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

#50

2018.05.09

강사:Innova Lee(이 상훈)

학생: 김시윤

과제

RTI_Blink (Real Time Interrupt Blink)

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

    gioSetDirection(hetPORT1, 0xFFFFFFFF);

    rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE0);

    _enable_IRQ_interrupt_();

    rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0);
            srand(time(NULL));

    while(1);
/* USER CODE END */

    return 0;
}
```

main.c 에 있는 메인함수 부터 살펴본다.

함수들만 봤을때 real time interrupt 를 초기화 하고 핀 방향 설정하고 rti compare 활성화 하고 인터럽트를 활성화 시키고 마지막에 카운터를 활성화 시킨다 이때부터 타이머가 동작하게된다.

우서 rtiInit()함수를 살펴본다.

```
void rtiInit(void)
{
/* USER CODE BEGIN (2) */
/* USER CODE END */
    /** @b Initialize @b RTI1: */

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

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

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

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

    /** - Reset up counter 0 */
    rtiREG1->CNT[0U].UCx = 0x00000000U;

    /** - Reset free running counter 0 */
    rtiREG1->CNT[0U].FRCx = 0x00000000U;

    /** - Setup up counter 0 compare value
    *     - 0x00000000: Divide by 2^32
    *     - 0x00000001-0xFFFFFFFF: Divide by (CPUC0 + 1)
    */
    rtiREG1->CNT[0U].CPUCx = 7U;

    /** - Reset up counter 1 */
    rtiREG1->CNT[1U].UCx = 0x00000000U;

    /** - Reset free running counter 1 */
    rtiREG1->CNT[1U].FRCx  = 0x00000000U;

    /** - Setup up counter 1 compare value
    *     - 0x00000000: Divide by 2^32
    *     - 0x00000001-0xFFFFFFFF: Divide by (CPUC1 + 1)
    */
    rtiREG1->CNT[1U].CPUCx = 7U;

    /** - Setup compare 0 value. This value is compared with selected free runni
    rtiREG1->CMP[0U].COMPx = 9375000U;

    /** - Setup update compare 0 value. This value is added to the compare 0 val
    rtiREG1->CMP[0U].UDCPx = 9375000U;

    /** - Setup compare 1 value. This value is compared with selected free runni
    rtiREG1->CMP[1U].COMPx = 46875U;

    /** - Setup update compare 1 value. This value is added to the compare 1 val
    rtiREG1->CMP[1U].UDCPx = 46875U;

    /** - Setup compare 2 value. This value is compared with selected free runni
    rtiREG1->CMP[2U].COMPx = 75000U;

    /** - Setup update compare 2 value. This value is added to the compare 2 val
    rtiREG1->CMP[2U].UDCPx = 75000U;
```

```
    rtiREG1->CMP[3U].COMPx = 93750U;

    /** - Setup update compare 3 value. Thi
    rtiREG1->CMP[3U].UDCPx = 93750U;

    /** - Clear all pending interrupts */
    rtiREG1->INTFLAG = 0x0007000FU;

    /** - Disable all interrupts */
    rtiREG1->CLEARINTENA = 0x00070F0FU;

    /**   @note This function has to be cal
    *           This function has to be exe
    *           This function does not star
    */

/* USER CODE BEGIN (3) */
/* USER CODE END */
}
```

```
 rtiREG1->GCTRL = (uint32)((uint32)0x5U << 16U) |
0x00000000U
```

19~16 번 비트가 5 이고 나머지는 0 이다.

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.

즉 외부 타임 베이스를 NTU1 을 사용하고 Counters are stopped while in halting debug mode 디버그 모드일 때 동안 카운터를 멈추고 커운터 블록 1 과 2 를 멈춘상태로 설정한다.

RTIUC0 clocks RTIFRC0.

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
Reserved	
R-0	

7	2	1	0
Reserved		INC	TBEXT
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.

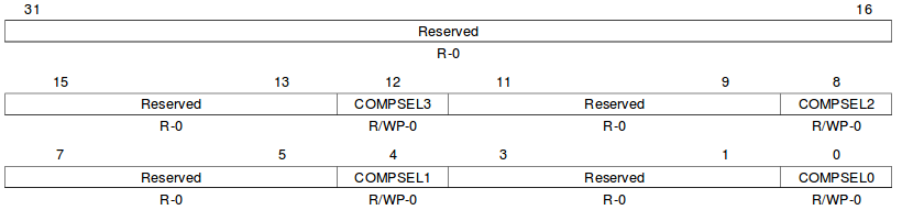
CAPCTRL 캡쳐 인터럽트 우리는 안써서 필요없음.

COMPCTRL= 0x00001000U | 0x00000100U | 0x00000000U | 0x00000000U;
8 번 12 번 비트 1 나머지는 0 으로 셋팅

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.

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



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

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.

`COMPSEL3= Value` will be compared with RTIFRC1

`COMPSEL2 = Value` will be compared with RTIFRC1.

`COMPSEL1 = Value` will be compared with RTIFRC0.
COMPSEL0= Value will be compared with RTIFRC0
comp counter 을 결해준다 우리는 compcounter 0 을 사용한다. 이 0 이 비교할 레지스터의 벨류를 물고있다. (RTIFRC0)

RTIFRC0 레지스터는 counter 0 번에 대한 현재값을 물고있다. 읽은 카운터의 현재 값을 반환한다. 시간을 카운팅 해주는 레지스터이다.

이 레지스터를 그 밑에 소스코드에서 초기화 시킨다.

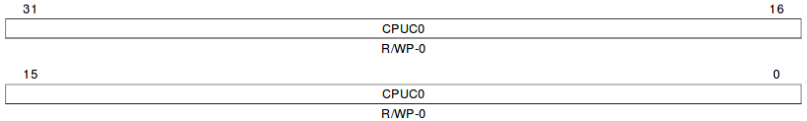
```
/** - Reset up counter 0 */
rtiREG1->CNT[0U].UCx = 0x00000000U;

/** - Reset free running counter 0 */
rtiREG1->CNT[0U].FRCx = 0x00000000U;
```

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]



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_{FRC0} = 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_{FRC0} = 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.

업 카운터 0 번이랑 비교된다. 비교 할라는 값이랑 카운터 값이 일치하면 이 값이 증가한다. RTIUC 0 이 카운터값과 RTICT0 과 일치하면 0

현재 CPUC0 은 7 이기 때문에 RTICLK 를 8 분주 한다는 뜻이다.
밑에 컴페이 벨류랑 업데이트 컴페어 밸류를 1000ms 로 설정해준 주파수 값을 설정해준다.

```
gioSetDirection(hetPORT1, 0xFFFFFFFF);
```

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

23.4.18 NHET Direction Register (HETDIR)

N2HET1: offset = FFF7 B84Ch; **N2HET2:** offset = FFF7 B94Ch

Figure 23-73. N2HET Direction Register (HETDIR)

31		16
HETDIR		
R/W-0		

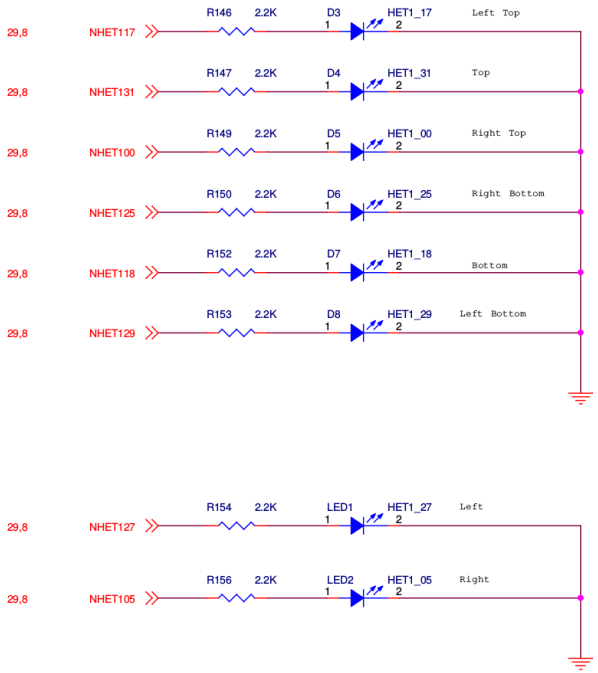
15		0
HETDIR		
R/W-0		

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

Table 23-34. N2HET Direction Register (HETDIR) Field Descriptions

Bit	Field	Value	Description
31-0	HETDIR[n]		Data direction of NHET pins
		0	Pin HET[n] is an input (and its output buffer is tristated).
		1	Pin HET[n] is an output.

NHET 핀의 데이터 방향을 설정 전체 1 로 설정했으므로 방향은 전체 출력




```
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 */
}
```

notification 은 1 로 들어옴
INTFLAG = 1
SETINTENA = 1
데이터 시트를 보면 0 으로 초기화 와 compare 0 활성화 라는 걸 알 수있다.

```
_enable_IRQ_interrupt_

        cpsie i
        bx    lr

        .endasmfunc
```

인터럽트 irq 동작시키겠다는 권한 설정

```
 rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0);
```

카운터 0 번을 활성화시킨다. 이때부터 타이머 동작

컴페어 벨류가 같을때 마다 인터럽트가 발생한다. 그 인터럽트를 서비스 루틴의 함수는 다음과 같다.

```
void rtiNotification(rtiBASE_t *rtiREG, uint32 notification)
```

우리는 타이머 1 초가 지날때 마다 이 서비스 루틴을 방문하게 하였다. 서비스 루틴이 연결되는 구간을 제대로 찾지는 못했는데 일단 알아낸것만 써보도록 한다.

```
#pragma WEAK(rtiNotification)

/** @fn void rtiNotification(uint32 notification)
*   @brief Notification of RTI module
*   @param[in] notification Select notification of RTI
module:
*              - rtiNOTIFICATION_COMPARE0: RTI compare 0
notification
*              - rtiNOTIFICATION_COMPARE1: RTI compare 1
notification
*              - rtiNOTIFICATION_COMPARE2: RTI compare 2
notification
*              - rtiNOTIFICATION_COMPARE3: RTI compare 3
notification
*              - rtiNOTIFICATION_TIMEBASE: RTI Timebase
notification
*              - rtiNOTIFICATION_COUNTER0: RTI counter 0
overflow notification
*              - rtiNOTIFICATION_COUNTER1: RTI counter 1
overflow notification
*
*   @note This function has to be provide by the user.
*/
```

위와 같은 상황일때 서비스 루틴을 간다고 되있는데 이걸 연결해주는 로직은 저걸로 부족할거같은데 더이상 찾지 못하였다.

```
        unsigned int val = 0xAA060021;
        unsigned int a[9] = {0,5,25,18,29,27,17,31};

        i = rand() % 9;
        if(t_flag == 0)
        {
            t_flag = 1;
            gioSetPort(hetPORT1, gioGetPort(hetPORT1) &
~(val));

        }
        else{
            gioSetPort(hetPORT1, gioGetPort(hetPORT1) |
1<<a[i]);
            t_flag = 0;
        }
```

랜덤일때의 로직

뱀꼬리 로직

```
        unsigned int a[9] = {0,5,25,18,29,27,17,31};

        i+=1;
        if(i>8)
        {i =0;
        }
            gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^
(1<<a[i]));
```

전체 켰을때의 로직

```
        unsigned int val = 0xAA060021;

            gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^val);
```