Esse é o décimo segundo artigo da série escrita pelo engenheiro Ismael Lopes da Silva, exclusivamente para o site "www.embarcados.com.br". Nessa série focarei no Microcontrolador da STMicroelectronics, o MCU STM32F103C8T6, que é um ARM Cortex-M3. Os pré-requisitos para uma boa compreensão dos artigos é ter o domínio da Linguagem C Embedded e conceitos de eletrônica.

Programação Bare Metal do STM32F103C8T6 – LED Blink

Vamos construir um aplicativo do zero (from scratch), que fará piscar o LED interno da Blue Pill (PC13), usando apenas o STM32CubeIDE, os documentos de referência do MCU STM32F103C8T6, e alguns conceitos que já vimos nos artigos anteriores.

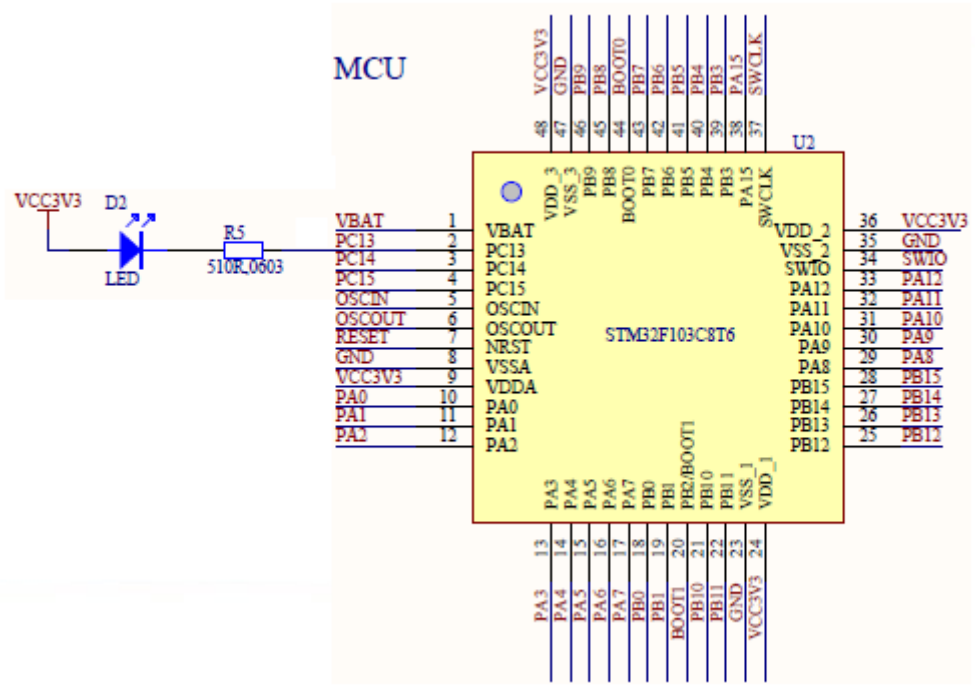

Figura 1 – Esquema de ligação do LED

Os documentos que vamos usar são:

- ST - STM32F103xx Manual de Referência (RM0008 - DocID13902 Rev 16)
- Blue Pill (original schematic STM32F103C8T6)

O LED interno da Blue Pill está conectado no pino 13 da porta C do MCU, conforme ilustrado na figura 1. Como estamos partindo do zero, precisamos saber o que vem como padrão? Quais os status dos Registradores quando o MCU é resetado? E o que temos que configurar?

Como padrão, para economia de energia, a maioria dos periféricos vem desabilitados, incluindo todas as portas do STM32F103C8T6. Como vamos utilizar a porta C, será necessário habilitar o clock para a porta C, mas, antes vamos analisar a arquitetura macro do MCU, ilustrada na figura 2, para termos uma boa noção do que temos que fazer.

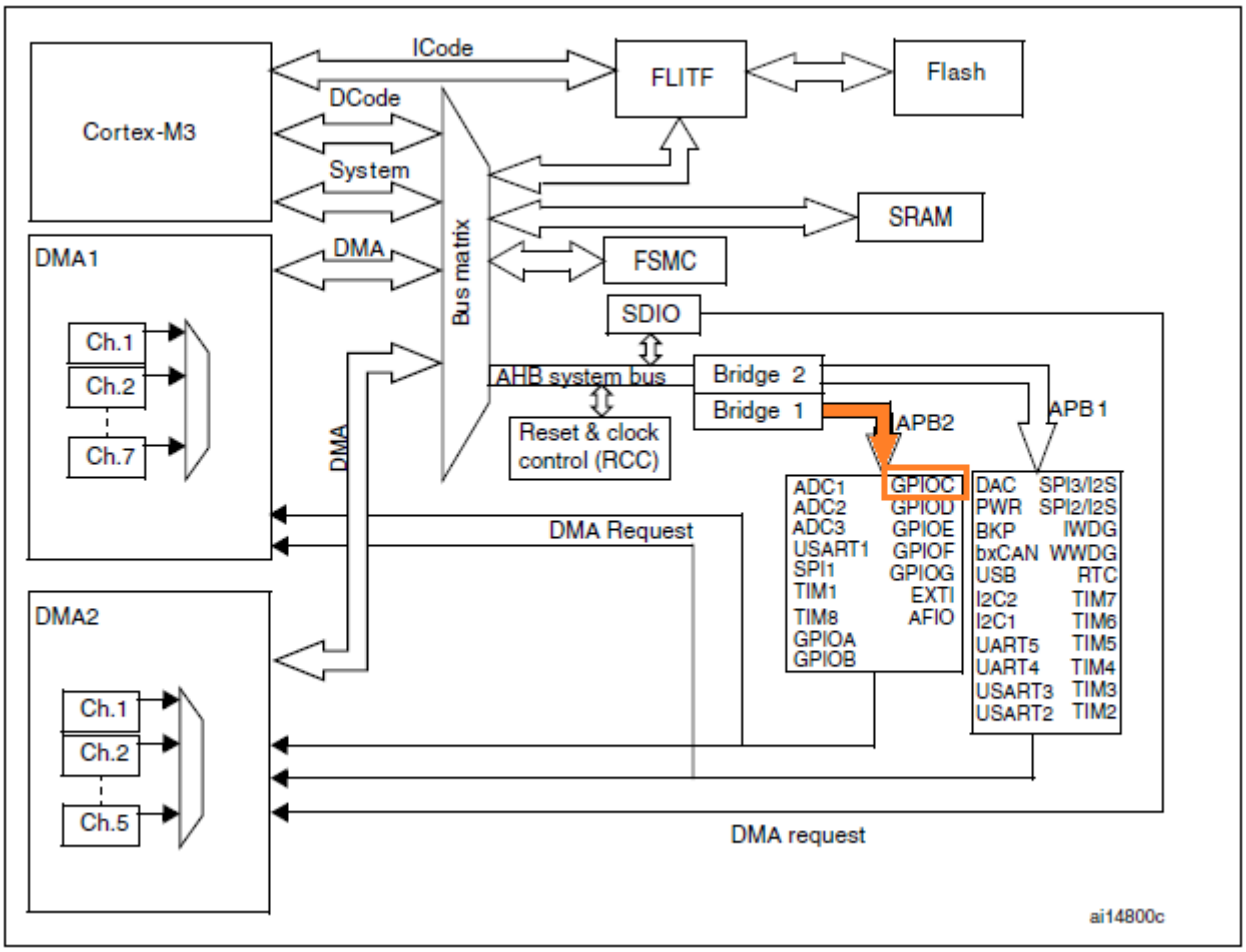

Figura 2 – Arquitetura STM32F103C8T6

A Porta C é acessada pelo Bus APB2, ver figura 2. Conforme o manual de referência, temos que configurar o Registrador RCC_APB2ENR, que habilita o clock dos periféricos do Bus APB2, ilustrado na figura 3. Se setarmos o bit 4 o clock para a Porta C será habilitado.

8.3.7 APB2 peripheral clock enable register (RCC_APB2ENR)

Address: 0x18

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
Res.	USART 1EN	Res.	SPI1 EN	TIM1 EN	ADC2 EN	ADC1 EN	Reserved		IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	Res.	AFIO EN
	rw		rw	rw	rw	rw			rw	rw	rw	rw	rw		rw

Figura 3 – Registrador RCC_APB2ENR

Observe na figura 3, que o Registrador RCC_APB2ENR tem seu endereço com deslocamento 0x18. Isso significa que teremos que encontrar o endereço base do Registrador RCC - Reset and Clock Control. No manual de referência, em mapa de memória, podemos obter o endereço de base, que é 0x4002.1000, portanto, o endereço do Registrador RCC_APB2ENR = 0x4002.1000 + 0x18. Usando

a calculadora do Programador, artigo especial dessa série, podemos efetuar essa soma na base hexadecimal. O resultado é 0x4002.1018, conforme ilustrado na figura 4.

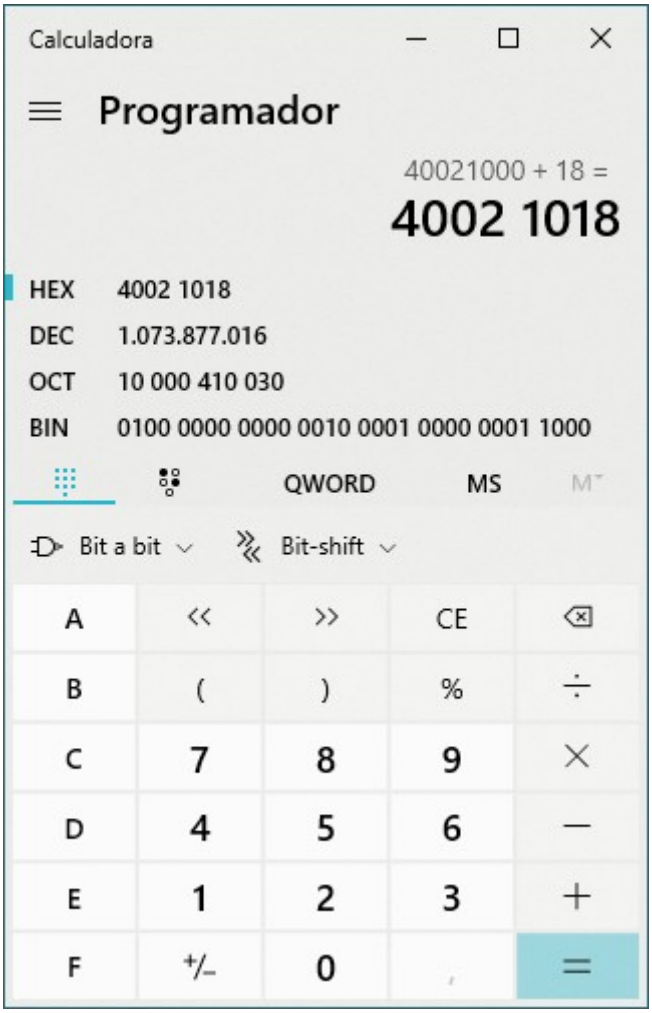

Figura 4 – Calculadora do programador

Também o clock do MCU deveria ser habilitado, mas, a condição inicial, isto é, o estado dos bits do Registrador RCC_CR depois do reset, como padrão, deixa configurado o clock interno de 8 MHz, então, vamos conferir. Na figura 5 temos o primeiro Registrador que Controla o Clock do MCU. Observe o valor depois do reset = 0x0000.XX83. Vamos verificar os bits afetados.

7.3.1 Clock control register (RCC_CR)

Address offset: 0x00

Reset value: 0x0000 XX83 where X is undefined.

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

Figura 5 – Registrador RCC_CR

Usando a calculadora do Programador o valor x83 = b1000.0011. Confira na figura 5 que os bits 0 e 1, indicam que o clock HSI foi setado e está pronto. O bit 7 significa que o clock HSI vem calibrado de fábrica com o valor 16 (padrão), para o HSITRIM[4:0], Bits 7:3. Esses bits fornecem um valor de corte adicional programável pelo usuário que é adicionado ao HSICAL, bits [7: 0]. Ele pode ser programado para se ajustar às variações de tensão e temperatura que influenciam a frequência do RC interno do HSI . O valor padrão é 16, que, quando adicionado ao valor HSICAL, deve ajustar o HSI para 8 MHz ± 1%.

Depois que verificamos o clock do MCU e habilitamos o clock da Porta C, agora temos outras configurações a realizar na Porta C, que são:

- Configurar o pino 13 da Porta C como uma saída digital;
- E saber como escrever, zero e um, no pino 13 da Porta C.

Na figura 6 podemos ver o circuito básico de entrada e saída de um pino do MCU. Obviamente dependendo da configuração, um determinado pino de uma porta pode ser uma entrada (caminho amarelo) ou uma saída (caminho azul). Em nosso caso precisamos ligar e desligar um LED, então, devemos configurar como uma saída digital.

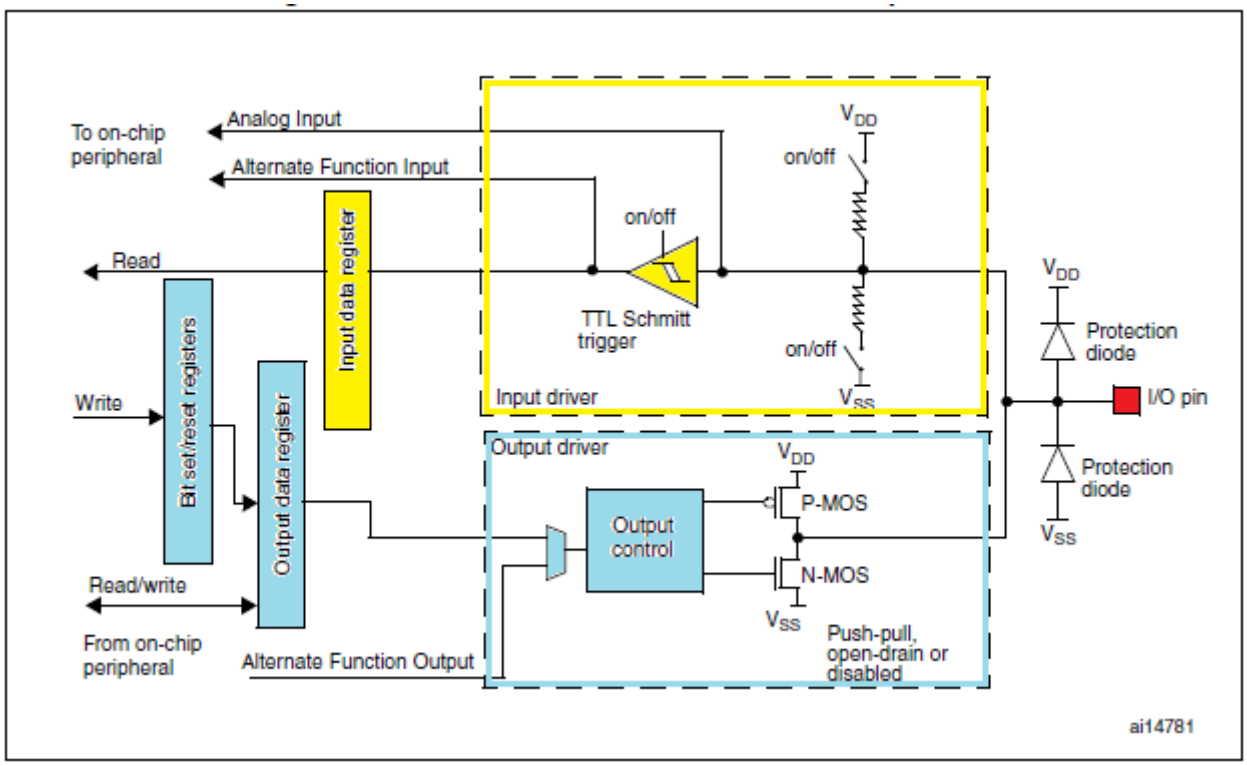

Figura 6 – Estrutura básica das portas de Entrada e Saída

Primeiro encontraremos o endereço base do Registrador da Porta C. No manual de referência, em mapa de memória, podemos obter o endereço de base da Porta C, que é 0x4001.1000. Agora configuraremos o Registrador GPIOC_CRH - Port C Configuration Register High. Vamos tratar somente a parte HIGH porque o pino 13, está contido na parte HIGH (pinos 8 à 15). Se fossemos usar, por exemplo, o pino 3, então, o Registrador seria o GPIOC_CRL - Port C Configuration Register Low (pinos 0 à 7).

Na figura 7 podemos ver que para cada pino da Porta C, temos quatro bits de configuração, divididos em dois grupos de dois bits, que são: CNF e MODE. Também o deslocamento do

endereço desse Registrador é 0x04, portanto, o endereço exato será 0x4001.1000 + 0x04 = 0x4001.1004.

9.2.2 Port configuration register high (GPIOC_CRH)

Address offset: 0x04

Reset value: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]		CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[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
CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]		CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figura 7 – Registrador GPIOC_CRH

CNF13[1:0]: Port x configuration bits **In output mode (MODE[1:0] > 00):** 00: General purpose output push-pull 01: General purpose output Open-drain 10: Alternate function output Push-pull 11: Alternate function output Open-drain	**MODE13 [1:0]:** Port x mode bits 00: Input mode (reset state) 01: Output mode, max speed 10 MHz. 10: Output mode, max speed 2 MHz. 11: Output mode, max speed 50 MHz.

Figura 8 – Bits do CNF13 e MODE13

Para configurar somente o pino 13 da Porta C, os bits envolvidos são CNF13[23:22] e MODE13[21:20], conforme pode ser confirmado na figura 7. Vamos fazer as seguintes configurações:

- MODE13[21:20] = 10 → Saída, com velocidade máxima de 2MHz;
- CNF13[23:22] = 00 → Saída de propósito geral push-pull.

Foi configurado a menor velocidade máxima de saída, porque para piscar um LED, onde o olho humano perceba as transições, não requer uma frequência alta, e 2 MHz é suficiente. Também o tipo push-pull utilizaremos os MOSFET, que são os Drivers de saída, puxando para nível 1 ou empurrando para nível 0.

9.2.4 Port output data register (GPIOC_ODR)

Address offset: 0x0C

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

Figura 9 – Registrador do Dado de Saída da Porta C

Para escrevermos no pino 13 da Porta C, temos que fazer isso através do Registrador GPIO_ODR, que escreve o dado na saída, como podemos vê-lo na figura 6, no circuito de saída. O endereço do Registrador GPIOC_ODR tem o deslocamento 0x0C, conforme ilustrado na figura 9. O endereço base da Porta C nós já havíamos obtido, que é 0x4001.1000, portanto, o endereço exato é 0x4001.100C.

Um resumo do que fizemos até aqui:

- O Clock HSI 8MHz já é configurado como padrão (estado de reset);
- Obtivemos o endereço do Registrador RCC_APB2ENR = 0x4002.1018;
  - ✔ Temos que setar o bit 4 do RCC_APB2ENR;
- Obtivemos o endereço do Registrador GPIOC_CRH = 0x4001.1004;
  - ✔ Temos que configurar MODE13[21:20] = 10 do GPIOC_CRH;
  - ✔ Temos que configurar CNF13[23:22] = 00 do GPIOC_CRH;
- Obtivemos o endereço do Registrador GPIOC_ODR = 0x4001.100C;
  - ✔ Temos que setar/resetar o bit 4 do RCC_APB2ENR para acender/apagar o LED 13.

Agora é escrever nossa aplicação no STM32CubeIDE, partindo do zero, conforme a proposta inicial desse artigo. Não vou detalhar os passos para criar um projeto, porque isso já foi visto várias vezes nos artigos dessa série. No STM32CubeIDE criei um projeto "06LEDBLINK" com o seguinte arquivo main.c:

```
#if !defined(__SOFT_FP__) && defined(__ARM_FP)
  #warning "FPU is not initialized, but the project is compiling for an FPU. Please initialize the FPU before use."
#endif

#include<stdio.h>

int main(void)
{
	//Define os ponteiros para os Registradores RCC_APB2ENR, GPIOC_CRH e GPIOC_ODR
	uint32_t *pRCC_APB2ENR = (uint32_t)0x40021018;
	uint32_t *pGPIOC_CRH = (uint32_t)0x40011004;
	uint32_t *pGPIOC_ODR = (uint32_t)0x4001100C;

	//1. Habilita o clock da Porta C
	*pRCC_APB2ENR |= 0x10;
	//CNF13[23:22] = 00 e MODE13[21:20] = 10
	//2a. Reseta os bits [23:20]
	*pGPIOC_CRH &= 0xFF0FFFFF;
	//2b. Seta o bit [21]
	*pGPIOC_CRH |= 0x00200000;
	//3. Seta o pino 13 da Porta C (apaga o LED)
	*pGPIOC_ODR |= 0x00002000;
```

```
	while(1)
	{
		//Delay para ver o LED piscar
		for (uint32_t i = 0; i < 500000; i++);
		//Reseta o pino 13 da Porta C (acende o LED)
		*pGPIOC_ODR &= 0xFFFFDFFF;
		//Delay para ver o LED piscar
		for (uint32_t i = 0; i < 500000; i++);
		//Seta o pino 13 da Porta C (apaga o LED)
		*pGPIOC_ODR |= 0x00002000;
	}
}
```

No STM32CubeIDE, na nova aplicação “06LEDBLINK”, dê um “Clean Project”, “Build Project” e “RUN as” → “STM32 Cortex-M C/C++ Application”, e o programa será escrito na Blue Pill e o LED BLINK. Também se quiser depurar o programa e ver passo a passo os Registradores sendo modificados, então, basta utilizar os recursos do STM32CubeIDE, conforme já mostrado nesta série.