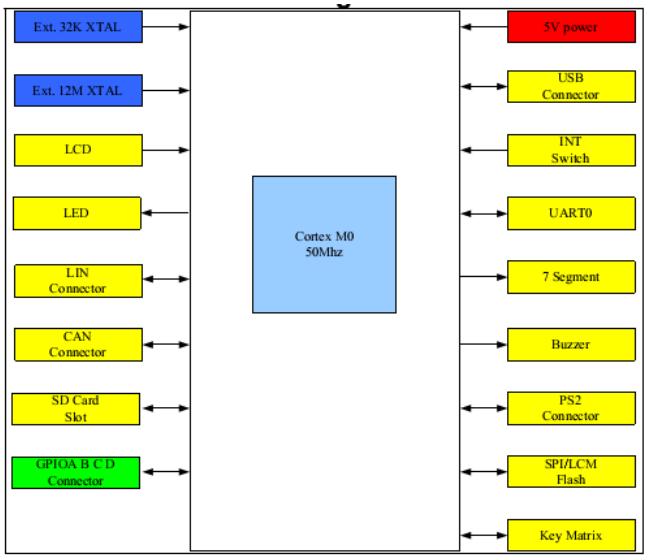
Konsep I/O pada NUC1XX series

1. **Tujuan Praktikum**
   Memahami konsep dasar I/O pada NU-LB NUC14O supaya bisa di aplikasikan pada mikrokontroler sejenisnya.
2. **Alat dan Bahan**
   Modul NU-LB NUC140
   CoID, CoSmart,GCC
3. **Landasan Teori**

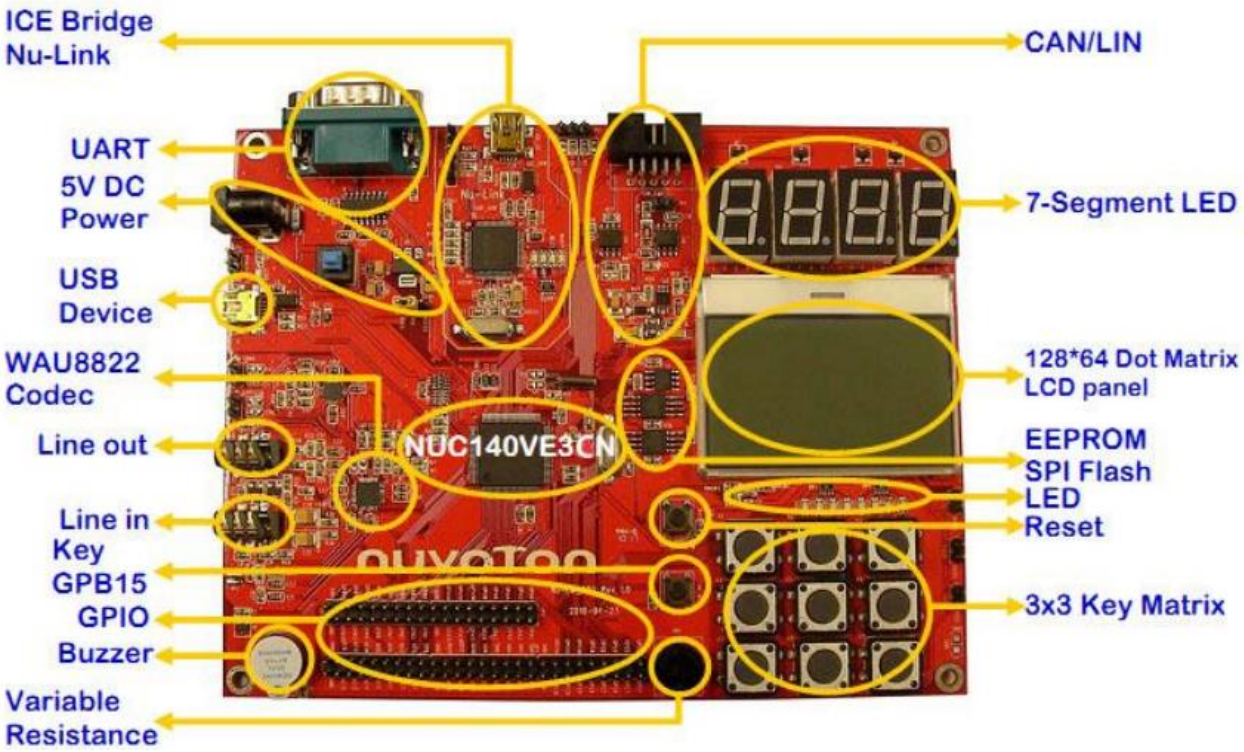
NUC1XX series adalah ARM ® Cortex ™-mikrokontroler dengan M0 core didalamnya yang cocok digunakan untuk kontrol industri dan aplikasi yang membutuhkan fungsi komunikasi khusus. Cortex ™-M0 adalah prosesor ARM terbaru dengan kinerja 32-bit dengan biaya yang setara dengan mikrokontroler 8-bit. NuMicro Seri NUC100 memiliki inti ARM Cortex M0 yang tertanam dengan kecepatan hingga 50 MHz, dilengkapi dengan memori flash untuk program 32K/64K/128Kbyte. SRAM sebesar 4K/8K/16K-byte dan memori flash loader untuk ISP (In System Programming) sebesar 4K-byte. Selain itu juga dilengkapi dengan berbagai macam periferal, seperti GPIO, Timer, Watchdog Timer, RTC, PDMA, UART, SPI/MICROWIRE, I2C, I2S, PWM, LIN, CAN, PS2, USB 2.0 FS Device, ADC 12-bit, komparator analog, Low Voltage Reset dan Brown-out Detector. Gambar 1 menunjukkan diagram blok dari NuMicro NUC130/140 Series.



Gambar 1. Diagram blok NUC130/140

NU-LB NUC140

Pada board NU-LB NUC140 dirancang khusus untuk pembelajaran dimana menggunakan catu daya 5 volt yang bisa diperoleh lewat melalui konektor USBatau konektor catu daya adaptor. Pada data sheetsheet chip NUC140VE3CN menunjukkan bahwa tegangan maksimumn untuk mencatu daya chip sebesar 5,5 volt. Pada board terdapat juga catudaya teregulasi 3.3V menggunakan chip AMS1117. Tegangan dari Powerjack 3 pin dan konektor USB dilewatkan melalui dioda sehingga aman dari kesalahan polaritas pemasangan, namun tidak melindungi dari kerusakan jika tegangan masuk melebihi 5.5V. Gambar 2 menunjukkan layout NUC140 Learning Board. Sedangkan Tabel 1 menunjukkan penggunaan pin pada Learning Board tsb.



Gambar 2. Lay out NU-LB-NUC140

Block	Pin	Function
ICE Bridge Nu-Link	ICE_CLK ICE_DATA	SWD interface
UART	GPB0 GPB1	UART0 Rx UART0 Tx
WAU8822 codec	GPC0 GPC1 GPC2 GPC3 GPA15	I2SLRCLK I2SBCLK I2SDI I2SDO I2SMCLK
	GPA8 GPA9	I2C0 SDA I2C0 SCL
	GPE14	Line out Enable/Disable
	GPE15	Line in plug in/out detect
Key GPB15	GPB15	INT0
CAN	GPD6 GPD7	CAN0 Rx CAN0 Tx
	GPB12~13	CAN transceiver speed
LIN	GPB4 GPB5	UART1 Rx UART1 Tx
	GPB6	LIN transceiver wakeup function
	GPB7	LIN transceiver Enable/Disable
7-Seg LED	GPE0~7	Row
	GPC4~7	Column

Black Dot Matrix LCD Panel	GPD8 GPD9 GPD10 GPD11	SPI3 SS30 SPI3 SPCLK SPI3 MISO0 SPI3 MOSI0
	GPD14	LCD backlight power
Variable Resistance	GPA7	ADC interface
Buzzer	GPB11	PWM4
Key Matrix	GPA0~5	GPIO
Reset	RESET	Reset
EEPROM	GPA10 GPA11	I2C1 SDA I2C1 SCL
SD Slot	GPD12	SD power
	GPD13	SD card detect
	GPC8~11	SD interface
FLASH	GPD0 GPD1 GPD2 GPD3 GPD4 GPD5	SPI2 SS20 SPI2 SPCLK SPI2 MISO0 SPI2 MOSI0 SPI2 MISO1 SPI2 MOSI1
LED	GPA12 GPA13 GPA14	PWM0 PWM1 PWM2
	GPC12~15	GPIO

Perangkat Lunak Kompilasi

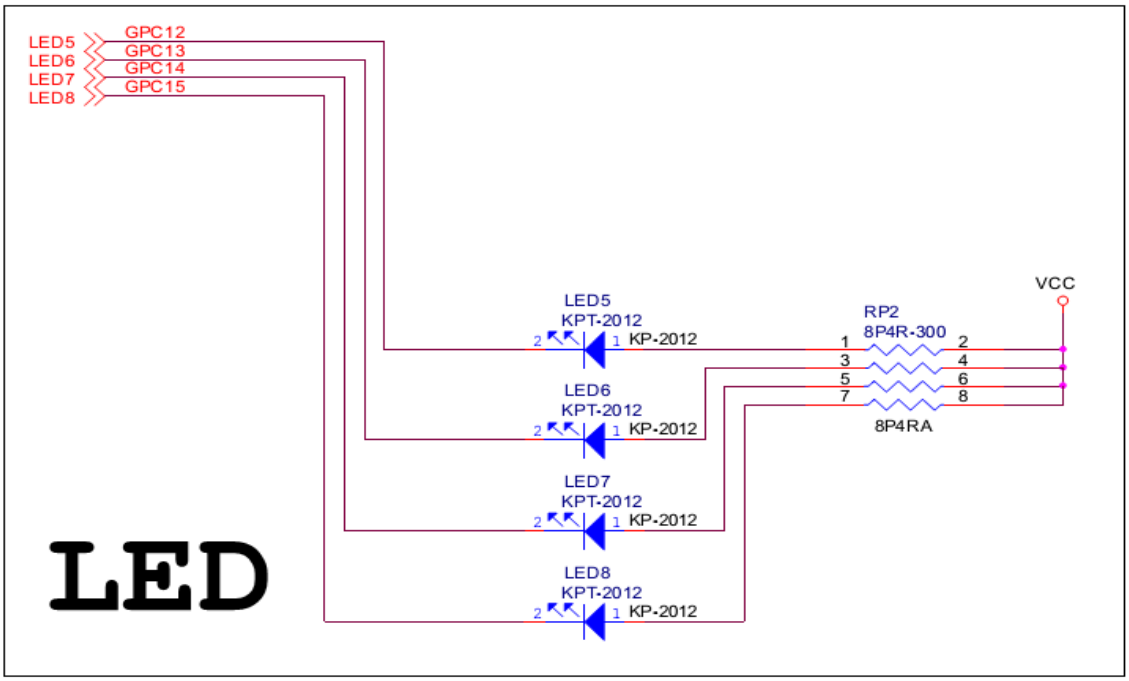
Untuk membuat sebuah proyek aplikasi menggunakan ARM® Cortex™-M0 dari NuMicro Nuvoton NUC1XX series, dapat digunakan CooCox IDE.Berikut langkah yang digunakan dalam membuat sebuah project baru dengan menggunakan CooCox IDE:

1. Klik menu Project >> New Project lalu tulis nama proyek aplikasi yang akan dibuat dan tentukan lokasi penyimpanan project tersebut. Setelah itu klik finish;
2. Lalu klik Nuvoton dan pilih jenis mikrokontroler NUC140VE3CN;
3. Kemudian akan muncul jendela repository. Pada bagian ini dapat dipilih pustaka components yang akan digunakan misalkan untuk I/O centang GPIO pada PERIPERAL NUVOTON.
4. Kemudian buka file main.c pada bagian jendela project. Pada main.c inilah dapat ditulis kode program yang akan digunakan pada mikrokontroler Nuvoton;
5. Untuk mengkompile program klik menu Project >> build atau tekan F7;
6. Untuk mendownload program yang telah ditulis, klik Flash >> Program Download.

PERCOBAAN I
APLIKASI LED dan LED RGB

Pada board NU-LB_NUC140 terdapat empar buah led dengan pin input telah terhubung pada port yang telah ditentukan seperti yang terlihat pada gambar 3. Dalam menyalakan LED pada program digunakan fungsi DrvGPIO_SetPortBits untuk membuat port GPIO bernilai HIGH dan LOW tertentu pada bit-bitnya. Untuk membuat program, klik panel repository lalu pilih M0 Cmsis Core, Cmsis Boot, System definition, SYS, dan GPIO.

Skema rangkaian LED pada NU-LB-NUC140



Gambar 3. Konfigurasi Rangkaian LED pada NU-LB-NUC140

Langkah Kerja

Gunakan CooCox IDE dalam membuat sebuah project baru untuk menyalakan led :

1. Klik menu Project >> New Project lalu tulis nama proyek aplikasi yang akan dibuat dan tentukan lokasi penyimpanan project tersebut. Setelah itu klik finish;
2. Lalu klik Nuvoton dan pilih jenis mikrokontroler NUC140VE3CN;
3. Kemudian akan muncul jendela repository, Untuk menyalakan Led cukup centang GPIO pada PERIPERAL NUVOTON. Maka akan tercentang M0 Cmsis Core, Cmsis Boot, System definition, SYS, dan GPIO secara otomatis
4. Kemudian buka file main.c pada bagian jendela project. Pada main.c inilah dapat ditulis kode program yang akan digunakan pada mikrokontroler Nuvoton;
5. Untuk mengkompile program klik menu Project >> build atau tekan F7;

6. Untuk mendownload program yang telah ditulis, klik Flash >> Program Download.

Percobaan 1 (a)
Lalu ketik listing program nyalakan LED berikut:

```
#include"DrvGPIO.h"
#include"DrvSYS.h"


void delay_loop(void)
{ uint32_t i,j;
  for(i=0;i<4;i++)
    { for(j=0;j<60000;j++);}
}


int main(void)
{/* mengatur penggunaan sumber clock */
        UNLOCKREG();
        DrvSYS_SetOscCtrl(E_SYS_XTL12M,1);

        DrvSYS_Delay(5000);
        DrvSYS_SelectHCLKSource(0);
        LOCKREG();
        DrvSYS_SetClockDivider(E_SYS_HCLK_DIV,0);

            /* Mengaktifkan fungsi GPC pin 12, 13, 14, dan 15 sebagai OUTPUT
                * Menggunakan fungsi @DrvGPIO_Open
                * Parameter untuk fungsi @DrvGPIO_Open bisa dilihat pada file DrvGPIO.c
                * Input parameter : PortC, Pin 12 / Pin 13 / Pin 14 / Pin 15, OUTPUT
                */
                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);


   while(1)
   {/* Mengeluarkan logika Low di GPC pin 12 / 13 / 14 / 15
        * Menggunakan fungsi @DrvGPIO_ClrBit
        * Parameter untuk fungsi @DrvGPIO_ClrBit bisa dilihat pada file DrvGPIO.c
        * Input parameter : PortC, [variabel i]
        */
        DrvGPIO_ClrBit(E_GPC,12);
        DrvGPIO_ClrBit(E_GPC,13);
        DrvGPIO_ClrBit(E_GPC,14);
```

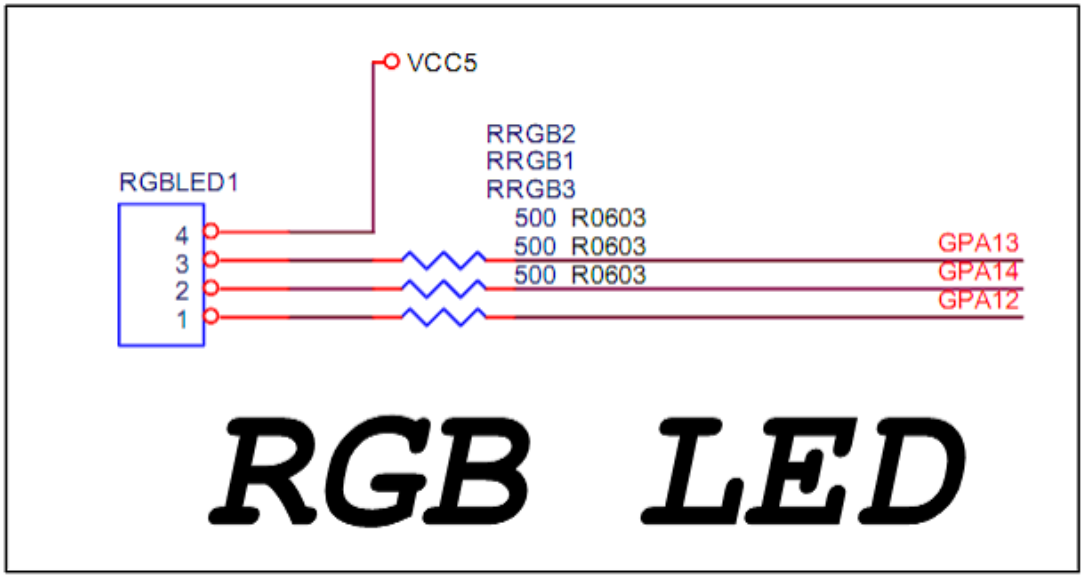
```
        DrvGPIO_ClrBit(E_GPC,15);

        DrvSYS_Delay(100000);

        /* Mengeluarkan logika Low di GPC pin 12 / 13 / 14 / 15
         * Menggunakan fungsi @DrvGPIO_SetBit
         * Parameter untuk fungsi @DrvGPIO_SetBit bisa dilihat pada file DrvGPIO.c
         * Input parameter : PortC, [variabel i]
         */
        DrvGPIO_SetBit(E_GPC,12);
        DrvGPIO_SetBit(E_GPC,13);
        DrvGPIO_SetBit(E_GPC,14);
        DrvGPIO_SetBit(E_GPC,15);
        DrvSYS_Delay(100000);
                        }
    }
```

Tugas

- Menampilkan biner 0000 sampai dengan 1111 dengan delay_loop sebagai jeda/delay.
- Menyalakan led pada port GPIOB

Percobaan 1 (b)

Pada latihan kedua akan dibuat program untuk menyalakan LED RGB yang tersedia pada port GPA12-14. Gambar 4. menunjukkan konfigurasi LED RGB yang terdapat pada NuMicro 1xx Series :



Gambar 4. Konfigurasi Rangkaian LED RGB

Buatlah program untuk menyalakan LED RGB tersebut dengan menuliskan listing program berikut :

```
#include "DrvSYS.h"
#include "DrvGPIO.h"

void delay_loop(void)
{ uint32_t i,j;
  for(i=0;i<4;i++)
    { for(j=0;j<60000;j++);}
}

int main(void)
{
int i=0;
UNLOCKREG();
DrvSYS_SetOscCtrl(E_SYS_XTL12M, 1);
DrvSYS_Delay(5000);
DrvSYS_SelectHCLKSource(0);
LOCKREG();
DrvSYS_SetClockDivider(E_SYS_HCLK_DIV, 0);
for(i=12;i<16;i++)
{
DrvGPIO_Open(E_GPC, i, E_IO_OUTPUT);
/*Set semua GPA12-15 sebagai output */
}
while(1)
{
     DrvGPIO_SetPortBits(E_GPC, 0xE000);
         delay_loop();
         DrvGPIO_SetPortBits(E_GPC, 0xD000);
         delay_loop();
         DrvGPIO_SetPortBits(E_GPC, 0xC000);
         delay_loop();
         DrvGPIO_SetPortBits(E_GPC, 0xB000);
         delay_loop();
         DrvGPIO_SetPortBits(E_GPC, 0xA000);
         delay_loop();
         DrvGPIO_SetPortBits(E_GPC, 0x9000);
         delay_loop();
         DrvGPIO_SetPortBits(E_GPC, 0x8000);
         delay_loop();
         DrvGPIO_SetPortBits(E_GPC, 0x7000);
         delay_loop();
         DrvGPIO_SetPortBits(E_GPC, 0x6000);
```

```
        delay_loop();
        DrvGPIO_SetPortBits(E_GPC, 0x5000);
        delay_loop();
        DrvGPIO_SetPortBits(E_GPC, 0x4000);
        delay_loop();
        DrvGPIO_SetPortBits(E_GPC, 0x3000);
        delay_loop();
        DrvGPIO_SetPortBits(E_GPC, 0x2000);
        delay_loop();
        DrvGPIO_SetPortBits(E_GPC, 0x1000);
        delay_loop();
        DrvGPIO_SetPortBits(E_GPC, 0x0000);
        delay_loop();
}
}
```

Tugas

- Menampilkan warna-warna pelangi pada LED RGB dengan delay 500ms.

PERCOBAAN II
APLIKASI SEVEN SEGMEN

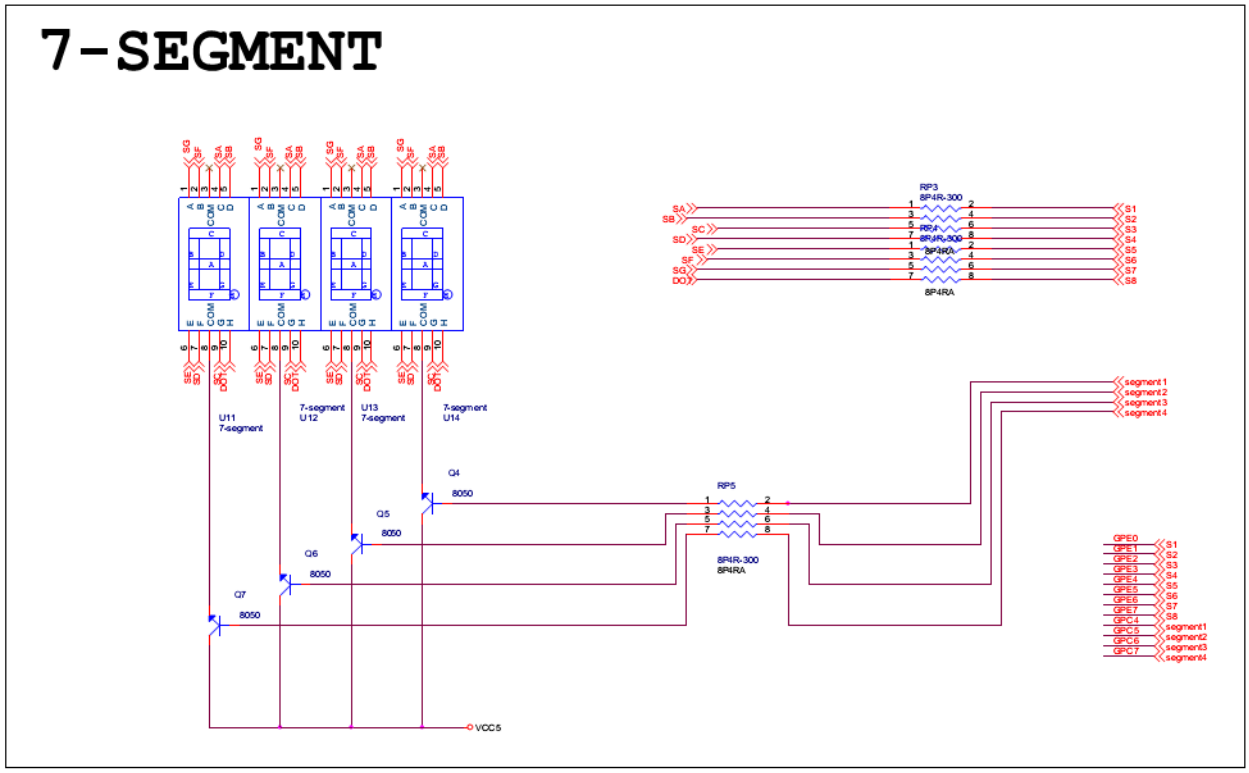
Pada NuMicro NU-LB-NUC140 terdapat 4 buah Seven Segment yang terdapat pada PORT GPE.0-7 sebagai Row (seven segment) dan GPC.4-7 sebagai column (switching seven segment). Untuk menggunakan Seven Segment ini, pengguna perlu memasukkan Seven_Segment.h dan Seven_Segment.c kedalam program. Pada program, pengguna dapat menggunakan syntax "show_seven_segment(column,row)" untuk menampilkan karakter pada Seven Segment. Karena Seven Segment tersebut terhubung dengan Multiplexer terlebih dahulu sebelum masuk ke mikrokontroler, maka pengguna harus mematikan Multiplexer terlebih dahulu sebelum memulai perintah selanjutnya dengan syntax close_seven_segment(). Gambar 5 berikut ini menunjukkan konfigurasi Seven Segment pada NuMicro 1XX series:

Untuk memasukan Seven_Segment.h dan Seven_Segment.c terdapat langkah-langkah sebagai berikut:

- Klik kanan pada folder di project
- Add file Seven_Segment.h (Misalkan C:\Nuvoton\BSP Library\ NUC100SeriesBSP_CMSIS_v1.05.003\NuvotonPlatform_IAR\Include\NUC1xx-LB_002 \ Seven_Segment.h)
- Open (akan muncul Seven_Segment.h dibawah main.c)

Kemudian masukan Seven_Segment.c sebagai berikut

- Klik kanan pada folder di project
- Add file Seven_Segment.c (Misalkan C:\Nuvoton\BSP Library\ NUC100SeriesBSP_CMSIS_v1.05.003\NuvotonPlatform_IAR\Src\NUC1xx-LB_002\ Seven_Segment.c)
- Open (akan muncul Seven_Segment.c dibawah main.c)



Gambar 5. Konfigurasi Rangkaian Seven Segment

Percobaan 2.

Listing program dibawah ini

```
#include "DrvSYS.h"
#include "DrvGPIO.h"
#include "Seven_Segment.h"

void delay_loop(void)
 {
 uint32_t i,j;
        for(i=0;i<3;i++)
        {
                for(j=0;j<60000;j++);
   }

 }

int main(void)
{
         int i=0,j=0;
                UNLOCKREG();
                DrvSYS_SetOscCtrl(E_SYS_XTL12M, 1);
                DrvSYS_Delay(5000);
```

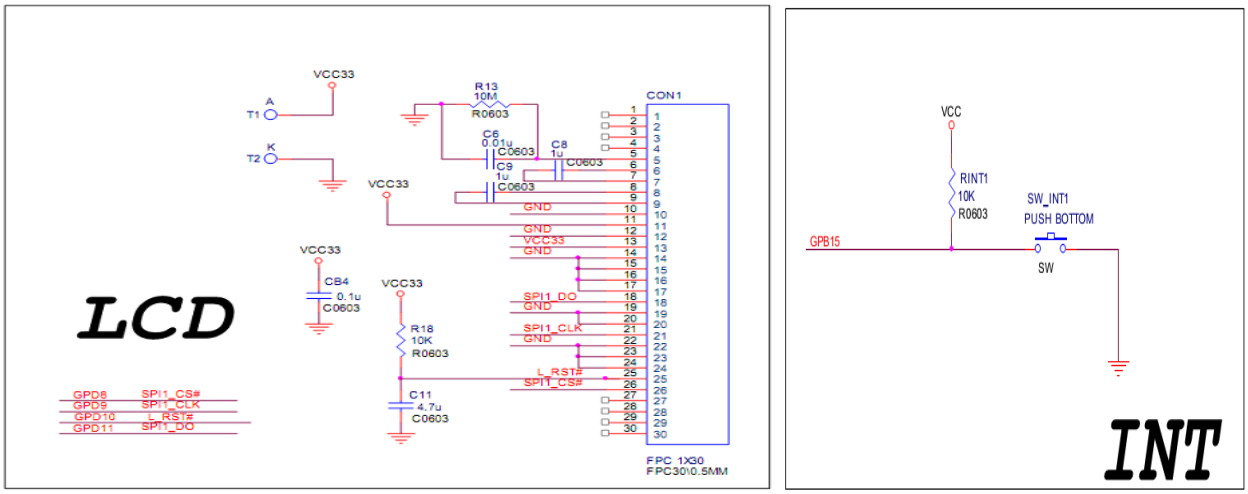
```
                DrvSYS_SelectHCLKSource(0);
                LOCKREG();

                DrvSYS_SetClockDivider(E_SYS_HCLK_DIV, 0);

  while(1)
  {   for(i=0;i<9;i++)
                { for(j=0;j<4;j++)
                  {     close_seven_segment();
                        show_seven_segment(j,i);
                        delay_loop();   }
                }
  }
}
```

Tugas

- Menampilkan NIM pada 7-segment!
- Menampilkan nilai dari 48 sampai 35 (Hitung Mundur)

PERCOBAAN III
APLIKASI LCD

Untuk menggunakan LCD pada NU-LB-NUC140, pengguna perlu memanggil sebuah library, yaitu: LCD_Driver.h, LCD_Driver.c dan Ascii_Table.c . Proses memasukan file tersebut bisa diliha seperti langkah-langkah pada percobaan sebelumnya.



Gambar 6. Konfigurasi Rangkaian LCD dan interupt

Setelah semua file dan library tersebut dimasukkan, tuliskan listing program berikut pada main.c.

Percobaan 3.
Listing program

```
#include "NUC1xx.h"
#include "DrvSYS.h"
#include "DrvGPIO.h"
#include "LCD_Driver.h"

void delay_loop(void)
 {
 uint32_t i,j;
        for(i=0;i<3;i++)
        {
                for(j=0;j<60000;j++);
   }
 }

int main(void)
{
        UNLOCKREG();
```

```
        DrvSYS_SetOscCtrl(E_SYS_XTL12M, 1);
        DrvSYS_Delay(5000);
        DrvSYS_SelectHCLKSource(0);
        LOCKREG();

        DrvSYS_SetClockDivider(E_SYS_HCLK_DIV, 0);

        Initial_pannel();
        clr_all_pannal();

        print_lcd(0, "fakultas teknik");
        print_lcd(1, "    UDINUS    ");

        DrvGPIO_ClrBit(E_GPD,14);  // backlight ON - comment this to turn OFF

        while(1)
        {
                if (DrvGPIO_GetBit(E_GPB, 15)==0)
                {
                        print_lcd(2, "   PRAKTIKUM   ");
                        print_lcd(3, " MIKROKONTROLER ");
                }
                else
                {
                        print_lcd(2, " MIKROKONTROLER ");
                        print_lcd(3, "   NUC140VE3CN  ");
                }

        }
}
}
```

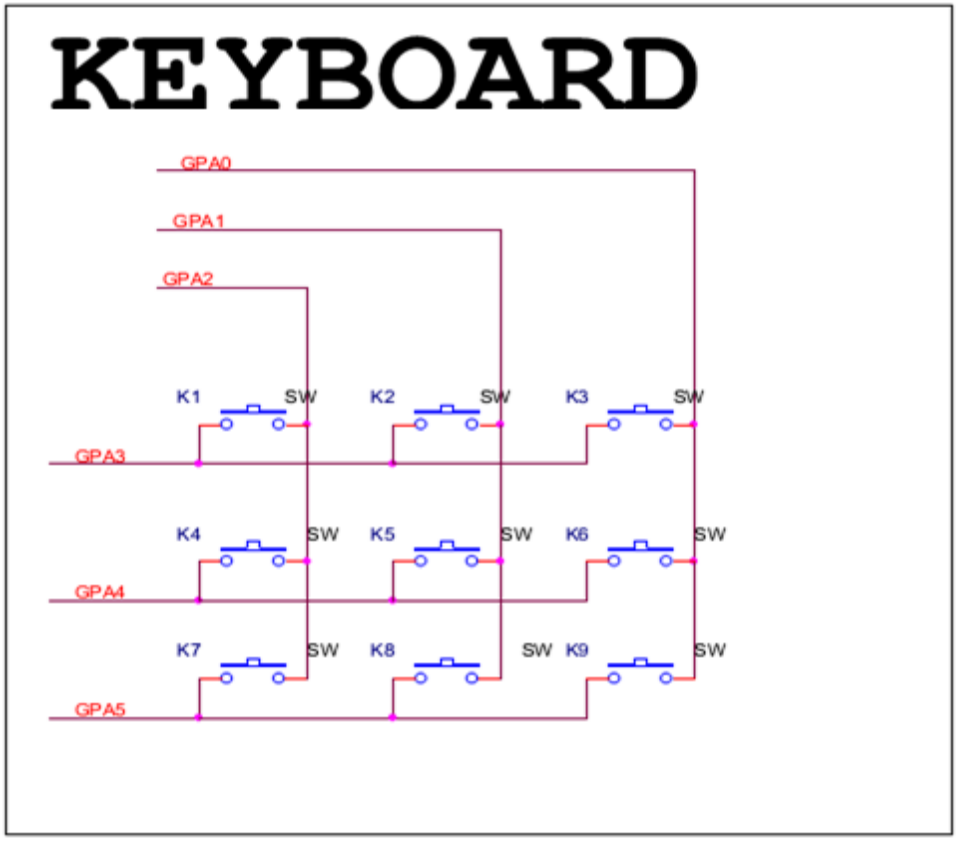
Pada listing program diatas menggunakan interupt dengan masukan INT pada GPB15 yaitu tombol push buttom dengan aktif LOW.

Tugas

- Menampilkan Nama dan NIM bergeser kesamping!
- Menampilkan nilai dari 48 sampai 35 (Hitung Mundur)

PERCOBAAN IV
APLIKASI KEYPAD

Seperti pada peercobaan sebelumnya, untuk dapat menggungakan keypad pada NU-LB-NUC140 perlu dilakukan pemasukan library keypad yaitu library ScanKey.h dan file ScanKey.c. terlebih dulu. Langkahnya bisa dilihat pada percobaan sebelumnya. Pada NU-LB-NUC140 menggunakan keypad matrik 3x3 dengan menggunakan PGA 0-5 seperti terlihat pada gambar 7.



Gambar 7. Konfigurasi rangkaian Keypad

Percobaan 4.
Berikut listing program penggunaan keypad:

```
#include  "NUC1xx.h"
#include  "DrvSYS.h"
#include  "DrvGPIO.h"
#include  "LCD_Driver.h"
#include  "ScanKey.h"

void delay_loop(void)
 {
 uint32_t  i,j;
        for(i=0;i<3;i++)
        {
```

```
                for(j=0;j<60000;j++);
    }
 }

int main(void)
{
        unsigned char temp;
        UNLOCKREG();
        DrvSYS_SetOscCtrl(E_SYS_XTL12M, 1);
        DrvSYS_Delay(5000);
        DrvSYS_SelectHCLKSource(0);
        LOCKREG();

        DrvSYS_SetClockDivider(E_SYS_HCLK_DIV, 0);
        Initial_pannel();  //call initial pannel function
        clr_all_pannal();

        print_lcd(0, "   PRAKTIKUM   ");
        print_lcd(1, " MIKROKONTROLER ");

        OpenKeyPad();

        while(1)
        {
                temp=Scankey();
                if (temp==1)
                {
                        print_lcd(3,"TOMBOL 1 ");
                }
                if (temp==2)
                {
                        print_lcd(3,"TOMBOL 2 ");
                }
                if (temp==3)
                {
                        print_lcd(3,"TOMBOL 3 ");
                }
                if (temp==4)
                {
                        print_lcd(3,"TOMBOL 4 ");
                }
                if (temp==5)
                {
                        print_lcd(3,"TOMBOL 5 ");
                }
                if (temp==6)
```

```
                {
                        print_lcd(3,"TOMBOL  6 ");
                }
                if (temp==7)
                {
                        print_lcd(3,"TOMBOL  7 ");
                }
                if (temp==8)
                {
                        print_lcd(3,"TOMBOL  8 ");
                }
                if (temp==9)
                {
                        print_lcd(3,"TOMBOL  9 ");
                }
        }
}
```

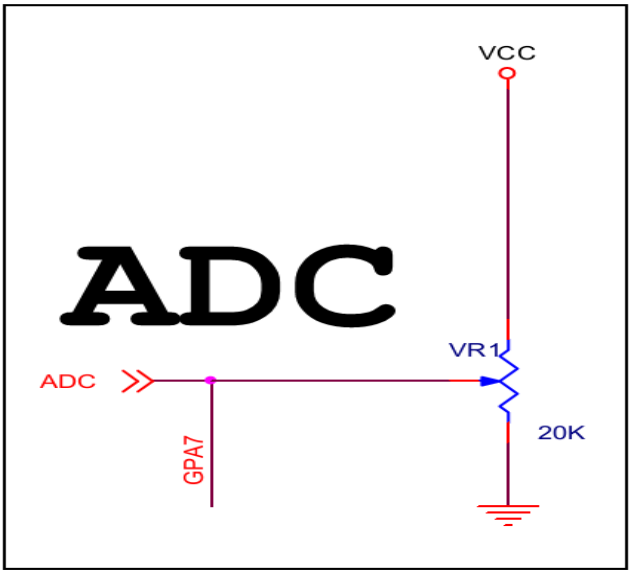
Pada program diatas, akan ditampilkan tombol yang telah ditekan pada LCD

Tugas

- Tampilkan angka pada keypad pada 7-segment dengan menekan key pad.
- Tampilkan nilai counter dari 1-10 saat keypad ditekan

PERCOBAAN V
APLIKASI ADC

Pada NuMicro 140 series terdapat variable resistor yang terhubung dengan port ADC dengan resolusi 12bit pada GPA.7 Gambar 8 berikut ini merupakan konfigurasi ADC pada NuMicro 140 series Development Board



Gambar 8 Konfigurasi Rangkaian ADC

Berikut program yang digunakan untuk menginisialisasi fungsi ADC pada NuMicro 1XX Series ditampilkan pada 7 segment

Percobaan 4(a).

Listing Program ADC

```
#include "DrvADC.h"
#include "DrvSYS.h"
#include "DrvGPIO.h"
#include "DrvPWM.h"

uint8_t gu8AdcIntFlag;
uint16_t u16ConversionData;
char temp[10];
S_DRVPWM_TIME_DATA_T sPt;

void AdcIntCallback(uint32_t u32UserData)
{
        /*
         * Mengambil nilai ADC di channel 7
         * Parameter input bisa dilihat di @DrvADC_GetConversionData di file DrvADC.
         */
```

```
        u16ConversionData  = DrvADC_GetConversionData(7);
        //ambil  data hasil  konversi
        sPt.u8HighPulseRatio  = ((u16ConversionData>>4)/3)+15;
        DrvPWM_SetTimerClk(DRVPWM_TIMER0,  &sPt);

        int digit=0,  value=0;

        digit  = (u16ConversionData/1000  %10);
        close_seven_segment();
        show_seven_segment(3,digit);
        DrvSYS_Delay(6000);
        digit  = (u16ConversionData/100  %10);
        close_seven_segment();
        show_seven_segment(2,digit);
        DrvSYS_Delay(6000);
        digit  = (u16ConversionData/10  %10);
        close_seven_segment();
        show_seven_segment(1,digit);
        DrvSYS_Delay(6000);
        digit  = (u16ConversionData  %10);
        close_seven_segment();
        show_seven_segment(0,digit);
        DrvSYS_Delay(6000);
}

int main(void)
{
UNLOCKREG();
        DrvSYS_SetOscCtrl(E_SYS_XTL12M,1);
        DrvSYS_Delay(5000);
        DrvSYS_SelectHCLKSource(0);
        LOCKREG();
        DrvSYS_SetClockDivider(E_SYS_HCLK_DIV,0);
        DrvGPIO_Open(E_GPA,7,E_IO_INPUT);
        DrvADC_Open(ADC_SINGLE_END,  ADC_CONTINUOUS_OP, 0x80,
EXTERNAL_12MHZ, 5);


        DrvADC_ConfigADCChannel7(EXTERNAL_INPUT_SIGNAL);
                //kanal  7 terhubung  ke input  analog  ekternal

        DrvADC_StartConvert();                  //konversi  ADC dimulai
        DrvADC_EnableADCInt(AdcIntCallback,  0);
        DrvPWM_Open();
        DrvGPIO_InitFunction(E_FUNC_PWM0);   //mengaktifkan  fungsi  PWM pada GPIO
        sPt.u8Mode = DRVPWM_AUTO_RELOAD_MODE;        //jenis  operasi auto reload
```

```
        sPt.u32Frequency  = 50;                         //frekuensi  = 50Hz = 20ms
        sPt.u8HighPulseRatio  =0;
        sPt.i32Inverter  = 0;

        /* Mengunnakan  PWM0 dan sumber  clock crystal  external  12MHz
         * Informasi  @DrvPWM_SelectClockSource()  dapat dilihat  pada file  DrvPWM.c
         */
        DrvPWM_SelectClockSource(DRVPWM_TIMER0,  DRVPWM_INTERNAL_22M);
        //sumber  clock  PWM = eksternal  12MHz
        /* Mengunnakan  PWM0
         * Informasi  @DrvPWM_SetTimerClk()  dapat dilihat  pada file  DrvPWM.c
         */
        DrvPWM_SetTimerClk(DRVPWM_TIMER0,  &sPt);
        //atur  konfigurasi  untuk  PWM Timer0
        /* Aktifkan  I/O untuk  PWM0
         * Informasi  @DrvPWM_SetTimerIO()  dapat dilihat  pada file  DrvPWM.c
         */
        DrvPWM_SetTimerIO(DRVPWM_TIMER0,  1);
        //mengaktifkan  I/O dari PWM Timer  0
        /* Aktifkan  fungsi  PWM0
         * Informasi  @DrvPWM_Enable()  dapat dilihat  pada file  DrvPWM.c
         */
        DrvPWM_Enable(DRVPWM_TIMER0,  1);
        //mengaktifkan  PWM Timer0.
        close_seven_segment();

        while(1)
    {
    }
}
```

Pada percobaan sebelumnya telah menampilakan data ADC pada 7 segment, sedangkan pada percobaan berikut menampilkan data ADC pada LCD.

Percobaan 4(b).

Listing Program ADC

```
#include  <stdio.h>
#include  "DrvGPIO.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
  ---------------------------------------------------------------------*/
int main (void)
{
	//Enable 12Mhz and set HCLK->12Mhz
	char adc_value[15]=" DATA ADC :";

	UNLOCKREG();
	SYSCLK->PWRCON.XTL12M_EN = 1;
	SYSCLK->CLKSEL0.HCLK_S = 0;
	LOCKREG();

	InitPWM();
	InitADC();

	Initial_pannel();  //call initial pannel function
	clr_all_pannal();

	/* 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+10,"%d",ADC->ADDR[7].RSLT);
		print_lcd(0,"     SISTEM    " );
		print_lcd(1,"  MIKROKONTROLER " );
		print_lcd(2," FAKULTAS TEKNIK" );
		print_lcd(3,  adc_value);
		Delay(20000);
```

```
            ADC->ADCR.ADST=1;
        }
}
```

Tugas

- Tampilkan hasil konversi ADC ke Tegangan.
- Tampilkan Hasil konversi ADC ke 7-Segment

PWM
PENGATURAN KECEPATAN MOTOR DC

PERCOBAAN V
MOTOR

Motor merupakan salah satu perangkat yang sering digunakan dalam kehidupan sehari-hari, industri ataupun robotika. Kebanyakan motor beroprasi melalui interaksi medan magnet dan konduktor yang dihasilkan oleh arus listrik. Pada terdapat stator (bagian yang tidak berputar) dan kumparan jangkar yang disebut rotor (bagian yang berputar). Motor stepper merupakan salah satu jenis motor yang prinsip kerjannya berdasarkan step by step. Besar pergeseran step tergantung dari kontruksi motor. Besar step ini disebut dengan derajat/step atau step angel, angka ini berkisar : 1,8 - 2,5 – 3,75 – 7,5 – 15 dan 30º.

Gambar 7 Driver dan Motor Servo

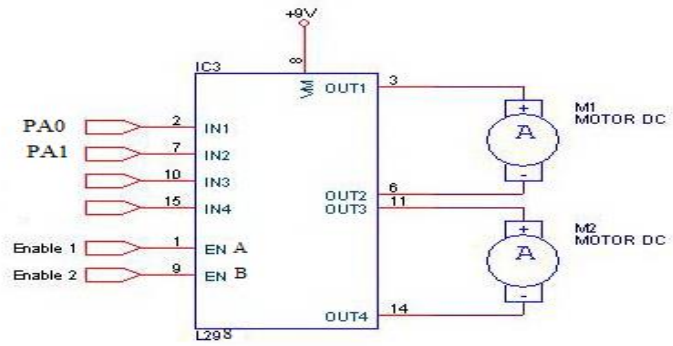
Pada percobaan ke-5

Percobaan 5(a).
Listing Program motor Servo.

```
#include "DrvSYS.h"
#include "DrvGPIO.h"


void delay_loop(void)
{ uint32_t i,j;
  for(i=0;i<4;i++)
    { for(j=0;j<60000;j++);}
}

int main(void)
{int i=0;
UNLOCKREG();
DrvSYS_SetOscCtrl(E_SYS_XTL12M, 1);
DrvSYS_Delay(5000);
DrvSYS_SelectHCLKSource(0);
LOCKREG();
DrvSYS_SetClockDivider(E_SYS_HCLK_DIV, 0);
for(i=12;i<16;i++)
{
DrvGPIO_Open(E_GPC, i, E_IO_OUTPUT);
}
```

```
  while(1)
  {
     DrvGPIO_ClrBit(E_GPC,0x7000);
     delay_loop();
     DrvGPIO_SetPortBits(E_GPC,0x7000);
     delay_loop();
     DrvGPIO_ClrBit(E_GPC,0xB000);
     delay_loop();
     DrvGPIO_SetPortBits(E_GPC,0xB000);
     delay_loop();
     DrvGPIO_ClrBit(E_GPC,0xD000);
     delay_loop();
     DrvGPIO_SetPortBits(E_GPC,0xD000);
     delay_loop();
     DrvGPIO_ClrBit(E_GPC,0xE000);
     delay_loop();
     DrvGPIO_SetPortBits(E_GPC,0xE000);


  }
}
```

Pada percobaan 5(b) menggunakan motor DC



Gambar 8 Driver dan Motor Stepper

Percobaan 5(a).

Listing Program motor DC

```
#include "DrvSYS.h"
#include "DrvGPIO.h"

void delay_loop(void)
{ uint32_t i,j;
  for(i=0;i<4;i++)
```

```
    { for(j=0;j<60000;j++);}
}

int main(void)
{int i=0;
UNLOCKREG();
DrvSYS_SetOscCtrl(E_SYS_XTL12M, 1);
DrvSYS_Delay(5000);
DrvSYS_SelectHCLKSource(0);
LOCKREG();
DrvSYS_SetClockDivider(E_SYS_HCLK_DIV, 0);
for(i=12;i<16;i++)
{
DrvGPIO_Open(E_GPC, i, E_IO_OUTPUT);
}


   while(1)
   {

      DrvGPIO_SetPortBits(E_GPC,0xE000);
      DrvGPIO_ClrBit(E_GPC,0xD000);
      delay_loop();

      DrvGPIO_ClrBit(E_GPC,0xE000);
      DrvGPIO_SetPortBits(E_GPC,0xD000);
      delay_loop();


   }
}
```