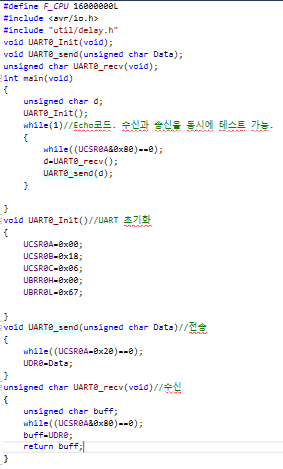
UART

이번장에서는 비동기 통신 방식 중 하나인 UART에 대해서 다룬다. UART는 RS-232C의 논리규격을 따르는 비동기 통신방식으로 매우 많이 쓰이고 있으며 그 범용성 때문에 대부분의 기기에 탑재되는 통신방식이다. 우리가 사용하는 ATmega128A에도 2개가 탑재되어 있는데 오늘은 이부분에 대해서 다룰 예정이다.

기본적으로 이 통신방식이 어떻게 동작하는지에 대해서 이해할 필요가 있다. UART는 2개의 선을 이용해서 동작한다. 한쪽은 송신이라는 뜻의 Transmit를 줄여서 TX라고 쓰며 나머지 한쪽은 수신이라는 뜻의 Receive를 줄여 Rx라고 쓴다. 실제로 사용할 때는 송신은 수신에다가 연결하는 식으로 결선한다. 이들 선은 전압크기로 신호를 주고받으며 전압이 Hi일 때 1 Low일 때 0인 식으로 데이터를 주고 받는다. 별도의 클럭은 없는 대신에 통신 속도,통신 데이터 비트 수 등을 정해 놓고 통신하고 송신과 수신을 둘로 나누었기 때문에 전 이중 통신이 가능한 통신방식이다.

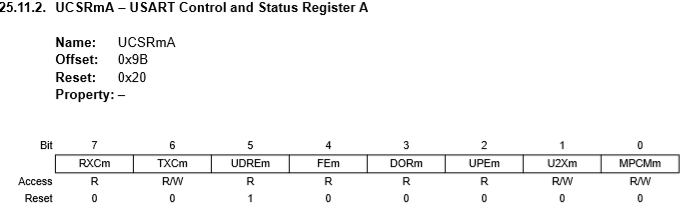
코드가 살짝 긴데 이번에는 보기 좋게 하기위해서 함수를 사용해서 코딩했을 뿐만 아니라 이 코드에는 송신과 수신을 같이 시연하고있다.

UART0\_init은 UART0을 초기화 하기위해 선언해놓은 함수로 여기에서 UART제어비트와 속도를 설정한다.

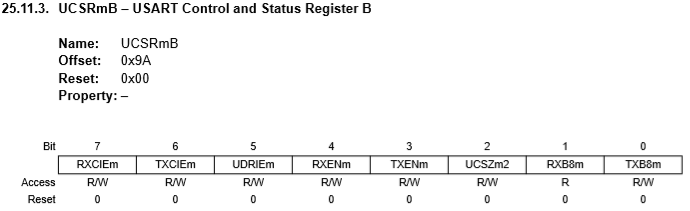
UART0\_send는 실제로 데이터 비트를 전송하는 함수이고 UART\_recv는 데이터를 받아오는 함수이다.

이 코드는 컴퓨터에서 입력을 한번 받아서 다시 컴퓨터로 전송해주는 코드로 되어있다. 이를 이용해 송신과 수신의 방법을 한번에 알수있다.

먼저 초기화 함수부터 확인해보자. 초기화 함수에는 레지스터가 무려 5개가 사용되었는데 하나 하나씩 살펴보기로 하자.

  
UCSR레지스터들은 전부 UART를 제어하기위한 레지스터이다. A는 전송과 관련된 플래그 비트들과 전송 속도제어를 위한 비트 한 개로 구성되는데 여기서 우리가 사용하는 비트만 확인하면

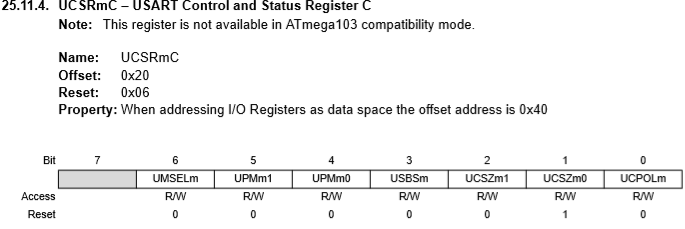
* RXC0=수신 확인 비트로 수신이 됨을 알려주는 비트다
* UDRE0=데이터 레지스터가 비었는지를 확인하는 비트



B레지스터는 기능의 동작 여부를 정하는 레지스터로 각각의 기능의 키고 끄는 것을 제어하는 레지스터이다.

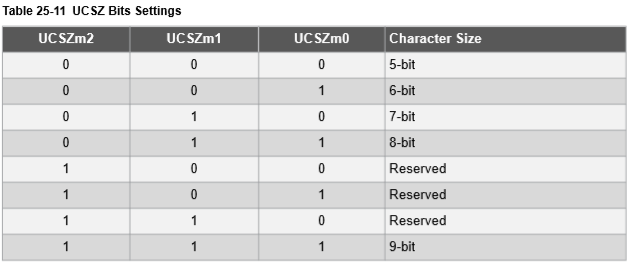
* RXCIE/TXCIE=각각 수신,송신 인터럽트를 제어하는 비트로 1로 셋할 경우 송신 인터럽트/수신 인터럽트가 동작한다
* UDRIE=데이터 레지스터 빈공간 인터럽트로 데이터 레지스터가 비었을 때 인터럽트를 보내는가에 대한 비트이다..
* RXEN/TXEN=각각 UART의 수신/송신기능을 활성화 시키는 비트로 이를 통해 둘 중 하나만 작동시키거나 한쪽만 사용하거나 할수 있다.
* UCSZm2=한번에 보내는 데이터 양에 대한 비트로 뒤에 나올 C레지스터와 이어진다.
* RXB8/TXB8=여분의 데이터 레지스터로 데이터 양과 관련있다.

여기에서는 따로 인터럽트를 쓰진 않고 제어비트를 통해 확인하기 때문에 RXEN/TXEN만 1로 세팅해 놨다.



C레지스터는 UART의 통신에 필요한 실제적인 통신 규격을 정하는 레지스터이다.

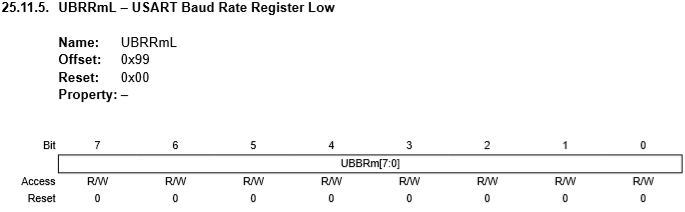
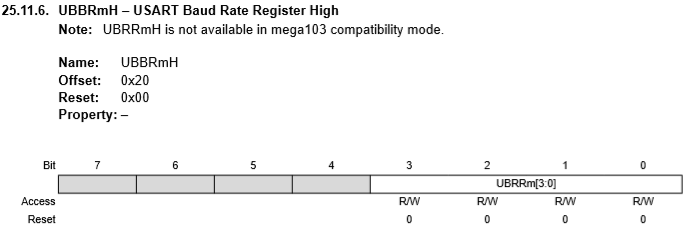
* UMSELm=통신 모드를 선택하는 비트로 비동기/동기 모드를 고 를수 있는데 동기 모드는 통신에 별도로 클럭을 인가 받는 통신모드다. 0일경우 비동기 모드이다
* UPM0~1=페리티 비트를 설정하는 레지스터로 페리티 비트는 오류정정 비트로 오류정정오류정 쓸 것 인가. 어떤 오류정정 비트를 쓸 것 인가를 고를 수 있다.
* USBS=정지비트의 수를 정하는 비트로 0일경우 1비트, 1일경우 2비트로 세팅된다



* UCSZm0~1=데이터 크기 비트로 한번에 통신할 때 몃 비트를 주고 받는지에 대해 정한다. B에 있는 레지스터를 포함해서 정하며 일반적으로는 8비트가 가장 많이 쓰인다
* UCPOL=동기 모드일 때 클럭에 대한 세팅으로 여기에서는 사용하지 않는다.

작성된 코드 에서는 별도의 페리티 비트를 쓰지 않고 동기통신도 쓰지 않기 때문에 데이터 크기만 8비트로 세팅하기 위해서 UCSZ01과 UCSZ00의 비트만 1로 만들어놨다.

다음은 통신 속도에 대한 세팅이다.



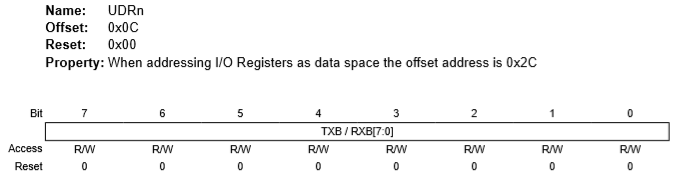
UBBR레지스터는 통신속도를 정하기 위한 레지스터로 2개로 나뉘어져 있는데 이는 입력 해야될 통신속도가 8비트의 한계를 넘어서는 경우가 존재하기 때문이다.

여기에 입력 해야될 통신 속도는 직접 구해야 되는데 일반적으로 여기에 쓰이는 통신속도는 마이크로 컨트롤러의 동작 주파수를 16분주 분주기를 통해 나눈 뒤에 UBBR+1의 값으로 한번 더 나눈 값이다. 이를 공식으로 쓰면 다음과 같이 나온다.

이 공식을 기반으로 UBBR에 입력할 값을 구하는 방법은 다음과 같다

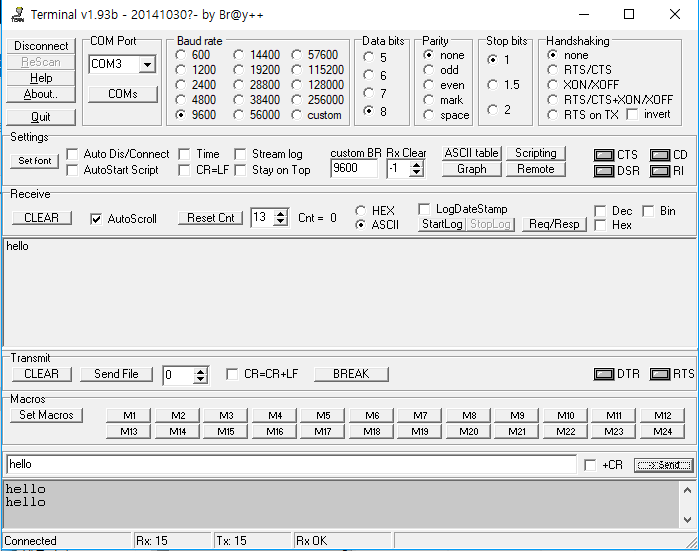
여기서는 통신에 쓰일 보레이트를 9600으로 잡아 놓고 계산했기 때문에 우리가 입력 해야 될 값은 103이며 8비트 레지스터에 다 들어가는 관계로 UBBRL에만 103을 16진수로 치환한 0x67을 넣었다. 실제로는 약간의 오차가 있어서 제대로 동작 안 할 가능성도 있긴 하지만 거의 대부분의 상황에서 제대로 동작하기 때문에 별 걱정을 하지 않아도 된다.

여기까지 해서 세팅이 모두 종료되었다. 이제부터는 통신으로 넘어간다.



UDR0는 송신데이터와 수신데이터가 들어가는 레지스터로 8비트로 구성되어 있으며 9비트 전송시에는 USCRB에 있는 1비트를 포함해서 전송 받는다. 송신은 USCR0A의 UDRE0비트를 확인해 데이터 레지스터가 비었으면 송신용 데이터를 보내고 수신은 RXC0비트를 확인해 수신 상황이라는것을 확인하고 데이터를 전송 받는다.

이제 하드웨어 적인 부분에 대해서 살펴보자. 타이머때 처럼 각각 용도가 설정된 핀들이 존재하는데 UART라고 예외가 아니다 마이크로 컨트롤러의의 핀 배치를 확인해보면 RX0와 TX0라고 되어있는 핀들이 있는데 이핀이 UART0에 배정된 핀이다. 128A에서는 RX0가 PE0에 TX가 PE1에 배정되어 있다. 이 핀이 데이터를 받고 보내주는 핀이다. 이것과 통신하려면 UART변환기라는 물건이 필요한데 변환기의 TX를 컨트롤러의 PE0에 연결하고 RX를 PE1에 연결하면 통신을 할 수 있고 이를 터미널 같은 프로그램을 불러서 연결해보면 보내주는 데이터를 그대로 다시 PC 로 보내준다.



이런 식으로 말이다

이를 통해서 우리는 이 통신이 제대로 동작 한다는 것을 알 수 있다.

지금 보내주는 데이터 함수는 1비트씩 보내는 식이라 후에 사용할 때는 약간 불편할 수 있는데 이는 후에 몃몃 명령어를 더 추가해서 printf문을 사용 할수 있도록 바꿔주면 된다. printf문을 추가할 경우 훨씬 더 많은 문장 데이터를 한꺼번에 보낼 수 있기 때문에 더욱더 편하게 쓸수있다.

통신 관련해서 주의 해야될 상황이 있다. UART는 전압 레벨을 통해 통신하는 방식이기 때문에 전압의 기준점에 대해 유의 해야 될 필요가 있다. 접지가 제대로 안되어 있다면 통신이 안되거나 하는 경우가 많이 발생하기 때문에 여기에 유의해야 된다. 또한 RS-232와 논리 규격에서 호환이 되지만 물리 규격에서 달라서 UART를 PC의 RS-232포트에 직접 연결하면 RS-232의 높은 전압 때문에 고장이 난다. 따라서 서로 통신하고 싶다면 MAX232등의 컨버터를 사용해서 연결 시켜줘야 된다

지금까지 UART를 다루는 방법에 대해서 배워보았다. UART는 많은 곳 에서 쓰이는 통신방식이기 때문에 향후 써먹을 기회가 많을 것이다.