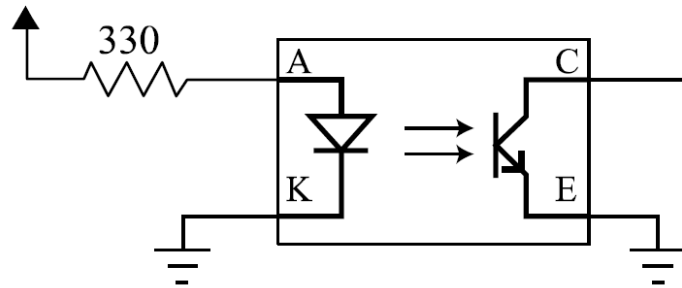

Lab6. Interrupt

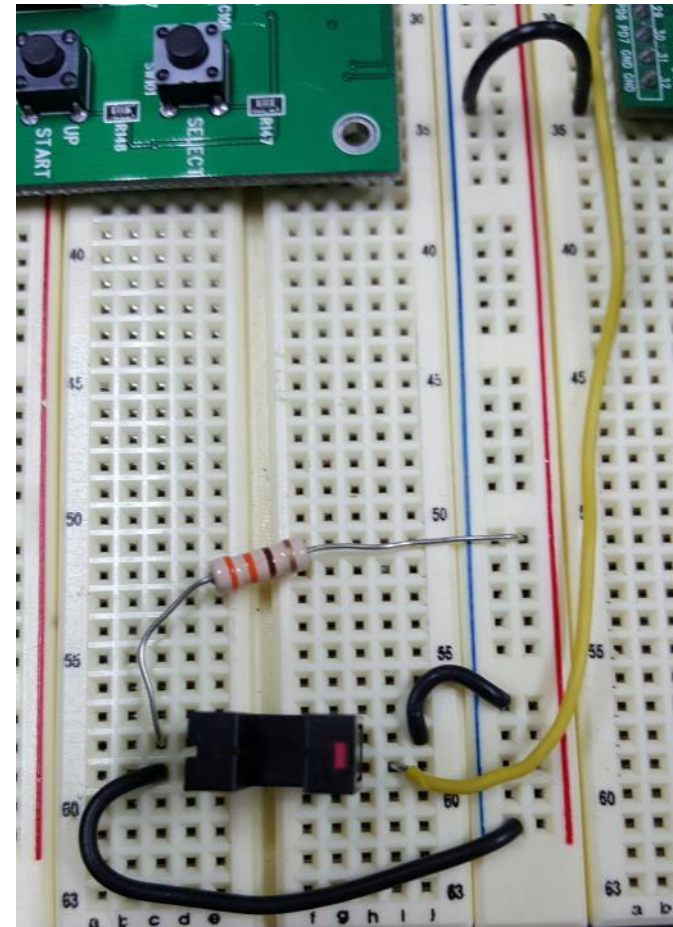
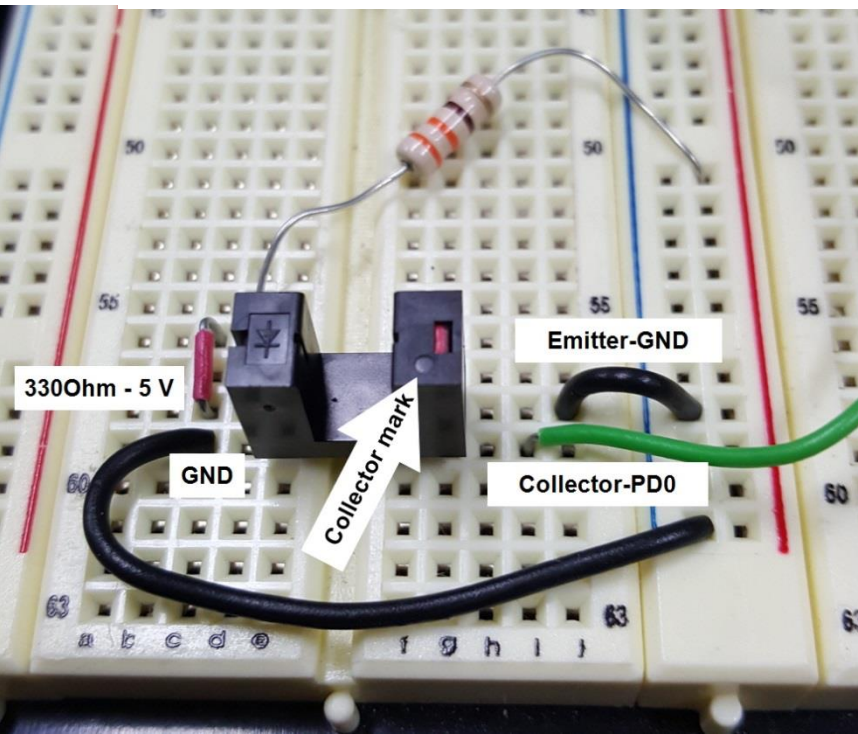
Photo Interrupter

회로 구성



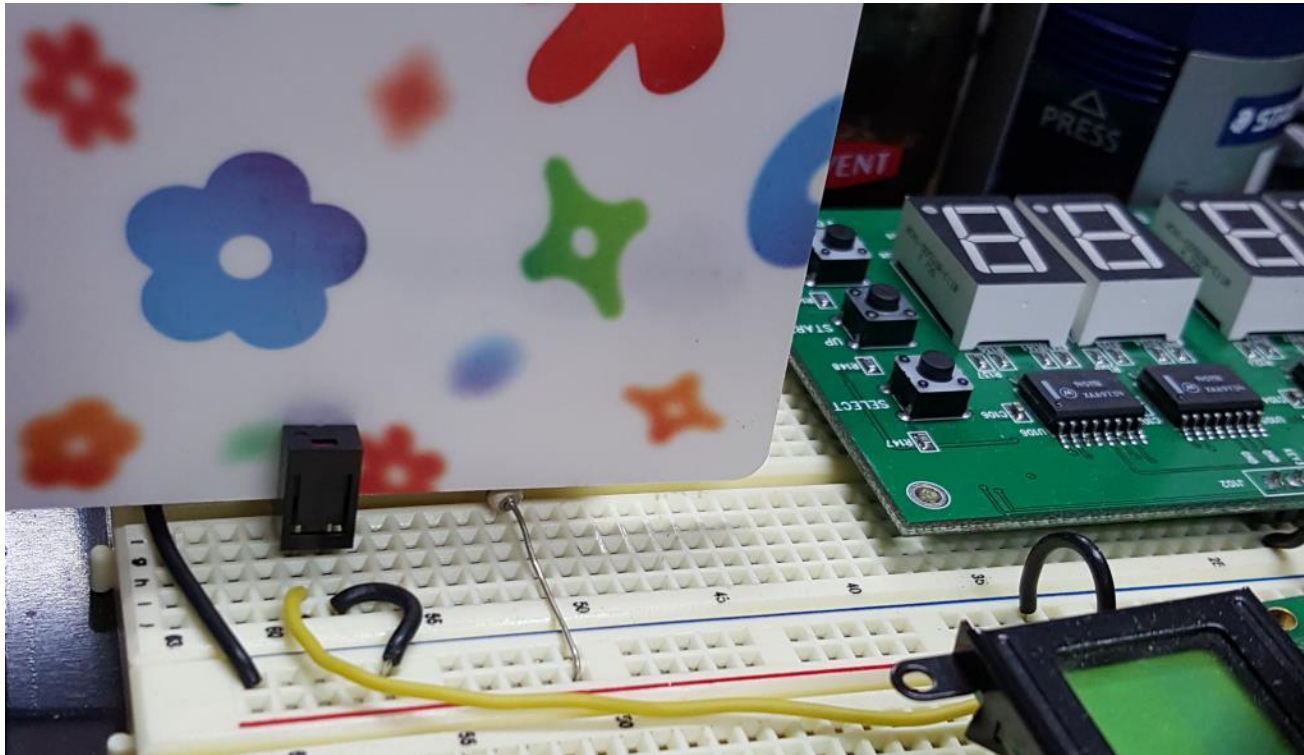
ATmega128 보드

PD0(INT0)



Sample code 실행

- interrupt.c 와 polling.c를 실행한다.



Interrupt Vectors in ATmega128

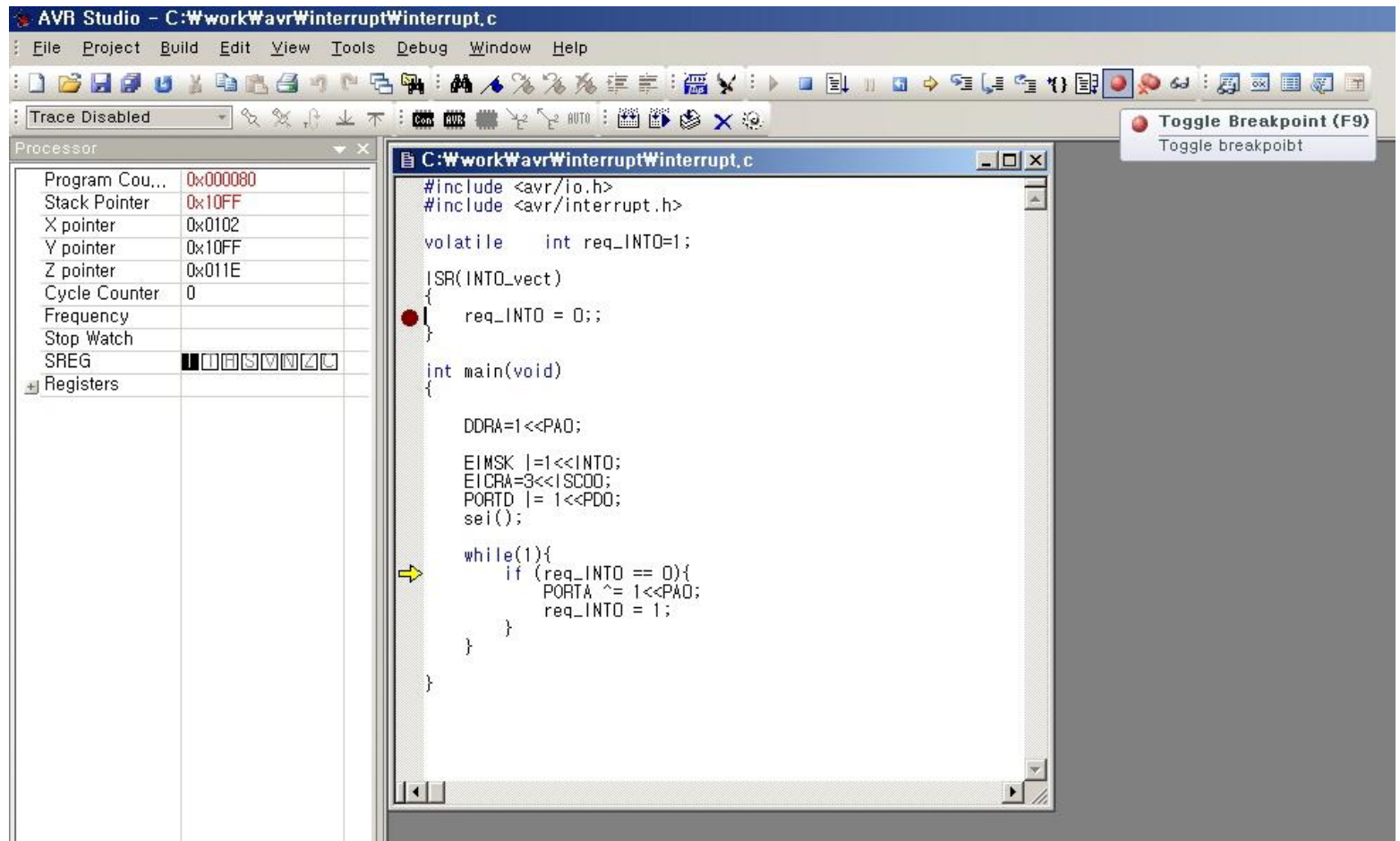
Table 23. Reset and Interrupt Vectors

Vector No.	Program Address ⁽²⁾	Source	Interrupt Definition
1	\$0000 ⁽¹⁾	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	\$0002	INT0	External Interrupt Request 0
3	\$0004	INT1	External Interrupt Request 1
4	\$0006	INT2	External Interrupt Request 2
5	\$0008	INT3	External Interrupt Request 3
6	\$000A	INT4	External Interrupt Request 4
7	\$000C	INT5	External Interrupt Request 5
8	\$000E	INT6	External Interrupt Request 6
9	\$0010	INT7	External Interrupt Request 7
10	\$0012	TIMER2 COMP	Timer/Counter2 Compare Match
11	\$0014	TIMER2 OVF	Timer/Counter2 Overflow
12	\$0016	TIMER1 CAPT	Timer/Counter1 Capture Event
13	\$0018	TIMER1 COMPA	Timer/Counter1 Compare Match A
14	\$001A	TIMER1 COMPB	Timer/Counter1 Compare Match B
15	\$001C	TIMER1 OVF	Timer/Counter1 Overflow
16	\$001E	TIMER0 COMP	Timer/Counter0 Compare Match
17	\$0020	TIMER0 OVF	Timer/Counter0 Overflow

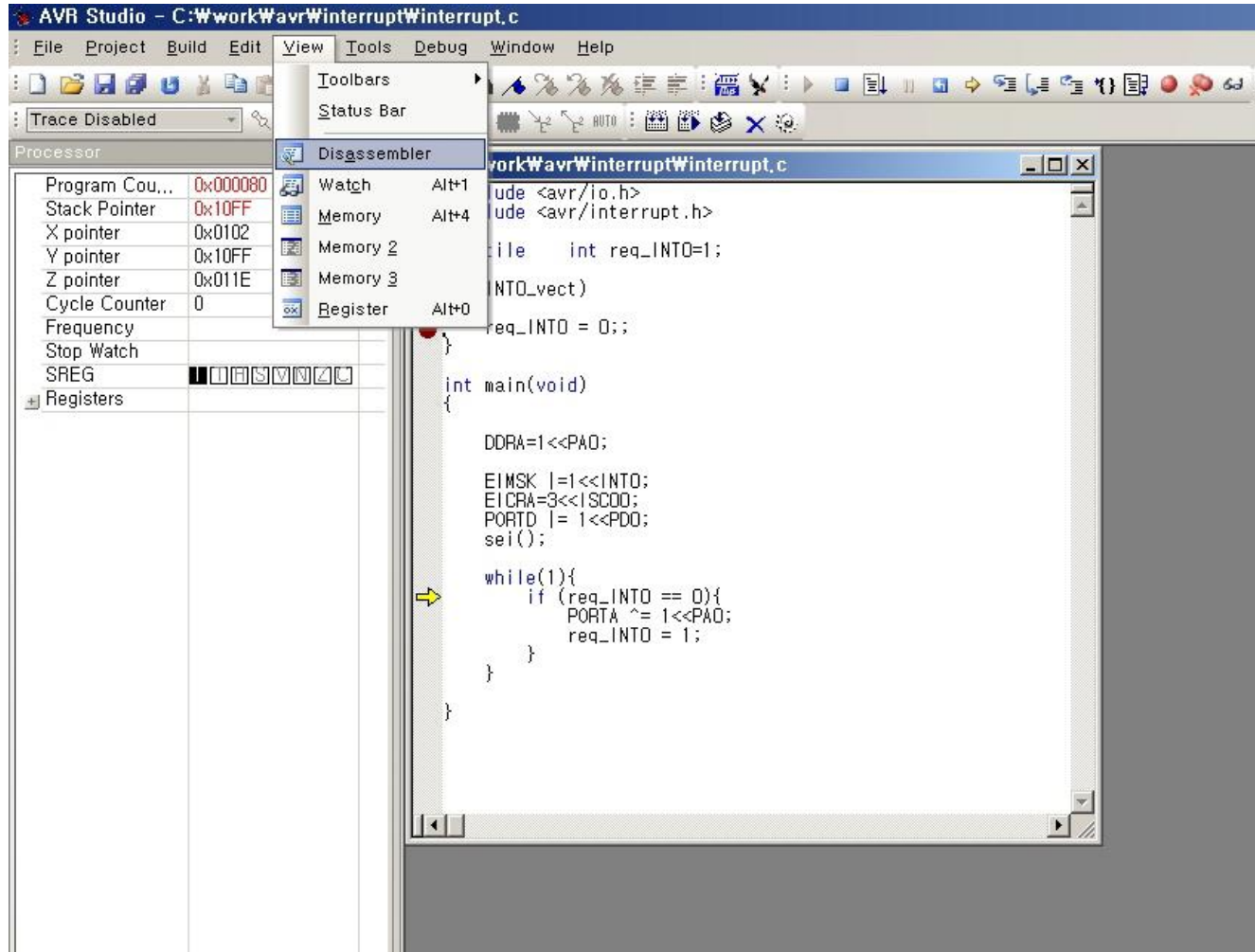
Interrupt Vectors in ATmega128

18	\$0022	SPI, STC	SPI Serial Transfer Complete
19	\$0024	USART0, RX	USART0, Rx Complete
20	\$0026	USART0, UDRE	USART0 Data Register Empty
21	\$0028	USART0, TX	USART0, Tx Complete
22	\$002A	ADC	ADC Conversion Complete
23	\$002C	EE READY	EEPROM Ready
24	\$002E	ANALOG COMP	Analog Comparator
25	\$0030 ⁽³⁾	TIMER1 COMPC	Timer/Counter1 Compare Match C
26	\$0032 ⁽³⁾	TIMER3 CAPT	Timer/Counter3 Capture Event
27	\$0034 ⁽³⁾	TIMER3 COMPA	Timer/Counter3 Compare Match A
28	\$0036 ⁽³⁾	TIMER3 COMPB	Timer/Counter3 Compare Match B
29	\$0038 ⁽³⁾	TIMER3 COMPC	Timer/Counter3 Compare Match C
30	\$003A ⁽³⁾	TIMER3 OVF	Timer/Counter3 Overflow

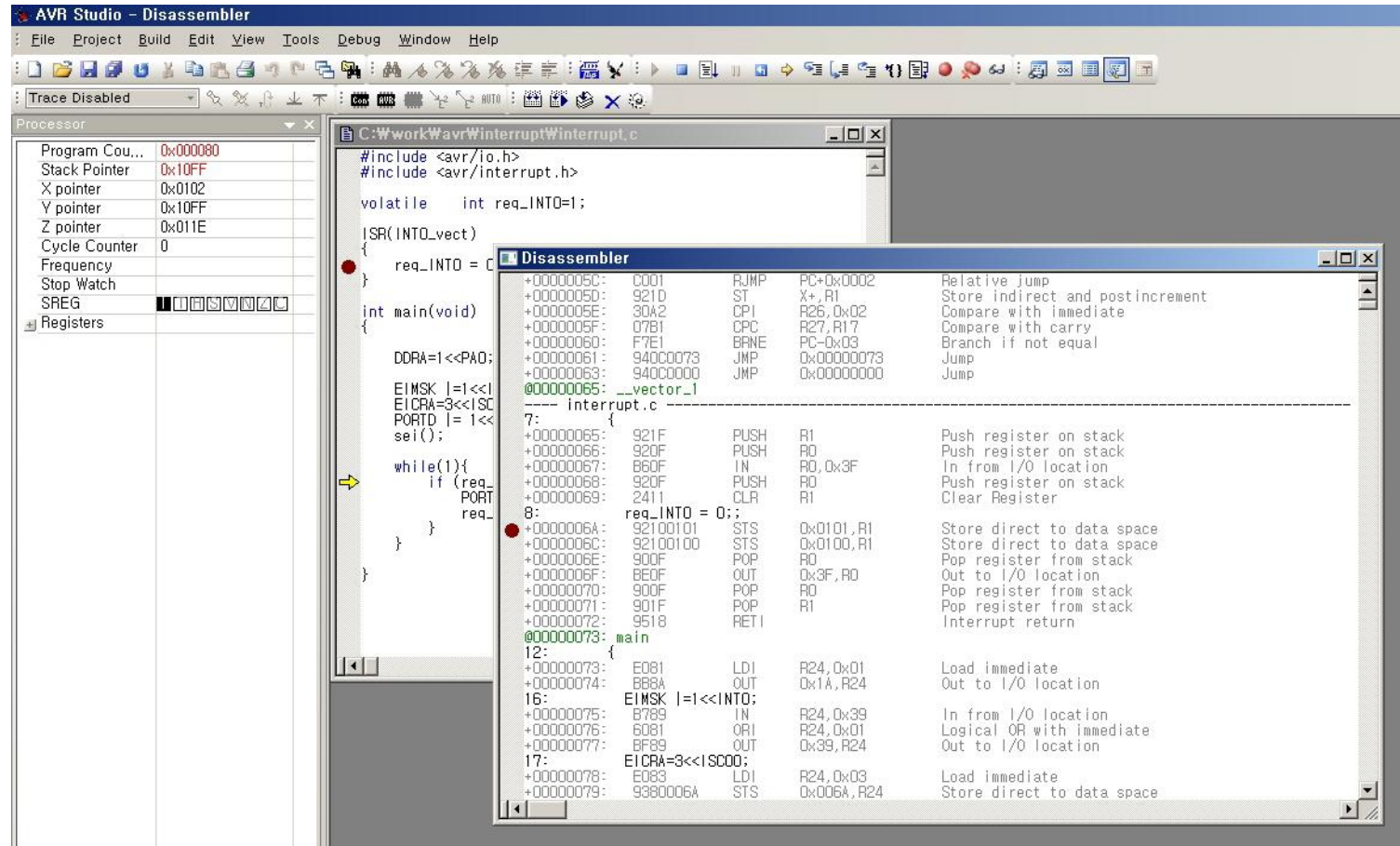
Toggle Breakpoint in ISR



Open Disassembler Window



Disassembler Window



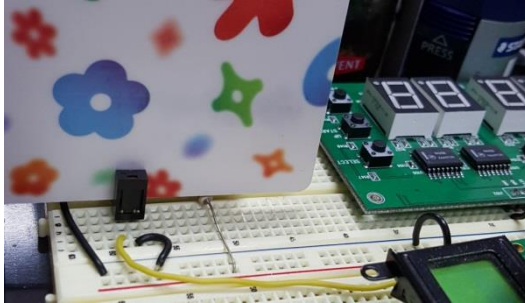
Interrupt Vector Table

The screenshot displays the AVR Studio - Disassembler interface. The main window shows the source code for `C:\work\avr\winterrupt\winterrupt.c`. The code includes `<avr/io.h>` and `<avr/interrupt.h>`, and defines a volatile integer `req_INT0=1;`. The `ISR(INT0_vect)` function is shown with a red dot indicating the start of the interrupt service routine. The `main` function is also visible, showing the initialization of `DDRA`, `EIMSK`, `EICRA`, and `PORTD`, followed by a `while(1)` loop that checks for `req_INT0` and calls `sei()`.

The Disassembler window is open, showing the assembly code for the interrupt vector table. The table consists of 16 entries, each starting with a jump instruction to the interrupt service routine.

Address	Instruction	Target Address	Comment
+00000000	JMP	0x00000046	Jump
+00000002	JMP	0x00000065	Jump
+00000004	JMP	0x00000063	Jump
+00000006	JMP	0x00000063	Jump
+00000008	JMP	0x00000063	Jump
+0000000A	JMP	0x00000063	Jump
+0000000C	JMP	0x00000063	Jump
+0000000E	JMP	0x00000063	Jump
+00000010	JMP	0x00000063	Jump
+00000012	JMP	0x00000063	Jump
+00000014	JMP	0x00000063	Jump
+00000016	JMP	0x00000063	Jump
+00000018	JMP	0x00000063	Jump
+0000001A	JMP	0x00000063	Jump
+0000001C	JMP	0x00000063	Jump
+0000001E	JMP	0x00000063	Jump
+00000020	JMP	0x00000063	Jump
+00000022	JMP	0x00000063	Jump
+00000024	JMP	0x00000063	Jump
+00000026	JMP	0x00000063	Jump
+00000028	JMP	0x00000063	Jump
+0000002A	JMP	0x00000063	Jump
+0000002C	JMP	0x00000063	Jump
+0000002E	JMP	0x00000063	Jump
+00000030	JMP	0x00000063	Jump
+00000032	JMP	0x00000063	Jump
+00000034	JMP	0x00000063	Jump
+00000036	JMP	0x00000063	Jump
+00000038	JMP	0x00000063	Jump
+0000003A	JMP	0x00000063	Jump
+0000003C	JMP	0x00000063	Jump
+0000003E	JMP	0x00000063	Jump
+00000040	JMP	0x00000063	Jump
+00000042	JMP	0x00000063	Jump

Stopped at the Breakpoint



AVR Studio - Disassembler

File Project Build Edit View Tools Debug Window Help

Trace Disabled

Processor

Program Counter	0x00006A
Stack Pointer	0x10FA
X pointer	0x0102
Y pointer	0x10FF
Z pointer	0x011E
Cycle Counter	0
Frequency	
Stop Watch	
SREG	
Registers	

C:\work\Wavr\Winterrupt\Winterrupt.c

```
#include <avr/io.h>
#include <avr/interrupt.h>

volatile int req_INT0=1;

ISR(INT0_vect)
{
    req_INT0 = 0;;
}

int main(void)
{
    DDRA=1<<PA0;

    EIMSK |=1<<INT0;
    EICRA=3<<ISC00;
    PORTD |= 1<<PD0;
    sei();

    while(1){
        if (req_INT0 == 0){
            PORTA ^= 1<<PA0;
            req_INT0 = 1;
        }
    }
}
```

Disassembler

Address	OpCode	Comment	Register
+0000005C:	C001	RJMP	PC+0x0002
+0000005D:	921D	ST	X+, R1
+0000005E:	30A2	CPI	R26, 0x02
+0000005F:	07B1	CPC	R27, R17
+00000060:	F7E1	BRNE	PC-0x03
+00000061:	940C0073	JMP	0x00000073
+00000063:	940C0000	JMP	0x00000000
+00000065:		__vector_1	
----- interrupt.c -----			
7:			
+00000065:	921F	PUSH	R1
+00000066:	920F	PUSH	R0
+00000067:	B60F	IN	R0, 0x3F
+00000068:	920F	PUSH	R0
+00000069:	2411	CLR	R1
8:			
+0000006A:	92100101	STS	0x0101, R1
+0000006C:	92100100	STS	0x0100, R1
+0000006E:	900F	POP	R0
+0000006F:	B60F	OUT	0x3F, R0
+00000070:	900F	POP	R0
+00000071:	901F	POP	R1
+00000072:	9518	RETI	
+00000073:		main	
12:			
+00000073:	E081	LDI	R24, 0x01
+00000074:	B68A	OUT	0x1A, R24
16:			
+00000075:	B789	IN	R24, 0x39
+00000076:	6081	ORI	R24, 0x01
+00000077:	BF89	OUT	0x39, R24
17:			
+00000078:	E083	LDI	R24, 0x03
+00000079:	9380006A	STS	0x006A, R24

Exercise

- 주어진 예제 코드 `interrupt_double.c`는 **rising edge**와 **falling edge**에서 모두 인터럽트가 발생하는 프로그램이다.
- 이 예제와 **delay** 함수를 이용하여 포토 인터럽터가 차단되는 시간을 측정하여 그 값을 msec값으로 **LCD 디스플레이(Memory mapped I/O LCD)**에 나타낸다.
- 나타내는 msec의 값은 0000~9999까지의 4자리 숫자로 나타낸다. 신용 카드 등을 이용하여 포토 인터럽터에 넣다가 빼면 즉시 차단 시간을 측정하여 나타낸다. 이 동작이 무한 계속된다.