8) TO SWITCH ON/OFF BULB USING RELAY

//

// 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\_GPA,0); // GPB11 = 0 to turn on Buzzer

DrvSYS\_Delay(10); // Delay

//DrvGPIO\_SetBit(E\_GPA,0); // 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\_GPA, 0, 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)

{

}

}

1. Int1/int0

#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

}

void EINT0Callback(void)

{

DrvGPIO\_ClrBit(E\_GPA,12); // GPB11 = 0 to turn on Buzzer

DrvSYS\_Delay(100000); // Delay

DrvGPIO\_SetBit(E\_GPA,12); // GPB11 = 1 to turn off Buzzer

DrvSYS\_Delay(100000); // 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

DrvGPIO\_Open(E\_GPA, 12, E\_IO\_OUTPUT);

//0 External Interrupt

DrvGPIO\_Open(E\_GPB, 5, E\_IO\_INPUT); // configure external interrupt pin GPB15

DrvGPIO\_EnableEINT1(E\_IO\_BOTH\_EDGE, E\_MODE\_EDGE, EINT1Callback); // configure external interrupt

DrvGPIO\_Open(E\_GPB, 14, E\_IO\_INPUT);

DrvGPIO\_EnableEINT0(E\_IO\_BOTH\_EDGE, E\_MODE\_EDGE, EINT0Callback);

while(1)

{

// DrvGPIO\_Open(E\_GPB, 15, E\_IO\_INPUT); // configure external interrupt pin GPB15

//DrvGPIO\_EnableEINT0(E\_IO\_BOTH\_EDGE, E\_MODE\_EDGE, EINT1Callback); // configure external interrupt

}

}

1. Get port number gpio

//#include <stdio.h>

#include "NUC1xx.h"

#include "Driver\DrvSYS.h"

#include "Driver\DrvGPIO.h"

int main (void)

{

int32\_t a;

char text[16];

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

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

Initial\_panel();

clr\_all\_panel();

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

{

DrvGPIO\_SetPortBits(E\_GPA,0);

a=DrvGPIO\_GetPortBits(E\_GPA);

sprintf(text,"port number %d",a); // delay

print\_lcd(0,text);

}

}

3)increment counter

//

// 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;

void GPIOAB\_INT\_CallBack(uint32\_t GPA\_IntStatus, uint32\_t GPB\_IntStatus)

{

int32\_t a;

char text[16];

DrvGPIO\_SetPortBits(E\_GPA,0);

a=DrvGPIO\_GetPortBits(E\_GPA);

if ((GPA\_IntStatus>>15) & 0x01) irqA\_counter++;

sprintf(text,"port number %d",a);

print\_lcd(3,"GPA interrupt !! ");

print\_lcd(2,text);

}

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\_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();

print\_lcd(0,"Smpl\_GPIO\_Intr");

while(1)

{

sprintf(TEXT,"IRQ\_A: %d",irqA\_counter);

print\_lcd(1, TEXT);

}

}

4)Adc channel 6 display analog value

#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

}

}

5)adc channel 0 and display on seven segment

#include <stdio.h>

#include "NUC1xx.h"

#include "Driver\DrvSYS.h"

#include "Seven\_Segment.h"

#include "DrvADC.h"

#include "LCD\_Driver.h"

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)

{ 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 , 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

seg\_display(value);

}

}

7)ADC Illumination led

#include <stdio.h>

#include "NUC1xx.h"

#include "LCD\_Driver.h"

#define BAUDRATE 9600

void InitADC(void)

{

/\* Step 1. GPIO initial \*/

GPIOA->OFFD|=0x00400000; //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 = 0x40;

/\* 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[6].RSLT<<4;

Show\_Word(0,11,' ');

Show\_Word(0,12,' ');

Show\_Word(0,13,' ');

sprintf(adc\_value+10,"%d",ADC->ADDR[7].RSLT);

print\_lcd(0, adc\_value);

Delay(20000);

ADC->ADCR.ADST=1;

}

}

6) adc pwm1 led intensity

// 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"

#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 InitPWM1(void)

{

/\* Step 1. GPIO initial \*/

SYS->GPAMFP.PWM1\_AD14=1;

/\* Step 2. Enable and Select PWM clock source\*/

SYSCLK->APBCLK.PWM01\_EN = 1;//Enable PWM clock

SYSCLK->CLKSEL1.PWM01\_S = 0;//Select 12Mhz for PWM clock source

PWMA->PPR.CP01=11; //Prescaler 0~255, Setting 0 to stop output clock

PWMA->CSR.CSR1=3; // PWM clock = clock source/(Prescaler + 1)/divider

//clock divider->0:/2, 1:/4, 2:/8, 3:/16, 4:/1

/\* Step 3. Select PWM Operation mode \*/

//PWM0

PWMA->PCR.CH1MOD=1; //0:One-shot mode, 1:Auto-load mode

//CNR and CMR will be auto-cleared after setting CH0MOD form 0 to 1.

PWMA->CNR1=0xFFFF;

PWMA->CMR1=0xFFFF;

PWMA->PCR.CH1INV=0; //Inverter->0:off, 1:on

PWMA->PCR.CH1EN=1; //PWM function->0:Disable, 1:Enable

PWMA->POE.PWM1=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();

InitPWM1();

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->CMR1=ADC->ADDR[7].RSLT<<4;

Show\_Word(0,11,' ');

Show\_Word(0,12,' ');

Show\_Word(0,13,' ');

sprintf(adc\_value+10,"%d",ADC->ADDR[7].RSLT);

print\_lcd(0, adc\_value);

Delay(20000);

ADC->ADCR.ADST=1;

}

}

9)ldr program

//

// Smpl\_ADC\_VR1 : use ADC7 to read Variable Resistor (on-board)

//

#include <stdio.h>

#include "NUC1xx.h"

#include "DrvSYS.h"

#include "NUC1xx-LB\_002\LCD\_Driver.h"

void InitADC(void)

{

/\* Step 1. GPIO initial \*/

GPIOA->OFFD|=0x00400000; //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 = 0x40;

/\* 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;

}

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

MAIN function

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

int32\_t main (void)

{

char TEXT1[16]="ADC Value: ";

UNLOCKREG();

//SYSCLK->PWRCON.XTL12M\_EN = 1; // enable external clock (12MHz)

//SYSCLK->CLKSEL0.HCLK\_S = 0; // select external clock (12MHz)

LOCKREG();

InitADC(); // initialize ADC

Initial\_panel(); // initialize LCD pannel

clr\_all\_panel(); // clear LCD panel

print\_lcd(0, "Smpl\_ADC\_VR1");

while(1)

{

while(ADC->ADSR.ADF==0); // wait till conversion flag = 1, conversion is done

ADC->ADSR.ADF=1; // write 1 to clear the flag

sprintf(TEXT1+10,"%4d",ADC->ADDR[6].RSLT); // convert ADC7 value into text

print\_lcd(1, TEXT1); // output TEXT to LCD

DrvSYS\_Delay(20000); // delay

ADC->ADCR.ADST=1; // restart ADC sample

}

}

10)stepper motor

//

// Sampl\_GPIO\_StepMotor

// 5V Step Motor 28BYJ-48, driver IC = ULN2003A

//

// Driver board connections:

// ULN2003A NUC140

// INA to GPA3

// INB to GPA2

// INC to GPA1

// IND to GPA0

//

#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(20000);//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(20000);//delay 2000us = 2ms

}

}

}

int main (void)

{

CW\_MOTOR(d2); // Clockwise for 360 degree

//CCW\_MOTOR(d2/2);// Counter-Clockwise for 180 degree

}