Xilinx Zynq FPGA, TI DSP, MCU 기반의 프로그래밍 및 회로 설계 전문가 과정

강사 – Innova Lee(이상훈)

gcccompil3r@gmail.com

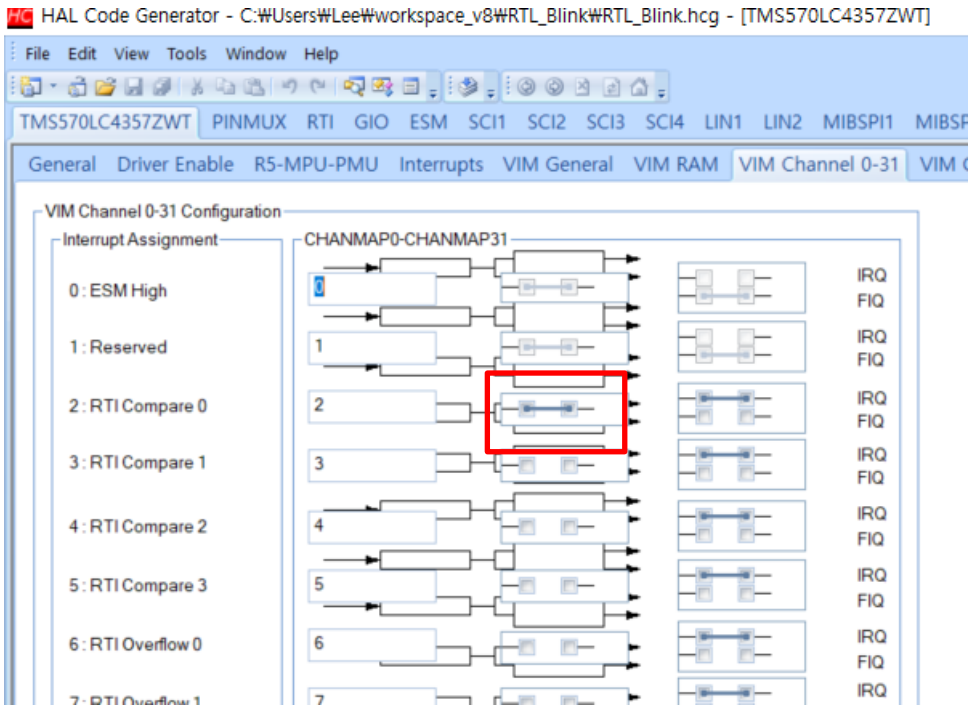
학생 – hoseong Lee(이호성)

hslee00001@naver.com

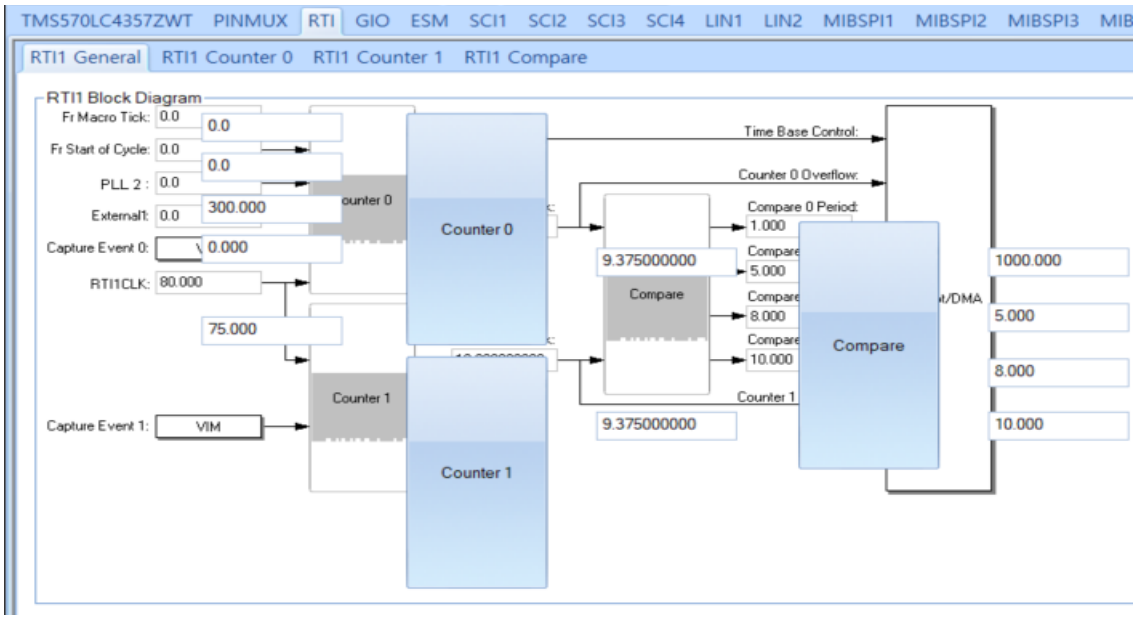
5월9일

CCS로 새프로젝트 만든후 HAL Code Generator로 새프로젝트 이름 같게해서 연결

1. VIM Channel 0-31 2: RTI Compare 0 : 연결 (Real Time Interrupt 실시간인터럽트)

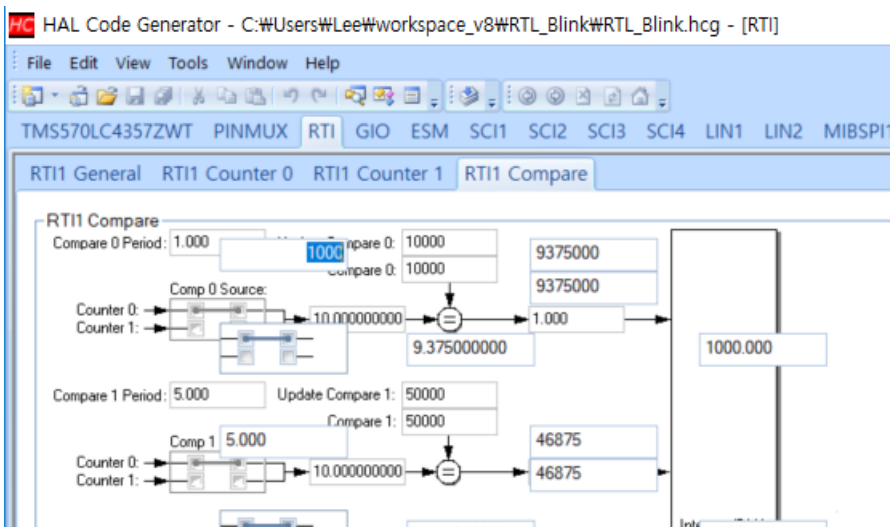


: RTI Compare 0 을 사용하겠다. IRQ or FIQ 어느 모드로 해도 상관없다. 이제 RTI를 본다.



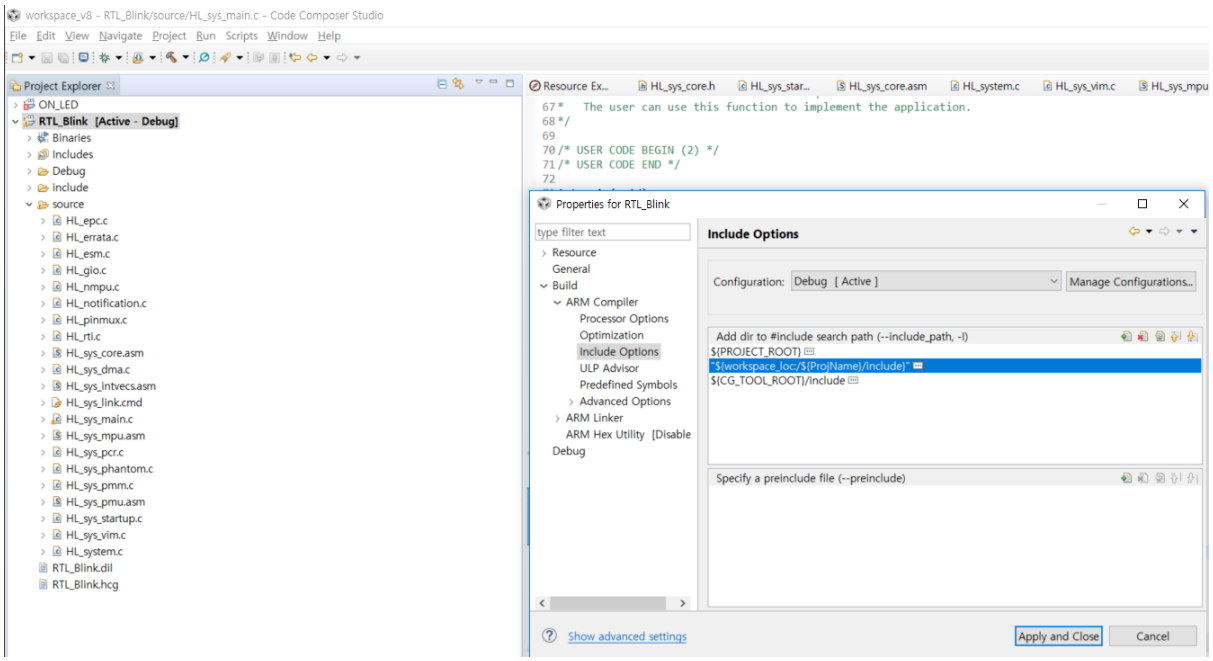
PLL2 (Network Time Unit 3) 300MHz 로 내보내고 있다.

2. RTI, Compare 0 Period : 1.000 -> 1000으로 1초를 만들어준다.



3. Generate code

다시 CCS 로 와서 workspace



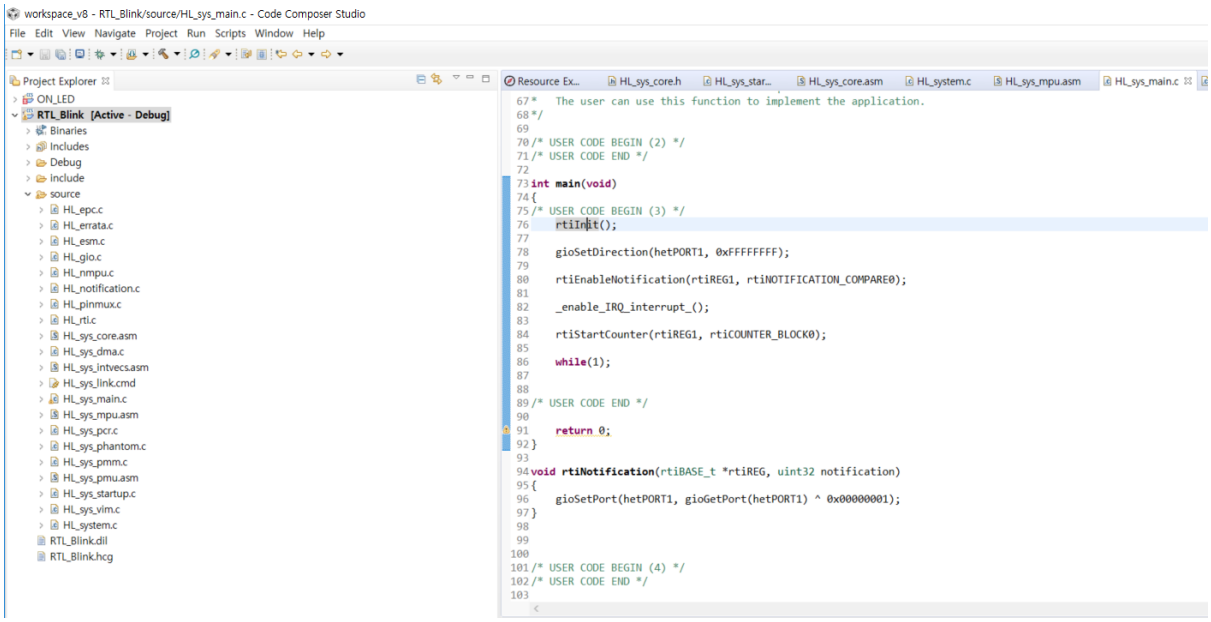
1초마다 인터럽트를 발생시켜 HetPORT1 (LED)를 껐다켰다 하는 코드

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

```
    rtiInit();

    gioSetDirection(hetPORT1, 0xFFFFFFFF);

    rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE0);

    _enable_IRQ_interrupt_();

    rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0);    // ( 0xFFFFFC00U, 0U)

    while(1);


/* USER CODE END */

    return 0;
}

void rtiNotification(rtiBASE_t *rtiREG, uint32 notification)
{

    gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0xAA0);   //#define hetPORT1
((gioPORT_t *)0xFFF7B84CU)  --> offset
}
```

.

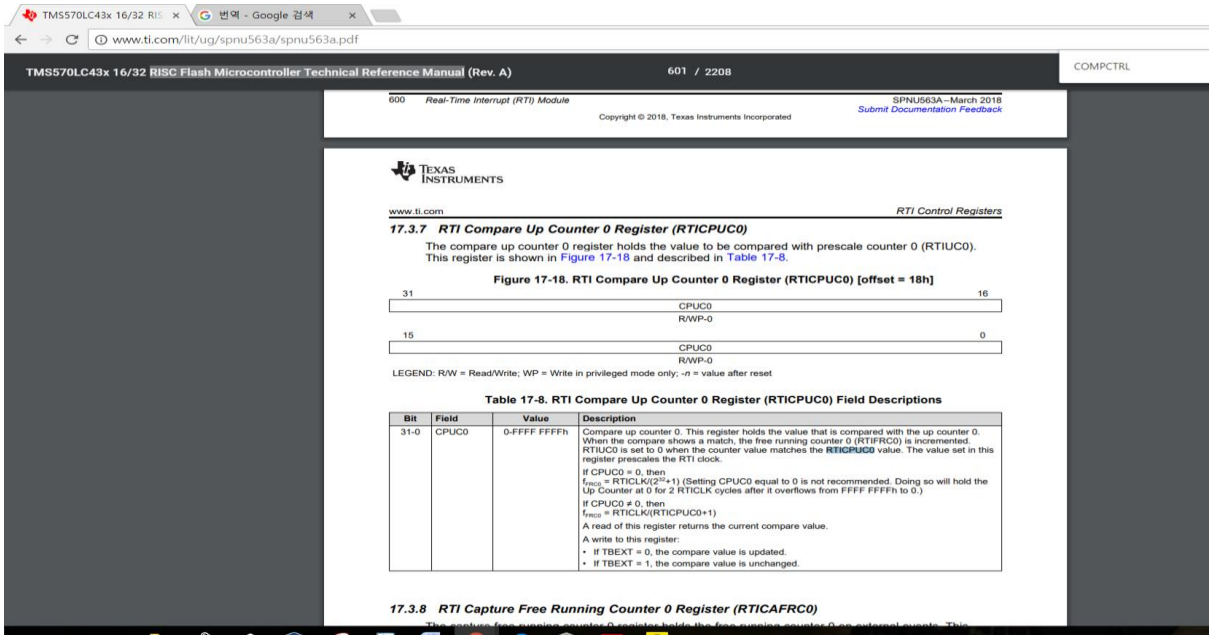
Main 분석코드 분석하기



1. rtiInit ()함수

```
Resource Ex...  HL_sys_core.h  HL_sys_star...  HL_sys_core.asm  HL_system.c  HL_sys_mpu.asm  HL_sys_ma
 69 void rtiInit(void)
 70 {
 71 /* USER CODE BEGIN (2) */
 72 /* USER CODE END */
 73     /** @b Initialize @b RTI1: */
 74
 75     /** - Setup NTU source, debug options and disable both counter blocks */
 76     rtiREG1->GCTRL = (uint32)((uint32)0x5U << 16U) | 0x00000000U;    // 101 0000 0000 0000 0000
 77
 78     /** - Setup timebase for free running counter 0 */
 79     rtiREG1->TBCTRL = 0x00000000U;
 80
 81     /** - Enable/Disable capture event sources for both counter blocks */
 82     rtiREG1->CAPCTRL = 0U | 0U;
 83
 84     /** - Setup input source compare 0-3 */
 85     rtiREG1->COMPCTRL = 0x00001000U | 0x00000100U | 0x00000000U | 0x00000000U;
 86
 87     /** - Reset up counter 0 */
 88     rtiREG1->CNT[0U].UCx = 0x00000000U;
 89
 90     /** - Reset free running counter 0 */
 91     rtiREG1->CNT[0U].FRCx = 0x00000000U;
 92
 93     /** - Setup up counter 0 compare value
 94     *      - 0x00000000: Divide by 2^32
 95     *      - 0x00000001-0xFFFFFFFF: Divide by (CPUC0 + 1)
 96     */
 97     rtiREG1->CNT[0U].CPUCx = 7U;
 98
 99     /** - Reset up counter 1 */
100     rtiREG1->CNT[1U].UCx = 0x00000000U;
101
102     /** - Reset free running counter 1 */
103     rtiREG1->CNT[1U].FRCx  = 0x00000000U;
104
105     /** - Setup up counter 1 compare value
106     *      - 0x00000000: Divide by 2^32
107     *      - 0x00000001-0xFFFFFFFF: Divide by (CPUC1 + 1)
108     */
109     rtiREG1->CNT[1U].CPUCx = 7U;
```

-100번째 RTIUC



TMS570LC43x 16/32 RISC Flash Microcontroller Technical Reference Manual (Rev. A) 601 / 2208

600 Real-Time Interrupt (RTI) Module SPNU563A–March 2018

Submit Documentation Feedback

www.ti.com RTI Control Registers

17.3.7 RTI Compare Up Counter 0 Register (RTICPUC0)

The compare up counter 0 register holds the value to be compared with prescale counter 0 (RTIUC0). This register is shown in Figure 17-18 and described in Table 17-8.

Figure 17-18. RTI Compare Up Counter 0 Register (RTICPUC0) [offset = 18h]

31	16
CPUC0	
R/WP-0	

15	0
CPUC0	
R/WP-0	

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

Table 17-8. RTI Compare Up Counter 0 Register (RTICPUC0) Field Descriptions

Bit	Field	Value	Description
31-0	CPUC0	0-FFFF FFFFh	Compare up counter 0. This register holds the value that is compared with the up counter 0. When the compare shows a match, the free running counter 0 (RTIFRC0) is incremented. RTIUC0 is set to 0 when the counter value matches the RTICPUC0 value. The value set in this register prescales the RTI clock. If CPUC0 = 0, then f_{RTC0} = RTICLK/(2^{32}+1) (Setting CPUC0 equal to 0 is not recommended. Doing so will hold the Up Counter at 0 for 2 RTICLK cycles after it overflows from FFFF FFFFh to 0.) If CPUC0 ≠ 0, then f_{RTC0} = RTICLK/(RTICPUC0+1) A read of this register returns the current compare value. A write to this register: • If TBEXT = 0, the compare value is updated. • If TBEXT = 1, the compare value is unchanged.

17.3.8 RTI Capture Free Running Counter 0 Register (RTICAFRC0)

-112번째줄 `rtiREG1->CMP[0U].COMPx = 9375000U;` → 여기서부터 녹음.파일

RTI (Real Time Interrupt)

17.3.15 RTI Compare 0 Register (RTICOMP0)

The compare 0 register holds the value to be compared with the counters. This register is shown in Figure 17-26 and described in Table 17-16.

Figure 17-26. RTI Compare 0 Register (RTICOMP0) [offset = 50h]

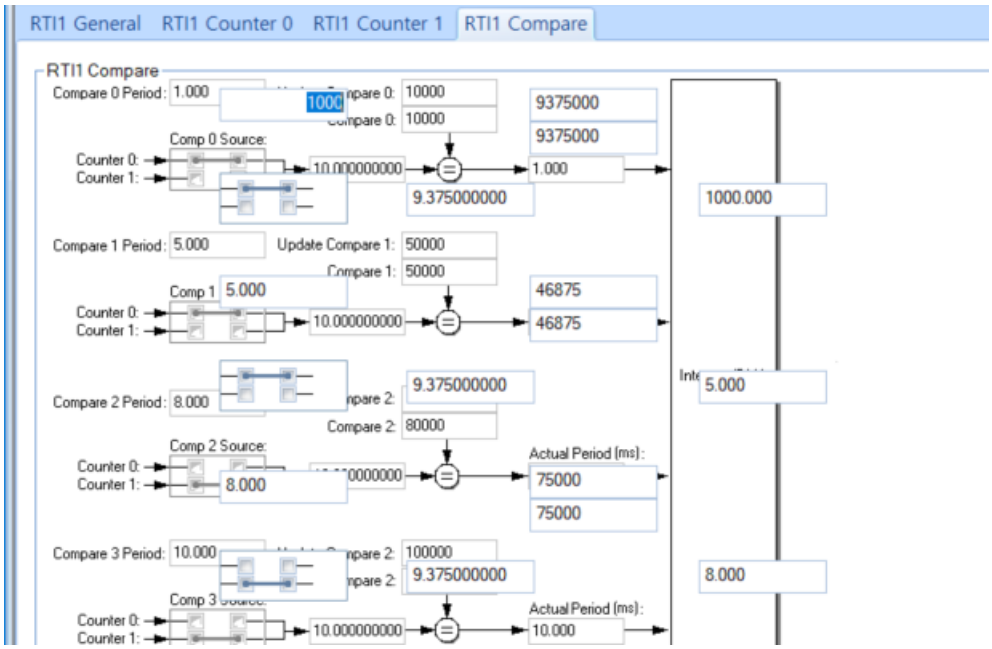
31	16
COMP0	
R/WP-0	

15	0
COMP0	
R/WP-0	

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

Table 17-16. RTI Compare 0 Register (RTICOMP0) Field Descriptions

Bit	Field	Value	Description
31-0	COMP0	0-FFFF FFFFh	Compare 0. This registers holds a value that is compared with the counter selected in the compare control logic. If RTIFRC0 or RTIFRC1, depending on the counter selected, matches the compare value, an interrupt is flagged. With this register it is also possible to initiate a DMA request. A read of this register will return the current compare value. A write to this register (in privileged mode only) will update the compare register with a new compare value.



T = 1/f = 1/9.375 = 1/9.375/(10^6) = 1/9375000= 0.106 * 10^(-6)

-115 rtiREG1->CMP[0U].UDCPx = 9375000U;

17.3.16 RTI Update Compare 0 Register (RTIUDCP0)

The update compare 0 register holds the value to be added to the compare register 0 value on a compare match. This register is shown in Figure 17-27 and described in Table 17-17.

Figure 17-27. RTI Update Compare 0 Register (RTIUDCP0) [offset = 54h]

31	16
UDCP0	
R/WP-0	

15	0
UDCP0	
R/WP-0	

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

Table 17-17. RTI Update Compare 0 Register (RTIUDCP0) Field Descriptions

Bit	Field	Value	Description
31-0	UDCP0	0-FFFF FFFFh	Update compare 0. This register holds a value that is added to the value in the compare 0 (RTICOMP0) register each time a compare matches. This function allows periodic interrupts to be generated without software intervention. A read of this register will return the value to be added to the RTICOMP0 register on the next compare match. A write to this register will provide a new update value.

-136번째줄 rtiREG1->INTFLAG = 0x0007000FU; //1,2,3,4 , 16,17,18

17.3.27 RTI Interrupt Flag Register (RTIINTFLAG)

The corresponding flags are set at every compare match of the RTIFRCx and RTICOMPx values, whether the interrupt is enabled or not. This register is shown in Figure 17-38 and described in Table 17-28.

Figure 17-38. RTI Interrupt Flag Register (RTIINTFLAG) [offset = 88h]

31	19	18	17	16
Reserved		OVL1INT	OVL0INT	TBINT
R-0		R/W1CP-0	R/W1CP-0	R/W1CP-0

15	4	3	2	1	0
Reserved		INT3	INT2	INT1	INT0
R-0		R/W1CP-0	R/W1CP-0	R/W1CP-0	R/W1CP-0

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset

Bit	Field	Value	Description
31-19	Reserved	0	Reads return 0. Writes have no effect.
18	OVL1INT		Free running counter 1 overflow interrupt flag. This bit determines if an interrupt is pending.
		0	*Read:* No interrupt is pending. *Write:* Bit is unchanged.
		1	*Read:* Interrupt is pending. *Write:* Bit is cleared to 0.
17	OVL0INT		Free running counter 0 overflow interrupt flag. This bit determines if an interrupt is pending.
		0	*Read:* No interrupt is pending. *Write:* Bit is unchanged.
		1	*Read:* Interrupt is pending. *Write:* Bit is cleared to 0.
16	TBINT		Timebase interrupt flag. This flag is set when the TBEXT bit is cleared by detection of a missing external clock edge. It will not be set by clearing TBEXT by software. It determines if an interrupt is pending.
		0	*Read:* No interrupt is pending. *Write:* Bit is unchanged.
		1	*Read:* Interrupt is pending. *Write:* Bit is cleared to 0.
15-4	Reserved	0	Reads return 0. Writes have no effect.
3	INT3		Interrupt flag 3. These bits determine if an interrupt due to a Compare 3 match is pending.
		0	*Read:* No interrupt is pending. *Write:* Bit is unchanged.
		1	*Read:* Interrupt is pending. *Write:* Bit is cleared to 0.
2	INT2		Interrupt flag 2. These bits determine if an interrupt due to a Compare 2 match is pending.
		0	*Read:* No interrupt is pending. *Write:* Bit is unchanged.
		1	*Read:* Interrupt is pending. *Write:* Bit is cleared to 0.
1	INT1		Interrupt flag 1. These bits determine if an interrupt due to a Compare 1 match is pending.
		0	*Read:* No interrupt is pending. *Write:* Bit is unchanged.
		1	*Read:* Interrupt is pending. *Write:* Bit is cleared to 0.

17,18 번째, Free running counter 0 overflow interrupt flag → 오타라고 생각하심 선생님

-139번째줄 `rtiREG1->CLEARINTENA = 0x00070F0FU;` 녹음 25분 // 0,1,2,3, 8,9,10,11, 16,17,18

17.3.26 RTI Clear Interrupt Enable Register (RTICLEARINTENA)

This register prevents the necessity of a read-modify-write operation if a particular interrupt should be disabled. This register is shown in Figure 17-37 and described in Table 17-27.

Figure 17-37. RTI Clear Interrupt Control Register (RTICLEARINTENA) [offset = 84h]

31					24
Reserved					
R-0					
23		**19**	**18**	**17**	**16**
Reserved			CLEAROVL1INT	CLEAROVL0INT	CLEARTBINT
R-0			R/WP-0	R/WP-0	R/WP-0
15	**12**	**11**	**10**	**9**	**8**
Reserved		CLEARDMA3	CLEARDMA2	CLEARDMA1	CLEARDMA0
R-0		R/WP-0	R/WP-0	R/WP-0	R/WP-0
7	**4**	**3**	**2**	**1**	**0**
Reserved		CLEARINT3	CLEARINT2	CLEARINT1	CLEARINT0
R-0		R/WP-0	R/WP-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 17-28. RTI Interrupt Flag Register (RTIINTFLAG) Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reads return 0. Writes have no effect.
18	OVL1INT		Free running counter 1 overflow interrupt flag. This bit determines if an interrupt is pending.
		0	*Read:* No interrupt is pending. *Write:* Bit is unchanged.
		1	*Read:* Interrupt is pending. *Write:* Bit is cleared to 0.
17	OVL0INT		Free running counter 0 overflow interrupt flag. This bit determines if an interrupt is pending.
		0	*Read:* No interrupt is pending. *Write:* Bit is unchanged.
		1	*Read:* Interrupt is pending. *Write:* Bit is cleared to 0.
16	TBINT		Timebase interrupt flag. This flag is set when the TBEXT bit is cleared by detection of a missing external clock edge. It will not be set by clearing TBEXT by software. It determines if an interrupt is pending.
		0	*Read:* No interrupt is pending. *Write:* Bit is unchanged.
		1	*Read:* Interrupt is pending. *Write:* Bit is cleared to 0.
15-4	Reserved	0	Reads return 0. Writes have no effect.
3	INT3		Interrupt flag 3. These bits determine if an interrupt due to a Compare 3 match is pending.
		0	*Read:* No interrupt is pending. *Write:* Bit is unchanged.
		1	*Read:* Interrupt is pending. *Write:* Bit is cleared to 0.
2	INT2		Interrupt flag 2. These bits determine if an interrupt due to a Compare 2 match is pending.
		0	*Read:* No interrupt is pending. *Write:* Bit is unchanged.
		1	*Read:* Interrupt is pending. *Write:* Bit is cleared to 0.
1	INT1		Interrupt flag 1. These bits determine if an interrupt due to a Compare 1 match is pending.
		0	*Read:* No interrupt is pending. *Write:* Bit is unchanged.
		1	*Read:* Interrupt is pending. *Write:* Bit is cleared to 0.

Table 17-28. RTI Interrupt Flag Register (RTIINTFLAG) Field Descriptions (continued)

Bit	Field	Value	Description
0	INT0		Interrupt flag 0. These bits determine if an interrupt due to a Compare 0 match is pending.
		0	*Read:* No interrupt is pending. *Write:* Bit is unchanged.
		1	*Read:* Interrupt is pending. *Write:* Bit is cleared to 0.

rtiInit() 함수가 끝났다 – 녹음 30분

이제 main에 돌아와서

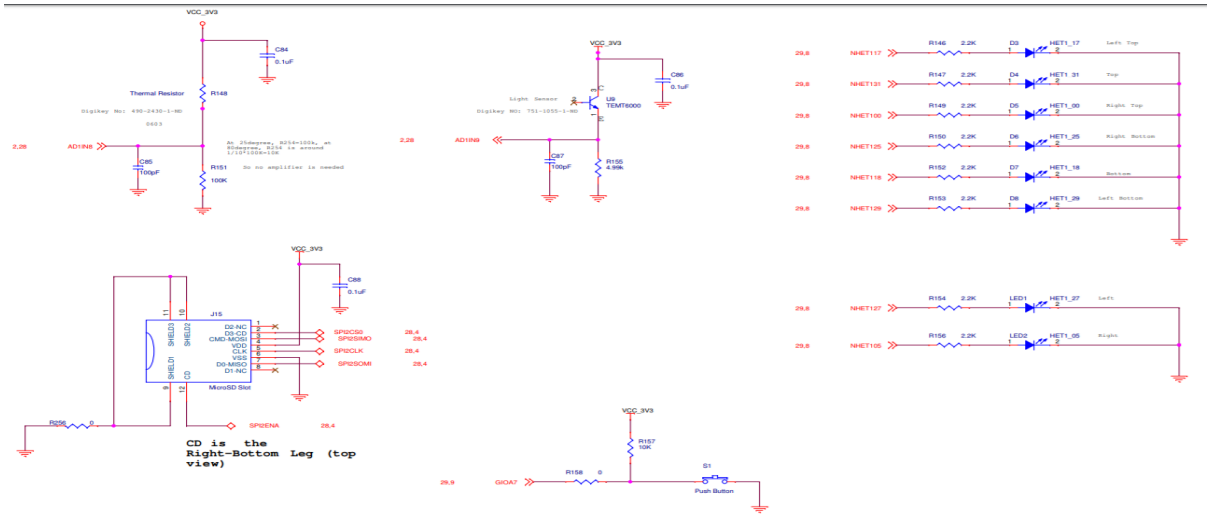
```
72
73 int main(void)
74 {
75 /* USER CODE BEGIN (3) */
76     rtiInit();
77
78     gioSetDirection(hetPORT1, 0xFFFFFFFF);
79
80     rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE0);
81
82     _enable_IRQ_interrupt_();
83
84     rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0);
85
86     while(1);
87
88
89 /* USER CODE END */
90
91     return 0;
```

HetPORT1 로들어가본다.

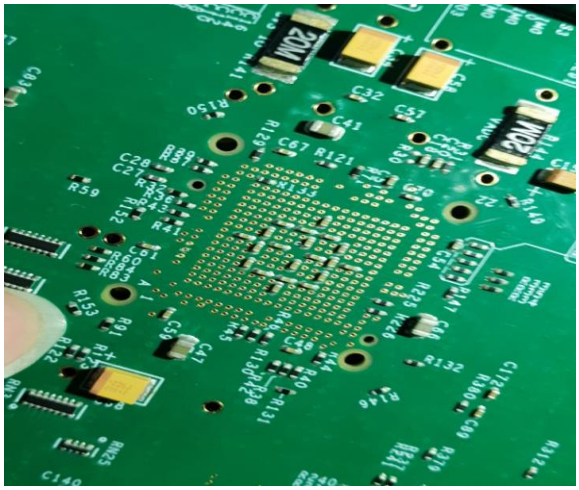
```
162 */
163 #define hetPORT1 ((gioPORT_t *)0xFFF7B84CU)
164
```

B84C 가안나온다. 그럼 FFF7을 찾아본다. 데이터시트.

Datasheet -> SPNR043 -> TMS570LC43_SCHEMATICS_REVE 회로도를 본다.

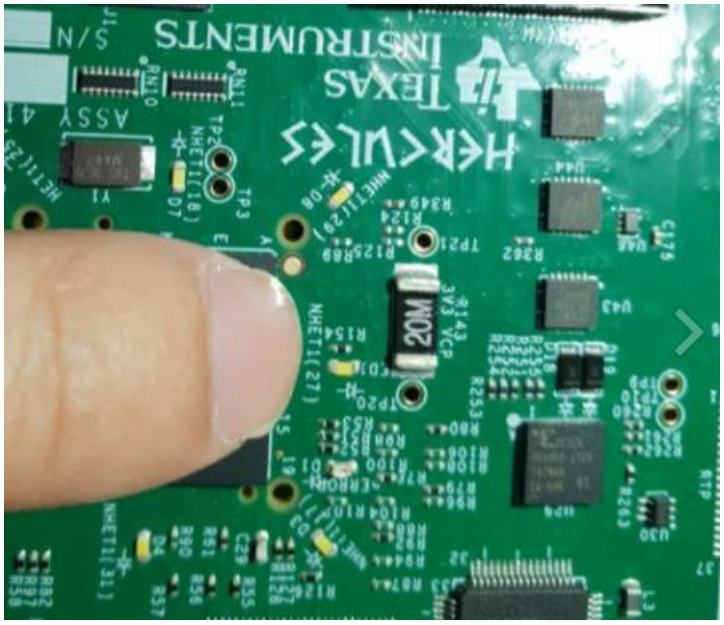


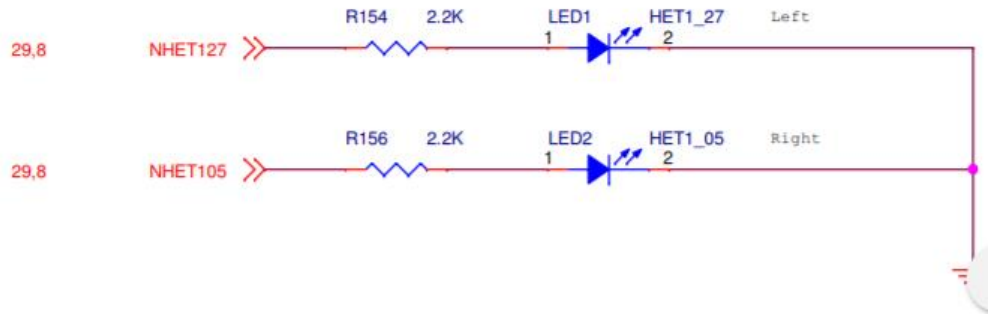
-확인

➔ R153 

실험에서 HET1_00 Right Top 이 밝혀진 것을 볼 수 있었다.

GPIO 로 쓰이고 있다는 것을 알 수 있다.





➔ 75 MHz -> 8분주 = 9.735 Mhz -> 1/9.75 = 0.106 *10^-6 * 9.735 = 1m (임 1ms 로 계산하면 편함으로) 여기다 우리는 1000을 곱해줘서 1초를 만들어 줬었다.

_enable_IRQ_interrupt 1시간 50분

```
73 int main(void)
74 {
75 /* USER CODE BEGIN (3) */
76     rtiInit();
77
78     gioSetDirection(hetPORT1, 0xFFFFFFFF);
79
80     rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE0);
81
82     _enable_IRQ_interrupt_();
83
84     rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0);
85
86     while(1);
87
88
89 /* USER CODE END */
90
91     return 0;
92 }
```

```
513 ; Requirements: HL_CONQ_CORE_SR8
514
515         .def _enable_IRQ_interrupt_
516         .asmfunc
517
518 _enable_IRQ_interrupt_
519
520         cpsie i
521         bx    lr
522
523         .endasmfunc
```

-84번째줄 main rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0); // (0xFFFFFC00U, 0U)

```
73 int main(void)
74 {
75 /* USER CODE BEGIN (3) */
76     rtiInit();
77
78     gioSetDirection(hetPORT1, 0xFFFFFFFF);
79
80     rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE0);
81
82     _enable_IRQ_interrupt_();
83
84     rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0);    // ( 0xFFFFFC00U, 0U)
85
86     while(1);
87
88
89 /* USER CODE END */
90
91     return 0;
92 }
```

```
163 void rtiStartCounter(rtiBASE_t *rtiREG, uint32 counter)
164 {
165 /* USER CODE BEGIN (4) */
166 /* USER CODE END */
167
168     rtiREG->GCTRL |= ((uint32)1U << (counter & 3U));
169
170     /**   @note The function rtiInit has to be called before this function can be used.\n
171     *           This function has to be executed in privileged mode.
172     */
173
174 /* USER CODE BEGIN (5) */
175 /* USER CODE END */
```

*전체동작 - 2시간

```
73 int main(void)
74 {
75 /* USER CODE BEGIN (3) */
76     rtiInit();
77
78     gioSetDirection(hetPORT1, 0xFFFFFFFF);
79
80     rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE0);
81
82     _enable_IRQ_interrupt_();
83
84     rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0);    // ( 0xFFFFFC00U, 0U)
85
86     while(1);
87
88
89 /* USER CODE END */
90
91     return 0;
92 }
93
94 void rtiNotification(rtiBASE_t *rtiREG, uint32 notification)
95 {
96     gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0x00000001);
97 }
98
```

Rtinotification -> 은 인터럽트가 발생하면 1초가지나면 동작함.

```
354 uint32 gioGetPort(gioPORT_t *port)
355 {
356 /* USER CODE BEGIN (9) */
357 /* USER CODE END */
358
359     return port->DIN;
360 }
```

gioGetPort :지금 이 포트에 들어와있는 신호가 있는지 없는지 보겠다.

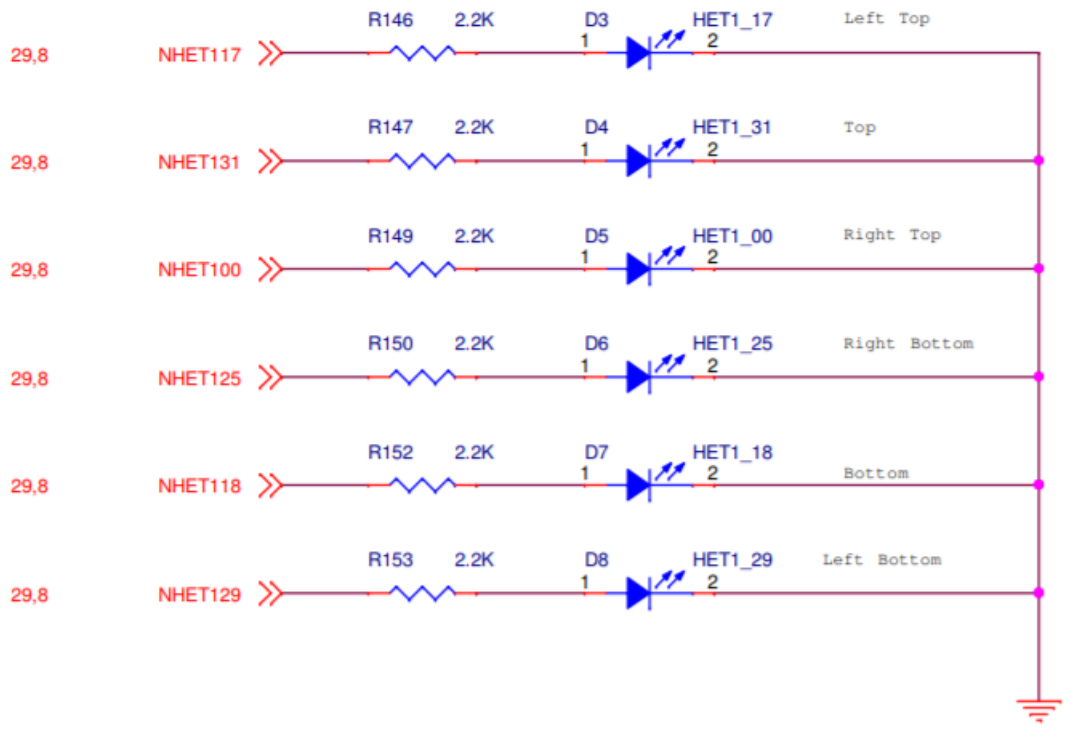
hetPORT1 : hetPORT1번만 가지고 제어를 해야한다.

```
163 #define hetPORT1 ((gioPORT_t *)0xFFF7B84CU)
164
```

QUIZ 1. N2HET 을 활용해 LED 8 개 전부 키기

```
        1111 1111 1111 0111 1011 1000 0100 1101
{
het1     1000 0000 0000 0000 0000 0000 0000 0001
       31 29 27 25     18,17           5
        1010 1010 0000 0110 0000 0000 0010 0001 = a a 0 6 0 0 2 1
value    0111 1111 1111 0111 1011 1000 0100 1100

          7   f    f    7    b    8   4    a
```

#define hetPORT1 ((gioPORT_t *)0xFFF7B84CU) 0Xfff7b85c 는 hetPORT1의 주소이다.



```
void rtiNotification(rtiBASE_t *rtiREG, uint32 notification)
{
    gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0x80000001);   //#define hetPORT1 ((gioPORT_t *)0xFFF7B84CU)  --> offset
}
```

GioGetPort(hetPORT1)는 모두 0x00000000 ^ 0x80000001 -> NHET100, NHET131 비트만 켜놓은 상태이다. 모두 키기 위해서는 31 29 25 18,17 5 키면됀다.

즉 0x 00000000 ^ a a 0 6 0 0 2 1 하면 됀다.

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

    gioSetDirection(hetPORT1, 0xFFFFFFFF);

    rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE0);

    _enable_IRQ_interrupt_();

    rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0);    // ( 0xFFFFFC00U, 0U)

    while(1);


/* USER CODE END */

    return 0;
}

void rtiNotification(rtiBASE_t *rtiREG, uint32 notification)
{

    gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0xAA0);   //#define hetPORT1
((gioPORT_t *)0xFFF7B84CU)  --> offset
}
```