MICROS 32 BITS

STM – PLL

ROBINSON JIMENEZ MORENO

UNIVERSIDAD MILITAR NUEVA GRANADA

INGENIERÍA MECATRÓNICA UMNG

RCC registers

El controlador RCC se encarga de manejar los relojes periférico y del Sistema. Este trabaja con un oscilador controlado por voltaje (VCO), de forma tal que, al cambiar valores de registros, se modifica el voltaje que controla la frecuencia de los relojes.

Es útil al usar múltiples módulos internos de forma simultánea que requieran de diferentes frecuencias.

También maneja diversos RESETs del dispositivo.

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
Res.	Res.	PLLSAIRDY	PLLSAION	PLLI2SRDY	PLLI2SON	PLLRDY	PLLON	Res.	Res.	Res.	Res.	CSSON	HSEBYP	HSERDY	HSEON
		r	rw	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.	HSIRDY	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

RCC clock control register (RCC_CR)

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

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

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 APB1 peripheral clock enable register (RCC_APB1ENR)

Address offset: 0x40

Reset value: 0x0000 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
UART8 EN	UART7 EN	DAC EN	PWR EN	CEC EN	CAN2 EN	CAN1 EN	I2C4 EN	I2C3 EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART 3 EN	USART 2 EN	SPDIFRX EN
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
SPI3 EN	SPI2 EN	Res.	Res.	WWDG EN	Res.	LPTIM1 EN	TIM14 EN	TIM13 EN	TIM12 EN	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

RCC clock configuration register (RCC_CFGR)

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	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPRE2[2:0]			PPRE1[2:0]			Res.	Res.	HPRE[3:0]				SWS1	SWS0	SW1	SW0
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	r	r	rw	rw

APB low-speed prescaler (APB1)

APB high-speed prescaler (APB2)

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 clock configuration register (RCC_CFGR)

Bits 15:13 **PPRE2:** APB high-speed prescaler (APB2)

Set and cleared by software to control APB high-speed clock division factor.

Caution: The software has to set these bits correctly not to exceed 108 MHz on this domain. The clocks are divided with the new prescaler factor from 1 to 16 AHB cycles after PPRE2 write.

0xx: AHB clock not divided
100: AHB clock divided by 2
101: AHB clock divided by 4
110: AHB clock divided by 8
111: AHB clock divided by 16

Bits 12:10 **PPRE1:** APB Low-speed prescaler (APB1)

Set and cleared by software to control APB low-speed clock division factor.

Caution: The software has to set these bits correctly not to exceed 54 MHz on this domain. The clocks are divided with the new prescaler factor from 1 to 16 AHB cycles after PPRE1 write.

0xx: AHB clock not divided
100: AHB clock divided by 2
101: AHB clock divided by 4
110: AHB clock divided by 8
111: AHB clock divided by 16

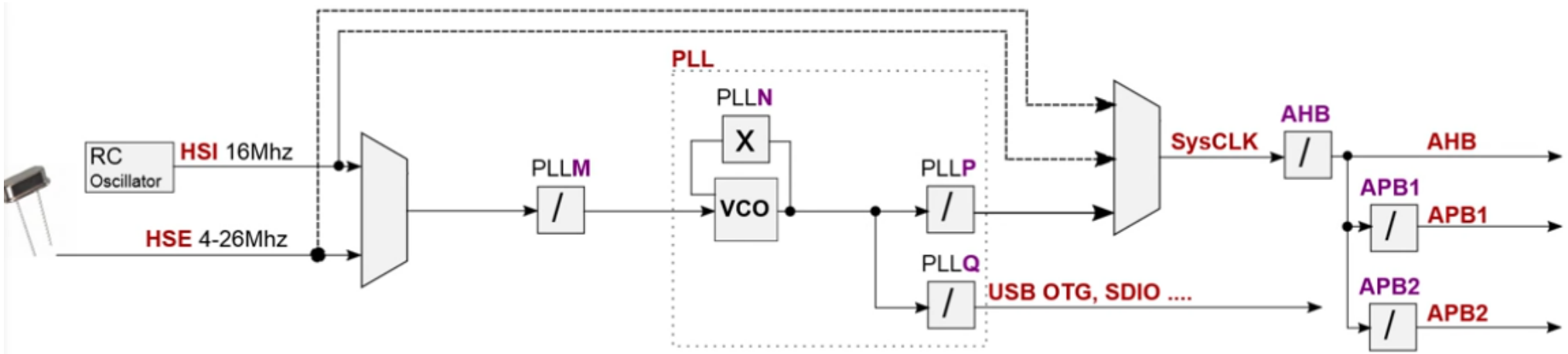
RCC PLL configuration register (RCC_PLLCFGR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	PLLQ[3:0]				Res.	PLLSRC	Res.	Res.	Res.	Res.	PLLP[1:0]	
				rw	rw	rw	rw		rw					rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PLLN[8:0]									PLLM[5:0]					
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

- $f_{(VCO\ clock)} = f_{(PLL\ clock\ input)} \times (PLLN / PLLM)$
- $f_{(PLL\ general\ clock\ output)} = f_{(VCO\ clock)} / PLLP$
- $f_{(USB\ OTG\ FS,\ SDMMC,\ RNG\ clock\ output)} = f_{(VCO\ clock)} / PLLQ$



Bits 5:0 **PLLM[5:0]:** Division factor for the main PLLs (PLL, PLLI2S and PLLSAI) 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

Bits 17:16 **PLLP[1:0]:** 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 216 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

Bits 14:6 **PLLN[8:0]:** 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 × PLLN with 50 ≤ PLLN ≤ 432

000000000: PLLN = 0, wrong configuration

000000001: PLLN = 1, wrong configuration

...

000110010: PLLN = 50

...

001100011: PLLN = 99

001100100: PLLN = 100

...

110110000: PLLN = 432

110110001: PLLN = 433, wrong configuration

...

111111111: PLLN = 511, wrong configuration

Note: Between 50 and 99, multiplication factors are possible for VCO input frequency higher than 1 MHz. However care must be taken to fulfill the minimum VCO output frequency as specified above.

Bits 27:24 **PLLQ[3:0]:** Main PLL (PLL) division factor for USB OTG FS, SDMMC and random number generator clocks

Set and cleared by software to control the frequency of USB OTG FS clock, the random number generator clock and the SDMMC clock. These bits should be written only if PLL is disabled.

Caution: The USB OTG FS requires a 48 MHz clock to work correctly. The SDMMC and the random number generator need a frequency lower than or equal to 48 MHz to work correctly.

USB OTG FS clock frequency = VCO frequency / PLLQ with $2 \leq PLLQ \leq 15$

0000: PLLQ = 0, wrong configuration

0001: PLLQ = 1, wrong configuration

0010: PLLQ = 2

0011: PLLQ = 3

0100: PLLQ = 4

...

1111: PLLQ = 15

Ejemplo PLL

```c
#include <stdio.h>
#include "STM32F7xx.h"


int time=500000;      //n ciclos de maquina


void PLL(void)
{
 int a=0;
 RCC->CR |= 0x10000;                    //Activar el oscilador HSE
 while((RCC->CR & 0x20000)==0);         //Esperar que el HSE este estabilizado
 RCC->APB1ENR = 0x10000000;             //Power interface clock enable
 RCC->CFGR = 0x9400;                    //APB2 (/2)    y APB1 (/4)
 RCC->PLLCFGR = 0x7405408;              //HSE reloj de entrada al PLL; PLLM=8; PLLN=336; PLLP=2
 RCC->CR |= 0x01000000;                 //Activar el PLL
 while((RCC->CR & 0x02000000)==0);      //Esperar que el PLL este listo
 FLASH->ACR = 0x205;                    //Activar el ART Accelerator y actualiza a 5wait states
 RCC->CFGR |= 2;                        //Seleccionar el PLL como la fuente de reloj del micro
 for (a=0;a<=500;a++);
}
```

Ejemplo PLL

```
int main(void)
{
  PLL();

  int i=0;
  //*********************************************************************
  //CONFIGURACION "CLOCK"
  RCC->AHB1ENR |= (1UL << 1);     //PRENDER EL CLOCK DEL PTB

  //*********************************************************************
  //CONFIGURACION DE PINES
  GPIOB->MODER |= 0x10004001;        //PTB0 - PTB7 - PTB14 -> OUTPUT
  GPIOB->OTYPER = 0;                //PUSH PULL -> PTB0 - PTB7 - PTB14
  GPIOB->OSPEEDR |= 0x10004001;      //MEDIUM SPEEED -> PTB0 - PTB7 - PTB14
  GPIOB->PUPDR |= 0x10004001;        //PULL-UP -> PTB0 - PTB7 - PTB14

  //*********************************************************************

  while(true){         //bucle infinito

        GPIOB->ODR |= (1UL<<14);     //enciende LED
        for(i=0;i<time;i++);         //Delay
        GPIOB->ODR &= ~(1UL<<14);    //apaga LED
        for(i=0;i<time;i++);         //Delay


  }//cierra while

}//cierra main
```