

R20AN0418EU0103

Renesas Synergy<sup>™</sup> Platform

Debugging Your Application with IAR C-SPY Rev.1.03 Feb 7, 2018

# Introduction

This Application Note uses the template project SK-S7G2 Synergy<sup>TM</sup> MCU Group Blinky with ThreadX<sup>®</sup>. Refer to *Getting Started with IAR Embedded Workbench<sup>®</sup> for Renesas Synergy*<sup>TM</sup> document to create a project through the Synergy Standalone Configurator (SSC).

# Contents

1.	Initial Settings	2
2.	Debug basics and Live Watch	5
3.	Conditional and Data Breakpoints	8
4.	C-SPY Macros	.11
5.	PRINTF via SWO and Semihosting	.12
6.	ITM Events	.14
7.	Interrupt Logging	.18
8.	Attach to running target	.19
9.	ThreadX RTOS plugin	.22
10.	Stack Analysis during Debug (View > Stack Usage)	.24
We	osite and Support	.27



## 1. Initial Settings

1. Open the project options for **S7G2-SK Blinky** with **ThreadX**<sup>®</sup> by **right clicking** on the project in the workspace. Select the project in **bold** in the **project tree** to get access to all the project options.



Figure 1 Debug project

- 2. Select the **J-Link**<sup>®</sup> driver that is populated on the Synergy MCU board. Go to **Step 4** for the final debug probe settings. You do not need to close the **Options** windows yet.
- 3. Under the Category window select the **Debugger** setting. Under the **Setup** tab and **Driver** window select **J-Link/J-Trace**.

Options for node "lab1"		×
Category: General Options Static Analysis Runtime Checking	Factory Settings	
Assembler Output Converter Custom Build Build Actions Linker Debugger	Driver Run to J-Link/J-Trace main Simulator Angel CMSIS DAP GDB Senser	
Angel CMSIS DAP GDB Server IAR ROM-monitor I-jet/JTAGjet	AR ROM-monitor UAR To Contemporate Contempor	
J-Link/J-Trace TI Stellaris Macraigor PE micro RDI ST-LINK	Third-Party Driver Ti XDS	
Third-Party Driver TI XDS	OK Cancel	

Figure 2 J-Link driver



4. In the **Category** window choose **J-Link/J-Trace**. Select the **JTAG/SWD speed** to **Auto** and set the **CPU clock** to **120 MHz**. It will be required for the **SWO clock** interface that will be used for the complete debugging in this application note.

Options for node "lab1"	×
Category: General Options Static Analysis Runtime Checking C/C++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator Angel CMSIS DAP GDB Server IAR ROM-monitor I-iet/ITAGiet J-Link/J-Trace	Factory Settings         Setup       Connection       Breakpoints         Reset       0       0         JTAG/SWD speed       0       0         JTAG/SWD speed       Clock setup       0         Initial       1000       kHz       CPU clock:         Initial       1000       kHz       SWO clock:         Adaptive       2000       kHz         ETM/ETB       Prefer ETB       0
PE micro RDI ST-LINK Third-Party Driver TI XDS	OK Cancel

### Figure 3 Setting the CPU clock

5. In the **Connection** tab click the **SWD** button under **Interface** and confirm all changes by clicking **OK**.

Options for node "lab1" Category: General Options Static Analysis Runtime Checking C/C++ Compiler Assembler Output Converter Custom Build Build Actions	Factory Settings         Setup       Connection         Breakpoints         Communication         USB:       Device 0         USB:       Device 0         Communication
Build Actions Linker Debugger Simulator Angel CMSIS DAP GDB Server IAR ROM-monitor I-jet/JTAGjet	IP address:       aaa.bbb.ccc.ddd       Serial no:         Interface       JTAG scan chain         JTAG       JTAG scan chain with multiple targets         TAP number:       0         SwD       Scan chain contains non-ARM devices         Preceeding bits:       0
J-Link()-Frace TI Stellaris Macraigor PE micro RDI ST-LINK Third-Party Driver TI XDS	Log communication     \$PROJ_DIR\$\cspycomm.log     OK Cancel

Figure 4 Selecting SWD interface



6. **Download** and **Debug** your application by selecting **Project** > **Download** and **Debug** (shortcut key combination is **CTRL** + **D**).



Figure 5 Selecting Download and Debug

7. The application should halt in the **main** function.



Figure 6 Halting at the main() function



8. Enable the line numbers in the source code. The line numbers can be enabled any time, even during the debug session in Tools > Options > Editor: Show line numbers and clicking OK.



#### Figure 7 Show line numbers

## 2. Debug basics and Live Watch

- 1. Open the View > Disassembly window if it's not already open. Single-step through the code, both on the C level and the assembler level.
- 2. Use the Step into, Step over, Step out, Restart Debugger, Go, and Break functionality function buttons.



#### Figure 8 Debug functions

- 3. **Open** the **View** > **Register** window and see how the **different register changes** values as you go through the code.
- 4. **Open** the **View** > **Call Stack** window and follow the calls during the single stepping.
- 5. **Break/halt** the application to see the **register** and **call stack** information.
- 6. **Double click** on the blinky\_thread\_entry.c file in the **Project Explorer**. If this is a new **Blinky Project** or you don't see the counter variable, add the below code at mentioned positions in blinky\_thread\_entry.c . A simple counter will be incremented in the **while loop** like the print screen in Figure 11 below.

```
A. int counter = 0; > line 15
```

B. counter++;  $\rightarrow$  line 40, inside the while loop





Figure 9 Increment counter



 Use the option Project > Make & Restart Debugger (shortcut key combination Ctrl+R) or the Make & Restart Debugger icon at the top after adding the changes in the source code. You could also Make the project first and then use Debug and Download button to run the code.



Figure 10 Make and restart debugger

- 8. **Open** the **View > Live Watch** window and write counter.
- 9. Reset the application to return to the main function.
- 10. Press **Go** and see how the counter value changes during runtime. You can monitor the value of variables without halting the application.
- 11. In some cases, you need a shorter update interval for the live watch. The default value is 1000 ms but you can decrease it to 1 ms for example. The update interval can be changed by right-clicking in the Live Watch window and selecting Options > Debugger > Update intervals (milliseconds) > Live watch. Alternatively, the Options window can be opened from the Tools tab: Tools > Options



mation Center for AR	M Renesas Synergy main.c blinky_	thread_entry.c main() + X	Characterity	L Ivo		1		1000-0
	#endif	TX_USER_TRACE_BUFFER_DECLA	Go to Disassen	V Memory	~ 🗈	^	Expression counter <click edi<="" th="" to=""><th>Valu 3</th></click>	Valu 3
	IDE Options			×	TST.V BNE.N	LR, #1 0x2442	<ul> <li>Default Form</li> </ul>	nat
	Common Fonts Key Bindings Language E Editor Messages Project Source Code Control Debugger Stack Register Filter Terminal I/O	When source resolves to multiple function instances         Automatically choose all instances         Source code color in disassembly window         Color         Step into functions         All functions         Functions with source only         Update intervals (milliseconds)         Live watch:       100         Memory window:       100	STL conta Depth: Default internal Decimal	iner expansion 10 ager format	VIDM LDM MSR MOVS STEA B.N CPSIE WFI LDD CMP STR BEQ.N LDR.N MOVS STR B.N MOVS STR B.N MOVS STR DC32	R12!,4 R12!, PSP,F R0,#1 R0,[F LR 00x2450 i R2,[F R2,R2 R2, [F R2, R2 R2, [F R2, R2 R2, [F R1, #2 00x2455 R2, [F R1, #1 00x2400 R0, R0, L1 F	Binary Form Octal Form Decimal For Hexadecimu Char Forma Show As Save to File. Options	aat st mat sl Forma t
	int main (void) { disable	0K	Cancel	Apply         Help           0x247c:         0x1ffe0008           0x2480:         0x40006120           0x2484:         0xe000ed04	DC32 DC32 DC32 DC32 DC32	107374 g_mair 107376 ICSR		

Figure 11 Updating Live Watch interval

12. Break the debugger and reset the application to force **counter** == 0 again.

# 3. Conditional and Data Breakpoints

To get the execution to stop at a certain point once a certain condition is met, insert a conditional breakpoint:

- 1. Find the following expression at line 57/58 in blinky\_thread\_entry.c
   /\* Delay \*/
   tx\_thread\_sleep (delay);
- 2. Right-click and select Toggle Breakpoint (Code).



### Figure 12 Break location



- 3. Right-click again and select Edit Breakpoint.
- 4. In the **Conditions > Expression** box, type **counter > 10**.
- 5. This will stop execution when the value is over 10.

Copy Paste Complete Word Complete Code Parameter Hint Match Brackets Toggle All Folds Insert Template Open Header/Source File Go to Definition Go to Declaration Find All References Find All Calls to Functions	<pre>Edit Breakpoint // There Edit Breakpoint // There Code Break At: CUSers\RafaelTa\Synergy\lab1\ Breakpoint type Override default Software Expression: Counter&gt;10 Conditions Expression: Counter&gt;10 Condition true Ski</pre>	x src\blinky_thread_entry.c).58.1 Edit ssion: p count: 0
Find All Calls from Functions Toggle Breakpoint (Code) Toggle Breakpoint (Log) Toggle Breakpoint (Trace Start) Toggle Breakpoint (Trace Stop) Toggle Breakpoint (Trace Filter) Enable/disable Breakpoint Edit Breakpoint Character Encoding Options 20 CX_UIIE	<pre>IOPORT_Li O Condition changed all board LEDs */ t i = 0; i &lt; leds.led_coun ct.p_api-&gt;pinWrite(leds.p_l , nu_sleep (delay);</pre>	OK Cancel at; i++) eds[i], level);

Figure 13 Stop execution when counter > 10

- 6. **Reset** the application to return to **main** and **force counter** ==  $\mathbf{0}$ .
- 7. **Run** the program again. Notice the **View** > **Live Watch** with the **counter changing**. The breakpoint will only be valid when the **condition is true**.
- 8. It is not shown, but the execution is stopped, the condition is evaluated, and if it is **FALSE** then the execution resumes. The complex breakpoint is intrusive but very powerful.
- 9. The following types of symbols can be used in expressions:
  - A. C/C++ symbols
  - B. Assembler symbols (register names and assembler labels)
  - C. C-SPY macro functions
  - D. C-SPY macro variable
- 10. You can write very complex expressions. These expressions will however need to be evaluated to decide if the expression is true or not, so there is a performance loss.
- 11. The action box can be used to evaluate another expression when the breakpoint triggers.
- 12. Remove the conditional breakpoint at line 58. One way of doing this is to open the **Breakpoints** window (**View** > **Breakpoints**) and remove all existing breakpoints. You can also set new breakpoints in this window.



13. Right-click on the global variable counter in line 15. In the menu, select Set Data Breakpoint for counter.

Debug	~	1	/ at	* * * * * * * * * * * * * * * * * * * *
	-		/********	
Files	22 📑 📩	2	* File Name	: blinky_thread_entry.c
🗆 🗇 lab1 - Debug	~	3	* Description	n : This is a very simple
_		4	******	******
Source Files				Cut
		5		Сору
		6	#include "b	Daste
Diinky_thread_entry	.C	7		ruste
hal_entry.c		8 🗆	/******	Complete Word
Synergy			* Abriof	Complete Code
		9	entier	Description of Line
		10	*	Parameter Hint
		11	* Blinks a	Match Brackets
		12	* Only ref	Toggle All Folds
uniter uniter uniter uniter framework		13	*	Insert Template >
느무 🗀 Output		14	*******	Open Header/Source File
lab1.log		15	int counter	openneadelyboarcerne
l labi.map		17		Go to Definition of 'counter'
		10	vola blinky	Go to Declaration of 'counter'
		17Ę	{	Find All References to 'counter'
Dilliky_thread.ontry.o		18	/* Defi	
binky_thead_entry.o		19	con	Find All Calls to 'counter'
		20	(* Cot	Find All Calls from 'counter'
bsp_common_leds.o		20	/~ Set	Find in Trans
		21	const u	Find in Trace
bsp_group_irg.o		22	/* Calc	Toggle Breakpoint (Code)
🛏 🗋 bsp_init.o		23	const u	Toggle Breakpoint (Log)
📥 🗋 bsp_irq.o		24	/* LED	Toggle Breakpoint (Trace Start)
bsp_leds.o		25	her led	
bsp_locking.o		25	psb_ted	Toggle Breakpoint (Trace Stop)
bsp_qspi.o		26	/* LED	Toggle Breakpoint (Trace Filter)
bsp_register_protection	n.o	27	ioport	Enable/disable Breakpoint
bsp_rom_registers.o		28		Set Data Breakpoint for 'counter'
bsp_sbrk.o		20	/* Cot	
bsp_vbatt.o		29	/ Gel	Set Data Log Breakpoint for 'counter'
common_data.o		30	R_BSP_I	Set Trace Start Breakpoint for 'counter'
		31		Set Trace Stop Breakpoint for 'counter'

Figure 14 Setting Breakpoint for variable

14. In the **Breakpoint** window, **View** > **Breakpoints**, **edit** the data breakpoint.

	23	const uinusz_u ae.
bsp_irq.o	24	Go to Source
	25	
bsp_qspi.o	26	Edit
bsp_register_protection.o	27	Delete e1
	28	Disable
bsp_vbatt.o	29	Enable All mo
	30	Disable All
	<	Delete All
* Breakpoint		New Breakpoint
🛛 🔽 🛛 Data @ counter [size 4] [Read/Wri	te] [0x1FFE10B	- OATFELT
10		
oint		
cak		

Figure 15 Editing a Breakpoint

15. Set the breakpoint Access type to Read and check the Enable Match data and set Value to 0x05 (5d).

Edit Breakpoint Data Break at: counter	Edit
Access type Read/Write Read Write	Size Auto (4) Manual Trigger range Requested:
Match data Pinable Value: 0x00000005 Mask: 0xFFFFFFFF	0x1FFE10B8 - 0x1FFE10BB         Effective:         0x1FFE10B8 - 0x1FFE10BB         Extend to cover requested range
	OK Cancel

Figure 16 Edit Breakpoint range

- 16. Click Go.
- 17. The application will halt when there is a read operation to the variable and the data matches.
- 18. Try additional settings for additional access types like Write and Read/Write.

# 4. C-SPY Macros

- 1. Remove all old **breakpoints**.
- 2. Go back to the expression at line 57/58 in blinky\_thread\_entry.c.
  - a. /\* Delay \*/
  - b. tx\_thread\_sleep (delay);
- 3. Right-click the line and select Toggle Breakpoint (Log), and then right-click again and select Edit Breakpoint.



4. Enable C-Spy macro messages and in the Message box add: data = , counter

Edit Breakpoint X
🕒 Log
Trigger at:
{C:\Users\RafaelTa\Synergy\ab1\src\blinky_thread_entry.c}.58.9 Edit
Message: C-Spy macro "message" style
"data = ", counter
Conditions
Conditions Expression:
Conditions Expression:
Conditions Expression:
Conditions Expression: © Condition true © Condition changed
Conditions Expression: O Condition true Condition changed
Conditions Expression: © Condition true © Condition changed

Figure 17 C-SPY message

5. **Run** the program. You will see messages in the **Debug Log** window. If you want, you can set conditions on this Breakpoint as well.

### 5. PRINTF via SWO and Semihosting

 Add a simple printf() function call inside the while loop in the application. For example: printf("Hello Synergy! SSP 1.10\n");

39	while (1)
40 🖨	{
41	counter++;
42	<pre>printf("Hello Synergy! SSP 1.10\n");</pre>
43	/* Determine the next state of the LEDs */
44	<pre>if(IOPORT LEVEL LOW == level)</pre>
. –	. – –

Figure 18 Printf function



13. Use the option **Project** > **Make & Restart Debugger** (shortcut key combination **Ctrl+R**) or the **Make & Restart Debugger** icon on the top after adding the changes in the source code.

File Edit View	Pro	ject Renesas Synergy Debug	) Disassemb	ly J-Link Tools	Wind	ow Help
1 🗅 🔛 🖴 🛛	G	Add Files		- < Q >	\$ ►=	< 📮 > 🔄 🖻 🔳 🗰 🖷 💽 C 😫 📜
Call Stack		Add Group		<b>→</b> ‡ ×	main	blinky_thread_entry x
	E1	Import File List		~	blinky	thread_entry()
🕈 main		Add Project Connection			19	const uint32 t threadx tick ra
Reset_Hand.		Edit Configurations		. <sup>1</sup> . ●	20	/* Set the blink frequency (must 1
<unknown ha<="" th=""><th></th><th></th><th></th><th>✓</th><th>21</th><th><pre>const uint32_t freq_in_hz = 2;</pre></th></unknown>				✓	21	<pre>const uint32_t freq_in_hz = 2;</pre>
	×	Remove			22	/* Calculate the delay in terms of
	+	Create New Project			23	const uint32_t delay = threadx_tic
		create New Project			25	han leds t leds.
	0	Add Existing Project			26	/* LED state variable */
	æ	Ontions	∆lt+F7	u_e	27	ioport level t level = IOPORT_LEVE
	T		ARTIT		28	
		Version Control System	+		29	/* Get LED information for this bo
					30	<pre>R_BSP_LedsGet(&amp;leds);</pre>
	•	Make	F7		31	
	Ð	Compile	Ctrl+F7		32	/* If this board has no leds then
	0	Rebuild All			34	$\square$
	۸	Clean			35	while(1); // There are no le
	5				36	- }
	•	Batch build	F8		37	
		C-STAT Static Analysis	•		38	while (1)
					39 [	
	8	Stop Build	Ctrl+Break		40	counter++;
	-		Chillin D		41	/* Determine the next state of
	•	Download and Debug	Ctri+D		42	L
<		Debug without Downloading			<	
Debug Log	۲	Attach to Running Target				
Log	G	Make & Restart Debugger	Ctrl+R			
Mon Jan 22, 2	C	Restart Debugger	Ctrl+Shift+R	lease set 'MOS	C_CLOC	CK' macro parameter to the actual XTAL/MOSC1

Figure 19 Make & Restart Debugger

- 3. Make sure the **Terminal I/O** window is open by selecting **View** > **Terminal I/O**.
- 4. Run the program, and observe how much time it takes to print the line till the end.
- 5. Exit the debugger.



6. Open the **Project options** by **right clicking** the **Project** in the **project explorer** window or from **Tab Project> Options** and change **General Options** > **Library Configuration** > **stdout/stderr** to **Via SWO**.

Options for node "lab1"	X
Category: General Options Static Analysis Runtime Checking C/C++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator Angel CMSIS DAP GDB Server IAR ROM-monitor I-jet/JTAGjet J-Link/J-Trace TI Stellaris Macraigor	Target Output       Library Configuration       Library Options       MISRA-C:200 ()         Library:       Description:         Normal       Use the normal configuration of the C/C++ nurtime library. No locale interface, C locale, no file descriptor support, no multibytes in printf and scarf, and no hex floats in stitod.         Configuration file:       STOOLKIT_DIR\$\INC\c\DLb_Config_Normal.h         Enable thread support in library       CMSIS         Ubrary low-level interface implementation       CMSIS         Wa semihosting       Use CMSIS         O None       \$tdout/stder         Wa semihosting       DSP library
RDI ST-LINK Third-Party Driver TI XDS	OK Cancel

Figure 20 Change stdout/stderr to Via SWO

- 7. **Build** and **download** the program.
- 8. Run the program, and observe how much time it takes to print the line till the end.
- 9. Do you see a difference?

As you can see, SWO can be sent at full speed while semi-hosting needs to halt the core to send data, and then continue execution.

### 6. ITM Events

To see **ITM event logging** to **measure time**, complete the following steps:

 Add an ITM event in the while loop when the IOPORT level is high in line 48 from blinky\_thread\_entry.c, and include the needed header file #include <arm\_itm.h> in line 7 otherwise you will get a build error. You do not need to transfer large amounts of data so select an 8-bit variant or ITM\_EVENT8\_WITH\_PC(1,1);



2. Add a second ITM event ITM\_EVENT8\_WITH\_PC(1,0); when the **IOPORT level** is **low** in line 53. You want to measure the time in between the two ITM events when the LEDs are being toggled.

40	while (1)
41 🛱	{
42	counter++;
43	<pre>printf("Hello Synergy! SSP 1.10\n");</pre>
44	/* Determine the next state of the LEDs */
45	<pre>if(IOPORT LEVEL LOW == level)</pre>
46	{
47	<u>level = TOPORT LEVEL HIGH;</u>
48	ITM EVENT8 WITH PC(1,1);
49 -	}
50	else
51 🖨	{
52	level = IOPORT LEVEL LOW;
53	ITM EVENT8 WITH PC(1,0);
54 -	}
55	

Figure 21 Setting ITM events

- 3. Build and download.
- 4. Select **Timeline** in the **J-Link** menu.
- 5. In the Timeline window, point within the Event log field, right-click and select Enable. See the following Figure.



Figure 22 Enable event logging

6. Run the program for a few seconds until you have some events in the timeline.



7. Finding the events in the Timeline window, you can measure how much time you have spent in the critical section. Note that you probably need to zoom in or out to find the events more easily. By right-clicking the ITM log field you can zoom the time line or just use the short keys plus (+) and minus (-).

x Debug Log	Zoom Events Enable Clear ITM1: Size Style Show Numerical Values Hexadecimal Go to Source Select Graphs Time Axis Unit Profile Selection	<ul> <li>Zoom to Selection Enter</li> <li>Zoom In +</li> <li>Zoom Out -</li> <li>10 ns</li> <li>100 ns</li> <li>1 us</li> <li>10 us</li> <li>100 us</li> <li>1 ms</li> <li>100 ms</li> <li>1 s</li> <li>1 0 s</li> <li>1 m</li> <li>10 m</li> <li>1 b</li> </ul>	F00D, PID: 001BB101 TSG performed ramming performed for 1 range (16384 bytes) needed: 0.314s (Prepare: 0.090s, Compare: 0.005s, Erase ytes/sec) Ta\Synergy\lab1\Debug\Exe\lab1.out performed
		·····	

Figure 23 Zooming the Timeline



- 8. To measure the time between events, find the first event in the timeline, **press shift** and then **right arrow** to step to **next event**. Hover the mouse cursor above the marked field to see the time between events.
- 9. The source code navigation is also straightforward with the ITM event. Just **double-click** the event and you will land in the correct line in the source code.



Figure 24 Finding the source from the timeline

14. Using the same method, can you measure how long the tx\_thread\_sleep (delay); is really taking? Hint: add the ITM event in lines 60 and 62.





Figure 25 Timing the tx\_thread\_sleep delay

# 7. Interrupt Logging

To use the **interrupt logging** capabilities, complete the following steps:

- 1. Remove all breakpoints.
- 2. Build and download.
- 3. Open the **Timeline**, **Interrupt log** and **Interrupt Log Summary** windows from the **J-Link** menu.

<ul> <li>□ 2 日 3 ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●</li></ul>		Watchpoints Vector Catch	* 0
ETT1 SWO		Disable Interrupts When Stepping	
Workspace ×	IAR Information Cente	Leave Target Running	\$7G2
Debug 🗸 🗸	30	ETM Trace Settings	erna
Files 🗠	31	ETM Trace Save	ir (vo:
🗆 🗊 test - Debug	33	ETM Trace	d * ;
He Synergy	34 🖂	Function Trace	- 0.
	36	SWO Configuration	fal
│	37 38 39 40	SWO Trace Window Settings SWO Trace Save SWO Trace	te su )_com
- Output	41 42	Interrupt Log	
	43 = 44 45 - 46	Interrupt Log Summary Data Log Data Log Summary	IX_IX_IX_IX_IX_IX_IX_IX_IX_IX_IX_IX_IX_I
	48	Event Log	(fir
	49	Power Log Setup	
	51 🖓	Power Log	
	53	Timeline	
	54 55	Function Profiler	
	56	Breakpoint Usage	

Figure 26 Interrupt Log Summary windows, J-Link menu



4. In the **Timeline** window, enable the **Interrupt Log**.



Figure 27 Timeline window, Interrupt Log

- 5. Run (Go) the application.
- 6. Now the **Timeline** window will show every interrupt.
- 7. By selecting and zooming, verify how many cycles it takes for the SysTick.

De	ata Log									
P	PendSV									
S S	SysTick					(	.64 us			
Ca	dl Stack									
Pov	wer Loa									
	Events									
2	<u>+</u>									
le l	2.0599955s	2.05	99960s	2.0599965s	2.0599970s	2.0599975s	2.0599980s	2.0599985s	2.0599990s	2.05
E <										
×		Time	Interrupt			Status	Progr	am Cou	Execution	n Time
	5s 879991 7	7 us	SysTick			Leave			0.64	4 us
	5s 899991.0	9 us	SvsTick			Enter				
	5s 899991.7	3 us	SysTick			Leave			0.64	4 us
	5s 919991.0	6 us	SysTick			Enter				
	5s 919991.7	0 us	SysTick			Leave			0.64	4 us
	5s 939991.0	3 us	SysTick			Enter				
2	5s 939991.6	7 us	SysTick			Leave			0.64	4 us
Ē	5s 959990.9	9 us	SysTick			Enter				
		~	0 7 1						~ ~ ~	

Figure 28 SysTick timing

# 8. Attach to running target

To connect to a target system without resetting or in other way interfering with the program that is running on the system already, complete the following steps:

- 1. Build, download and run the application (Go). You'll see the LEDs blinking, showing that the application is running.
- 2. View the variable **counter** in the **View** > **LiveWatch** window.



3. Select the **Leave Target Running** from the **J-link** in the menu bar as shown below.

×	lab1 - IAR Embedde	ed Workbench IDE - ARM 7.71.2
<u>File Edit View Project Renesas Synergy Debug</u> Disassembly	J-Link ThreadX Tools Window Help	
🗅 🚔 🖬 🞒 👗 🛍 💼 🗠 억	Watchpoints	> ♥ ●   Di 😳 Ni 👂 🛃 👉 🐠   🗁 -   😫   当 み よ 当 ;
ETM <mark>SWO</mark>	Vector Catch	
Workspace × blinky thread entry.c	Disable Interrupts When Stepping	main() 👻 🗙 Disassembly
Debug V 35	<ul> <li>Leave Target Running</li> </ul>	Go to
Files 8: 26: 36 37	ETM Trace Settings	Disas
In ab1 - Debug 38 In the synergy 39	ETM Trace Save FTM Trace	ommon init is done before threads start run initialized_semaphore, "SSP Common Init Sem
40 Source Files 40	Function Trace	main:
	SWO Configuration SWO Trace Window Settings	ENABLE_TRACE TRACE;
45 -	SWO Trace Save	
47	SWO Trace	<pre>pused_memory);</pre>

Figure 29 Leave Target Running Option

4. Exit the debugger. Note that it will not stop the execution.



Figure 30 Exit the Debugger

5. Note that the LEDs are still running.



- File Edit View Project Renesas Synergy J-Link Tools Window Help Add Files... 11 🚹 🔛 🗗 - < Q Add Group... Workspace read\_entry Import File List... Debug Add Project Connection... g\_ssp\_com Files while (g\_: Edit Configurations... 🖂 🌒 lab1 - Deb ł 🔁 🧭 Synergy × Remove err = 🕀 🛋 Source 🕀 🛑 src | D Create New Project... if (T) - 🕀 🔳 s 7 ł Add Existing Project... 🕀 🖸 🕀 t: -🕀 🖸 h 🌣 Options... Alt+F7 -🗉 🗐 syne 1 Version Control System • -🕀 🛑 Output g ssp Make F7 } } Compile Ctrl+F7 } Rebuild All int main(void) Clean { P Batch build... F8 disable irq tx kernel ente C-STAT Static Analysis • return 0; Ctrl+Break Stop Build } Download and Debug Ctrl+D lab1 ۲ Debug without Downloading D Attach to Running Target Build Make & Restart Debugger С CtrI+R Messages Restart Debugger Ctrl+Shift+R C Building confi
- 6. Select the **Attach to running target** from **Project** in the menu bar.

Figure 31 Attach to running target

- 7. Observe that the **counter variable** has progressed further since the target was not reset. The target is running, but when you press **Stop**, the debugger has full control so the running program stops. You can visualize all the variables and registers when it's more convenient for you. This features is useful when investigating some crash of your application.
- 8. Stop the debugger. All features in Options > Debugger > Download should be disabled.



# 9. ThreadX RTOS plugin

To use the built-in **RTOS awareness** plugin for ThreadX in the ISDE, complete the following steps:

1. Enable the RTOS awareness plugin for ThreadX in Options > Debugger > Plugins. Right click on the project in the project explorer to get the Options menu.

Category:		Factory Setting	s
General Options Static Analysis Runtime Checking C/C++ Compiler	Setup Downlos	ad Images Extra Options Multicore Plugins	
Assembler	Select plugins t	to load:	
Custom Build Build Actions Linker	MQX FreeRTOS RTXC Quad	and OpenRTOS dros	
Simulator	ThreadX		
Angel CMSIS DAP		or versions 2.86 and earlier	
GDB Server IAR ROM-monitor	Description:	RTOS awareness for ThreadX.	
I-jet/JTAGjet	Location:	C:\Embedded Workbench 7.2 EWARM RS_2\arm\plugins\	
TI Stellaris	Originator:	IAR Systems	
Macraigor PE micro	Version:	7.40.5.11235	
Third-Party Driver			

Figure 32 ThreadX awareness plugin

2. Start the debugger again with **Ctrl+D** or the **Download** and **Debug** button.



3. Notice that there is a new ThreadX menu available in your debug session.

💦 lab1 - IAR Embed	ded Workbench IDE		5						
File Edit View F	Project Debug Disassembly J-Link	ThreadX	Tools Wind	dow Help					
D 🚔 🖬 🕼 🛛	Thread List		~ ~	🍾 🍾 ً⊻ 🔄 🖻 🐢 🛷 ∢	8	🔊 🗇 🕼 😲	P 🕅 🗩 🖉	🕁 🔄	
5-112	Message Queues								
ETM SWO	Semaphores								
Workspace	Mutexes		ad.c main.c	main() •	• X	Disassembly			
Debug	Byte Pools					Go to		~	Memory
Files	Block Pools			<pre>void g_hal</pre>		Disasse	embl∨		
🗆 🗇 lab1 - D	Timers						,	int :	main(vo
— 🖓 🗇 Syner	Event Flag Groups			#if define	d			{	
└── 🔁 Sou	Execution Profile			#define WE.	A a	main:	0	01-5.04	
	Communication Performance Metric			#elif defi	n		UX3462:	UXD58	, disal
	Memory Performance Metrics			#define WE	A		0x3464:	0xb673	2
	Synchronization Performance Metric			#endif				ł	tx_kerr
	Thread Performance Metrics						0x3466:	Oxf7f:	f Oxffl
	Timer Performance Metrics			#1Idel TX_	u		0x346a:	0x200	)
	Isic	18 -	J	#endif			0x346c:	0xbd03	2
		19					0x346e:	0x000	כ
	l 🔄 driver	20		<pre>void tx_ap</pre>	F	_tx_tł	nread_ini 0v3470	tialize N⊽b510	∋: า
		01					0A3470.	OVDOI	,

Figure 33 ThreadX menu

4. **Open** the **ThreadX** > **Thread List** window.

🞉 lab1 - IAR Embe	dded Workbench IDE
File Edit View	Project Debug Disassembly J-Link ThreadX
🗅 🛩 🖬 🞒	Thread List
5-182	Message Queues
ETM <mark>SWO</mark>	Semaphores
Workspace	Mutexes
Debug	Byte Pools
	Right Pools

#### Figure 34 Showing ThreadX List window

5. The **ThreadX List** window shows the list of active tasks with all additional information from **stack**, **maximum stack usage**, **state of the thread** and how many times the thread has been called.

8										
<u> </u>										
2										
		No Task								
	0x1ffe1008	Blinky Thread	1	Sleep	21	0x1ff∈2688	0x1ffe2318	0x1ffe2717	1024	144
ID		Name	Priority -	State	Run Count	Stack Ptr	Stack Start	Stack End	Stack Size	Max Stack Usage

Figure 35 ThreadX List window

- 6. If you do not see any task you must leave the application running for a while or at least until the multi-tasking is started in the application.
- 7. Notice the **max stack usage** for the **thread** and the **available stack**. This information is useful to set the size of the thread in your application
- 8. Stop and exit the debugger.



# 10. Stack Analysis during Debug (View > Stack Usage)

1. To enable graphical stack analysis during debugging with the SSP package in the IAR EW for Synergy, you must enable the following extra options for the debugger in Project > Options > Debugger > Extra Options > Use command line options:

proc_stack_main=g_main_stack,g_main_stack+sizeor(g_main_stack)	proc_	stack	_main=g_	main	stack,	g_main_	_stack+sizeof(	g_main	_stack)
--	-------	-------	----------	------	--------	---------	----------------	--------	---------

Options for node "I Category: General Options Static Analysis Runtime Checking C/C++ Compiler Assembler Output Converte Custom Build Build Actions Linker Debugger Simulator Angel CMSIS DAP GDB Server IAR ROM-monit I-jet/JTAGjet J-Link/J-Trace TI Stellaris Macraigor PE micro	Ab4" × Factory Settings Setup Download Images Extra Options Multicore Plugins Setup Download Images Extra Options Multicore Plugins Command line options: (one per line) -proc_stack_main=g_main_stack.g_main_stack+sizeof(g_main_stac or
RDI ST-LINK Third-Party Driv TI XDS	er OK Cancel

Figure 36 Command line options for stack analysis

2. Enable the graphical stack display in Tools > Options > Stack: Enable Graphical Stack display.

Coptions		▶ ㅋ ㅋ @ ♪ ♥   01 第 93 ♪ ♪		
Pies Files Surce Files Cutout Babl Trop Babl Trop Babl Tout	Filename Extensions Configure Viewers Configure Custom Argument Variables Configure Tools	Common Center for Renesas Synergy [ab Linao mon.c]      IDE Options      Common Fonts     Key Bindings     W Enable graphical stack display and stack usage tracking		
		7     Language     50 % stack usage threshold       9     E Editor     Wam when exceeding stack threshold       11     Project       12     Source Code Cont       14     Stack pointer(s) not valid until program reaches:       15     Stack pointer(s) not valid until program reaches:       16     Stack pointer(s) not valid until program reaches:       18     Stack pointer(s) not valid until program reaches:       18     Stack pointer(s) not valid until program reaches:       19     Stack pointer(s) not valid until program reaches:       10     Log       21     Log       22     Log       23     Limit stack display to		
		29 30 31 32 33 34 34 35 35 36 36 37 37 38 38 38 38 38 38 38 38 38 38		





- 3. Start the debugger again with **Ctrl+D** or the **Download and Debug** button.
- $4. \quad \mbox{Once you reach main(), open the Stack window in View } > Stack > Stack 1.$

22 Jah1 - JAR Embedded Workbench IDE							
File Edi View Project Debug Disassembly J-Link Tools Window Help							
Workspace							
Workspace C-STAT	anter for Deneras Supergy light man main c						
	#define WEAK REF ATTRIBUTE						
Breakpoints	<pre>#pragma weak tx_application_define_user = tx_application</pre>						
Disassembly	<pre>#elif defined(_GNUC) #define WFAV DFF ATTPIRUTE</pre>						
Hemory Memory	<pre>#endif</pre>						
L Gut Symbolic Memory							
Register	#ITGET IX_USER_IRACE_BUFFER_DECLARE TX_USER_TRACE_BUFFER_DECLARE;						
Watch	<pre>#endif</pre>						
Locals	<pre>void tx application define internal(void * first unused memory):</pre>						
Statics							
Auto	<pre>void tx_application_define_internal(void * first_unused_memory) </pre>						
Live Watch	/* Does nothing. Default implementation of tx_application_define						
Ouick Watch	SSP_PARAMETER_NOT_USED(first_unused_memory);						
Macros	2 •						
Call Stack	<pre>void tx_application_define_internal(void * first_unused_memory);</pre>						
Stack	<pre>void tx application_define_user(void * first_unused_memory) WEAK_REF_ Stack 1</pre>						
Terminal I/O	Stack 2 application_define(void * first_unused_memory)						
Images	blinky thread create();						
Correc							
Corles	#ifdef TX_USER_ENABLE_TRACE						
Code Coverage	#endif						
C-RUN							
Symbols	g_nat_init();						
Toolbars	<pre>tx_application_define_user(first_unused_memory);</pre>						
✓ Status Bar	j.						
44 int main (void)							

Figure 38 Open Stack window

5. Leave the application running for a while (5 to 10 seconds).



6. Halt the debugger and observe the stack usage in the **Stack** window. Notice the log of the **maximum usage** displayed graphically on the top of the stack window.



Figure 39 Stack window



# Website and Support

Support: <u>https://synergygallery.renesas.com/support</u>

Technical Contact Details:

- America: <u>https://www.renesas.com/en-us/support/contact.html</u>
- Europe: <u>https://www.renesas.com/en-eu/support/contact.html</u>
- Japan: <u>https://www.renesas.com/ja-jp/support/contact.html</u>

All trademarks and registered trademarks are the property of their respective owners.



# **Revision History**

		Description	
Rev.	Date	Page	Summary
Rev.1.00	Sept 23, 2016	-	First revision
Rev.1.01	Jul 4, 2017	All	Updated for SSP v1.3.0
Rev.1.02	Aug 2, 2017	-	Initial release
Rev.1.03	Feb 7, 2018	-	Updated for SSP v1.4.0

#### Notice

- 1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
- Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
- 3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
- 4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
- 5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
  - "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc. Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.

- 6. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
- 7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and malfunction of measures to the final products or systems manufactured by you.
- 8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
- 9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
- 10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
- 11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics
- 12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.
- (Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
- (Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

Refer to "http://www.renesas.com/" for the latest and detailed information

(Rev.4.0-1 November 2017)

http://www.renesas.com



#### SALES OFFICES

**Renesas Electronics Corporation** 

Renesas Electronics America Inc. 1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A. Tel: +1408-432-8888, Fax: +1-408-434-5351 Renesas Electronics Canada Limited 9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3 Tel: +1-905-237-2004 Renesas Electronics Europe Limited Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K Tel: +44-1628-651-700, Fax: +44-1628-651-804 Renesas Electronics Europe GmbH Arcadiastrasse 10, 40472 Düsseldorf, Germany Tel: +49-211-6503-0, Fax: +49-211-6503-1327 Renesas Electronics (China) Co., Ltd. Room 1709 Quantum Plaza, No.27 ZhichunLu, Haidian District, Beijing, 100191 P. R. China Tel: +86-10-8235-1155, Fax: +86-10-8235-7679 Renesas Electronics (Shanghai) Co., Ltd. Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, 200333 P. R. China Tel: +2226-0888, Fax: +86-21-2226-0989

Renesas Electronics Hong Kong Limited Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong Tel: +852-2265-6688, Fax: +852 2886-9022 Renesas Electronics Taiwan Co., Ltd. 13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan Tel: +886-2-8175-9600, Fax: +868 2-8175-9670 Renesas Electronics Singapore Pte. Ltd. 80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949 Tel: +656-213-0200, Fax: +656-213-0300 Renesas Electronics Malaysia Sdn.Bhd. Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia Tel: +60-57955-9300, Fax: +60-37955-9510 Renesas Electronics India Pvt. Ltd. No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India Tel: +91-80-67208700, Fax: +60-37208707 Renesas Electronics Korea Co., Ltd. 17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea Tel: +82-558-3737, Fax: +82-2-558-5338