EDDI
Electronic Design
Development Institute

에디로봇아카데미

임베디드 마스터 Lv1 과정

Ch9.

제 3기

2022. 2. 28

여건영

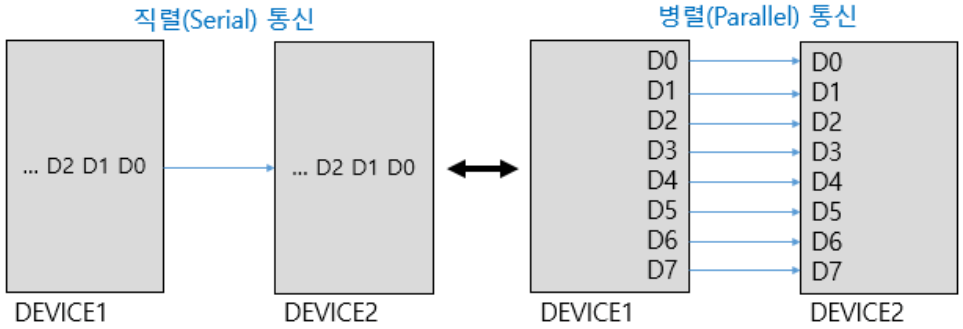
CONTENTS

UART 란?

UART(범용 비동기화 송수신기 : Universal asynchronous receiver/transmitter)는
병렬 데이터의 형태를 직렬 방식으로 전환하여 데이터를 전송하는 컴퓨터 하드웨어의 일종이다.

UART는 일반적으로 RS-232, RS-485, RS-422과 같은 통신 표준과 함께 사용한다.

* UART 통신 이론
 직렬(Serial) 통신 -> 1개의 입출력 핀을 통해 8개 비트를 한 번에 전송하는 방법
병렬(Parallel) 통신 -> n비트의 데이터를 전송하기 위해 n개의 입출력 핀을 사용하는 방법



MCU 에서는 흔히 직렬(Serial) 통신 방식이며, 그 중 많이 쓰이는 방법 중 하나가 UART 이다.

UART 란

시리얼 통신을 사용하기 위해서 보내는 쪽(TX)와 받는 쪽(RX)을 정하는 약속을 프로토콜(Protocol)이라고 한다.
MCU에서는 0은 GND, 1은 VCC로 데이터를 전송하고, 받는 쪽에서는 GND와 VCC를 다시 0과 1의 이진값으로 변환하여 사용한다.
TX - RX 간 원활한 데티어 교류를 위해, 데이터를 보내는 속도에 대한 약속(Protocol)이 정해져 있어야 한다.

그것을 보율(baud rate)라고 한다.
bps는 (bit per second) " 1초에 전송할 수 있는 비트의 수" baud rate는 " 1초에 몇 개의 신호가 전달되는가를 나타내는 단위 "

1200bps는 1초에 1200개의 비트가 전달되며, 9600bps는 1초에 9600개의 비트가 전달된다.
대부분의 신호전달은 하나의 신호에 하나의 비트로 대응된다. 이 경우에 bps와 baud rate가 같다. (Di-Bit 또는 Tri-Bit의 경우엔

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	$BAUD = \frac{f_{OSC}}{8(UBRRn + 1)}$	$UBRRn = \frac{f_{OSC}}{2BAUD} - 1$

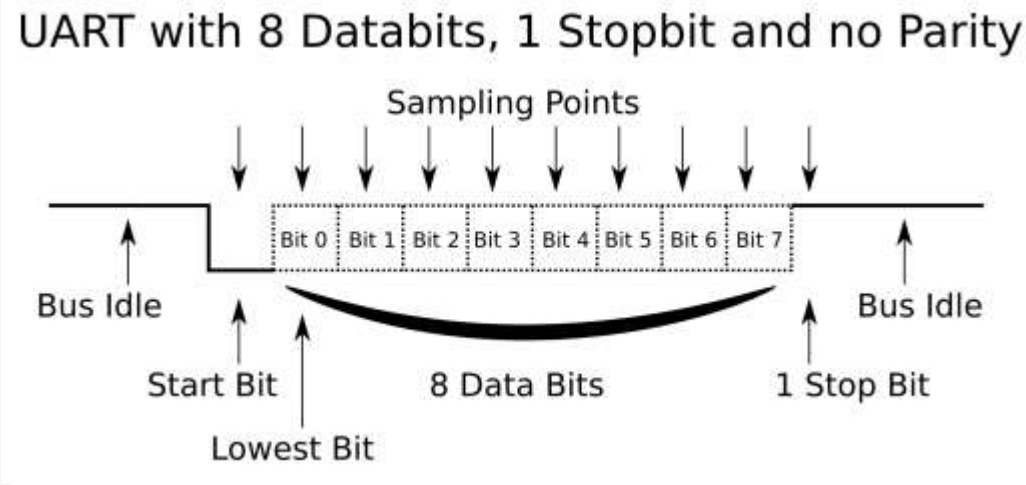
Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps)

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

UART 란

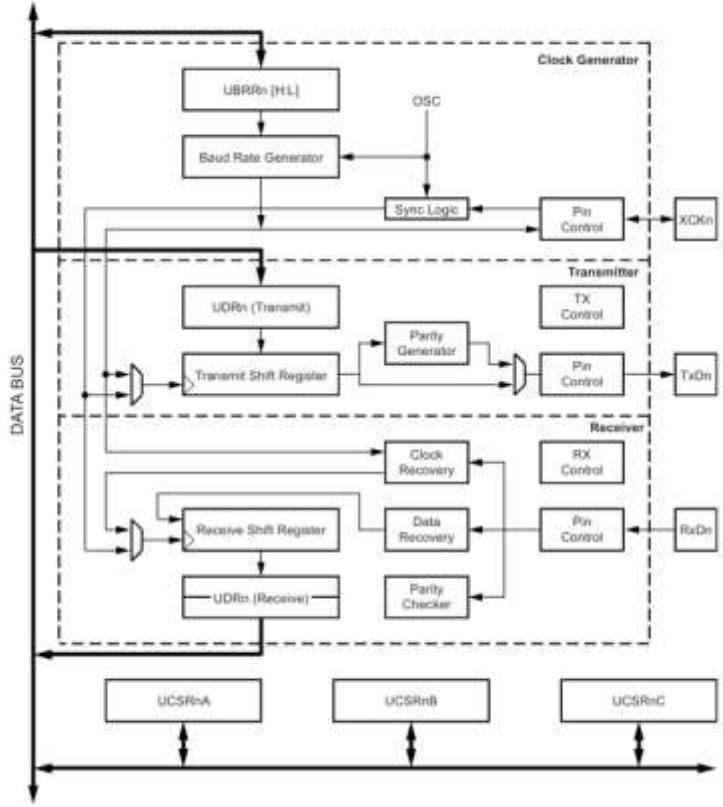
RX는 어디서부터 어디까지가 TX에서 보낸 데이터인지 알아내기 위해
'0'의 시작 비트(start bit)와 '1'의 정지 비트(stop bit)를 사용한다.

UART는 바이트 단위 통신을 주로 사용하며, 시작 비트(start bit)와 정지 비트(stop bit)가 추가되어,
10비트 데이터를 전송하는 것이 일반 적이다. (패리티 비트를 사용하지 않을 경우에)



UART

Figure 19-1. USART Block Diagram[1]



송신 설명 : TX_Shift_Register는 사용자가 프로그램적으로 접근할 수 없고,
회로적으로 UDRn의 내용이 로드 되어 BAUD 클럭에 맞추어 쉬프트 되면서 TXD 핀으로 송출 된다.

송신 Sequence :
UDRn이 비어있음을 확인(UDREn : SET) →
UDRn에 data를 씀(UDREn : Clear) →
UDRn의 내용이 TX_Shift_Register로 로드 됨 →
BAUD 클럭에 맞추어 쉬프트 되면서 TXD 핀으로 송출 됨

수신 설명 : RX_Shift_Register는 RXD 핀으로 수신된 신호가 수신 BAUD 클럭에 맞추어 쉬프트 되다가 스톱 비트가 들어오면 UDRn로 저장된다.
UDRn의 내용은 수신된 데이터로 데이터 버스를 통해 사용자가 읽어서 처리한다.

수신 Sequence :
RXD 핀으로 수신된 신호가 BAUD 클럭에 맞추어 RX_Shift_Register로 쉬프트 됨 →
스톱 비트가 들어오면 UDRn로 저장됨(RXCn : SET) →
RXCIE : SET 되어 있는 경우 인터럽트 발생 →
UDRn의 데이터를 읽어 처리한다.(RXCn : Clear)

UART 사용

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	

```
void uart_init(void)
{
    //sbi(UCSR0A, U2X0);
    UCSR0A |= (1 << U2X0);

    UBRR0H = 0x00;
    UBRR0L = 207;

    UCSR0C |= 0x6;

    UCSR0B |= (1 << RXEN0);
    UCSR0B |= (1 << TXEN0);
    //sbi(UCSR0B, RXEN0);
    //sbi(UCSR0B, TXEN0);
}
```

* UCSR0A 레지스터 : UART 통신의 제어와 상태 확인을 위한 레지스터
  - 2배속 설정 : U2X0 비트

비트 7 : RXCn (USARTn Receiver Complete)
데이터가 수신되어 수신 쉬프트 레지스터에서 UDR 레지스터로 전달될 때 이 비트가 SET 됨. 수신 인터럽트(RXCIE)가 인에이블 되어 있는 경우 수신 인터럽트가 발생 됨.
UDR 레지스터를 읽으면 RXC 플래그는 자동으로 Clear 됨

비트 6 : TXCn 송신완료(Transmit Complete)
송신 쉬프트 레지스터의 전체 내용이 모두 전송되고(stop bit까지) 비어 있고,
UDR도 비어 있을 때 이 비트가 1로 SET 됨. TXIE 비트가 1로 설정되었을 때,b송신 인터럽트 발생.
인터럽트가 처리될 때 TXC는 회로적으로 클리어 되거나 TXC에 1을 쓸때 bTXC 비트가 클리어 됨

비트 5 : UDREn (USARTn Data Register Empty)
새로운 송신 데이터를 받기 위한 상태 플래그로 송신 쉬프트 레지스터로 UDR의
내용이 로드 될 때 1로 SET 됨. UDRIE 비트가 1로 SET되어 있으면 인터럽트 발생가능.
UDR에 데이터를 쓸 때 UDRE는 0으로 클리어 됨

Double Speed Operation(U2Xn)

UCSRnA에서 U2Xn 비트를 설정하여 전송 속도를 두 배로 늘릴 수 있다.
(비동기 모드 동작에서만)

해당 모드 사용 시, 송신기에서는 성능 저하가 발생하지 않지만
수신기에서는 더 정확한 baud rate 설정과 system clock이 요구된다.

UART 사용

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	

```
void uart_init(void)
{
    //sbi(UCSR0A, U2X0);
    UCSR0A |= (1 << U2X0);

    UBRR0H = 0x00;
    UBRR0L = 207;

    UCSR0C |= 0x6;

    UCSR0B |= (1 << RXEN0);
    UCSR0B |= (1 << TXEN0);
    //sbi(UCSR0B, RXEN0);
    //sbi(UCSR0B, TXEN0);
}
```

* UBRR0H, UBRR0L 레지스터
  8비트 레지스터 조합에 의한 가상 16비트 레지스터
  - 12비트로 Baud Rate 결정

* Baud Rate에 따른 레지스터 계산값은 정수가 아니므로 근사값으로 설정
  - 약간의 오차 발생
  - 비트 데이터 확인을 위해 여러 번의 샘플링 필요
    2배 모드에서는 샘플링 횟수가 절반

	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%

UART 사용

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	

* 통신 방법 및 데이터 형식 지정
  - UCSZn1~0 : 데이터 비트 수 설정 // 8bit, UCSR0B 레지스터와 함께 사용
  - UMSELn1~0 : 통신 모드 설정 // 비동기 USART 모드
  - UPMn1~0 : 패리티 비트 설정 // no parity bit
  - USBSn : 정지 비트 설정 (0 : 1비트, 1 : 2비트)
  - UCPOLn : 클록 극성 설정 (동기 모드에서만 사용)

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

```
void uart_init(void)
{
    //sbi(UCSR0A, U2X0);
    UCSR0A |= (1 << U2X0);

    UBRR0H = 0x00;
    UBRR0L = 207;

    UCSR0C |= 0x6;

    UCSR0B |= (1 << RXEN0);
    UCSR0B |= (1 << TXEN0);
    //sbi(UCSR0B, RXEN0);
    //sbi(UCSR0B, TXEN0);
}
```

UART 사용

UCSRnB – USART MSPIM Control and Status Register n B

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIE	RXENn	TXENn	-	-	-	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R	R	
Initial Value	0	0	0	0	0	1	1	0	

* UART통신의 송신 및 수신을 가능하게 하기 위해 사용
  - 수신 가능 설정 비트 : RXEN0
  - 송신 가능 설정 비트 : TXEN0

```
void uart_init(void)
{
    //sbi(UCSR0A, U2X0);
    UCSR0A |= (1 << U2X0);

    UBRR0H = 0x00;
    UBRR0L = 207;

    UCSR0C |= 0x6;

    UCSR0B |= (1 << RXEN0);
    UCSR0B |= (1 << TXEN0);
    //sbi(UCSR0B, RXEN0);
    //sbi(UCSR0B, TXEN0);
}
```

UART 사용

UDRn – USART I/O Data Register n

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDRn (Read)
	TXB[7:0]								UDRn (Write)
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	

* USART Data Register
* UDR0 레지스터에 데이터를 쓰면 TXB 레지스터에 기록되고,
  UDR0 레지스터에서 데이터를 읽으면 RXB 레지스터의 값이 읽힘

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	

```
unsigned char uart_rx(void)
{
    while(!(UCSR0A & (1 << RXC0)))
    {
        ;
    }
    return UDR0;
}
void uart_tx(char data)
{
    while(!(UCSR0A & (1 << UDRE0)))
    {
        ;
    }
    UDR0 = data;
}
```

* 송신 가능 상태 확인 : UDRE0
  UDR0이 비어 있어 데이터를 받을 준비가 되어 있을 때 1이 된다.
  UCSR0B 레지스터의 UDRIE0 비트와 함께 사용 되어 송신 데이터 레지스터 준비 완료 인터럽트를 발생 시킬 수 있다.

  수신 완료 상태 확인 : RXC0
  UDR0에 읽지 않은 문자가 있을 때는 1이 되고 버퍼가 비어 있을 때는 0이 된다.

UART 사용

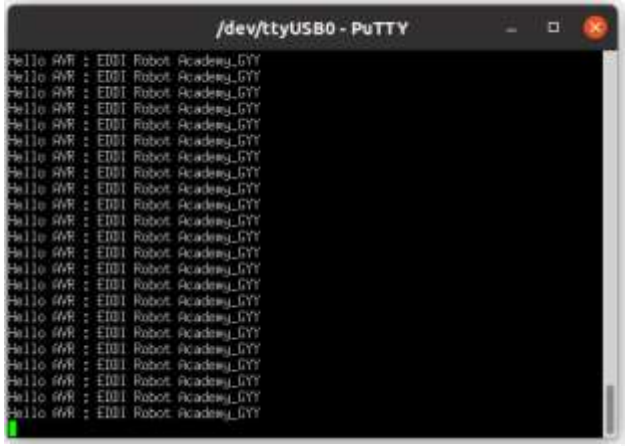
```c
#define F_CPU   16000000UL
#include <avr/io.h>
#include <avr/delay.h>

#include "uart.h"
#include "spi_25LC010A.h"
int main(void)
{
    // 필요한 사항
    // 1. UART
    // 2. SPI
    uint8_t i;
    uart_init();
    spi_init();

    while(1)
    {
          uart_string_tx("Hello AVR : EDDI Robot Academy_GYY\n\r\0");
          _delay_ms(500);
    }

    return 0;
}
```

```c
void uart_string_tx(char *str)
{
    // 문자열이 끝날때까지 1byte 씩 전송을하다 NULL에서 멈춤
    while(*str != '\0')
    {
        uart_tx(*str);
        str++;
    }
}
```



* uart_string_tx(char *str)
  문자열이 NULL(\0) 값까지 1byte 씩 전송한다.

* uart_tx(char data)
  UDR0이 비어 있어 UDRE0이 1이 되어 있고,
  while(!(UCSR0A & (1 << UDRE0))); 문을
  빠져 나가게 되고
  함수의 인자로 받게 된 data가 UDR0에 입력되고
  데이터가 전송 된다.