

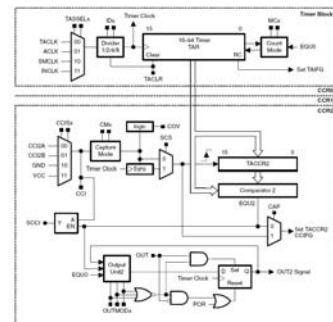
Build new Project

- Select correct microcontroller
- Select FET debugger
- COP disable
- Set bus frequency

```
void main( void ){
    WDTCTL = WDTPW + WDTCTL; // Stop watchdog timer
    P1DIR |= 0x01;           // Set P1.0 to output direction
    BCSCTL1 = CALBC1_16MHZ; // running at 16 MHz
    DCOCTL = CALDCO_16MHZ;
    for (;;) {
        volatile unsigned short i; // prevent optimization
        P1OUT ^= 0x01;           // Toggle P1.0
        i = 65535;               // Delay
        do{
            i--;
        }
        while (i != 0);
    }
}
```

TimerA2

- Single thread periodic event
- SMCLK is bus clock
- MCx continuous mode
- TACCR2 = TACCR2+period



```
;TimerA2, 2MHz from 16MHz SMCLK, TAIFG disarmed
TimerA2_Init MACRO
    mov.w #002E0h,&TACTL    ;SMCLK, /8, continuous
    mov.w #02010h,&TACCTL0  ;compare mode, arm CCIFG
    mov.w #00000h,&TACCTL1  ;TACCR1 not used
    mov.w #00000h,&TACCR0   ;time for first interrupt
    ENDM
;Synchronize to periodic clock
;Sleep in LPM1 until it is time to sample ADC
TimerA2_Sync MACRO period
    mov.w &TACCR0,R12      ;time of last interrupt
    add.w #period,R12      ;time for next interrupt
    mov.w R12,&TACCR0
    bic.w #00001h,&TACCTL0 ;clear CCIFG
    bis.w #LPM1+GIE,SR     ;low power sleep, wait for
    ENDM
; Interrupt called when periodic timer complete
; allows CPU to sleep during free time
TACCR0ISR
;clears CCIFG automatically
    bic.w #LPM1,0(SP)      ;exit low power sleep
    reti
```

SPI

- Master
- 16-bit data
- Busy-wait


```

mov.w #address,&ADC10SA      ; starting address
bis.w #00003h,&ADC10CTL0    ; ENC+ADC10SC, start
bis.w #LPM3+GIE,SR         ; low power sleep, wait
ENDM

; Interrupt called when ADC complete
; allows CPU to sleep during conversion
ADC10ISR
    bic.w #ADC10IFG,&ADC10CTL0 ; clear ADC10IFG
    xor.b #001h,&P1OUT        ; Toggle P1.0
    bic.w #LPM3,0(SP)        ; exit low power sleep
    reti

adapted from slaa335, go to www.ti.com, search "slaa335"
unsigned short buffer[4];           // buffer for 4 ADC samples
void Sample (void){
    unsigned volatile short i;
    ADC10CTL0 = ADC10ON + REFON + ADC10SHT_1 + MSC + ADC10IE + SREF_1;
    // ADC on, ref = 1.5V, sampling = 8 clocks
    // NOTE: REF takes 30us to settle, But because of other instructs
    // no need for additional delay loop
    ADC10CTL1 = INCH_4 + CONSEQ_2;    // Chan A4, repeat single channel
    ADC10DTC1 = 4;                   // Do 4 conversions
    ADC10SA = (unsigned short)buffer; // Start address for DTC
    i = 6;                             // Delay for OA settling
    do i--;
    while (i != 0);
    ADC10CTL0 |= ENC + ADC10SC;      // Enable and start conversions
    LPM3;                             // Enter LPM3 while 4 conversions made
    ADC10CTL0 &= ~ENC;              // Clear ENC to stop conversion
    ADC10CTL0 = 0;                  // Turn off ADC and reference
}
// ADC10 interrupt service routine
#pragma vector=ADC10_VECTOR
__interrupt void ADC10 (void){
    LPM3_EXIT;
}

```

16-bit sigma delta ADC, go to www.ti.com, search "slaa283a"
10-bit ADC, go to www.ti.com, search "slaa309"