ADC

```
#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 resistor");

  DrvADC_Open(ADC_SINGLE_END,ADC_SINGLE_OP , 0x80,INTERNAL_HCLK , 1);
  while(1)
   {
    DrvADC_StartConvert();   // start A/D conversion
    while(DrvADC_IsConversionDone()==FALSE);
    value = ADC->ADDR[7].RSLT & 0xFFF;

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

    }
}
```

Getport

```
#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvSYS.h"
#include "Driver\DrvGPIO.h"
#include "NUC1xx-LB_002\LCD_Driver.h"
int main(void)
{
  int32_t number;
  char TEXT0[16] = "SmplKeypad";
  char TEXT1[16];
  UNLOCKREG();          // unlock register for programming
  DrvSYS_Open(48000000); // set System Clock to run at 48MHz
  LOCKREG();            // lock register from programming
                        // Initialize LEDs (four on-board LEDs below LCD panel)
  Initial_panel();
  clr_all_panel();
  print_lcd(0, TEXT0); // print title
  while (1)           // forever loop to keep flashing four LEDs one at a time
  {
    number = DrvGPIO_GetPortBits(E_GPA);
    sprintf(TEXT1, "%x", number); // print scankey input to string
    print_lcd(1, TEXT1);
    if (number == 0xfffe)
      print_lcd(2, "A0");
    else if (number == 0xfffd)
      print_lcd(2, "A1");
    else if (number == 0xfffb)
      print_lcd(2, "A2");
    else if (number == 0xfff7)
      print_lcd(2, "A3");
```

```
   else if (number == 0xffef)
    print_lcd(2, "A4");
   else if (number == 0xffdf)
    print_lcd(2, "A5");
   else if (number == 0xffbf)
    print_lcd(2, "A6");
   else if (number == 0xff7f)
    print_lcd(2, "A7");
   else if (number == 0xfeff)
    print_lcd(2, "A8");
  }
}
```

7 Seg

```
#include <stdio.h>
#include "NUC1xx.h"
#include "DrvSYS.h"
#include "Seven_Segment.h"

// display an integer on four 7-segment LEDs
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 i =0;

  UNLOCKREG();
  DrvSYS_Open(48000000);
  LOCKREG();

  while(i<10000)
  {
    seg_display(i);    // display i on 7-segment display
    DrvSYS_Delay(10000); // delay for keeping display
    i++;         // increment i
  }
}
```

Onboard interrupt

```
// Smpl_GPIO_EINT1 : External Interrupt pin to trigger interrupt //on GPB15, then Buzz  INT1(GPB.15) pin
INT0(GPB.14) pin

#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvGPIO.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(10);     // Delay
  DrvGPIO_SetBit(E_GPB,11); // 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_GPB, 11, 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)
  {
  }
}
```

PI

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
GPIO.output(18, GPIO.HIGH)

time.sleep(3)

GPIO.output(18, GPIO.LOW)
GPIO.cleanup()
```