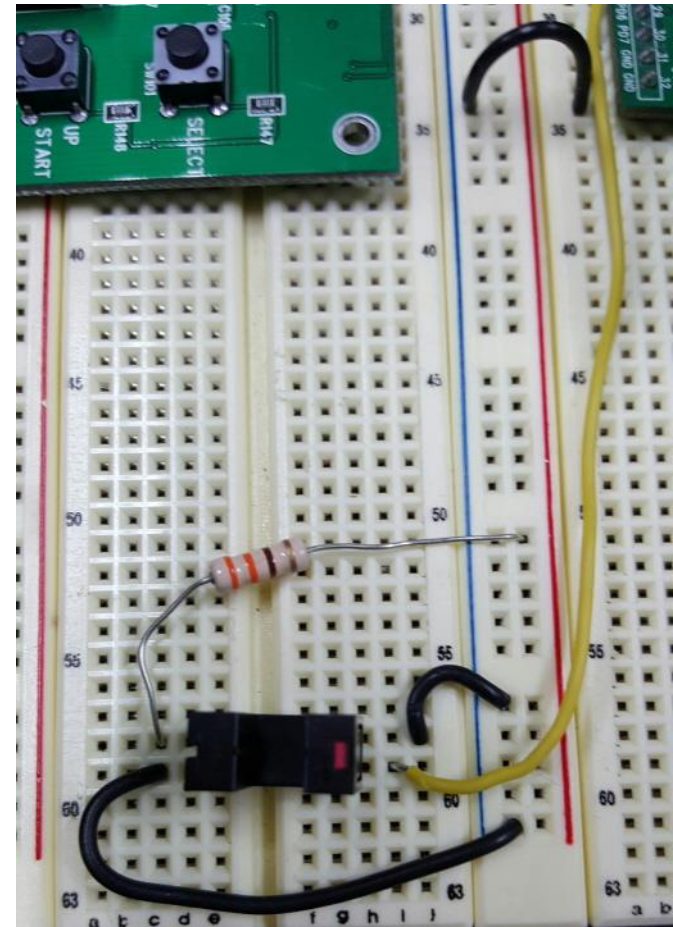
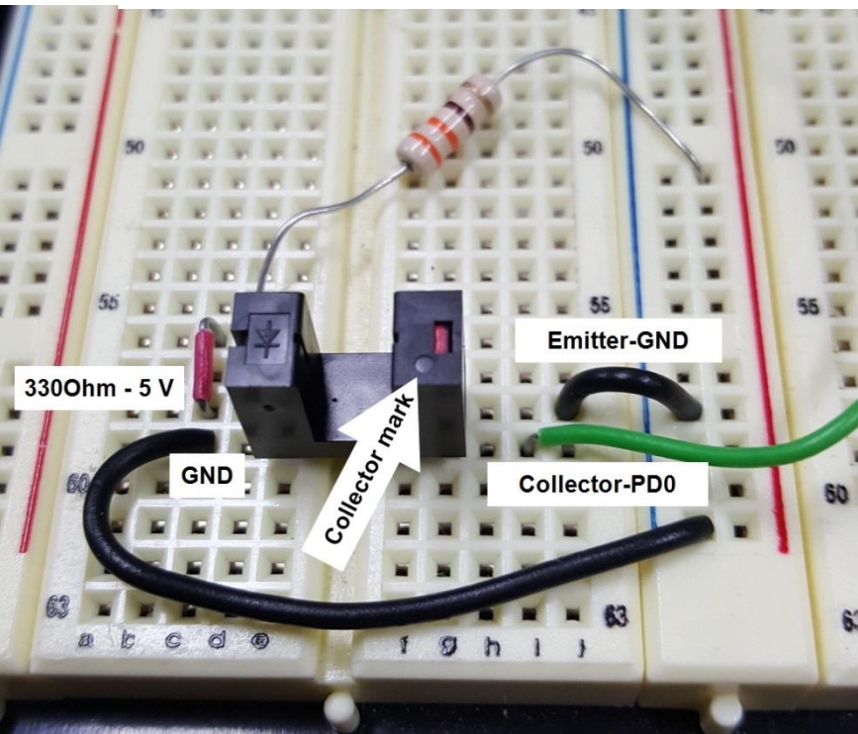
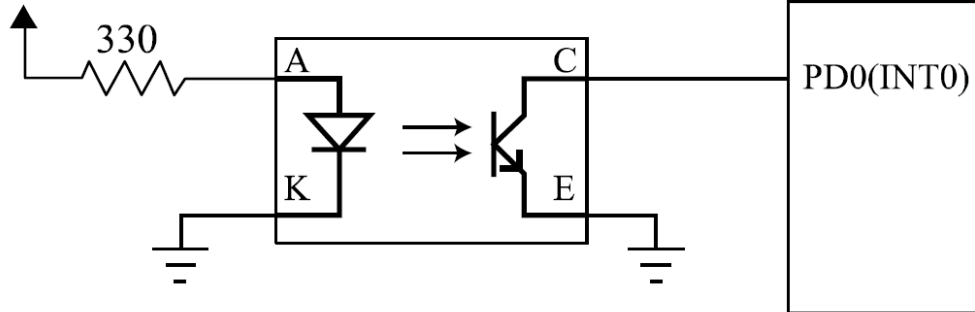




Lab4. Interrupt

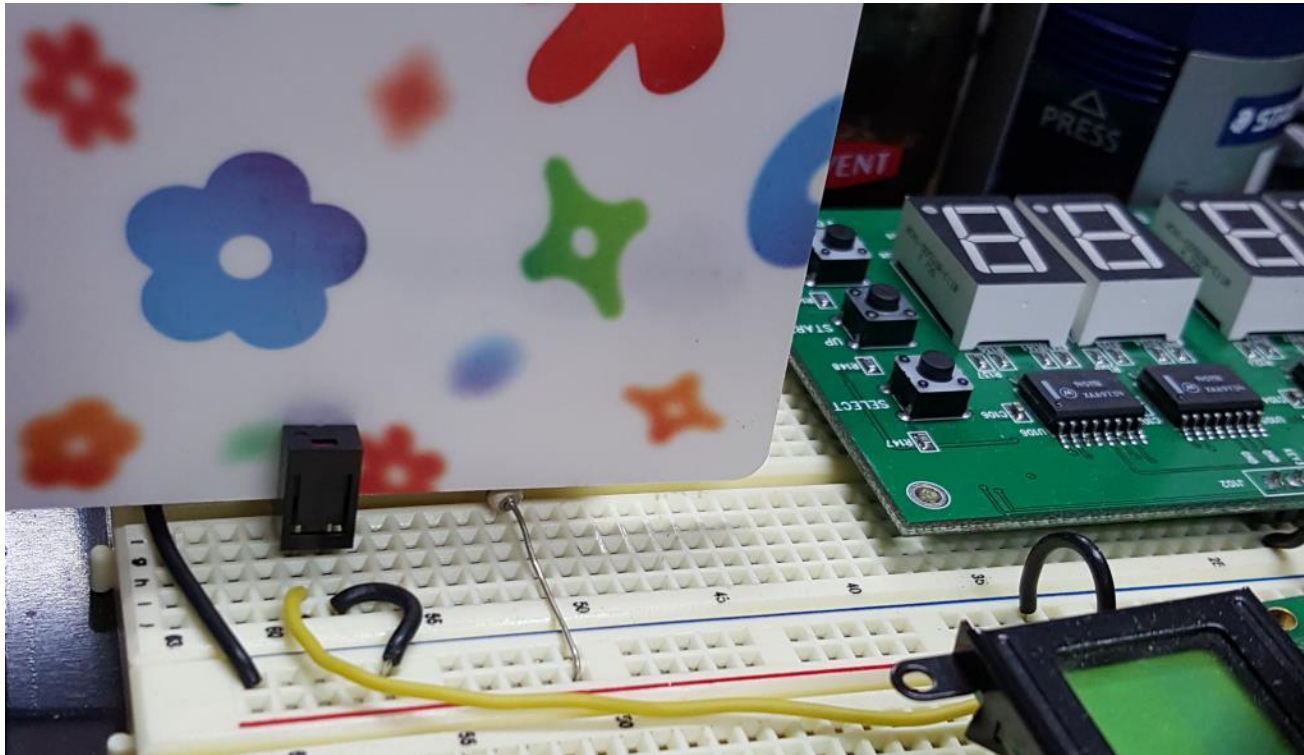
회로 구성

ATmega128 보드



Sample code 실행

- ▶ interrupt.c 와 polling.c를 실행한다.



Interrupt Vectors in ATmega128

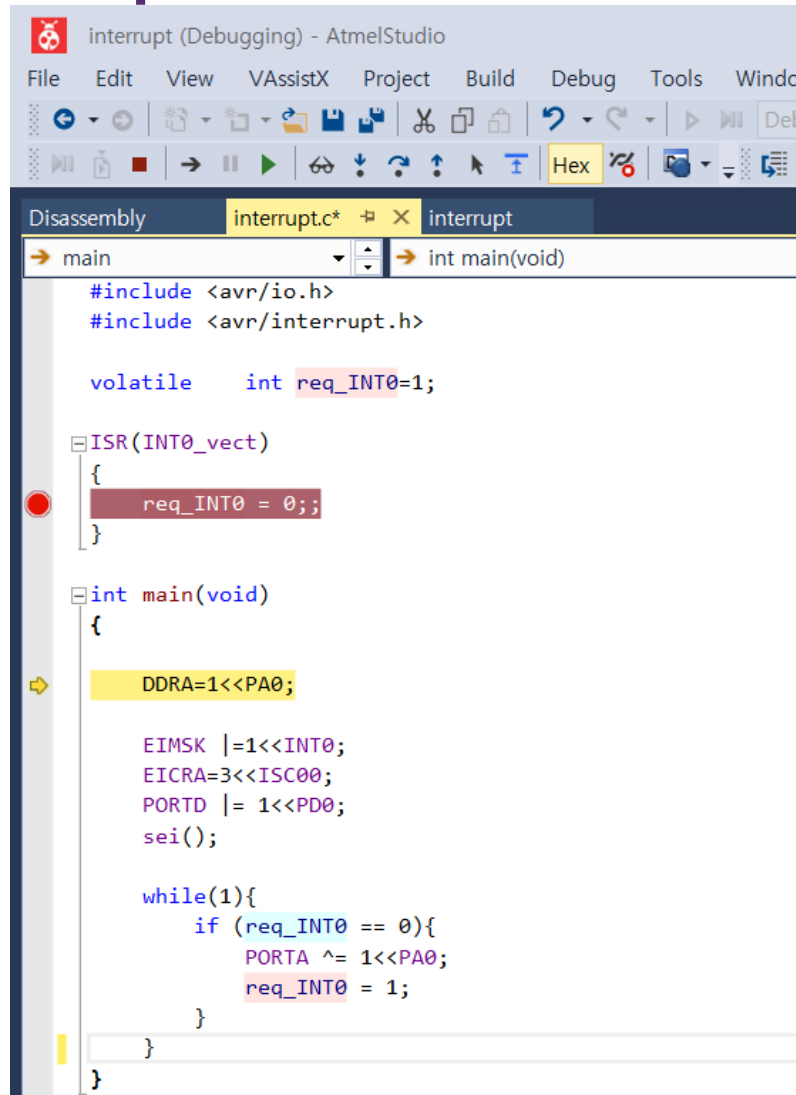
Table 23. Reset and Interrupt Vectors

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 Interrupt Service Routine



The screenshot shows the AtmelStudio IDE in 'Debugging' mode. The 'interrupt.c*' file is open, and the 'interrupt' function is selected. The code is as follows:

```
#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;
        }
    }
}
```

A red circle breakpoint is set on the line `req_INT0 = 0;;` within the `ISR(INT0_vect)` function. A yellow arrow breakpoint is set on the line `DDRA=1<<PA0;` within the `main` function. The 'Hex' view is selected in the bottom right corner of the editor.

Open Disassembler Window

The screenshot shows the AtmelStudio IDE interface. The main window displays the source code for `interrupt.c`. The `main` function is visible, and the cursor is positioned at the line `DDRA=1<<PA0;`. The `ISR(INT0_vect)` function is also shown, containing the line `req_INT0 = 0;;`. The `Debug` menu is open, showing various debugging options. The `Windows` submenu is also open, displaying a list of windows and their keyboard shortcuts. The `Disassembly` window is highlighted in the `Windows` submenu, with the shortcut `Alt+8`.

interrupt (Debugging) - AtmelStudio

File Edit View VAssistX Project Build Debug Tools Window Help

Windows

- Start Debugging and Break Alt+F5
- Attach to Target
- Stop Debugging Ctrl+Shift+F5
- Start Without Debugging Ctrl+Alt+F5
- Disable debugWIRE and Close
- Continue F5
- Execute Stimulifile
- Set Stimulifile
- Restart
- Break All Ctrl+F5
- QuickWatch... Shift+F9
- Step Into F11
- Step Over F10
- Step Out Shift+F11
- Run To Cursor Ctrl+F10
- Reset Shift+F5
- Other Debug Targets
- Toogle Breakpoint F9

Breakpoints Alt+F9

Data Breakpoints

Processor Status

I/O

Live Watch

Program Counter Trace

Output

Watch

- Autos Ctrl+Alt+V, A
- Locals Alt+4
- Immediate Ctrl+Alt+I

Call Stack Alt+7

Threads Ctrl+Alt+H

Modules Ctrl+Alt+U

Processes Ctrl+Shift+Alt+P

Memory

- Disassembly Alt+8
- Registers Alt+5

Disassembly interrupt.c* interrupt

main int main(void)

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

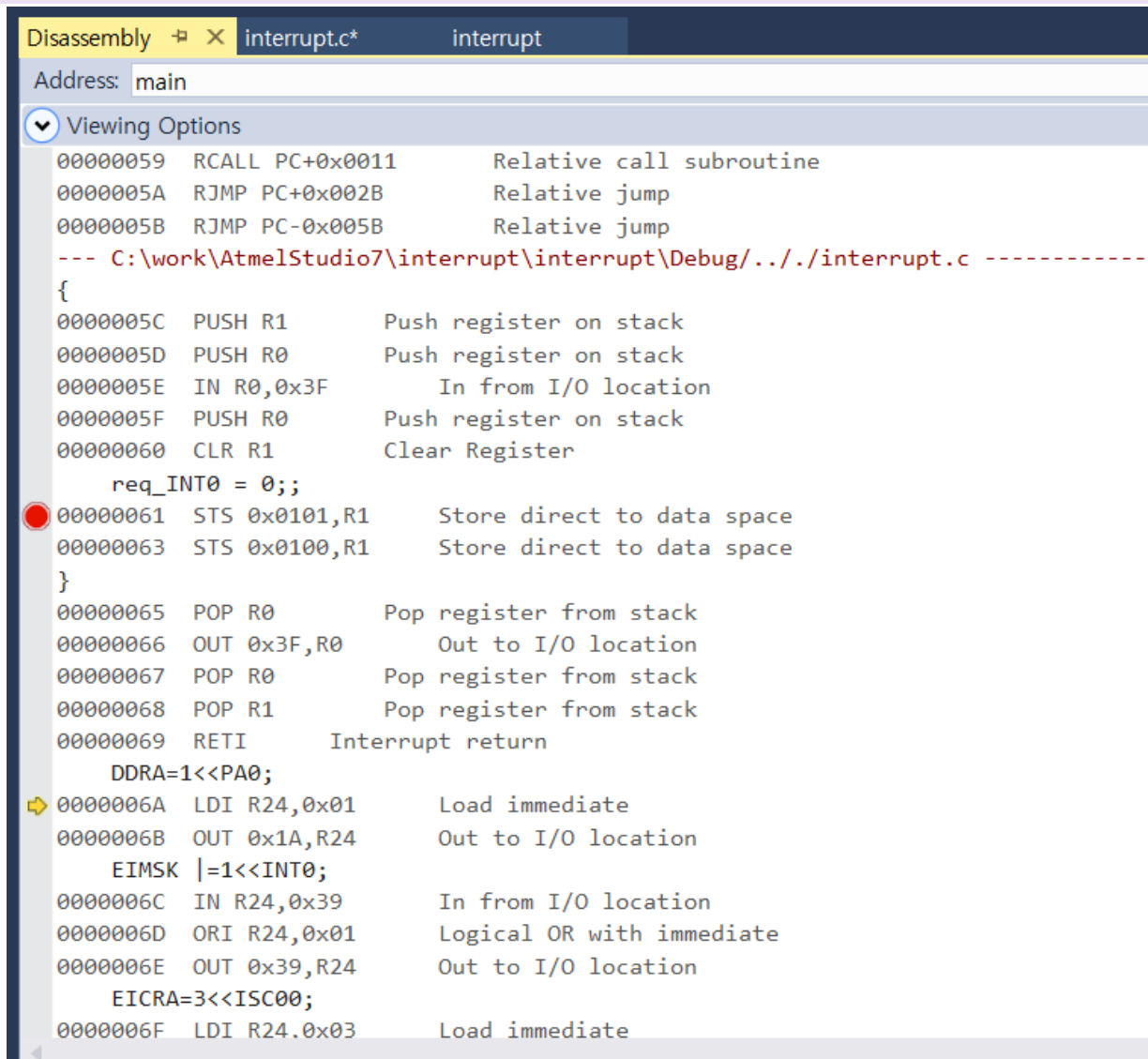
volatile int req_INT0=1;

ISR(INT0_vect)
{
    req_INT0 = 0;;
}

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

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


Disassembler Window

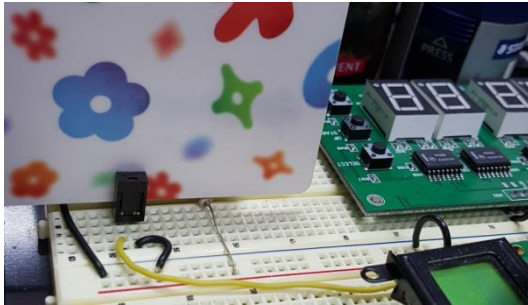


```
Disassembly  interrupt.c*  interrupt
Address: main
Viewing Options
00000059 RCALL PC+0x0011 Relative call subroutine
0000005A RJMP PC+0x002B Relative jump
0000005B RJMP PC-0x005B Relative jump
--- C:\work\AtmelStudio7\interrupt\interrupt\Debug\../interrupt.c -----
{
0000005C PUSH R1 Push register on stack
0000005D PUSH R0 Push register on stack
0000005E IN R0,0x3F In from I/O location
0000005F PUSH R0 Push register on stack
00000060 CLR R1 Clear Register
    req_INT0 = 0;;
00000061 STS 0x0101,R1 Store direct to data space
00000063 STS 0x0100,R1 Store direct to data space
}
00000065 POP R0 Pop register from stack
00000066 OUT 0x3F,R0 Out to I/O location
00000067 POP R0 Pop register from stack
00000068 POP R1 Pop register from stack
00000069 RETI Interrupt return
    DDRA=1<<PA0;
0000006A LDI R24,0x01 Load immediate
0000006B OUT 0x1A,R24 Out to I/O location
    EIMSK |=1<<INT0;
0000006C IN R24,0x39 In from I/O location
0000006D ORI R24,0x01 Logical OR with immediate
0000006E OUT 0x39,R24 Out to I/O location
    EICRA=3<<ISC00;
0000006F LDI R24,0x03 Load immediate
```


Interrupt Vector Table

Disassembly interrupt.c* interrupt		
Address: main		
Viewing Options		
--- No source file ---		
00000000	RJMP PC+0x0046	Relative jump
00000001	NOP	No operation
00000002	RJMP PC+0x005A	Relative jump
00000003	NOP	No operation
00000004	RJMP PC+0x0057	Relative jump
00000005	NOP	No operation
00000006	RJMP PC+0x0055	Relative jump
00000007	NOP	No operation
00000008	RJMP PC+0x0053	Relative jump
00000009	NOP	No operation
0000000A	RJMP PC+0x0051	Relative jump
0000000B	NOP	No operation
0000000C	RJMP PC+0x004F	Relative jump
0000000D	NOP	No operation
0000000E	RJMP PC+0x004D	Relative jump
0000000F	NOP	No operation
00000010	RJMP PC+0x004B	Relative jump
00000011	NOP	No operation
00000012	RJMP PC+0x0049	Relative jump
00000013	NOP	No operation
00000014	RJMP PC+0x0047	Relative jump
00000015	NOP	No operation
00000016	RJMP PC+0x0045	Relative jump
00000017	NOP	No operation
00000018	RJMP PC+0x0043	Relative jump
00000019	NOP	No operation

Stopped at the Breakpoint



```
Disassembly    interrupt.c  X  interrupt
→ interrupt.c  C:\work\AtmelStudio7\W...

#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();

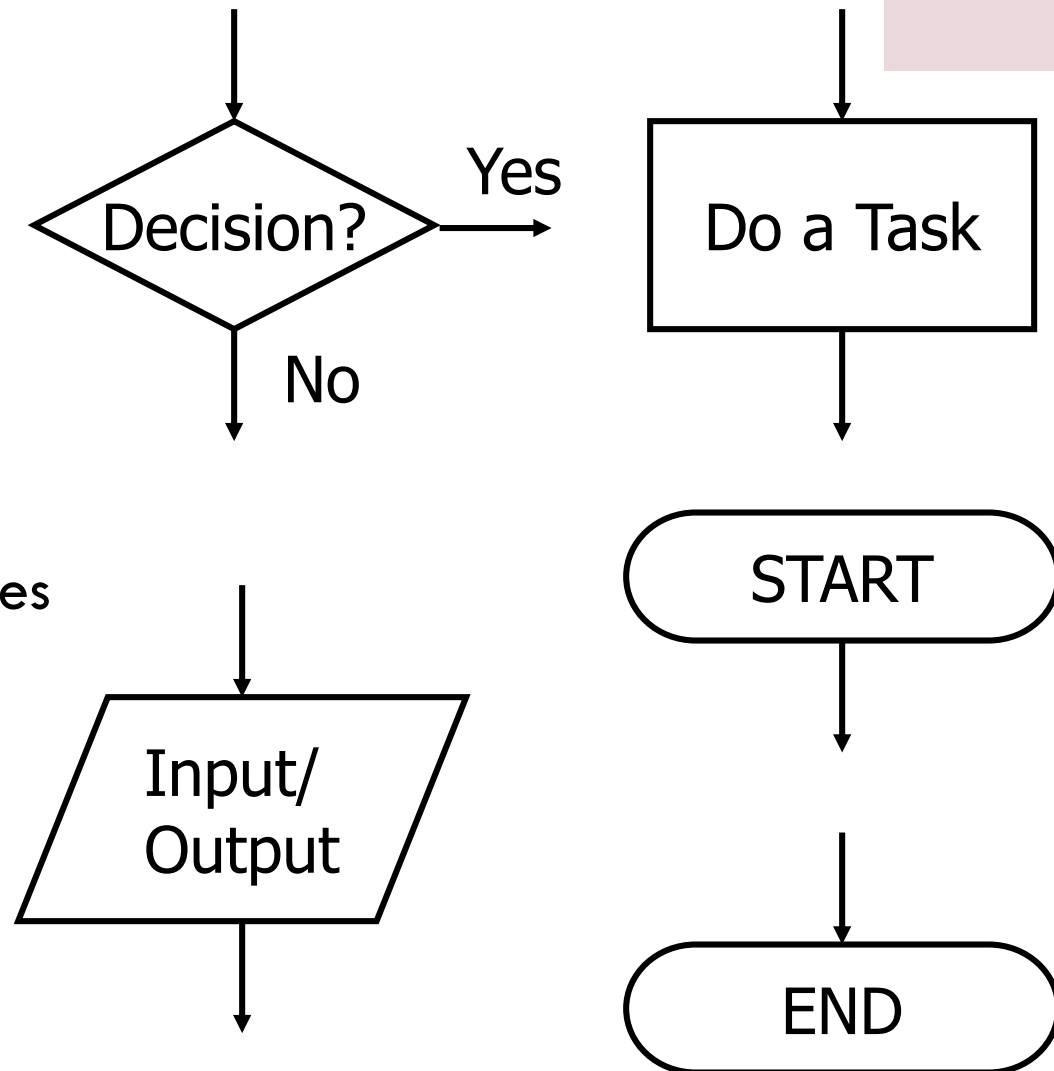
    while(1){
        if (req_INT0 == 0){
            PORTA ^= 1<<PA0;
            req_INT0 = 1;
        }
    }
}
```

Exercise

- ▶ 포토 인터럽터에서 발생하는 인터럽트의 횟수를 character LCD에 나타낸다.
- ▶ 처음 시작 시 LCD는 00을 표시하고, 인터럽트를 기다린다.
- ▶ 인터럽트가 발생할 때 마다 1씩 증가.
- ▶ 카운트 값이 99에 도달한 후 다음 인터럽트가 발생하면 다시 0으로 돌아간다.(시험할 필요 없음)
- ▶ 버튼을 누르면 카운트 값은 0으로 돌아간다.(버튼 입력은 PG4사용)

Flowchart

- ▶ Shows flow of control in a processing activity (what gets done)
- ▶ Used to show steps in a process, including decision-making
- ▶ Does not scale well: becomes confusing if larger than a page



Flowchart Example

