET example available in the S32K1xx SDK. The example is only available in version EAR 0.8.6 onwards.

- In order to be able to get the example working you will need:
  - S32K148 EVB
  - Automotive ENET daughter card ADTJA1101-RMII
  - Media converter (Automotive Ethernet to 10/100 Base-TX) (https://store.intrepidcs.com/product-p/rad-moon.htm)
  - SDK version "S32K_SDK_EAR_0.8.6" or beyond
  - Any program able to set up a TCP client (e.g. SocketTest - Test My Socket download | SourceForge.net)

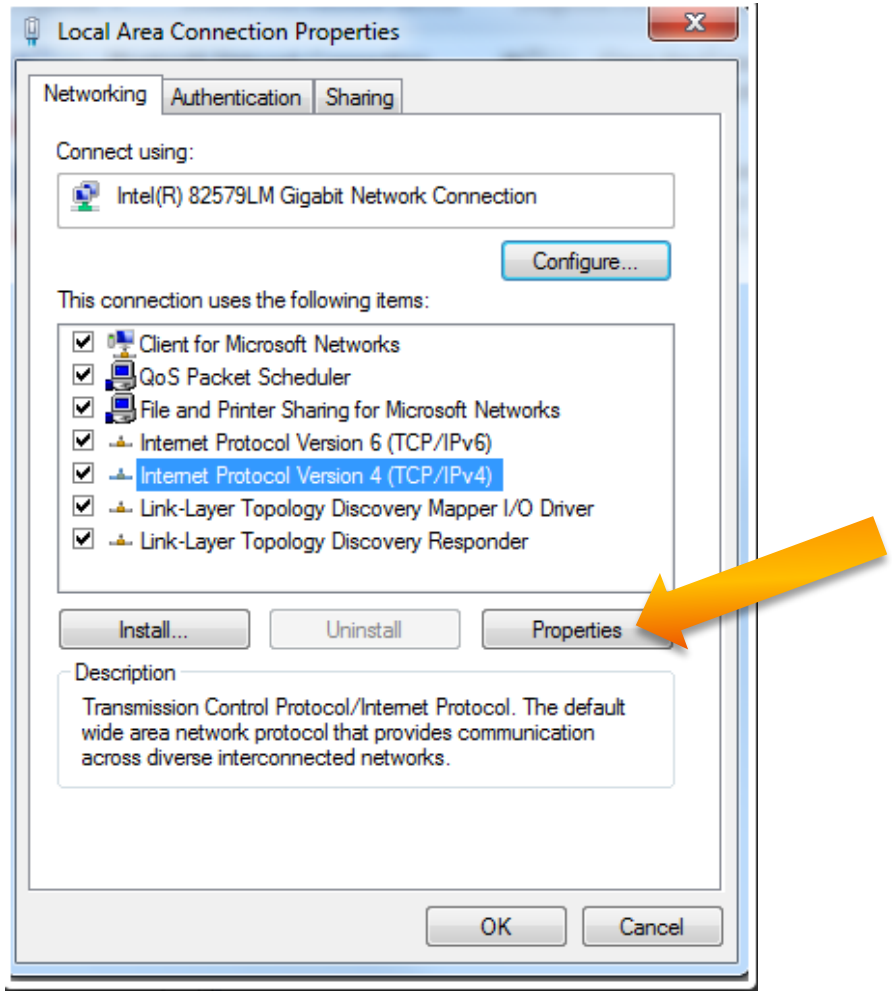
Running ENET on the S32K148 EVB

- Before trying the example you will need to change the IP of your computer in order to be an static IP address. Go to the network adapter settings of your computer, right click the local area network of your computer and select properties.



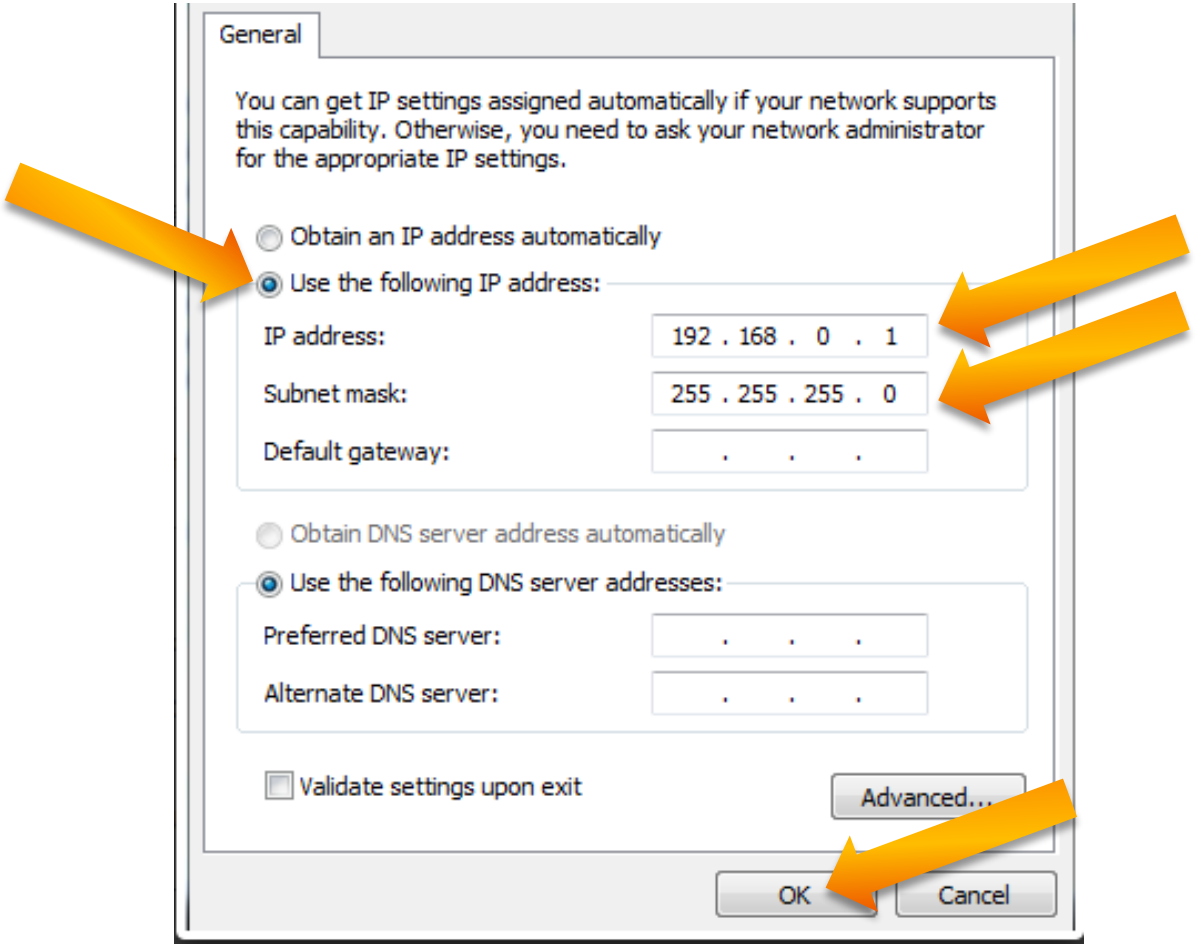
Running ENET on the S32K148 EVB

- The following window will open. Select the option ***Internet Protocol Version 4 (TCP/IPv4)*** and then click on properties.



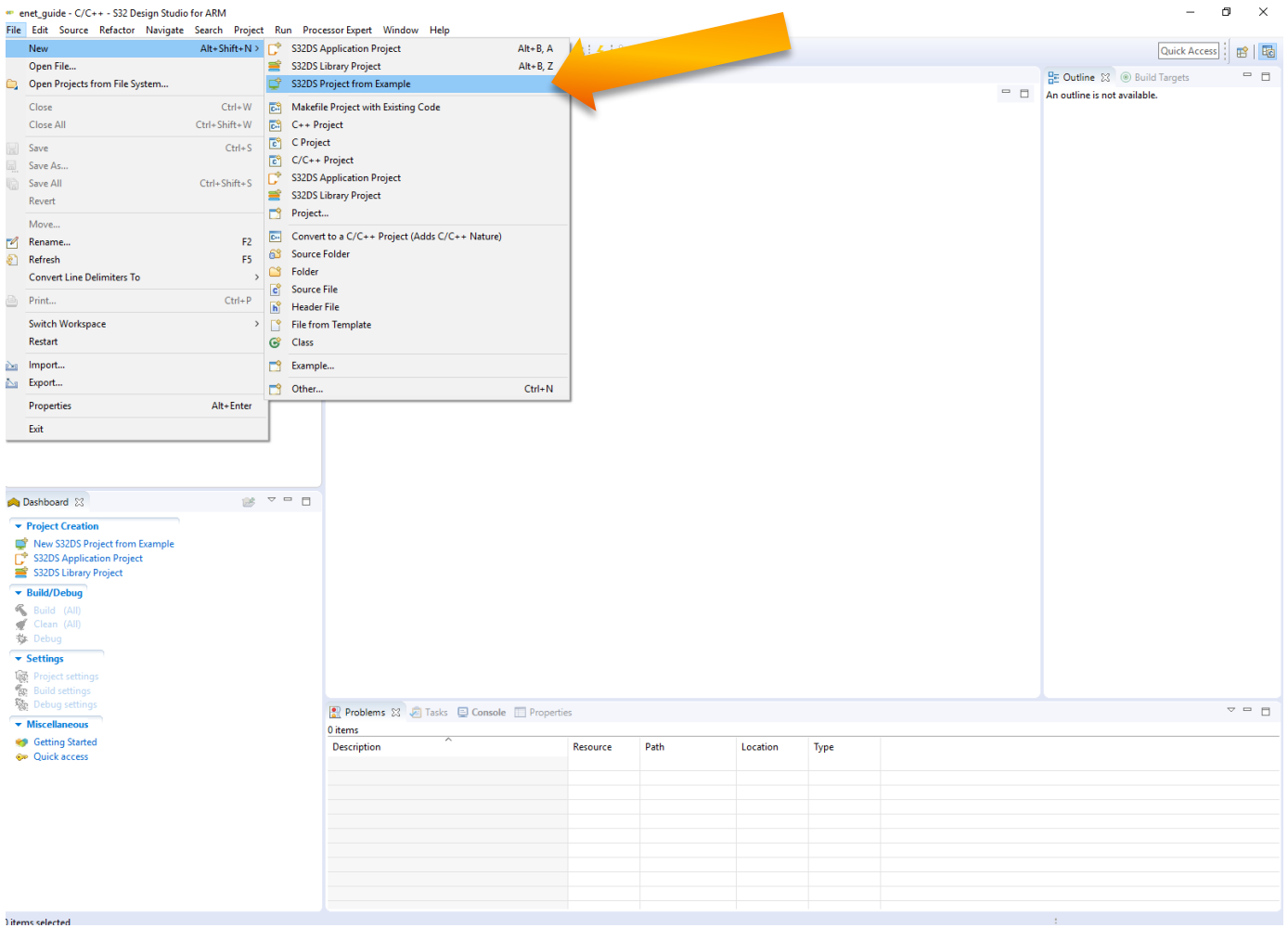
Running ENET on the S32K148 EVB

- The SDK example assigns an IP value of ***192.168.0.200*** to the S32K148. The computer must have an IP in the same network.
- Fill out the values just as in the following figure:



Running ENET on the S32K148 EVB

- Once the PC setup is done. Import the SDK example into the S32DS. Open the file tab and click on the create new project from example option.

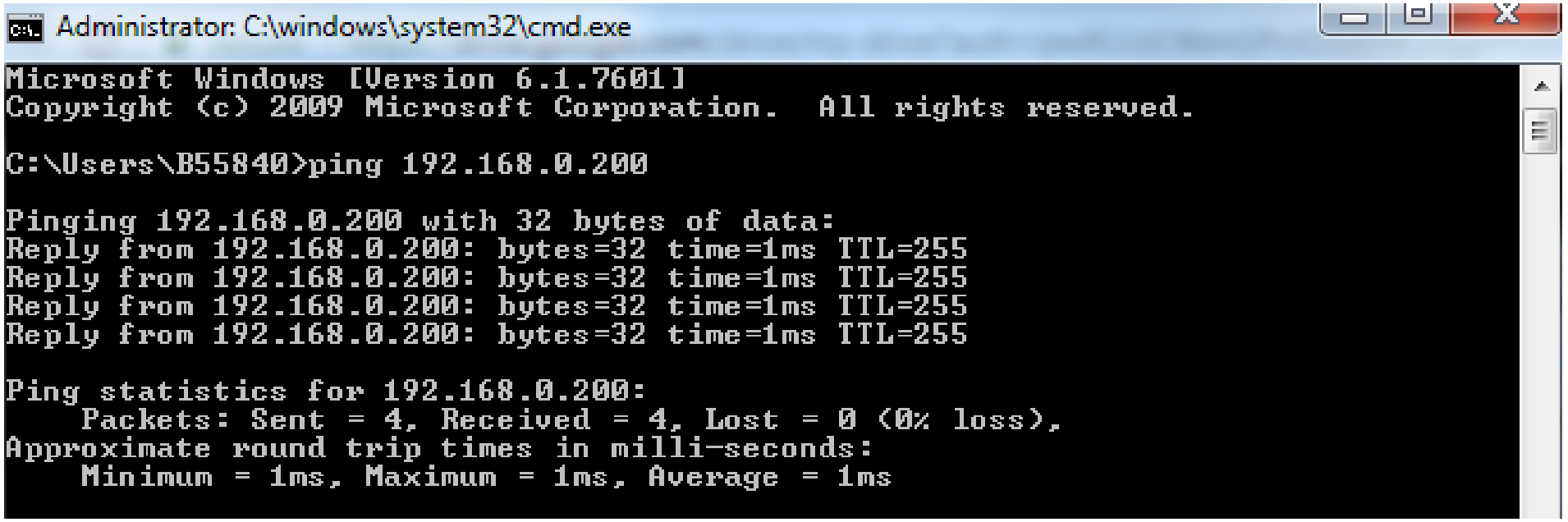


Running ENET on the S32K148 EVB

- Compile the example and download it to the S32K148 EVB with the ADTJA1101-RMII already connected. The media converter should also be connected between the board and the computer. As shown in the picture.

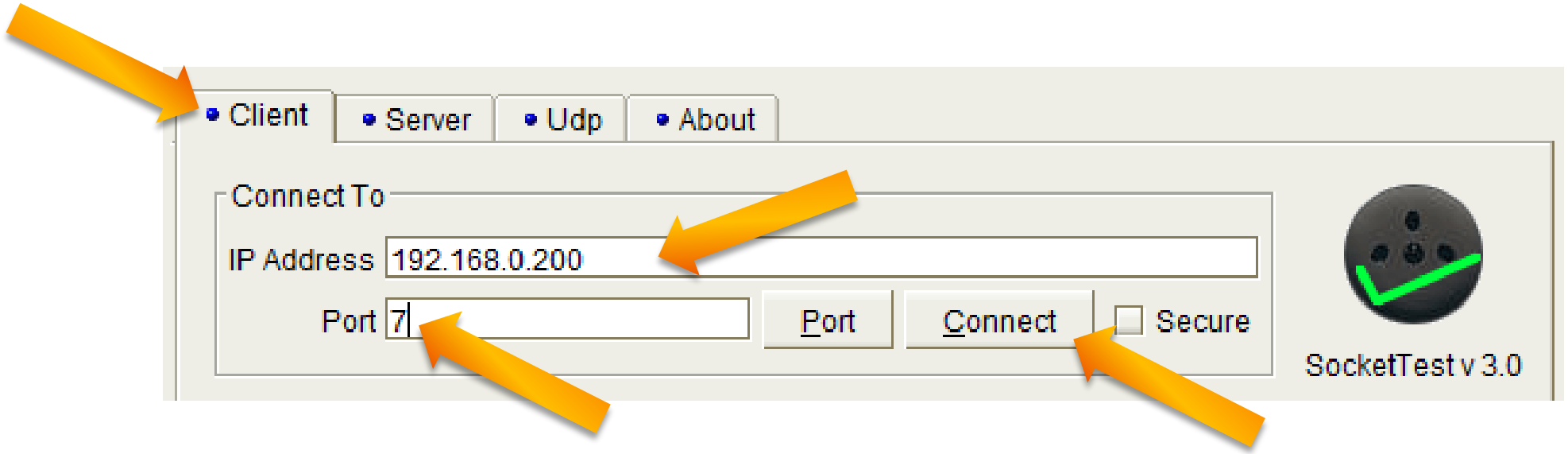
Running ENET on the S32K148 EVB

Once you have everything connected, run the example. It should not have any issue. There are two ways to verify that the example is correctly running. The first one is to "ping" the board using the windows command of the windows console. Use the command ***ping 192.168.0.200*** and the board should answer as shown in the following figure:



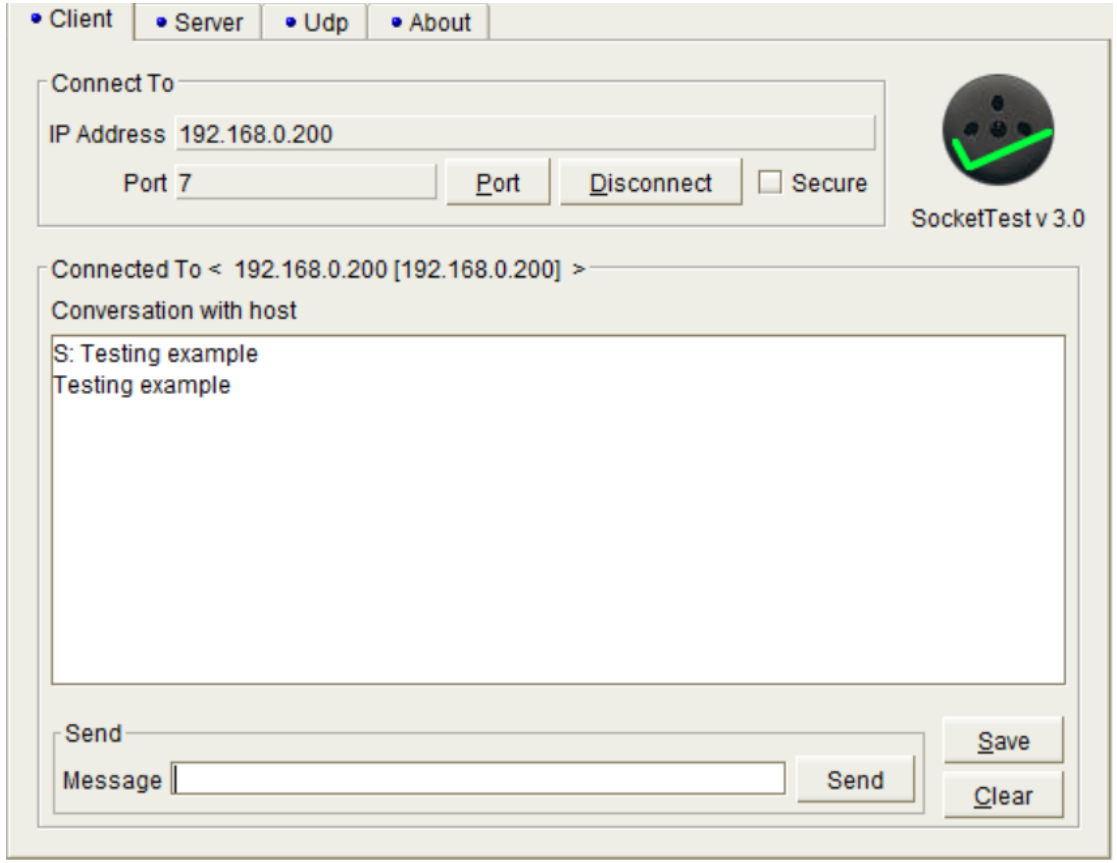
Running ENET on the S32K148 EVB

- The second way is to use the SocketTest program to echo the board at a TCP level. Open the SocketTest (or the TCP client program you are using), fill the options with the following information and click on connect:
  - IP Address: 192.168.0.200
  - Port: 7



Running ENET on the S32K148 EVB

- Connection should be established without any issue. Once the computer is connected with the S32K148, you can send any message and the S32K148 will make an echo of anything you send. Just as shown in the following image:



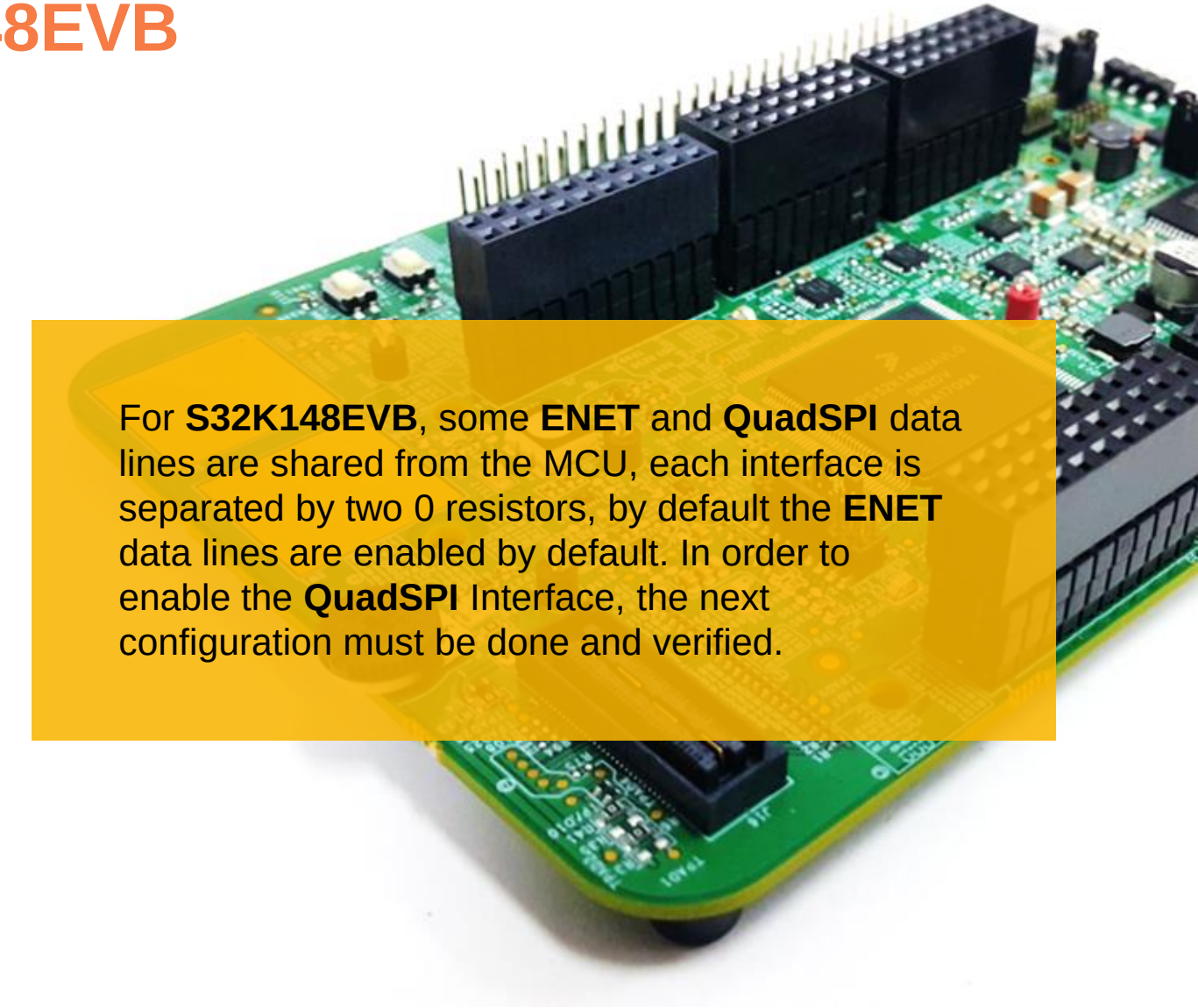
USING QUADSPI

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. Just run it and start testing the module.

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