# EE 200

# Design Tools

# Laboratory 29

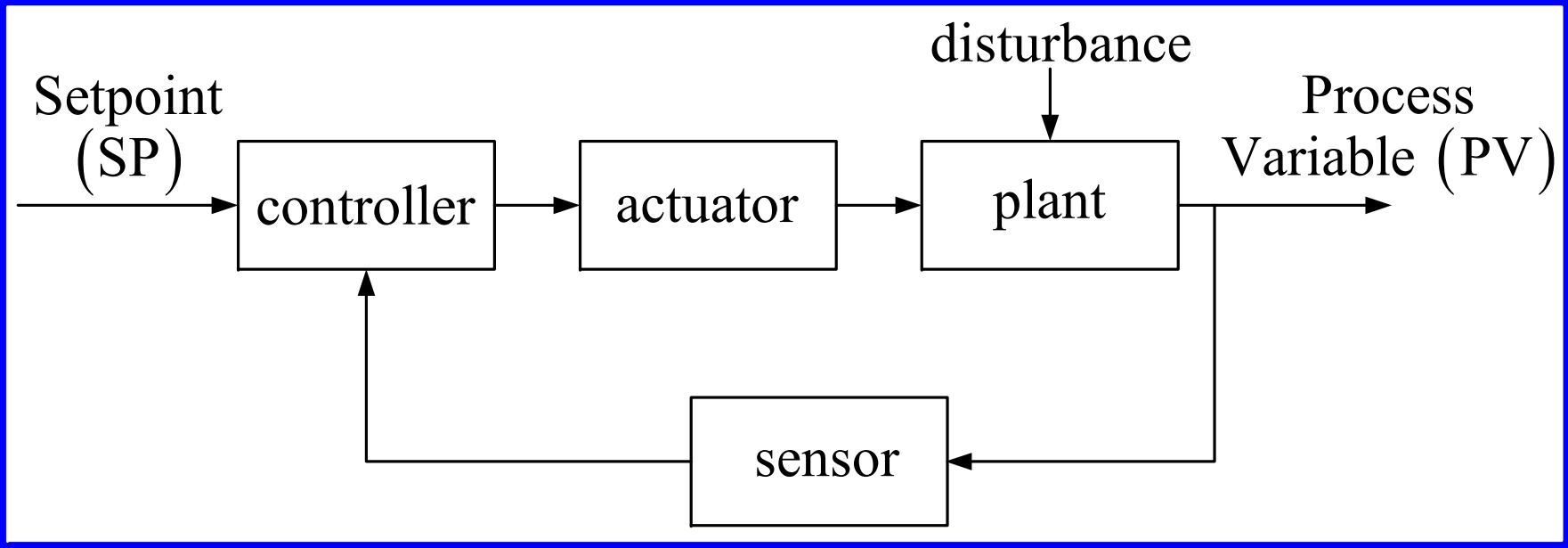
PennState
College of Engineering


School of Electrical Engineering and Computer Science

# Laboratory 29 Topics

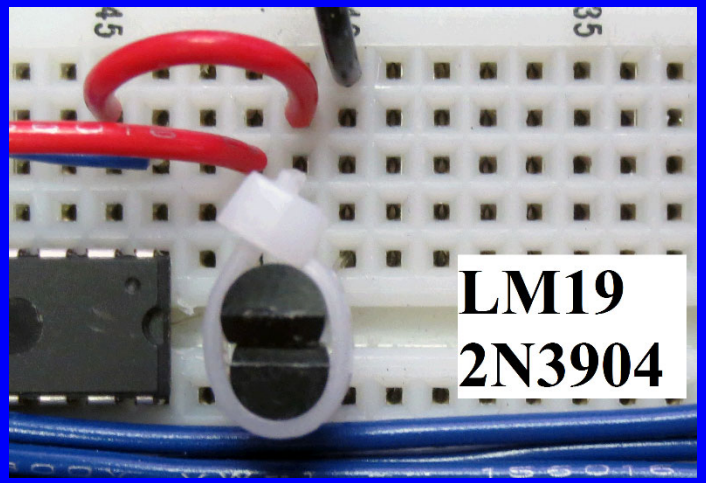
- Feedback control realization using a microcontroller and the C programming language

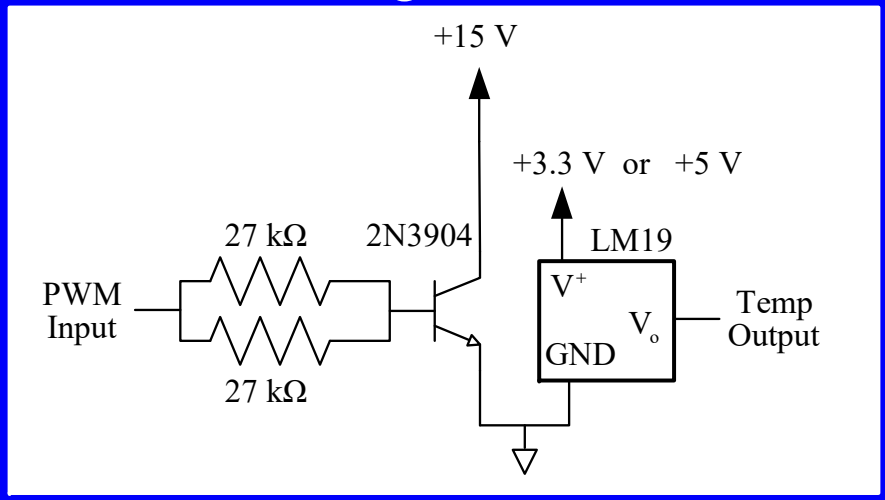
# Feedback Control System



# EE 200 Feedback Exercises

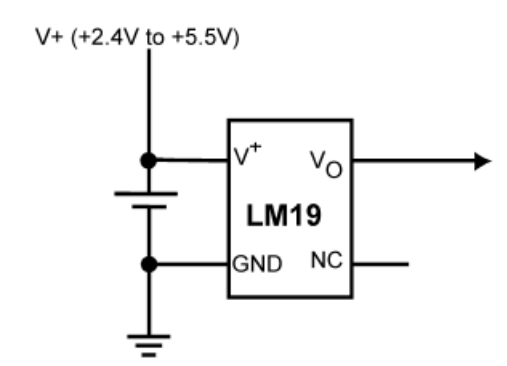
- Regulate temperature using proportional plus integral (PI) control
- Heat a LM19 temperature sensor using a NPN transistor




- Regulate the LM19 temperature to a desired value by controlling the duty cycle of the base current

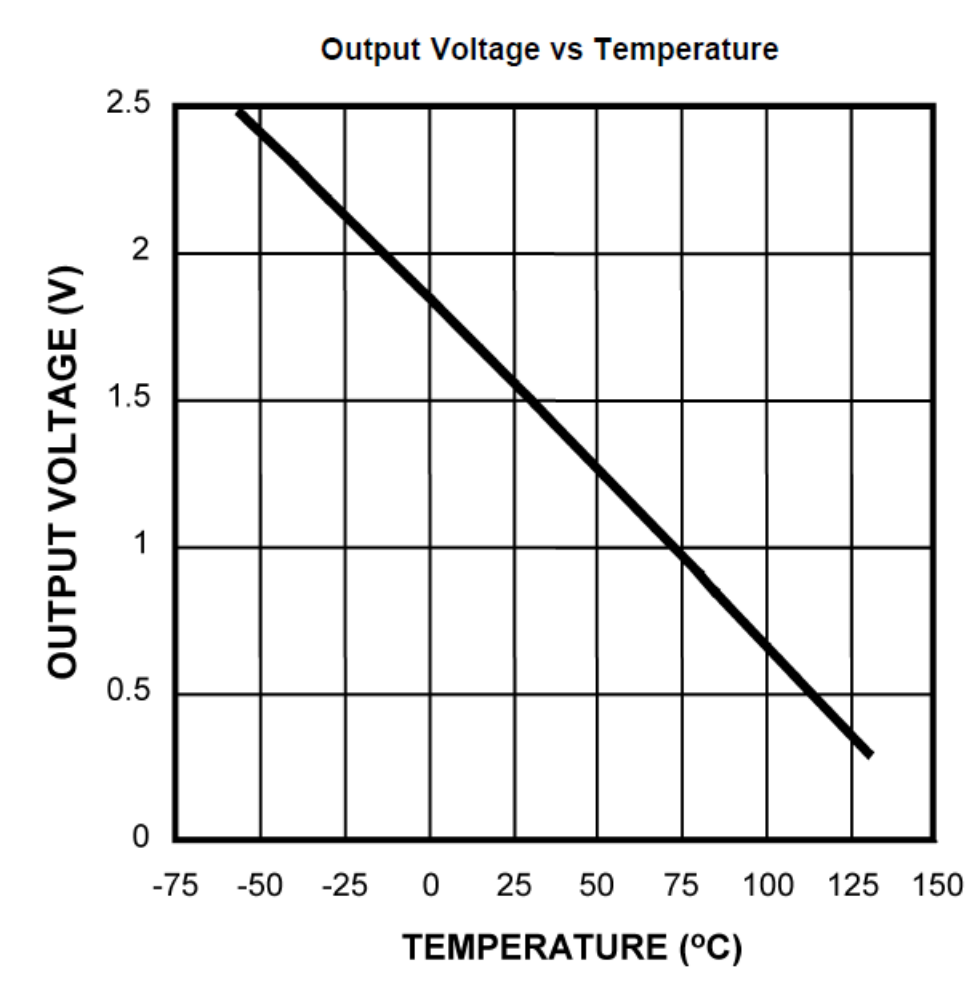
# LM19 Temperature Sensor




$$V_O = (-3.88\times10^{-6}\times T^2) + (-1.15\times10^{-2}\times T) + 1.8639$$

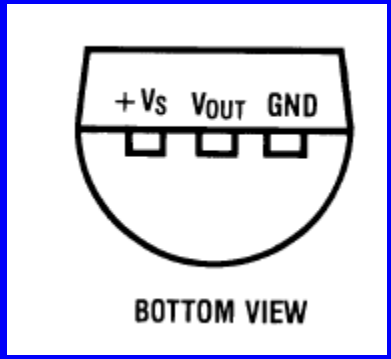
or

$$T = -1481.96 + \sqrt{2.1962 \times 10^{6} + \frac{(1.8639 - V_O)}{3.88 \times 10^{-6}}}$$



# LM19 Temperature Sensor

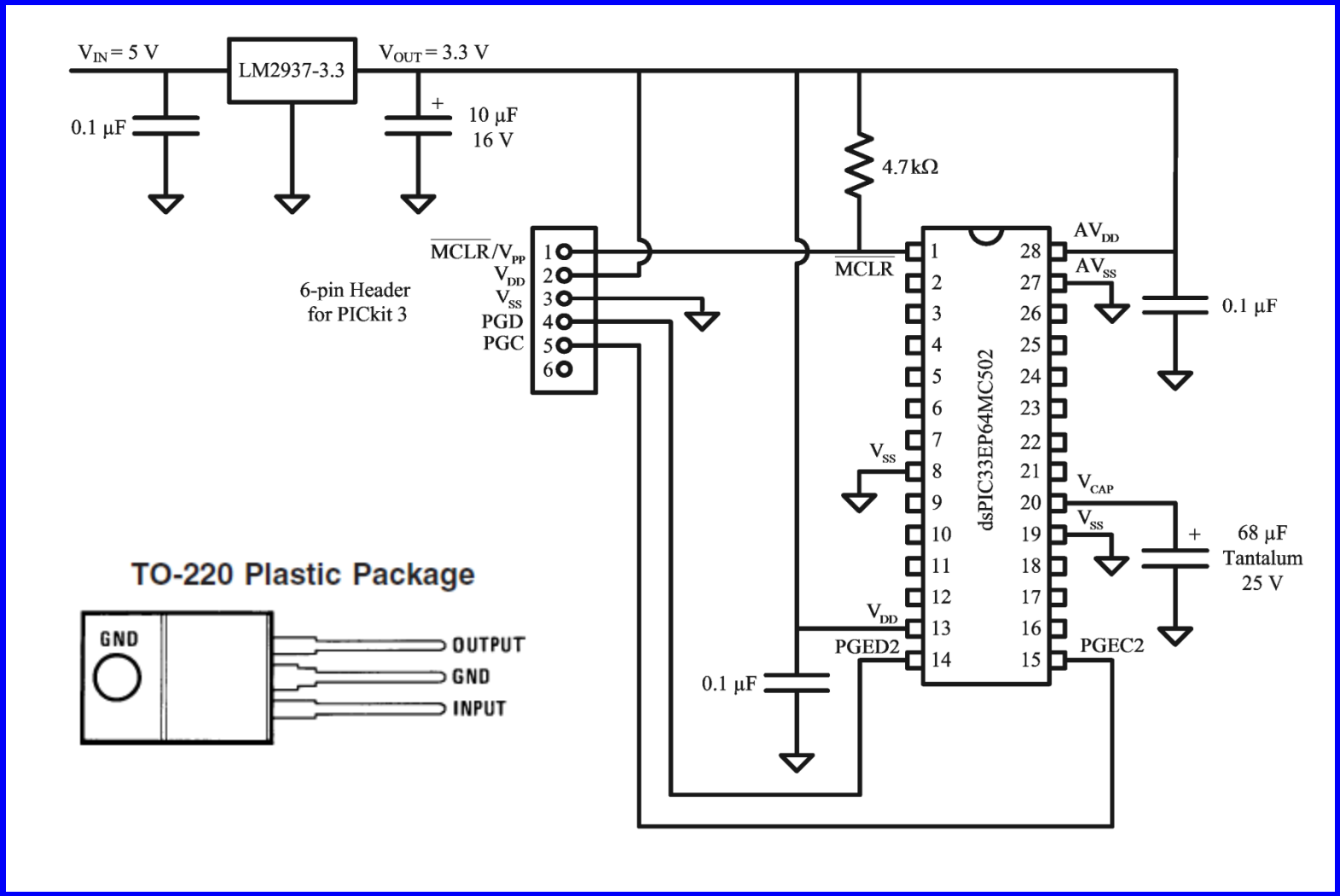
- **Warning:**
  - **Reversing GND and V+ will destroy the sensor**
  - **Verify connections before powering the circuit**



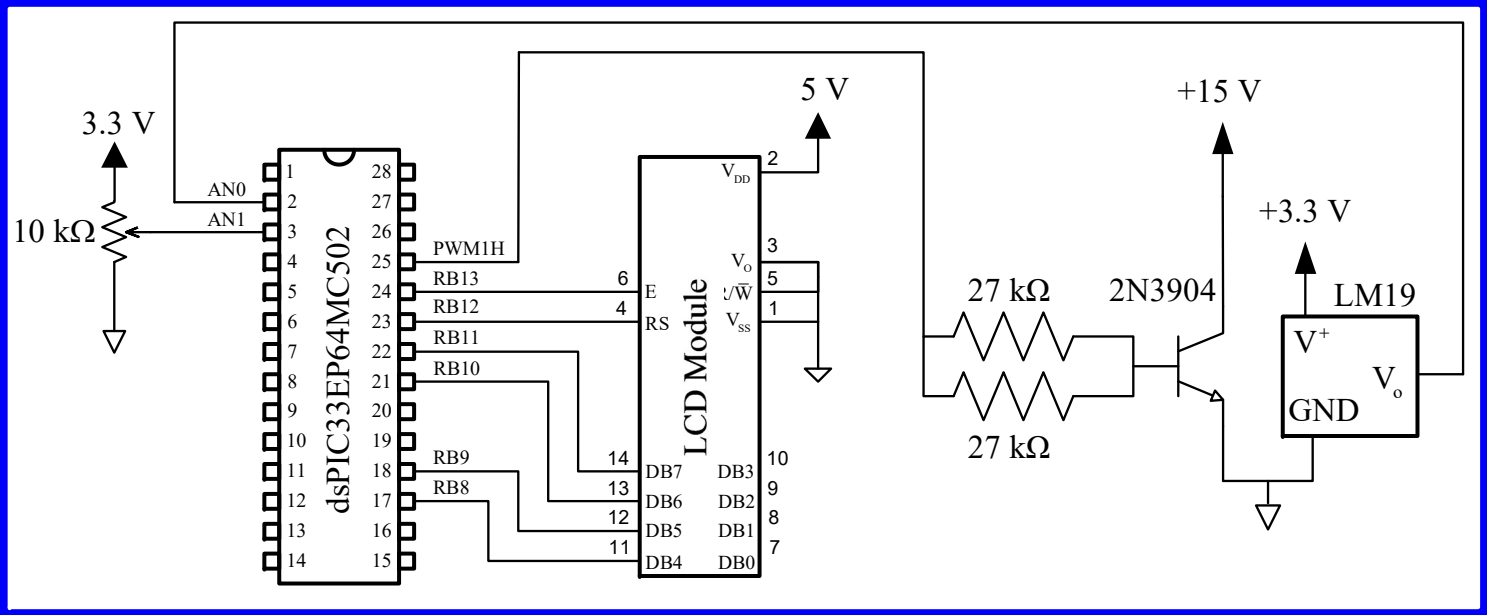
# Exercise 1

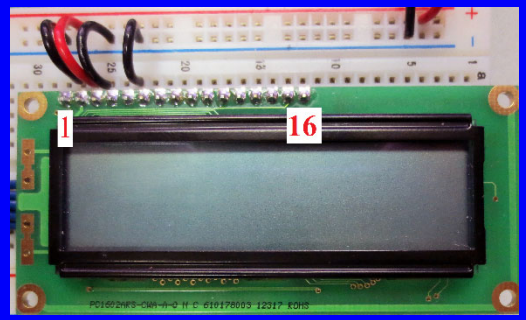
- Write a C program that displays the sensor output voltage and corresponding temperature in °F
  - Acquire the analog signal using pin 2 (ANO)
  - Update the display every 500 ms

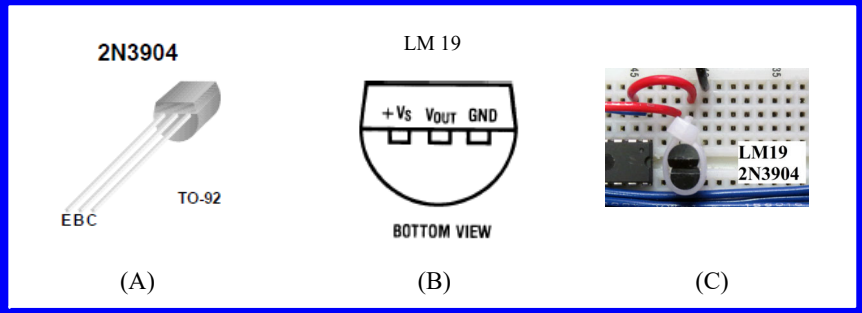
# Laboratory 29 Circuit Part 1



# Laboratory 29 Circuit Part 2



# Exercise 1

```c
/* define FCY before including libpic30.h */
#define FCY 3685000UL

#include <p33EP64MC502.h>
#include <libpic30.h>
#include <math.h>
#include <stdio.h>
#include <xc.h>
#include "EE200_LCD.h"

/* set Configuration Bits */
#pragma config ICS = PGD2    // communicate on PGED2 (pin 14) and PGEC2 (pin 15)

/* declare functions */
void Init_ADC(void);

void Init_ADC (void) {
    ANSELAbits.ANSA0 = 1;  // set pin 2 (AN0) for analog input
    TRISAbits.TRISA0 = 1;  // configure for input (disable DO)
    AD1CON1bits.ADON = 1;  // turn ADC module on
  }

```

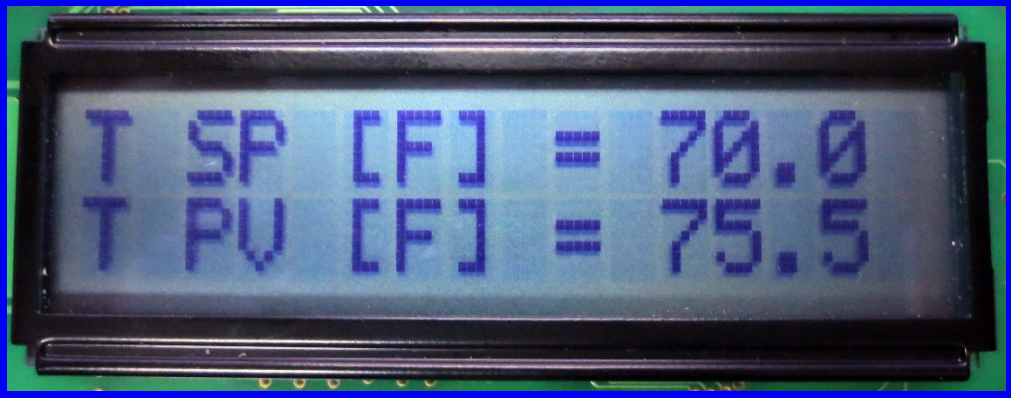
```c
int main(void) {
    char Line_char_Array[16];
    unsigned int TS;
    double V_PV, TC, TF_PV;

    /* Set Parameters */
    TS = 500; // sample period in ms

    Init_LCD_Module();
    Init_ADC ( );

    while (1) {
       __delay_ms(TS);              // set sample rate to 500 ms

       /* Capture analog input from LM19 on pin 2 (AN0) */
       AD1CON1bits.SAMP = 1;        // start Sample; place inp signal across cap
       __delay_us(10);              // wait for cap voltage to track analog inp
       AD1CON1bits.SAMP = 0;        // start SA converter
       while (!AD1CON1bits.DONE);  // wait for SA conv to complete
       V_PV = 3.3 * (double) ADC1BUF0 / 1023.0; // convert to units of Volts

       /* Determine temp in units of degrees F */
       TC = -1481.96 + sqrt(2.1962e6 + (1.8639 - V_PV)/3.88e-6);
       TF_PV = (9.0/5.0)*TC + 32.0;

       /* Update LCD */
       Position_LCD_Cursor(0x00);  // place cursor at cell 0x00
       sprintf(Line_char_Array,"V PV [V] = %4.3f", V_PV);
       Write_LCD_String(Line_char_Array);

       Position_LCD_Cursor(0x40);  // place cursor at cell 0x40
       sprintf(Line_char_Array,"T PV [F] = %4.1f",TF_PV);
       Write_LCD_String(Line_char_Array);

       ClrWdt(); // restart the watchdog timer
    }
    return 0;
}
```

# Exercise 1

- Verify the process variable measurement by pinching the thermal sensor with your fingers
  - Does the process variable temperature increase?
- Based on your understanding of the C code, is the true sample period equal to or greater than 500 ms?

# Exercise 2

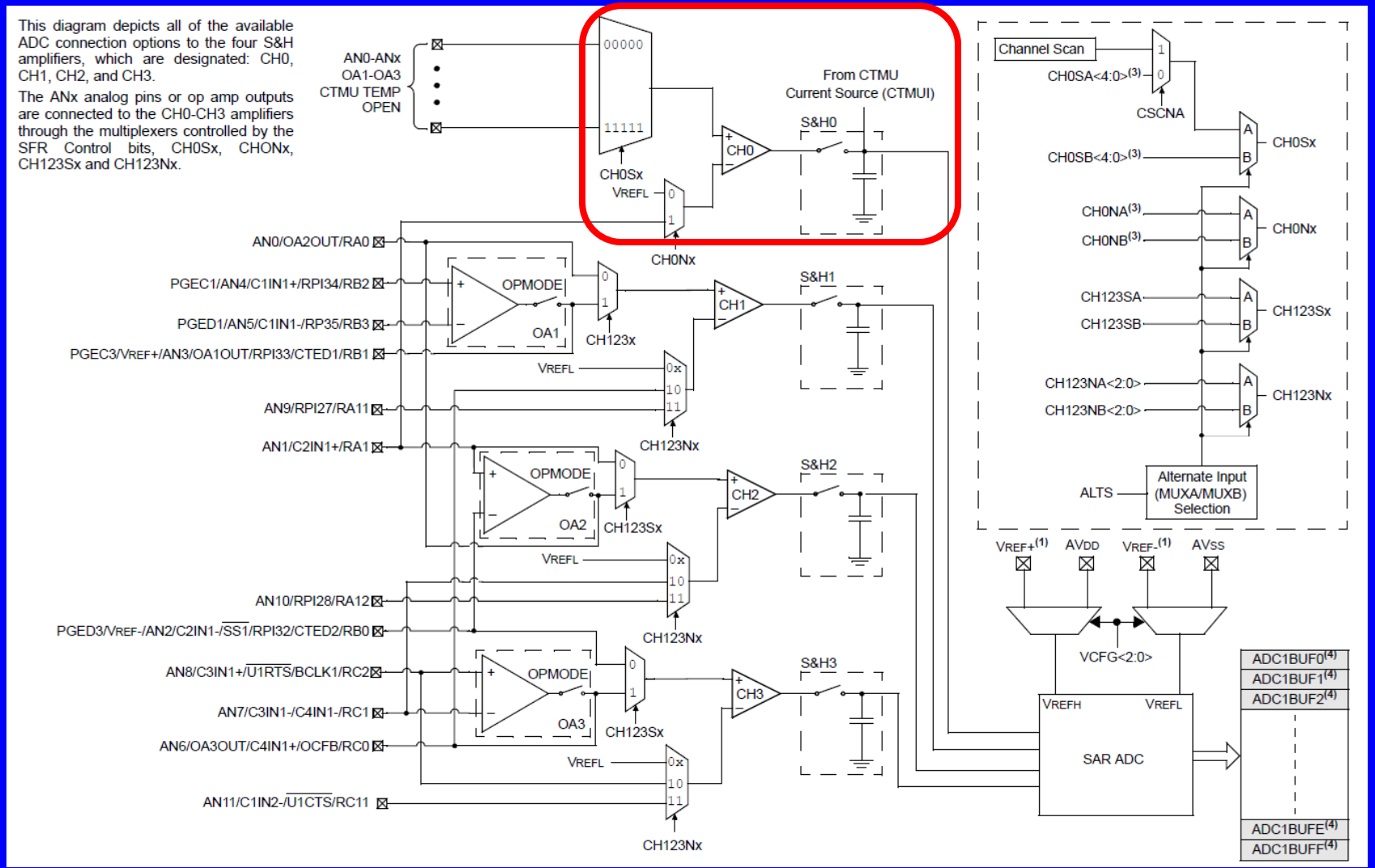
- Use a potentiometer to generate a set point temperature
  - Acquire the analog signal using AN1
  - Linearly map the input voltage (0 to 3.3 V) to a set point temperature (70 to 99 °F)
- Display the set point (SP) temperature and measured process variable (PV) temperature on lines 1 and 2 of the LCD module



# Set Analog Input Multiplexers

This diagram depicts all of the available ADC connection options to the four S&H amplifiers, which are designated: CH0, CH1, CH2, and CH3.

The ANx analog pins or op amp outputs are connected to the CH0-CH3 amplifiers through the multiplexers controlled by the SFR Control bits, CH0Sx, CHONx, CH123Sx and CH123Nx.



# AD1CHS0

**REGISTER 23-6: AD1CHS0: ADC1 INPUT CHANNEL 0 SELECT REGISTER**

| R/W-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CH0NB | — | — | CH0SB<4:0> | | | | |
| bit 15 | | | | | | | bit 8 |

| R/W-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CH0NA | — | — | CH0SA<4:0> | | | | |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 15 **CH0NB:** Channel 0 Negative Input Select for Sample MUXB bit
1 = Channel 0 negative input is AN1(1)
0 = Channel 0 negative input is VREFL

bit 14-13 **Unimplemented:** Read as '0'

bit 12-8 **CH0SB<4:0>:** Channel 0 Positive Input Select for Sample MUXB bits(1)
11111 = Open; use this selection with CTMU capacitive and time measurement
11110 = Channel 0 positive input is connected to CTMU temperature measurement diode (CTMU TEMP)
11101 = Reserved
11100 = Reserved
11011 = Reserved
11010 = Channel 0 positive input is output of OA3/AN6(2)
11001 = Channel 0 positive input is output of OA2/AN0(2)
11000 = Channel 0 positive input is output of OA1/AN3(2)
10111 = Reserved
•
•
•
10000 = Reserved
01111 = Channel 0 positive input is AN15(3)
01110 = Channel 0 positive input is AN14(3)
01101 = Channel 0 positive input is AN13(3)
•
•
•
00010 = Channel 0 positive input is AN2(3)
00001 = Channel 0 positive input is AN1(3)
00000 = Channel 0 positive input is AN0(3)

bit 7 **CH0NA:** Channel 0 Negative Input Select for Sample MUXA bit
1 = Channel 0 negative input is AN1(1)
0 = Channel 0 negative input is VREFL

bit 6-5 **Unimplemented:** Read as '0'

bit 4-0 **CH0SA<4:0>:** Channel 0 Positive Input Select for Sample MUXA bits(1)
11111 = Open; use this selection with CTMU capacitive and time measurement
11110 = Channel 0 positive input is connected to CTMU temperature measurement diode (CTMU TEMP)
11101 = Reserved
11100 = Reserved
11011 = Reserved
11010 = Channel 0 positive input is output of OA3/AN6(2)
11001 = Channel 0 positive input is output of OA2/AN0(2)
11000 = Channel 0 positive input is output of OA1/AN3(2)
10110 = Reserved
•
•
•
10000 = Reserved
01111 = Channel 0 positive input is AN15(1)
01110 = Channel 0 positive input is AN14(1)
01101 = Channel 0 positive input is AN13(1)
•
•
•
00010 = Channel 0 positive input is AN2(1)
00001 = Channel 0 positive input is AN1(1)
00000 = Channel 0 positive input is AN0(1)

- Page 329 of dsPIC33 data sheet
- Choose CH-0 positive input by setting AD1CHS0

# Exercise 2

- Use a single C function that returns the analog voltage at either pin 2 (AN0) or pin 3 (AN1)
  - Set AD1CHS0 = 0x000 for ADC on pin 2 (AN0)
  - Set AD1CHS0 = 0x001 for ADC on pin 3 (AN1)

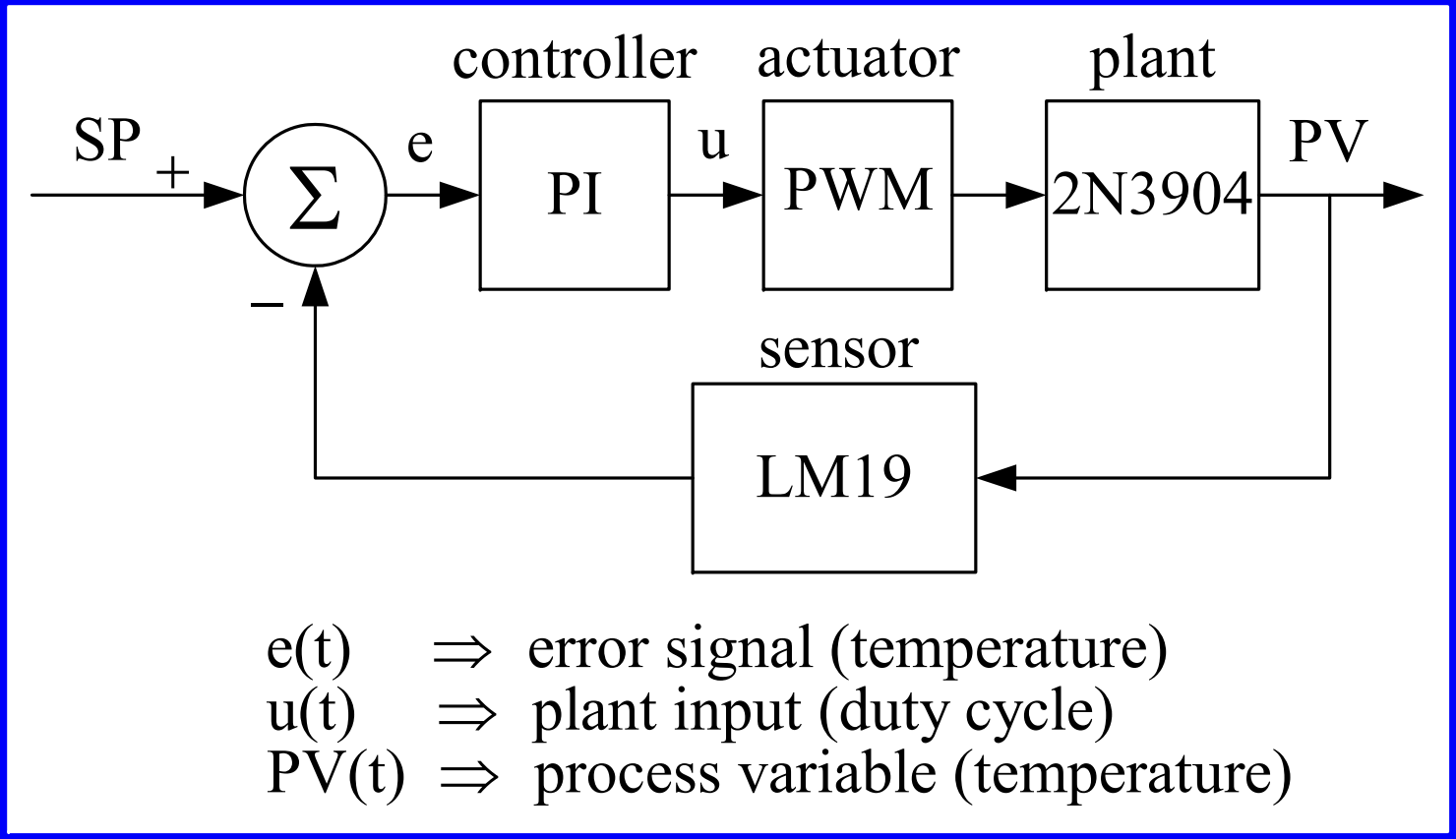
```
36  double Read_ADC(int channel) {
37      double Vmeas;
38
39      AD1CHS0 = channel;              // choose CH-0 positive input
40      AD1CON1bits.SAMP = 1;           // start Sample; place inp signal across cap
41      __delay_us(10);                 // wait for cap voltage to track analog inp
42      AD1CON1bits.SAMP = 0;           // start SA converter
43      while (!AD1CON1bits.DONE);      // wait for SA conv to complete
44      Vmeas = 3.3 * (double) ADC1BUF0 / 1023.0; // convert to units of Volts
45
46      return Vmeas;
47  }
```

# Exercise 2

```c
/* define FCY before including libpic30.h */
#define FCY 3685000UL

#include <p33EP64MC502.h>
#include <libpic30.h>
#include <stdio.h>
#include <math.h>
#include <xc.h>
#include "EE200_LCD.h"

/* set Configuration Bits */
#pragma config ICS = PGD2    // communicate on PGED2 (pin 14) and PGEC2 (pin 15)
#pragma config JTAGEN = OFF  // disable JTAG inorder to use RB8 through RB11

/* declare functions */
void Init_ADC(void);
double Read_ADC(int channel);

void Init_ADC (void) {
    ANSELAbits.ANSA0 = 1;  // set pin 2 (AN0) for analog input
    TRISAbits.TRISA0 = 1;  // configure for input (dispable DO)

    ANSELAbits.ANSA1 = 1;  // set pin 3 (AN1) for analog input
    TRISAbits.TRISA1 = 1;  // configure for input (dispable DO)

    AD1CON1bits.ADON = 1;  // turn ADC module on
  }

double Read_ADC(int channel) {
    double Vmeas;

    AD1CHS0 = channel;             // choose CH-0 positive input
    AD1CON1bits.SAMP = 1;          // start Sample; place inp signal across cap
    __delay_us(10);                // wait for cap voltage to track analog inp
    AD1CON1bits.SAMP = 0;          // start SA converter
    while (!AD1CON1bits.DONE);     // wait for SA conv to complete
    Vmeas = 3.3 * (double) ADC1BUF0 / 1023.0; // convert to units of Volts

    return Vmeas;
}
```

```c
int main(void) {
    char Line_char_Array[16];
    unsigned int TS;
    double V_PV, V_SP, TC, TF_PV, TF_SP;

    /* Set Parameters */
    TS = 500; // sample period in ms

    Init_LCD_Module();
    Init_ADC();

    while (1) {
        __delay_ms(TS);      // set sample rate to 500 ms

        /* Acquire analog input signals */
        V_PV = Read_ADC(0); // read PV voltage on pin 2 (AN0)
        V_SP = Read_ADC(1); // read SP voltage on pin 3 (AN1)

        /* Determine process variable temp in degrees F */
        TC = -1481.96 + sqrt(2.1962e6 + (1.8639 - V_PV)/3.88e-6);
        TF_PV = (9.0/5.0)*TC + 32.0;

        /* Determine setpoint temp in degrees F */
        TF_SP = 29.0*V_SP/3.3 + 70.0;

        /* Update LCD */
        Position_LCD_Cursor(0x00); // place cursosr at cell 0x00
        sprintf(Line_char_Array,"T SP [F] = %4.1f", TF_SP);
        Write_LCD_String(Line_char_Array);

        Position_LCD_Cursor(0x40); // place cursosr at cell 0x40
        sprintf(Line_char_Array,"T PV [F] = %4.1f", TF_PV);
        Write_LCD_String(Line_char_Array);

        ClrWdt(); // restart the watchdog timer
    }
    return 0;
}
```

# Block Diagram Representation




e(t) ⇒ error signal (temperature)
u(t) ⇒ plant input (duty cycle)
PV(t) ⇒ process variable (temperature)

# PI Controller

- Representation

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau) d\tau$$

- The proportional ($K_P$) and integral ($K_I$) gains are design parameters chosen to obtain desired
  - Transient response characteristics
  - Steady-state accuracy

- EE 380 and EE 482 show how to determine control gains for analog and digital control systems, respectively

# Sampled-Data Control

- The output voltage $V_o$ of the LM19 is sampled in time
- Signals in the microcontroller are indexed in time by k

$$V_o(k) = V_o(t)\Big|_{t=kT}$$

$$T \Rightarrow \text{sample period}$$

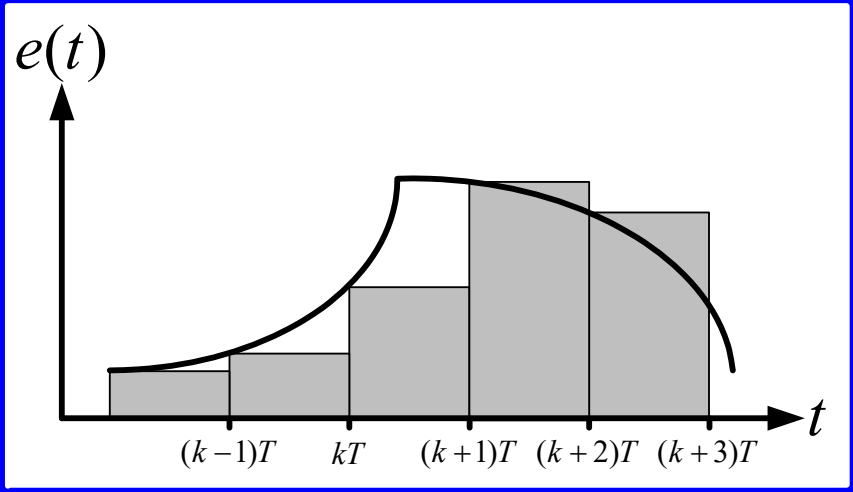
- The amplitude of $V_o(k)$ is quantized as it is represented by a finite-length register

# Approximating the Integral

- Using the samples e(k), the microcontroller must approximate

$$x(t) = \int_0^t e(\tau)\,d\tau$$

- Represent the area x as a sum of rectangles whose width is the sample period T and whose height is e(k)




$$\begin{aligned} x(k) &= \sum_{n=0}^{k} e(n)T \\ &= \sum_{n=0}^{k-1} e(n)T + Te(k) \\ &= x(k-1) + Te(k) \end{aligned}$$

# PI Controller

- Analog Representation

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau) d\tau$$

- Sampled Data Representation

$$x(k) = x(k-1) + Te(k)$$

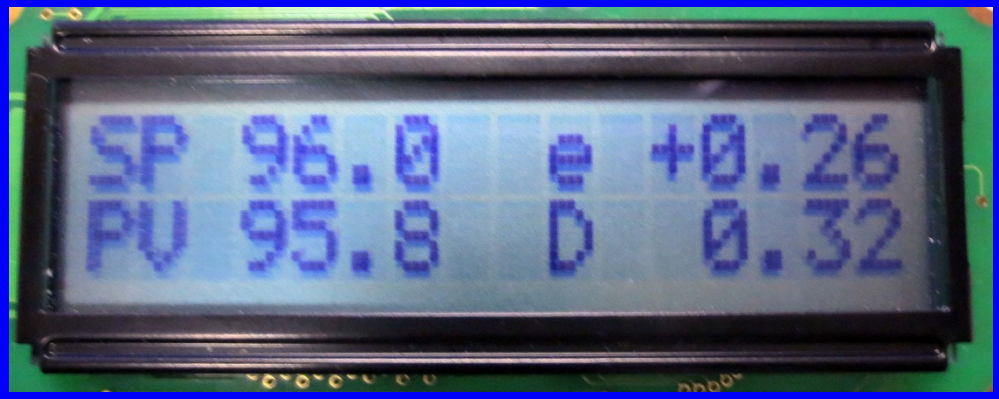
$$\mathrm{u}(k) = K_P e(k) + K_I x(\mathrm{k})$$

# Exercise 3

- Realize a PI controller using
  - sample period $T_S$ = 500 ms
  - proportional gain $K_P$ = 0.25
  - integral gain $K_I$ = 0.05
- Drive the 2N3904 NPN transistor using the PWM signal PWM1H (pin 25)
  - PWM frequency = 1 kHz (PTPER = 7370)
  - Limit the fractional duty cycle between 0 and 1 (PDC1 between 0 and 7370)

# Exercise 3

- Use the LCD module to display:
  - set point (SP) temperature [°F]
  - measured process variable (PV) temperature [°F]
  - error (e) [°F] where e = SP – PV
  - fractional duty cycle (D)



# Exercise 3

```c
/* define FCY before including libpic30.h */
#define FCY 3685000UL

#include <p33EP64MC502.h>
#include <libpic30.h>
#include <stdio.h>
#include <math.h>
#include <xc.h>
#include "EE200_LCD.h"

/* set Configuration Bits */
#pragma config ICS = PGD2    // communicate on PGED2 (pin 14) and PGEC2 (pin 15)
#pragma config JTAGEN = OFF  // disable JTAG inorder to use RB8 through RB11
#pragma config PWMLOCK = OFF // PWM Lock Enabl Bit

/* declare functions */
void Init_ADC(void);
void Init_PWM_module(void);
double Read_ADC(int channel);

void Init_ADC (void) {
    ANSELAbits.ANSA0 = 1;          // set pin 2 (AN0) for analog input
    TRISAbits.TRISA0 = 1;          // configure for input (disable DO)

    ANSELAbits.ANSA1 = 1;          // set pin 3 (AN1) for analog input
    TRISAbits.TRISA1 = 1;          // configure for input (disable DO)

    AD1CON1bits.ADON = 1;          // turn ADC module on
  }

```

```c
double Read_ADC(int channel) {
    double Vmeas;

    AD1CHS0 = channel;             // choose CH-0 positive input
    AD1CON1bits.SAMP = 1;          // start Sample; place inp signal across cap
    __delay_us(10);                // wait for cap voltage to track analog inp
    AD1CON1bits.SAMP = 0;          // start SA converter
    while (!AD1CON1bits.DONE);     // wait for SA conv to complete
    Vmeas = 3.3 * (double) ADC1BUF0 / 1023.0; // convert to units of Volts

    return Vmeas;
}

void Init_PWM_Module(void) {
    PTCONbits.PTEN = 0;      // disable the PWM module
    PTPER = 7370;            // period as a multiple of clock cycles
    PDC1  = 3685;            // fract duty cycle as a multiple of clock cycles
    IOCON1bits.PMOD  = 1;    // PWM1 I/O pin pair is in redundant mode
    IOCON1bits.PENH  = 1;    // PWM module controls pin 25, PWM1H

    /* RB10 to RB13 must be configured to control the LCD module */
    IOCON2bits.PENL  = 0;    // GPIO module controls pin 24, RB13
    IOCON2bits.PENH  = 0;    // GPIO module controls pin 23, RB12
    IOCON3bits.PENL  = 0;    // GPIO module controls pin 22, RB11
    IOCON3bits.PENH  = 0;    // GPIO module controls pin 21, RB10

    FCLCON1bits.FLTMOD = 3; // fault input disabled
    PTCONbits.PTEN = 1;      // enable the PWM module
}

```

# Exercise 3

```
68  int main(void) {
69      char Line_char_Array[16];
70      unsigned int TS;
71      double V_PV, V_SP, TC, TF_PV, TF_SP, KP, KI, x, e, u;
72
73      /* Set Parameters */
74      TS = 500;               // sample period in ms
75      KP = 0.25;              // proportional gain
76      KI = 0.05;              // integral gain
77      x   =  0;               // initial value of integrator
78
79      Init_ADC();
80      Init_PWM_Module();
81      Init_LCD_Module();
82
```

```
83      while (1) {
84          __delay_ms(TS);       // set sample rate to 500 ms
85
86          /* Acquire analog input signals */
87          V_PV = Read_ADC(0); // read PV voltage on pin 2 (AN0)
88          V_SP = Read_ADC(1); // read SP voltage on pin 3 (AN1)
89
90          /* Determine process variable temp in degrees F */
91          TC = -1481.96 + sqrt(2.1962e6 + (1.8639 - V_PV)/3.88e-6);
92          TF_PV = (9.0/5.0)*TC + 32.0;
93
94          /* Determine setpoint temp in degrees F */
95          TF_SP = 29.0*V_SP/3.3 + 70.0;
96
97          /* Realize PI Controller */
98           e = TF_SP - TF_PV;
99           x = x + TS*e/1000;
100          u = KP*e + KI*x;
101
102         /* Limit fractional duty cycle between 0 and 1 */
103         if (u > 1)
104             u = 1;
105         else if (u < 0)
106             u = 0;
107
108         /* Set PWM duty cycle */
109         PDC1  = floor(7370 * u);
110
111         /* Update LCD  */
112         Position_LCD_Cursor(0x00); // place cursosr at cell 0x00
113         sprintf(Line_char_Array,"SP %4.1f  e %+4.2f", TF_SP, e);
114         Write_LCD_String(Line_char_Array);
115
116         Position_LCD_Cursor(0x40); // place cursosr at cell 0x40
117         sprintf(Line_char_Array,"PV %4.1f  D  %4.2f", TF_PV, u);
118         Write_LCD_String(Line_char_Array);
119
120         ClrWdt(); // restart the watchdog timer
121     }
122     return 0;
123 }
```

# Exercise 3

- For a fixed set point temperature, the process variable, error, and fractional duty cycle approach steady-state values
- In steady-state, the power dissipated by the 2N3904 NPN transistor matches the heat loss of the LM19/2N3904 structure to the environment
- Verify operation of the closed-loop system by bringing the blade of a room temperature screwdriver in contact with the case of the 2N3904 and/or LM19
  - Should the fractional duty cycle increase or decrease? Why?

# EE 200

# Design Tools

# Laboratory 29

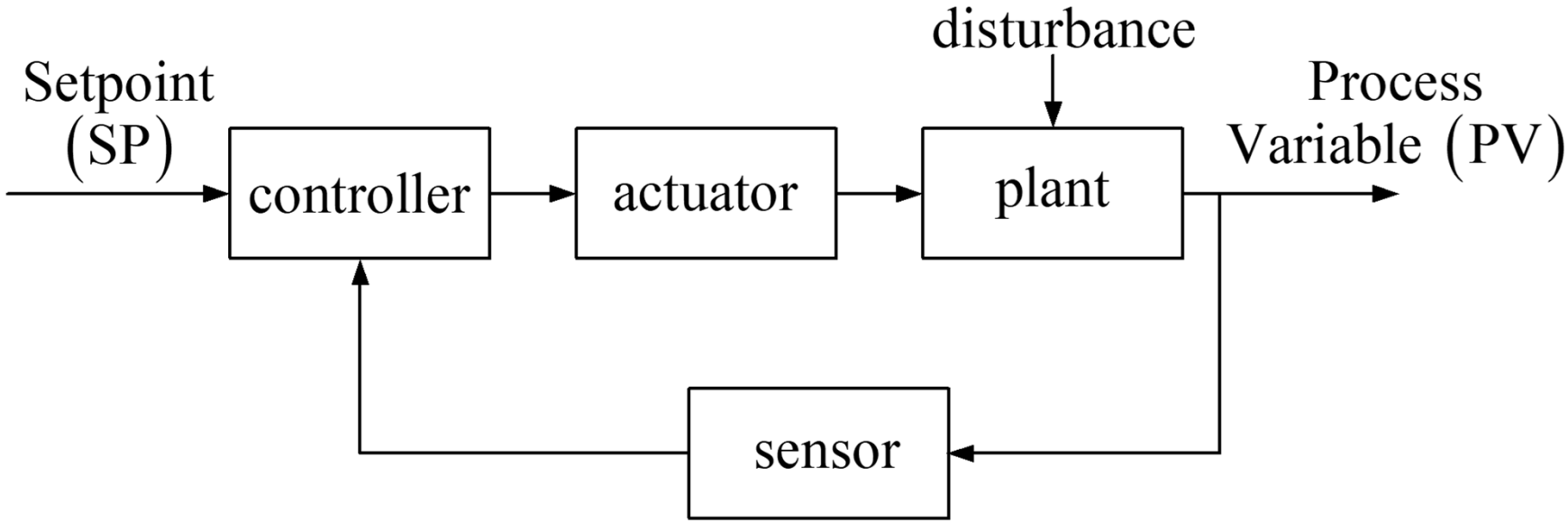
PennState College of Engineering


School of Electrical Engineering and Computer Science

# Laboratory 29 Topics

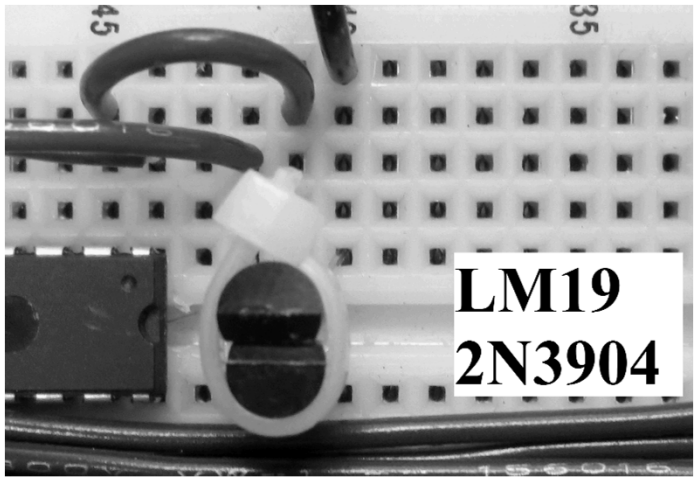
- Feedback control realization using a microcontroller and the C programming language

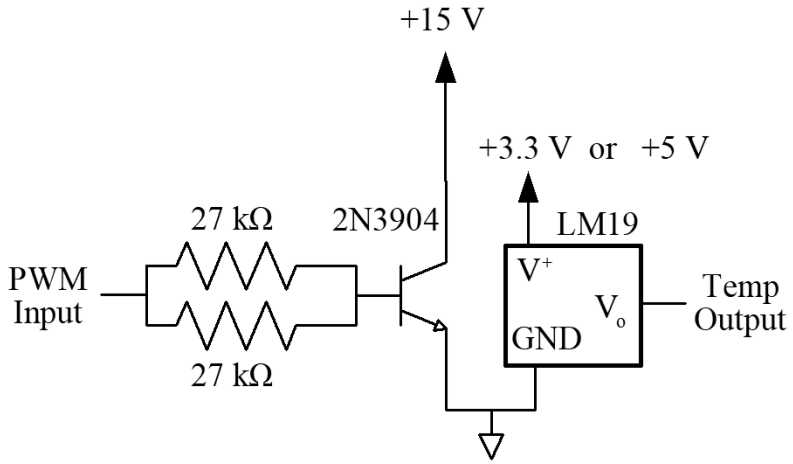
# Feedback Control System



# EE 200 Feedback Exercises

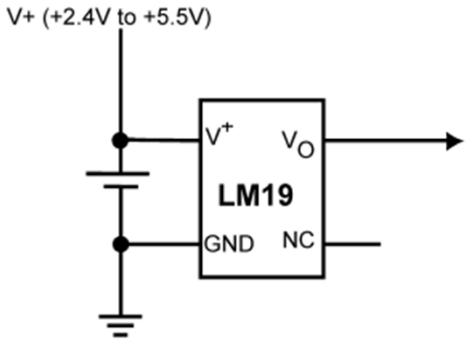
- Regulate temperature using proportional plus integral (PI) control
- Heat a LM19 temperature sensor using a NPN transistor




- Regulate the LM19 temperature to a desired value by controlling the duty cycle of the base current

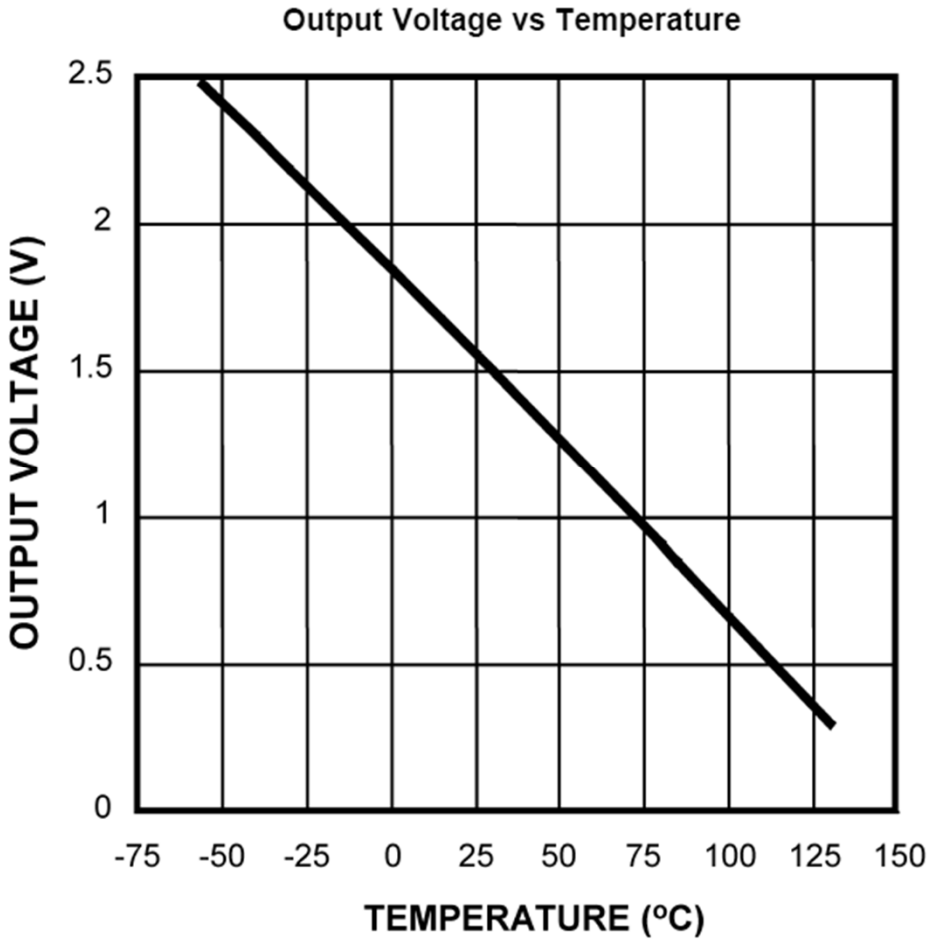
# LM19 Temperature Sensor




$$V_O = (-3.88\times10^{-6}\times T^2) + (-1.15\times10^{-2}\times T) + 1.8639$$

or

$$T = -1481.96 + \sqrt{2.1962 \times 10^{6} + \frac{(1.8639 - V_O)}{3.88 \times 10^{-6}}}$$




PennState College of Engineering


School of Electrical Engineering and Computer Science

# LM19 Temperature Sensor

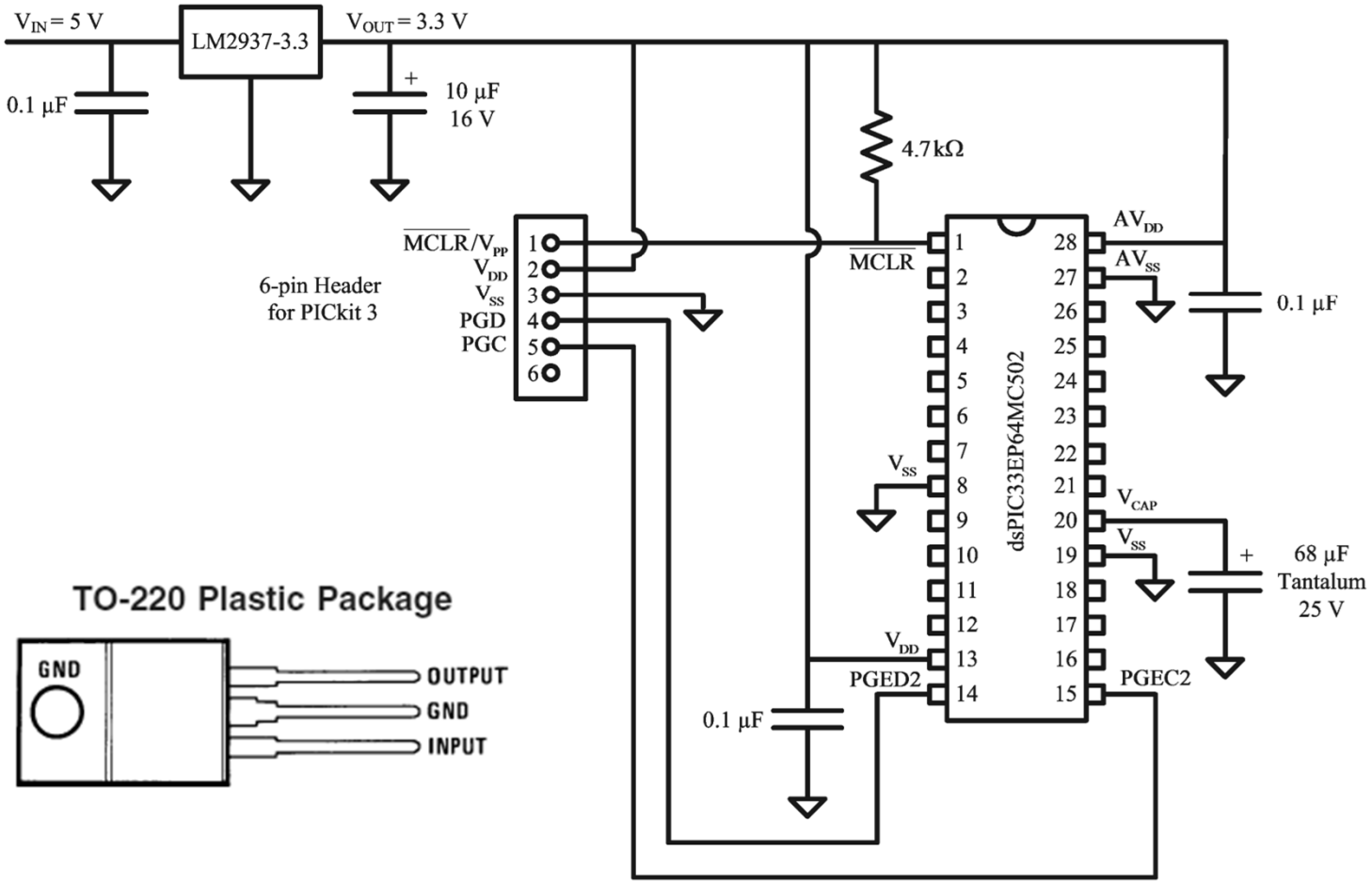
- **Warning:**
  - **Reversing GND and V+ will destroy the sensor**
  - **Verify connections before powering the circuit**

BOTTOM VIEW

# Exercise 1

- Write a C program that displays the sensor output voltage and corresponding temperature in °F
  - Acquire the analog signal using pin 2 (ANO)
  - Update the display every 500 ms

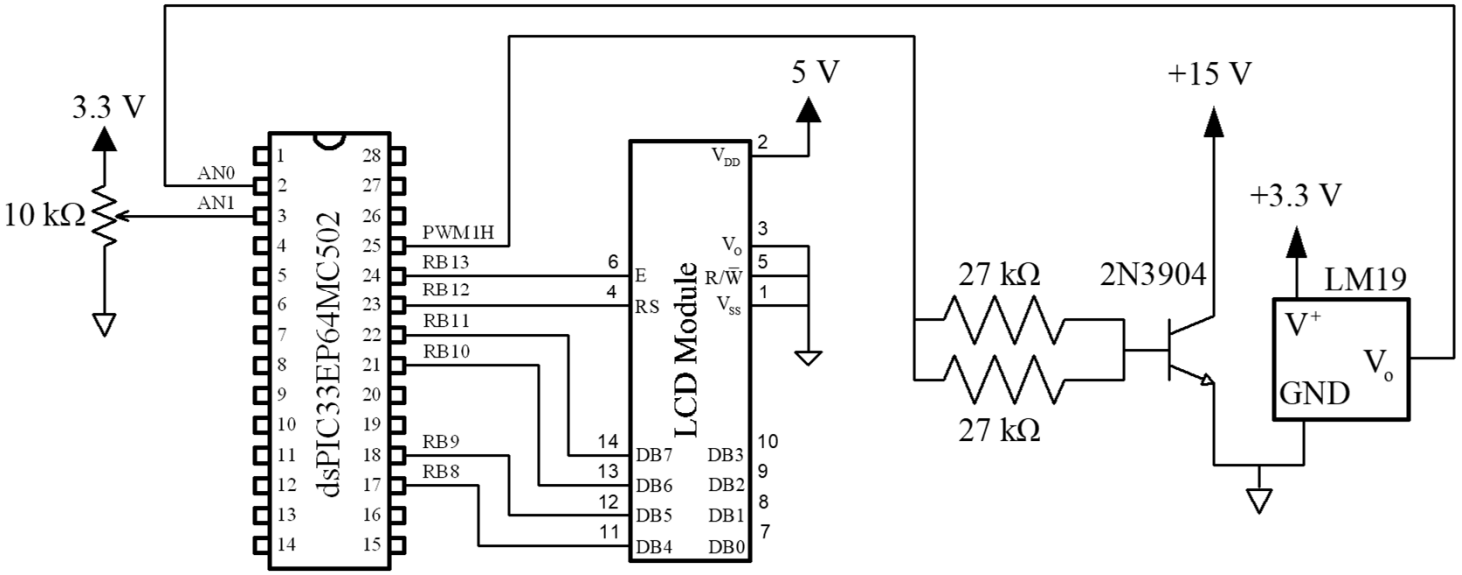
# Laboratory 29 Circuit Part 1

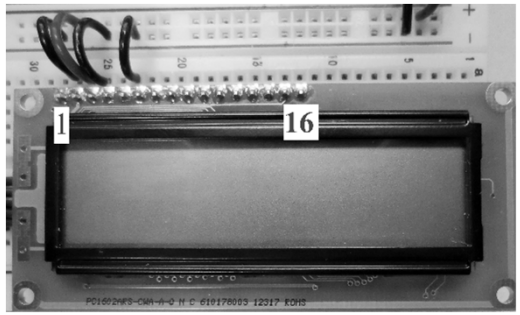



PennState College of Engineering


School of Electrical Engineering and Computer Science

# Laboratory 29 Circuit Part 2

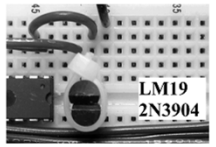




(A)


(B)




(C)

PennState College of Engineering


School of Electrical Engineering and Computer Science

# Exercise 1

```
8   /* define FCY before including libpic30.h */
9   #define FCY 3685000UL
10
11  #include <p33EP64MC502.h>
12  #include <libpic30.h>
13  #include <math.h>
14  #include <stdio.h>
15  #include <xc.h>
16  #include "EE200_LCD.h"
17
18  /* set Configuration Bits */
19  #pragma config ICS = PGD2    // communicate on PGED2 (pin 14) and PGEC2 (pin 15)
20
21  /* declare functions */
22  void Init_ADC(void);
23
24  void Init_ADC (void) {
25      ANSELAbits.ANSA0 = 1;  // set pin 2 (ANO) for analog input
26      TRISAbits.TRISA0 = 1;  // configure for input (disable DO)
27      AD1CON1bits.ADON = 1;  // turn ADC module on
28    }
29
```

```
30  int main(void) {
31      char Line_char_Array[16];
32      unsigned int TS;
33      double V_PV, TC, TF_PV;
34
35      /* Set Parameters */
36      TS = 500; // sample period in ms
37
38      Init_LCD_Module();
39      Init_ADC ( );
40
41      while (1) {
42          __delay_ms(TS);             // set sample rate to 500 ms
43
44          /* Capture analog input from LM19 on pin 2 (ANO) */
45          AD1CON1bits.SAMP = 1;       // start Sample; place inp signal across cap
46          __delay_us(10);             // wait for cap voltage to track analog inp
47          AD1CON1bits.SAMP = 0;       // start SA converter
48          while (!AD1CON1bits.DONE);  // wait for SA conv to complete
49          V_PV = 3.3 * (double) ADC1BUF0 / 1023.0; // convert to units of Volts
50
51          /* Determine temp in units of degrees F */
52          TC = -1481.96 + sqrt(2.1962e6 + (1.8639 - V_PV)/3.88e-6);
53          TF_PV = (9.0/5.0)*TC + 32.0;
54
55          /* Update LCD */
56          Position_LCD_Cursor(0x00);  // place cursor at cell 0x00
57          sprintf(Line_char_Array,"V PV [V] = %4.3f", V_PV);
58          Write_LCD_String(Line_char_Array);
59
60          Position_LCD_Cursor(0x40);  // place cursor at cell 0x40
61          sprintf(Line_char_Array,"T PV [F] = %4.1f",TF_PV);
62          Write_LCD_String(Line_char_Array);
63
64          ClrWdt(); // restart the watchdog timer
65      }
66      return 0;
67  }
```

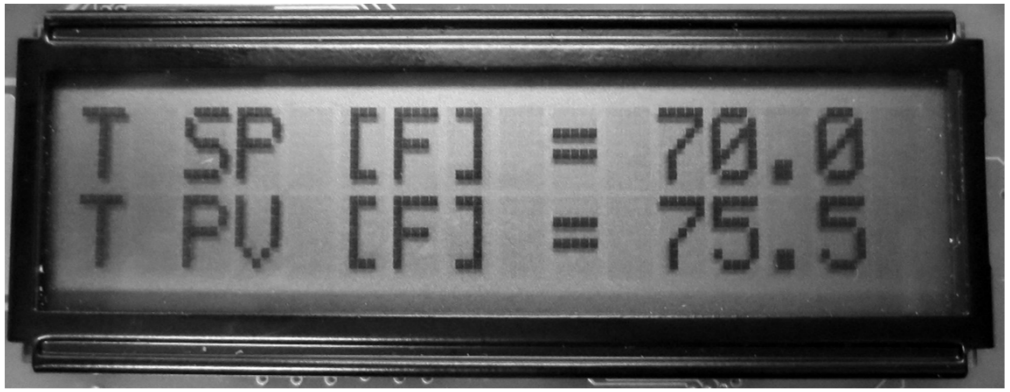
# Exercise 1

- Verify the process variable measurement by pinching the thermal sensor with your fingers
  - Does the process variable temperature increase?

- Based on your understanding of the C code, is the true sample period equal to or greater than 500 ms?

# Exercise 2

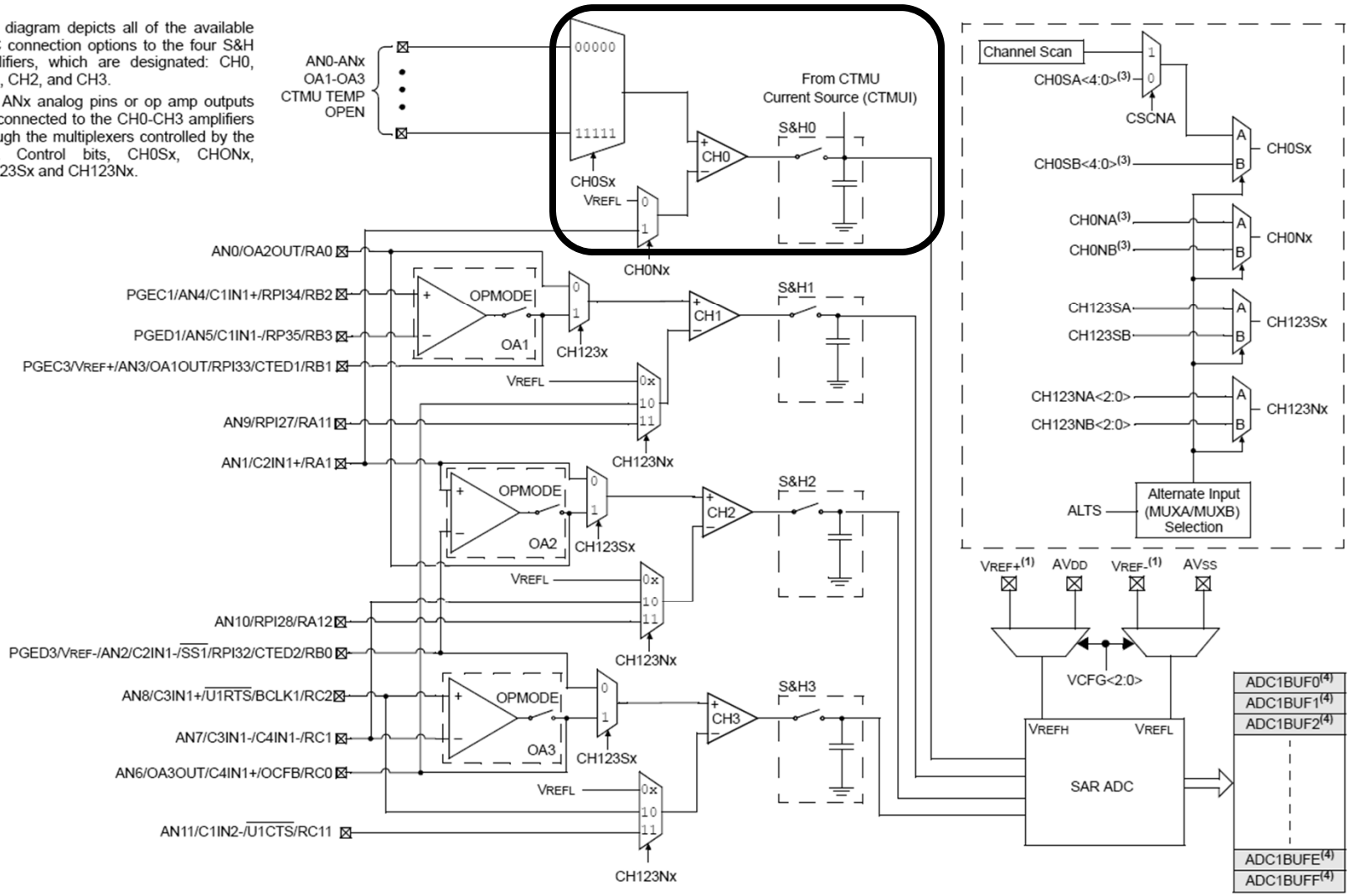
- Use a potentiometer to generate a set point temperature
  - Acquire the analog signal using AN1
  - Linearly map the input voltage (0 to 3.3 V) to a set point temperature (70 to 99 °F)
- Display the set point (SP) temperature and measured process variable (PV) temperature on lines 1 and 2 of the LCD module



# Set Analog Input Multiplexers

This diagram depicts all of the available ADC connection options to the four S&H amplifiers, which are designated: CH0, CH1, CH2, and CH3.

The ANx analog pins or op amp outputs are connected to the CH0-CH3 amplifiers through the multiplexers controlled by the SFR Control bits, CH0Sx, CHONx, CH123Sx and CH123Nx.



# AD1CHS0

**REGISTER 23-6: AD1CHS0: ADC1 INPUT CHANNEL 0 SELECT REGISTER**

| R/W-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CH0NB | — | — | CH0SB<4:0> | | | | |
| bit 15 | | | | | | | bit 8 |

| R/W-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CH0NA | — | — | CH0SA<4:0> | | | | |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 15 **CH0NB:** Channel 0 Negative Input Select for Sample MUXB bit
1 = Channel 0 negative input is AN1(1)
0 = Channel 0 negative input is VREFL

bit 14-13 **Unimplemented:** Read as '0'

bit 12-8 **CH0SB<4:0>:** Channel 0 Positive Input Select for Sample MUXB bits(1)
11111 = Open; use this selection with CTMU capacitive and time measurement
11110 = Channel 0 positive input is connected to CTMU temperature measurement diode (CTMU TEMP)
11101 = Reserved
11100 = Reserved
11011 = Reserved
11010 = Channel 0 positive input is output of OA3/AN6(2)
11001 = Channel 0 positive input is output of OA2/AN0(2)
11000 = Channel 0 positive input is output of OA1/AN3(2)
10111 = Reserved
•
•
•
10000 = Reserved
01111 = Channel 0 positive input is AN15(3)
01110 = Channel 0 positive input is AN14(3)
01101 = Channel 0 positive input is AN13(3)
•
•
•
00010 = Channel 0 positive input is AN2(3)
00001 = Channel 0 positive input is AN1(3)
00000 = Channel 0 positive input is AN0(3)

bit 7 **CH0NA:** Channel 0 Negative Input Select for Sample MUXA bit
1 = Channel 0 negative input is AN1(1)
0 = Channel 0 negative input is VREFL

bit 6-5 **Unimplemented:** Read as '0'

bit 4-0 **CH0SA<4:0>:** Channel 0 Positive Input Select for Sample MUXA bits(1)
11111 = Open; use this selection with CTMU capacitive and time measurement
11110 = Channel 0 positive input is connected to CTMU temperature measurement diode (CTMU TEMP)
11101 = Reserved
11100 = Reserved
11011 = Reserved
11010 = Channel 0 positive input is output of OA3/AN6(2)
11001 = Channel 0 positive input is output of OA2/AN0(2)
11000 = Channel 0 positive input is output of OA1/AN3(2)
10110 = Reserved
•
•
•
10000 = Reserved
01111 = Channel 0 positive input is AN15(1)
01110 = Channel 0 positive input is AN14(1)
01101 = Channel 0 positive input is AN13(1)
•
•
•
00010 = Channel 0 positive input is AN2(1)
00001 = Channel 0 positive input is AN1(1)
00000 = Channel 0 positive input is AN0(1)

- Page 329 of dsPIC33 data sheet
- Choose CH-0 positive input by setting AD1CHS0

PennState College of Engineering

# Exercise 2

- Use a single C function that returns the analog voltage at either pin 2 (AN0) or pin 3 (AN1)
  - Set AD1CHS0 = 0x000 for ADC on pin 2 (AN0)
  - Set AD1CHS0 = 0x001 for ADC on pin 3 (AN1)

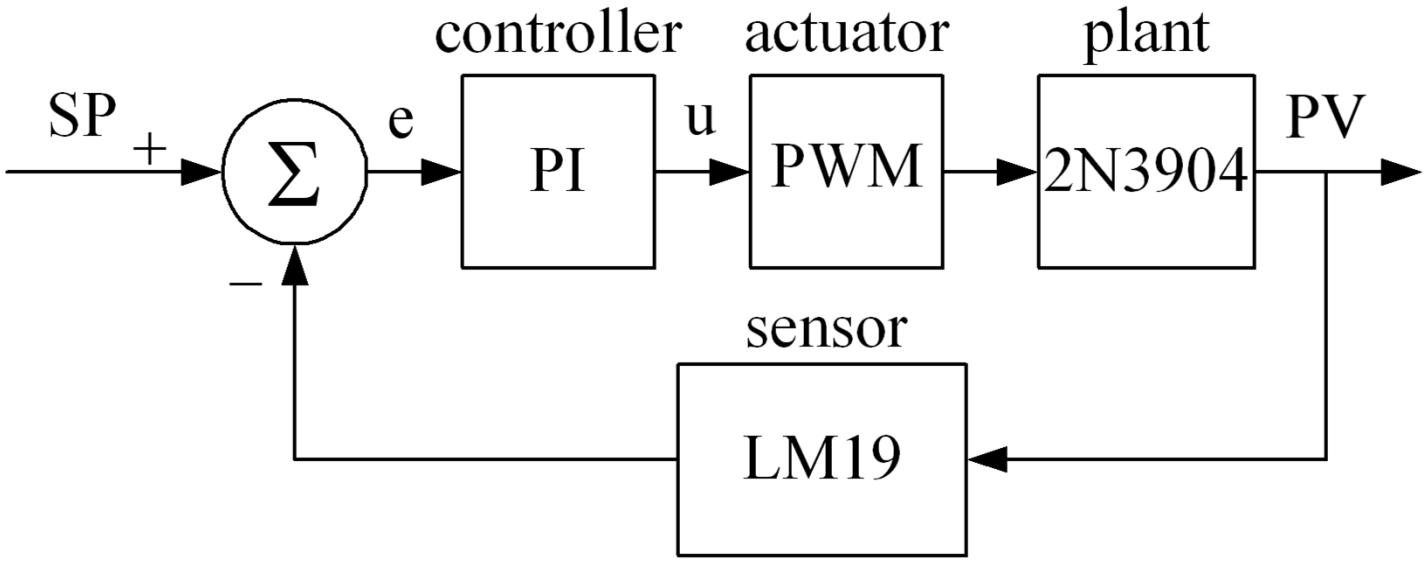
```
36  double Read_ADC(int channel) {
37      double Vmeas;
38
39      AD1CHS0 = channel;              // choose CH-0 positive input
40      AD1CON1bits.SAMP = 1;           // start Sample; place inp signal across cap
41      __delay_us(10);                 // wait for cap voltage to track analog inp
42      AD1CON1bits.SAMP = 0;           // start SA converter
43      while (!AD1CON1bits.DONE);      // wait for SA conv to complete
44      Vmeas = 3.3 * (double) ADC1BUF0 / 1023.0; // convert to units of Volts
45
46      return Vmeas;
47  }
```

# Exercise 2

```c
/* define FCY before including libpic30.h */
#define FCY 3685000UL

#include <p33EP64MC502.h>
#include <libpic30.h>
#include <stdio.h>
#include <math.h>
#include <xc.h>
#include "EE200_LCD.h"

/* set Configuration Bits */
#pragma config ICS = PGD2    // communicate on PGED2 (pin 14) and PGEC2 (pin 15)
#pragma config JTAGEN = OFF  // disable JTAG inorder to use RB8 through RB11

/* declare functions */
void Init_ADC(void);
double Read_ADC(int channel);

void Init_ADC (void) {
    ANSELAbits.ANSA0 = 1;  // set pin 2 (ANO) for analog input
    TRISAbits.TRISA0 = 1;  // configure for input (dispable DO)

    ANSELAbits.ANSA1 = 1;  // set pin 3 (AN1) for analog input
    TRISAbits.TRISA1 = 1;  // configure for input (dispable DO)

    AD1CON1bits.ADON = 1;  // turn ADC module on
  }

double Read_ADC(int channel) {
    double Vmeas;

    AD1CHS0 = channel;             // choose CH-0 positive input
    AD1CON1bits.SAMP = 1;          // start Sample; place inp signal across cap
    __delay_us(10);                // wait for cap voltage to track analog inp
    AD1CON1bits.SAMP = 0;          // start SA converter
    while (!AD1CON1bits.DONE);     // wait for SA conv to complete
    Vmeas = 3.3 * (double) ADC1BUF0 / 1023.0; // convert to units of Volts

    return Vmeas;
}
```

```c
int main(void) {
    char Line_char_Array[16];
    unsigned int TS;
    double V_PV, V_SP, TC, TF_PV, TF_SP;

    /* Set Parameters */
    TS = 500; // sample period in ms

    Init_LCD_Module();
    Init_ADC();

    while (1) {
       __delay_ms(TS);      // set sample rate to 500 ms

       /* Acquire analog input signals */
       V_PV = Read_ADC(0); // read PV voltage on pin 2 (ANO)
       V_SP = Read_ADC(1); // read SP voltage on pin 3 (AN1)

       /* Determine process variable temp in degrees F */
       TC = -1481.96 + sqrt(2.1962e6 + (1.8639 - V_PV)/3.88e-6);
       TF_PV = (9.0/5.0)*TC + 32.0;

       /* Determine setpoint temp in degrees F */
       TF_SP = 29.0*V_SP/3.3 + 70.0;

       /* Update LCD */
       Position_LCD_Cursor(0x00); // place cursosr at cell 0x00
       sprintf(Line_char_Array,"T SP [F] = %4.1f", TF_SP);
       Write_LCD_String(Line_char_Array);

       Position_LCD_Cursor(0x40); // place cursosr at cell 0x40
       sprintf(Line_char_Array,"T PV [F] = %4.1f", TF_PV);
       Write_LCD_String(Line_char_Array);

       ClrWdt(); // restart the watchdog timer
    }
    return 0;
}
```

# Block Diagram Representation




$e(t) \Rightarrow$ error signal (temperature)
$u(t) \Rightarrow$ plant input (duty cycle)
$PV(t) \Rightarrow$ process variable (temperature)

# PI Controller

- Representation

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau)\, d\tau$$

- The proportional ($K_P$) and integral ($K_I$) gains are design parameters chosen to obtain desired
  - Transient response characteristics
  - Steady-state accuracy

- EE 380 and EE 482 show how to determine control gains for analog and digital control systems, respectively

# Sampled-Data Control

- The output voltage $V_o$ of the LM19 is sampled in time
- Signals in the microcontroller are indexed in time by k

$$V_o(k) = V_o(t)\Big|_{t=kT}$$

$$T \Rightarrow \text{sample period}$$

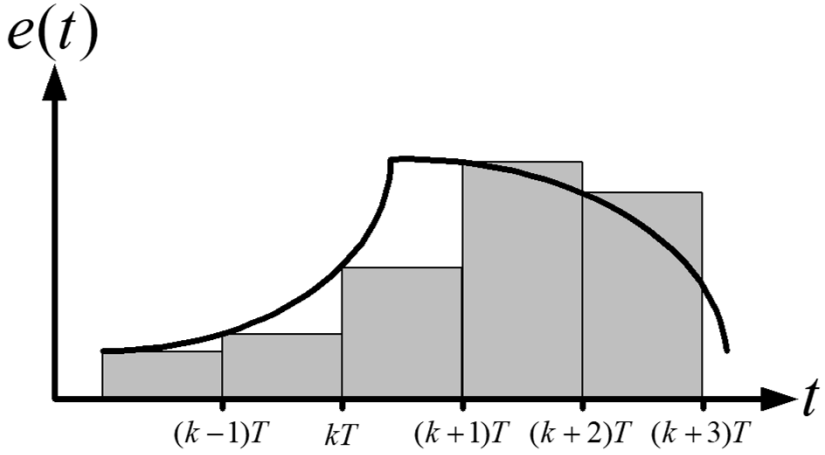
- The amplitude of $V_o(k)$ is quantized as it is represented by a finite-length register

# Approximating the Integral

- Using the samples e(k), the microcontroller must approximate

$$x(t) = \int_0^t e(\tau)\, d\tau$$

- Represent the area x as a sum of rectangles whose width is the sample period T and whose height is e(k)




$$\begin{aligned} x(k) &= \sum_{n=0}^{k} e(n)T \\ &= \sum_{n=0}^{k-1} e(n)T + Te(k) \\ &= x(k-1) + Te(k) \end{aligned}$$

# PI Controller

- Analog Representation

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau) d\tau$$

- Sampled Data Representation

$$x(k) = x(k-1) + Te(k)$$

$$u(k) = K_P e(k) + K_I x(k)$$

# Exercise 3

- Realize a PI controller using
  - sample period $T_S$ = 500 ms
  - proportional gain $K_P$ = 0.25
  - integral gain $K_I$ = 0.05
- Drive the 2N3904 NPN transistor using the PWM signal PWM1H (pin 25)
  - PWM frequency = 1 kHz (PTPER = 7370)
  - Limit the fractional duty cycle between 0 and 1 (PDC1 between 0 and 7370)

# Exercise 3

- Use the LCD module to display:
  - set point (SP) temperature [°F]
  - measured process variable (PV) temperature [°F]
  - error (e) [°F] where e = SP – PV
  - fractional duty cycle (D)

# Exercise 3

```c
/* define FCY before including libpic30.h */
#define FCY 3685000UL

#include <p33EP64MC502.h>
#include <libpic30.h>
#include <stdio.h>
#include <math.h>
#include <xc.h>
#include "EE200_LCD.h"

/* set Configuration Bits */
#pragma config ICS = PGD2    // communicate on PGED2 (pin 14) and PGEC2 (pin 15)
#pragma config JTAGEN = OFF  // disable JTAG inorder to use RB8 through RB11
#pragma config PWMLOCK = OFF // PWM Lock Enabl Bit

/* declare functions */
void Init_ADC(void);
void Init_PWM_module(void);
double Read_ADC(int channel);

void Init_ADC (void) {
    ANSELAbits.ANSA0 = 1;          // set pin 2 (AN0) for analog input
    TRISAbits.TRISA0 = 1;          // configure for input (disable DO)

    ANSELAbits.ANSA1 = 1;          // set pin 3 (AN1) for analog input
    TRISAbits.TRISA1 = 1;          // configure for input (disable DO)

    AD1CON1bits.ADON = 1;          // turn ADC module on
  }

```

```c
double Read_ADC(int channel) {
    double Vmeas;

    AD1CHS0 = channel;             // choose CH-0 positive input
    AD1CON1bits.SAMP = 1;          // start Sample; place inp signal across cap
    __delay_us(10);                // wait for cap voltage to track analog inp
    AD1CON1bits.SAMP = 0;          // start SA converter
    while (!AD1CON1bits.DONE);     // wait for SA conv to complete
    Vmeas = 3.3 * (double) ADC1BUF0 / 1023.0; // convert to units of Volts

    return Vmeas;
}

void Init_PWM_Module(void) {
    PTCONbits.PTEN = 0;      // disable the PWM module
    PTPER = 7370;            // period as a multiple of clock cycles
    PDC1  = 3685;            // fract duty cycle as a multiple of clock cycles
    IOCON1bits.PMOD  = 1;    // PWM1 I/O pin pair is in redundant mode
    IOCON1bits.PENH  = 1;    // PWM module controls pin 25, PWM1H

    /* RB10 to RB13 must be configured to control the LCD module */
    IOCON2bits.PENL  = 0;    // GPIO module controls pin 24, RB13
    IOCON2bits.PENH  = 0;    // GPIO module controls pin 23, RB12
    IOCON3bits.PENL  = 0;    // GPIO module controls pin 22, RB11
    IOCON3bits.PENH  = 0;    // GPIO module controls pin 21, RB10

    FCLCON1bits.FLTMOD = 3; // fault input disabled
    PTCONbits.PTEN = 1;      // enable the PWM module
}

```

# Exercise 3

```
68  int main(void) {
69      char Line_char_Array[16];
70      unsigned int TS;
71      double V_PV, V_SP, TC, TF_PV, TF_SP, KP, KI, x, e, u;
72
73      /* Set Parameters */
74      TS = 500;               // sample period in ms
75      KP = 0.25;              // proportional gain
76      KI = 0.05;              // integral gain
77      x   =  0;               // initial value of integrator
78
79      Init_ADC();
80      Init_PWM_Module();
81      Init_LCD_Module();
82
```

```
83      while (1) {
84          __delay_ms(TS);      // set sample rate to 500 ms
85
86          /* Acquire analog input signals */
87          V_PV = Read_ADC(0); // read PV voltage on pin 2 (AN0)
88          V_SP = Read_ADC(1); // read SP voltage on pin 3 (AN1)
89
90          /* Determine process variable temp in degrees F */
91          TC = -1481.96 + sqrt(2.1962e6 + (1.8639 - V_PV)/3.88e-6);
92          TF_PV = (9.0/5.0)*TC + 32.0;
93
94          /* Determine setpoint temp in degrees F */
95          TF_SP = 29.0*V_SP/3.3 + 70.0;
96
97          /* Realize PI Controller */
98           e = TF_SP - TF_PV;
99           x = x + TS*e/1000;
100          u = KP*e + KI*x;
101
102         /* Limit fractional duty cycle between 0 and 1 */
103         if (u > 1)
104             u = 1;
105         else if (u < 0)
106             u = 0;
107
108         /* Set PWM duty cycle */
109         PDC1  = floor(7370 * u);
110
111         /* Update LCD  */
112         Position_LCD_Cursor(0x00); // place cursosr at cell 0x00
113         sprintf(Line_char_Array,"SP %4.1f  e %+4.2f", TF_SP, e);
114         Write_LCD_String(Line_char_Array);
115
116         Position_LCD_Cursor(0x40); // place cursosr at cell 0x40
117         sprintf(Line_char_Array,"PV %4.1f  D  %4.2f", TF_PV, u);
118         Write_LCD_String(Line_char_Array);
119
120         ClrWdt(); // restart the watchdog timer
121     }
122     return 0;
123 }
```

# Exercise 3

- For a fixed set point temperature, the process variable, error, and fractional duty cycle approach steady-state values

- In steady-state, the power dissipated by the 2N3904 NPN transistor matches the heat loss of the LM19/2N3904 structure to the environment

- Verify operation of the closed-loop system by bringing the blade of a room temperature screwdriver in contact with the case of the 2N3904 and/or LM19
  - Should the fractional duty cycle increase or decrease? Why?