

Harici Led Yakma

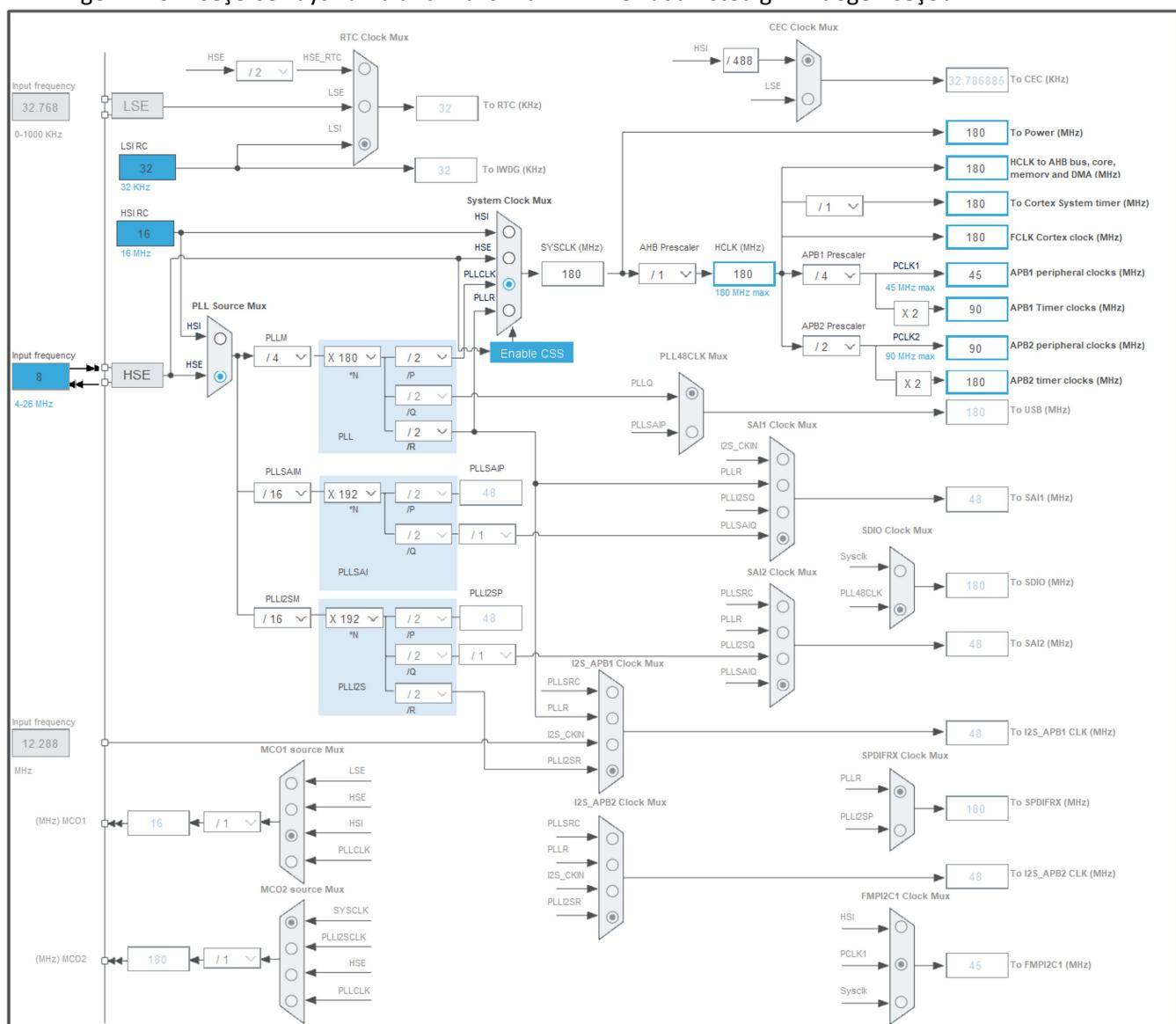
25 Aralık 2021 Cumartesi 00:52

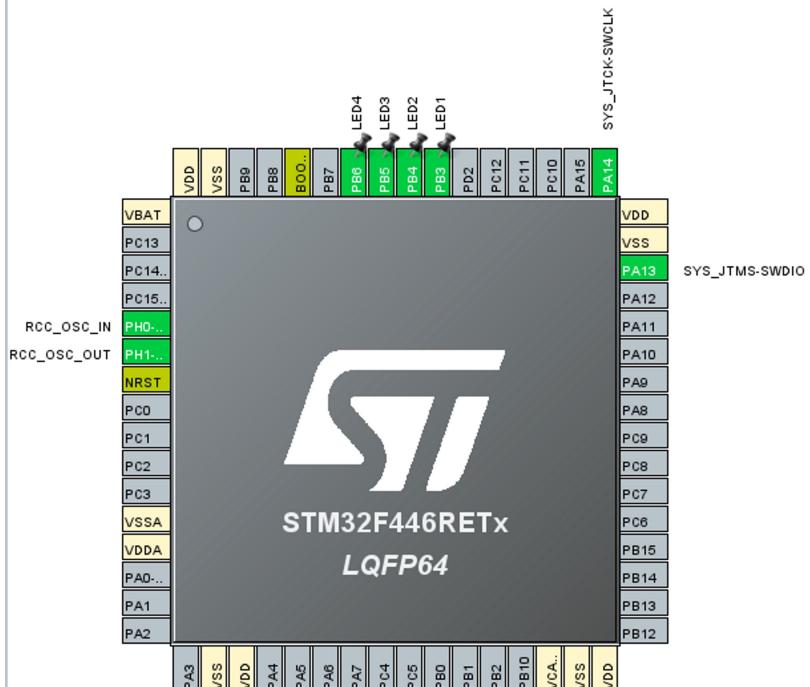
Harici Led Yakma

➤ HAL

Konfigürasyon Kısmı

- Bir saniye içerisinde ne kadar talimat çalıştırıysa frekansı o kadardır.
- Sistem saatini ayarlamak için öncelikle RCC kısmından HSE'den Crysal/Ceramic Resonator seçilir. Bu seçim işlemi ile beraber Clock Configuration da kutucuk bölme açılır. LSE'nin HSE ile aralarındaki farkı düşük hızlı olması ve düşük güç tüketimi için kullanılır.
- System Clock Mux kısmında 3 seçenekimiz var. Bunlar HSI, HSE ve PLLCLK'tur. HSI dahili iken HSE harici kaynaktır.
- HSI seçersek kutucuktaki değer olur, HSE seçersek giriş frekansı belli MHz aralıktadır. Biz burada kart üzerinde 8MHz olduğundan bu şekilde yapıp kullanıyoruz.
- Eğer PLLCLK seçersek ayarlamalarla maksimum MHz'e kadar istediğimiz değeri seçebiliriz.





- System Core kısmından SYS tıklıyoruz ve Debug'da Serial Wire diyoruz.

| Pin Name | Signal on Pin | GPIO output ... | GPIO mode | GPIO Pull-u... | Maximum o... | User Label | Modified |
|----------|----------------|-----------------|-----------|----------------|--------------|------------|--------------------------|
| PA13 | SYS_JTMS-SWDIO | n/a | n/a | n/a | n/a | | <input type="checkbox"/> |
| PA14 | SYS_JTCK-SWCLK | n/a | n/a | n/a | n/a | | <input type="checkbox"/> |

- RCC tıklıyoruz HSE'de Crystal/Ceramic Resonator işaretliyoruz.

| Pin Name | Signal on Pin | GPIO output... | GPIO mode | GPIO Pull-u... | Maximum ou... | User Label | Modified |
|-------------|---------------|----------------|-----------|----------------|---------------|------------|--------------------------|
| PH0-OSC_IN | RCC_OSC_IN | n/a | n/a | n/a | n/a | | <input type="checkbox"/> |
| PH1-OSC_OUT | RCC_OSC_OUT | n/a | n/a | n/a | n/a | | <input type="checkbox"/> |

- B portundaki 3., 4., 5. ve 6.pinlere led bağladık.

| Pin Name | Signal | GPIO o... | GPIO mode | GPIO Pull-up/Pull-do... | Maximum output ... | User Label | Modified |
|----------|--------|-----------|-----------------|--------------------------|--------------------|------------|-------------------------------------|
| PB3 | n/a | Low | Output Push ... | No pull-up and no pul... | Very High | LED1 | <input checked="" type="checkbox"/> |
| PB4 | n/a | Low | Output Push ... | No pull-up and no pul... | Very High | LED2 | <input checked="" type="checkbox"/> |
| PB5 | n/a | Low | Output Push ... | No pull-up and no pul... | Very High | LED3 | <input checked="" type="checkbox"/> |
| PB6 | n/a | Low | Output Push ... | No pull-up and no pul... | Very High | LED4 | <input checked="" type="checkbox"/> |

Kod Kısmı

- CubeMX'de ayarlamaları yaptıktan sonra main.c kısmında yaptığımız kısımları kendisi kod içerisinde oluşturmuştur.
 - RCC ve GPIO için iki ayrı fonksiyon oluşturmuştur.
 - Oluşturulan iki fonksiyon kod içerisinde yazmadan önce private function prototypes kısmında belirtilir.

```
48 /* Private function prototypes ----- */
```

49 void SystemClock_Config(void);

```
50 static void MX_GPIO_Init(void);
```

- Daha sonra bu iki fonksiyon kod içerişine dahil edilir.

```

64@ int main(void)
65 {
66     /* USER CODE BEGIN 1 */
67
68     /* USER CODE END 1 */
69
70     /* MCU Configuration-----*/
71
72     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
73     HAL_Init();
74
75     /* USER CODE BEGIN Init */
76
77     /* USER CODE END Init */
78
79     /* Configure the system clock */
80     SystemClock_Config();
81
82     /* USER CODE BEGIN SysInit */
83
84     /* USER CODE END SysInit */
85
86     /* Initialize all configured peripherals */
87     MX_GPIO_Init();
88     /* USER CODE BEGIN 2 */
89
90     /* USER CODE END 2 */
91
92     /* Infinite loop */
93     /* USER CODE BEGIN WHILE */
94     while (1)
95     {
96         /* USER CODE END WHILE */
97
98         /* USER CODE BEGIN 3 */
99     }
100    /* USER CODE END 3 */
101 }

```

- hal_gpio.c kısmından WritePin ve Toggle Pin fonksiyonlarına ulaşabiliriz.

```

410@void HAL_GPIO_WritePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)
411 {
412     /* Check the parameters */
413     assert_param(IS_GPIO_PIN(GPIO_Pin));
414     assert_param(IS_GPIO_PIN_ACTION(PinState));
415
416     if(PinState != GPIO_PIN_RESET)
417     {
418         GPIOx->BSRR = GPIO_Pin;
419     }
420     else
421     {
422         GPIOx->BSRR = (uint32_t)GPIO_Pin << 16U;
423     }
424 }

```

```

433@void HAL_GPIO_TogglePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
434 {
435     uint32_t odr;
436
437     /* Check the parameters */
438     assert_param(IS_GPIO_PIN(GPIO_Pin));
439
440     /* get current Output Data Register value */
441     odr = GPIOx->ODR;
442
443     /* Set selected pins that were at low level, and reset ones that were high */
444     GPIOx->BSRR = ((odr & GPIO_Pin) << GPIO_NUMBER) | (~odr & GPIO_Pin);
445 }

```

- Döngümüze aşağıdaki kodu yazarız.

```

94     while (1)
95     {
96         /* USER CODE END WHILE */
97
98         /* USER CODE BEGIN 3 */
99         HAL_GPIO_WritePin(GPIOB, LED1_Pin | LED2_Pin | LED3_Pin | LED4_Pin, GPIO_PIN_SET);
100         HAL_Delay(2000);
101         HAL_GPIO_TogglePin(GPIOB, LED1_Pin | LED2_Pin | LED3_Pin | LED4_Pin);
102         HAL_Delay(500);

```

```

94     while (1)
95     {
96         /* USER CODE END WHILE */
97
98         /* USER CODE BEGIN 3 */
99         HAL_GPIO_WritePin(GPIOB, LED1_Pin | LED2_Pin | LED3_Pin | LED4_Pin, GPIO_PIN_SET);
100        HAL_Delay(2000);
101        HAL_GPIO_TogglePin(GPIOB, LED1_Pin | LED2_Pin | LED3_Pin | LED4_Pin);
102        HAL_Delay(500);
103    }

```

Programlama Kısmı

- STM32CubeIDE üzerinden hex dosyası oluşturmak için <https://www.youtube.com/watch?v=7FpsxwEXfLc> link üzerinden bilgi alabiliriz.
- Proje dosyasına sağ tıklayıp Properties tıklarız. Burada C/C++ Build sekmesinde Setting tıklarız ve burada Build Steps sekmesine geliriz. Burada Post-build steps kısmında Command kutucوغuna "arm-none-eabi-objcopy -O ihex \${ProjName}.elf \${ProjName}.hex" yazarız. Bu kod ile .elf dosyasını .hex dosyasına çeviririz.
- Hex dosyasını yüklemek için STM32 ST-LINK Utility ya da STM32Cube Programmer programlarını kullanabiliriz. Eğer STM32Cube Programmer programını kullanıyorsak kodu yükledikten sonra işlemci üzerindeki reset tuşuna basarak yüklediğimiz kod çalıştırırız.
- Eğer STM32Cube Programmer programını kullanıyorsak kodu yükledikten sonra işlemci üzerindeki reset tuşuna basarak yüklediğimiz kod çalıştırırız.

Git Kısmı

- STM32CubeIDE üzerinden oluşturduğumuz projeyi git ile kullanmak için https://www.youtube.com/watch?v=_96FSH7uIOE linkteki videoya ve <https://shadyelectronics.com/how-to-use-github-with-stm32cubeide/> linkteki makaleye göz atabiliriz.

➤ REGISTER

Konfigürasyon Kısmı

- main.c dosyasındaki yorum satırları silip sadece hale getiriyoruz.

```

1 #include "stm32f4xx.h"
2
3 int main(void)
4 {
5     while (1)
6     {
7
8     }
9 }

```

- system_stmf4xx.c dosyasındaki 182.satırda SystemCoreClock kısmına Ctrl ile sağ tıklıyoruz ve bizi system_stmf4xx.h dosyasındaki 59.satırı götürüyor. Bu satırı kopyalayıp main.c dosyasına yapıştırıyoruz.

```
182     uint32_t SystemCoreClock = 168000000;
```

```
59 extern uint32_t SystemCoreClock;           /*!< System Clock Frequency (Core Clock) */
```

```

1 #include "stm32f4xx.h"
2
3 extern uint32_t SystemCoreClock;
4
5 uint32_t systemClock;
6
7 int main(void)
8 {
9     systemClock=SystemCoreClock;
10
11     while (1)
12     {
13
14     }
15 }
```

| Expression | Type | Value |
|------------------|----------|-----------|
| (*)= systemClock | uint32_t | 168000000 |

```

1 #include "stm32f4xx.h"
2
3 extern uint32_t SystemCoreClock;
4
5 uint32_t systemClock;
6
7 int main(void)
8 {
9     systemClock=SystemCoreClock;           //168 000 000
10
11     RCC_DeInit();                      //HSI ON PLL OFF
12
13     SystemCoreClockUpdate();
14     systemClock=SystemCoreClock;         //16 000 000
15
16     while (1)
17     {
18
19     }
20 }
```

| Expression | Type | Value |
|------------------|----------|-----------|
| (x)= systemClock | uint32_t | 16000000\ |

- STM32F407VG mikrodenetleyicinin Reference Manuals'de RCC kısmına bakıyoruz. Mikrodenetleyici 32 bittir. Bu sebeple kaydediciler olan registerler 32 bittir.
- Bit değerlerinde yazan r ile rw'nin anlamı r'nin read yani okunabilir, w'nin write yani yazılabılır anlamı vardır. r olan yerlere müdahale edemiyoruz ama rw yazan bitlere müdahale edebiliyoruz.

RCC clock control register (RCC_CR)

Address offset: 0x00

Reset value: 0x0000 XX83 where X is undefined.

Access: no wait state, word, half-word and byte access

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
|-------------|----|----|------------|-----------|---------|-------|----------|--------------|----|--------|---------|---------|--------|------|---------|-------|
| Reserved | | | PLL12S RDY | PLL12S ON | PLLRD Y | PLLON | Reserved | | | CSS ON | HSE BYP | HSE RDY | HSE ON | | | |
| | | | r | rw | r | rw | | | | rw | rw | r | rw | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| HSICAL[7:0] | | | | | | | | HSITRIM[4:0] | | | | | | Res. | HSI RDY | HSION |
| r | r | r | r | r | r | r | r | rw | rw | rw | rw | rw | r | rw | | |

- Adress offset bizim ilk olarak kaydedilen yeri gösterir.
Reset value ise girilen değer sonrası tüm bitler sıfırlanır.
- Burada resetleme işlemi yapıyoruz.

RCC->CR &= 0x00000083;

- "&=" anlam kaydet anlamındadır.
- Harici osilatör kullanacağımdan HSEON olan 16.biti 1 yapmam gerekiyor.

Bit 16 **HSEON**: HSE clock enable

Set and cleared by software.

Cleared by hardware to stop the HSE oscillator when entering Stop or Standby mode. This bit cannot be reset if the HSE oscillator is used directly or indirectly as the system clock.

0: HSE oscillator OFF

1: HSE oscillator ON

- "|= anlamı ise öncekine 1 ekle ve eşitle anlamındadır. Bunu kullanırken eklemeye yani ötelemeye yapıyoruz.
- 16.biti 1 yapmak için ötelemeye kullanırken 1 << 16 yazıyoruz yani 0.bitte 1 yazıp 16 bit öteleiyor.

RCC->CR |= 1 << 16;

Bunu denetlemek için 17.biti kullanıyoruz. Yani 16.bit 1 olup olmadığını HSE osilatör 6 clock cycles yaptıktan sonra 17.bit olan HSERDY biti 1 oluyor.

Bit 17 **HSERDY**: HSE clock ready flag

Bit 17 HSERDY: HSE clock ready flag

Set by hardware to indicate that the HSE oscillator is stable. After the HSEON bit is cleared, HSERDY goes low after 6 HSE oscillator clock cycles.

- 0: HSE oscillator not ready
- 1: HSE oscillator ready

- While döngüsü 1 olduğu sürece çalışır. HSERDY biti 1 olana dek 0 olacağından ve döngünün çalışabilmesi için başına "!" işaretini koyarak tersliyoruz.
- 17.bit 1 olduğunda döngüden çıkışacaktır.
- & biti lojik kapılarda olduğu gibi 0&0 ile 0&1 olduğunda çıkışında 0 verir. 1&1 olduğunda çıkışında 1 verir.
- Buradaki $1 \ll 17$ işlemini yapabilmesi için 6 clock cycles zaman geçmesi gerekiyor. Öteleme işlem yaptığını anlamak için RCC->CR 1 olduğunda 17.bit 1 olmuş oluyor. Böylece 1&1'den 1 oluyor ve !1'den 0 olarak döngüden çıkışıyor.

```
while(!(RCC->CR & (1 << 17)));
```

- HSI kapatmak için HSION bitini 0 yapmam gerekiyor.

Bit 0 HSION: Internal high-speed clock enable

Set and cleared by software.

Set by hardware to force the HSI oscillator ON when leaving the Stop or Standby mode or in case of a failure of the HSE oscillator used directly or indirectly as the system clock. This bit cannot be cleared if the HSI is used directly or indirectly as the system clock.

- 0: HSI oscillator OFF
- 1: HSI oscillator ON

- $1 \ll 0$ ile 0.bit 1 yapılır ardından ~ işaretini ile 0 yapılır.

```
RCC->CR &= ~(1 << 0)
```

- Şu an mikrodenetleyici 8 000 000Hz'de çalışıyor. PLL ile 168 000 000Hz'e çıkaracağız.

Bit 19 CSSON: Clock security system enable

Set and cleared by software to enable the clock security system. When CSSON is set, the clock detector is enabled by hardware when the HSE oscillator is ready, and disabled by hardware if an oscillator failure is detected.

0: Clock security system OFF (Clock detector OFF)

1: Clock security system ON (Clock detector ON if HSE oscillator is stable, OFF if not)

- 19.bit 1 yapılır.

```
RCC->CR |= 1 << 19;
```

PLLON 24.bit 1 yapılmadan önce PLL'nin konfigürasyonlarını yapmamız gerekiyor.

RCC PLL configuration register (RCC_PLLCFGR)

Address offset: 0x04

Reset value: 0x2400 3010

Access: no wait state, word, half-word and byte access.

This register is used to configure the PLL clock outputs according to the formulas:

- $f_{(VCO\ clock)} = f_{(PLL\ clock\ input)} \times (PLL_N / PLL_M)$
- $f_{(PLL\ general\ clock\ output)} = f_{(VCO\ clock)} / PLL_P$
- $f_{(USB\ OTG\ FS,\ SDIO,\ RNG\ clock\ output)} = f_{(VCO\ clock)} / PLL_Q$

| | | | | | | | | | | | | | | | |
|----------|------|----|-------|-------|-------|-------|----------|--------|----------|-------|-------|--------|--------|-------|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | PLLQ3 | PLLQ2 | PLLQ1 | PLLQ0 | Reserved | PLLSRC | Reserved | | | PLL_P1 | PLL_P0 | | |
| | | | rw | rw | rw | rw | | rw | | | | rw | rw | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | PLLN | | | | | | | | PLLM5 | PLLM4 | PLLM3 | PLLM2 | PLLM1 | PLLM0 | |
| | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | |

- PLL için M, N ve P değerlerine göre sistem saatı 168MHz yapıyoruz. Bunun için M değerine 4, N değerine 168 ve P değerine 2 verdigimizde bu işlemi sağlamış oluyoruz.
- M için PLLM kısmında ilk 6 bitte yazacağımız 000100 değerini 0., 1., 3., 4. ve 5.bite 0 yapıyorken 2.biti 1 yapıyoruz.

2.biti 1 yaparken ekleme yaparak yani "|=" işaretini kullanırken diğerlerine sadece 0 bite kaydetmek için

"&=" işaretini kullanıyoruz.

Bits 5:0 **PLLM**: Division factor for the main PLL (PLL) and audio PLL (PLLI2S) input clock

Set and cleared by software to divide the PLL and PLLI2S input clock before the VCO.

These bits can be written only when the PLL and PLLI2S are disabled.

Caution: The software has to set these bits correctly to ensure that the VCO input frequency ranges from 1 to 2 MHz. It is recommended to select a frequency of 2 MHz to limit PLL jitter.

VCO input frequency = PLL input clock frequency / PLLM with $2 \leq PLLM \leq 63$

000000: PLLM = 0, wrong configuration

000001: PLLM = 1, wrong configuration

000010: PLLM = 2

000011: PLLM = 3

000100: PLLM = 4

...

111110: PLLM = 62

111111: PLLM = 63

```
RCC->PLLCFGR &= ~(1 << 0);          //PLLM0 0
RCC->PLLCFGR &= ~(1 << 1);          //PLLM1 0
RCC->PLLCFGR |= (1 << 2);           //PLLM2 1
RCC->PLLCFGR &= ~(1 << 3);          //PLLM3 0
RCC->PLLCFGR &= ~(1 << 4);          //PLLM4 0
RCC->PLLCFGR &= ~(1 << 5);          //PLLM5 0
```

- N için PLLN kısmını kullanıyoruz.

Bits 14:6 **PLLN**: Main PLL (PLL) multiplication factor for VCO

Set and cleared by software to control the multiplication factor of the VCO. These bits can be written only when PLL is disabled. Only half-word and word accesses are allowed to write these bits.

Caution: The software has to set these bits correctly to ensure that the VCO output frequency is between 100 and 432 MHz.

VCO output frequency = VCO input frequency \times PLLN with $50 \leq PLLN \leq 432$

00000000: PLLN = 0, wrong configuration

00000001: PLLN = 1, wrong configuration

...

0001100010: PLLN = 50

...

0011000011: PLLN = 99

001100100: PLLN = 100

...

110110000: PLLN = 432

110110001: PLLN = 433, wrong configuration

...

111111111: PLLN = 511, wrong configuration

Note: Multiplication factors ranging from 50 and 99 are possible for VCO input frequency higher than 1 MHz. However care must be taken that the minimum VCO output frequency respects the value specified above.

| | |
|-----|-----------|
| HEX | A8 |
| DEC | 168 |
| OCT | 250 |
| BIN | 1010 1000 |

- 168 decimal sayısının binary karşılığı 1010 1000'dir. Biz burada 9 bit olduğundan 010101000 yazacağız.
Biz bununla uğraşmak yerine 6 bit öteleyip 168 sayısını yazacağız.

```
RCC->PLLCFGR |= (168 << 6);          //PLLN 168
```

Aynı işlemi M için de yapabiliriz.

```
RCC->PLLCFGR |= (4 << 0);          //PLLM 4
```

P için PLLP kısmını kullanıyoruz. 2 değeri için iki biti 0 yap diyor.

Bits 17:16 **PLLP**: Main PLL (PLL) division factor for main system clock

Set and cleared by software to control the frequency of the general PLL output clock. These bits can be written only if PLL is disabled.

Caution: The software has to set these bits correctly not to exceed 168 MHz on this domain.

PLL output clock frequency = VCO frequency / PLLP with PLLP = 2, 4, 6, or 8

- 00: PLLP = 2
- 01: PLLP = 4
- 10: PLLP = 6
- 11: PLLP = 8

```
RCC->PLLCFGR &= ~(1 << 16);           //PLLP1 0  
RCC->PLLCFGR &= ~(1 << 17);           //PLLP2 0
```

- PLL için başlangıçta tüm bitleri 0 yapıyoruz.
- Burada 8 tane sıfır var. Buradaki her biri 32 bit'de 4 bite karşılık geliyor.

```
RCC->CFGR = 0x00000000;
```

- Tüm bitleri başlangıçta 0 yaptığımız için PLL'de P için yaptığımız 0'lama işlemine gerek kalmadı.

```
7 void RCC_Config(void)  
8 {  
9     //RCC->CR &= 0x00000083;           //RESET  
10  
11    RCC->CR &= ~(1 << 0);          //HSION  
12    RCC->CR |= 1 << 16;             //HSEON  
13    while(!(RCC->CR & (1 << 17))); //HSERDY  
14    RCC->CR |= 1 << 19;             //CSSON  
15    RCC->CFGR = 0x00000000;  
16    //RCC->PLLCFGR &= ~(1 << 0);    //PLLM0 0  
17    //RCC->PLLCFGR &= ~(1 << 1);    //PLLM1 0  
18    //RCC->PLLCFGR |= (1 << 2);      //PLLM2 1  
19    //RCC->PLLCFGR &= ~(1 << 3);    //PLLM3 0  
20    //RCC->PLLCFGR &= ~(1 << 4);    //PLLM4 0  
21    //RCC->PLLCFGR &= ~(1 << 5);    //PLLM5 0  
22    RCC->PLLCFGR |= (4 << 0);       //PLLM 4  
23    RCC->PLLCFGR |= (168 << 6);     //PLLN 168  
24    //RCC->PLLCFGR &= ~(1 << 16);   //PLLP1 0  
25    //RCC->PLLCFGR &= ~(1 << 17);   //PLLP2 0  
26 }
```

| Register | Address | Value |
|----------|------------|---------|
| > DMA1 | | |
| RCC | | |
| CR | 0x40023800 | 0xb7083 |
| PLLCFGR | 0x40023804 | 0xa04 |
| PLLQ3 | [27:1] | 0x0 |
| PLLQ2 | [26:1] | 0x0 |
| PLLQ1 | [25:1] | 0x0 |
| PLLQ0 | [24:1] | 0x0 |
| PLLSRC | [22:1] | 0x0 |
| PLLP1 | [17:1] | 0x0 |
| PLLP0 | [16:1] | 0x0 |
| PLLN8 | [14:1] | 0x0 |
| PLLN7 | [13:1] | 0x1 |
| PLLN6 | [12:1] | 0x0 |
| PLLN5 | [11:1] | 0x1 |
| PLLN4 | [10:1] | 0x0 |
| PLLN3 | [9:1] | 0x1 |
| PLLN2 | [8:1] | 0x0 |
| PLLN1 | [7:1] | 0x0 |
| PLLN0 | [6:1] | 0x0 |
| PLLM5 | [5:1] | 0x0 |
| PLLM4 | [4:1] | 0x0 |
| PLLM3 | [3:1] | 0x0 |
| PLLM2 | [2:1] | 0x1 |
| PLLM1 | [1:1] | 0x0 |
| PLLM0 | [0:1] | 0x0 |

- PLLSRC ile PLL sürücüsü seçilir. Bu bit için HSI kullanırsak 0, HSE kullanırsak 1 yapılır. Biz HSE kullandığımızdan bu biti 1 yapıyoruz.

Bit 22 **PLLSRC**: Main PLL(PLL) and audio PLL (PLLI2S) entry clock source

Set and cleared by software to select PLL and PLLI2S clock source. This bit can be written only when PLL and PLLI2S are disabled.

0: HSI clock selected as PLL and PLLI2S clock entry

1: HSE oscillator clock selected as PLL and PLLI2S clock entry

RCC->**CFGR |= (1 << 22);**

Bit 24 **PLLON**: Main PLL (PLL) enable

Set and cleared by software to enable PLL.

Cleared by hardware when entering Stop or Standby mode. This bit cannot be reset if PLL clock is used as the system clock.

0: PLL OFF

1: PLL ON

RCC->**CR |= 1 << 24;**

Bit 25 **PLLRDY**: Main PLL (PLL) clock ready flag

Set by hardware to indicate that PLL is locked.

0: PLL unlocked

1: PLL locked

while(! (RCC->CR & (1 << 25)));

RCC clock configuration register (RCC_CFGR)

Address offset: 0x08

Reset value: 0x0000 0000

Access: 0 ≤ wait state ≤ 2, word, half-word and byte access

1 or 2 wait states inserted only if the access occurs during a clock source switch.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------------|----|---------------|------------|----|---------------|----------|-----------|--------|------|----|-------------|------|-----|-----|----|
| MCO2 | | MCO2 PRE[2:0] | | | MCO1 PRE[2:0] | | | I2SSCR | MCO1 | | RTCPRE[4:0] | | | | |
| rw | | rw | rw | rw | rw | rw | rw | rw | rw | | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PPRE2[2:0] | | | PPRE1[2:0] | | | Reserved | HPRE[3:0] | | | | SWS1 | SWS0 | SW1 | SW0 | |
| rw | rw | rw | rw | rw | rw | | rw | rw | rw | rw | r | r | rw | rw | |

- İlk önce 0.bit 0 yapıldı daha sonra 1.bit 1 yapılarak sistemin saatini PLL ile ayarladığımızı belirtiyoruz.

Bits 1:0 **SW**: System clock switch

Set and cleared by software to select the system clock source.

Set by hardware to force the HSI selection when leaving the Stop or Standby mode or in case of failure of the HSE oscillator used directly or indirectly as the system clock.

00: HSI oscillator selected as system clock

01: HSE oscillator selected as system clock

10: PLL selected as system clock

11: not allowed

RCC->**CFGR** &= ~(1 << 0);

RCC->**CFGR** |= (1 << 1);

Bits 3:2 **SWS**: System clock switch status

Set and cleared by hardware to indicate which clock source is used as the system clock.

00: HSI oscillator used as the system clock

01: HSE oscillator used as the system clock

10: PLL used as the system clock

11: not applicable

while(!(RCC->**CR** & (1 << 1)));

```

7 @void RCC_Config(void)
8 {
9     //RCC->CR &= 0x00000083;           //RESET
10
11    RCC->CR &= ~(1 << 0);          //HSION
12    RCC->CR |= 1 << 16;            //HSEON
13    while(!(RCC->CR & (1 << 17))); //HSERDY
14    RCC->CR |= 1 << 19;            //CSSON
15    RCC->CFGR = 0x00000000;
16    RCC->PLLCFGR |= (1 << 22);    //PLLCSR
17    //RCC->PLLCFGR &= ~(1 << 0); //PLLMO 0
18    //RCC->PLLCFGR &= ~(1 << 1); //PLLM1 0
19    //RCC->PLLCFGR |= (1 << 2); //PLLM2 1
20    //RCC->PLLCFGR &= ~(1 << 3); //PLLM3 0
21    //RCC->PLLCFGR &= ~(1 << 4); //PLLM4 0
22    //RCC->PLLCFGR &= ~(1 << 5); //PLLM5 0
23    RCC->PLLCFGR |= (4 << 0);    //PLLM 4
24    RCC->PLLCFGR |= (168 << 6); //PLLN 168
25    //RCC->PLLCFGR &= ~(1 << 16); //PLLP1 0
26    //RCC->PLLCFGR &= ~(1 << 17); //PLLP2 0
27
28    RCC->CR |= 1 << 24;           //PLLON
29    while(!(RCC->CR & (1 << 25))); //PLLRDY
30
31    RCC->CFG &= ~(1 << 0);
32    RCC->CFG |= (1 << 1);        //SW
33
34    while(!(RCC->CR & (1 << 1))); //SWS
35 }
36 @int main(void)
37 {
38     //systemClock=SystemCoreClock;   //168 000 000
39
40     //RCC_DeInit();                //HSI ON PLL OFF
41
42     //SystemCoreClockUpdate();
43     //systemClock=SystemCoreClock;   //16 000 000
44
45     RCC_Config();
46     SystemCoreClockUpdate();
47     systemClock=SystemCoreClock;   //168 000 000
48
49     while (1)
50     {
51
52     }
53 }
54 }
```

| Expression | Type | Value |
|------------------|----------|-----------|
| (x)= systemClock | uint32_t | 168000000 |

GPIO port mode register (GPIOx_MODER) ($x = A..I/J/K$)

Address offset: 0x00

Reset values:

- 0xA800 0000 for port A
- 0x0000 0280 for port B
- 0x0000 0000 for other ports

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|--------------|--------------|--------------|--------------|--------------|--------------|-------------|-------------|----|----|----|----|----|----|----|----|
| MODER15[1:0] | MODER14[1:0] | MODER13[1:0] | MODER12[1:0] | MODER11[1:0] | MODER10[1:0] | MODER9[1:0] | MODER8[1:0] | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MODER7[1:0] | MODER6[1:0] | MODER5[1:0] | MODER4[1:0] | MODER3[1:0] | MODER2[1:0] | MODER1[1:0] | MODER0[1:0] | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

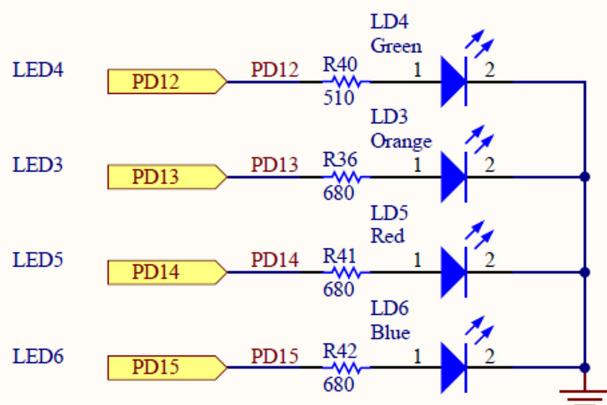
- Pinlerin nasıl kullanacağımızı seçtiğimiz kısımdır. Her iki bit bir pini temsil ediyor.
- Pini giriş, çıkış, analog ya da alternatif fonksiyon olarak mı kullanacağımızı belirtiyoruz. Alternatif fonksiyon ile kast edilen pinin çevresel birimlerden I2C, SPI olarak kullanılacağını belirtiyor.

Bits 2y:2y+1 **MODERy[1:0]**: Port x configuration bits ($y = 0..15$)

These bits are written by software to configure the I/O direction mode.

- 00: Input (reset state)
- 01: General purpose output mode
- 10: Alternate function mode
- 11: Analog mode

- STM32f407VG mikrodenetleyicisinin board üzerindeki led pinleri D12, D13, D14 ve D15'tir.



LEDs

- D portundaki belirlediğimiz pinlerimizi output yani çıkış olarak tanımlıyoruz ama öncesinde D portunu clock hattına aktarmamız gerekiyor.

```

GPIOD->MODER |= 1 << 24;           //PD12
GPIOD->MODER &= ~(1 << 25);
GPIOD->MODER |= 1 << 26;           //PD13
GPIOD->MODER &= ~(1 << 27);
GPIOD->MODER |= 1 << 28;           //PD14
GPIOD->MODER &= ~(1 << 29);
GPIOD->MODER |= 1 << 30;           //PD15
GPIOD->MODER &= ~(1 << 31);
    
```

- Output ayarı için her bir pine bir bit ayrılmıştır. Kullanacağımız push-pull reset durumunda 0 olduğundan bir ayar yapmamıza gerek yok.

GPIO port output type register (GPIOx_OTYPER) (x = A..I/J/K)

Address offset: 0x04

Reset value: 0x0000 0000

| | | | | | | | | | | | | | | | |
|----------|------|------|------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OT15 | OT14 | OT13 | OT12 | OT11 | OT10 | OT9 | OT8 | OT7 | OT6 | OT5 | OT4 | OT3 | OT2 | OT1 | OT0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 OTy: Port x configuration bits (y = 0..15)

These bits are written by software to configure the output type of the I/O port.

0: Output push-pull (reset state)

1: Output open-drain

- Pinlerin very high speed olmasını istiyoruz. Pinleri 1 yapıyoruz.

GPIO port output speed register (GPIOx_OSPEEDR) (x = A..I/J/K)

Address offset: 0x08

Reset values:

- 0x0C00 0000 for port A
- 0x0000 00C0 for port B
- 0x0000 0000 for other ports

| | | | | | | | | | | | | | | | |
|-----------------|----|-----------------|----|-----------------|----|-----------------|----|-----------------|----|-----------------|----|----------------|----|----------------|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| OSPEEDR15 [1:0] | | OSPEEDR14 [1:0] | | OSPEEDR13 [1:0] | | OSPEEDR12 [1:0] | | OSPEEDR11 [1:0] | | OSPEEDR10 [1:0] | | OSPEEDR9 [1:0] | | OSPEEDR8 [1:0] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OSPEEDR7[1:0] | | OSPEEDR6[1:0] | | OSPEEDR5[1:0] | | OSPEEDR4[1:0] | | OSPEEDR3[1:0] | | OSPEEDR2[1:0] | | OSPEEDR1 [1:0] | | OSPEEDR0 1:0] | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 2y:2y+1 OSPEEDRy[1:0]: Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O output speed.

00: Low speed

01: Medium speed

10: High speed

11: Very high speed

Note: Refer to the product datasheets for the values of OSPEEDRy bits versus V_{DD} range and external load.

```

GPIOD->OSPEEDR |= 1 << 24;
GPIOD->OSPEEDR |= 1 << 25;
GPIOD->OSPEEDR |= 1 << 26;
GPIOD->OSPEEDR |= 1 << 27;
GPIOD->OSPEEDR |= 1 << 28;
GPIOD->OSPEEDR |= 1 << 29;
GPIOD->OSPEEDR |= 1 << 30;
GPIOD->OSPEEDR |= 1 << 31;

```

- Bu şekilde tek tek yazmak yerine 32 bitinin son sekize 1 yazarak hex kısmı yazabiliriz.

| | |
|---|---|
| HEX | FF00 0000 |
| DEC | -16.777.216 |
| OCT | 37 700 000 000 |
| BIN | 1111 1111 0000 0000 0000 0000 0000 0000 |
| | WORD DWORD MS M ⁺ |
| 0 | 60 56 52 48 |
| 0 | 44 40 36 32 |
| 1 1 1 1 1 1 1 1 0 | 28 24 20 16 |
| 0 | 12 8 4 0 |

GPIOD->OSPEEDR |= 0xFF000000;

- 8 biti 1 yaptığımızda HEX olarak FF veriri. 24.bite eklemeli şekilde de yapabiliriz.

GPIOD->OSPEEDR |= FF << 24;

- Kullanacağımız no pull-up, no pull-down reset durumunda 0 olduğundan bir ayar yapmamıza gerek yok.

GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A..I/J/K)

Address offset: 0x0C

Reset values:

- 0x6400 0000 for port A
- 0x0000 0100 for port B
- 0x0000 0000 for other ports

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|--------------|--------------|--------------|--------------|--------------|--------------|-------------|-------------|----|----|----|----|----|----|----|----|
| PUPDR15[1:0] | PUPDR14[1:0] | PUPDR13[1:0] | PUPDR12[1:0] | PUPDR11[1:0] | PUPDR10[1:0] | PUPDR9[1:0] | PUPDR8[1:0] | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PUPDR7[1:0] | PUPDR6[1:0] | PUPDR5[1:0] | PUPDR4[1:0] | PUPDR3[1:0] | PUPDR2[1:0] | PUPDR1[1:0] | PUPDR0[1:0] | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 2y:2y+1 PUPDR_y[1:0]: Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O pull-up or pull-down

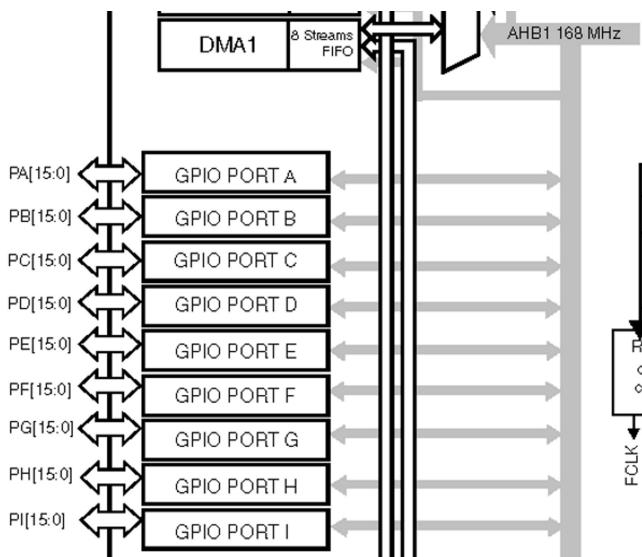
00: No pull-up, pull-down

01: Pull-up

10: Pull-down

11: Reserved

- Öncesinde clock hattını aktif etmemiz gerekiyor. Portlar AHB1 kısmına gidiyor.



RCC AHB1 peripheral clock enable register (RCC_AHB1ENR)

Address offset: 0x30

Reset value: 0x0010 0000

Access: no wait state, word, half-word and byte access.

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
|---------------|-------------------------|-------------|---------------------|--------------------|--------------------|--------------|------------|-------------|-------------|------------|------------------|-------------|---------------|-------------|-------------|----------|--|
| Reser- ved | OTGH S ULPIE N | OTGH SEN | ETHM ACPTP EN | ETHM ACRXE N | ETHM ACTXE N | ETHMA CEN | Reserved | | DMA2E N | DMA1E N | CCMDAT ARAMEN | Res. | BKPSR AMEN | Reserved | | | |
| | rw | rw | rw | rw | rw | rw | | | rw | rw | | | rw | | | | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Reserved | | | CRCE N | Reserved | | | GPIOE N | GPIOH EN | GPIOG EN | GPIOF N | GPIOEEN | GPIOD EN | GPIOC EN | GPIO BEN | GPIO AEN | Reserved | |
| | | | rw | | | | rw | rw | rw | rw | rw | rw | rw | rw | | | |

- Biz D portunu kullandığımızdan sadece bunu aktif ediyoruz.

Bit 3 **GPIODEN**: IO port D clock enable

Set and cleared by software.

0: IO port D clock disabled

1: IO port D clock enabled

RCC->AHB1ENR |= (1 << 3);

- Pinleri set ya da reset edeceğimiz kısımdır. Bunları while döngüsü içerisinde yazıyoruz.

GPIO port output data register (GPIOx_ODR) (x = A..I/J/K)

Address offset: 0x14

Reset value: 0x0000 0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|-------|----------|-------|-------|-------|-------|------|------|------|------|------|------|------|------|------|------|----|
| | Reserved | | | | | | | | | | | | | | | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ODR15 | ODR14 | ODR13 | ODR12 | ODR11 | ODR10 | ODR9 | ODR8 | ODR7 | ODR6 | ODR5 | ODR4 | ODR3 | ODR2 | ODR1 | ODR0 | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODRy**: Port output data (y = 0..15)

These bits can be read and written by software.

Note: For atomic bit set/reset, the ODR bits can be individually set and reset by writing to the GPIOx_BSRR register (x = A..I/J/K).

- ```

GPIOD->ODR |= 1 << 12;
GPIOD->ODR |= 1 << 13;
GPIOD->ODR |= 1 << 14;
GPIOD->ODR |= 1 << 15;

• Clock ayarlarını aşağıdaki gibi düzelttik.
• Önceki yaptıklarımızın hepsini HEX formatında olacak şekilde düzelttik.
• Konunun videosu eski olduğundan önceki yaptığımız örneklerden farklılık olarak eskilerde RCC_CFGR register kısmın 4:7 arası bitler koda eklenmişken yenilerde ekli olan 0:3 arası bitler eklenmemiş.
 Eski videoda döngüde açılan flagların kapatılması için RCC_CIR registerin 11. ve 23.bitleri resetlemiş.

```

```

8 void RCC_Config(void)
9 {
10 RCC->CR |= 0x00030000; //HSEON, HSERDY
11 while(!(RCC->CR & 0x00020000)); //HSERDY
12 RCC->CR |= 0x00080000; //CSSON
13 RCC->CFGR = 0x00000000;
14 RCC->PLLCFGR |= 0x00400000; //PLLSRC
15 RCC->PLLCFGR |= 0x00000004; //PLLM 4
16 RCC->PLLCFGR |= 0x00002A00; //PLLN 168
17 RCC->PLLCFGR |= 0x00000000; //PLLP 2
18 RCC->CR |= 0x01000000; //PLLON
19 while(!(RCC->CR & 0x02000000)); //PLLRDY
20 RCC->CFGR |= 0x00000001; //SW
21 while(!(RCC->CR & 0x00000001)); //SWS
22 }

• GPIO ayarlarını aşağıdaki gibi düzelttik.

19 void GPIO_Config(void)
20 {
21 RCC->AHB1ENR |= (1 << 3); //D clock enable
22
23 //output mode
24 GPIOD->MODER |= 0x55000000; //PD12, PD13, PD14, PD15
25
26 //Very high speed
27 GPIOD->OSPEEDR |= 0xFF000000;
28 }

```

### Kod Kısmı

- RCC ve GPIO için yazdığımız fonksiyonları ekledik ardından while dönüsü içerisinde led blink uygulamasını gerçekleştirdik.

```
36 int main(void)
37 {
38
39 RCC_Config();
40 SystemCoreClockUpdate();
41
42 GPIO_Config();
43
44 while (1)
45 {
46 //set
47 GPIOD->ODR |= 1 << 12;
48 GPIOD->ODR |= 1 << 13;
49 GPIOD->ODR |= 1 << 14;
50 GPIOD->ODR |= 1 << 15;
51
52 for(int i=0; i<1680000; i++);
53
54 //reset
55 GPIOD->ODR &= ~(1 << 12);
56 GPIOD->ODR &= ~(1 << 13);
57 GPIOD->ODR &= ~(1 << 14);
58 GPIOD->ODR &= ~(1 << 15);
59
60 for(int i=0; i<1680000; i++);
61 }
62 }
```