EDDI
Electronic Design
Development Institute

에디로봇아카데미

임베디드 마스터 Lv2 과정
[TMS 570_GIO/RTI]

제 1기

2022. 02. 11

박태인

기본 GPIO 설정

workspace_v11 - external_gpio_test/source/HL_sys_main.c - Code Composer Studio

File Edit View Navigate Project Run Scripts Window Help

Project Explorer

- external_gpio_test [Active - Debug]
  - Binaries
  - Includes
  - Debug
  - include
  - source
    - HL_epc.c
    - HL_errata_SSWF021_45.c
    - HL_errata.c
    - HL_esm.c
    - HL_gio.c
    - HL_nmpu.c
    - HL_notification.c
    - HL_pinmux.c
    - HL_sys_core.asm
    - HL_sys_dma.c
    - HL_sys_intvecs.asm
    - HL_sys_link.cmd
    - HL_sys_main.c
    - HL_sys_mpu.asm
    - HL_sys_pcr.c
    - HL_sys_phantom.c
    - HL_sys_pmm.c
    - HL_sys_pmu.asm
    - HL_sys_startup.c
    - HL_sys_vim.c
    - HL_system.c
  - targetConfigs
  - external_gpio_test.dil
  - external_gpio_test.hcg

Getting Started | *HL_sys_main.c | HL_gio.h | HL_gio.c

```
55
56 /** @fn void main(void)
57 *   @brief Application main function
58 *   @note This function is empty by default.
59 *
60 *   This function is called after startup.
61 *   The user can use this function to implement the application.
62 */
63
64 /* USER CODE BEGIN (2) */
65 /* USER CODE END */
66
67 int main(void)
68 {
69 /* USER CODE BEGIN (3) */
70     gioInit();
71     gioSetDirection(gioPORTA, 0xffffffff);
72     gioSetPort(gioPORTA, 0xffffffff);
73     gioSetBit(gioPORTA, 4, 1);
74
75     for (;;);
76
77 /* USER CODE END */
78
```

초기화
PORTA set (DDRA 같은 설정/사용여부)
Setport (출력 input/Output 설정)
SetBit (해당 비트[4] high 출력)

Console

CDT Build Console [external_gpio_test]

```
/home/taein/ti/ccs1110/ccs/utils/bin/gmake -k -j 4 all -O

gmake[1]: 'external_gpio_test.out' is up to date.

**** Build Finished ****
```

Problems | Advice | Memory Allocation | Stack Usage

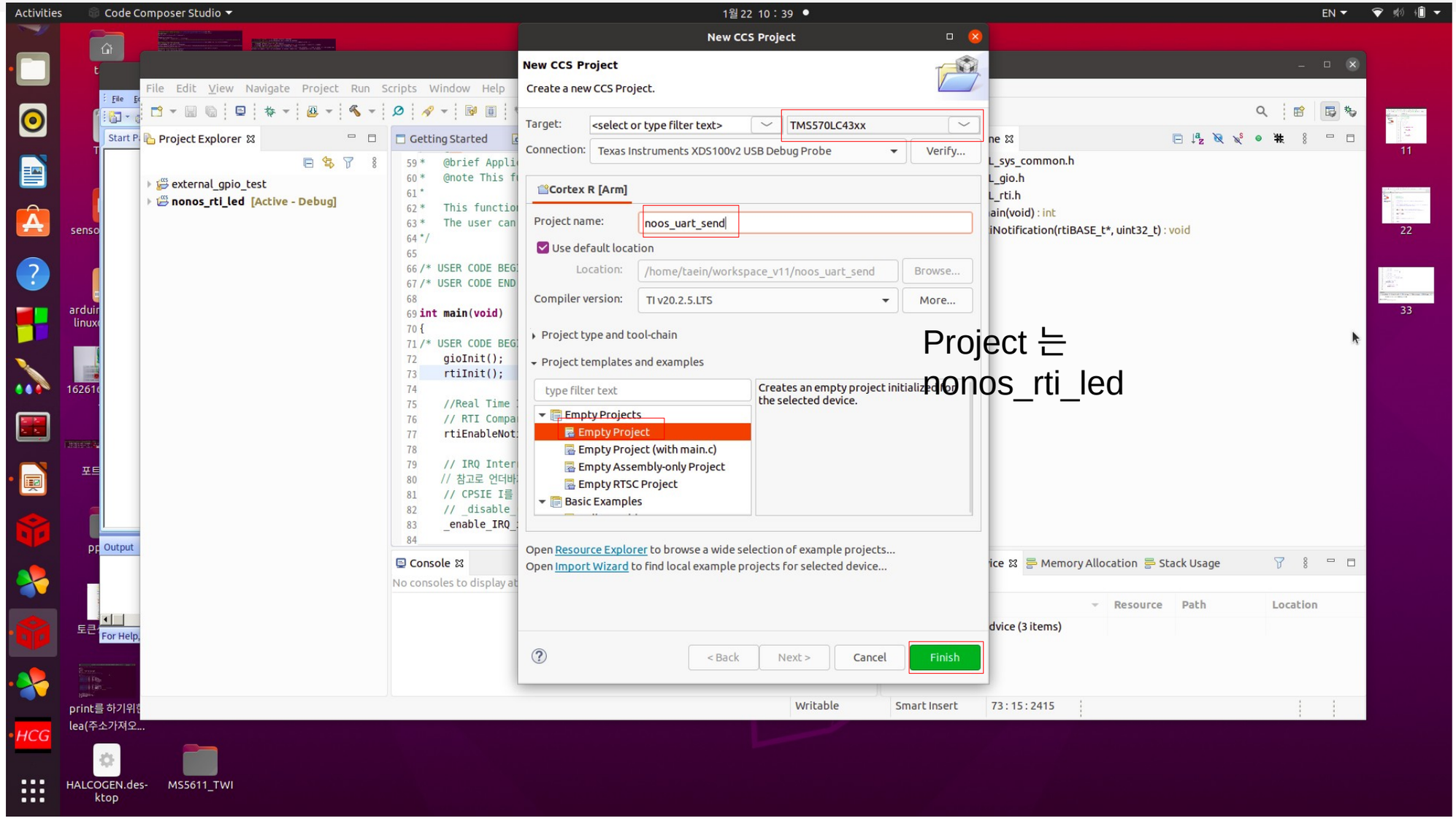
0 errors, 2 warnings, 0 others

Description	Resource	Path	Locati
Warnings (2 items)			

Writable | Smart Insert | 71 : 44 : 2421

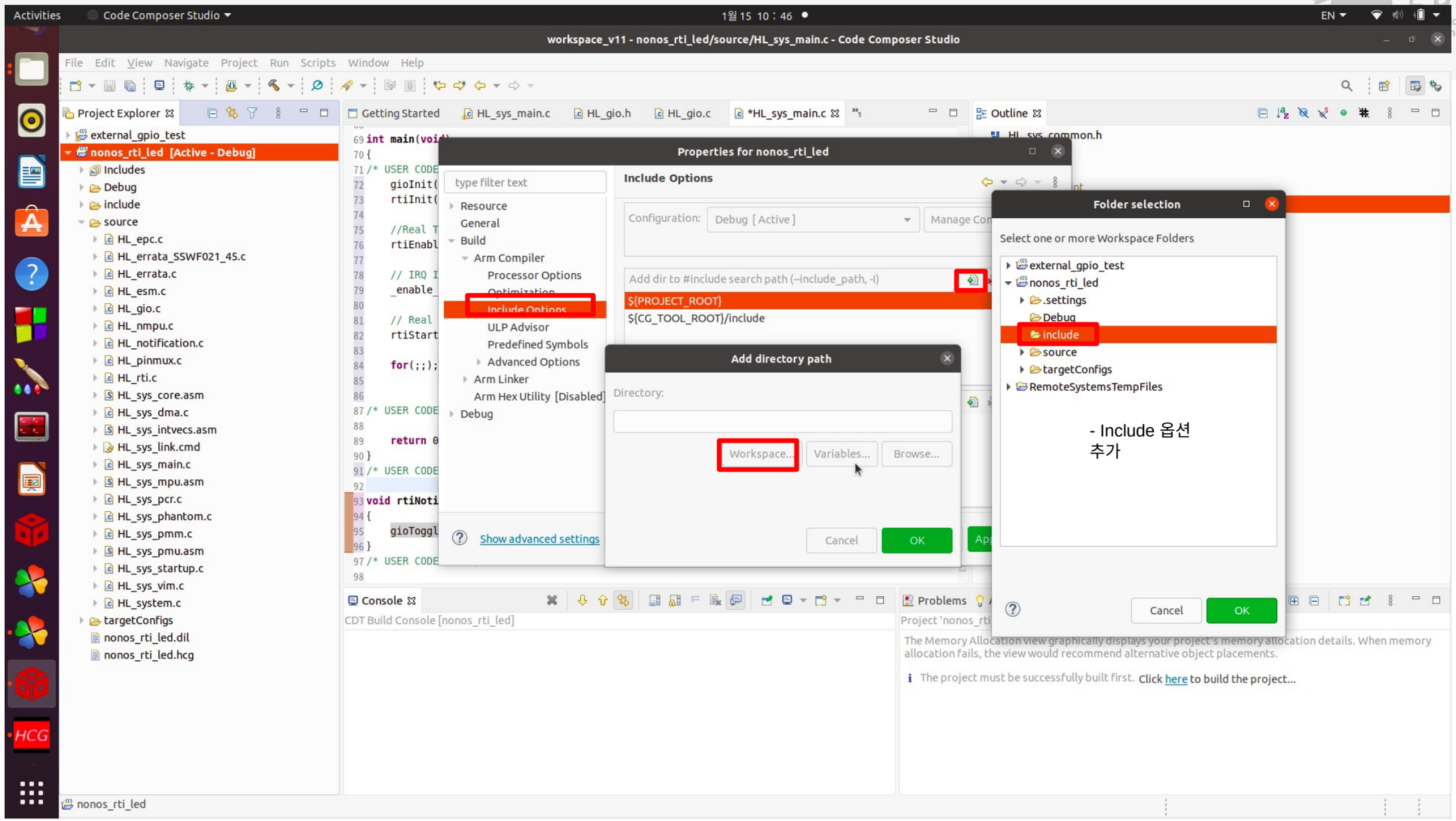
RTI 동작

- CCS 설정
  ㄴ 코딩 및 컴파일
프로그램

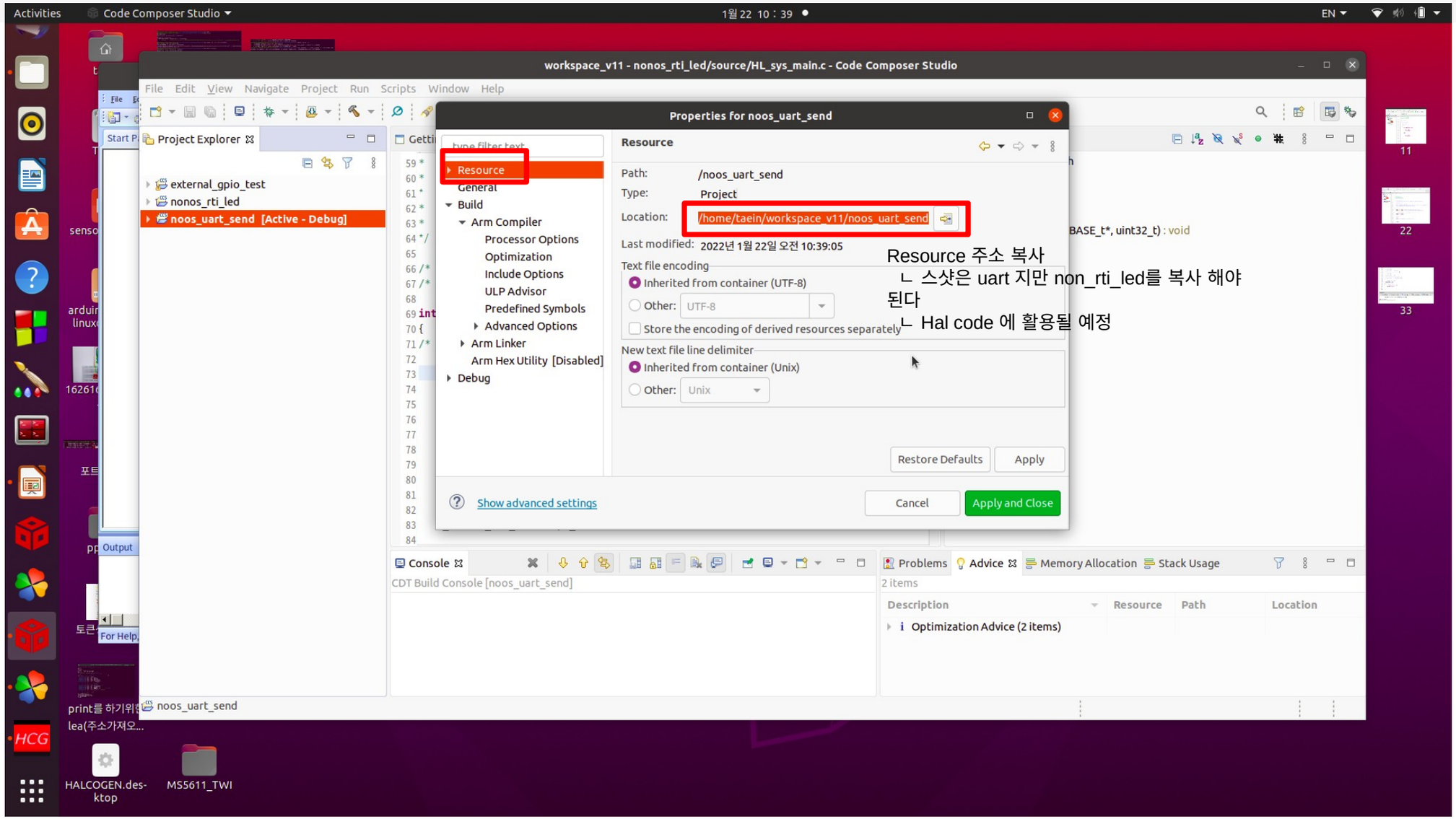


Project 는
nonos_rti_led

RTI 동작


workspace_v11 - nonos_rti_led/source/HL_sys_main.c - Code Composer Studio
Properties for nonos_rti_led
Include Options
Add directory path
Folder selection
Select one or more Workspace Folders
- Include 옵션
추가

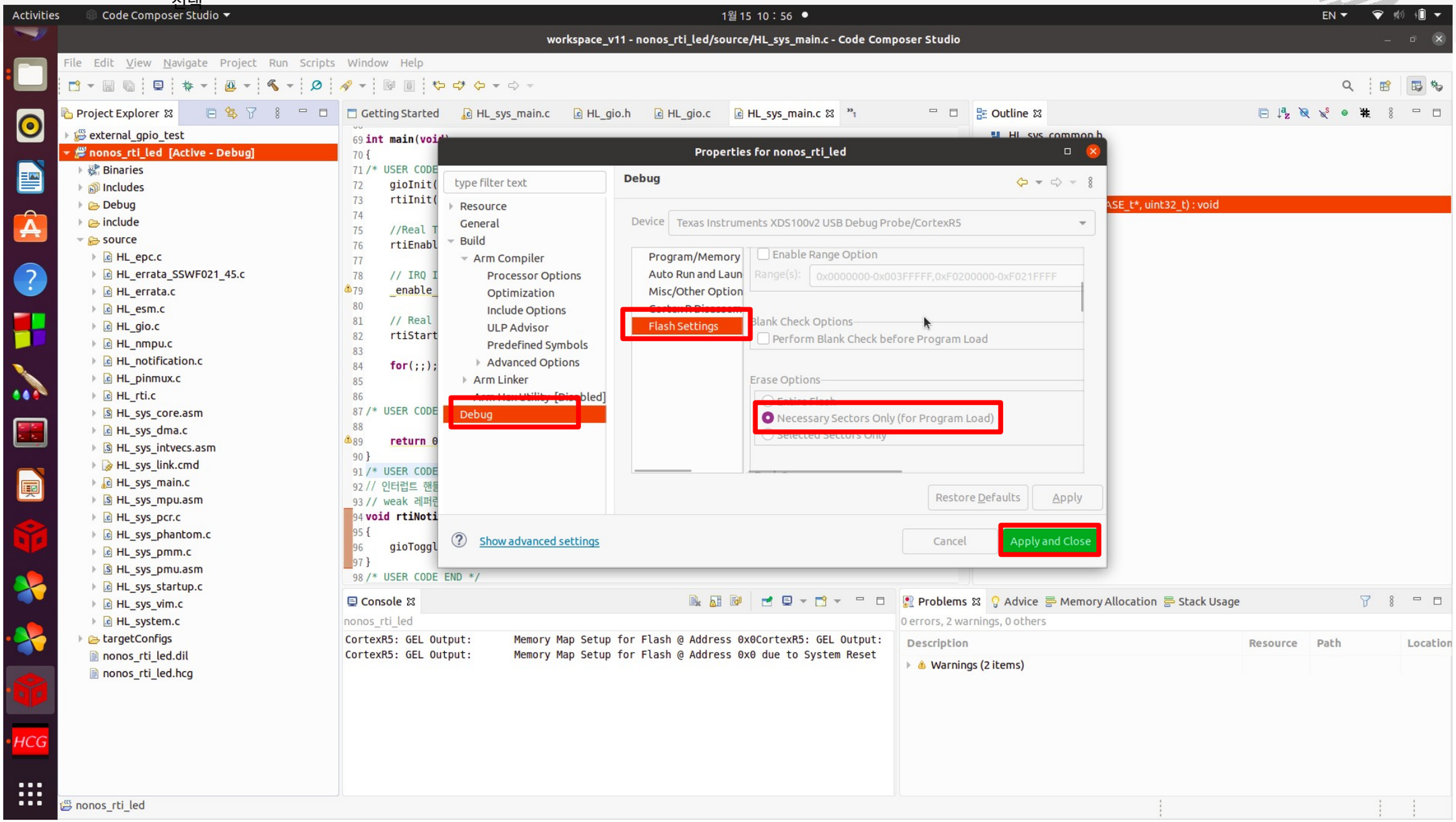
RTI 동작



Resource 주소 복사
ㄴ 스샷은 uart 지만 non_rti_led를 복사 해야 된다
ㄴ Hal code 에 활용될 예정

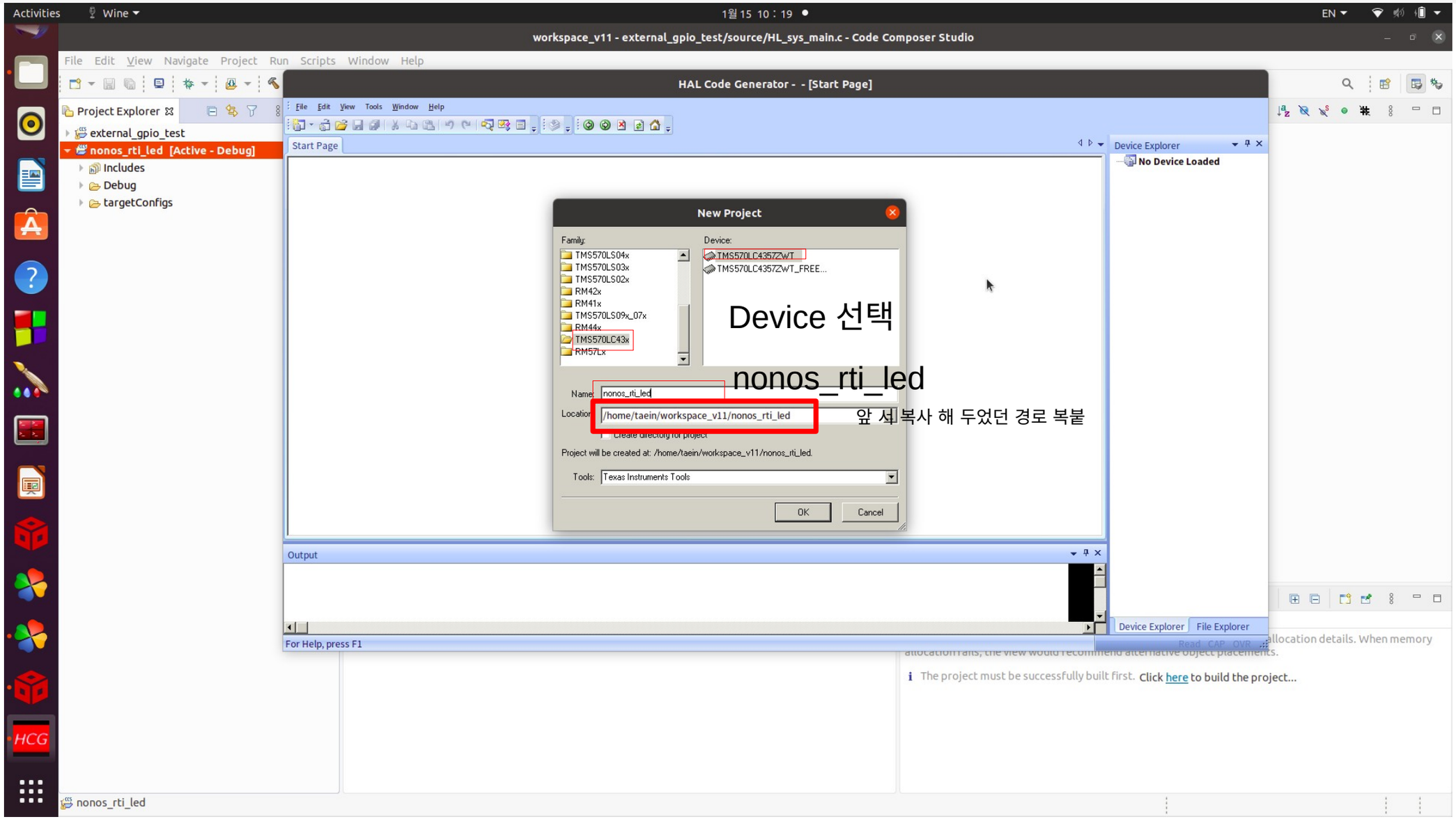
RTI 동작

- 팁 : flash 할 때 속도 높이는 방법
  ㄴ Debug 옵션에서 아래 옵션 선택



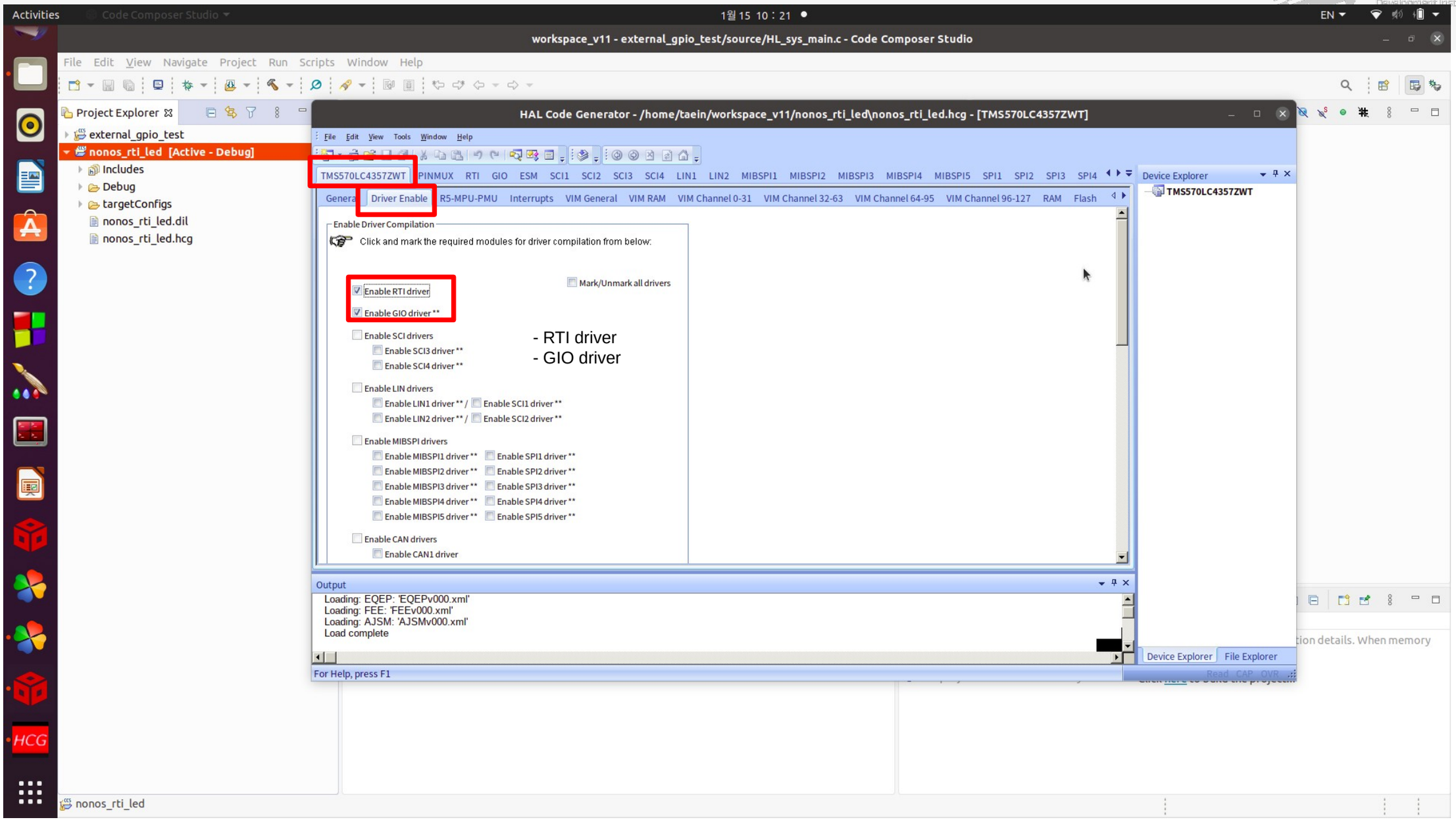
RTI 동작

- HAL Code 설정
  ㄴ Code Generator 같은 느낌의 프로그램

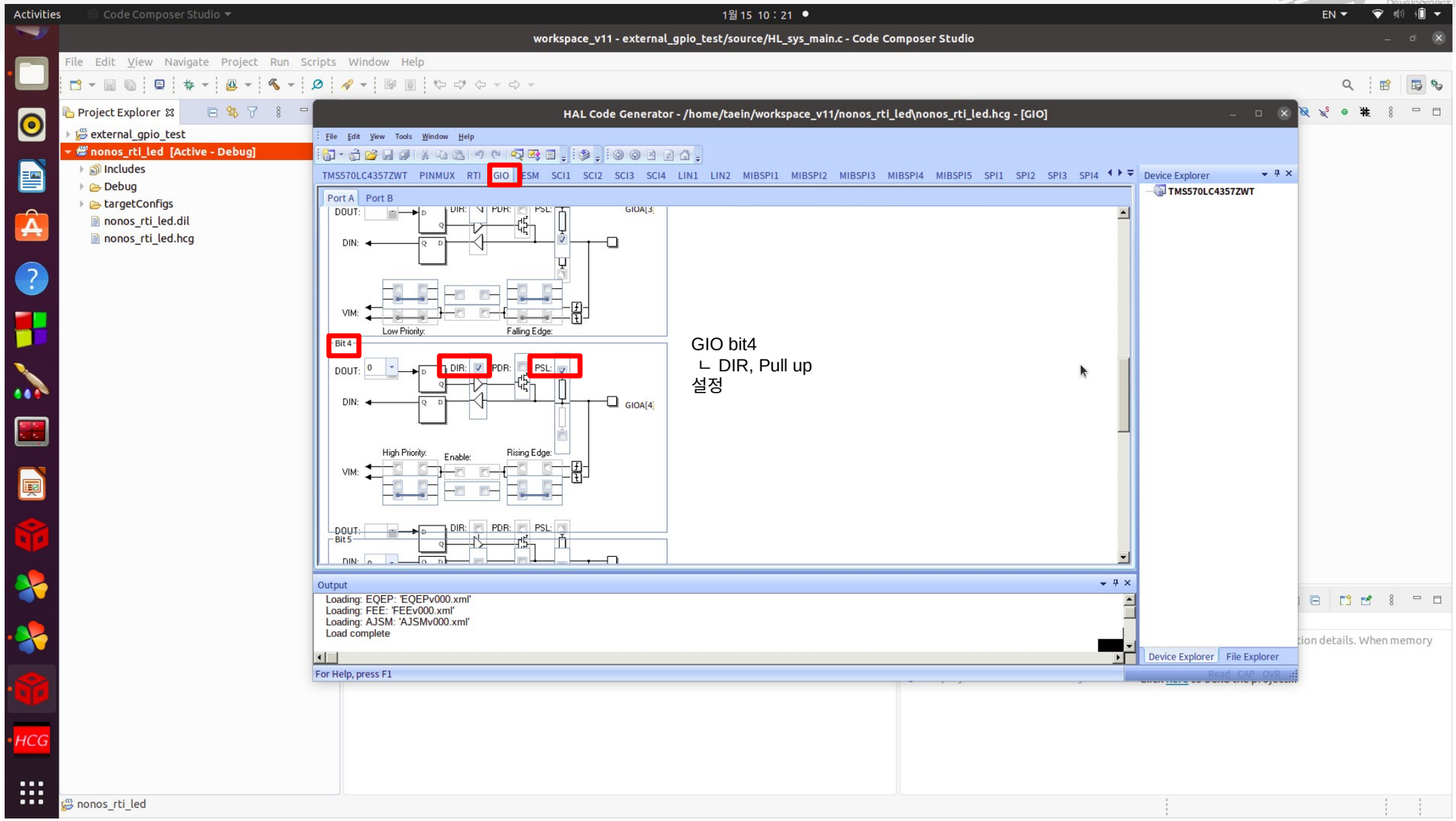


RTI 동작

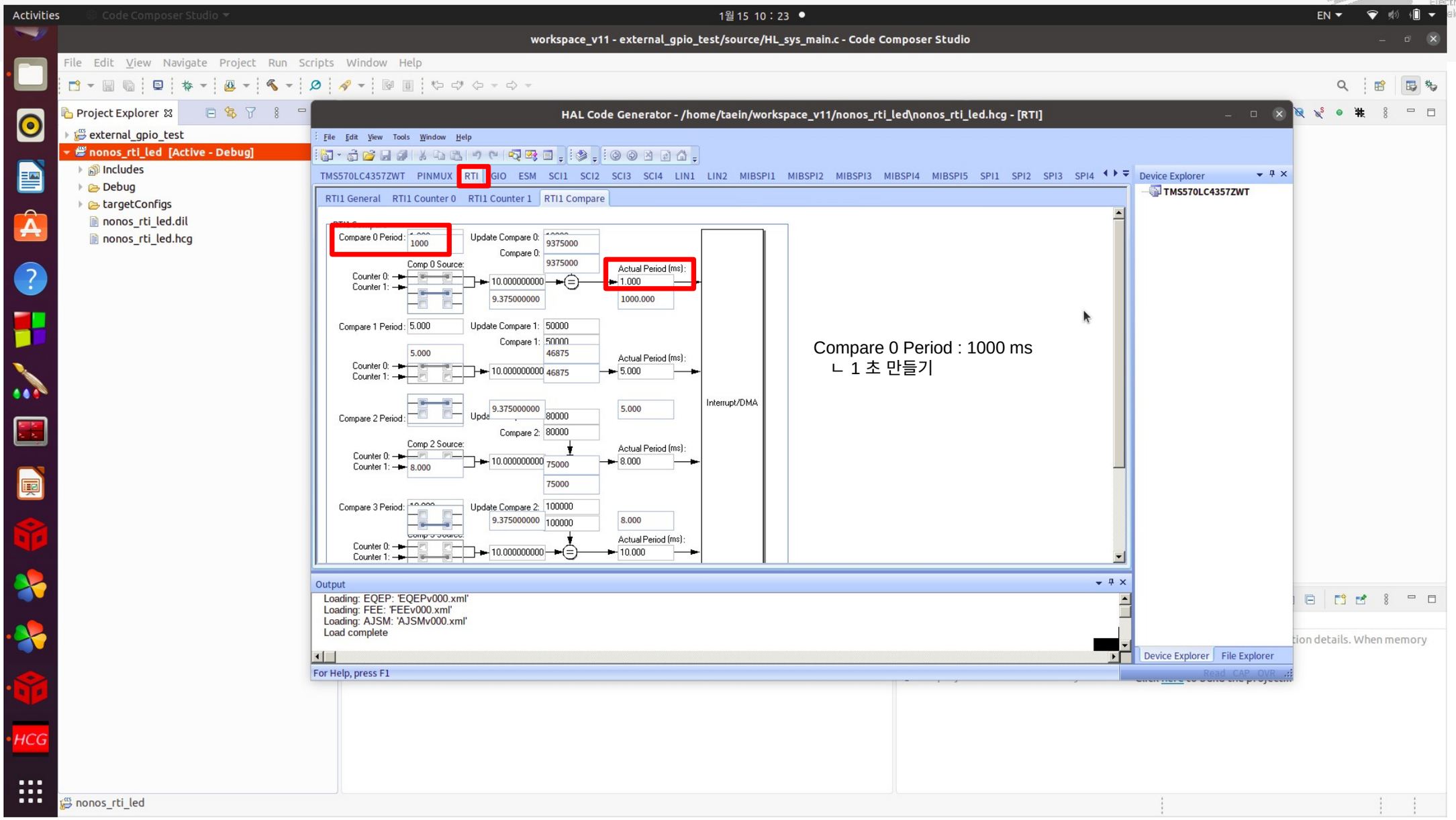
- HAL Code 설정



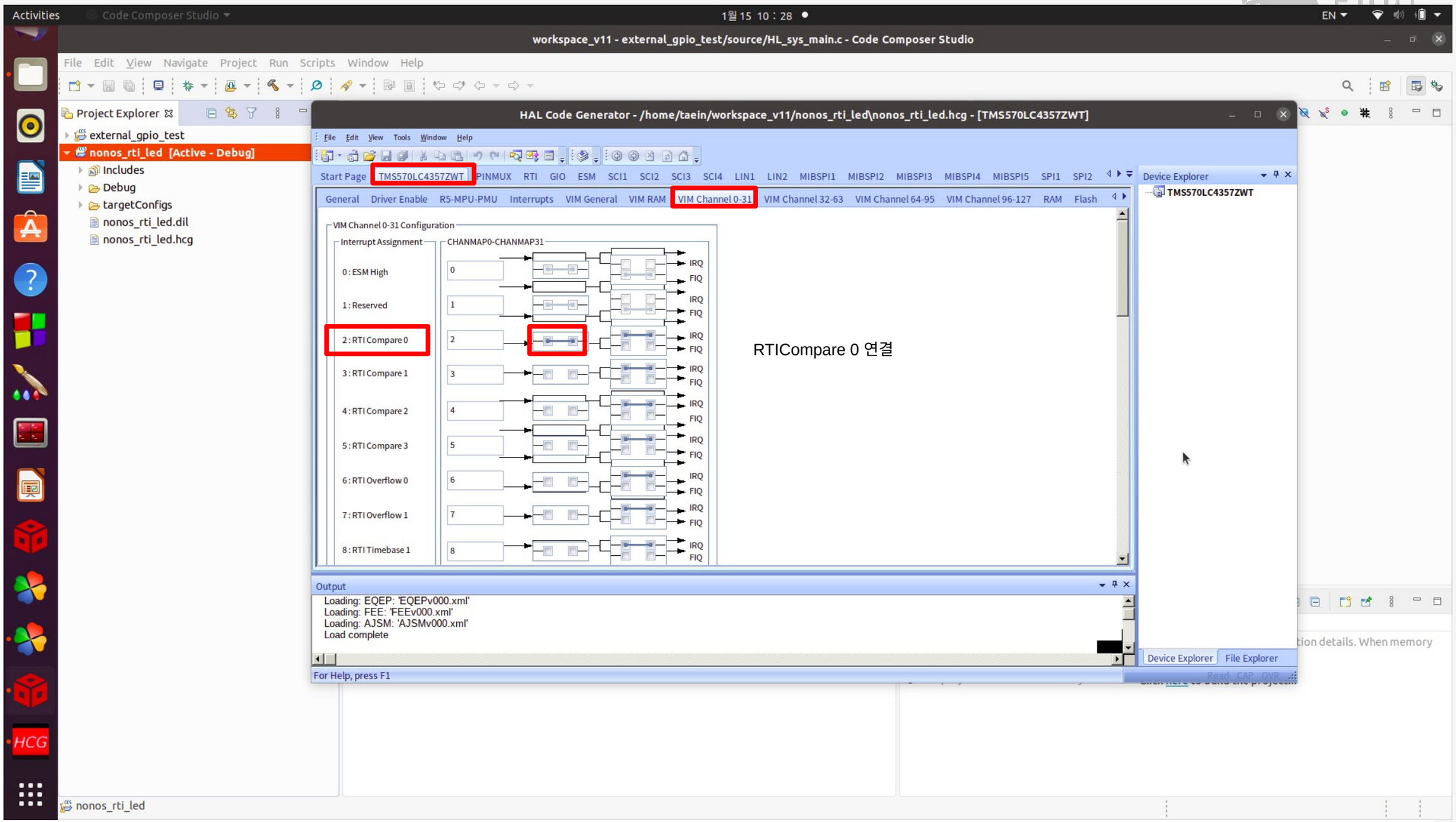
RTI 동작


workspace_v11 - external_gpio_test/source/HL_sys_main.c - Code Composer Studio
HAL Code Generator - /home/taein/workspace_v11/nonos_rti_led\nonos_rti_led.hcg - [GIO]
TMS570LC4357ZWT PINMUX RTI GIO ESM SCI1 SCI2 SCI3 SCI4 LIN1 LIN2 MIBSPI1 MIBSPI2 MIBSPI3 MIBSPI4 MIBSPI5 SPI1 SPI2 SPI3 SPI4
Port A Port B
Bit 4
DOUT: DIN: VIM: DIR: PDR: PSL:
Low Priority: Falling Edge:
High Priority: Enable: Rising Edge:
GIOA[4]
Bit 5
Output
Loading: EQEP: 'EQEPv000.xml'
Loading: FEE: 'FEEv000.xml'
Loading: AJSM: 'AJSMv000.xml'
Load complete
Device Explorer
TMS570LC4357ZWT
Project Explorer
external_gpio_test
nonos_rti_led [Active - Debug]
Includes
Debug
targetConfigs
nonos_rti_led.dil
nonos_rti_led.hcg
GIO bit4
ㄴ DIR, Pull up
설정

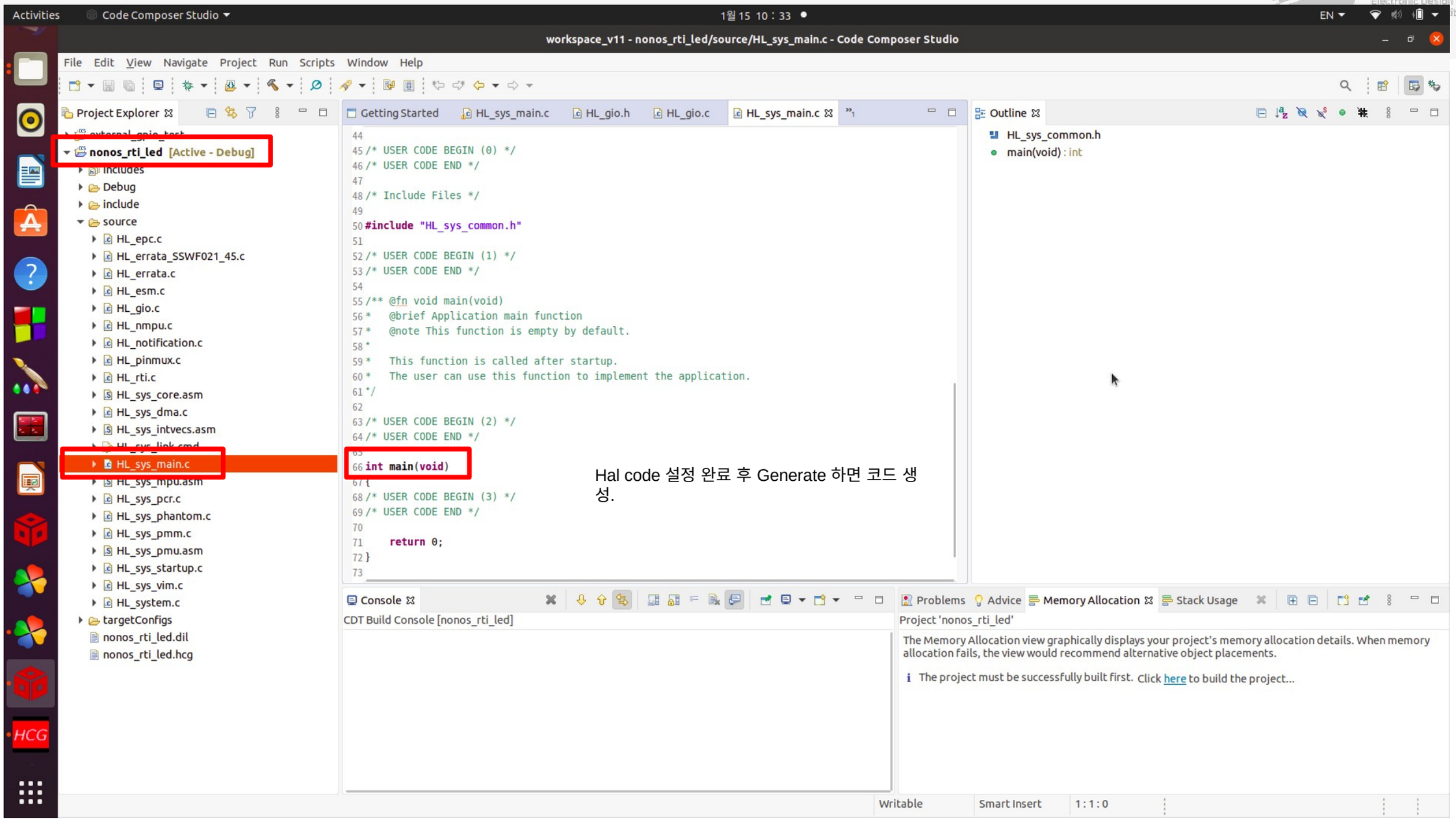
RTI 동작


workspace_v11 - external_gpio_test/source/HL_sys_main.c - Code Composer Studio
HAL Code Generator - /home/taein/workspace_v11/nonos_rti_led\nonos_rti_led.hcg - [RTI]
Project Explorer
external_gpio_test
nonos_rti_led [Active - Debug]
Includes
Debug
targetConfigs
nonos_rti_led.dil
nonos_rti_led.hcg
TMS570LC4357ZWT
PINMUX
RTI
GIO
ESM
SCI1
SCI2
SCI3
SCI4
LIN1
LIN2
MIBSPI1
MIBSPI2
MIBSPI3
MIBSPI4
MIBSPI5
SPI1
SPI2
SPI3
SPI4
RTI1 General
RTI1 Counter 0
RTI1 Counter 1
RTI1 Compare
Compare 0 Period: 1000
Actual Period (ms): 1.000
Compare 1 Period: 5.000
Actual Period (ms): 5.000
Compare 2 Period:
Actual Period (ms): 8.000
Compare 3 Period:
Actual Period (ms): 10.000
Interrupt/DMA
Compare 0 Period : 1000 ms
ㄴ 1 초 만들기
Output
Loading: EQEP: 'EQEPv000.xml'
Loading: FEE: 'FEEv000.xml'
Loading: AJSM: 'AJSMv000.xml'
Load complete
For Help, press F1
Device Explorer
File Explorer

RTI 동작


workspace_v11 - external_gpio_test/source/HL_sys_main.c - Code Composer Studio
HAL Code Generator - /home/taein/workspace_v11/nonos_rti_led\nonos_rti_led.hcg - [TMS570LC4357ZWT]
Start Page
TMS570LC4357ZWT
PINMUX
RTI
GIO
ESM
SCI1
SCI2
SCI3
SCI4
LIN1
LIN2
MIBSPI1
MIBSPI2
MIBSPI3
MIBSPI4
MIBSPI5
SPI1
SPI2
General
Driver Enable
R5-MPU-PMU
Interrupts
VIM General
VIM RAM
VIM Channel 0-31
VIM Channel 32-63
VIM Channel 64-95
VIM Channel 96-127
RAM
Flash
VIM Channel 0-31 Configuration
Interrupt Assignment
CHANMAP0-CHANMAP31
0 : ESM High
1 : Reserved
2 : RTI Compare 0
3 : RTI Compare 1
4 : RTI Compare 2
5 : RTI Compare 3
6 : RTI Overflow 0
7 : RTI Overflow 1
8 : RTI Timebase 1
IRQ
FIQ
RTICompare 0 연결
Output
Loading: EQEP: 'EQEPv000.xml'
Loading: FEE: 'FEEv000.xml'
Loading: AJSM: 'AJSMv000.xml'
Load complete
For Help, press F1
Device Explorer
File Explorer
Project Explorer
external_gpio_test
nonos_rti_led [Active - Debug]
Includes
Debug
targetConfigs
nonos_rti_led.dil
nonos_rti_led.hcg

RTI 동작



Hal code 설정 완료 후 Generate 하면 코드 생성.

RTI 동작

헤더 파일
선언

```c
/* USER CODE BEGIN (0) */
/* USER CODE END */

/* Include Files */

#include "HL_sys_common.h"

/* USER CODE BEGIN (1) */

#include "HL_gio.h"
#include "HL_rti.h"
/* USER CODE END */
```

Main 문

```c
int main(void)
{
/* USER CODE BEGIN (3) */
    gioInit();
    rtiInit();

    //Real Time Interrupt 를 누가 처리 할지
    // RTI Compare0에서 발생하는 인터럽트 허용!!
    rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE0);   // 인터럽트를 구통화 시키세요

    // IRQ Interrupt 활성화
    // 참고로 언더바가 두개면 거의 건드리지 말라는 의미이다.
    // 어셈블리 명령어인 CPSIE I를 통해 CPSR Register 의 I 비트인 IRQ를 활성화 한다.
    // _disable_() 의 경우 CPSID I를 통해 IRQ를 비활성화 한다.
    _enable_IRQ_interrupt_();

    // Real Time Interrupt 카운터 시작!
    // 실제로는 RTI Counter Block 0 을 시작하여 카운팅을 진행
    rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0);

    for(;;);

/* USER CODE END */

    return 0;
}
```

Gio, RTI 초기화

```c
/* USER CODE BEGIN (4) */
// 인터럽트 핸들러 인데 weak 레퍼런스가 지정된 것이고, 원래 이것은 인터럽트 실행시에 실행 되는것인데
// weak 레퍼런스라고 지정되어 있으므로 이것이 수정을 해서 덮어 쓸 수 있는 것이 가능한 것이다.
// 이부분은 RTI Interrupt Handler 로 Timer 가 차면 구동됨
void rtiNotification(rtiBASE_t *rtiREG, uint32_t notification)
{
    // GPIO PortA의 4번을 토글(On/Off) 시킴
    gioToggleBit(gioPORTA, 4);
}
/* USER CODE END */
```

RTI 동작

```
int main(void)
{
/* USER CODE BEGIN (3) */
    gioInit();
    rtiInit();
```

Rti init 알아 보기

인터럽트 (RTI) 모듈 기능 설명이라 한다. RTI는
실시간 운영체제
RTOS 를 지원하는 운영 체제
타이머라 한다.

Real-Time Interrupt (RTI) Module

This chapter describes the functionality of the real-time interrupt (RTI) module. The RTI is designed as an operating system timer to support a real time operating system (RTOS).

17.2 Module Operation

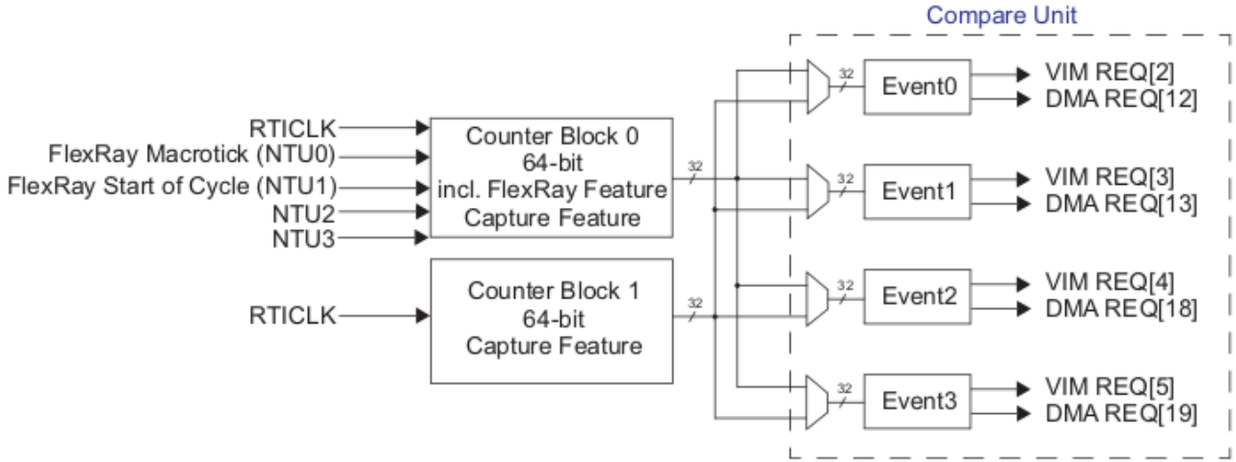
Figure 17-1 illustrates the high level block diagram of the RTI module.

The RTI module has two independent counter blocks for generating different timebases: counter block 0 and counter block 1. The two counter blocks provide the same basic functionality, but counter block 0 has the additional functionality of being able to work with the FlexRay Macrotick (NTU0) or Start of Cycle (NTU1) and perform clock supervision to detect a missing signal.

A compare unit compares the counters with programmable values and generates four independent interrupt or DMA requests on compare matches. Each of the compare registers can be programmed to be compared to either counter block 0 or counter block 1.

The following sections describe the individual functions in more detail.

Figure 17-1. RTI Block Diagram



RTI 모듈에는 서로 다른 타임베이스 생성하기 위한
두 개의 독립적인 카운터 블록 (0 , 1) 이 있습니다.

두 카운터 블록은 동일한 기본 기능을 제공 하지만

카운터 블록 0 에는 FlexRay Macrotick (NTU0)
과 함께 작동 할 수 있는 추가 기능이 있습니다.

또는!!

사이클 시작(NTU1)을 실행하고
클럭 감시를 수행하여 누락된 신호를 감지 합니다.

비교 장치는 카운터를 프로그래밍 가능한 값과 비교하고
4개의 독립적인 값을 생성합니다.

각 비교 레지스터는
카운터 블록 0 또는 블록 1 과 비교되도록 프로그래밍 할 수 있습니다.

RTI 동작

17.3 RTI Control Registers

Table 17-1 provides a summary of the registers. The registers support 8-bit, 16-bit, and 32-bit writes. The offset is relative to the associated peripheral select. See the following sections for detailed descriptions of the registers. The base address for the control registers is FFFF FC00h. The address locations not listed are reserved.

Table 17-1. RTI Registers

Offset	Acronym	Register Description	Section
00h	RTIGCTRL	RTI Global Control Register	Section 17.3.1
04h	RTITBCTRL	RTI Timebase Control Register	Section 17.3.2
08h	RTICAPCTRL	RTI Capture Control Register	Section 17.3.3
0Ch	RTICOMPCTRL	RTI Compare Control Register	Section 17.3.4
10h	RTIFRC0	RTI Free Running Counter 0 Register	Section 17.3.5
14h	RTIUC0	RTI Up Counter 0 Register	Section 17.3.6
18h	RTICPUC0	RTI Compare Up Counter 0 Register	Section 17.3.7
20h	RTICAFRC0	RTI Capture Free Running Counter 0 Register	Section 17.3.8
24h	RTICAUC0	RTI Capture Up Counter 0 Register	Section 17.3.9
30h	RTIFRC1	RTI Free Running Counter 1 Register	Section 17.3.10
34h	RTIUC1	RTI Up Counter 1 Register	Section 17.3.11
38h	RTICPUC1	RTI Compare Up Counter 1 Register	Section 17.3.12
40h	RTICAFRC1	RTI Capture Free Running Counter 1 Register	Section 17.3.13
44h	RTICAUC1	RTI Capture Up Counter 1 Register	Section 17.3.14
50h	RTICOMP0	RTI Compare 0 Register	Section 17.3.15
54h	RTIUDCP0	RTI Update Compare 0 Register	Section 17.3.16
58h	RTICOMP1	RTI Compare 1 Register	Section 17.3.17
5Ch	RTIUDCP1	RTI Update Compare 1 Register	Section 17.3.18
60h	RTICOMP2	RTI Compare 2 Register	Section 17.3.19
64h	RTIUDCP2	RTI Update Compare 2 Register	Section 17.3.20
68h	RTICOMP3	RTI Compare 3 Register	Section 17.3.21
6Ch	RTIUDCP3	RTI Update Compare 3 Register	Section 17.3.22
70h	RTITBLCOMP	RTI Timebase Low Compare Register	Section 17.3.23
74h	RTITBHCOMP	RTI Timebase High Compare Register	Section 17.3.24
80h	RTISETINTENA	RTI Set Interrupt Enable Register	Section 17.3.25
84h	RTICLEARINTENA	RTI Clear Interrupt Enable Register	Section 17.3.26
88h	RTIINTFLAG	RTI Interrupt Flag Register	Section 17.3.27
90h	RTIDWDCTRL	Digital Watchdog Control Register	Section 17.3.28
94h	RTIDWDPRLD	Digital Watchdog Preload Register	Section 17.3.29
98h	RTIWDSTATUS	Watchdog Status Register	Section 17.3.30
9Ch	RTIWDKEY	RTI Watchdog Key Register	Section 17.3.31
A0h	RTIDWDCNTR	RTI Digital Watchdog Down Counter Register	Section 17.3.32
A4h	RTIWWDRXNCTRL	Digital Windowed Watchdog Reaction Control Register	Section 17.3.33
A8h	RTIWWDSIZECTRL	Digital Windowed Watchdog Window Size Control Register	Section 17.3.34
ACh	RTIINTCLRENABLE	RTI Compare Interrupt Clear Enable Register	Section 17.3.35
B0h	RTICOMP0CLR	RTI Compare 0 Clear Register	Section 17.3.36
B4h	RTICOMP1CLR	RTI Compare 1 Clear Register	Section 17.3.37
B8h	RTICOMP2CLR	RTI Compare 2 Clear Register	Section 17.3.38
BCh	RTICOMP3CLR	RTI Compare 3 Clear Register	Section 17.3.39

레지스터 8, 16, 32 bit 쓰기를 지원

오프셋은 연결된 주변기기 선택을 기준으로 한다

레지스터의 기본 주소는 FFFF FC00h

17.3.1 RTI Global Control Register (RTIGCTRL)

The global control register starts/stops the counters and selects the signal compared with the timebase control circuit. This register is shown in Figure 17-12 and described in Table 17-2.

Figure 17-12. RTI Global Control Register (RTIGCTRL) [offset = 00]

31–20	19–16
Reserved	NTUSEL
R-0	R/WP-0

15	14–2	1	0
COS	Reserved	CNT1EN	CNT0EN
R/WP-0	R-0	R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

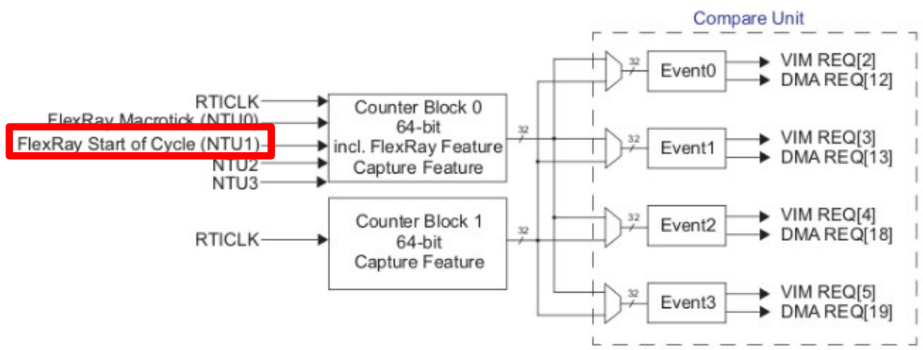
Table 17-2. RTI Global Control Register (RTIGCTRL) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	NTUSEL		Select NTU signal. These bits determine which NTU input signal is used as external timebase
		0h	NTU0
		5h	NTU1
		Ah	NTU2
		Fh	NTU3
		All other values	Tied to 0
15	COS		Continue on suspend. This bit determines if both counters are stopped when the device goes into halting debug mode or if they continue counting.
		0	Counters are stopped while in halting debug mode.
		1	Counters are running while in halting debug mode.
14-2	Reserved	0	Reads return 0. Writes have no effect.
1	CNT1EN		Counter 1 enable. This bit starts and stops counter block 1 (RTIUC1 and RTIFRC1).
		0	Counter block 1 is stopped.
		1	Counter block 1 is running.
0	CNT0EN		Counter 0 enable. This bit starts and stops counter block 0 (RTIUC0 and RTIFRC0).
		0	Counter block 0 is stopped.
		1	Counter block 0 is running.

글로벌 제어 레지스터는 카운터를 시작/중지하고 타임 베이스 제어 회로와 비교하여 신호를 선택한다.

```
/** - Setup NTU source, debug options and disable both counter blocks */
 /*NTU 1 선택 */
rtiREG1->GCTRL = (uint32)((uint32)0x5U << 16U) | 0x00000000U;
```

전체 값을 0 으로 셋팅
0x5 [0b0110] 을 좌측 쉬프트 16bit
NTU 1 선택



17.3.2 RTI Timebase Control Register (RTITBCTRL)

The timebase control register selects if the free running counter 0 is incremented by RTICLK or NTU. This register is shown in Figure 17-13 and described in Table 17-3.

Figure 17-13. RTI Timebase Control Register (RTITBCTRL) [offset = 04h]

31 – 8	7 – 2	1	0
Reserved	Reserved	INC	TBEXT
R-0	R-0	R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 17-3. RTI Timebase Control Register (RTITBCTRL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reads return 0. Writes have no effect.
1	INC		Increment free running counter 0. This bit determines whether the free running counter 0 (RTIFRC0) is automatically incremented if a failing clock on the NTU signal is detected.
		0	RTIFRC0 will not be incremented on a failing external clock.
		1	RTIFRC0 will be incremented on a failing external clock.
0	TBEXT		Timebase external. This bit selects whether the free running counter 0 (RTIFRC0) is clocked by the internal up counter 0 (RTIUC0) or from the external signal NTU. Setting the TBEXT bit from 0 to 1 will not increment RTIFRC0, since RTIUC0 is reset.
			When the timebase supervisor circuit detects a missing clock edge, then the TBEXT bit is reset.
			Only the software can select whether the external signal should be used.
		0	RTIUC0 clocks RTIFRC0.
		1	NTU clocks RTIFRC0.

```
/** - Setup timebase for free running counter 0 */
/* RTIUC0 clocks RTIFRC0 */
rtiREG1->TBCTRL = 0x00000000U;
```

타임베이스 제어 레지스터는 자유 실행 카운터 0이 RTICLK 또는 NTU만큼 증가되는지 선택합니다.

카운터 초기 값?

17.3.3 RTI Capture Control Register (RTICAPCTRL)

The capture control register controls the capture source for the counters. This register is shown in Figure 17-14 and described in Table 17-4.

Figure 17-14. RTI Capture Control Register (RTICAPCTRL) [offset = 08h]

31 – 8	7 – 2	1	0
Reserved	Reserved	CAPCNTR1	CAPCNTR0
R-0	R-0	R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 17-4. RTI Capture Control Register (RTICAPCTRL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reads return 0. Writes have no effect.
1	CAPCNTR1		Capture counter 1. This bit determines which external interrupt source triggers a capture event of RTIUC1 and RTIFRC1.
		0	Capture of RTIUC1/ RTIFRC1 is triggered by capture event source 0.
		1	Capture of RTIUC1/ RTIFRC1 is triggered by capture event source 1.
0	CAPCNTR0		Capture counter 0. This bit determines which external interrupt source triggers a capture event of RTIUC0 and RTIFRC0.
		0	Capture of RTIUC0/ RTIFRC0 is triggered by capture event source 0.
		1	Capture of RTIUC0/ RTIFRC0 is triggered by capture event source 1.

캡처 제어 레지스터는
카운터의 캡처 소스를 제어 합니다.

ㄴ 송수신기가 아닐 땐 필요 없긴 하다.

```
/** - Enable/Disable capture event sources for both counter blocks */
rtiREG1->CAPCTRL = 0U | 0U;
```

17.3.4 RTI Compare Control Register (RTICOMPCTRL)

The compare control register controls the source for the compare registers. This register is shown in Figure 17-15 and described in Table 17-5.

비교 제어 레지스터는
비교 레지스터의 소스를 제어 합니다.

Figure 17-15. RTI Compare Control Register (RTICOMPCTRL) [offset = 0Ch]

31–16	15–13	12	11–9	8
Reserved	Reserved	COMPSEL3	Reserved	COMPSEL2
R-0	R-0	R/WP-0	R-0	R/WP-0

7–5	4	3–1	0
Reserved	COMPSEL1	Reserved	COMPSEL0
R-0	R/WP-0	R-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

```
/** - Setup input source compare 0-3 */
/* RTIFRC1 과 비교*/
rtiREG1->COMPCTRL = 0x00001000U | 0x00000100U | 0x00000000U | 0x00000000U;
```

Table 17-5. RTI Compare Control Register (RTICOMPCTRL) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reads return 0. Writes have no effect.
12	COMPSEL3		Compare select 3. This bit determines the counter with which the compare value held in compare register 3 (RTICOMP3) is compared.
		0	Value will be compared with RTIFRC0.
		1	Value will be compared with RTIFRC1.
11-9	Reserved	0	Reads return 0. Writes have no effect.
8	COMPSEL2		Compare select 2. This bit determines the counter with which the compare value held in compare register 2 (RTICOMP2) is compared.
		0	Value will be compared with RTIFRC0.
		1	Value will be compared with RTIFRC1.
7-5	Reserved	0	Reads return 0. Writes have no effect.
4	COMPSEL1		Compare select 1. This bit determines the counter with which the compare value held in compare register 1 (RTICOMP1) is compared.
		0	Value will be compared with RTIFRC0.
		1	Value will be compared with RTIFRC1.
3-1	Reserved	0	Reads return 0. Writes have no effect.
0	COMPSEL0		Compare select 0. This bit determines the counter with which the compare value held in compare register 0 (RTICOMP0) is compared.
		0	Value will be compared with RTIFRC0.
		1	Value will be compared with RTIFRC1.

비교 레지스터 3(RTICOMP3) 에 있는 비교 값이 비교 되는 카운터를 결정.

비교 레지스터 2(RTICOMP2) 에 있는 비교 값이 비교 되는 카운터를 결정.

비교되는 카운터가 2개??

RTI 동작

```c
int main(void)
{
/* USER CODE BEGIN (3) */
    gioInit();
    rtiInit();

    //Real Time Interrupt 를 누가 처리 할지
    // RTI Compare0에서 발생하는 인터럽트 허용!!
    rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE0);   // 인터럽트를 구통화 시키세요

    // IRQ Interrupt 활성화
    // 참고로 언더바가 두개면 거의 건드리지 말라는 의미이다.
    // 어셈블리 명령어인 CPSIE I를 통해 CPSR Register 의 I 비트인 IRQ를 활성화 한다.
    // _disable_() 의 경우 CPSID I를 토해 IRQ를 비활성화 한다.
    _enable_IRQ_interrupt_();

    // Real Time Interrupt 카운터 시작!
    // 실제로는 RTI Counter Block 0 을 시작하여 카운팅을 진행
    rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0);

    for(;;);

/* USER CODE END */

    return 0;
}
```

RTI 동작

- rtiEnableNotification 함수
  ㄴ RTI 인터럽트를 누가 (rtiREG1) 어떤 방식 (rtiNOTIFICATION_COMPARE0) 으로 처리 할지 정하는 함수

```
//Real Time Interrupt 를 누가 처리 할지
// RTI Compare0에서 발생하는 인터럽트 허용!!
rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE0);   // 인터럽트를 구통화 시키세요
```

```
void rtiEnableNotification(rtiBASE_t *rtiREG, uint32 notification)
{
/* USER CODE BEGIN (38) */
/* USER CODE END */

    rtiREG->INTFLAG = notification;
    rtiREG->SETINTENA   = notification;

    /**   @note The function rtiInit has to be called before this function can be used.\n
    *           This function has to be executed in privileged mode.
    */

/* USER CODE BEGIN (39) */
/* USER CODE END */
}
```

rtiREG1의 포인터에

→ INTFLAG 와
SETINTENA 에

rtiNOTIFICATION_COMPARE0 대입

RTI 동작

```c
int main(void)
{
/* USER CODE BEGIN (3) */
    gioInit();
    rtiInit();

    //Real Time Interrupt 를 누가 처리 할지
    // RTI Compare0에서 발생하는 인터럽트 허용!!
    rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE0);   // 인터럽트를 구통화 시키세요

    // IRQ Interrupt 활성화
    // 참고로 언더바가 두개면 거의 건드리지 말라는 의미이다.
    // 어셈블리 명령어인 CPSIE I를 통해 CPSR Register 의 I 비트인 IRQ를 활성화 한다.
    // _disable_() 의 경우 CPSID I를 토해 IRQ를 비활성화 한다.
    _enable_IRQ_interrupt_();

    // Real Time Interrupt 카운터 시작!
    // 실제로는 RTI Counter Block 0 을 시작하여 카운팅을 진행
    rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0);

    for(;;);

/* USER CODE END */

    return 0;
}
```

- HL_epc.c
- HL_errata_SSWF021_45.c
- HL_errata.c
- HL_esm.c
- HL_gio.c
- HL_nmpu.c
- HL_notification.c
- HL_pinmux.c
- HL_rti.c
- HL_sys_core.asm
- HL_sys_dma.c
- HL_sys_intvecs.asm
- HL_sys_link.cmd
- HL_sys_main.c

```asm
;-----------------------------------------
; Enable interrupts - CPU IRQ
; SourceId : CORE_SourceId_026
; DesignId : CORE_DesignId_022
; Requirements: HL_CONQ_CORE_SR8

        .def _enable_IRQ_interrupt_
        .asmfunc

_enable_IRQ_interrupt_

        cpsie i
        bx    lr

        .endasmfunc
;-----------------------------------------
```

.def : C 언어의 함수 연결
.asmfunc : 어셈으로 했다.

Cpsie I => CPSR Register 의 I 비트인 IRQ 활성화 한다.

Bx lr => lr에 복귀주소 저장. bx는 복귀 주소로 돌아 가라(점프, branch)

참고 : 어셈 명령어 call 은 push 와 점프를 같이
bx는 점프만

- 결론적으로 ARM Core 의 IRQ를 CPSR 레지스터를 제어 해서 on 하고 토낌.

RTI 동작

```c
int main(void)
{
/* USER CODE BEGIN (3) */
    gioInit();
    rtiInit();

    //Real Time Interrupt 를 누가 처리 할지
    // RTI Compare0에서 발생하는 인터럽트 허용!!
    rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE0);    // 인터럽트를 구통화 시키세요

    // IRQ Interrupt 활성화
    // 참고로 언더바가 두개면 거의 건드리지 말라는 의미이다.
    // 어셈블리 명령어인 CPSIE I를 통해 CPSR Register 의 I 비트인 IRQ를 활성화 한다.
    // _disable_() 의 경우 CPSID I를 토해 IRQ를 비활성화 한다.
    _enable_IRQ_interrupt_();

    // Real Time Interrupt 카운터 시작!
    // 실제로는 RTI Counter Block 0 을 시작하여 카운팅을 진행
    rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0);

    for(;;);

/* USER CODE END */

    return 0;
}
```

```c
void rtiStartCounter(rtiBASE_t *rtiREG, uint32 counter)
{
/* USER CODE BEGIN (4) */
/* USER CODE END */

    rtiREG->GCTRL |= ((uint32)1U << (counter & 3U));

    /**   @note The function rtiInit has to be called before this function can be used.\n
    *           This function has to be executed in privileged mode.
    */

/* USER CODE BEGIN (5) */
/* USER CODE END */
}
```

17.3.1 RTI Global Control Register (RTIGCTRL)

The global control register starts/stops the counters and selects the signal compared with the timebase control circuit. This register is shown in Figure 17-12 and described in Table 17-2.

Figure 17-12. RTI Global Control Register (RTIGCTRL) [offset = 00]

31–20	19–16
Reserved	NTUSEL
R-0	R/WP-0

15	14–2	1	0
COS	Reserved	CNT1EN	CNT0EN
R/WP-0	R-0	R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; -*n* = value after reset

Table 17-2. RTI Global Control Register (RTIGCTRL) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	NTUSEL		Select NTU signal. These bits determine which NTU input signal is used as external timebase
		0h	NTU0
		5h	NTU1
		Ah	NTU2
		Fh	NTU3
		All other values	Tied to 0
15	COS		Continue on suspend. This bit determines if both counters are stopped when the device goes into halting debug mode or if they continue counting.
		0	Counters are stopped while in halting debug mode.
		1	Counters are running while in halting debug mode.
14-2	Reserved	0	Reads return 0. Writes have no effect.
1	CNT1EN		Counter 1 enable. This bit starts and stops counter block 1 (RTIUC1 and RTIFRC1).
		0	Counter block 1 is stopped.
		1	Counter block 1 is running.
0	CNT0EN		Counter 0 enable. This bit starts and stops counter block 0 (RTIUC0 and RTIFRC0).
		0	Counter block 0 is stopped.
		1	Counter block 0 is running.

Counter 가 0 이 들어 갔고
그게 3U와 & 가 되었으므로 위의 데이터시트상의 두 비트가 둘 다 0 이 된다.

결국 Counter 0 enable.

RTI 동작

```c
/* USER CODE BEGIN (4) */
// 인터럽트 핸들러 인데 weak 레퍼런스가 지정된 것이고, 원래 이것은 인터럽트 실행시에 실행 되는것인데
// weak 레퍼런스라고 지정되어 있으므로 이것이 수정을 해서 덮어 쓸 수 있는 것이 가능한 것이다.
// 이부분은 RTI Interrupt Handler 로 Timer 가 차면 구동됨
void rtiNotification(rtiBASE_t *rtiREG, uint32_t notification)
{
    // GPIO PortA의 4번을 토글(On/Off) 시킴
    gioToggleBit(gioPORTA, 4);
}
/* USER CODE END */
```

PORTA 의 4 bit

Clear / SET

RTI 를 Hal code 에서 1 초로 설정 했었었고,

해당 bit 의 값이 1초에 한번씩

Toggle 됨 으로써

LED 를 연결하게 되면 LED가 점멸하게 된다.

```c
void gioToggleBit(gioPORT_t *port, uint32 bit)
{
/* USER CODE BEGIN (10) */
/* USER CODE END */

    if ((port->DIN & (uint32)((uint32)1U << bit)) != 0U)
    {
        port->DCLR = (uint32)1U << bit;
    }
    else
    {
        port->DSET = (uint32)1U << bit;
    }
}
```