# TI DSP,MCU 및 Xilinux Zynq FPGA

# 프로그래밍 전문가 과정

| | |
|---|---|
| 이름 | 문지희 |
| 학생 이메일 | mjh8127@naver.com |
| 날짜 | 2018/5/9 |
| 수업일수 | 50 일차 |
| 담당강사 | Innova Lee(이상훈) |
| 강사 이메일 | gcccompil3r@gmail.com |

## 목차

1. RTI

2. 아두이노를 이용한 서보모터 제어

3. Uniform Circular Montion (등속도 운동)

# RTI

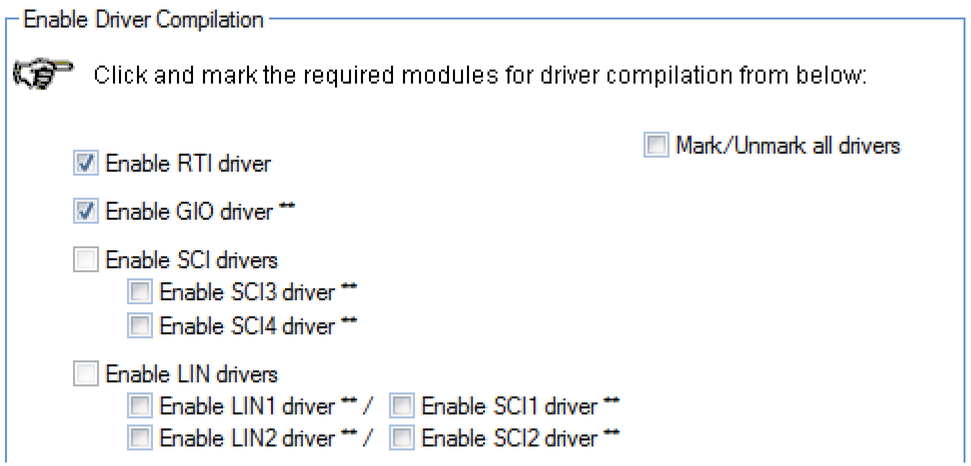
## [CCS]

File - New - CCS Project 로 RT_Blink 라는 이름으로 프로젝트 생성

## [HALcogen]

기존에 생성한 것처럼 프로젝트 생성.

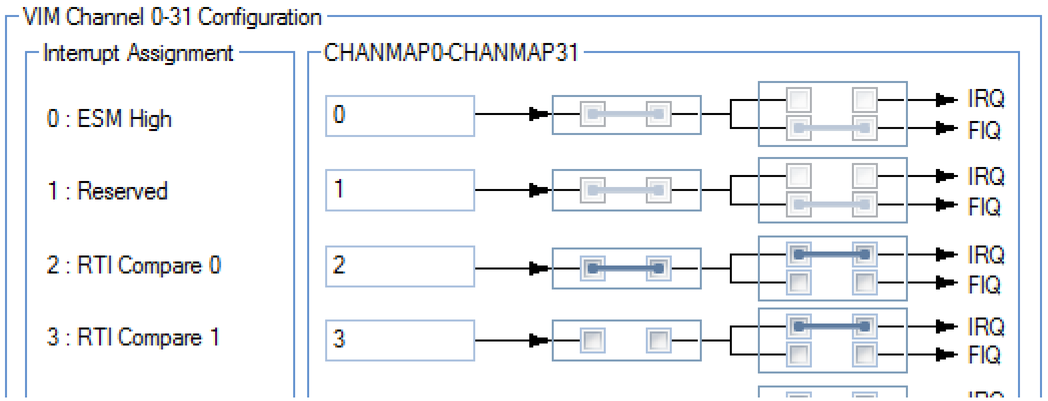
설정


Enable Driver Compilation

Click and mark the required modules for driver compilation from below:

- [ ] Mark/Unmark all drivers
- [✓] Enable RTI driver
- [✓] Enable GIO driver **
- [ ] Enable SCI drivers
  - [ ] Enable SCI3 driver **
  - [ ] Enable SCI4 driver **
- [ ] Enable LIN drivers
  - [ ] Enable LIN1 driver ** / [ ] Enable SCI1 driver **
  - [ ] Enable LIN2 driver ** / [ ] Enable SCI2 driver **

Enable RTI driver 와 Enable GIO driver 만 클릭


VIM Channel 0-31 Configuration

Interrupt Assignment

0 : ESM High

1 : Reserved

2 : RTI Compare 0

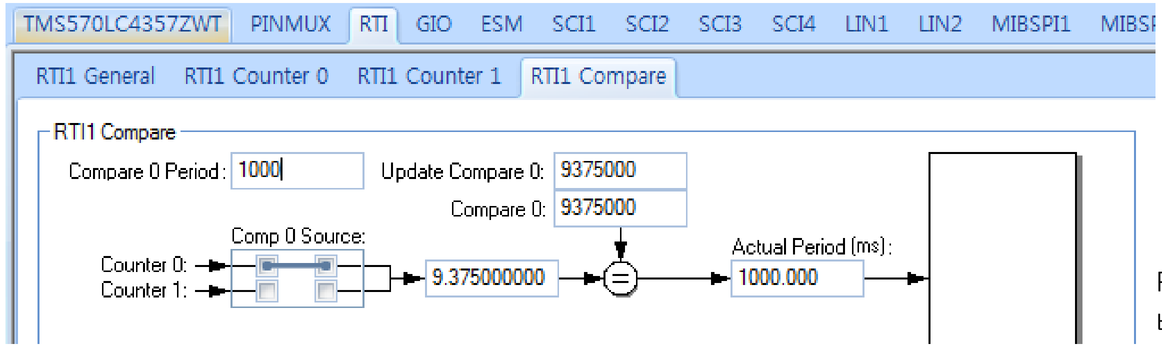
3 : RTI Compare 1

CHANMAP0-CHANMAP31

| CHANMAP | |
|---|---|
| 0 | IRQ / FIQ |
| 1 | IRQ / FIQ |
| 2 | IRQ / FIQ |
| 3 | IRQ / FIQ |

TMS570LC4357ZWT - VIM Channel 0-31 에서 사진과 같이 체크함

2 : RTI Compare 0 의 네모 박스 체크



RTI1 Compare

Compare 0 Period: 1000

Update Compare 0: 9375000

Compare 0: 9375000

Comp 0 Source:

Counter 0:

Counter 1:

9.375000000

Actual Period (ms): 1000.000

RTI - RTI Compare 에서 Compare 0 Period 를 1000 으로 바꾸기

File - Generate Code 하여 코드생성

**[CCS]**

Alt + Enter 를 눌러 properties - Build - Include Options 에서 Add dir to #incldue search path 를 추가하여 “${workspace_loc:/${ProjName}/include}” 를 입력하고 Apply and Close 한다.

HL_sys_main.c 파일에서 아래의 소스코드 작성

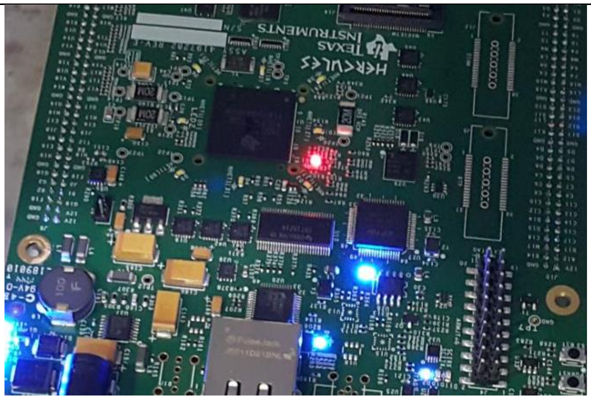
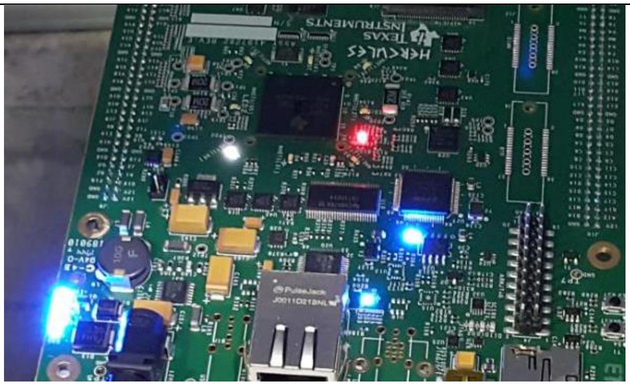
```
#include "HL_sys_common.h"
#include "HL_system.h"
#include "HL_mibspi.h"
#include "HL_sys_core.h"
#include "HL_esm.h"
#include "HL_rti.h"
#include "HL_gio.h"
#include "HL_het.h"

int main(void)
{

    rtiInit();
    gioSetDirection(hetPORT1, 0xFFFFFFFF);
    rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE0);
    _enable_IRQ_interrupt_();
    rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0);

    while(1);

    return 0;
}
```

소스코드 작성 후 컴파일하기.

이후 main 함수 아래의 소스코드 추가

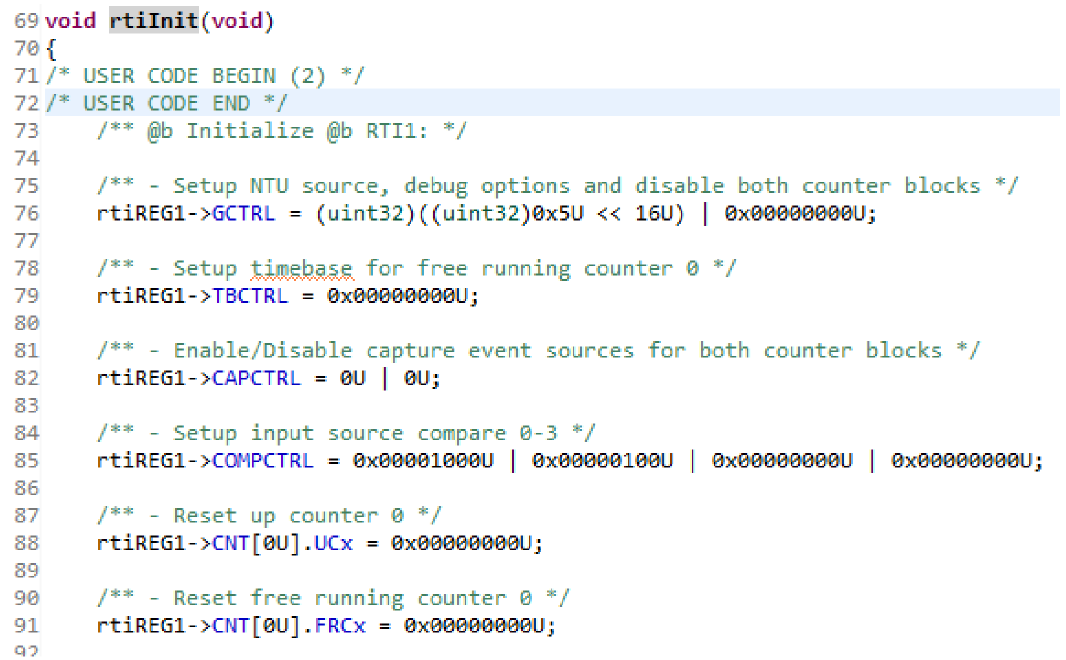
```
void rtiNotification(rtiBASE_t *rtiREG, uint32 notification)
{
      gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0x00000001);
}
```

**결과**

| | |
|---|---|
|  전 |  후 |

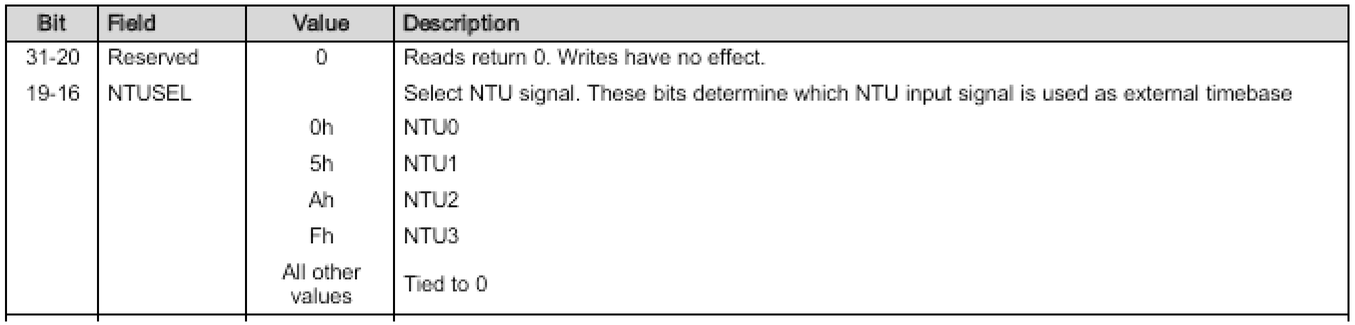
## 해석

rtInit(1)



```
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;
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
```

: 76 GCTRL



| 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 |

NTU1 신호가 선택된다.

NTU 입력신호는 외부의 시간을 기초로 사용한다.

: 79 TBCTRL

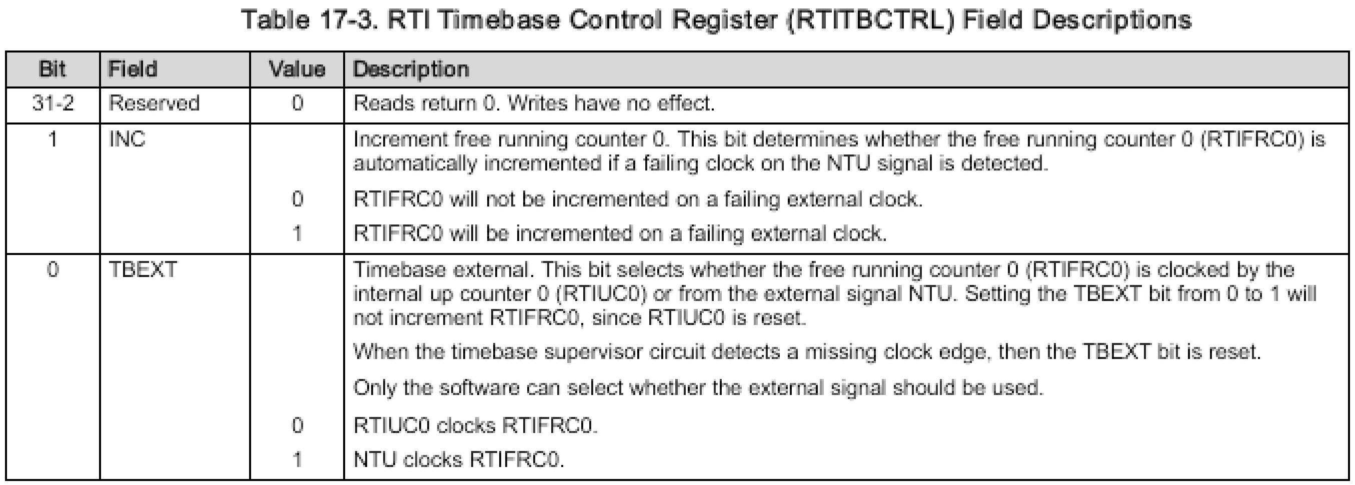


**Table 17-3. RTI Timebase Control Register (RTITBCTRL) Field Descriptions**

| 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. |

RTIFRC0 가 0 이면 증가하지 않는다.

RTIFRC0 을 사용할 것인지 NTU 를 사용할 것인지 결정하는 비트.

0 이면 RTIFRC0 는 RTIUC0 를 사용한다.

1 이면 NTU 를 사용

:82 CAPCTRL

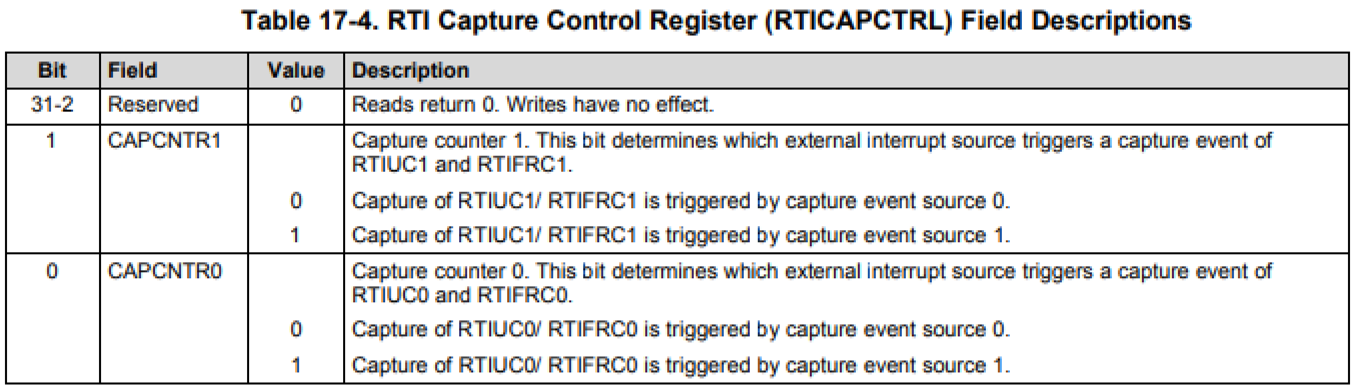


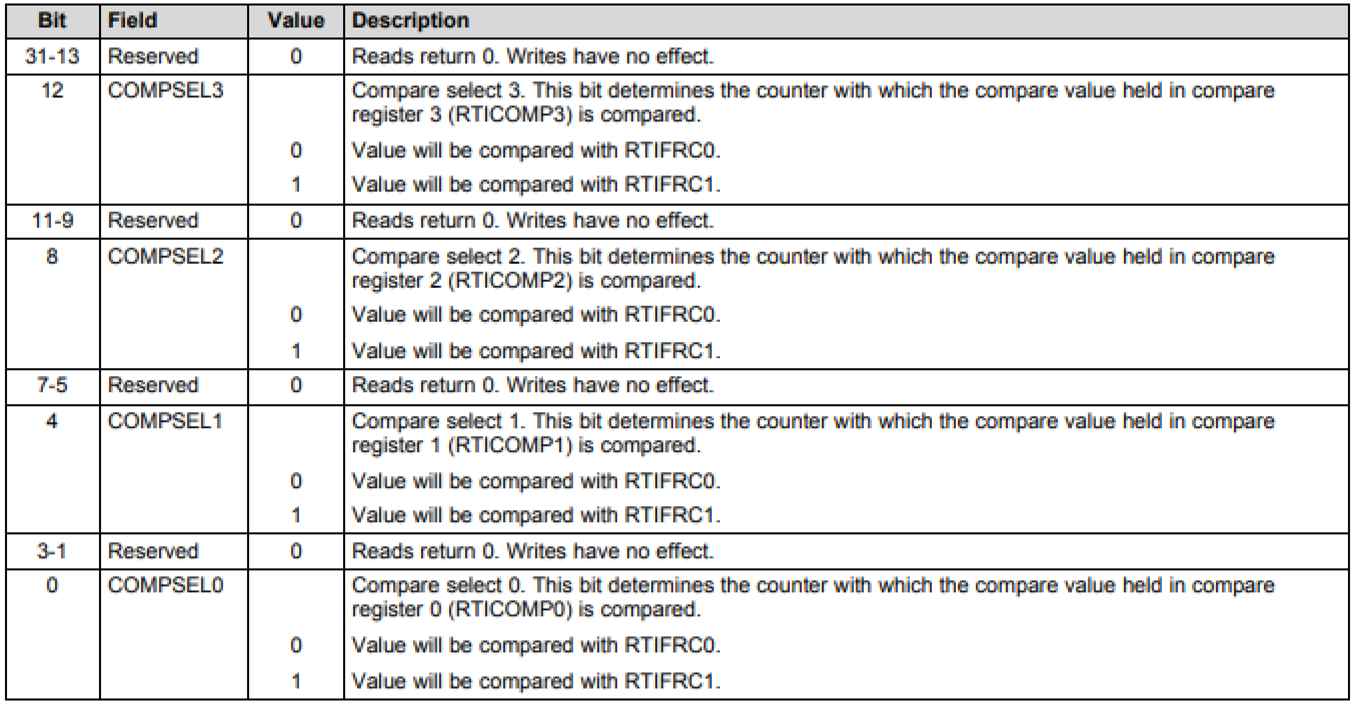
**Table 17-4. RTI Capture Control Register (RTICAPCTRL) Field Descriptions**

| 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. |

이 비트는 외부 인터럽트 소스 트리거의 RTIUC0 와 RTIFRC0 의 이벤트를 발견하면 인터럽트를 일으킴. 0 이므로 이벤트 소스 0 에 의해 RTIUC0/RTIFRC0, RTIUC1/RTIFRC1 가 실행됨.

:85 COMPCTRL 4,8

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. |

4 번과 8 번비트가 1 이고 0 번비트가 0 으로 고정이므로

RTICOMP2/1/0 레지스터가 비교 값과 카운터 값을 비교한다

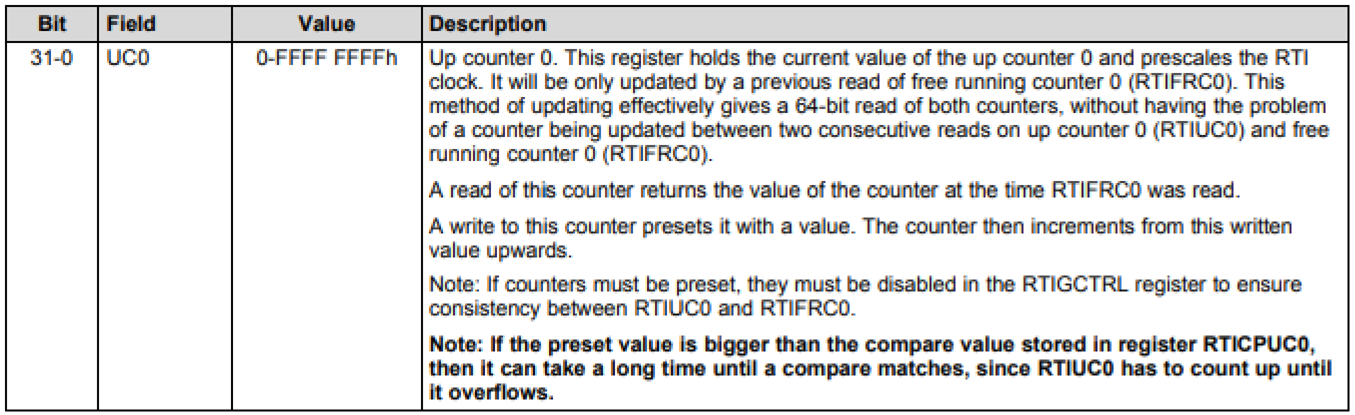
RTICOMP2 레지스터가 RTIFRC1 과 비교 값이 비교함.

RTICOMP1 레지스터가 RTIFRC1 과 비교 값을 비교함.

RTICOMP0 레지스터가 비교 값과 RTIFRC0 과 비교함

:88 CNT[0U].UCx

**Table 17-7. RTI Up Counter 0 Register (RTIUC0) Field Descriptions**



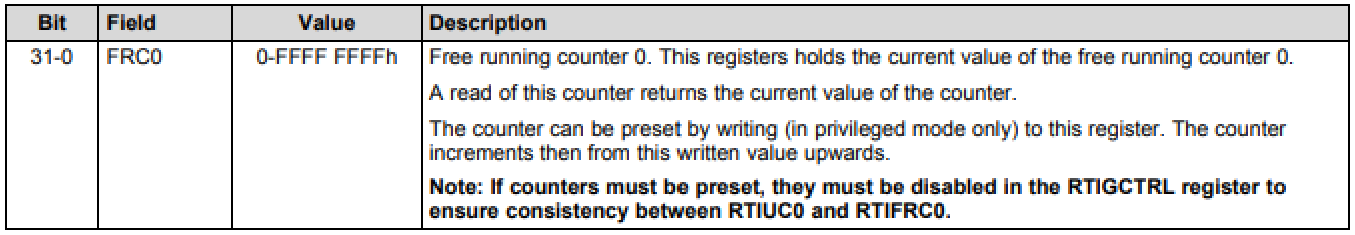
| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | UC0 | 0-FFFF FFFFh | Up counter 0. This register holds the current value of the up counter 0 and prescales the RTI clock. It will be only updated by a previous read of free running counter 0 (RTIFRC0). This method of updating effectively gives a 64-bit read of both counters, without having the problem of a counter being updated between two consecutive reads on up counter 0 (RTIUC0) and free running counter 0 (RTIFRC0).<br>A read of this counter returns the value of the counter at the time RTIFRC0 was read.<br>A write to this counter presets it with a value. The counter then increments from this written value upwards.<br>Note: If counters must be preset, they must be disabled in the RTIGCTRL register to ensure consistency between RTIUC0 and RTIFRC0.<br>**Note: If the preset value is bigger than the compare value stored in register RTICPUC0, then it can take a long time until a compare matches, since RTIUC0 has to count up until it overflows.** |

Up counter 0 레지스터는 현재의 up counter 0 을 유지하고 RTI clock 의 값을 지정한다.

값을 0 으로 지정함. RTI clock 값 초기화

:91 CNT[0U].FRCx

**Table 17-6. RTI Free Running Counter 0 Register (RTIFRC0) Field Descriptions**



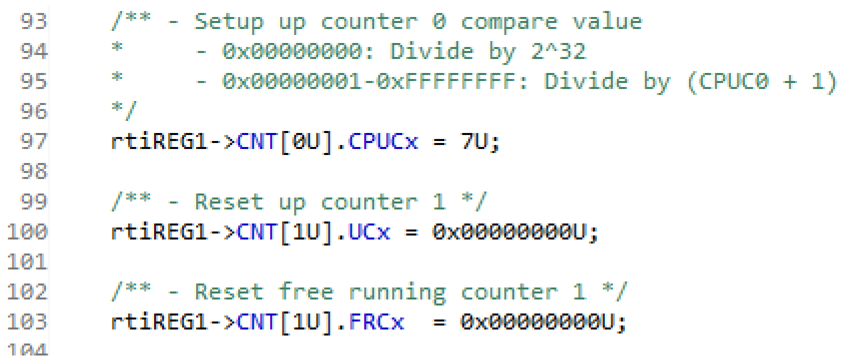
| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | FRC0 | 0-FFFF FFFFh | Free running counter 0. This registers holds the current value of the free running counter 0.<br>A read of this counter returns the current value of the counter.<br>The counter can be preset by writing (in privileged mode only) to this register. The counter increments then from this written value upwards.<br>**Note: If counters must be preset, they must be disabled in the RTIGCTRL register to ensure consistency between RTIUC0 and RTIFRC0.** |

free running counter 0 의 현재 값을 레지스터에 저장.

카운터를 읽으면 카운터의 현재 값을 리턴한다

레지스터에 0 을 저장. FRC0 초기화.

ritInit(2)



```
    /** - 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;
```

CPUCx

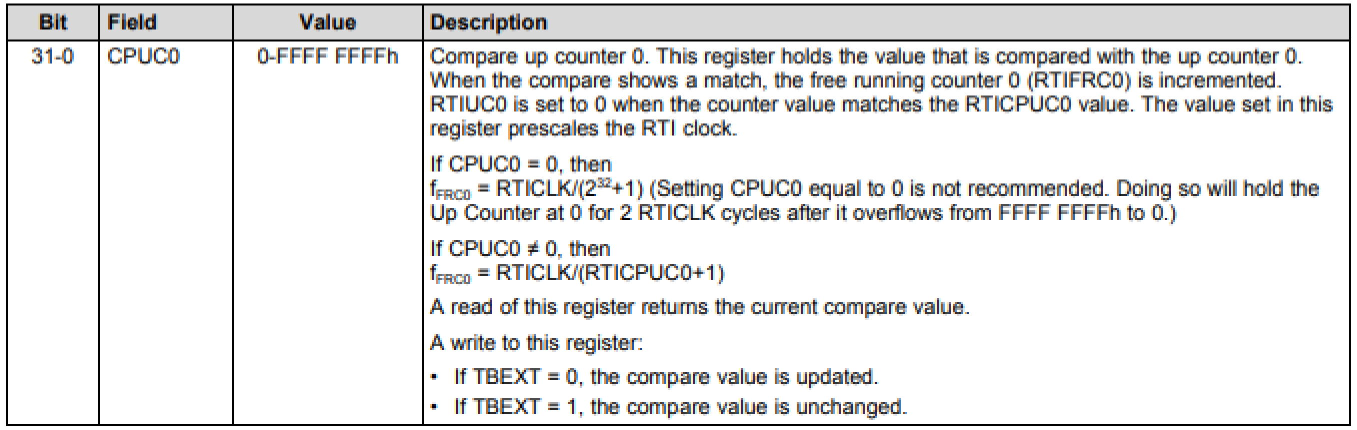


**Table 17-8. RTI Compare Up Counter 0 Register (RTICPUC0) Field Descriptions**

| 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.<br>If CPUC0 = 0, then<br>$f_{FRC0}$ = 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.)<br>If CPUC0 ≠ 0, then<br>$f_{FRC0}$ = RTICLK/(RTICPUC0+1)<br>A read of this register returns the current compare value.<br>A write to this register:<br>• If TBEXT = 0, the compare value is updated.<br>• If TBEXT = 1, the compare value is unchanged. |

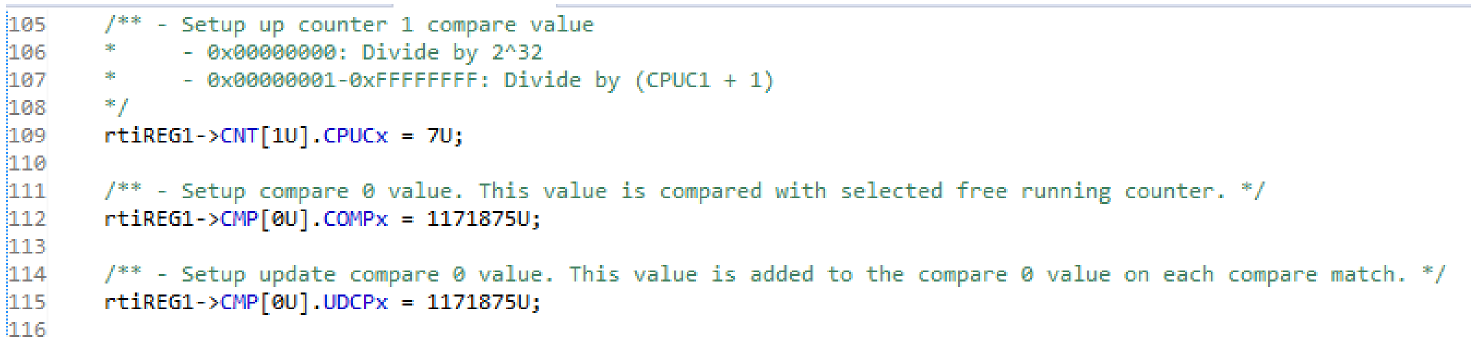
CPUC0 이 0 이 아니므로

RTICLK 을 8(7+1)분주한다.

레지스터 값을 읽으면 현재의 값이 반환된다.

CNT[1U].FRCx, CNT[1U].UCx 가 0 으로 초기화 됨.

rtiInit(3)

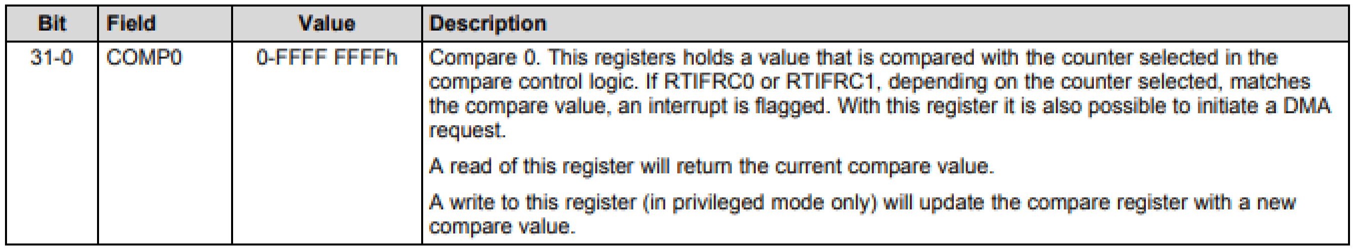


```
    /** - Setup up counter 1 compare value
    *     - 0x00000000: Divide by 2^32
    *     - 0x00000001-0xFFFFFFFF: Divide by (CPUC1 + 1)
    */
    rtiREG1->CNT[1U].CPUCx = 7U;

    /** - Setup compare 0 value. This value is compared with selected free running counter. */
    rtiREG1->CMP[0U].COMPx = 1171875U;

    /** - Setup update compare 0 value. This value is added to the compare 0 value on each compare match. */
    rtiREG1->CMP[0U].UDCPx = 1171875U;
```

CNT[1U].CPUCx 도 RTICLK 을 8 분주함.

CNT[0U].COMPx, CNT[0U].UDCPx,

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

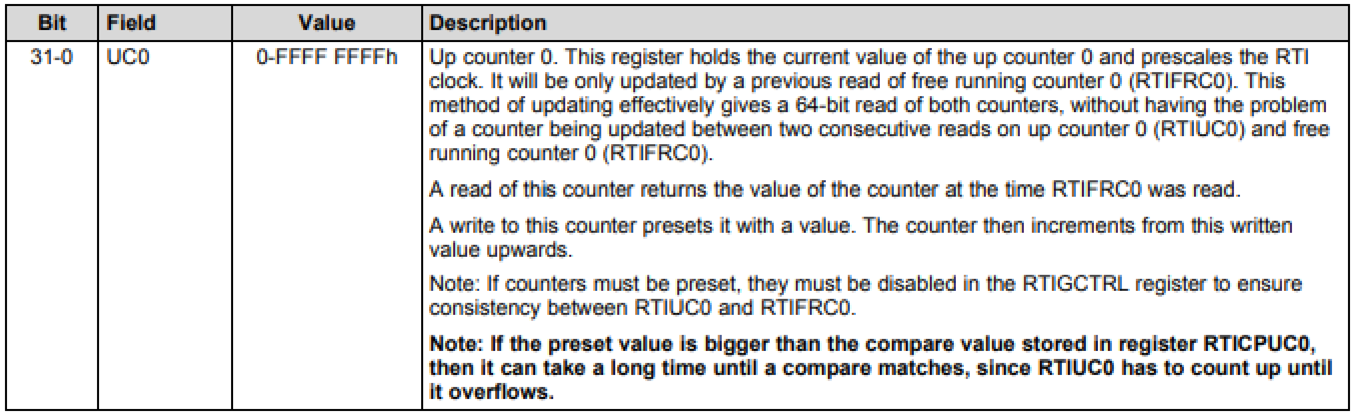


| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | COMP0 | 0-FFFF FFFFh | Compare 0. This registers holds a value that is compared with the counter selected in the compare control logic. If RTIFRC0 or RTIFRC1, depending on the counter selected, matches the compare value, an interrupt is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

RTICOMP0 레지스터에 비교할 논리 값을 설정함. 또한 이 레지스터는 DMA 요청 시작 가능하다

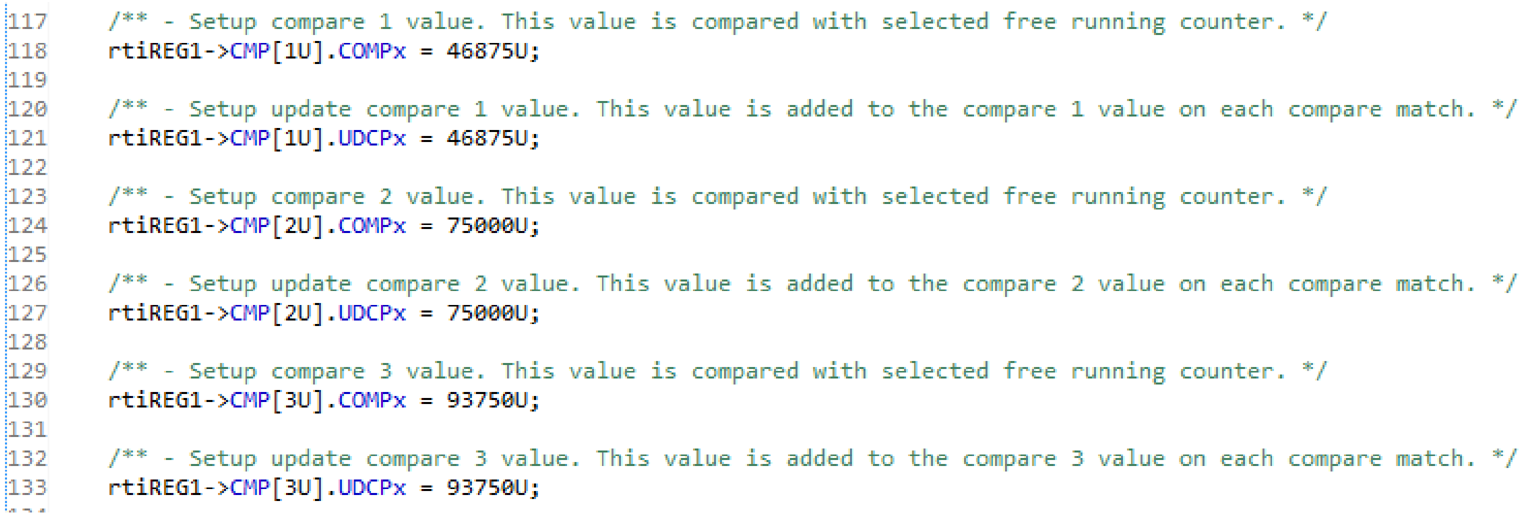
1171875 로 설정함

**Table 17-7. RTI Up Counter 0 Register (RTIUC0) Field Descriptions**



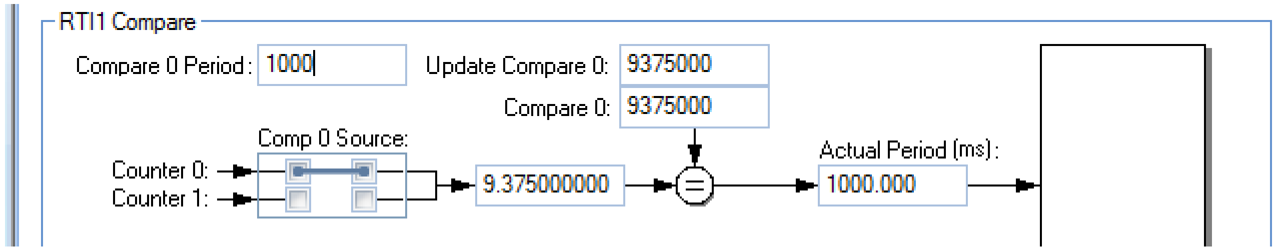
| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | UC0 | 0-FFFF FFFFh | Up counter 0. This register holds the current value of the up counter 0 and prescales the RTI clock. It will be only updated by a previous read of free running counter 0 (RTIFRC0). This method of updating effectively gives a 64-bit read of both counters, without having the problem of a counter being updated between two consecutive reads on up counter 0 (RTIUC0) and free running counter 0 (RTIFRC0).<br>A read of this counter returns the value of the counter at the time RTIFRC0 was read.<br>A write to this counter presets it with a value. The counter then increments from this written value upwards.<br>Note: If counters must be preset, they must be disabled in the RTIGCTRL register to ensure consistency between RTIUC0 and RTIFRC0.<br>**Note: If the preset value is bigger than the compare value stored in register RTICPUC0, then it can take a long time until a compare matches, since RTIUC0 has to count up until it overflows.** |

up counter0 과 미리 설정한 RTI 클럭 값을 설정한다.



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

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

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

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

    /** - Setup compare 3 value. This value is compared with selected free running counter. */
    rtiREG1->CMP[3U].COMPx = 93750U;

    /** - Setup update compare 3 value. This value is added to the compare 3 value on each compare match. */
    rtiREG1->CMP[3U].UDCPx = 93750U;
```

CNT[1U].COMPx, CNT[1U].UDCPx 과 CNT[2U].COMPx, CNT[2U].UDCPx, CNT[3U].COMPx, CNT[U].UDCPx 의 값들을 각각 46875, 75000, 93750 으로 설정한다.



RTI1 Compare

Compare 0 Period: 1000

Update Compare 0: 9375000

Compare 0: 9375000

Comp 0 Source:

Counter 0:

Counter 1:

9.375000000

Actual Period (ms): 1000.000

1 초를 만들기 위해 비교기에서 9375000 만큼의 클록을 카운터하여 비교하면 1 초가 됨.

**추가 제어**

LED 8 개 전부 다 켜기

```
#include "HL_sys_common.h"
#include "HL_system.h"
#include "HL_mibspi.h"
#include "HL_sys_core.h"
#include "HL_esm.h"
#include "HL_rti.h"
#include "HL_gio.h"
#include "HL_het.h"

int main(void)
{
	rtiInit();
	gioSetDirection(hetPORT1, 0xFFFFFFFF);
	rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE0);
	_enable_IRQ_interrupt_();
	rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0);

	while(1);

	return 0;
}

void rtiNotification(rtiBASE_t *rtiREG, uint32 notification)
{
	gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0xAA060021);
}
```

LED 뱀처럼 켜기

```
#include "HL_sys_common.h"
#include "HL_system.h"
#include "HL_mibspi.h"
#include "HL_sys_core.h"
#include "HL_esm.h"
#include "HL_rti.h"
#include "HL_gio.h"
#include "HL_het.h"

int main(void)
{
    rtiInit();
    gioSetDirection(hetPORT1, 0xFFFFFFFF);
    rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE0);
    _enable_IRQ_interrupt_();
    rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0);

    while(1);

    return 0;
}

void rtiNotification(rtiBASE_t *rtiREG, uint32 notification)
{
    gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0x1);
    gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0x20);
    gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0x20000);
    gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0x40000);
    gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0x2000000);
    gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0x8000000);
    gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0x20000000);
    gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0x80000000);
    gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0xAA060021);
}
```

순서 바꿔가며 LED 키기

```
#include "HL_sys_common.h"
#include "HL_system.h"
#include "HL_mibspi.h"
#include "HL_sys_core.h"
#include "HL_esm.h"
#include "HL_rti.h"
#include "HL_gio.h"
#include "HL_het.h"

int main(void)
{
    rtiInit();
    gioSetDirection(hetPORT1, 0xFFFFFFFF);
    rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE0);
    _enable_IRQ_interrupt_();
    rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0);

    while(1);

    return 0;
}

void rtiNotification(rtiBASE_t *rtiREG, uint32 notification)
{
    gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0x00000001);
    gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0x00000021);
    gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0x00020020);
    gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0x00060000);
    gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0x02040000);
    gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0x0A000000);
    gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0x28060000);
    gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0xA0000000);
    gioSetPort(hetPORT1, gioGetPort(hetPORT1) ^ 0x80000000);
}
```

# 아두이노를 이용한 서보모터 제어

아두이노로 서보모터 제어

```
#include <Servo.h>

Servo myservo;
int pos = 0;
void setup() {
  // put your setup code here, to run once:
myservo.attach(9);
}

void loop() {
  // put your main code here, to run repeatedly:
for(pos = 0;pos<180; pos +=1)
{
  myservo.write(pos);
  delay(15);
}
for(pos=180; pos>=1; pos-=1)
{
  myservo.write(pos);
  delay(15);
}
}
```

# Uniform Circular Montion (등속도 운동)

## 미분 :

(기하학적) 접선의 기울기

$$\lim_{x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} = \frac{df(x)}{dx}$$

(신호처리) 샘플링타임

$$\lim_{x \to 0} \frac{f(x + st) - f(x)}{st}$$

속도 : 시간에 따른 위치의 변화량

각속도 : 시간에 따른 각도의 변화량 == 회전속도

가속도 : 시간에 따른 속도의 변화량

각가속도 : 시간에 따른 각속도의 변화량

## 등속도 운동

(직선) $F = ma = m\frac{dv}{dt}$

(회전) $\tau = I\alpha$ ($\tau$: *토크*, $I$: *관성모멘트*, $\alpha$: *각가속도*)

(등속일 때 = 주기가 일정)

$$T = \frac{2\pi r}{v} \iff v = \frac{2\pi r}{T}$$

$$f = \frac{1}{T}$$

$$w = \frac{\theta}{t} = \frac{2\pi}{T} \Leftrightarrow \theta = wt$$

$$s = vt = r\theta \ \ v = \frac{r\theta}{t}$$

$$a = \frac{dv}{dt} = \frac{d}{dt}rw = \frac{dr}{dt}w + r\frac{dw}{dt} = vw + 0 = v\frac{v}{r} = \frac{v^2}{r} = rw^2$$

$$v = rw \Leftrightarrow w = \frac{w^2}{r}$$