# Report

Configure the board with different

Clock rate

Target: STM32f103c6

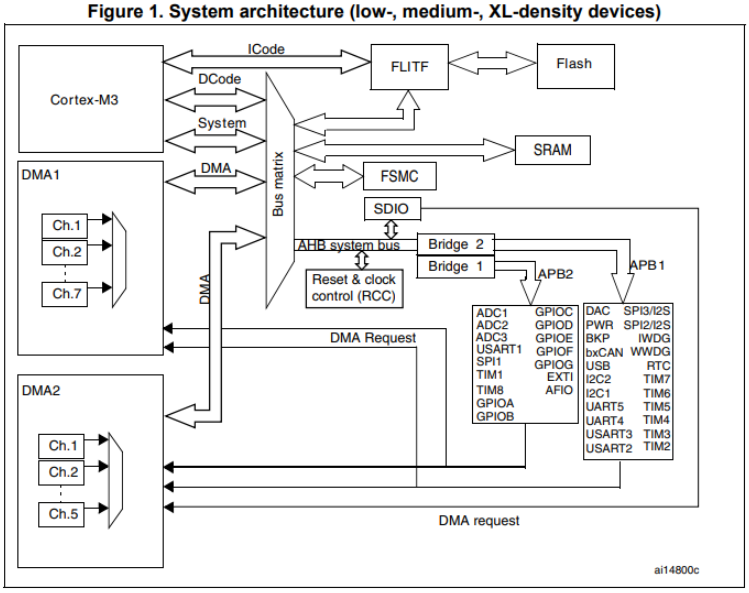
**Lab 1:**

Make GPIOA pin 13 be blinking and simulate the code using KEIL micro vision logic analyzer.

**Steps:**

1- First we should see memory map to get base address of GPIOA according its bus connected.
2- According to TRM we found out that GPIOA is connected to BUS APB2

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




3- According to memory map the base address of GPIOA is: 0x40010800

| | | | |
|---|---|---|---|
| 0x4001 3000 - 0x4001 33FF | SPI1 | APB2 | Section 25.5 on page 742 |
| 0x4001 2C00 - 0x4001 2FFF | TIM1 timer | | Section 14.4.21 on page 363 |
| 0x4001 2800 - 0x4001 2BFF | ADC2 | | Section 11.12.15 on page 252 |
| 0x4001 2400 - 0x4001 27FF | ADC1 | | |
| 0x4001 2000 - 0x4001 23FF | GPIO Port G | | Section 9.5 on page 194 |
| 0x4001 1C00 - 0x4001 1FFF | GPIO Port F | | |
| 0x4001 1800 - 0x4001 1BFF | GPIO Port E | | |
| 0x4001 1400 - 0x4001 17FF | GPIO Port D | | |
| 0x4001 1000 - 0x4001 13FF | GPIO Port C | | |
| 0x4001 0C00 - 0x4001 0FFF | GPIO Port B | | |
| 0x4001 0800 - 0x4001 0BFF | GPIO Port A | | |
| 0x4001 0400 - 0x4001 07FF | EXTI | | Section 10.3.7 on page 214 |
| 0x4001 0000 - 0x4001 03FF | AFIO | | Section 9.5 on page 194 |

4- We navigate to see GPIOA registers
We found to enable PORTA as output with max 2 MHZ we should write 1 on bit called mood13 bit number 21 in a register called CRH has offset 0x04.

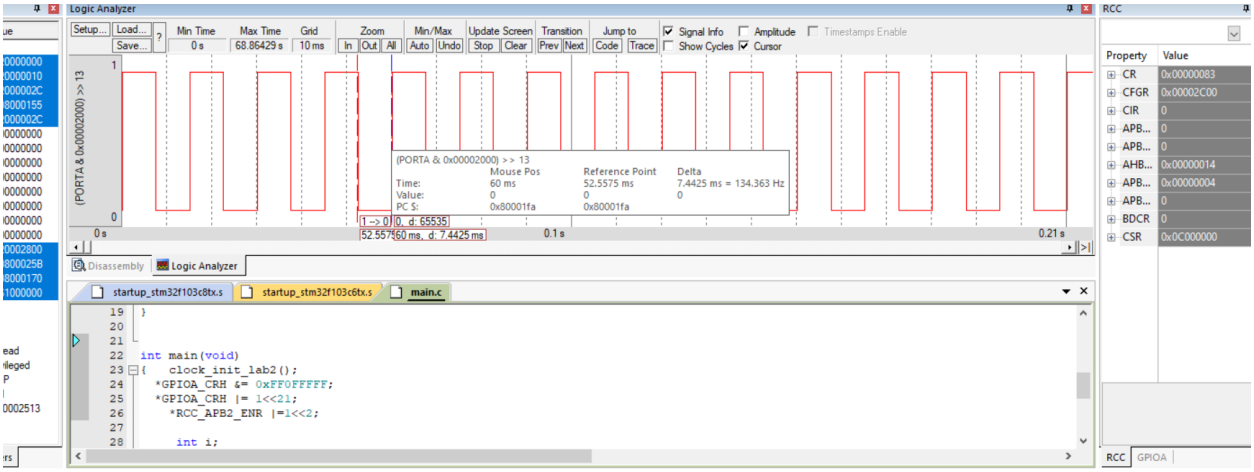
5- We also need the offset of ODR register to write on pin 13 to be blinking, the offset is 0x0c.

6- Finally we need to enable clock for PORTA
From RCC registers we found out to make that we need to write on bit number 2 on register called RCC_APB2ENR in clock and reset unit that has base address of 0x40021000
And the register has offset address of 0x18.

7- Lets see code:

```c
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#define GPIOA_BASE 0x40010800
volatile unsigned int* GPIOA_CRH=(volatile unsigned int *)0x40010804;
volatile unsigned int* GPIOA_ODR=(volatile unsigned int *)0x4001080c;
volatile unsigned int* RCC_APB2_ENR=(volatile unsigned int *)0x40021018;



int main(void)
{   *GPIOA_CRH &= 0xFF0FFFFF;
    *GPIOA_CRH |= 1<<21;
     *RCC_APB2_ENR |=1<<2;
     int i;
    while(1)
    {
        *GPIOA_ODR |=1<<13;
        for(i=0;i<50000;i++);
        *GPIOA_ODR &= ~(1<<13);
        for(i=0;i<5000;i++);

    }

}
```

8- Logic analyzer




# Lab2:

Configure board to run with following rates

APB1 Bus frequency 4 MHZ

APB2 Bus frequency 2 MHZ

AHB Bus frequency 8 MHZ

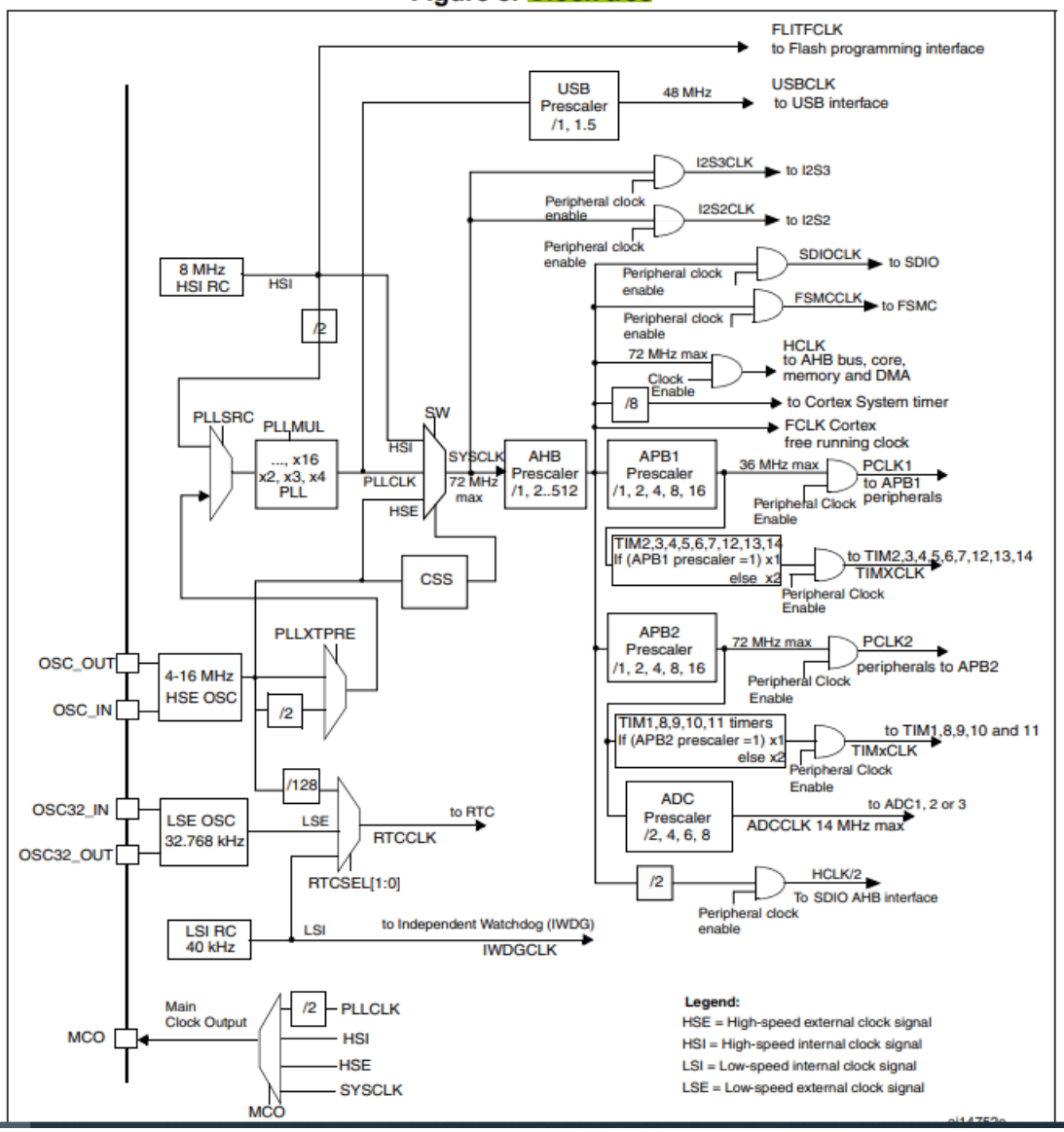
Sysclk frequency 2 MHZ

Use only internal HSI RC

## Steps:

1- First lets take a look on clock tree

**Figure 8. Clock tree**




2- to match requirements, we need to make clock source MUX SW select

HSI input to be the system clock 8 MHZ, make AHB prescaler divide by 1 to make the AHB bus clocking with 8 MHZ,

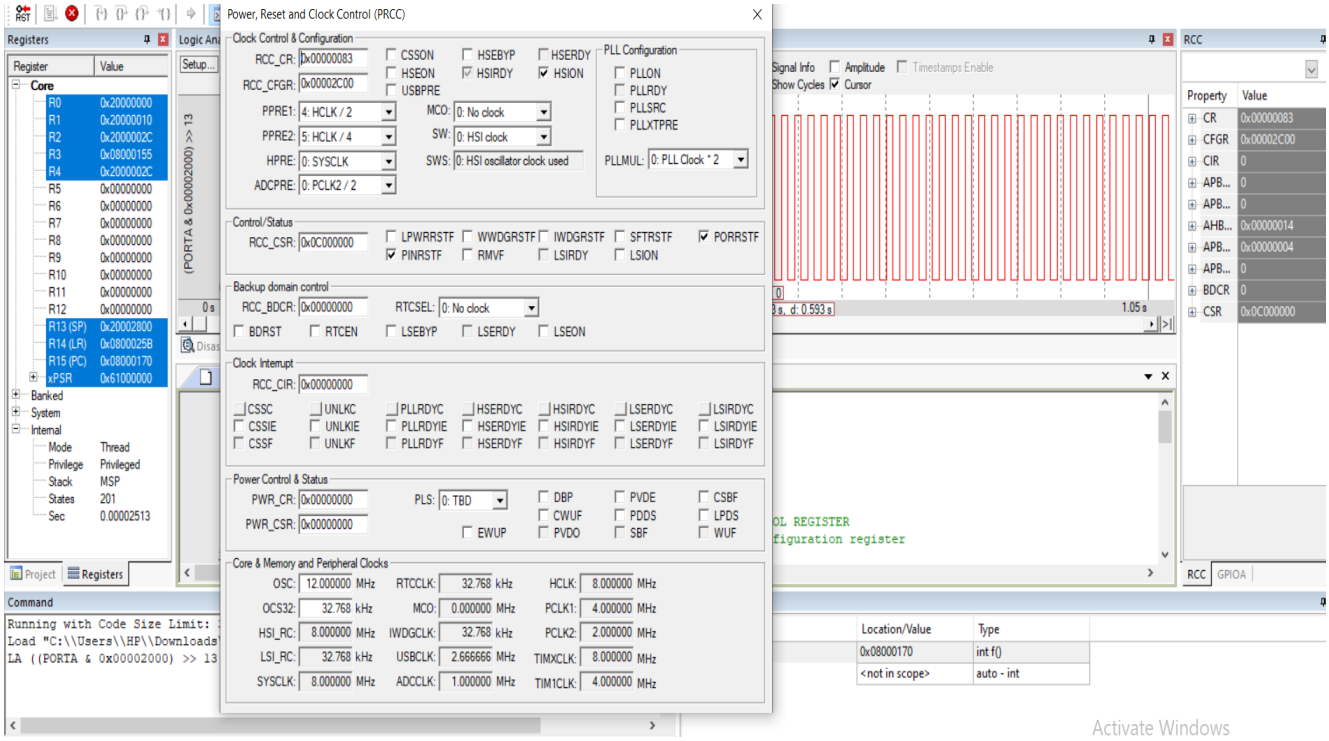
make APB1 prescaler divide by 2 to make APB1 bus clocking with 4 MHZ and make APB2 prescaler divide by 4 to make APB1 bus clocking with 2 MHZ.

3- According to TRM we found out that internal clock is enabled by default as a reset value of the register CR in bit HSION

4- Lets see code

```c
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#define GPIOA_BASE 0x40010800
volatile unsigned int* GPIOA_CRH=(volatile unsigned int *)0x40010804;
volatile unsigned int* GPIOA_ODR=(volatile unsigned int *)0x4001080c;
volatile unsigned int* RCC_APB2_ENR=(volatile unsigned int *)0x40021018;
volatile unsigned int* RCC_CR=(volatile unsigned int *)0x40021000; // CONTROL REGISTER
volatile unsigned int* RCC_CFGR=(volatile unsigned int *)0x40021004; // Configuration register

void clock_init_lab2()
{
// lab 2
//RCC_CR by default enables HSI clock source
//RCC_CFGR by default reset value selects HSI as clock source
//  *RCC_CFGR |=(0b0000<<4); // AHB NOT divided 8 Mhz
*RCC_CFGR |= (1<<10); // APB1 divided by 2
    *RCC_CFGR |=((1<<11)|(1<<13)); // APB2 divided by 4
}


int main(void)
{   clock_init_lab2();
    *GPIOA_CRH &= 0xFF0FFFFF;
    *GPIOA_CRH |= 1<<21;
    *RCC_APB2_ENR |=1<<2;

     int i;
    while(1)
    {
        *GPIOA_ODR |=1<<13;
        for(i=0;i<5000;i++);
        *GPIOA_ODR &= ~(1<<13);
        for(i=0;i<5000;i++);

    }

}
```

5- Lets see the output in KEIL




# Lab3:

Configure board to run with following rates

APB1 Bus frequency 16 MHZ

APB2 Bus frequency 8 MHZ

AHB Bus frequency 32 MHZ

Sysclk frequency 32 MHZ

Use only internal HSI RC

Steps:

1- Repeat the previous steps in lab 2 that related with getting base address and offset address of registers
2- We need to enable PLL, PLL SRC select HSI as a clock source for PLL, select PLL as a clock source using SW bit, multiply PLL MUL by 8 to out 32 MHZ as a system clock, APB1 prescaler divide by 2 to make APB1 clocking at 16 MHZ
   And APB2 prescaler divide by 4 to make APB1 clocking at 8 MHZ
3- Lets see the code

main.c

```c
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#define GPIOA_BASE 0x40010800
volatile unsigned int* GPIOA_CRH=(volatile unsigned int *)0x40010804;
volatile unsigned int* GPIOA_ODR=(volatile unsigned int *)0x4001080c;
volatile unsigned int* RCC_APB2_ENR=(volatile unsigned int *)0x40021018;
volatile unsigned int* RCC_CR=(volatile unsigned int *)0x40021000; // CONTROL REGISTER
volatile unsigned int* RCC_CFGR=(volatile unsigned int *)0x40021004; // Configuration register

/*void clock_init_lab2()
{
    // lab 2
    //RCC_CR by default enables HSI clock source
    //RCC_CFGR by default reset value selects HSI as clock source
    //  *RCC_CFGR |=(0b0000<<4); // AHB NOT divided 8 Mhz
 *RCC_CFGR |= (1<<10); // APB1 divided by 2
 *RCC_CFGR |=((1<<11)|(1<<13)); // APB2 divided by 4
}*/
void clock_init_lab3()
{   // RCC_CFGR bit 16 by default is 0 to select HSI as input to PLL


    *RCC_CFGR |=(1<<1); // select PLL as system clock
    *RCC_CFGR |=(1<<10); // divide APB1 prescaler by 2 to make APB1 clock by 32/2=16 MHZ
    *RCC_CFGR |=(1<<13)|(1<<11); // divide APB2 prescaler by 4 to make APB1 clock by 32/4=8 MHZ
    *RCC_CFGR |=(1<<20); // adjust PLL MULL to MUL by 8 to make sysclck is 4*8=32 MHZ
    *RCC_CFGR |=(1<<19);
    *RCC_CR |=(1<<24); // Enable PLL
}

int main(void)
{ //  clock_init_lab2();
    clock_init_lab3();
    *GPIOA_CRH &= 0xFF0FFFFF;
    *GPIOA_CRH |= 1<<21;
    *RCC_APB2_ENR |=1<<2;

```

# 4- The output on KEIL