

임베디드시스템설계

Embedded System Capstone Design

ICE3015-001



과제 4

실습팀 4

정보통신공학과 12181855 황용하

정보통신공학과 12191720 곽재현

정보통신공학과 12191765 박승재

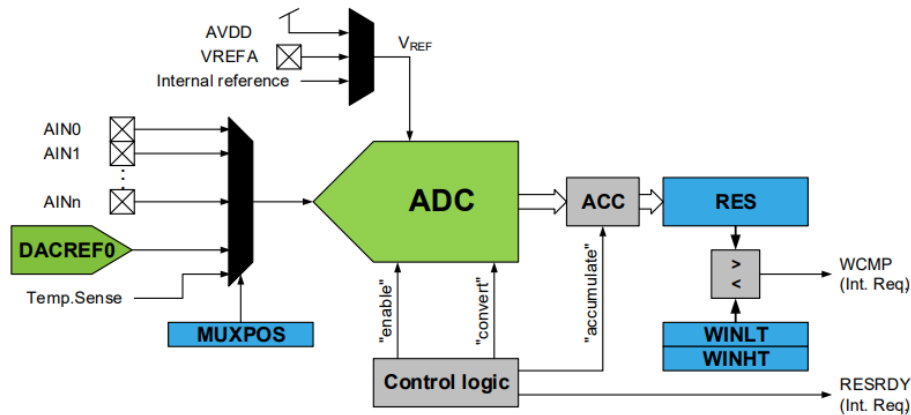
Introduction

ADC

아날로그 신호를 디지털 신호로 변환해주는 장치이다.

Block Diagram

Figure 29-1. Block Diagram



MUXPOS 를 통해 Input source 를 선택한다. ADC 는 기본적으로 10bit 결과값을 출력하며 값은 오른쪽으로 정렬되어 있다.

Signal Description

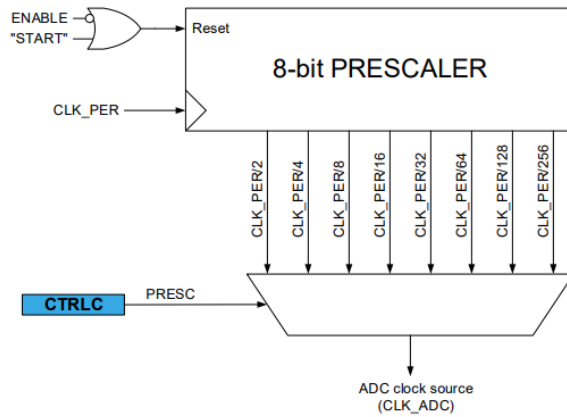
Pin Name	Type	Description
AIN[n:0]	Analog input	Analog input pin
VREFA	Analog input	External voltage reference pin

ADC 의 출력은 $1023 * V_{in} / V_{ref}$ 로 계산된다. ADC 초기화 방법은 아래와 같다.

1. ADC 해상도를 설정합니다.
2. (선택) Free-Running 모드를 설정합니다.
3. (선택) 변환당 누적 샘플 수를 설정합니다.
4. Vref 를 설정합니다. 기본값은 내부 Vref 와 동일합니다.
5. Prescaler 비트 필드를 수정하여 CLK_ADC 를 설정합니다.
6. MUXPOS 를 수정하여 입력 받을 값을 설정합니다.
7. (선택) Start Event 를 활성화합니다.
8. ADC 를 활성화합니다.

Clock Generation

Figure 29-2. ADC Prescaler



Prescaler 를 이용해 clock 속도를 조절하여 변환 시간 간격을 수정할 수 있다.

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0	RUNSTBY					RESSEL	FREERUN	ENABLE
0x01	CTRLB	7:0							SAMPNUM[2:0]	
0x02	CTRLC	7:0		SAMPCAP	REFSEL[1:0]				PRESC[2:0]	
0x03	CTRLD	7:0		INITDLY[2:0]		ASDV			SAMPDLY[3:0]	
0x04	CTRLF	7:0							WINCM[2:0]	
0x05	SAMPCTRL	7:0						SAMPLEN[4:0]		
0x06	MUXPOS	7:0						MUXPOS[4:0]		
0x07	Reserved									
0x08	COMMAND	7:0								STCONV
0x09	EVCTRL	7:0								STARTEI
0x0A	INTCTRL	7:0							WCMP	RESRDY
0x0B	INTFLAGS	7:0							WCMP	RESRDY
0x0C	DBGCTRL	7:0								DBGGRUN
0x0D	TEMP	7:0					TEMP[7:0]			
0x0E	Reserved									
...										
0x0F										
0x10	RES	7:0					RES[7:0]			
		15:8					RES[15:8]			
0x12	WINLT	7:0					WINLT[7:0]			
		15:8					WINLT[15:8]			
0x14	WINHT	7:0					WINHT[7:0]			
		15:8					WINHT[15:8]			
0x16	CALIB	7:0								DUTYCYC

ADC 의 레지스터 offset 이다. 확인해 볼 비트는 RESSEL, ENABLE, REFSEL, MUXPOS, STCONV, RES 이다.

Bit 2 – RESSEL Resolution Selection

This bit selects the ADC resolution.

Value	Description
0	Full 10-bit resolution. The 10-bit ADC results are accumulated or stored in the ADC Result (ADC.RES) register.
1	8-bit resolution. The conversion results are truncated to eight bits (MSbs) before they are accumulated or stored in the ADC Result (ADC.RES) register. The two Least Significant bits (LSbs) are discarded.

RESSEL 은 해상도를 설정하는 값이며 기본값은 10bit resolution 이다.

Bits 5:4 – REFSEL[1:0] Reference Selection

These bits select the voltage reference for the ADC.

Value	Name	Description
0x0	INTERNAL	Internal reference
0x1	VDD	V _{DD}
0x2	VREFA	External reference V _{REFA}
Other	-	Reserved

REFSEL 는 Vref 를 설정한다. 기본값은 내부 Vref 이다.

Bits 4:0 – MUXPOS[4:0] MUXPOS

This bit field selects which single-ended analog input is connected to the ADC. If these bits are changed during a conversion, the change will not take effect until this conversion is complete.

MUXPOS	Name	Input
0x00-0x0F	AIN0-AIN15	ADC input pin 0 - 15
0x10-0x1B	-	Reserved
0x1C	DACREF0	DAC reference in AC0
0x1D	-	Reserved
0x1E	TEMPSENSE	Temperature sensor
0x1F	GND	GND
Other	-	Reserved

MUXPOS 는 어떤 source 의 값을 ADC 에 넣을지 선택한다.

29.5.8 Command

Name: COMMAND
Offset: 0x08
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								STCONV
Access								R/W
Reset								0

Bit 0 – STCONV Start Conversion

Writing a '1' to this bit will start a single measurement. If in Free-Running mode, this will start the first conversion. STCONV will read as '1' as long as a conversion is in progress. When the conversion is complete, this bit is automatically cleared.

STCONV 가 1 이 되면 ADC 가 변환을 시작한다.

29.5.14 Result

Name: RES
Offset: 0x10
Reset: 0x00
Property: -

The ADCn.RESL and ADCn.RESH register pair represents the 16-bit value, ADCn.RES. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01.

If the analog input is higher than the reference level of the ADC, the 10-bit ADC result will be equal the maximum value of 0x3FF. Likewise, if the input is below 0V, the ADC result will be 0x000. As the ADC cannot produce a result above 0x3FF values, the accumulated value will never exceed 0xFFC0 even after the maximum allowed 64 accumulations.

Bit	15	14	13	12	11	10	9	8
	RES[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
	RES[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – RES[15:8] Result high byte

These bits constitute the MSB of the ADCn.RES register, where the MSb is RES[15]. The ADC itself has a 10-bit output, ADC[9:0], where the MSb is ADC[9]. The data format in ADC and Digital Accumulation is 1's complement, where 0x0000 represents the zero, and 0xFFFF represents the largest number (full scale).

Bits 7:0 – RES[7:0] Result low byte

These bits constitute the LSB of ADC/Accumulator Result, (ADCn.RES) register. The data format in ADC and Digital Accumulation is 1's complement, where 0x0000 represents the zero, and 0xFFFF represents the largest number (full scale).

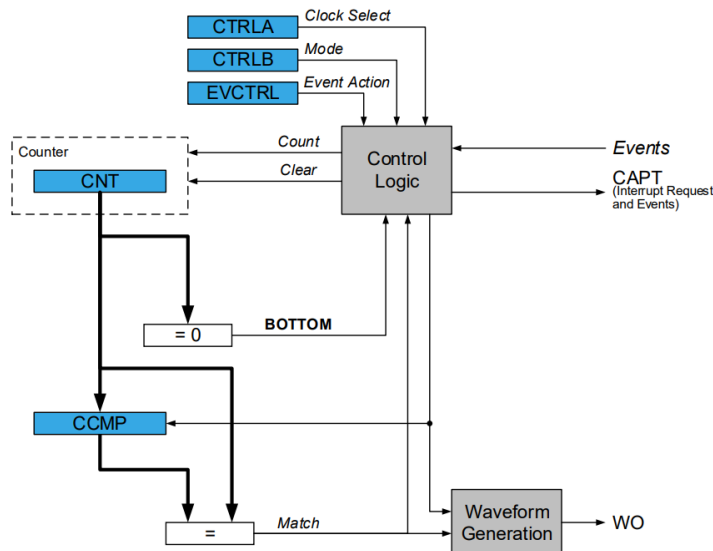
RES 는 RESL 과 RESH 로 나뉘며 각각 8 비트(unsigned char)이다. 따라서 RES 는 16 비트 unsigned short 이다. 해상도가 10bit 라면 RESL 로 전부 표현할 수 없기 때문에 RES 를 사용해야 한다.

TCB

TCB 는 16-Bit Timer/Counter Type B 를 의미한다. ATmega4809 에는 1 개의 TCA 와 4 개의 TCB 를 가지고 있다. Timer 에는 A 타입과 B 타입이 있는데, A 타입은 다양한 모드를 지원하며 일반적인 목적으로 사용하는 범용 타이머이고 B 타입은 정밀하게 PWM 을 제어하기 위한 타이머이다.

Block Diagram

Figure 21-1. Timer/Counter Type B Block



실습에서 사용하는 TCB0 는 A2 핀으로 출력을 보내기 때문에 해당 핀을 OUTPUT 으로 설정해야 한다.

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	CTRLA	7:0		RUNSTDBY		SYNCUPD		CLKSEL[1:0]		ENABLE
0x01	CTRLB	7:0		ASYNC	CCMPINIT	CCMPEN			CNTMODE[2:0]	
0x02	Reserved									
...										
0x03										
0x04	EVCTRL	7:0		FILTER		EDGE				CAPTEI
0x05	INTCTRL	7:0								CAPT
0x06	INTFLAGS	7:0								CAPT
0x07	STATUS	7:0								RUN
0x08	DBGCTRL	7:0								DBGRUN
0x09	TEMP	7:0	TEMP[7:0]							
0x0A	CNT	7:0	CNT[7:0]							
		15:8	CNT[15:8]							
0x0C	CCMP	7:0	CCMP[7:0]							
		15:8	CCMP[15:8]							

TCB 의 레지스터 offset 이다. 확인해 볼 비트는 CLKSEL, ENABLE, CCMPEN, CNTMODE, CAPT, CCMP 이다.

21.5.1 Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		RUNSTDBY		SYNCUPD		CLKSEL[1:0]		ENABLE
Access		R/W		R/W		R/W	R/W	R/W
Reset		0		0		0	0	0

Bits 2:1 – CLKSEL[1:0] Clock Select

Writing these bits selects the clock source for this peripheral.

Value	Name	Description
0x0	CLKDIV1	CLK_PER
0x1	CLKDIV2	CLK_PER / 2
0x2	CLKTCA	Use CLK_TCA from TCA0
0x3		Reserved

Bit 0 – ENABLE Enable

Writing this bit to '1' enables the Timer/Counter type B peripheral.

CLKSEL 은 TCB 에서 사용하는 클럭을 설정한다. TCB_CLKSEL_CLKDIV2_gc 를 넣으면 사용하는 클럭이 절반이 된다. (더 느려진다)

실습에서는 최대한 천천히 TCB 를 동작시키기 위해 메인 클럭도 최소값으로 설정했다.

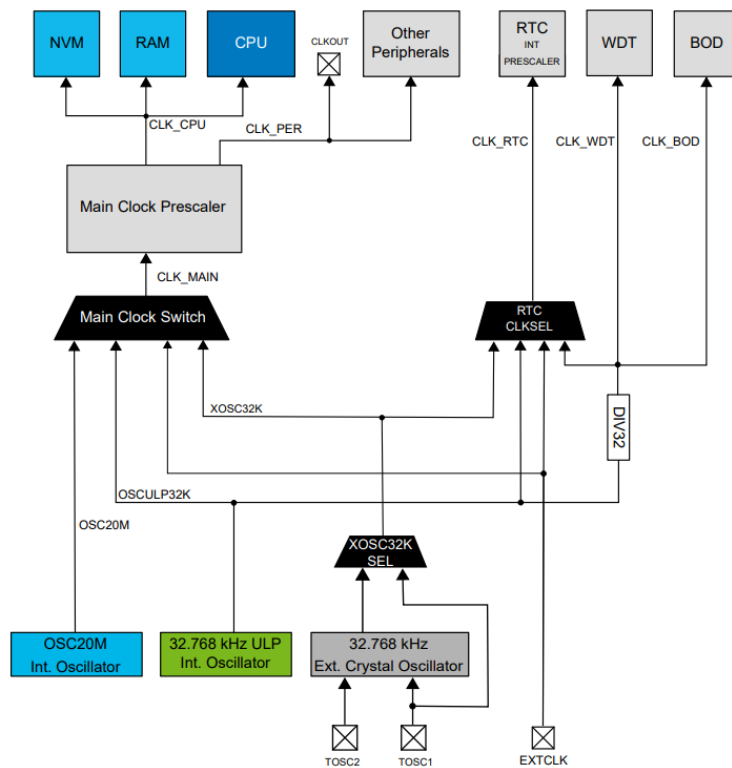
```
CPU_CCP = CCP_IOREG_gc;
CLKCTRL_MCLKCTRLB = CLKCTRL_PDIV_64X_gc | CLKCTRL_PEN_bm; // Enable
Prescaler and set Prescaler Division to 64
CLKCTRL_MCLKCTRLA = CLKCTRL_CLKSEL_OSCULP32K_gc; // Select 32KHz Internal
Ultra Low Power Oscillator (OSCULP32K)
while (CLKCTRL_MCLKSTATUS & CLKCTRL_SOSC_bm); // Wait for system oscillator
changing to finish
```

CCP_IOREG_gc 는 Protected register 값을 수정할 수 있도록 설정한다.

10. CLKCTRL - Clock Controller

10.1 Features

- All Clocks and Clock Sources are Automatically Enabled when Requested by Peripherals
- Internal Oscillators:
 - 16/20 MHz oscillator (OSC20M)
 - 32.768 kHz ultra low-power oscillator (OSCULP32K)
- External Clock Options:
 - 32.768 kHz crystal oscillator (XOSC32K)
 - External clock
- Main Clock Features:
 - Safe run-time switching
 - Prescaler with 1x to 64x division in 12 different settings



CLKCTRL_PDIV_64X_gc 으로 1/64 배 Prescaler 를 활성화했고, CLKCTRL_CLKSEL_OSCULP32K_gc 으로 주파수가 가장 낮은 Oscillator 를 사용하도록 했다.

21.5.2 Control B

Name: CTRLB
Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		ASYNC	CCMPINIT	CCMPEN		CNTMODE[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bit 4 – CCMPEN Compare/Capture Output Enable

Writing this bit to '1' enables the waveform output. This will make the waveform output available on the corresponding pin, overriding the value in the corresponding PORT output register. The corresponding pin direction must be configured as an output in the PORT peripheral.

Value	Description
0	Waveform output is not enabled on the corresponding pin
1	Waveform output will override the output value of the corresponding pin

Bits 2:0 – CNTMODE[2:0] Timer Mode

Writing these bits selects the Timer mode.

Value	Name	Description
0x0	INT	Periodic Interrupt mode
0x1	TIMEOUT	Time-out Check mode
0x2	CAPT	Input Capture on Event mode
0x3	FRQ	Input Capture Frequency Measurement mode
0x4	PW	Input Capture Pulse-Width Measurement mode
0x5	FRQPW	Input Capture Frequency and Pulse-Width Measurement mode
0x6	SINGLE	Single-Shot mode
0x7	PWM8	8-Bit PWM mode

CCMPEN 은 지정된 핀으로 값을 출력하도록 설정한다. TCB0 은 A2 와 연결되어 있으므로 해당 핀으로 값이 출력될 것이다.

CNTMODE 은 타이머 모드를 설정한다. TCB_CNTMODE_PWM8_gc 은 TCB 가 PWM 을 만들도록 명령한다.

21.5.4 Interrupt Control

Name: INTCTRL
Offset: 0x05
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								CAPT
Access								R/W
Reset								0

Bit 0 – CAPT Capture Interrupt Enable
 Writing this bit to '1' enables interrupt on capture.

CAPT 가 1 이면 인터럽트를 활성화한다.

Table 21-6. Interrupt Sources Set Conditions by Counter Mode

Counter Mode	Interrupt Set Condition	TOP Value	CAPT
8-Bit PWM mode	Set when the counter reaches CCML	CCML	CNT == CCML

8-bit PWM 모드에서는 CNT == CCML 일 때 인터럽트가 호출된다.

```
ISR(TCB0_INT_vect) {  
    }  
}
```

인터럽트를 받는 코드는 위와 같다.

21.5.10 Capture/Compare

Name: CCMP
Offset: 0x0C
Reset: 0x00
Property: -

The TCBn.CCMPL and TCBn.CCMPH register pair represents the 16-bit value TCBn.CCMP. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01.

This register has different functions depending on the mode of operation:

- For Capture operation, these registers contain the captured value of the counter at the time the capture occurs
- In Periodic Interrupt/Time-Out and Single-Shot mode, this register acts as the TOP value
- In 8-bit PWM mode, TCBn.CCMPL and TCBn.CCMPH act as two independent registers: The period of the waveform is controlled by CCMPH, while CCMPH controls the duty cycle.

Bit	15	14	13	12	11	10	9	8
	CCMP[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
	CCMP[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – CCMP[15:8] Capture/Compare Value High Byte

These bits hold the MSB of the 16-bit compare, capture, and top value.

Bits 7:0 – CCMP[7:0] Capture/Compare Value Low Byte

These bits hold the LSB of the 16-bit compare, capture, and top value.

CCMP 는 16 비트 값으로 ADC 의 RES 와 마찬가지로 CCMPH 와 CCMPH 로 나뉜다. ADC 와 다른점은 High 와 Low 이 한 세트가 아니라 역할이 나뉘어 있다. CCMPH 는 PWM 에서 duty 를 의미하며 CCMPH 은 top 을 의미한다.

15.3.6 PORTMUX Control for TCB

Name: TCBROUTEA
Offset: 0x05
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					TCB3	TCB2	TCB1	TCB0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 0 – TCB0 TCB0 Output

Write this bit to '1' to select alternative output pin for 16-bit timer/counter B 0.

Value	Name	Description
0x0	DEFAULT	TCB0 on PA2
0x1	ALT1	TCB0 on PF4

TCB0 은 A2 와 연결되어 있다고 했는데 TCBROUTEA 를 이용해서 F4 로 값을 출력하도록 변경할 수 있다. A2 와 F4 이외의 다른 핀으로는 값을 출력할 수 없다.

Problem

Write a review of contents of the practice.

Result

ADC – Single conversion

```
#define F_CPU 3333333

#include <avr/io.h>
#include <stdio.h>
#include <util/delay.h>

#define BAUD_RATE 9600
#define USART_BAUD_RATE(BAUD_RATE) ((float) (F_CPU * 64 / (16 * (float) BAUD_RATE)) + 0.5)

void USART3_init(void) {
    PORTB_DIR |= PIN0_bm; // B0: Output
    PORTB_DIR &= ~PIN1_bm; // B1: Input
    USART3_BAUD = (unsigned short) USART_BAUD_RATE(BAUD_RATE);
    USART3_CTRLB |= USART_TXEN_bm | USART_RXEN_bm;
}

void USART3_transmit(unsigned char c) {
    while (!(USART3_STATUS & USART_DREIF_bm));
    USART3_TXDATA = c;
}

int USART3_transmit2(char c, FILE* _) {
    USART3_transmit(c);
    return 0;
}

FILE USART_stream = FDEV_SETUP_STREAM(USART3_transmit2, NULL,
_FDEV_SETUP_WRITE);

void adc_init(void) {
    PORTD_PIN0CTRL = PORT_ISC_INPUT_DISABLE_gc; // PD0: Disable digital
input buffer
    ADC0_CTRLA = ADC_RESSEL_8BIT_gc; // 8bit resolution
    ADC0_MUXPOS = 0x00;
    ADC0_CTRLA |= ADC_ENABLE_bm;
```

```

}

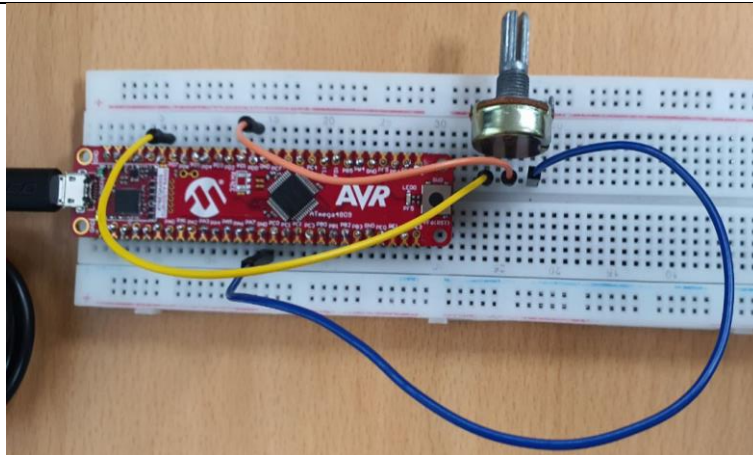
unsigned char adc_read(void) {
    ADC0_COMMAND |= ADC_STCONV_bm; // Start conversion
    while (!(ADC0_COMMAND & ADC_STCONV_bm);
    return ADC0_RES0;
}

int main(void) {
    // Init serial printf
    USART3_init();
    stdout = &USART_stream;

    // Init ADC
    adc_init();

    // Read ADC
    while (1) {
        unsigned char Ain = adc_read();
        double vref = 3.3;
        unsigned int Ain_mV = vref * Ain * 1000 / 255;
        printf("Ain value: %d\r\n", Ain);
        printf("Ain voltage: %d [mV]\r\n", Ain_mV);
        _delay_ms(1000);
    }
    return 0;
}

```



이 코드는 ATmega4809 에서 아날로그 값을 읽어와 ADC 모듈을 통해 디지털 값으로 변환한 후, USART 을 사용해 Serial 모니터에 들어온 전압 값을 보여주는 코드이다.

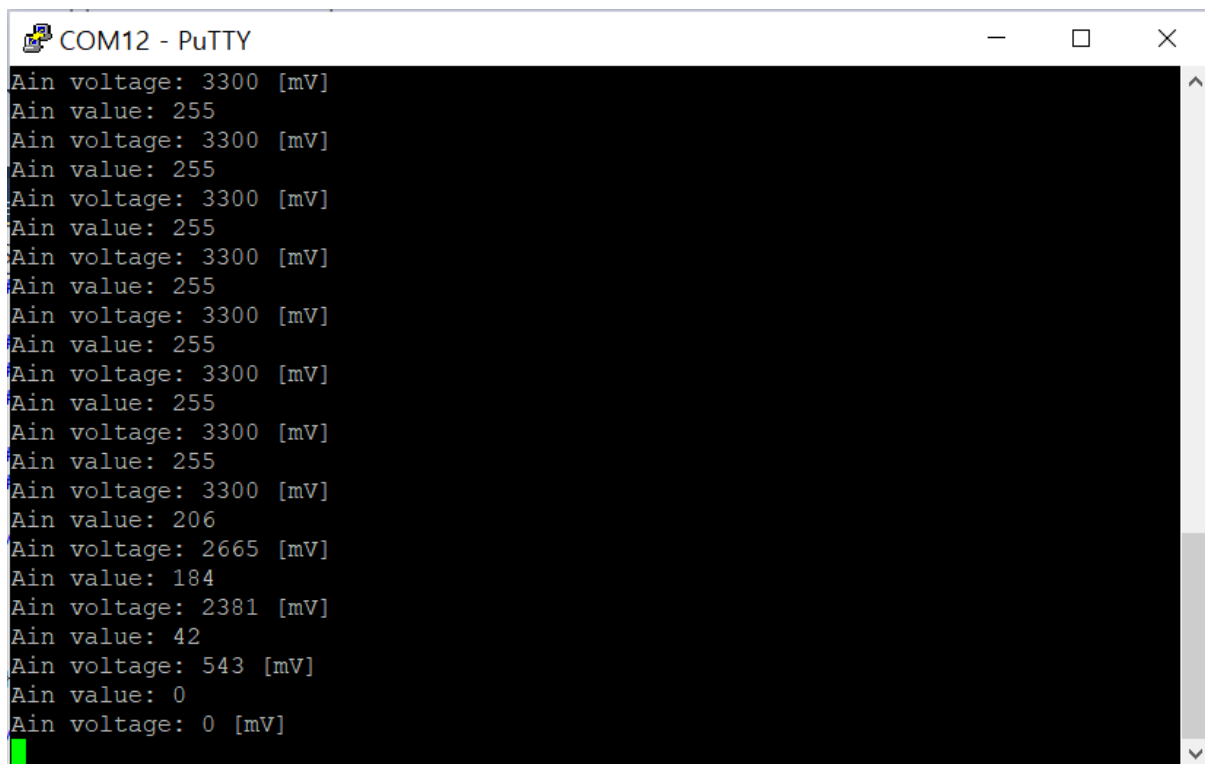
USART3_init 함수에서 USART 을 초기화한다. B0 는 데이터를 전송하기 위한 출력으로 설정하고, B1 은 데이터를 수신하기 위한 입력으로 설정한다. USART3_BAUD = (unsigned short)

`USART_BAUD_RATE(BAUD_RATE)`; 이 코드를 사용해 전송 속도를 설정한다. `USART3_transmit` 함수는 코드는 전송 버퍼가 비어있고, 새 데이터가 수신 가능할 때 `USART` 모듈을 사용해 데이터를 송신한다.

`adc_read(unsigned char select)` 함수는 아날로그 입력핀에서 읽어온 아날로그 값을 디지털로 변환을 위해 사용된다. `ADC0_COMMAND |= ADC_STCONV_bm`;으로 ADC 에게 변환을 시작하도록 명령하면 `ADC0_RES1` 에 디지털 값이 들어오게 된다.

main 함수는 ADC 모듈을 초기화하고, `Ain` 변수에 아날로그 값을 저장해 `vref * Ain * 1000 / 255`;으로 아날로그 값으로부터 전압 값[mV]을 계산한다. ATmega4809 의 `printf`에서는 %f 로 실수 형태의 숫자 출력이 안되기 때문에 1000 을 곱해 mV 단위로 변환했다.

가변 저항을 돌리게 되면 Serial 모니터에 0mV 부터 3300mV 까지의 값이 1 초마다 출력되게 된다.



```
COM12 - PuTTY
Ain voltage: 3300 [mV]
Ain value: 255
Ain voltage: 3300 [mV]
Ain value: 255
Ain voltage: 3300 [mV]
Ain value: 255
Ain voltage: 3300 [mV]
Ain value: 255
Ain voltage: 3300 [mV]
Ain value: 255
Ain voltage: 3300 [mV]
Ain value: 255
Ain voltage: 3300 [mV]
Ain value: 255
Ain voltage: 3300 [mV]
Ain value: 206
Ain voltage: 2665 [mV]
Ain value: 184
Ain voltage: 2381 [mV]
Ain value: 42
Ain voltage: 543 [mV]
Ain value: 0
Ain voltage: 0 [mV]
```

ADC – 10bit conversion

```
#define F_CPU 333333

#include <avr/io.h>
#include <stdio.h>
#include <util/delay.h>

#define BAUD_RATE 9600
#define USART_BAUD_RATE(BAUD_RATE) ((float) (F_CPU * 64 / (16 * (float)
```

```

BAUD_RATE)) + 0.5)

void USART3_init(void) {
    PORTB_DIR |= PIN0_bm; // B0: Output
    PORTB_DIR &= ~PIN1_bm; // B1: Input
    USART3_BAUD = (unsigned short) USART_BAUD_RATE(BAUD_RATE);
    USART3_CTRLB |= USART_TXEN_bm | USART_RXEN_bm;
}

void USART3_transmit(unsigned char c) {
    while (!(USART3_STATUS & USART_DREIF_bm));
    USART3_TXDATA = c;
}

int USART3_transmit2(char c, FILE* _) {
    USART3_transmit(c);
    return 0;
}

FILE USART_stream = FDEV_SETUP_STREAM(USART3_transmit2, NULL,
_FDEV_SETUP_WRITE);

void adc_init(void) {
    PORTD_PIN2CTRL = PORT_ISC_INPUT_DISABLE_gc; // PD2: Disable digital
input buffer
    ADC0_CTRLA = ADC_RESSEL_10BIT_gc; // 10bit resolution
    ADC0_CTRLC |= ADC_REFSEL_VDDREF_gc;
    ADC0_CTRLA |= ADC_ENABLE_bm;
}

unsigned short adc_read(unsigned char select) {
    ADC0_MUXPOS = select;
    ADC0_COMMAND |= ADC_STCONV_bm; // Start conversion
    while (!(ADC0_COMMAND) & ADC_STCONV_bm);
    return ADC0_RES;
}

int main(void) {
    // Init serial printf
    USART3_init();
    stdout = &USART_stream;

    // Init ADC
    adc_init();

    // Read ADC
    while (1) {

```

ADC의 해상도를 10bit로 변경한 코드이다. ADC의 출력이 10bit이기 때문에 `adc_read`의 반환 타입도 `unsigned char`에서 `unsigned short`로 수정하고 `ADC0_RES`을 출력하도록 했다. `main` 함수도 거의 동일하지만 읽어온 값의 타입을 `unsigned short`로 바꿔주는 것을 주의해야 한다.

[illegible]

TCB – Basic TCB

```
#define F_CPU 3333333

#include <avr/io.h>
#include <util/delay.h>

int main(void) {
    // Init clock
    CPU_CCP = CCP_IOREG_gc; // Enable writing to protected register
    CLKCTRL MCLKCTRLB = CLKCTRL PDIV 64X_gc | CLKCTRL PEN bm; // Enable
```

Prescaler and set Prescaler Division to 64

```
CLKCTRL_MCLKCTRLA = CLKCTRL_CLKSEL_OSCULP32K_gc; // Select 32KHz
Internal Ultra Low Power Oscillator (OSCULP32K)
while (CLKCTRL_MCLKSTATUS & CLKCTRL_SOSC_bm); // Wait for system
oscillator changing to finish

// Init port
PORTA_DIR |= PIN2_bm; // A2: Output
PORTA_OUT |= PIN2_bm; // A2: Output HIGH

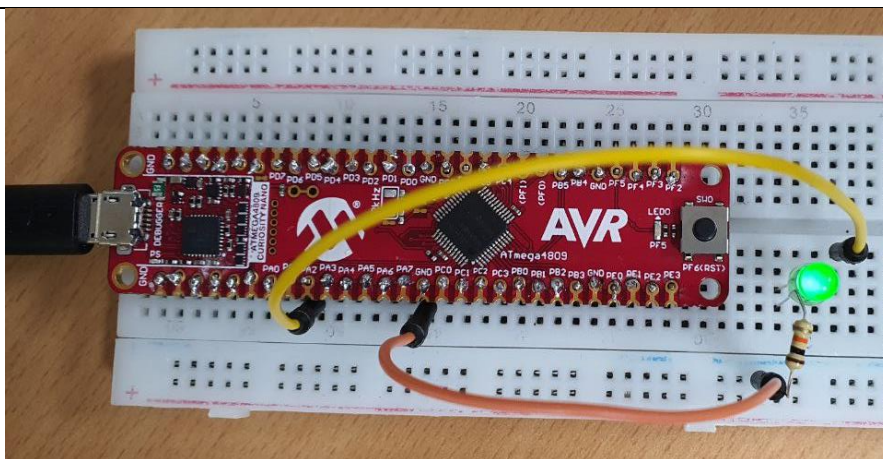
// Init TCB0
// Load CCMP register with the period and duty cycle of the PWM
TCB0_CCMPH = 0x80; // Duty
TCB0_CCMLP = 0xff; // TOP

TCB0_CTRLA |= TCB_ENABLE_bm | TCB_CLKSEL_CLKDIV2_gc; // Enable TCB0 and
Divide CLK_PER by 2

TCB0_CTRLB |= TCB_CCMPEN_bm | TCB_CNTMODE_PWM8_gc; // Enable Pin Output
and configure TCB in 8-bit PWM mode

while (1) {
    for (unsigned char i = 0; i < 255; i++) {
        TCB0_CCMPH = i;
        _delay_ms(10);
    }
}

return 0;
}
```



이 코드는 A2 핀의 LED 를 PWM 신호를 이용해 조절하는 코드이다.

CPU_CCP = CCP_IOREG_gc;를 통해 Protected 레지스터를 수정할 수 있게 만들었다.

CLKCTRL_MCLKCTRLB = CLKCTRL_PDIV_64X_gc | CLKCTRL_PEN_bm; CLKCTRL_MCLKCTRLA =

CLKCTRL_CLKSEL_OSCULP32K_gc;는 클럭 주파수를 낮추고, 클럭 생성에 OSCULP32K(Ultra Low Power Oscillator)를 사용하게 한다.

PORTA_DIR |= PIN2_bm; PORTA_OUT |= PIN2_bm; 를 통해서 A2 핀을 출력으로 설정한다.
TCB0_CCMPH = 0x80; 를 통해 PWM duty cycle 을 설정하고, TCB0_CCMPH = 0xff; 에 top 값을 설정한다. TCB0_CTRLA |= TCB_ENABLE_bm | TCB_CLKSEL_CLKDIV2_gc;을 통해 PWM 신호를 활성화하고, TCB0_CTRLB |= TCB_CCMPEN_bm | TCB_CNTMODE_PWM8_gc;을 통해 8 비트에 PWM 신호를 사용하도록 설정했다. 위 코드를 작동시키면 LED 의 밝기가 연속적으로 변화하게 된다.

TCB – Interrupt mode

```
#define F_CPU 3333333

#include <avr/interrupt.h>
#include <avr/io.h>
#include <util/delay.h>

int duty = 0x80;

int main(void) {
    // Init clock
    CPU_CCP = CCP_IOREG_gc;
    CLKCTRL_MCLKCTRLB = CLKCTRL_PDIV_64X_gc | CLKCTRL_PEN_bm;
    CLKCTRL_MCLKCTRLA = CLKCTRL_CLKSEL_OSCULP32K_gc;
    while (CLKCTRL_MCLKSTATUS & CLKCTRL_SOSC_bm);

    // Init port
    PORTMUX_TCBROUTEA |= PORTMUX_TCB0_ALT1_gc; // Set output pin to PF4
    PORTF_DIR |= PIN4_bm; // F4: Output
    PORTF_OUT |= PIN4_bm; // F4: Output HIGH

    // Init TCB0
    TCB0_CCMPH = duty; // Duty
    TCB0_CCMPH = 0xff; // TOP

    TCB0_CTRLA |= TCB_ENABLE_bm | TCB_CLKSEL_CLKDIV2_gc;
    TCB0_CTRLB |= TCB_CCMPEN_bm | TCB_CNTMODE_PWM8_gc;

    TCB0_INTCTRL |= TCB_CAPT_bm; // Enable Capture or Timeout interrupt

    sei(); // Enable Global Interrupts

    while (1) {
        _delay_ms(1000);
    }
}
```

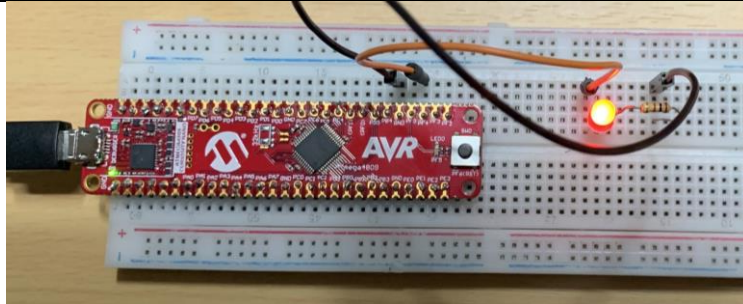


```

    }
    return 0;
}

ISR(TCB0_INT_vect) {
    TCB0_CCMPH = duty++;
    if (duty == 0x1F) { // Reset duty to 0x80
        duty = 0x80;
    }
}
}

```



실습 3 은 main 함수의 while 에서 duty 를 변화시켰지만, 이 코드는 ISR 을 이용해 duty 를 변화시킨다. 타이머 카운터 인터럽트를 사용하기 위해서는 TCB0_INTCTRL |= TCB_CAPT_bm 으로 Interrupt Control 을 설정해야 한다. sei()는 ISR 을 쓰기 위해 전역 인터럽트를 활성화하는 함수이다. 코드를 실행하면 LED 가 깜빡이는 주기가 점점 줄어들게 되고, 시간이 지나면 LED 가 계속 켜져 있는 것처럼 보인다.

Conclusion

ADC – Single conversion

USART 을 사용하기 위해서 코드에서 RX 와 TX 설정이 필요하다. RX 는 수신을 의미하며, USART_RXEN_bm 코드를 사용해 USART3 의 수신을 활성화시킨다. TX 는 송신을 의미하며, USART_TXEN_bm 코드를 사용해 USART3 의 송신을 활성화시킨다. 이 때 데이터 송수신에 전달되는 비트의 크기는 ADC0_CTRLA = ADC_RESSEL_8BIT_gc; 코드를 통해 8bit 데이터를 사용한다는 것을 알 수 있다.

해당 문제에서는 FILE USART_stream =FDEV_SETUP_STREAM(USART3_transmit2, NULL, _FDEV_SETUP_WRITE); 코드가 사용되었는데, FDEV_SETUP_STREAM () 함수를 사용해 사용자가 정의한 buffer 를 stdio(standard input output) stream 으로 바꿔주는 역할을 한다. 이를 통해 USART_stream 에서 printf 와 같은 I/O 기능을 사용할 수 있게 만들어졌다.

ADC – 10bit conversion

해당 문제도 Single conversion 과 마찬가지로 `adc_init` 함수와 `adc_read` 함수를 사용해 함수를 정의했다. main 함수에서는 `adc_read` 를 사용해 읽어온 아날로그 값을 10bit 로 표현된 전압 레벨로 변환하고, 공식을 사용해 측정 전압을 계산하고 출력시킨다. 이를 통해 아날로그 입력 값을 측정하고, 해당 값을 전압 값으로 변환하여 확인할 수 있다.

TCB – Basic TCB

`TCB0_CCMPH = 0x80;` 코드를 통해 먼저 PWM 을 정의하고, 이후 아래의 코드 `while (1) {for (unsigned char i = 0; i < 255; i++) {TCB0_CCMPH = i; _delay_ms(10);}}` 코드를 while 구문 안에 넣어서 duty cycle 의 값으로 PWM 신호를 생성하고 있다. 이 while 구문 안에서는 LED 의 밝기를 지속적으로 조절하고 있다. duty cycle 의 값이 업데이트가 되면서 LED 의 밝기가 조절된다는 것을 알 수 있다.

TCB – Interrupt mode

TCB0 는 A2 와 F4 로만 값을 출력할 수 있다. 기본 출력은 A2 핀이다. 실습 4 에서는 A2 핀 이외의 핀에서 PWM 을 출력하기 위해 `PORTMUX_TCBROUTEA |= PORTMUX_TCB0_ALT1_gc;`를 사용하여 F4 핀으로 출력을 내보내게 했다. 또한, 실습 3 과 다르게 인터럽트를 사용해 `TCB0_INTCTRL |= TCB_CAPT_bm;`으로 top 값에 도달하면 인터럽트를 발생시키도록 설정했다. 마지막으로, `ISR(TCB0_INT_vect)` 함수를 사용해 duty cycle 값을 감소시켜 LED 의 깜빡임의 간격을 줄이는 기능을 구현했다.