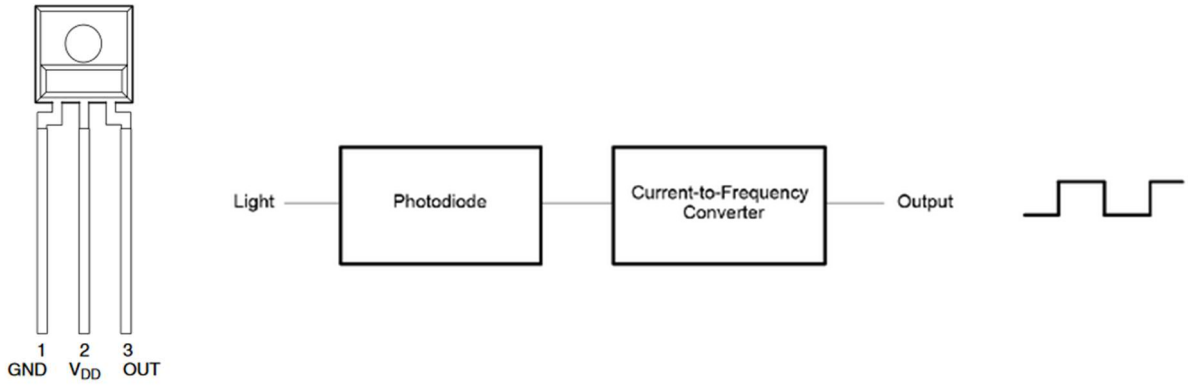
TECH 3233
Lab #7
Ver 2.1

Background: The TSL237 is a light-to-frequency converter. The more intense the light, the higher the frequency output.



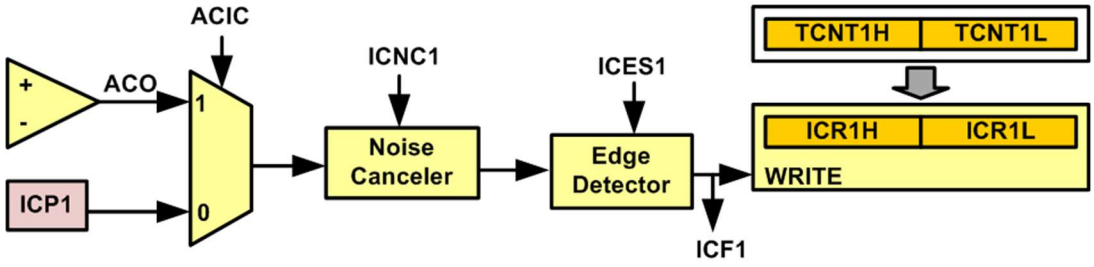
A quick test of the sensor gave the following readings:

Light Condition	Freq out (KHz)
Dark (shaded by hand over sensor)	1.6
Bright Light (using cell phone as flashlight)	806

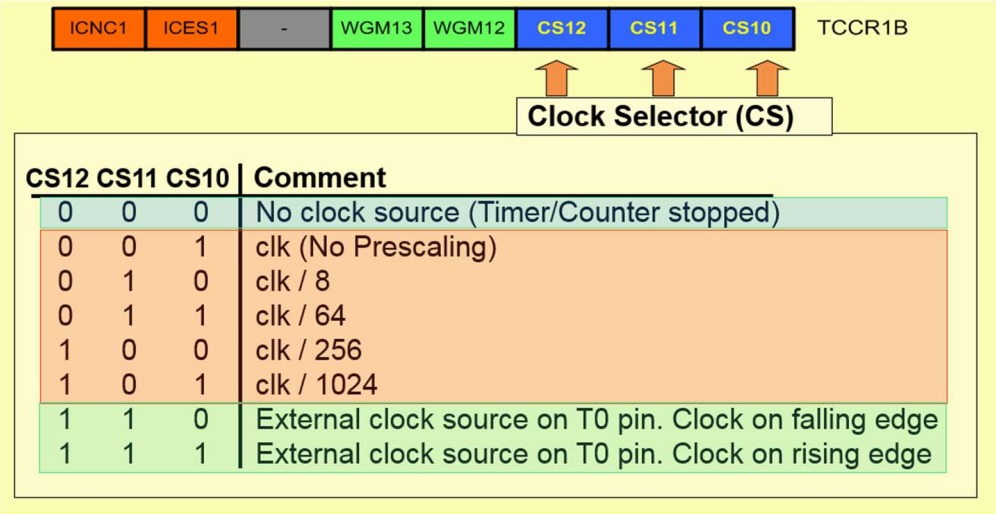
This lab will use the Timer Input Capture feature of the microcontroller to calculate the frequency of the generated square wave of the TSL237.

Most microcontrollers have a "Free running counter" that just increments at some known rate (related to the clock of the microcontroller and a prescaler). The Atmel328p is no exception.

For this experiment, we will use TIMER1:

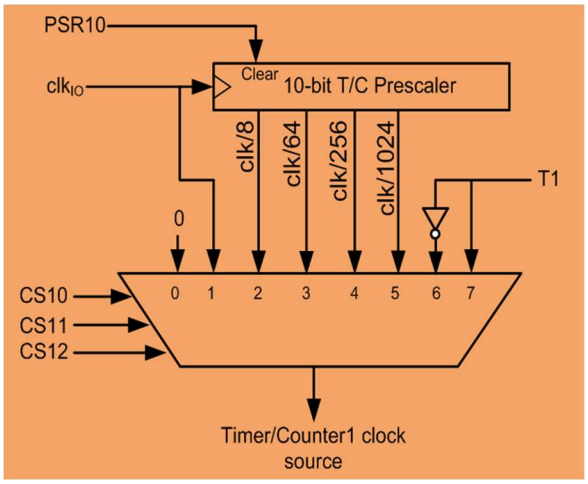


You will need to figure out what clock prescaler to use to allow for the frequency range of the device (listed above).



CS12	CS11	CS10	Comment
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk (No Prescaling)
0	1	0	clk / 8
0	1	1	clk / 64
1	0	0	clk / 256
1	0	1	clk / 1024
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

This prescaler is used to select the speed at which TCNT1 counts (as shown below):



Since we know the Arduino boards have a 16MHz clock. We know with no prescale, each count of the timer will be:

$$t = \frac{1}{f}$$
$$= \frac{1}{16000000}$$
$$= 62.5pSec$$

This can also be considered the theoretical minimum time that can be measured (although really you should design so there are always multiple counts for the smallest time you wish to measure).

Since we know that TIMER1 is a 16bit counter, we know that there are 65535 counts (0xFFFF) we can calculate the maximum time that one full count can measure by multiplying the time (above) by the maximum number of counts.

Create a spreadsheet that will calculate the time per count, max time and max frequency and minimum frequency that can be measured by TIMER1 for each possible prescale factor. Highlight the prescaler that will be used for this experiment and explain why it was chosen (writing your answer in a cell below the created table.

It should be noted that other prescalers can be used if you do count the overflows that occur, but this complicates the code and should NOT be necessary for this experiment if the right prescaler is used.

Registers needed for this experiment include:

TCCR1A – Timer/Counter1 Control Register A

Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
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	

COM1A1:0 and COM1B1:0 are associated with Output Compare and should be set to zero (Normal Port Operation) for this experiment.

WGM11:0 should also be set to zero for this experiment for the same reason.

TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ICNC1 is used for Input Capture Noise Cancelation. If the bit is set to 1, it will wait for 4 equal inputs before triggering TCNT being copied to ICR1H:L. Due to the high variability of the input generated by the sensor in the experiment, it should be turned off (bit set to zero)

ICES1: Input Capture Edge Select

This bit selects which edge on the Input Capture pin (ICP1) that is used to trigger a capture event. When the

ICES1 bit is written to zero, a falling (negative) edge is used as trigger, and when the ICES1 bit is written to one,

a rising (positive) edge will trigger the capture.

When a capture is triggered according to the ICES1 setting, the counter value is copied into the Input Capture

Register (ICR1). The event will also set the Input Capture Flag (ICF1), and this can be used to cause an Input

Capture Interrupt, if this interrupt is enabled.

WGM13:2 are both set to zero (Normal mode – SEE TCCR1A reg for other two bits required)

CS12:0 selects the pre scaler as per the table below:

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (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 T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

Although it is not directly referenced, the 16bit free running counter TCNT1H:L is used. This register is copied to the register ICR1H:L when the edge (selected by ICES1) is detected on pin PB0.

You can trigger an interrupt when this occurs via the register TIMSK1, but we will not be using interrupts in this program. We will be using POLLING (checking a bit and waiting for a required status).

TIFR1 – Timer/Counter1 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	TIFR1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

We will be waiting for the ICF1 bit to be set (this occurs when the edge is detected).

NOTE that you must right a ONE to this bit to clear it (since we are using polling, if we were using an interrupt, the pin would be cleared automatically).

Two more bits need to be set. Ensure the TIMER/Counter1 is turned on (PRTIM1) in

PRR – Power Reduction Register

Bit	7	6	5	4	3	2	1	0	
(0x64)	PRTWI	PRTIM2	PRTIM0	–	PRTIM1	PRSPI	PRUSART0	PRADC	PRR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Also ensure that ACIC in the register below is set to 0. This will prevent an unwanted interrupt when the selected edge is detected.

ACSR – Analog Comparator Control and Status Register

Bit	7	6	5	4	3	2	1	0	
0x30 (0x50)	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	N/A	0	0	0	0	0	

Procedure

After determining the Prescaler needed (via excel spreadsheet and calculations explained in the background). Setup the TIMER1 for Input Capture on the falling edge and the proper prescaler.

Clear the flag.

When the first edge is detected, store ICR1 (ICR1H:L) and clear the flag.

When the second edge is detected. Calculate the Frequency of the square wave generated by the sensor then display the count difference and frequency.

Go back to “clear the flag”.

Reminders

- Don’t forget your clock prescaler in your calculation
- Since Frequency will be a calculation using division, you will need to use floating point values
- With Floating point you will need to turn on the ability to printf with floating point values. See ADC lab for instructions.

Make sure to use a proper prescale value to be able to read 1.6KHz to 806KHz range of the sensor.