AVR - HW3

임베디드스쿨1기
Lv1과정
2020. 09. 21
손표훈

1. Interrupt

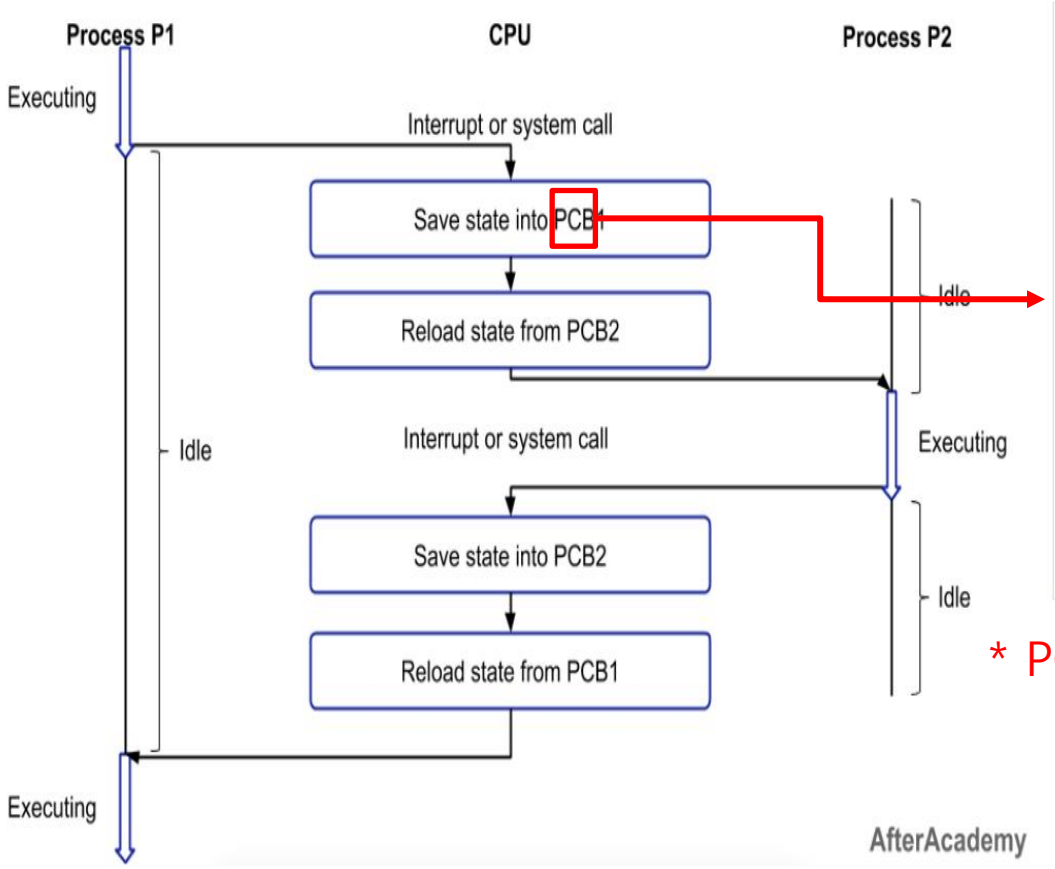
```c
/*
 * Ext_Interrupt.c
 *
 * Created: 2020-09-19 오전 9:44:39
 * Author : SON
 */
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#define sbi(PORTX, BitX) (PORTX |= (1<<BitX))

SIGNAL(INT1_vect){
    PORTB = 0x20; // PORTB(PIN 3) : ON
    _delay_ms(200);
}
int main(void)
{
    sbi(SREG, 7);

    sbi(EIMSK, INT1);

    EICRA = 0x08;

    DDRB = 0x20;

    DDRD=0x00;

    PORTD = 0xff;

    /* Replace with your application code */
    while (1)
    {
        PORTB = 0x00; //PORTB(PIN 3) : OFF
    }
}
```

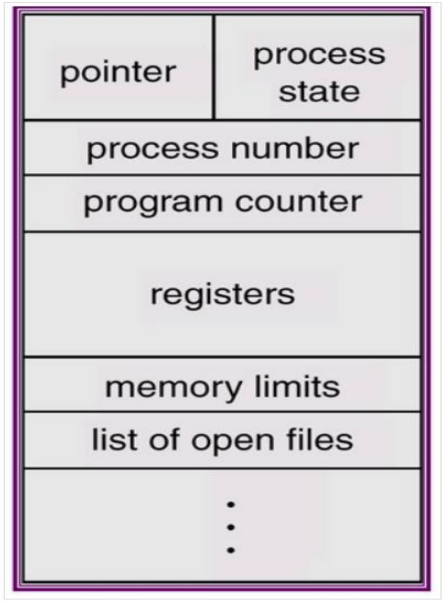
- 매크로함수 : PORTX에 BitX 만큼 1bit 우 쉬프트 후 OR Bit연산을 위해 정의함

```c
/** \def SIGNAL(vector)
    \ingroup avr_interrupts

    Introduces an interrupt handler function that runs with global interrupts
    initially disabled.

    This is the same as the ISR macro without optional attributes.
    \deprecated Do not use SIGNAL() in new code. Use ISR() instead.
*/
#  define SIGNAL(vector)
#else  /* real code */

#ifdef __cplusplus
#  define SIGNAL(vector)                    \
    extern "C" void vector(void) __attribute__ ((signal, __INTR_ATTRS));    \
    void vector (void)
#else
#  define SIGNAL(vector)                    \
    void vector (void) __attribute__ ((signal, __INTR_ATTRS));      \
    void vector (void)
#endif
```

1. Context Switching

1) Context Switching이란? 시스템 호출(함수호출)과 인터럽트 처리, 사용자와 커널모드 전환 시 현재 실행중인 프로세스를 중단하지 않고 발생된 프로세스를 처리 후 원래 프로세스를 처리하는 기술





* PCB(Process Control Block): 프로세스 관리 구조

1. Context Switching

2) Context Switching 처리과정(Interrupt 발생)

2) 인터럽트 발생

```c
/*
 * Ext_Interrupt.c
 *
 * Created: 2020-09-19 오전 9:44:39
 * Author : SON
 */
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#define sbi(PORTX, BitX) (PORTX |= (1<<BitX))

SIGNAL(INT1_vect){
    PORTB = 0x20; // PORTB(PIN 3) : ON
    _delay_ms(200);
}
int main(void)
{
    sbi(SREG, 7);

    sbi(EIMSK, INT1);

    EICRA = 0x08;

    DDRB = 0x20;

    DDRD=0x00;

    PORTD = 0xff;

    /* Replace with your application code */
    while (1)
    {
        PORTB = 0x00; //PORTB(PIN 3) : OFF
    }
}
```

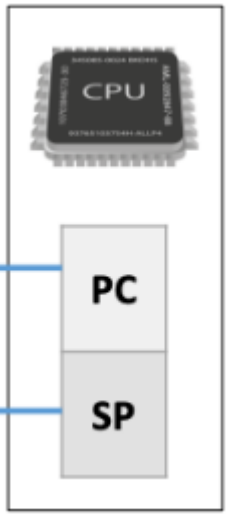
```c
/** \def SIGNAL(vector)
    \ingroup avr_interrupts

    Introduces an interrupt handler function that runs with global interrupts
    initially disabled.

    This is the same as the ISR macro without optional attributes.
    \deprecated Do not use SIGNAL() in new code. Use ISR() instead.
*/
#  define SIGNAL(vector)
#else  /* real code */

#ifdef __cplusplus
#  define SIGNAL(vector)                                    \
    extern "C" void vector(void) __attribute__ ((signal, __INTR_ATTRS));    \
    void vector (void)
#else
#  define SIGNAL(vector)                                    \
    void vector (void) __attribute__ ((signal, __INTR_ATTRS));      \
    void vector (void)
#endif
```

5) Vecter Table 참조 후 Interrupt ISR실행

6) ISR의 명령어 실행

7) PC값을 로드하여 while문 실행

4) PC : 다음 실행될 명령어 주소를 가리킴

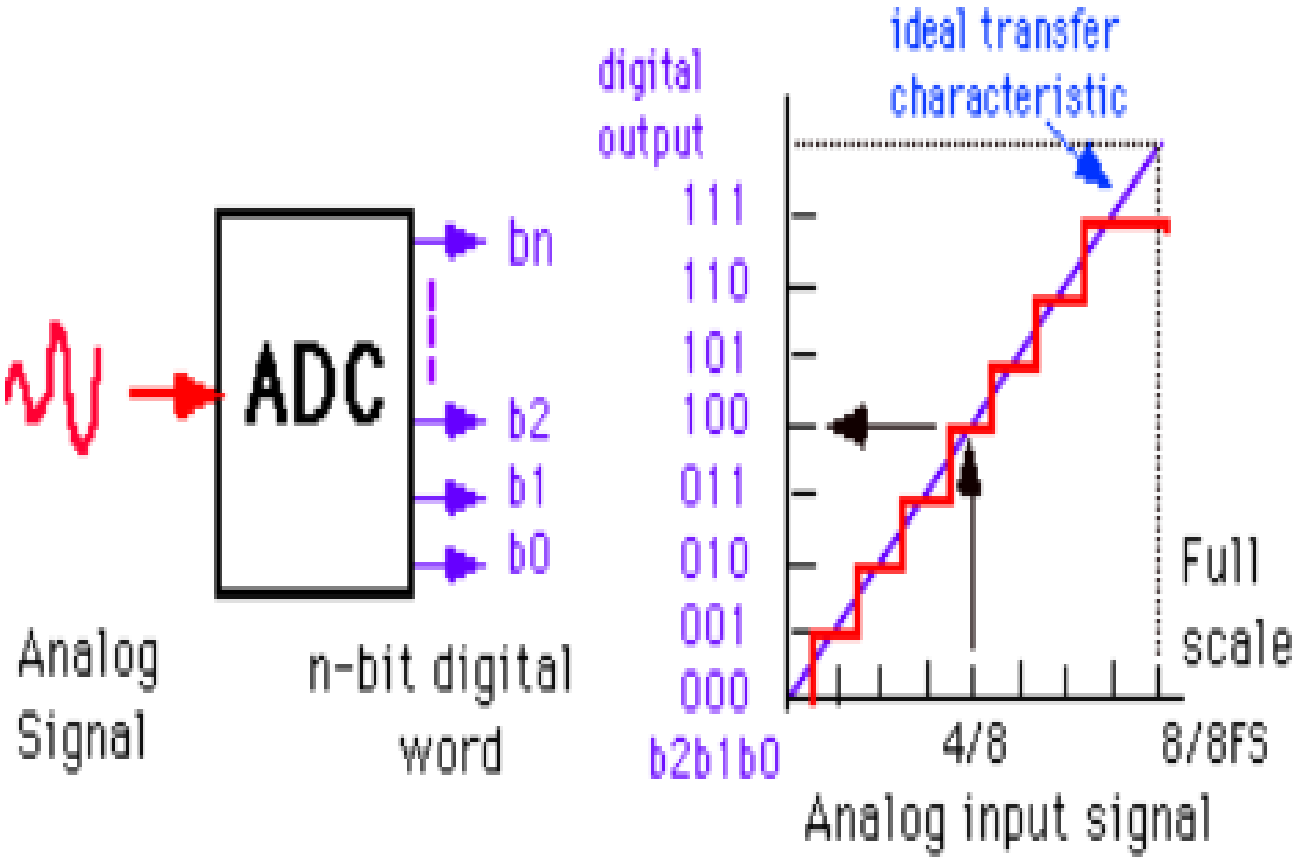


1) while문 실행

3) 현 상태 stack에 저장

2. ADC(Analog to Digital Converter)

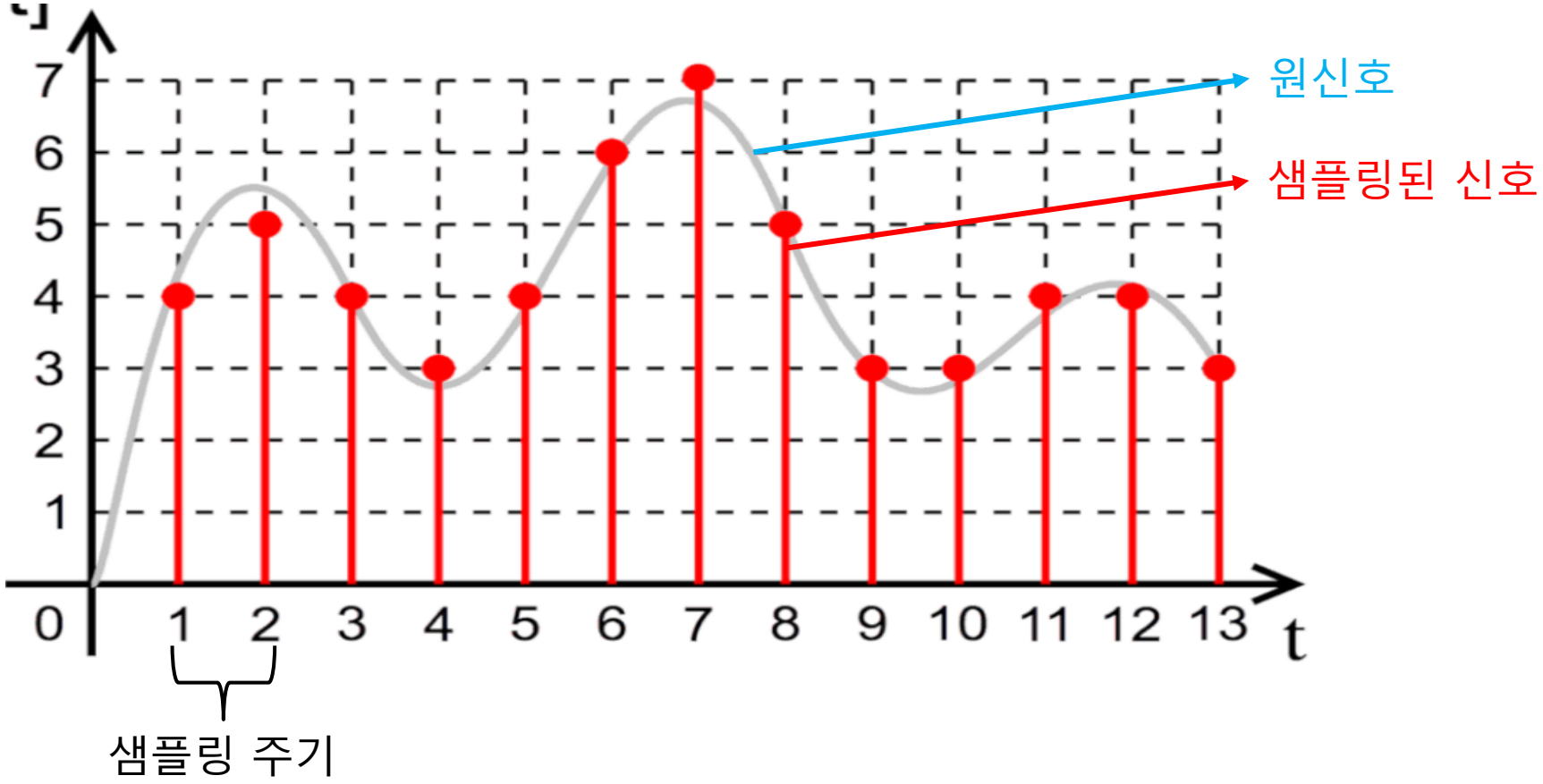
1) ADC란? 아날로그 신호를 디지털 신호로 변환해주는 장치
   - 아날로그 신호란? 연속시간에서 표현되는 신호들
   - 디지털 신호란? 0과 1로 표현된 신호들



2. ADC(Analog to Digital Converter)

2) ADC 변환 과정

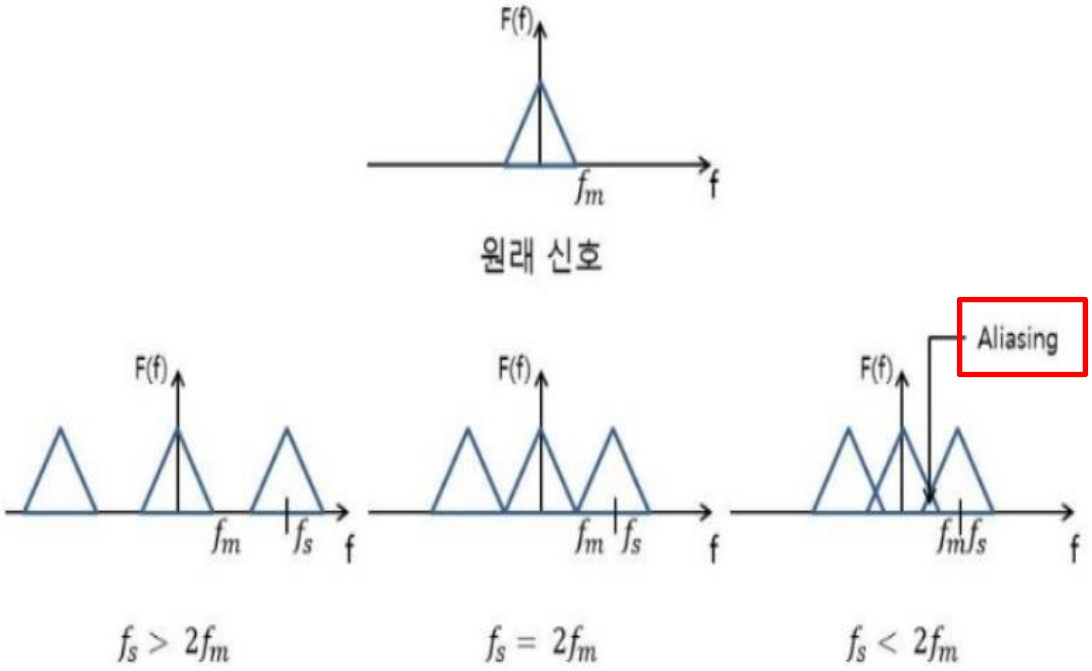
- 샘플링 : 원 신호를 얼마나 잘게 쪼갤 것인가?

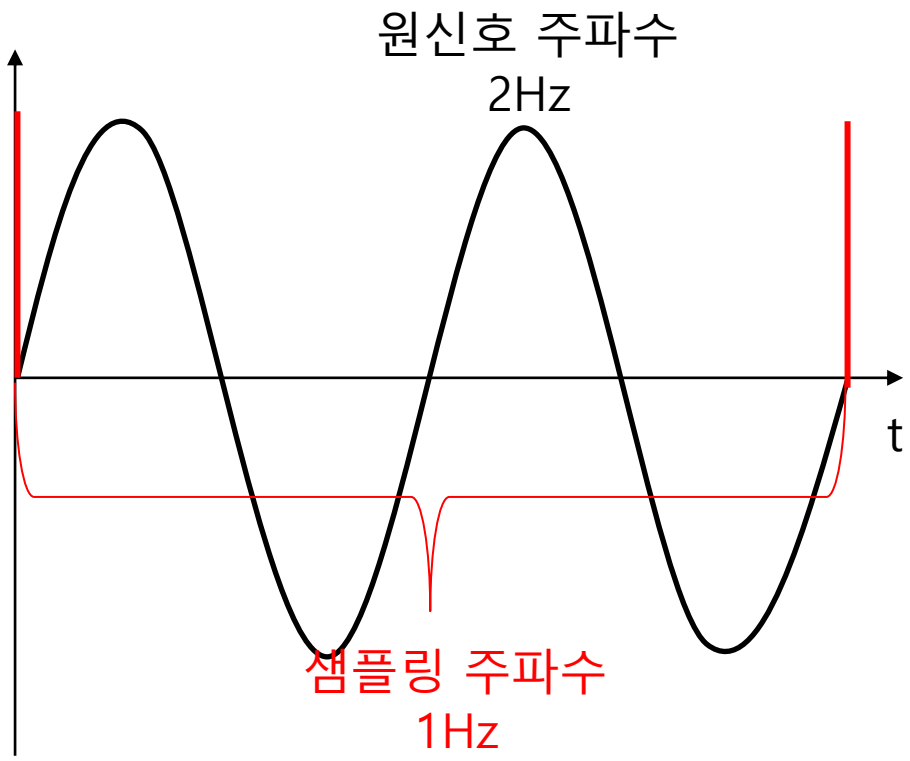


- Digital변환 시 샘플링 주기가 짧을수록 원신호에 가깝게 표현 할 수 있게 된다.
- 샘플링 주기는 무조건 짧기만 하면 된다? 아니다..(나이퀴스트 샘플링)

2. ADC(Analog to Digital Converter)

※ 나이퀴스트 샘플링 이론 : 샘플링 주파수는 원 신호 주파수의 2배 이상이 되어야 한다.



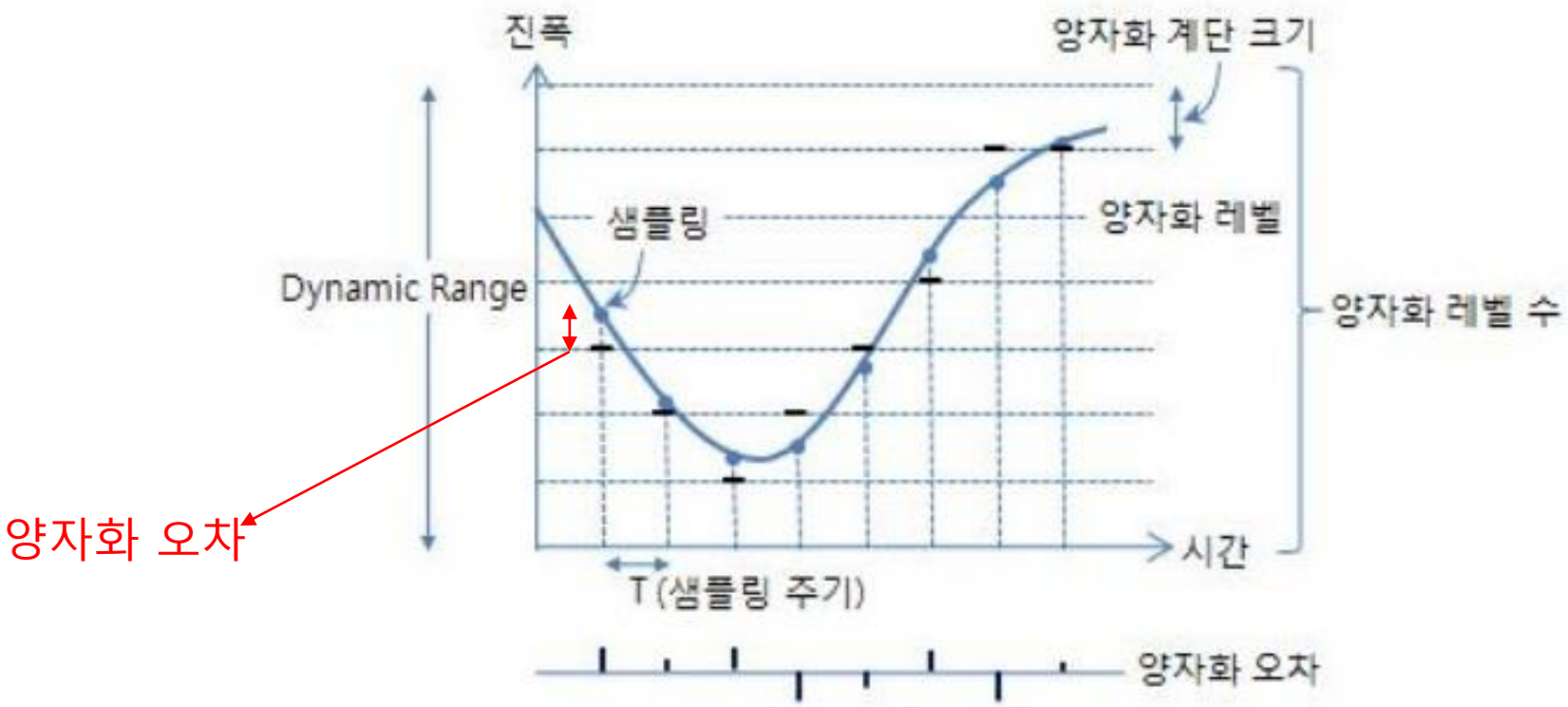


- 샘플링 주파수가 원 신호의 주파수 보다 낮으면 우측 그림과 같이 계속 변환된 신호는 계속 0이 된다.

- 주파수 측에서 볼 때 스펙트럼이 겹치는 현상(Aliasing 왜곡)이 발생한다.

2. ADC(Analog to Digital Converter)

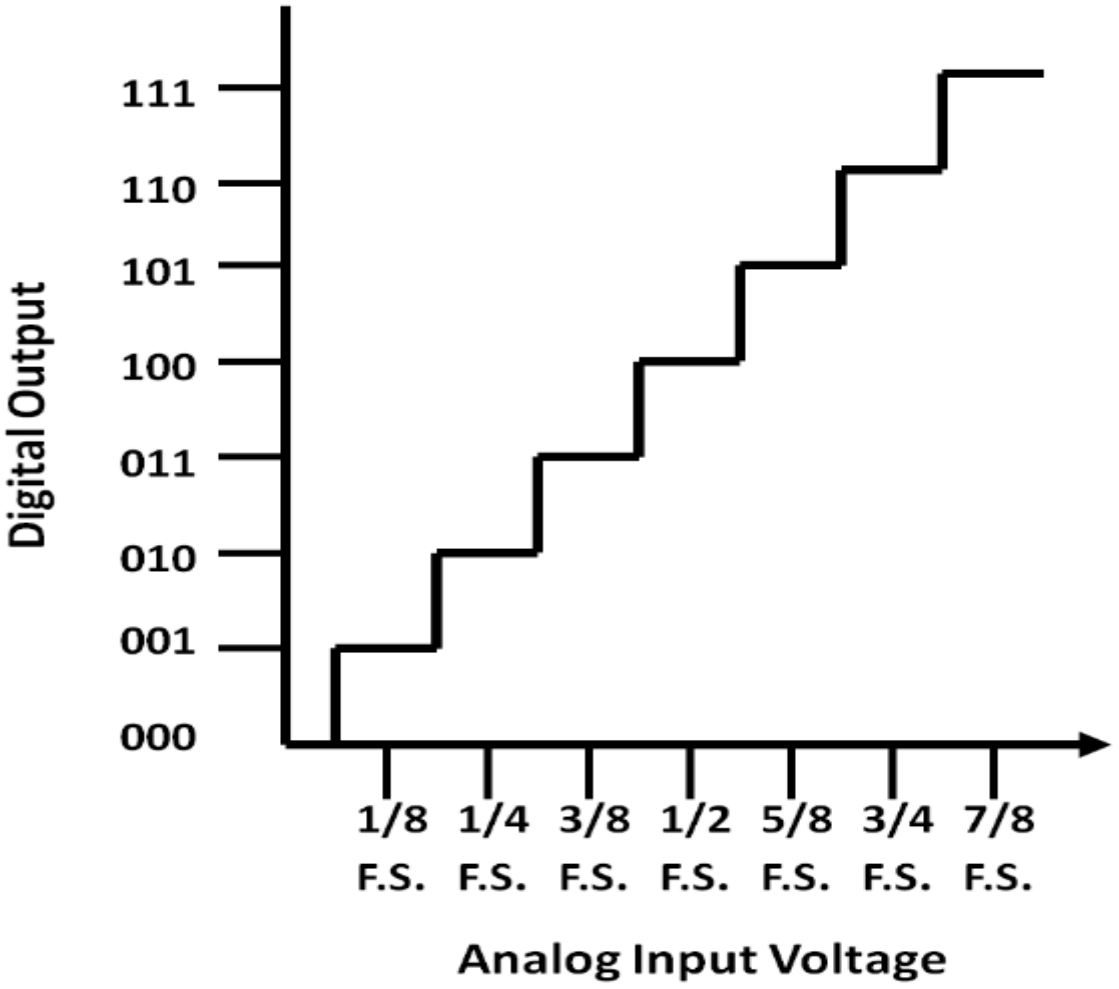
- 양자화 : 무한대로 이루어진 셀 수 없는 아날로그 정보를 셀 수 있을 만큼의 간격으로 만들어 유의미한 정보를 사용할 수 있게 해주는 과정



- 양자화 오차 : 빨간색 표시 부분처럼 양자화시 오차가 발생한다.
- 분해능이 높을수록(샘플의 수가 많을수록) 양자화 오차는 줄어든다.
- 양자화 레벨은 bit수에 대응된다.
  12bit ADC의 경우 4096개의 양자화된 레벨을 가진다.

2. ADC(Analog to Digital Converter)

- 부호화 : 양자화된 신호에 디지털 신호의 레벨(0 또는 1)로 변환하는 과정



2. ADC(Analog to Digital Converter)

3) Atmega328의 ADC 특징

Analog-to-Digital Converter

Features

- 10-bit Resolution
- 0.5 LSB Integral Non-linearity
- ± 2 LSB Absolute Accuracy
- 13 - 260µs Conversion Time
- Up to 76.9kSPS (Up to 15kSPS at Maximum Resolution)
- 6 Multiplexed Single Ended Input Channels
- 2 Additional Multiplexed Single Ended Input Channels (TQFP and VQFN Package only)
- Temperature Sensor Input Channel
- Optional Left Adjustment for ADC Result Readout
- 0 - V_{CC} ADC Input Voltage Range
- Selectable 1.1V ADC Reference Voltage
- Free Running or Single Conversion Mode
- Interrupt on ADC Conversion Complete
- Sleep Mode Noise Canceler

분해능 : 0~1023의 디지털 값

오차 : 2LSB로 Vref에 따라 다름
Ex) 4Vref -> 3.9062mV의 오차를 가짐

샘플속도 : ADC모듈이 입력전압을 변환하는 속도
15ksps = 초당 15000개의 샘플 신호를 만듬

6개의 채널을 가짐
각 채널은 single ended입력
*single ended는 GND 기준 신호
반대는 differential

내부 1.1Vref 전압을 설정 할 수 있음

2. ADC(Analog to Digital Converter)

3) ADC CLK, Sample Rate, Input Signal Band Width

Figure 23-5. ADC Timing Diagram, Single Conversion

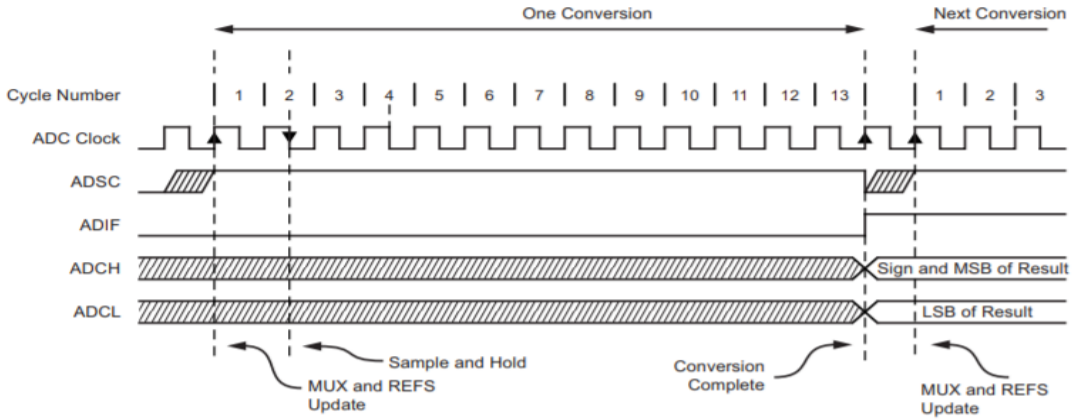


Figure 23-3. ADC Prescaler

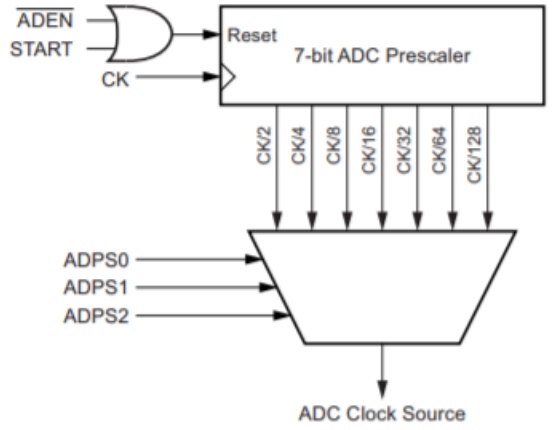


Table 23-1. ADC Conversion Time

Condition	Sample and Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	13.5	25
Normal conversions, single ended	1.5	13
Auto triggered conversions	2	13.5

- ADC CLK = 200KHz
- 10Bit ADC 1 Sampling Frequency(Fs) = 200KHz/13(Normal Conversion) ≒ 15.384KHz
- Sample Rate = 15384SPS(초당 15384개의 샘플을 획득)
- Input Band Width는 나이퀴스트 샘플링에 의해 7.69KHz(Fs/2)

2. ADC(Analog to Digital Converter)

질문

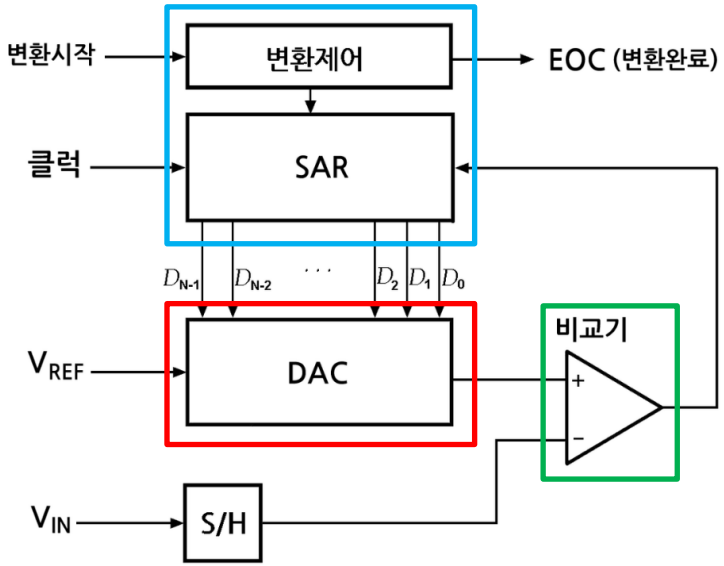
By default, the successive approximation circuitry requires an input clock frequency between 50kHz and 200kHz to get maximum resolution. If a lower resolution than 10 bits is needed, the input clock frequency to the ADC can be higher tha 200kHz to get a higher sample rate.

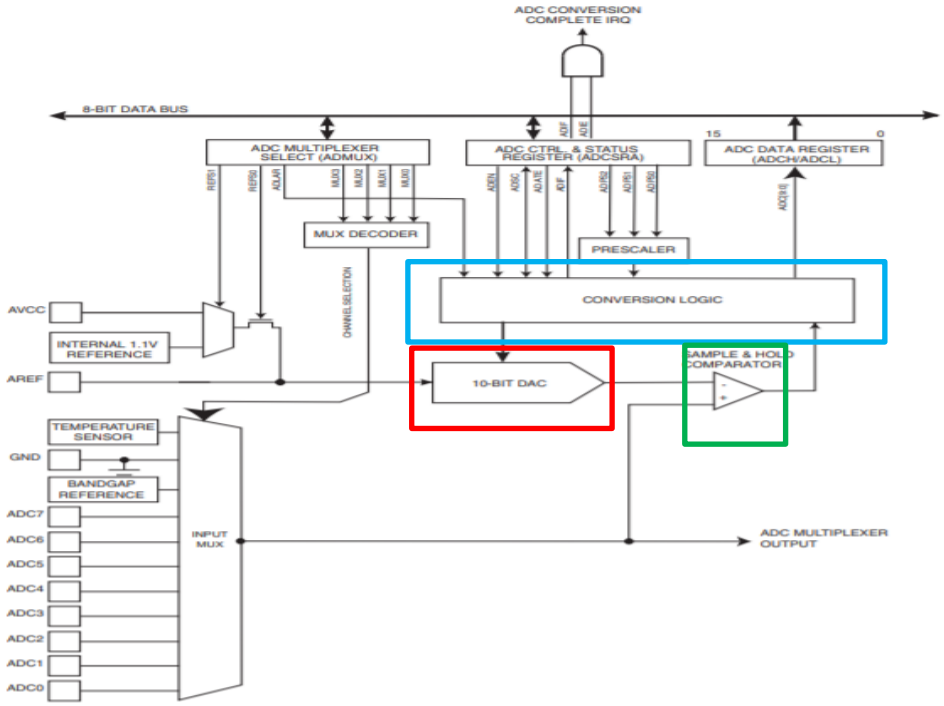
- 데이터시트에 따르면 최대 분해능 10bit를 얻기 위해서 ADC CLK은 50~200KHz가 필요하다고 한다.

- 빨간색200KHz이상으로 ADC CLK을 인가하면 10bit보다 낮은 분해능을 얻는다는데 왜 낮을수록 낮은 분해능인지?

2. ADC(Analog to Digital Converter)

- SAR 타입의 ADC 회로를 사용한다...
- SAR 타입?
  https://ko.wikipedia.org/wiki/%EC%B6%95%EC%B0%A8_%EB%B9%84%EA%B5%90%ED%98%95_%EC%95%84%EB%82%A0%EB%A1%9C%EA%B7%B8-%EB%94%94%EC%A7%80%ED%84%B8_%EB%B3%80%ED%99%98%ED%9A%8C%EB%A1%9C





- 그 외 ADC 회로 : dual-slope ADC, Flash ADC, Delta-Sigma ADC, Tracking ADC

2. ADC(Analog to Digital Converter)

- 데이터 계산방법

ADC Conversion Result

After the conversion is complete (ADIF is high), the conversion result can be found in the ADC Result Registers (ADCL, ADCH).

For single ended conversion, the result is

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

- Vref = 2.5V, Vin = 2V라면,
  ADC = 2V * 1024/2.5V ≒ 819 = 0x333

- 만약 Vin이 Vref보다 높으면 ADC 결과는 1023이 된다.

- 설계의도에 따라 Vref를 내부 1.1V ~ VCC로 설정 한다.

2. ADC(Analog to Digital Converter)

4) ATmega328의 ADC 관련 레지스터
- ADC MUX Register

ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – REFS1:0: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 23-3. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 23-3. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, internal V_{REF} turned off
0	1	AV_{CC} with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 1.1V voltage reference with external capacitor at AREF pin

Table 23-4. Input Channel Selections

MUX3..0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	ADC8(1)
1001	(reserved)
1010	(reserved)
1011	(reserved)
1100	(reserved)
1101	(reserved)
1110	1.1V (V_{BG})
1111	0V (GND)

Note: 1. For temperature sensor.

- Bit 0~3 설정으로 외부 신호 ADC 0 ~ 7
- ADC 8 : On chip 온도센서 선택
- Bit 6~7 설정으로 내부 기준전압 설정을 할 수 있다.

2. ADC(Analog to Digital Converter)

4) ATmega328의 ADC 관련 레지스터
- ADC Ctrl and Status Register A

23.9.2 ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ADEN: ADC Enable**

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

- **Bit 6 – ADSC: ADC Start Conversion**

In single conversion mode, write this bit to one to start each conversion. In free running mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

- **Bit 5 – ADATE: ADC Auto Trigger Enable**

When this bit is written to one, auto triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC trigger select bits, ADTS in ADCSRB.

- **Bit 4 – ADIF: ADC Interrupt Flag**

This bit is set when an ADC conversion completes and the data registers are updated. The ADC conversion complete interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a read-modify-write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

- **Bit 3 – ADIE: ADC Interrupt Enable**

When this bit is written to one and the I-bit in SREG is set, the ADC conversion complete interrupt is activated.

- **Bits 2:0 – ADPS2:0: ADC Prescaler Select Bits**

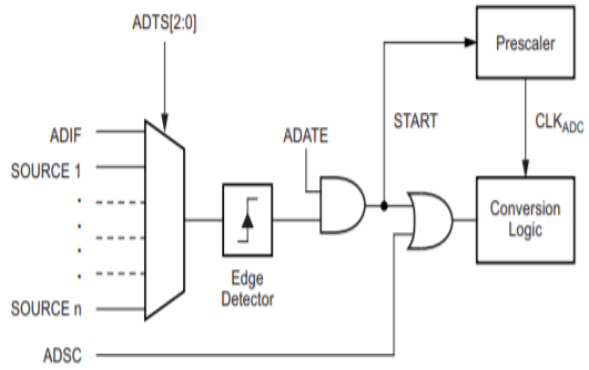
These bits determine the division factor between the system clock frequency and the input clock to the ADC.

Table 23-5. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

- Bit 7 : ADC 모듈 활성화
- Bit 6 : 변환 시작신호
- Bit 5 : 자동 변환 모드를 활성화
- Bit 4 : ADC Interrupt Flag
- Bit 3 : ADC Interrupt Enable
- Bit 0~2 : ADC CLK 분주비 설정

Figure 23-2. ADC Auto Trigger Logic



2. ADC(Analog to Digital Converter)

4) ATmega328의 ADC 관련 레지스터
- ADC Ctrl and Status Register B

23.9.4 ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
(0x7B)	–	ACME	–	–	–	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R	R/W	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7, 5:3 – Res: Reserved Bits**

These bits are reserved for future use. To ensure compatibility with future devices, these bits must be written to zero when ADCSRB is written.

- **Bit 2:0 – ADTS2:0: ADC Auto Trigger Source**

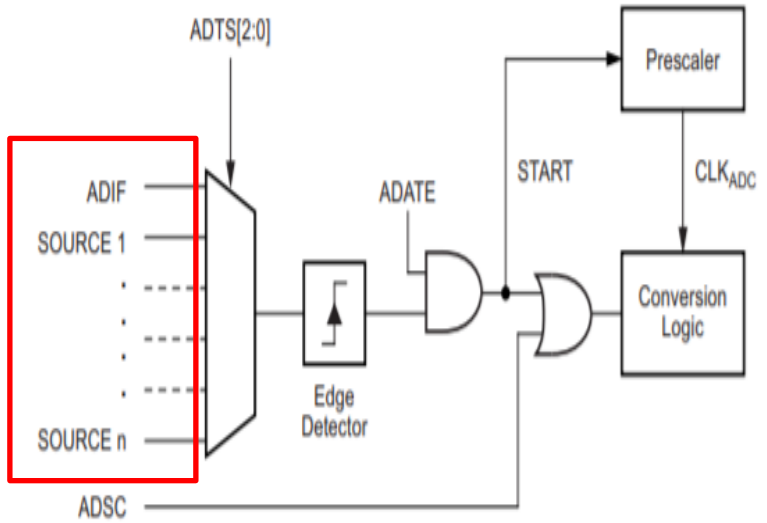
If ADATE in ADCSRA is written to one, the value of these bits selects which source will trigger an ADC conversion. If ADATE is cleared, the ADTS2:0 settings will have no effect. A conversion will be triggered by the rising edge of the selected interrupt flag. Note that switching from a trigger source that is cleared to a trigger source that is set, will generate a positive edge on the trigger signal. If ADEN in ADCSRA is set, this will start a conversion. Switching to free running mode (ADTS[2:0]=0) will not cause a trigger event, even if the ADC interrupt flag is set.

Table 23-6. ADC Auto Trigger Source Selections

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free running mode
0	0	1	Analog comparator
0	1	0	External interrupt request 0
0	1	1	Timer/Counter0 compare match A
1	0	0	Timer/Counter0 overflow
1	0	1	Timer/Counter1 compare match B
1	1	0	Timer/Counter1 overflow
1	1	1	Timer/Counter1 capture event

- Bit 0~2 : Auto Trigger모드의 트리거 소스를 결정
  * Free Runing Mode 설정
  * Analog Comp 출력에 따른 트리거
  * Timer 소스

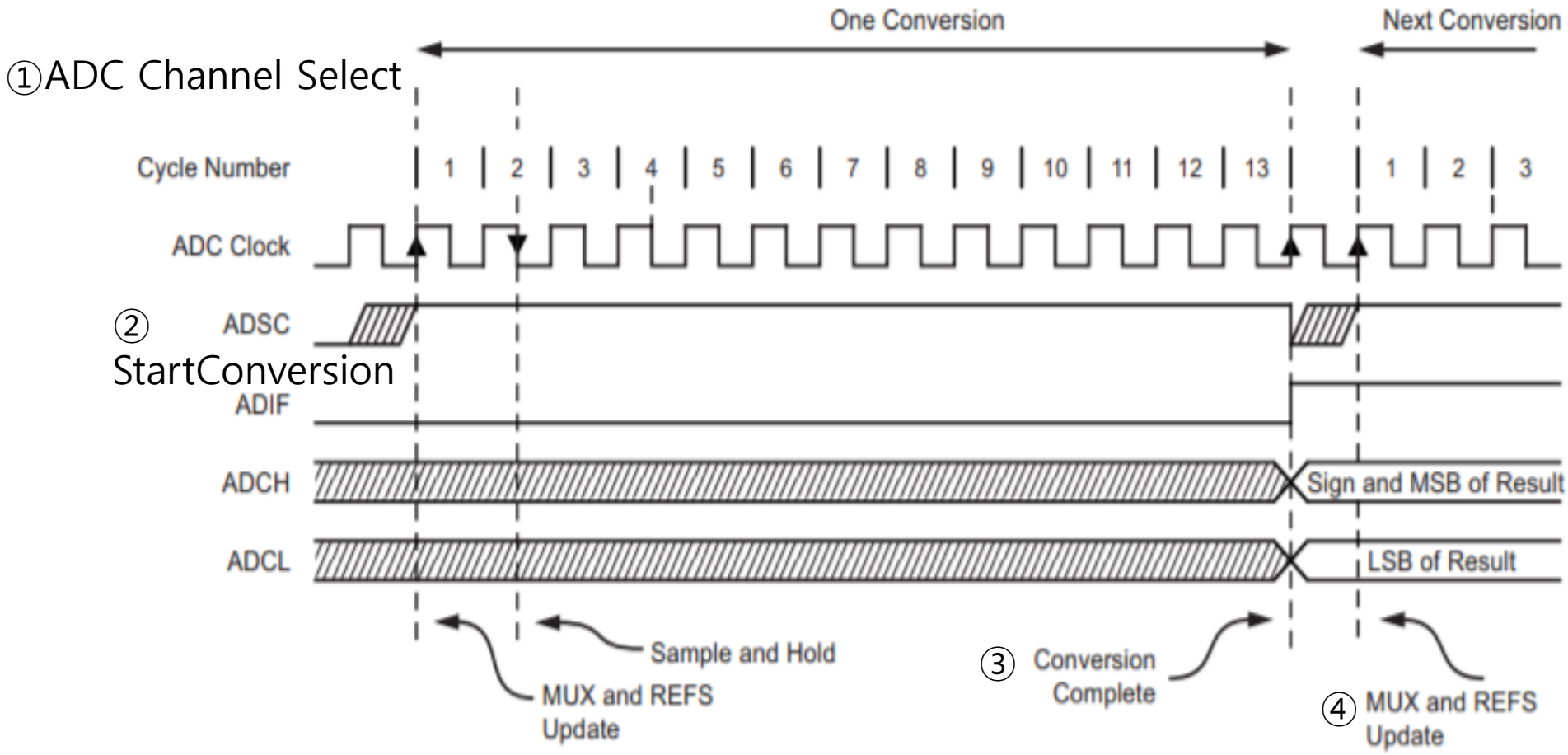
Figure 23-2. ADC Auto Trigger Logic



2. ADC(Analog to Digital Converter)

4) ATmega328의 ADC 동작
- Single Conversion Mode

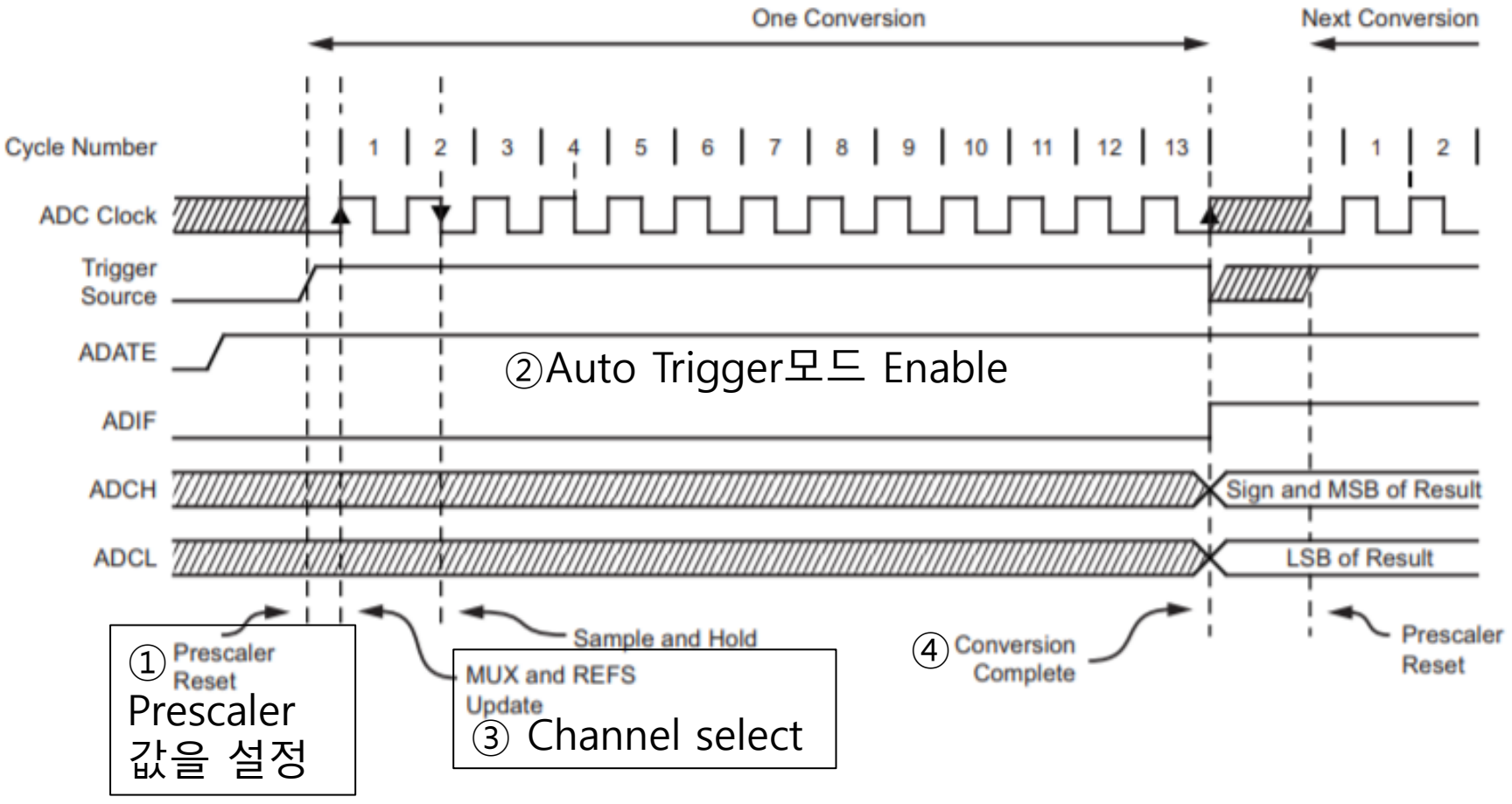
Figure 23-5. ADC Timing Diagram, Single Conversion



2. ADC(Analog to Digital Converter)

4) ATmega328의 ADC 동작
- Auto Trigger Mode

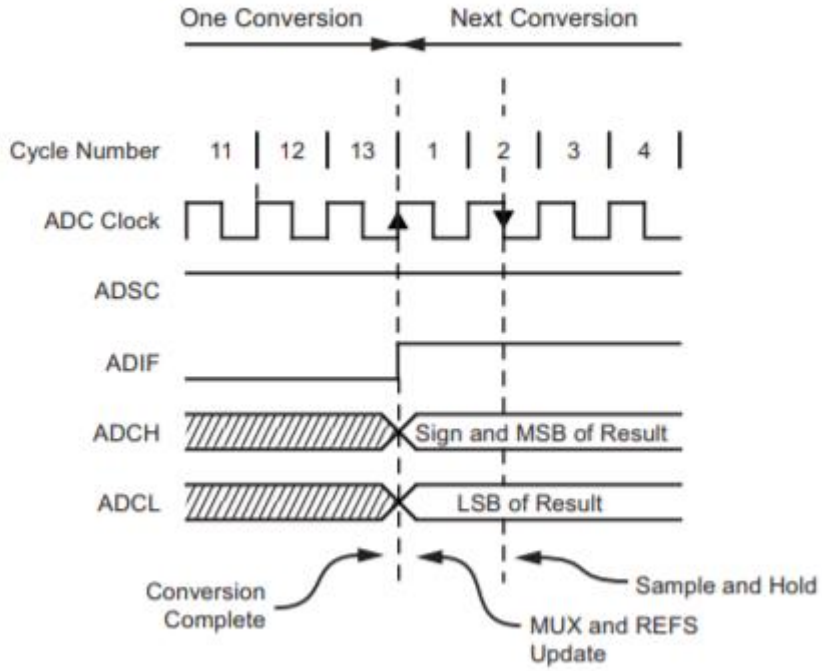
Figure 23-6. ADC Timing Diagram, Auto Triggered Conversion



2. ADC(Analog to Digital Converter)

4) ATmega328의 ADC 동작
- Free Running Mode

Figure 23-7. ADC Timing Diagram, Free Running Conversion



- 13 ADC clk Cycle마다 지속적으로 변환을 수행함
- Conversion Start 신호는 'H' 상태가 유지되어야 한다.

3. UART 통신

1) UART 통신이란? 범용 비동기 직렬통신으로 별도의 동기 신호가 없고 통신 프로토콜(?)의 Start Bit & Stop Bit를 시간적으로 알아내어 통신한다.
따라서 송/수신측의 통신속도가 일치해야만 통신이 가능!

2) 프로토콜이란? 비동기 통신을 위한 데이터 프레임으로 UART통신의 프로토콜은 아래와 같은 데이터 프레임을 갖는다.

비트 수	1	2	3	4	5	6	7	8	9	10	11
	시작 비트 (Start bit)	5-8 데이터 비트								패리티 비트 (parity bit)	종료 비트 (Stop bit(s))
	Start	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Parity	Stop

•시작 비트 : 통신의 시작을 의미하며 한 비트 시간 길이 만큼 유지한다. 지금 부터 정해진 약속에 따라 통신을 시작한다.

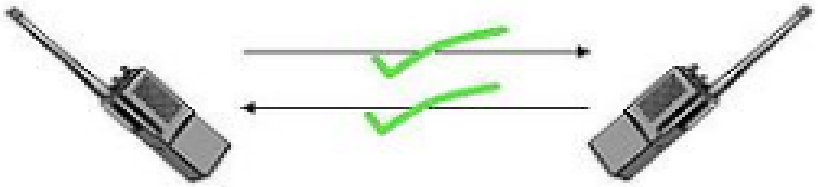
•데이터 비트 : 5~8비트의 데이터 전송을 한다. 몇 비트를 사용할 것인지는 해당 레지스터 설정에 따라 결정된다.

•패리티 비트 : 오류 검증을 하기 위한 패리티 값을 생성하여 송신하고 수신쪽에 오류 판단한다. 사용안함, 짝수, 홀수 패리티 등의 세가지 옵션으로 해당 레지스터 설정에 따라 선택할 수 있다. '사용안함'을 선택하면 이 비트가 제거된다.

•끝 비트 : 통신 종료를 알린다. 세가지의 정해진 비트 만큼 유지해야 한다. 1, 1.5, 2비트로 해당 레지스터 설정에 따라 결정된다.

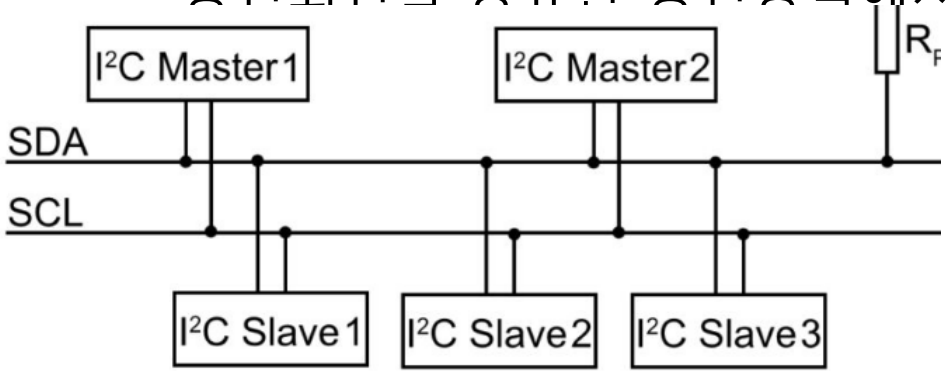
3. UART 통신

3) 전이중(Full-Duplex)통신 방식을 지원한다.
- 전이중(Full-Duplex)란? 독립된 송/수신 회선을 사용하는 통신 방식
송신 중 수신을 할 수 있고, 수신 중 송신을 할 수 있음



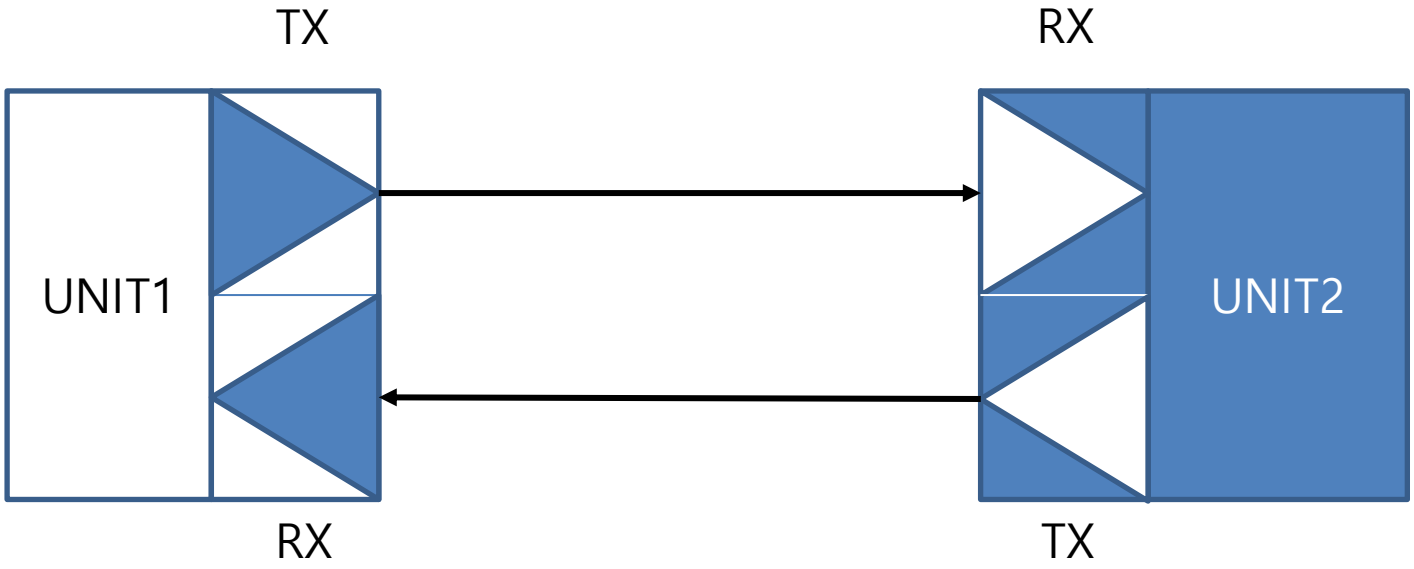
단순 전이중 방식을 그린 그림

* 반이중(Half-Duplex)통신으로 한쪽이 송신하면 다른쪽은 수신하는 통신 방식
통신회선을 공유한 통신방식에서 사용된다.



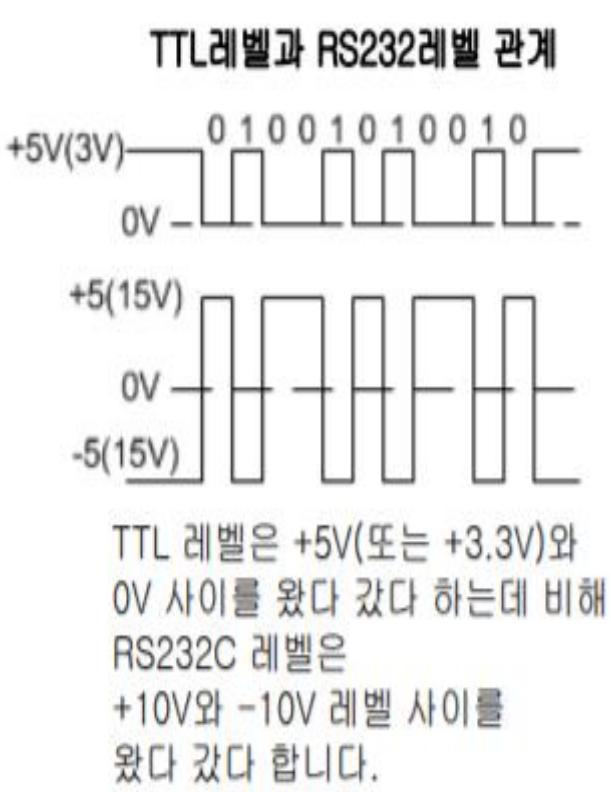
3. UART 통신

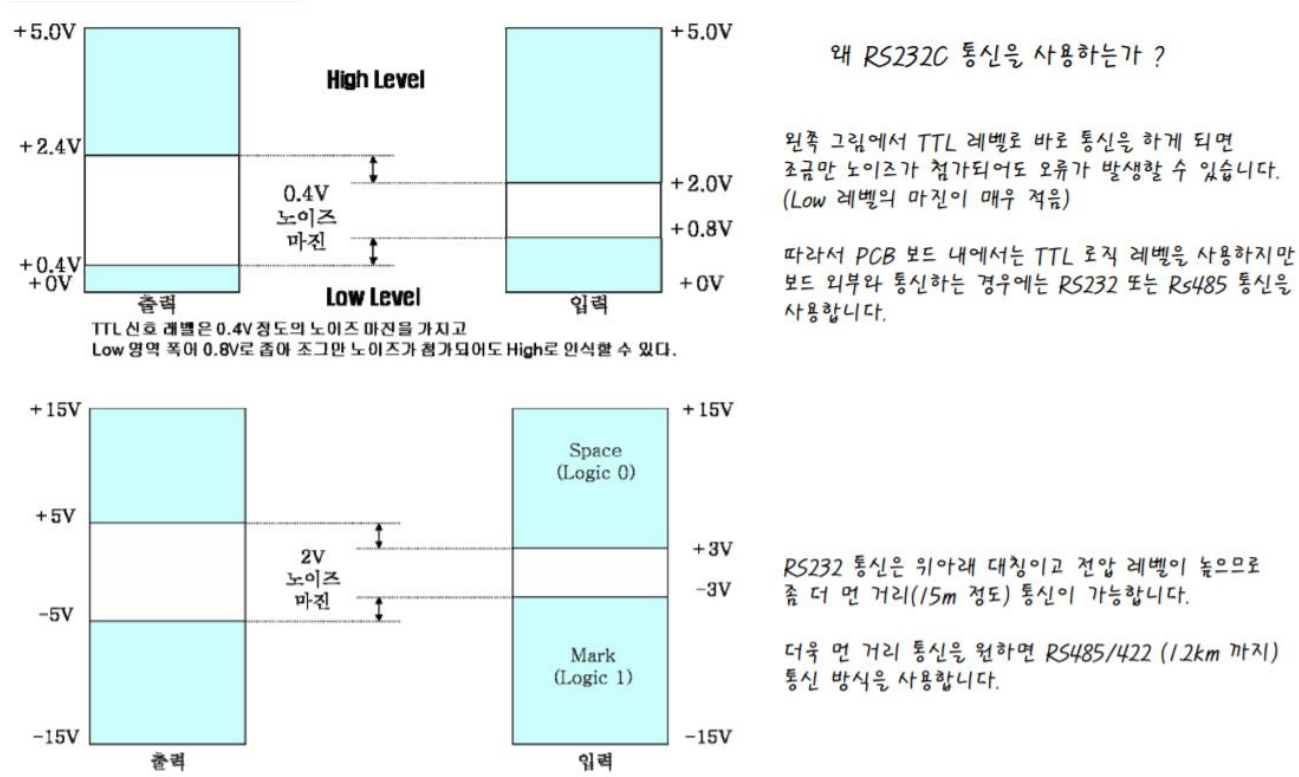
4) 1:1 통신 구조로 TX/RX 통신 회선이 있으며, 송수신간 통신선로 연결은 아래와 같이 한다.



3. UART 통신

5) RS-232, RS-422, RS-485와 같은 통신규격과 같이 사용될 수 있다.
UART통신은 보통 MCU내부의 주변장치로 TTL 신호레벨이 사용됨..





왜 RS232C 통신을 사용하는가 ?

왼쪽 그림에서 TTL 레벨로 바로 통신을 하게 되면 조금만 노이즈가 첨가되어도 오류가 발생할 수 있습니다. (Low 레벨의 마진이 매우 적음)

따라서 PCB 보드 내에서는 TTL 로직 레벨을 사용하지만 보드 외부와 통신하는 경우에는 RS232 또는 Rs485 통신을 사용합니다.

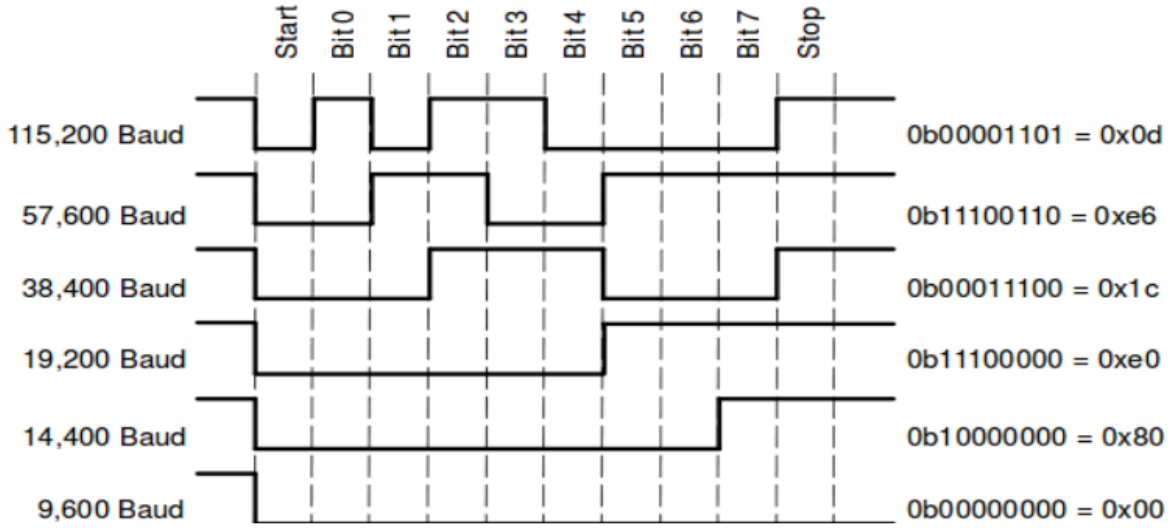
RS232 통신은 위아래 대칭이고 전압 레벨이 높으므로 좀 더 먼 거리(15m 정도) 통신이 가능합니다.

더욱 먼 거리 통신을 원하면 RS485/422 (1.2km 까지) 통신 방식을 사용합니다.

- UART통신만가지고 장거리 통신을 하게 되면 노이즈에 의해 통신이 제대로 안됨
- 따라서 RS-232, RS-422, RS-485와 같은 통신규격과 별도의 트랜시버 칩을 사용하여 통신함..

3. UART 통신

6) 통신속도(BaudRate)란? 초당 전송되는 데이터 수(펄수의 수)로 아래와 같이 표준 통신속도가 있음..



- Baud Rate와 전송되는 데이터 크기와의 관계(ex : 115200bps)
- 115200bps는 1/115200하여 8.86us 간격으로 1bit씩 송신을 말한다.
- 115200bps는 UART의 8bit 데이터 송신 시 start+stop을 합쳐 10bit를 송수신한다.
- 1byte를 보내는데 10bit가 필요, 115200/10 = 11520 byte per second = 92160 bit per second가 된다.

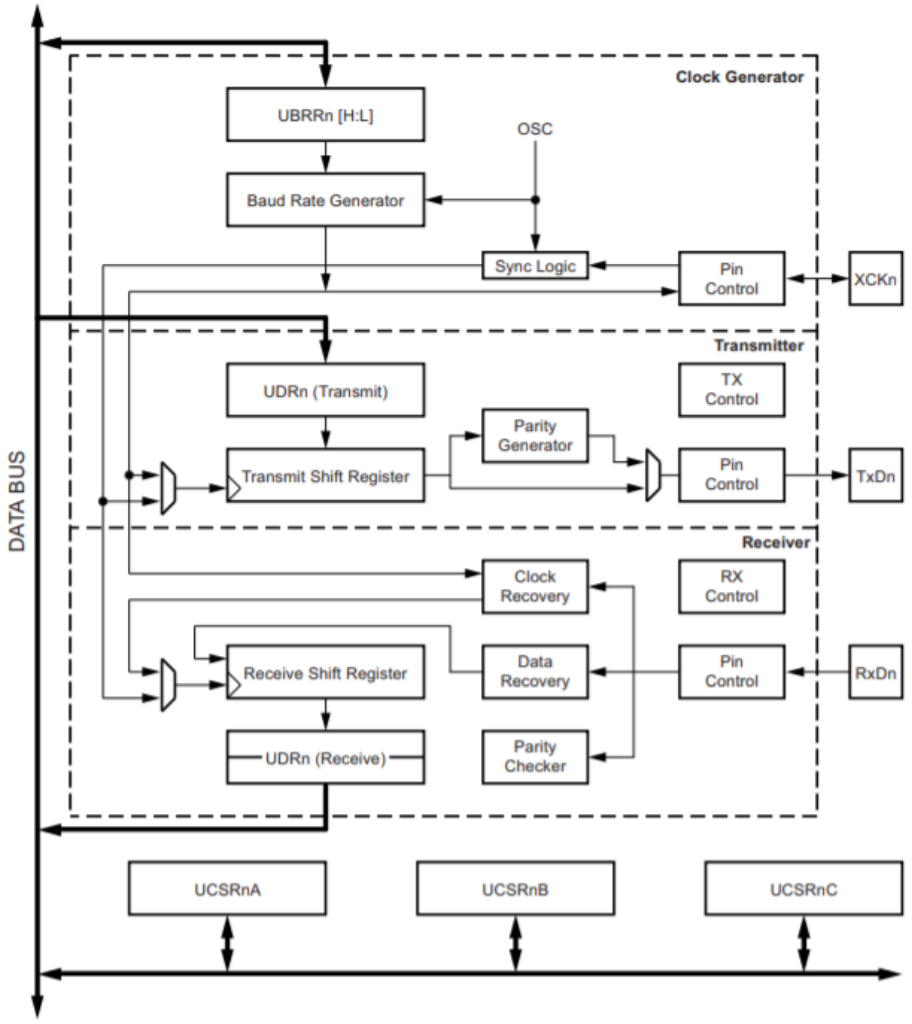
3. UART 통신

7) ATmega328의 USART0모듈의 특징 및 Block

Features

- Full duplex operation (independent serial receive and transmit registers)
- Asynchronous or synchronous operation
- Master or slave clocked synchronous operation
- High resolution baud rate generator
- Supports serial frames with 5, 6, 7, 8, or 9 data bits and 1 or 2 stop bits
- Odd or even parity generation and parity check supported by hardware
- Data overrun detection
- Framing error detection
- Noise filtering includes false start bit detection and digital low pass filter
- Three separate interrupts on TX complete, TX data register empty and RX complete
- Multi-processor communication mode
- Double speed asynchronous communication mode

Figure 19-1. USART Block Diagram(1)



3. UART 통신

7) ATmega328의 USART0 통신 속도 설정

Figure 19-2. Clock Generation Logic, Block Diagram

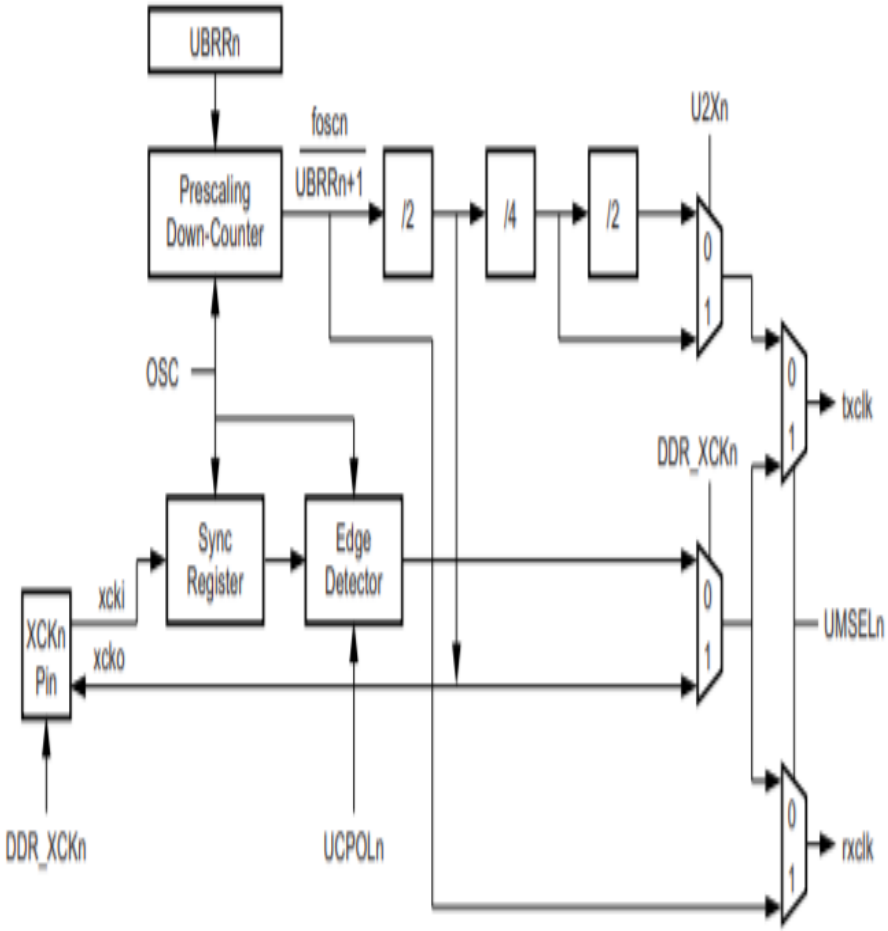


Table 19-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRRn Value
Asynchronous normal mode (U2Xn = 0)	$BAUD = \frac{f_{OSC}}{16(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous double speed mode (U2Xn = 1)	$BAUD = \frac{f_{OSC}}{8(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous master mode	$BAUD = \frac{f_{OSC}}{8(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{2BAUD} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps)

BAUD	Baud rate (in bits per second, bps)
f_{OSC}	System oscillator clock frequency
UBRRn	Contents of the UBRRnH and UBRRnL registers, (0-4095)

7) ATmega328의 USART0 Data Format and Parity Check

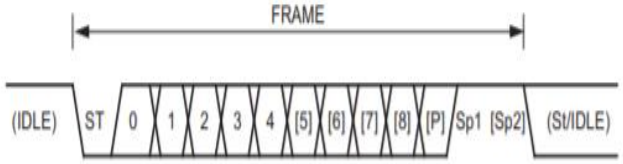
Frame Formats

A serial frame is defined to be one character of data bits with synchronization bits (start and stop bits), and optionally a parity bit for error checking. The USART accepts all 30 combinations of the following as valid frame formats:

- 1 start bit
- 5, 6, 7, 8, or 9 data bits
- no, even or odd parity bit
- 1 or 2 stop bits

A frame starts with the start bit followed by the least significant data bit. Then the next data bits, up to a total of nine, are succeeding, ending with the most significant bit. If enabled, the parity bit is inserted after the data bits, before the stop bits. When a complete frame is transmitted, it can be directly followed by a new frame, or the communication line can be set to an idle (high) state. Figure 19-4 illustrates the possible combinations of the frame formats. Bits inside brackets are optional.

Figure 19-4. Frame Formats



St	Start bit, always low.
(n)	Data bits (0 to 8).
P	Parity bit. Can be odd or even.
Sp	Stop bit, always high.
IDLE	No transfers on the communication line (RxDn or TxDn). An IDLE line must be high.

Parity Bit Calculation

The parity bit is calculated by doing an exclusive-or of all the data bits. If odd parity is used, the result of the exclusive or is inverted. The relation between the parity bit and data bits is as follows:

$$P_{even} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$
$$P_{odd} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

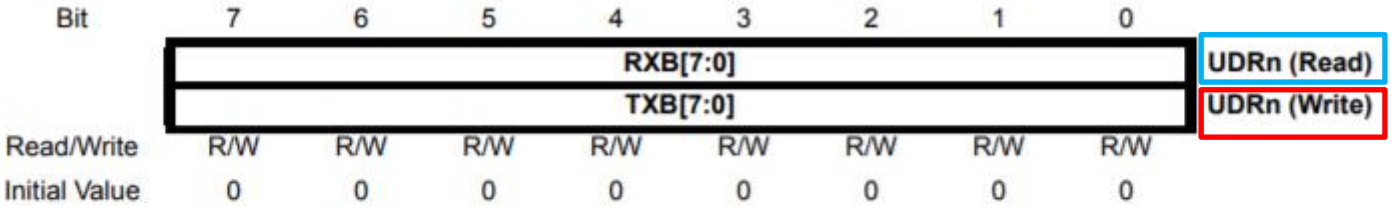
P_{even}	Parity bit using even parity
P^{odd}	Parity bit using odd parity
d_n	Data bit n of the character

If used, the parity bit is located between the last data bit and first stop bit of a serial frame.

3. UART 통신

7) ATmega328의 USART0 관련 레지스터
- 데이터 송/수신 레지스터

19.10.1 UDRn – USART I/O Data Register n



The USART transmit data buffer register and USART receive data buffer registers share the same I/O address referred to as USART data register or UDRn. The transmit data buffer register (TXB) will be the destination for data written to the UDRn register location. Reading the UDRn register location will return the contents of the receive data buffer register (RXB).

For 5-, 6-, or 7-bit characters the upper unused bits will be ignored by the transmitter and set to zero by the receiver.

The transmit buffer can only be written when the UDREn flag in the UCSRnA register is set. Data written to UDRn when the UDREn flag is not set, will be ignored by the USART transmitter. When data is written to the transmit buffer, and the transmitter is enabled, the transmitter will load the data into the transmit shift register when the shift register is empty. Then the data will be serially transmitted on the TxDn pin.

The receive buffer consists of a two level FIFO. The FIFO will change its state whenever the receive buffer is accessed. Due to this behavior of the receive buffer, do not use read-modify-write instructions (SBI and CBI) on this location. Be careful when using bit test instructions (SBIC and SBIS), since these also will change the state of the FIFO.

3. UART 통신

7) ATmega328의 USART0 관련 레지스터
- CTRL 레지스터

19.10.2 UCSRnA – USART Control and Status Register n A

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

UCSRnB – USART Control and Status Register n B

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

7) ATmega328의 USART0 관련 레지스터 - CTRL 레지스터

UCSRnC – USART Control and Status Register n C

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

- **Bits 7:6 – UMSELn1:0 USART Mode Select**

These bits select the mode of operation of the USARTn as shown in Table 19-4.

Table 19-4. UMSELn Bits Settings

UMSELn1	UMSELn0	Mode
0	0	Asynchronous USART
0	1	Synchronous USART
1	0	(Reserved)
1	1	Master SPI (MSPIM)(1)

Note: 1. See Section 20. "USART in SPI Mode" on page 166 for full description of the master SPI mode (MSPIM) operation

- **Bits 5:4 – UPMn1:0: Parity Mode**

These bits enable and set type of parity generation and check. If enabled, the transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The receiver will generate a parity value for the incoming data and compare it to the UPMn setting. If a mismatch is detected, the UPEn flag in UCSRnA will be set.

Table 19-5. UPMn Bits Settings

UPMn1	UPMn0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, even parity
1	1	Enabled, odd parity

- **Bit 3 – USBSn: Stop Bit Select**

This bit selects the number of stop bits to be inserted by the transmitter. The receiver ignores this setting.

- **Bit 2:1 – UCSZn1:0: Character Size**

The UCSZn1:0 bits combined with the UCSZn2 bit in UCSRnB sets the number of data bits (character size) in a frame the receiver and transmitter use.

Table 19-7. UCSZn Bits Settings

UCSZn2	UCSZn1	UCSZn0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

- **Bit 0 – UCPOLn: Clock Polarity**

This bit is used for synchronous mode only. Write this bit to zero when asynchronous mode is used. The UCPOLn bit sets the relationship between data output change and data input sample, and the synchronous clock (XCKn).

Table 19-8. UCPOLn Bit Settings

UCPOLn	Transmitted Data Changed (Output of TxDn Pin)	Received Data Sampled (Input on RxDn Pin)
0	Rising XCKn edge	Falling XCKn edge
1	Falling XCKn edge	Rising XCKn edge

3. UART 통신

7) ATmega328의 USART0 관련 레지스터
- Baud Rate(Arduino Board = 16MHz)

19.10.5 UBRRnL and UBRRnH – USART Baud Rate Registers

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRRn[11:8]				UBRRnH
	UBRRn[7:0]								UBRRnL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Table 19-12. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies

	f_{osc} = 16.0000MHz			
	U2Xn = 0		U2Xn = 1	
Baud Rate (bps)	UBRRn	Error	UBRRn	Error
2400	416	–0.1%	832	0.0%
4800	207	0.2%	416	–0.1%
9600	103	0.2%	207	0.2%
14.4k	68	0.6%	138	–0.1%
19.2k	51	0.2%	103	0.2%
28.8k	34	–0.8%	68	0.6%
38.4k	25	0.2%	51	0.2%
57.6k	16	2.1%	34	–0.8%
76.8k	12	0.2%	25	0.2%
115.2k	8	–3.5%	16	2.1%
230.4k	3	8.5%	8	–3.5%
250k	3	0.0%	7	0.0%
0.5M	1	0.0%	3	0.0%
1M	0	0.0%	1	0.0%
Max.(1)	1Mbps		2Mbps	

Note: 1. UBRRn = 0, error = 0.0%