



**BUKU PANDUAN PRAKTIKUM**  
**MIKROKONTROLLER**

**LABORATORIUM TEKNIK ELEKTRO**  
**UNIVERSITAS MUHAMMADIYAH MALANG**

**STANDART OPERASIONAL PROSEDUR  
LABORATORIUM TEKNIK ELEKTRO  
UNIVERSITAS MUHAMMADIYAH MALANG**

**A. PRA PRAKTIKUM**

1. Ka Laboratorium bersama Ketua Prodi menetapkan daftar Mata Praktikum yang akan dilaksanakan pada semester berjalan.
2. Laboran atau Staf mengumumkan daftar Mata Praktikum dan pengumuman lainnya via web lab-elektro.umm.ac.id.
3. Staf / Laboran menerima pendaftaran calon praktikan yang mengulang.
4. Staf / Laboran mengumumkan daftar peserta Mata Praktikum berdasarkan data peserta mata kuliah dan peserta mengulang di web lab-elektro.umm.ac.id.
5. Kepala lab dan wakil kepala lab menetapkan daftar Instruktur dan Asisten Mata Praktikum dan diusulkan untuk ditetapkan SK Dekan.
6. Ka. Lab mengundang Peserta Mata Praktikum untuk mengikuti pertemuan persiapan dan pembagian jadwal peserta mengikuti praktikum dan peraturan serta prosedur praktikum dan K3.
7. Instruktur dan Asisten mengundang peserta Mata Praktikum untuk mengikuti Ujian Pra Praktikum (Memberikan Tugas Pra Praktikum).

**B. PRA PELAKSANAAN PERCOBAAN PRAKTIKUM**

1. Asisten dan Praktikan hadir 15 menit sebelum dimulai jam praktikum.
2. Asisten mempersiapkan instrumen ukur serta modul praktikum dan peralatan pendukung seperti kabel, jumper dan lain lain.
3. Praktikan membaca petunjuk praktikum dan mempersiapkan kebutuhan peralatan sebelum masuk ruang/lab.
4. Asisten memberikan salam dan ucapan selamat datang dengan senyum serta memberikan arahan kepada kelompok Praktikan tentang prosedur pelaksanaan praktikum dan penjelasan daftar peralatan dan modul.
5. Asisten menunjuk peserta yang menjadi petugas pencatat, melakukan pengukuran dan pembantu pelaksanaan.
6. Asisten meminta kelompok Praktikum untuk membaca doa/Basmalah sebelum dimulai pemasangan dan instalasi praktikum dan dipandu oleh Asisten.

**C. PRAKTIKUM BERLANGSUNG**

1. Asisten memberikan instruksi kepada kelompok praktikan pemasangan atau instalasi modul dan mengawasi dan mengevaluasi serta memeriksa hasil pemasangan dan memastikan kebenaran instalasi.
2. Praktikan dan asisten saling menjaga kenyamanan dan ketertiban praktikum sesuai tata tertib yang berlaku serta menjaga keamanan perangkat lab selama pelaksanaan praktikum dari satu percobaan ke percobaan berikutnya.
3. Asisten berhak menegur dan menindak praktikan apabila ketahuan merusak, mengubah atau memindahkan perlengkapan lab tanpa ijin.
4. Asisten melakukan penilaian dan pengawasan tiap praktikan melakukan pengukuran selama percobaan.
5. Asisten dan kelompok praktikan mengakhiri praktikum dengan membaca hamdallah dan mengucapkan salam serta meminta praktikan untuk merapikan peralatan dan modul serta kursi dan membuang sampah di sekitarnya.

#### **D. PRAKTIKUM BERAKHIR**

1. Praktikan meninggalkan ruangan dengan rapi dan teratur.
2. Asisten Mengkondisikan ruangan kembali,
  - a. Mengembalikan/mengatur kursi kembali.
  - b. Merapikan sampah yang ditemukan berserakan dalam ruangan.
  - c. Mengembalikan peralatan dan modul ke Lemari Alat dan Modul sesuai nama jenis Mata Praktikum.
  - d. Mengunci pintu.
  - e. Mematikan lampu apabila tidak ada praktikum berikutnya.
3. Asisten menandatangani presensi kelompok dan memberikan daftar penilaian kerja percobaan kelompok ke ruang administrasi (Laboran).
4. Instruktur dan atau asisten melakukan evaluasi reguler praktikum jika diperlukan.

#### **E. PASCA PRAKTIKUM**

1. Praktikan menyusun laporan semua percobaan
2. Praktikan melakukan asistensi laporan ke Asisten Praktikum minimal 4 kali.
3. Setelah laporan praktikum ditandatangani oleh Asisten, Tiap Praktikum menghadap Instruktur sesuai jadwal yang ditetapkan Instruktur.
4. Instruktur menguji praktikum mengenai proses pelaksanaan praktikum.
5. Instruktur memberikan nilai akhir praktikan.
6. Nilai akhir praktikum diserahkan ke Lab untuk proses administrasi.

#### **F. SANKSI**

1. Keterlambatan asistensi pertama kali sanksi point 1
  2. Tidak memenuhi minimal 4 kali asistensi sanksi point 2
  3. Datang terlambat 15 menit dari waktu yang telah ditentukan sanksi point 3
  4. \* Tidak mengikuti proses praktikum tanpa adanya konfirmasi sanksi point 4
  5. \* Tidak mengikuti ujian koordinator tanpa adanya konfirmasi sanksi point 5
  6. Keterlambatan pengumpulan laporan resmi sanksi point 6
  7. \* Tidak mengikuti ujian instruktur sesuai dengan jadwal yang ditentukan instruktur sanksi point 7
  8. Pemalsuan tanda tangan selama proses praktikum berlangsung sanksi point 8
  9. Merusakkan peralatan Lab. Teknik Elektro sanksi point 9
- \* Maksimal konfirmasi 2 x 24 jam sejak jadwal resmi diumumkan untuk penggantian jadwal ujian

Point 1	Menulis materi modul bab 1.
Point 2	Menulis materi modul bab 1-3 & Pengurangan nilai.
Point 3	Menulis materi 1 bab & Pengurangan nilai.
Point 4	Mengulang ( tidak konfirmasi sesuai waktu yang telah ditentukan ) atau Pengurangan Nilai.
Point 5	Mengulang ( tidak konfirmasi sesuai waktu yang telah ditentukan ) atau Pengurangan Nilai.
Point 6	Membeli buku berkaitan dengan bidang Teknik Elektro.
Point 7	Pengurangan Nilai Instruktur.
Point 8	Mengulang Praktikum atau mendapat Nilai E.
Point 9	Mengganti peralatan tersebut sesuai dengan spesifikasi atau mirip dan memiliki fungsi yang sama.

## **G. KESELAMATAN DAN KESEHATAN KERJA (K3)**

1. Sebelum memulai praktikum, praktikan memahami tata tertib dan keselamatan di Laboratorium
2. Mengetahui tempat dan cara penggunaan peralatan Laboratorium
3. Memperhatikan dan waspada terhadap tempat-tempat sumber listrik ( stop kontak dan circuit breaker)
4. Praktikan harus memperhatikan dan menaati peringatan (warning) yang biasa tertera pada badan peralatan praktikum maupun rambu peringatan yang terdapat di ruangan Laboratorium
5. Jika melihat ada kerusakan yang berpotensi menimbulkan bahaya, segera laporkan ke asisten terkait atau dapat langsung melapor ke laboran.
6. Hindari daerah atau benda yang berpotensi menimbulkan bahaya listrik ( sengatan listrik) secara tidak sengaja, missal seperti jala-jala kabel yang terkelupas
7. Keringkan bagian tubuh yang basah, seperti keringat atau sisa air wudhu
8. Selalu waspada terhadap bahaya listrik pada setiap aktifitas praktikum.
9. Jika terjadi kecelakaan akibat bahaya listrik, berikut ini adalah hal-hal yang harus diikuti praktikan:
  - a) Jangan panik
  - b) Matikan semua peralatan elektronik dan sumber listrik di meja masing-masing dan di meja praktikum yang tersengat arus listrik.
  - c) Bantu praktikan yang tersengat arus listrik untuk melepaskan diri dari sumber listrik.
  - d) Beritahukan dan minta bantuan kepada laboran, praktikan lain dan orang di sekitar anda tentang terjadinya kecelakaan akibat bahaya listrik.
  - e) Menjauh dari ruang praktikum.
10. Jangan membawa benda-benda yang mudah terbakar (korek api, gas, dll) ke dalam ruangan laboratorium bila tidak disyaratkan dalam modul praktikum.
11. Jangan melakukan sesuatu yang menimbulkan api, percikan api, atau panas yang berlebihan.
12. Jangan melakukan sesuatu yang menimbulkan bahaya api atau panas berlebih pada diri sendiri atau orang lain.
13. Selalu waspada terhadap bahaya api atau panas berlebih pada setiap aktivitas di laboratorium.
14. Dilarang membawa benda tajam (pisau, gunting dan sejenisnya) ke ruang praktikum bila tidak diperlukan untuk pelaksanaan percobaan
15. Dilarang memakai perhiasan dari logam misalnya cincin, kalung, gelang, dll
16. Hindari daerah, benda atau logam yang memiliki bagian tajam dan dapat melukai.
17. Tidak melakukan sesuatu yang dapat menimbulkan luka pada diri sendiri atau orang lain.

# Pendahuluan

## Konsep I/O pada NUC1xx Series

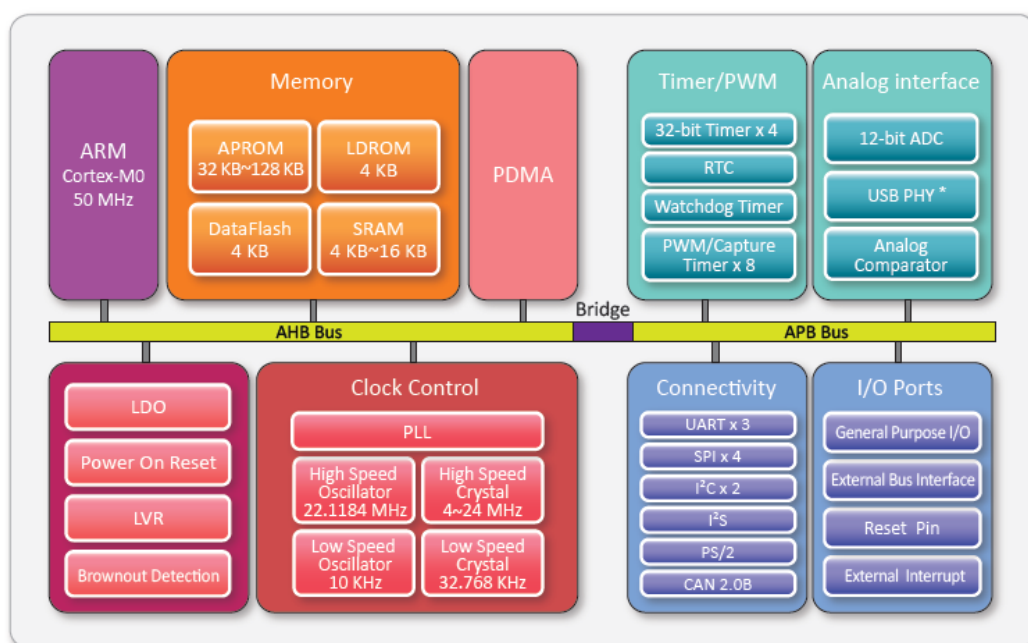
Tujuan:

Memahami konsep Masukan/Keluaran (I/O) pada mikrokontroler ARM Cortex M0 NuMicro 1xx Series (NUC140VE3CN)

Pendahuluan

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 setara dengan mikrokontroler 8 bit.

NuMicro seri NUC1xx memiliki inti ARM Cortex M0 yang tertanam dengan kecepatan hingga 50 MHz, dilengkapi dengan memori flash untuk program 32KB/64KB/128KB, SRAM sebesar 4KB/8KB/16KB dan memori flash loader untuk ISP (In System Programming) sebesar 4KB. Selain itu juga dilengkapi dengan berbagai macam periperhal, 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. Bagan 1 menunjukkan diagram blok dari NuMicro NUC130/140 Series.



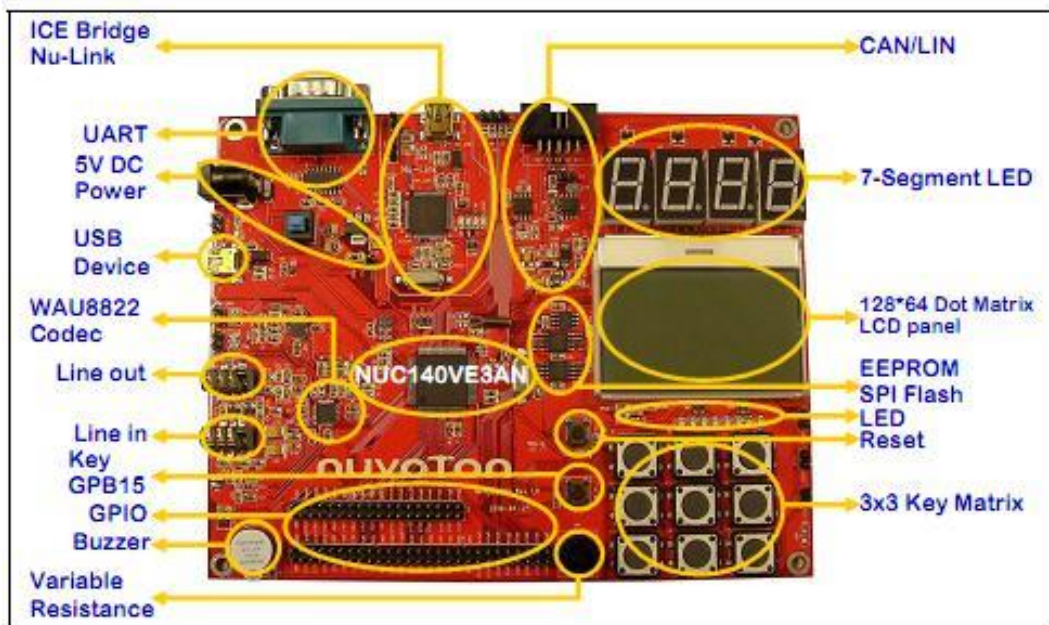
\*USB PHY not offer support for NUC130

Bagan 1 Diagram Blok NuMicro NUC130/140 Series

NuC140 Learning Board

Board ini menggunakan catudaya 5V, yang dapat diperoleh dari konektor USB ataupun melalui konektor catudaya adaptor. Tegangan ini langsung menjadi VDD untuk chip NUC140VE3CN, sehingga perlu diperhatikan tegangan input ini maksimal adalah 5.5V (menurut datasheet NUC140).

Pada board terdapat juga catudaya teregulasi 3.3V menggunakan chip LM1117. Tipe chip regulator ini tidak dinyatakan dalam skematik. Tegangan dari powerjack 3 in dan konektor USB dilewatkan melalui dioda sehingga aman dari kesalahan polaritas pemasangan, namun tidak melindungi dari kerusakan jika tegangan masuk melebihi 5.5V. Bagan 1 menunjukkan layout NUC140 Learning Board. Sedangkan Tabel 1 menunjukkan penggunaan pin pada Learning Board tersebut.



Bagan 2 Layout NUC140 Learning Board dari Nuvoton

Tabel 1 Konfigurasi Sistem NuMicro Ixx Series Learning Board

Blok	Pin	Fungsi
ICE Bridge Nu Link	ICE_CLK ICE_DATA	Antarmuka SWD
UART	GPB0 GPB1	Rx UART0 Tx UART0
Push button GPB15	GPB15	INT0
CAN	GPD6 GPD7	Rx CAN0 Tx CAN0
	GPB12-13	CAN transciever speed
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 Enable/Disable
	LIN	GPB4 GPB5 GPB6 GPB7
7 Segment LED	GPE0-7	Baris
	GPC4-7	Kolom
LCD Panel Dot Matrix	GPD8	SPI3 SS30

	GPD9 GPD10 GPD11	SPI3 SPCLK SPI3 MISO0 SPI3 MOSIO
	GPD14	LCD Backlight
Variable Resistor	GPA7	Antarmuka ADC
Buzzer	GPB11	PWM4
Keypad Matrix	GPA0-5	GPIO
Reset	RESET	Reset
EEPROM	GPA10 GPA11	I2C1 SDA I2C1 SCL
SD Card Slot	GPD12	Catudaya SD Card
	GPD13	Deteksi SD Card
	GPC8-11	Antarmuka SD Card
Flash	GPD0	SPI2 SS20
	GPD1	SPI2 SPCLK
	GPD2	SPI2 MISO0
	GPD3	SPI2 MOSIO
	GPD4	SPI2 MISO1
	GPD5	SPI2 MOSI1
LED	GPA12	PWM0
	GPA13	PWM1
	GPA14	PWM2
	GPC12-15	GPIO

#### Langkah Penggunaan NUVOTON NL-NUC140

1. Buatlah project baru disetiap pergantian dengan memilih MENU PROJECT – NEW PROJECT setelah muncul kotak dialog beri nama project yang akan dibuat dan pilih tempat penyimpanan project dan pilih NEXT.
2. Maka akan muncul kotak dialog selanjutnya dan pilih CHIP – NEXT dan pilih device yang akan digunakan NUVOTON – NUC140VE3CN – FINISH
3. Langkah selanjutnya Pilih REPOSITORY dan centang sesuai kebutuhan dari sistem sesuai pada masing-masing bab percobaan.
4. Pada PROJECT yang telah dibuat terdapat file “ **main.c** “ dimana digunakan untuk menuliskan program yang akan dimasukkan kedalam device.
5. Pilih MENU PROJECT – BUILD untuk *compile* program.
6. Jika *BUILD SUCCESS* maka dapat langsung di downloadkan ke dalam device dengan memilih MENU – FLASH – PROGRAM DOWNLOAD.
7. Tunggu beberapa saat sampai device siap digunakan.



# BAB 1

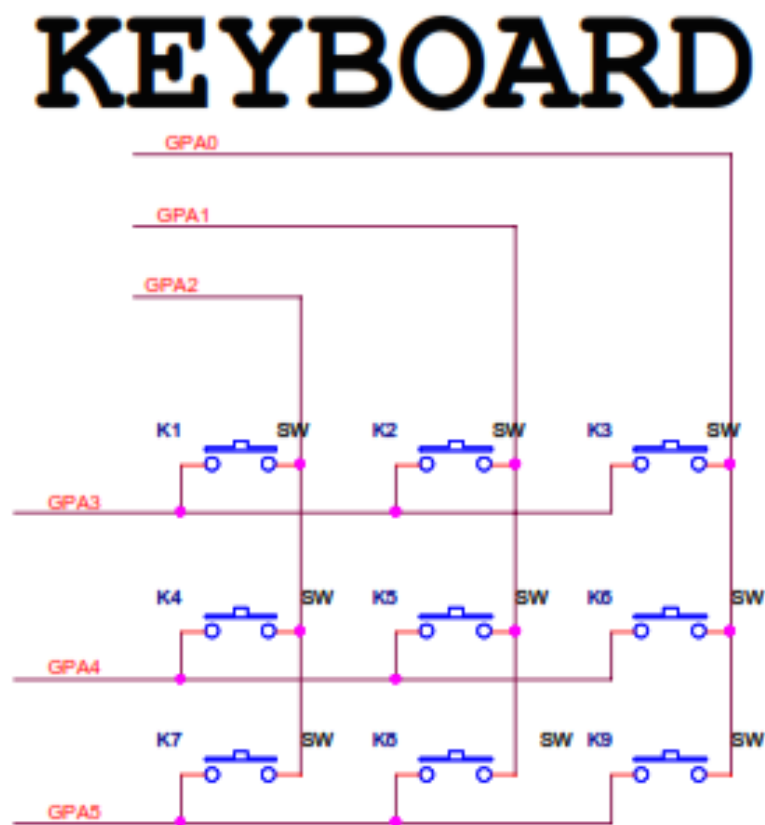
## Keypad Matrix 3x3

### 1.1 Tujuan

Memahami konsep masukan melalui keypad matrix 3x3 dan menampilkannya melalui seven segment.

### 1.2 Pendahuluan

Pada learning board NuMicro 1xx terdapat Keypad Matrix yang terhubung dengan GPIO Port A pin 0 sampai 2 sebagai kolom, dan GPIO Port A pin 3 sampai 5 sebagai baris. Berikut skema dari rangkaian keypad matrix 3x3 pada learning board NuMicro 1xx.

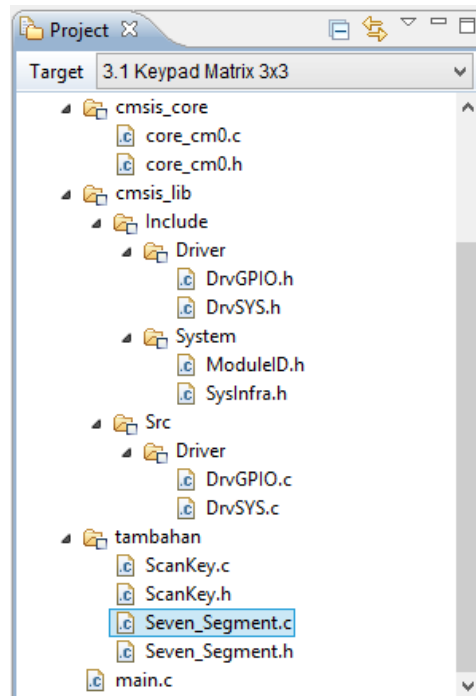


Gambar 1.1 Skema rangkaian keypad matrix 3x3 pada NuMicro 1xx Learning Board

Pengaturan Repository

Step 3 Select Basic Components [ Nuvoton / NUC140VE3CN ]

COMMON			
<input type="checkbox"/>	C Library	Implement the minimal functionality required to allow newlib to link	Available <a href="#">CooCox</a>
<input type="checkbox"/>	Retarget printf	Implementation of printf(), sprintf() to reduce memory footprint	Available <a href="#">CooCox</a>
<input type="checkbox"/>	Semihosting	Implementation of Semihosting GetChar/SendChar	Available <a href="#">CooCox</a>
<input checked="" type="checkbox"/>	M0 Cmsis Core	CMSIS Core For Cortex M0	Available <a href="#">CooCox</a>
BOOT			
<input checked="" type="checkbox"/>	CMSIS Boot	CMSIS Boot for Nuvoton NUC1xx	Available <a href="#">CooCox</a>
PERIPHERAL NUVOTON			
<input checked="" type="checkbox"/>	System Definitions	NUC1xx System Definitions	Available <a href="#">CooCox</a>
<input checked="" type="checkbox"/>	SYS	NUC1xx System Manager and Clock Controller	Available <a href="#">CooCox</a>
<input type="checkbox"/>	UART	NUC1xx Universal Asynchronous Receiver/Transmitter Driver	Available <a href="#">CooCox</a>
<input type="checkbox"/>	TIMER	NUC1xx Timer Controller Driver	Available <a href="#">CooCox</a>
<input checked="" type="checkbox"/>	GPIO	NUC1xx General Purpose I/O Driver	Available <a href="#">CooCox</a>
<input type="checkbox"/>	ADC	NUC1xx Analog-to-Digital Converter Driver	Available <a href="#">CooCox</a>
<input type="checkbox"/>	SPI	NUC1xx Serial Peripheral Interface Driver	Available <a href="#">CooCox</a>
<input type="checkbox"/>	I2C	NUC1xx I2C Serial Interface Driver	Available <a href="#">CooCox</a>



Untuk menggunakan keypad matrix 3x3 pada NuMicro 1xx Learning Board, hal pertama yang harus kita lakukan adalah memasukkan library ScanKey.h dan ScanKey.c. Setelah dilakukan pemanggilan library tersebut, pengguna dapat memanggil fungsi Scankey() saat akan menggunakan keypad matrix tersebut.

Contoh program:

```

_key = Scankey();
if(_key==1)
{
    show_seven_segment(0,_key);
}
close_seven_segment();

```

### 1.3 Keypad + Seven Segment

Program berikut akan menampilkan angka pada seven segment sesuai dengan tombol keypad yang ditekan.

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

int main(void)
{
    UNLOCKREG();
    DrvSYS_SetOscCtrl(E_SYS_XTL12M,1);
    DrvSYS_Delay(5000);
    DrvSYS_SelectHCLKSource(0);
    LOCKREG();
    DrvSYS_SetClockDivider(E_SYS_HCLK_DIV,0);
    OpenKeypad();

    int8_t_keypressed;
    while(1)
    {
        _keypressed = Scankey();
        show_seven_segment(0,_keypressed);
        DrvSYS_Delay(5000);
        close_seven_segment();
    }
}
```

*Bagan 1 source code percobaan 1.3*

### 1.4 Keypad sebagai trigger

Untuk percobaan kali ini kita akan menggunakan keypad sebagai trigger untuk menampilkan 2 program yang berbeda, gambarannya sebagai berikut, ketika kita menekan keypad 1, maka akan menampilkan angka 0 sampai F mulai dari kolom paling kanan menuju kolom paling kiri, dan ketika keypad 2 ditekan akan terjadi sebaliknya.

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

int main(void)
{
    UNLOCKREG();
    DrvSYS_SetOscCtrl(E_SYS_XTL12M,1);
    DrvSYS_Delay(5000);
    DrvSYS_SelectHCLKSource(0);
    LOCKREG();
    DrvSYS_SetClockDivider(E_SYS_HCLK_DIV,0);
```

```

OpenKeyPad();

int8_t _keypad;
while(1)
{
    close_seven_segment();
    _keypad = Scankey();
    if(_keypad==1)
    {
        int i=0, j=0, k=0;
        for(i=0;i<16;i++)
        {
            close_seven_segment();
            show_seven_segment(j,i);
            DrvSYS_Delay(500000);
            if(i>0){
                k=(i+1)%4;
                if(k==0){j++;}
            }
        }
    }
    else if(_keypad==2)
    {
        int i=0, j=3, k=0;
        for(i=0;i<16;i++)
        {
            close_seven_segment();
            show_seven_segment(j,i);
            DrvSYS_Delay(500000);
            if(i>0){
                k=(i+1)%4;
                if(k==0){j--;}
            }
        }
    }
}
}

```

*Bagan 2 source code percobaan 1.4*

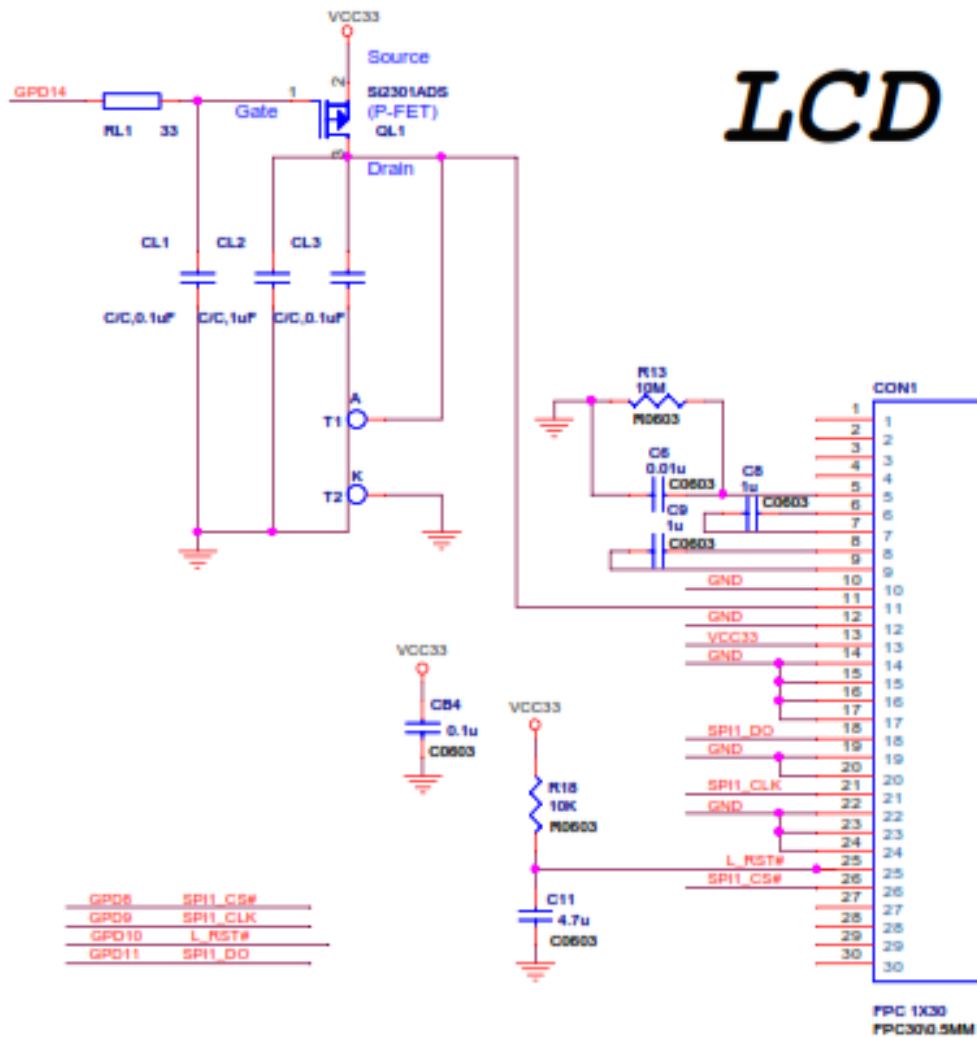
# BAB 2 LCD

## 2.1 Tujuan

Memahami penggunaan LCD pada mikrokontroler ARM Cortex M0 NuMicro 1xx Series, yang dihubungkan dengan masukan dari keypad matrix 3x3.

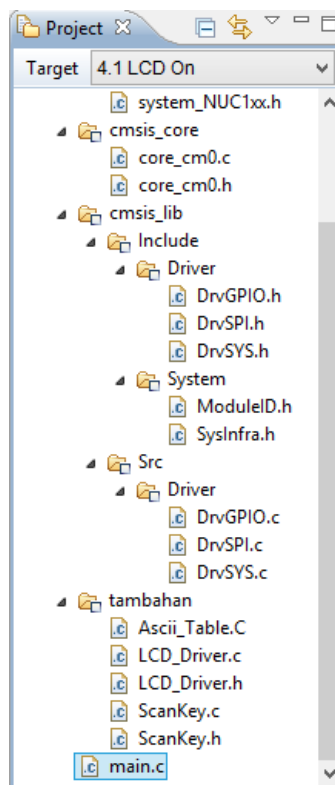
## 2.2 Pendahuluan

Pada learning board NuMicro 1xx series terdapat sebuah dot matrix LCD panel. Berikut adalah skema rangkaian dot matrix LCD yang ada pada learning board NuMicro1xx.



Gambar 2.1 skema rangkaian LCD pada NuMicro 1xx Series Pengaturan Repository

COMMON			
<input type="checkbox"/>	C Library	Implement the minimal functionality required to allow newlib to link	Available <a href="#">CooCox</a>
<input type="checkbox"/>	Retarget printf	Implementation of printf(), sprintf() to reduce memory footprint	Available <a href="#">CooCox</a>
<input type="checkbox"/>	Semihosting	Implementation of Semihosting GetChar/SendChar	Available <a href="#">CooCox</a>
<input checked="" type="checkbox"/>	M0 Cmsis Core	CMSIS Core For Cortex M0	Available <a href="#">CooCox</a>
BOOT			
<input checked="" type="checkbox"/>	CMSIS Boot	CMSIS Boot for Nuvoton NUC1xx	Available <a href="#">CooCox</a>
PERIPHERAL NUVOTON			
<input checked="" type="checkbox"/>	System Definitions	NUC1xx System Definitions	Available <a href="#">CooCox</a>
<input checked="" type="checkbox"/>	SYS	NUC1xx System Manager and Clock Controller	Available <a href="#">CooCox</a>
<input type="checkbox"/>	UART	NUC1xx Universal Asynchronous Receiver/Transmitter Driver	Available <a href="#">CooCox</a>
<input type="checkbox"/>	TIMER	NUC1xx Timer Controller Driver	Available <a href="#">CooCox</a>
<input checked="" type="checkbox"/>	GPIO	NUC1xx General Purpose I/O Driver	Available <a href="#">CooCox</a>
<input type="checkbox"/>	ADC	NUC1xx Analog-to-Digital Converter Driver	Available <a href="#">CooCox</a>
<input checked="" type="checkbox"/>	SPI	NUC1xx Serial Peripheral Interface Driver	Available <a href="#">CooCox</a>
<input type="checkbox"/>	I2C	NUC1xx I2C Serial Interface Driver	Available <a href="#">CooCox</a>



## 2.3 LCD

Untuk menggunakan dot Matrix LCD, library dari LCD perlu dipanggil terlebih dahulu, library yang kita gunakan adalah LCD\_Driver.h. Kemudian tambahkan file LCD\_Driver.h, LCD\_Driver.c dan ASCII\_Table.c pada bagian project.

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

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

```
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_panel();
clr_all_panel();

print_lcd(0, "Praktikum");
print_lcd(1, "Mikrokontroler");
print_lcd(2, "ARM Cortex");
print_lcd(3, "uC M0 32 Bit");

while(1)
{
}
}

```

*Bagan 1 source code percobaan 2.3*

Pada percobaan diatas, dengan menggunakan fungsi print\_lcd(), LCD akan menampilkan suatu kalimat dalam 4 baris dengan layar latar belakang menyala terang.

#### b. LCD dan Keypad

Pada percobaan kali ini kita akan menghubungkan LCD dengan Keypad, sehingga tombol yang ditekan melalui keypad matrix 3x3 akan ditampilkan ke layar dot matrix LCD. Berikut source code programnya, tuliskan dalam main.c.

```

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

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_GPD,14,E_IO_OUTPUT);
    DrvGPIO_ClrBit(E_GPD,14);

    OpenKeyPad();
    Initial_panel();
    clr_all_panel();

    print_lcd(0, "Praktikum uC");
    print_lcd(1, "ARM Cortex");
    print_lcd(2, "M0 32 Bit");
    print_lcd(3, "Key Pressed:");

    while(1)
    {
        char getKeyPressed=Scankey();
    }
}

```

```

        if(!getKeyPressed)
        {
            Show_Word(3, 12, 32);
        }
        else
        {
            Show_Word(3, 12, 48 + getKeyPressed);
        }
    }
}

```

### *Bagan 2 source code percobaan 2.3*

Percobaan diatas akan menampilkan nilai keypad yang ditekan kelayar LCD menggunakan fungsi Show\_Word().

#### c. LCD Draw

Untuk percobaan kali ini kita akan menggunakan fungsi LCD\_Draw(), fungsi ini memanfaatkan \*buffer array yang dibentuk dengan matrik 8x128.

```

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

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_GPD,14,E_IO_OUTPUT);
    DrvGPIO_ClrBit(E_GPD,14);
    Initial_panel();
    clr_all_panel();

    char b[128*8];
    unsigned int i;
    for(i=0;i<=(128*8);i++)
    {
        b[i]=0x0;
    }
    //b[0]=0x18; b[1]=0x24; b[2]=0x42; b[3]=0x81; b[4]=0x81; b[5]=0x42; b[6]=0x24;
b[7]=0x18;
    //b[0]=0x3c; b[1]=0x42; b[2]=0x81; b[3]=0x81; b[4]=0x81; b[5]=0x81; b[6]=0x42;
b[7]=0x3c;
    b[0]=0x3c; b[1]=0x42; b[2]=0x8d; b[3]=0xa1; b[4]=0xa1; b[5]=0x8d; b[6]=0x42;
b[7]=0x3c;
    //b[128*1+63]=0xff;

```



```
//b[128*2+1]=0xff;
draw_LCD(b);

while(1)
{
    //sengaja dikosongkan
}
}
```

*Bagan 3 source code percobaan 2.3*

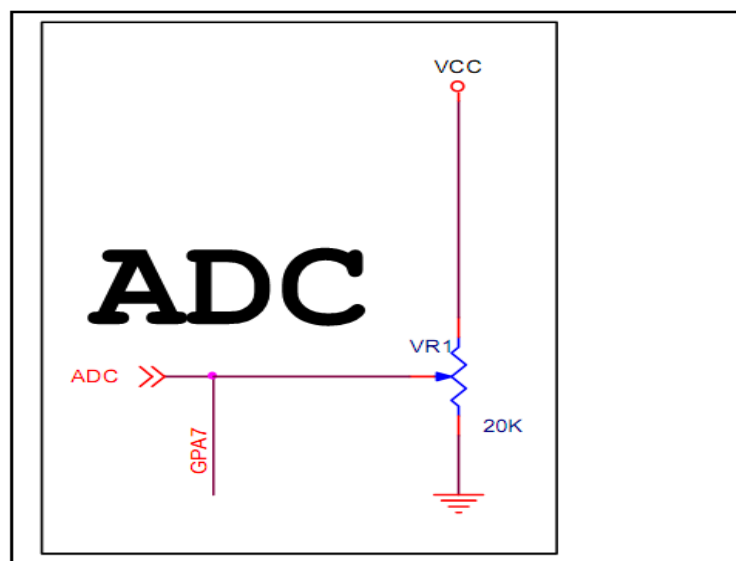
# BAB 3 ADC

## 3.1 Tujuan

Praktikan memahami penggunaan ADC pada NuMicro 1XX Series Development Board dengan menggunakan CoCoox IDE

Pada NuMicro 1XX series terdapat variable resistor yang terhubung dengan port ADC dengan resolusi 12bit pada GPA.7

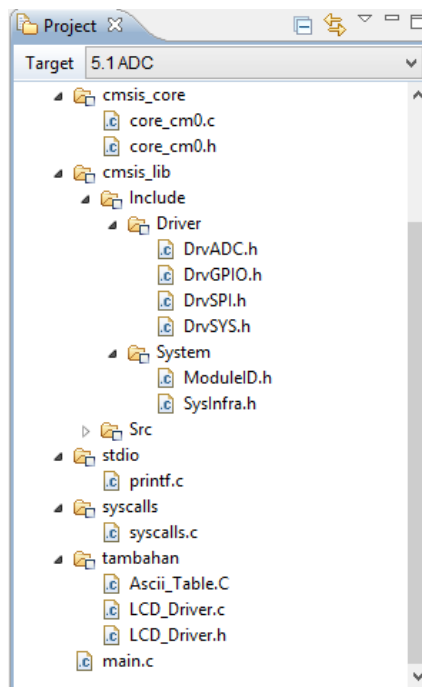
Gambar 5 berikut ini merupakan konfigurasi ADC pada NuMicro 1XX series Development Board



*bagan 1 Konfigurasi ADC pada NuMicro 1XX series Development Board*

## Pengaturan Repository

COMMON			
<input checked="" type="checkbox"/>	C Library	Implement the minimal functionality required to allow newlib to link	Available <a href="#">CooCox</a>
<input checked="" type="checkbox"/>	Retarget printf	Implementation of printf(), sprintf() to reduce memory footprint	Available <a href="#">CooCox</a>
<input type="checkbox"/>	Semihosting	Implementation of Semihosting GetChar/SendChar	Available <a href="#">CooCox</a>
<input checked="" type="checkbox"/>	M0 Cmsis Core	CMSIS Core For Cortex M0	Available <a href="#">CooCox</a>
BOOT			
<input checked="" type="checkbox"/>	CMSIS Boot	CMSIS Boot for Nuvoton NUC1xx	Available <a href="#">CooCox</a>
PERIPHERAL NUVOTON			
<input checked="" type="checkbox"/>	System Definitions	NUC1xx System Definitions	Available <a href="#">CooCox</a>
<input checked="" type="checkbox"/>	SYS	NUC1xx System Manager and Clock Controller	Available <a href="#">CooCox</a>
<input type="checkbox"/>	UART	NUC1xx Universal Asynchronous Receiver/Transmitter Driver	Available <a href="#">CooCox</a>
<input type="checkbox"/>	TIMER	NUC1xx Timer Controller Driver	Available <a href="#">CooCox</a>
<input checked="" type="checkbox"/>	GPIO	NUC1xx General Purpose I/O Driver	Available <a href="#">CooCox</a>
<input checked="" type="checkbox"/>	ADC	NUC1xx Analog-to-Digital Converter Driver	Available <a href="#">CooCox</a>
<input checked="" type="checkbox"/>	SPI	NUC1xx Serial Peripheral Interface Driver	Available <a href="#">CooCox</a>
<input type="checkbox"/>	I2C	NUC1xx I2C Serial Interface Driver	Available <a href="#">CooCox</a>



Berikut program yang digunakan untuk menginisialisasi fungsi ADC pada NuMicro 1XX series:

```
#include "stdio.h"
#include "DrvADC.h"
#include "DrvSYS.h"
#include "DrvGPIO.h"
#include "LCD_Driver.h"

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

void AdcIntCallback(uint32_t u32UserData)
{
    u16ConversionData = DrvADC_GetConversionData(7);

    sprintf(temp, "ADC7 : %d", u16ConversionData);

    clr_all_panel();

    print_lcd(0, temp);

    DrvSYS_Delay(500000);
}

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_panel();

clr_all_panel();

DrvGPIO_Open(E_GPA,7,E_IO_INPUT);

DrvGPIO_Open(E_GPD,14,E_IO_OUTPUT);

DrvGPIO_ClrBit(E_GPD,14);

DrvADC_Open(ADC_SINGLE_END, ADC_CONTINUOUS_OP, 0x80,
EXTERNAL_12MHZ, 5);

DrvADC_ConfigADCChannel7(EXTERNAL_INPUT_SIGNAL);

DrvADC_StartConvert();

DrvADC_EnableADCInt(AdcIntCallback, 0);

while(1)
{
}
}
```

## BAB IV

### LCD

#### 4.1 Tujuan

Mengetahui cara menggunakan LCD 16x2 pada mikrokontroler STM32F1

#### 4.2 Dasar Teori

Teknologi Display LCD ini memungkinkan produk-produk elektronik dibuat menjadi jauh lebih tipis jika dibanding dengan teknologi Tabung Sinar Katoda (Cathode Ray Tube atau CRT). Jika dibandingkan dengan teknologi CRT, LCD juga jauh lebih hemat dalam mengonsumsi daya karena LCD bekerja berdasarkan prinsip pemblokiran cahaya sedangkan CRT berdasarkan prinsip pemancaran cahaya. Namun LCD membutuhkan lampu backlight (cahaya latar belakang) sebagai cahaya pendukung karena LCD sendiri tidak memancarkan cahaya. Beberapa jenis backlight yang umum digunakan untuk LCD diantaranya adalah backlight CCFL (Cold cathode fluorescent lamps) dan backlight LED (Light-emitting diodes).



Gambar 4.1 LCD 16x2

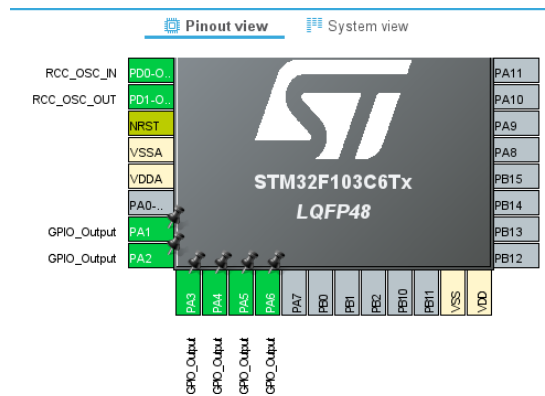
LCD (Liquid Cristal Display) berfungsi untuk menampilkan karakter angka, huruf ataupun simbol dengan lebih baik dan dengan konsumsi arus yang rendah. LCD (Liquid Cristal Display) dot matrik M1632 merupakan modul LCD buatan hitachi. Modul LCD (Liquid Cristal Display) dot matrik M1632 terdiri dari bagian penampil karakter (LCD) yang berfungsi menampilkan karakter dan bagian sistem prosesor LCD dalam bentuk modul dengan mikrokontroler yang diletakkan dibagian belakan LCD tersebut yang berfungsi untuk mengatur tampilan LCD serta mengatur komunikasi antara LCD dengan mikrokontroler yang menggunakan modul LCD tersebut. LCD M1632 merupakan modul LCD dengan tampilan 2×16 (2 baris x 16 kolom) dengan konsumsi daya rendah

#### 4.3 Langkah Percobaan

##### 4.3.1. Langkah Percobaan STM32CUBE IDE

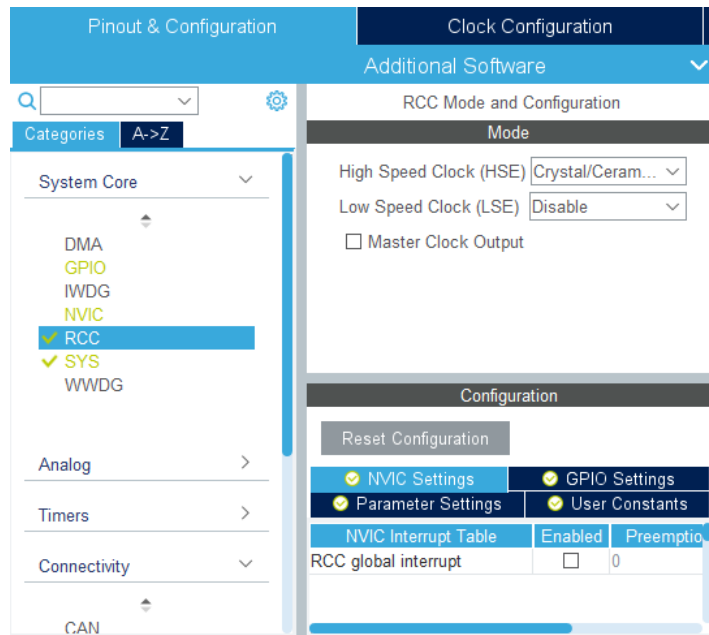
- Membuat project baru menggunakan software STM32CubeIDE
- Memakai Device STM32F103C6

- Mengatur PortA sesuai dengan gambar dibawah ini



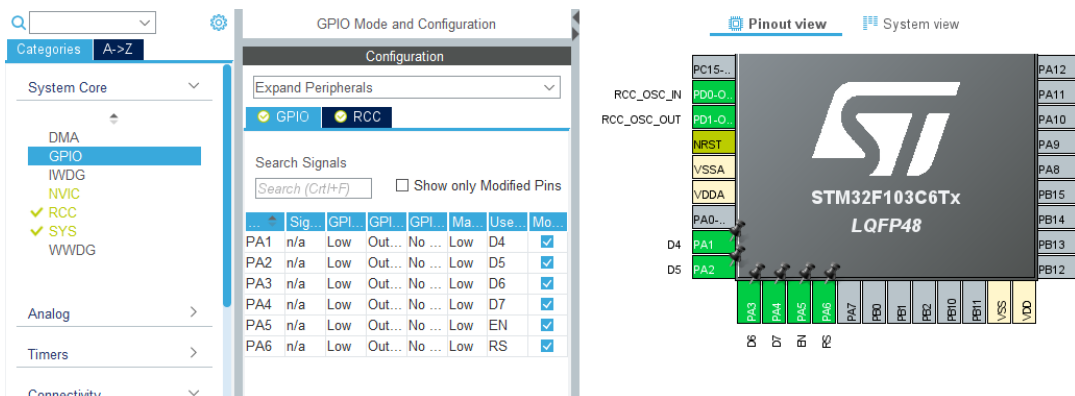
Gambar 4.2 GPIO\_OUTPUT

- ✓ Atur RCC – Crystal/Ceramic



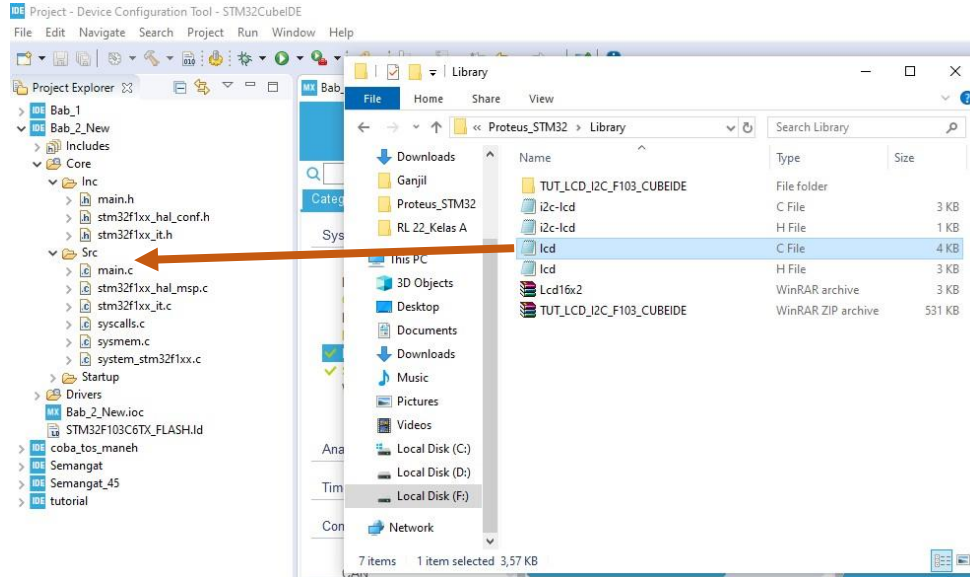
Gambar 4.3 RCC Mode And Configuration

- ✓ Pilih GPIO dan beri keterangan pada User Label “D4” pada PA1 yang telah dipilih, dan disesuaikan dengan gambar dibawah ini.



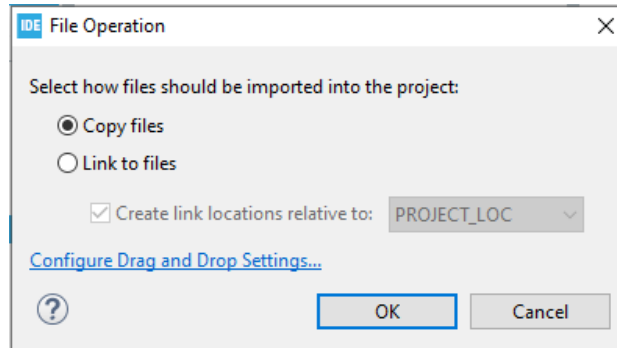
Gambar 4.4 User Label GPIO

- ✓ Setelah selesai pilih generate code sesuai materi dipendahuluan
- ✓ Selanjutnya masukkan library tambah “LCD” dengan cara drag and drop file ke dalam folder yang dibutuhkan sesuai dengan extension file.



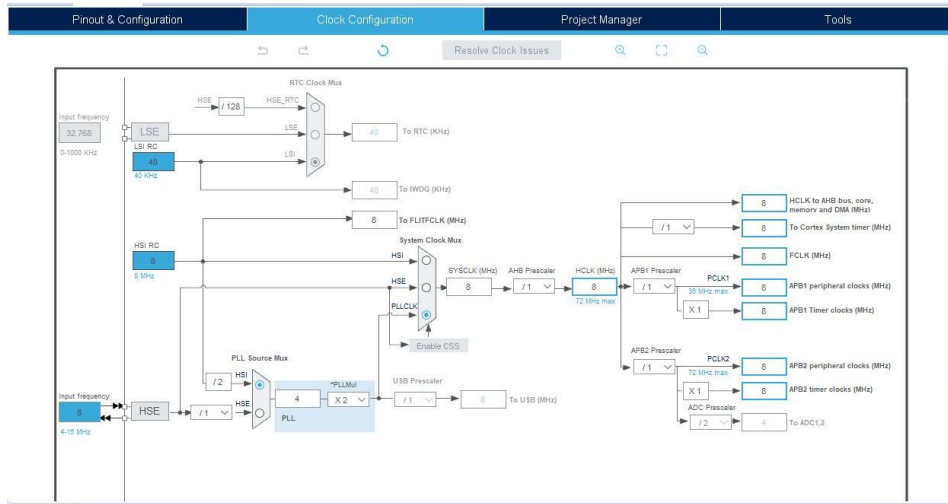
Gambar 4.5 Library LCD

- ✓ Pilih “OK”



Gambar 4.6 File Operation

- ✓ Ulangi Langkah tersebut untuk memasukkan library LCD dengan extension .h dengan cara yang sama seperti sebelumnya.
- ✓ Kemudian pilih menu Clock Configuration dan sesuai kan Item Clock Configuration seperti gamabr di bawah ini :



Gambar 4.7 Clock Configuration

- ✓ Masuk kedalam Project explorer dan pilih “Core – src – main.c” dan tulis program seperti gambar dibawah ini.

```

main.c | lcd.c | Bab_3_LCD.ioc
88
89 /* Initialize all configured peripherals */
90 MX_GPIO_Init();
91 /* USER CODE BEGIN 2 */
92 Lcd_PortType ports[] = {
93     D4_GPIO_Port, D5_GPIO_Port, D6_GPIO_Port, D7_GPIO_Port
94 };
95
96 Lcd_PinType pins[] = {
97     D4_Pin, D5_Pin, D6_Pin, D7_Pin
98 };
99
100 Lcd_HandleTypeDef lcd = Lcd_create(ports, pins, RS_GPIO_Port, RS_Pin, EN_GPIO_Port, EN_Pin, LCD_4_BIT_MODE);
101 /* USER CODE END 2 */
102
103 /* Infinite loop */
104 /* USER CODE BEGIN WHILE */
105 while (1)
106 {
107     /* USER CODE END WHILE */
108     Lcd_cursor(&lcd, 0,0);
109     Lcd_string(&lcd, "Robotikid");
110     Lcd_clear(&lcd);
111
112
113     Lcd_cursor(&lcd, 0,0);
114     Lcd_int(&lcd, 100);
115     HAL_Delay(50);
116 /* USER CODE BEGIN 3 */
117 }
118 /* USER CODE END 3 */
119 }

```

Gambar 4.8 Source Code Bab 3

- ✓ Lakukan proses build, pilih menu “Project – Build Project”
- ✓ Perhatikan hasil build (error, warning dan file .hex)

```

Console
DT Build Console [Bab_3_LCD]
7576 120 1584 9280 2440 Bab_3_LCD.e1f
Finished building: default.size.stdout
Finished building: Bab_3_LCD.bin

Finished building: Bab_3_LCD.list
Finished building: Bab_3_LCD.hex

11:09:12 Build Finished. 0 errors, 0 warnings. (took 6s.681ms)

```

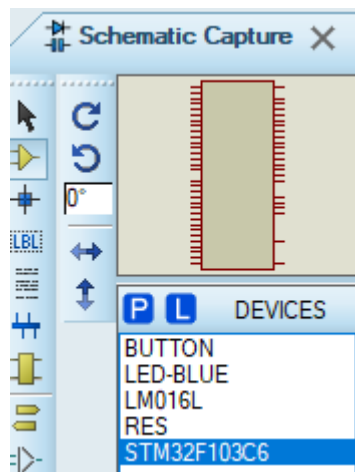
Gambar 4.9 Console Bab 3



- ✓ Abaikan jika terdapat warning.

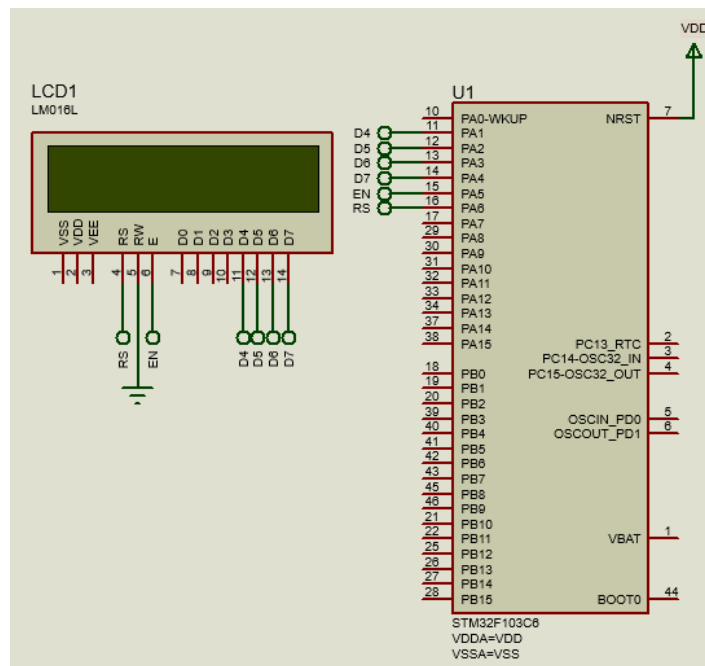
#### 4.3.2 Langkah Percobaan Proteus

- ✓ Membuat project baru dengan nama “Bab 3” sesuai dengan materi pendahuluan di atas
- ✓ Setelah selesai membuat proct baru dan berada pada tampilan simulasi silanjutnya silahkan memilih Component Mode – Pick From Library
- ✓ Maka akan muncul seperti gambar silahkan isi keywords dengan mengetik “LM016L” lalu “Double Click”.



Gambar 4.10 Component list

- ✓ Setelah itu silahkan ditambahkan sesuai dengan komponen list yang diinginkan.
- ✓ Jika benar maka akan muncul komponen pada component list
- ✓ Selanjutnya buat lah rangkaian seperti gambar di bawah ini



Gambar 4.11 Rangkaian Bab 3

- ✓ Setelah selesai merangkai sesuai gambar diatas dapat langsung menjalankan proses simulasi untuk mengetahui hasilnya dengan cara pilih tombol “Run the Simulation”



terletak kiri bawah.

- ✓ Untuk mengganti nilai dari komponen dapat dengan meng-klik 2 pada gambar komponen tersebut.

#### 4.4 Analisa Percobaan

#### 4.5 Kesimpulan

## BAB V

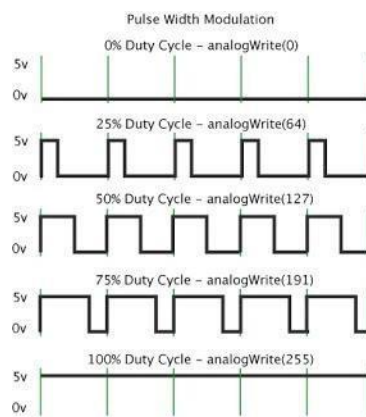
### PWM DENGAN INTERNAL TIMER

#### 5.1 Tujuan

Tujuan dari percobaan ini adalah untuk menguji pembangkit PWM dengan internal timer pada STM32.

#### 5.2 Dasar Teori

PWM adalah kepanjangan dari Pulse Width Modulation atau dalam bahasa Indonesia dapat diterjemahkan menjadi Modulasi Lebar Pulsa. Jadi pada dasarnya, PWM adalah suatu teknik modulasi yang mengubah lebar pulsa (pulse width) dengan nilai frekuensi dan amplitudo yang tetap. PWM dapat dianggap sebagai kebalikan dari ADC (Analog to Digital Converter) yang mengkonversi sinyal Analog ke Digital, PWM atau Pulse Width Modulation ini digunakan menghasilkan sinyal analog dari perangkat Digital (contohnya dari Mikrokontroler).



Gambar 5.1 Duty Cycle

Persentase waktu di mana sinyal PWM tetap pada kondisi TINGGI (ON Time) disebut dengan “siklus kerja” atau “*Duty Cycle*”. Kondisi yang sinyalnya selalu dalam kondisi ON disebut sebagai 100% *Duty Cycle* (Siklus Kerja 100%), sedangkan kondisi yang sinyalnya selalu dalam kondisi OFF (mati) disebut dengan 0% *Duty Cycle* (Siklus Kerja 0%).

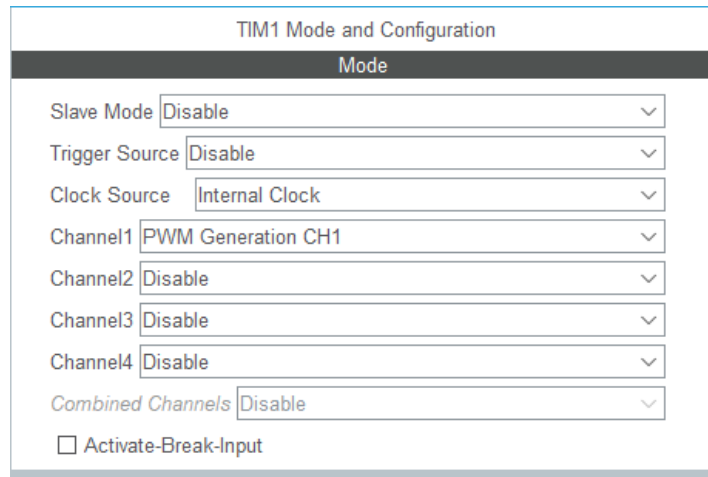
#### 5.3 Langkah Percobaan

##### 5.3.1 Langkah Percobaan STM32CUBE IDE

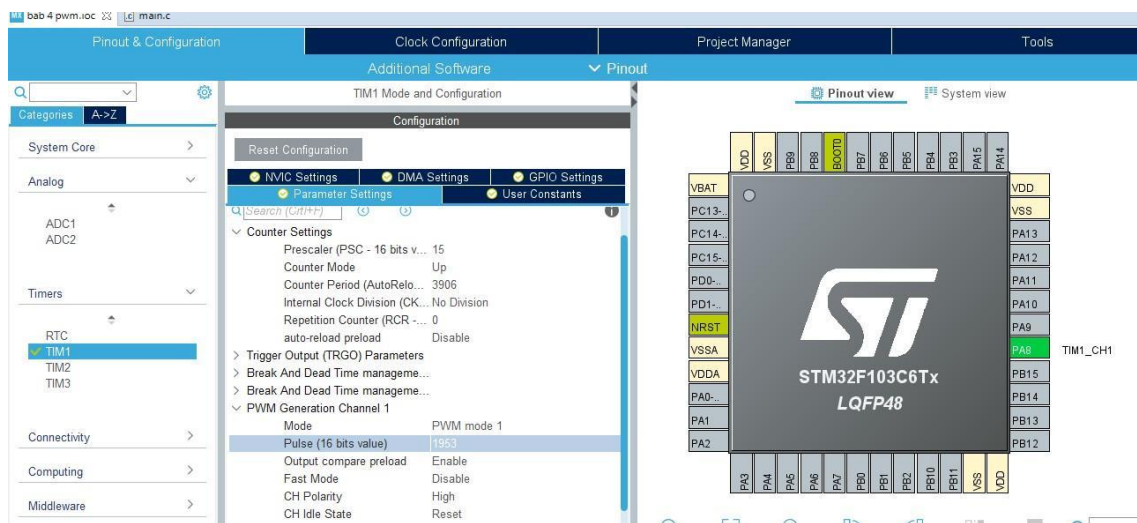
###### 5.3.1.1 Membuat project baru menggunakan software STM32CubeIDE

###### 5.3.1.2 Memakai Device STM32F103C6

###### 5.3.1.3 Pilih kebutuhan internal TIMER dengan memilih TIM1 dan mengatur Mode serta Parameters Settings seperti gambar dibawah ini.

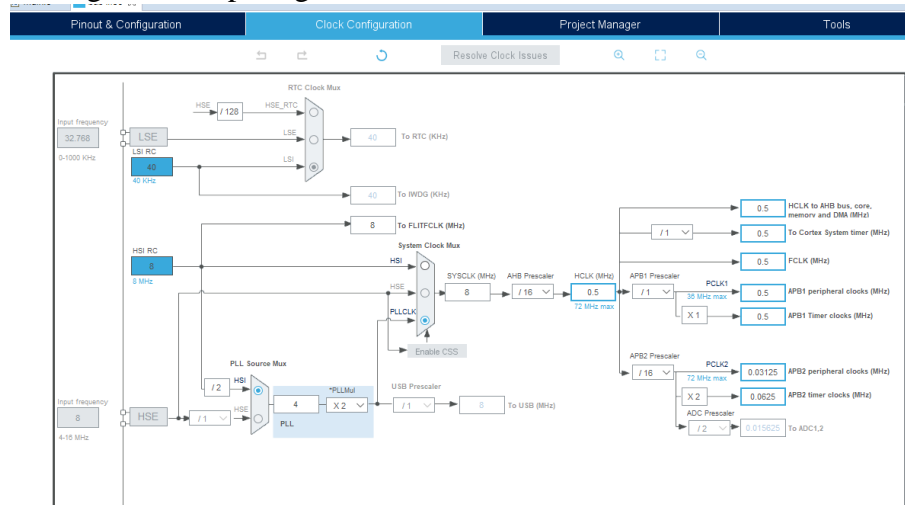


Gambar 5.2 TIM1 Mode and Configuration



Gambar 5.3 Parameter Setting TIM1

5.3.1.4 Kemudian pilih menu Clock Configuration dan sesuai kan Item Clock Configuration seperti gamabr di bawah ini :



Gambar 5.4 Clock Configuration

5.3.1.5 Setelah selesai pilih generate code sesuai materi dipendahuluan

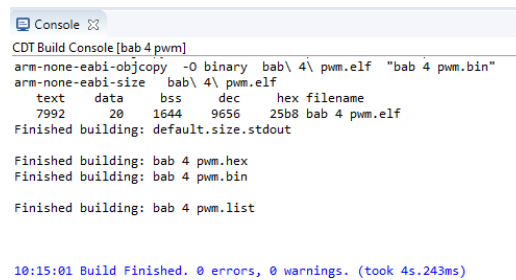
5.3.1.6 Masuk kedalam Project explorer dan pilih “Core – src – main.c” dan tulis program seperti gambar dibawah ini.

```
/* USER CODE END Init */
/* Configure the system clock */
SystemClock_Config();
/* USER CODE BEGIN SysInit */
/* USER CODE END SysInit */
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_TIM1_Init();
/* USER CODE BEGIN 2 */
HAL_TIM_Base_Start(&htim1);
HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
/* USER CODE END 2 */
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
/* USER CODE END WHILE */
/* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
}
```

Gambar 5.5 Source Code Bab 4

5.3.1.7 Lakukan proses build, pilih menu “Project – Build Project”

5.3.1.8 Perhatikan hasil build (error, warning dan file .hex)



```
CDT Build Console [bab 4 pwm]
arm-none-eabi-objcopy -O binary bab\ 4\ pwm.elf "bab 4 pwm.bin"
arm-none-eabi-size bab\ 4\ pwm.elf
text data bss dec hex filename
7992 20 1644 9656 2508 bab 4 pwm.elf
Finished building: default.size.stdout
Finished building: bab 4 pwm.hex
Finished building: bab 4 pwm.bin
Finished building: bab 4 pwm.list
10:15:01 Build Finished. 0 errors, 0 warnings. (took 4s.243ms)
```

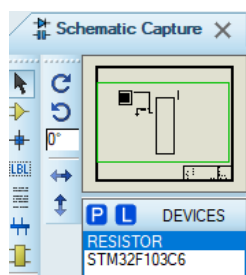
Gambar 5.6 Console Bab 4

5.3.2 Langkah Percobaan Proteus

5.3.2.1 Membuat project baru dengan nama “Bab 4” sesuai dengan materi pendahuluan di atas

5.3.2.2 Setelah selesai membuat proct baru dan berada pada tampilan simulasi silanjutnya silahkan memilih Component Mode – Pick From Library

5.3.2.3 Maka akan muncul seperti gambar silahkan isi keywords dengan mengetik “RESISTOR” lalu “Double Click”.

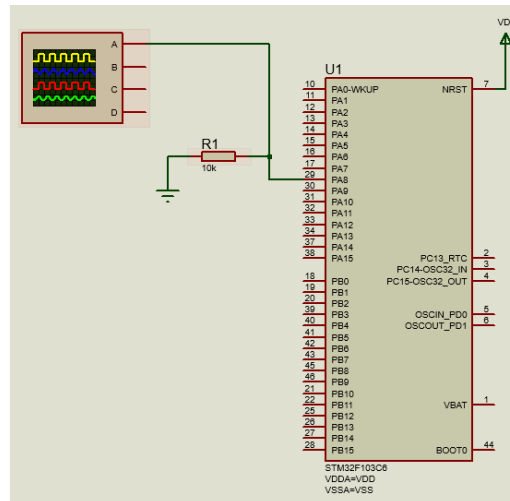


Gambar 5.7 Component list

5.3.2.4 Setelah itu silahkan ditambahkan sesuai dengan komponen list yang diinginkan.

5.3.2.5 Jika benar maka akan muncul komponen pada component list.

5.3.2.6 Selanjutnya buat lah rangkaian seperti gambar di bawah ini



Gambar 5.8 rangkain Bab 4

5.3.2.7 Untuk penambahan alat ukur OSCILOSCOPE berada dalam “virtual instrument mode”

5.3.2.8 Setelah selesai merangkai sesuai gambar diatas dapat langsung menjalankan proses simulasi untuk mengetahui hasilnya dengan cara pilih tombol “Run the Simulation”



- ✓ Untuk mengganti nilai dari komponen dapat dengan meng-klik 2 pada gambar komponen tersebut.

5.4 Analisa Percobaan

5.5 Kesimpulan

## BAB VI

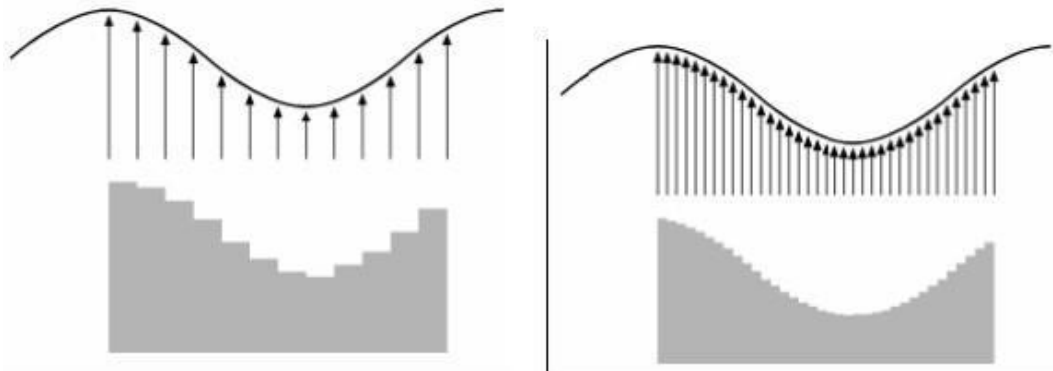
### ANALOG TO DIGITAL CONVERTER

#### 6.1 Tujuan

Tujuan dari percobaan ini adalah untuk menguji nilai Bit ADC pada STM32

#### 6.2 Dasar Teori

Analog To Digital Converter (ADC) adalah pengubah input analog menjadi kode – kode digital. ADC banyak digunakan sebagai Pengatur proses industri, komunikasi digital dan rangkaian pengukuran/ pengujian. Umumnya ADC digunakan sebagai perantara antara sensor yang kebanyakan analog dengan sistim komputer seperti sensor suhu, cahaya, tekanan/ berat, aliran dan sebagainya kemudian diukur dengan menggunakan sistim digital (komputer). ADC (Analog to Digital Converter) memiliki 2 karakter prinsip, yaitu kecepatan sampling dan resolusi. Kecepatan sampling suatu ADC menyatakan seberapa sering sinyal analog dikonversikan ke bentuk sinyal digital pada selang waktu tertentu. Kecepatan sampling biasanya dinyatakan dalam sample per second (SPS).



Gambar 6.1 ADC

Dengan kecepatan Sampling Rendah dan kecepatan Sampling tinggi Resolusi ADC menentukan ketelitian nilai hasil konversi ADC. Sebagai contoh: ADC 8 bit akan memiliki output 8 bit data digital, ini berarti sinyal input dapat dinyatakan dalam 255 ( $2^n - 1$ ) nilai diskrit. ADC 12 bit memiliki 12 bit output data digital, ini berarti sinyal input dapat dinyatakan dalam 4096 nilai diskrit. Dari contoh diatas ADC 12 bit akan memberikan ketelitian nilai hasil konversi yang jauh lebih baik daripada ADC 8 bit.

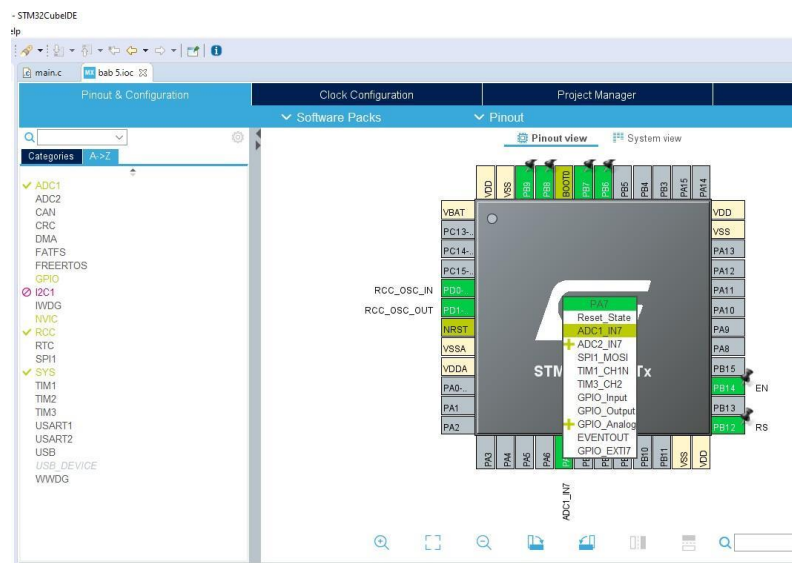
#### 6.3 Langkah Percobaan

##### 6.3.1 Langkah Percobaan STM32CUBE IDE

6.3.1.1 Membuat project baru menggunakan software STM32CubeIDE

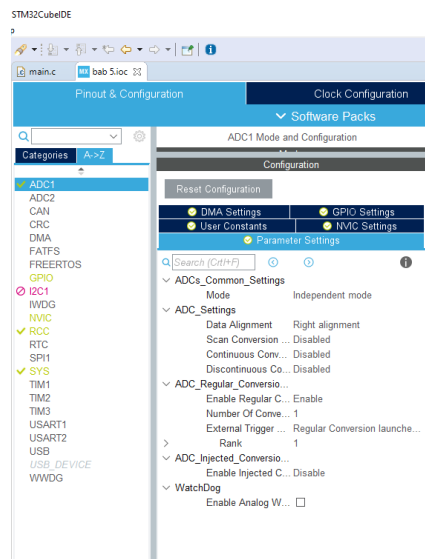
6.3.1.2 Memakai Device STM32F103C6

### 6.3.1.3 Mengatur PA7 sebagai ADC1\_IN7



Gambar 6.2 ADC1 Mode and Configuration

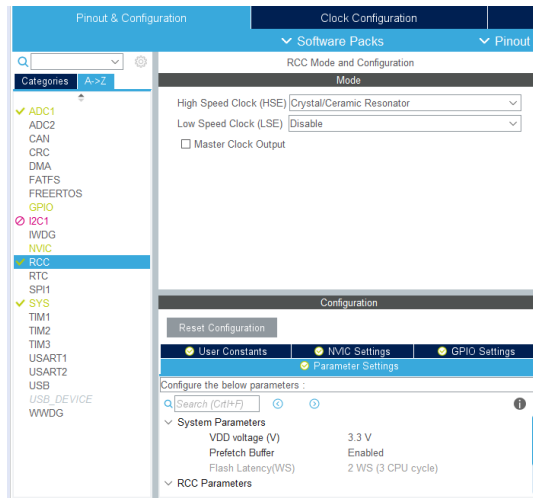
6.3.1.4 Sesuai kan Item pada Configuration pada ADC1\_IN7 seperti pada gambar di bawah ini:



Gambar 6.3 Parameter Setting ADC1

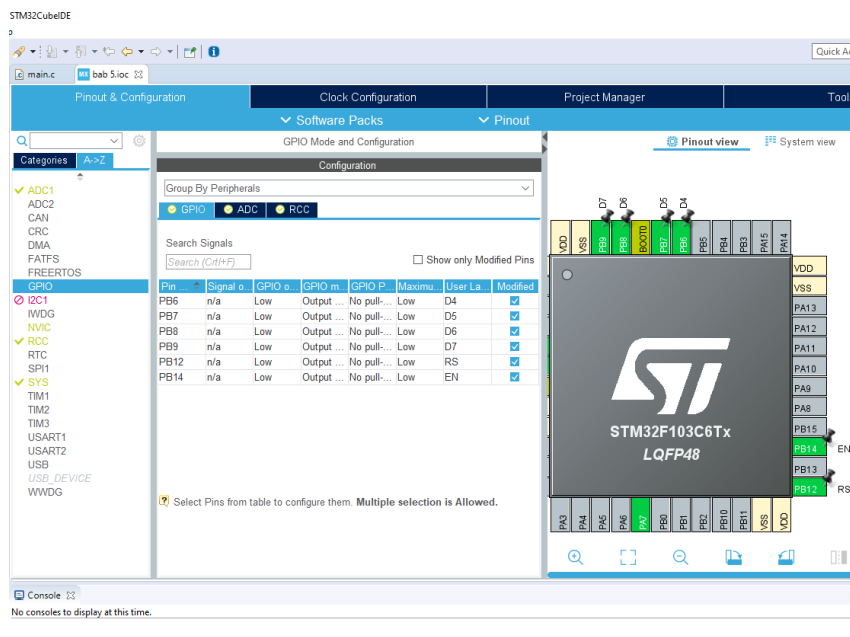


6.3.1.5 Pilih menu RCC,dan ubah pada HSE dari Disable ke rystal/Ceramic Res, seperti pada gambar di bawah ini:



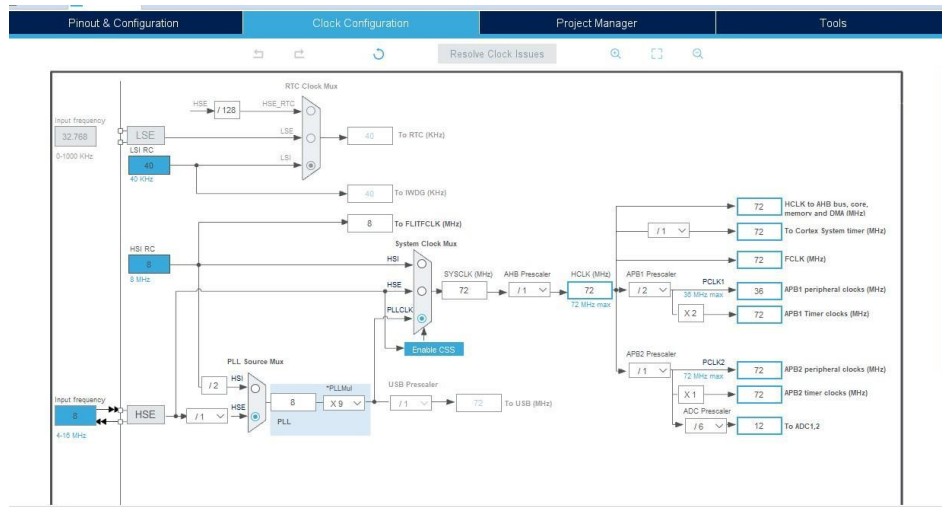
Gambar 6.4 RCC Mode and Configuration

6.3.1.6 Atur PB6 PB7, PB8, PB9 PB12 PB14 Menjadi GPIO\_OUTPUT dan beri label pada pin tersebut sesuai gambar di bawah ini :



Gambar 6.5 User Label GPIO

6.3.1.7 Kemudian pilih menu Clock Configuration dan sesuai kan Item Clock Configuration seperti gamabr di bawah ini :



Gambar 6.6 Clock Configuration

6.3.1.8 Setelah selesai pilih generate code sesuai materi dipendahuluan

6.3.1.9 Masuk kedalam Project explorer dan pilih “Core – src – main.c” dan tulis program seperti gambar dibawah ini dan jangan lupa masukan library LCD seperti bab 3:

```

21 #include "main.h"
22 #include "lcd.h"
23 #include "stdio.h"
24
25 ADC_HandleTypeDef hadc1; //
26 Lcd_HandleTypeDef lcd;
27
28
29
30
31
32 int main(void)
33 {
34
35     uint16_t adc_value = 0;
36
37     HAL_Init();
38
39
40     SystemClock_Config();
41     MX_GPIO_Init();
42     Lcd_PortType port []=
43     {
44         D4_GPIO_Port,D5_GPIO_Port,D6_GPIO_Port,D7_GPIO_Port
45     };
46     Lcd_PinType pin[] =
47     {
48         D4_Pin,D5_Pin,D6_Pin,D7_Pin,
49     };
50
51     lcd = Lcd_create(port, pin, RS_GPIO_Port, RS_Pin, EN_GPIO_Port, EN_Pin, LCD_4_BIT_MODE);
52
53     MX_ADC1_Init();
54     HAL_ADCEx_Calibration_Start(&hadc1);
55     while (1)
56     {
57         HAL_ADC_Start(&hadc1);
58         adc_value = HAL_ADC_GetValue(&hadc1);
59         HAL_Delay(1);
60         Lcd_cursor (&lcd,0,0);
61         Lcd_string(&lcd, "Value:");
62         Lcd_cursor(&lcd, 0, 0);
63         Lcd_int(&lcd, adc_value);
64         HAL_Delay(10);
65     }
66
67 }
68

```

Gambar 6.7 Source Code Bab 5

6.3.1.10 Lakukan proses build, pilih menu “Project – Build Project”

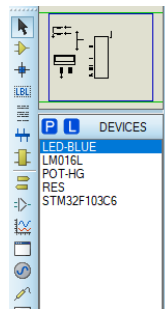
6.3.1.11 Perhatikan hasil build (error, warning dan file .hex)

### 6.3.2 Langkah Percobaan Proteus

6.3.2.1 Membuat project baru dengan nama “Bab 5” sesuai dengan materi pendahuluan di atas

6.3.2.2 Setelah selesai membuat project baru dan berada pada tampilan simulasi silanjutnya silahkan memilih Component Mode – Pick From Library

6.3.2.3 Maka akan muncul seperti gambar silahkan isi keywords dengan mengetik “POT\_HG”,”RES”,”LM016L” lalu “Double Click” setiap pemilihan item.

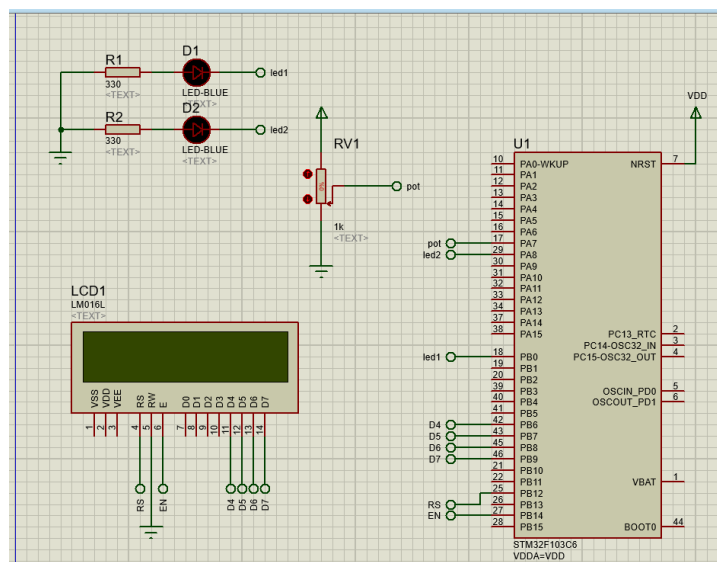


Gambar 6.8 Component list

6.3.2.4 Setelah itu silahkan ditambahkan sesuai dengan komponen list yang diinginkan.

6.3.2.5 Jika benar maka akan muncul komponen pada component list

6.3.2.6 Selanjutnya buat lah rangkaian seperti gambar di bawah ini



Gambar 6.9 Rangkaian Bab 5

6.3.2.7 Untuk penambahan alat ukur voltmeter maupun amperemeter berada dalam “virtual instrument mode” pilih DC Voltmeter

6.3.2.8 Setelah selesai merangkai sesuai gambar diatas dapat langsung menjalankan proses simulasi untuk mengetahui hasilnya dengan cara pilih tombol “Run the Simulation”



terletak kiri bawah.

- ✓ Untuk mengganti nilai dari komponen dapat dengan meng-klik 2 pada gambar komponen tersebut.

6.4 Analisa Percobaan

6.5 Kesimpulan