**전자회로실험 보고서**



학과 전자공학과

학번 1218165

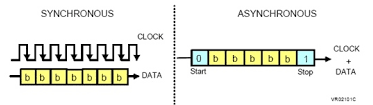
이름 김재욱

제출일 2020-09-24

**1. UART와 USRT**

**실습 이론**

UART는 비동기식 시리얼통신 방법이며 USRT는 동기식 시리얼통신방법이다. MCU는 동기식/비동기식 모두 지원하므로 USART라고 한다.



USRT는 통신시 클럭을 같이 보내며 이 클럭을 이용해 데이터를 취득할 수 있다.

시작비트와 스탑비트를 사용하지 않아 데이터 전송 효율을 높일수 있지만 클럭 핀을 필요로 한다.

UART는 통신시 클럭을 동반하지 않으며 송,수신 측의 통신속도를 미리 맞춰 놓는다. 시작비트와 스탑비트를 보내 데이터를 취득할 수 있다.

**2. 시리얼통신**

직렬통신이란 송/수신측 간 연결이 하나로만 이루어져 있어 주어진 하나의 선으로 데이터의 전송이 이뤄지며 문자를 송/수신 하는 경우 그 문자의 아스키코드를 전달한다.

아스키코드란 표준화된 7비트 부호체계이며 단순하게 각 문자 숫자를 할당하고 2진수변환이라고 생각하면 쉽다

EX) 0을 보내면 '0'에 해당하는 아스키코드는 48이며 이를 2진수로 나타내면 00110000이다.

**3. ATMEGA128의 직렬통신포트**

ATMEGA128에서는 2개의 직렬통신포트(USART0,USART1)를 가지고 있다. 앞서 말한것 처럼 비동기식 통신에서는 통신속도를 맞추는 등의 송/수신측 간 약속이 필요하며 이 약속에 대한 부분을 레지스터를 이용해 설정한다.

관련된 레지스터는 UDRn UCSRnA UCSRnB UCSRnC UBRRnH/L이 있다

3-1.UDRn

송신과 수신 buffur로 나누어진 8bit 레지스터이며 송/수신 데이터가 버퍼에 담긴다.

3-2 UCSRnA

송신과 수신 상태를 알수 있는 레지스터이며 주요한 기능을 정리하였다.

수신 buffur에 수신할 데이터가 들어오면 RXCn(bit 7번)는 1이며 수신완료 인터럽트가 발생되고 들어오지 않으면 0이다.

송신 buffur의 데이터가 시프트레지스터로 옮겨서 buffur가 비면 TXCn(bit 6번)1이며 송신완료 인터럽트가 발생된고 그렇지 않으면 0이다..

송신 buffur에 송신할 데이터가 있으면 UDREn(bit 5번)은 0이며 비어있으면 1이다.

**3-3 UCSRnB**

송/수신 사용여부를 결정하며 송/수신 완료 인터럽트 허가여부를 결정하는 레지스터이며 주요한 기능을 정리하였다.

RXCIEn (bit 7번): 수신완료 인터럽트 허가

TXCIEn (bit 6번): 송신완료 인터럽트 허가

RXENn (bit 4번): USART 수신부 사용허가

TXENN (bit 3번): USART 송신부 사용허가

UCSZn2 (bit 2번): 전송 문자의데이터 비트수 결정 (UCSRnC의 UCSZn1,UCSZn0과 함께쓰임 )

**3-4 UCSRnC**

송/수신측 간의 통신을 설정해주는 레지스터이다.

UMSELn : 동기,비동기인 USART의 동작방식 결정

UPMn1 : 패리티 설정

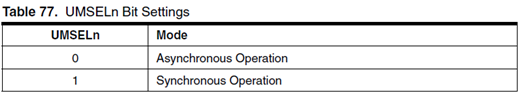
UPMn0 : 패리티 설정

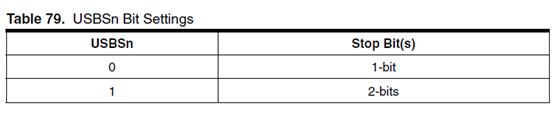
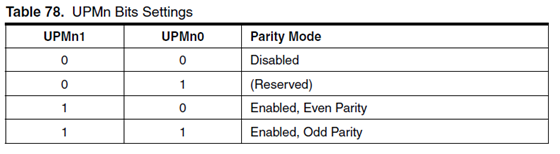
USBSn : 정지비트 선택

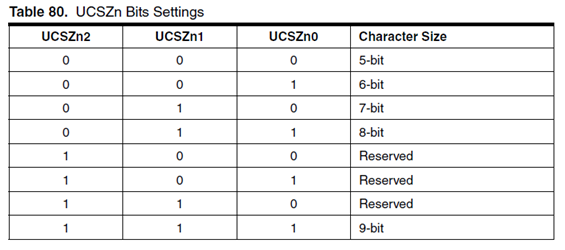
UCSZn1 : 문자의 데이터 비트수 결정

UCSZn0 : 문자의 데이터 비트수 결정

UCPOLn : 클럭극성 설정



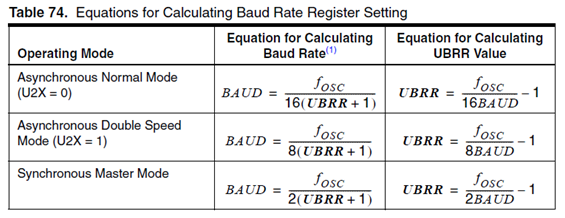




**3-5 UBRRnH/L**

송/수신측 간의 송/수신 속도를 설정하는 레지스터이며 HIGH는 4bit만 사용하며 와 LOW는 8bit모두 사용한다.

아래의 테이블 중 비동기 1배속에서는 첫번째 ROW(제목 제외)를 사용해 계산할 수 있다.



**실습 1**

**실습내용 및 결과**

|  |
| --- |
|  |
| /\*  \* week2-1.c  \*  \* Created: 2020-09-18 오전 11:52:59  \* Author : lobgd  \*/  // USART0, fosc=14.7456MHz, 9600bps  // 패리티 없음, 데이터 8비트, 정지비트 1비트  #define F\_CPU 16000000  #include <avr/io.h>  #include <avr/interrupt.h>  #include <util/delay.h>  #include <stdio.h>  volatile unsigned char ch ;  ISR(USART0\_RX\_vect){ //수신이 완료되었을때 인터럽트의 함수  ch = UDR0 ; ///usart0를 통해 데이터 읽기  PORTB = ch ;  }  int main(void)  {  DDRB = 0xff ;  UCSR0A = 0x00;  UCSR0B = 0b10011000; // RXCIE0=1, TXEN0=1, RXEN0=1 ==> 수신완료 인터럽트 허가, 수신허가.송신허가  UCSR0C = 0b10000110; // 비동기 통신, 패리티 없음, ==>고정  // 데이터 비트 : 8비트, 정지비트 : 1비트  UBRR0H = 0; //보오 레이트 레지스터  UBRR0L = 103; //보오 레이트 레지스터, fosc=14.7456MHz, BAUD=9600bps,  SREG = 0x80;  while(1);  } |
| 실습1 결과 동영상 첨부  컴퓨터의 화상키보드를 이용하여 숫자를 입력하였다.  직렬통신에서 데이터 송/수신이 아스키코드로 이뤄지기 때문에 수신과정에서 아스키코드를 수신되어 저장되는 변수의 타입에 따라 변환된 데이터가 저장된다.   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | ASCII | 16진수 | 2진수 | ASCII | 16진수 | 2진수 | | 48 | 30 |  | 53 | 35 |  | | 49 | 31 |  | 54 | 36 |  | | 50 | 32 |  | 55 | 37 |  | | 51 | 33 |  | 56 | 38 |  | | 52 | 34 |  | 57 | 39 |  | |

**실습 2**

|  |
| --- |
|  |
| /\*  \* week2-1.c  \*  \* Created: 2020-09-18 오전 11:52:59  \* Author : lobgd  \*/  // USART0, fosc=14.7456MHz, 9600bps  // 패리티 없음, 데이터 8비트, 정지비트 1비트  #define F\_CPU 16000000  #include <avr/io.h>  #include <avr/interrupt.h>  #include <util/delay.h>  #include <stdio.h>  volatile unsigned char rx\_data ;  volatile unsigned char tx\_data ;  ISR(USART0\_RX\_vect){ //수신이 완료되었을때 인터럽트의 함수  rx\_data = UDR0 ; ///usart0를 통해 데이터 읽기  if((rx\_data>='a')&&(rx\_data<='z')){  tx\_data = rx\_data-'a'+'A';  UDR0=tx\_data;  }  //아스키 코드를 이용해서 대문자로 변환하기  }  int main(void)  {  UCSR0A = 0x00;  UCSR0B = 0b10011000; // RXCIE0=1, TXEN0=1, RXEN0=1 ==> 수신완료 인터럽트 허가, 수신허가.송신허가  UCSR0C = 0b10000110; // 비동기 통신, 패리티 없음, ==>고정  // 데이터 비트 : 8비트, 정지비트 : 1비트  UBRR0H = 0; //보오 레이트 레지스터  UBRR0L = 103; //보오 레이트 레지스터, fosc=14.7456MHz, BAUD=9600bps,  SREG = 0x80;  while(1);  } |
| 실습2 결과 동영상 첨부  컴퓨터의 화상키보드를 이용하여 어떤 문자(대,소)를 전달하는지 실습결과 동영상에 나타내었다. |

**실습 3**

|  |
| --- |
|  |
| /\*  \* week2-1.c  \*  \* Created: 2020-09-18 오전 11:52:59  \* Author : lobgd  \*/  // USART0, fosc=14.7456MHz, 9600bps  // 패리티 없음, 데이터 8비트, 정지비트 1비트  #define F\_CPU 16000000  #include <avr/io.h>  #include <avr/interrupt.h>  #include <util/delay.h>  #include <stdio.h>  volatile unsigned char rx0\_data ;  volatile unsigned char rx1\_data ;  ISR(USART0\_RX\_vect){ //수신이 완료되었을때 인터럽트의 함수  rx0\_data = UDR0 ; ///usart0를 통해 데이터 읽기  UDR1 = rx0\_data;  }  ISR(USART1\_RX\_vect){ //수신이 완료되었을때 인터럽트의 함수  rx1\_data = UDR1 ; ///usart1를 통해 데이터 읽기  UDR0 = rx1\_data;  }  int main(void)  {  //pc1  UCSR0A = 0x00;  UCSR0B = 0b10011000; // RXCIE0=1, TXEN0=1, RXEN0=1 ==> 수신완료 인터럽트 허가, 수신허가.송신허가  UCSR0C = 0b10000110; // 비동기 통신, 패리티 없음, ==>고정  // 데이터 비트 : 8비트, 정지비트 : 1비트  UBRR0H = 0; //보오 레이트 레지스터  UBRR0L = 103; //보오 레이트 레지스터, fosc=14.7456MHz, BAUD=9600bps,    //pc2  UCSR1A = 0x00;  UCSR1B = 0b10011000; // RXCIE1=1, TXEN1=1, RXEN1=1 ==> 수신완료 인터럽트 허가, 수신허가.송신허가  UCSR1C = 0b10000110; // 비동기 통신, 패리티 없음, ==>고정  // 데이터 비트 : 8비트, 정지비트 : 1비트  UBRR1H = 0; //보오 레이트 레지스터  UBRR1L = 103; //보오 레이트 레지스터, fosc=14.7456MHz, BAUD=9600bps,  SREG = 0x80;  while(1);  } |
| 실습3 결과 동영상 첨부  PUTTY를 두개 켜서 양쪽에 PUTTY에 각각 송신한 데이터가 출력되도록 하였다. |

Atmega 128의 USART핀을이용하여 비동기식 직렬통신을 배울수 있었다. 아두이노를 통해 시리얼통신을 할때는 USART 자체가 직렬 비동기 통신을 의미하는 줄로만 알았는데 이번 실험을 통해 USART가 UART,USRT를 설정에따라 사용할 수있게 때문임을 알게 되었으며 그 차이에는 데이터 전송시의 클럭의 존재유무로 설명할 수 있었다.

**고찰**

위실험에서 가장 먼저 논의 되어야할 것은 통신하는 각 개체의 RX와 TX의 연결이다.

MCU와 다른 통신 개체가 연결될때는 RX핀과 TX핀이 서로 교차 되어 있어야만 통신이 가능하다. 송신 개체의 TX핀으로 데이터를 수신 객체에서 RX핀으로 받는다면 TX와 RX가 연결 되어있어야 함을 직관적으로도 알수 있다. 실제 하드웨어 연결에서는 USB to TTL과 MCU연결시 RX와 TX를 교차하여 연결한다.

하지만 proteus상에서 atmega128 과 compim을 연결할때는 RX핀끼리 TX핀끼리 연결해야만 실행됨을 확인했다.

compim의 특성때문에 교차없이 연결을 해도 통신이 되는 것 같아 자료를 찾아보았는데 왜 그렇게 연결해야하는지 확실이 알수 없었다. 하지만 compim의 대략적인 역할에 대해 알수 있었다.

compim의 물리적 시리얼 포트를 모델링 한것이며 시리얼 통신을 수신하여 회로에 디지털 신호로 표시하는 역할이다.

추가적으로 실습 1에서 사용했던 max232의 역할에 대해서도 알수 있었다.

max232는 TTL레벨을 RS레벨로 바꿔주는 역할이다. 이는 컴퓨터와 Atmega 128의 통신을 위해 1로 정의되는 신호를 변화하는 것이다. Atmega128에서의 신호인식은 5v(TTL레벨)이고 컴퓨터는 12v(RS레벨)를 신호로 인식하기 때무네 서로의 통신을위해 12v->5v 5v->12v의 변환이 필요하다.

하지만 proteus에서 compim자체가 TTL레벨과 RS레벨간 변환을 지원하기 때문에 max232를 달지 않아도 가상시리얼통신이 가능했다. 그래서 max232를 제외하고 compim만을 이용하여 회로구성을 하였다.

실제 회로에서는 꼭 max232와 연결되어야 한다.

송/수신시 발생하는 인터럽트를 활용하여 수신시 인터럽트를 통해 데이터를 저장할 수 있었다. 수신 버퍼의 데이터를 확인하는 번거로움을 줄일 수 있었다.

인터럽트 발생시 호출되는 함수를 ISR또는 인터럽트 핸들러라고 부른다. atmega128에서는 ISR을 등록하기 위해 ISR()을 사용하는데 이때 매개변수로 인터럽트가 일어난 곳을 알려준다 이번 실습에서는 USART0\_RX\_VECT,USART1\_RX\_VECT를 사용하였다. 아래에 INTERRUT VECTOR TABLE의 일부이다.



