

Software Installation Guide for S32 Design Studio IDE (S32DS):

- FRDM-KEAZ128Q80
- FRDM-KEAZ64Q64
- FRDM-KEAZN32Q64

Ultra-Reliable MCUs for Industrial and Automotive

www.freescale.com/FRDM-KEA

Contents:

- [Installing S32 Design Studio](#)
- [How to Create a New Project in S32DS](#)
- [How to import SDK evaluation drivers](#)
- [Basic Configuration and Debug](#)
- [Installing FreeMaster GUI Monitor](#)



Install S32 Design Studio IDE (step by step)



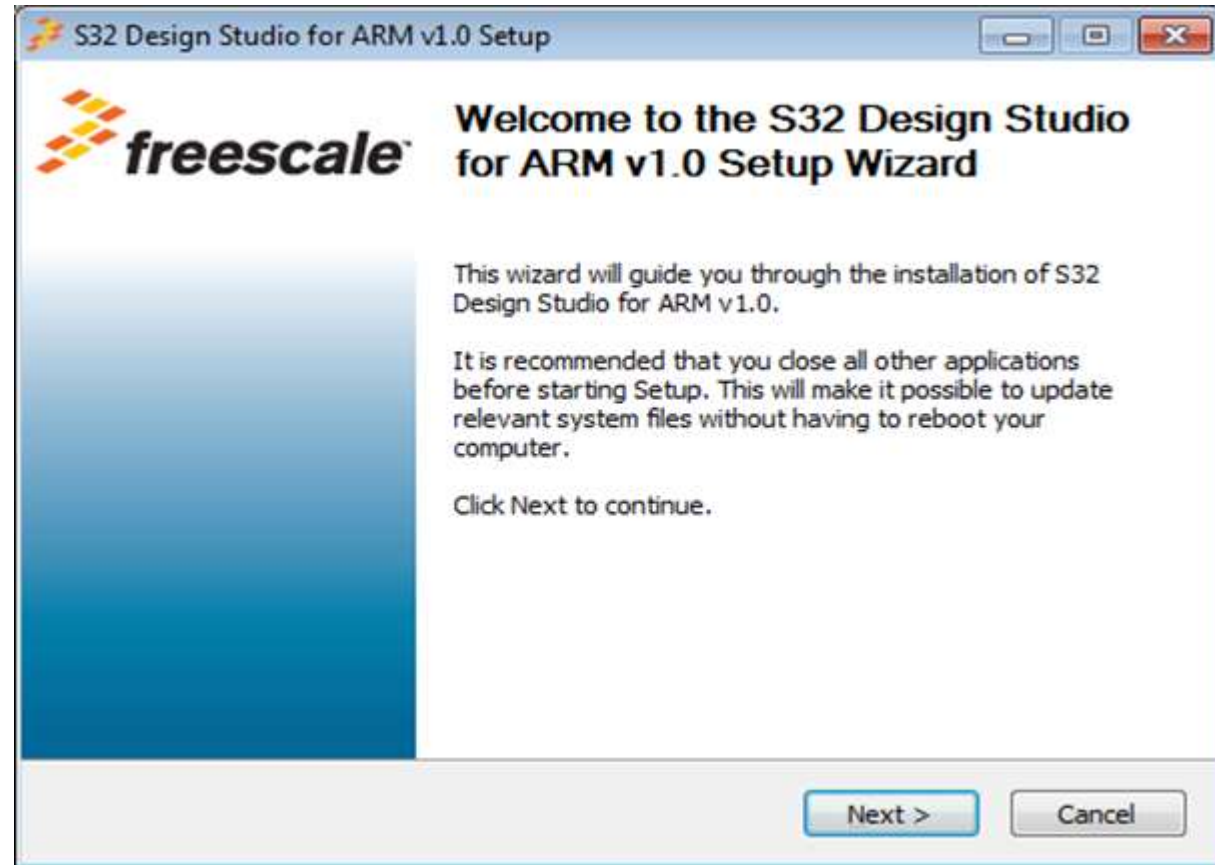
Download and Install S32 Design Studio Software

- From Downloads folder, double-click on **S32_ARM_Win32_v1.0_b150626.exe** to start installation
- Click through any administrative privilege issues resulting from unknown software publisher.



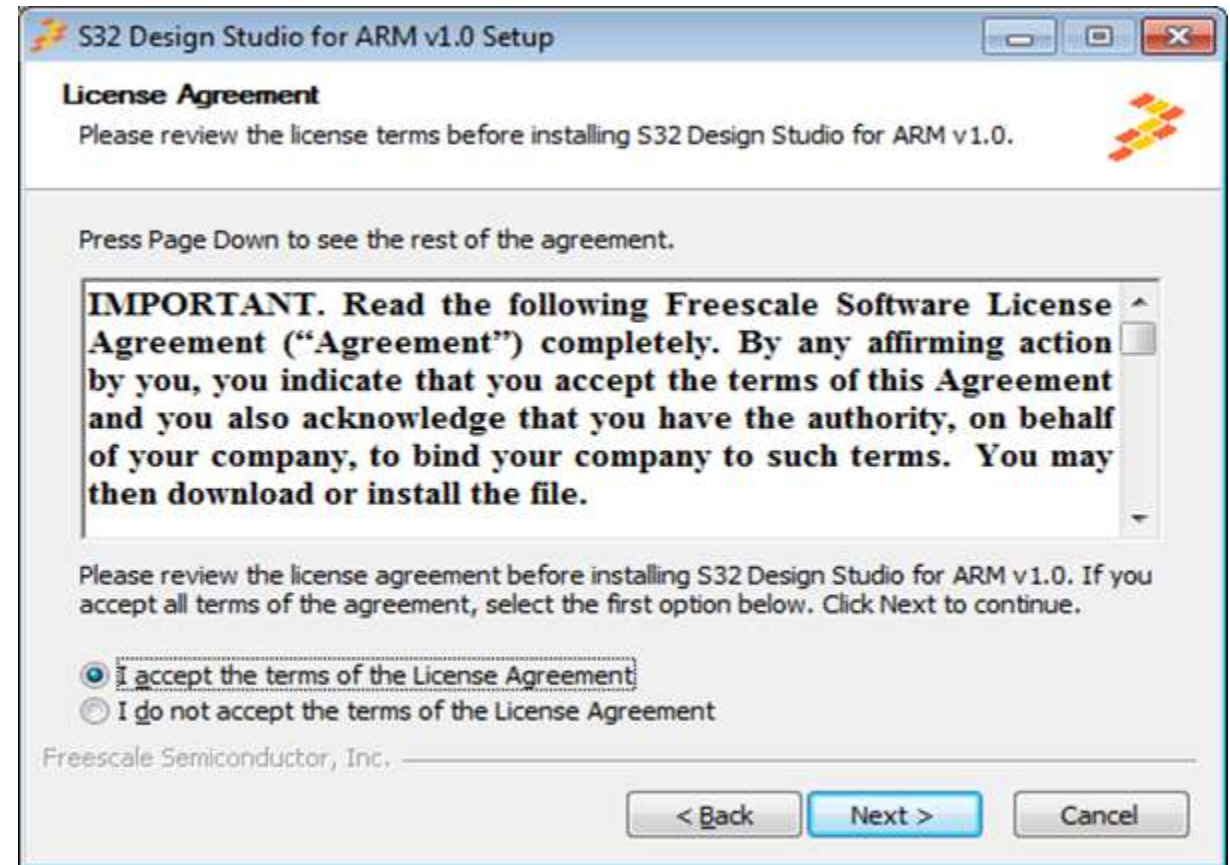
Download and Install S32 Design Studio Software

- An Installer welcome window will be displayed, click Next to continue



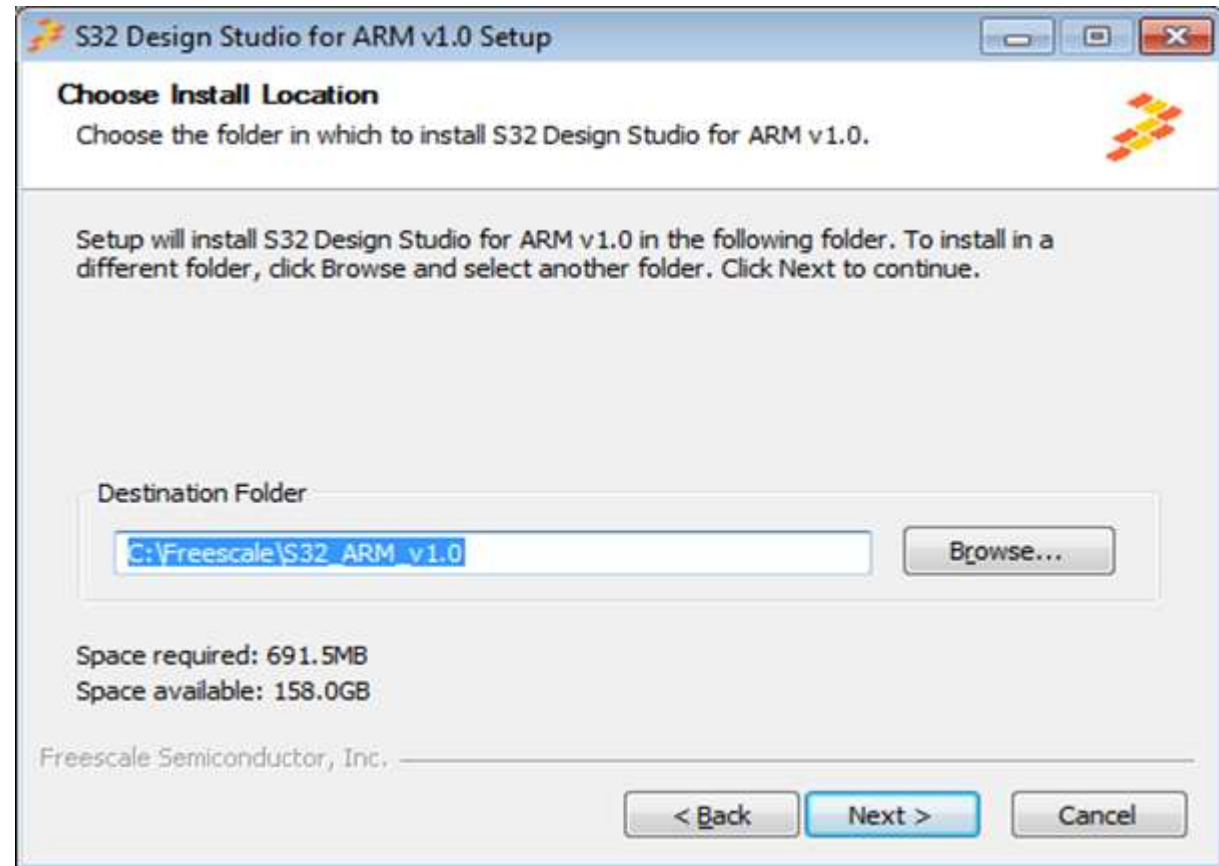
Download and Install S32 Design Studio Software

- Read license agreement, then select the radio button acknowledging the license agreement terms and click **Next** to continue.



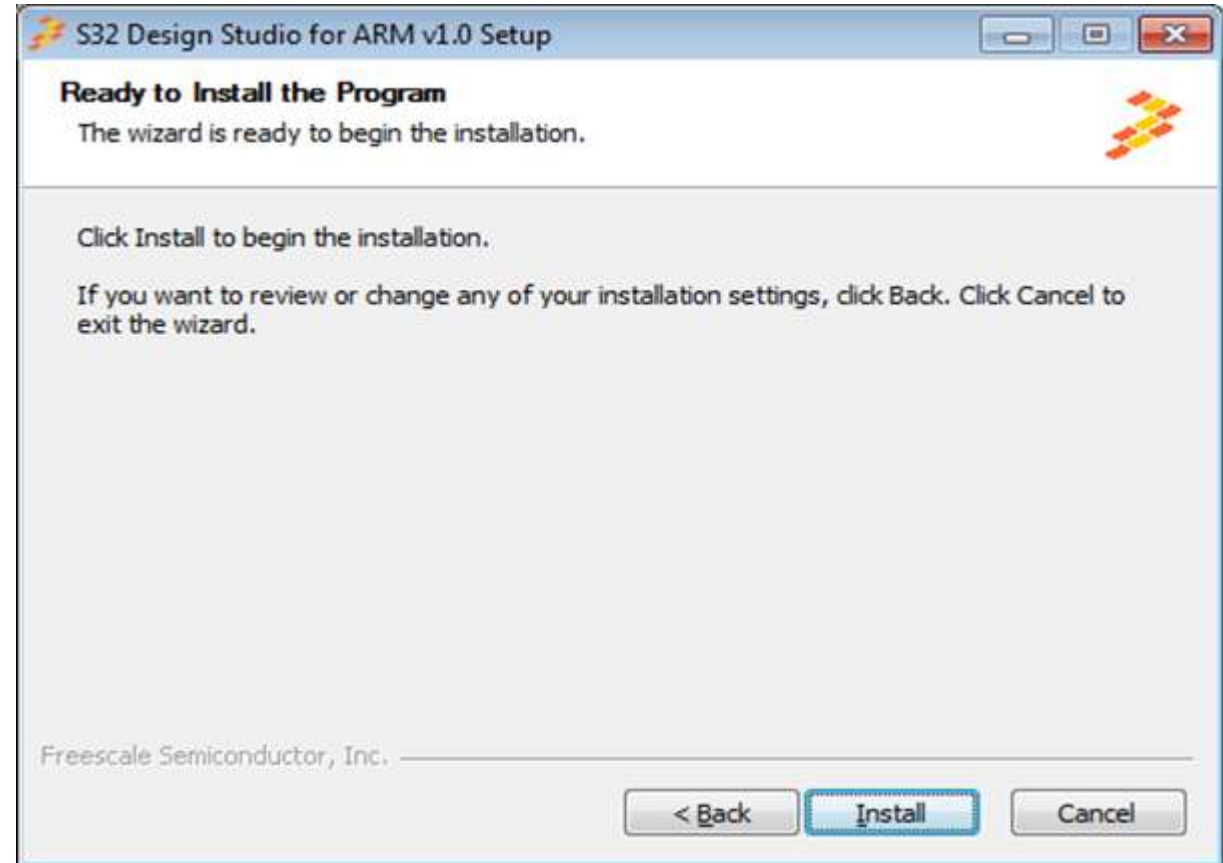
Download and Install S32 Design Studio Software

- Click **Next** to accept the default installation location (could be changes, but recommended to install into path without spaces).



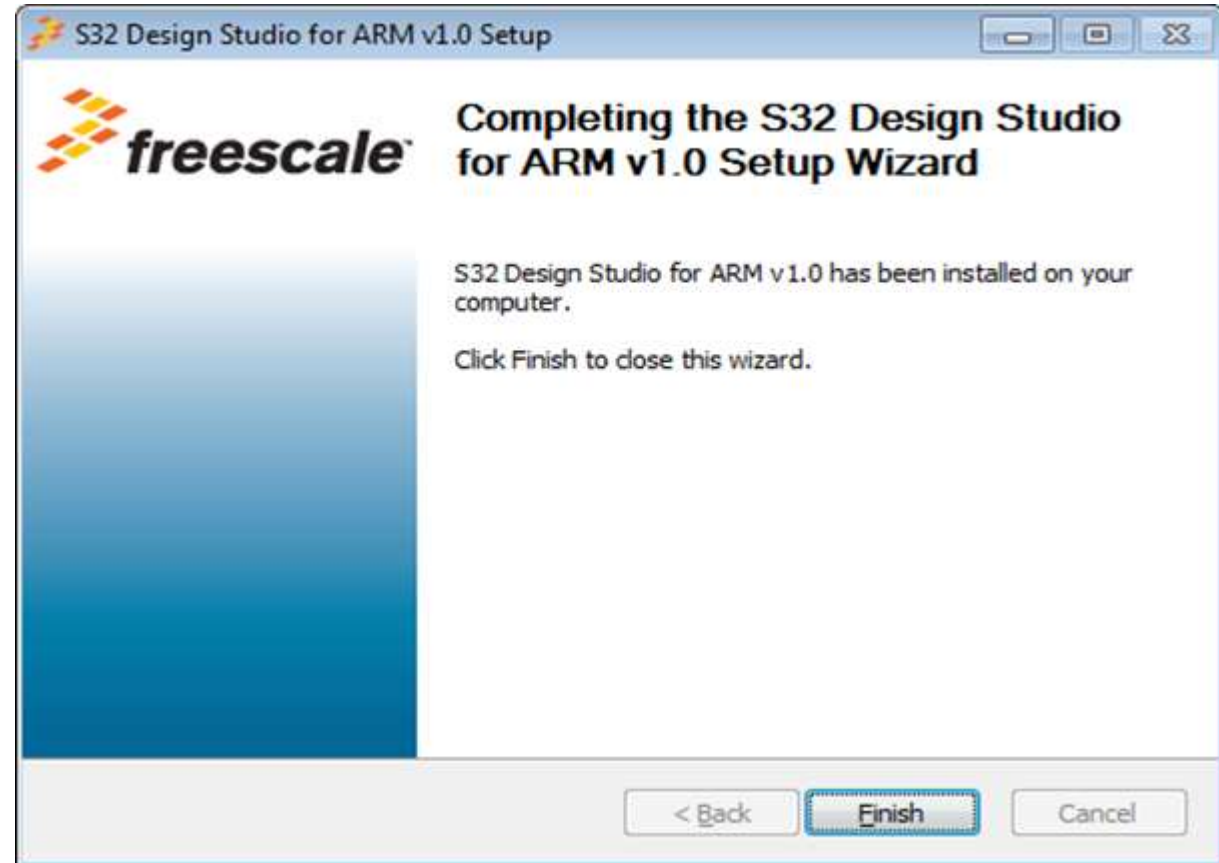
Download and Install S32 Design Studio Software

- Click **Install** to start installation.



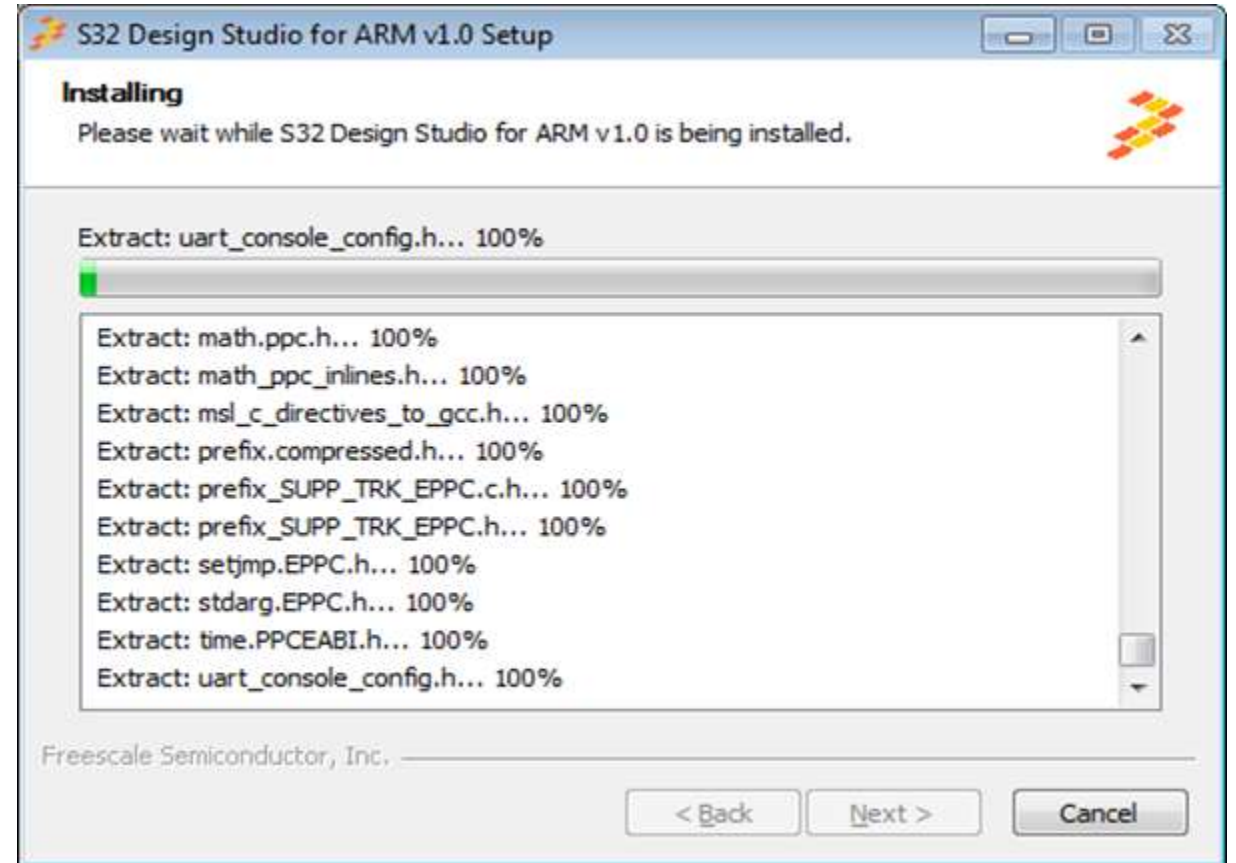
Download and Install S32 Design Studio Software

- Click Finish to complete S32DS for ARM® installation.



Download and Install S32 Design Studio Software

- Wait while the installation proceeds.



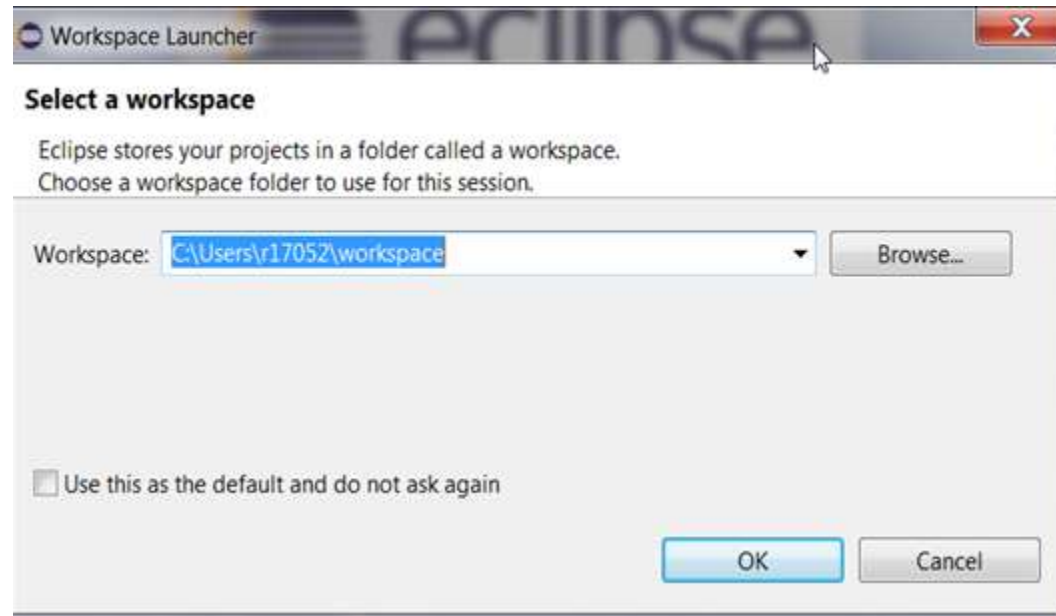


Create a New Project in S32 Design Studio



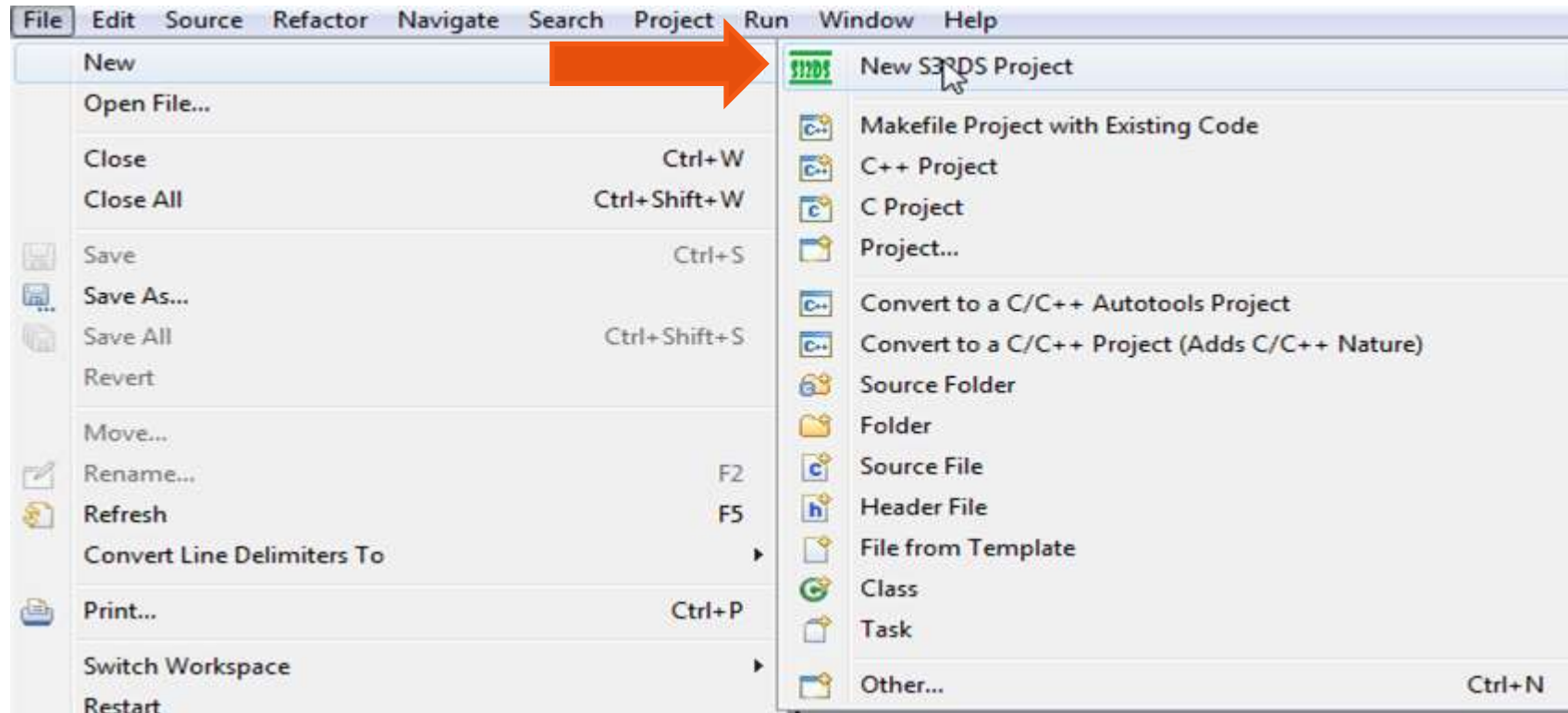
Create New Project: First Time – Select a Workspace

- Start program: Click on “S32 Design Studio for ARM v1.0” 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 Project



Create New Project: S32DS Project

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



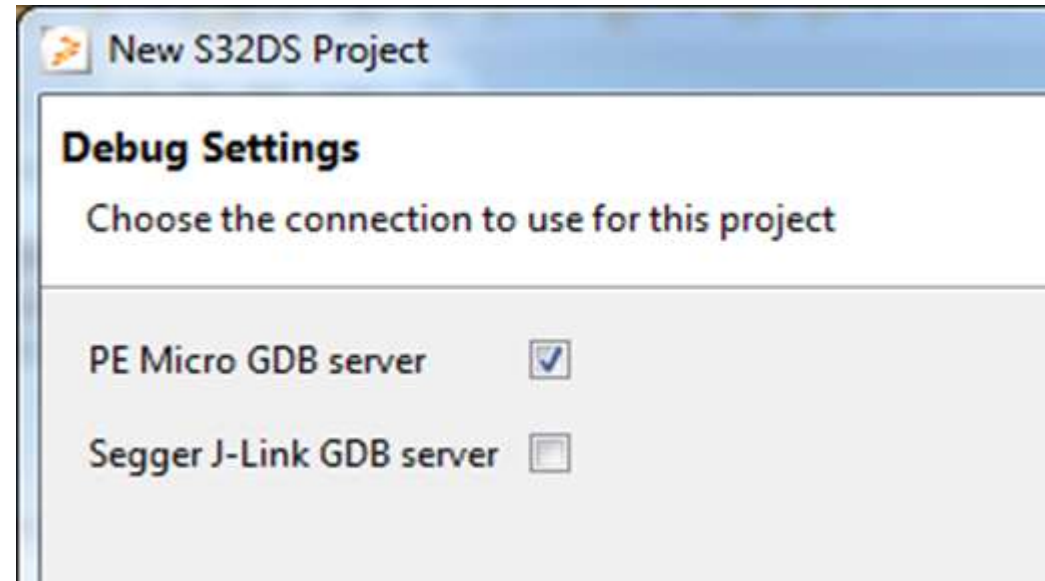
Create New Project: Target Processor

- KEA Families: select micro
- Float ABI: default or none
- I/O Support (for any console msgs)
 - UART
 - Debugger console
 - No I/O (None)
- Library Support (ARM libraries)
 - newlib
 - newlib_nano (suggested starting point)
 - ewl
- Language
- Click “Next”



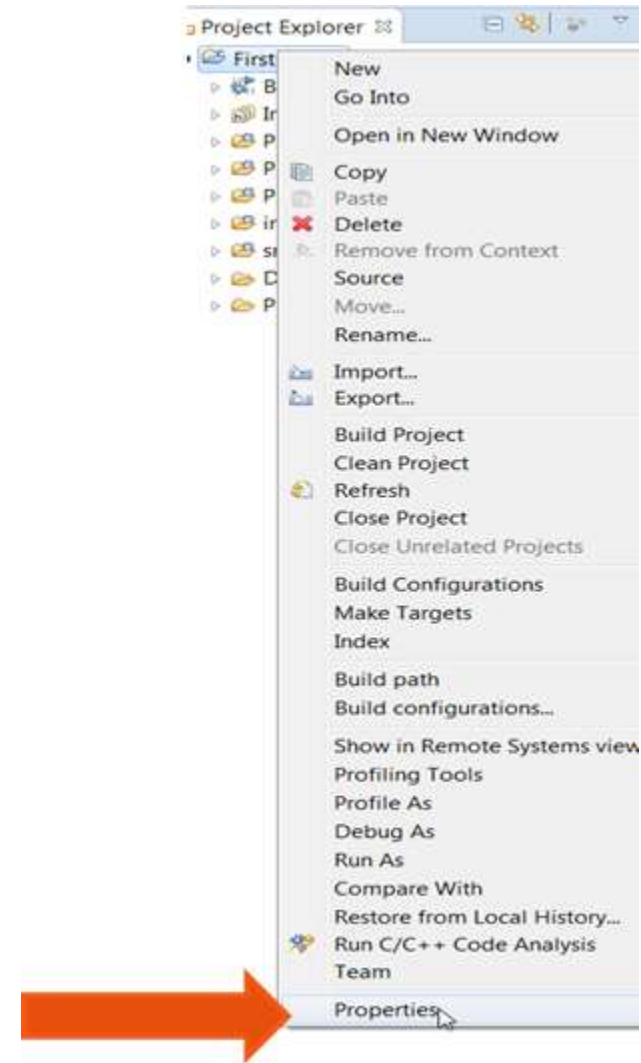
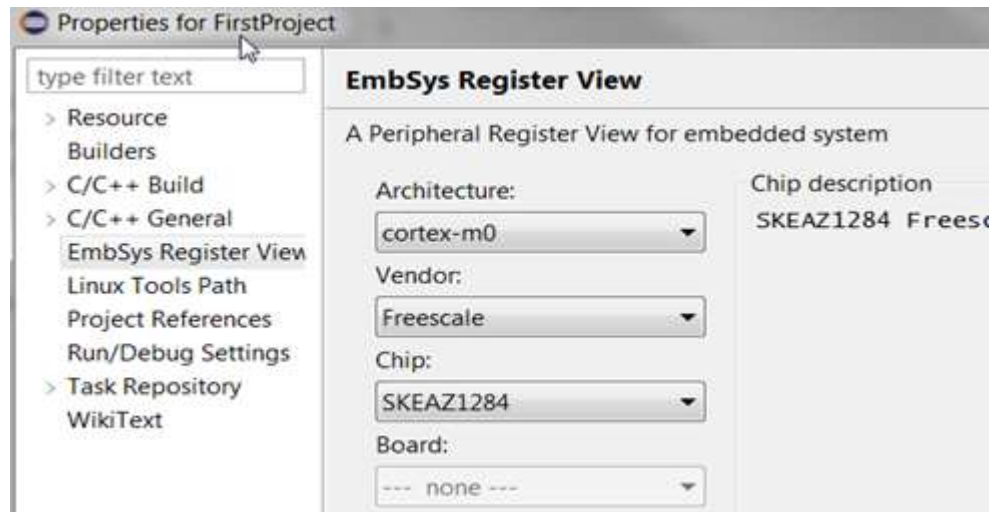
Create New Project: Target Debug

- Run Control Support:
 - PE Micro GDB server
 - Click “Finish”



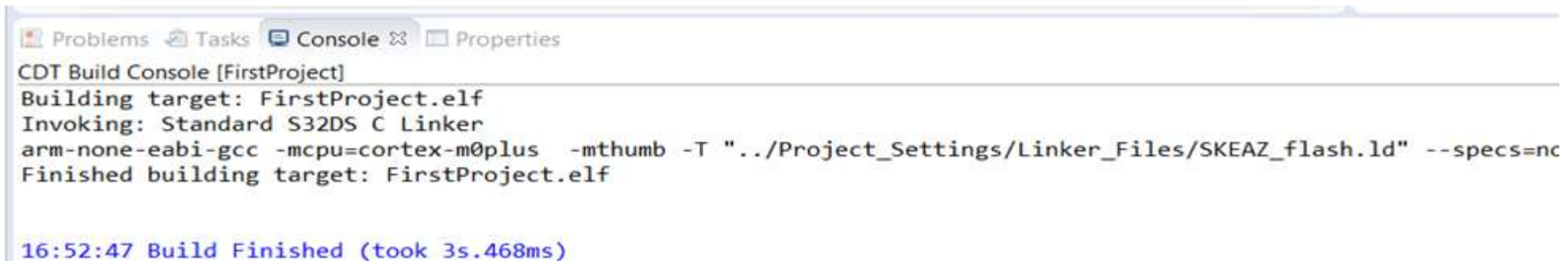
Create New Project: Enabling Debugger Viewing I/O

- (Temporary step for early release of software)
- Right click on project
- Select “Properties” at bottom
- Select “EmbSys Register View”,
make selections as per below & click “OK”



Create New Project: Build Project

- Select project
- If code was altered, save before building
- Project – Build Project
- Console tab has message that build finished



The screenshot shows the CDT Build Console interface with tabs for Problems, Tasks, Console, and Properties. The Console tab is active, displaying the following text:

```
CDT Build Console [FirstProject]
Building target: FirstProject.elf
Invoking: Standard S32DS C Linker
arm-none-eabi-gcc -mcpu=cortex-m0plus -mthumb -T "../Project_Settings/Linker_Files/SKEAZ_flash.ld" --specs=nc
Finished building target: FirstProject.elf

16:52:47 Build Finished (took 3s.468ms)
```



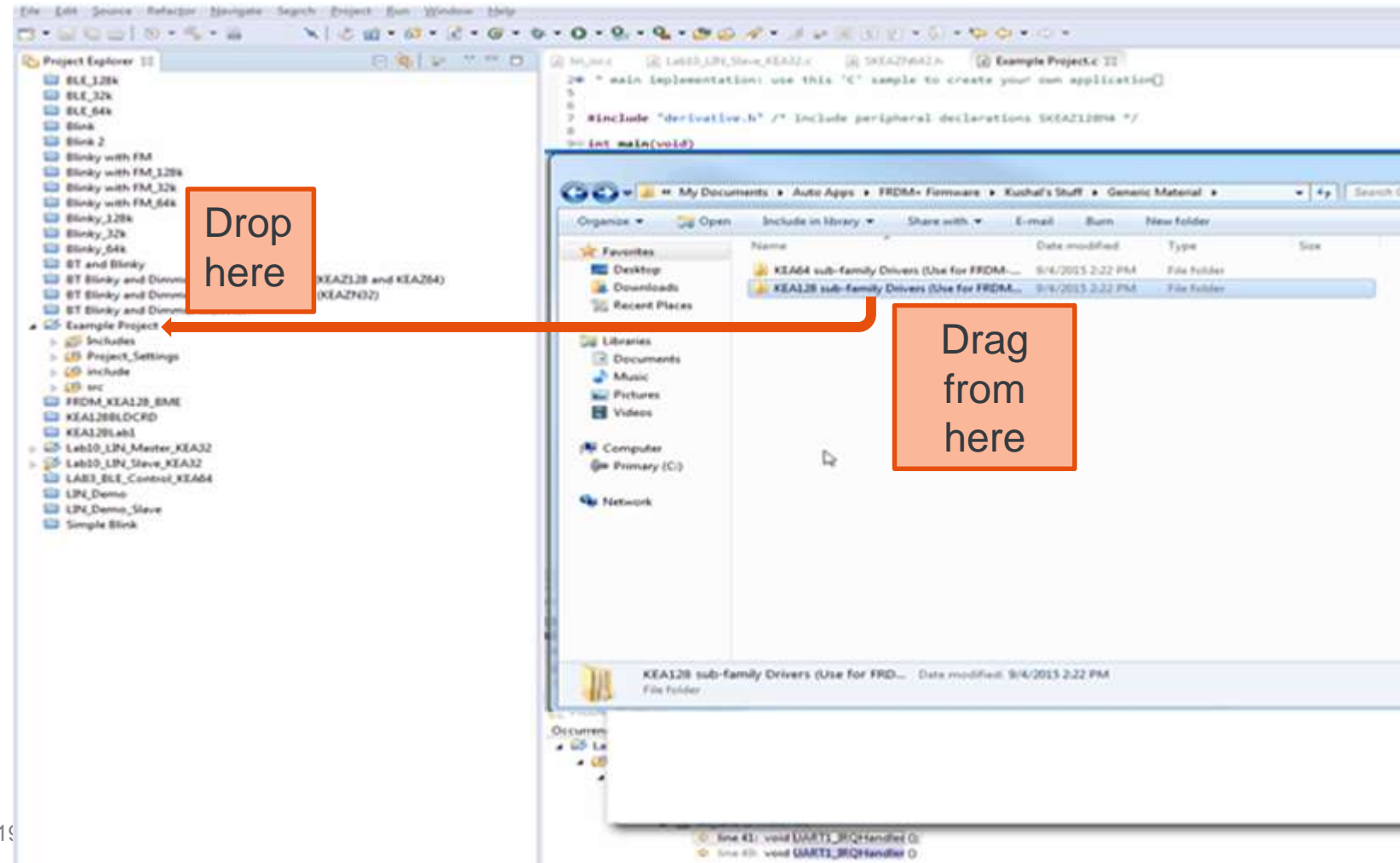
Import SDK Evaluation Drivers into Project



Copy Drivers into Project Directory

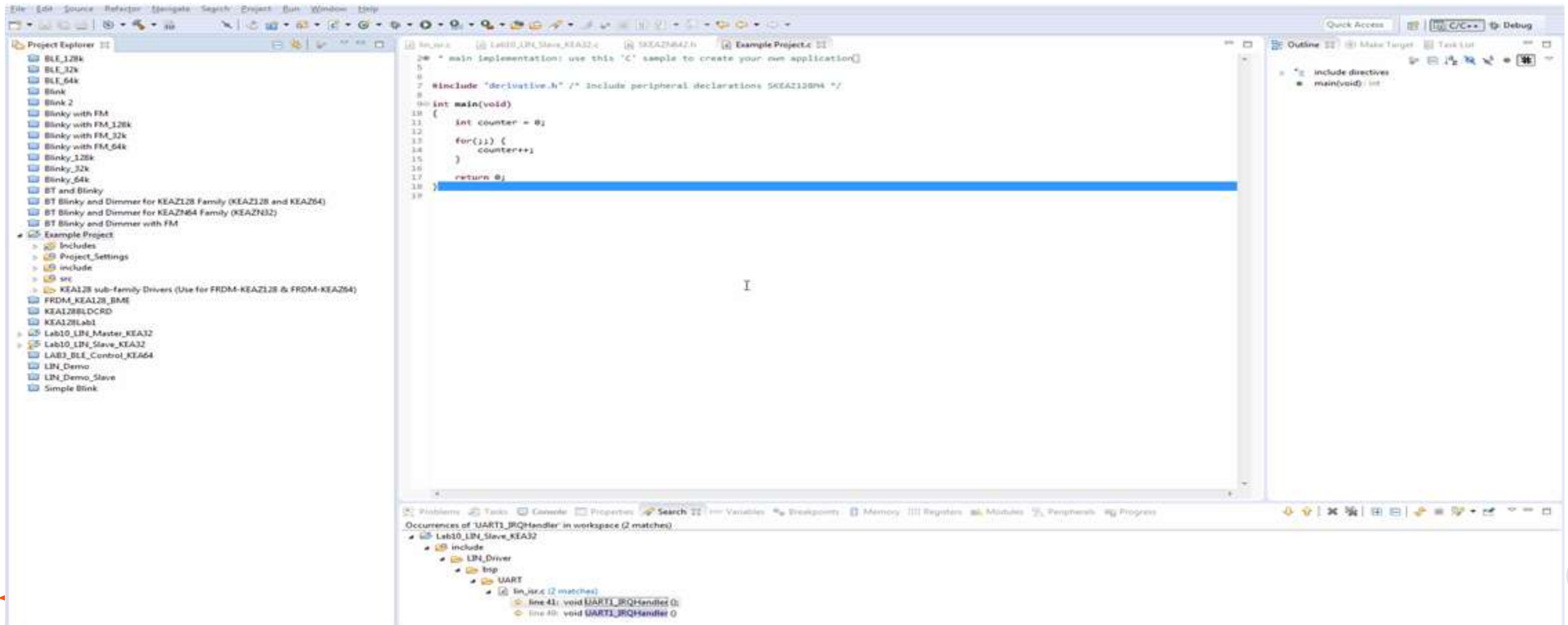
NOTE: The following steps to import SDK drivers into a S32DS for ARM project are only needed until the next release of S32DS in Q1'16 when they will be integrated into tool

- Drag and drop the SDK drivers into your project workspace in S32DS
 - To get SDK driver package go to www.freescale.com/kea and download the Quick Start Package (QSP) under “Getting Started” section
- You can choose to copy or link the driver folders in S32DS
 - In this example, folders are copied



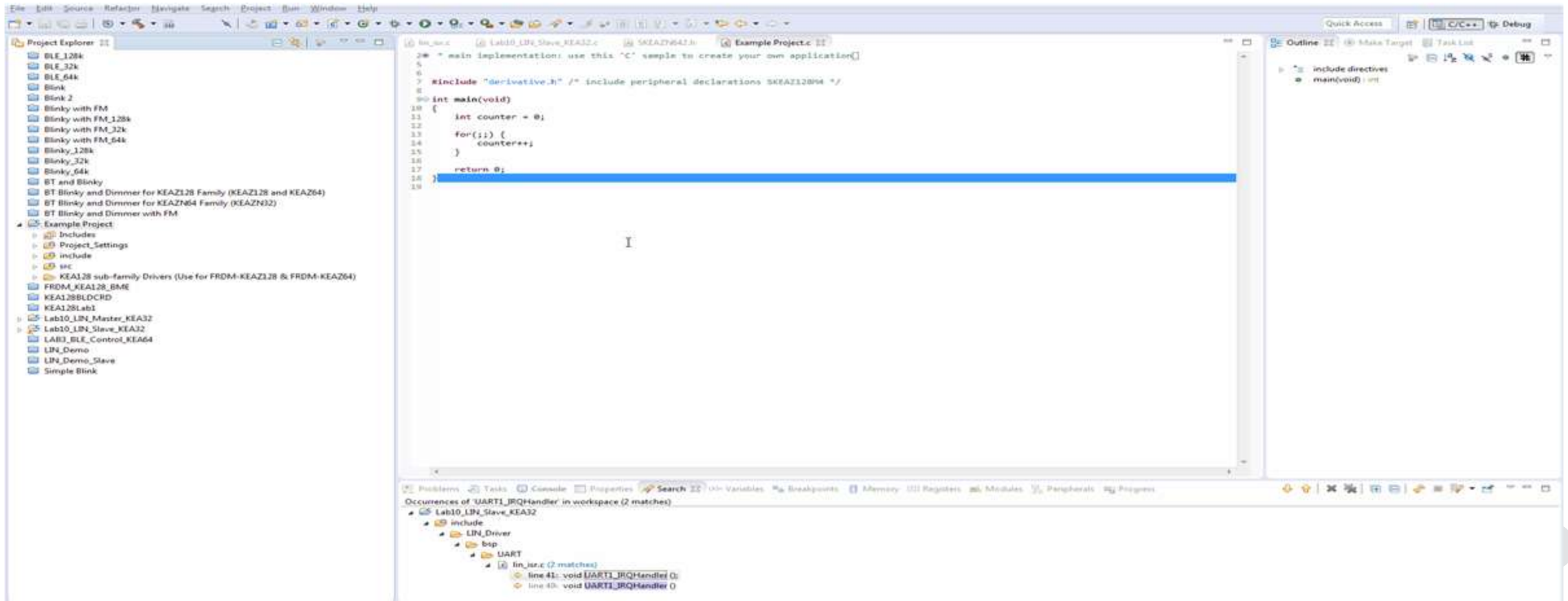
Include Driver Folder Paths

- Right-click on project>*Properties*>*C/C++ Build*>*Settings*>*Standard S32DS C Compiler*>*Includes*>*Include paths (-I)*
- Add the driver folder paths by clicking on the icon with the green '+' and specifying to folder directory. Include *all* folders and subfolders



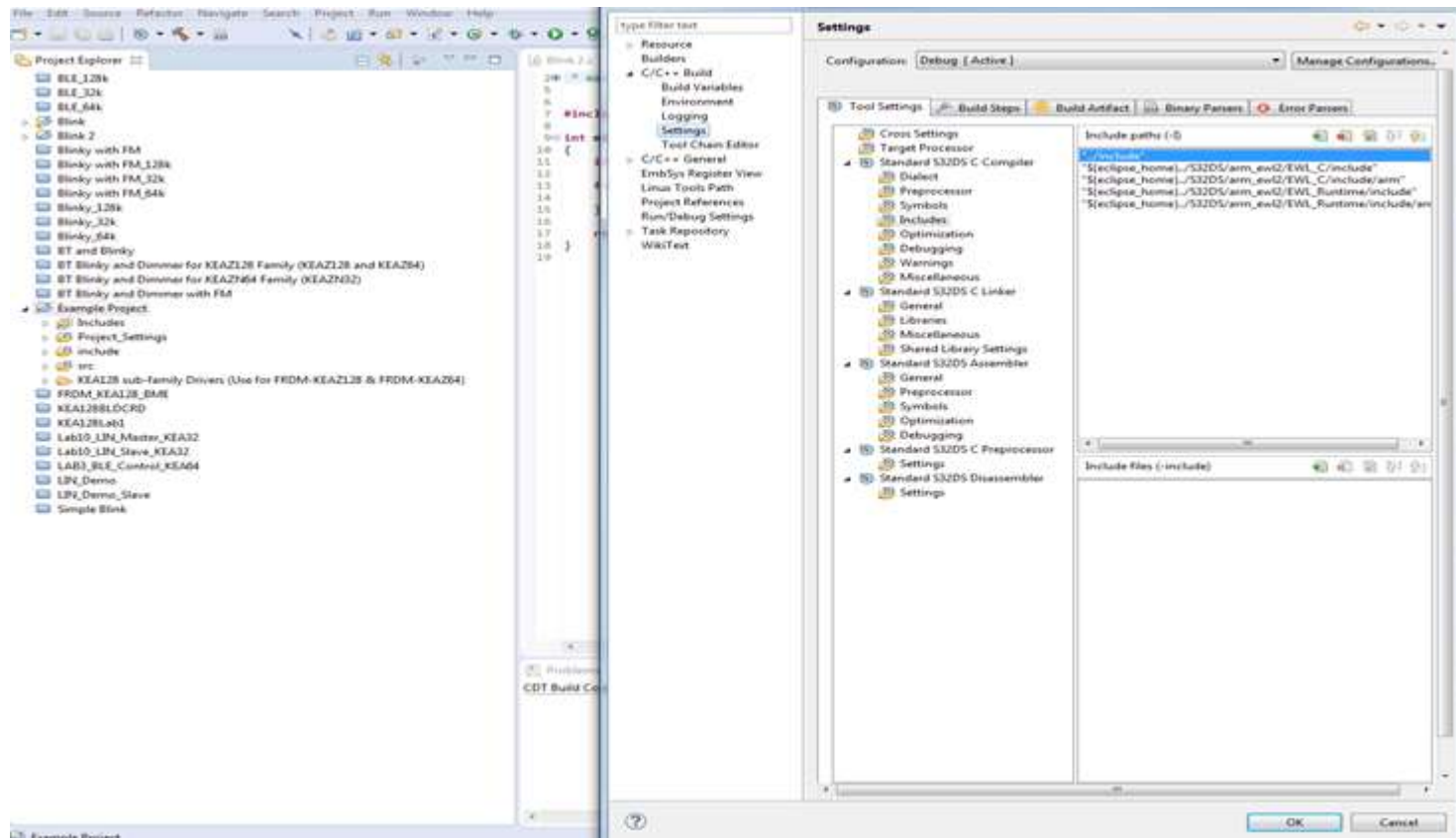
Add the Drivers to Build Configurations

- Right-click on project>*Build Configurations*
- Click the top level driver folder to include all drivers into the build for Debug, Release, and Debug_RAM modes. Make sure they have green '+' instead of grey 'x' next to them
- Otherwise, compiling will not include the driver files



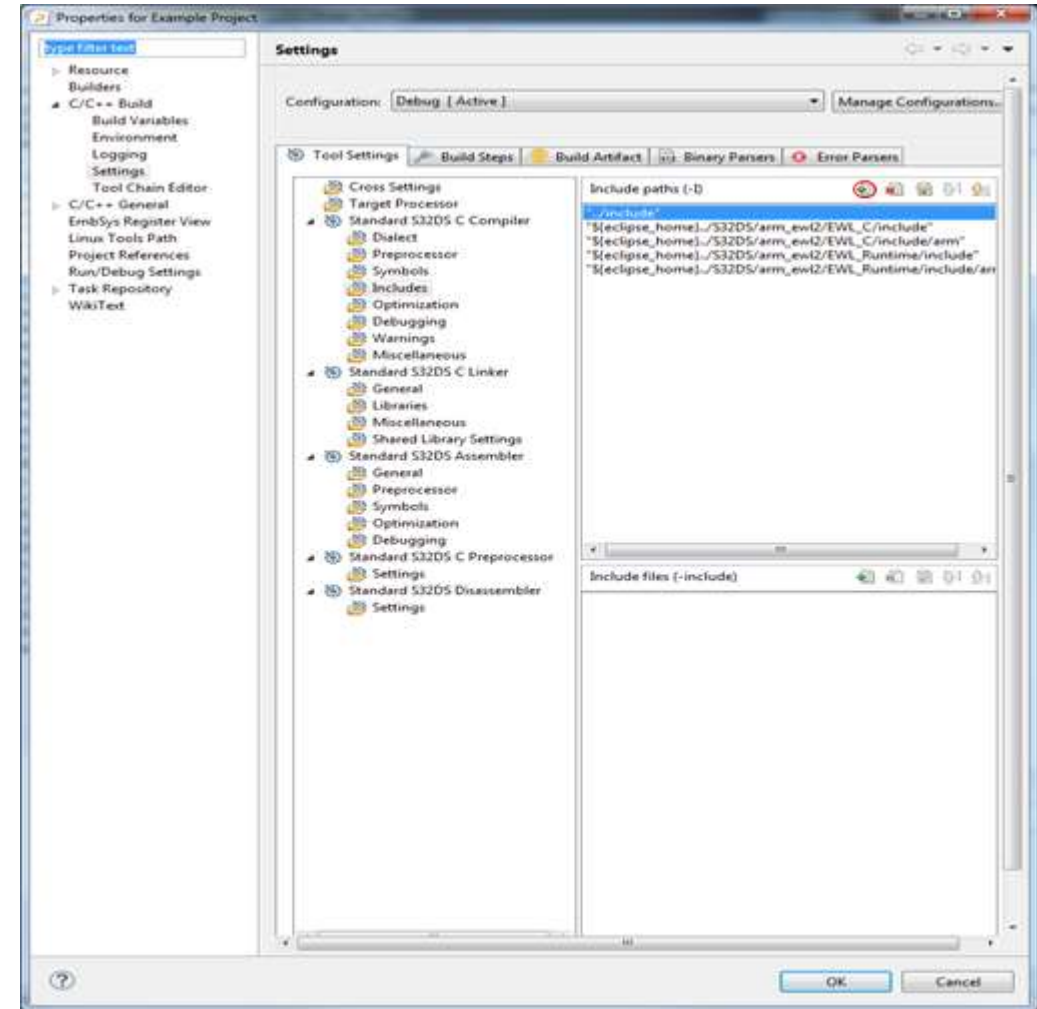
Include Driver Folder Paths

- Right-click on the project
- Go to *Properties>C/C++ Build>Settings>Standard S32DS C Compiler>Includes*



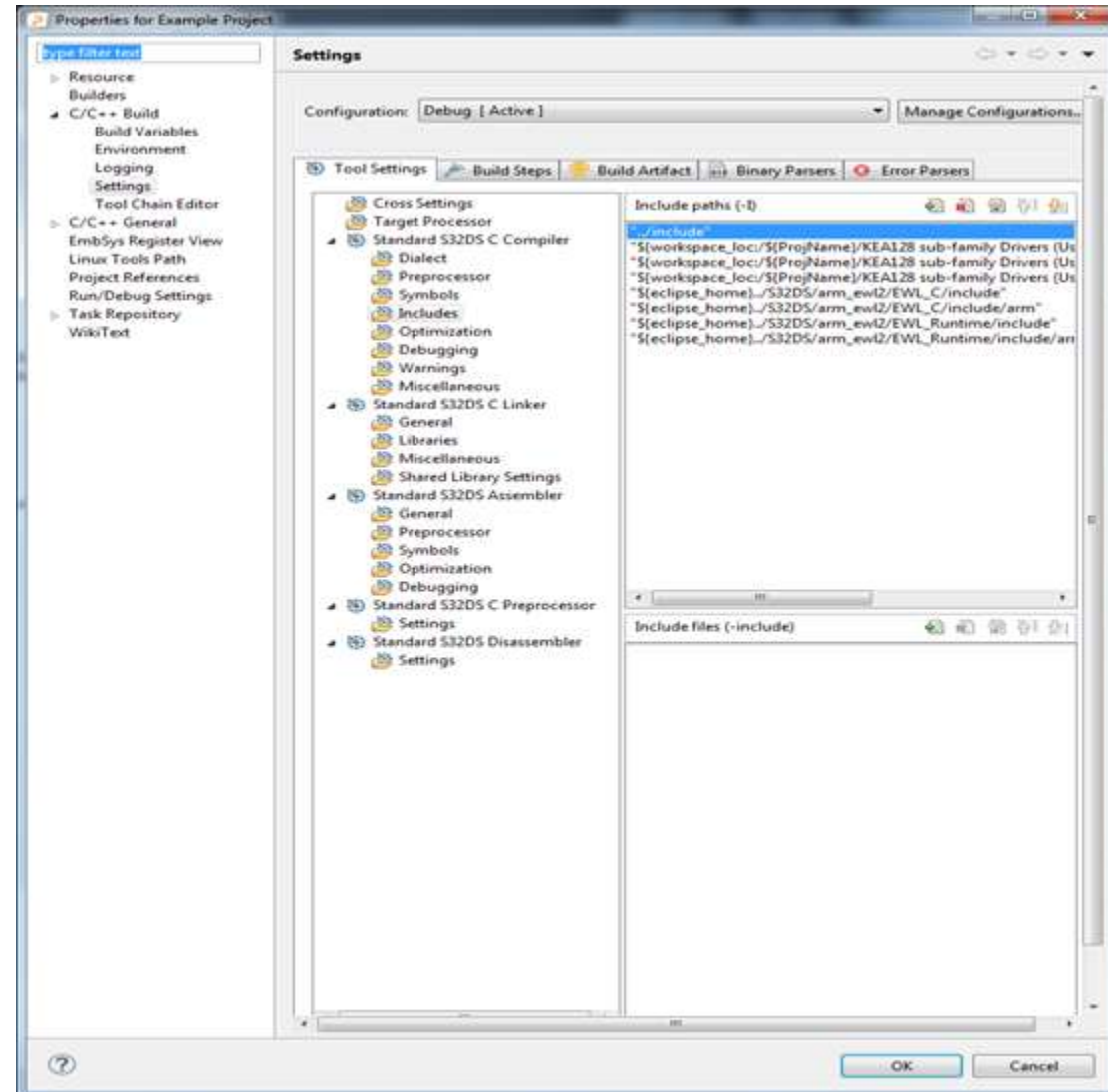
Include Driver Folder Paths

- Click on the '+' icon to include the SDK folders
- This lets the tool know where to look for the folders



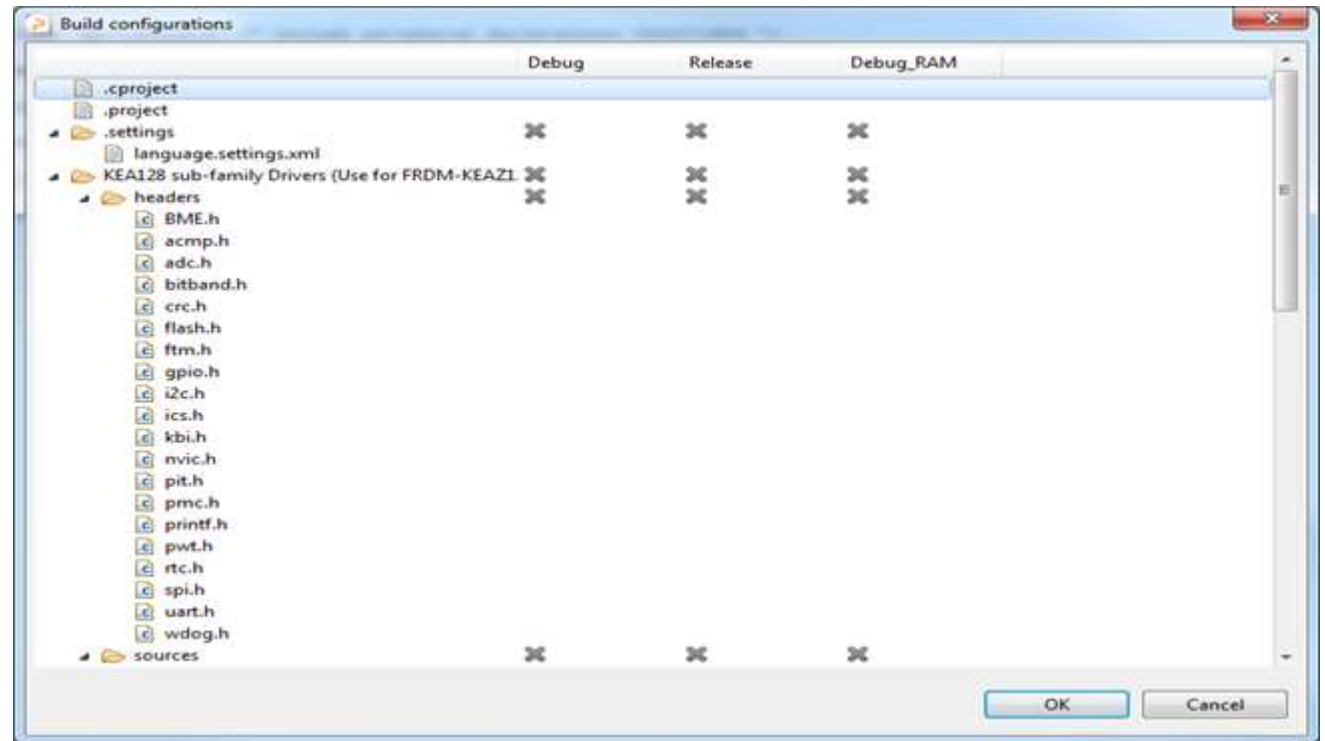
Include Driver Folder Paths

- Add all SDK folders and subfolders
 - *SDK*
 - *SDK/headers*
 - *SDK/sources*
- Hit OK



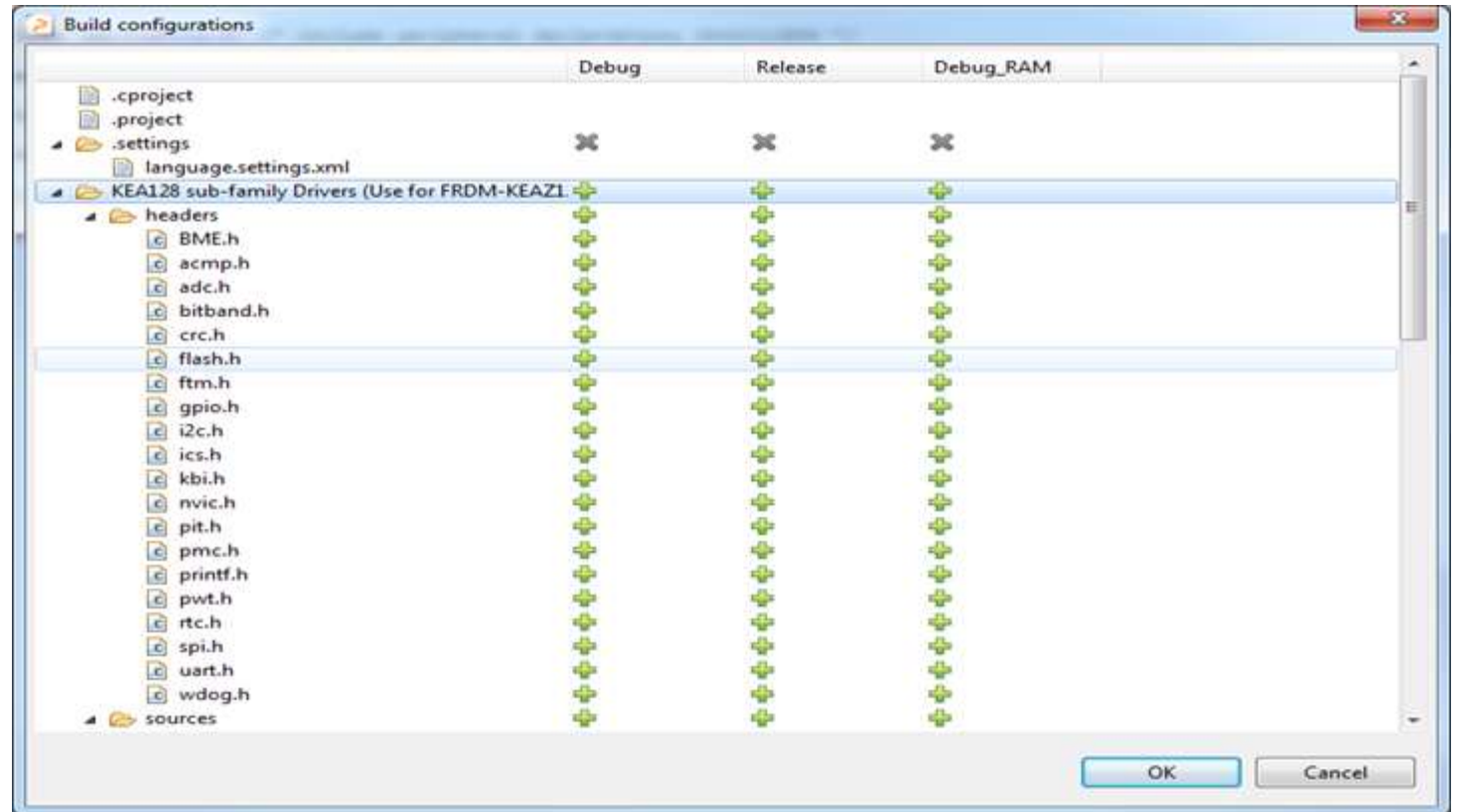
Add SDK to Build Configurations

- Enable the SDK folders in the build configurations
- Right-click on the project, go to *Build configurations*
- This lets S32DS know to incorporate the SDK into the build, having told it where to look.



Add SDK to Build Configurations

- The grey 'X' icon means the folder is not included in the build
- Click on the 'X' to turn them into green '+'
- Hit *OK*



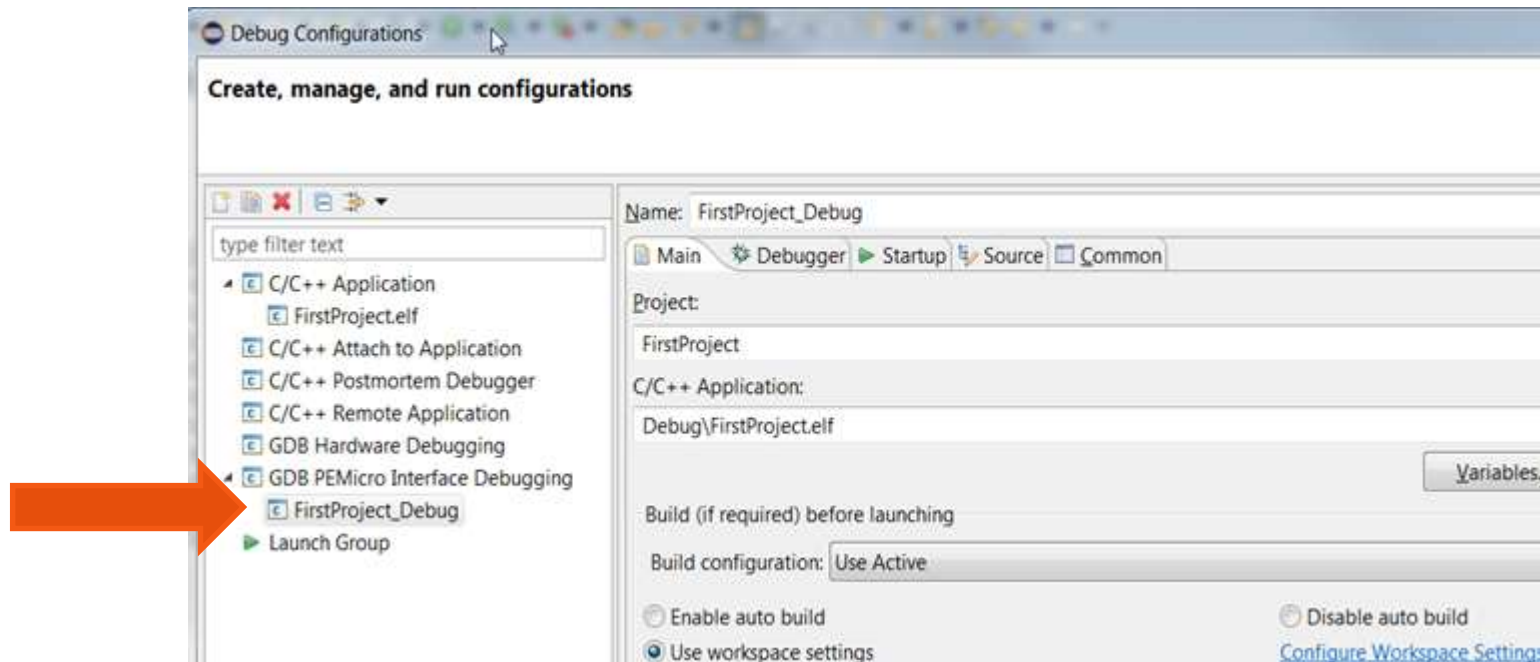


Basic Configuration



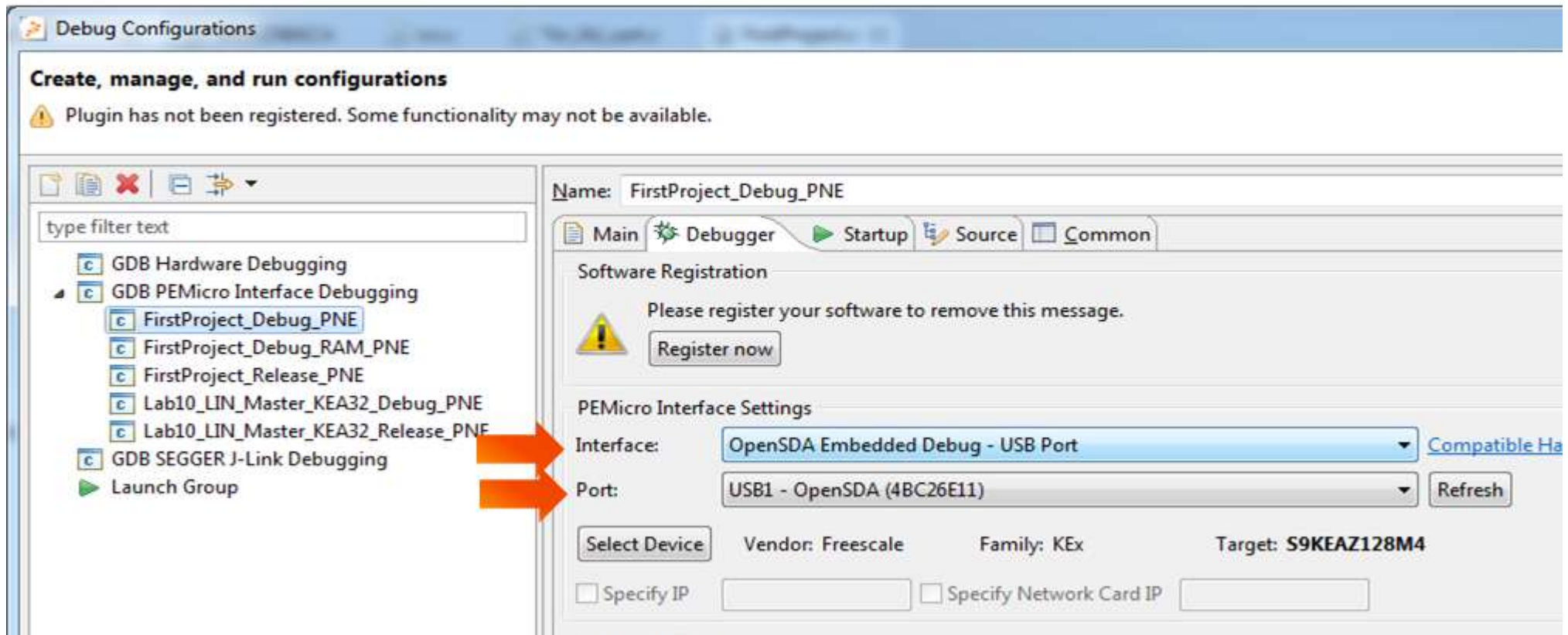
Debug Configuration: Select Interface & Project

- Connect target to PC
- Run – Debug Configurations
- Example: FirstProject_Debug - GDP PEMicro Interface Debugging
- Main tab displays selections



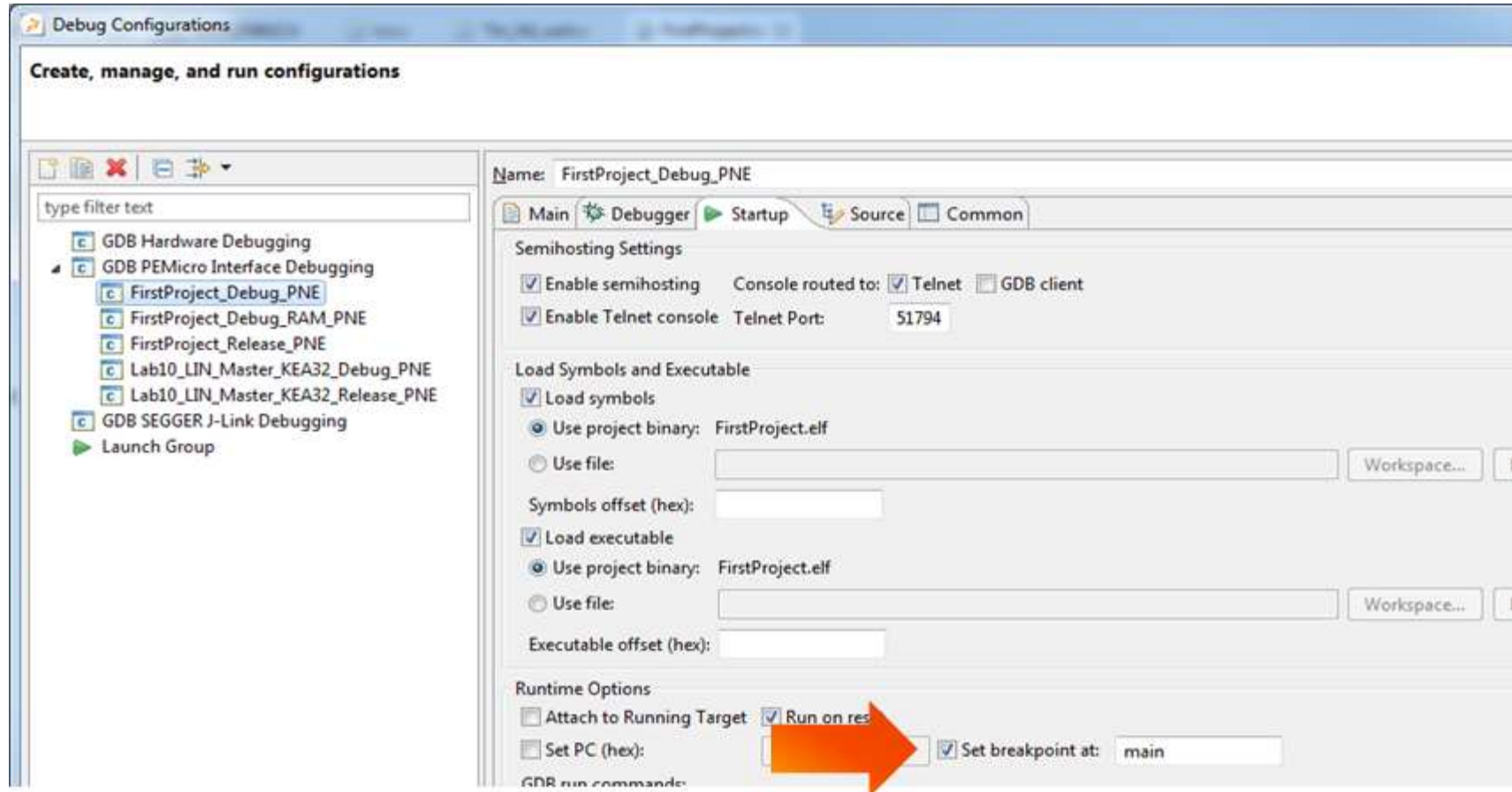
Debug Configuration: Verify Debugger Settings

- Click on Debugger tab
- Verify Interface is correct or make proper selection
- Verify Port is found. If the target is disconnected from USB port, connect target and hit refresh



Debug Configuration: First Breakpoint

- Click on Startup tab. Modify breakpoint if desired.






Basic Configuration

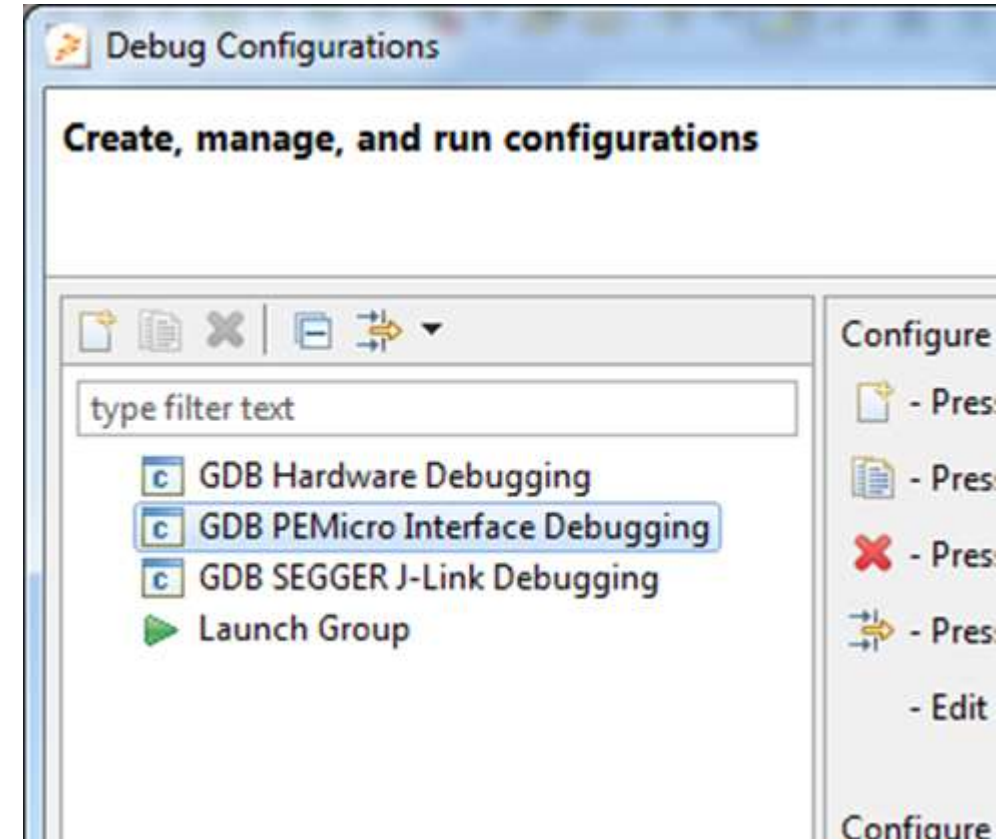
Creating Your Own Debug Configuration

Sometimes after you build a project, the debug configuration for that project does not generate automatically. You will have to create one from scratch



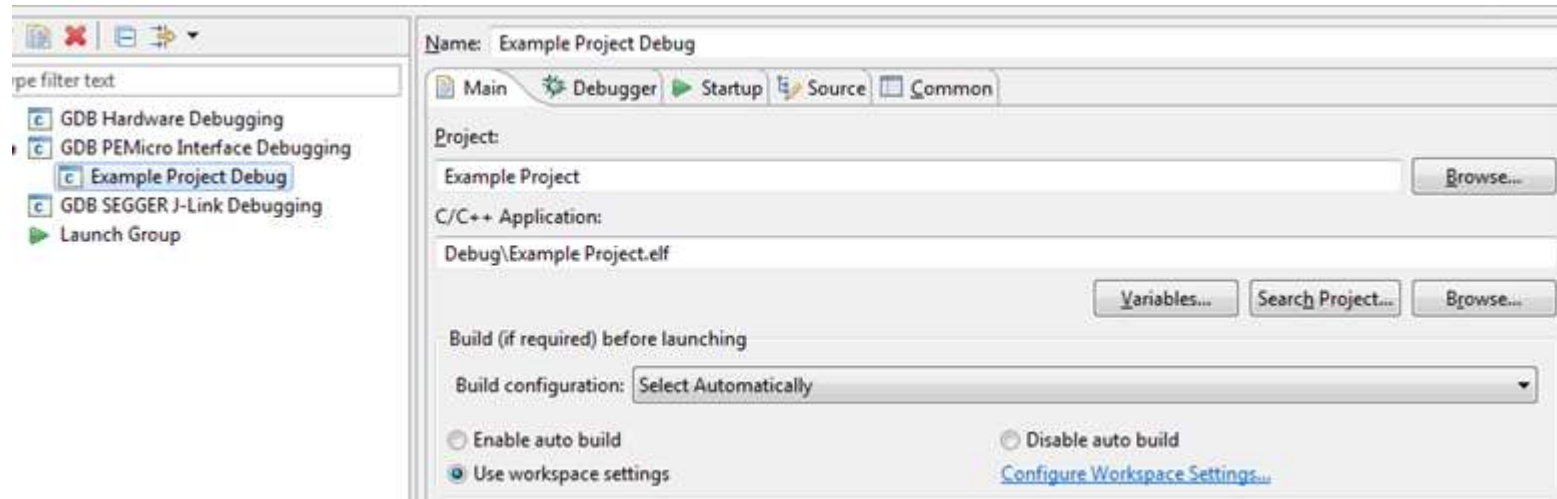
Go to the Debug Configurations

- In this example, a blank project for KEAZ64 called Example Project was created and compiled under Debug mode
 - You can  ct which mode to build under by selecting from the drop-down menu next to the hammer icon
 - Building in Debug mode will generate a Debug folder in the project workspace; Release mode would generate a Release folder
- Click on the drop-down menu next to the bug icon, select Debug Configurations



Create New Configuration

- The KEA connects via P&Emicro
- Right-click on GDB PEMicro Interface Debugging and select New
- A blank debug config will generate
 - Default name and settings of new config will depend on which project is selected in your project workspace at the time the debug config window is opened
- Because Example Project was selected, config called Example Project Debug and Example Project and its .elf file are selected



Select Correct Interface

- Go to the Debugger tab of the new config
- USB Multilink is selected by default under Interface
- KEA talks through OpenSDA, select OpenSDA for the interface
- *Port* field will populate automatically when you select OpenSDA if board is recognized. If still blank, unplug/replug your board

The image shows two screenshots of the 'PEmicro Interface Settings' dialog box, connected by a large yellow arrow pointing downwards. The top screenshot shows the 'Interface' dropdown set to 'USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB Port' and the 'Port' dropdown is empty. The bottom screenshot shows the 'Interface' dropdown set to 'OpenSDA Embedded Debug - USB Port' and the 'Port' dropdown is populated with 'USB1 - OpenSDA (4BC76E6B)'. Both screenshots include a 'Select Device' button with a red error icon, a 'Refresh' button, and a 'Compatible Hardware' link. The 'Vendor:', 'Family:', and 'Target:' fields are also visible but empty in both.

Select Correct Device

- Press the Select Device button and find your device from the available list
- Your device will be printed on the top of the chip

PEMicro Interface Settings

Interface: OpenSDA Embedded Debug - USB Port ▼ [Compatible Hardware](#)

Port: USB1 - OpenSDA (4BC76E6B) ▼ Refresh

Select Device Vendor: Freescale Family: KEx Target: S9KEAZ64M4

Check Debugger Pointer

- In Eclipse, you have to specify where the debugger executable resides
- Pointer is also in *Debugger* tab, at GDB Client Settings>Executable
- When you create a config from scratch, S32DS will sometimes generate an incorrect pointer or none at all. Make sure the relative path is as in the screenshot:
`${cross_prefix}gdb${cross_suffix}`
- You are now ready to debug under your new debug config

GDB Server Settings

☒ Launch Server Locally

Hostname or IP: Port Number:

Server Parameters:

GDB Client Settings

Executable:

Other options:

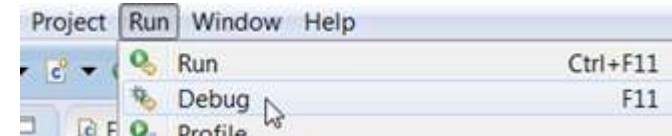


Debug Basics



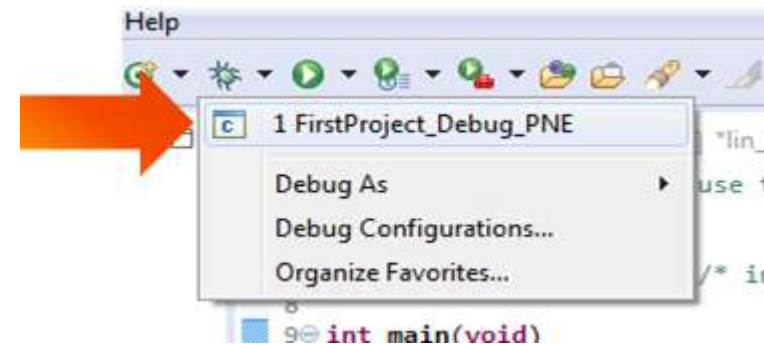
Debug Basics: Starting the Debugger

- 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)



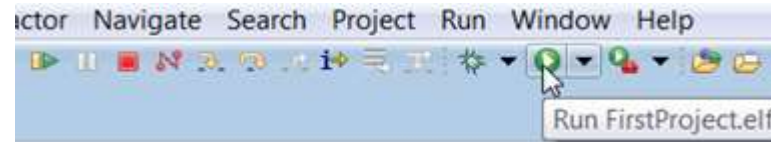
Note: This method currently selects the desktop target (*project.elf*) and gives an error. Do not use until this is changed.

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



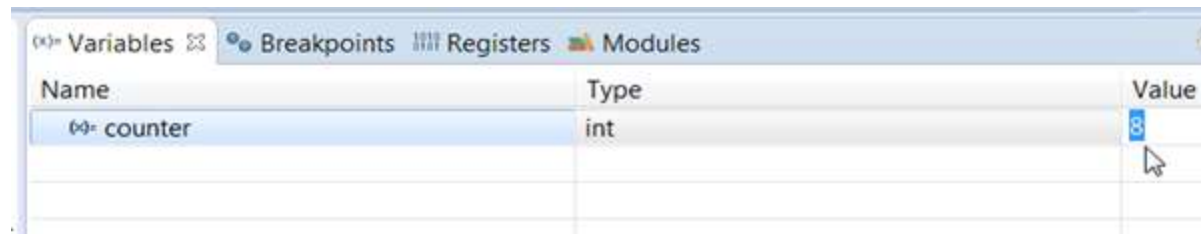
Debug Basics: Step, Run, Suspend, Resume

- 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

| Name | Value |
|-------------------|-----------|
| General Registers | |
| r0 | 3 |
| r1 | 5 |
| r2 | 536866944 |
| r3 | 8 |
| r4 | 0 |

- View peripheral registers in the EmbSys Registers tab

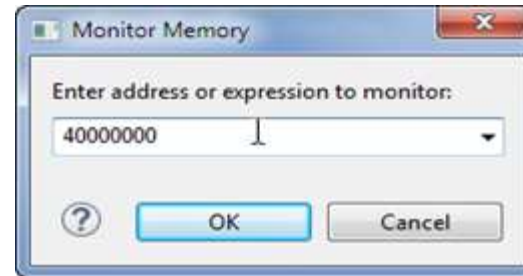
| Register | Hex | Bin | Reset | Access | Address | Description |
|------------------|------|----------|-------|--------|------------|--|
| IRQ | | | | | | Interrupt |
| SC | 0x00 | 00000000 | 0x00 | RW | 0x40031000 | Interrupt Pin Request Status and Control |
| IRQMOD (bit 0) | 0x0 | 0 | | | | 0: IRQ event is detected only on 1 |
| IRQIE (bit 1) | 0x0 | 0 | | | | 0: Interrupt request when IRQF sc |
| 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 IRQI |
| RESERVED (bit 7) | 0x0 | 0 | | | | no description available |

Debug Basics: View & Alter Memory

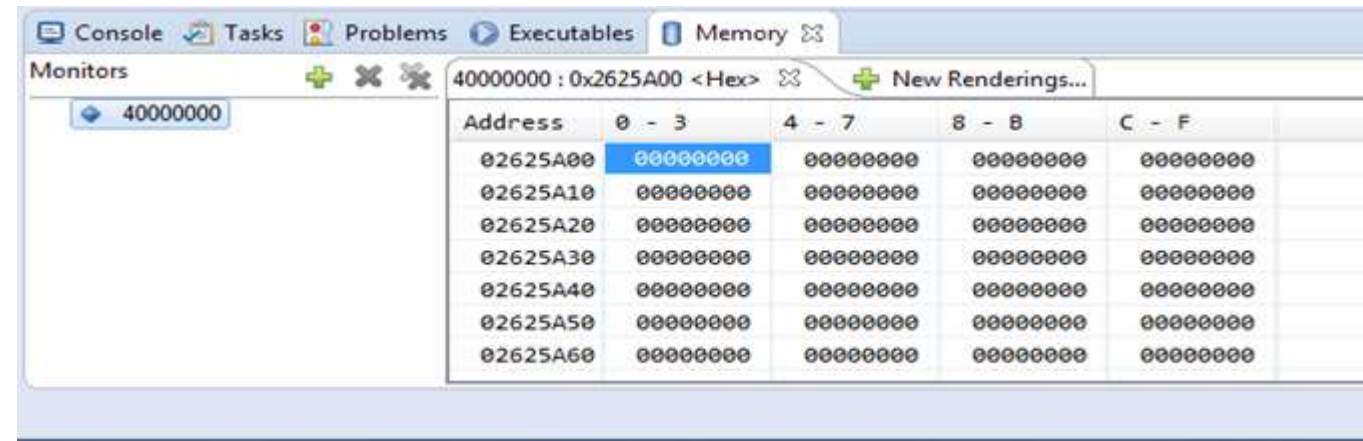
- Add Memory Monitor



- Select Base Address to Start at : 40000000



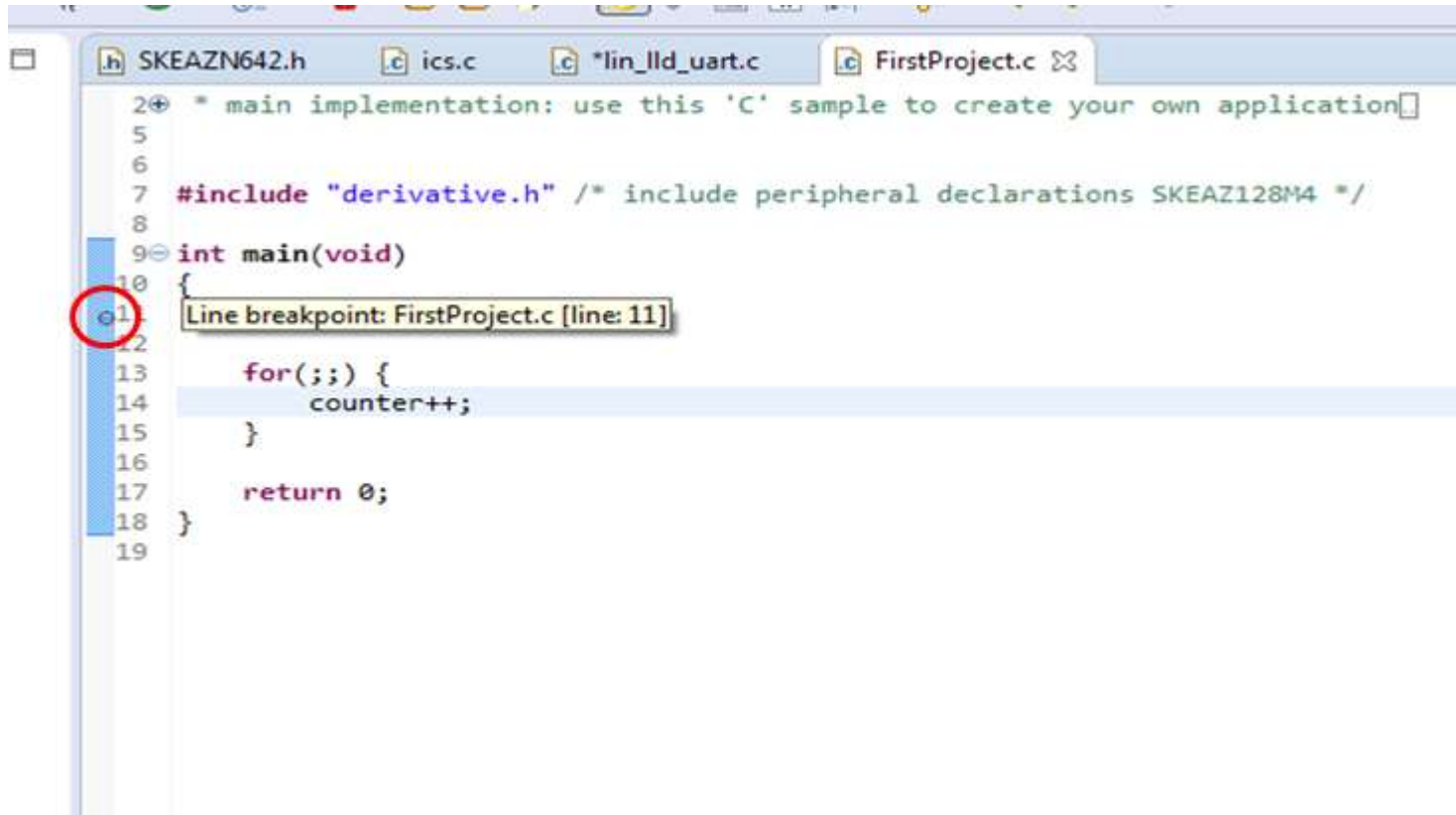
- View Memory



Debug Basics: Breakpoints

Add Breakpoint: Point and Click

- light blue dot represents debugger breakpoint



The screenshot shows a code editor with several tabs: SKEAZN642.h, ics.c, *lin_lld_uart.c, and FirstProject.c. The code in FirstProject.c is as follows:

```
2+ * main implementation: use this 'C' sample to create your own application
5
6
7 #include "derivative.h" /* include peripheral declarations SKEAZ128M4 */
8
9 int main(void)
10 {
11     for(;;) {
12         counter++;
13     }
14     return 0;
15 }
```

A light blue dot, representing a debugger breakpoint, is placed on line 11. A tooltip box appears over the dot, displaying the text: "Line breakpoint: FirstProject.c [line: 11]".

Debug Basics: Reset & Terminate Debug Session

- Reset program counter
- Terminate Ctrl+F2()





Startup Code Reset



Startup Code: Reset*

- System reset begins with:
 - on-chip regulator in full regulation
 - System clocking generation from an internal reference
- On reset exit the following is preformed:
 - Note: Vector-table is located at 0x0000_0000 on KEA128
 - Reads start SP (SP_main) from vector-table offset 0
 - Reads start program counter (PC) from vector-table offset 4
 - Link Register (LR) is set to 0xFFFF_FFFF
- On chip peripherals are disabled
- Non-analog I/O pins configured as disabled except:
 - SWD_DIO/SWD/DCLK
 - NMI
 - RESET
- Analog pins are set to their default analog function

* KEA128 Sub-Family Reference Manual, Rev. 2, July 2014 chapter 6



Install FreeMaster – Serial Real-Time Monitor Tool

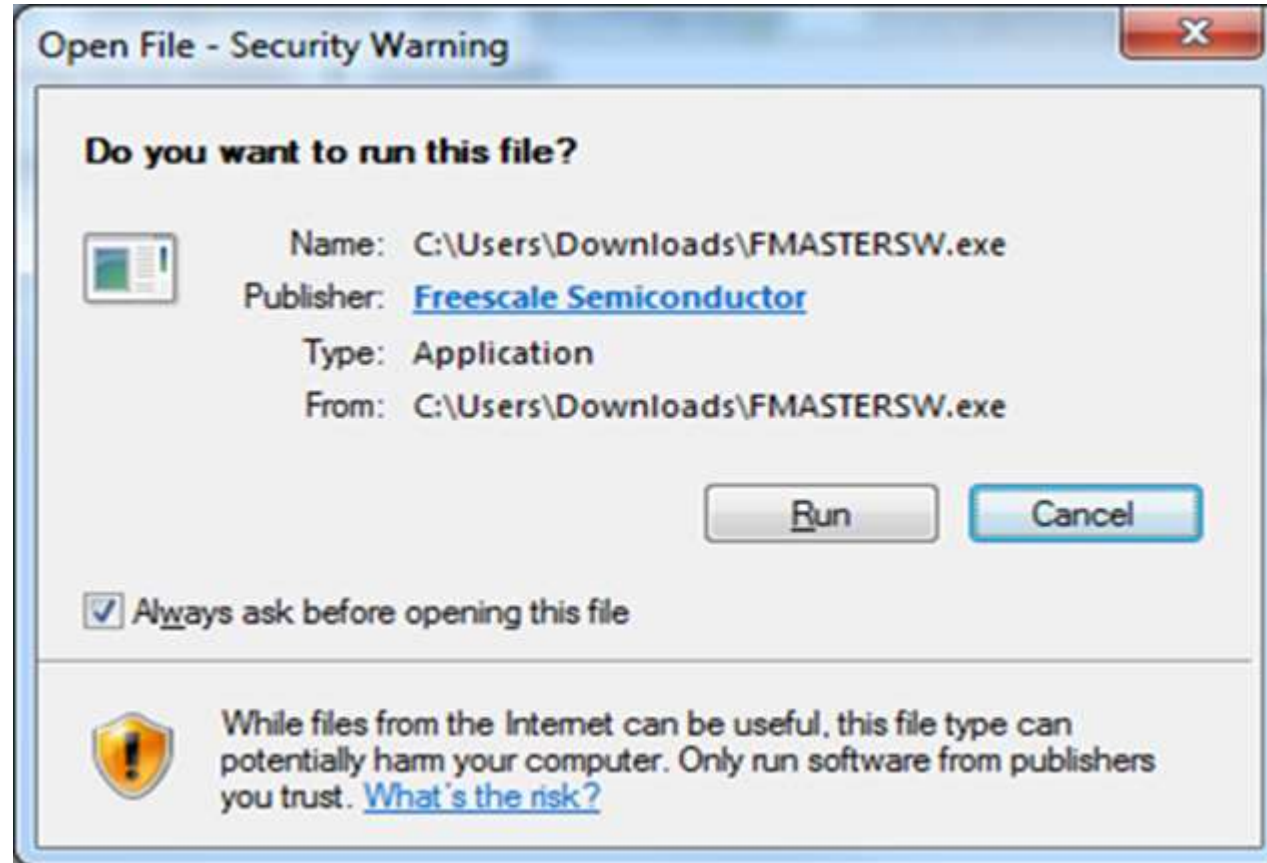


Download FreeMASTER for Real-Time Monitoring

- Go to www.freescale.com/freemaster
- Download both *FreeMASTER* (Rev 2.0 or latest) and *FreeMASTER Communication Driver* (Rev 1.9 or latest)

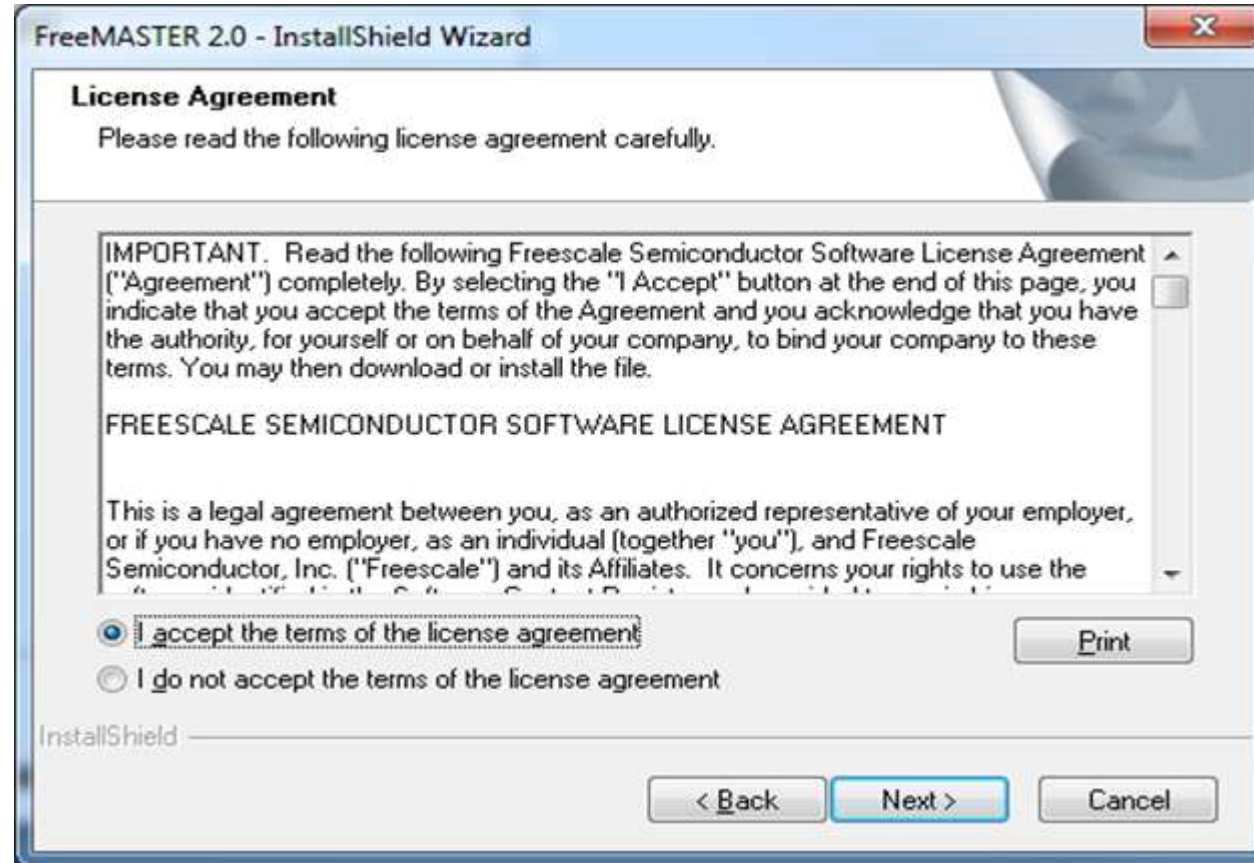
Downloading FreeMASTER for Real-Time Monitoring

- Double-click on *FMASW.exe*
- Trust Freescale ☺
- Hit *Run*



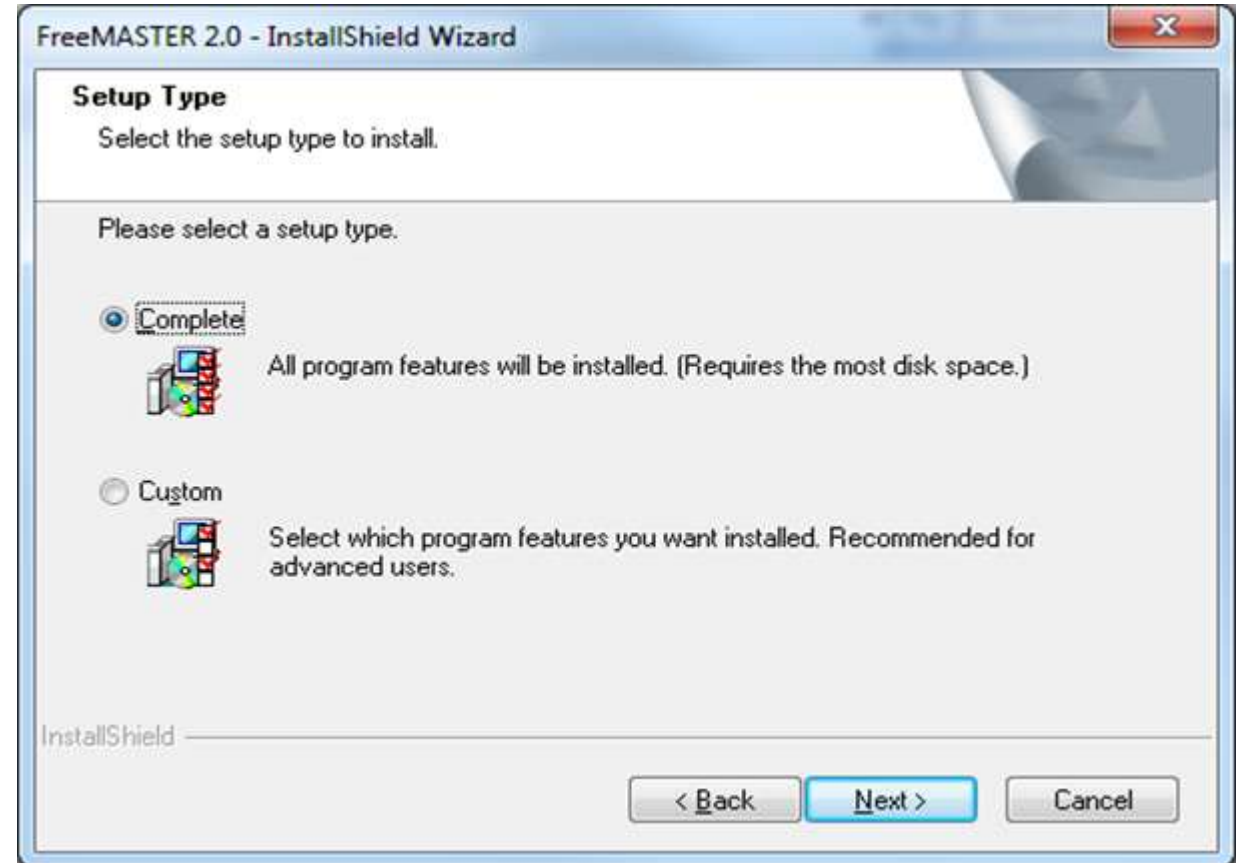
Downloading FreeMASTER for Real-Time Monitoring

- Accept the Freescale license agreement
- Hit *Next*



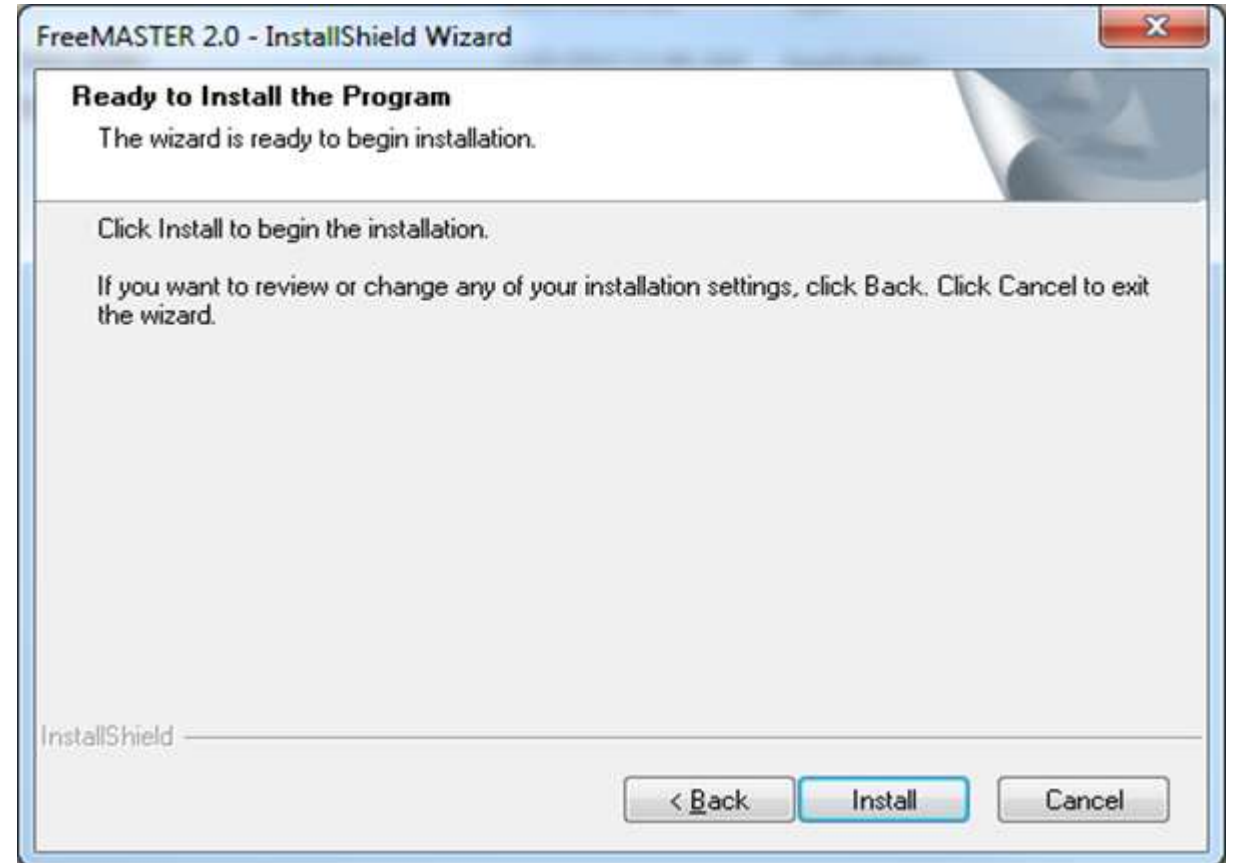
Downloading FreeMASTER for Real-Time Monitoring

- Select the complete install
- Hit *Next*



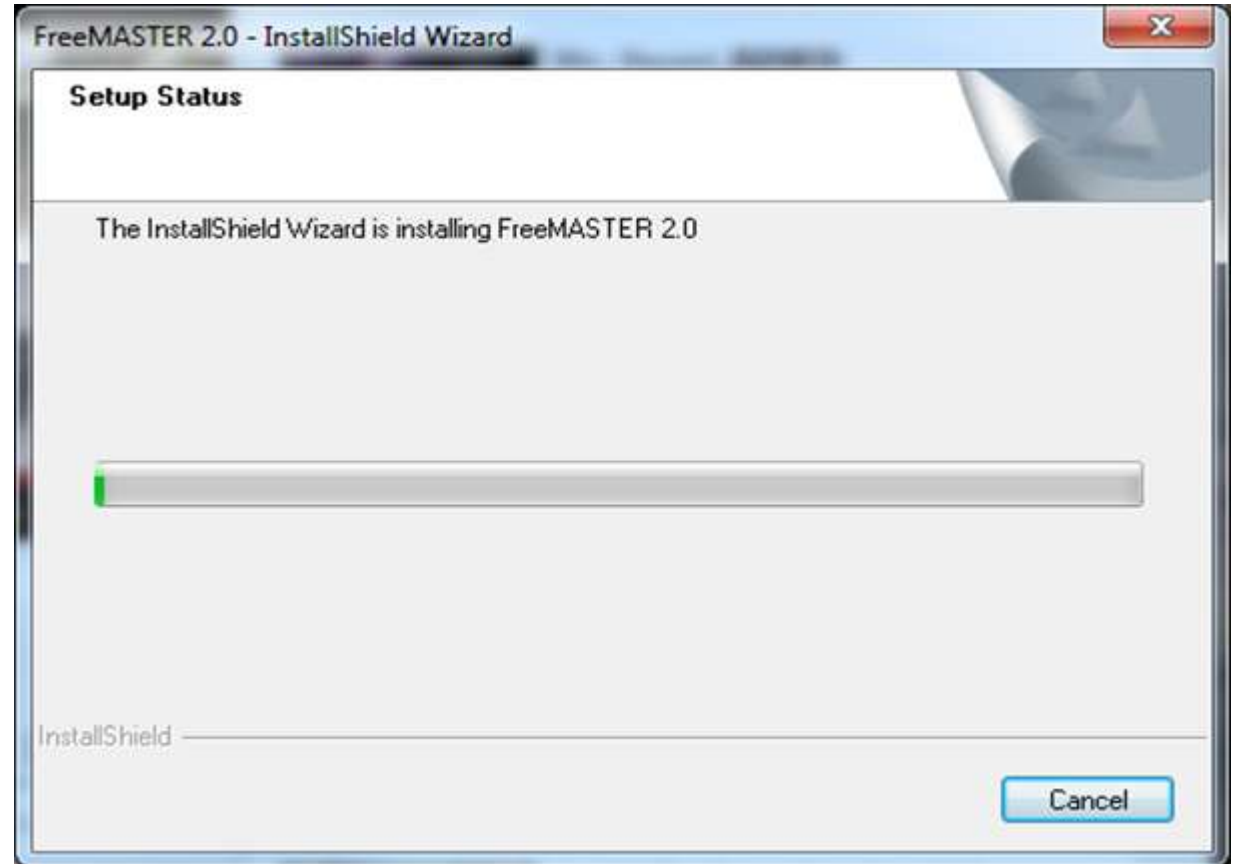
Downloading FreeMASTER for Real-Time Monitoring

- Install



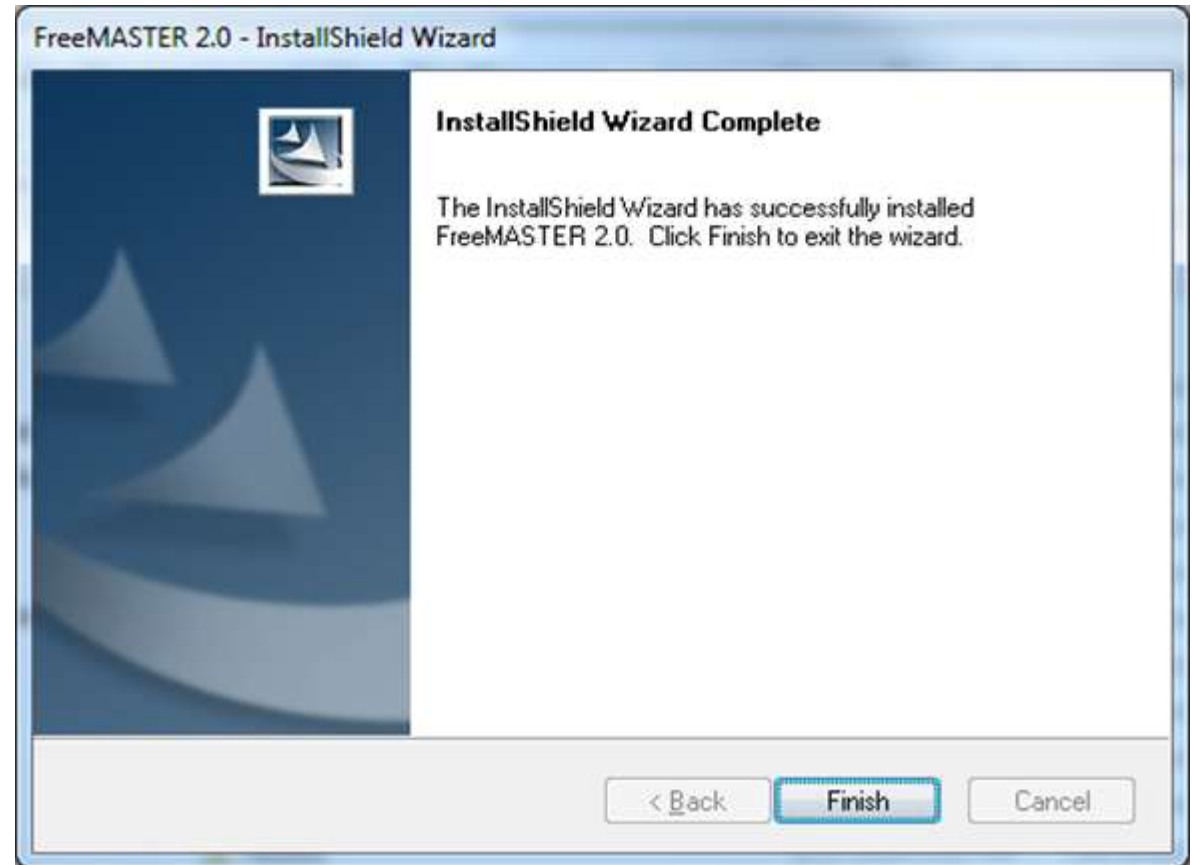
Downloading FreeMASTER for Real-Time Monitoring

- Wait for the installer to complete the installation



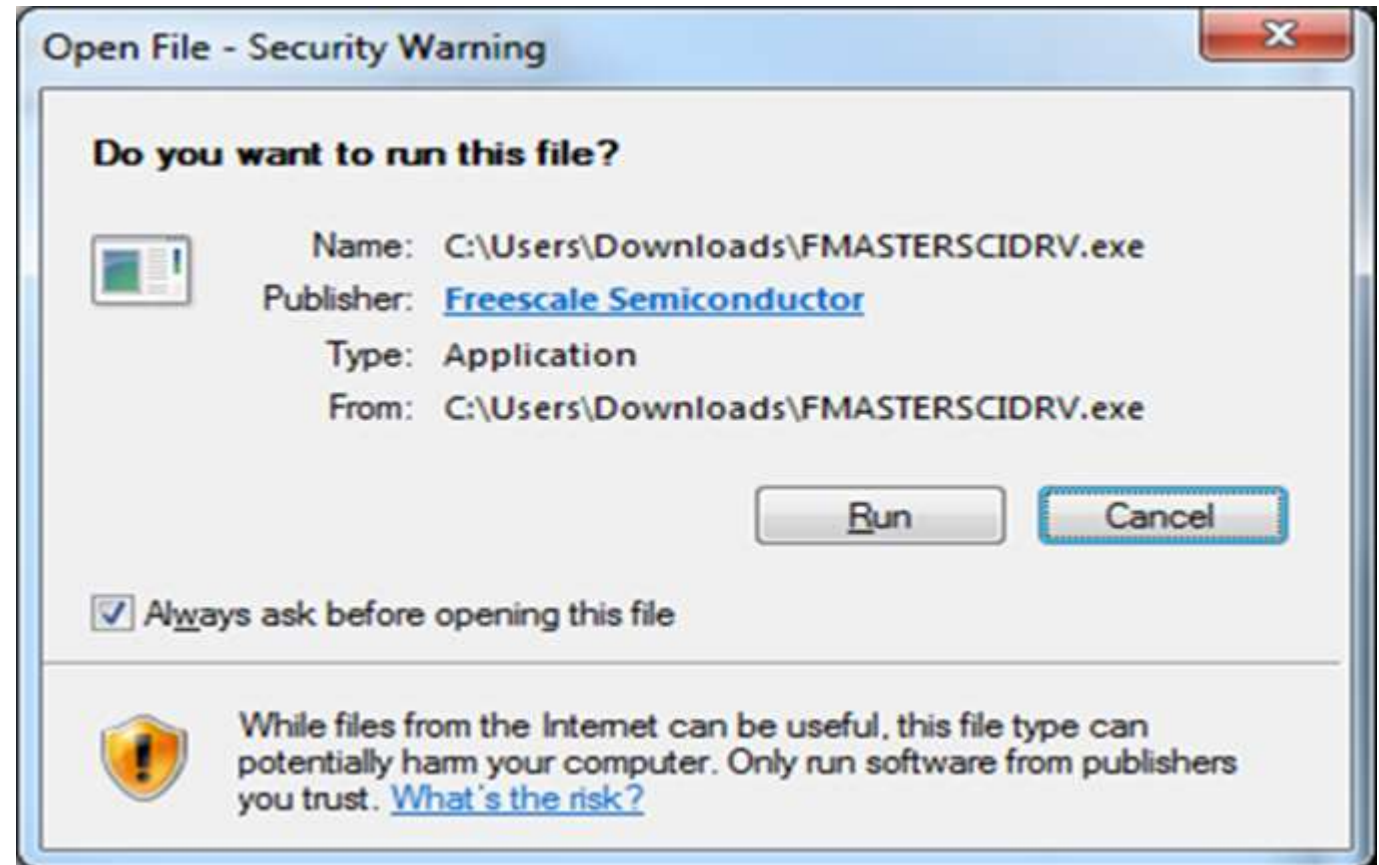
Downloading FreeMASTER for Real-Time Monitoring

- Click *Finish*
- FreeMASTER does not let you select an install path. The default location is C:\Program Files (x86)\Freescale\FreeMASTER 2.0\



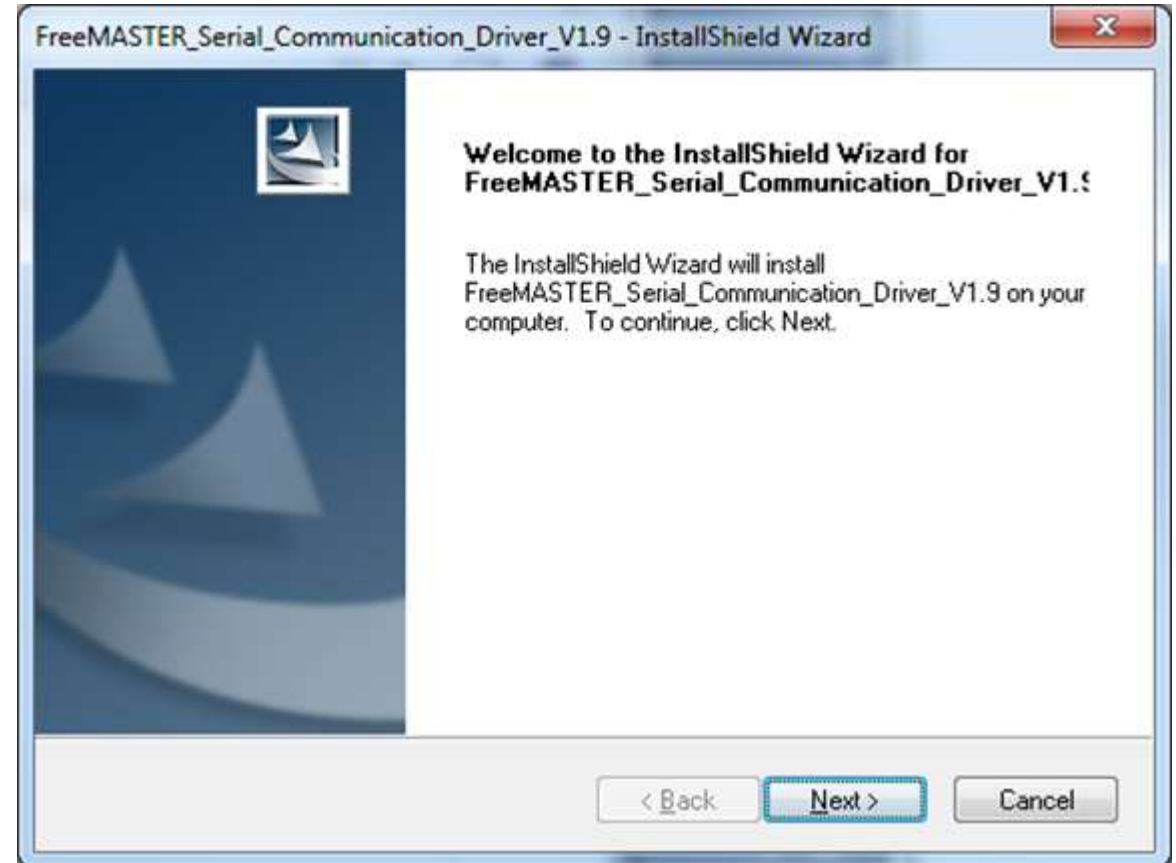
Downloading FreeMASTER for Real-Time Monitoring Communication Driver

- Double-click *FMMASTERSCIDRV.exe*
- Click *Run* when prompted



Downloading FreeMASTER for Real-Time Monitoring Communication Driver

- This dialog-box will appear
- Hit *Next*



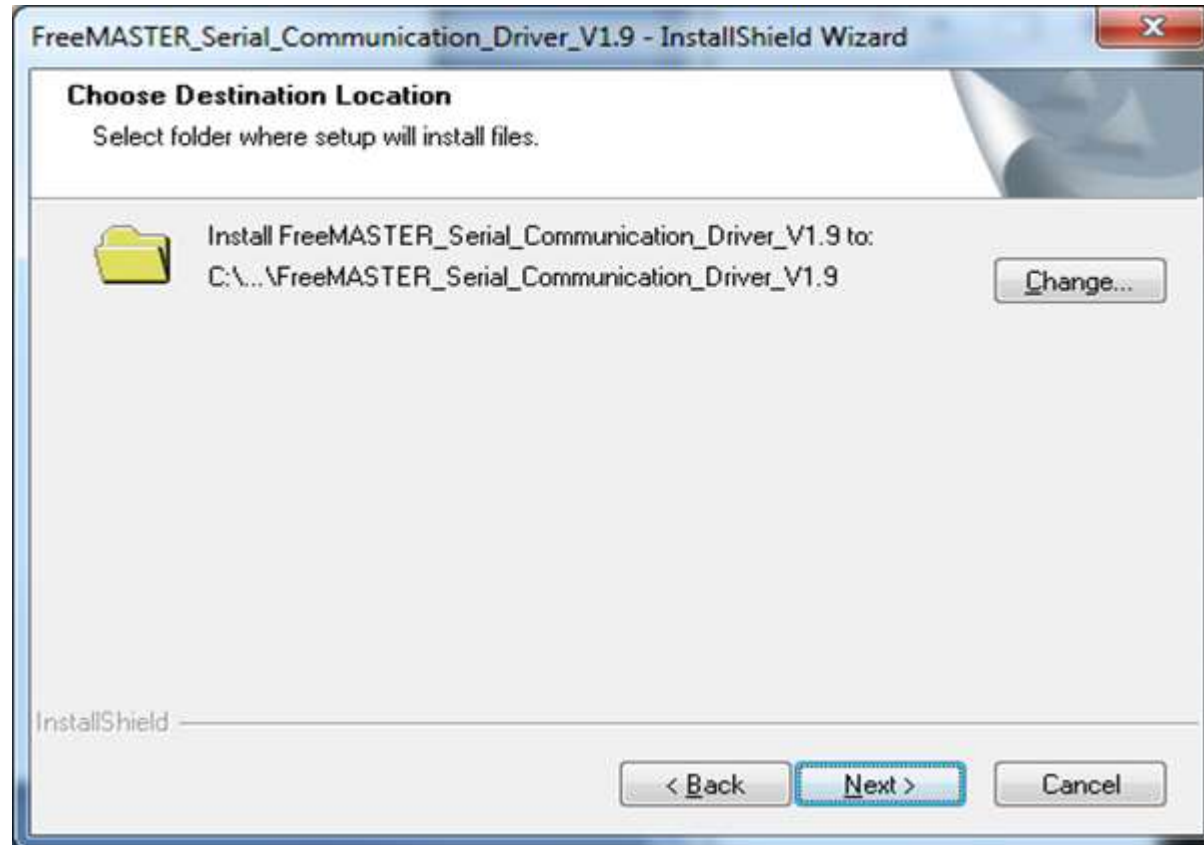
Downloading FreeMASTER for Real-Time Monitoring Communication Driver

- Accept the Freescale license agreement



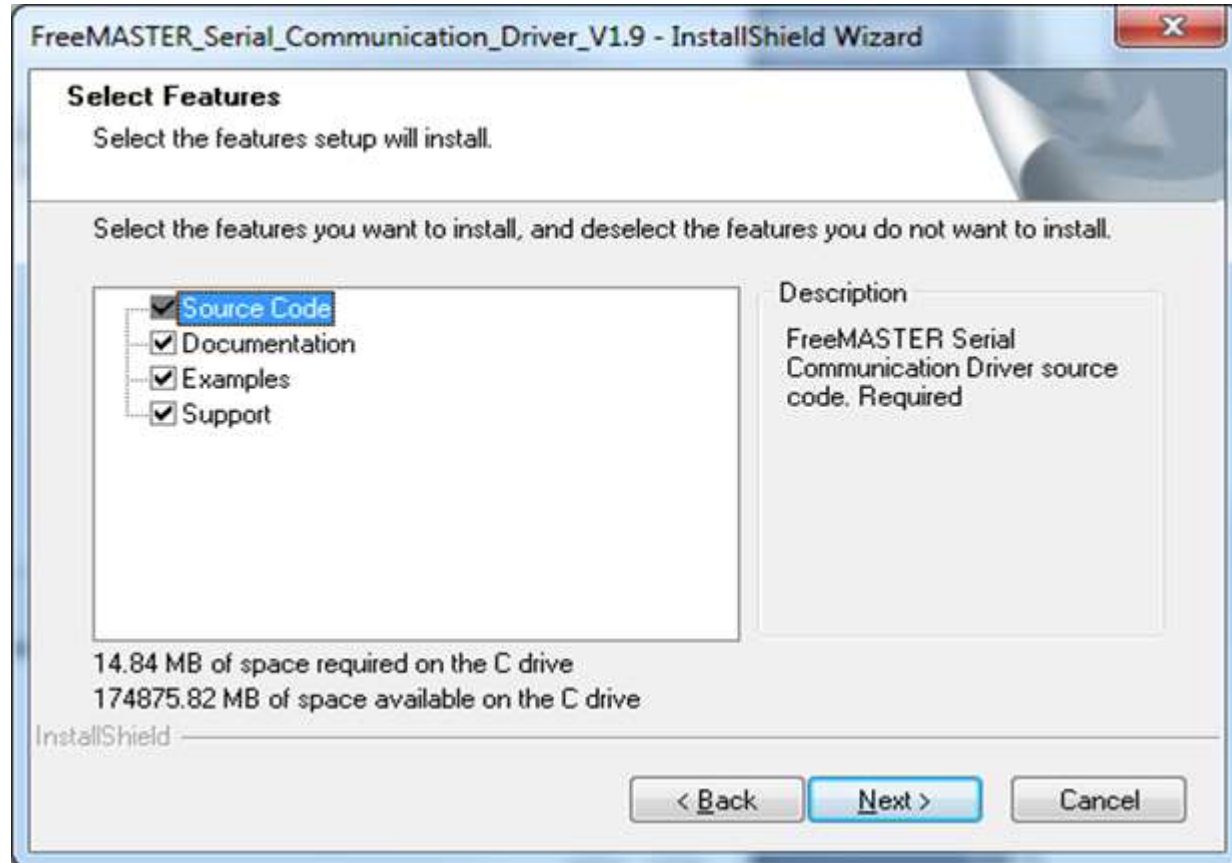
NXP Downloading FreeMASTER for Real-Time Monitoring Communication Driver

- It is recommended that you install FreeMASTER Communication Driver in the default location, since FreeMASTER must know where to look for it
- But you can change the directory if you so choose. Just make sure FreeMASTER settings are adjusted to reflect the change



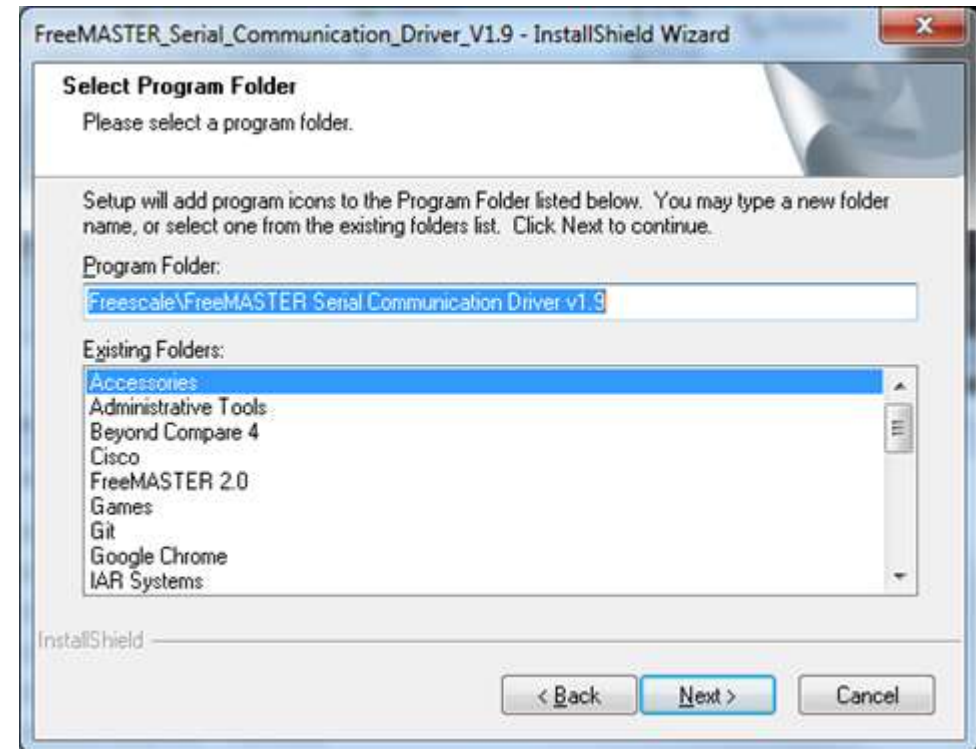
Downloading FreeMASTER for Real-Time Monitoring Communication Driver

- Select everything



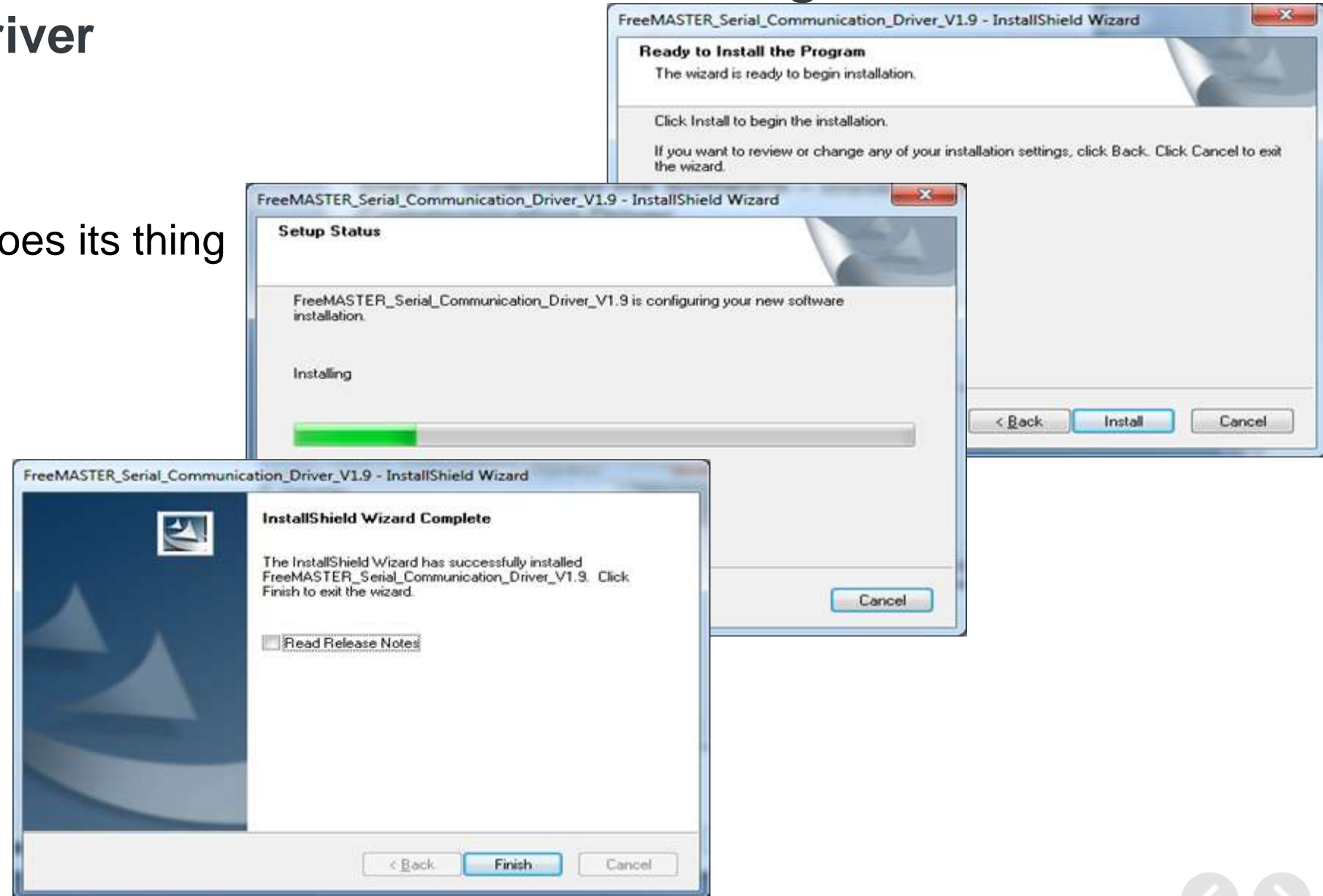
Unloading FreeMASTER for Real-Time Monitoring Communication Driver

- Choose the program folder
- This differs from the installation folder. The program folder is where the FMSCDrv appears in the Windows start menu
- Again, it is recommended to use the default location



Downloading FreeMASTER for Real-Time Monitoring Communication Driver

- Click Install
- Wait as the installer does its thing
- Hit *Finish*





www.Freescale.com