GPIO

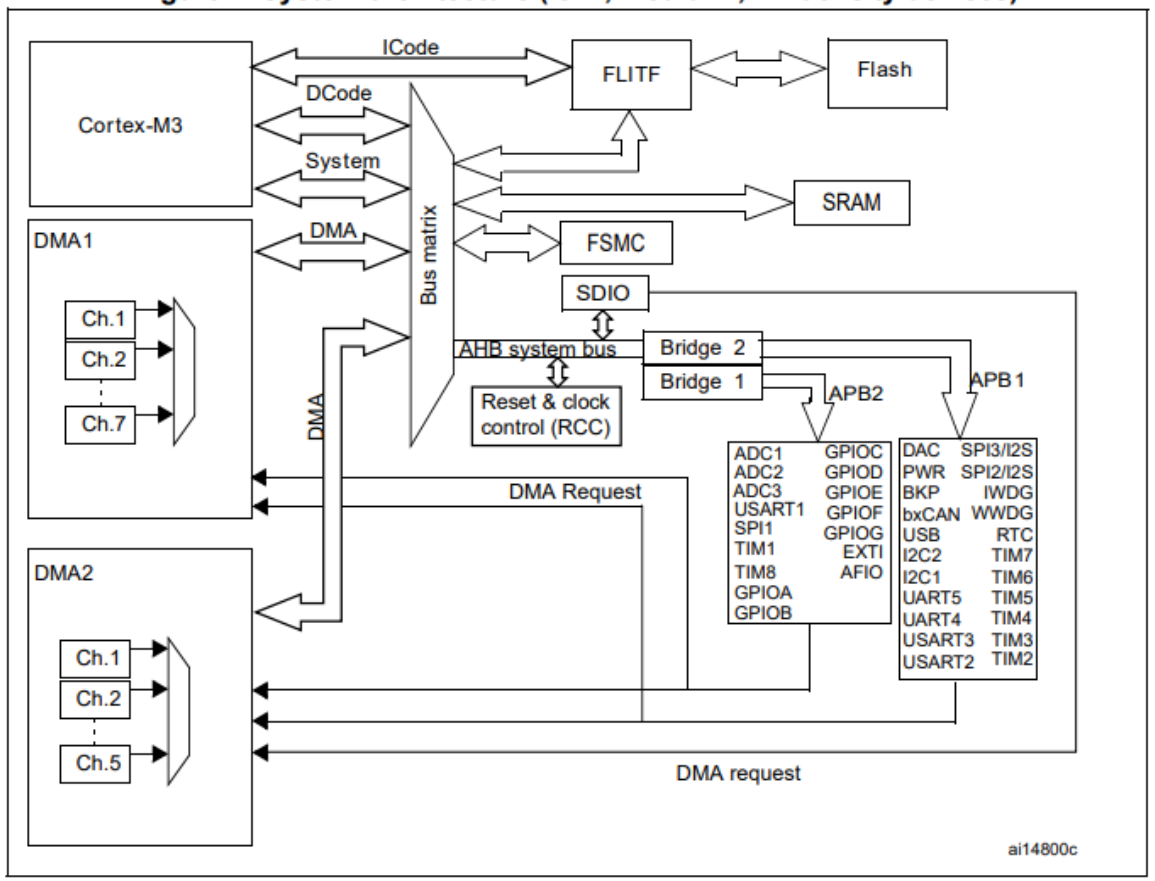
Programming by Registers

Any thoughts ?

```c
#include "stm32f103x6.h"

int main(void)
{
    RCC->APB2ENR = RCC_BASE+0x18;
    RCC->APB2ENR |= RCC_APB2ENR_IOPCEN;

    GPIOC->CRH = GPIOC_BASE+0x04;
    GPIOC->CRH &= 0x44044444;
    GPIOC->CRH |= 0x00200000;

    GPIOC->ODR = GPIOC_BASE+0x0C;

    while(1)
    {
        GPIOC->ODR |= (1<<13);
        for (volatile int i = 0; i < 500000; i++);
        GPIOC->ODR &= ~(1<<13);
        for (volatile int i = 0; i < 500000; i++);
    }
}
```

System Architecture

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



Therefore…

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.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										TIM11 EN	TIM10 EN	TIM9 EN	Reserved		
										rw	rw	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3 EN	USART 1EN	TIM8 EN	SPI1 EN	TIM1 EN	ADC2 EN	ADC1 EN	IOPG EN	IOPF EN	IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	Res.	AFIO EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

1 Lsh 4 =

10

HEX	10
DEC	16
OCT	20
BIN	0001 0000

Bit 4 **IOPCEN:** IO port C clock enable
Set and cleared by software.
0: IO port C clock disabled
1: IO port C clock enabled

```
RCC->APB2ENR = RCC_BASE+0x18;
RCC->APB2ENR |= RCC_APB2ENR_IOPCEN;
```

```
#define RCC_APB2ENR_IOPCEN_Pos          (4U)
#define RCC_APB2ENR_IOPCEN_Msk          (0x1UL << RCC_APB2ENR_IOPCEN_Pos)
#define RCC_APB2ENR_IOPCEN              RCC_APB2ENR_IOPCEN_Msk
```

Set Port C Pin 13 as Output

9.2.2 Port configuration register high (GPIOx_CRH) (x=A..G)

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

Bits 31:30, 27:26, 23:22, 19:18, 15:14, 11:10, 7:6, 3:2 **CNFy[1:0]:** Port x configuration bits (y= 8 .. 15)
These bits are written by software to configure the corresponding I/O port.
Refer to *Table 20: Port bit configuration table*.
In input mode (MODE[1:0]=00):
00: Analog mode
01: Floating input (reset state)
10: Input with pull-up / pull-down
11: Reserved
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

Bits 29:28, 25:24, 21:20, 17:16, 13:12, 9:8, 5:4, 1:0 **MODEy[1:0]:** Port x mode bits (y= 8 .. 15)
These bits are written by software to configure the corresponding I/O port.
Refer to *Table 20: Port bit configuration table*.
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.

20 0000

HEX	20 0000
DEC	2.097.152
OCT	10 000 000
BIN	0010 0000 0000 0000 0000 0000

```
GPIOC->CRH = GPIOC_BASE+0x04;
GPIOC->CRH &= 0x44044444;
GPIOC->CRH |= 0x00200000;
```

Set and Reset Port C Pin 13

9.2.4 Port output data register (GPIOx_ODR) (x=A..G)

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODRy:** Port output data (y= 0 .. 15)

These bits can be read and written by software and can be accessed in Word mode only.

Note: For atomic bit set/reset, the ODR bits can be individually set and cleared by writing to the GPIOx_BSRR register (x = A .. G).


```
GPIOC->ODR |= (1<<13);
for (volatile int i = 0; i < 500000; i++);
GPIOC->ODR &= ~(1<<13);
for (volatile int i = 0; i < 500000; i++);
```

Practices :

- Make a Traffic Light using LEDs in Port B
- Read Input from Button (clue : read about GPIOx_IDR)

Thank you!