마이크로프로세서 응용 보고서 (Interrupt, PWM)

자동차IT융합학과 20183376

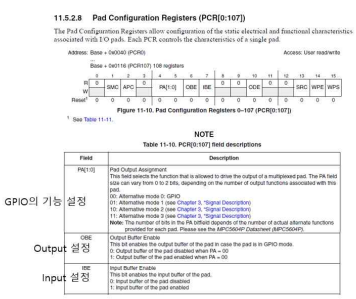
박선재

*주석으로 설명 대체, 칸 수 부족으로 각 비트에 대한 그림 몇 개 제외

1. Interrupt (LED를 한 칸씩 shift(<<1>> => 이전 LED off, <<2>> => 이전 LED 유지))

(include, 함수 선언, 구현하지 않은 함수 사용하는 부분 생략)

```
char LED[4] = { 1, 1, 1, 1 }; //LED
uint32_t Pit1cnt = 0;
int main(void)
{
    init_PIT();
    // Timer 1의 interrupt가 발생하면 PIT1ISR function으로 이동함
    // 6 : priority level
    // 60 : Vector number
    // priority level이 더 큰 것이 우선 동작, 우선 순위가 같으면 Vector Number가 덜 큰 것이 우선 동작
    INTC_InstallINTCInterruptHandler(PIT1ISR, 60, 6);
    RegisterO_set();
    /* Loop forever */
    for (;;)
    {
        LED_ctrl();
    }
}
void RegisterO_set(void)        // Output Register setting
{
    // LED를 Output으로 사용하도록 세팅
    // .B => Register의 특정 비트의 특정 위치, .R => Register all
    // Pad Configuration Registers(Port D)의 OBE bit를 enable하여 output으로 사용하도록 설정
    SIU.PCR[52].R = 0x0200;  // LED1 // Register 전체 중 OBE bit만 따로 enable
    SIU.PCR[53].R = 0x0200;  // LED2
    SIU.PCR[54].R = 0x0200;  // LED3
    SIU.PCR[55].B.OBE = 0b1; // LED4 // OBE bit만 따로 불러와 enable
}
void LED_ctrl(void)             // LED Output
{
    //LED Output (Pad data output)
    SIU.GPDO[52].B.PDO = LED[0]; // LED1
    SIU.GPDO[53].B.PDO = LED[1]; // LED2
    SIU.GPDO[54].B.PDO = LED[2]; // LED3
    SIU.GPDO[55].B.PDO = LED[3]; // LED4
}
void init_PIT(void)             // PIT initial
{
    //Enable PIT and Config Stop in debug mode
    PIT.PITMCR.R = 0x00000001;
    // Register에 설정 된 value에서 system clock의 주기로 0까지
count
    // Timeout = (6.4M) x 1sec / 64M sysclks = 100ms
    PIT.CH[1].LDVAL.R = 6400000;
```



11.5.2.8 Pad Configuration Registers (PCR[0:107])

The Pad Configuration Registers allow configuration of the static electrical and functional characteristics associated with I/O pads. Each PCR controls the characteristics of a single pad.

Address: Base + 0x0040 (PCR0) Access: User read/write

Base + 0x0116 (PCR107) 108 registers

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	SMC	APC	0	PA[1:0]		OBE	IBE	0	0	ODE	0	0	SRC	WPE	WPS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-10. Pad Configuration Registers 0–107 (PCR[0:107])

1 See Table 11-11.

NOTE

Table 11-10. PCR[0:107] field descriptions

Field	Description
PA[1:0]	Pad Output Assignment This field selects the function that is allowed to drive the output of a multiplexed pad. The PA field size can vary from 0 to 2 bits, depending on the number of output functions associated with this pad. 00: Alternative mode 0: GPIO 01: Alternative mode 1 (see Chapter 3, "Signal Description) 10: Alternative mode 2 (see Chapter 3, "Signal Description) 11: Alternative mode 3 (see Chapter 3, "Signal Description) Note: The number of bits in the PA bitfield depends of the number of actual alternate functions provided for each pad. Please see the MPC5604P Datasheet (MPC5604P).
OBE	Output Buffer Enable This bit enables the output buffer of the pad in case the pad is in GPIO mode. 0: Output buffer of the pad disabled when PA = 00 1: Output buffer of the pad enabled when PA = 00
IBE	Input Buffer Enable This bit enables the input buffer of the pad. 0: Input buffer of the pad disabled 1: Input buffer of the pad enabled

GPIO의 기능 설정

Output 설정

Input 설정

Figure 36-2. PIT Module Control Registers (PITMCR)

Table 36-3. PITMCR Field Descriptions

Field	Description
MDIS	Module Disable. This is used to disable the module clock. This bit should be enabled before any other setup is done. 0 Clock for PIT Timers is enabled (default) 1 Clock for PIT Timers is disabled
FRZ	Freeze. Allows the timers to be stopped when the device enters debug mode. 0 = Timers continue to run in debug mode. 1 = Timers are stopped in debug mode.

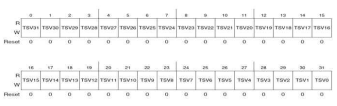


Figure 36-3. Timer Load Value Register (LDVAL)

Table 36-4. LDVAL Field Descriptions

Field	Description
TSVn	Time Start Value Bits. These bits set the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer; instead the value will be loaded once the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again (see Figure 36-8).

```
    //Enable PIT1 Interrupt Enabled & Start, PIT counting
        PIT.CH[1].TCTRL.R = 0x00000003;
}
void PIT1ISR(void) //Interrupt service routine // PIT running
{
    Pit1cnt++; //Interrupt 발생 시 +1씩 연산
    //추가하지 않으면 interrput가 계속 살아 있어서 통신 interrupt를 받지 못함 // Interrupt finish (Clear PIT0 flag)
    PIT.CH[1].TFLG.B.TIF = 1;
    // <<1>> 0부터 Pit1cnt가 순차적으로 증가되기 때문에 %4 연산 시 0, 1, 2, 3이 순차적으로 도출.
    //        LED가 순차적으로 on, off됨 (이전 LED off)
    LED[0] = !(Pit1cnt % 4 == 3);
    LED[1] = !(Pit1cnt % 4 == 2);
    LED[2] = !(Pit1cnt % 4 == 1);
    LED[3] = !(Pit1cnt % 4 == 0);
    // <<2>> 0부터 Pit1cnt가 순차적으로 증가되기 때문에 %5 연산 시 0, 1, 2, 3, 4가 순차적으로 도출.
    //       LED가 순차적으로 on됨(이전 LED on유지)
    LED[0] = !(Pit1cnt%5>=4);
    LED[1] = !(Pit1cnt%5>=3);
    LED[2] = !(Pit1cnt%5>=2);
    LED[3] = !(Pit1cnt%5>=1);
}
```

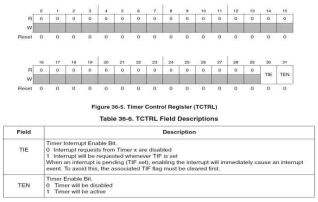
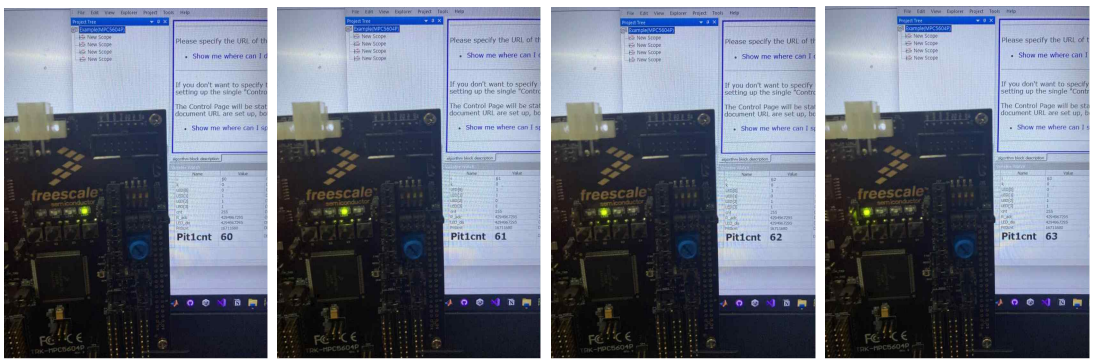


Figure 36-5. Timer Control Register (TCTRL)

Table 36-6. TCTRL Field Descriptions

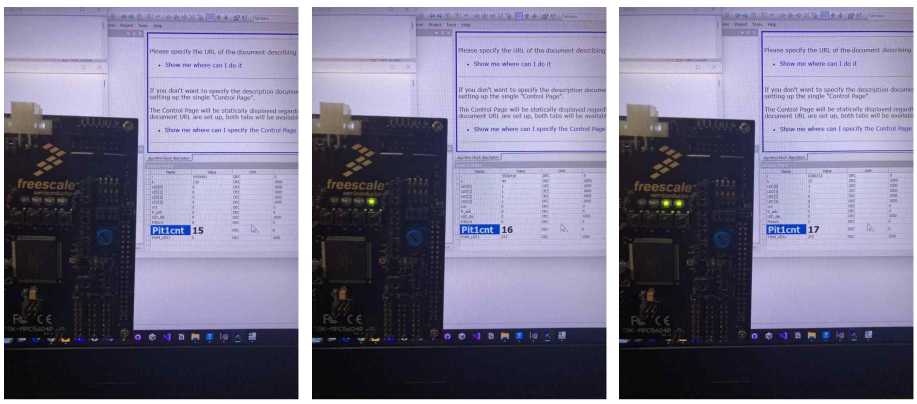
Field	Description
TIE	Timer Interrupt Enable Bit 0 Interrupt requests from Timer x are disabled 1 Interrupt will be requested whenever TIF is set When an interrupt is pending (TIF set), enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TIF flag must be cleared first.
TEN	Timer Enable Bit 0 Timer will be disabled 1 Timer will be active

<<1>>

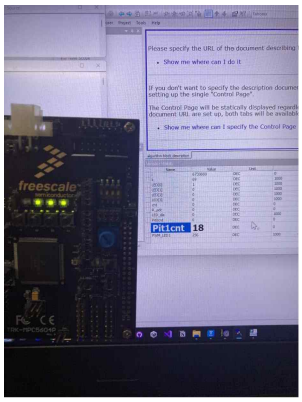


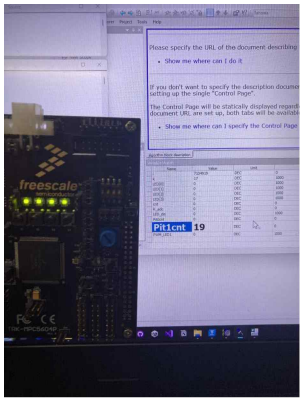
=> Pit1cnt%4 == 0 실행 => Pit1cnt%4 == 1 실행 => Pit1cnt%4 == 2 실행 => Pit1cnt%4 == 3 실행

<<2>>



=> 아무 조건도 걸리지 않음 => Pit1cnt%5 >= 1 실행 => Pit1cnt%5 >= 2 실행



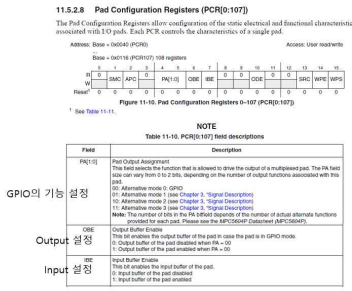


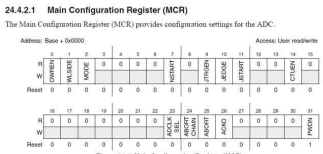
=> Pit1cnt%5 >= 3 실행 => Pit1cnt%5 >= 4 실행

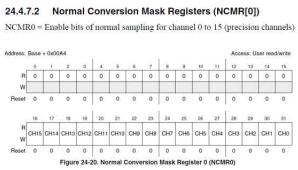
2. PWM(가변저항에 따른 LED 밝기 조정)

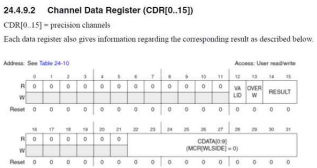
(include, 함수 선언, 구현하지 않은 함수 사용하는 부분 생략)

```
char LED[4] = { 1, 1, 1, 1 }; //LED int R_adc = 0; int LED_dis = 0; uint16_t PWM_LED1 = 6400;
int main(void)
{
    init_ADC1();
    Init_FlexPWM();
    RegisterO_set();
    /* Loop forever */
    for (;;)
    {
        ADCRead_1();
        LED_ctrl();
        PWM_out();
    }
}
void RegisterO_set(void)    // Output Register setting
{
    // LED를 Output으로 사용하도록 세팅 .R => Register all
    // Pad Configuration Registers(Port D)의 OBE bit를 enable하여 output으로 사용하도록 설정
    SIU.PCR[52].R = 0x0E00; // LED1 // Register 전체 중 OBE bit만 따로 enable
    SIU.PCR[64].R = 0x2400;              //가변저항으로 사용
}
void LED_ctrl(void)         // LED Output
{
    //LED Output (Pad data output)
    SIU.GPDO[52].B.PDO = LED[0]; // LED1
}
void init_ADC1(void)            // 가변저항 initial
{
    //ADC 모듈에 대한 기본적인 설정
    ADC_1.MCR.B.ABORT = 1;      //Abort ADC_1
    ADC_1.MCR.B.OWREN = 0;     //disable overwritting
    ADC_1.MCR.B.WLSIDE = 0;    //conversion data is written right_aligned
    ADC_1.MCR.B.MODE = 0;      //One Shot mode
    ADC_1.MCR.B.CTUEN = 0;    //disable CTU triggered
```

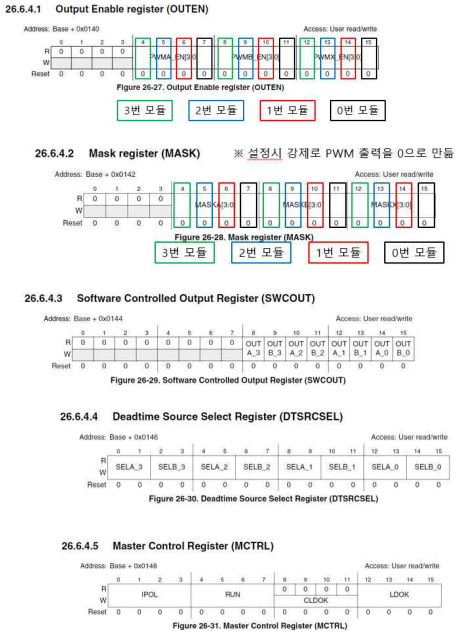




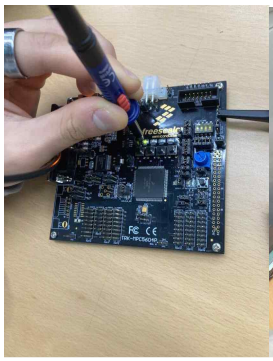
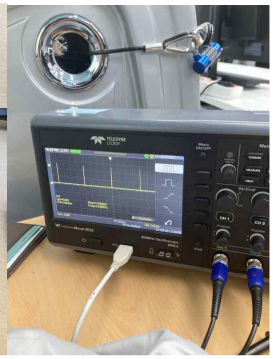




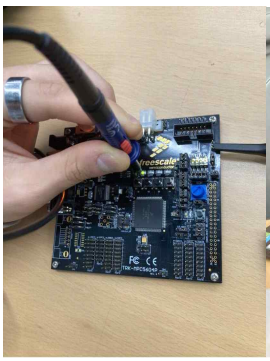
```
    ADC_1.MCR.B.ADCLKSEL = 0; //Set ADClock 32MHz
    ADC_1.MCR.B.ACK0 = 0;     //disable auto clock off
    ADC_1.MCR.B.PWDN = 0;     //disable power down mode
    //ADC sampling 수에 따른 변환 시간 설정
    ADC_1.CTR[0].R = 0x00008208;
    //ADC Channel 사용 설정
    ADC_1.NCMR[0].R = 0x00000020;
    //ADC 데이터 저장 변수 초기화
    ADC_1.CDR[0].R = 0x00000000;
    ADC_1.MCR.B.ABORT = 0; //Exit Abort ADC_1

}
void ADCRead_1(void)          // 가변저항 value read
{
    ADC_1.MCR.B.NSTART = 1;
    asm("nop");       //Assembly어 -> 시작 전까지 기다려라
    //변환이 끝나면 Data를 input (1: 변환중, 0: 변환완료)
    while (ADC_1.MCR.B.NSTART) asm("nop");
    R_adc = ADC_1.CDR[5].B.CDATA; //가변저항값
}
void Init_FlexPWM(void)         // pwm  initial
{
    FLEXPWM_0.OUTEN.B.PWMB_EN = 0b1000; //PWM B Output Enabled Submodule 3
    FLEXPWM_0.MASK.R = 0x0000;       //Mask를 사용하지 않는다
    FLEXPWM_0.SWCOUT.B.OUTB_3 = 1; //S/W Controlled O/P OUTB_3 Enable
    // module 활성화
    FLEXPWM_0.MCTRL.B.LDOK |= 0xF; // Load update (Load config values into buffers)
    FLEXPWM_0.MCTRL.B.RUN |= 0xF;  // PWM start (1, 2, 3, 4 RUN)
    //10kHZ로 update
    FLEXPWM_0.SUB[3].INIT.R = 0;                  // 캐리어파의 시작점
    FLEXPWM_0.SUB[3].VAL[0].R = 3200;             // 캐리어파의 중간값
    FLEXPWM_0.SUB[3].VAL[1].R = 6400;             // 캐리어파의 끝점
    //독립 제어
    FLEXPWM_0.SUB[3].CTRL2.B.INDEP = 1; // B를 독립적으로 사용(A와 함께 사용 X)
    // full, half enable
    // PWM의 값을 Half와 Full에서 update함.
    FLEXPWM_0.SUB[3].CTRL.B.HALF = 1;             // Half에서 update
    FLEXPWM_0.SUB[3].CTRL.B.FULL = 1;             // Full에서 update
    FLEXPWM_0.SUB[3].DISMAP.B.DISB = 0; //Fault시 data를 차단함.
    FLEXPWM_0.SUB[3].DISMAP.B.DISA = 0; //Fault시 data를 차단함.
}
void PWM_out(void)            // 가변저항에 대한 pwm 변화
{
     //가변저항의 최대값이 1023이고 PWM의 캐리어파 끝점을 6400으로 설정했기 때문에 가변저항이 커질 수록
LED의 빛이 어두워짐
    PWM_LED1 = (unsigned short)(6400 * R_adc / 1023);
    FLEXPWM_0.SUB[3].VAL[4].R = 0; //4, 5번 관장 => PWMB
    FLEXPWM_0.SUB[3].VAL[5].R = PWM_LED1; //PWMB의 5번에 값을 출력해줌
    FLEXPWM_0.MCTRL.B.LDOK |= 0xF;  // Load update (Load config values into buffers)
    FLEXPWM_0.MCTRL.B.RUN |= 0xF;  // PWM start (1, 2, 3, 4 RUN)
}
```



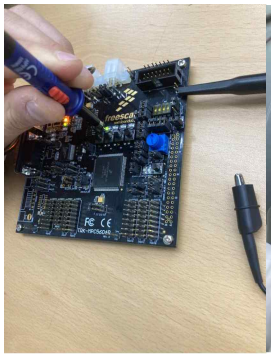
<<PWM Duty>>

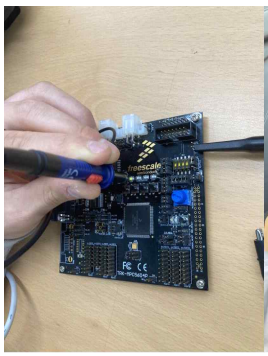
=> Duty : 0 (LED) => Duty : 0 (OSIL)



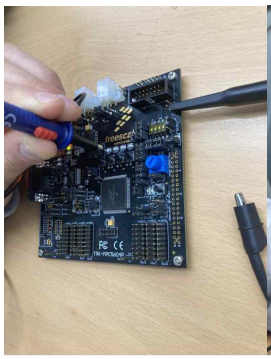
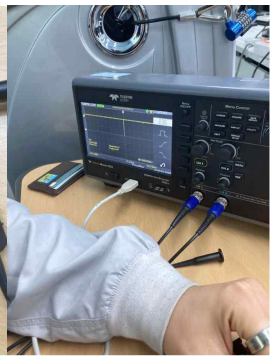

=> Duty : 1600 (LED) => Duty : 1600 (OSIL)



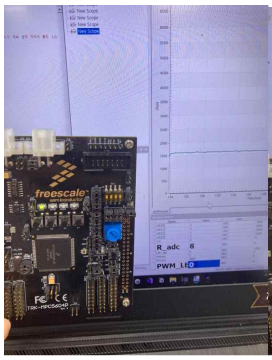
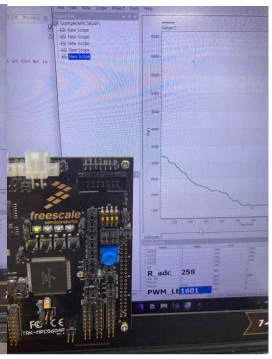

=> Duty : 3200 (LED) => Duty : 3200 (OSIL)



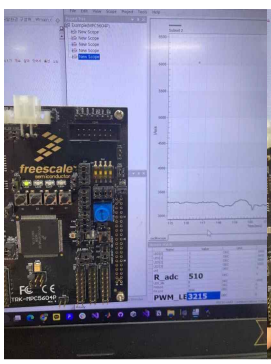
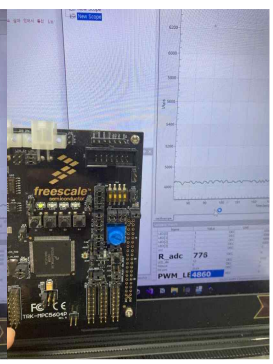

=> Duty : 4800 (LED) => Duty : 4800 (OSIL)

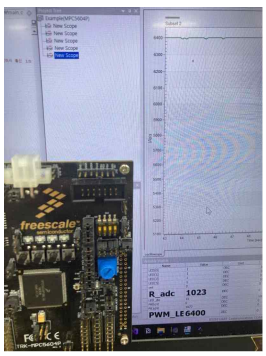
=> Duty : 6400 (LED) => Duty : 6400 (OSIL)

=> Duty : 0 (LED, N) => Duty : 1600 (LED, N)

=> Duty : 3200 (LED, N) => Duty : 4800 (LED, N)



=> Duty : 6400 (LED, N)