



# **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.	Intr	oduction	.4
1	l.1.	Description	.4
1	.2.	Content	.4
2.	Get	tting started	. 5
2	2.1.	Prerequisites	. 5
2	2.2.	ARIS BSP	. 5
2	2.3.	Creating and building a new project	. 5
3.	Use	e of a VSA component	. 8
3	3.1.	Files required	. 8
3	3.2.	Configuration	.9
3	3.3.	Sample code	13
3	3.4.	Board configuration and connection	14
3	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.	info@reloc.it – <u>www.reloc.it</u>
HEADQUARTERS	Land +39-0521-1913460
Via Borsari, 23/A 43126 – Parma (Italy)	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:

- e<sup>2</sup> studio v5.0.0.043 or later release
- Synergy Software Package v1.1.1 or later release

Please refer to the <u>Synergy Gallery</u> 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 e<sup>2</sup> studio:

- Select: File  $\rightarrow$  New  $\rightarrow$  Synergy Project in the menu of e<sup>2</sup> 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 <u>Synergy Gallery</u> website for additional information about obtaining a custom license.



e2 studio - Project Configuration (Synergy Project)			
e2 studio - Project Configuration (Synergy Project) Specify the new project details.			
Project		Toolchains	
Project name test		GCC ARM Embedded IAR ARM Toolchain	
Location: C:\Users\Lorenzo\e2_studio\workspace\test	Browse		
License License file: Change license file C:\Renesas\e2_studio\internal\projectgen\arm\Licenses\SSP_License_Example_EvalLicense_20160205.xml			
License Details:			
CUSTOMER INFORMATION: Company: Reness Electronics America Inc. UserName: Reness Synergy Evaluation User Email: noreply@renesas.com			
Issued: 06/02/2016 SUPPORTED COMPONENTS: Component: Synergy BSP Permissions: Source=yes,Edit=yes,Save=yes,View=yes,Compile=yes			
Visit the Apps Gallery for license file and Pack file downloads			
?	< Back	Next > Finish	Cancel

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

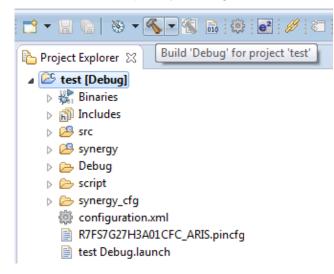
-	Configuration (Synergy Project) nort that you require.	
Device Selection SSP version: 1.1.1 Board: aris	▼	
Device: R7FS Select Tools Toolchain: Toolchain version: Debugger:	GCC ARM Embedded 4.9.3.20150529 J-Link ARM	Available Tools    GCC ARM Embedded 4.9.3.20150529   Debuggers J-Link ARM  RTOS Express Logic ThreadX Smart Manual IO Registers Supported Software Manual Supported
2		< Back Next > Finish Cancel



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

e² e2 studio - Project Configuration (Synergy Project)	- • ×
e2 studio - Project Configuration (Synergy Project) Select the type of project you wish to create.	
Project Template Selection	
BSP Base Board Support Package for the chosen Synergy family. No RTOS included.	
It is a finalized BSP. Modifications to the BSP will not be saved. [user.aris111.pack]	
Code Generation Settings	
✓ Use Synergy Code Formatter	
?       < Back	Cancel

• 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.

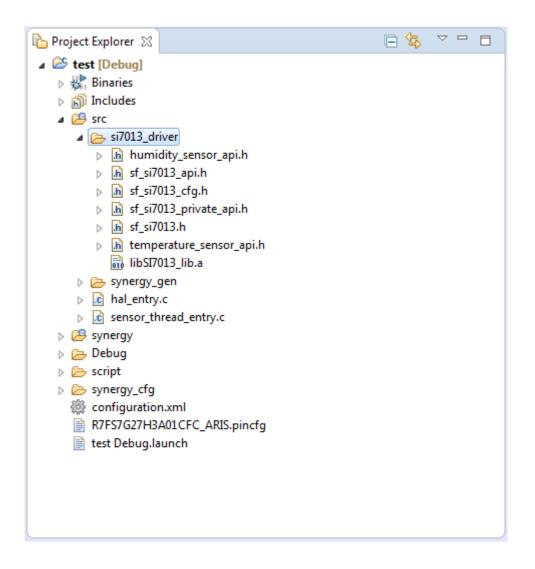
This chapter describes how to use a component using the Si7013 temperature and humidity driver as an example; all the steps required to include it into a new project will be discussed.

#### 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  $e^2$  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:

⊳ ၍) Includes	oject Explorer 🛛 📄 🔩 👻 🖻	▽ 🖻 🔲 🏟 *(test) Synergy Configuration 🔯	
> @ src     > @ spragy       > @ bebug     > @ sprigy.       > @ sprigy.dg     > g.egc CGC Driver on r.egc       g.egc ELC Driver on r.egc     g.egc ELC Driver on r.egc       g.egt ELC Driver on r.egc     g.egt ELC Driver on r.egc       g.egt ELC Driver on r.egc     g.egt ELC Driver on r.egc       g.egt ELC Driver on r.egc     g.egt ELC Driver on r.egc       g.egt ELC Driver on r.egc     g.egt ELC Driver on r.egc       g.egt ELC Driver on r.egc     g.egt ELC Driver on r.egc       g.egt ELC Driver on r.egc     g.egt ELC Driver on r.egc       g.egt ELC Driver on r.egc     g.egt ELC Driver on r.egc       g.egt ELC Driver on r.egc     g.egt ELC Driver on r.egc       g.egt ELC Driver on r.egc     g.egt ELC Driver on r.egc       g.egt ELC Driver on r.egc     g.egt ELC Driver on r.egc       g.egt ELC Driver on r.egc     g.egt ELC Driver on r.egc       g.egt ELC Driver on r.egc     g.egt ELC Driver on r.egc       g.egt ELC Driver on r.egc     g.egt ELC Driver on r.egc       g.egt ELC Driver on r.egc     g.egt ELC Driver on r.egc       g.egt ELC Driver on r.egc     g.egt ELC Driver on r.egc       g.egt ELC Driver on r.egc     g.egt ELC Driver on r.egc       g.egt ELC Driver on r.egc     g.egt ELC Driver on r.egc       g.egt ELC Driver on r.egc     g.egt ELC Driver on r.egc       g.egt ELC Driver on r.egc     g.egt ELC Driver	🖑 Binaries	Threads	Generate Project Content
Summary 🙆 BSP [Clocks Pins] 🚱 Threads [Messaging] [CU] Components	Ge src Ge Sphergy Bebug Bebug Bestigt Best	Image: Additional and the second s	1

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

Properties 🛛 🛃 Problems	
Property	Value
▲ Thread	
Symbol	sensor_thread
Name	Sensor management
Stack size (bytes)	1024
Priority	1
Auto start	Enabled
Time slicing interval (ticks)	1

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":

	Summary 💩 BSP Clocks Pins Threads Messaging ICU Components
🔲 Properties 🙁 🤶 Problems	
Property	Value
ID code 3	0xFFFFFFF
ID code 4	0xFFFFFFF
MCU Vcc (mV)	3300
Parameter checking	Enabled
RTOS being used	ThreadX
Assert Failures	Return SSP_ERR_ASSERTION
Error Log	No Error Log
•	III



Return back to the 'Threads' tab and press the button in 'Sensor management Stacks' to add a new Framework, selecting Connectivity  $\rightarrow$  I2C Framework Device on sf\_i2c.

Threads	<b>e e</b>	Sensor management Stacks		Driver >		8888 7000
HAL/Common g_cgc CGC Driver on r_cgc				Framework	Analog	• <u>ы</u> ыыы
g_elc ELC Driver on r_elc				P400 32	Audio	F
g_ioport I/O Port Driver on r_ioport			•	Communications Framework on sf_el_nx_comms	Connectivity	•
Consor management				Communications Framework on sf_el_ux_comms	File System	
			+	Communications Framework on sf_uart_comms	Graphics	F
			•	I2C Framework Device on sf_i2c	Input	- +
			+	I2C Framework Shared Bus on sf_i2c	Networking	- +
			+	SPI Framework Device on sf_spi	RTOS	- +
				SPI Framework Shared Bus on sf_spi	Services	- +
Sensor management Objects	A A			P800 32 P801 32	USB	

Select the highlighted component to add a new I2C driver, New  $\rightarrow$  'I2C Driver on r\_riic':

Threads 🔮 着	Sensor management Stacks
<ul> <li>HAL/Common g_cgc CGC Driver on r_cgc g_elc ELC Driver on r_elc g_ioport I/O Port Driver on r_ioport</li> <li>Sensor management g_sf_i2c_device0 I2C Framework Device on sf_i2c</li> </ul>	g_sf_i2c_device0 I2C Framework Device on sf_i2c g_sf_i2c_bus0 I2C Framework Shared Bus on sf_i2c
	New

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).

	Summary 💩 BSP Clocks Pins Threads Messaging ICU Components
🗏 Properties 🔀 🔝 Problems	
Property	Value
⊿ Common	
Parameter Checking	Default (BSP)
⊿ ICU	
IIC1 RXI	Priority 10
IIC1 TXI	Priority 10
IIC1 TEI	Priority 10
IIC1 ERI	Priority 10
⊿ Module	
Name	g_i2c0
Channel	1
Rate	Standard
Slave Address	0x40
Address Mode	7-Bit
Callback	NULL

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.

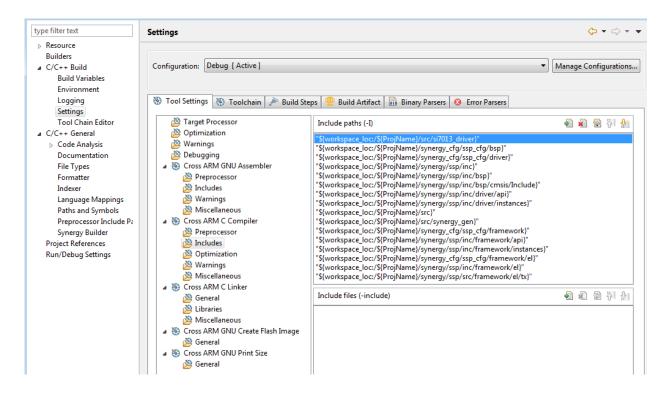
🌼 [test] Synergy Configuration 🔀			
Threads		O Generate Project Content	
Threads       Image: Sensor management         g_sf_i2c_device0 12C Framew         Sensor management         g_sf_i2c_device0 12C Framew	Sensor management Stacks g_sf_i2c_device0 I2C Framew sf_i2c g_sf_i2c_bus0 I2C Framework Shared Bus on sf_i2c g_i2 g_i2	vork Device on	
Summary 💩 BSP Clocks Pins Th	reads Messaging ICU Compone	ents TET TA C	
⊿ Module			
Name		g_sf_i2c_bus0	
I2C Implementation		RIIC	
Channel		1	

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

🏟 [test] Synergy Configuration 🔀		
Threads		Generate Project Content
Threads	Sensor management Stacks	
HAL/Common g_cgc CGC Driver on r_cgc g_elc ELC Driver on r_elc g_ioport I/O Port Driver on r_ioport	g_sf_i2c_device0 I2C Framework Device on sf_i2c	
Sensor management g_sf_i2c_device0 I2C Framework Device on sf_	i <sup>2</sup> c g_sf_i <sup>2</sup> c_bus <sup>0</sup> I <sup>2</sup> C Framework Shared Bus on sf_i <sup>2</sup> C Bus on sf_i <sup>2</sup> C	



Select the C/C++ Perspective, right click on the test project and select Properties: in C/C++ Build  $\rightarrow$  Settings  $\rightarrow$  Tool Settings  $\rightarrow$  Cross ARM C Compiler  $\rightarrow$  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  $\rightarrow$  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).

oe filter text	Settings		
Resource			
Builders			
C/C++ Build	Configuration: Debug [ Active ]		<ul> <li>Manage Configuration</li> </ul>
Build Variables			
Environment			
Logging	🛞 Tool Settings 🛞 Toolchain 🎤 Build St	eps  🚇 Build Artifact  🗟 Binary Parsers 🕺 Error Pa	rsers
Settings			
Tool Chain Editor	Processor Target Processor	Libraries (-I)	🛃 📾 🖗 🖓
C/C++ General	Optimization	SI7013 lib	
Project References	Warnings	DSP_Lib	
Run/Debug Settings	pebugging 🖉	tx	
	Cross ARM GNU Assembler		
	Preprocessor		
	Includes		
	2 Warnings		
	Miscellaneous		
	Cross ARM C Compiler     Preprocessor		
	Optimization		
	2 Warnings		
	A Miscellaneous		
	Cross ARM C Linker		
	A General	Library search path (-L)	🔞 🗐 🗟 취 🚽
	(A Libraries	"\${workspace_loc:/\${ProjName}/src/si7013_driver}"	
	Miscellaneous	"S{workspace_loc:/S{ProjName}/src/si/015_driver}	msis/DSP_Lib/cm0plus_acc}"
	Cross ARM GNU Create Flash Image	"\${workspace_loc:/\${ProjName}/synergy/ssp/src/bsp/c	
	🖄 General	"\${workspace_loc:/\${ProjName}/synergy/ssp/src/bsp/c	
	A 🐼 Cross ARM GNU Print Size	\${workspace_loc:/\${ProjName}/synergy/ssp/src/frame	work/el/tx/cm4_gcc}"
	🖄 General		



#### 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->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( hum_sens.p_api->close( temp_sens.p_ctrl ) == SSP_SUCCESS);
ASSERT( hum_sens.p_api->close( temp_sens.p_ctrl ) == SSP_SUCCESS);
```

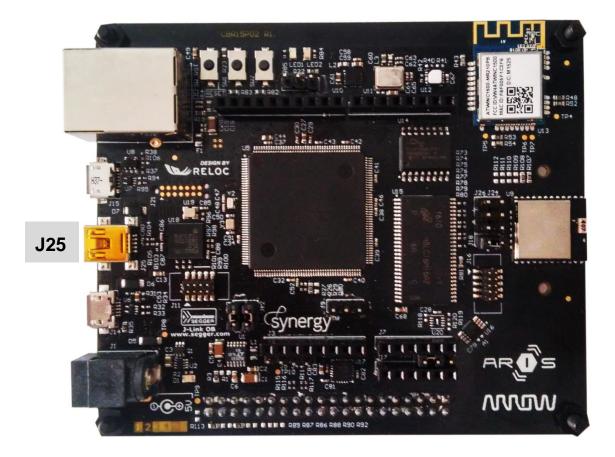
Support 'Panic' function can be defined at the end of "sensor\_thread\_entry.c" file.



```
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  $\rightarrow$  Toggle Breakpoint from the menu:

_C187
D,

You can now start the Debug by selecting Run  $\rightarrow$  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:

🔅 [test] Synergy Configuration		Configuration	🝺 sensor_thread_entry.c 🛛	💽 startup_S7G2.c	🖻 main.c
24					
25	25 26 /* Sensor management entry function */				
27		void sensor	_thread_entry(void)		
28	28 00004298 {				
29		/* variables for temperature and humidity */			
30		float temperature;			
31		float h	float humidity;		
32	32				
33		/* open	function on driver and	l check of the retu	urn value */
34	0000429a	ASSERT (	<pre>temp_sens.p_api-&gt;open(</pre>	<pre>temp_sens.p_ctrl;</pre>	<pre>, temp_sens.p_cfg ) == SSP_SUCCESS );</pre>
35	000042aa	ASSERT(	<pre>hum_sens.p_api-&gt;open(</pre>	hum_sens.p_ctrl, H	<pre>hum_sens.p_cfg ) == SSP_SUCCESS );</pre>
36					
🟓 🛛 37	000042b8	ASSERT (	<pre>temp_sens.p_api-&gt;readT</pre>	emperature( temp_	<pre>sens.p_ctrl, &amp;temperature ) == SSP_SUCCESS );</pre>
38		/* Veri	fy read value */		
39					
40	000042c4	ASSERT (	hum_sens.p_api->readHu	midity( hum_sens.	<pre>p_ctrl, &amp;humidity ) == SSP_SUCCESS );</pre>
41		/* Veri	fy 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 🔀 💁 Breakpoints	🚻 Registers 🛋 Modules 🙀 Expressions	👴 Eventpoints 🛛 🔲 IO Registers
Name	Туре	Value
(×)= temperature	float	30.3599777
(×)= humidity	float	29.2058411