1. Program to use GPIO with LED / Buzzer with interrupt int1/int0.

```
#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvUART.h"
#include "Driver\DrvSYS.h"

void Init_LED() // Initialize GPIO pins
{
   DrvGPIO_Open(E_GPC, 15, E_IO_OUTPUT); // GPC12 pin set to output mode
   DrvGPIO_SetBit(E_GPC, 15);          // Goutput Hi to turn off LED
}

void EINT0Led_CALLBACK(void)
{
   DrvGPIO_ClrBit(E_GPC, 15); //turns on LED
   DrvSYS_Delay(300000);
   DrvGPIO_SetBit(E_GPC, 15); //turns off LED
   DrvSYS_Delay(300000);
}

void EINT1Callback(void)
{
   DrvGPIO_ClrBit(E_GPB,11);  //turns on Buzzer
```

```
   DrvSYS_Delay(100000);
   DrvGPIO_SetBit(E_GPB,11);  //turns off Buzzer
   DrvSYS_Delay(100000);
}

int main (void)
{
          UNLOCKREG();
  DrvSYS_Open(48000000);
  LOCKREG();

  Init_LED();
  DrvGPIO_Open(E_GPB, 14, E_IO_INPUT); //for LED
  DrvGPIO_EnableEINT0(E_IO_RISING,  E_MODE_EDGE,  EINT0Led_CALLBACK);        //GPIO  port E_GPB, pin 14

  DrvGPIO_Open(E_GPB, 11, E_IO_OUTPUT); //for buzzer
  DrvGPIO_Open(E_GPB, 15, E_IO_INPUT);  // configure external interrupt pin GPB15

  DrvGPIO_EnableEINT1(E_IO_BOTH_EDGE, E_MODE_EDGE, EINT1Callback);

  while(1)
   {
   }
```

```
}
```

2.Program to use GPIO as input from A port and display the port bit number.

```
#include<stdio.h>
#include "Driver\DrvGPIO.h"
#include "Driver\DrvUART.h"
#include "Driver\DrvSYS.h"
#include "LCD_Driver.h"

int32_t main()
{
        char TEXT[16];
        int32_t a;

        UNLOCKREG();  //present in setting up clock for pwm (last page of file)
        SYSCLK->PWRCON.XTL12M_EN = 1;
        DrvSYS_Delay(5000);  //this is not present in the file
        SYSCLK->CLKSEL0.HCLK_S = 0;
        LOCKREG();

        DrvGPIO_SetPortBits(E_GPA,15);
        a=DrvGPIO_GetPortBits(E_GPA);

        Initial_panel();
```

```
    clr_all_panel();

    sprintf(TEXT,"port :: %x",a);//for decimal --> %d
    print_lcd(0,TEXT);
}
```

3.Program interrupt with port A and identify the A port bit that was interrupted and increment the counter to count the number of interrupts.

```
#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvUART.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvSYS.h"
#include "LCD_Driver.h"

volatile uint32_t irqA_counter = 0;

void GPIOAB_INT_CallBack(uint32_t GPA_IntStatus, uint32_t GPB_IntStatus)
{
    int32_t a;
    char text[16];
    DrvGPIO_SetPortBits(E_GPA,0);
```

```
    a=DrvGPIO_GetPortBits(E_GPA);
     if ((GPA_IntStatus>>0) & 0x01) irqA_counter++;
     sprintf(text,"port number %d",a);
     print_lcd(3,"GPA interrupt !! ");
     print_lcd(2,text);
}



int32_t main()
{
    char TEXT[16];

    UNLOCKREG();
    SYSCLK->PWRCON.XTL12M_EN=1;
    DrvSYS_Delay(5000);                                  // Waiting for 12M Xtal stalble
    SYSCLK->CLKSEL0.HCLK_S=0;
    LOCKREG();

  // setup GPA15 & GPD15 to get interrupt input

    DrvGPIO_Open(E_GPA,0,E_IO_INPUT);
    DrvGPIO_EnableInt(E_GPA, 0, E_IO_RISING, E_MODE_EDGE);
  DrvGPIO_SetDebounceTime(5, 1);
```

```
    DrvGPIO_EnableDebounce(E_GPA, 0);

 DrvGPIO_SetIntCallback(GPIOAB_INT_CallBack,NULL);

 Initial_panel();
    clr_all_panel();

    print_lcd(0,"Smpl_GPIO_Intr");

    while(1)
    {
            sprintf(TEXT,"IRQ_A: %d",irqA_counter);
            print_lcd(1, TEXT);

    }
}4.Program for using ADC  channel 6  and display analog value on the LCD.

    #include <stdio.h>

    #include "NUC1xx.h"
    #include "Driver\DrvSYS.h"
    #include "Seven_Segment.h"
    #include "DrvADC.h"
    #include "LCD_Driver.h"
```

```
int32_t main (void)
{  uint16_t value;
   char TEXT[16];

        UNLOCKREG();
        SYSCLK->PWRCON.XTL12M_EN = 1;   //Enable 12Mhz and set HCLK->12Mhz
        SYSCLK->CLKSEL0.HCLK_S = 0;
        LOCKREG();
        Initial_panel();  // initialize LCD pannel
  clr_all_panel();  // clear LCD panel
  print_lcd(0,"variable reistor");

        DrvADC_Open(ADC_SINGLE_END,ADC_SINGLE_OP , 0x40,INTERNAL_HCLK , 1);
        while(1)
  {
   DrvADC_StartConvert();   // start A/D conversion
   while(DrvADC_IsConversionDone()==FALSE);
   value = ADC->ADDR[6].RSLT & 0xFFF;

   sprintf(TEXT,"Value: %d",value); // convert ADC0 value into text
   print_lcd(1, TEXT); // output TEXT to LCD
```

```
    }
  }
```

5. Program for using ADC channel 0 and display value on the 7 segment.

```
#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvSYS.h"
#include "Seven_Segment.h"
void InitADC(void)
{
    /* Step 1. GPIO initial */
    //Should be 0x00010000
    GPIOA->OFFD|=0x00010000; //Disable digital input path
    SYS->GPAMFP.ADC7_SS21_AD6=1; //Set ADC function

    /* Step 2. Enable and Select ADC clock source, and then enable ADC module */
    SYSCLK->CLKSEL1.ADC_S = 2; //Select 22Mhz for ADC
    SYSCLK->CLKDIV.ADC_N = 1; //ADC clock source = 22Mhz/2 =11Mhz;
    SYSCLK->APBCLK.ADC_EN = 1; //Enable clock source
    ADC->ADCR.ADEN = 1; //Enable ADC module

    /* Step 3. Select Operation mode */
    ADC->ADCR.DIFFEN = 0; //single end input
```

```
	ADC->ADCR.ADMD = 0; //single mode
	//Should be 0x01(In Q4 0x40)

	/* Step 4. Select ADC channel 0*/
	ADC->ADCHER.CHEN = 0x01;

	/* Step 5. Enable ADC interrupt */
	ADC->ADSR.ADF =1; //clear the A/D interrupt flags for safe
	ADC->ADCR.ADIE = 1;
	// NVIC_EnableIRQ(ADC_IRQn);

	/* Step 6. Enable WDT module */
	ADC->ADCR.ADST=1;
}
void seg_display(int16_t value)
{
	int8_t digit;
	digit = value / 1000;
	close_seven_segment();
	show_seven_segment(3,digit);
	DrvSYS_Delay(5000);

	value = value - digit * 1000;
	digit = value / 100;
```

```
        close_seven_segment();
        show_seven_segment(2,digit);
        DrvSYS_Delay(5000);

        value = value - digit * 100;
        digit = value / 10;
        close_seven_segment();
        show_seven_segment(1,digit);
        DrvSYS_Delay(5000);

        value = value - digit * 10;
        digit = value;
        close_seven_segment();
        show_seven_segment(0,digit);
        DrvSYS_Delay(5000);
}
int32_t main (void)
{
        int32_t adc_value;
        UNLOCKREG();
        SYSCLK->PWRCON.XTL12M_EN = 1; //Enable 12Mhz and set HCLK->12Mhz
        SYSCLK->CLKSEL0.HCLK_S = 0;
        LOCKREG();
        InitADC();
```

```
        while(1)
        {
                while(ADC->ADSR.ADF==0); // ADC Flag, wait till 1 (A/DC conversion done)
                ADC->ADSR.ADF=1; // write 1 to ADF is to clear the flag

                adc_value=ADC->ADDR[0].RSLT; // input 12-bit ADC value
                seg_display(adc_value); // display value to 7-segment display
                ADC->ADCR.ADST=1;       //from step 6
        }
}
```

6.Program pwm1 and adc channel 6 and change the illumination of led (use ADC and PWM).

```
#include <stdio.h>
#include "NUC1xx.h"
#include "LCD_Driver.h"
#define BAUDRATE 9600

void InitADC(void)
{
/* Step 1. GPIO initial */
GPIOA->OFFD|=0x00400000; //Disable digital input path
SYS->GPAMFP.ADC7_SS21_AD6=1; //Set ADC function
/* Step 2. Enable and Select ADC clock source, and then enable ADC module */
SYSCLK->CLKSEL1.ADC_S = 2; //Select 22Mhz for ADC
SYSCLK->CLKDIV.ADC_N = 1; //ADC clock source = 22Mhz/2 =11Mhz;
SYSCLK->APBCLK.ADC_EN = 1; //Enable clock source
ADC->ADCR.ADEN = 1; //Enable ADC module

/* Step 3. Select Operation mode */
ADC->ADCR.DIFFEN = 0;     //single end input
ADC->ADCR.ADMD   = 0;     //single mode
```

```
/* Step 4. Select ADC channel */
ADC->ADCHER.CHEN = 0x40;
/* Step 5. Enable ADC interrupt */
ADC->ADSR.ADF =1;    //clear the A/D interrupt flags for safe
ADC->ADCR.ADIE = 1;
// NVIC_EnableIRQ(ADC_IRQn);
/* Step 6. Enable WDT module */
ADC->ADCR.ADST=1;
}
//-----------------------------------------
void InitPWM(void)
{
  /* Step 1. GPIO initial */
SYS->GPAMFP.PWM0_AD13=1;
/* Step 2. Enable and Select PWM clock source*/
SYSCLK->APBCLK.PWM01_EN = 1;//Enable PWM clock
SYSCLK->CLKSEL1.PWM01_S = 3;//Select 22.1184Mhz for PWM clock source

PWMA->PPR.CP01=1; //Prescaler 0~255, Setting 0 to stop output clock
PWMA->CSR.CSR0=0; // PWM clock = clock source/(Prescaler + 1)/divider

/* Step 3. Select PWM Operation mode */
//PWM0
PWMA->PCR.CH0MOD=1; //0:One-shot mode, 1:Auto-load mode
//CNR and CMR will be auto-cleared after setting CH0MOD form 0 to 1.
PWMA->CNR0=0xFFFF;
PWMA->CMR0=0xFFFF;

PWMA->PCR.CH0INV=0; //Inverter->0:off, 1:on
PWMA->PCR.CH0EN=1; //PWM function->0:Disable, 1:Enable
  PWMA->POE.PWM0=1; //Output to pin->0:Diasble, 1:Enable
}

void Delay(int count)
{
while(count--)
{
// __NOP;
}
}

/*---------------------------------------------------------------------------
  MAIN function
  ---------------------------------------------------------------------------*/
```

```
int32_t main (void)
{
//Enable 12Mhz and set HCLK->12Mhz
char adc_value[15]="ADC Value:";
UNLOCKREG();
SYSCLK->PWRCON.XTL12M_EN = 1;
SYSCLK->CLKSEL0.HCLK_S = 0;
LOCKREG();

InitPWM();
InitADC();

Initial_panel();  //call initial pannel function
clr_all_panel();
/* Synch field transmission & Request Identifier Field transmission*/

while(1)
{
while(ADC->ADSR.ADF==0);
ADC->ADSR.ADF=1;
PWMA->CMR0=ADC->ADDR[6].RSLT<<4;
Show_Word(0,11,' ');
Show_Word(0,12,' ');
Show_Word(0,13,' ');
sprintf(adc_value+10,"%d",ADC->ADDR[6].RSLT);
print_lcd(0, adc_value);
Delay(20000);
ADC->ADCR.ADST=1;
}
}
```

7.Using pwm0 change the illumination of external led connected to port A12.

```
#include <stdio.h>
#include "NUC1xx.h"
#include "LCD_Driver.h"
#define BAUDRATE 9600

void InitADC(void)
{
/* Step 1. GPIO initial */
GPIOA->OFFD|=0x00800000; //Disable digital input path
SYS->GPAMFP.ADC7_SS21_AD6=1; //Set ADC function
```

```
/* Step 2. Enable and Select ADC clock source, and then enable ADC module */
SYSCLK->CLKSEL1.ADC_S = 2; //Select 22Mhz for ADC
SYSCLK->CLKDIV.ADC_N = 1; //ADC clock source = 22Mhz/2 =11Mhz;
SYSCLK->APBCLK.ADC_EN = 1; //Enable clock source
ADC->ADCR.ADEN = 1; //Enable ADC module

/* Step 3. Select Operation mode */
ADC->ADCR.DIFFEN = 0;     //single end input
ADC->ADCR.ADMD   = 0;     //single mode
/* Step 4. Select ADC channel */
ADC->ADCHER.CHEN = 0x80;
/* Step 5. Enable ADC interrupt */
ADC->ADSR.ADF =1;     //clear the A/D interrupt flags for safe
ADC->ADCR.ADIE = 1;
// NVIC_EnableIRQ(ADC_IRQn);
/* Step 6. Enable WDT module */
ADC->ADCR.ADST=1;
}
//-----------------------------------------
void InitPWM(void)
{
  /* Step 1. GPIO initial */
SYS->GPAMFP.PWM0_AD13=1;
/* Step 2. Enable and Select PWM clock source*/
SYSCLK->APBCLK.PWM01_EN = 1;//Enable PWM clock
SYSCLK->CLKSEL1.PWM01_S = 3;//Select 22.1184Mhz for PWM clock source

PWMA->PPR.CP01=1; //Prescaler 0~255, Setting 0 to stop output clock
PWMA->CSR.CSR0=0; // PWM clock = clock source/(Prescaler + 1)/divider

/* Step 3. Select PWM Operation mode */
//PWM0
PWMA->PCR.CH0MOD=1; //0:One-shot mode, 1:Auto-load mode
//CNR and CMR will be auto-cleared after setting CH0MOD form 0 to 1.
PWMA->CNR0=0xFFFF;
PWMA->CMR0=0xFFFF;

PWMA->PCR.CH0INV=0; //Inverter->0:off, 1:on
PWMA->PCR.CH0EN=1; //PWM function->0:Disable, 1:Enable
  PWMA->POE.PWM0=1; //Output to pin->0:Diasble, 1:Enable
}

void Delay(int count)
{
```

```
while(count--)
{
// __NOP;
}
}

/*---------------------------------------------------------------------------
  MAIN function
 ---------------------------------------------------------------------------*/
int32_t main (void)
{
//Enable 12Mhz and set HCLK->12Mhz
char adc_value[15]="ADC Value:";
UNLOCKREG();
SYSCLK->PWRCON.XTL12M_EN = 1;
SYSCLK->CLKSEL0.HCLK_S = 0;
LOCKREG();

InitPWM();
InitADC();

Initial_panel();  //call initial pannel function
clr_all_panel();
/* Synch field transmission & Request Identifier Field transmission*/

while(1)
{
while(ADC->ADSR.ADF==0);
ADC->ADSR.ADF=1;
PWMA->CMR0=ADC->ADDR[7].RSLT<<4;
Show_Word(0,11,' ');
Show_Word(0,12,' ');
Show_Word(0,13,' ');
sprintf(adc_value+10,"%d",ADC->ADDR[7].RSLT);
print_lcd(0, adc_value);
Delay(20000);
ADC->ADCR.ADST=1;
}
}
```

8) TO SWITCH ON/OFF BULB USING RELAY

```
//
// Smpl_GPIO_EINT1 : External Interrupt pin to trigger interrupt on GPB15, then Buzz

#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvUART.h"
#include "Driver\DrvSYS.h"

// External Interrupt Handler (INT button to trigger GPB15)
void EINT1Callback(void)
{
  DrvGPIO_ClrBit(E_GPA,0); // GPB11 = 0 to turn on Buzzer
        DrvSYS_Delay(10);         // Delay
        //DrvGPIO_SetBit(E_GPA,0); // GPB11 = 1 to turn off Buzzer
        DrvSYS_Delay(10000);              // Delay
}

int main (void)
{
        UNLOCKREG();
        DrvSYS_SetOscCtrl(E_SYS_XTL12M, 1); // external 12MHz Crystal
        //DrvSYS_Delay(5000);              // delay for stable clock
        DrvSYS_SelectHCLKSource(0);       // clock source = 12MHz Crystal
        LOCKREG();

        DrvGPIO_Open(E_GPA, 0, E_IO_OUTPUT); // initial GPIO pin GPB11 for
controlling Buzzer

//0 External Interrupt
  DrvGPIO_Open(E_GPB, 15, E_IO_INPUT);                    // configure external interrupt
pin GPB15
  DrvGPIO_EnableEINT1(E_IO_BOTH_EDGE, E_MODE_EDGE, EINT1Callback); // configure
external interrupt

  while(1)
        {
```

```
        }
    }
```

9. ldr program

```
//
// Smpl_ADC_VR1 : use ADC7 to read Variable Resistor (on-board)
//
#include <stdio.h>


#include "NUC1xx.h"
#include "DrvSYS.h"
#include "NUC1xx-LB_002\LCD_Driver.h"

void InitADC(void)
{
    /* Step 1. GPIO initial */
    GPIOA->OFFD|=0x00400000;        //Disable digital input path
    SYS->GPAMFP.ADC7_SS21_AD6=1;        //Set ADC function

    /* Step 2. Enable and Select ADC clock source, and then enable ADC module */
    SYSCLK->CLKSEL1.ADC_S = 2; //Select 22Mhz for ADC
    SYSCLK->CLKDIV.ADC_N = 1;  //ADC clock source = 22Mhz/2 =11Mhz;
    SYSCLK->APBCLK.ADC_EN = 1;      //Enable clock source
    ADC->ADCR.ADEN = 1;                 //Enable ADC module
```

```
        /* Step 3. Select Operation mode */
        ADC->ADCR.DIFFEN = 0;         //single end input
        ADC->ADCR.ADMD   = 0;         //single mode

        /* Step 4. Select ADC channel */
        ADC->ADCHER.CHEN = 0x40;

        /* Step 5. Enable ADC interrupt */
        ADC->ADSR.ADF =1;                       //clear the A/D interrupt flags for safe
        ADC->ADCR.ADIE = 1;
//      NVIC_EnableIRQ(ADC_IRQn);

        /* Step 6. Enable WDT module */
        ADC->ADCR.ADST=1;
}

/*---------------------------------------------------------------------------
  MAIN function
 ---------------------------------------------------------------------------*/
int32_t main (void)
{
        char TEXT1[16]="ADC Value:     ";
        UNLOCKREG();
        //SYSCLK->PWRCON.XTL12M_EN = 1; // enable external clock (12MHz)
```

```
    //SYSCLK->CLKSEL0.HCLK_S = 0;          // select external clock (12MHz)
    LOCKREG();

    InitADC();                    // initialize ADC

    Initial_panel();  // initialize LCD pannel
    clr_all_panel();  // clear LCD panel
    print_lcd(0, "Smpl_ADC_VR1");

    while(1)
    {
        while(ADC->ADSR.ADF==0); // wait till conversion flag = 1, conversion is done
        ADC->ADSR.ADF=1;                  // write 1 to clear the flag
        sprintf(TEXT1+10,"%4d",ADC->ADDR[6].RSLT); // convert ADC7 value into text
        print_lcd(1, TEXT1);      // output TEXT to LCD
        DrvSYS_Delay(20000);              // delay
        ADC->ADCR.ADST=1;                 // restart ADC sample
    }
}
```

10)stepper motor

```
//
// Sampl_GPIO_StepMotor
// 5V Step Motor 28BYJ-48, driver IC = ULN2003A
```

```
//
// Driver board connections:
// ULN2003A    NUC140
// INA      to GPA3
// INB      to GPA2
// INC      to GPA1
// IND      to GPA0
//
#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvSYS.h"

// Definitions for Step Motor turning degree
#define d360 512
#define d180 512/2
#define d90  512/4
#define d45  512/8
#define d2 51

unsigned char CW[8] ={0x09,0x01,0x03,0x02,0x06,0x04,0x0c,0x08}; //Clockwise Sequence
unsigned char CCW[8]={0x08,0x0c,0x04,0x06,0x02,0x03,0x01,0x09};  //Counter-Clockwise Sequence

void CW_MOTOR(uint16_t deg)
{
 int i=0,j=0;

for(j=0;j<(deg);j++)
{
   for(i=0;i<8;i++)
            {
            GPIOA->DOUT=CW[i];
            DrvSYS_Delay(20000);//delay 2000us = 2ms
            }
 }
}

void CCW_MOTOR(uint16_t deg)
{
 int i=0,j=0;

for(j=0;j<(deg);j++)
{
```

```
   for(i=0;i<8;i++)
            {
            GPIOA->DOUT=CCW[i];
            DrvSYS_Delay(20000);//delay 2000us = 2ms
            }
 }
}

int main (void)
{
            CW_MOTOR(d2); // Clockwise        for 360 degree
            //CCW_MOTOR(d2/2);// Counter-Clockwise for 180 degree
}
```