LED

//

// 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 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)

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

{

DrvGPIO\_ClrBit(E\_GPC, 12); // output Low to turn on LED

DrvSYS\_Delay(300000); // delay

DrvGPIO\_SetBit(E\_GPC, 12); // output Hi to turn off LED

DrvSYS\_Delay(300000); // delay

}

}

BUZZER

//

// 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"

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, 11, E\_IO\_OUTPUT); // initial GPIO pin GPB11 for controlling Buzzer

while(1) {

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

}

}

RGB LED

//

// 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 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();

while (1)

{

// 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);

}

}

SEVEN SEGMENT

//

// Smpl\_7seg : counting from 0 to 9999 and display on 7-segment LEDs

//

#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

}

}

/\*---------------------------------------------------------------------------------------------------------\*/

/\* \*/

/\* Sample Code : Smpl\_7seg\_ADC7 \*/

/\* input : ADC[7] (12-bit) \*/

/\* output : Four Digit on 7-segment display \*/

/\* \*/

/\*---------------------------------------------------------------------------------------------------------\*/

#include <stdio.h>

#include "NUC1xx.h"

#include "Seven\_Segment.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 Delay(int32\_t count)

{

while(count--)

{

// \_\_NOP;

}

}

void seg\_display(int16\_t value)

{

int8\_t digit;

digit = value / 1000;

close\_seven\_segment();

show\_seven\_segment(3,digit);

Delay(5000);

value = value - digit \* 1000;

digit = value / 100;

close\_seven\_segment();

show\_seven\_segment(2,digit);

Delay(5000);

value = value - digit \* 100;

digit = value / 10;

close\_seven\_segment();

show\_seven\_segment(1,digit);

Delay(5000);

value = value - digit \* 10;

digit = value;

close\_seven\_segment();

show\_seven\_segment(0,digit);

Delay(5000);

}

/\*----------------------------------------------------------------------------

MAIN function

----------------------------------------------------------------------------\*/

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[7].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

}

}