1.write a sample program to recognise port pins using getport.c
2.write a program to use on board interrupt INT1 and external INT0
3.write a program to change the intersity of led using PWM PWM_LED.c
4.write a sample program using setport.c using Interrupt
5.write a program to use adc with functions.c choosing channel 6
6.write a program to use Smpl_LCD_Text.c using Interrupt

7.Smpl_GPIO_Buzzer.c using Interrupt
8.smpl_GPIO_Interrupt.c using port pins of Port A and Port E
9.Smpl_GPIO_LED1.c using Interrupt InTO or INT1
10.Smpl_GPIO_RGBled.c using Interrupt
11. Using SSH blink led using rasberry Pi from remote system

Right click on target, click on options for target and go to c/c++. In include paths, include Include folder, Include/ NUC…, Include/Driver. Go to Debugger and press NUlink Debugger. Go to utilities and add the NUlink debugger.
Right click on the source group and add files to the group. Add the necessary files according to code. GO to CMSIS and core support and add the .c file. Do the same for device support(Need to go 2 folders in) as well. Also add the program file to the source group.

1

```
#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");
//else if(number==0xfeff)
        //print_lcd(2,"A7");
/*else if(number==0xff7f)
        print_lcd(2,"A7");
else if(number==0xff7f)
        print_lcd(2,"A7");
else if(number==0xff7f)
        print_lcd(2,"A7");
else if(number==0xff7f)
        print_lcd(2,"A7");
else if(number==0xff7f)
        print_lcd(2,"A7");
else if(number==0xff7f)
        print_lcd(2,"A7");
else if(number==0xff7f)
        print_lcd(2,"A7");
else if(number==0xff7f)
        print_lcd(2,"A7");*/


        }

        }
//Add ASCII.c as well. Put one female in GND and the other in GPA14.and 12 13
```

2

```
//
// 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(10000);            // Delay
        DrvGPIO_SetBit(E_GPB,11); // GPB11 = 1 to turn off Buzzer
        DrvSYS_Delay(10000);            // Delay
        DrvGPIO_ClrBit(E_GPB,11); // GPB11 = 0 to turn on Buzzer
        DrvSYS_Delay(10000);            // Delay
        DrvGPIO_SetBit(E_GPB,11); // GPB11 = 1 to turn off Buzzer
        DrvSYS_Delay(10000);            // Delay
        DrvGPIO_ClrBit(E_GPB,11); // GPB11 = 0 to turn on Buzzer
        DrvSYS_Delay(10000);            // 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)
        {

  }
}
//Connect to GND and then for 11 it will buzz continuously and for 15 it will break.
```

3

```
//
// Smpl_ADC_PWM : ADC7 to read VR1 resistance value, PWM0 output to control LED
(GPA12)
//
#include <stdio.h>


#include "NUC1xx.h"
#include "LCD_Driver.h"


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+4,"%d",ADC->ADDR[7].RSLT);
		print_lcd(0, adc_value);
		Delay(20000);
		ADC->ADCR.ADST=1;
	}
}
//Just turn the VR1 and the intensity changes.
```

4

```
//
// Smpl_GPIO_LED1 : GPC12--15  GPA 12_14 to control on-board LEDs
//                  low-active output to control Red LEDs
//
#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvGPIO.h"

#include "Driver\DrvSYS.h"
void EINT1Callback(void)
```

```
{
        DrvGPIO_SetPortBits(E_GPC,0xffff0fff); // output Low to turn on LED
        DrvSYS_Delay(300000);          // delay
DrvGPIO_SetPortBits(E_GPC,0xffffffff)        ; // output Hi to turn off LED
        DrvSYS_Delay(300000);          // delay

}

void Init_LED() // Initialize GPIO pins
{
 DrvGPIO_Open(E_GPC, 12, E_IO_OUTPUT); // GPC12 pin set to output mode
           DrvGPIO_Open(E_GPC, 13, E_IO_OUTPUT);       // Goutput Hi to turn off LED
        DrvGPIO_Open(E_GPC, 14, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPC, 15, E_IO_OUTPUT);
}

int main (void)
{
        UNLOCKREG();                                // unlock register for programming
  DrvSYS_Open(48000000);// set System Clock to run at 48MHz
                        // 12MHz crystal input, PLL output 48MHz
        LOCKREG();                                  // lock register from programming

  Init_LED();       // Initialize LEDs (four on-board LEDs below LCD panel)
        DrvGPIO_Open(E_GPB, 15, E_IO_INPUT);
DrvGPIO_EnableEINT1(E_IO_BOTH_EDGE, E_MODE_EDGE, EINT1Callback);
        while (1)                                   // forever loop to keep flashing four LEDs
one at a time
        {
        //DrvGPIO_SetPortBits(E_GPC,0xffff0fff); // output Low to turn on LED
        //DrvSYS_Delay(300000);        // delay
//DrvGPIO_SetPortBits(E_GPC,0xffffffff)       ; // output Hi to turn off LED
        //DrvSYS_Delay(300000);        // delay
        }

}
//Connect to ground and ports 12,13,14.
```

5

```
// Smpl_7seg_ADC7 : ADC7 to read and display on 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

    }
}
//Take the meter (a Long stick-like thing) and connect the three wires. Signal to Channel
6.

6
//
// Smpl_LCD_Text: display 4 lines of Text on LCD
//
#include <stdio.h>
```

```
#include "NUC1xx.h"
#include "Driver\DrvSYS.h"
#include "Driver\DrvGPIO.h"
#include "NUC1xx-LB_002\LCD_Driver.h"

void EINT1Callback(void)
{
        Initial_panel();
        clr_all_panel();
        print_lcd(0, "Smpl_LCD_Text   ");
        print_lcd(1, "Nu-LB-NUC140    ");
        print_lcd(2, "Test LCD Display");
        print_lcd(3, "Nuvoton NuMicro ");

}

int main(void)
{
        UNLOCKREG();
        DrvSYS_Open(48000000); // set to 48MHz
        LOCKREG();


        DrvGPIO_Open(E_GPB, 15, E_IO_INPUT);
DrvGPIO_EnableEINT1(E_IO_BOTH_EDGE, E_MODE_EDGE, EINT1Callback);

        while(1)
        {
        }

}
//Just reset and press RINT1
```

7

```
//
// Smpl_GPIO_Buzzer : GPB11 low-active output control Buzzer
// Note: Nu-LB-NUC140 R1 should be 0 ohm
//
#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvSYS.h"
#include "Driver\DrvGPIO.h"
```

```
#include "Driver\DrvADC.h"
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);

}

int main (void)
{
          UNLOCKREG();                              // unlock register for programming
  DrvSYS_Open(48000000);     // set System Clock to run at 48MHz
          LOCKREG();                                // lock register from programming


DrvGPIO_Open(E_GPB, 15, E_IO_INPUT);
DrvGPIO_EnableEINT1(E_IO_BOTH_EDGE, E_MODE_EDGE, EINT1Callback);

          while(1) {
    // Delay
          }
}
//Just press reset and the RINT1.
```

8

```
//
// smpl_GPIO_Interrupt
//
// GPA15 to input interrupt
// GPD15 to input interrupt

#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;
volatile uint32_t irqE_counter = 0;

void GPIOAB_INT_CallBack(uint32_t GPA_IntStatus, uint32_t GPB_IntStatus)
```

```
{
        if ((GPA_IntStatus>>15) & 0x01) irqA_counter++;
        print_lcd(3,"GPA interrupt !!");
}

void GPIOCDE_INT_CallBack(uint32_t GPC_IntStatus, uint32_t GPD_IntStatus,
uint32_t GPE_IntStatus)
{
        if ((GPE_IntStatus>>15) & 0x01) irqE_counter++;
        print_lcd(3,"GPC interrupt !!");
}

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,15,E_IO_INPUT);
        DrvGPIO_Open(E_GPE,15,E_IO_INPUT);
  DrvGPIO_EnableInt(E_GPA, 15, E_IO_RISING, E_MODE_EDGE);
  DrvGPIO_EnableInt(E_GPE, 15, E_IO_RISING, E_MODE_EDGE);
  DrvGPIO_SetDebounceTime(5, 1);
        DrvGPIO_EnableDebounce(E_GPA, 15);
        DrvGPIO_EnableDebounce(E_GPE, 15);

  DrvGPIO_SetIntCallback(GPIOAB_INT_CallBack, GPIOCDE_INT_CallBack);

  Initial_panel();
        clr_all_panel();

        print_lcd(0,"Smpl_GPIO_Intr");

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

```
                print_lcd(2, TEXT);
        }
}
```
//Connect one wire to A15 and one more to E15 and then connect the other part the each of the wires to ground and vcc. Counter will increase.

9

```
//
// Smpl_GPIO_LED1 : GPC12--15  GPA 12_14 to control on-board LEDs
//                 low-active output to control Red LEDs
//
#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvGPIO.h"

#include "Driver\DrvSYS.h"

void EINT1Callback(void)
{
        DrvGPIO_ClrBit(E_GPC, 13); // output Low to turn on LED
        DrvSYS_Delay(300000);          // delay
        DrvGPIO_SetBit(E_GPC, 13); // output Hi to turn off LED
        DrvSYS_Delay(300000);

}


void Init_LED() // Initialize GPIO pins
{
 DrvGPIO_Open(E_GPC, 12, E_IO_OUTPUT); // GPC12 pin set to output mode
        DrvGPIO_SetBit(E_GPC, 12);          // Goutput Hi to turn off LED
}

int main (void)
{
        UNLOCKREG();                                // unlock register for programming
  DrvSYS_Open(48000000);// set System Clock to run at 48MHz
                        // 12MHz crystal input, PLL output 48MHz
        LOCKREG();                                  // lock register from programming

  Init_LED();       // Initialize LEDs (four on-board LEDs below LCD panel)

DrvGPIO_Open(E_GPB, 15, E_IO_INPUT);
DrvGPIO_EnableEINT1(E_IO_BOTH_EDGE, E_MODE_EDGE, EINT1Callback);
```

```
        while (1)                                   // forever loop to keep flashing four LEDs
one at a time
        {
          // delay
        }

}
//Just press the RINT1
```

10

```
//
// Smpl_GPIO_RGBled : GPA12,13,14 output control RGB LED
//                    output low to enable LEDs
#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvUART.h"
#include "Driver\DrvSYS.h"

// Initial GPIO pins (GPA 12,13,14) to Output mode
void EINT1Callback(void)
{
  // GPA12 = Blue,  0 : on, 1 : off
        // GPA13 = Green, 0 : on, 1 : off
        // GPA14 = Red,   0 : on, 1 : off

        // set RGBled to Blue
   DrvGPIO_ClrBit(E_GPA,12); // GPA12 = Blue,  0 : on, 1 : off
   DrvGPIO_SetBit(E_GPA,13);
   DrvGPIO_SetBit(E_GPA,14);
        DrvSYS_Delay(1000000);

        // set RGBled to Green
   DrvGPIO_SetBit(E_GPA,12);
   DrvGPIO_ClrBit(E_GPA,13); // GPA13 = Green, 0 : on, 1 : off
   DrvGPIO_SetBit(E_GPA,14);
        DrvSYS_Delay(1000000);

        // set RGBled to Red
   DrvGPIO_SetBit(E_GPA,12);
   DrvGPIO_SetBit(E_GPA,13);
   DrvGPIO_ClrBit(E_GPA,14); // GPA14 = Red,   0 : on, 1 : off
```

```
        DrvSYS_Delay(1000000);

        // set RGBled to off
   DrvGPIO_SetBit(E_GPA,12); // GPA12 = Blue,  0 : on, 1 : off
   DrvGPIO_SetBit(E_GPA,13); // GPA13 = Green, 0 : on, 1 : off
   DrvGPIO_SetBit(E_GPA,14); // GPA14 = Red,   0 : on, 1 : off
        DrvSYS_Delay(1000000);
}


void Init_LED()
{
        // initialize GPIO pins
        DrvGPIO_Open(E_GPA, 12, E_IO_OUTPUT); // GPA12 pin set to output mode
        DrvGPIO_Open(E_GPA, 13, E_IO_OUTPUT); // GPA13 pin set to output mode
        DrvGPIO_Open(E_GPA, 14, E_IO_OUTPUT); // GPA14 pin set to output mode
        // set GPIO pins output Hi to disable LEDs
        DrvGPIO_SetBit(E_GPA, 12); // GPA12 pin output Hi to turn off Blue  LED
        DrvGPIO_SetBit(E_GPA, 13); // GPA13 pin output Hi to turn off Green LED
        DrvGPIO_SetBit(E_GPA, 14); // GPA14 pin output Hi to turn off Red   LED
}

int main (void)
{
        UNLOCKREG();                            // unlock register for programming
   DrvSYS_Open(48000000);     // set System Clock to run at 48MHz (PLL with 12MHz
crystal input)
        LOCKREG();                              // lock register from programming

        Init_LED();
                                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)
        {


        }
}
//Connect to GPA12,13,14
```

11

Raspberry pi

```
import RPi_GPO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(13,GPIO.OUT)l
let = int(input('Press a key')
If(let==1):
        GPIO.output(13,GPIO.high)
         time.sleep(1)
         while(true):
                  let1 = int(input())
                  if let1 == 0:
                           GPIO.output(13,GPIO.LOW)
                           break
GPIO.cleanup()
```