

Infineon TC275 Getting Started

Hyeongrae Kim

Architecture and Compiler for Embedded System LAB.

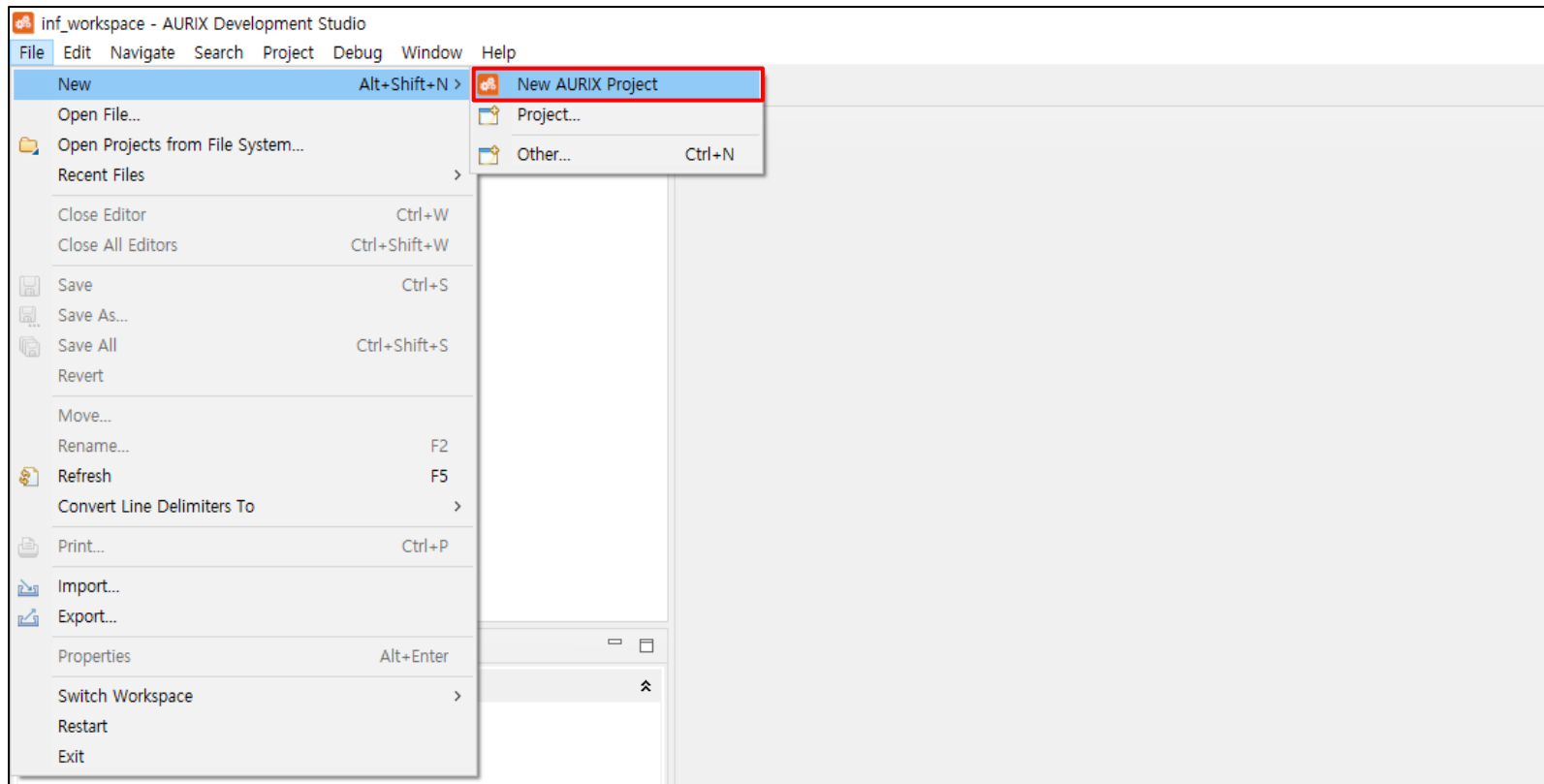
School of Electronics Engineering, KNU, KOREA

2021-12-23



New Project

1. AURIX Development Studio를 실행하고 왼쪽 상단의 '**File – New –New AURIX Project**'를 클릭한다.



New Project

2. Project name을 입력하고, '**Next**'를 클릭한다.

New AURIX Development Studio Project

New AURIX Project

Specify the name and the location of the new project

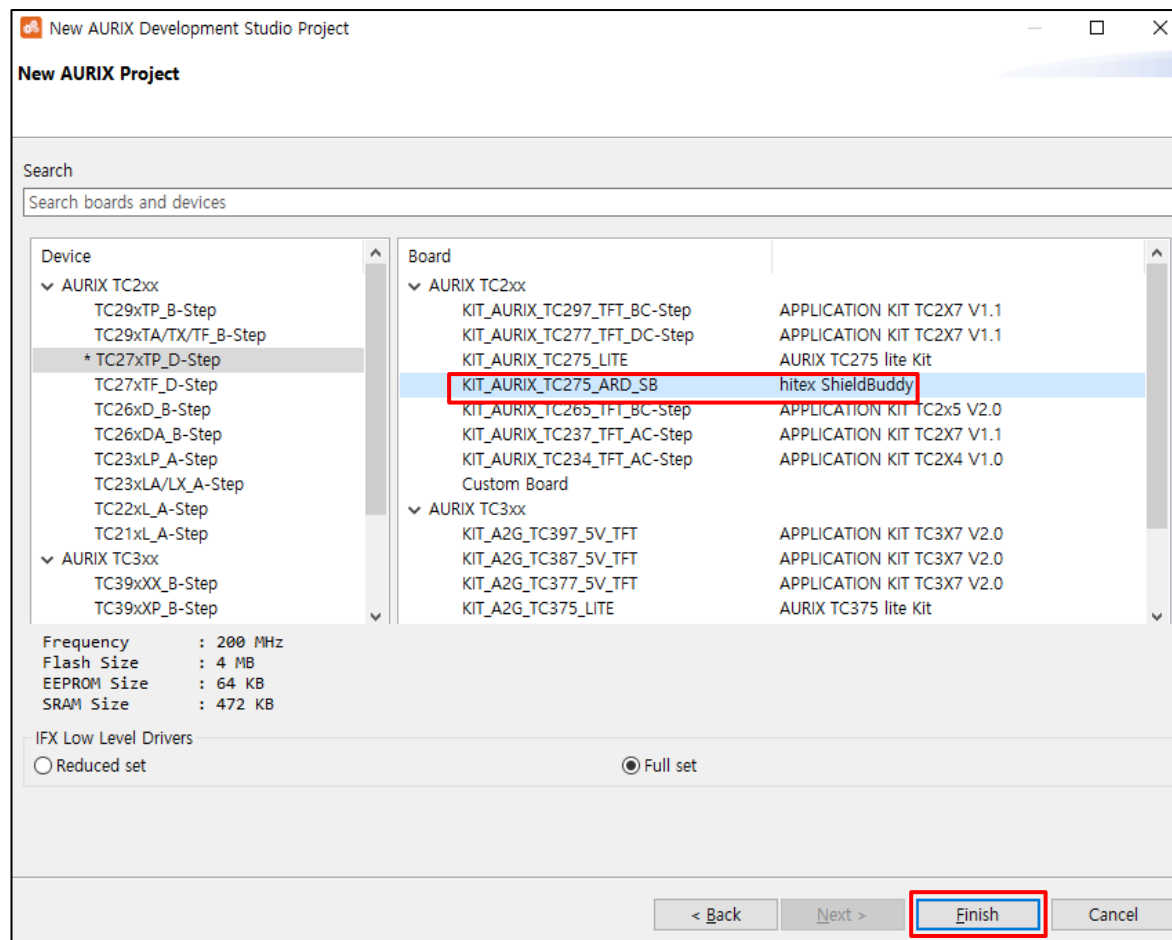
Project name:

☒ Use default location

Location:

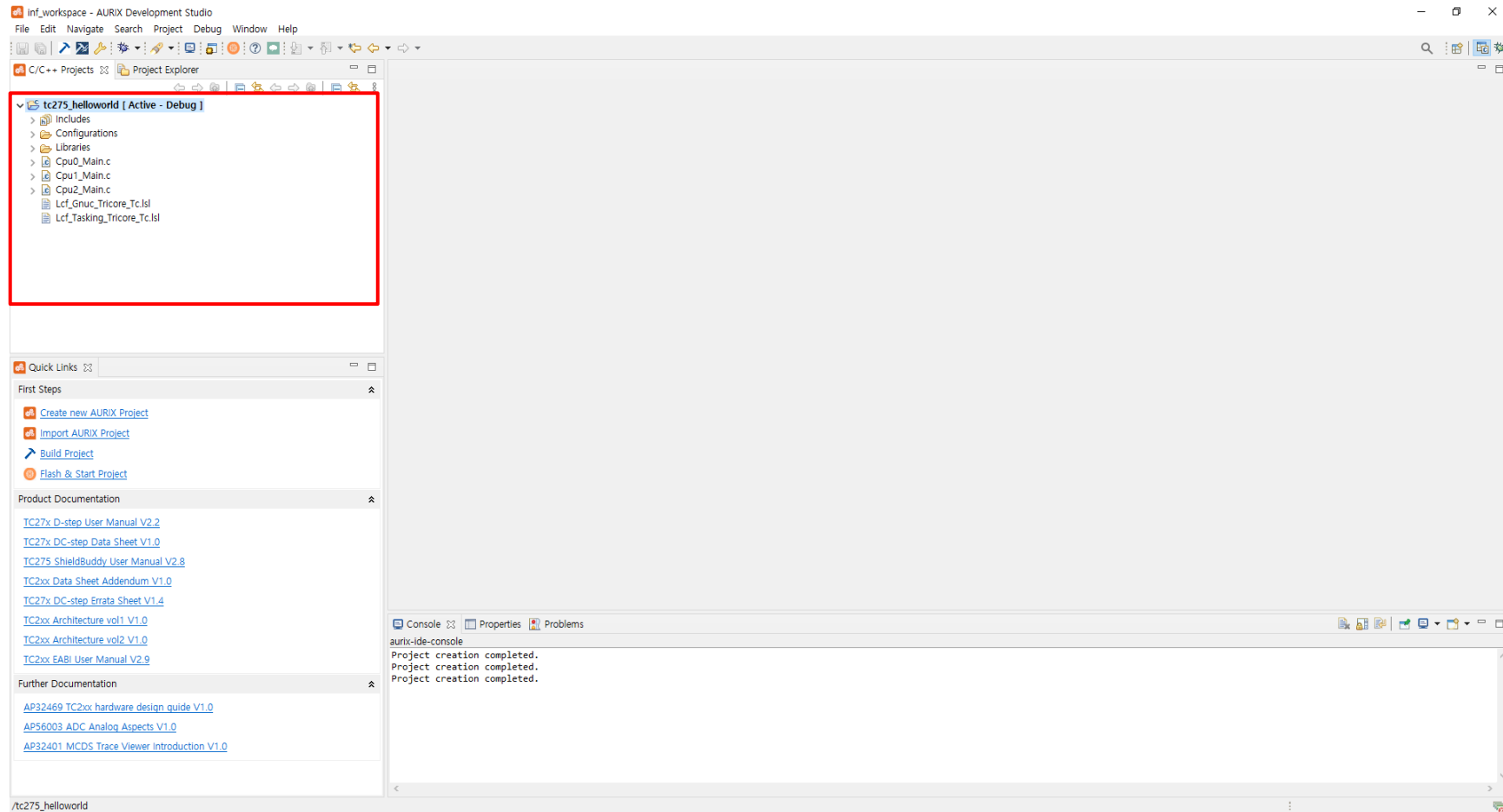
New Project

3. Device에서 '**AURIX TC2xx- TC27XTP_D-Step -KIT_AURIX_TC275_ARD_SB**'를 선택하고, 다른 설정은 그대로 유지한 채 '**Finish**'를 클릭한다.



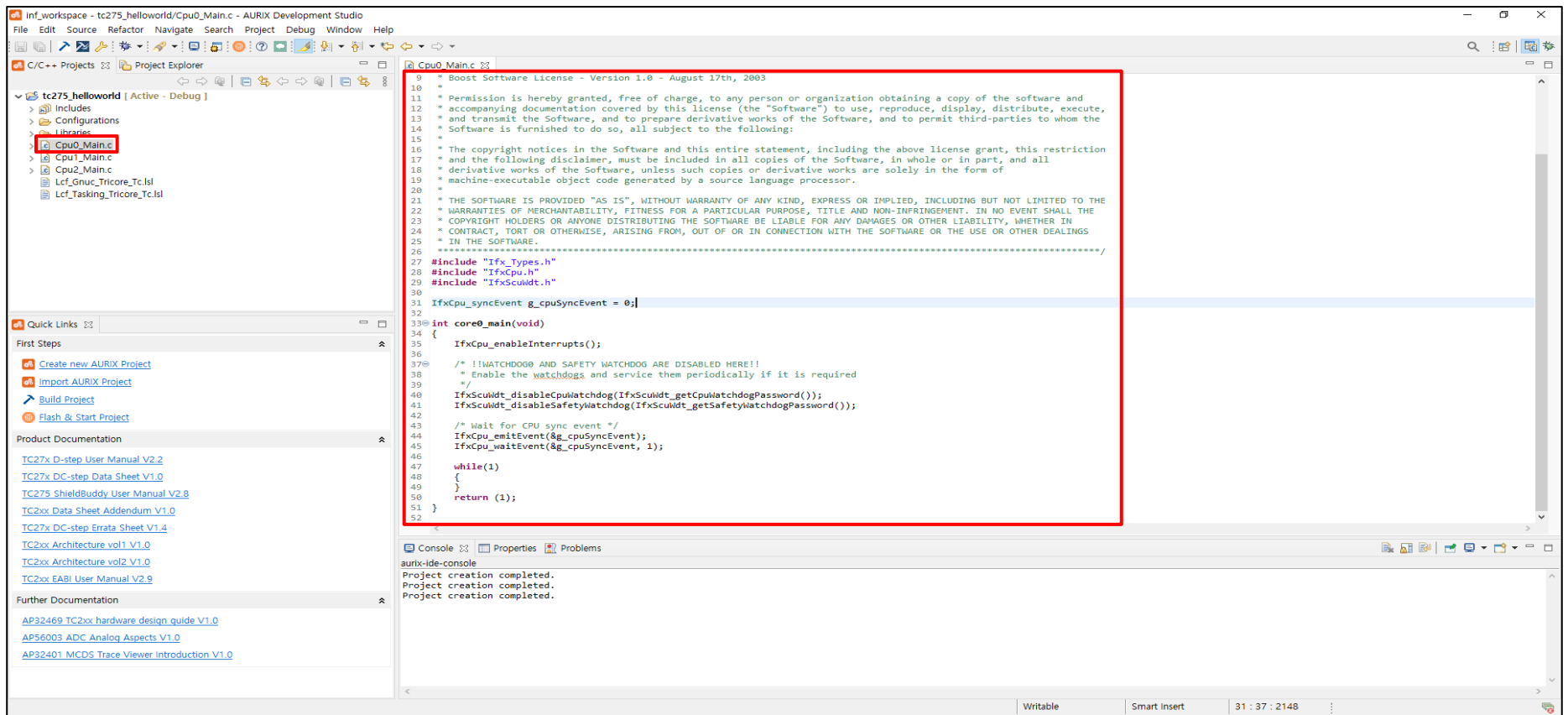
New Project

4. 왼쪽의 Project Explorer 창에서 프로젝트가 생성된 것을 확인한다.



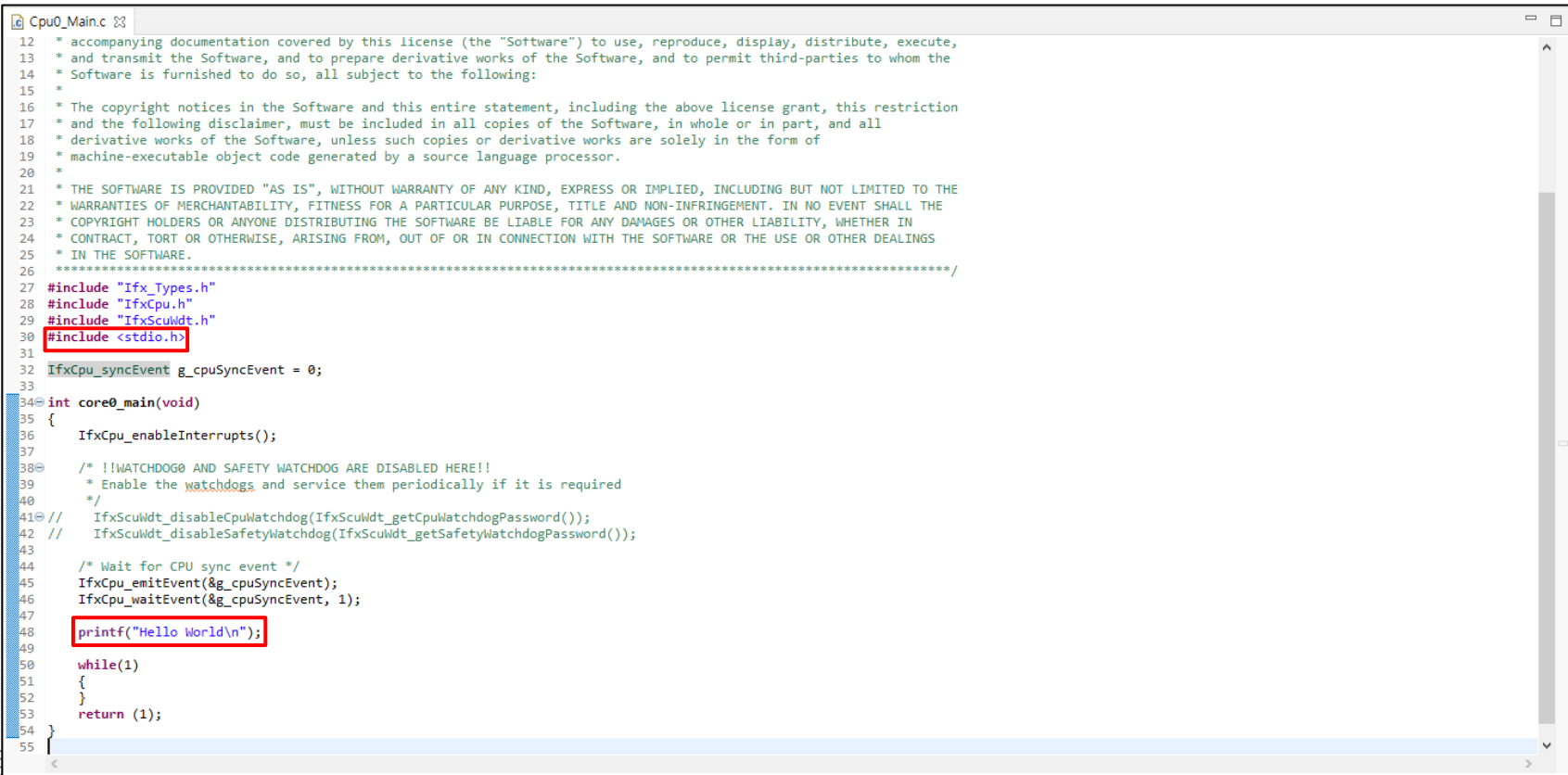
Edit Project

5. 왼쪽의 Project Explorer 창에서 **Project name**으로 생성된 파일인 '**Cpu0_Main.c**' 파일을 더블 클릭하여 활성화한다.



Edit Project

6. 'Cpu0_main.c' 파일은 **core0_main** 함수를 포함하고 있으며 이를 수정하여 동작을 설계한다. (본 실습에서는 멀티코어를 사용하지 않으므로 core1_main과 core2_main 함수는 사용하지 않는다.)



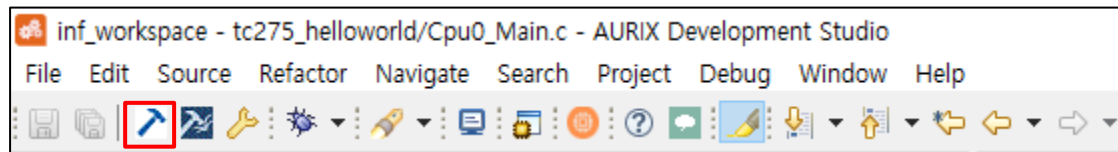
```
12  * accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute,
13  * and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the
14  * Software is furnished to do so, all subject to the following:
15  *
16  * The copyright notices in the Software and this entire statement, including the above license grant, this restriction
17  * and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all
18  * derivative works of the Software, unless such copies or derivative works are solely in the form of
19  * machine-executable object code generated by a source language processor.
20  *
21  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
22  * WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE
23  * COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN
24  * CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
25  * IN THE SOFTWARE.
26  *
27  #include "Ifx_Types.h"
28  #include "IfxCpu.h"
29  #include "IfxScuWdt.h"
30  #include <stdio.h>
31
32  IfxCpu_syncEvent g_cpuSyncEvent = 0;
33
34  int core0_main(void)
35  {
36      IfxCpu_enableInterrupts();
37
38      /* !!WATCHDOG0 AND SAFETY WATCHDOG ARE DISABLED HERE!!
39       * Enable the watchdogs and service them periodically if it is required
40       */
41      // IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword());
42      // IfxScuWdt_disableSafetyWatchdog(IfxScuWdt_getSafetyWatchdogPassword());
43
44      /* Wait for CPU sync event */
45      IfxCpu_emitEvent(&g_cpuSyncEvent);
46      IfxCpu_waitEvent(&g_cpuSyncEvent, 1);
47
48      printf("Hello World\\n");
49
50      while(1)
51      {
52      }
53      return (1);
54  }
55
```



Build

7. 상단의 메뉴에서 '**Build**' 버튼을 클릭하여 Build를 실행한다.

(Build/Debug는 Active Project에 대해 수행되기 때문에 Build를 수행할 Project를 Active Project로 미리 설정해야 한다. 'Project Explorer – 대상 Project'에서 우클릭 한 뒤, 'Set Active Project'를 클릭하여 Active Project로 설정할 수 있다.)

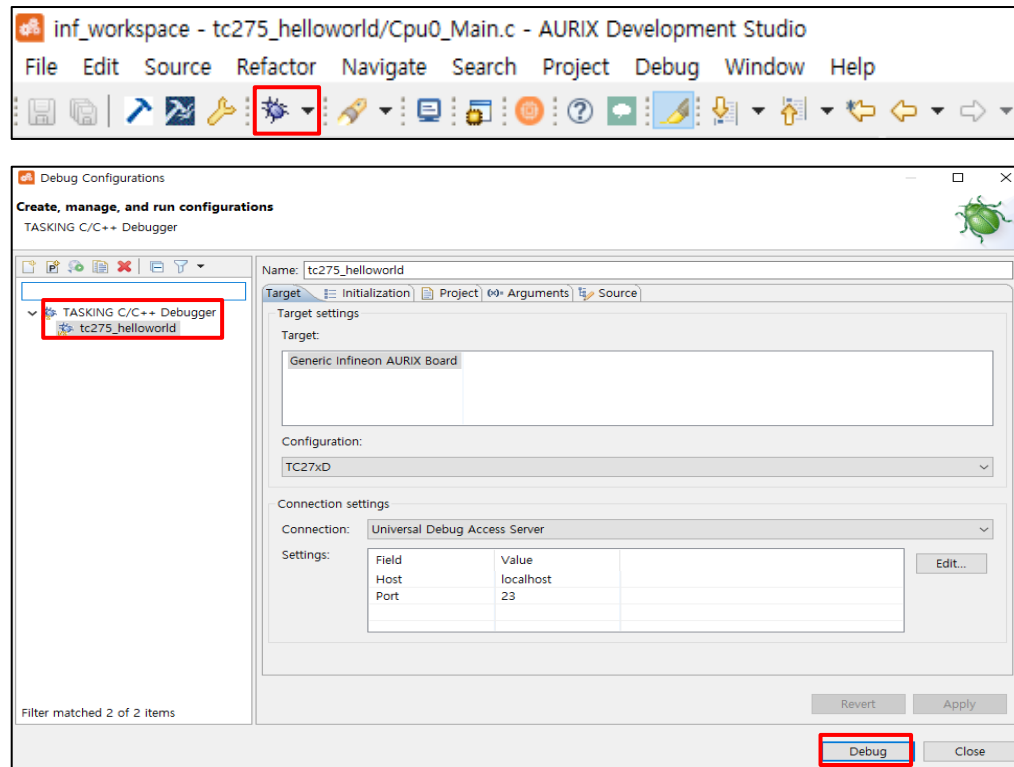


Debug

8. 상단의 메뉴에서 '**Debug**' 버튼을 클릭하여 Debug를 실행한다.

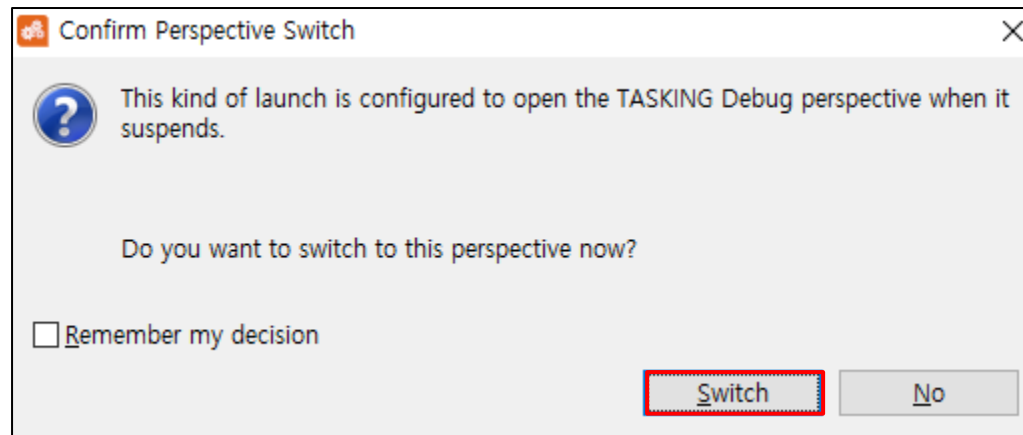
- ✓ 'Debug' 버튼을 처음으로 클릭하면 'Debug Configurations' 창이 활성화된다. 왼쪽의 창에서 '**TASKING C/C++ Debugger – Project name**'을 확인하고 '**Debug**' 버튼을 클릭한다.

(이후에는 Debug가 바로 실행되며 'Debug Configurations' 창을 활성화하기 위해서는 'Debug' 버튼의 오른쪽 화살표를 클릭한 후 'Debug Configurations...' 버튼을 클릭한다.)



Debug

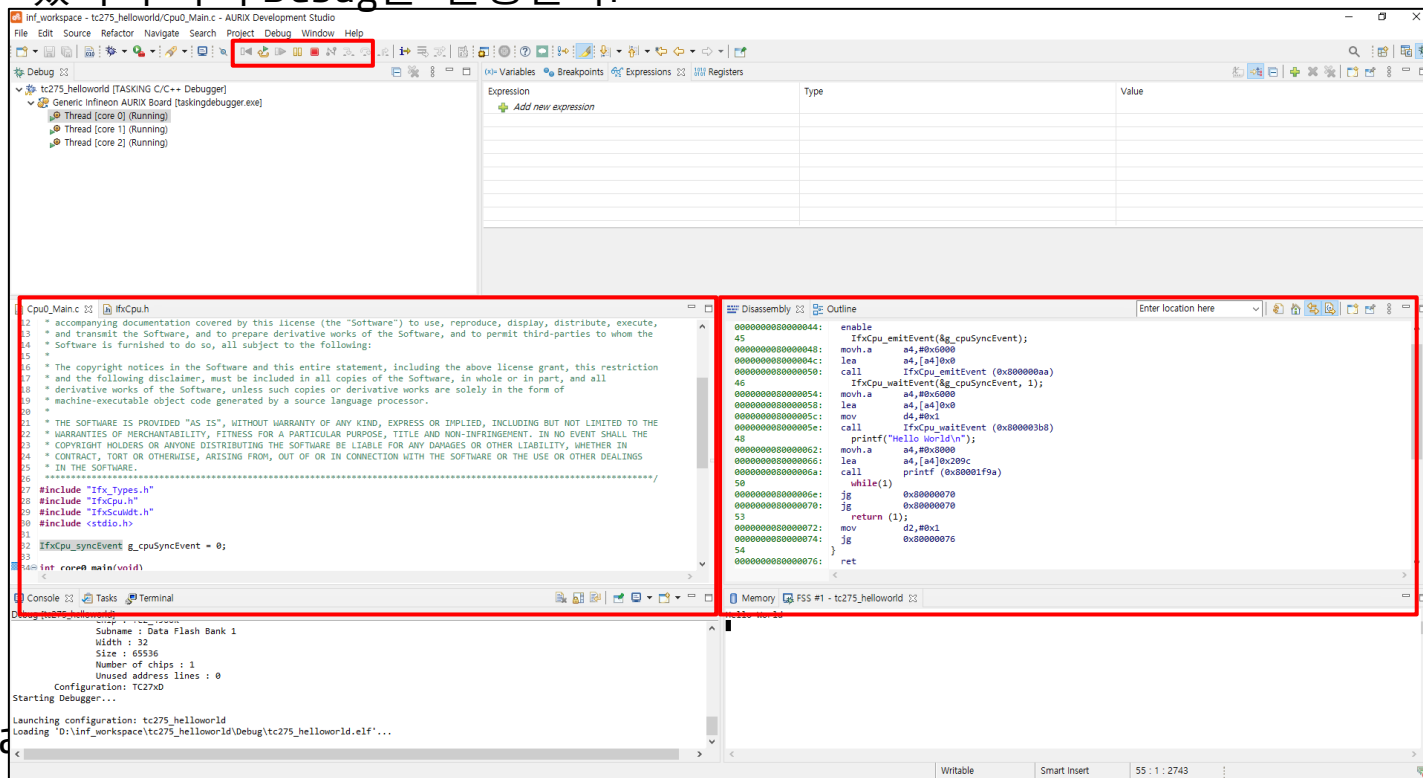
8. 상단의 메뉴에서 '**Debug**' 버튼을 클릭하여 Debug를 실행한다.
 - ✓ 'Confirm Perspective Switch' 창이 뜨면 Switch를 눌러 디버그 창으로 전환한다.



Debug

9. Debug 화면을 확인한다.

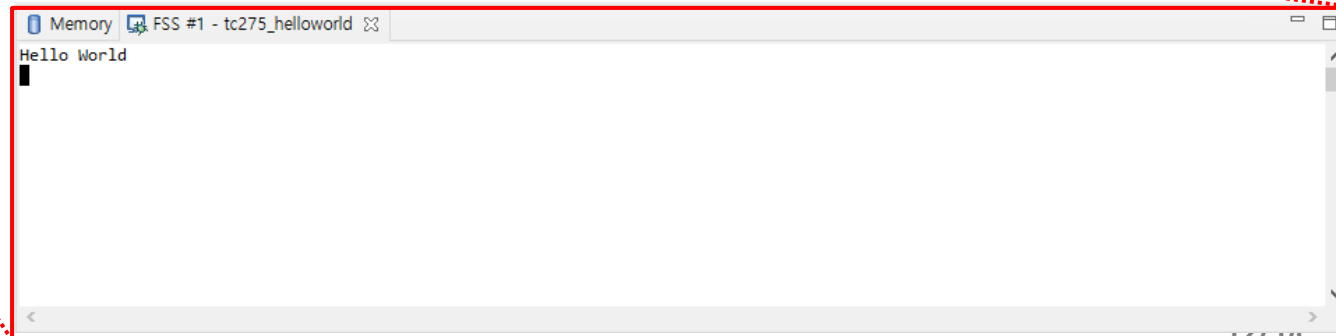
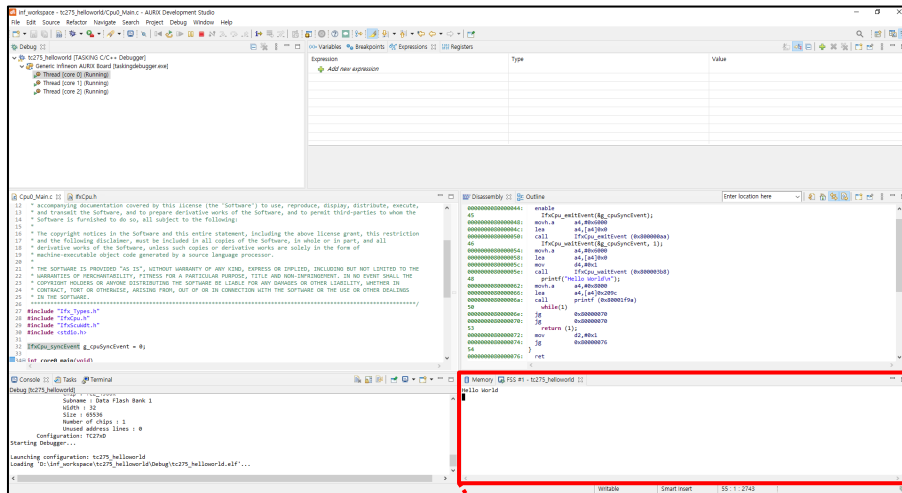
- ✓ 상단의 Restart / Resume / Terminate / Step Into / Step Over 등을 클릭하여 실행을 제어가능
- ✓ 가운데 창을 통해 소스 코드 및 어셈블리 코드를 확인할 수 있다.
- ✓ 만약, 소스 코드 화면에 에러 메시지가 표시되면 'Terminate' 버튼을 클릭하여 Debug를 종료했다가 다시 Debug를 실행한다.



Debug

10. Debug를 통해 실행을 제어하고 결과를 확인한다.

- ✓ 상단의 '**Resume**' 버튼을 클릭하여 동작을 실행시킨다.
- ✓ 동작의 실행 결과로 우측 하단 창 (FSS)에 'Hello world'라는 메시지가 표시되는 것을 확인한다.



Memory Access Speed Test

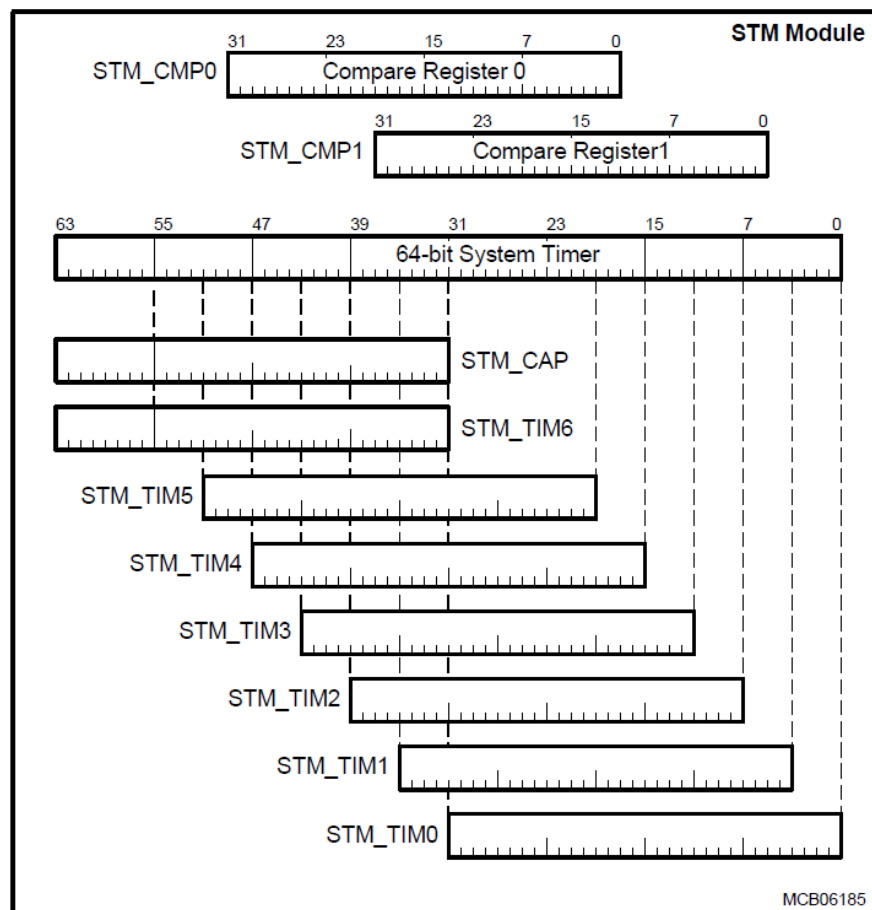


Figure 17-1 General Block Diagram of the STM Module

Table 17-1 Registers Address Space

Module	Base Address	End Address	Note
STM0	F000 0000 _H	F000 00FF _H	STM for CPU0
STM1	F000 0100 _H	F000 01FF _H	STM for CPU1
STM2	F000 0200 _H	F000 02FF _H	STM for CPU2

Table 17-2 Registers Overview - STM Control Registers

Short Name	Description	Offset Addr.	Access Mode		Reset	Description See
			Read	Write		
CLC	Clock Control Register	00 _H	U, SV	SV, E, P	Application ¹⁾	Page 17-8
-		04 _H	BE	BE	-	-
ID	Identification Register	08 _H	U, SV	BE	Application	Page 17-9
-		0C _H	BE	BE	-	-
TIM0	Timer 0 Register	10 _H	U, SV	BE	Application	Page 17-10

User's Manual
STM, V1.11

17-5

V2.2, 2014-12



TC27x D-Step

System Timer (STM)

Table 17-2 Registers Overview - STM Control Registers

Short Name	Description	Offset Addr.	Access Mode		Reset	Description See
			Read	Write		
TIM1	Timer 1 Register	14 _H	U, SV	BE	Application	Page 17-10
TIM2	Timer 2 Register	18 _H	U, SV	BE	Application	Page 17-11
TIM3	Timer 3 Register	1C _H	U, SV	BE	Application	Page 17-11
TIM4	Timer 4 Register	20 _H	U, SV	BE	Application	Page 17-11
TIM5	Timer 5 Register	24 _H	U, SV	BE	Application	Page 17-12
TIM6	Timer 6 Register	28 _H	U, SV	BE	Application	Page 17-12
CAP	Timer Capture Register	2C _H	U, SV	BE	Application	Page 17-13
CMP0	Compare Register 0	30 _H	U, SV	U, SV	Application	Page 17-13

System Timer Test

```
#include "Ifx_Types.h"
#include "IfxCpu.h"
#include "IfxScuWdt.h"
#include <stdio.h>

#define SYSTEM_TIMER_31_0  *(unsigned int *) (0xF0000000+0x10)
#define SYSTEM_TIMER_PERIOD 10 // 100MHz

unsigned int systemtick[2];
unsigned int tick_cnt;
unsigned int delay_time_ns;

IfxCpu_syncEvent g_cpuSyncEvent = 0;

int core0_main(void)
{
    IfxCpu_enableInterrupts();

    /* !!WATCHDOG0 AND SAFETY WATCHDOG ARE DISABLED HERE!!
     * Enable the watchdogs and service them periodically if it is required
     */
    IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword());
    IfxScuWdt_disableSafetyWatchdog(IfxScuWdt_getSafetyWatchdogPassword());

    /* Wait for CPU sync event */
    IfxCpu_emitEvent(&g_cpuSyncEvent);
    IfxCpu_waitEvent(&g_cpuSyncEvent, 1);

    // Delay Check
    systemtick[0] = SYSTEM_TIMER_31_0;
    for( int i=0; i<83; i++)
        systemtick[1] = SYSTEM_TIMER_31_0;

    // Calculate tick cnt
    tick_cnt = systemtick[1] - systemtick[0];

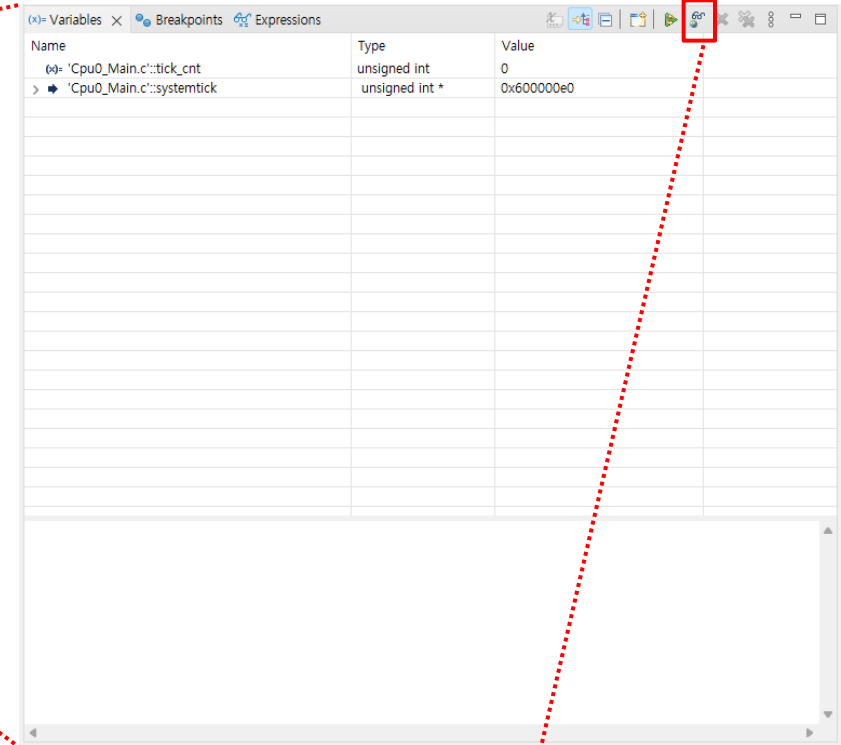
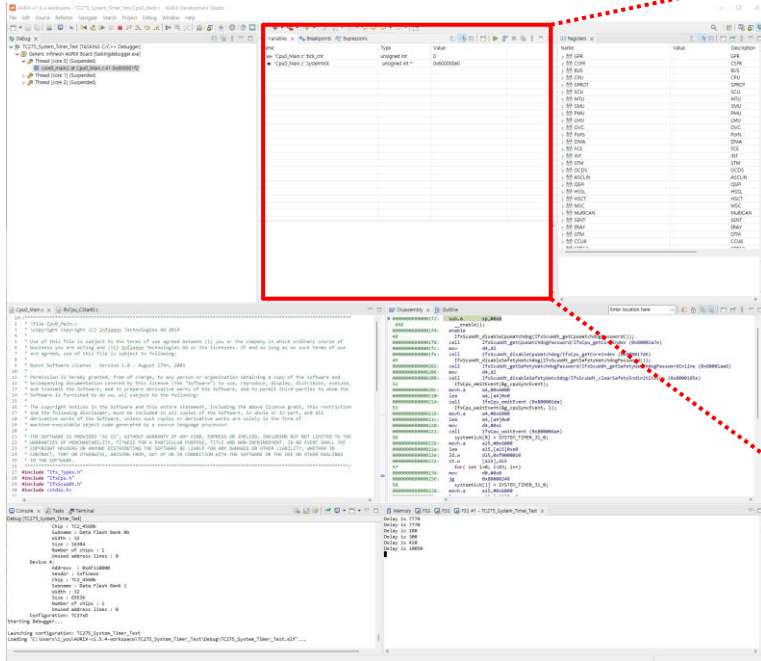
    // Calculate delay time, cnt * 10ns
    delay_time_ns = tick_cnt * SYSTEM_TIMER_PERIOD;

    printf("Delay is %d \n", delay_time_ns);

    while(1)
    {
    }

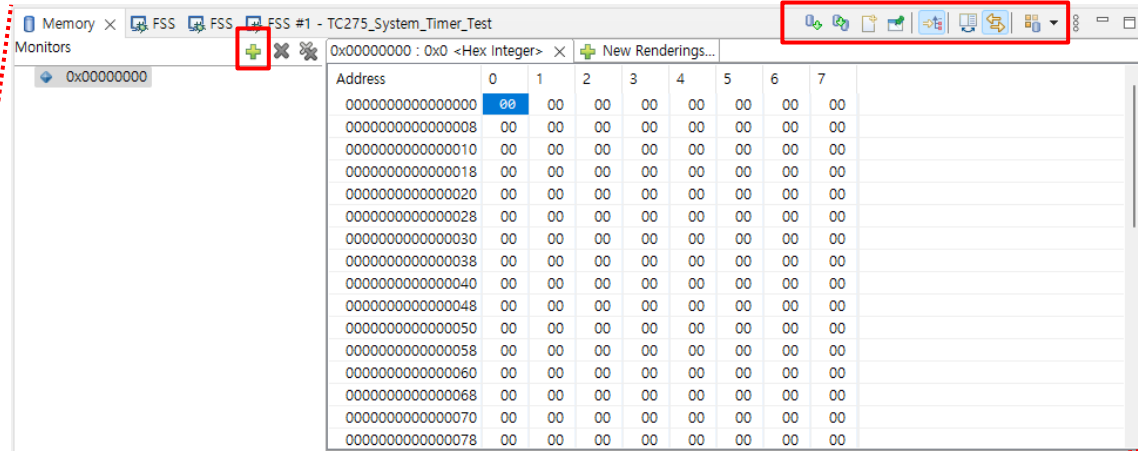
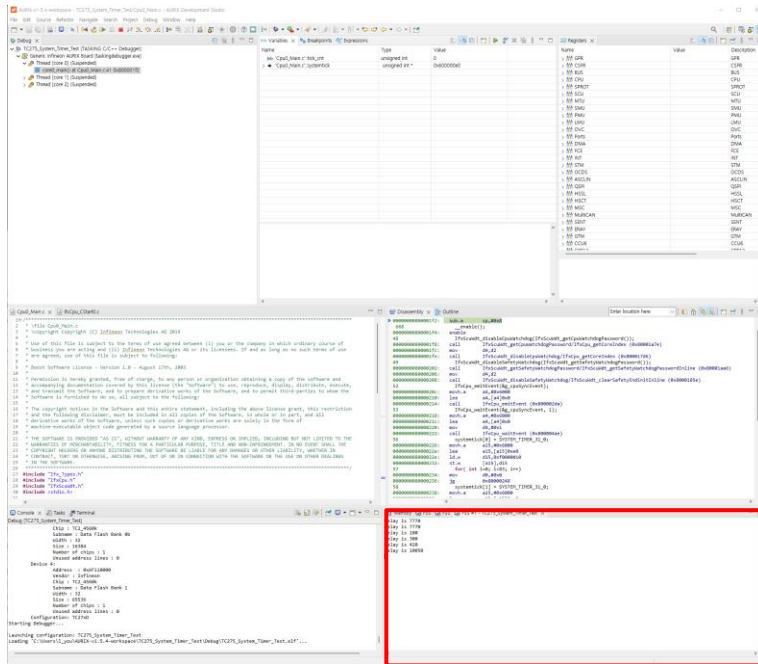
    return (1);
}
```

Variable



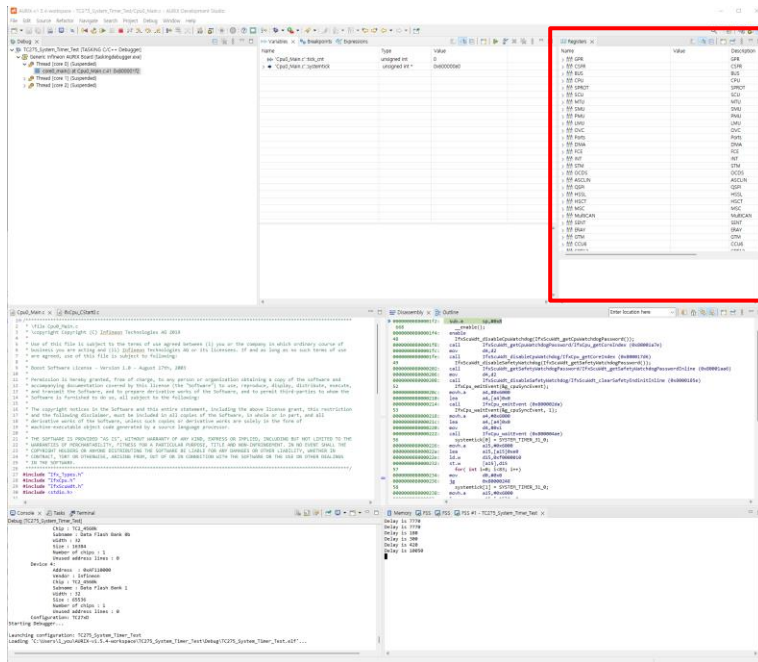
- Variable 창에 Add Variable을 통해서 전역 변수 추가 가능
- 지역변수는 자동으로 표시됨.
- 변수값은 Suspend 상태에서에서만 표시됨.

Memory



- Memory Tab을 클릭하여 사용가능
- +를 클릭해서 확인하고자 하는 Address를 입력해서 Dump 할 수 있다.
- 8-bit 기준으로 설정한 주소를 기준으로 Display 된다.

Registers



Name	Value	Description
> GPR		GPR
> CSFR		CSFR
> BUS		BUS
> CPU		CPU
> SPROT		SPROT
> SCU		SCU
> MTU		MTU
> SMU		SMU
> PMU		PMU
> LMU		LMU
> OVC		OVC
> Ports		Ports
> DMA		DMA
> FCE		FCE
> INT		INT
> STM		STM
> OCDS		OCDS
> ASCLIN		ASCLIN
> QSPI		QSPI
> HSSL		HSSL
> HSCT		HSCT
> MSC		MSC
> MultICAN		MultICAN
> SENT		SENT
> ERAY		ERAY
> GTM		GTM
> CCU6		CCU6
> CRT13		CRT13

- Registers Tab을 이용하여 사용가능
- 각 Peripheral Register 상태를 확인이 가능하다.
- Read only / Write Only를 고려해서 확인을 해야 한다.

Break

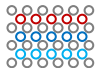
The screenshot shows the AURIX Development Studio interface. In the top-left pane, the 'Breakpoints' menu is open, and the 'Toggle Line Breakpoint' option is selected. In the bottom-left pane, the source code for 'Cpu0_Main.c' is displayed, and line 75 is highlighted with a green background, indicating where a breakpoint has been set. The right pane shows the 'Variables' and 'Expressions' tabs, with 'Cpu0_Main.c [line: 75]' selected. The code in the right pane is as follows:

```

52
53 /* Wait for CPU sync event */
54 IfxCpu_emitEvent(&g_cpuSyncEvent);
55 IfxCpu_waitEvent(&g_cpuSyncEvent, 1);
56
57 // Delay Check
58 systemtick[0] = SYSTEM_TIMER_31_0;
59 for( int i=0; i<83; i++)
60 systemtick[1] = SYSTEM_TIMER_31_0;
61
62 // Calculate tick cnt
63 tick_cnt = systemtick[1] - systemtick[0];
64
65 // Calculate delay time, cnt * 10ns
66 delay_time_ns = tick_cnt * SYSTEM_TIMER_PERIOD;
67
68 printf("Delay is %d \n", delay_time_ns);
69
70 while(1)
71 {
72     i++;
73     if(i == 100)
74     {
75         i = 0;
76     }
77 }
78
79 return (1);
80 }
81

```

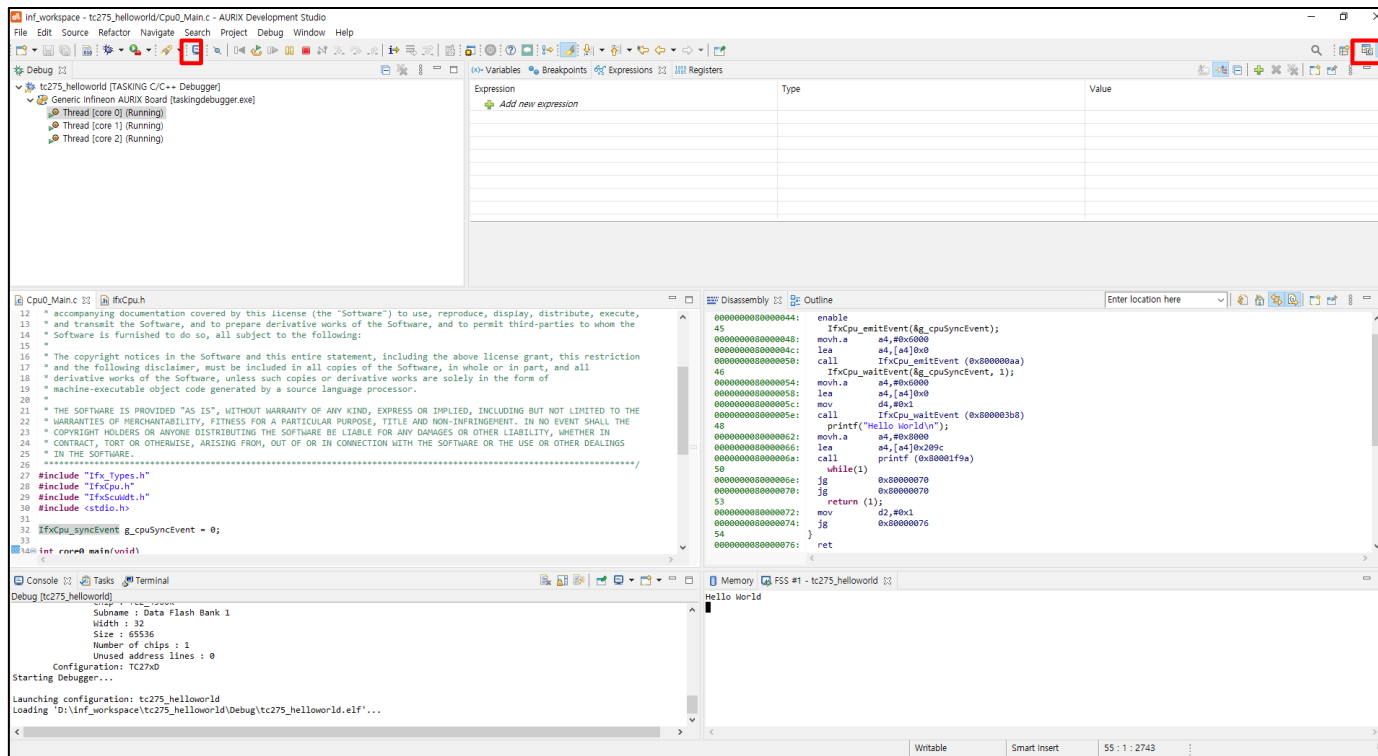
- Code tab에서 원하는 line을 클릭해서 Debugger menu를 클릭해서 break를 선택
- 또는 Code line을 Click하면 Break가 가능한 영역은 파란 띠가 나타나고 파란 라인에 Double Click 하면 Break를 설정 가능하다.
- Breakpoints tab을 통해서 설정된 break를 확인가능



Debug

11. Debug를 종료한다.

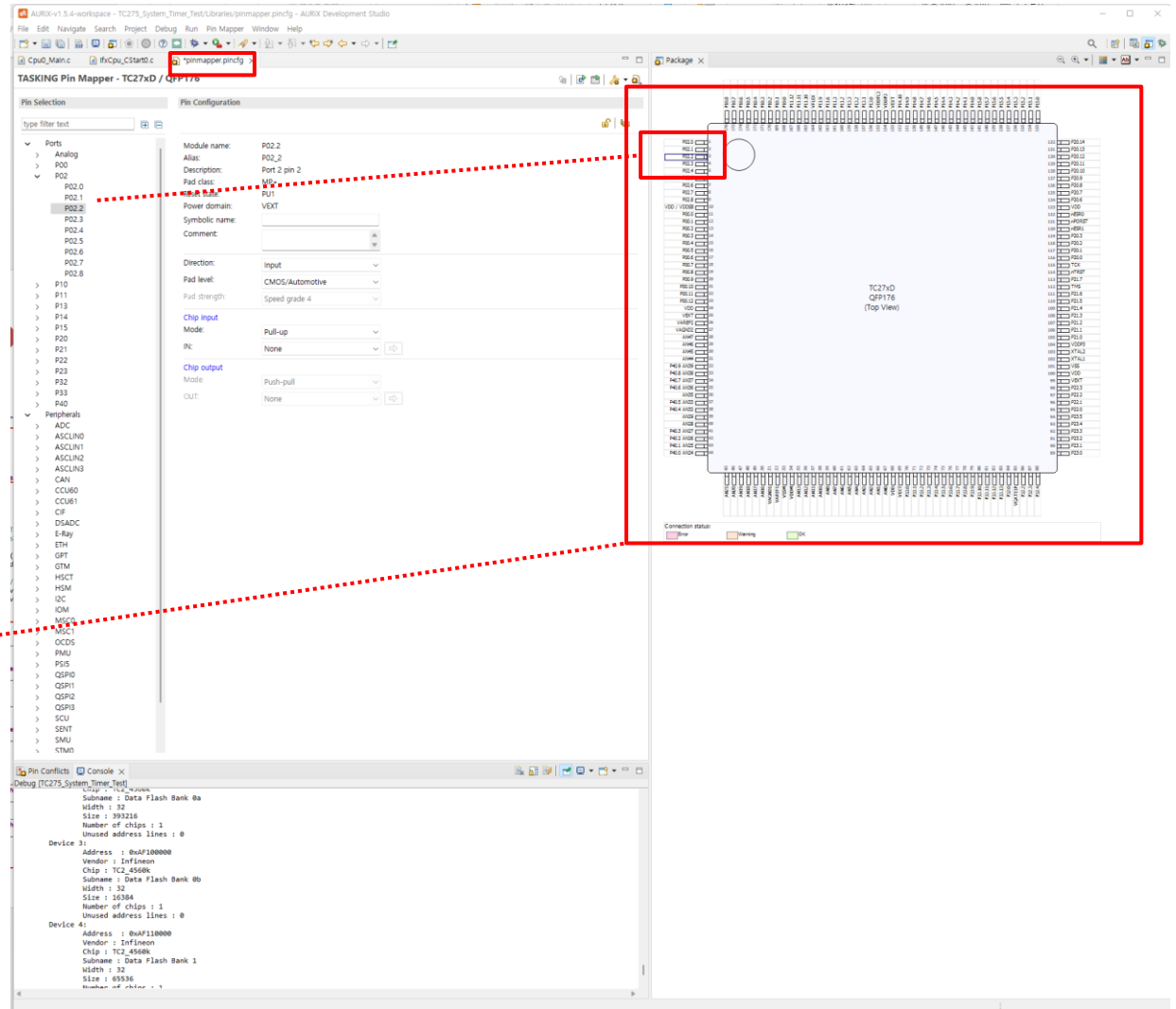
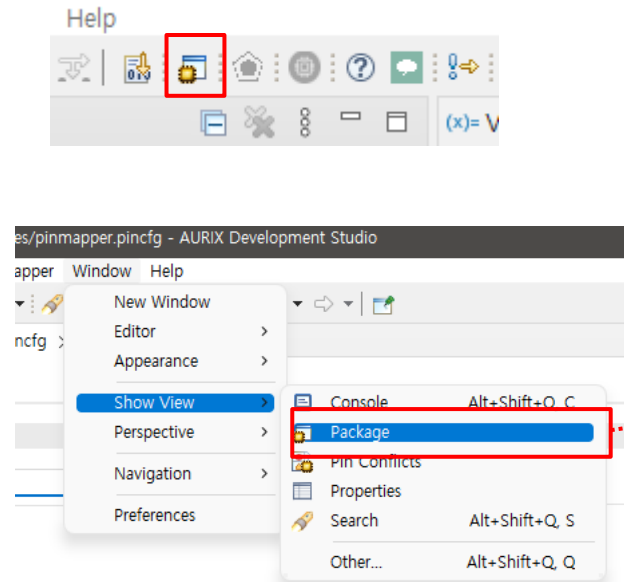
- ✓ 상단의 '**Terminate**' 버튼을 클릭하여 Debug를 종료한다.
(Debug 종료 시, 꼭 '**Terminate**' 버튼을 클릭하여 정상적으로 종료한다.)
- ✓ Debug 종료 후, 소스코드 편집 화면으로 돌아가기 위해서는 우측 상단의 '**C/C++**'을 클릭한다.



단축키

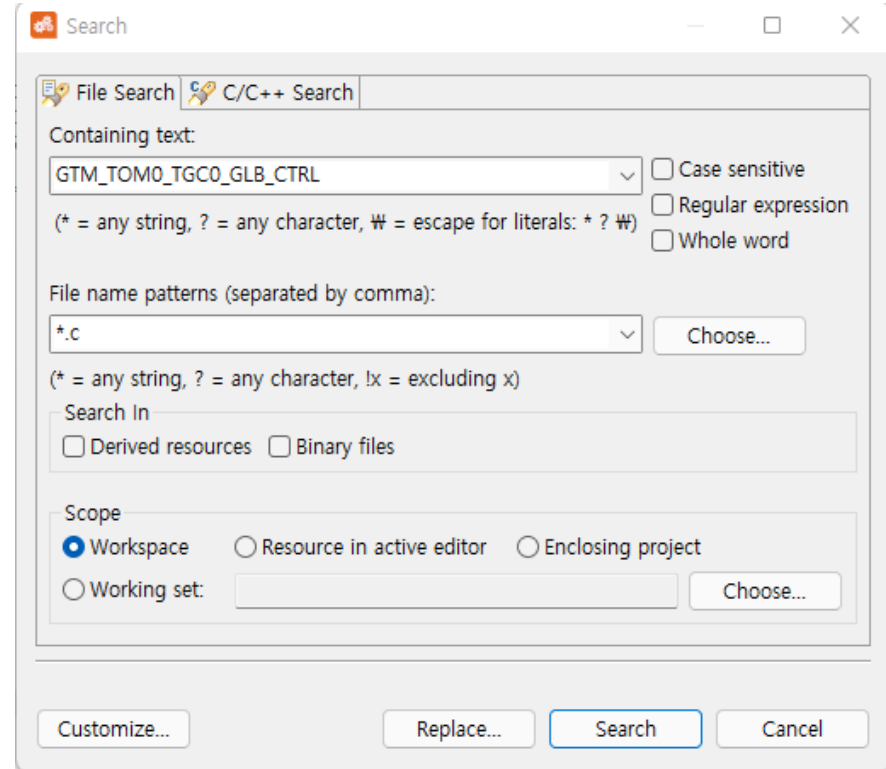
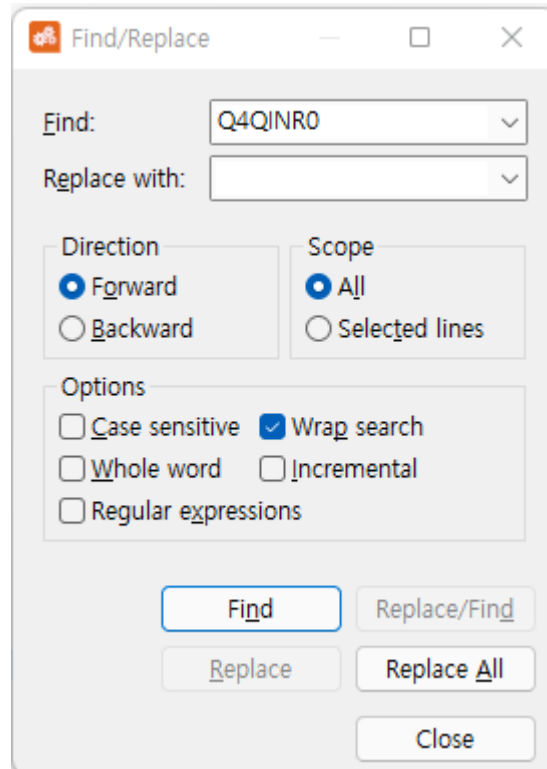
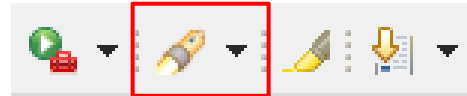
✓ Ctrl-B	Build	
✓ Ctrl-F2	Exit Debug	
✓ F5	Step Into	중단점 다음 라인, 다음 라인이 함수라면 함수 내부로 들어간다.
✓ F6	Step Over	중단점 다음 라인, 다음 라인이 함수라면 실행하되 내부로 들어가지는 않는다.
✓ F7	Step Return	현재 함수의 리턴으로 이동한다. 함수를 빠져 나온다.
✓ F8	Run	
✓ F3	Open Declation	

Pin Map



- Pin Map Icon을 클릭하면 Pinmapper Tab이 생기면서 Pin Map 확인이 가능하다.
- Pin을 클릭하면 PIN 설정을 control 할 수 있다.

찾기



- Find(Ctrl + F)를 통한 검색 : 현재 편집 중인 파일 내 검색
- Search를 통한 검색 : file들 간의 검색

System Timer Test

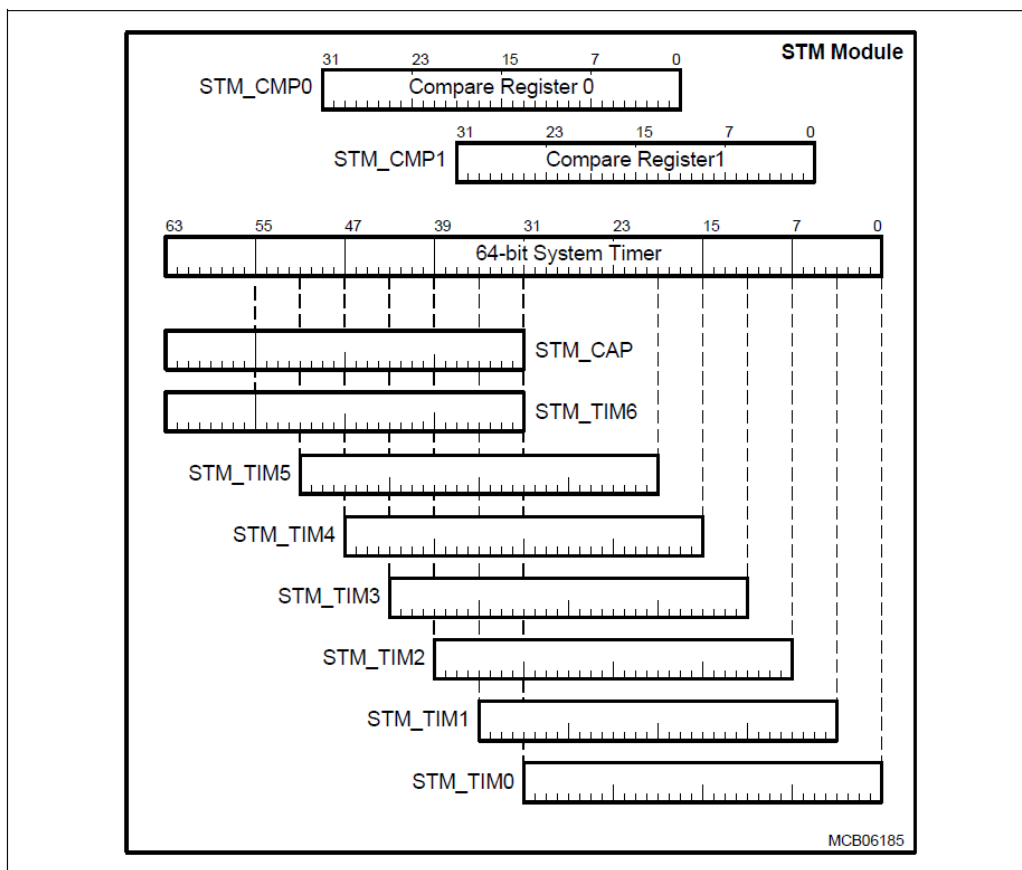


Figure 17-1 General Block Diagram of the STM Module

Table 17-1 Registers Address Space

Module	Base Address	End Address	Note
STM0	F000 0000 _H	F000 00FF _H	STM for CPU0
STM1	F000 0100 _H	F000 01FF _H	STM for CPU1
STM2	F000 0200 _H	F000 02FF _H	STM for CPU2

Table 17-2 Registers Overview - STM Control Registers

Short Name	Description	Offset Addr.	Access Mode		Reset	Description See
			Read	Write		
CLC	Clock Control Register	00 _H	U, SV	SV, E, P	Application ¹⁾	Page 17-8
-		04 _H	BE	BE	-	-
ID	Identification Register	08 _H	U, SV	BE	Application	Page 17-9
-		0C _H	BE	BE	-	-
TIM0	Timer 0 Register	10 _H	U, SV	BE	Application	Page 17-10

User's Manual
STM, V1.11

17-5

V2.2, 2014-12



TC27x D-Step

System Timer (STM)

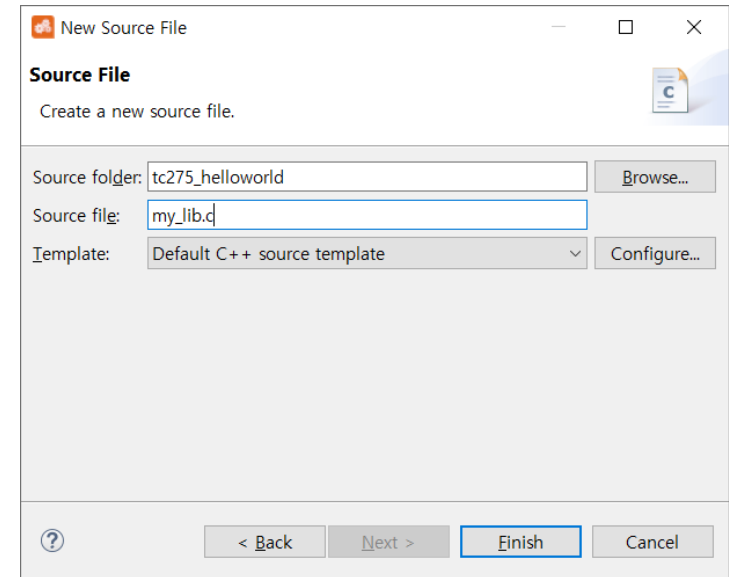
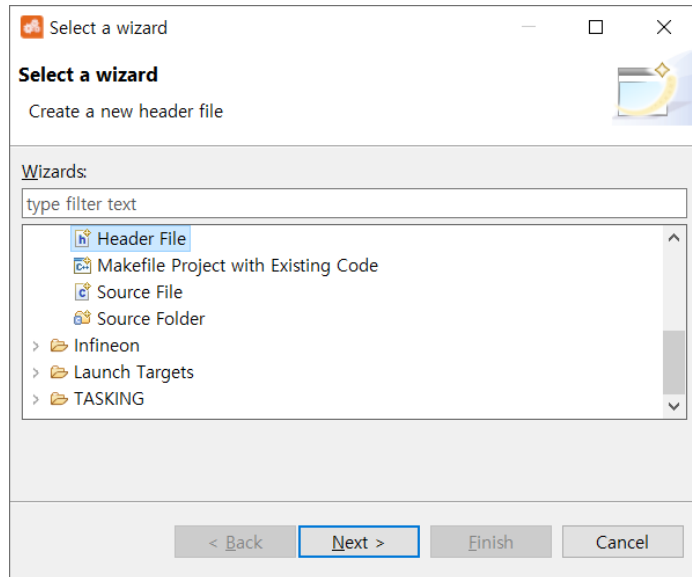
Table 17-2 Registers Overview - STM Control Registers

Short Name	Description	Offset Addr.	Access Mode		Reset	Description See
			Read	Write		
TIM1	Timer 1 Register	14 _H	U, SV	BE	Application	Page 17-10
TIM2	Timer 2 Register	18 _H	U, SV	BE	Application	Page 17-11
TIM3	Timer 3 Register	1C _H	U, SV	BE	Application	Page 17-11
TIM4	Timer 4 Register	20 _H	U, SV	BE	Application	Page 17-11
TIM5	Timer 5 Register	24 _H	U, SV	BE	Application	Page 17-12
TIM6	Timer 6 Register	28 _H	U, SV	BE	Application	Page 17-12
CAP	Timer Capture Register	2C _H	U, SV	BE	Application	Page 17-13
CMP0	Compare Register 0	30 _H	U, SV	U, SV	Application	Page 17-13

Header File, C File 추가

File → New → Other

my_lib.h, my_lib.c 추가



```
my_lib.h x
1 @ /*
2  * my_lib.h
3  *
4  * Created on: 2022. 3. 1.
5  * Author: changmin
6  */
7
8 #ifndef MY_LIB_H_
9 #define MY_LIB_H_
10
11
12
13
14
15 #endif /* MY_LIB_H_ */
16
17 void test_0(void);
18
```

```
my_lib.c x
1 @ /*
2  * my_lib.c
3  *
4  * Created on: 2022. 3. 1.
5  * Author: changmin
6  */
7
8 #include <stdio.h>
9
10 void test_0(void)
11 {
12     printf("Hello World @ test_0\n");
13 }
14
15
```



```

14 * Software is furnished to do so, all subject to the following:
15 *
16 * The copyright notices in the Software and this entire statement, including the above license grant, this
17 * and the following disclaimer, must be included in all copies of the Software, in whole or in part, and in
18 * derivative works of the Software, unless such copies or derivative works are solely in the form of
19 * machine-executable object code generated by a source language processor.
20 *
21 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED
22 * TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT
23 * SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER
24 * IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
25 * IN THE SOFTWARE.
26 *****
27 #include "Ifx_Types.h"
28 #include "IfxCpu.h"
29 #include "IfxScuWdt.h"
30 #include <stdio.h>
31 #include <string.h>
32 #include "my_lib.h"
33
34 IfxCpu_syncEvent g_cpuSyncEvent = 0;
35
36 unsigned int systemtick[4];
37
38 volatile int checksum_0;
39 volatile int checksum_1;
40
41 #define SYSTEM_TIMER_31_0 *(unsigned int *) (0xF0000000+0x10)
42
43 int core0_main(void)
44 {
45     IfxCpu_enableInterrupts();
46
47     /* !!WATCHDOG0 AND SAFETY WATCHDOG ARE DISABLED HERE!!
48      * Enable the watchdogs and service them periodically if it is required
49      */
50     IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword());
51     IfxScuWdt_disableSafetyWatchdog(IfxScuWdt_getSafetyWatchdogPassword());
52
53     /* Wait for CPU sync event */
54     IfxCpu_emitEvent(&g_cpuSyncEvent);
55     IfxCpu_waitEvent(&g_cpuSyncEvent, 1);
56
57     printf("Hello World\n");
58
59     // CPU0 Data Scratch-Pad RAM
60     systemtick[0] = SYSTEM_TIMER_31_0;
61     checksum_0 = 0;
62     for( int i=0; i<0x2000; i++)
63         checksum_0 += *((volatile int *)0x70008000+i);
64     systemtick[1] = SYSTEM_TIMER_31_0;
65
66     // CPU1 Data Scratch-Pad RAM
67     systemtick[2] = SYSTEM_TIMER_31_0;
68     checksum_1 = 0;
69     for( int i=0; i<0x2000; i++)
70         checksum_1 += *((volatile int *)0x60008000+i);
71     systemtick[3] = SYSTEM_TIMER_31_0;
72
73     printf("0x7000 access @ cpu0 : %d\n", systemtick[1]-systemtick[0]);
74     printf("0x6000 access @ cpu0 : %d\n", systemtick[3]-systemtick[2]);
75
76     systemtick[0] = SYSTEM_TIMER_31_0;
77     memcpy((char *)0x70008000, (char *)0x70008000, 0x8000);
78     systemtick[1] = SYSTEM_TIMER_31_0;
79
80     systemtick[2] = SYSTEM_TIMER_31_0;
81     memcpy((char *)0x60008000, (char *)0x60008000, 0x8000);
82     systemtick[3] = SYSTEM_TIMER_31_0;
83
84     printf("0x7000 memcpy @ cpu0 : %d\n", systemtick[1]-systemtick[0]);
85     printf("0x6000 memcpy @ cpu0 : %d\n", systemtick[3]-systemtick[2]);
86
87     test_0();
88
89     while(1)
90     {

```

MAP 파일

- Map 파일 경로 : Debugger 폴더 아래에 *.map로 존재

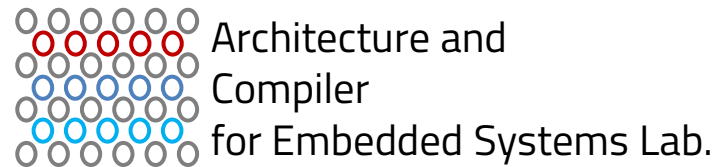
***** Link Result *****						
[in] File	[in] Section	[in] Size (MAU)	[out] Offset	[out] Section	[out] Size (MAU)	
Cpu0_Main.o	.bss.Cpu0_Main.delay_time_ns (7784)	0x00000004	0x0	.bss.Cpu0_Main.delay_time_ns (7784)	0x00000004	
Cpu0_Main.o	.bss.Cpu0_Main.systemtick (7782)	0x00000008	0x0	.bss.Cpu0_Main.systemtick (7782)	0x00000008	
Cpu0_Main.o	.bss.Cpu0_Main.tick_cnt (7783)	0x00000004	0x0	.bss.Cpu0_Main.tick_cnt (7783)	0x00000004	
atexit.o	.bss._atexitarr.libcs_fpu (8224)	0x00000080	0x0	.bss._atexitarr.libcs_fpu (8224)	0x00000080	

* Memory usage in bytes					
Memory	Code	Data	Reserved	Free	Total
mpe:df1s0	0x0	0x0	0x104000	0x0	0x104000
mpe:dsram0	0x0	0x000080	0x002c00	0x019380	0x01c000
mpe:dsram1	0x0	0x0002a5	0x003c00	0x01a15b	0x01e000
mpe:dsram2	0x0	0x000080	0x002c00	0x01b380	0x01e000
mpe:edmem	0x0	0x0	0x0	0x100000	0x100000
mpe:lmuram	0x0	0x0	0x0	0x008000	0x008000
mpe:pfls0	0x002822	0x00027b	0x0	0x1fd563	0x200000
mpe:pfls1	0x0	0x0	0x0	0x200000	0x200000
mpe:psram0	0x0	0x0	0x0	0x006000	0x006000
mpe:psram1	0x0	0x0	0x0	0x008000	0x008000
mpe:psram2	0x0	0x0	0x0	0x008000	0x008000
Total	0x002822	0x000620	0x10d400	0x569d8e	0x67a000

+ Space mpe:vtc:linear (MAU = 8bit)						
Chip	Group	Section	Size (MAU)	Space addr	Chip addr	Alignment
mpe:dsram2		ustack_tc2 (8256)	0x00000800	0x5001ae00	0x0001ae00	0x00000008
mpe:dsram2		istack_tc2 (8257)	0x00000400	0x5001b700	0x0001b700	0x00000008
mpe:dsram2		csa_tc2 (8266)	0x00002000	0x5001bc00	0x0001bc00	0x00000040
mpe:dsram1		.data.Cpu0_Main.g_cpuSyncEvent (7785)	0x00000004	0x60000000	0x0	0x00000004
mpe:dsram1		.data.IfxCpuCcu.IfxCpuCcu_xtalFrequency (5848)	0x00000004	0x60000004	0x00000004	0x00000004
mpe:dsram1		.data._atexitarr.libcs_fpu (8223)	0x00000001	0x60000008	0x00000008	0x00000004
mpe:dsram1		.data._end.libcs_fpu (8210)	0x00000004	0x6000000c	0x0000000c	0x00000004
mpe:dsram1		.data._iob.libcs_fpu (7913)	0x00000008	0x60000010	0x00000010	0x00000004
mpe:dsram1		.data.libcpax_fpu (8129)	0x00000004	0x60000014	0x00000014	0x00000004
mpe:dsram1		.bss.Cpu0_Main.delay_time_ns (7784)	0x00000004	0x6000001c	0x0000001c	0x00000004
mpe:dsram1		.bss.Cpu0_Main.systemtick (7782)	0x00000008	0x60000020	0x00000020	0x00000004
mpe:dsram1		.bss.Cpu0_Main.tick_cnt (7783)	0x00000004	0x60000028	0x00000028	0x00000004
mpe:dsram1		.bss._atexitarr.libcs_fpu (8224)	0x00000080	0x6000002c	0x0000002c	0x00000004
mpe:dsram1		.bss._dbg_request.libcs_fpu (8029)	0x00000014	0x60000030	0x00000030	0x00000004
mpe:dsram1		.bss._malloc_start.libcs_fpu (8201)	0x00000004	0x60000034	0x00000034	0x00000004
mpe:dsram1		.bss.libcpax_fpu (8157)	0x00000004	0x60000038	0x00000038	0x00000004
mpe:dsram1		.bss.stdin_buf.libcs_fpu (7911)	0x00000050	0x6000003c	0x0000003c	0x00000004
mpe:dsram1		.bss.stdout_buf.libcs_fpu (7912)	0x00000050	0x60000040	0x00000040	0x00000004
mpe:dsram1		heap (8258)	0x00001000	0x60019e00	0x00019e00	0x00000008
mpe:dsram1		ustack_tcl (8254)	0x00000800	0x6001ae00	0x0001ae00	0x00000008
mpe:dsram1		istack_tcl (8255)	0x00000400	0x6001b700	0x0001b700	0x00000008
mpe:dsram1		csa_tcl (8267)	0x00002000	0x6001bc00	0x0001bc00	0x00000040
mpe:dsram0		ustack_tc0 (8252)	0x00000800	0x70018e00	0x00018e00	0x00000008
mpe:dsram0		istack_tc0 (8253)	0x00000400	0x70019700	0x00019700	0x00000008
mpe:dsram0		csa_tc0 (8268)	0x00002000	0x70019c00	0x00019c00	0x00000040
mpe:pfls0	bmh_0	.rodatta.bmh_0 (7687)	0x00000020	0x80000000	0x0	0x00000002
mpe:pfls0	reset	.text.start (7685)	0x00000006	0x80000002	0x00000002	0x00000002
mpe:pfls0		_ic_cforce (8242)	0x00000004	0x80000006	0x00000006	0x00000004
mpe:pfls0		.text.Exit.libc (8108)	0x00000004	0x8000000a	0x0000000a	0x00000004
mpe:pfls0		.text.librt (8241)	0x00000020	0x8000000e	0x0000000e	0x00000008
mpe:pfls0		.text..cocofun_1.libcs_fpu (7865)	0x0000000a	0x80000012	0x00000012	0x00000002
mpe:pfls0		.text..cocofun_1.libcs_fpu (8023)	0x0000000a	0x80000018	0x00000018	0x00000002
mpe:pfls0		.text..cocofun_2.libcs_fpu (7864)	0x00000010	0x8000001c	0x0000001c	0x00000002
mpe:pfls0		.text.CompilerTasking.Ifxc_C_Init (7746)	0x00000006	0x80000020	0x00000020	0x00000002
mpe:pfls0		.rodatta.IfxcCpu_cfg.IfxcCpu_cfg_indexMap (5696)	0x00000018	0x80000024	0x00000024	0x00000004
mpe:pfls0		.rodatta.IfxcCcuCcu.IfxcCcuCcu_aDefaultPllConfigSteps (5847)	0x00000024	0x80000028	0x00000028	0x00000004

Q & A

Thank you for your attention



School of Electronics Engineering, KNU

ACE Lab (hn02301@gmail.com)