실험제목 : TWI(Two Wire Interface)

실험목적

Atmel사에서 개발한 직렬 통신의 한 방법인 TWI(TWO Wire Interface)의 개념을 이해하고
ATmega328PB와 TWI 방식의 3축 가속도 센서 소자인 ADXL345 간의 데이터 통신에 대해 이해한다.

실험 준비물

Microchip Studio 7

RealTerm / TeraTerm

Atmega328PB Xplained Mini

ADXL345 3-Axis Accelerometer Module (GY-291)

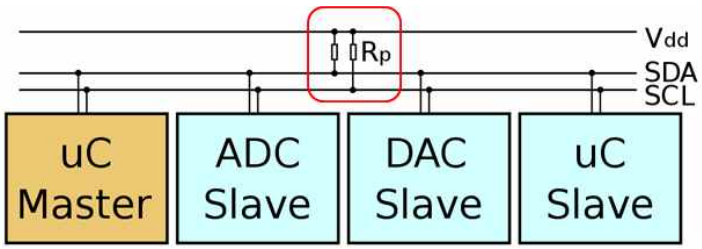
실험에 필요한 예비지식

1. TWI 정의

Philips Semiconductor에서 개발한 직렬 통신의 한 방법인 I²C (Inter-Integrated Circuit)의 일부 기능만을 구현한 것이 Atmel사에서 개발한 TWI(Two Wire Interface) 통신 방식이다. 이 인터페이스 방식은 클럭 신호선(SCL)과 데이터 신호선(SDA)의 두 가닥으로 직렬 통신을 구현한 동기식 통신 방식이며, 반 이중방식 통신(Half Duplex)이다. 하나의 Master와 여러 개의 Slave를 가질 수 있는데 각 Slave는 7비트로 구성되는 고유한 주소를 가진다. 주로 근거리 통신, 즉 마이크로컨트롤러와 주변장치간의 저속 직렬 통신에 사용된다.

2. TWI 연결

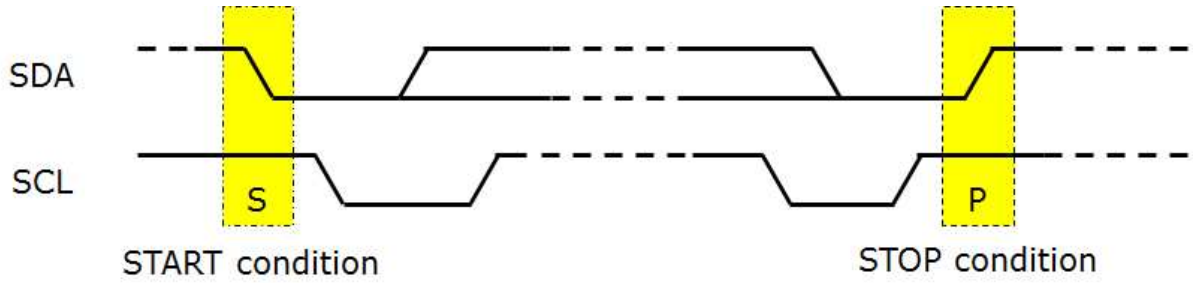
① SCL (Serial Clock) : Master Device로부터 출력되어 Slave Device로 입력되는 신호.
② SDA (Serial Data) : Transmitter로부터 출력되어 Receiver로 입력되는 신호.



3. TWI 동작에 따른 장치 구분

Term	Description
Master	The device that initiates and terminates a transmission. The Master also generates the SCL clock.
Slave	The device addressed by a Master.
Transmitter	The device placing data on the bus.
Receiver	The device reading data from the bus.

4. START, STOP, Repeated START Conditions



1) START condition

SCL 신호가 논리 '1'인 동안에 SDA 신호가 논리 '1'에서 '0'으로 천이되는 것.

2) STOP condition

SCL 신호가 논리 '1'인 동안에 SDA 신호가 논리 '0'에서 '1'로 천이되는 것.

3) Repeated START condition

START condition 발생 후에 STOP condition이 발생하지 않고 다시 START condition 발생하는 것.

5. Master Transmitter의 통신 절차 예

① Master Device에서 START condition을 송신하고, 이어서 7비트로 구성된 Slave Device의 주소와 1비트로 구성된 R/W 비트를 송신한다.

② Master Device에서 송출된 번지와 동일한 주소를 갖는 Slave Device는 ACK 신호를 발생시켜 Master Device에게 해당 Slave Device의 존재를 알린다.

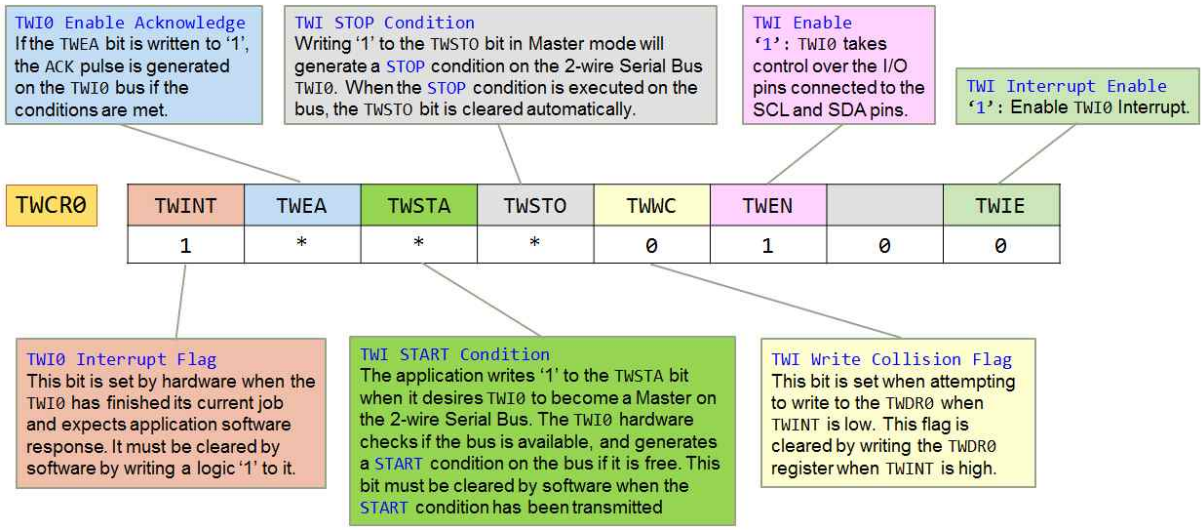
③ Master Device에서 Slave Device로 8비트의 데이터를 보낸다. 이 데이터를 수신한 Slave Device는 ACK 신호를 발생시켜 Master Device에게 해당 데이터의 수신을 알린다.

④ Master Device에서 STOP condition을 송신하여 Slave Device에게 통신 종료를 알린다.

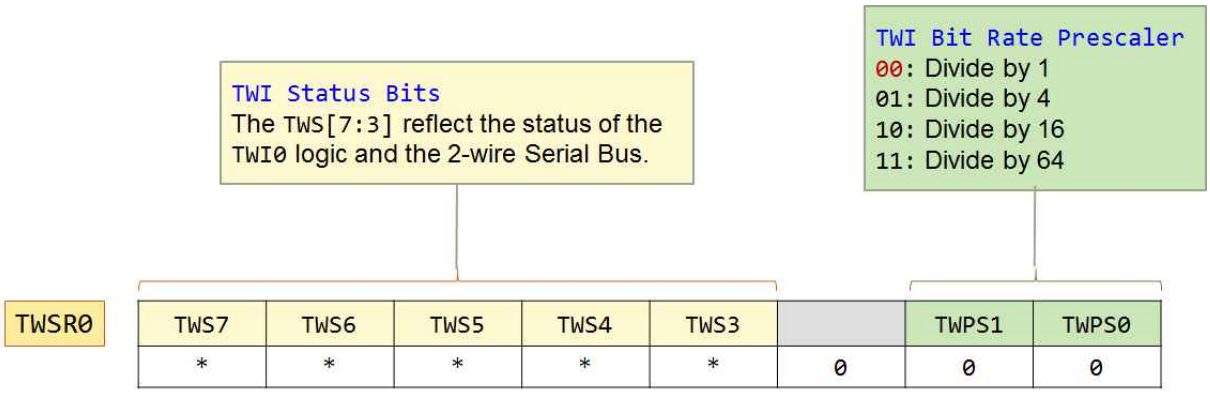
	C Example Code
1	`TWCR0 = (1 << TWINT) | (1 << TWSTA) | (1 << TWEN)`
2	`while (!(TWCR0 & (1<<TWINT)));`
3	`if ((TWSR0 & 0xF8) != START)` `  ERROR();`
	`TWDR0 = SLA_W;` `TWCR0 = (1 << TWINT) | (1 << TWEN);`
4	`while (!(TWCR0 & (1 << TWINT)));`
5	`if ((TWSR0 & 0xF8) != MT_SLA_ACK)` `  ERROR();`
	`TWDR0 = DATA;` `TWCR0 = (1 << TWINT) | (1 << TWEN);`
6	`while (!(TWCR0 & (1 << TWINT)));`
7	`if ((TWSR0 & 0xF8) != MT_DATA_ACK)` `  ERROR();`
	`TWCR0 = (1 << TWINT) | (1 << TWEN) | (1 << TWSTO);`

6. ATmega328PB TWI Register 설명

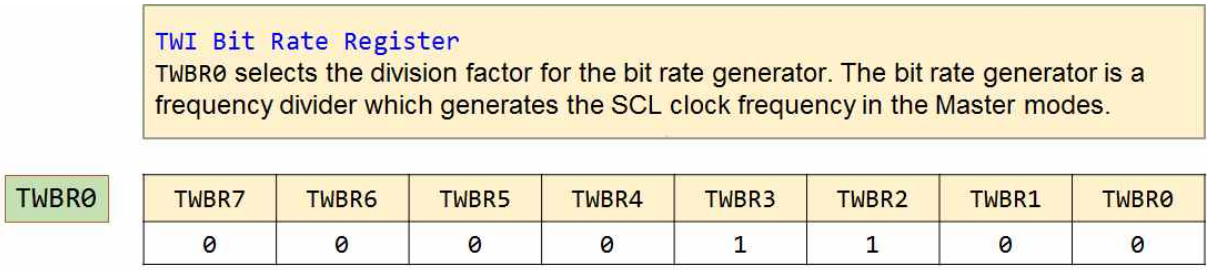
1) TWI Control Register



2) TWI Status Register

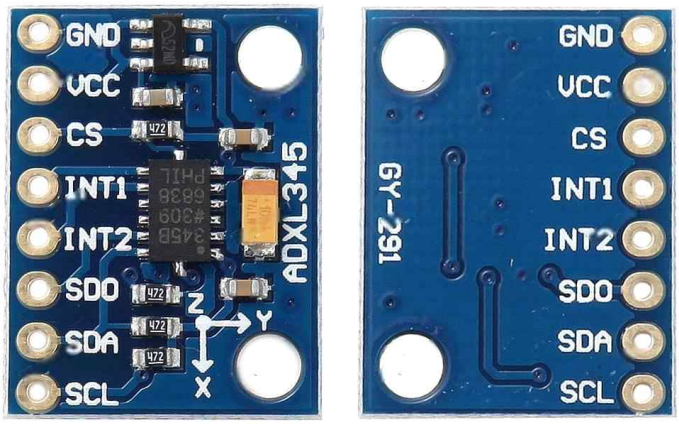


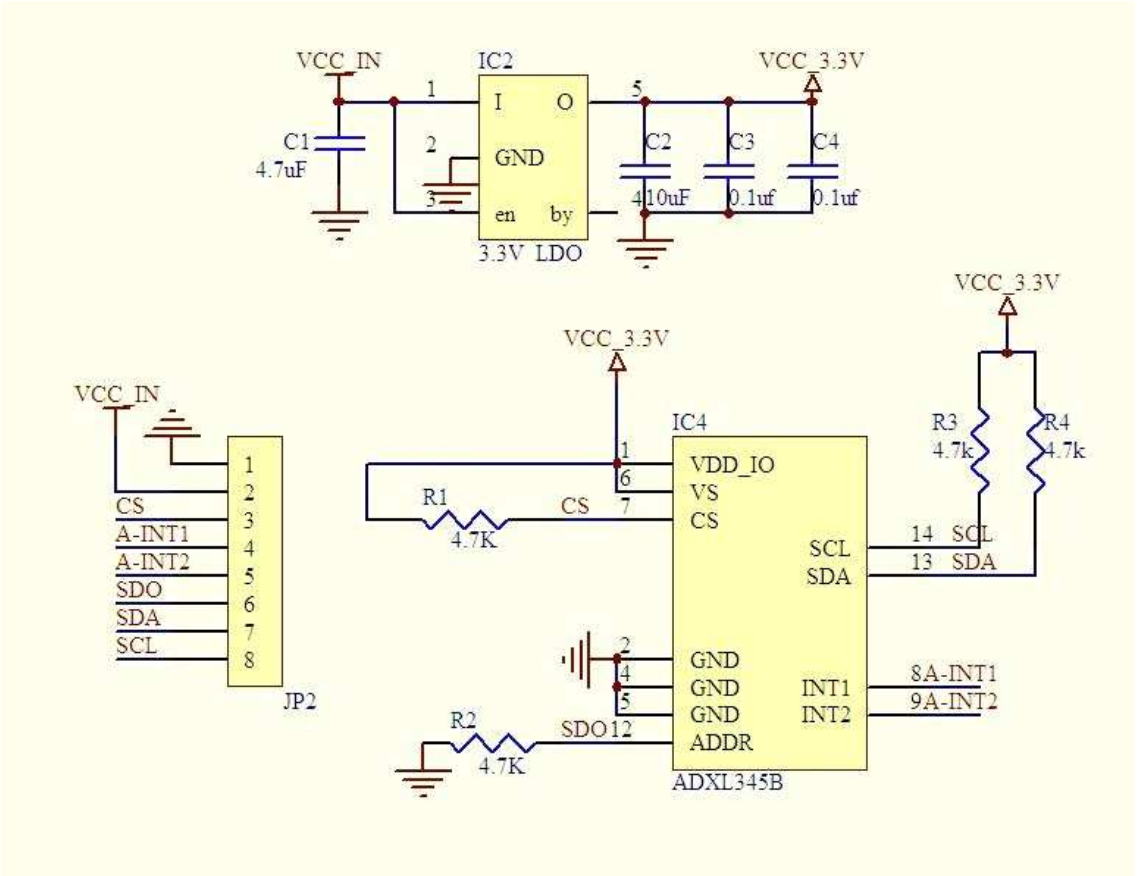
2) TWI Bit Rate Register



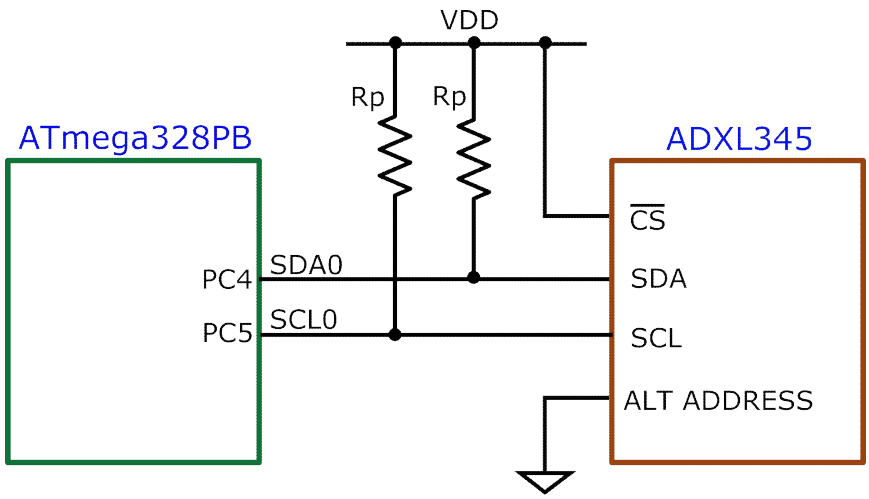
실험내용

GY-291 Board에는 Ananlog Device사의 3축 가속도 센서인 ADXL345가 아래의 그림과 같이 연결되어 있으며, 3축의 가속도 값을 SPI 혹은 IIC 방식을 통해 출력한다.





1. ATmega328PB에 아래와 같이 3축 가속도 센서 모듈인 GY-291 Board를 연결하고, TWI를 사용하여 3축의 가속도 값을 읽어내어 UART0에 연결된 터미널 프로그램의 화면에 출력하는 프로그램을 다음 순서에 따라 작성하시오. TWI0 Clock 주파수는 400kHz로 하고, UART0는 polling 방법을 사용하며 UART0의 baud rate는 38,400 bps로 설정하시오.



가. TWI0와 관련된 다음과 같은 함수를 만들어서 사용하시오.

1) void twi0_init(void)
ATmega328PB TWSR0/TWBR0 레지스터의 초기화

2) void twi0_write_adxl345_reg(*uint8_t* reg_addr, *uint8_t* data)
ADXL345의 해당 레지스터(reg_addr)에 데이터(data)를 출력
TWSR0 Register(SPSR0)의 SPIF0 비트를 읽어 쓰기 동작이 완료된 것을 확인한 후에 리턴.

3) *uint8_t* twi0_read_adxl345_reg(*uint8_t* reg_addr)
ADXL345의 해당 레지스터(reg_addr)로부터 데이터(data)를 읽은 후에 이 값을 반환.

4) void twi0_read_adxl345_reg_multi(*uint8_t* num, *uint8_t* start_addr, *uint8_t* *buff)
ADXL345의 시작 레지스터(start_addr)부터 연속해서 num 번지의 레지스터 값을 읽어 buff로 지정된 메모리에 저장한 후 리턴.
(DATAX0, DATAX1, DATAY0, DATAY1, DATAZ0, DATAZ1의 데이터를 읽는데 사용)

나. H/W 연결 상태 확인

twi0_read_adxl345_reg() 함수를 이용하여 ADXL345 소자의 Device ID를 읽어 USART0으로 출력하고, 그 값을 기록한다.

다. ADXL345 설정

twi0_write_adxl345_reg() 함수를 이용하여 아래에 표에 나타낸 값을 ADXL345 소자의 해당 레지스터에 써 넣는다.

Register Address	Name	Type	Set Value	Description
0x2C	BW_RATE	R/W	00001010	BW=50Hz (Output Data Rate = 100Hz)
0x31	DATA_FORMAT	R/W	00001000	Full Resolution +/-2g, 4mg/LSB
0x2D	POWER_CTL	R/W	00001000	Enter Measurement Mode

라. ADXL345로부터 3축 가속도 값 읽기

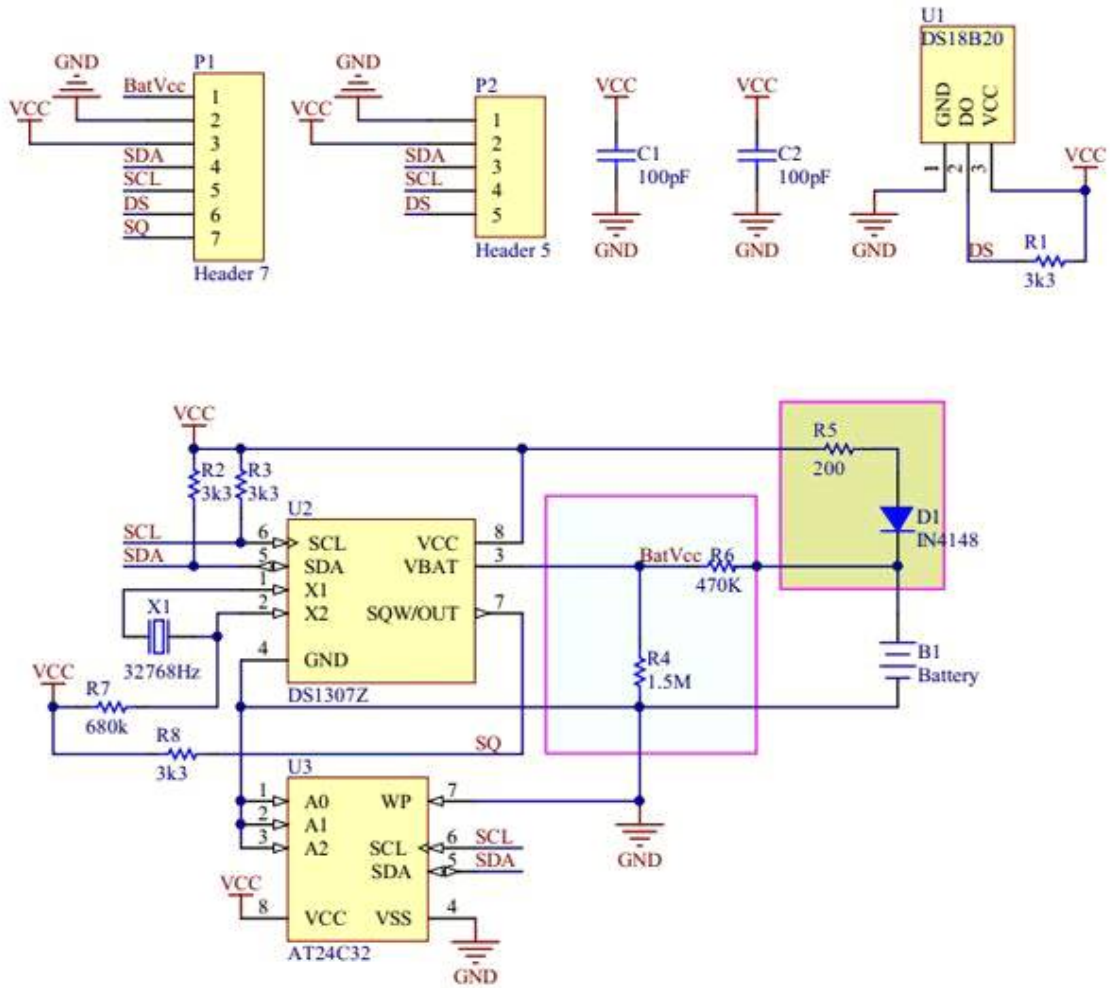
twi0_read_adxl345_reg_multi() 함수를 이용하여 아래에 표에 나타낸 ADXL345 소자의 해당 레지스터의 값을 읽어내어 UART0를 통해 출력한다. 먼저 INT_SOURCE (Address 0x30) 레지스터의 Bit 7을 읽어 새로운 가속도 데이터가 준비된 것을 확인한 후에 3축의 가속도 센서의 출력 데이터(6 바이트)를 차례대로 모두 읽어내어 출력한다.

Register Address	Name	Type	Reset Value	Description
0x32	DATAX0	R	00000000	Low Byte of X-Axis Data
0x33	DATAX1	R	00000000	High Byte of X-Axis Data
0x34	DATAY0	R	00000000	Low Byte of Y-Axis Data
0x35	DATAY1	R	00000000	High Byte of Y-Axis Data
0x36	DATAZ0	R	00000000	Low Byte of Z-Axis Data
0x37	DATAZ1	R	00000000	High Byte of Z-Axis Data

```c
#include <avr/io.h>
#include <stdio.h>

int main(void)
{
    // Declare variables
    uint8_t buff[6];
    int accelX, accelY, accelZ;

    // Init UART0

    // Init TWI0

    // Read ADXL345 ID and display it

    // Set ADXL345 BW_RATE

    // Set ADXL345 DATA_FORMAT (Full Resolution, +/-2g (4mg/LSB))

    // Enter MEASUREMENT mode

    while (1)
    {
        // Wait until new data is available

        // Read 3-axis data with twi0_read_adxl345_reg_multi()

        // Display 3-axis data
    }
}

// Refer to Fig. 37 of ADXL345 Datasheet
void twi0_write_adxl345_reg(uint8_t reg_addr, uint8_t data)
{

}

uint8_t twi0_read_adxl345_reg(uint8_t reg_addr)
{

}

void twi0_read_adxl345_reg_multi_reg(uint8_t num, uint8_t start_addr, uint8_t *buff)
{

}
```

숙제

Tiny RTC Module에는 아래의 그림과 같이 TWI를 이용하여 통신할 수 있는 DS1307 RTC(Real Time Clock) 소자와 AT23C32 EEPROM 소자가 외부 핀으로 연결되어 있다.



1. DS1307 RTC에 대해 조사

2. DS1307 RTC를 이용하여 매 1초 간격으로 시간을 읽어내어 UART0에 연결된 터미널 프로그램의 화면에 현재의 시간을 출력하는 프로그램을 작성하시오. UART0 polling 방법을 사용하고, baud rate는 9,600 bps로 설정하시오. SQW/OUT 출력을 ATmega328PB의 적절한 핀에 연결하여 인터럽트 처리.

3. AT24C32 EEPROM에 대해 조사

4. UART0에 연결된 터미널 프로그램(예: TeraTerm 등)을 이용하여 AT24C32 EEPROM의 특정 번지에 데이터를 써넣고 읽어내는 프로그램을 작성하시오. UART0 polling 방법을 사용하고, baud rate는 9,600 bps로 설정하시오.