IoT Lab

Compiled by Ishan

Q1. Write a sample program to recognize port pins using getport.c

```
#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 title[16] = "SmplKeypad"; //Title
    char ch[16];
    Initial_panel(); //Initialize panel
    clr_all_panel(); //Clear panel
    print_lcd(0, title); //print the title
    while(1){
        number = DrvGPIO_GetPortBits(E_GPA); //Get what you're asked
        sprintf(ch, "%x", number); //Save it to the string "ch"
        print_lcd(1, ch);
        if(number==0xfffe) //1111 1111 1111 1110
            print_lcd(2, "A0");
        else if(number==0xfffd) //1111 1111 1111 1101
            print_lcd(2, "A1");
        //and so on up to A8
        else if(number==0xffbf) //1111 1111 1011 1111
            print_lcd(2, "A6");
        //and so on
    }
}
```

Q2. Write a program to use on board interrupt (buzzer) INT1 and external INT0

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

```c
#include "Driver\DrvSYS.h"

//External Interrupt Handler
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){
    DrvGPIO_Open(E_GPB, 11, E_IO_OUTPUT); //Configure GPB11 pin

    //External Interrupt
    DrvGPIO_Open(E_GPB, 15, E_IO_INPUT); //Configure GPB15 pin (for INT1) or GPB14 (for INT0)
    DrvGPIO_EnableEINT1(E_IO_BOTH_EDGE, E_MODE_EDGE, EINT1Callback);

    //Empty while loop
    while(1){}
}
```

Q3. Write a program to change the intensity of LED using PWM

```c
#include<stdio.h>
#include "NUC1xx.h"
#include "LCD_Driver.h"

int32_t main(void){
    char adc_value[15] = "ADC Value: ";

    InitPWM(); //Initialize Pulse Width Modulation
    InitADC(); //Initialize ADC

    Initial_panel(); //Initalize panel
    clr_all_panel(); //Clear panel

    //No idea what this means rn
    while(1){
```

```
        while(ADC->ADSR.ADF==0);
        ADC->ADSR.ADF = 1;
        PWMA->CMR0 = ADC->ADDR[7].RSLT<<4;
        sprintf(adc_value+4, "%d". ADC->ADDR[7].RSLT);
        print_lcd(0, adc_value);
        ADC->ADCR.ADST = 1
    }
}
```

Q4. Write a program to glow all LED using setport.c using interrupts

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

void Init_LED(){ //Initialize GPIO pins
    DrvGPIO_Open(E_GPC, 12, E_IO_OUTPUT);
    DrvGPIO_Open(E_GPC, 13, E_IO_OUTPUT);
    DrvGPIO_Open(E_GPC, 14, E_IO_OUTPUT);
    DrvGPIO_Open(E_GPC, 15, E_IO_OUTPUT);
}

int main(void){
    Init_LED();
    while(1){
        DrvGPIO_SetPortBits(E_GPC, 0xffff0fff); //0 to turn LED on
        DrvSYS_Delay(300000);
        DrvGPIO_SetPortBits(E_GPC, 0xffffffff); //1 to turn LED off
        DrvSYS_Delay(300000);
    }
}
```

Q5. Write a program to use ADC with functions.c choosing channel 6 (variable resistors)

Connections:
GND-GND
VCC-VCC
Signal-GPA6

```c
#include<stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvADC.h"
#include "Seven_Segment.h"
#include "LCD_Driver.h"

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

    Initial_panel(); //Initialize panel
    clr_all_panel(); //Clear panel
    print_lcd(0, "variable resistor");

    //What is this configuration oof
    DrvADC_Open(ADC_SINGLE_END, ADC_SINGLE_OP, 0x40, INTERNAL_HCLK, 1);

    while(1){
        DrvADC_StartConvert(); //Start AD conversion
        while(DrvADC_IsConversionDone()==FALSE); //wait till it's done
        value = ADC->ADDR[6].RSLT & 0xFFF; //get the value
        sprintf(TEXT, "Value: %d", value); //store value to buffer
        print_lcd(1, TEXT); //print it to LCD
    }
}
```

Q6. Write a program to use Smpl_LCD_Text.c using Interrupt

```c
#include<stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvSYS.h"
#include "NUC1xx-LB_002\LCD_Driver.h"

//External Interrupt Handler
void EINT1Callback(void){
    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){
    Initial_panel(); //Initialize panel
    clr_all_panel(); //Clear panel

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

    while(1){}
}
```

Q8. Smpl_GPIO_Interrupt.c using port pins of port A and port E

Connections:
First connect to GND and then to VCC

```
#include<stdio.h>
#include "NUC1xx.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;

//External interrupt handlers
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(4, "GPE interrupt!")
}
```

```c
int main(void){
    char TEXT[16];

    //Configure
    DrvGPIO_Open(E_GPA, 15, E_IO_INPUT);
    DrvGPIO_Open(E_GPE, 15, E_IO_INPUT);

    //Enable interrupts
    DrvGPIO_EnableInt(E_GPA, 15, E_IO_RISING, E_MODE_EDGE);
    DrvGPIO_EnableInt(E_GPE, 15, E_IO_RISING, E_MODE_EDGE);

    //Enable debounce
    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);
    }
}
```

Q9. Smpl_GPIO_LED1.c using interrupt INT0 or INT1

```c
#include<stdio.h>
#include "NU1cxx.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvSYS.h"

void Init_LED(){
    DrvGPIO_Open(E_GPC, 12, E_IO_OUTPUT);
}
```

```
//External interrupt handler
void EINT1Callback(void){
    DrvGPIO_ClrBit(E_GPC, 12); //0 to turn on LED
    DrvSYS_Delay(300000);
    DrvGPIO_SetBit(E_GPC, 12); //1 to turn off LED
    DrvSYS_Delay(300000);
}

int main(void){
    Init_LED();

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

    while(1){}
}
```

Q10. Smpl_GPIO_RGBled.c using interrupt

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

void Init_LED(){
    DrvGPIO_Open(E_GPA, 12, E_IO_OUTPUT);
    DrvGPIO_Open(E_GPA, 13, E_IO_OUTPUT);
    DrvGPIO_Open(E_GPA, 14, E_IO_OUTPUT);

    DrvGPIO_SetBit(E_GPA, 12);
    DrvGPIO_SetBit(E_GPA, 13);
    DrvGPIO_SetBit(E_GPA, 14);
}

//External interrupt handler
void ENT1Callback(void){
    //Blue (GPA12)
    DrvGPIO_ClrBit(E_GPA, 12);
    DrvGPIO_SetBit(E_GPA, 13);
```

```
    DrvGPIO_SetBit(E_GPA, 14);

    //Green (GPA13)
    DrvGPIO_SetBit(E_GPA, 12);
    DrvGPIO_ClrBit(E_GPA, 13);
    DrvGPIO_SetBit(E_GPA, 14);

    //Red (GPA14)
    DrvGPIO_SetBit(E_GPA, 12);
    DrvGPIO_SetBit(E_GPA, 13);
    DrvGPIO_ClrBit(E_GPA, 14);

    //Off
    DrvGPIO_SetBit(E_GPA, 12);
    DrvGPIO_SetBit(E_GPA, 13);
    DrvGPIO_SetBit(E_GPA, 14);
}

int main(void){
    Init_LED();

    //14 with INT0, 15 with INT1
    DrvGPIO_Open(E_GPB, 15, E_IO_INPUT);
    DrvGPIO_EnableEINT1(E_IO_BOTH_EDGE, E_MODE_EDGE, EINT1Callback);

    while(1){}
}
```

Q10. Using SSH blink LED using Raspberry Pi from remote system.

```
import RPi.GPIO as GPIO
import time

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

let = int(input('Press a key')
if let==1:
    GPIO.output(18, GPIO.HIGH)
```

```
    time.sleep(1)

    while(True):
        let1 = int(input())
        if let1 == 0:
            GPIO.output(18, GPIO.LOW)
            break
GPIO.cleanup()
```