LABS

MCU fundamentals

Clocks

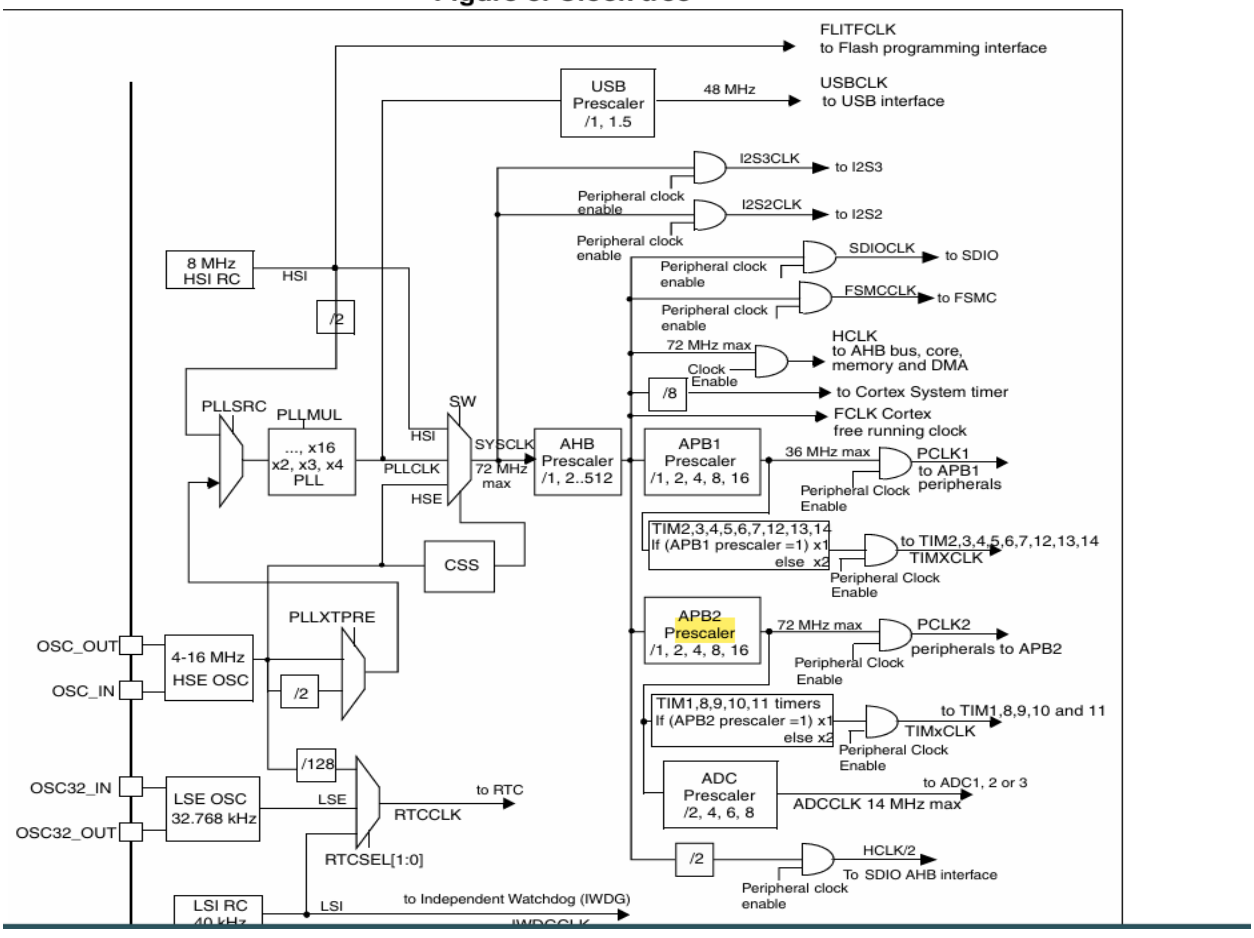
Eng: Ziad El-Sayed Ibrahim

-In this report we will know how to configure the clocks for STM32 for GPIOA peripheral, and we will use different frequences and inputs for the Clock.

-At the first it uses by default 8MHz HSI(high-speed internal clock signal).

-the clock tree:

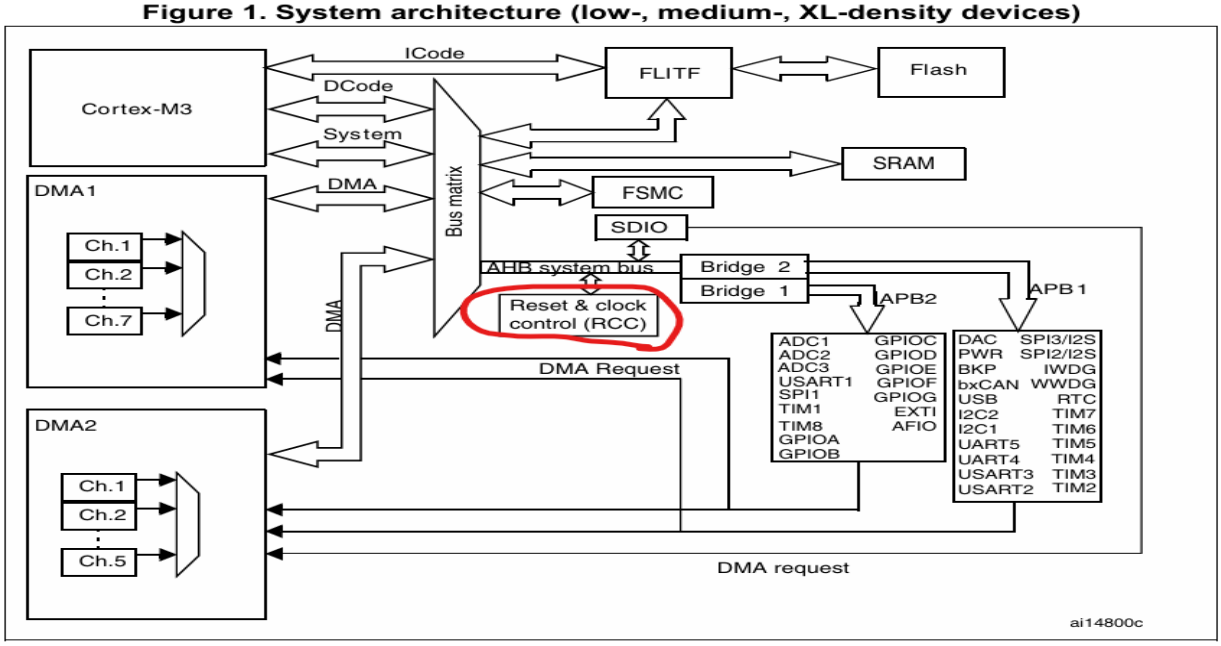
Figure 8. Clock tree



1.Lab1 :

-To configure the clock for STM32, from the data sheets, it is called RCC.

Figure 1. System architecture (low-, medium-, XL-density devices)



-Its memory map is

3.3 Memory map

See the datasheet corresponding to your device for a comprehensive diagram of the memory map. *Table 3* gives the boundary addresses of the peripherals available in all STM32F10xxx devices.

Table 3. Register boundary addresses

Boundary address	Peripheral	Bus	Register map
0xA000 0000 - 0xA000 0FFF	FSMC	AHB	*Section 21.6.9 on page 564*
0x5000 0000 - 0x5003 FFFF	USB OTG FS		*Section 28.16.6 on page 913*
0x4003 0000 - 0x4FFF FFFF	Reserved		-
0x4002 8000 - 0x4002 9FFF	Ethernet		*Section 29.8.5 on page 1069*
0x4002 3400 - 0x4002 7FFF	Reserved		-
0x4002 3000 - 0x4002 33FF	CRC		*Section 4.4.4 on page 65*
0x4002 2000 - 0x4002 23FF	Flash memory interface		-
0x4002 1400 - 0x4002 1FFF	Reserved		-
0x4002 1000 - 0x4002 13FF	Reset and clock control RCC		*Section 7.3.11 on page 121*
0x4002 0800 - 0x4002 0FFF	Reserved		-
0x4002 0400 - 0x4002 07FF	DMA2		

-To configure the clock for GPIOA, from the data sheets, it is on APB2 bus, Its memory is.

7.3.7 APB2 peripheral clock enable register (RCC_APB2ENR)

Address: 0x18

Reset value: 0x0000 0000

Access: word, half-word and byte access

No wait states, except if the access occurs while an access to a peripheral in the APB2 domain is on going. In this case, wait states are inserted until the access to APB2 peripheral is finished.

Note: When the peripheral clock is not active, the peripheral register values may not be readable by software and the returned value is always 0x0.

-For GPIOA:

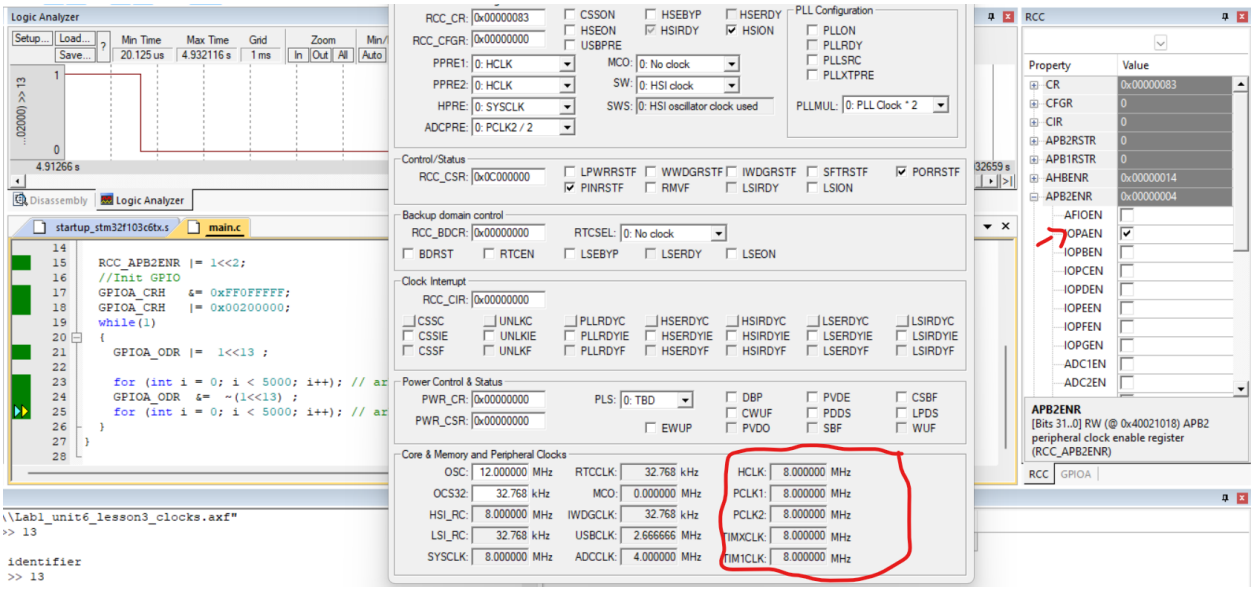
Bit 2 **IOPAEN:** IO port A clock enable
Set and cleared by software.
0: IO port A clock disabled
1: IO port A clock enabled

Bit 1 Reserved, must be kept at reset value.

-So, the code is:

```c
typedef volatile unsigned int vuint32_t ;
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>
// register address
#define GPIOA_BASE      0x40010800
#define GPIOA_CRH       *(volatile uint32_t *)(GPIOA_BASE + 0x04)
#define GPIOA_ODR       *(volatile uint32_t *)(GPIOA_BASE + 0x0C)
#define RCC_base        0x40021000
#define RCC_APB2ENR     *(volatile uint32_t*)(RCC_base + 0x18)
int main(void)
{
    //Init clock for GPIOA

    RCC_APB2ENR |= 1<<2;
    //Init GPIO
    GPIOA_CRH   &= 0xFF0FFFFF;
    GPIOA_CRH   |= 0x00200000;
    while(1)
    {
        GPIOA_ODR |=  1<<13 ;

        for (int i = 0; i < 5000; i++); // arbitrary delay
        GPIOA_ODR  &=  ~(1<<13) ;
        for (int i = 0; i < 5000; i++); // arbitrary delay
    }
}
```

-From the simulation:



2. **Lab2:**

-Requirements:

Configure Board to run with the Following rates:

- APB1 Bus frequency 4MHZ.
- APB2 Bus frequency 2MHZ.
- AHB frequency 8 MHZ.
- SysClk 8 MHZ .
- Use only internal HSI_RC.

-By default it uses HIS, SysClk and AHB equal 8MHz.

- to make APB2=2MHz and APB1=4MHz:

7.3.2 Clock configuration register (RCC_CFGR)

Address offset: 0x04

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 clock source switch.

Bits 13:11 **PPRE2:** APB high-speed prescaler (APB2)

Set and cleared by software to control the division factor of the APB high-speed clock (PCLK2).

0xx: HCLK not divided

100: HCLK divided by 2

101: HCLK divided by 4 ←

110: HCLK divided by 8

111: HCLK divided by 16

Bits 10:8 **PPRE1:** APB low-speed prescaler (APB1)

Set and cleared by software to control the division factor of the APB low-speed clock (PCLK1).

Warning: the software has to set correctly these bits to not exceed 36 MHz on this domain.

0xx: HCLK not divided

100: HCLK divided by 2 ←

101: HCLK divided by 4

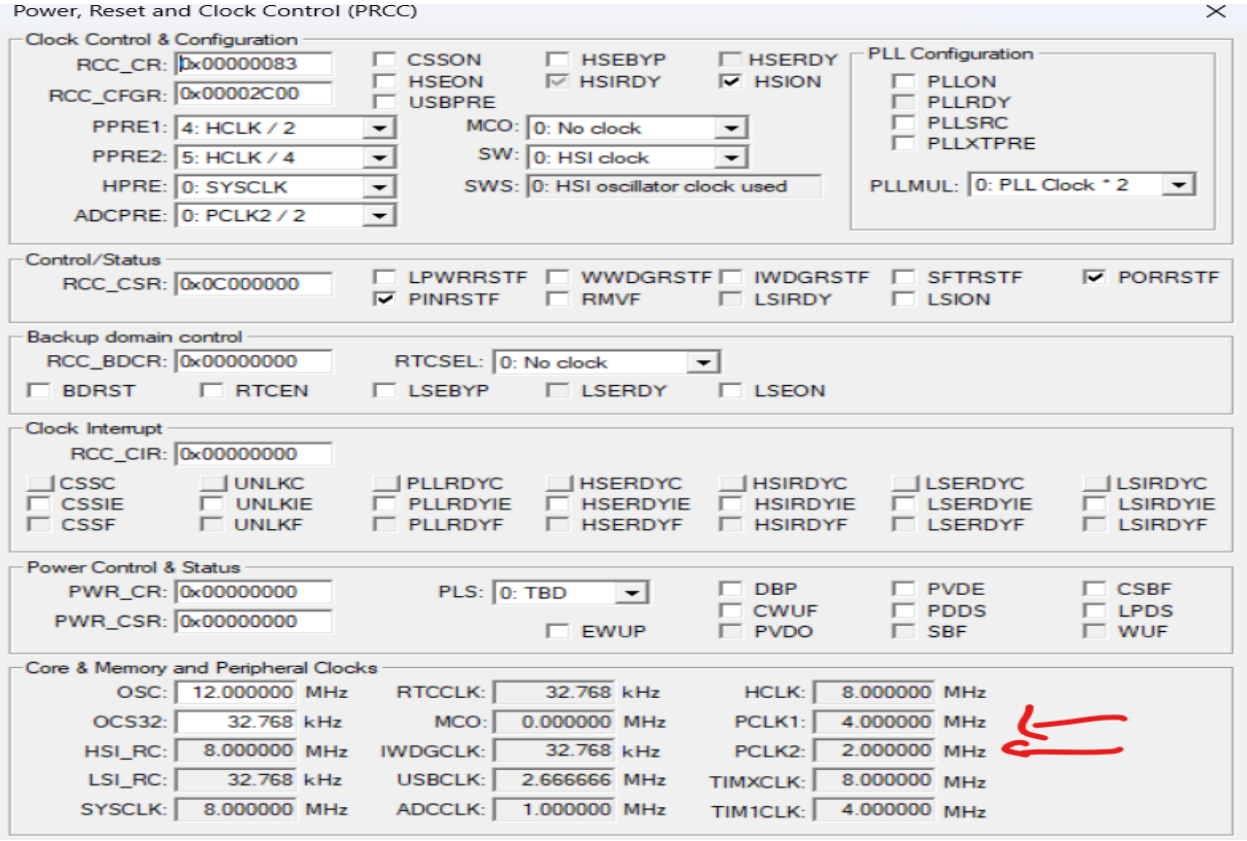
110: HCLK divided by 8

111: HCLK divided by 16

-The code:

```c
typedef volatile unsigned int vuint32_t ;
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>
// register address
#define GPIOA_BASE      0x40010800
#define GPIOA_CRH       *(volatile uint32_t *)(GPIOA_BASE + 0x04)
#define GPIOA_ODR       *(volatile uint32_t *)(GPIOA_BASE + 0x0C)
#define RCC_base        0x40021000
#define RCC_APB2ENR     *(volatile uint32_t*)(RCC_base + 0x18)
#define RCC_CFGR        *(volatile uint32_t*)(RCC_base + 0x04) //RCC configure
int main(void)
{
    //make APB1 clock = 4MHz

    RCC_CFGR |= (0b100)<<8 ;


    //make APB2 clock = 2MHz
    RCC_CFGR |= (0b101)<<11;
    //Init clock for GPIOA

    RCC_APB2ENR |= 1<<2;
    //Init GPIO
    GPIOA_CRH   &= 0xFF0FFFFF;
    GPIOA_CRH   |= 0x00200000;
    while(1)
    {
        GPIOA_ODR |=  1<<13 ;

        for (int i = 0; i < 5000; i++); // arbitrary delay
        GPIOA_ODR  &=  ~(1<<13) ;
        for (int i = 0; i < 5000; i++); // arbitrary delay
    }
}
```

-the Simulation:



3. **Lab3:**

-Requirements:

- Configure Board to run with the Following rates:
  - APB1 Bus frequency 16MHZ.
  - APB2 Bus frequency 8MHZ.
  - AHB frequency 32 MHZ.
  - SysClk 32 MHZ .

- Use only internal HSI_RC.

-To make SysClk = 32MHz, we should use PLL:

7.3.1 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						PLL RDY	PLLON	Reserved				CSS ON	HSE BYP	HSE RDY	HSE ON
						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

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **PLLRDY:** PLL clock ready flag
Set by hardware to indicate that the PLL is locked.
0: PLL unlocked
1: PLL locked

Bit 24 **PLLON:** PLL enable
Set and cleared by software to enable PLL.
Cleared by hardware when entering Stop or Standby mode. This bit can not be reset if the PLL clock is used as system clock or is selected to become the system clock.
0: PLL OFF
1: PLL ON

Bits 23:20 Reserved, must be kept at reset value.

Bits 21:18 **PLLMUL:** PLL multiplication factor
These bits are written by software to define the PLL multiplication factor. These bits can be written only when PLL is disabled.
Caution: The PLL output frequency must not exceed 72 MHz.
0000: PLL input clock x 2
0001: PLL input clock x 3
0010: PLL input clock x 4
0011: PLL input clock x 5
0100: PLL input clock x 6
0101: PLL input clock x 7
0110: PLL input clock x 8
0111: PLL input clock x 9
1000: PLL input clock x 10
1001: PLL input clock x 11
1010: PLL input clock x 12
1011: PLL input clock x 13
1100: PLL input clock x 14
1101: PLL input clock x 15
1110: PLL input clock x 16
1111: PLL input clock x 16

-to select PLL:

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

Set and cleared by hardware to indicate which clock source is used as system clock.
00: HSI oscillator used as system clock
01: HSE oscillator used as system clock
10: PLL used as system clock
11: not applicable

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

Set and cleared by software to select SYSCLK source.
Set by hardware to force HSI selection when leaving Stop and Standby mode or in case of failure of the HSE oscillator used directly or indirectly as system clock (if the Clock Security System is enabled).
00: HSI selected as system clock
01: HSE selected as system clock
10: PLL selected as system clock
11: not allowed

- to make APB1=16MHz:

Bits 10:8 **PPRE1:** APB low-speed prescaler (APB1)

Set and cleared by software to control the division factor of the APB low-speed clock (PCLK1).
Warning: the software has to set correctly these bits to not exceed 36 MHz on this domain.
0xx: HCLK not divided
100: HCLK divided by 2
101: HCLK divided by 4
110: HCLK divided by 8
111: HCLK divided by 16

-The simulation:

-The code:

```c
typedef volatile unsigned int vuint32_t ;
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>
// register address
#define GPIOA_BASE      0x40010800
#define GPIOA_CRH       *(volatile uint32_t *)(GPIOA_BASE + 0x04)
#define GPIOA_ODR       *(volatile uint32_t *)(GPIOA_BASE + 0x0C)
#define RCC_base        0x40021000
#define RCC_APB2ENR     *(volatile uint32_t*)(RCC_base + 0x18)
#define RCC_CFGR        *(volatile uint32_t*)(RCC_base + 0x04) //RCC configure
#define RCC_CR          *(volatile uint32_t*)(RCC_base + 0x0)
int main(void)
{
    //to use the PLL
    RCC_CR |= ( 1<<24 );

    // to make SW choose the PPL
    RCC_CFGR |= (0b10<<0);
    RCC_CFGR |= (0b10<<2);

    //to multiply the clock by 8
    RCC_CFGR |= (0b0110<<18);
    //make APB1 clock = 16MHz

    RCC_CFGR |= (0b100<<8) ;


    //make APB2 clock = 8MHz
    //RCC_CFGR |= (0b110<<11);
    //Init clock for GPIOA

    RCC_APB2ENR |= 1<<2;
    //Init GPIO
    GPIOA_CRH   &= 0xFF0FFFFF;
    GPIOA_CRH   |= 0x00200000;
    while(1)
    {
        GPIOA_ODR |=  1<<13 ;

        for (int i = 0; i < 5000; i++); // arbitrary delay
        GPIOA_ODR  &=  ~(1<<13) ;
        for (int i = 0; i < 5000; i++); // arbitrary delay
    }
}

```