Today's Plan:

- Announcements
- Using Analog to Digital Converter – continuation
- Using Pulse Width Modulation
- Activity 3
- Ultrasonic Distance Sensor

In weeks 3-4 you have to finish parts 4, 5, 6 and 7 of the manual

Notes will be due before each lab during the week 5 of the labs.

Using peripherals: Analog to digital converter

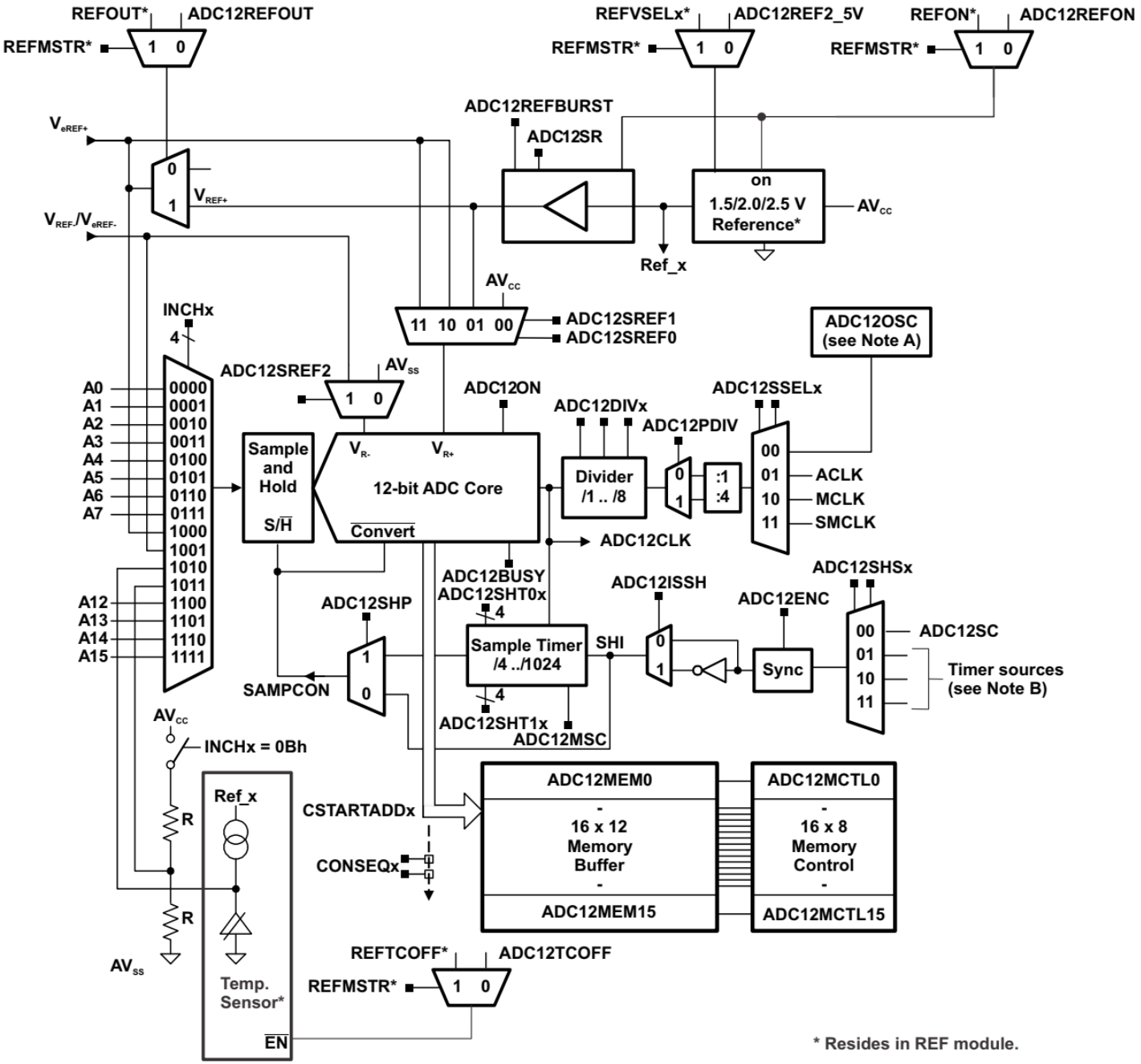
READING THE DATASHEET IS ESSENTIAL!

```
#include <msp430f5529.h>
#include<stdio.h>

int main(void)
{
  WDTCTL = WDTPW + WDTHOLD;
  ADC12CTL0 = ADC12SHT02 + ADC12ON;        // Sampling time, ADC12 on
  ADC12CTL1 = ADC12SHP;                    // sampling timer
  P6SEL |= 0b00000001;;                     // P6.0 allow ADC on pin 6.0
  P1DIR |= 0b00000001;;                     // set pin P1.0 as output
  ADC12CTL0 |= ADC12ENC;                   // ADC enable
  while (1)
  {

    ADC12CTL0 |= ADC12SC;                  // Start sampling
    while (ADC12CTL1 & ADC12BUSY);//while bit ADC12BUSY in register ADC12CTL1 is high wait

    if(ADC12MEM0>=3072)    //This value depends on the input voltage
      P1OUT |= BIT0;
    else
     P1OUT &= ~BIT0;
  }
}
```

Using peripherals: Analog to digital converter

READING THE DATASHEET of the family IS ESS

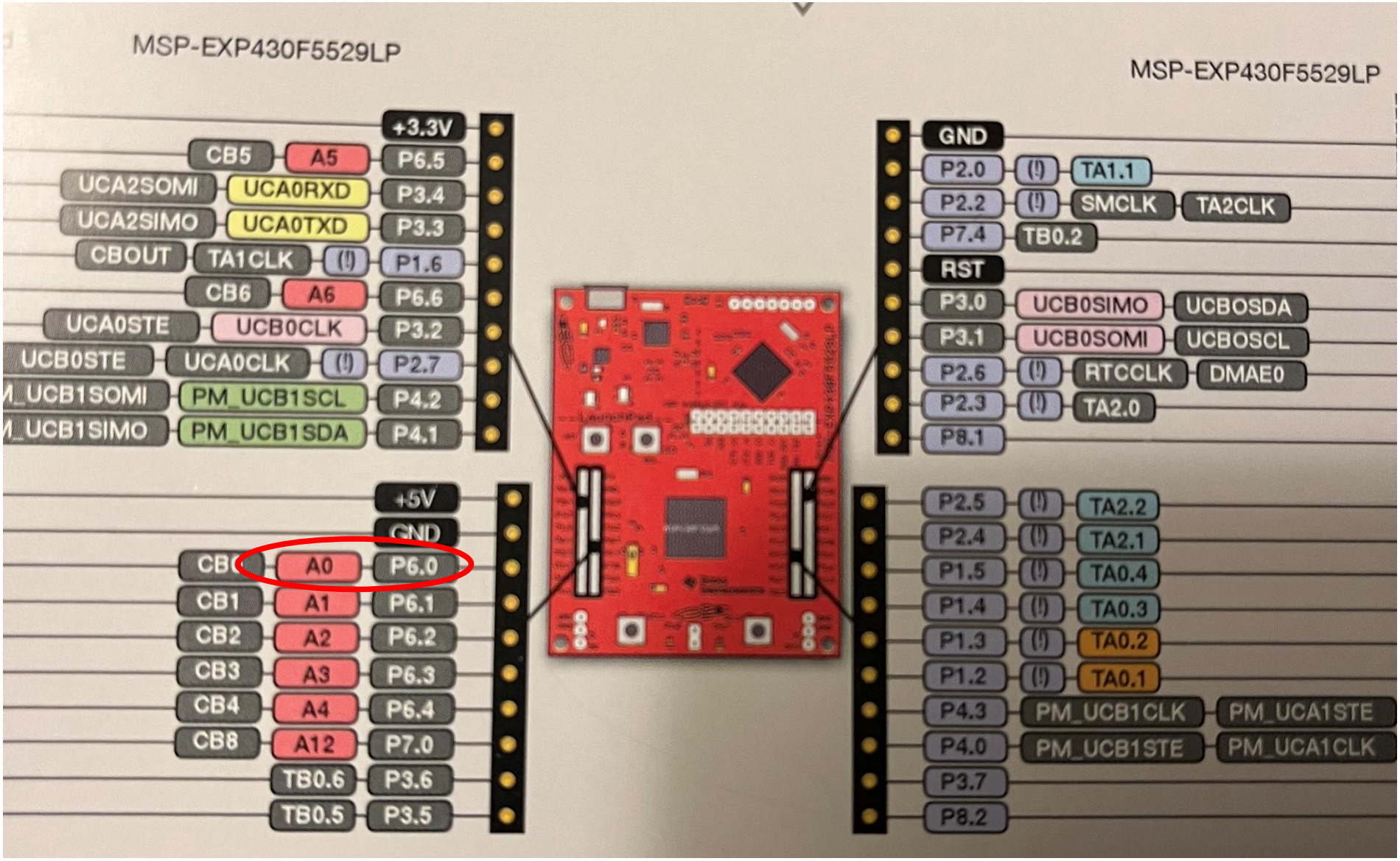
```
#include <msp430f5529.h>
#include<stdio.h>

int main(void)
{
  WDTCTL = WDTPW + WDTHOLD;
  ADC12CTL0 = ADC12SHT02 + ADC12ON;       // Sampling time 16 cycles, ADC12 on
  ADC12CTL1 = ADC12SHP;                   // sampling timer
  ADC12CTL0 |= ADC12ENC;                  // ADC enable
  P6SEL |= 0b00000001;                    // P6.0 allow ADC on pin 6.0
  P1DIR |= 0b00000001;                    // set pin P1.0 as output

  while (1)
  {

    ADC12CTL0 |= ADC12SC;                 // Start sampling
    while (ADC12CTL1 & ADC12BUSY);//while bit ADC12BUSY in register ADC12CTL1 is high wait

    if(ADC12MEM0>=3072)    //This value depends on the input voltage
       P1OUT |= BIT0;
    else
      P1OUT &= ~BIT0;
  }
```



A ADC12OSC refers to the MODCLK from the UCS. See the UCS chapter for more information.

B See the device-specific data sheet for timer sources available.

Figure 28-1. ADC12_A Block Diagram (Devices With REF Module)


MSP-EXP430F5529LP
MSP-EXP430F5529LP
+3.3V
CB5 A5 P6.5
UCA2SOMI UCA0RXD P3.4
UCA2SIMO UCA0TXD P3.3
CBOUT TA1CLK (!) P1.6
CB6 A6 P6.6
UCA0STE UCB0CLK P3.2
UCB0STE UCA0CLK (!) P2.7
M_UCB1SOMI PM_UCB1SCL P4.2
M_UCB1SIMO PM_UCB1SDA P4.1
+5V
GND
CB A0 P6.0
CB1 A1 P6.1
CB2 A2 P6.2
CB3 A3 P6.3
CB4 A4 P6.4
CB8 A12 P7.0
TB0.6 P3.6
TB0.5 P3.5
GND
P2.0 (!) TA1.1
P2.2 (!) SMCLK TA2CLK
P7.4 TB0.2
RST
P3.0 UCB0SIMO UCB0SDA
P3.1 UCB0SOMI UCB0SCL
P2.6 (!) RTCCLK DMAE0
P2.3 (!) TA2.0
P8.1
P2.5 (!) TA2.2
P2.4 (!) TA2.1
P1.5 (!) TA0.4
P1.4 (!) TA0.3
P1.3 (!) TA0.2
P1.2 (!) TA0.1
P4.3 PM_UCB1CLK PM_UCA1STE
P4.0 PM_UCB1STE PM_UCA1CLK
P3.7
P8.2

ADC12CTL0 Register

ADC12_A Control Register 0

Figure 28-13. ADC12CTL0 Register

15	14	13	12	11	10	9	8
ADC12SHT1x				ADC12SHT0x			
(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
2MSC	ADC12REF2_5V	ADC12REFON	ADC12ON	ADC12OVIE	ADC12TOVIE	ADC12ENC	ADC12SC
(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Can be modified only when ADC12ENC = 0

Table 28-4. ADC12CTL0 Register Description

	Field	Type	Reset	Description
	ADC12SHT1x	RW	0h	ADC12_A sample-and-hold time. These bits define the number of ADC12CLK cycles in the sampling period for registers ADC12MEM8 to ADC12MEM15.
	ADC12SHT0x	RW	0h	ADC12_A sample-and-hold time. These bits define the number of ADC12CLK cycles in the sampling period for registers ADC12MEM0 to ADC12MEM7. 0000b = 4 ADC12CLK cycles 0001b = 8 ADC12CLK cycles 0010b = 16 ADC12CLK cycles 0011b = 32 ADC12CLK cycles 0100b = 64 ADC12CLK cycles 0101b = 96 ADC12CLK cycles 0110b = 128 ADC12CLK cycles 0111b = 192 ADC12CLK cycles 1000b = 256 ADC12CLK cycles 1001b = 384 ADC12CLK cycles 1010b = 512 ADC12CLK cycles 1011b = 768 ADC12CLK cycles 1100b = 1024 ADC12CLK cycles 1101b = 1024 ADC12CLK cycles 1110b = 1024 ADC12CLK cycles 1111b = 1024 ADC12CLK cycles
	ADC12MSC	RW	0h	ADC12_A multiple sample and conversion. Valid only for sequence or repeated modes. 0b = The sampling timer requires a rising edge of the SHI signal to trigger each sample-and-convert. 1b = The first rising edge of the SHI signal triggers the sampling timer, but further sample-and-conversions are performed automatically as soon as the prior conversion is completed.
	ADC12REF2_5V	RW	0h	ADC12_A reference generator voltage. ADC12REFON must also be set. In devices with the REF module, this bit is only valid if the REFMSTR bit of the REF module is set to 0. In the F54xx devices (non-A), the REF module is not available. 0b = 1.5 V 1b = 2.5 V
	ADC12REFON	RW	0h	ADC12_A reference generator on. In devices with the REF module, this bit is only valid if the REFMSTR bit of the REF module is set to 0. In the F54xx devices (non-A), the REF module is not available. 0b = Reference off 1b = Reference on
	ADC12ON	RW	0h	ADC12_A on 0b = ADC12_A off 1b = ADC12_A on

Using peripherals: Analog to digital converter

READING THE DATASHEET of the family IS ESS

```
#include <msp430f5529.h>
#include<stdio.h>

int main(void)
{
  WDTCTL = WDTPW + WDTHOLD;
  ADC12CTL0 = ADC12SHT02 + ADC12ON;       // Sampling time 16 cycles, ADC12 on
  ADC12CTL1 = ADC12SHP;                   // sampling timer
  P6SEL |= 0b00000001;;                     // P6.0 allow ADC on pin 6.0
  P1DIR |= 0b00000001;;                     // set pin P1.0 as output
  ADC12CTL0 |= ADC12ENC;                  // ADC enable
  while (1)
  {

    ADC12CTL0 |= ADC12SC;                 // Start sampling
    while (ADC12CTL1 & ADC12BUSY);//while bit ADC12BUSY in register ADC12CTL1 is high wait

    if(ADC12MEM0>=3072)    //This value depends on the input voltage
      P1OUT |= BIT0;
    else
     P1OUT &= ~BIT0;
  }
}
```

Table 28-4. ADC12CTL0 Register Description (continued)

Bit	Field	Type	Reset	Description
3	ADC12OVIE	RW	0h	ADC12MEMx overflow-interrupt enable. The GIE bit must al the interrupt. 0b = Overflow interrupt disabled 1b = Overflow interrupt enabled
2	ADC12TOVIE	RW	0h	ADC12_A conversion-time-overflow interrupt enable. The GI set to enable the interrupt. 0b = Conversion time overflow interrupt disabled 1b = Conversion time overflow interrupt enabled
1	ADC12ENC	RW	0h	ADC12_A enable conversion 0b = ADC12_A disabled 1b = ADC12_A enabled
0	ADC12SC	RW	0h	ADC12_A start conversion. Software-controlled sample-and- ADC12SC and ADC12ENC may be set together with one ins is reset automatically. 0b = No sample-and-conversion-start 1b = Start sample-and-conversion

Using peripherals: Analog to digital converter

READING THE DATASHEET of the family IS ESS

```
#include <msp430f5529.h>
#include<stdio.h>

int main(void)
{
  WDTCTL = WDTPW + WDTHOLD;
  ADC12CTL0 = ADC12SHT02 + ADC12ON;       // Sampling time 16 cycles, ADC12 on
  ADC12CTL1 = ADC12SHP;                   // sampling timer

  P6SEL |= 0b00000001;;                    // P6.0 allow ADC on pin 6.0
  P1DIR |= 0b00000001;                     // set pin P1.0 as output
 ADC12CTL0 |= ADC12ENC;                   // ADC enable
  while (1)
  {

    ADC12CTL0 |= ADC12SC;                 // Start sampling
    while (ADC12CTL1 & ADC12BUSY);//while bit ADC12BUSY in register ADC12CTL1 is high wait

    if(ADC12MEM0>=3072)    //This value depends on the input voltage
       P1OUT |= BIT0;
    else
      P1OUT &= ~BIT0;
  }
}
```

Table 28-5. ADC12CTL1 Register Description

Bit	Field	Type	Reset	Description
15-12	ADC12CSTARTADDx	RW	0h	ADC12_A conversion start address. These bits select which ADC12_A conversion-memory register is used for a single conversion or for the first conversion in a sequence. The value of CSTARTADDx is 0 to 0Fh, corresponding to ADC12MEM0 to ADC12MEM15.
11-10	ADC12SHSx	RW	0h	ADC12_A sample-and-hold source select 00b = ADC12SC bit 01b = Timer source (see device-specific data sheet for exact timer and locations) 10b = Timer source (see device-specific data sheet for exact timer and locations) 11b = Timer source (see device-specific data sheet for exact timer and locations)
9	ADC12SHP	RW	0h	ADC12_A sample-and-hold pulse-mode select. This bit selects the source of the sampling signal (SAMPCON) to be either the output of the sampling timer or the sample-input signal directly. 0b = SAMPCON signal is sourced from the sample-input signal. 1b = SAMPCON signal is sourced from the sampling timer.
8	ADC12ISSH	RW	0h	ADC12_A invert signal sample-and-hold 0b = The sample-input signal is not inverted. 1b = The sample-input signal is inverted.
7-5	ADC12DIVx	RW	0h	ADC12_A clock divider 000b = Divide by 1 001b = Divide by 2 010b = Divide by 3 011b = Divide by 4 100b = Divide by 5 101b = Divide by 6 110b = Divide by 7 111b = Divide by 8
4-3	ADC12SSELx	RW	0h	ADC12_A clock source select 00b = ADC12OSC (MODCLK) 01b = ACLK 10b = MCLK 11b = SMCLK
2-1	ADC12CONSEQx	RW	0h	ADC12_A conversion sequence mode select 00b = Single-channel, single-conversion 01b = Sequence-of-channels 10b = Repeat-single-channel 11b = Repeat-sequence-of-channels
0	ADC12BUSY	R	0h	ADC12_A busy. This bit indicates an active sample or conversion operation. 0b = No operation is active. 1b = A sequence, sample, or conversion is active.

Using peripherals: Analog to digital converter

READING THE DATASHEET of the family IS ESS

```
#include <msp430f5529.h>
#include<stdio.h>

int main(void)
{
  WDTCTL = WDTPW + WDTHOLD;
  ADC12CTL0 = ADC12SHT02 + ADC12ON;       // Sampling time 16 cycles, ADC12 on
  ADC12CTL1 = ADC12SHP;                   // sampling timer
  P6SEL |= 0b00000001;                     // P6.0 allow ADC on pin 6.0
  P1DIR |= 0b00000001;                     // set pin P1.0 as output
 ADC12CTL0 |= ADC12ENC;                   // ADC enable
  while (1)
  {

    ADC12CTL0 |= ADC12SC;                 // Start sampling
    while (ADC12CTL1 & ADC12BUSY);//while bit ADC12BUSY in register ADC12CTL1 is high wait

    if(ADC12MEM0>=3072)    //This value depends on the input voltage
      P1OUT |= BIT0;
    else
      P1OUT &= ~BIT0;
  }
}
```

28.3.5 ADC12MCTLx Register

ADC12_A Conversion Memory Control Register

Figure 28-17. ADC12MCTLx Register

7	6	5	4	3	2	1	0
ADC12EOS	ADC12SREFx			ADC12INCHx			
rw	rw	rw	rw	rw	rw	rw	rw

Can be modified only when ADC12ENC = 0

Table 28-8. ADC12MCTLx Register Description

Bit	Field	Type	Reset	Description
7	ADC12EOS	RW	0h	End of sequence. Indicates the last conversion in a sequence. 0b = Not end of sequence 1b = End of sequence
6-4	ADC12SREFx	RW	0h	Select reference 000b = V_{R+} = AVCC and V_{R-} = AVSS 001b = V_{R+} = VREF+ and V_{R-} = AVSS 010b = V_{R+} = VeREF+ and V_{R-} = AVSS 011b = V_{R+} = VeREF+ and V_{R-} = AVSS 100b = V_{R+} = AVCC and V_{R-} = VREF-/VeREF- 101b = V_{R+} = VREF+ and V_{R-} = VREF-/VeREF- 110b = V_{R+} = VeREF+ and V_{R-} = VREF-/VeREF- 111b = V_{R+} = VeREF+ and V_{R-} = VREF-/VeREF-
3-0	ADC12INCHx	RW	0h	Input channel select 0000b = A0 0001b = A1 0010b = A2 0011b = A3 0100b = A4 0101b = A5 0110b = A6 0111b = A7 1000b = VeREF+ 1001b = VREF-/VeREF- 1010b = Temperature diode 1011b = (AVCC – AVSS) / 2 1100b = A12. On devices with the Battery Backup System, VBAT can be measured internally by the ADC. 1101b = A13 1110b = A14 1111b = A15

Notice that there are 16 control registers like that corresponding to 16 ADC12MEMx memory registers where the conversion results appear

```
while (ADC12CTL1 & ADC12BUSY);//while bit ADC12BUSY in register ADC12CTL1 is high wait

   if(ADC12MEM0>=3072)    //This value depends on the input voltage
     P1OUT |= BIT0;
   else
    P1OUT &= ~BIT0;
 }
```

```
while (a & b){

}
```

As compared to:

```
while(a & b); {


}
```

Activity 3 due before each lab in week 5 of the labs. 2 points

The result of ADC conversion on MSP430 is a 12 bit binary number corresponding to voltage range 0-3.3 V.

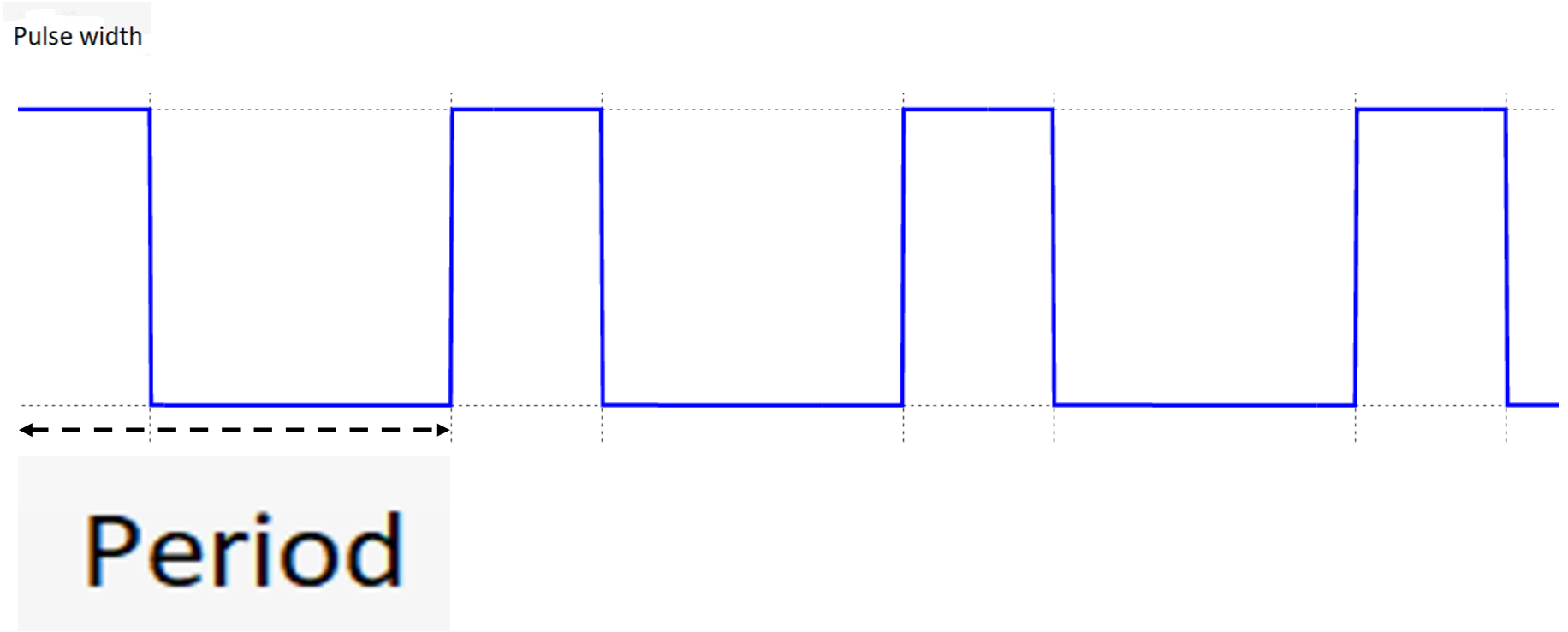
1. What is the accuracy (resolution) of the conversion in volts?

2. What command or commands will you use to convert this 12 bit number to 8 bit number without loosing the 0-3.3 V range.

3. How such a conversion affects the accuracy (resolution) in volts.

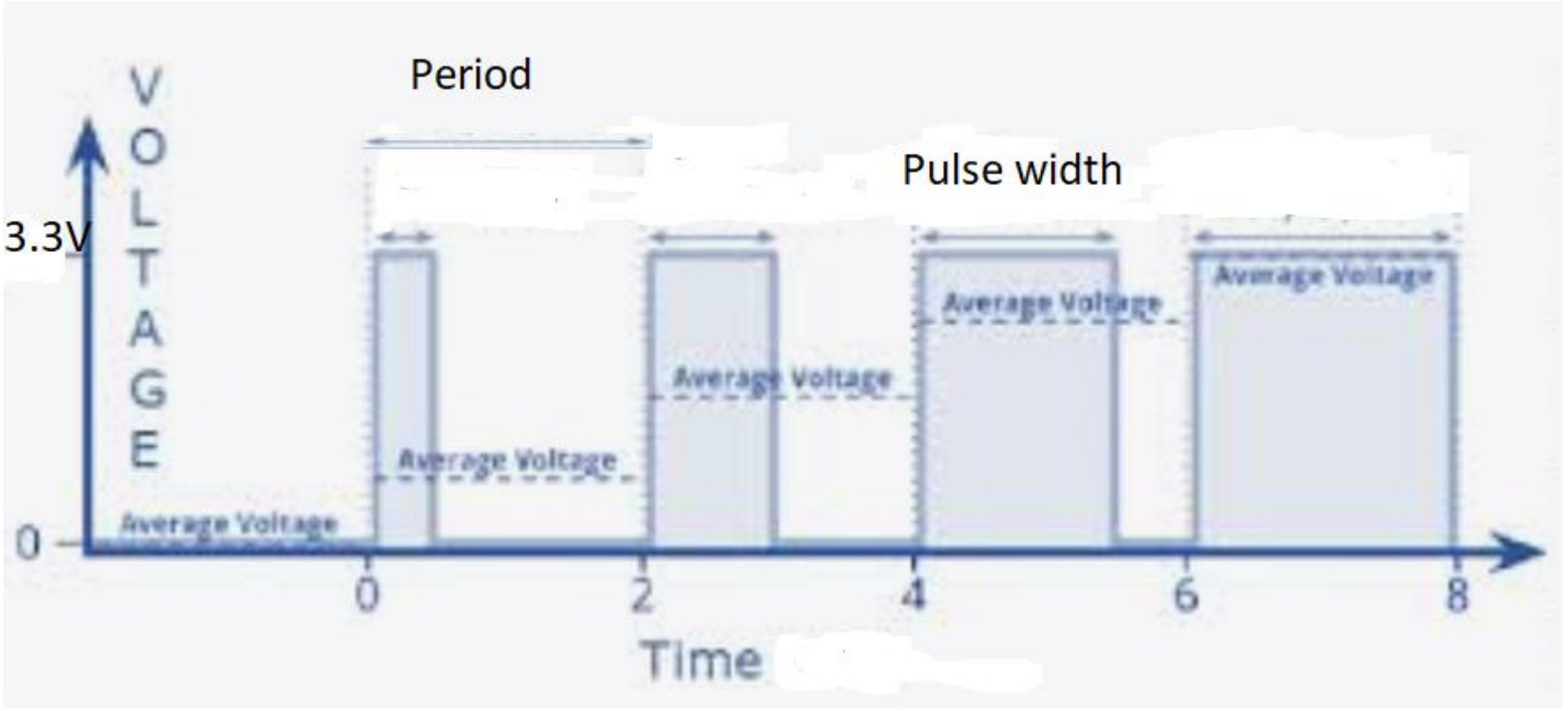
Pulse Width Modulation (PWM)

Full range square wave like this one



Pulse Width Modulation (PWM)

Controlling power with PWM



Duty cycle of PWM

Ratio of pulse width to period usually expressed in %. Sometimes the pulse width is referred as duty cycle.

Pulse Width Modulation (PWM)

Applications:

Power control without energy loss. (Compare with voltage divider).

Controlling the servos.

Pulse Width Modulation (PWM)

PWM outputs on MSP430F5529.

P1.2

P1.3

P1.4

P1.5

P2.4

P2.5

Pulse Width Modulation (PWM)

```
#include <msp430.h>
int i = 0;
void main(void){ WDTCTL = WDTPW|WDTHOLD; // Stop WDT
 P1DIR |= BIT2;       // Output on Pin 1.2
 P1SEL |= BIT2;       // Pin 1.2 selected as PWM
 TA0CCR0 = 500;     // PWM period 500/1.048 microseconds
 TA0CCR1 = 50;       // PWM duty cycle
 TA0CCTL1 = OUTMOD_7; // TA0CCR1 reset/set-high voltage
                                    // below count, low voltage when past
 TA0CTL = TASSEL_2 + MC_1 + ID_0 + TAIE;
 // Timer A control set to SMCLK, count up mode MC_1, no division
  //- keep 1.048 MHz, enable interrupt
 _bis_SR_register(LPM0_bits); // Enter Low power mode 0
}
```

One needs to read a description in

Microprocessor family

Names of Registers and Bits

They are all in include file "msp430.h"

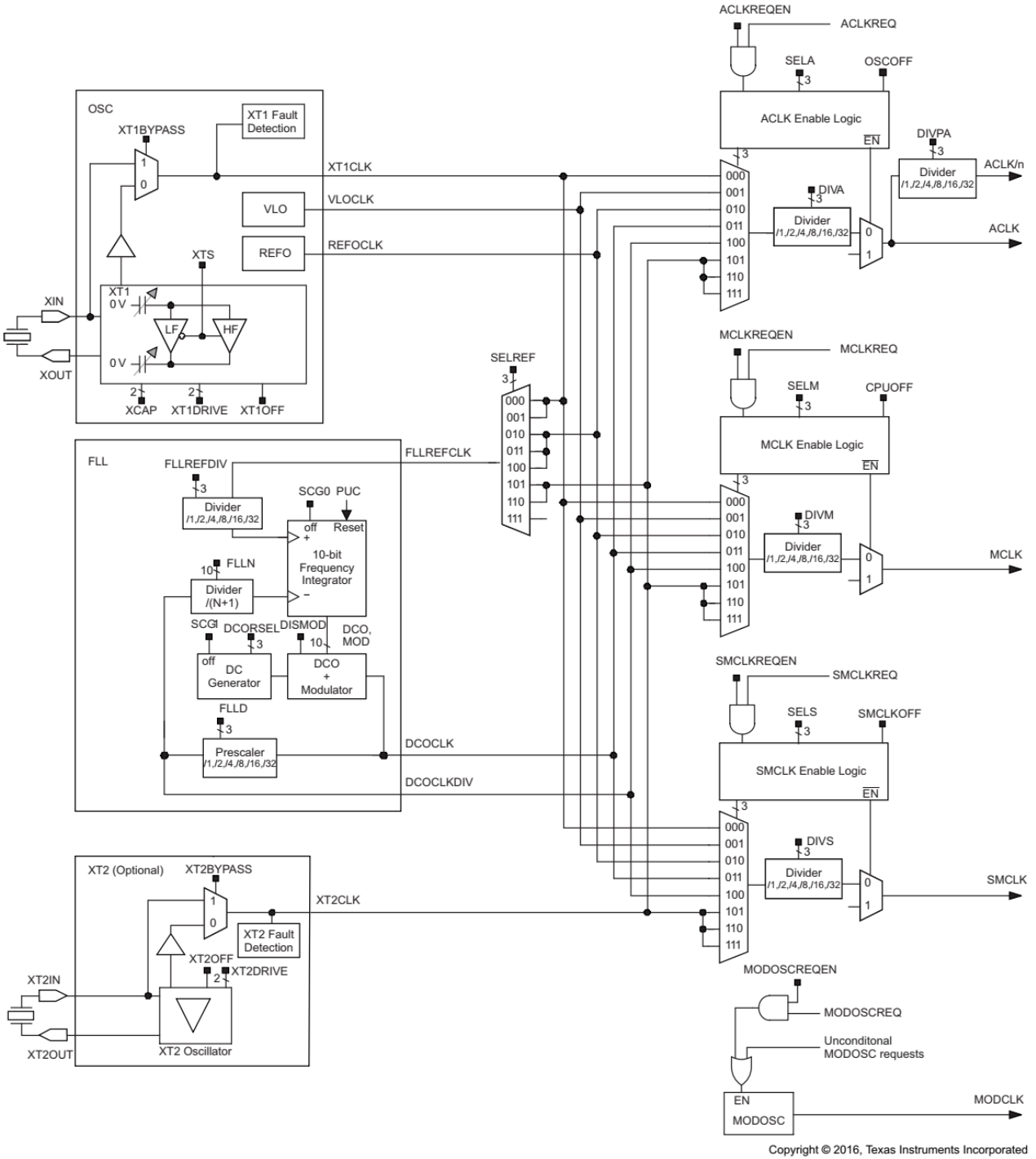


Figure 5-1. UCS Block Diagram

17.3.1 TAxCTL Register

Timer_Ax Control Register

Figure 17-16. TAxCTL Register

15	14	13	12	11	10	9	8
Reserved						TASSEL	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ID		MC		Reserved	TACLR	TAIE	TAIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	w-(0)	rw-(0)	rw-(0)

Table 17-4. TAxCTL Register Description

Bit	Field	Type	Reset	Description
15-10	Reserved	RW	0h	Reserved
9-8	TASSEL	RW	0h	Timer_A clock source select 00b = TAxCLK 01b = ACLK 10b = SMCLK 11b = INCLK
7-6	ID	RW	0h	Input divider. These bits along with the TAIDEX bits select the divider for the input clock. 00b = /1 01b = /2 10b = /4 11b = /8
5-4	MC	RW	0h	Mode control. Setting MC = 00h when Timer_A is not in use conserves power. 00b = Stop mode: Timer is halted 01b = Up mode: Timer counts up to TAxCCR0 10b = Continuous mode: Timer counts up to 0FFFFh 11b = Up/down mode: Timer counts up to TAxCCR0 then down to 0000h
3	Reserved	RW	0h	Reserved
2	TACLR	RW	0h	Timer_A clear. Setting this bit clears TAR, the clock divider logic (the divider setting remains unchanged), and the count direction. The TACLR bit is automatically reset and is always read as zero.
1	TAIE	RW	0h	Timer_A interrupt enable. This bit enables the TAIFG interrupt request. 0b = Interrupt disabled 1b = Interrupt enabled
0	TAIFG	RW	0h	Timer_A interrupt flag 0b = No interrupt pending

17.3.3 TAxCCTLn Register

Timer_Ax Capture/Compare Control n Register

Figure 17-18. TAxCCTLn Register

15	14	13	12	11	10	9	8
CM		CCIS		SCS	SCCI	Reserved	CAP
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r-(0)	r-(0)	rw-(0)
7	6	5	4	3	2	1	0
OUTMOD			CCIE	CCI	OUT	COV	CCIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r	rw-(0)	rw-(0)	rw-(0)

Table 17-6. TAxCCTLn Register Description

Bit	Field	Type	Reset	Description
15-14	CM	RW	0h	Capture mode 00b = No capture 01b = Capture on rising edge 10b = Capture on falling edge 11b = Capture on both rising and falling edges
13-12	CCIS	RW	0h	Capture/compare input select. These bits select the TAxCCR0 input signal. See the device-specific data sheet for specific signal connections. 00b = CCIxA 01b = CCIxB 10b = GND 11b = VCC
11	SCS	RW	0h	Synchronize capture source. This bit is used to synchronize the capture input signal with the timer clock. 0b = Asynchronous capture 1b = Synchronous capture
10	SCCI	RW	0h	Synchronized capture/compare input. The selected CCI input signal is latched with the EQUx signal and can be read from this bit.
9	Reserved	R	0h	Reserved. Reads as 0.
8	CAP	RW	0h	Capture mode 0b = Compare mode 1b = Capture mode
7-5	OUTMOD	RW	0h	Output mode. Modes 2, 3, 6, and 7 are not useful for TAxCCR0 because EQUx = EQU0. 000b = OUT bit value 001b = Set 010b = Toggle/reset 011b = Set/reset 100b = Toggle 101b = Reset 110b = Toggle/set 111b = Reset/set
4	CCIE	RW	0h	Capture/compare interrupt enable. This bit enables the interrupt request of the corresponding CCIFG flag. 0b = Interrupt disabled 1b = Interrupt enabled
3	CCI	R	0h	Capture/compare input. The selected input signal can be read by this bit.
2	OUT	RW	0h	Output. For output mode 0, this bit directly controls the state of the output. 0b = Output low 1b = Output high

12.2.5.2 Output Example — Timer in Up Mode

The OUTx signal is changed when the timer *counts* up to the TA0CCRx value, and rolls from TA0CCR0 to zero, depending on the output mode. An example is shown in Figure 12-12 using TA0CCR0 and TA0CCR1

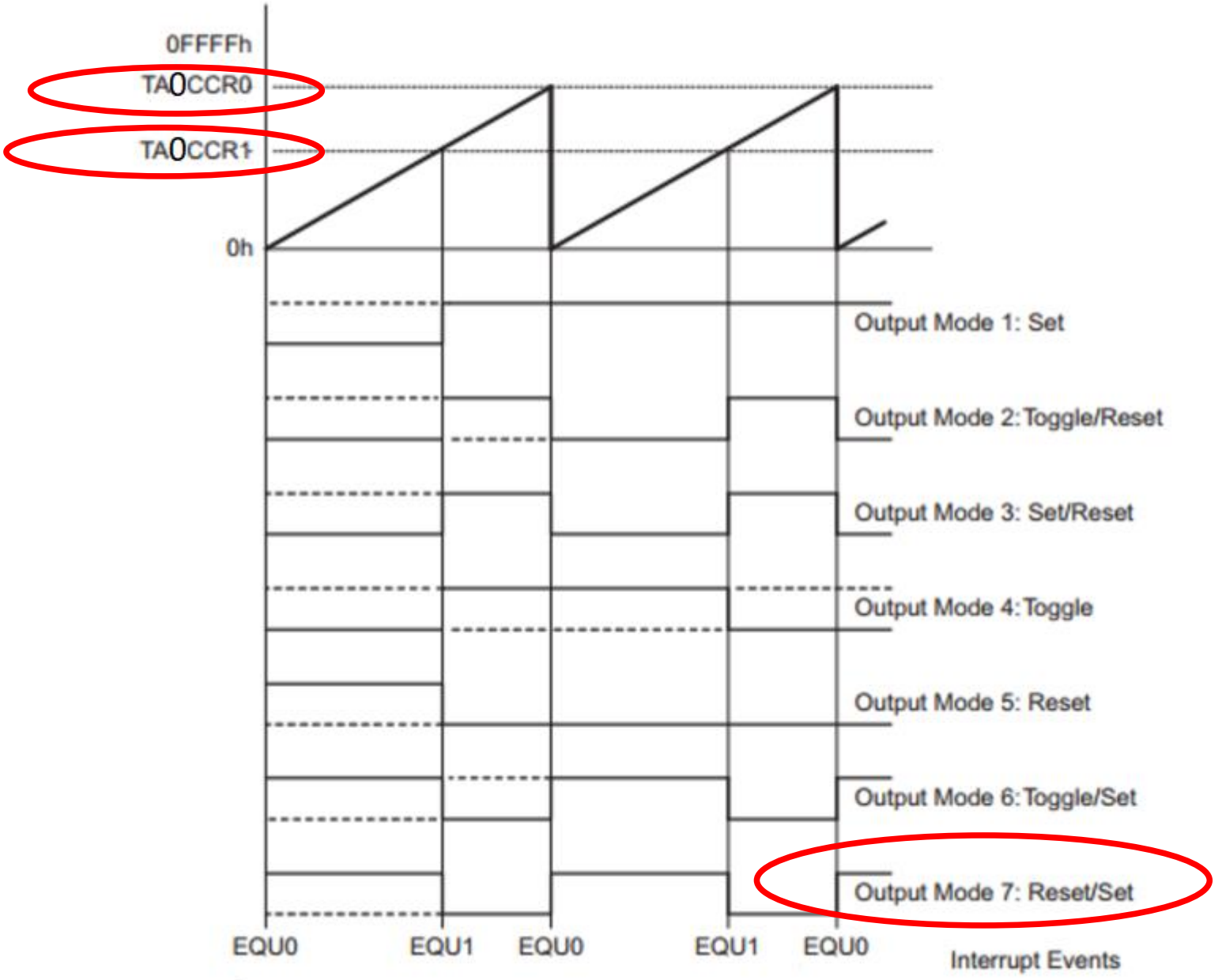


Figure 12-12. Output Example—Timer in Up Mode

Ultrasonic distance sensor

Idea comes from bats and dolphins

- Send a pulse (chirp) of high frequency sound
- Wait for the echo
- Estimate or calculate distance from a time between the send time and return time

Radars work on the same principle but with electromagnetic waves.

LADAR (LIDAR)

Connecting the ultrasonic distance sensor to test it:

5V

Pulse from Generator

Oscilloscope

Pulse amplitude should be 2V, offset 0 (no negative part), short with frequency allowing echo to return from a long distance (say 15 m) before the next pulse comes in.

You should see the width of the echo pulse on the oscilloscope changing when you move an object (for example box) farther or closer to the sensor.

Distance measurement with real-time display on the computer.
Program should:

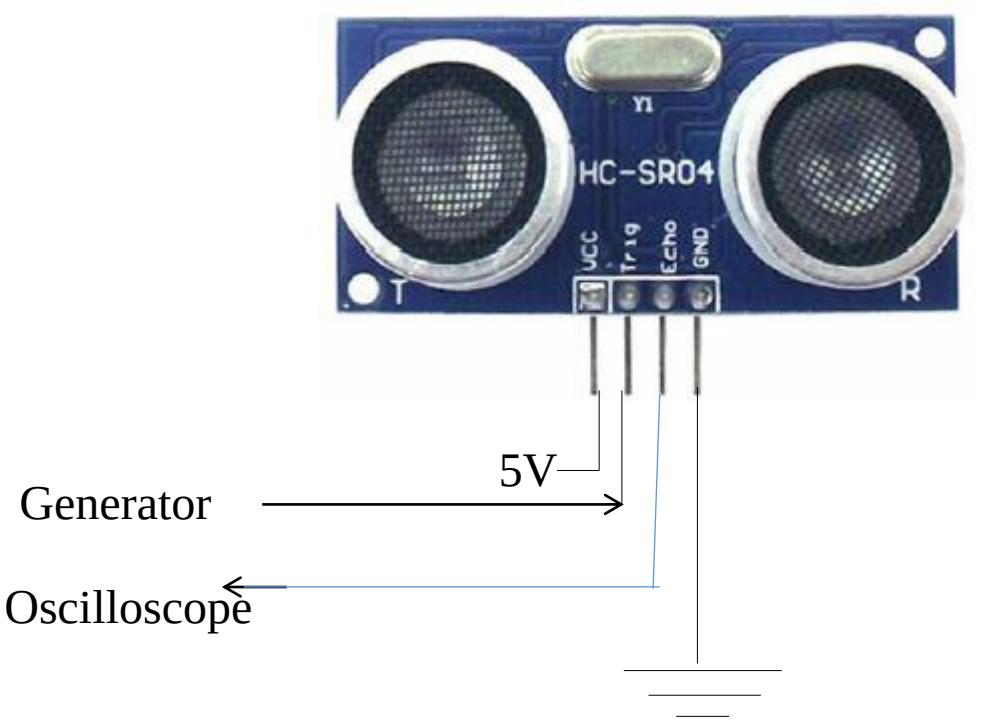
1. Trigger the ultrasonic distance measurement sensor to begin a measurement.

2. Do nothing till the “echo” pin goes high

3. Time the interval while the echo pin is high

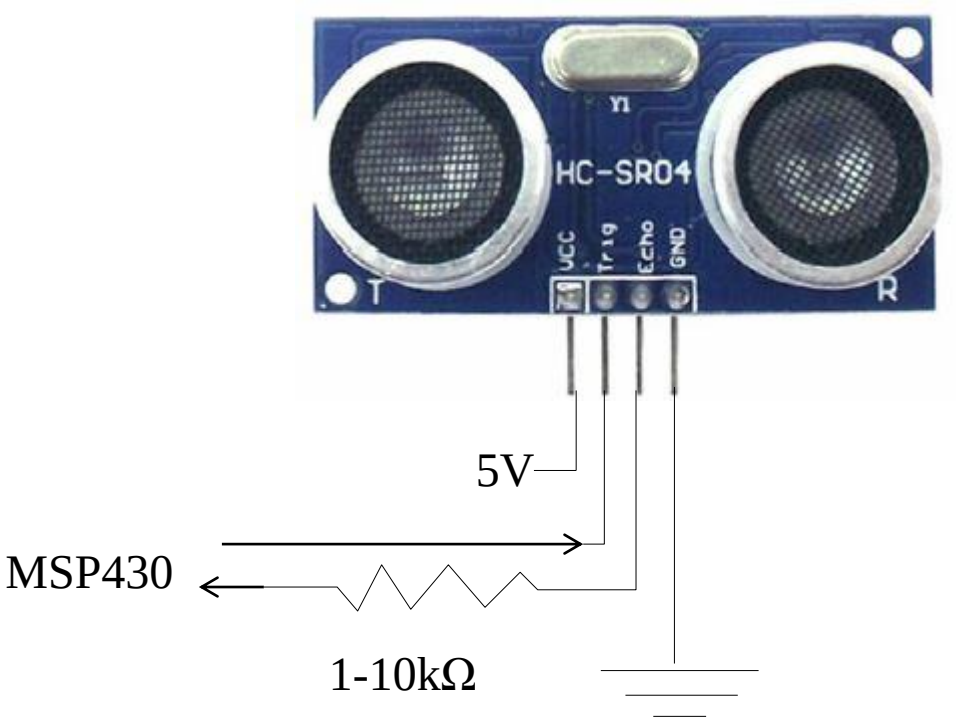
4. Transmit the time interval through the serial port to the host computer where a python program will receive it and print or plot it.

Connecting the ultrasonic distance sensor to Hantek for testing:



The SR04 is a 5V device! To protect the MSP430 input from it, you should put a resistor in between the 'Echo' line and the MSP430. 3.3 V pulse is high enough to trigger it.

Connecting the ultrasonic distance sensor:



The SR04 is a 5V device! To protect the MSP430 input from it, you should put a resistor in between the 'Echo' line and the MSP430. 3.3 V pulse is high enough to trigger it.

Python and GUI's

For part 8 you will need Python installed and example temperature programs tested.

Python is a “high-level language” (complicated things are often easier)

It is interpreted, not compiled which make it faster to change program but the execution is slower and it take more memory.

It has the same format on windows, mac and linux

Remember that in python white spaces and indenting matters (unlike C where it's just for readability)

•