1)

```
//Connect GPB15 to Ground(GND)
//
// 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_GPB,11); // GPB11 = 0 to turn on Buzzer
        DrvSYS_Delay(100000);          // Delay
        DrvGPIO_SetBit(E_GPB,11); // GPB11 = 1 to turn off Buzzer
        DrvSYS_Delay(100000);          // Delay

        DrvGPIO_ClrBit(E_GPC, 15); // output Low to turn on LED
  DrvSYS_Delay(300000);    // delay
  DrvGPIO_SetBit(E_GPC, 15); // output Hi to turn off LED
  DrvSYS_Delay(300000);
}

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_GPC, 15, E_IO_OUTPUT); // GPC12 pin set to output mode
  DrvGPIO_SetBit(E_GPC, 15);

        DrvGPIO_Open(E_GPB, 11, E_IO_OUTPUT); // initial GPIO pin GPB11 for controlling Buzzer

// 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)
        {
  }
}
```

2)

```
// Dont connect any interrupts just load the program
//No connections
#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvUART.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvSYS.h"
#include "LCD_Driver.h"
int32_t main()
{
    char TEXT[16];
    int32_t a;

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

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


  Initial_panel();
  clr_all_panel();

//to print decimal: sprintf(TEXT,”port is %d”,a);
  sprintf(TEXT,"port :: %x",a);
    print_lcd(0, TEXT);

}
```

3)
Program interrupt with port A and identify Aport bit that was
interrupted and increment the counter to count no of interrupts

```
//GPA 15 to GND then VCC repeat
#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;

//USER DEFINED FUNCTION
void GPIOAB_INT_CallBack(uint32_t GPA_IntStatus, uint32_t GPB_IntStatus)
{
            int32_t a;
            char TEXT1[16];

//Highlighted one not needed
  if ((GPA_IntStatus>>15) & 0x01) irqA_counter++;
             print_lcd(3,"GPA interrupt !!");
            a=DrvGPIO_GetPortBits(E_GPA);
  sprintf(TEXT1,"port :: %x",a);
  print_lcd(0, TEXT1);

}
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 to get interrupt input
            DrvGPIO_Open(E_GPA,15,E_IO_INPUT);
  DrvGPIO_EnableInt(E_GPA, 15, E_IO_RISING, E_MODE_EDGE);

  DrvGPIO_SetDebounceTime(5, 1);
  DrvGPIO_EnableDebounce(E_GPA, 15);

  DrvGPIO_SetIntCallback(GPIOAB_INT_CallBack,NULL);
  Initial_panel();
                    clr_all_panel();

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

                    }
}
```

5)

Explained changes for Q4

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

```
//CONNECTIONS
//Potentiometer                Board
//GND                          GND
//VCC                          VCC(3.3)
//SIG                          GPA0          //For Q4 GPA6

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

void InitADC(void)
{
          /* Step 1. GPIO initial */
//Should be 0x00010000 (In Q4 0x00040000)
          GPIOA->OFFDI=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
         //Should be 0 (In Q4 6)
        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;           // activate next ADC sample
                              // 1 : conversion start
                                    // 0 : conversion stopped, ADC enter idle state
    }
}
```

10

//STEPPER MOTOR

```
//
// Sampl_GPIO_StepMotor
// 5V Step Motor 28BYJ-48, driver IC = ULN2003A
//
// Driver board connections:
// ULN2003A    NUC140

//Connections
// IN1     to GPA3
// IN2     to GPA2
// IN3     to GPA1
// IN4     to GPA0
//VCC to  VCC(not 3.3 or 5)
//GND to GND

#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(2000);//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(2000);//delay 2000us = 2ms
            }
 }
}
```

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