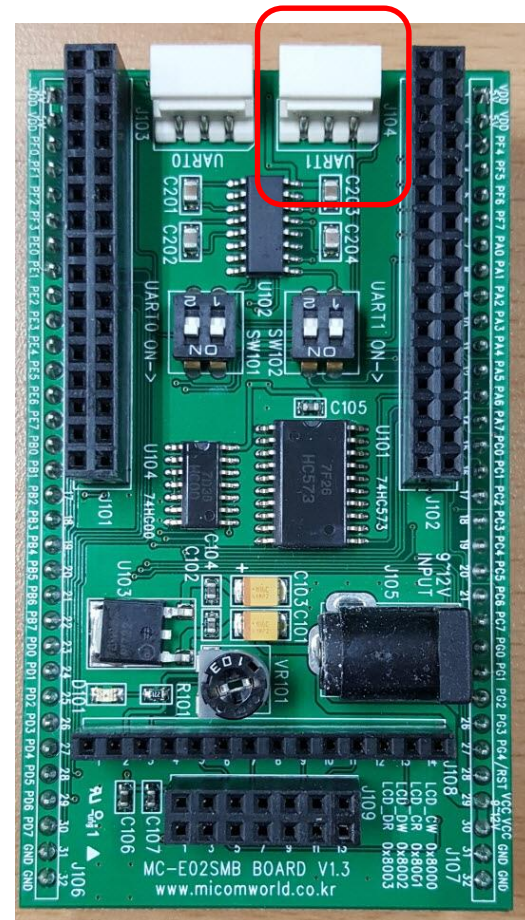


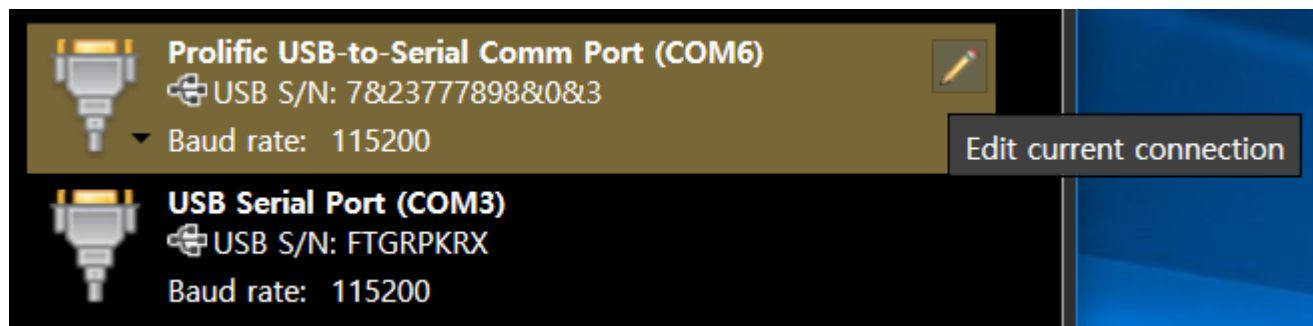
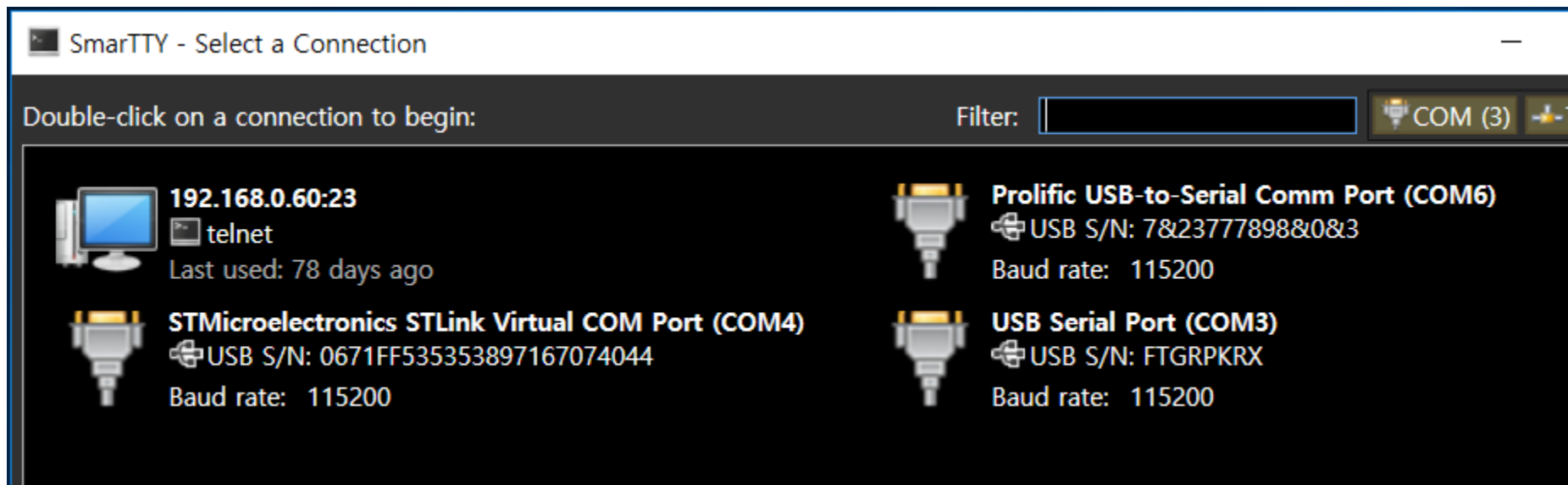
Lab 11. Serial Port

Connection

- Connect serial cable to UART1



Terminal Program



Raw Terminal Settings



COM6 settings

Bits per second: 115200

Data bits: 5 6 7 8

Parity: None Odd Even Mark Space

Stop bits: 0 1 1.5 2

Flow control: None Software (XON/XOFF) Hardware (CTS/RTS) Both

Maximum speed: 7.5 KB/s

▶ Test connection now



Terminal colors

Background color:  Received text color: 

Background color (disconnected):  Echoed characters color: 



Terminal settings

☐ Echo back typed characters (with different color)

Line ending style: CR (␣r) LF (␣n) CRLF (␣r␣n)

☐ Treat incoming LF as CRLF when in text mode

Display mode: Text only Hex only Text and hex

Send data: As you type it When Enter or 'Send' is pressed

Interpret typed data as: Characters Hexadecimal numbers

Maximum bytes per line: 16

Additional line breaks

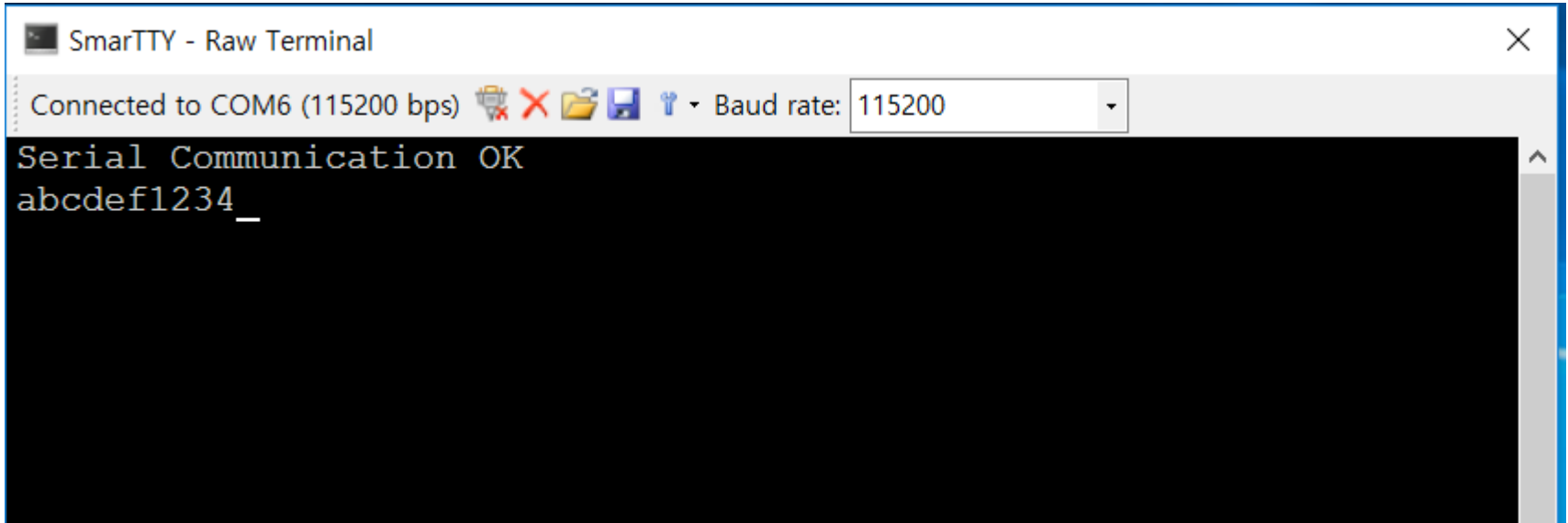
☒ Between echoed and received data

☐ After 0 msec of inactivity

OK

Cancel

Sample Code: serial1.c

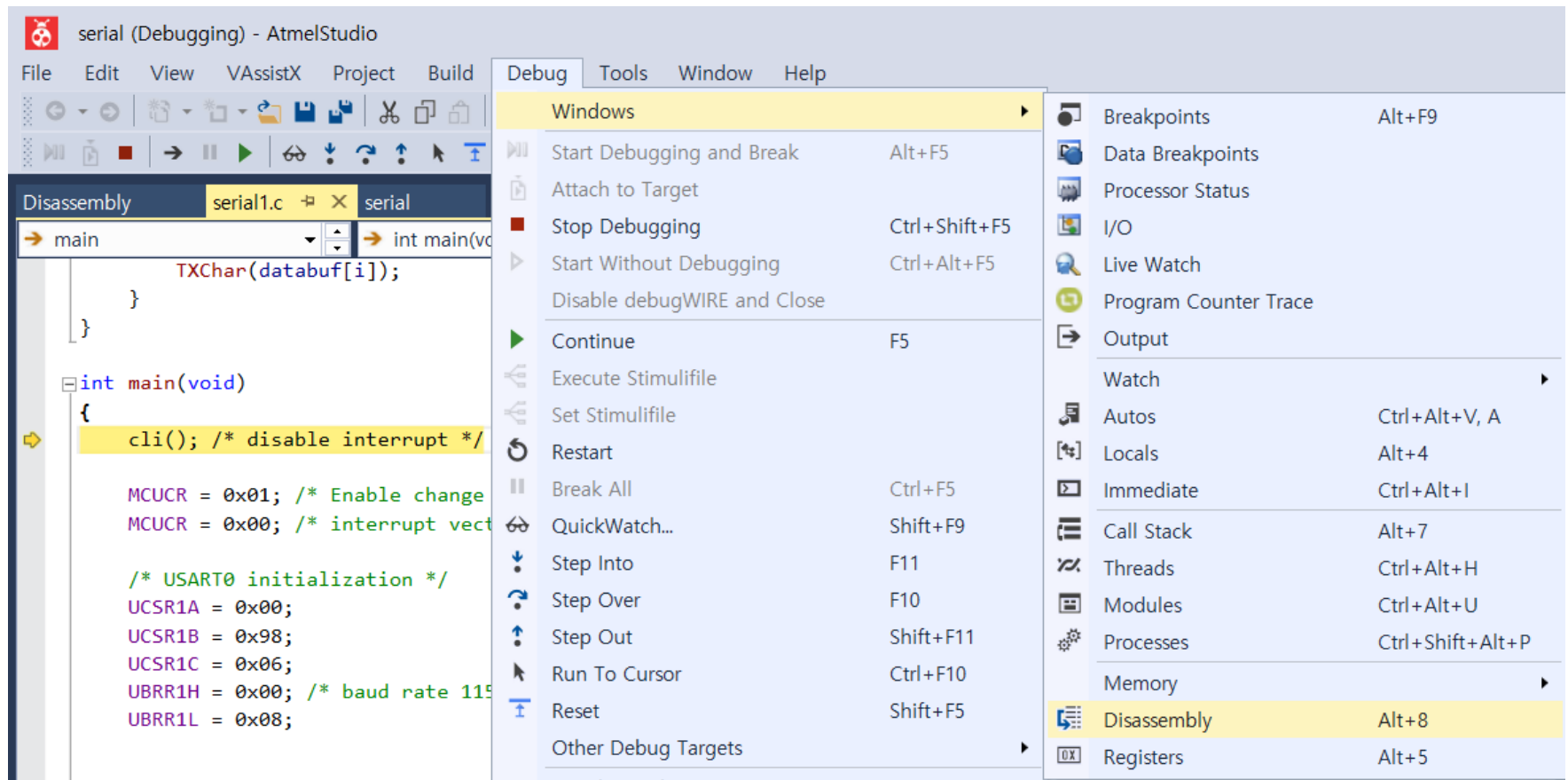


The screenshot shows a terminal window with a title bar that reads "SmarTTY - Raw Terminal". Below the title bar is a status bar indicating the connection: "Connected to COM6 (115200 bps)" followed by several icons (a printer, a red X, a folder, a floppy disk, and a key) and a dropdown menu for "Baud rate:" set to "115200". The main area of the terminal is black with white text. The text displayed is "Serial Communication OK" on the first line and "abcdef1234_" on the second line. A vertical scrollbar is visible on the right side of the terminal window.

```
SmarTTY - Raw Terminal
Connected to COM6 (115200 bps) [printer] [X] [folder] [floppy] [key] Baud rate: 115200
Serial Communication OK
abcdef1234_
```

Debugger & ISR

- USART1, RX complete ISR address: 0x3C
- Start Debugging and Break



ISR Address: 0x3C

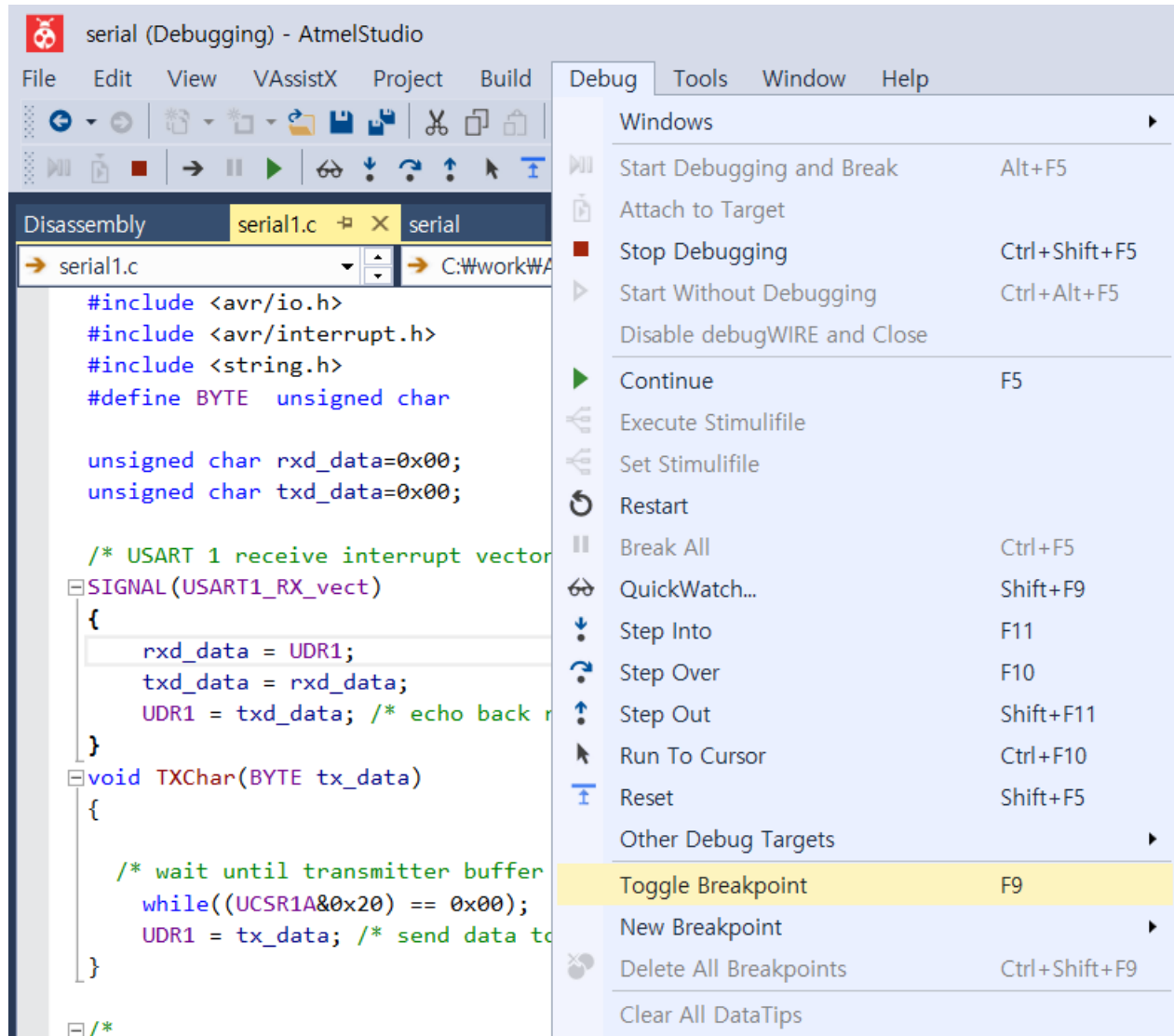
- $0x3C + 0x28 = 0x64$

Disassembly		serial1.c	serial
Address: main			
Viewing Options			
00000037	NOP	No operation	
00000038	RJMP PC+0x002B	Relative jump	
00000039	NOP	No operation	
0000003A	RJMP PC+0x0029	Relative jump	
0000003B	NOP	No operation	
0000003C	RJMP PC+0x0028	Relative jump	
0000003D	NOP	No operation	
0000003E	RJMP PC+0x0025	Relative jump	
0000003F	NOP	No operation	
00000040	RJMP PC+0x0023	Relative jump	
00000041	NOP	No operation	
00000042	RJMP PC+0x0021	Relative jump	
00000043	NOP	No operation	
00000044	RJMP PC+0x001F	Relative jump	
00000045	NOP	No operation	
--- ../../../../crt1/gcrt1.S -----			
00000046	CLR R1	Clear Register	
00000047	OUT 0x3F,R1	Out to I/O location	
00000048	SER R28	Set Register	
00000049	LDI R29,0x10	Load immediate	
0000004A	OUT 0x3E,R29	Out to I/O location	
0000004B	OUT 0x3D,R28	Out to I/O location	

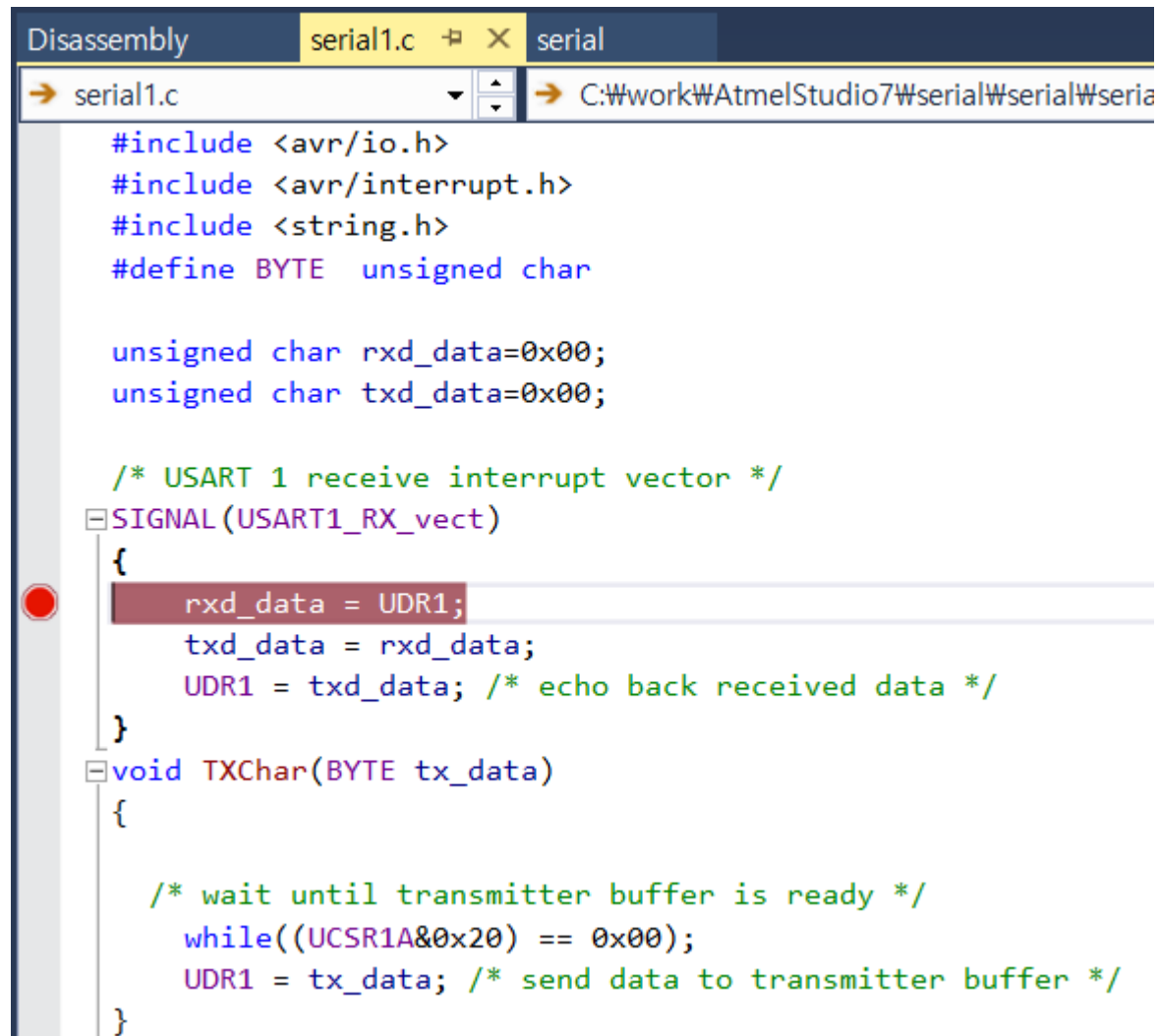
ISR Address

```
Disassembly  serial1.c  serial
Address: main
Viewing Options
--- C:\work\AtmelStudio7\serial\serial\Debug\.././serial1.c -----
{
00000064  PUSH R1          Push register on stack
00000065  PUSH R0          Push register on stack
00000066  IN R0,0x3F       In from I/O location
00000067  PUSH R0          Push register on stack
00000068  CLR R1          Clear Register
00000069  IN R0,0x3B       In from I/O location
0000006A  PUSH R0          Push register on stack
0000006B  PUSH R24         Push register on stack
--- C:\work\AtmelStudio7\serial\serial\Debug\.././serial1.c -----
0000006C  PUSH R30         Push register on stack
0000006D  PUSH R31         Push register on stack
    rxd_data = UDR1;
0000006E  LDI R30,0x9C     Load immediate
0000006F  LDI R31,0x00     Load immediate
00000070  LDD R24,Z+0      Load indirect with displacement
00000071  STS 0x011B,R24   Store direct to data space
    txd_data = rxd_data;
00000073  STS 0x011A,R24   Store direct to data space
    UDR1 = txd_data; /* echo back received data */
00000075  STD Z+0,R24      Store indirect with displacement
}
```


Breakpoint



Breakpoint



The screenshot shows an IDE window titled "Disassembly" with tabs for "serial1.c" and "serial". The "serial1.c" tab is active, showing the following code:

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <string.h>
#define BYTE unsigned char

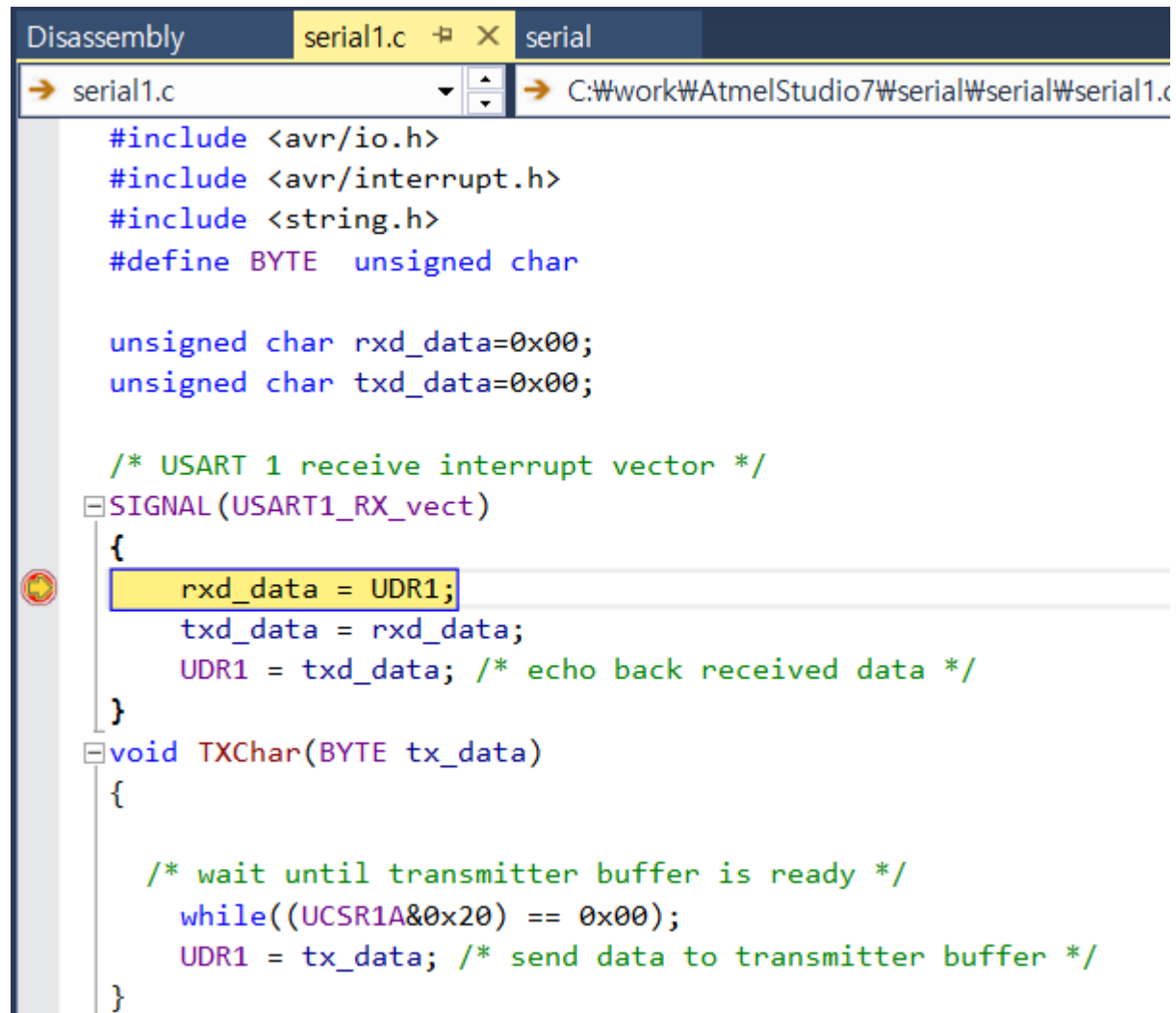
unsigned char rxd_data=0x00;
unsigned char txd_data=0x00;

/* USART 1 receive interrupt vector */
SIGNAL(USART1_RX_vect)
{
    rxd_data = UDR1;
    txd_data = rxd_data;
    UDR1 = txd_data; /* echo back received data */
}

void TXChar(BYTE tx_data)
{
    /* wait until transmitter buffer is ready */
    while((UCSR1A&0x20) == 0x00);
    UDR1 = tx_data; /* send data to transmitter buffer */
}
```

A red circular breakpoint icon is positioned to the left of the line `rxdata = UDR1;` within the `SIGNAL(USART1_RX_vect)` function. The line is highlighted with a light blue background.

Breakpoint



The screenshot shows an IDE window titled "Disassembly" with tabs for "serial1.c" and "serial". The "serial1.c" tab is active, showing the following code:

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <string.h>
#define BYTE unsigned char

unsigned char rxd_data=0x00;
unsigned char txd_data=0x00;

/* USART 1 receive interrupt vector */
SIGNAL(USART1_RX_vect)
{
    rxdata = UDR1;
    txd_data = rxdata;
    UDR1 = txd_data; /* echo back received data */
}

void TXChar(BYTE tx_data)
{
    /* wait until transmitter buffer is ready */
    while((UCSR1A&0x20) == 0x00);
    UDR1 = tx_data; /* send data to transmitter buffer */
}
```

A red circle with a yellow arrow icon is positioned to the left of the line `rxdata = UDR1;`, indicating a breakpoint. The line `rxdata = UDR1;` is highlighted with a yellow background.

Exercise

- 터미널 프로그램(SmarTTY)에서 임의의 **message string**을 입력한 후 **ENTER** 키를 치면, 입력된 **string**이 **LCD**에 나타나서 연속적으로 왼쪽으로 흐르는 프로그램을 작성한다.
- 처음 시작 시 “**Enter a message:**” 메시지를 터미널 프로그램으로 출력한 후, 입력을 기다린다. **String** 입력 시 **echo back** 기능을 사용하여 입력하는 문자가 터미널에서 보이도록 한다.
- **String**의 길이는 편의상 **8~20 character** 라고 가정한다. (경우의 수를 줄여서 쉽게 하기 위함)