ADC

Ports-

9.0,8.4,8.2

Setup Code-

```
char ch;
   int temp1 = 0;
   int temp2 = 0;
   int temp3 = 0;
   int average1 = 0;
   int average2= 0;
   int average3= 0;
   int data1[1000];
   int data2[1000];
   int data3[1000];
   int min1 = 0;
   int min2 = 0;
   int min3 = 0;
   int max1 = 0;
   int max2 = 0;
   int max3 = 0;
   double stddev1 = 0;
   double stddev2 = 0;
   double stddev3 = 0;
   int bump = 0, lastbump=0;
   uint8_t* where;
   uint32_t try32;
   uint16_t try16;
   uint8_t out;
   uint32_t n = 0;
   Clock_Init48MHz();    // makes SMCLK=12 MHz
   UART0_Initprintf();   // initialize UART and printf
   UART0_OutString("\nTest program for UART and ADC\n");
   P8->SEL0 &= ~0xE1;    // configure P8 0,5,6,7 GPIO output
   P8->SEL1 &= ~0xE1;
   P8->DIR |= 0xE1;      // P8 0,5,6,7 output
   where = &(P8->DIR);
   P4->SEL0 &= ~0xFD;
   P4->SEL1 &= ~0xFD;   // 1) configure P4 as GPIO (except .1)
   P4->DIR &= ~0xFD;    // 2) make P4 in
   P4->REN |= 0xFD;     // 3) enable pull resistors on P4
//    P6->DIR |= 0x04;
//    P6->OUT = 0;
//    P10->DIR |= 0x05;
//    P10->OUT = 0;
   P1->DIR |= 0x01;
```

```
  P1->OUT = 0;
  ADC0_InitSWTriggerCh17_21_23();
```

Main While Code-

```
while (1) {
//      P6->OUT |= 0x04;
//      P10->OUT |= 0x05;
//      Clock_Delay1ms(1);
      P1->OUT |= 0x01
      ADC_In17_21_23(&v17, &v21, &v23);
//      P6->OUT &= ~0x04;
//      P10->OUT &= ~0x05;
      P1->OUT &= ~0x01
      data1[n]=v17;
      data2[n]=v21;
      data3[n]=v23;
      temp1 = temp1 + data1[n];
      temp2 = temp2 + data2[n];
      temp3 = temp3 + data3[n];
      n++;
      if(n == 1000){
         average1 = temp1/1000;
         average2 = temp2/1000;
         average3 = temp3/1000;
         for(int i=0;i<n;i++){
            stddev1 += pow(data1[i]-average1,2);
            stddev2 += pow(data2[i]-average2,2);
            stddev3 += pow(data3[i]-average3,2);
            if(data1[i] < data1[min1]){
               min1 = i;
            }
            if(data2[i] < data2[min2]){
               min2 = i;
            }
            if(data3[i] < data3[min3]){
               min3 = i;
            }
            if(data1[i] > data1[max1]){
               max1 = i;
            }
            if(data2[i] > data2[max2]){
               max2 = i;
            }
            if(data3[i] > data3[max3]){
               max3 = i;
            }
         }
```

```
        printf("Average1:%6d Average2:%6d Average3:%6d \r\n", average1, average2, average3);
        printf("Stardard Dev1: %6f Stardard Dev2: %6f Stardard Dev3: %6f
\r\n",sqrt(stddev1/1000),sqrt(stddev2/1000),sqrt(stddev3/1000));
        printf("Min1: %6d Min2: %6d Min3: %6d Max1: %6d Max2: %6d Max3: %6d \r\n",
data1[min1],data2[min2],data3[min3],data1[max1],data2[max2],data3[max3]);
        temp1 = 0;
        temp2 = 0;
        temp3 = 0;
        stddev1 = 0;
        stddev2 = 0;
        stddev3 = 0;
        min1=0;
        min2=0;
        min3=0;
        max1=0;
        max2=0;
        max3=0;
        n = 0;
    }


    //printf("%6d %6d %6d\r\n", v17, v21, v23);
    bump = (int) (P4->IN&0xFD);
    if ((EUSCI_A0->IFG&0x01) != 0) {
        printf("%c\r\n", (char)(EUSCI_A0->RXBUF));
        where = &(P8->OUT);
        out = *where;
        out = out ^ 0xE1;
        *where = out;
        // verbose version of P8->OUT ^= 0xE1;  // toggle
    }
    if (bump != lastbump) {
        printf("%x\r\n", (int) (bump));
        lastbump = bump;
        where = &(P8->OUT);
        out = *where;
        out = out ^ 0xE1;
        *where = out;
        // verbose version of P8->OUT ^= 0xE1;  // toggle
    }
```

ISR Code-

N/A

Event Timer-

Pins-

6.2,10.2,10.4

Setup Code-

```
int increments[] = {3000,300,2700,24000,3000,300,300,26400,3000,300,2700,24000};
int i = 0;
WDT_A->CTL = WDT_A_CTL_PW |           // Stop WDT
        WDT_A_CTL_HOLD;

  // Configure GPIO
  P1->DIR |= BIT0;
  P1->OUT |= BIT0;
  P6->DIR |= BIT2;
  P6->OUT |= BIT2;
  P10->DIR |= BIT2 | BIT4;
  P10->OUT = BIT2 | BIT4;
  TIMER_A0->CCTL[0] = TIMER_A_CCTLN_CCIE; // TACCR0 interrupt enabled
  TIMER_A0->CCR[0] = 50000;
  TIMER_A0->CTL = TIMER_A_CTL_SSEL__SMCLK | // SMCLK, continuous mode
        TIMER_A_CTL_MC__CONTINUOUS;

  SCB->SCR |= SCB_SCR_SLEEPONEXIT_Msk;    // Enable sleep on exit from ISR

  // Ensures SLEEPONEXIT takes effect immediately
  __DSB();

  // Enable global interrupt
  __enable_irq();

  NVIC->ISER[0] = 1 << ((TA0_0_IRQn) & 31);
```

Main loop Code:

```
while (1)
  {
    __sleep();

    __no_operation();                // For debugger
  }
```

ISR Code:

```
void TA0_0_IRQHandler(void) {
  TIMER_A0->CCTL[0] &= ~TIMER_A_CCTLN_CCIFG;
  P1->OUT ^= BIT0;
  if(i > 11){
        i=0;
  }
  switch(i){
    case 0:
      P6->OUT |= BIT2;
```

```
        break;
      case 1:
        //adc
        break;
      case 2:
        P6->OUT &= ~BIT2;
        break;
      case 3:
        //adc
        break;
      case 4:
        P10->OUT |= BIT2;
        break;
      case 5:
        //adc
        break;
      case 6:
        P10->OUT &= ~BIT2;
        break;
      case 7:
        //adc
        break;
      case 8:
        P10->OUT |= BIT4;
        break;
      case 9:
        //adc
        break;
      case 10:
        P10->OUT &= ~BIT4;
        break;
      case 11:
        //adc
        break;
    }
    TIMER_A0->CCR[0] += increments[i];          // Add Offset to TACCR0
    i++;
}
```

Motor PWM

Pins-

2.6,2.7

Setup Code-

```
int tick = 0;
int n = 100;
 WDT_A->CTL = WDT_A_CTL_PW |          // Stop WDT
```

```
        WDT_A_CTL_HOLD;

   // Configure GPIO
//    P7->DIR |= BIT6 | BIT7;              // P7.6~7 set TA1.1~2
//    P7->SEL0 |= BIT6 | BIT7;
//    P7->SEL1 &= ~(BIT6 | BIT7);

   P2->DIR |= BIT6 | BIT7;
   P2->SEL0 |= BIT6 | BIT7;
   P2->SEL1 &= ~(BIT6 | BIT7);
   //Pin7
//    TIMER_A1->CCR[0] = 1000 - 1;          // PWM Period
//    TIMER_A1->CCTL[1] = TIMER_A_CCTLN_OUTMOD_7; // CCR1 reset/set
//    TIMER_A1->CCR[1] = 750;               // CCR1 PWM duty cycle
//    TIMER_A1->CCTL[2] = TIMER_A_CCTLN_OUTMOD_7; // CCR2 reset/set
//    TIMER_A1->CCR[2] = 250;               // CCR2 PWM duty cycle
   //Pin2
   TIMER_A0->CCR[0] = 1000 - 1;          // PWM Period
   TIMER_A0->CCTL[0] = TIMER_A_CCTLN_CCIE; // TACCR0 interrupt enabled
   TIMER_A0->CCTL[3] = TIMER_A_CCTLN_OUTMOD_7; // CCR1 reset/set
   TIMER_A0->CCR[3] = 750;               // CCR1 PWM duty cycle
   TIMER_A0->CCTL[4] = TIMER_A_CCTLN_OUTMOD_7; // CCR2 reset/set
   TIMER_A0->CCR[4] = 250;               // CCR2 PWM duty cycle
   TIMER_A0->CTL = TIMER_A_CTL_SSEL__SMCLK | // SMCLK
         TIMER_A_CTL_MC__UP |          // Up mode
         TIMER_A_CTL_CLR | TIMER_A_CTL_IE;             // Clear TAR

  // __DSB();
   __enable_irq();

   NVIC->ISER[0] = 1 << ((TA0_0_IRQn) & 31);
```

Main Loop Code-

```
while(1){
      if(tick>=n){
         tick = 0;
         TIMER_A0->CCR[3] += n;
         TIMER_A0->CCR[4] += n;
      }
      else if(TIMER_A0->CCR[3]>=999){
         TIMER_A0->CCR[3]=0;
      }
      else if(TIMER_A0->CCR[4]>=999){
         TIMER_A0->CCR[4]=0;
      }
   }
```

ISR Code:

```
void TA0_0_IRQHandler(void) {
   TIMER_A0->CCTL[0] &= ~TIMER_A_CCTLN_CCIFG;
   tick++;
}
```

LED Control-

Pins-

6.2,10.2,10.4

Setup Code-

```
   P6->DIR |= 0x04;
   P6->OUT = 0;
   P10->DIR |= 0x05;
   P10->OUT = 0;
   P1->DIR |= 0x01;
   P1->OUT = 0;
```

Main Loop Code-

```
P6->OUT |= 0x04;
P10->OUT |= 0x05;
P6->OUT &= ~0x04;
P10->OUT &= ~0x05;
```