Interfacing ADC with msp430

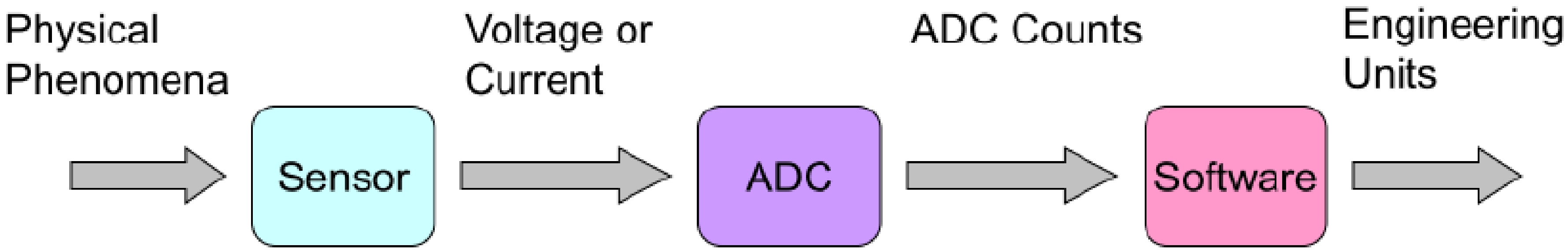
Engr. Shahzada Fahim Jan

- We live in an analog world
- Temperature, humidity, pressure, are analog.
- We use transducers to convert physical quantity to electrical quantity such as voltage or current.
- For interfacing these sensors to microcontrollers we require to convert the analog output of these sensors to digital so that the controller can read it. Some microcontrollers have built-in Analog to Digital Convertor (ADC) so there is no need for external ADC. For microcontrollers that don’t have internal ADC external ADC is used.

Going from analog to digital

- What we want

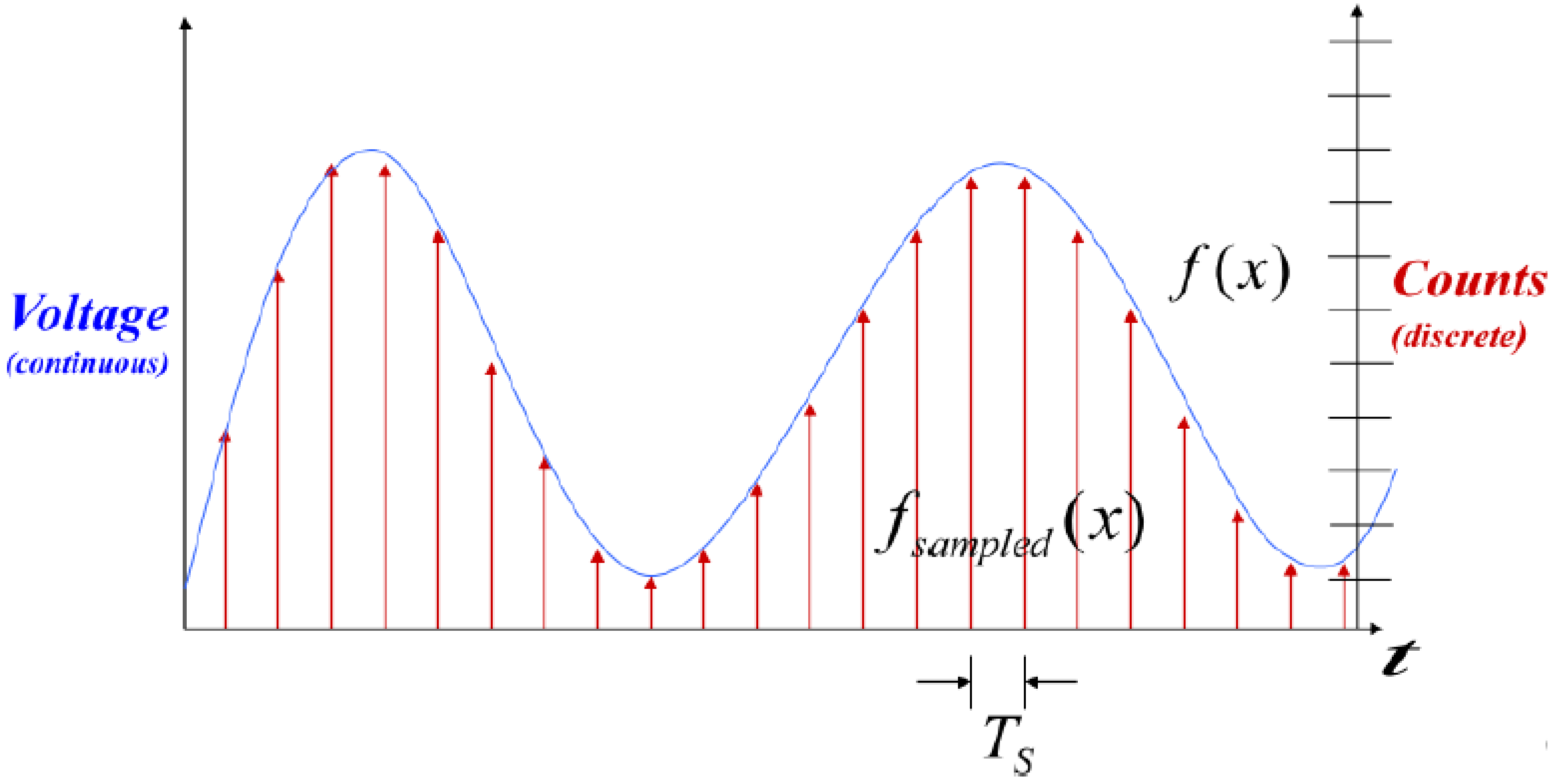
- How we have to get there



Representing an analog signal digitally

- How do we represent an analog signal (e.g. continuous voltage)?
  - As a time series of discrete values

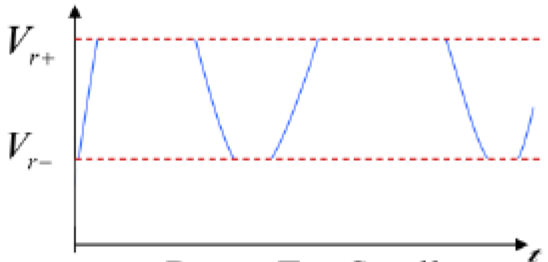
    → On MCU: read ADC data register (counts) periodically (T_s)



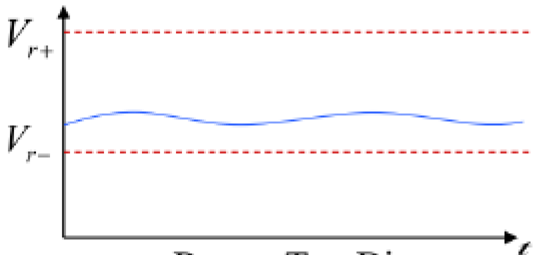
Choosing the range

- Fixed # of bits (e.g. 8-bit ADC)
- Span a particular input voltage range
- What do the sample values represent?
  - Some fraction within the range of values

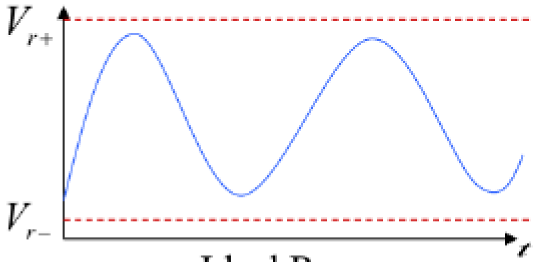
    → What range to use?



Range Too Small



Range Too Big



Ideal Range

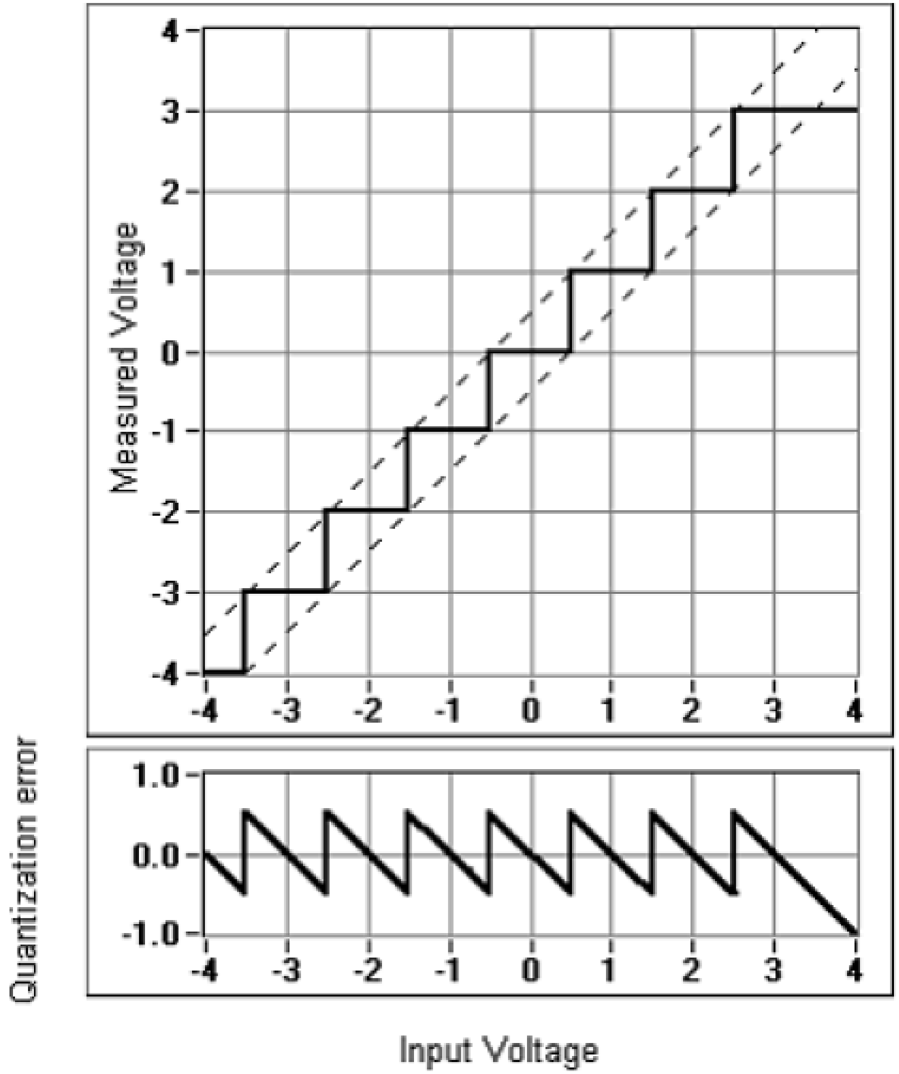
Choosing the granularity

- Resolution
  - Number of discrete values that represent a range of analog values
  - MSP430: 12-bit ADC
    - 4096 values
    - Range / 4096 = Step

  Larger range ➔ less info / bit

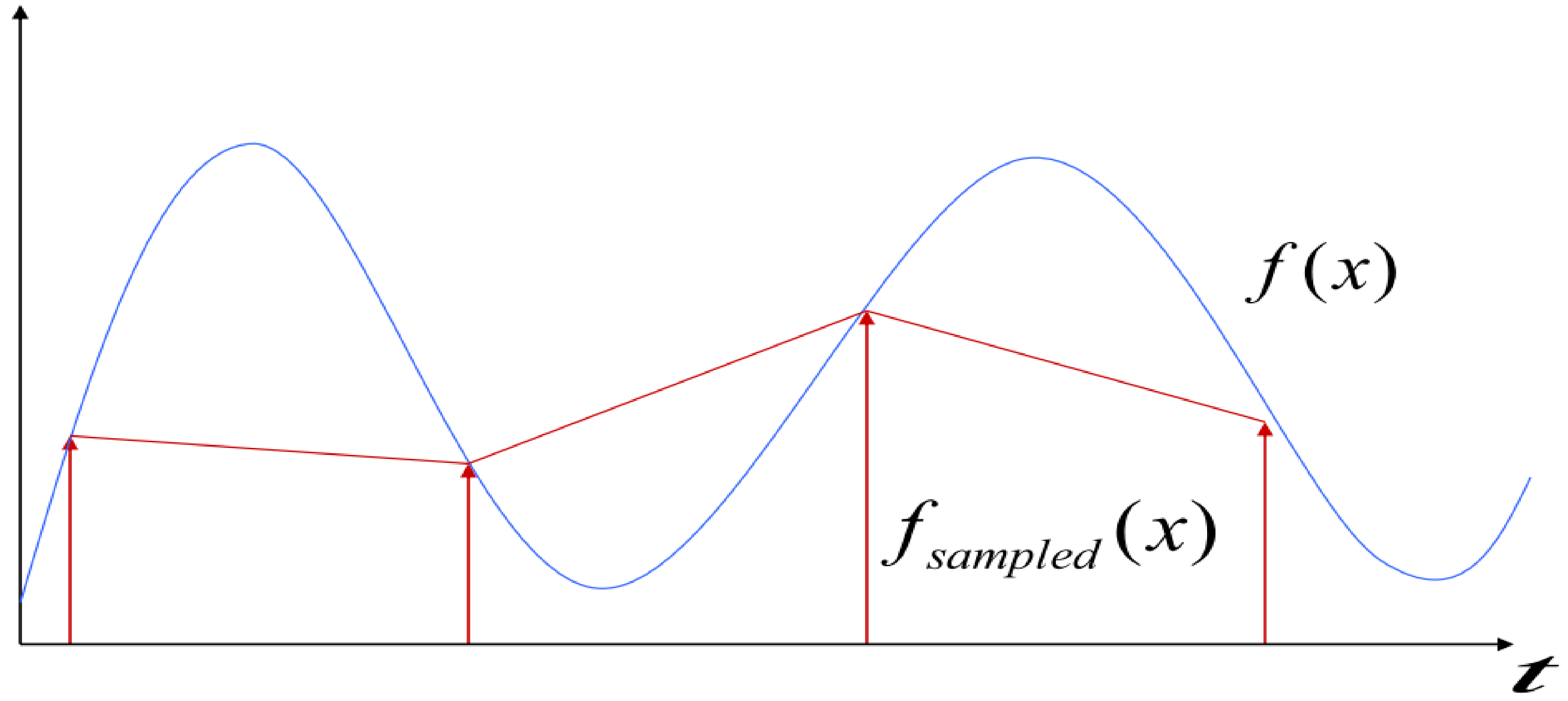
- Quantization Error
  - How far off discrete value is from actual
  - ½ LSB → Range / 8192

  Larger range ➔ larger error



Choosing the sample rate

- What sample rate do we need?
  - Too little: we can’t reconstruct the signal we care about
  - Too much: waste computation, energy, resources



Shannon-Nyquist sampling theorem

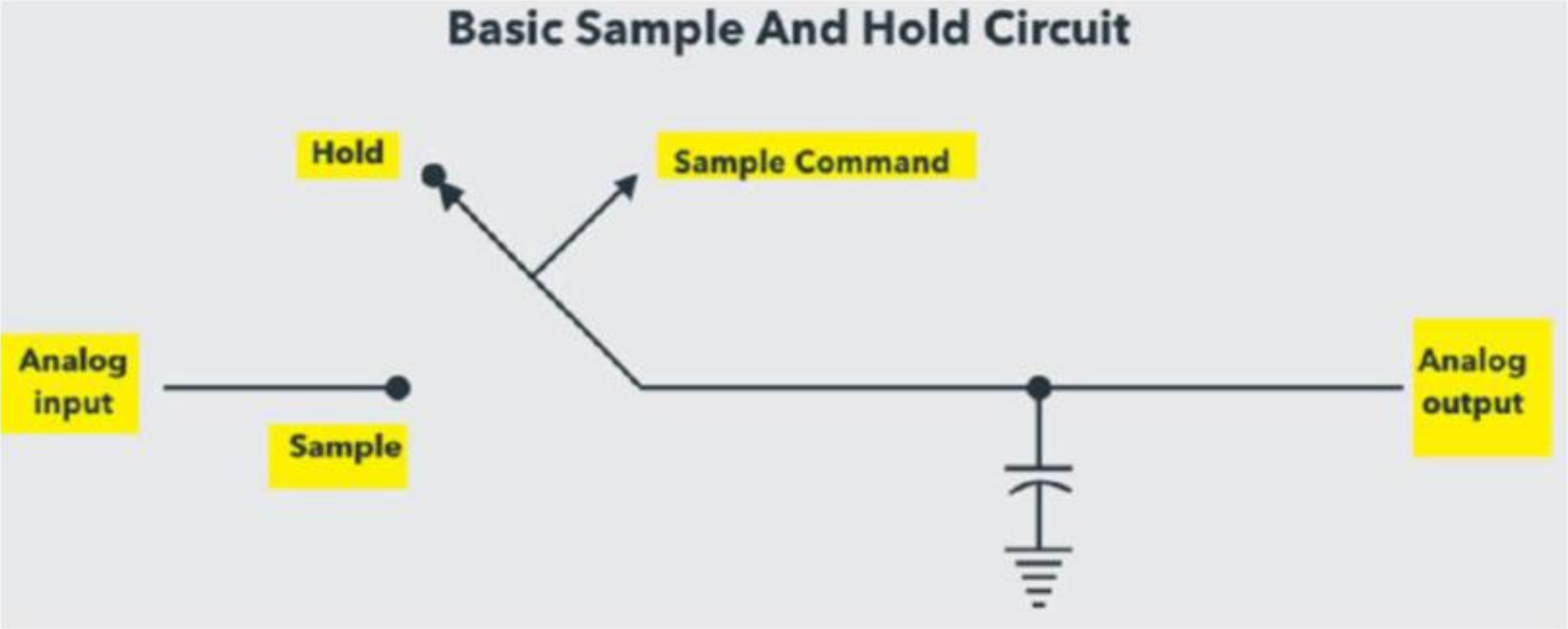
- *If a continuous-time signal $f(x)$ contains no frequencies higher than $f_{\max}$, it can be completely determined by discrete samples taken at a rate:*

$$f_{\text{samples}} > 2 f_{\max}$$

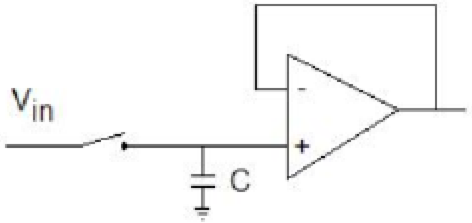
- Example:
  - Humans can process audio signals 20 Hz – 20 KHz
  - Audio CDs: sampled at 44.1 KHz

Sample and Hold Circuits
If the input analog voltage of an ADC changes more than ±1/2 LSB, then there is a severe chance that the output digital value is an error.

For the ADC to produce accurate results, the input analog voltage should be held constant for the duration of the conversion.



Sample and Hold of ADC



- If the switch is left open, but momentarily close it when we want to grab a measurement, it is a sample and hold circuit.

21.3.1 *ADCCTL0 Register*

ADC Control Register 0

Figure 21-21. ADCCTL0 Register

15	14	13	12	11	10	9	8
Reserved				ADCSHTx			
r0	r0	r0	r0	rw-(0)	rw-(0)	rw-(0)	rw-(1)
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADCMSC	Reserved		ADCON	Reserved		ADCENC	ADCSC
rw-(0)	r0	r0	rw-(0)	r0	r0	rw-(0)	rw-(0)

Can be modified only when ADCENC = 0. Resetting ADCENC = 0 by software and changing these fields immediately shows an effect when a conversion is active.

7	ADCMSC	RW	0h	ADC multiple sample-and-conversion. Valid only for sequence or repeated modes. Can be modified only when ADCENC = 0. Resetting ADCENC = 0 by software and changing these fields immediately shows an effect when a conversion is active. 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.
6-5	Reserved	R	0h	Reserved. Always reads as 0.
4	ADCON	RW	0h	ADC on. Can be modified only when ADCENC = 0. Resetting ADCENC = 0 by software and changing these fields immediately shows an effect when a conversion is active. 0b = ADC off 1b = ADC on
3-2	Reserved	R	0h	Reserved. Always reads as 0.

21.3.2 ADCCTL1 Register

ADC Control Register 1

Figure 21-22. ADCCTL1 Register

15	14	13	12	11	10	9	8
Reserved				ADCSHSx		ADCSHP	ADCISSH
r0	r0	r0	r0	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADCDIVx			ADCSSELx		ADCCONSEQx		ADCBUSY
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r-(0)

Can be modified only when ADCENC = 0. Resetting ADCENC = 0 by software and changing these fields immediately shows an effect when a conversion is active.

Table 21-4. ADCCTL1 Register Description

Bit	Field	Type	Reset	Description
15-12	Reserved	R	0h	Reserved. Always reads as 0.
11-10	ADCSHSx	RW	0h	ADC sample-and-hold source select. Can be modified only when ADCENC = 0. Resetting ADCENC = 0 by software and changing these fields immediately shows an effect when a conversion is active. 00b = ADCSC bit 01b = Timer trigger 0 (see device-specific data sheet) 10b = Timer trigger 1 (see device-specific data sheet) 11b = Timer trigger 2 (see device-specific data sheet)
9	ADCSHP	RW	0h	ADC 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. Can be modified only when ADCENC = 0. Resetting ADCENC = 0 by software and changing these fields immediately shows an effect when a conversion is active. 0b = SAMPCON signal is sourced from the sample input signal. 1b = SAMPCON signal is sourced from the sampling timer.
2-1	ADCCONSEQx	RW	0h	ADC conversion sequence mode select. Can be modified only when ADCENC = 0. Resetting ADCENC = 0 by software and changing these fields immediately shows an effect when a conversion is active. 00b = Single-channel single-conversion 01b = Sequence-of-channels 10b = Repeat-single-channel 11b = Repeat-sequence-of-channels

21.3.6 ADCMCTL0 Register

ADC Conversion Memory Control Register

Figure 21-26. ADCMCTL0 Register

15	14	13	12	11	10	9	8
Reserved							Reserved
r0	r0	r0	r0	r0	r0	r0	rw-(0)
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	ADCSREFx			ADCINCHx			
r0	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Can be modified only when ADCENC = 0. Resetting ADCENC = 0 by software and changing these fields immediately shows an effect when a conversion is active.

				Reserved. Always reads as 0.
6-4	ADCSREFx	RW	0h	Select reference. It is not recommended to change this setting while a conversion is ongoing. Can be modified only when ADCENC = 0. Resetting ADCENC = 0 by software and changing these fields immediately shows an effect when a conversion is active. 000b = {V_{R+} = AVCC and V_{R-} = AVSS } 001b = {V_{R+} = VREF and V_{R-} = AVSS} 010b = {V_{R+} = VEREF+ buffered and V_{R-} = AVSS} 011b = {V_{R+} = VEREF+ and V_{R-} = AVSS } 100b = {V_{R+} = AVCC and V_{R-} = VEREF-} 101b = {V_{R+} = VREF and V_{R-} = VEREF-} 110b = {V_{R+} = VEREF+ buffered and V_{R-} = VEREF-} 111b = {V_{R+} = VEREF+ and V_{R-} = VEREF-}
3-0	ADCINCHx	RW	0h	Input channel select. Writing these bits select the channel for a single-conversion or the highest channel for a sequence of conversions. Reading these bits in ADCCONSEQ = 01,11 returns the channel currently converted. Can be modified only when ADCENC = 0. Resetting ADCENC = 0 by software and changing these fields immediately shows an effect when a conversion is active. 0000b = A0 (see device-specific data sheet) 0001b = A1 (see device-specific data sheet) 0010b = A2 (see device-specific data sheet) 0011b = A3 (see device-specific data sheet) 0100b = A4 (see device-specific data sheet) 0101b = A5 (see device-specific data sheet) 0110b = A6 (see device-specific data sheet) 0111b = A7 (see device-specific data sheet) 1000b = A8 (see device-specific data sheet) 1001b = A9 (see device-specific data sheet) 1010b = A10 (see device-specific data sheet) 1011b = A11 (see device-specific data sheet) 1100b = A12 (see device-specific data sheet) 1101b = A13 (see device-specific data sheet) 1110b = A14 (see device-specific data sheet) 1111b = A15 (see device-specific data sheet)

```c
P1DIR |= BIT0;                                      // Set P1.0 to output direction
P1OUT &= ~BIT0;                                     // Clear P1.0 LED

// Configure ADC A1 pin
SYSCFG2 |= ADCPCTL1;                                 // Enable A1 for ADC input

// Configure ADC
ADCCTL0 |= ADCON | ADCMSC;                          // Enable ADC, start conversion
ADCCTL1 |= ADCSHP | ADCSHS_2 | ADCCONSEQ_2;          // Repeated single channel, TA1.1 trig sample start
ADCCTL2 |= ADCRES;                                  // 10-bit conversion
ADCMCTL0 |= ADCINCH_1 | ADCSREF_1;                  // A1 ADC input select, Vref=1.5V
ADCIE |= ADCIE0;                                    // Enable ADC interrupt

// Configure Timer A1 for ADC triggering
TA1CCR0 = 1024 - 1;                                  // PWM Period
TA1CCR1 = 512 - 1;                                   // TA1.1 ADC trigger
TA1CCTL1 = OUTMOD_4;                                 // TA1CCR0 toggle
TA1CTL = TASSEL__ACLK | MC_1 | TACLR;                // ACLK, up mode

// Configure reference
PMMCTL0_H = PMMPW_H;                                // Unlock PMM registers
PMMCTL2 |= INTREFEN;                                // Enable internal reference
__delay_cycles(400);                                // Delay for reference settling

ADCCTL0 |= ADCENC;                                  // ADC Enable
```

__bis_SR_register(LPM0_bits | GIE);   // Enter low power mode with interrupts enabled}

```c
#pragma vector=ADC_VECTOR
__interrupt void ADC_ISR(void)
#elif defined(__GNUC__)
void __attribute__ ((interrupt(ADC_VECTOR))) ADC_ISR (void)
#else
#error Compiler not supported!
#endif
{
    switch(__even_in_range(ADCIV, ADCIV_ADCIFG))
    {
        case ADCIV_NONE:
            break;
        case ADCIV_ADCIFG:
            ADC_Result = ADCMEM0;                                    // Read ADC result
            displayNumber(ADC_Result);                                // Display result on LCD

            if (ADCMEM0 < 0x155)                                     // Check if ADC value is below 0.5V
                P1OUT &= ~BIT0;                                      // Turn off LED
            else
                P1OUT |= BIT0;                                       // Turn on LED

            ADCIFG = 0;                                              // Clear interrupt flag
            break;
        default:
            break;
    }
}
```

TASK

- Run the code given in the lecture
- From temperature sensor (LM35) read temperature and convert it into Digital value by using ADC0804 and display the value on the LCD. In LCD at first line write your registration Number and on the second line display the value of the temperature sensor attached with ADC0804. (use proteus can also use MSP430FR4133)
- Attach temperature sensor with msp430 and display the temperature on the LCD.