TECH 3233
Lab #6
Interrupts

Ver INT/TOF - 0.6

In this experiment we will use both external interrupts (INT0 and INT1) and a Timer Overflow Interrupt to turn on and off a flashing LED.

Flashing an LED

If you recall, in Lab 3 we blinked the onboard LED by using the following code:

```
#include <atmel_start.h>
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
      /* Initializes MCU, drivers and middleware */
      atmel_start_init();
    DDRB = 0b00100000; // configure pin 5 of PORTB as output
(digital pin 13 on the Arduino Uno)

    while(1)
    {
          PORTB = 0b00100000; // set 5th bit to HIGH
          _delay_ms(1000);
          PORTB = 0b00000000; // set 5th bit to LOW
          _delay_ms(1000);
    }
}
```

But, after the setup, all we did in main was turn on a bit, pause, turn off a bit, pause and repeat. But what if we want to make better use of the processor while waiting for the next on/off? We can use interrupts to allow the processor to do "other things" while still blinking the LED at a constant rate.

To do this we will use TIMER0 and the Overflow Interrupt.

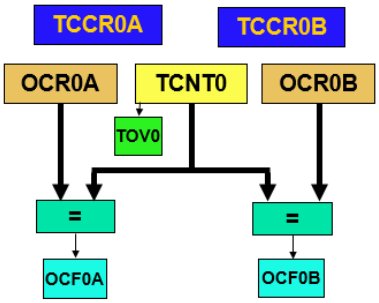


Figure 1- TIMER0 Overview

Timer0, can be used to measure and generate square waves, but in this experiment, we are just using the free running 8-bit timer (TCNT0) and the Timer Overflow (TOV0).

TCNT0 just counts at a known interval. That interval is defined by the OSC on the Arduino board (16MHz) and the Timer Prescaler defined by TCCR0B Register:

TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 15-9. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

If external pin modes are used for the Timer/Counter0, transitions on the T0 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

(For this experiment FOC0A and FOC0B should be left at 0)

So if CS02:CS00 was 001, TNCT0 would increment by one every clock cycle (since it is clk/(no presscaling) it would increment at a frequency of 16MHz or every 63pSec.

If we use CS02:CS00 as 101, TCNT0 would increment at 16Mhz/1024, or 15625HZ or once every 64uSec.

Since we wish to flash the LED 1 sec on then 1 sec off, we could in theory, use any prescaler, but any prescale value we use will still be much lest than 1 second. For example, lets take the clk/1024 prescaler:

$$f_{scaled} = \frac{16MHz}{1024}$$
$$= 15625\ Hz$$

$$time_{count} = \frac{1}{f_{scaled}}$$
$$= 64uSec$$

$$t_{overflow} = 256 * time_{count}$$
$$= 16.384mSec$$

So we need

$$Number\ of\ Overflows = \frac{1Sec}{16.384mSec} = 61$$

So, for this we will need to add a counter within the interrupt to count how many times the interrupt has occurred, then every 61 interrupts we toggle the LED. (so an "IF" inside the interrupt, looking at the counter)

The Timer does have different operational modes. For this experiment we will leave it in NORMAL Mode (Mode 0) as defined by this table:

Table 15-8. Waveform Generation Mode Bit Description

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on[1][2]
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

Figure 2- Timer Modes of Operation

Bit WGM02 is found in register TCCR0B (already discussed) and the other two bits can be found in:

TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

(We will not be using the high nibble bits in this experiment, just leave them at zero)

In Normal Mode, TCNT0 counts from 0 – 255 then resets back to 0 and an Overflow. Other modes will be explained when needed.

Lastly, we will need to turn on the Timer Overflow Interrupt Enable bit found in:

TIMSK0 – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:3 – Reserved**

These bits are reserved bits in the ATmega48A/PA/88A/PA/168A/PA/328/P and will always read as zero.

- **Bit 0 – TOIE0: Timer/Counter0 Overflow Interrupt Enable**

When the TOIE0 bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in the Timer/Counter 0 Interrupt Flag Register – TIFR0.

(we will not be using OCIE0B or OCIE0A – output compare interrupts for this experiment so they will be explained in a latter lab, just leave them 0 for this lab)

Don't forget, since this is a MASKABLE INTERRUPT, you must execute the SEI(); command to enable the Interrupt Flag in the Flag Register for the interrupt to be enabled.

External Interrupts (INT0 and INT1):

To turn on/off the flashing led we will be using both external interrupts (INT0 and INT1)

In addition to setting up DDRD and using PORTD, you will need to use the following registers to set up the external interrupts (INT0 and INT1):

EIMSK – External Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	–	–	–	–	–	–	INT1	INT0	EIMSK
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:2 – Reserved**

These bits are unused bits in the ATmega48A/PA/88A/PA/168A/PA/328/P, and will always read as zero.

- **Bit 1 – INT1: External Interrupt Request 1 Enable**

When the INT1 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control1 bits 1/0 (ISC11 and ISC10) in the External Interrupt Control Register A (EICRA) define whether the external interrupt is activated on rising and/or falling edge of the INT1 pin or level sensed. Activity on the pin will cause an interrupt request even if INT1 is configured as an output. The corresponding interrupt of External Interrupt Request 1 is executed from the INT1 Interrupt Vector.

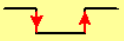
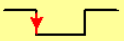
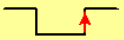
- **Bit 0 – INT0: External Interrupt Request 0 Enable**

When the INT0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control0 bits 1/0 (ISC01 and ISC00) in the External Interrupt Control Register A (EICRA) define whether the external interrupt is activated on rising and/or falling edge of the INT0 pin or level sensed. Activity on the pin will cause an interrupt request even if INT0 is configured as an output. The corresponding interrupt of External Interrupt Request 0 is executed from the INT0 Interrupt Vector.

EICRA – External Interrupt Control Register A

The External Interrupt Control Register A contains control bits for interrupt sense control.

Bit	7	6	5	4	3	2	1	0	
(0x69)	–	–	–	–	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ISCx1	ISCx0	
0	0	
0	1	
1	0	
1	1	

And remember that we need to use the correct Interrupt Service Routine (ISR) names as defined by ATMEL Studio:

Interrupt	Vector Name
External Interrupt Request 0	INT0_vect
External Interrupt Request 1	INT1_vect

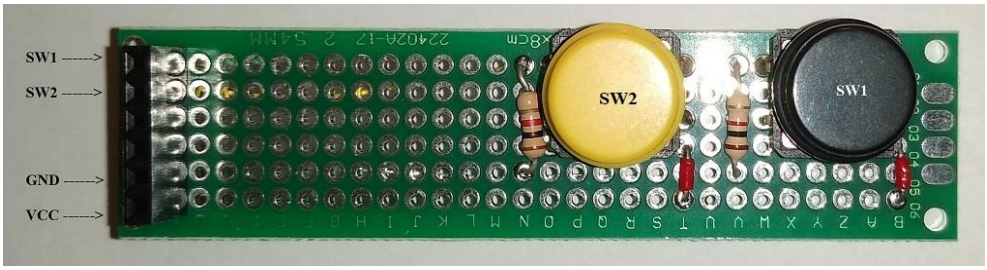
And lastly to turn on all maskable interrupts by using the c function:

sei();

Requirements:

Using only interrupts INT0, INT1 and Timer0 (Timer Overflow Interrupt), create a program that will flash the LED built onto the Arduino board and control by pin 13 (PB5). It should flash at the rate of one second on and one second off.

Using the prewired push button switches shown below:



Connect SW1 to D2 (PD2), the SW2 to D3 (PD3) and Vcc (+5v) and ground (as shown above).

Write a program that will do the following:

- On a rising edge of PD2 (INT0) – flash the LED
- On a falling edge of PD3 (INT1) – stop flashing the LED and ensure it is off

Even though you do not have to put anything inside the While(1) loop in main, we will print the value of the 0-61 counter value to the screen using printf.

Submit the commented code in a zip file via online submission for credit.