1. PENDAHULUAN

Modul ini dibuat untuk membantu memahami beberapa fungsi dasar pada mikrokontroler berbasis STM32. Fungsi – fungsi dasar tersebut diantaranya manipulasi GPIO, membaca input, komunikasi serial, komunikasi I2C, membuat sinyal PWM dan input capture.

Dalam mermbuat modul ini penulis menggunakan serangkaian hardware dan software sebagai berikut :

- Blue Pill STM32F103C8T6 development board
- STM32CubeMX version 4.25.1
- Atollic TrueSTUDIO for ARM v8.1.0

Meskipun dalam modul ini digunakan mikrokontroler STM32F103C8T6 namun tutorial ini dapat diaplikasikan pada mikrokontroler berbasis STM32Fx lainnya dengan sedikit penyesuaian.

Dengan adanya modul ini diharapkan pembaca dapat memahami workflow dan beberapa fungsi dasar mikrokontroler berbasis STM32. Source code program yang digunakan dalam modul ini dapat diakses di; https://bitbucket.org/Abizhar/set_stm32/src.

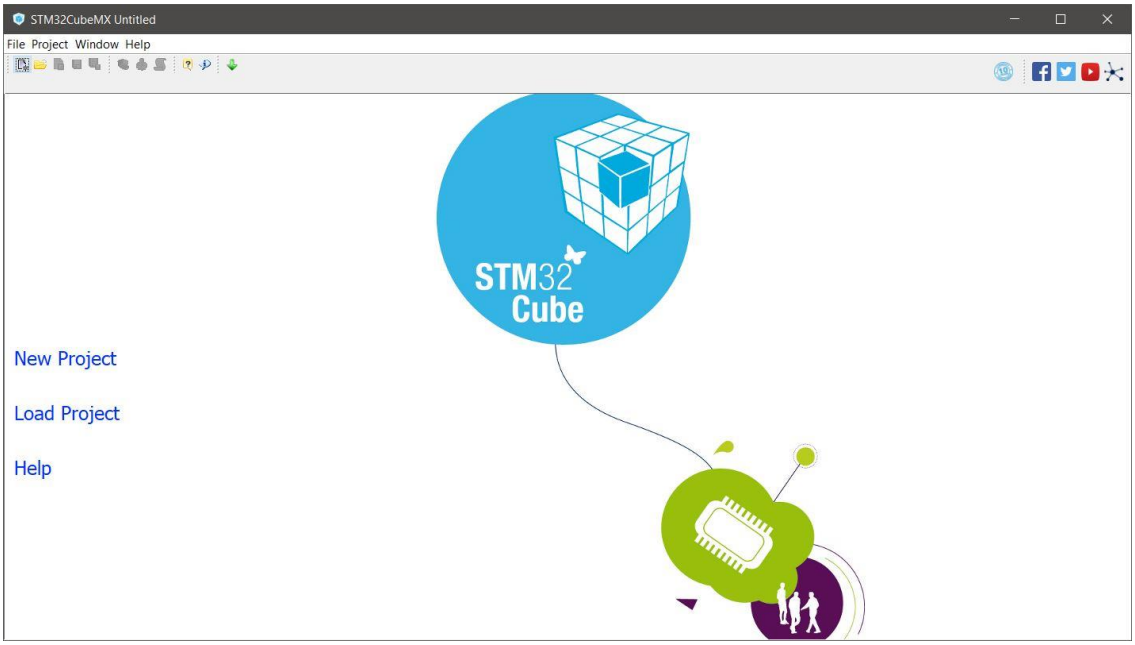
2. PROJECT HELLO_WORLD

Project pertama ini akan mendemonstrasikan cara membuat project baru pada STM32CubeMX dan menuliskan program pada Atollic TrueSTUDIO untuk memanipulasi dan membaca GPIO pada mikrokontroler STM32.

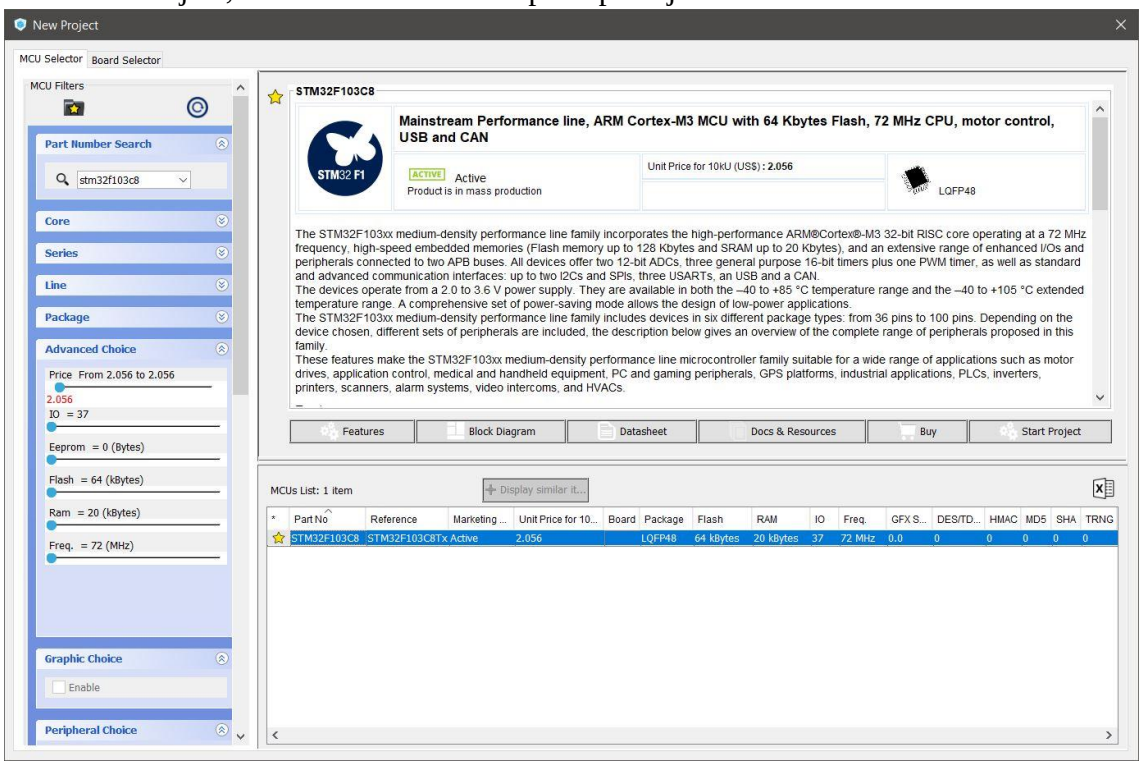
Sebelum memulai project ini, pembaca diharapkan sudah memahami beberapa konsep dasar berikut:

- Pull-up
- Pull-down
- Open-drain
- Push-pull
- Current sink
- Current source

2.1. Untuk membuat project baru, buka aplikasi STM32CubeMX. Ketika pertama kali membuka STM32CubeMX kita akan dihadapkan dengan tampilan berikut :

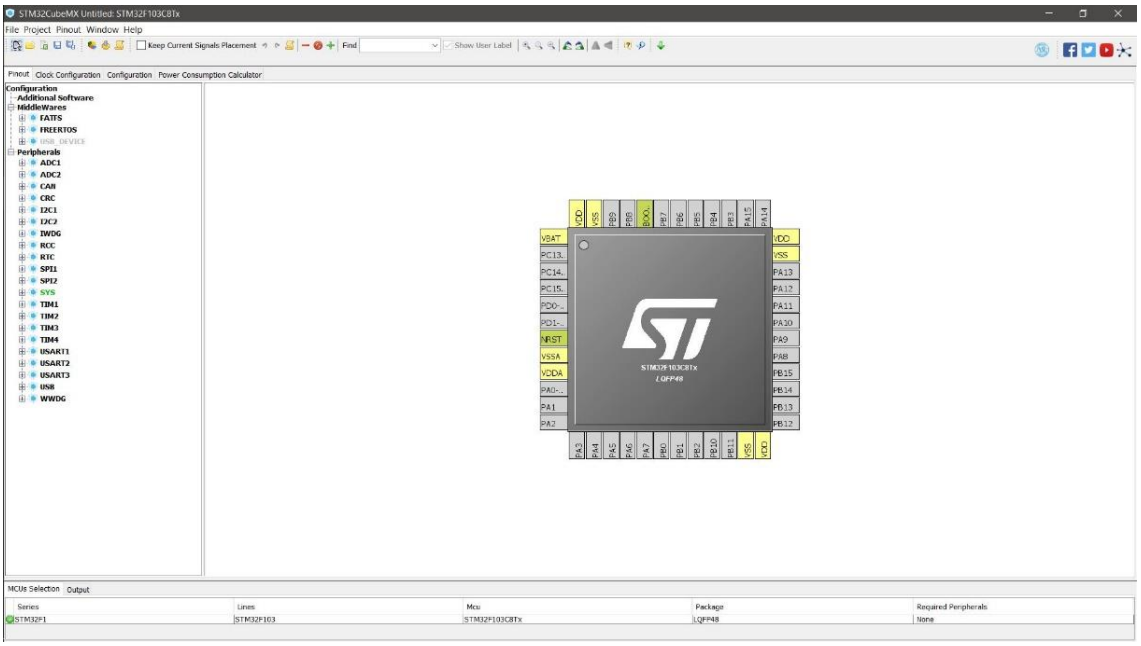


2.2. Klik New Project, maka kita akan dihadapkan pada jendela MCU selector.



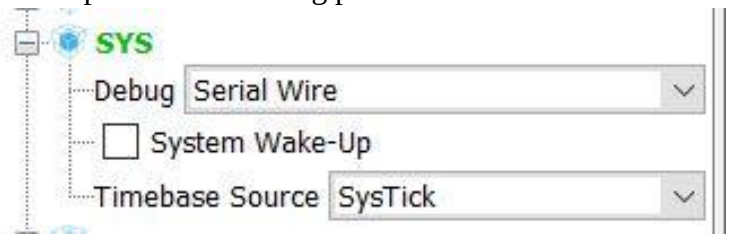
Pada jendela ini masukkan seri mikrokontroller yang akan digunakan pada kolom Part Number Serach. Kemudian double-click mikrokontroller yang akan digunakan pada kolom sebelah kanan-bawah.

2.3. Setelah memilih mikrokontroler yang akan digunakan, pengguna akan disajikan dengan tampilan Pinout View

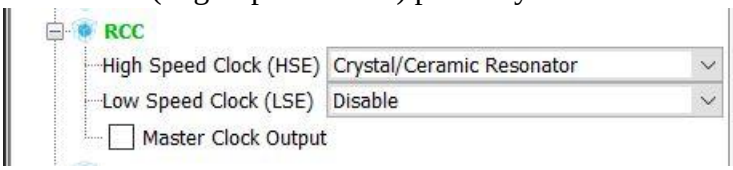


Pada tampilan ini kita dapat mengatur fungsi masing - masing pin pada mikrokontroler dan mengaktifkan berbagai peripheral pada mikrokontroler.

2.4. Hal pertama yang harus dilakukan pada jendela ini adalah memilih programmer yang akan digunakan. Karena pada project ini akan menggunakaan ST-Link programmer maka expand panel SYS dan pada kolom Debug pilih Serial Wire.

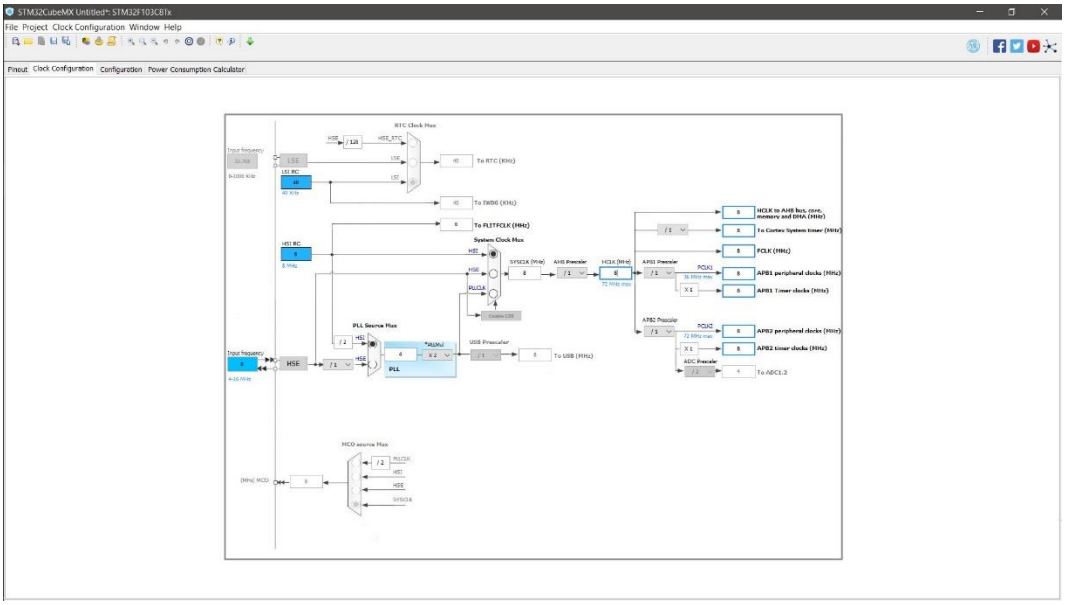


2.5. Selanjutnya adalah melakukan pengaturan RCC (Reset and Clock Control). Expand panel RCC dan pada kolom HSE (High Speed Clock) pilih Crystal/Ceramic resonator.

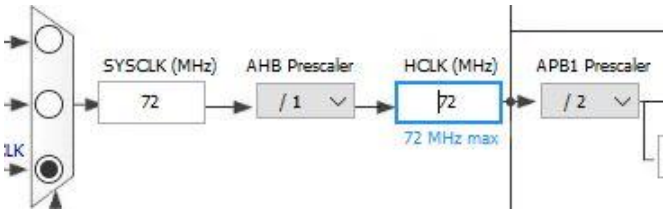


Dengan demikian mikrokontroler akan menggunakan 8MHz Crystal pada board Blue Pill sebagai sumber clock.

2.6. Setelah melakukan konfigurasi RCC buka tab Clock Configuration.

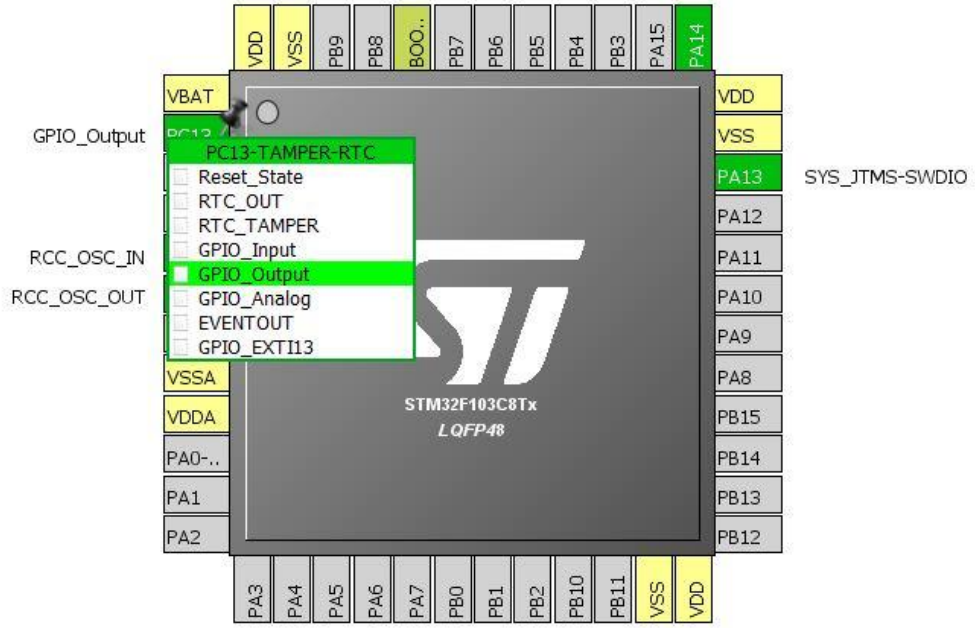


Tampilan Clock Configuration menampilkan clock tree yang terdapat pada mikrokontroler yang digunakan. Tampilan pada jendela ini terlihat sedikit membingungkan, namun pada project ini kita hanya perlu memperhatikan satu parameter yaitu HCLK.

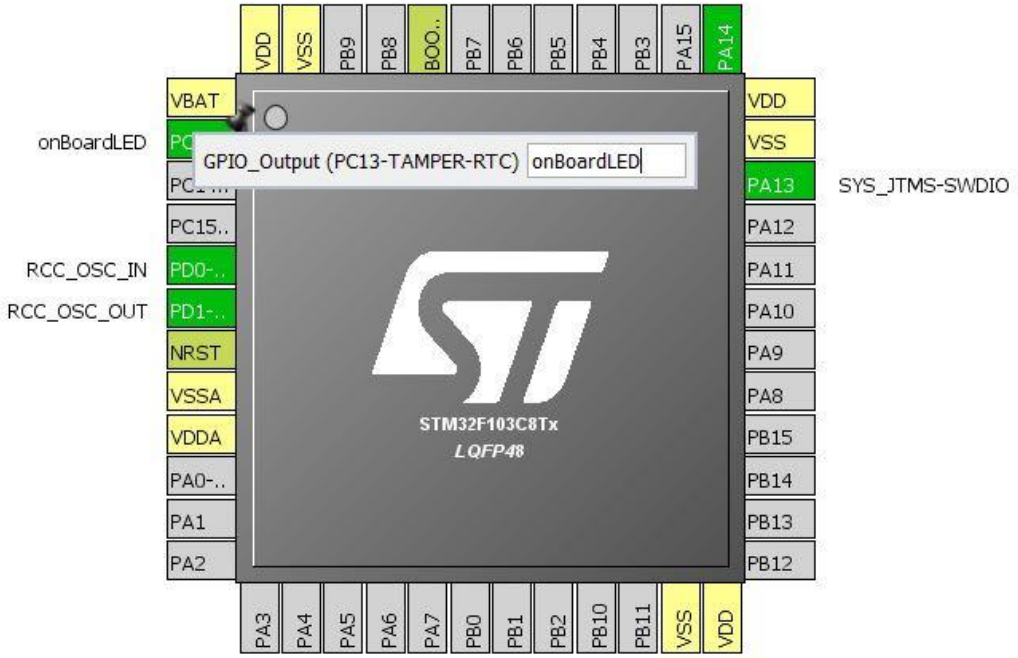


HCLK adalah sumber clock bagi CPU dan APB (Advanced Peripheral Bus). Masukkan nilai 72 pada HCLK, enter dan klik OK pada jendela peringatan yang muncul. Pada project ini kita akan menggunakan nilai 72MHz sehingga CPU akan bekerja apada 72MHz. Pengguna juga dapat menginputkan nilai frekuensi lain seperti 8MHz, 16MHz, 48MHz dan sebagainya.

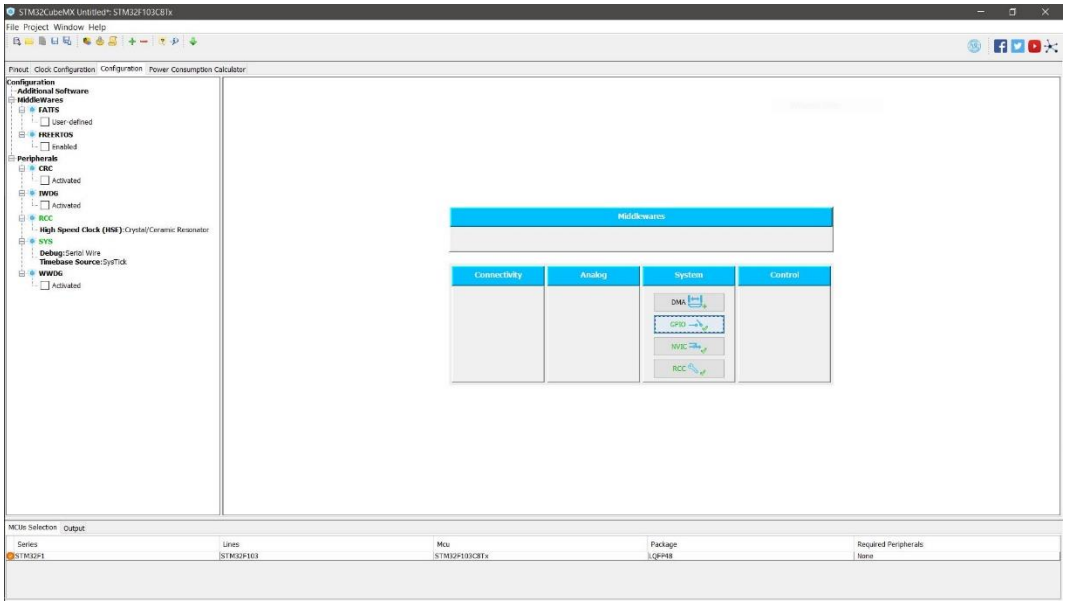
2.7. Selanjutnya buka kembali tampilan Pinout View. Pada development board yang kita gunakan terdapat LED yang terhubung pada pin C13. Untuk menggunakan LED tersebut kita perlu mengatur pin C13 sebagai output. Untuk melakukan hal tersebut klik pada PC13 dan pilih GPIO_Output.



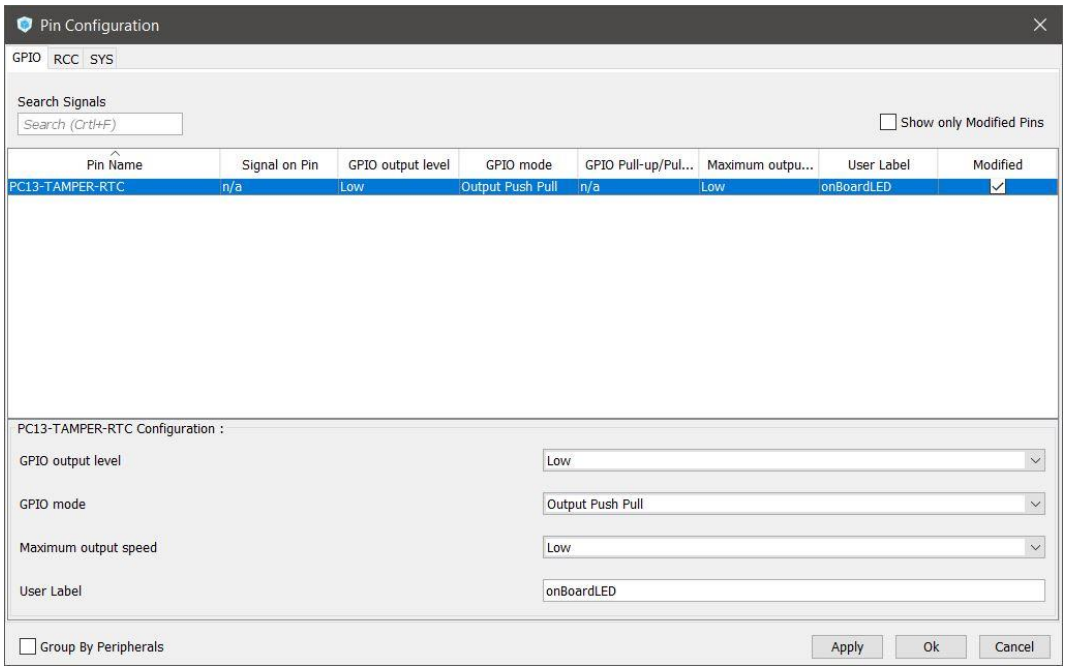
2.8. Pengguna juga dapat memberikan label pada pin C13 yang dapat mempermudah dalam penulisan program. Klik kanan pada PC13 dan pilih Enter User Label.



2.9. Klik pada tab Configuration. Pada Configuration view kita dapat melakukan pengaturan yang lebih spesifik pada masing – masing peripheral yang akan digunakan.

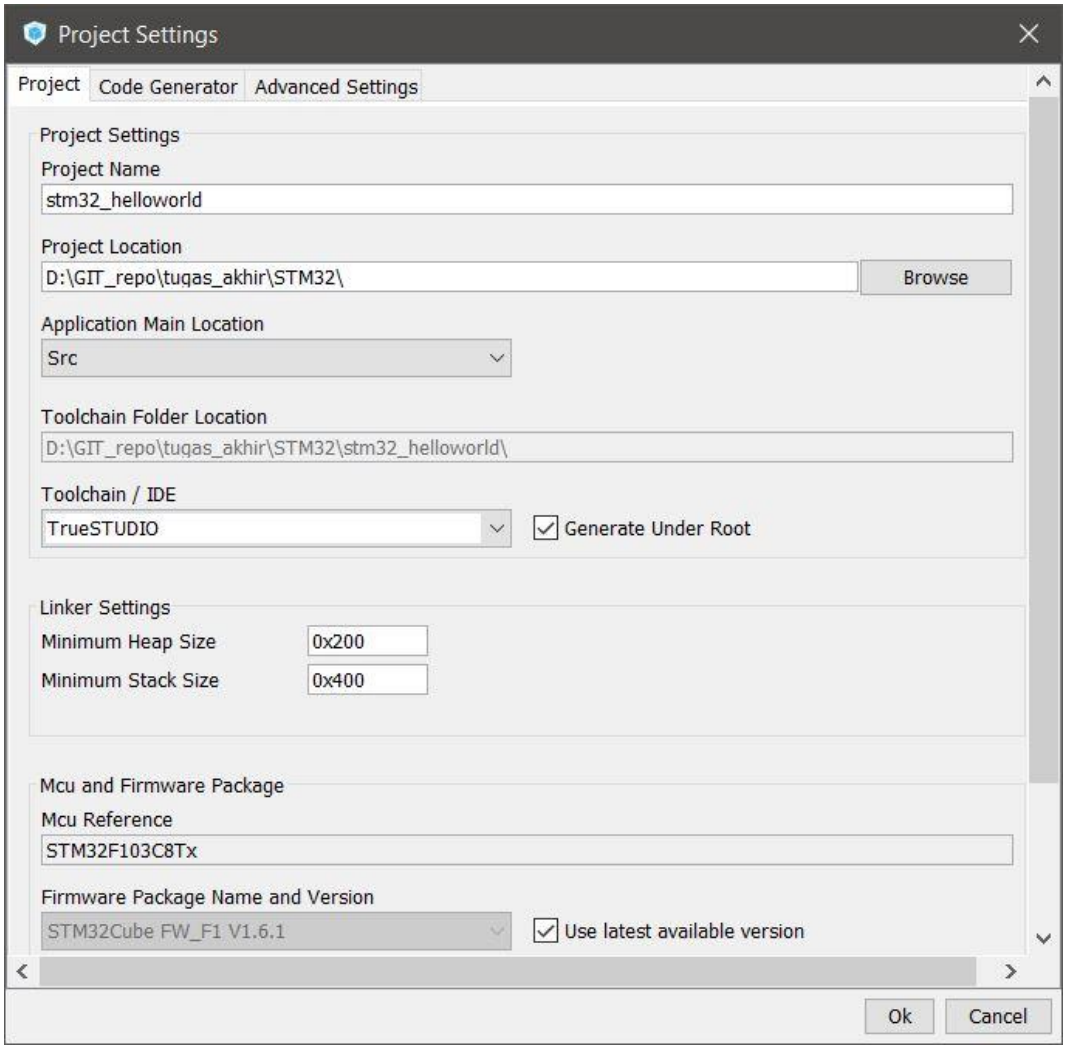


Pada tampilan Configuration view click GPIO, maka kita akan dihadapkan dengan tampilan berikut.



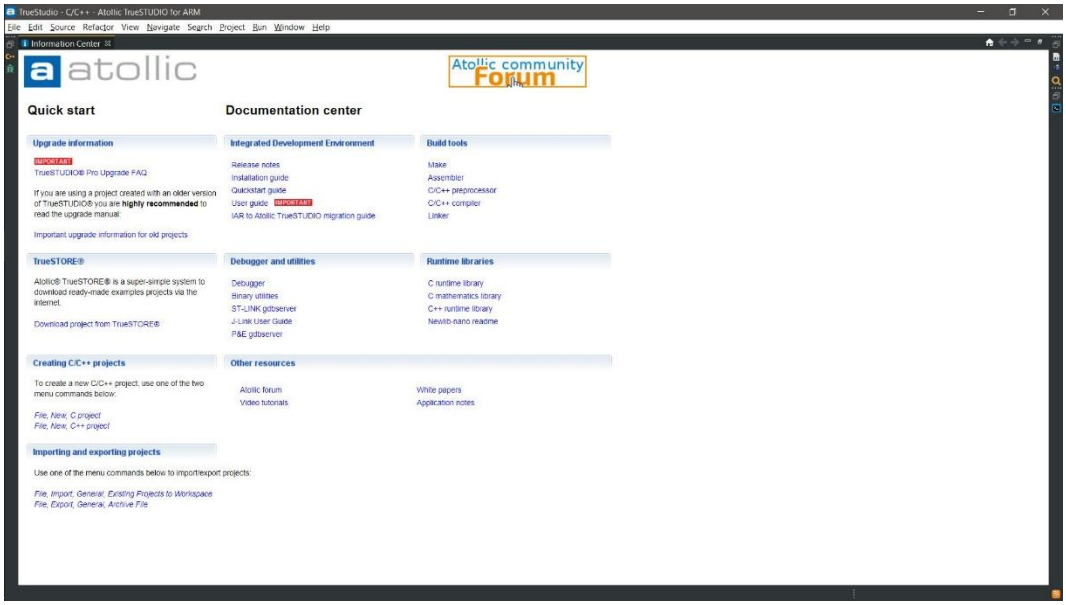
Pilih PC-13. Pada kolom GPIO mode pilih Output Push Pull, dengan demikian pin C13 dapat berperan sebagai current sink dan currect source. Kita juga dapat mengatur GPIO mode menjadi open-drain namun pada mode ini pin C13 hanya dapat berperan sebagain current sink.

2.10. Pada tahap ini pengaturan mikrokontroler telah selesai. Klik project - Generate Code (Ctrl + Shift + G)

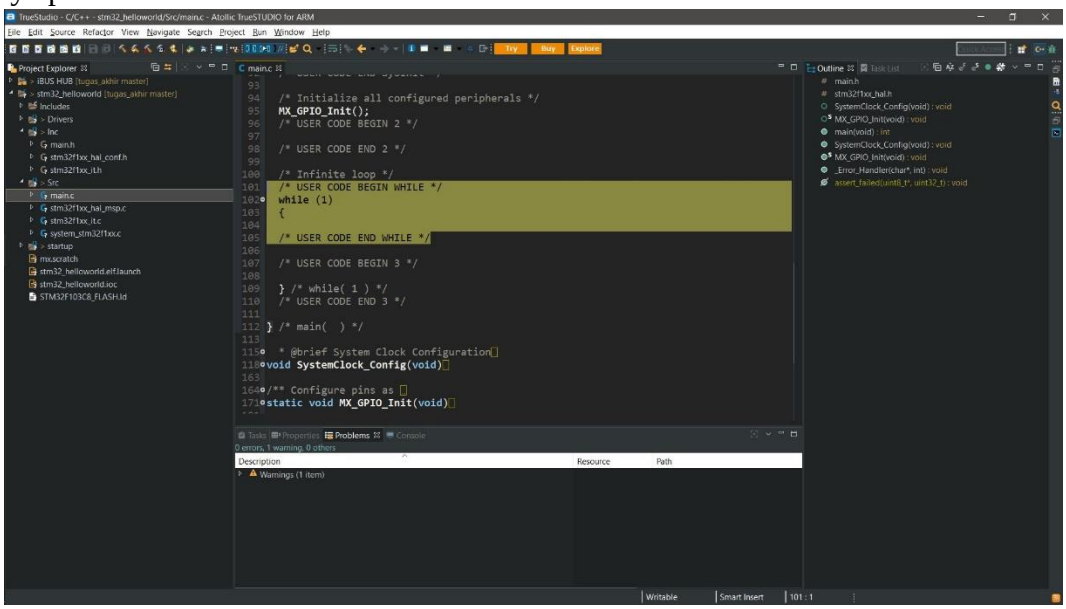


Berikan nama project dan directory project. Pada kolom Toolchain / IDE pilih TrueSTUDIO. Klik Open Project pada jendela yang akan muncul. Jangan tutup aplikasi STM32CubeMX cukup minimize saja.

2.11. Aplikasi Atollic TrueSTUDIO akan otomatis terbuka. Ketika Atollic TrueSTUDIO pertama kali dibuka kita akan dihadapkan pada tab Information Center.



Tutup tab Information Center untuk memasuki jendela editor utama. Pada project ini kita hanya perlu memodifikasi file main.c.



2.12. Sisipkan program berikut pada file main.c

```
  /* USER CODE BEGIN WHILE */
  while (1)
  {
      HAL_GPIO_TogglePin(onBoardLED_GPIO_Port, onBoardLED_Pin);
      HAL_Delay(300);

//    HAL_GPIO_WritePin(onBoardLED_GPIO_Port, onBoardLED_Pin, 1);
//    HAL_Delay(300);
//    HAL_GPIO_WritePin(onBoardLED_GPIO_Port, onBoardLED_Pin, 0);
//    HAL_Delay(300);
  /* USER CODE END WHILE */
```

Ketika menuliskan program pada project yang dibuat, pastikan hanya menuliskan program diantara baris :
/* USER CODE BEGIN xxxxx */ dan /* USER CODE END xxxxx */ karena program diluar kedua baris tersebut akan otomatis terhapus ketika kita me-generate ulang project menggunakan STM32CubeMX.

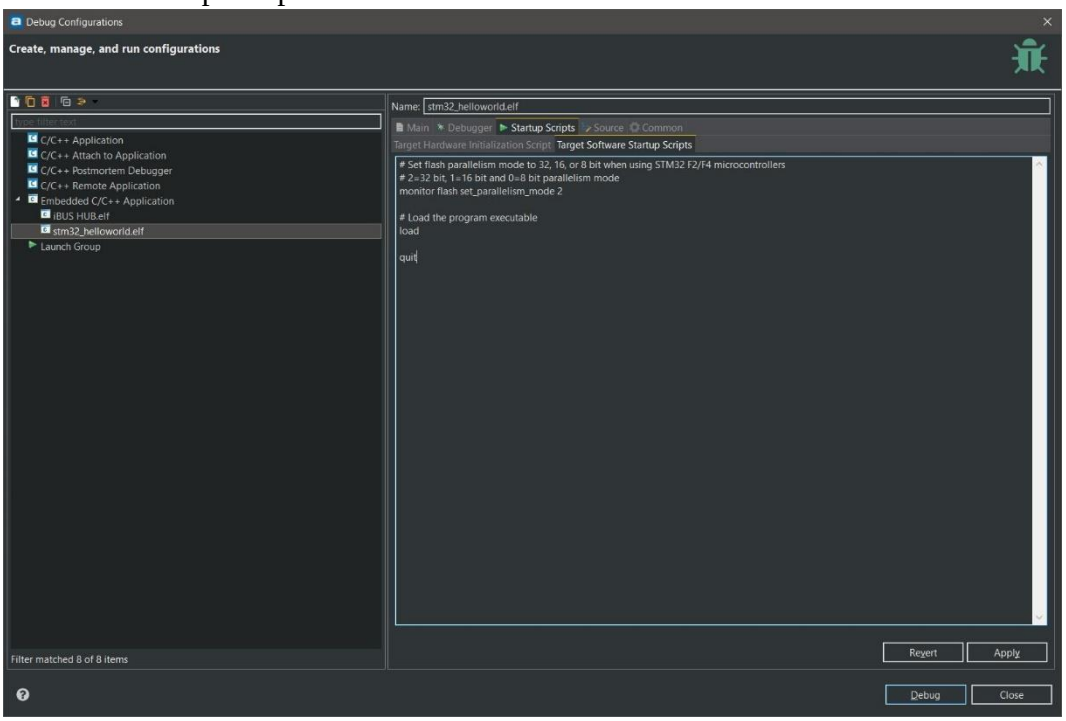
Program yang kita tuliskan diatas baris while (1) { akan dijalankan satu kali layaknya void setup() pada pemrograman Arduino, sedangkan program dibawah while (1) { akan dijalankan berulang – ulang layaknya pada void loop().

Pada project ini kita akan membuat blinking LED yang akan menyala selama 300ms dan mati selama 300ms. Output yang sama juga dapat diperoleh dengan menggunakan empat baris program dibawahnya (commented).

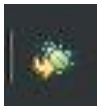
2.13. Hingga tahap ini, program sudah siap untuk di-upload pada mikrokontroler. Untuk melakukan upload pada mikrokontroler kita perlu melakukan pengaturan proses debug terlebih dahulu. Pada toolbar, klik ikon debug setting.

Jendela debug configuration akan terbuka. Pada kolom sebelah kiri expand panel Embedded C/C++ Aplication dan pilih file ber-extensi .elf dengan nama file yang sama dengan nama project yang sedang aktif.

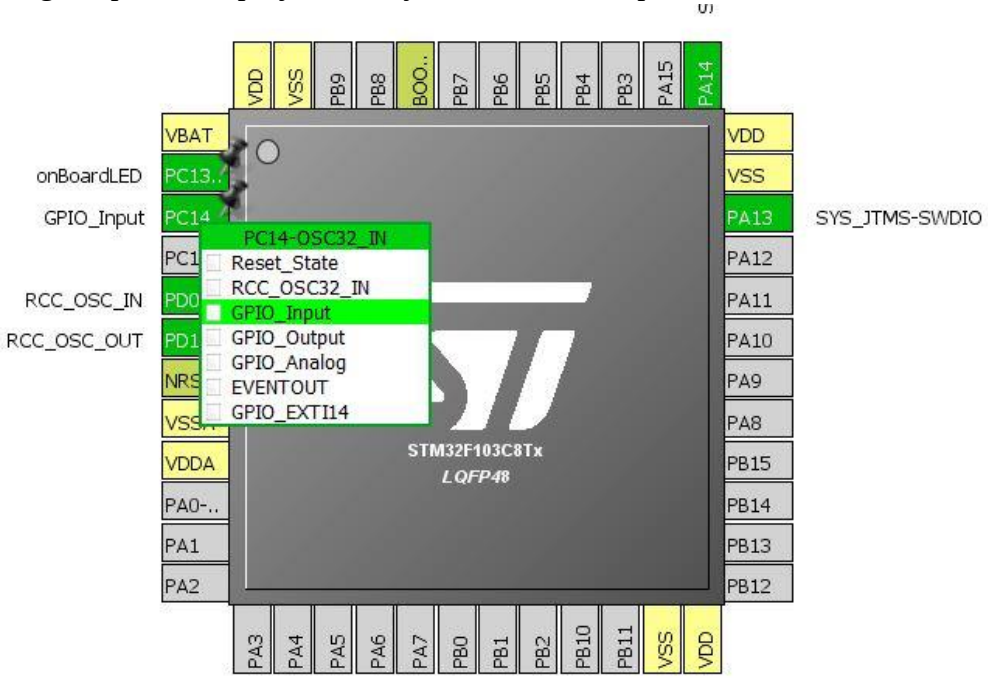
2.14. Buka tab Startup Scripts



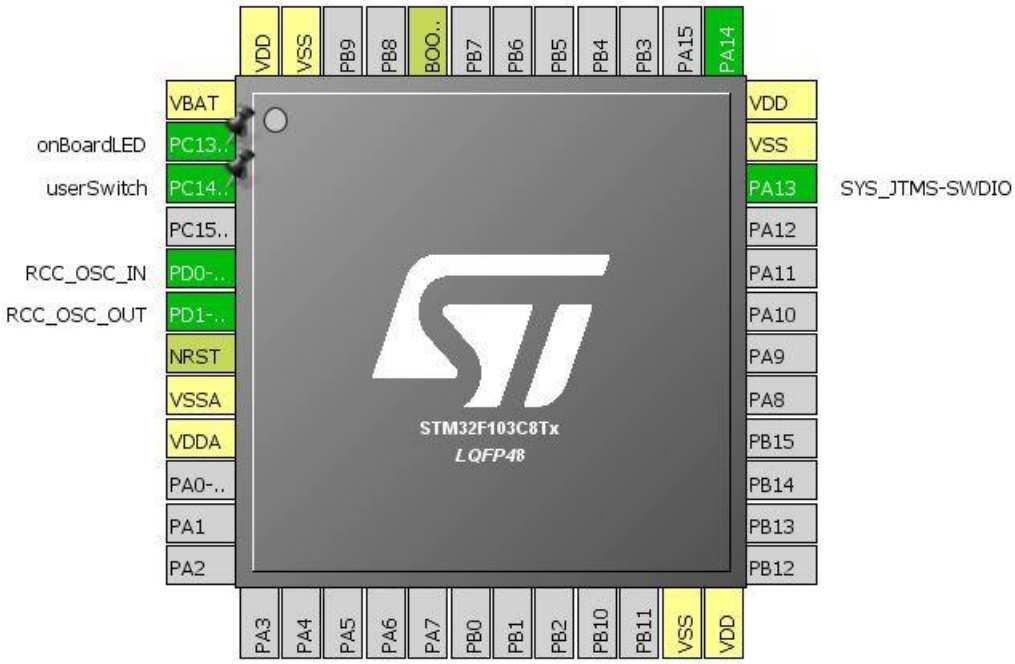
Hapus semua text dibawah baris load dan tuliskan quit sebagai penggantinya. Klik apply.

2.15. Untuk mekakukan proses upload pada mikrokontroler klik ikon debug  dan tunggu hingga proses selesai.

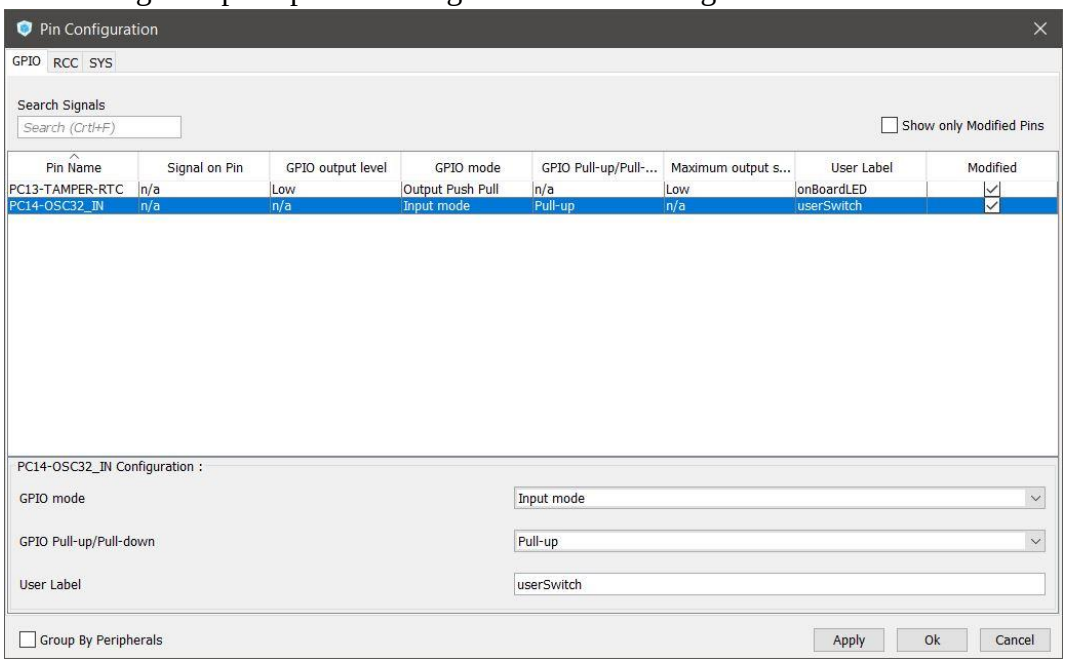
2.16. Pada langkah selanjutnya kita akan menambahkan fungsi untuk membaca input pada GPIO. Buka kembali pinout view pada STM32CubeMX. Pilih salah satu pin dan atur sebagai input. Pada project ini saya mencontohkan pin C14.



2.17. Berikan label seperti pada langkah 2.8



2.18. Laukan konfigurasi pada pin C14 dengan membuka configuration viev – GPIO



Pilih PC14 dan pada kolom GPIO Pull-up/Pull-down pilih mode yang diinginkan. Klik apply.

2.19. Pada langkah ini kita akan me-generate ulang project. Klik Project - Generate Code (Ctrl + Shift + G). Klik close pada jendela yang akan muncul.

2.20. Buka Atollic TrueSTUDIO. Edit file main.c seperti pada gambar berikut

```
/* USER CODE BEGIN WHILE */
while (1)
{
    if(HAL_GPIO_ReadPin(userSwitch_GPIO_Port, userSwitch_Pin) == 0){
        HAL_GPIO_TogglePin(onBoardLED_GPIO_Port, onBoardLED_Pin);
        HAL_Delay(300);
        //      HAL_GPIO_WritePin(onBoardLED_GPIO_Port, onBoardLED_Pin, 1);
        //      HAL_Delay(300);
        //      HAL_GPIO_WritePin(onBoardLED_GPIO_Port, onBoardLED_Pin, 0);
        //      HAL_Delay(300);

    } /* if( HAL_GPIO_...ch_Pin) == 0 ) */
/* USER CODE END WHILE */
```

Kali ini LED akan berhenti berkedip ketika pin C14 bernilai 1.

2.21. Upload program pada mikrokontroler seperti pada langkah 2.15

3. PROJECT USART

Pada project ini kita akan membahas mengenai komunikasi serial USART (Universal Synchronous/Asynchronous Receiver/Transmitter). Komunikasi USART banyak digunakan pada mikrokontroler karena hanya membutuhkan dua koneksi data (mode asynchronous) sehingga praktis. Komunikasi USART juga mudah untuk diimplementasikan dalam program. Komunikasi USART pada STM32F103 dapat dioprasikan menjadi tiga mode aplikasi yaitu mode blocking, interrupt dan DMA(Direct Memory Access). Pada project ini USART akan dioperasikan pada mode yang paling mendasar yaitu mode blocking.

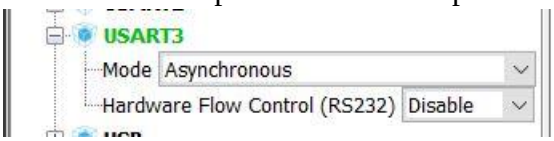
Sebelum memulai project ini, pembaca diharapkan sudah memahami beberapa konsep dasar berikut:

- Baud rate
- Synchronous USART
- Asynchronous USART
- UART

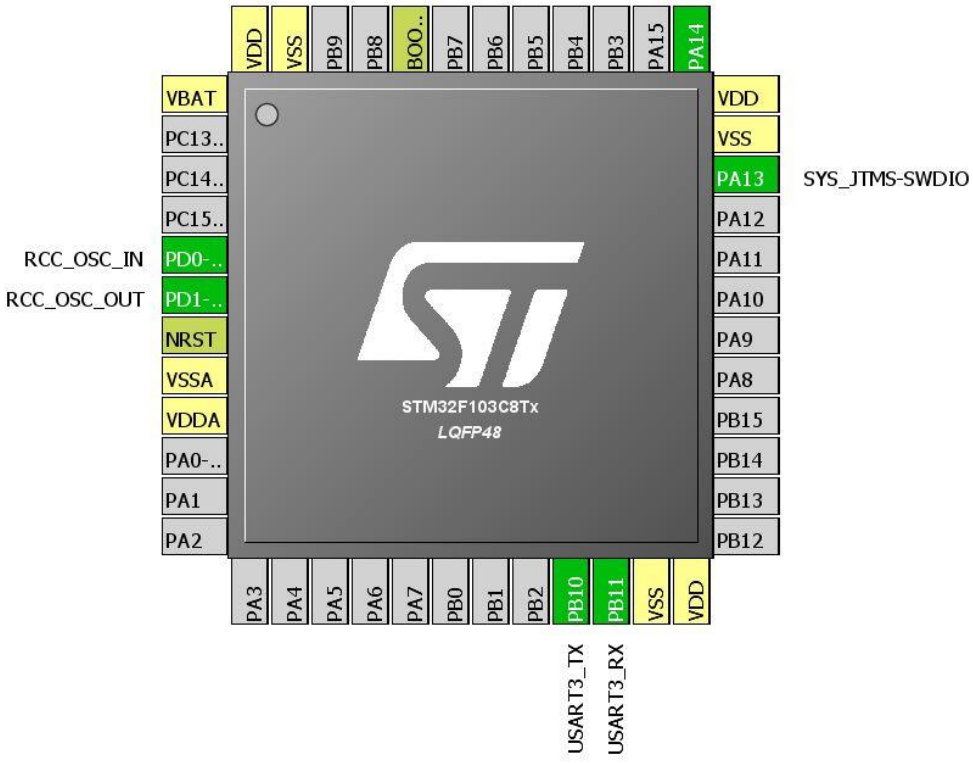
3.1. Buat project baru seperti pada langkah 2.1 hingga 2.3

3.2. Atur SYS, RCC dan HCLK seperti pada langkah 2.4 hingga 2.6

3.3. Pada STM32F103C8T6 terdapat 3 hardware USART. Kita dapat menggunakan salah satu daru hardware USART tersebut. Pada tutorial kita akan menggunakan hardware USART3. Expand panel USART3 dan pada kolom mode pilih Asynchronous.

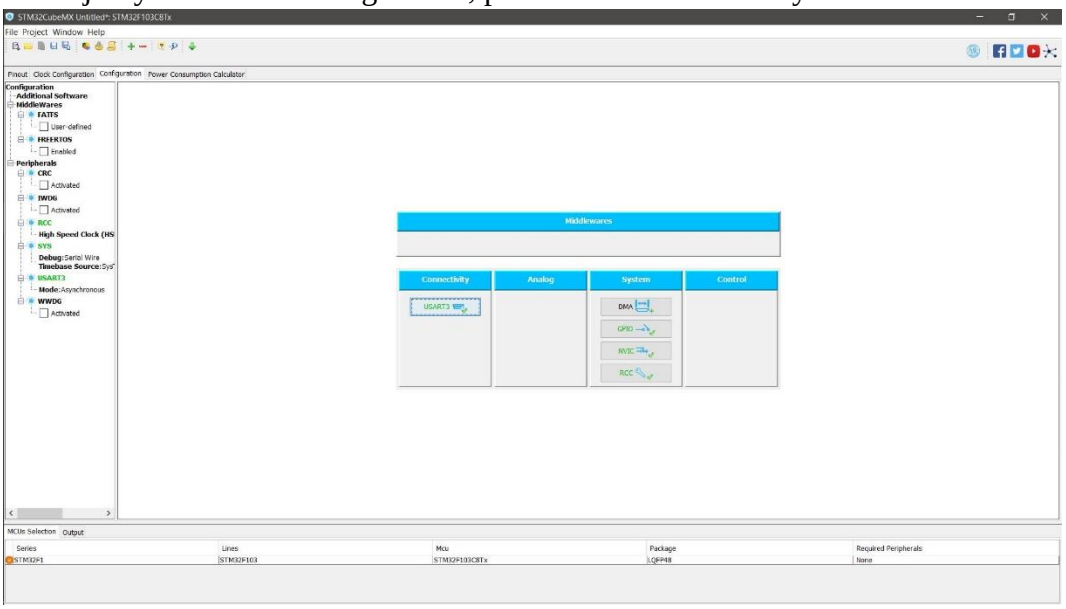


Dengan demikian pin B10 dan pin B11 akan dengan sendirinya terinisialisasi sebagai USART_TX dan USART_TX.

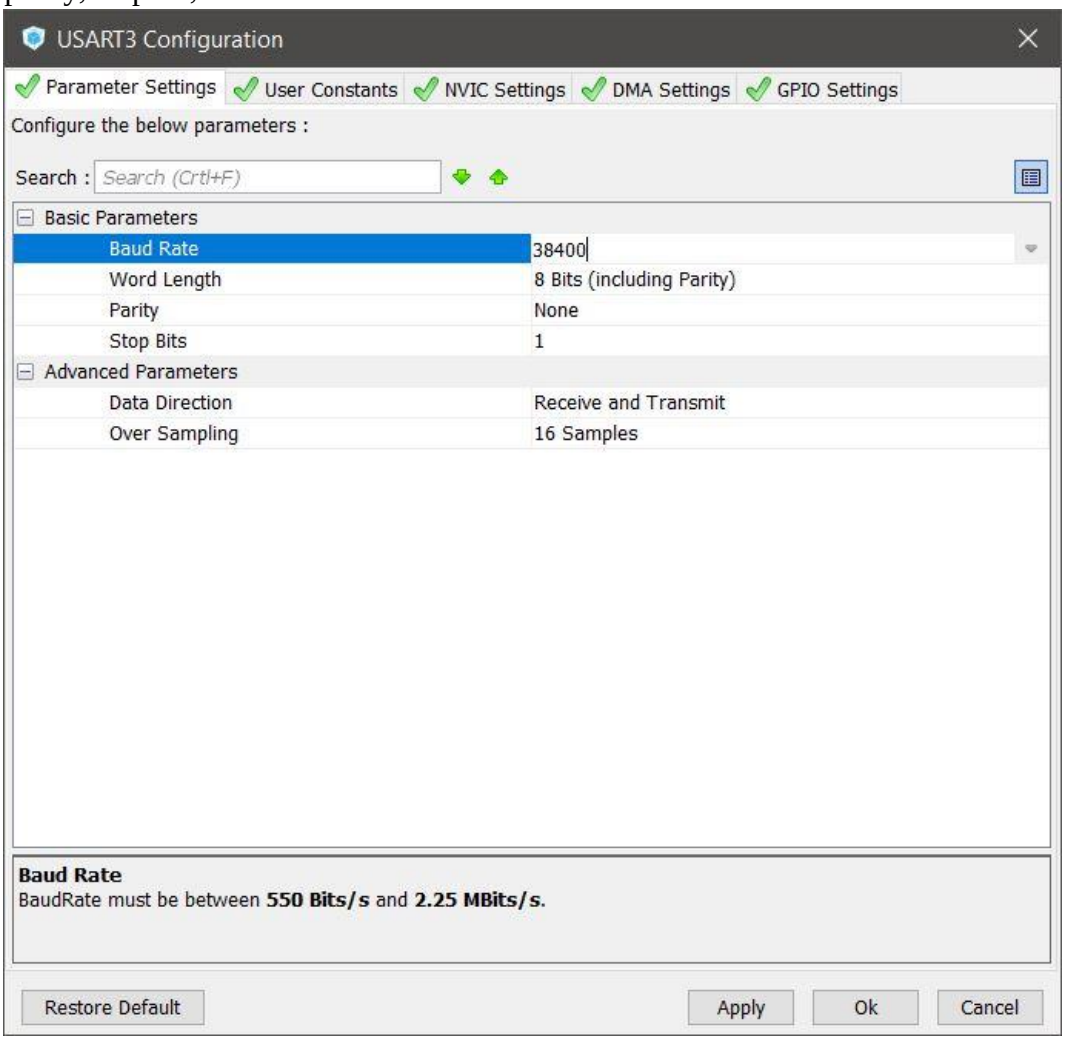


Perlu diperhatikan bahwa STM32F103C8T6 pin B10 dan B11 bekerja pada level tegangan 3.3V, namun pin tersebut 5V tolerant, yang berarti dapat menerima input hingga 5V. Tidak semua pin pada STM32F103C8T6 5V tolerant sehingga ketika menggunakan hardware USART yang lain kita perlu memastikan pin yang digunakan dapat menerima input 5V.

3.4. Selanjutnya buka tab Configuration, pada kolom Connectivity klik USART3.



3.5. Pada langkah ini kita akan melakukan konfigurasi pada USART3. Pada jendela USART3 Configuration, kita dapat menentukan parameter protokol komunikasi USART3 seperti parity, stop bit, baud rate dan lain – lain.



Pada project ini kita hanya perlu memperhatikan satu parameter yaitu Baud Rate. Isikan nilai Baud Rate yang ingin digunakan pada panel Basic Setting, kolom Baud Rate.

3.6. Pada tahap ini pengaturan mikrokontroler telah selesai. Kita hanya perlu me-generate dan membuka project yang dihasilkan seperti pada langkah 2.10 hingga 2.11

3.7. Pada project ini kita hanya perlu melakukan modifikasi pada file main.c. sisipkan program berikut pada file main.c

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
uint8_t receive_buffer[50];
while (1)
{
    HAL_UART_Receive(&huart3, receive_buffer, 50, 100);
    int jumlah_data_diterima = 49 - huart3.RxXferCount;

    if(jumlah_data_diterima > 0)
        HAL_UART_Transmit(&huart3, receive_buffer, jumlah_data_diterima, 20);
/* USER CODE END WHILE */
```

Variable receive buffer berfungsi untuk menyimpan data yang diterima pada komunikasi serial. Fungsi

```
HAL_StatusTypeDef HAL_UART_Receive(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout)
```

digunakan untuk menerima data serial sebanyak maksimal `Size` byte data. Jika ternyata data yang diterima kurang dari `Size` byte data maka fungsi tersebut akan menunggu selama `Timeout` mS. Untuk mengetahui banyaknya data yang telah diterima selayaknya Serial.available() Arduino kita perlu menggunakan rumus berikut :

$$JumlahDataDiterima(byte) = JumlahDataMaximal - 1 - \text{huartx.RxXferCount}$$

jumlah_data_diterima = 50 - 1 - huartx.RxXferCount

Untuk mengirimkan data pada komunikasi serial kita dapat menggunakan fungsi

```
HAL_StatusTypeDef HAL_UART_Transmit_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size)
```

Fungsi tersebut akan mengirimkan data yang tersimpan pada array `*pData` sebanyak `Size` byte.

Program pada project ini akan menerima data sebanyak maksimal 50 byte dari komputer yang terhubung. Data yang diterima tersebut akan dikirimkan kembali untuk ditampilkan pada serial monitor.

3.8. Upload program pada mikrokontroler seperti pada langkah 2.14 hingga 2.15

3.9. Hubungkan mikrokontroler dengan PC menggunakan USB to Serial TTL converter CP2102/PL2303 atau FTDI FT232. Hubungkan pin B10 pada RX dan pin B11 pada TX.

3.10. Buka aplikasi Serial Monitor pada PC seperti CoolTerm, TeraTerm, Putty atau Arduino Serial Monitor. Pada contoh ini saya menggunakan Arduino Serial Monitor.

3.11. Atur Baudrate pada Serial Monitor sesuai dengan nilai baudrate pada langkah 3.5

Masukkan pesan pada kolom input dan klik send. Mikrokontroler akan mengirimkan kembali pesan tersebut pada serial terminal.

4. PROJECT I2C

I2C adalah salah satu jenis komunikasi serial yang banyak digunakan pada mikrokontroler. Komunikasi I2C sering digunakan untuk mengakses modul – modul seperti MPU6050, DS3231, dan lain – lain. Pada project ini kita akan membahas mengenai komunikasi serial USART (Universal Synchronous/Asynchronous Receiver/Transmitter). Seperti pada komunikasi USART, I2C pada STM32F103 juga dapat dioprasikan menjadi tiga mode aplikasi yaitu mode blocking, interrupt dan DMA(Direct Memory Access). Pada project ini I2C akan dioperasikan pada mode blocking. Project ini akan mencontohkan komunikasi I2C antara STM32F103 dengan modul RTC DS3231 untuk membaca data menit dan detik. Pada project ini kita juga akan menggunakan komunikasi Serial untuk menampilkan data pada PC.

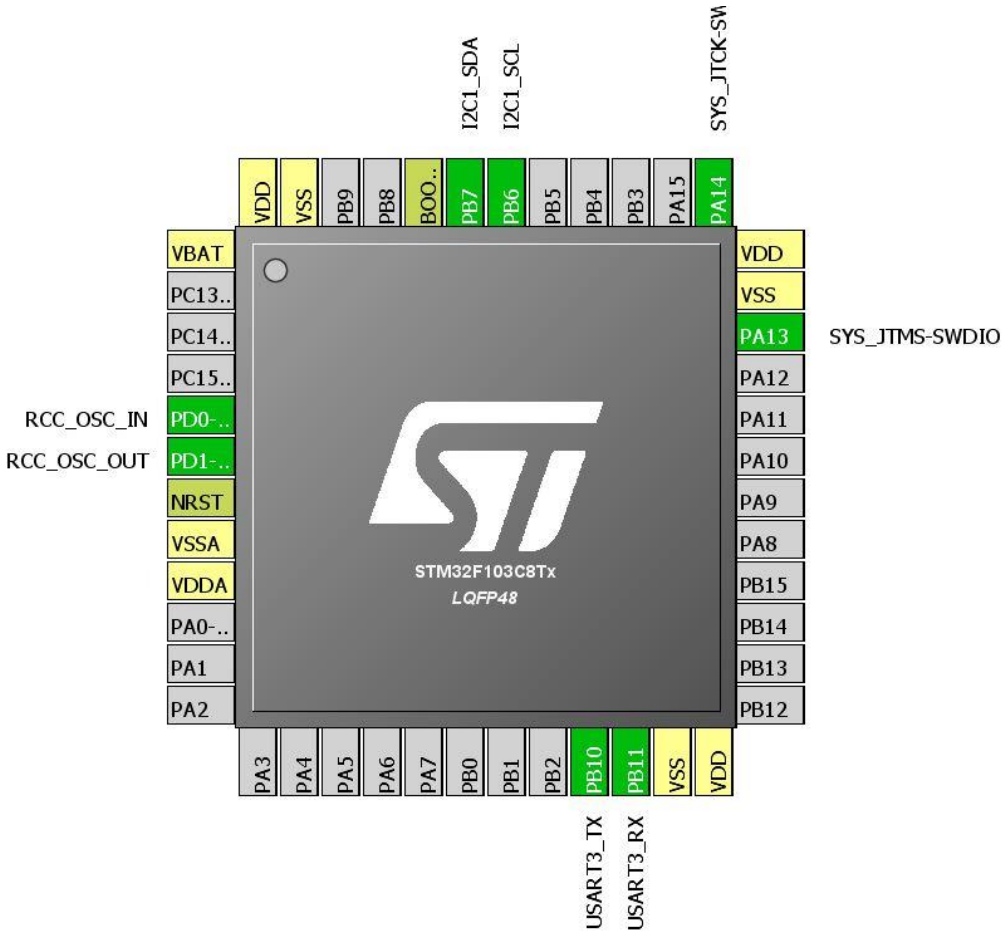
Sebelum memulai project ini, pembaca diharapkan sudah memahami beberapa konsep dasar berikut:

- I2C Slave
- I2C Master
- Device address
- Register address

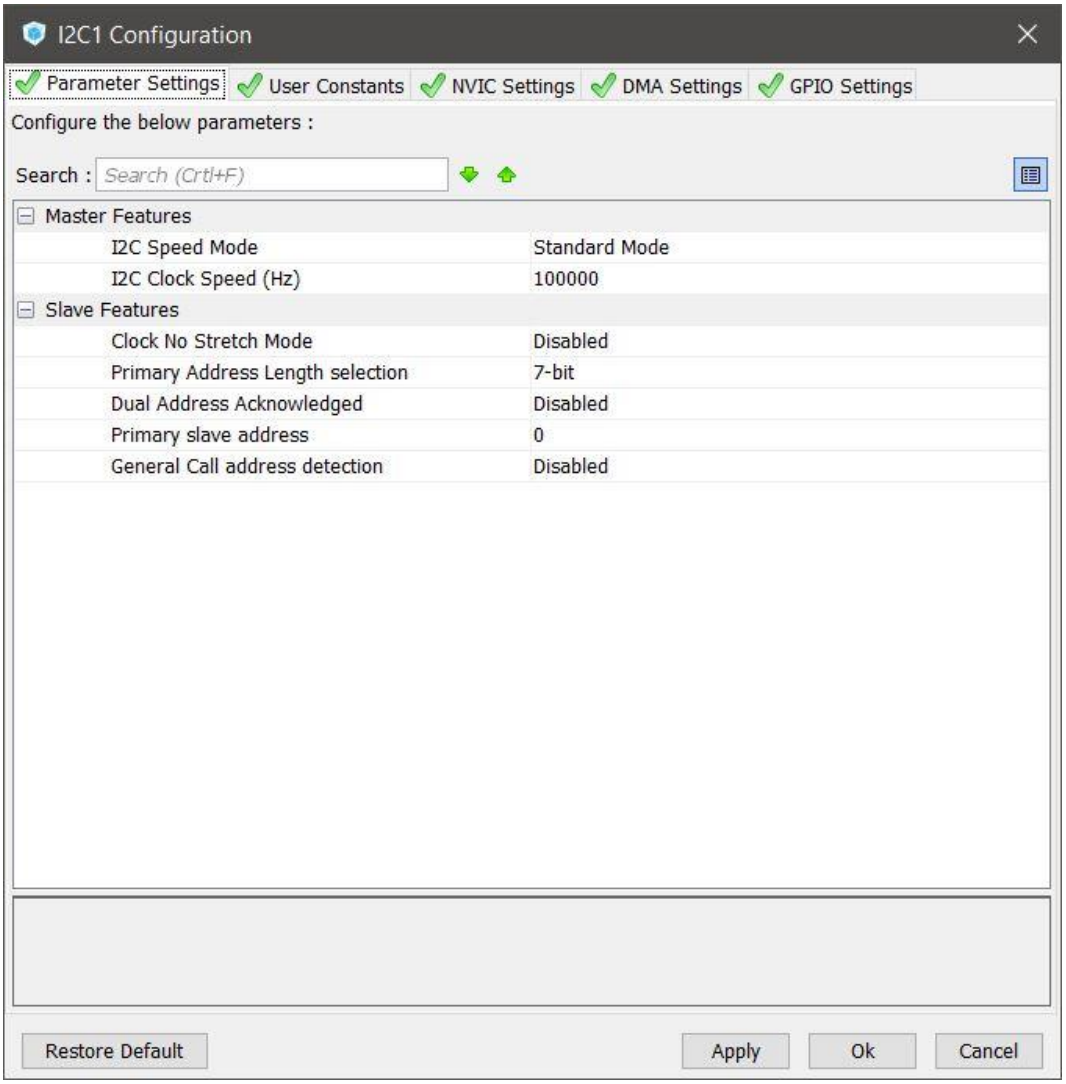
4.1. Buat project baru, atur SYS, RCC, HCLK dan USART3 seperti pada langkah 3.1 hingga 3.5

4.2. Pada STM32F103C8T6 terdapat 2 hardware I2C. Kita dapat menggunakan salah satu daru hardware I2C tersebut. Pada tutorial kita akan menggunakan hardware I2C1. Buka Pinout tab, expand panel I2C1 dan pada kolom I2C pilih I2C.

Dengan demikian pin B7 dan pin B6 akan dengan sendirinya terinisialisasi sebagai I2C_SDA dan I2C_SCL.



4.3. Selanjutnya buka tab Configuration, pada kolom Connectivity, klik I2C1.



Pada jendela I2C1 Configuration kita dapat mengatur beberapa parameter I2C seperti Speed Mode, Clock Stretch, Adress Length dan lain – lain. Pada project ini tidak ada yang perlu dirubah pada jendela I2C1.

4.4. Pada tahap ini pengaturan mikrokontroler telah selesai. Kita perlu me-generate dan membuka project yang dihasilkan seperti pada langkah 2.10 hingga 2.11

4.5. Program pada project ini dibagi menjadi 4 bagian. Bagian pertama berfungsi untuk mengecek apakah Slave sudah siap untuk melakukan transfer data.

```c
/* USER CODE BEGIN WHILE */
uint8_t i2c_receive_buffer[2];
uint8_t i2c_send_buffer[3];
uint8_t i2c_reg_addr = 0x00;
uint8_t uart_send_buffer[50];
uint8_t sec = 0;
uint8_t min = 0;

//cek apakah slave ready
sprintf((char*) uart_send_buffer, "slave ready?\r\n");
HAL_UART_Transmit(&huart3, uart_send_buffer,
    (uint16_t) strlen((char*) uart_send_buffer), 50);

if (HAL_I2C_IsDeviceReady(&hi2c1, 0xD0, 2, 100) == HAL_OK) {
  sprintf((char*) uart_send_buffer, "slave ready!\r\n");
  HAL_UART_Transmit(&huart3, uart_send_buffer,
      (uint16_t) strlen((char*) uart_send_buffer), 50);
} else {
  sprintf((char*) uart_send_buffer, "slave error\r\n");
  HAL_UART_Transmit(&huart3, uart_send_buffer,
      (uint16_t) strlen((char*) uart_send_buffer), 50);
  while (1);
}
```

`HAL_StatusTypeDef HAL_I2C_IsDeviceReady(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint32_t Trials, uint32_t Timeout)`

Fungsi tersebut akan mengecek apakah slave dengan device address `DevAddress`, sudah siap untuk melakukan komunikasi. Jika slave siap untuk melakukan komunikasi maka fungsi akan memberikan return value berupa *HAL_OK*. Jika slave belum siap maka proses tersebut akan diulang sebanyak `Trials` kali percobaan.

Bagian kedua berfungsi untuk mendapatkan nilai menit dan detik.

```c
    //Baca menit dan detik
    sprintf((char*) uart_send_buffer, "Baca menit dan waktu\r\n");
    HAL_UART_Transmit(&huart3, uart_send_buffer,
            (uint16_t) strlen((char*) uart_send_buffer), 50);

    HAL_I2C_Master_Transmit(&hi2c1, 0xD0, &i2c_reg_addr, 1, 20);
    HAL_I2C_Master_Receive(&hi2c1, 0xD0, i2c_receive_buffer, 2, 20);

    sec = i2c_receive_buffer[0] & 0b00001111;
    sec += (i2c_receive_buffer[0] >> 4) * 10;

    min = i2c_receive_buffer[1] & 0b00001111;
    min += (i2c_receive_buffer[1] >> 4) * 10;

    sprintf((char*) uart_send_buffer, "menit: %d, detik: %d\r\n", min, sec);
    HAL_UART_Transmit(&huart3, uart_send_buffer,
            (uint16_t) strlen((char*) uart_send_buffer), 50);
```

`HAL_StatusTypeDef HAL_I2C_Master_Transmit(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint8_t *pData, uint16_t Size, uint32_t Timeout)`
Fungsi tersebut digunakan menuliskan data `*pData` sebanyak `Size` byte, pada slave dengan device address `DevAddress`.

`HAL_StatusTypeDef HAL_I2C_Master_Receive(I2C_HandleTypeDef *hi2c, uint16_t DevAddress, uint8_t *pData, uint16_t Size, uint32_t Timeout)`
Fungsi tersebut digunakan menerima data dari slave dengan device address `DevAddress` sebanyak `Size` byte, yang disimpan pada buffer `*pData`.

Bagian ke-tiga digunakan untuk menuliskan/mengatur menit dan detik pada slave menjadi menit = 0 dan detik = 0.

```
150     //set menit dan detik
151     i2c_send_buffer[0] = 0x00; //register address
152     i2c_send_buffer[1] = 0x00; //data untuk register 0x00 (detik)
153     i2c_send_buffer[2] = 0x00; //data untuk register 0x01 (menit)
154     HAL_I2C_Master_Transmit(&hi2c1, 0xD0, i2c_send_buffer, 3, 20);
155
156     sprintf((char*) uart_send_buffer, "set menjadi 0detik dan 0menit\r\n");
157     HAL_UART_Transmit(&huart3, uart_send_buffer,
158             (uint16_t) strlen((char*) uart_send_buffer), 50);
```

Bagian terakhit digunakan untuk membaca nilai menit dan detik setiap 1 detik.

```
159     while (1) {
160         HAL_I2C_Master_Transmit(&hi2c1, 0xD0, &i2c_reg_addr, 1, 20);
161         HAL_I2C_Master_Receive(&hi2c1, 0xD0, i2c_receive_buffer, 2, 20);
162
163         sec = i2c_receive_buffer[0] & 0b00001111;
164         sec += (i2c_receive_buffer[0] >> 4) * 10;
165
166         min = i2c_receive_buffer[1] & 0b00001111;
167         min += (i2c_receive_buffer[1] >> 4) * 10;
168
169         sprintf((char*) uart_send_buffer, "menit : %d, detik: %d\r\n", min,
170                 sec);
171         HAL_UART_Transmit(&huart3, uart_send_buffer,
172                 (uint16_t) strlen((char*) uart_send_buffer), 50);
173
174         HAL_Delay(1000);
175   /* USER CODE END WHILE */
```

4.6. Upload program pada mikrokontroler seperti pada langkah 2.14 hingga 2.15
4.7. Hubungkan mikrokontroler dengan PC seperti pada langkah 3.8 hingga 3.11

5. PROJECT PWM

PWM (Pulse Width Modulation) adalah salah satu fitur yang dimiliki oleh timer. Pada STM32F103 masing – masing timer dapat memiliki hingga 4 output PWM, bergantung pada jumlah channel output yang dimiliki oleh masing – masing timer.

Sebelum memulai project ini, pembaca diharapkan sudah memahami beberapa konsep dasar berikut:

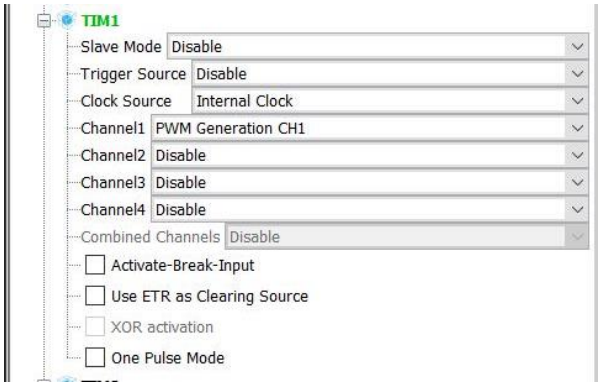
- PWM Periode

- Duty cycle

5.1. Buat project baru seperti pada langkah 2.1 hingga 2.3

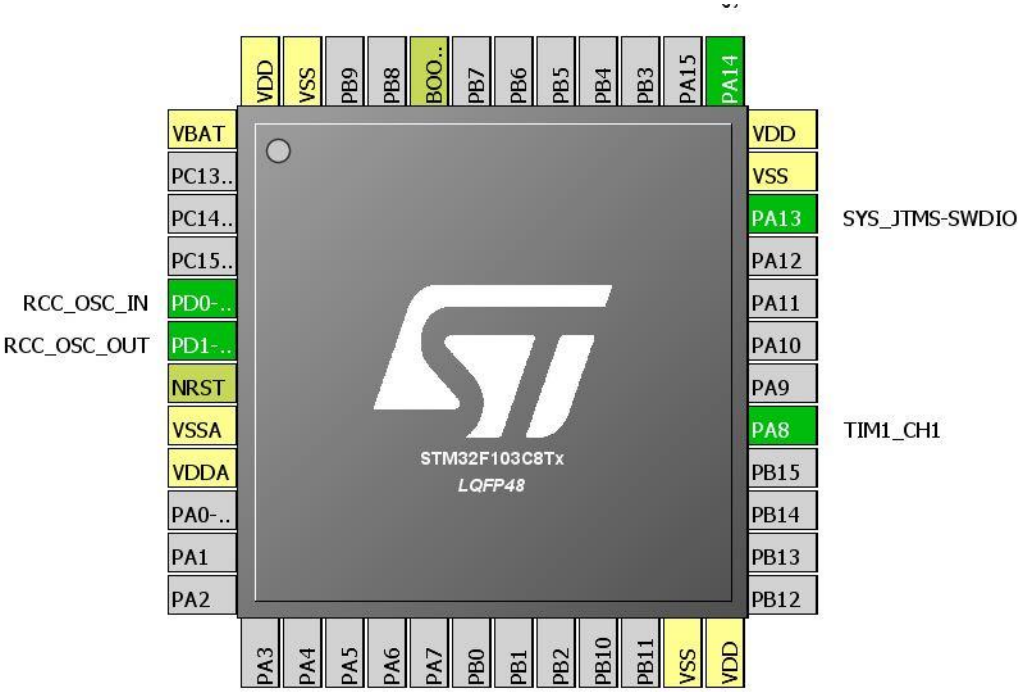
5.2. Atur SYS, RCC dan HCLK seperti pada langkah 2.4 hingga 2.6

5.3. Pada STM32F103C8T6 terdapat 4 buah timer, pada tutorial ini kita akan menggunakan timer 1

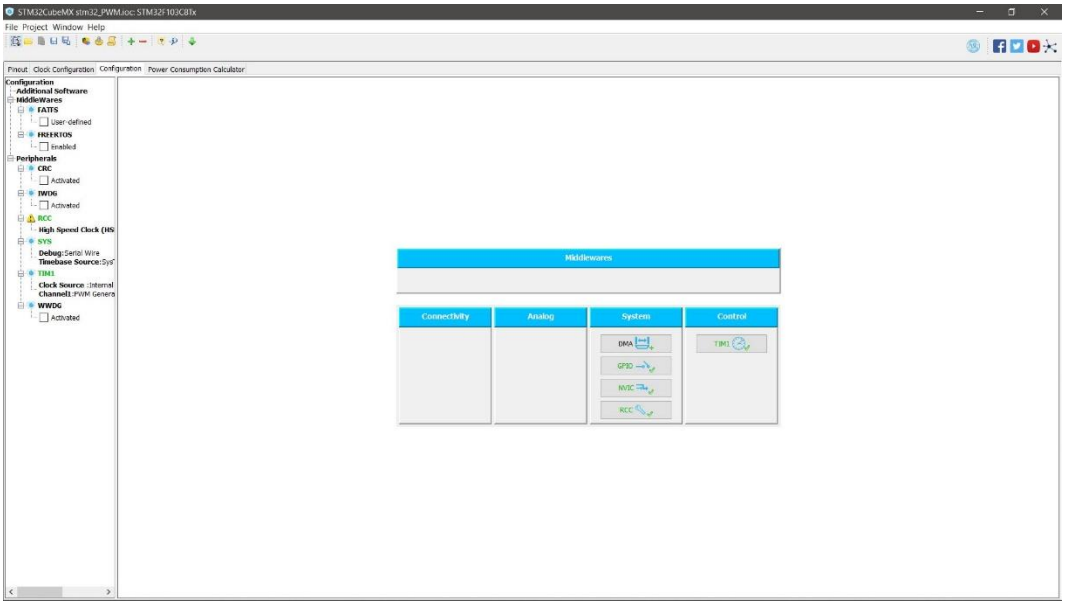


Expand panel TIM1. Pada kolom Clock Source pilih Internal Clock. Pilih PWM Generation pada kolom Channel yang diinginkan, pada contoh ini saya menggunakan Channel1. Kita juga dapat mengaktifkan PWM generation pada lebih dari satu channel untuk menghasilkan lebih dari satu sinyal PWM pada pin GPIO yang berbeda.

Ketika kita mengaktifkan PWM Generation pada Channel1 maka pin A8 akan secara otomatis terinisiasi sebagai output TIM1_CH1 dan pada pin tersebut sinyal PWM akan dihasilkan.



5.4. Buka tab Configuration. Pada kolom Control click TIM1.



5.5. Pada jendela TIM1 Configuration kita perlu mengatur cukup banyak parameter diantaranya PSC(Prescaler), ARR(Auto Reload Register), CKD(Clock Division) dan CCRx(Capture/Compare register).

Nilai – nilai parameter tersebut bisa didapatkan dari beberapa rumus berikut.

$$FrekuensiPWM(Hz) = \frac{TIMxClockSource}{(PSC+1)(ARR+1)(CKD)}$$

$$PeriodePWM(S) = \frac{1}{FrekuensiPWM}$$

$$DutyCycle(\%) = \frac{CCRx}{ARR}$$

Pada STM32F103 TIM1 memiliki sumber clock yang berasal dari clock APB1 (Advanced Peripheral Bus). Nilai frekuensi clock APB1 dapat dilihat pada tab Clock Configuration. Pada project ini frekuensi clock APB1 adalah 72Mhz, sehingga nilai TIMxClockSource adalah 72Mhz. Untuk mengetahui clock source pada peripheral timer yang lain, kita dapat mengacu pada user manual STM32F103.

Nilai CCRx pada STM32CubeMX disebut juga dengan "pulse". Nilai pulse/CCRx pada masing – masing channel dapat berbeda – beda, sehingga setiap channel pada satu timer dapat menghasilkan sinyal PWM dengan duty cycle yang berbeda – beda pula. CCRx/pulse tidak harus diinisialisasi pada STM32CubeMX karena nilai CCRx/pulse dapat dirubah dengan mudah pada program yang akan dibuat. Perlu diperhatikan juga bahwa nilai CCRx/pulse tidak dapat lebih besar dari nilai ARR.

5.6. Pada tahap ini pengaturan mikrokontroler telah selesai. Kita hanya perlu me-generate dan membuka project yang dihasilkan seperti yang sudah dijelaskan pada langkah 2.10 hingga 2.11

5.7. Pada project ini kita hanya perlu melakukan modifikasi pada file main.c. Sisipkan program berikut pada file main.c

```
106   /* USER CODE BEGIN WHILE */
107   HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
108   htim1.Instance->CCR1 = 5000;
109   while (1)
110   {
111   /* USER CODE END WHILE */
```

`HAL_StatusTypeDef HAL_TIM_PWM_Start(TIM_HandleTypeDef *htim, uint32_t Channel)`

Fungsi tersebut digunakan untuk memulai/mengaktifkan PWM pada timer `*htim`, channel `Channel`.

Untuk mengubah nilai CCR pada timer 1 channel 1 gunakan `htim1.Instance->CCR1`, untuk mengubah nilai CCR pada timer 3 channel 2 gunakan `htim3.Instance->CCR2`, untuk mengubah CCR pada timer x channel x gunakan `htimx.Instance->CCRx`.

5.8. Upload program pada mikrokontroler seperti pada langkah 2.14 hingga 2.15

6. PROJECT INPUT_CAPTURE

Input Capture adalah salah satu fitur yang dimiliki oleh timer. Input Capture dapat digunakan untuk mengetahui frekuensi, periode, duty cycle dan parameter – parameter lain dari sebuah input. Salah satu aplikasi input capture yang banyak ditemui adalah untuk membaca sinyal PPM(pulse position modulation).

Sebelum memulai project ini, pembaca diharapkan sudah memahami beberapa konsep dasar berikut:

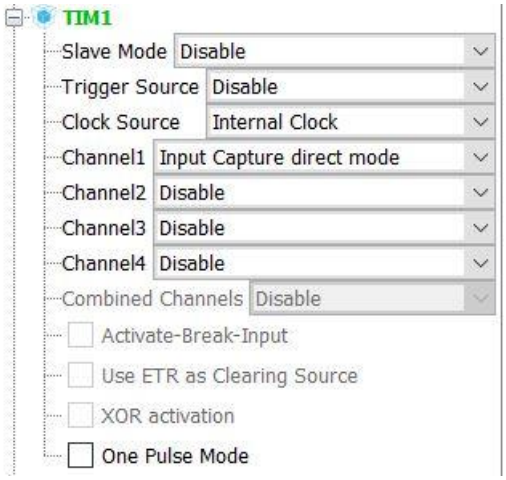
- Rising edge
- Falling edge

- IRQ (Interrupt Request)
- ISR (Interrupt Service Routine)
- Timer/Counter overflow

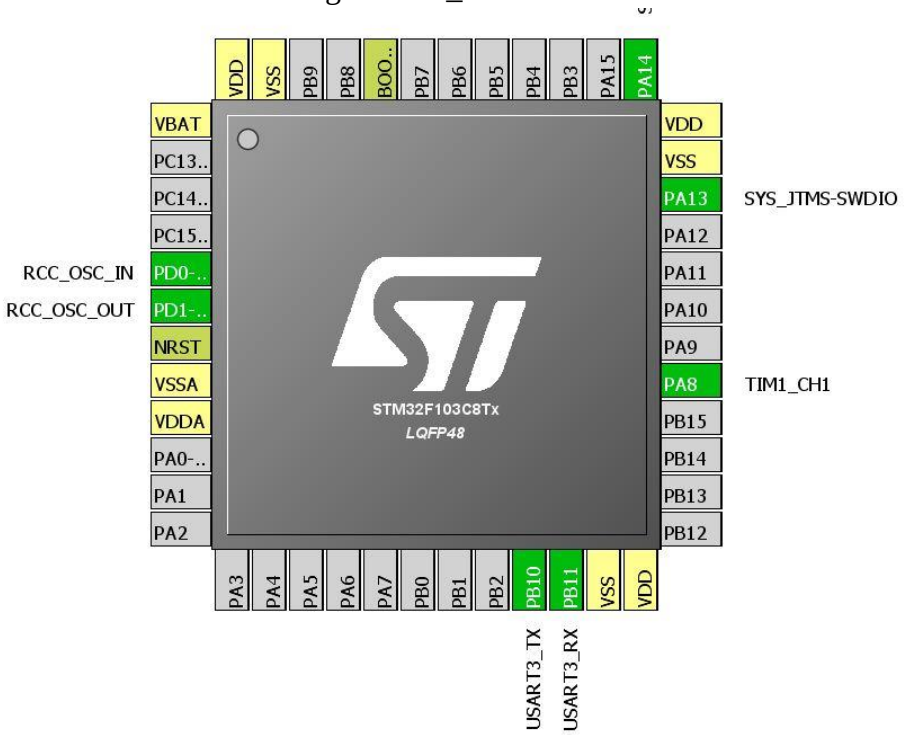
6.1. Buat project baru, atur SYS, RCC, HCLK dan USART3 seperti pada langkah 3.1 hingga 3.5

6.2. Pada STM32F103C8T6 terdapat 4 buah timer, pada tutorial ini kita akan menggunakan timer 1

6.3. Expand panel TIM1. Pada kolom Clock Source pilih Internal Clock. Pilih Input Capture Direct Mode pada kolom Channel yang diinginkan, pada contoh ini saya menggunakan Channel1.

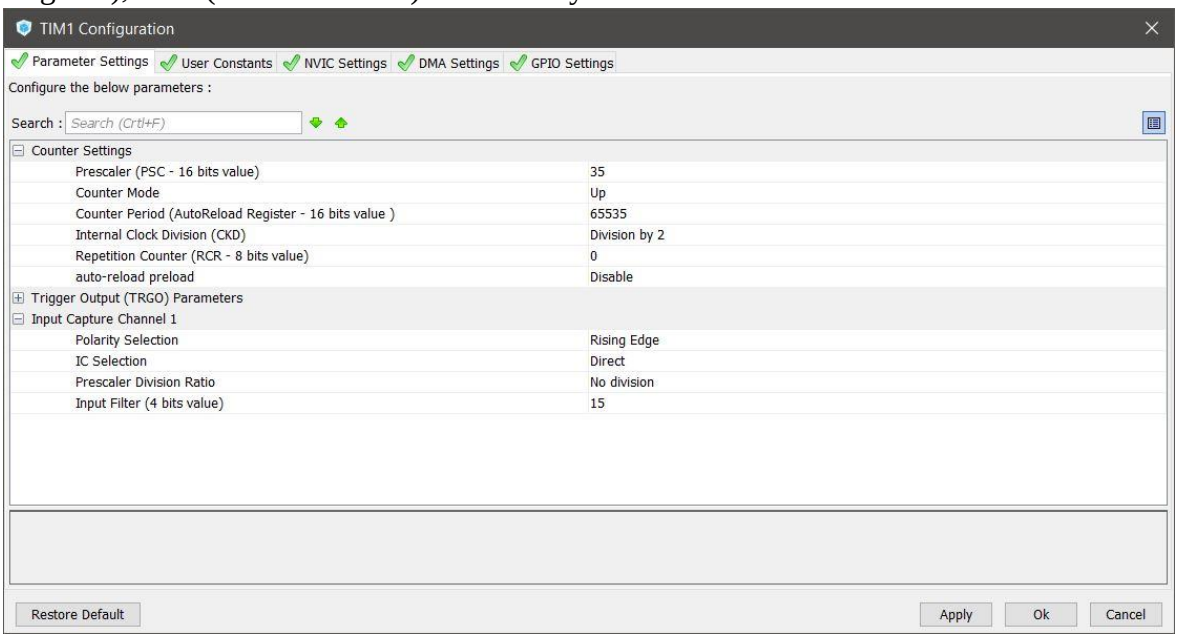


Ketika kita mengaktifkan Input Capture Direct Mode pada Channel1 maka pin A8 akan secara otomatis terinisialisasi sebagai TIM1_CH1.



6.4. Buka tab Configuration. Pada kolom Control click TIM1.

6.5. Seperti pada tutorial yang sebelumnya, pada jendela TIM1 Configuration kita perlu mengatur beberapa parameter dari TIM1 diantaranya; PSC(Prescaler), ARR(Auto Reload Register), CKD(Clock Division) dan Polarity.



Nilai – nilai parameter tersebut bisa didapatkan dari beberapa rumus berikut.

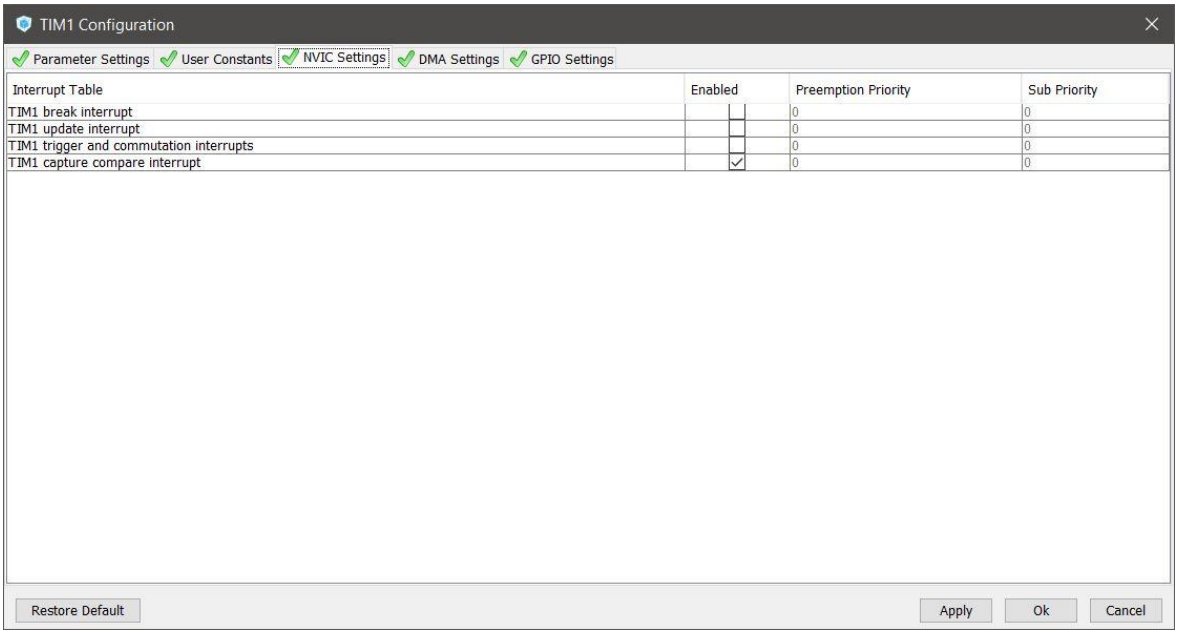
$$PeriodeMaksimumInput(S) = \frac{(ARR + 1)(PSC + 1)(CKD)}{TIMxClockSource\ x\ 2}$$

$$PeriodeInput(S) = \frac{(CCRx + 1)(PSC + 1)(CKD)}{TIMxClockSource\ x\ 2}$$

$$Ketelitian(\pm S) = \frac{(PSC + 1)(CKD)}{TIMxClockSource}$$

Parameter Polarity Selection digunakan untuk menentukan kapan Input Capture menghasilkan IRQ. Jika kita memilih rising edge maka IRQ akan dihasilkan pada rising edge sinyal input.

6.6. Selanjutnya buka tab NVIC(Nested Vector Interrupt Controller) setting pada jendela TIM1 Configuration



Pada tab ini kita akan melakukan pengaturan pada NVIC. Pada tab ini kita hanya perlu mengaktifkan TIM1 capture compare interrupt.

6.7. Pada tahap ini pengaturan mikrokontroler telah selesai. Kita hanya perlu me-generate dan membuka project yang dihasilkan seperti yang sudah dijelaskan pada langkah 2.10 hingga 2.11

6.8. Pada project ini program akan dibagi menjadi tiga bagian yaitu definisi variable global, program utama dan ISR.

Bagian pertama dari program adalah definisi variable global. Variabel global didefinisikan diluar fungsi main(). Variable IC_value adalah variable yang akan menyimpan periode gelombang input. Karena variable IC_value juga akan digunakan pada ISR maka perlu ditambahkan kata kunci “volatile”. Perlu diperhatikan definisi variable global juga harus terletak diantara /* USER CODE BEGIN xxxxx */ dan /* USER CODE END xxxxx */.

```
68 /* USER CODE BEGIN 0 */
69 volatile uint32_t IC_value;
70 /* USER CODE END 0 */
```

Program utama diawali dengan fungsi:

```
HAL_StatusTypeDef HAL_TIM_IC_Start (TIM_HandleTypeDef *htim, uint32_t Channel)
```

Fungsi tersebut digunakan untuk memulia InputCapture channel1 pada timer1. Selanjutnya kita akan menampilkan data frekuensi dan periode pada serial monitor.

```
108  /* USER CODE BEGIN WHILE */
109  uint8_t uart_send_buffer[30];
110
111  HAL_TIM_IC_Start_IT(&htim1, TIM_CHANNEL_1);
112  while (1)
113  {
114      uint16_t f = 1000000/IC_value;
115      sprintf((char*)uart_send_buffer, "frekuensi = %dHz, Periode = %duS\r\n", f, IC_value);
116      HAL_UART_Transmit(&huart3, uart_send_buffer, strlen((char*)uart_send_buffer), 50);
117      HAL_Delay(200);
118  /* USER CODE END WHILE */
```

Pada bagian terakhir, kita akan mendefinisikan ISR. Pada fungsi ISR kita akan melakukan dua hal, yang pertama kita perlu me-reset counter pada timer 1 agar tidak terjadi overflow, dan yang kedua kita menyimpan periode gelombang pada variable IC_value.

```
/* USER CODE BEGIN 4 */
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim){
    if(htim->Instance == TIM1){
        __HAL_TIM_SET_COUNTER(&htim1, 0);
        IC_value = __HAL_TIM_GET_COMPARE(&htim1, TIM_CHANNEL_1)/2;
    } /* if( htim->Instance == TIM1 ) */
} /* HAL_TIM_IC_Ca...llback( htim ) */

/* USER CODE END 4 */
```

6.9. Upload program pada mikrokontroler seperti pada langkah 2.14 hingga 2.15

6.10. Hubungkan mikrokontroler dengan PC seperti pada langkah 3.8 hingga 3.11