

Report

Configure the board with different
Clock rate

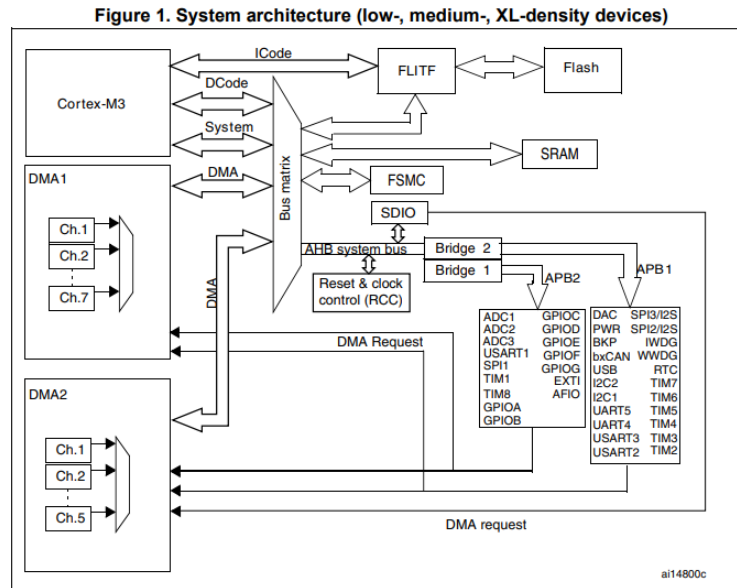
Target: STM32f103c6

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



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

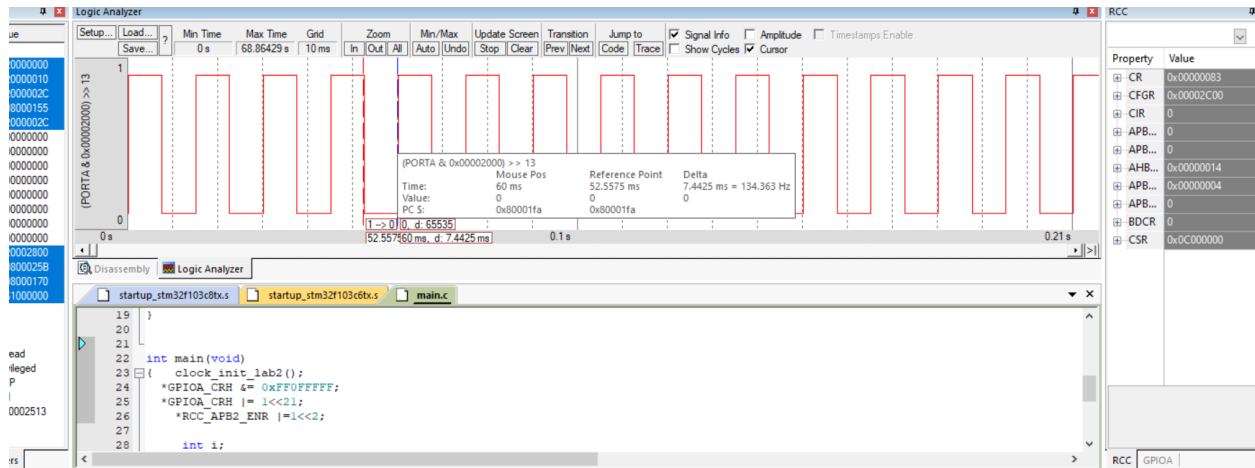
0x4001 3000 - 0x4001 33FF	SPI1	APB2	Section 25.5 on page 742
0x4001 2C00 - 0x4001 2FFF	TIM1 timer		Section 14.4.21 on page 363
0x4001 2800 - 0x4001 2BFF	ADC2		Section 11.12.15 on page 252
0x4001 2400 - 0x4001 27FF	ADC1		
0x4001 2000 - 0x4001 23FF	GPIO Port G		Section 9.5 on page 194
0x4001 1C00 - 0x4001 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		Section 10.3.7 on page 214
0x4001 0800 - 0x4001 0BFF	GPIO Port A		
0x4001 0400 - 0x4001 07FF	EXTI		
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:

```
1 #include <stdio.h>
2 #include <stdint.h>
3 #include <stdlib.h>
4 #define GPIOA_BASE 0x40010800
5 volatile unsigned int* GPIOA_CRH=(volatile unsigned int *)0x40010804;
6 volatile unsigned int* GPIOA_ODR=(volatile unsigned int *)0x4001080c;
7 volatile unsigned int* RCC_APB2_ENR=(volatile unsigned int *)0x40021018;
8
9
10
11 int main(void)
12 {
13     *GPIOA_CRH &= 0xFF0FFFFF;
14     *GPIOA_CRH |= 1<<21;
15     *RCC_APB2_ENR |= 1<<2;
16     int i;
17     while(1)
18     {
19         *GPIOA_ODR |= 1<<13;
20         for(i=0;i<50000;i++);
21         *GPIOA_ODR &= ~(1<<13);
22         for(i=0;i<50000;i++);
23     }
24 }
25
26
```

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

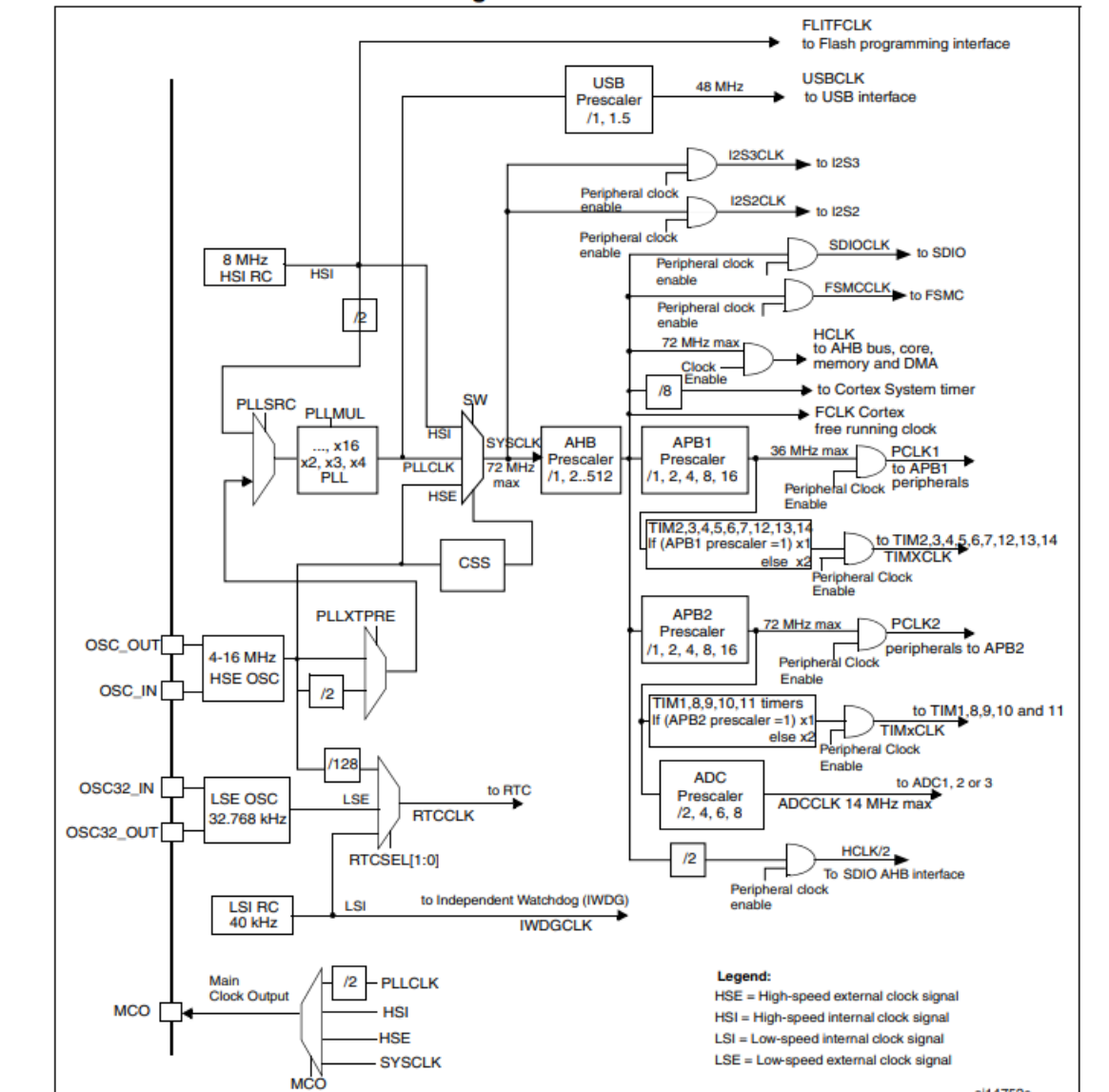
Sysclk frequency 2 MHZ

Use only internal HSI RC

Steps:

1- First lets take a look on clock tree

Figure 8. Clock tree

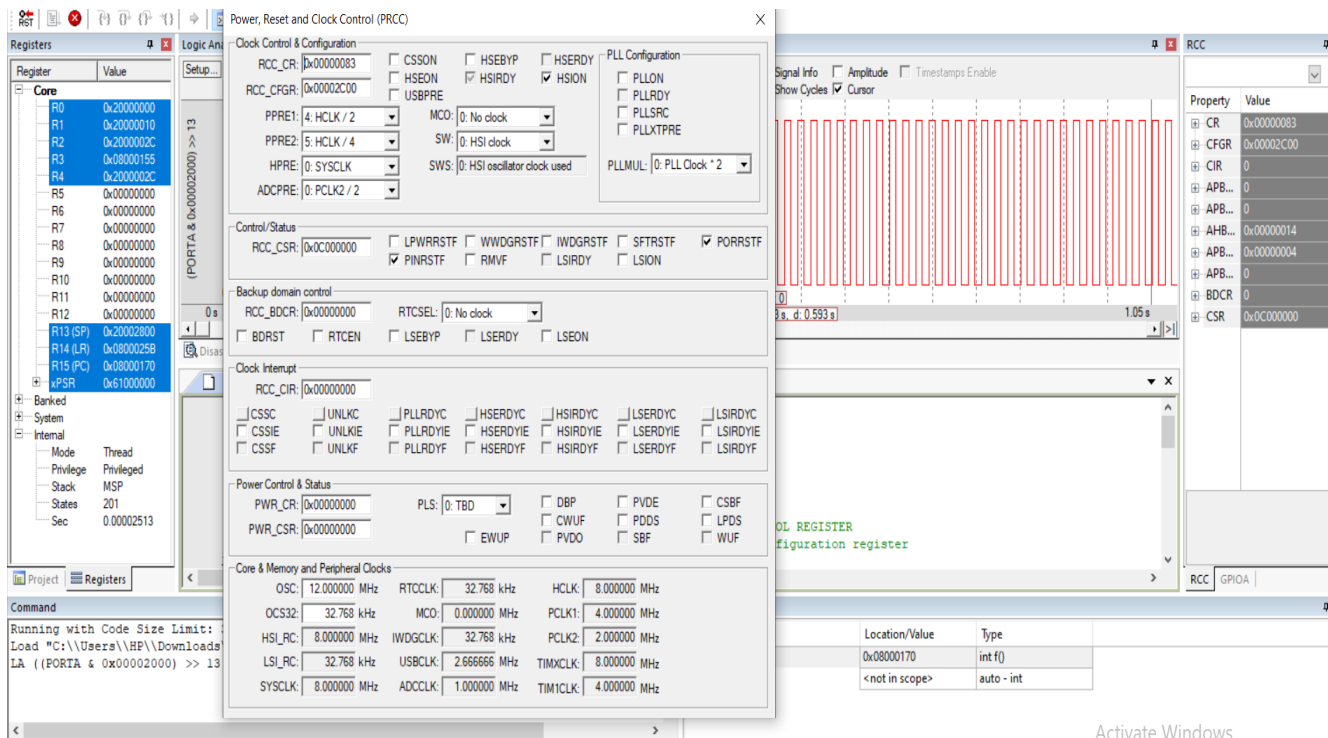


- 2- to match requirements, we need to make clock source MUX SW select HSI input to be the system clock 8 MHz, make AHB prescaler divide by 1 to make the AHB bus clocking with 8 MHz,

- 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

```
1 #include <stdio.h>
2 #include <stdint.h>
3 #include <stdlib.h>
4 #define GPIOA_BASE 0x40010800
5 volatile unsigned int* GPIOA_CRH=(volatile unsigned int *)0x40010804;
6 volatile unsigned int* GPIOA_ODR=(volatile unsigned int *)0x4001080c;
7 volatile unsigned int* RCC_APB2_ENR=(volatile unsigned int *)0x40021018;
8 volatile unsigned int* RCC_CR=(volatile unsigned int *)0x40021000; // CONTROL REGISTER
9 volatile unsigned int* RCC_CFGR=(volatile unsigned int *)0x40021004; // Configuration register
10
11 void clock_init_lab2()
12 {
13     // lab 2
14     //RCC_CR by default enables HSI clock source
15     //RCC_CFGR by default reset value selects HSI as clock source
16     // *RCC_CFGR |= (0b0000<<4); // AHB NOT divided 8 Mhz
17     *RCC_CFGR |= (1<<10); // APB1 divided by 2
18     *RCC_CFGR |= ((1<<11)|(1<<13)); // APB2 divided by 4
19 }
20
21
22 int main(void)
23 {
24     clock_init_lab2();
25     *GPIOA_CRH &= 0xFF0FFFFF;
26     *GPIOA_CRH |= 1<<21;
27     *RCC_APB2_ENR |= 1<<2;
28
29     int i;
30     while(1)
31     {
32         *GPIOA_ODR |= 1<<13;
33         for(i=0;i<5000;i++);
34         *GPIOA_ODR &= ~(1<<13);
35         for(i=0;i<5000;i++);
36     }
37 }
```

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
1 #include <stdio.h>
2 #include <stdint.h>
3 #include <stdlib.h>
4 #define GPIOA_BASE 0x40010800
5 volatile unsigned int* GPIOA_CRH=(volatile unsigned int *)0x40010804;
6 volatile unsigned int* GPIOA_ODR=(volatile unsigned int *)0x4001080c;
7 volatile unsigned int* RCC_APB2_ENR=(volatile unsigned int *)0x40021018;
8 volatile unsigned int* RCC_CR=(volatile unsigned int *)0x40021000; // CONTROL REGISTER
9 volatile unsigned int* RCC_CFGR=(volatile unsigned int *)0x40021004; // Configuration register
10
11 /*void clock_init_lab2()
12 {
13     // lab 2
14     //RCC_CR by default enables HSI clock source
15     //RCC_CFGR by default reset value selects HSI as clock source
16     // *RCC_CFGR |= (0b0000<<4); // AHB NOT divided 8 Mhz
17     *RCC_CFGR |= (1<<10); // APB1 divided by 2
18     *RCC_CFGR |= ((1<<11)|(1<<13)); // APB2 divided by 4
19 }*/
20 void clock_init_lab3()
21 { // RCC_CFGR bit 16 by default is 0 to select HSI as input to PLL
22
23
24     *RCC_CFGR |= (1<<1); // select PLL as system clock
25     *RCC_CFGR |= (1<<10); // divide APB1 prescaler by 2 to make APB1 clock by 32/2=16 MHZ
26     *RCC_CFGR |= (1<<13)|(1<<11); // divide APB2 prescaler by 4 to make APB1 clock by 32/4=8 MHZ
27     *RCC_CFGR |= (1<<20); // adjust PLL MULL to MUL by 8 to make sysclock is 4*8=32 MHZ
28     *RCC_CFGR |= (1<<19);
29     *RCC_CR |= (1<<24); // Enable PLL
30 }
31
32 int main(void)
33 { // clock_init_lab2();
34     clock_init_lab3();
35     *GPIOA_CRH &= 0xFFFFFFF;
36     *GPIOA_CRH |= 1<<21;
37     *RCC_APB2_ENR |= 1<<2;
38 }
```


4- The output on KEIL

Power, Reset and Clock Control (PRCC)

Registers

Register	Value
R0	0x20000000
R1	0x20000014
R2	0x20000030
R3	0x08000155
R4	0x20000030
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20002800
R14 (LR)	0x0800024F
R15 (PC)	0x080001E4
μPSR	0x61000000

Core

Mode: Thread

Privilege: Privileged

Stack: MSP

States: 211

Sec: 0.00002638

Command

Running with Code Size Limit:

Load "C:\Users\HP\Downloads\IA ((PORTA & 0x00002000)) >> 13

Logic Analyzer

Signal Info

Amplitude

Timestamps Enable

Show Cycles

Cursor

25.964 s

25.975 s

RCC

Property Value

CR	0x03000083
CFGR	0x00182C0A
CIR	0
APB...	0
APB...	0
AHB...	0x00000014
APB...	0x00000004
APB...	0
BDCR	0
CSR	0x0C000000

Core & Memory and Peripheral Clocks

OSC	RTCCCLK	HCLK
12.000000 MHz	32.768 kHz	32.000000 MHz

Core & Memory and Peripheral Clocks

OSCS32	MCO	PCLK1
32.768 kHz	0.000000 MHz	16.000000 MHz

Core & Memory and Peripheral Clocks

HSI_RC	IWDGCLK	PCLK2
8.000000 MHz	32.768 kHz	8.000000 MHz

Core & Memory and Peripheral Clocks

LSI_RC	USBCCLK	TIMXCLK
32.768 kHz	21.333333 MHz	32.000000 MHz

Core & Memory and Peripheral Clocks

SYSCCLK	ADCCCLK	TIM1CLK
32.000000 MHz	4.000000 MHz	16.000000 MHz

Location/Value Type

0x080001E4	int f()
<not in scope>	auto - int