



King Mongkut's University of Technology Thonburi

Department of Electronics and Telecommunication Engineering Faculty of Engineering

EIE/ENE 335 Digital Circuit and Microprocessor Lab

for the 3rd year student

Experiment: Analog-to-Digital Converter (ADC)

Objectives

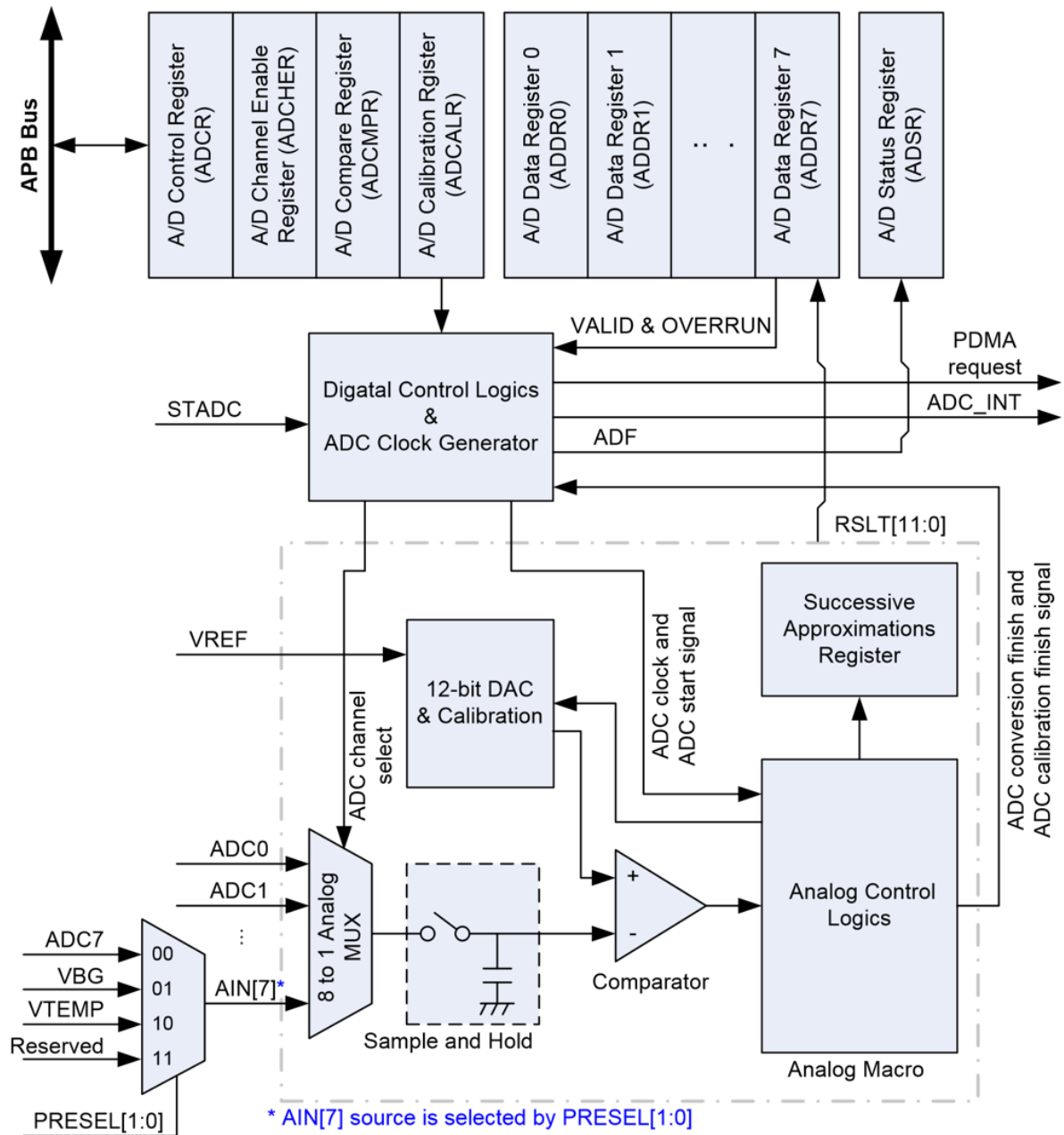
- How to use
 - o the NuMicro™ NUC100 series driver to do the fast application software development
 - o ADC

Background Theory

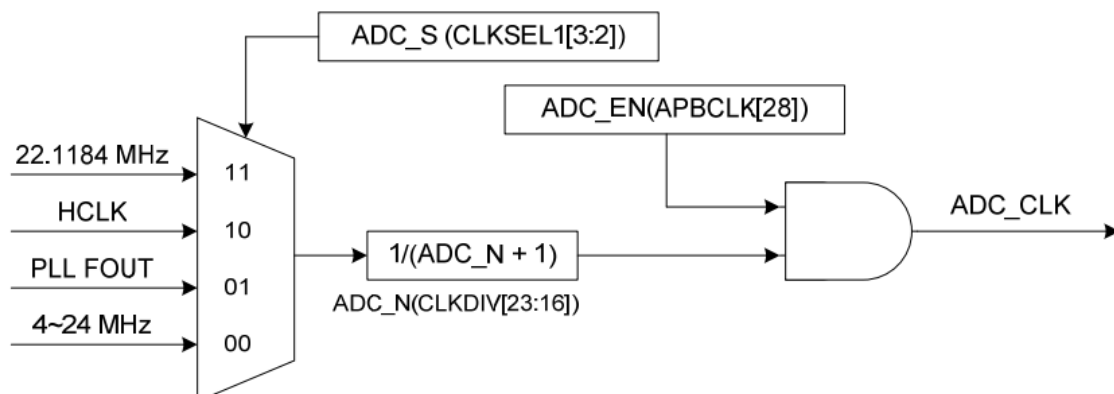
NuMicro™ NUC100 Series contains one 12-bit successive approximation analog-to-digital converters (SAR A/D converter) with 8 input channels. The A/D converter supports three operation modes: single, single-cycle scan and continuous scan mode. The A/D converters can be started by software and external STADC pin.

- Up to 8 single-end analog input channels or 4 differential analog input channels
- Maximum ADC clock frequency is 16 MHz
- Up to 700K SPS conversion rate
- Three operating modes
 - o Single mode: A/D conversion is performed one time on a specified channel
 - o Single-cycle scan mode: A/D conversion is performed one cycle on all specified channels with the sequence from the lowest numbered channel to the highest numbered channel
 - o Continuous scan mode: A/D converter continuously performs Single-cycle scan mode until software stops A/D conversion
- An A/D conversion can be started by
 - o Software write 1 to ADST bit
 - o External pin STADC
- Conversion results are held in data registers for each channel with valid and overrun indicators
- Conversion result can be compared with specify value and user can select whether to generate an interrupt when conversion result is equal to the compare register setting

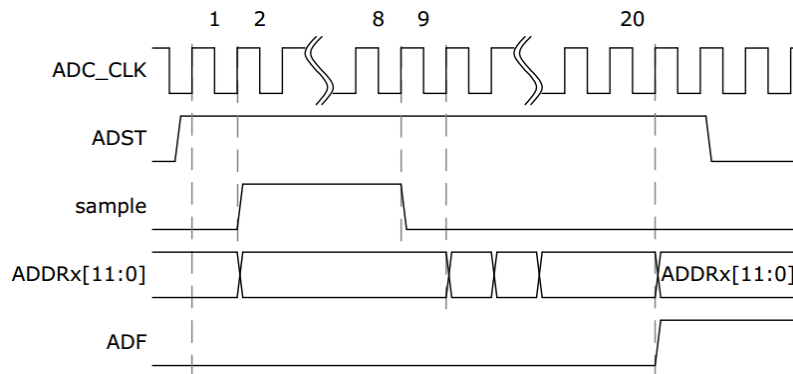
Block Diagram



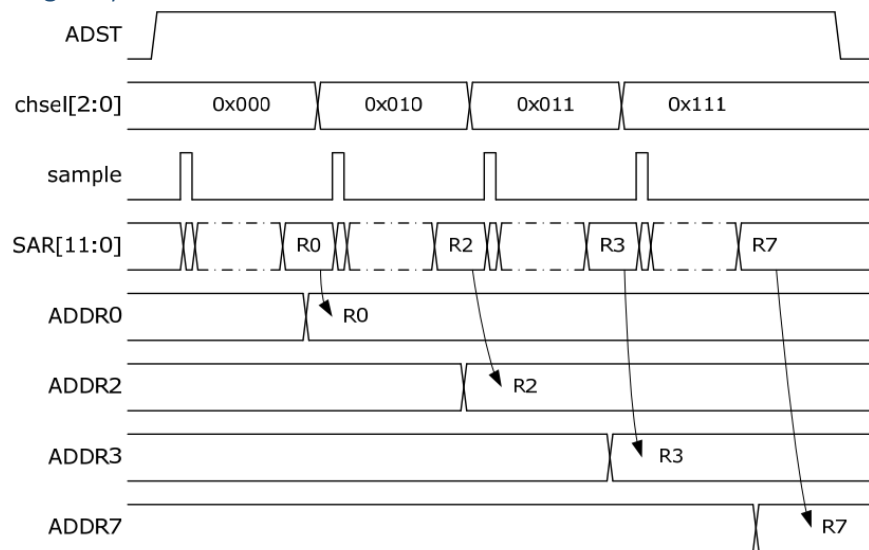
ADC Clock Control



Single Mode



Single-Cycle Scan Mode



Single-cycle scan on channel 0, 2, 3 and 7 (ADCHER[7:0] = 0x10001101b)

```

20 //-----ADC7
21 void InitADC(void) {
22     /* Step 1. GPIO initial */
23     GPIOA->OFFD |= 0x00800000; // Disable digital input path (when input is analog signal)
24     SYS->GPAMFP.ADC7_SS21_AD6 = 1; // Set ADC function
25
26     /* Step 2. Enable and Select ADC clock source, and then enable ADC module */
27     SYSCLK->CLKSEL1.ADC_S = 3; // Select 22Mhz for ADC
28     // 0:12MHz, 1:PLL FOUT, 2:HCLK, 3:22.1184 MHz
29     SYSCLK->CLKDIV.ADC_N = 1; // ADC clock source = 22Mhz/2 =11Mhz;
30     // The ADC clock frequency = (ADC clock source frequency)/(ADC_N+1) ;8-bits
31     SYSCLK->APBCLK.ADC_EN = 1; // Enable clock source
32     ADC->ADCR.ADEN = 1; // Enable ADC module
33
34     /* Step 3. Select Operation mode */
35     ADC->ADCR.DIFFEN = 0; // Single-end analog input mode
36     ADC->ADCR.ADMO = 0; // A/D Converter Operation Mode
37     // 0:Single conversion, 2:Single-cycle scan, 3:Continuous scan
38
39     /* Step 4. Select ADC channel */
40     ADC->ADCHER.CHEN = 0x0080; // 8-bits -> ch7
41
42     /* Step 5. Enable ADC interrupt */
43     //ADC->ADSR.ADF = 1; // clear the A/D interrupt flags
44     //ADC->ADCR.ADIE = 1;
45     //NVIC_EnableIRQ(ADC_IRQn);
46
47     /* Step 6. A/D Conversion Start */
48     ADC->ADCR.ADST = 1;
49     // ADST will be cleared to 0 by hardware automatically
50     // at the ends of single mode and single cycle scan mode.
51 }

```

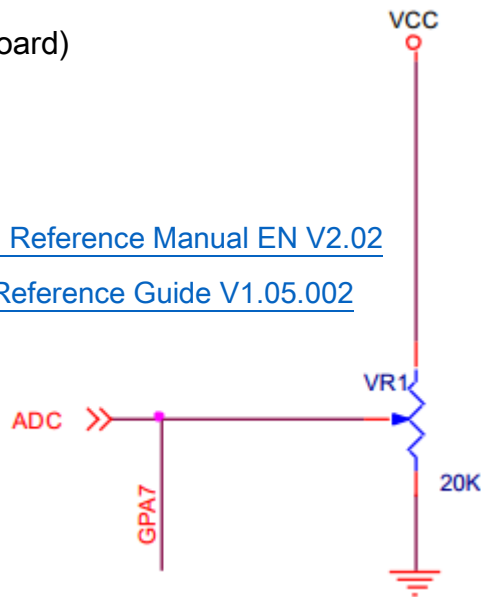
Equipment required

- Nu_LB-002 (Nuvoton learning board)

Reference:

1. [Nu_LB-002 Rev 2.1 User's Manual](#)
2. [NuMicro™ NUC130_140 Technical Reference Manual EN V2.02](#)
3. [NuMicro™ NUC100 Series Driver Reference Guide V1.05.002](#)

ADC connect GPA7



Procedure 1: ADC connect GPA7

1. Replace the content of the 'Smpl_Start_Kit.c' with the '[ADC](#)' lab file.
2. Compile the project, and run the program.
3. Study the program and answer the following questions.

```

52 //-----PWM0
53 void InitPWM(void) {
54     /* Step 1. GPIO initial */
55     SYS->GPAMFP.PWM0_AD13 = 1;
56
57     /* Step 2. Enable and Select PWM clock source*/
58     SYSCLK->APBCLK.PWM01_EN = 1; // Enable PWM clock
59     SYSCLK->CLKSEL1.PWM01_S = 3; // Select 22.1184Mhz for PWM clock source
60     // 0:12MHz, 1:32.768 kHz, 2:HCLK, 3:22.1184 MHz
61     PWMA->PPR.CP01 = 1; // Prescaler 0~255, Setting 0 to stop output clock
62     PWMA->CSR.CSR0 = 0; // clock divider -> 0:/2, 1:/4, 2:/8, 3:/16, 4:/1
63     // PWM frequency = PWMxy_CLK/[(prescale+1)*(clock divider)*(CNR+1)]
64     // Ex:= 22.1184M/[(1+1)*(2)*(2^16)] = 84.375 Hz -> T = 11.85 ms.
65
66     /* Step 3. Select PWM Operation mode */
67     PWMA->PCR.CH0MOD = 1; // 0:One-shot mode, 1:Auto-load mode
68     //CNR and CMR will be auto-cleared after setting CH0MOD form 0 to 1.
69     PWMA->CNR0 = 0xFFFF; // CMR >= CNR: PWM output is always high
70     PWMA->CMR0 = 0xFFFF;
71     // CMR = 0: PWM low width = (CNR) unit; PWM high width = 1 unit
72
73     PWMA->PCR.CH0INV = 0; // Inverter -> 0:off, 1:on
74     PWMA->PCR.CH0EN = 1; // PWM function -> 0:Disable, 1:Enable
75     PWMA->POE.PWM0 = 1; // Output to pin -> 0:Disable, 1:Enable
76 }

```

```

77 //-----Timer0
78 void InitTIMER0(void) {
79     /* Step 1. Enable and Select Timer clock source */
80     SYSClk->CLKSEL1.TMR0_S = 0; // Select 12Mhz for Timer0 clock source
81     // 0 = 12 MHz, 1 = 32 kHz, 2 = HCLK, 7 = 22.1184 MHz
82     SYSClk->APBCLK.TMR0_EN = 1; // Enable Timer0 clock source
83
84     /* Step 2. Select Operation mode */
85     TIMER0->TCSR.MODE = 1; // 1 -> Select periodic mode
86     // 0 = One shot, 1 = Periodic, 2 = Toggle, 3 = continuous counting mode
87
88     /* Step 3. Select Time out period
89     = (Period of timer clock input) * (8-bit Prescale + 1) * (24-bit TCMP)*/
90     TIMER0->TCSR.PRESCALE = 11; // Set Prescale [0~255]
91     TIMER0->TCMPR = 300000; // Set TCMPR [0~16777215]
92     // (1/12000000)*(11+1)*(300000)= 0.3 sec
93
94     /* Step 4. Enable interrupt */
95     TIMER0->TCSR.IE = 1;
96     TIMER0->TISR.TIF = 1; // Write 1 to clear the interrupt flag
97     NVIC_EnableIRQ(TMR0_IRQn); // Enable Timer0 Interrupt
98
99     /* Step 5. Enable Timer module */
100    TIMER0->TCSR.CRST = 1; // Reset up counter
101    TIMER0->TCSR.CEN = 1; // Enable Timer0
102 }
103
104 void TMR0_IRQHandler(void) { // Timer0 interrupt subroutine
105     char adc_value[15] = "ADC Value:";
106     while(ADC->ADSR.ADF == 0); // A/D Conversion End Flag
107     // A status flag that indicates the end of A/D conversion.
108
109     ADC->ADSR.ADF = 1; // This flag can be cleared by writing 1 to self
110     PWMA->CMR0 = ADC->ADDR[7].RSLT << 4;
111     Show_Word(0,11,' ');
112     Show_Word(0,12,' ');
113     Show_Word(0,13,' ');
114     sprintf(adc_value+10, "%d", ADC->ADDR[7].RSLT);
115     print_lcd(0, adc_value);
116     ADC->ADCR.ADST = 1; // 1 = Conversion start
117
118     TIMER0->TISR.TIF = 1; // Write 1 to clear the interrupt flag
119 }
120
121 //-----MAIN
122 int32_t main (void) {
123     // Enable 12Mhz and set HCLK->12Mhz
124     UNLOCKREG();
125     SYSClk->PWRCON.XTL12M_EN = 1;
126     SYSClk->CLKSEL0.HCLK_S = 0;
127     LOCKREG();
128
129     InitPWM();
130     InitADC();
131     InitTIMER0();
132
133     Initial_pannel(); // call initial pannel function
134     clr_all_pannal();
135
136     while(1) {
137         __NOP();
138     }
139 }
140

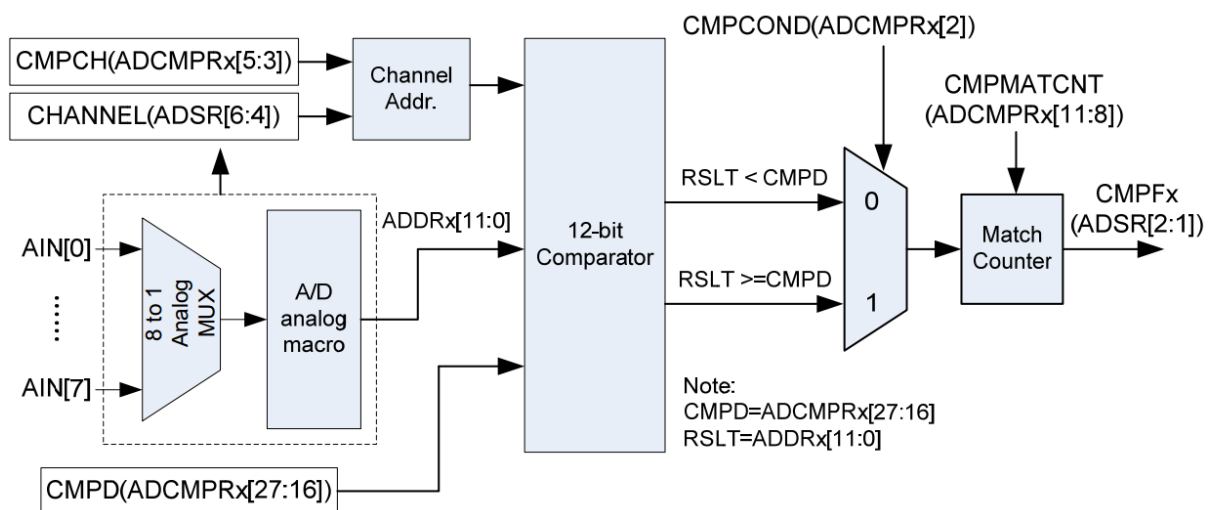
```

Questions (ADC)

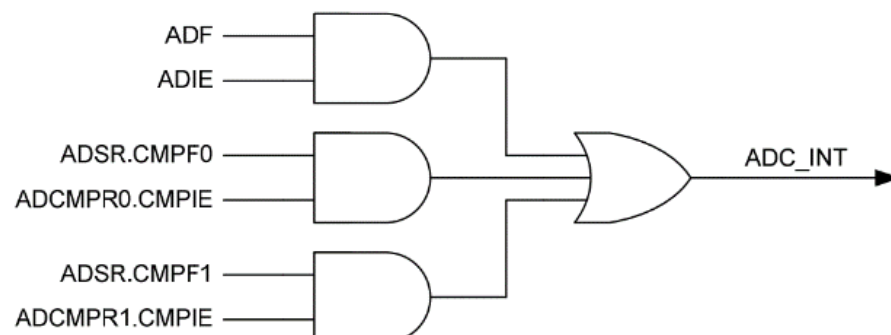
1. How to disable digital input path, if we want to use ADC0-3?
2. How to assign the value to ADC->ADCHER.CHEN, if we want to use ADC0-3?

ADC: Conversion Result Monitor by Compare Function

ADC controller provide two sets of compare register ADCMPR0 and ADCMPR1, to monitor maximum two specified channels conversion result from A/D conversion controller, refer to Figure below. Software can select which channel to be monitored by set CMPCH(ADCMPRx[5:3]) and CMPCOND bit is used to check conversion result is less than specify value or greater than (equal to) value specified in CMPD[11:0]. When the conversion of the channel specified by CMPCH is completed, the comparing action will be triggered one time automatically. When the compare result meets the setting, compare match counter will increase 1, otherwise, the compare match counter will be clear to 0. When counter value reach the setting of (CMPMATCNT+1) then CMPF bit will be set to 1, if CMPIE bit is set then an ADC_INT interrupt request is generated. Software can use it to monitor the external analog input pin voltage transition in scan mode without imposing a load on software. Detail logics diagram is shown as below:



ADC: Interrupt Sources



Procedure 2: ADC connect GPA7 with compare

1. Replace the content of the 'Smpl_Start_Kit.c' with the '[ADCcompare](#)' lab file.
2. Compile the project, and run the program.
3. Study the program and do the assignment in the class.

```

20 //-----ADC7compare
21 void InitADC(void) {
22     /* Step 1. GPIO initial */
23     GPIOA->OFRD |= 0x00800000;    // Disable digital input path
24                                   // (when input is analog signal)
25     SYS->GPAMFP.ADC7_SS21_AD6 = 1; // Set ADC function
26
27     /* Step 2. Enable and Select ADC clock source, and then enable ADC module */
28     SYSCLK->CLKSEL1.ADC_S = 3;    // Select 22Mhz for ADC
29     // 0:12MHz, 1:PLL FOUT, 2:HCLK, 3:22.1184 MHz
30     SYSCLK->CLKDIV.ADC_N = 1;    // ADC clock source = 22Mhz/2 =11Mhz;
31     // The ADC clock frequency = (ADC clock source frequency)/(ADC_N+1) ;8-bits
32     SYSCLK->APBCLK.ADC_EN = 1;    // Enable clock source
33     ADC->ADCR.ADEN = 1;          // Enable ADC module
34
35     /* Step 3. Select Operation mode */
36     ADC->ADCR.DIFFEN = 0;         // Single-end analog input mode
37     ADC->ADCR.ADMOD = 0;         // A/D Converter Operation Mode
38     // 0:Single conversion, 2:Single-cycle scan, 3:Continuous scan
39
40     /* Step 4. Select ADC channel */
41     ADC->ADCHER.CHEN = 0x0080;   // 8-bits -> ch7
42
43     /* Step 5. Enable ADC interrupt */
44     //ADC->ADSR.ADF = 1;         // clear the A/D interrupt flags
45     //ADC->ADCR.ADIE = 1;
46     //NVIC_EnableIRQ(ADC_IRQn);
47
48     /* Step x. compare setup */
49     ADC->ADCMPR[0].CMPD = 0x7FF; // Comparison Data
50     ADC->ADCMPR[0].CMPCH = 7;    // Compare Channel Selection
51     ADC->ADCMPR[0].CMPCOND = 1; // Compare Condition
52     // 1: greater or equal, 0: less than
53     ADC->ADCMPR[0].CMPIE = 1;    // Compare Interrupt Enable
54     ADC->ADCMPR[0].CMPEN = 1;    // Compare Enable
55     NVIC_EnableIRQ(ADC_IRQn);
56
57     /* Step 6. A/D Conversion Start */
58     ADC->ADCR.ADST=1;
59     // ADST will be cleared to 0 by hardware automatically
60     // at the ends of single mode and single cycle scan mode.
61 }
62
63 void ADC_IRQHandler(void) {
64     print_lcd(3, "ADC interrupt");
65
66     ADC->ADCMPR[0].CMPIE = 0;    // Disable compare interrupt
67 }

```



```

68 //-----PWM0
69 void InitPWM(void) {
70     /* Step 1. GPIO initial */
71     SYS->GPAMFP.PWM0_AD13 = 1;
72
73     /* Step 2. Enable and Select PWM clock source*/
74     SYSCLK->APBCLK.PWM01_EN = 1; // Enable PWM clock
75     SYSCLK->CLKSEL1.PWM01_S = 3; // Select 22.1184Mhz for PWM clock source
76     // 0:12MHz, 1:32.768 kHz, 2:HCLK, 3:22.1184 MHz
77     PWMA->PPR.CP01 = 1; // Prescaler 0~255, Setting 0 to stop output clock
78     PWMA->CSR.CSR0 = 0; // clock divider -> 0:/2, 1:/4, 2:/8, 3:/16, 4:/1
79     // PWM frequency = PWMxy_CLK/[(prescale+1)*(clock divider)*(CNR+1)]
80     // Ex:= 22.1184M/[(1+1)*(2)*(2^16)] = 84.375 Hz -> T = 11.85 ms.
81
82     /* Step 3. Select PWM Operation mode */
83     PWMA->PCR.CHOMOD = 1; // 0:One-shot mode, 1:Auto-load mode
84     //CNR and CMR will be auto-cleared after setting CHOMOD form 0 to 1.
85     PWMA->CNR0 = 0xFFFF; // CMR >= CNR: PWM output is always high
86     PWMA->CMR0 = 0xFFFF;
87     // CMR = 0: PWM low width = (CNR) unit; PWM high width = 1 unit
88
89     PWMA->PCR.CHOINV = 0; // Inverter -> 0:off, 1:on
90     PWMA->PCR.CHOEN = 1; // PWM function -> 0:Disable, 1:Enable
91     PWMA->POE.PWM0 = 1; // Output to pin -> 0:Disable, 1:Enable
92 }
93 //-----Timer0
94 void InitTIMER0(void) {
95     /* Step 1. Enable and Select Timer clock source */
96     SYSCLK->CLKSEL1.TMR0_S = 0; // Select 12Mhz for Timer0 clock source
97     // 0 = 12 MHz, 1 = 32 kHz, 2 = HCLK, 7 = 22.1184 MHz
98     SYSCLK->APBCLK.TMR0_EN = 1; // Enable Timer0 clock source
99
100    /* Step 2. Select Operation mode */
101    TIMER0->TCSR.MODE = 1; // 1 -> Select periodic mode
102    // 0 = One shot, 1 = Periodic, 2 = Toggle, 3 = continuous counting mode
103
104    /* Step 3. Select Time out period
105    = (Period of timer clock input) * (8-bit Prescale + 1) * (24-bit TCMP)*/
106    TIMER0->TCSR.PRESCALE = 11; // Set Prescale [0~255]
107    TIMER0->TCMPR = 300000; // Set TCMPR [0~16777215]
108    // (1/12000000)*(11+1)*(300000)= 0.3 sec
109
110    /* Step 4. Enable interrupt */
111    TIMER0->TCSR.IE = 1;
112    TIMER0->TISR.TIF = 1; // Write 1 to clear the interrupt flag
113    NVIC_EnableIRQ(TMR0_IRQn); // Enable Timer0 Interrupt
114
115    /* Step 5. Enable Timer module */
116    TIMER0->TCSR.CRST = 1; // Reset up counter
117    TIMER0->TCSR.CEN = 1; // Enable Timer0
118 }
119
120 void TMR0_IRQHandler(void) { // Timer0 interrupt subroutine
121     char adc_value[15] = "ADC Value:";
122     while(ADC->ADSR.ADF == 0); // A/D Conversion End Flag
123     // A status flag that indicates the end of A/D conversion.
124
125     ADC->ADSR.ADF = 1; // This flag can be cleared by writing 1 to self
126     PWMA->CMR0 = ADC->ADDR[7].RSLT << 4;
127     Show_Word(0,11,' ');
128     Show_Word(0,12,' ');
129     Show_Word(0,13,' ');
130     sprintf(adc_value+10, "%d", ADC->ADDR[7].RSLT);
131     print_lcd(0, adc_value);
132     ADC->ADCR.ADST = 1; // 1 = Conversion start
133
134     TIMER0->TISR.TIF = 1; // Write 1 to clear the interrupt flag
135 }
136
137 //-----MAIN
138 int32_t main (void) {
139     // Enable 12Mhz and set HCLK->12Mhz
140     UNLOCKREG();
141     SYSCLK->PWRCON.XTL12M_EN = 1;
142     SYSCLK->CLKSEL0.HCLK_S = 0;
143     LOCKREG();
144
145     InitPWM();
146     InitADC();
147     InitTIMER0();
148
149     Initial_panel(); // call initial pannel function
150     clr_all_pannel();
151
152     while(1) {
153         __NOP();
154     }
155 }

```


Assignment(s)

Summarize what you suppose to learn in this class.