Lab2. Debugging, Memory Access, Toggle Key

Bits & Bytes

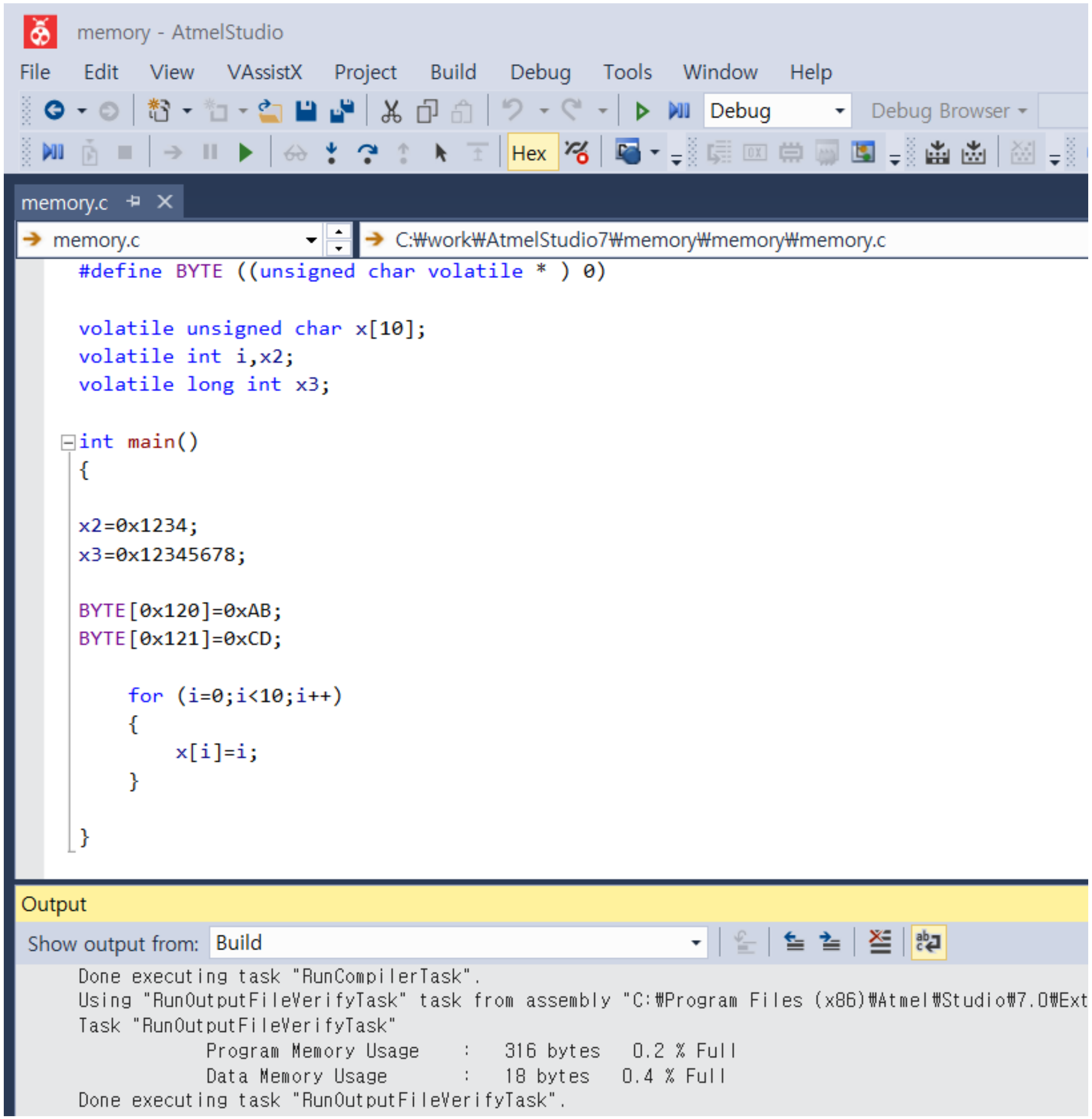
00000000 = 0		1111000 = 248
00000001 = 1		1111001 = 249
00000010 = 2		1111010 = 250
00000011 = 3	(9 thru 247)	1111011 = 251
00000100 = 4		1111100 = 252
00000101 = 5		1111101 = 253
00000110 = 6		1111110 = 254
00000111 = 7		1111111 = 255
00001000 = 8		

00000001 = 0x01 = 1
00000010 = 0x02 = 2
00000100 = 0x04 = 4
00001000 = 0x08 = 8
00010000 = 0x10 = 16
00100000 = 0x20 = 32
01000000 = 0x40 = 64
10000000 = 0x80 = 128

Hexadecimal

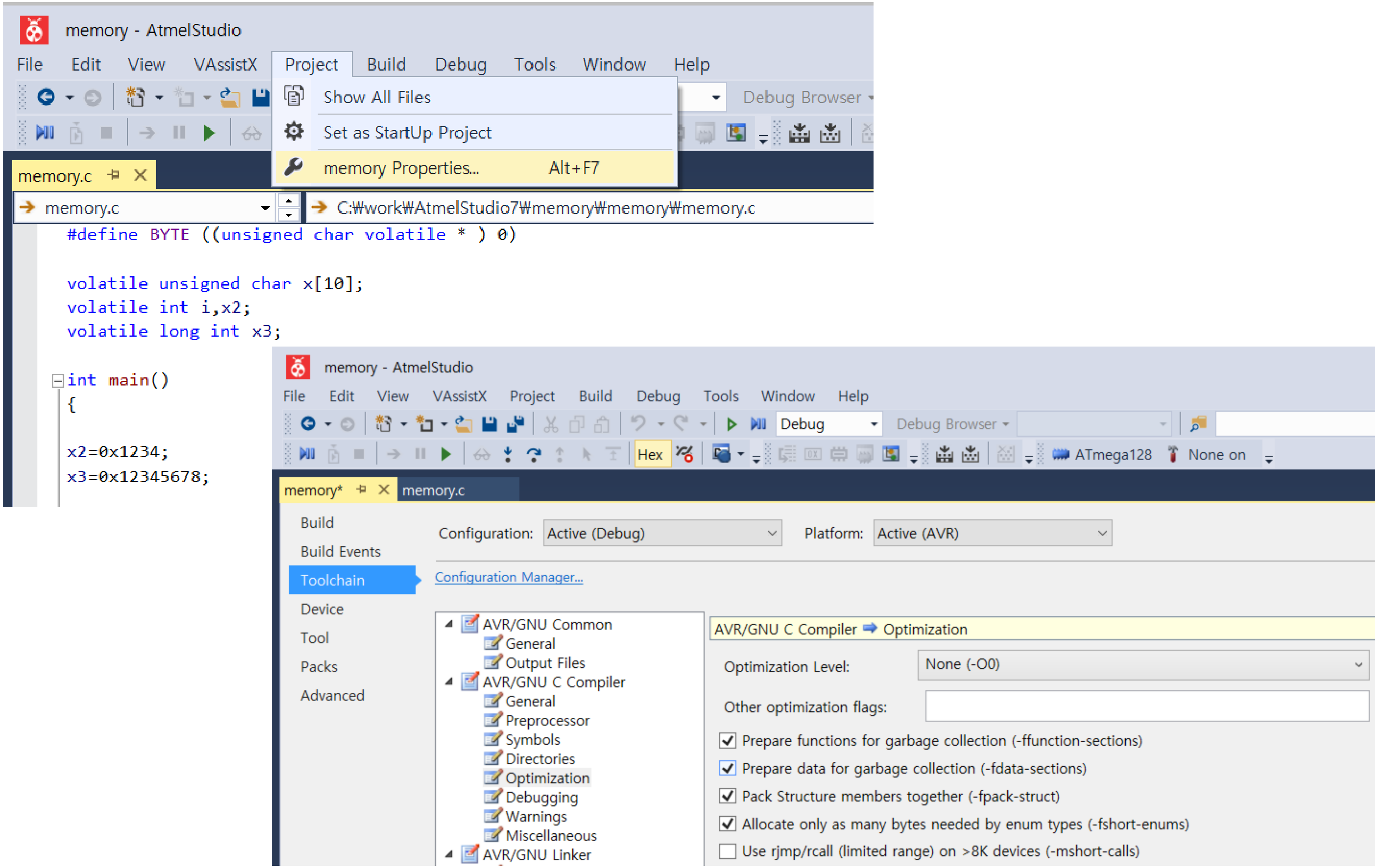
0 = 0000 = 0x0
1 = 0001 = 0x1
2 = 0010 = 0x2
3 = 0011 = 0x3
4 = 0100 = 0x4
5 = 0101 = 0x5
6 = 0110 = 0x6
7 = 0111 = 0x7
8 = 1000 = 0x8
9 = 1001 = 0x9
10 = 1010 = 0xA
11 = 1011= 0xB
12 = 1100 = 0xC
13 = 1101 = 0xD
14 = 1110 = 0xE
15 = 1111 = 0xF

memory.c



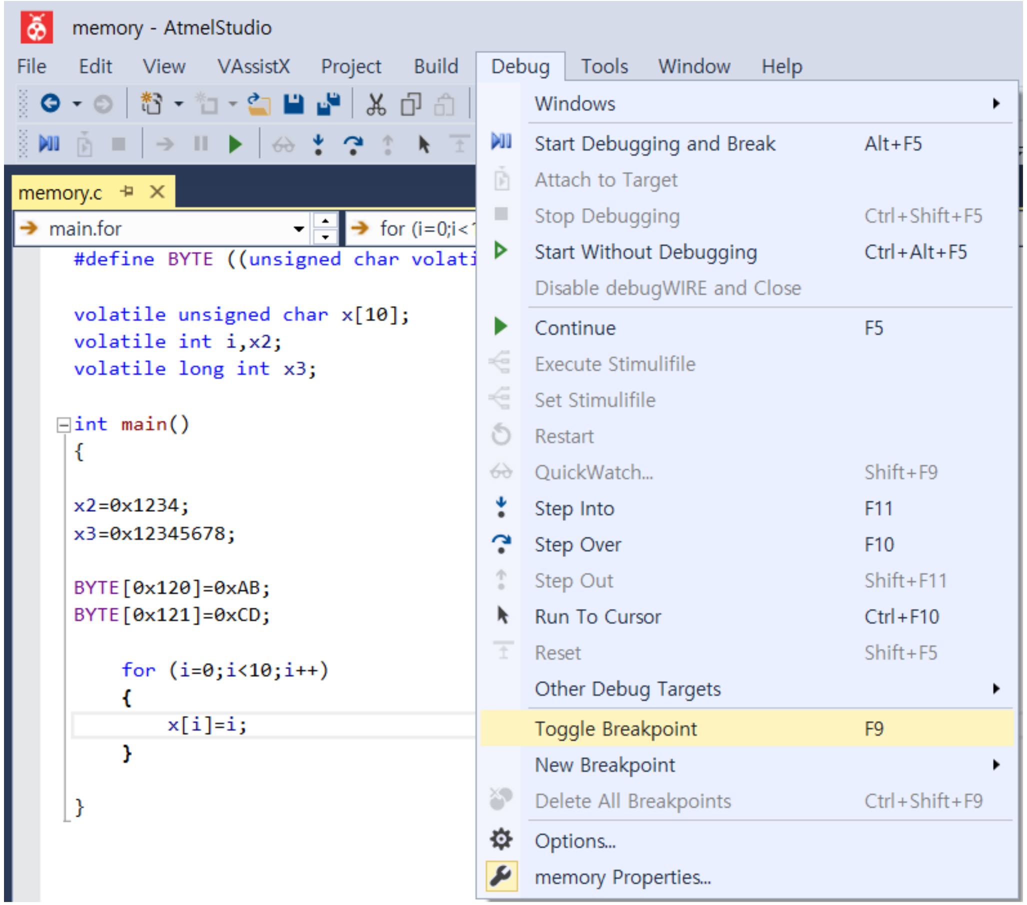
memory - AtmelStudio
File Edit View VAssistX Project Build Debug Tools Window Help
Debug
Debug Browser
Hex
memory.c
memory.c
C:₩work₩AtmelStudio7₩memory₩memory₩memory.c
#define BYTE ((unsigned char volatile *) 0)
volatile unsigned char x[10];
volatile int i,x2;
volatile long int x3;
int main()
{
x2=0x1234;
x3=0x12345678;
BYTE[0x120]=0xAB;
BYTE[0x121]=0xCD;
for (i=0;i<10;i++)
{
x[i]=i;
}
}
Output
Show output from: Build
Done executing task "RunCompilerTask".
Using "RunOutputFileVerifyTask" task from assembly "C:₩Program Files (x86)₩Atmel₩Studio₩7.0₩Ext
Task "RunOutputFileVerifyTask"
Program Memory Usage : 316 bytes 0.2 % Full
Data Memory Usage : 18 bytes 0.4 % Full
Done executing task "RunOutputFileVerifyTask".

Optimization & volatile



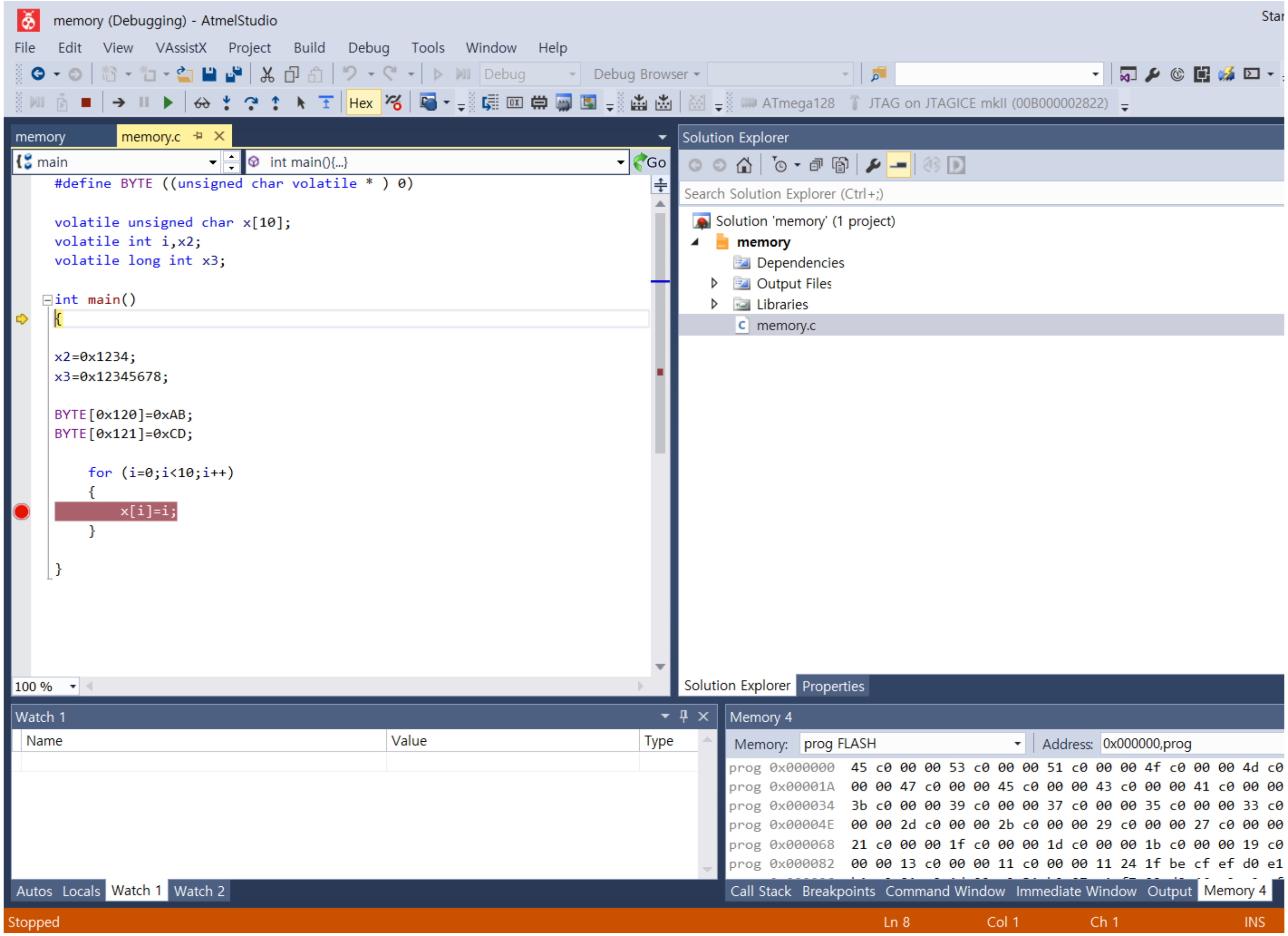
Toggle Breakpoint

- Position the cursor to the desired position and toggle breakpoint.



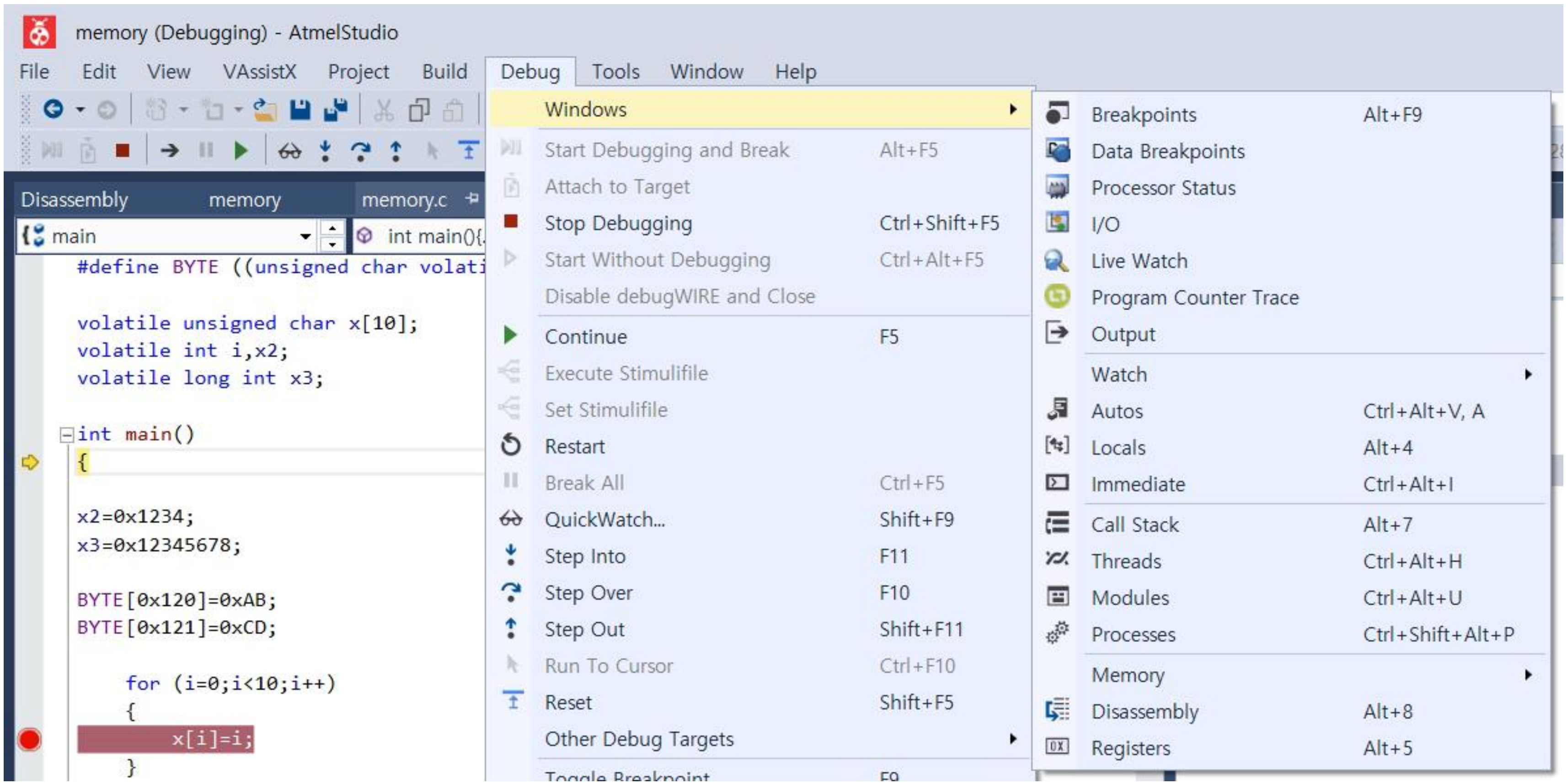
Breakpoint

- Start Debugging and Break

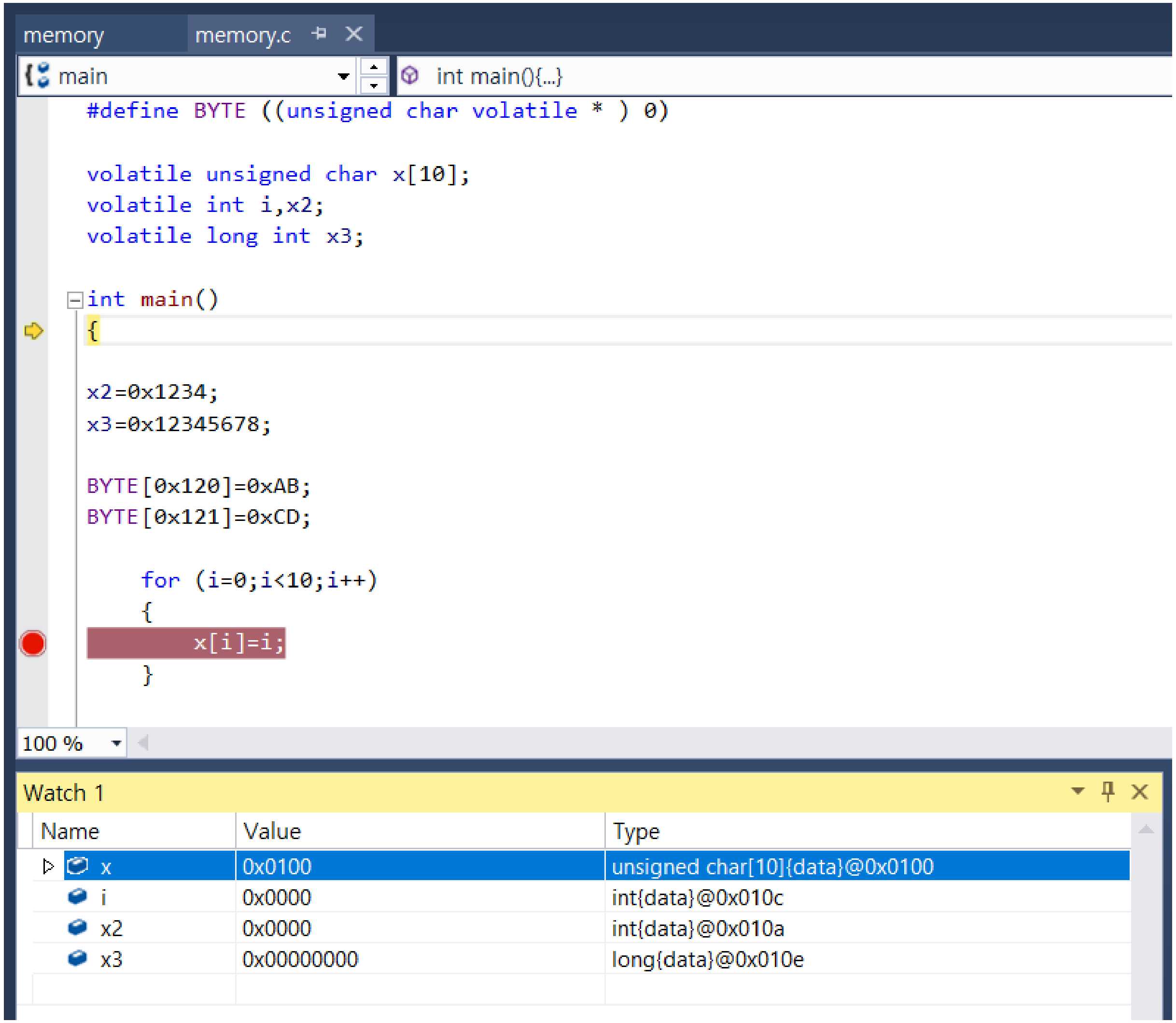


Watch & Memory Window

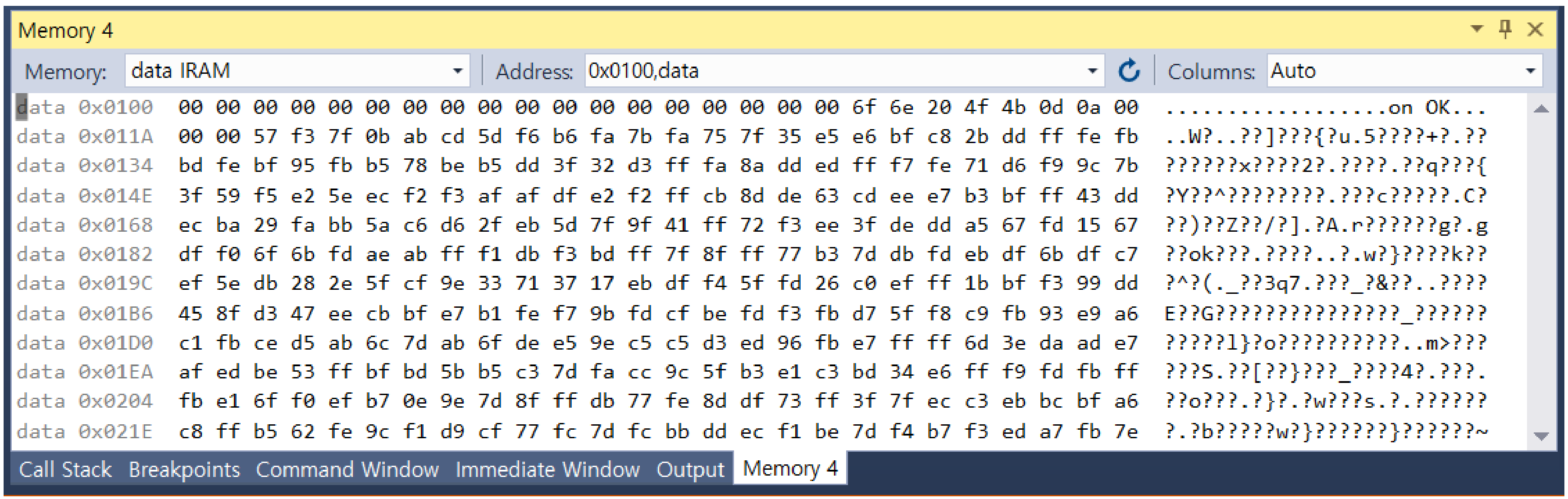
- Select Watch & Memory window

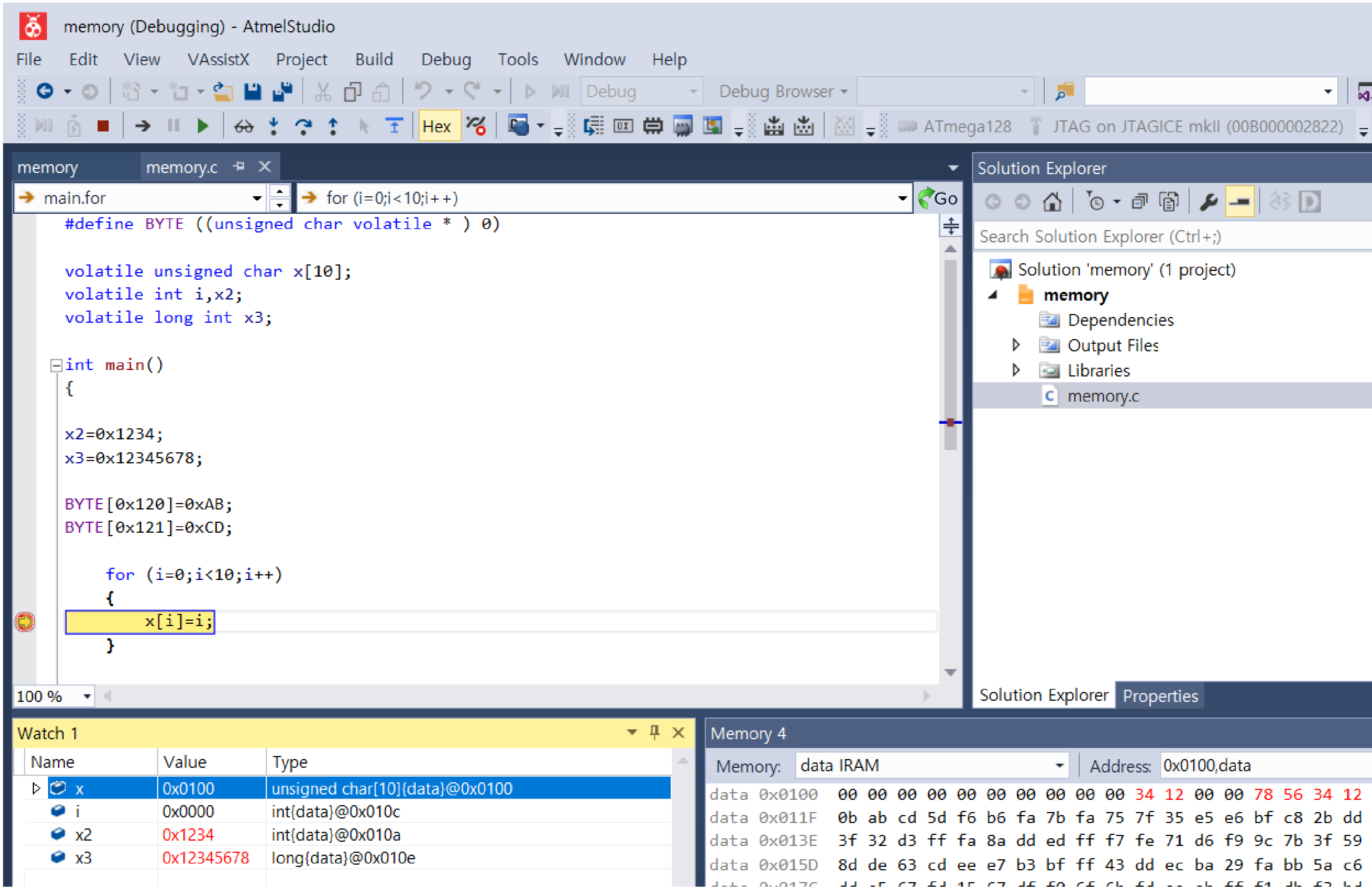


Watch variables

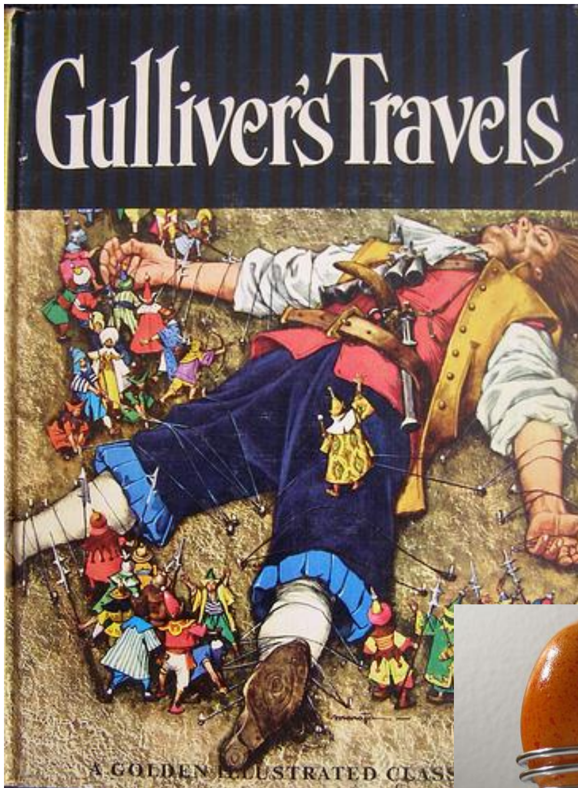


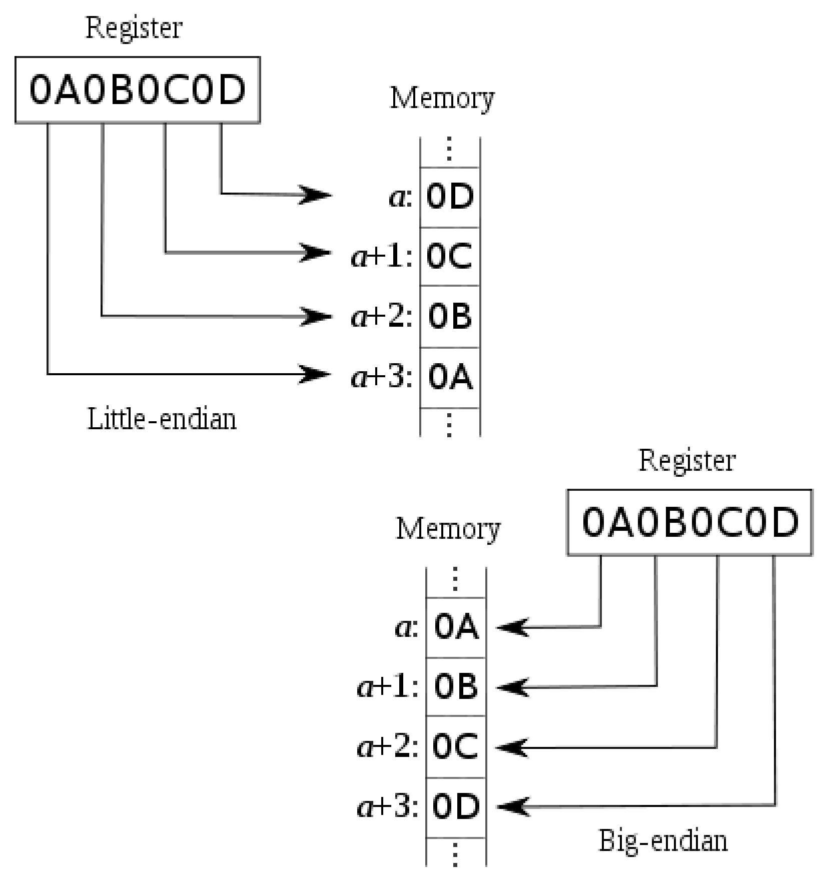
Memory

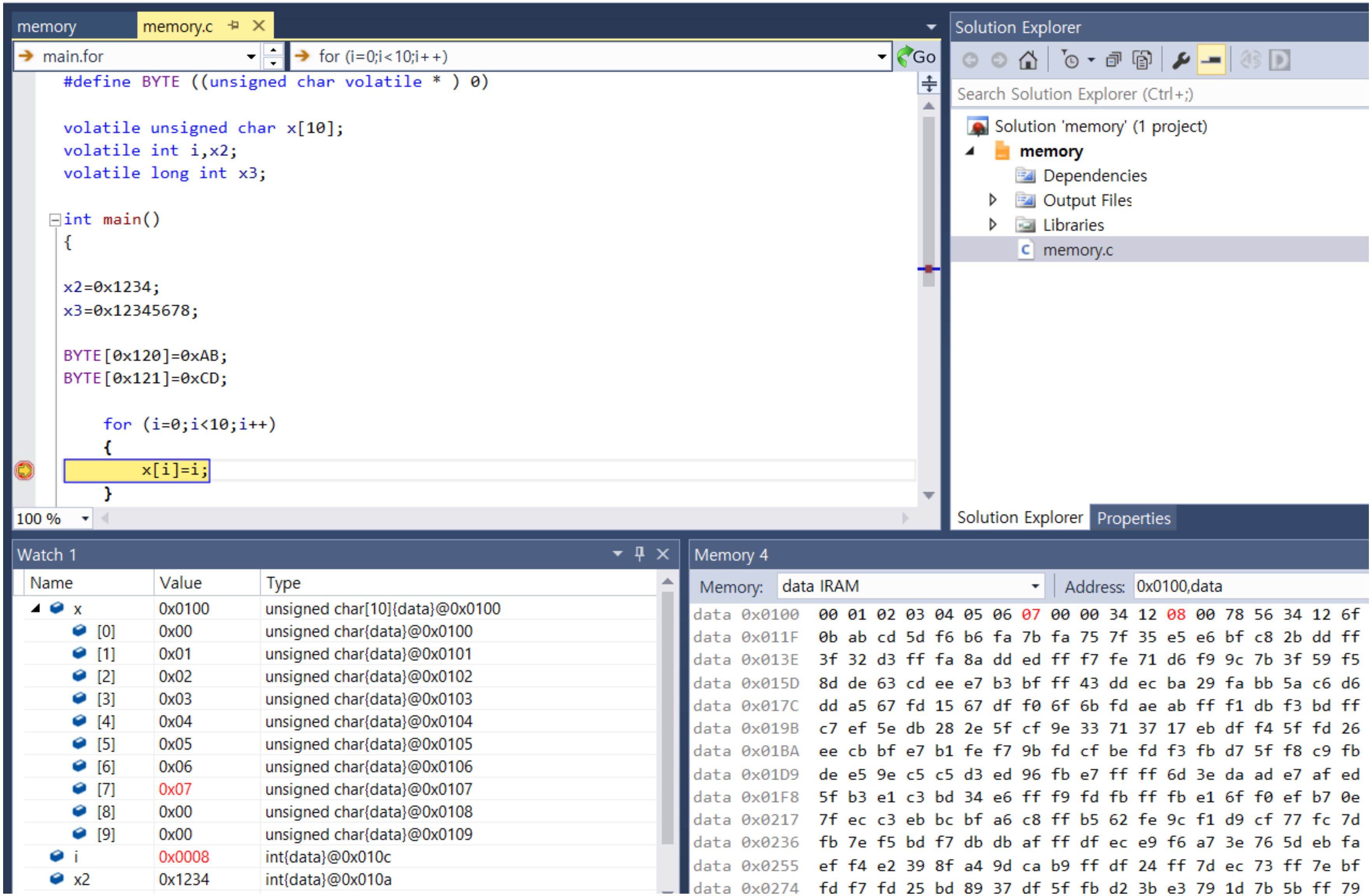

Memory 4
Memory: data IRAM
Address: 0x0100,data
Columns: Auto
data 0x0100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 6f 6e 20 4f 4b 0d 0a 00 ....on OK...
data 0x011A 00 00 57 f3 7f 0b ab cd 5d f6 b6 fa 7b fa 75 7f 35 e5 e6 bf c8 2b dd ff fe fb ..W?..??]???{?u.5????+?.??
data 0x0134 bd fe bf 95 fb b5 78 be b5 dd 3f 32 d3 ff fa 8a dd ed ff f7 fe 71 d6 f9 9c 7b ??????x????2?.????.??q???{
data 0x014E 3f 59 f5 e2 5e ec f2 f3 af af df e2 f2 ff cb 8d de 63 cd ee e7 b3 bf ff 43 dd ?Y??^????????.???c?????.C?
data 0x0168 ec ba 29 fa bb 5a c6 d6 2f eb 5d 7f 9f 41 ff 72 f3 ee 3f de dd a5 67 fd 15 67 ??)??Z??/?].?A.r??????g?.g
data 0x0182 df f0 6f 6b fd ae ab ff f1 db f3 bd ff 7f 8f ff 77 b3 7d db fd eb df 6b df c7 ??ok???.????..?.w?}????k??
data 0x019C ef 5e db 28 2e 5f cf 9e 33 71 37 17 eb df f4 5f fd 26 c0 ef ff 1b bf f3 99 dd ?^?(._??3q7.???_?&??..????
data 0x01B6 45 8f d3 47 ee cb bf e7 b1 fe f7 9b fd cf be fd f3 fb d7 5f f8 c9 fb 93 e9 a6 E??G???????????????_??????
data 0x01D0 c1 fb ce d5 ab 6c 7d ab 6f de e5 9e c5 c5 d3 ed 96 fb e7 ff ff 6d 3e da ad e7 ?????l}?o??????????..m>???
data 0x01EA af ed be 53 ff bf bd 5b b5 c3 7d fa cc 9c 5f b3 e1 c3 bd 34 e6 ff f9 fd fb ff ???S.??[??}???_????4?.???.
data 0x0204 fb e1 6f f0 ef b7 0e 9e 7d 8f ff db 77 fe 8d df 73 ff 3f 7f ec c3 eb bc bf a6 ??o???.?}?.?w???s.?.??????
data 0x021E c8 ff b5 62 fe 9c f1 d9 cf 77 fc 7d fc bb dd ec f1 be 7d f4 b7 f3 ed a7 fb 7e ?.?b?????w?}??????}??????~
Call Stack
Breakpoints
Command Window
Immediate Window
Output
Memory 4


memory (Debugging) - AtmelStudio
File Edit View VAssistX Project Build Debug Tools Window Help
Debug
Debug Browser
Hex
ATmega128
JTAG on JTAGICE mkII (00B000002822)
memory
memory.c
main.for
for (i=0;i<10;i++)
Go
#define BYTE ((unsigned char volatile *) 0)
volatile unsigned char x[10];
volatile int i,x2;
volatile long int x3;
int main()
{
x2=0x1234;
x3=0x12345678;
BYTE[0x120]=0xAB;
BYTE[0x121]=0xCD;
for (i=0;i<10;i++)
{
x[i]=i;
}
100 %
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'memory' (1 project)
memory
Dependencies
Output Files
Libraries
memory.c
Solution Explorer
Properties
Watch 1
Name Value Type
x 0x0100 unsigned char[10]{data}@0x0100
i 0x0000 int{data}@0x010c
x2 0x1234 int{data}@0x010a
x3 0x12345678 long{data}@0x010e
Memory 4
Memory: data IRAM
Address: 0x0100,data
data 0x0100 00 00 00 00 00 00 00 00 00 00 34 12 00 00 78 56 34 12
data 0x011F 0b ab cd 5d f6 b6 fa 7b fa 75 7f 35 e5 e6 bf c8 2b dd
data 0x013E 3f 32 d3 ff fa 8a dd ed ff f7 fe 71 d6 f9 9c 7b 3f 59
data 0x015D 8d de 63 cd ee e7 b3 bf ff 43 dd ec ba 29 fa bb 5a c6

Little Endian & Big Endian






memory
memory.c
main.for
for (i=0;i<10;i++)
Go
#define BYTE ((unsigned char volatile *) 0)
volatile unsigned char x[10];
volatile int i,x2;
volatile long int x3;
int main()
{
x2=0x1234;
x3=0x12345678;
BYTE[0x120]=0xAB;
BYTE[0x121]=0xCD;
for (i=0;i<10;i++)
{
x[i]=i;
}
100 %
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'memory' (1 project)
memory
Dependencies
Output Files
Libraries
memory.c
Solution Explorer
Properties
Watch 1
Name Value Type
x 0x0100 unsigned char[10]{data}@0x0100
[0] 0x00 unsigned char{data}@0x0100
[1] 0x01 unsigned char{data}@0x0101
[2] 0x02 unsigned char{data}@0x0102
[3] 0x03 unsigned char{data}@0x0103
[4] 0x04 unsigned char{data}@0x0104
[5] 0x05 unsigned char{data}@0x0105
[6] 0x06 unsigned char{data}@0x0106
[7] 0x07 unsigned char{data}@0x0107
[8] 0x00 unsigned char{data}@0x0108
[9] 0x00 unsigned char{data}@0x0109
i 0x0008 int{data}@0x010c
x2 0x1234 int{data}@0x010a
Memory 4
Memory: data IRAM
Address: 0x0100,data
data 0x0100 00 01 02 03 04 05 06 07 00 00 34 12 08 00 78 56 34 12 6f
data 0x011F 0b ab cd 5d f6 b6 fa 7b fa 75 7f 35 e5 e6 bf c8 2b dd ff
data 0x013E 3f 32 d3 ff fa 8a dd ed ff f7 fe 71 d6 f9 9c 7b 3f 59 f5
data 0x015D 8d de 63 cd ee e7 b3 bf ff 43 dd ec ba 29 fa bb 5a c6 d6
data 0x017C dd a5 67 fd 15 67 df f0 6f 6b fd ae ab ff f1 db f3 bd ff
data 0x019B c7 ef 5e db 28 2e 5f cf 9e 33 71 37 17 eb df f4 5f fd 26
data 0x01BA ee cb bf e7 b1 fe f7 9b fd cf be fd f3 fb d7 5f f8 c9 fb
data 0x01D9 de e5 9e c5 c5 d3 ed 96 fb e7 ff ff 6d 3e da ad e7 af ed
data 0x01F8 5f b3 e1 c3 bd 34 e6 ff f9 fd fb ff fb e1 6f f0 ef b7 0e
data 0x0217 7f ec c3 eb bc bf a6 c8 ff b5 62 fe 9c f1 d9 cf 77 fc 7d
data 0x0236 fb 7e f5 bd f7 db db af ff df ec e9 f6 a7 3e 76 5d eb fa
data 0x0255 ef f4 e2 39 8f a4 9d ca b9 ff df 24 ff 7d ec 73 ff 7e bf
data 0x0274 fd f7 fd 25 bd 89 37 df 5f fb d2 3b e3 79 1d 7b 5b ff 79

Exercise 1

- 앞의 memory.c 에서 x 변수의 정의를 아래와 같이 변경하고, 앞 페이지와 동일하게 i=8 에서 디버거가 정지한 화면을 캡쳐 하시오. 그리고 앞 페이지의 그림과 무엇이 다른지를 파악한 후 이유를 설명하시오.

volatile unsigned int x[10];

Exercise 2: Pointers Example

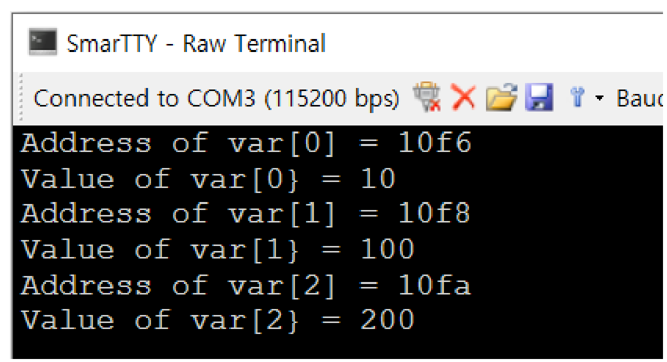
```c
#include <avr/io.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

void uart_putchar(uint8_t u8Data, FILE *stream )
{
    while(!(UCSR1A&(1<<UDRE1))){};
    UDR1 = u8Data;
}
FILE uart_output = FDEV_SETUP_STREAM((void *)uart_putchar, NULL, _FDEV_SETUP_WRITE);

const int MAX = 3;
int main(void)
{
    int var[] = {10, 100, 200};
    int i, *ptr;

    /* USART1 initialization */
    UCSR1A = 0x00;
    UCSR1B = 0x98;
    UCSR1C = 0x06;
    UBRR1H = 0x00; /* baud rate 115200 UBRR1=8*/
    UBRR1L = 0x08;
    stdout = &uart_output;

    ptr = var;
    for (i=0;i < MAX; i++)
    {
        printf("Address of var[%d] = %x\n\r", i, ptr);
        printf("Value of var[%d} = %d\n\r", i, *ptr);
        ptr++;
    }
while(1);
}
```



```c
    ptr = var;
    for (i=0;i < MAX; i++)
    {
        printf("Address of var[%d] = %x\n\r", i, ptr);
        printf("Value of var[%d} = %d\n\r", i, *ptr);
        ptr++;
    }
while(1);
}
```

91 %

Watch 1

Name	Value	Type
var	0x10f6	int[3]{data}@0x10f6 ([R28]+5)
[0]	0x000a	int{data}@0x10f6
[1]	0x0064	int{data}@0x10f8
[2]	0x00c8	int{data}@0x10fa

Memory 4

Memory: data IRAM   Address: 0x10F6,data

```
data 0x10F6  0a 00 64 00 c8 00 10 ff 00 62 62 62 62 62 62
data 0x110A  62 62 62 62 62 62 62 62 62 62 62 62 62 62 62
data 0x111E  62 62 62 62 62 62 62 62 62 62 62 62 62 62 62
data 0x1132  62 62 62 62 62 62 62 62 62 62 62 62 62 62 62
```

Exercise 3: 아래의 예제를 실행

```c
#include <stdio.h>
#include <string.h>

int main ()
{
   char str1[12] = "Hello";
   char str2[12] = "World";
   char str3[12];
   int  len ;

   /* copy str1 into str3 */
   strcpy(str3, str1);
   printf("strcpy( str3, str1) :  %s\n", str3 );

   /* concatenates str1 and str2 */
   strcat( str1, str2);
   printf("strcat( str1, str2):   %s\n", str1 );
```

동일한 결과를 확인

```
    /* total lenghth of str1 after concatenation */
    len = strlen(str1);
    printf("strlen(str1) :  %d\n", len );

    return 0;
}
```

When the above code is compiled and executed, it produces the following result:

```
strcpy( str3, str1) :  Hello
strcat( str1, str2):   HelloWorld
strlen(str1) :  10
```

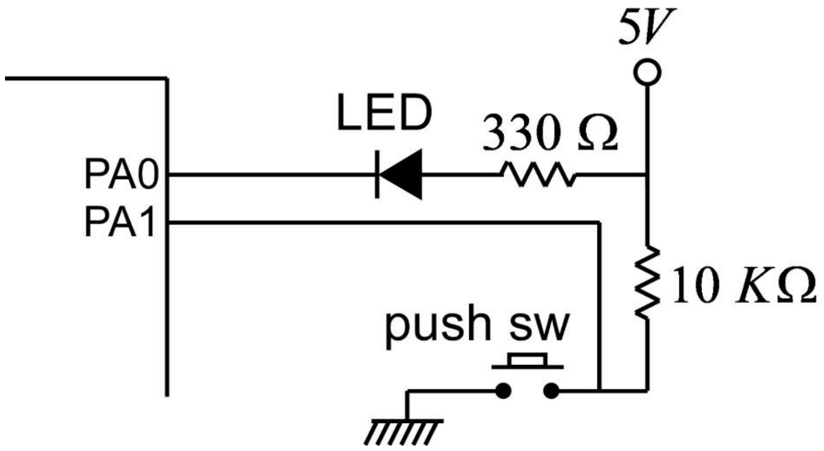
Toggle Key

- 예제 프로그램 Key_toggle2.c 는 키를 한번 누를 때 마다 led를 켜거나 끄는 동작을 한다. 이와 같은 동작을 토글(toggle) 동작이라고 하며, 전자 제품의 전원 스위치 등에 많이 사용된다. 즉, 스위치를 한번 누르면 켜지고, 다시 누르면 꺼지는 동작이다. 이 프로그램을 실행해서 토글 동작이 정상 작동하는지 확인한다.

Review: led_key.c

```
#include <avr/io.h>
int main(void)
{
        DDRA = 0x0f;

        while(1)
        {
                if (PINA & 0x02)
                {
                        PORTA = 0x0f;
                }
                else
                {
                        PORTA = 0x0;
                }
        }
}
```



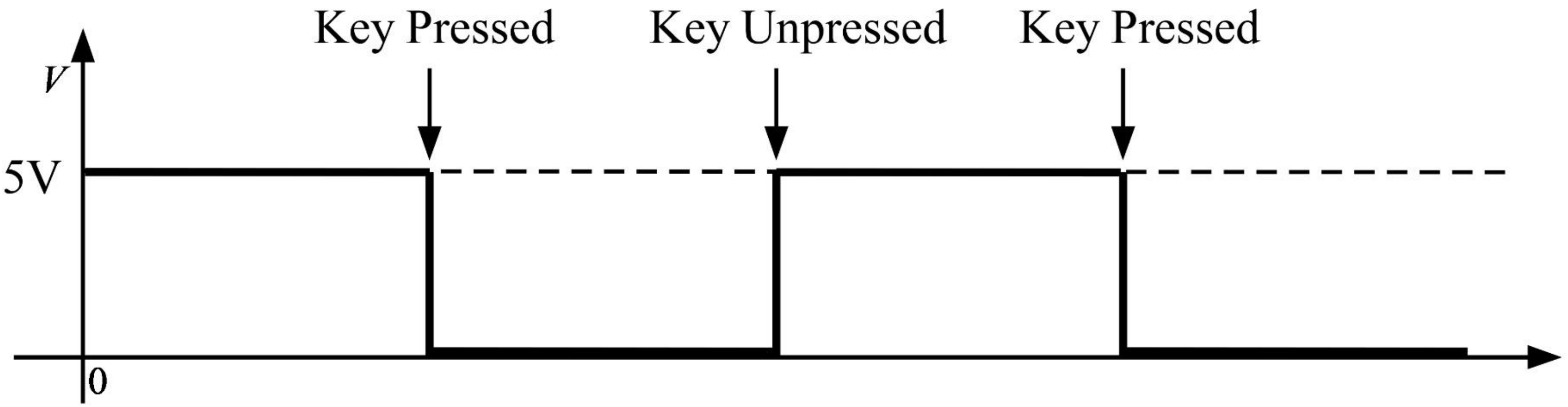
- DDRA: data direction register A

  0: input(default), 1: output
- PINA: input Port A
- PORTA: output Port A

PINA	**PA7**	**PA6**	**PA5**	**PA4**	**PA3**	**PA2**	**PA1**	**PA0**
0x02	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**
PINA & 0x10	**0**	**0**	**0**	**0**	**0**	**0**	**PA1**	**0**

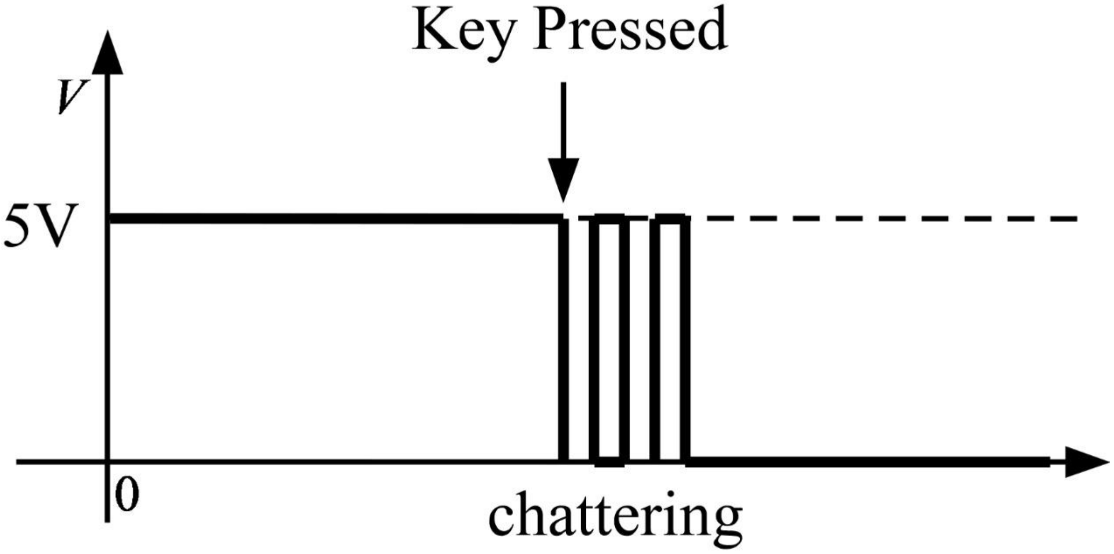
- When the key is pressed, PA1=0

Toggle Action



- if (Key_Pressed == TRUE) ?

Chattering



Exercise 4

- 주어진 Key_toggle2.c 프로그램을 수정하여 브레드 보드에 꽂힌 led 대신에 CPU 보드의 CPU 옆에 부착된 4개의 led 중 가장 우측의 led (Lab1에서 led.c 프로그램에서 사용되었던 led)가 꺼지거나 켜지도록 한다.

Exercise 5

- 주어진 Key_toggle2.c 프로그램을 아래의 조건에 맞도록 변형하여 동작을 확인한다.
- 키 스위치를 한번 씩 누를 때 마다 변수 state 는 0,1,2,3,0,1,2,3,0,... 과 같이 계속 변한다. 변수 state의 최초 값은 0이다.
- 변수 state 값이 0 일 때는 가장 우측의 led가 켜지고, 변수 state 값이 1 일 때는 가장 우측에서 2번째 led가 켜지고, 변수 state 값이 2 일 때는 3번째 led가 켜지고, 변수 state 값이 3 일 때는 4번째 led가 켜진다.