



ARROW ARIS Board

Software User's Guide

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by RELOC s.r.l. without notice.

Outline

- 1. Introduction 4
 - 1.1. Description 4
 - 1.2. Content 4
- 2. Getting started 5
 - 2.1. Prerequisites 5
 - 2.2. ARIS BSP 5
 - 2.3. Creating and building a new project 5
- 3. Use of a VSA component..... 8
 - 3.1. Files required 8
 - 3.2. Configuration..... 9
 - 3.3. Sample code 13
 - 3.4. Board configuration and connection 14
 - 3.5. Debugging the project 15

Revisions

REVISION	DATE	DESCRIPTION	STATUS	AUTHOR	REVISER
0.1	18/07/2016	Document created	draft	L. Dal Bello	
0.2	25/07/2016	Description for library integration	draft	L. Dal Bello	
1.0	27/07/2016	Document released	release	L. Dal Bello	A. Ricci

Disclaimer

All rights strictly reserved. Reproduction in any form is not permitted without written authorization from RELOC s.r.l.

RELOC s.r.l.

HEADQUARTERS

Via Borsari, 23/A 43126 – Parma (Italy)

info@reloc.it – www.reloc.it

Land +39-0521-1913460

Fax +39-0521-1913461

1. Introduction

1.1. Description

ARIS board, developed by RELOC for Arrow Electronics, is a ready-to-use Internet of Things (IoT) hardware and software platform that enables users to get their IoT applications up and running quickly, exploiting the Renesas Synergy development framework.

Based on Renesas Synergy S7 MCU with 240 MHz ARM Cortex-M4 core, the ARIS board has a host of features that equip it for IoT operation. Communication with other devices and the cloud is enabled via Bluetooth Low Energy (BLE 4.1/4.2), Wi-Fi b/g/n support as well as an Ethernet 10/100 port. NFC tag functionality is also included along with a crypto bootloader and support for over-the-air (OTA) firmware updates. Board sensing capabilities include motion detection, i.e. 3-axes accelerometer and 2-axes gyroscope, environmental temperature and humidity sampling.

The Renesas Synergy Platform helps to accelerate IoT designs: a proven combination of hardware and software makes development easier and faster, thus encouraging innovation and product differentiation. The combination of Arrow ARIS board and Renesas Synergy software platform enables developers to reduce time-to-market and decreases the total cost of ownership of a product over its lifetime.

1.2. Content

This document represents a quick start guide for the creation of a new project and use of VSA components with ARIS board R1.1 (i.e. production release).

Please refer to ARIS Hardware User's Guide document for additional information about the ARIS board.

2. Getting started

2.1. Prerequisites

The following development tools should be installed before trying to use any example based on ARIS board:

- e² studio v5.0.0.043 or later release
- Synergy Software Package v1.1.1 or later release

Please refer to the [Synergy Gallery](#) website for more information and for obtaining the required software.

2.2. ARIS BSP

ARIS Board Support Package (BSP) provides a default configuration for ARIS board peripherals.

ARIS BSP is provided with ARIS software examples, and is available for download at the following link:

<http://www.reloc.it/download/products/ARIS/user.aris.1.1.1.zip>

In order to install it, please unzip the archive and copy the file “user.aris.1.1.1.pack” in the following directory:

`%e2_studio_install_dir%\internal\projectgen\arm\Packs`

where

`%e2_studio_install_dir%` is typically the folder “C:\Renesas\e2_studio\”

2.3. Creating and building a new project

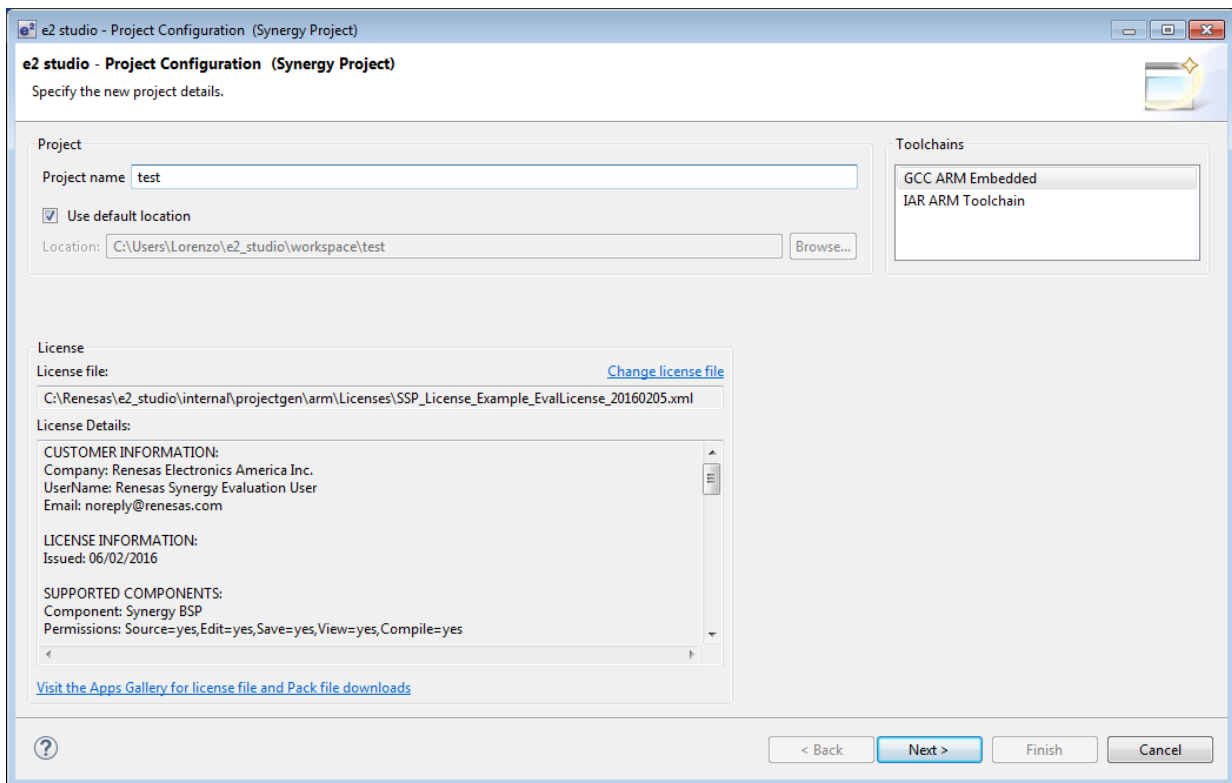
To create a Synergy project with ARIS board as a target, please follow these steps within e² studio:

- Select: File → New → Synergy Project in the menu of e² studio
- Assign a new project name and select GCC toolchain

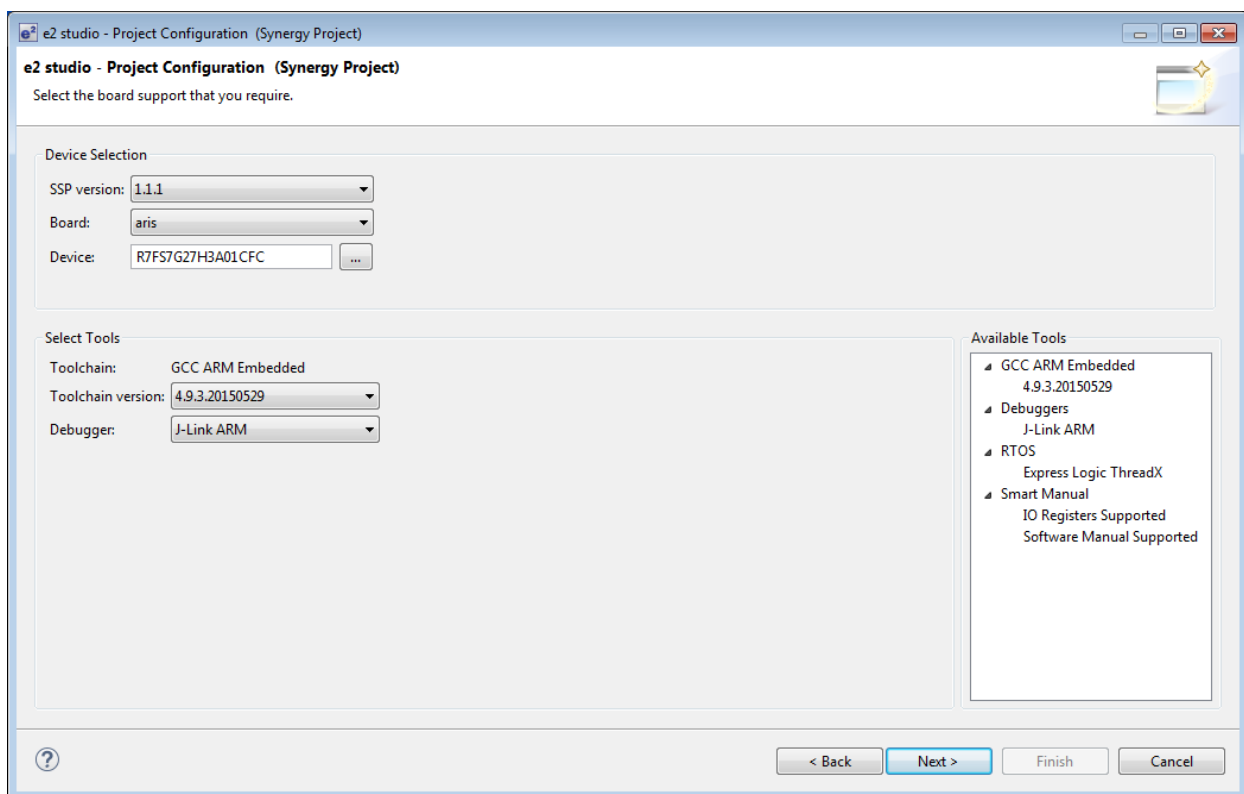
A license file is required for using Synergy platform, an evaluation license is provided with SSP in the following directory:

`%e2_studio_install_dir%\internal\projectgen\arm\Licenses`

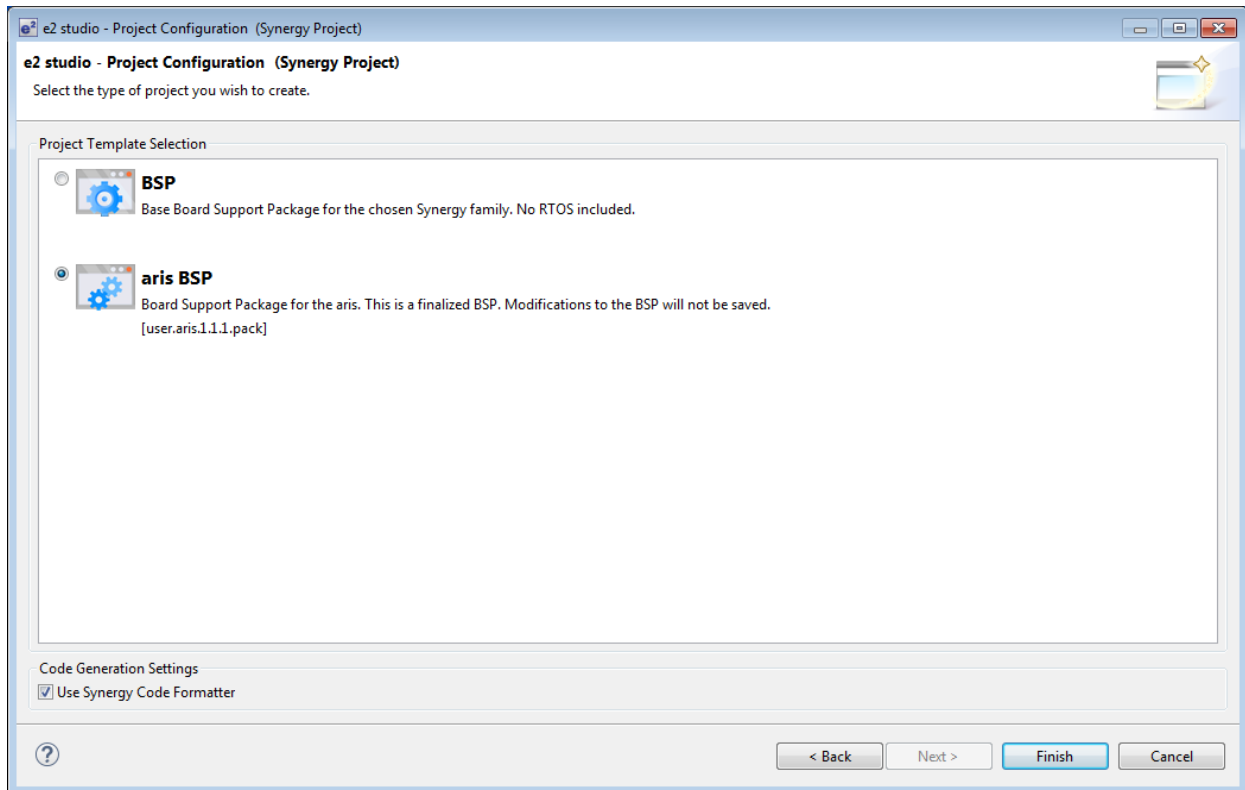
Please refer to [Synergy Gallery](#) website for additional information about obtaining a custom license.



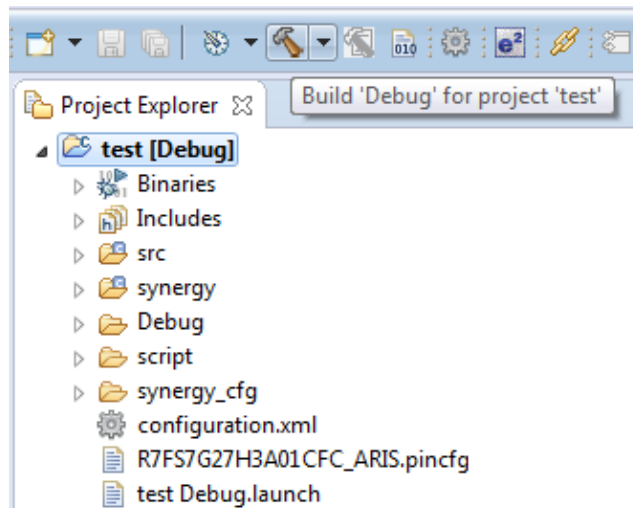
- Check that correct parameters are selected, according to the picture below:



- Select the 'aris BSP' and confirm with Finish button:



- You should now be able to build the project by clicking on the Build button:



Compilation should end without any error or warning.

- You can now start working on the project adding your own code to `src` → `hal_entry.c`

3. Use of a VSA component

VSA components made by RELOC are provided in the Synergy Gallery in binary form; the sample application contains all the required code to build and test the components.

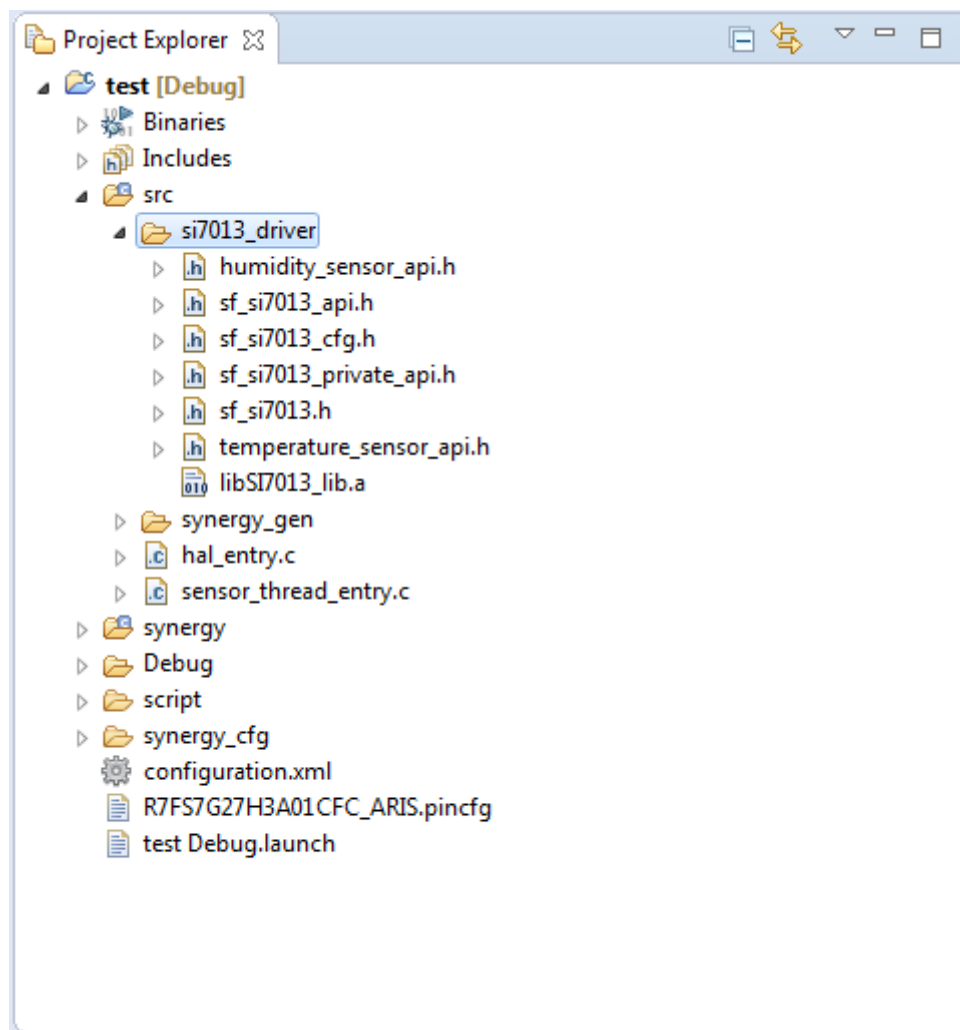
This chapter describes how to use a component using the Si7013 temperature and humidity driver as an example; all the steps required to include it into a new project will be discussed.

3.1. Files required

Both the library ('.a' extension) and the header files containing the API interface are needed.

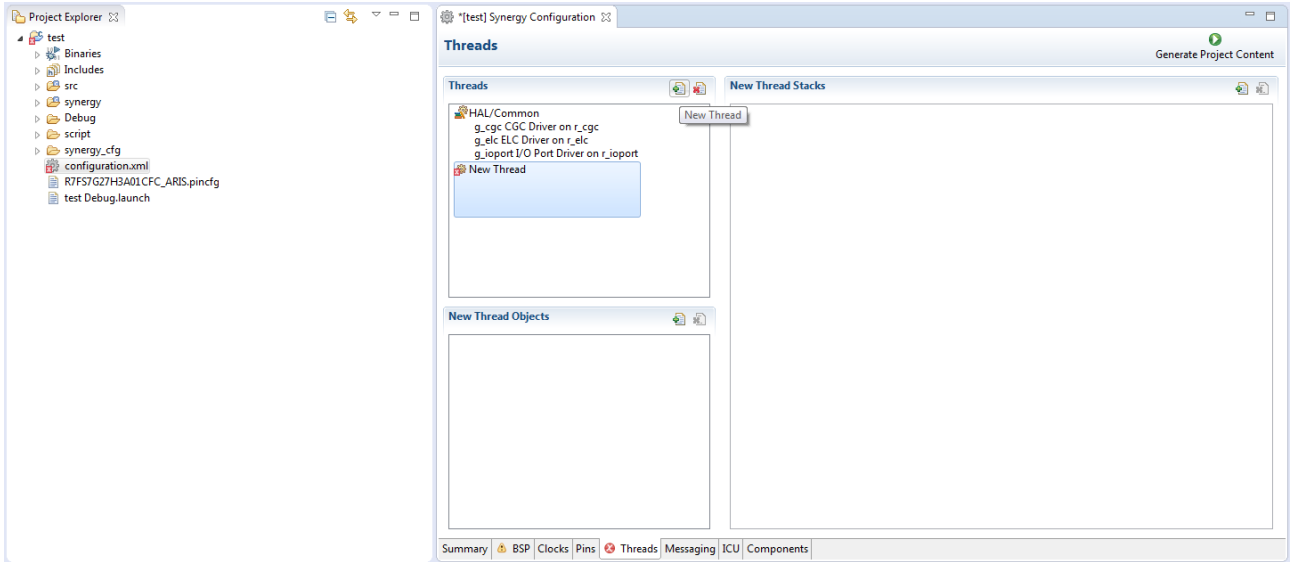
Starting from the project created in previous chapter, please create a new subfolder inside "src" directory with name "si7013_driver" and copy all the required files.

The following file structure should be visible within e² studio:

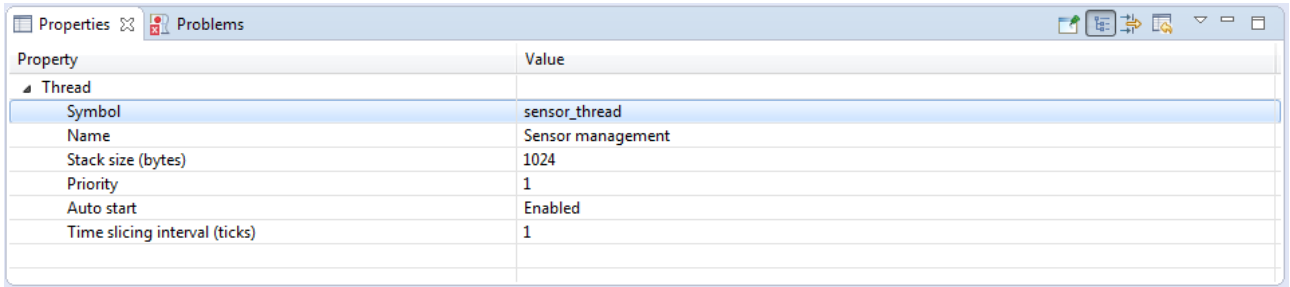


3.2. Configuration

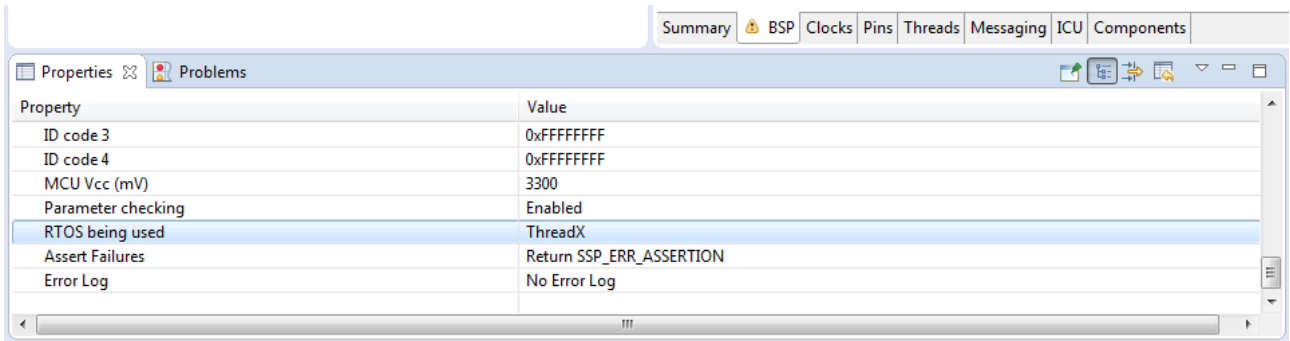
In order to add the new component to an existing project, some configuration must be performed. Open “configuration.xml” and add a new thread to the project:



You can change the name of the new thread and associated symbol in Properties panel; please change the values matching the following picture:



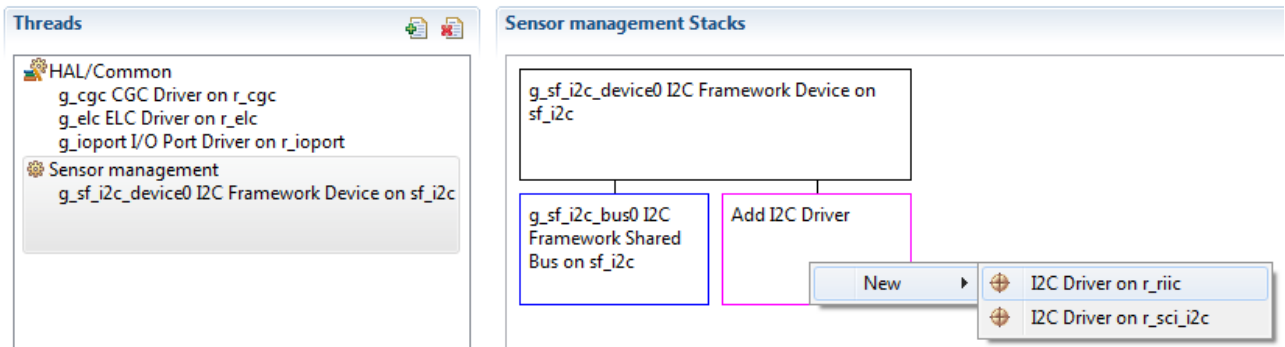
In order to use a thread we need to enable the RTOS: select the ‘BSP’ tab, property “RTOS being used” must be changed to “ThreadX”:



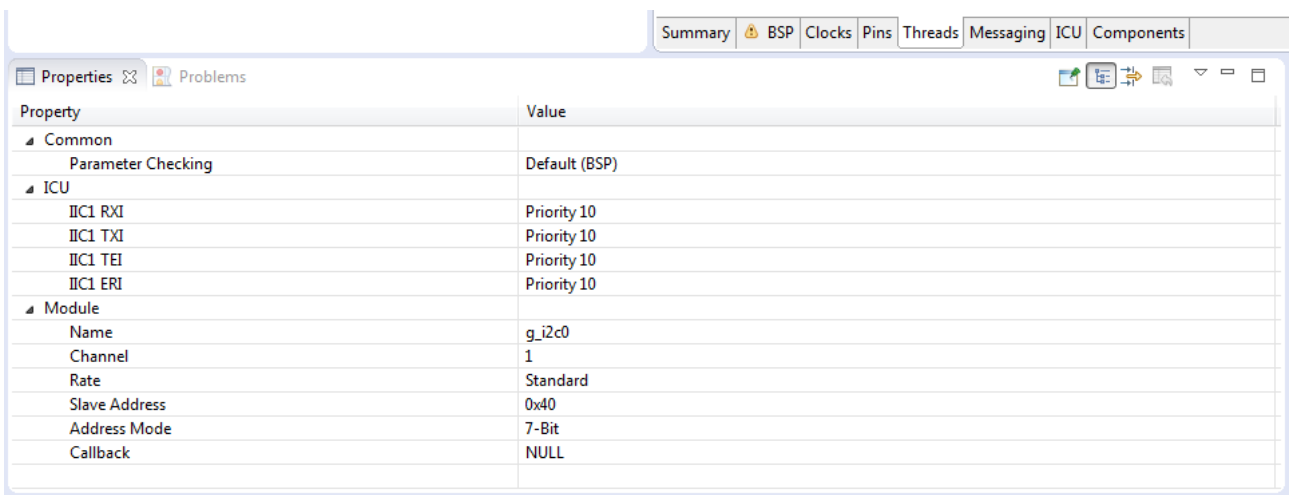
Return back to the 'Threads' tab and press the button in 'Sensor management Stacks' to add a new Framework, selecting Connectivity → I2C Framework Device on sf_i2c.



Select the highlighted component to add a new I2C driver, New → 'I2C Driver on r_riic':

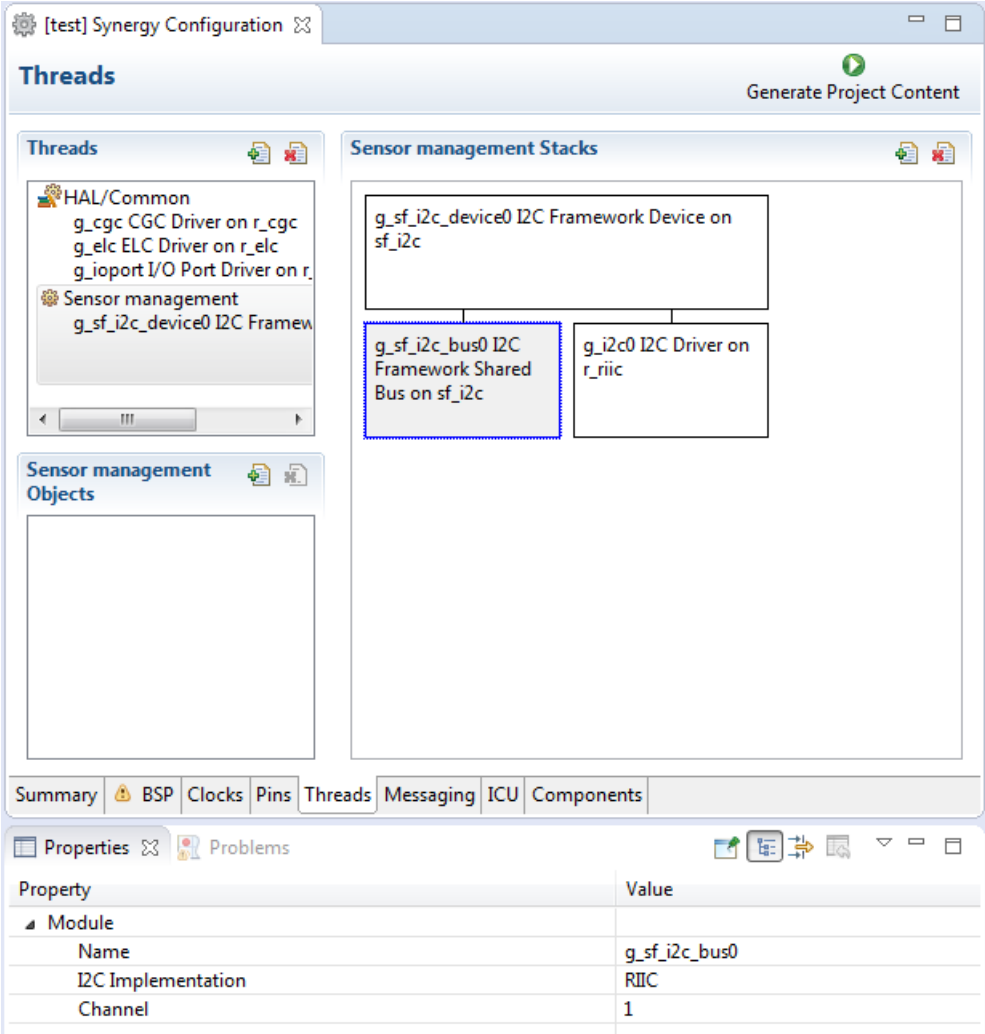


With the component still selected, change Channel to 1 (the sensor is connected to I2C channel 1 on the ARIS board) and set Slave Address to 0x40 (unique identifier of the sensor on I2C bus).

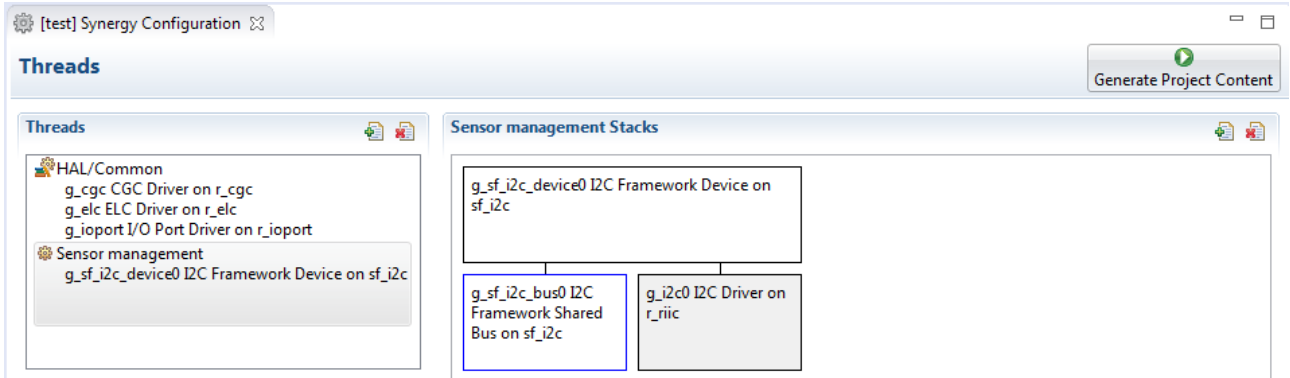


In ICU section, enable all interrupts giving them appropriate priority (10 in the example).

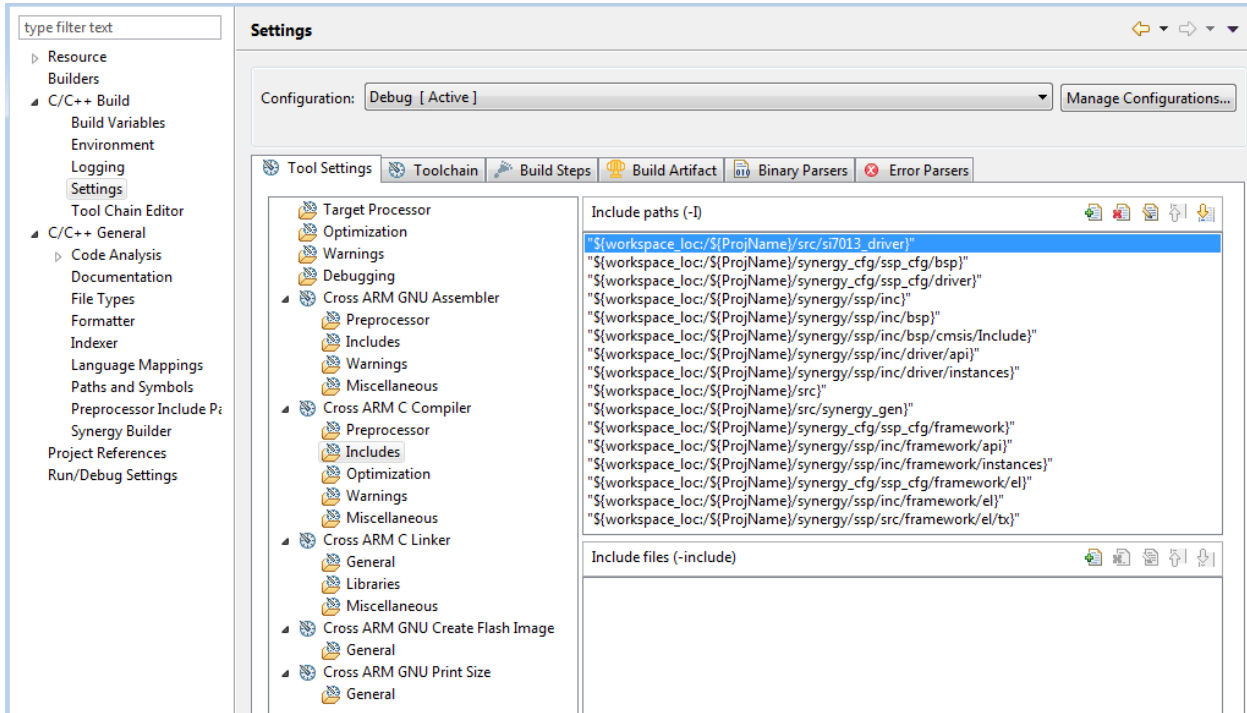
Selecting the Framework Shared Bus, please adjust the Channel to 1 matching the parameters previously set.



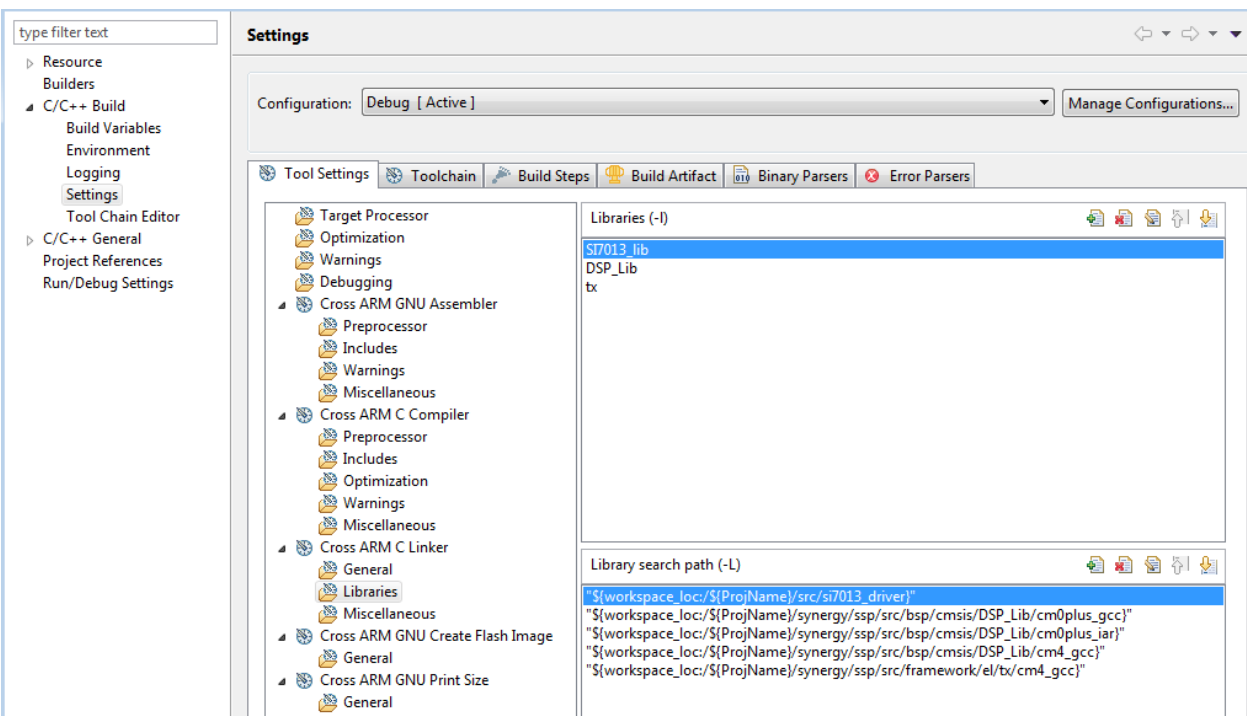
You can now save the modified configuration and press the “Generate Project Content” to update the Synergy files.



Select the C/C++ Perspective, right click on the test project and select Properties: in C/C++ Build → Settings → Tool Settings → Cross ARM C Compiler → Includes, and press the Plus button to add the directory containing header files to the Include paths, like in the following picture:



The same folder should be added to Library search path (in Cross ARM C Linker → Libraries section) and the specific library file should be added as well (please note that the library name should NOT contain "lib" in the beginning or '.a' extension in the end).



3.3. Sample code

The following code should be added in beginning of “sensor_thread_entry.c” file in order to include the relevant files and additional variables:

```
#include "sf_si7013_api.h"
#include "sf_si7013.h"
#include "temperature_sensor_api.h"
#include "humidity_sensor_api.h"

/* Definition of a new ASSERT function and prototype for the implementation */
#define ASSERT(a)          panic(a)
void panic( bool condition );

/* Definition of control and configuration structures needed for management of the sensor */
static sf_si7013_ctrl_t si7013_ctrl;
static const sf_si7013_cfg_t si7013_cfg = { .device = &g_sf_i2c_device0 };

static temperature_sensor_ctrl_t temp_sens_ctrl;
static const temperature_sensor_cfg_t temp_sens_cfg = { .p_extend_cfg = &si7013_cfg, .p_extend_ctrl = &si7013_ctrl };
static const temperature_sensor_instance_t temp_sens = { .p_ctrl = &temp_sens_ctrl, .p_cfg = &temp_sens_cfg, .p_api =
&g_sf_si7013_temperature_api };

static humidity_sensor_ctrl_t hum_sens_ctrl;
static const humidity_sensor_cfg_t hum_sens_cfg = { .p_extend_cfg = &si7013_cfg, .p_extend_ctrl = &si7013_ctrl };
static const humidity_sensor_instance_t hum_sens = { .p_ctrl = &hum_sens_ctrl, .p_cfg = &hum_sens_cfg, .p_api =
&g_sf_si7013_humidity_api };
```

This sample code can be put inside sensor_thread_entry function to test the behavior:

```
/* variables for temperature and humidity */
float temperature;
float humidity;

/* open function on driver and check of the return value */
ASSERT( temp_sens.p_api->open( temp_sens.p_ctrl, temp_sens.p_cfg ) == SSP_SUCCESS );
ASSERT( hum_sens.p_api->open( hum_sens.p_ctrl, hum_sens.p_cfg ) == SSP_SUCCESS );

ASSERT( temp_sens.p_api->readTemperature( temp_sens.p_ctrl, &temperature ) == SSP_SUCCESS );
/* Verify read value */

ASSERT( hum_sens.p_api->readHumidity( hum_sens.p_ctrl, &humidity ) == SSP_SUCCESS );
/* Verify read value */

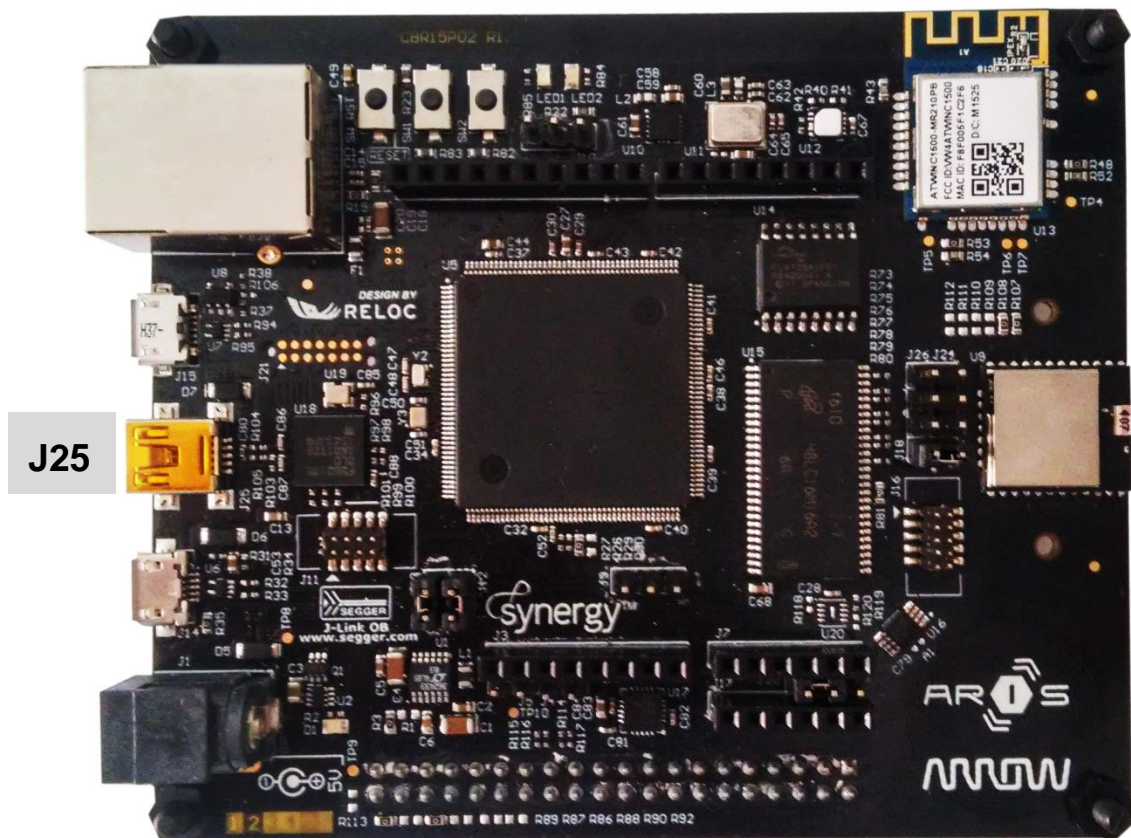
/* close function on driver and check of the return value */
ASSERT( hum_sens.p_api->close( hum_sens.p_ctrl ) == SSP_SUCCESS );
ASSERT( temp_sens.p_api->close( temp_sens.p_ctrl ) == SSP_SUCCESS );
```

Support ‘Panic’ function can be defined at the end of “sensor_thread_entry.c” file.

```
void panic( bool condition )
{
    if( condition == false )
    {
        while(1);
    }
}
```

Saving the modified file and pressing the Build button will start the compilation that should end without any error or warning.

3.4. Board configuration and connection



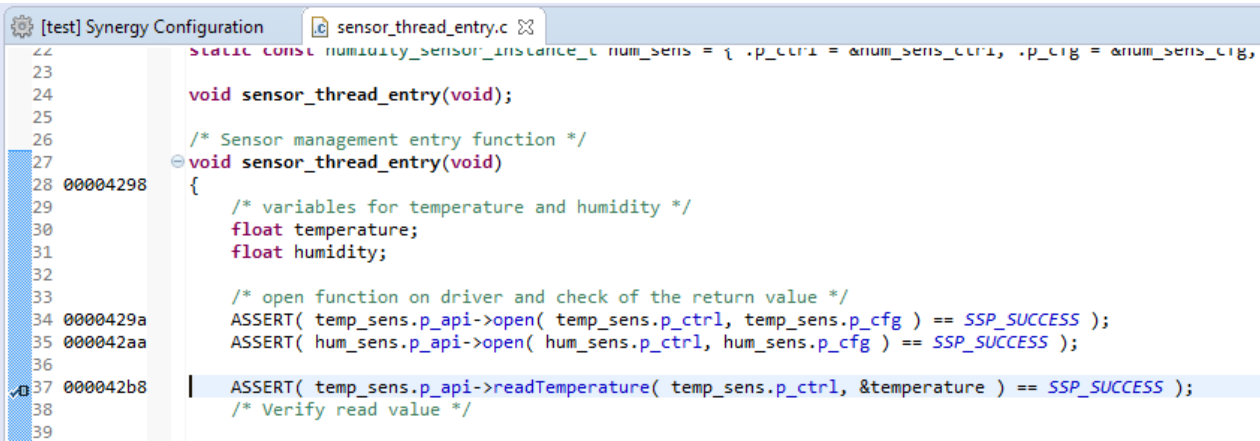
Connect ARIS board to your PC through the mini-USB port (J25): this connection will provide power supply to the board and will provide an interface for flashing and debug.

J12 and J13 (populated in the default configuration) provide power to the Synergy MCU. They can also be used to measure the current consumption of the S7 device.

Please refer to ARIS Hardware User's Guide for additional information about the power supply of the board.

3.5. Debugging the project

In order to check the correct behavior of the sample application, select the line of “sensor_thread_entry.c” with ‘readTemperature’ function and select Run → Toggle Breakpoint from the menu:



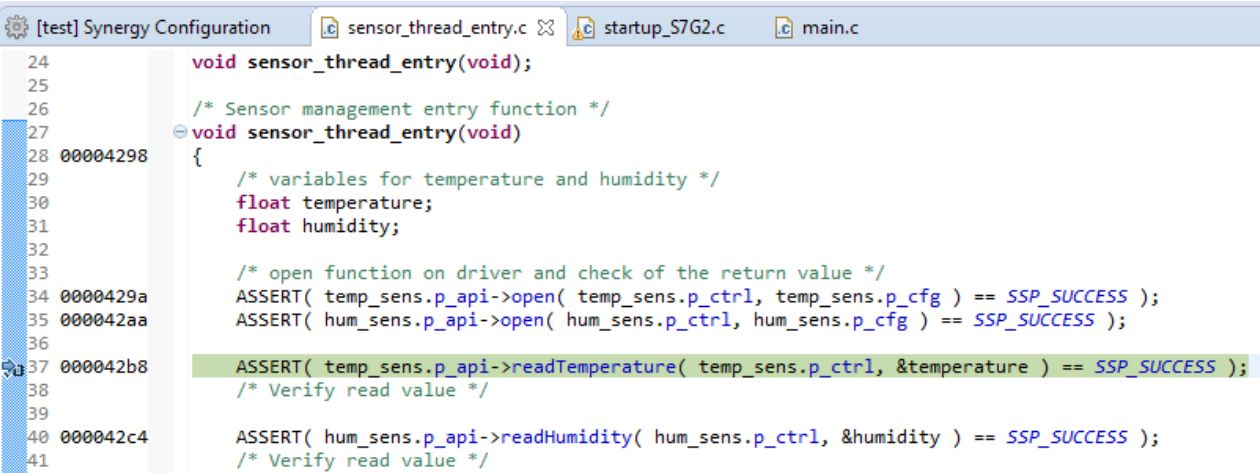
```

22 static const humidity_sensor_instance_t hum_sens = { .p_ctrl = &num_sens_ctrl, .p_cfg = &num_sens_cfg,
23
24 void sensor_thread_entry(void);
25
26 /* Sensor management entry function */
27 void sensor_thread_entry(void)
28 {
29     /* variables for temperature and humidity */
30     float temperature;
31     float humidity;
32
33     /* open function on driver and check of the return value */
34 ASSERT( temp_sens.p_api->open( temp_sens.p_ctrl, temp_sens.p_cfg ) == SSP_SUCCESS );
35 ASSERT( hum_sens.p_api->open( hum_sens.p_ctrl, hum_sens.p_cfg ) == SSP_SUCCESS );
36
37 ASSERT( temp_sens.p_api->readTemperature( temp_sens.p_ctrl, &temperature ) == SSP_SUCCESS );
38     /* Verify read value */
39

```

You can now start the Debug by selecting Run → Debug from the menu; if there is not any error the sample application will be downloaded to ARIS board and the development environment will switch automatically to Debug Perspective.

Click on Resume (F8) a couple of times and the execution should stop on the selected line:



```

24 void sensor_thread_entry(void);
25
26 /* Sensor management entry function */
27 void sensor_thread_entry(void)
28 {
29     /* variables for temperature and humidity */
30     float temperature;
31     float humidity;
32
33     /* open function on driver and check of the return value */
34 ASSERT( temp_sens.p_api->open( temp_sens.p_ctrl, temp_sens.p_cfg ) == SSP_SUCCESS );
35 ASSERT( hum_sens.p_api->open( hum_sens.p_ctrl, hum_sens.p_cfg ) == SSP_SUCCESS );
36
37 ASSERT( temp_sens.p_api->readTemperature( temp_sens.p_ctrl, &temperature ) == SSP_SUCCESS );
38     /* Verify read value */
39
40 ASSERT( hum_sens.p_api->readHumidity( hum_sens.p_ctrl, &humidity ) == SSP_SUCCESS );
41     /* Verify read value */

```

You can now click on Step Over (F6) a couple of times and the measurement taken from the sensors will appear on Variables window.

(x)= Variables		
Name	Type	Value
(x)- temperature	float	30.359777
(x)- humidity	float	29.2058411