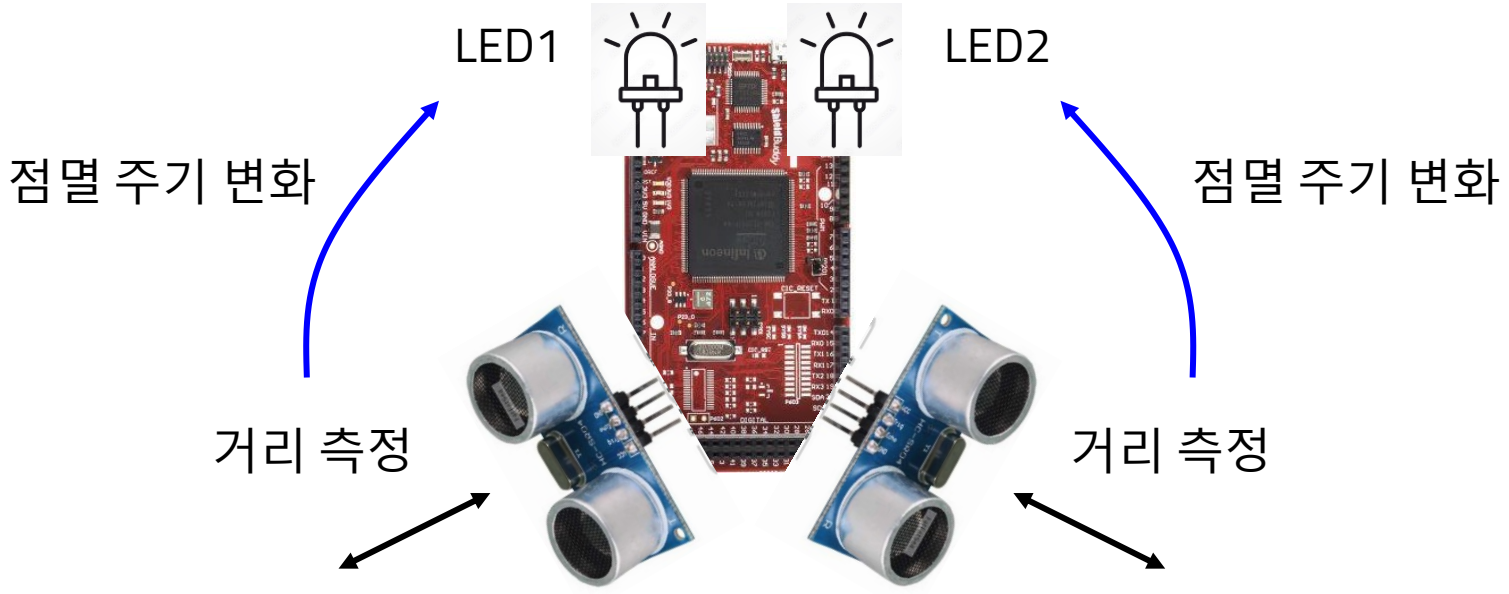
임베디드 기반 SW 개발 프로젝트

AURIX TC275 보드와 초음파 센서 사용
좌/우측 사각지대 LED 점멸

현대자동차 입문교육
박대진 교수

실습 목표

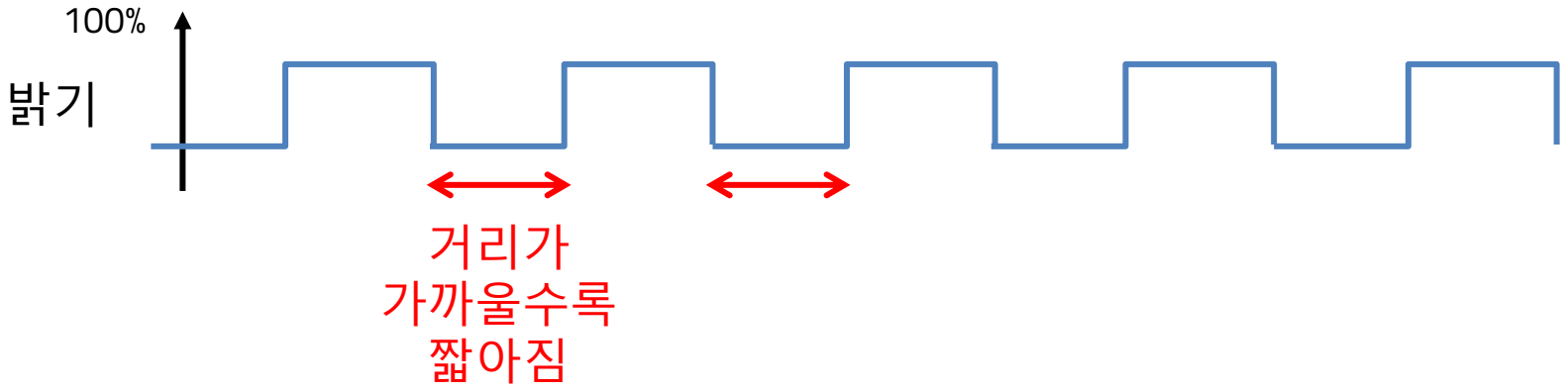
- 차량의 양측 사각지대 물체와의 거리를 초음파 센서로 측정하는 상황을 가정
- 물체와의 거리가 가까울수록 **LED** 점멸 주기를 짧게, 멀수록 길게 변화시켜 본다.
- **2**개의 초음파 센서를 사용해본다.
  1. GPIO로 LED 구동
  2. PWM으로 LED의 Dimming 구동



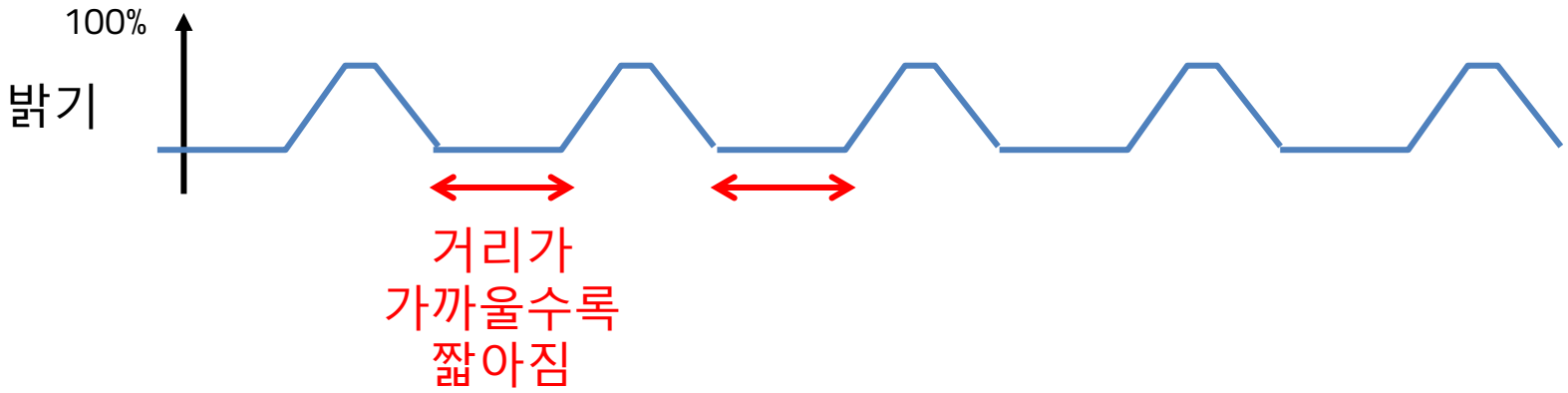
LED 동작 개요

: 거리에 따른 GPIO / PWM 점멸 주기 변화 방식

- **GPIO로 LED 점멸 하는 경우**



- **PWM으로 LED Dimming 하는 경우**



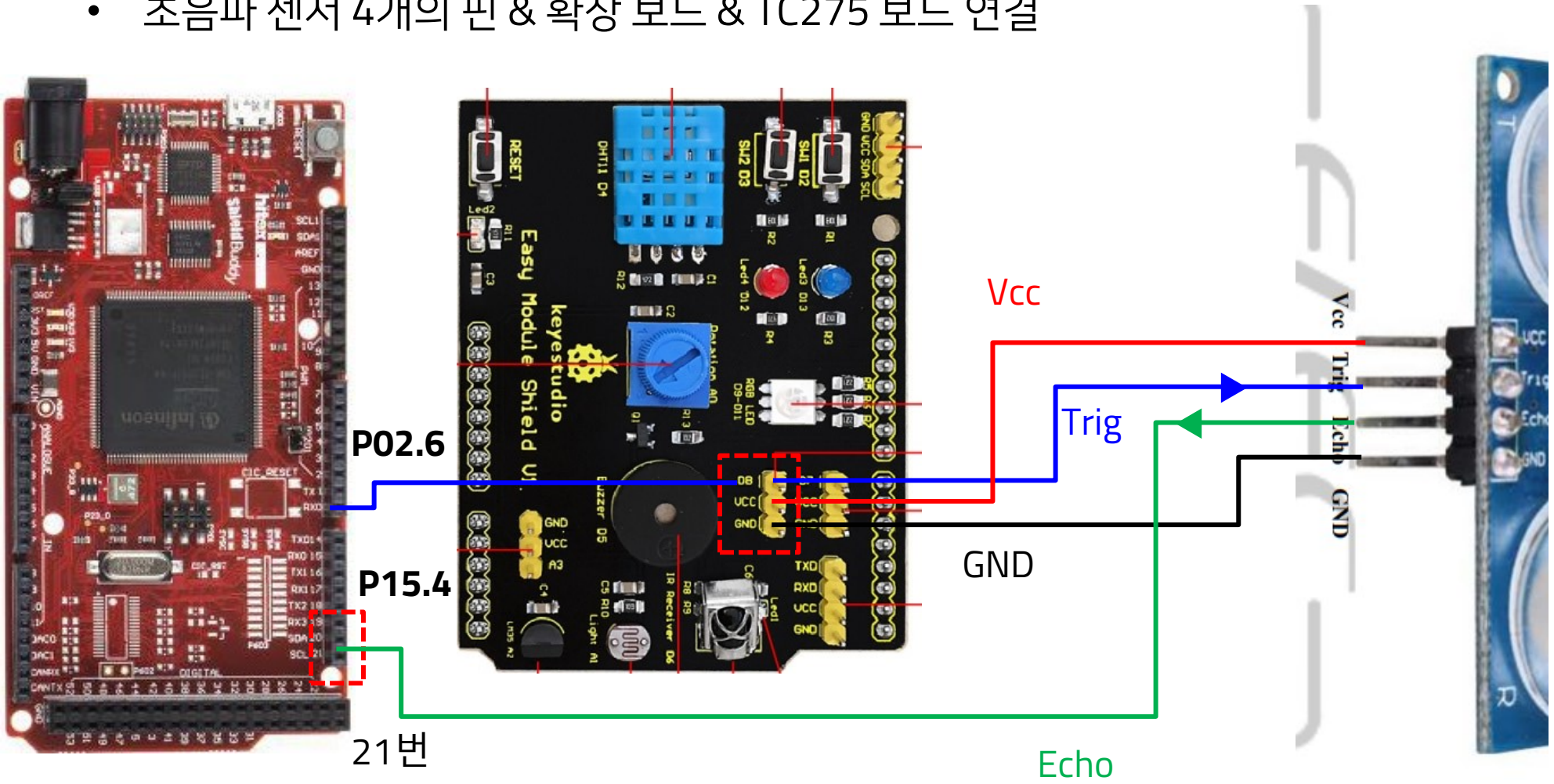
HC-SR04 초음파 센서 데이터 시트 참고 (HCSR04.pdf)

TC275 보드와 2개의 초음파 센서 연결

TC275 보드와 1번 초음파 센서 연결

:확장 보드의 핀과 초음파 센서 연결 – 모든 핀 연결한 모습

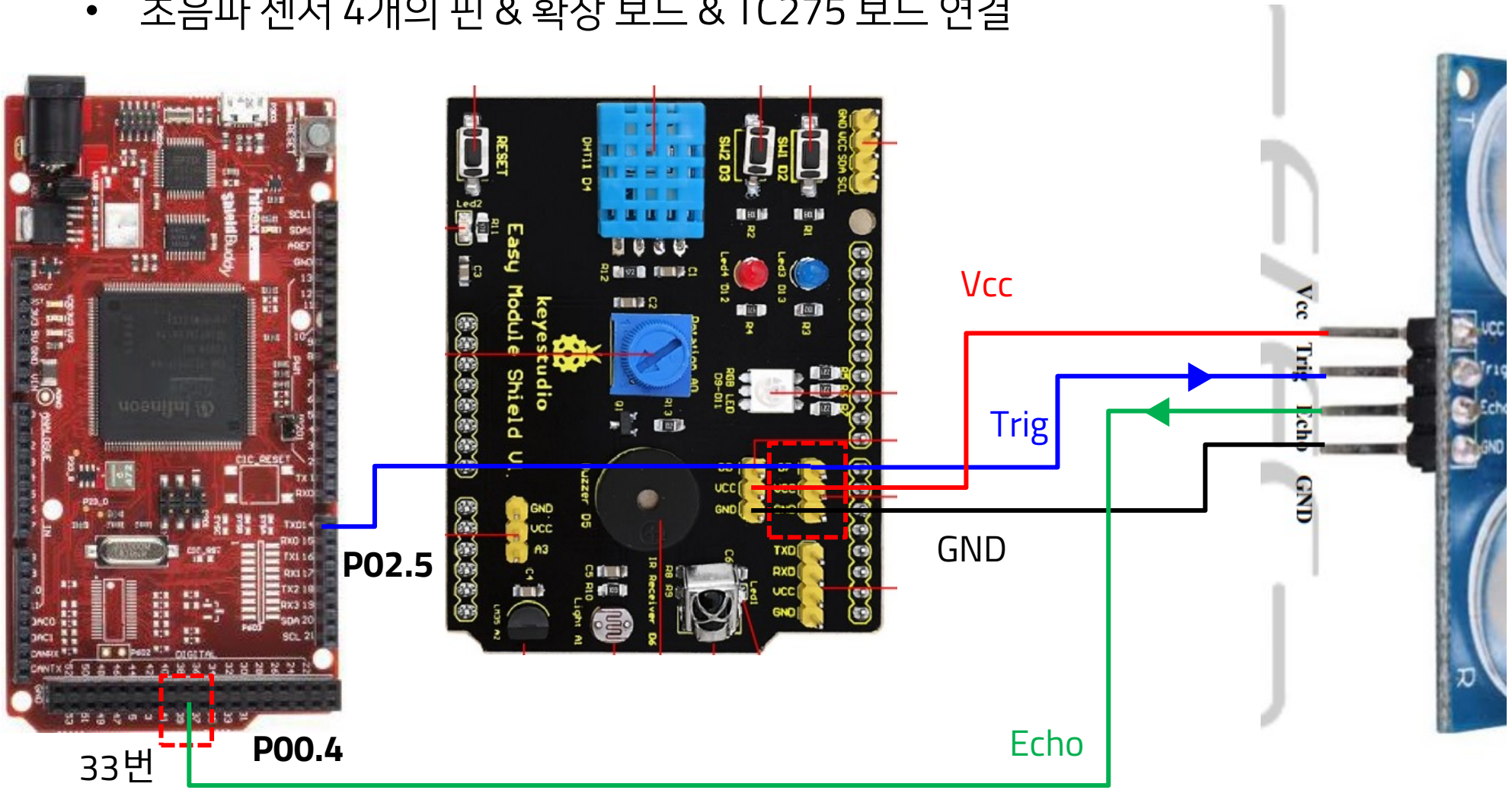
- 초음파 센서 4개의 핀 & 확장 보드 & TC275 보드 연결



TC275 보드와 2번 초음파 센서 연결

:확장 보드의 핀과 초음파 센서 연결 – 모든 핀 연결한 모습

- 초음파 센서 4개의 핀 & 확장 보드 & TC275 보드 연결



초음파 센서 거리 기반
GPIO LED 점멸 주기 변화

Lab1

: 헤더 파일 / define 정의 / 전역 변수 / 함수 prototype

```
26  ***************************************
27  #include "Ifx_Types.h"
28  #include "IfxCpu.h"
29  #include "IfxScuWdt.h"
30
31  #include "IfxCcu6_reg.h"
32  #include "IfxGtm_reg.h"
33
34  // SCU registers
35  #define LCK_BIT_LSB_IDX             1
36  #define ENDINIT_BIT_LSB_IDX         0
37
38  // GTM registers
39  #define DISS_BIT_LSB_IDX            1
40  #define DISR_BIT_LSB_IDX            0
41
42  IfxCpu_syncEvent g_cpuSyncEvent = 0;
43
44  void initLED(void);
45  void initERU(void);
46  void initCCU60(void);
47  void initCCU61(void);
48  void usonicTrigger_1(void);
49  void usonicTrigger_2(void);
50  void initUSonic(void);
51
52  unsigned int range_1, range_2;
53  unsigned char range_valid_flag_1 = 0;
54  unsigned char range_valid_flag_2 = 0;
55
```

Lab2

: ERU (외부 인터럽트) ISR

```
57
58  __interrupt(0x0A) __vector_table(0)
59  void ERU0_ISR(void)
60  {
61      if( P15_IN.B.P4 != 0x0 )      // rising edge of echo
62      {
63          //                  _________
64          //        echo ______|
65          //                  ^
66          CCU61_TCTR4.B.T12RS = 0x1;        // start CCU61 T12 counter
67      }
68      else    // falling edge of echo
69      {
70          //            ______
71          //        echo      |______
72          //                  ^
73          CCU61_TCTR4.B.T12RR = 0x1;        // stop CCU61 T12 counter
74
75          // (1 / t_freq) * counter * 1000000 / 58  =  centimeter
76          range_1 = ((CCU61_T12.B.T12CV * 1000000) / 48828) / 58;
77          range_valid_flag_1 = 1;
78
79          CCU61_TCTR4.B.T12RES = 0x1;       // reset CCU61 T12 counter
80      }
81  }
82
```

```
82
83  __interrupt(0x0E) __vector_table(0)
84  void ERU2_ISR(void)
85  {
86      if( P00_IN.B.P4 != 0x0 )      // rising edge of echo
87      {
88          //                  _________
89          //        echo ______|
90          //                  ^
91          CCU61_TCTR4.B.T13RS = 0x1;        // start CCU61 T13 counter
92      }
93      else    // falling edge of echo
94      {
95          //            ______
96          //        echo      |______
97          //                  ^
98          CCU61_TCTR4.B.T13RR = 0x1;        // stop CCU61 T13 counter
99
100         // (1 / t_freq) * counter * 1000000 / 58  =  centimeter
101         range_2 = ((CCU61_T13.B.T13CV * 1000000) / 48828) / 58;
102         range_valid_flag_2 = 1;
103
104         CCU61_TCTR4.B.T13RES = 0x1;       // reset CCU61 T13 counter
105     }
106 }
107
```

Lab3

: CCU6 타이머 인터럽트

```
108 __interrupt(0x0B) __vector_table(0)
109 void CCU60_T12_ISR(void)
110 {
111     // end of 10us Trig
112     // GPIO P02.6 --> LOW
113     P02_OUT.B.P6 = 0x0;
114 }
115
116 __interrupt(0x0C) __vector_table(0)
117 void CCU60_T13_ISR(void)
118 {
119     // end of 10us Trig
120     // GPIO P02.5 --> LOW
121     P02_OUT.B.P5 = 0x0;
122 }
123
```

Lab4

: main 함수

```c
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);

    initERU();
    initCCU60();
    initCCU61();
    initLED();
    initUSonic();
```

```c
    while(1)
    {
        for(unsigned int i = 0; i < 10000000; i++);

        usonicTrigger_1();
        while( range_valid_flag_1 == 0  );
        P10_OUT.B.P1 = 0x1;
        for(unsigned int i = 0; i < 1000000; i++);
        P10_OUT.B.P1 = 0x0;
        for(unsigned int i = 0; i < (range_1 * 500000); i++);


        usonicTrigger_2();
        while( range_valid_flag_2 == 0  );
        P10_OUT.B.P2 = 0x1;
        for(unsigned int i = 0; i < 1000000; i++);
        P10_OUT.B.P2 = 0x0;
        for(unsigned int i = 0; i < (range_2 * 500000); i++);

    }
    return (1);
}
```

Lab5

: LED, ERU configuration

```
167 void initLED(void)
168 {
169     P10_IOCR0.B.PC1 = 0x10;      // set P10.1 general output
170     P10_IOCR0.B.PC2 = 0x10;      // set P10.2 general output
171 }
172
173 void initERU(void)
174 {
175     // Ultrasonic 1 Echo
176     SCU_EICR0.B.EXIS0 = 0x0;     // ERS0 - In00
177     SCU_EICR0.B.FEN0  = 0x1;     // falling edge
178     SCU_EICR0.B.REN0  = 0x1;     // rising edge
179     SCU_EICR0.B.EIEN0 = 0x1;
180     SCU_EICR0.B.INP0  = 0x0;
181     SCU_IGCR0.B.IGP0  = 0x1;
182
183     // SRC Interrupt setting
184     SRC_SCU_SCU_ERU0.B.SRPN = 0x0A;
185     SRC_SCU_SCU_ERU0.B.TOS  = 0x00;
186     SRC_SCU_SCU_ERU0.B.SRE  = 0x01;
187
188     // Ultrasonic 2 Echo
189     SCU_EICR1.B.EXIS0 = 0x2;     // ERS2 - Input 2 (P00.4)
190     SCU_EICR1.B.FEN0  = 0x1;     // falling edge
191     SCU_EICR1.B.REN0  = 0x1;
192     SCU_EICR1.B.EIEN0 = 0x1;
193     SCU_EICR1.B.INP0  = 0x1;     // OGU0
194     SCU_IGCR0.B.IGP1  = 0x1;
195
196     // SRC Interrupt setting
197     SRC_SCU_SCU_ERU1.B.SRPN = 0x0E;
198     SRC_SCU_SCU_ERU1.B.TOS  = 0x0;
199     SRC_SCU_SCU_ERU1.B.SRE  = 0x1;
200 }
201
```

Lab6

: CCU60 타이머 configuration

```
void initCCU60(void)
{
    // Password Access to unlock SCU_WDTSCON0
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
    // Modify Access to clear ENDINIT
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
    CCU60_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCU
    // Password Access to unlock SCU_WDTSCON0
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
    // Modify Access to set ENDINIT
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
    // CCU60 T12 configurations
    while((CCU60_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0);// wait until CCU60 module enabled

    // T12 configurations
    CCU60_TCTR0.B.T12CLK = 0x2;                    // f_CCU6 = 50 MHz, prescaler = 4
//    CCU60_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX;   // f_T12 --> 12.5 MHz
    CCU60_TCTR0.B.CTM = 0x0;                       // T12 auto reset when period match (PM) occur
    CCU60_T12PR.B.T12PV = 125 - 1;                 // PM interrupt freq. = f_T12 / (T12PR + 1)
    CCU60_TCTR4.B.T12STR = 0x1;                    // load T12PR from shadow register
    CCU60_TCTR2.B.T12SSC = 0x1;                    // Single Shot Control
    CCU60_T12.B.T12CV = 0x0;                       // clear T12 counter register
    CCU60_INP.B.INPT12 = 0x0;                      // service request output SR0 selected
    CCU60_IEN.B.ENT12PM = 0x1;                     // enable T12 PM interrupt
    SRC_CCU6_CCU60_SR0.B.SRPN = 0x0B;              // set priority 0x0B
    SRC_CCU6_CCU60_SR0.B.TOS = 0x0;                // CPU0 service T12 PM interrupt
    SRC_CCU6_CCU60_SR0.B.SRE = 0x1;                // SR0 enabled

    // T13 configurations
    CCU60_TCTR0.B.T13CLK = 0x2;                    // f_CCU6 = 50 MHz, prescaler = 4
//    CCU60_TCTR0.B.T13PRE = 0x1;                   // f_T13 --> 12.5 MHz
    CCU60_TCTR0.B.CTM = 0x0;                       // T13 auto reset when period match (PM) occur
    CCU60_T13PR.B.T13PV = 125 - 1;
    CCU60_TCTR4.B.T13STR = 0x1;
    CCU60_TCTR2.B.T13SSC = 0x1;                    // Single Shot Control
    CCU60_T13.B.T13CV = 0x0;
    CCU60_IEN.B.ENT13PM = 0x1;
    CCU60_INP.B.INPT13 = 0x1;
    SRC_CCU6_CCU60_SR1.B.SRPN = 0x0C;
    SRC_CCU6_CCU60_SR1.B.TOS = 0x0;
    SRC_CCU6_CCU60_SR1.B.SRE = 0x1;
}
```

Lab7

: CCU61 타이머 configuration

```
void initCCU61(void)
{
    // Password Access to unlock SCU_WDTSCON0
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked

    // Modify Access to clear ENDINIT
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked

    CCU61_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCU

    // Password Access to unlock SCU_WDTSCON0
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked

    // Modify Access to set ENDINIT
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked

    // CCU60 T12 configurations
    while((CCU61_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0);// wait until CCU60 module enabled

    // f_T12 = f_CCU6 / prescaler = 12.5 MHz
    CCU61_TCTR0.B.T12CLK = 0x2;          // f_CCU6 = 50 MHz, prescaler = 4
    CCU61_TCTR0.B.T12PRE = 0x1;          // f_T12 = f_CCU6 / 256 = 48,828 Hz
    CCU61_T12PR.B.T12PV = 100000 -1;     // PM interrupt freq. = f_T12 / (T12PR + 1)
    CCU61_TCTR4.B.T12STR = 0x1;          // load T12PR from shadow register
    CCU61_T12.B.T12CV = 0x0;             // clear T12 counter register

    CCU61_TCTR0.B.T13CLK = 0x2;          // f_CCU6 = 50 MHz, prescaler = 4
    CCU61_TCTR0.B.T13PRE = 0x1;          // f_T13 = f_CCU6 / 256 = 48,828 Hz
    CCU61_T13PR.B.T13PV = 100000 -1;     // PM interrupt freq. = f_T13 / (T13PR + 1)
    CCU61_TCTR4.B.T13STR = 0x1;          // load T13PR from shadow register
    CCU61_T13.B.T13CV = 0x0;             // clear T13 counter register
}
```

Lab8

: 초음파 센서 사용 위한 configuration 및 trigger 함수

```
342
343 void initUSonic(void)
344 {
345     // USonic 1
346     P15_IOCR4.B.PC4 = 0x01;         // set P15.4 general input (pull-down connected) [Echo]
347     P02_IOCR4.B.PC6 = 0x10;         // set P02.6 push-pull general output            [Trig]
348     P02_OUT.B.P6 = 0x0;
349
350     // Usonic 2
351     P00_IOCR4.B.PC4 = 0x01;         // set P00.4 general input (pull-down connected) [Echo]
352     P02_IOCR4.B.PC5 = 0x10;         // set P02.5 push-pull general output            [Trig]
353     P02_OUT.B.P5 = 0x0;
354 }
355
356 void usonicTrigger_1(void)
357 {
358     // start of 10us Trigger Pulse
359     // GPIO P02.6 --> HIGH
360     P02_OUT.B.P6 = 0x1;
361     range_valid_flag_1 = 0;
362     CCU60_TCTR4.B.T12RS = 0x1;          // T12 start counting
363 }
364
365 void usonicTrigger_2(void)
366 {
367     // start of 10us Trigger Pulse
368     // GPIO P02.5 --> HIGH
369     P02_OUT.B.P5 = 0x1;
370     range_valid_flag_2 = 0;
371     CCU60_TCTR4.B.T13RS = 0x1;          // T13 start counting
372 }
373
```

초음파 센서 거리 및 PWM 기반 LED DIMMING 주기 변화

Lab1

: 헤더 파일 / define 정의 / 전역 변수 / 함수 prototype

```
26  ************************************************
27  #include "Ifx_Types.h"
28  #include "IfxCpu.h"
29  #include "IfxScuWdt.h"
30
31  #include "IfxCcu6_reg.h"
32  #include "IfxGtm_reg.h"
33
34  // SCU registers
35  #define LCK_BIT_LSB_IDX             1
36  #define ENDINIT_BIT_LSB_IDX         0
37
38  // GTM registers
39  #define DISS_BIT_LSB_IDX            1
40  #define DISR_BIT_LSB_IDX            0
41
42  IfxCpu_syncEvent g_cpuSyncEvent = 0;
43
44  void initLED(void);
45  void initERU(void);
46  void initCCU60(void);
47  void initCCU61(void);
48  void usonicTrigger_1(void);
49  void usonicTrigger_2(void);
50  void initUSonic(void);
51  void initGTM(void);
52
53  unsigned int range_1, range_2;
54  unsigned char range_valid_flag_1 = 0;
55  unsigned char range_valid_flag_2 = 0;
56
```

Lab2

: ERU (외부 인터럽트) ISR

```
57
58  __interrupt(0x0A) __vector_table(0)
59  void ERU0_ISR(void)
60  {
61      if( P15_IN.B.P4 != 0x0 )      // rising edge of echo
62      {
63          //                  _________
64          //        echo ______|
65          //                  ^
66          CCU61_TCTR4.B.T12RS = 0x1;        // start CCU61 T12 counter
67      }
68      else    // falling edge of echo
69      {
70          //            ______
71          //        echo      |______
72          //                  ^
73          CCU61_TCTR4.B.T12RR = 0x1;        // stop CCU61 T12 counter
74
75          // (1 / t_freq) * counter * 1000000 / 58  =  centimeter
76          range_1 = ((CCU61_T12.B.T12CV * 1000000) / 48828) / 58;
77          range_valid_flag_1 = 1;
78
79          CCU61_TCTR4.B.T12RES = 0x1;       // reset CCU61 T12 counter
80      }
81  }
82
```

```
82
83  __interrupt(0x0E) __vector_table(0)
84  void ERU2_ISR(void)
85  {
86      if( P00_IN.B.P4 != 0x0 )      // rising edge of echo
87      {
88          //                  _________
89          //        echo ______|
90          //                  ^
91          CCU61_TCTR4.B.T13RS = 0x1;        // start CCU61 T13 counter
92      }
93      else    // falling edge of echo
94      {
95          //            ______
96          //        echo      |______
97          //                  ^
98          CCU61_TCTR4.B.T13RR = 0x1;        // stop CCU61 T13 counter
99
100         // (1 / t_freq) * counter * 1000000 / 58  =  centimeter
101         range_2 = ((CCU61_T13.B.T13CV * 1000000) / 48828) / 58;
102         range_valid_flag_2 = 1;
103
104         CCU61_TCTR4.B.T13RES = 0x1;       // reset CCU61 T13 counter
105     }
106 }
107
```

Lab3

: CCU6 타이머 인터럽트

```
108 __interrupt(0x0B) __vector_table(0)
109 void CCU60_T12_ISR(void)
110 {
111     // end of 10us Trig
112     // GPIO P02.6 --> LOW
113     P02_OUT.B.P6 = 0x0;
114 }
115
116 __interrupt(0x0C) __vector_table(0)
117 void CCU60_T13_ISR(void)
118 {
119     // end of 10us Trig
120     // GPIO P02.5 --> LOW
121     P02_OUT.B.P5 = 0x0;
122 }
123
```

Lab4

: main 함수

```c
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);

    initGTM();
    initERU();
    initCCU60();
    initCCU61();
    initLED();
    initUSonic();
```

```c
    while(1)
    {
        for(unsigned int i = 0; i < 10000000; i++);

        usonicTrigger_1();
        while( range_valid_flag_1 == 0  );
        for(unsigned int i = 0; i < 12500; i++) {
            GTM_TOM0_CH1_SR1.U = i;
            for(unsigned int j = 0; j < 2000; j++);
        }
        for(unsigned int i = 0; i < 5000; i++);
        for(unsigned int i = 12500; i > 0; i--) {
            GTM_TOM0_CH1_SR1.U = i;
            for(unsigned int j = 0; j < 2000; j++);
        }
        for(unsigned int i = 0; i < (range_1 * 500000); i++);


        usonicTrigger_2();
        while( range_valid_flag_2 == 0  );
        for(unsigned int i = 0; i < 12500; i++) {
            GTM_TOM0_CH2_SR1.U = i;
            for(unsigned int j = 0; j < 2000; j++);
        }
        for(unsigned int i = 0; i < 5000; i++);
        for(unsigned int i = 12500; i > 0; i--) {
            GTM_TOM0_CH2_SR1.U = i;
            for(unsigned int j = 0; j < 2000; j++);
        }
        for(unsigned int i = 0; i < (range_2 * 500000); i++);

    }
    return (1);
}
```

Lab5

: LED, ERU configuration

```
180
181 void initLED(void)
182 {
183     P10_IOCR0.B.PC1 = 0x11;      // set P10.1 GTM PWM output
184     P10_IOCR0.B.PC2 = 0x11;      // set P10.2 GTM PWM output
185 }
186
187 void initERU(void)
188 {
189     // Ultrasonic 1 Echo
190     SCU_EICR0.B.EXIS0 = 0x0;     // ERS0 - In00
191     SCU_EICR0.B.FEN0  = 0x1;     // falling edge
192     SCU_EICR0.B.REN0  = 0x1;     // rising edge
193     SCU_EICR0.B.EIEN0 = 0x1;
194     SCU_EICR0.B.INP0  = 0x0;
195     SCU_IGCR0.B.IGP0  = 0x1;
196
197     // SRC Interrupt setting
198     SRC_SCU_SCU_ERU0.B.SRPN = 0x0A;
199     SRC_SCU_SCU_ERU0.B.TOS  = 0x00;
200     SRC_SCU_SCU_ERU0.B.SRE  = 0x01;
201
202     // Ultrasonic 2 Echo
203     SCU_EICR1.B.EXIS0 = 0x2;     // ERS2 - Input 2 (P00.4)
204     SCU_EICR1.B.FEN0  = 0x1;     // falling edge
205     SCU_EICR1.B.REN0  = 0x1;
206     SCU_EICR1.B.EIEN0 = 0x1;
207     SCU_EICR1.B.INP0  = 0x1;     // OGU0
208     SCU_IGCR0.B.IGP1  = 0x1;
209
210     // SRC Interrupt setting
211     SRC_SCU_SCU_ERU1.B.SRPN = 0x0E;
212     SRC_SCU_SCU_ERU1.B.TOS  = 0x0;
213     SRC_SCU_SCU_ERU1.B.SRE  = 0x1;
214 }
215
```

Lab6

: CCU60 타이머 configuration

```
void initCCU60(void)
{
    // Password Access to unlock SCU_WDTSCON0
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
    // Modify Access to clear ENDINIT
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
    CCU60_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCU
    // Password Access to unlock SCU_WDTSCON0
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
    // Modify Access to set ENDINIT
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
    // CCU60 T12 configurations
    while((CCU60_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0);// wait until CCU60 module enabled

    // T12 configurations
    CCU60_TCTR0.B.T12CLK = 0x2;                    // f_CCU6 = 50 MHz, prescaler = 4
//    CCU60_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX;   // f_T12 --> 12.5 MHz
    CCU60_TCTR0.B.CTM = 0x0;                       // T12 auto reset when period match (PM) occur
    CCU60_T12PR.B.T12PV = 125 - 1;                 // PM interrupt freq. = f_T12 / (T12PR + 1)
    CCU60_TCTR4.B.T12STR = 0x1;                    // load T12PR from shadow register
    CCU60_TCTR2.B.T12SSC = 0x1;                    // Single Shot Control
    CCU60_T12.B.T12CV = 0x0;                       // clear T12 counter register
    CCU60_INP.B.INPT12 = 0x0;                      // service request output SR0 selected
    CCU60_IEN.B.ENT12PM = 0x1;                     // enable T12 PM interrupt
    SRC_CCU6_CCU60_SR0.B.SRPN = 0x0B;              // set priority 0x0B
    SRC_CCU6_CCU60_SR0.B.TOS = 0x0;                // CPU0 service T12 PM interrupt
    SRC_CCU6_CCU60_SR0.B.SRE = 0x1;                // SR0 enabled

    // T13 configurations
    CCU60_TCTR0.B.T13CLK = 0x2;                    // f_CCU6 = 50 MHz, prescaler = 4
//    CCU60_TCTR0.B.T13PRE = 0x1;                   // f_T13 --> 12.5 MHz
    CCU60_TCTR0.B.CTM = 0x0;                       // T13 auto reset when period match (PM) occur
    CCU60_T13PR.B.T13PV = 125 - 1;
    CCU60_TCTR4.B.T13STR = 0x1;
    CCU60_TCTR2.B.T13SSC = 0x1;                    // Single Shot Control
    CCU60_T13.B.T13CV = 0x0;
    CCU60_IEN.B.ENT13PM = 0x1;
    CCU60_INP.B.INPT13 = 0x1;
    SRC_CCU6_CCU60_SR1.B.SRPN = 0x0C;
    SRC_CCU6_CCU60_SR1.B.TOS = 0x0;
    SRC_CCU6_CCU60_SR1.B.SRE = 0x1;
}
```

Lab7

: CCU61 타이머 configuration

```
void initCCU61(void)
{
    // Password Access to unlock SCU_WDTSCON0
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked

    // Modify Access to clear ENDINIT
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked

    CCU61_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCU

    // Password Access to unlock SCU_WDTSCON0
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked

    // Modify Access to set ENDINIT
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked

    // CCU60 T12 configurations
    while((CCU61_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0);// wait until CCU60 module enabled

    // f_T12 = f_CCU6 / prescaler = 12.5 MHz
    CCU61_TCTR0.B.T12CLK = 0x2;          // f_CCU6 = 50 MHz, prescaler = 4
    CCU61_TCTR0.B.T12PRE = 0x1;          // f_T12 = f_CCU6 / 256 = 48,828 Hz
    CCU61_T12PR.B.T12PV = 100000 -1;     // PM interrupt freq. = f_T12 / (T12PR + 1)
    CCU61_TCTR4.B.T12STR = 0x1;          // load T12PR from shadow register
    CCU61_T12.B.T12CV = 0x0;             // clear T12 counter register

    CCU61_TCTR0.B.T13CLK = 0x2;          // f_CCU6 = 50 MHz, prescaler = 4
    CCU61_TCTR0.B.T13PRE = 0x1;          // f_T13 = f_CCU6 / 256 = 48,828 Hz
    CCU61_T13PR.B.T13PV = 100000 -1;     // PM interrupt freq. = f_T13 / (T13PR + 1)
    CCU61_TCTR4.B.T13STR = 0x1;          // load T13PR from shadow register
    CCU61_T13.B.T13CV = 0x0;             // clear T13 counter register
}
```

Lab8

: PWM 생성을 위한 GTM configuration

```
263 void initGTM(void)
264 {
265     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0XFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
266     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);
267
268     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0XFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
269     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
270
271     GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX); // enable VADC
272
273     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
274     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);
275
276     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
277     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
278
279     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0);
280
281     GTM_CMU_CLK_EN.B.EN_FXCLK = 0x2;
282
283     // P10.1 -> TOM0_1 -> TOUT103
284     GTM_TOM0_TGC0_GLB_CTRL.B.UPEN_CTRL1 = 0x2;
285     GTM_TOM0_TGC0_ENDIS_CTRL.B.ENDIS_CTRL1 = 0x2;
286     GTM_TOM0_TGC0_OUTEN_CTRL.B.OUTEN_CTRL1 = 0x2;
287     GTM_TOM0_CH1_CTRL.B.SL = 0x1;
288     GTM_TOM0_CH1_CTRL.B.CLK_SRC_SR = 0x1;
289     GTM_TOM0_CH1_SR0.B.SR0 = 12500 - 1;
290     GTM_TOM0_CH1_SR1.B.SR1 = 12500 - 1;
291     GTM_TOUTSEL6.B.SEL15 = 0x0;
292
293     // P10.2 -> TOM0_2 -> TOUT104
294     GTM_TOM0_TGC0_GLB_CTRL.B.UPEN_CTRL2 = 0x2;
295     GTM_TOM0_TGC0_ENDIS_CTRL.B.ENDIS_CTRL2 = 0x2;
296     GTM_TOM0_TGC0_OUTEN_CTRL.B.OUTEN_CTRL2 = 0x2;
297     GTM_TOM0_CH2_CTRL.B.SL = 0x1;
298     GTM_TOM0_CH2_CTRL.B.CLK_SRC_SR = 0x1;
299     GTM_TOM0_CH2_SR0.B.SR0 = 12500 - 1;
300     GTM_TOM0_CH2_SR1.B.SR1 = 12500 - 1;
301     GTM_TOUTSEL7.B.SEL0 = 0x0;
302
303     GTM_TOM0_TGC0_GLB_CTRL.B.HOST_TRIG = 0x1;
304 }
```

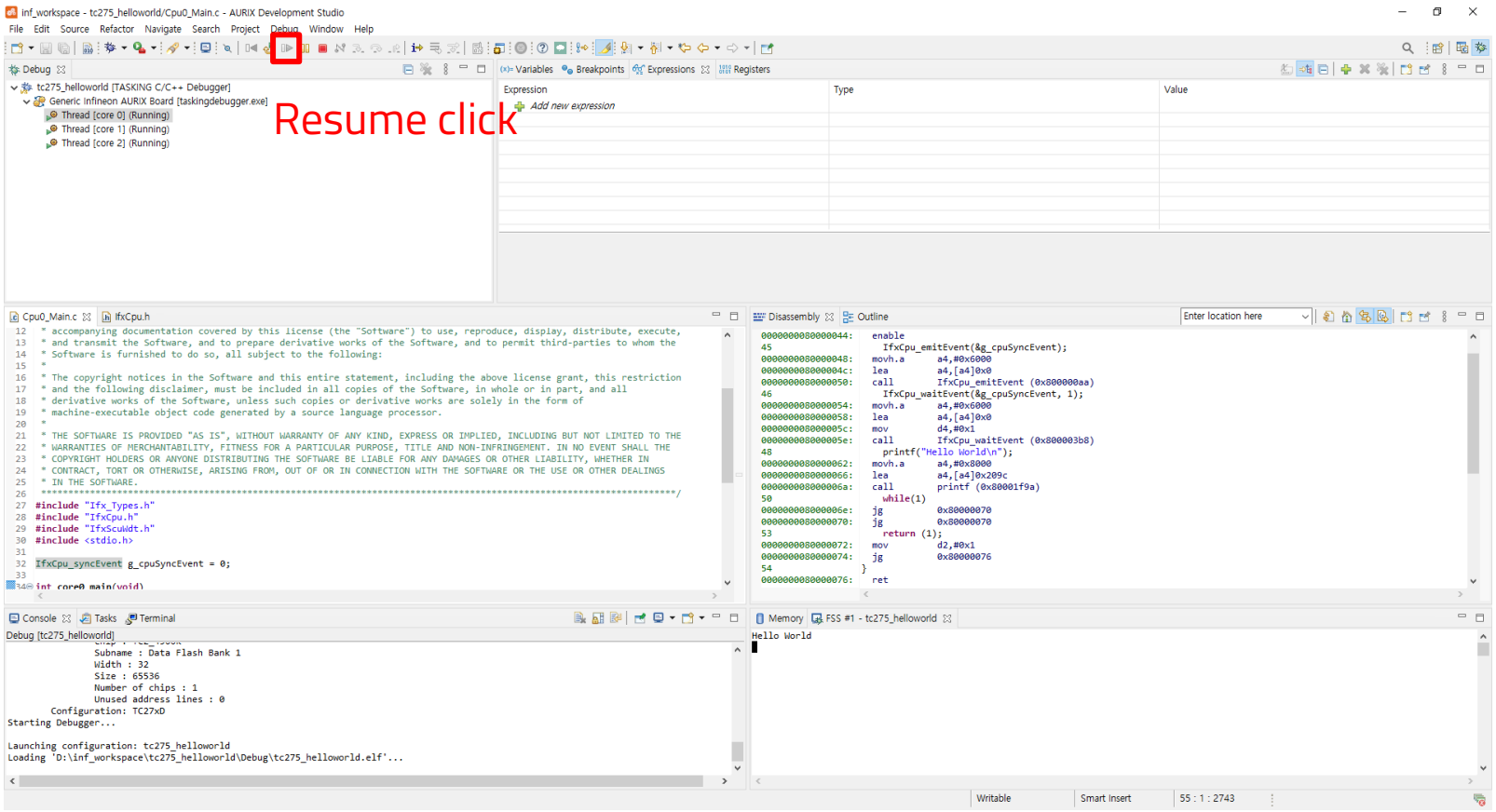
Lab9

: 초음파 센서 사용 위한 configuration 및 trigger 함수

```
342
343 void initUSonic(void)
344 {
345     // USonic 1
346     P15_IOCR4.B.PC4 = 0x01;         // set P15.4 general input (pull-down connected) [Echo]
347     P02_IOCR4.B.PC6 = 0x10;         // set P02.6 push-pull general output            [Trig]
348     P02_OUT.B.P6 = 0x0;
349
350     // Usonic 2
351     P00_IOCR4.B.PC4 = 0x01;         // set P00.4 general input (pull-down connected) [Echo]
352     P02_IOCR4.B.PC5 = 0x10;         // set P02.5 push-pull general output            [Trig]
353     P02_OUT.B.P5 = 0x0;
354 }
355
356 void usonicTrigger_1(void)
357 {
358     // start of 10us Trigger Pulse
359     // GPIO P02.6 --> HIGH
360     P02_OUT.B.P6 = 0x1;
361     range_valid_flag_1 = 0;
362     CCU60_TCTR4.B.T12RS = 0x1;          // T12 start counting
363 }
364
365 void usonicTrigger_2(void)
366 {
367     // start of 10us Trigger Pulse
368     // GPIO P02.5 --> HIGH
369     P02_OUT.B.P5 = 0x1;
370     range_valid_flag_2 = 0;
371     CCU60_TCTR4.B.T13RS = 0x1;          // T13 start counting
372 }
373
```

Build 및 Debug

- 프로젝트 빌드 (ctrl + b)
- 디버그 수행하여 보드에 실행 파일 flash



감사합니다. 휴식~~