

MICROS 32 BITS

STM - GPIO

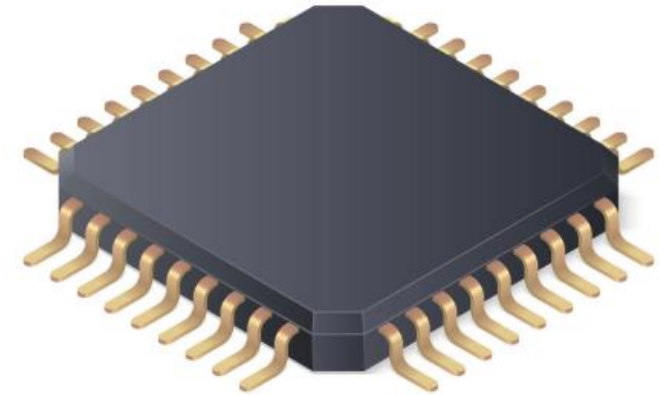
ROBINSON JIMENEZ MORENO

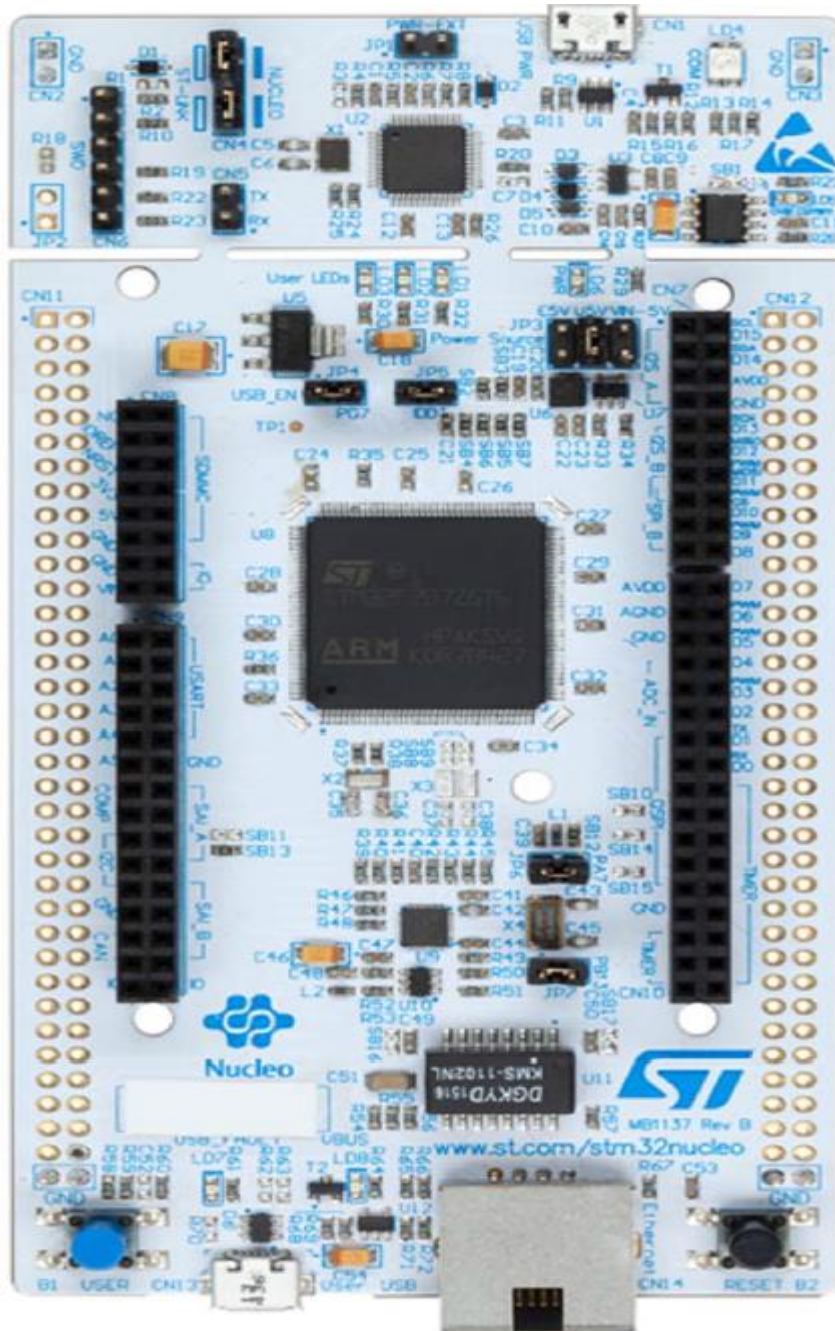


Arquitectura de un sistema embebido

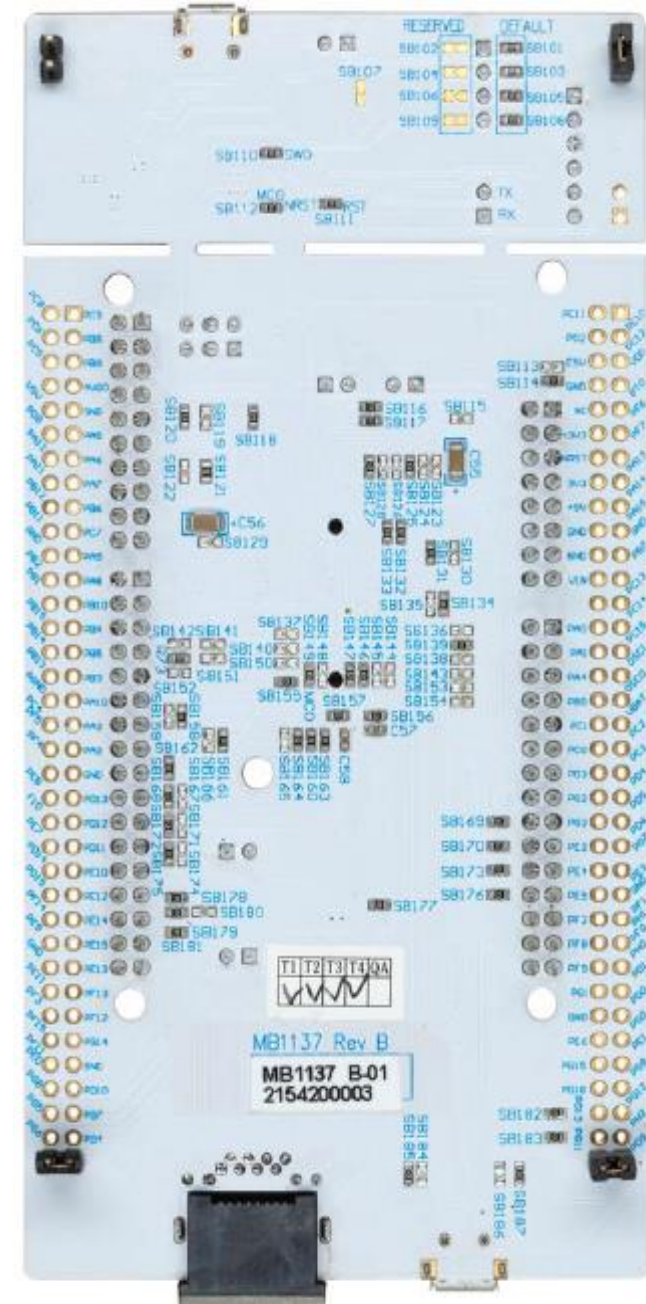
➤ Para un sistema embebido específico, como la STM32F407VG, se encuentran componentes como:

- Microcontrolador ARM 32-bit *Cortex*TM
- Oscilador de 168 MHz
- *Memory Flash* de 1 Mbyte
- SRAM de 192+4 Kbyte
- CCM (Core Coupled Memory) de 64 Kbyte
- Interfaz paralela de LCD
- Interfaz paralela a 8-14 bits de cámara a 54 Mbytes/s
- Unidad de cálculo CRC
- RTC: Precisión inferior a segundos y Calendario Hardware
- 3 conversores A/D de 12 bits
- 2 conversores D/A de 12 bits
- 17 temporizadores, 15 de estos a 16 bits y 2 a 32 bits, con funciones IC/OC/PWM
- 140 I/O con capacidad de interrupción
- 15 interfaces de comunicación
- 4 LEDs de uso general
- Conexiones para periféricos de entrada y salida

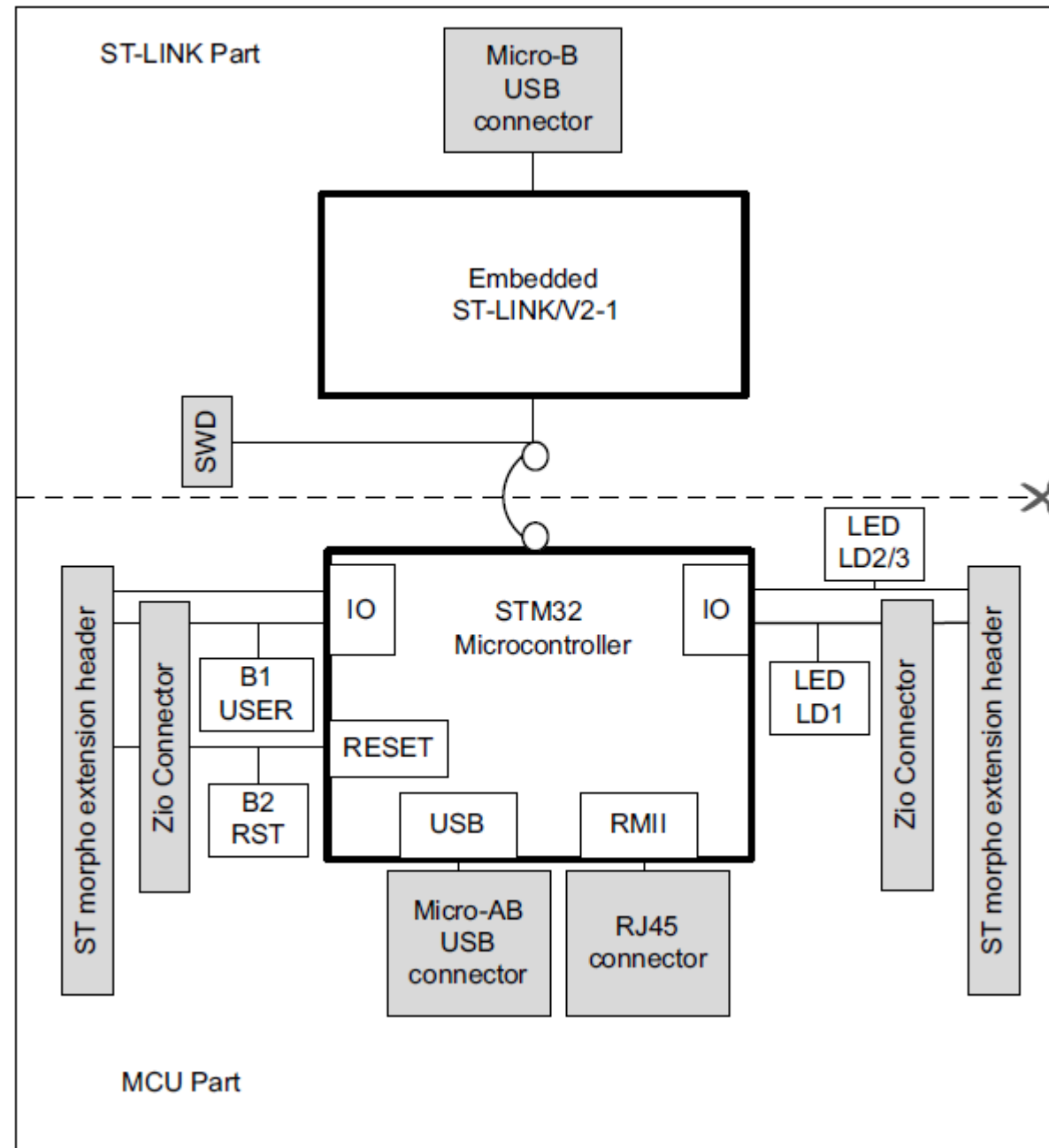


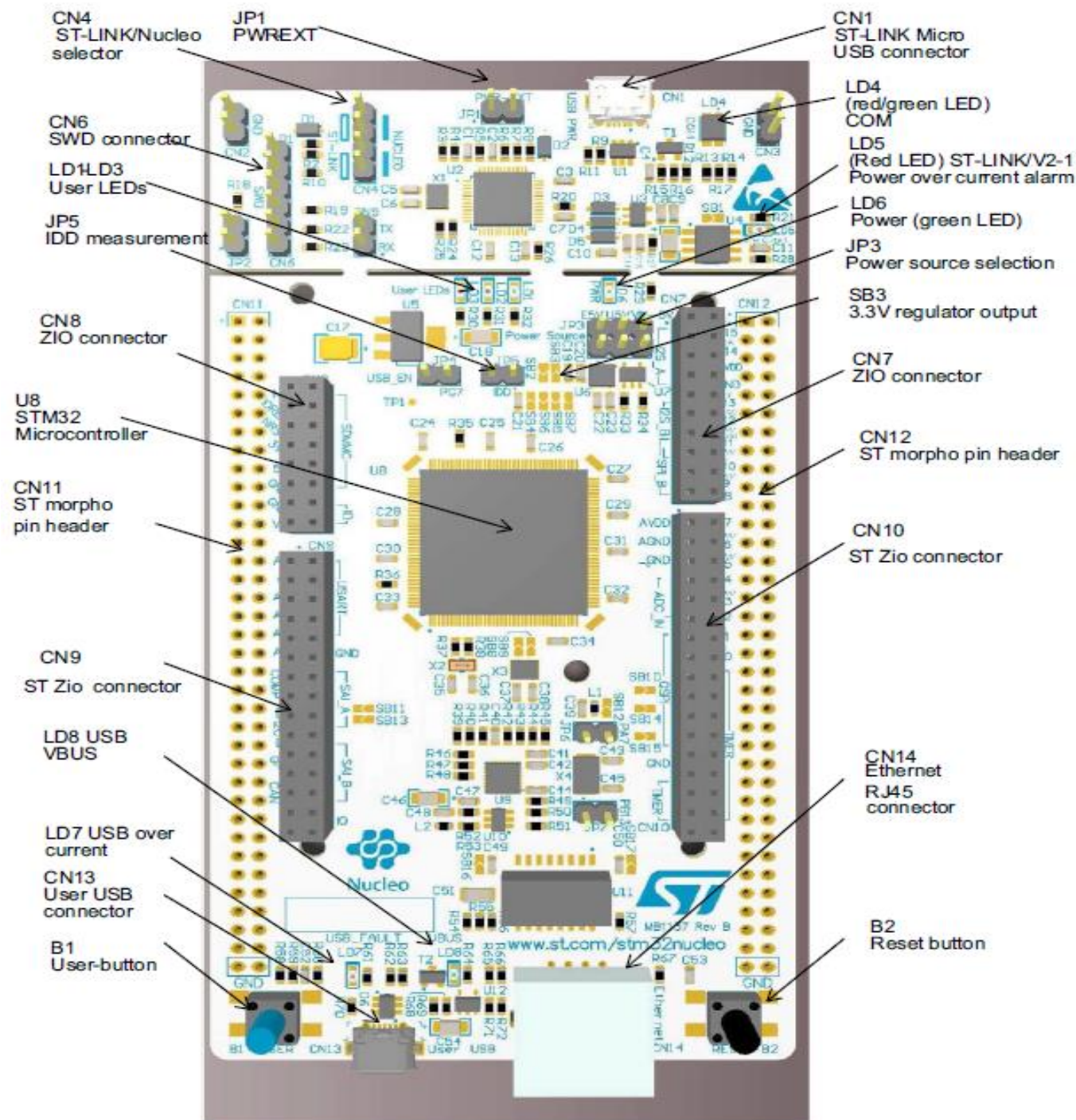


STM32F746



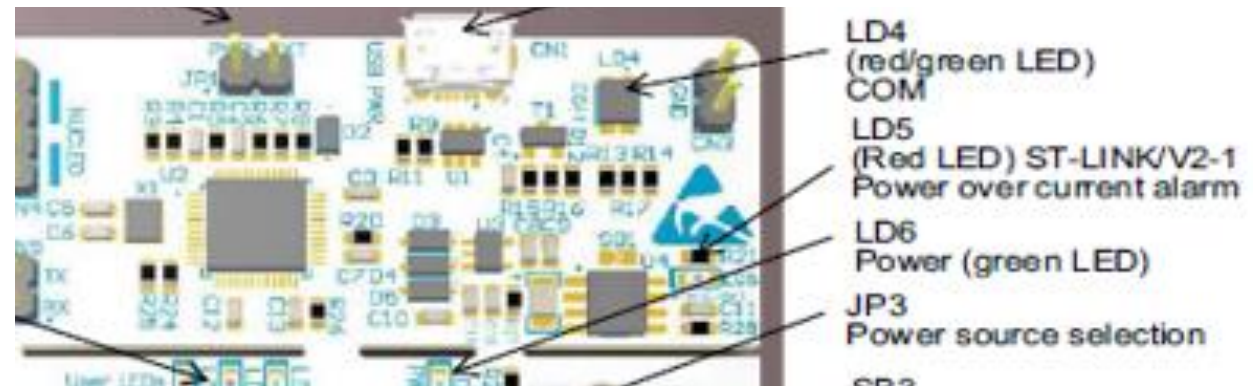






LD4 COM: the tricolor LED LD4 (green, orange, red) provides information about ST-LINK communication status. LD4 default color is red. LD4 turns to green to indicate that communication is in progress between the PC and the ST-LINK/V2-1, with the following setup:

- Slow blinking red/off: at power-on before USB initialization
- Fast blinking red/off: after the first correct communication between PC and ST-LINK/V2-1 (enumeration)
- Red LED on: when the initialization between the PC and ST-LINK/V2-1 is complete
- Green LED on: after a successful target communication initialization
- Blinking red/green: during communication with target
- Green on: communication finished and successful
- Orange on: communication failure



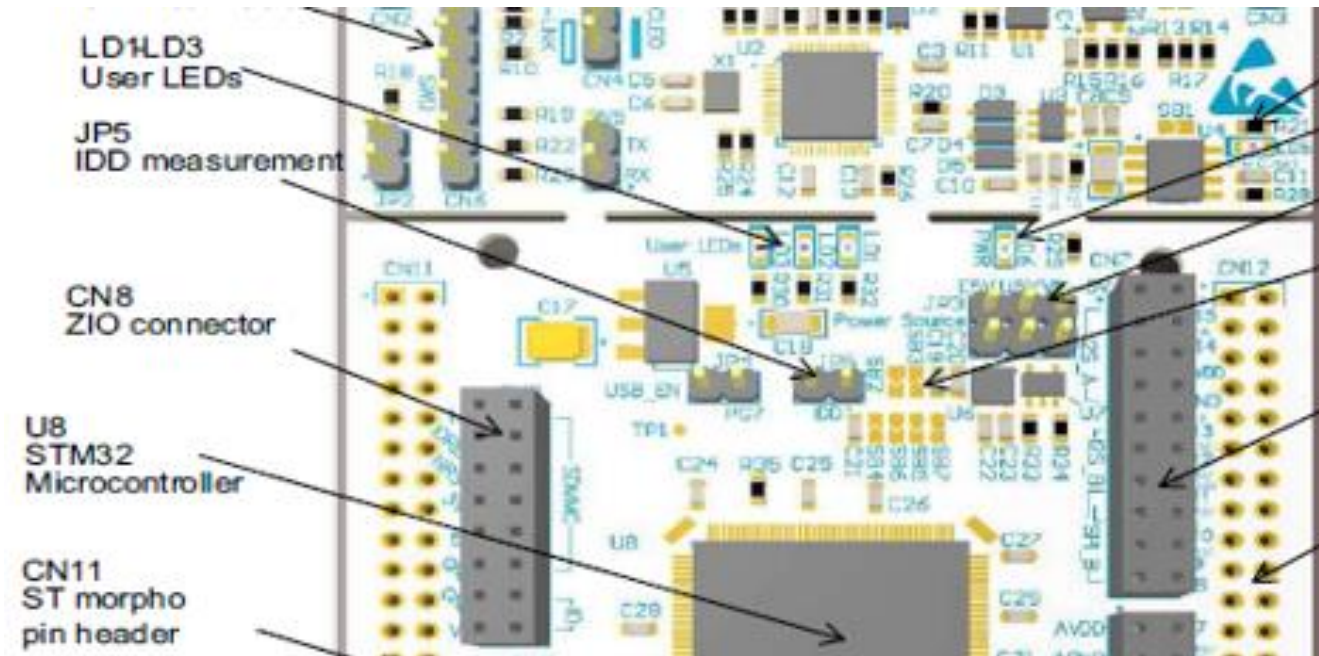
LEDs

User LD1: a green user LED is connected to the STM32 I/O PB0 (SB120 ON and SB119 OFF) or PA5 (SB119 ON and SB120 OFF) corresponding to the ST Zio D13.

User LD2: a blue user LED is connected to PB7.

User LD3: a red user LED is connected to PB14.

These user LEDs are on when the I/O is HIGH value, and are off when the I/O is LOW.

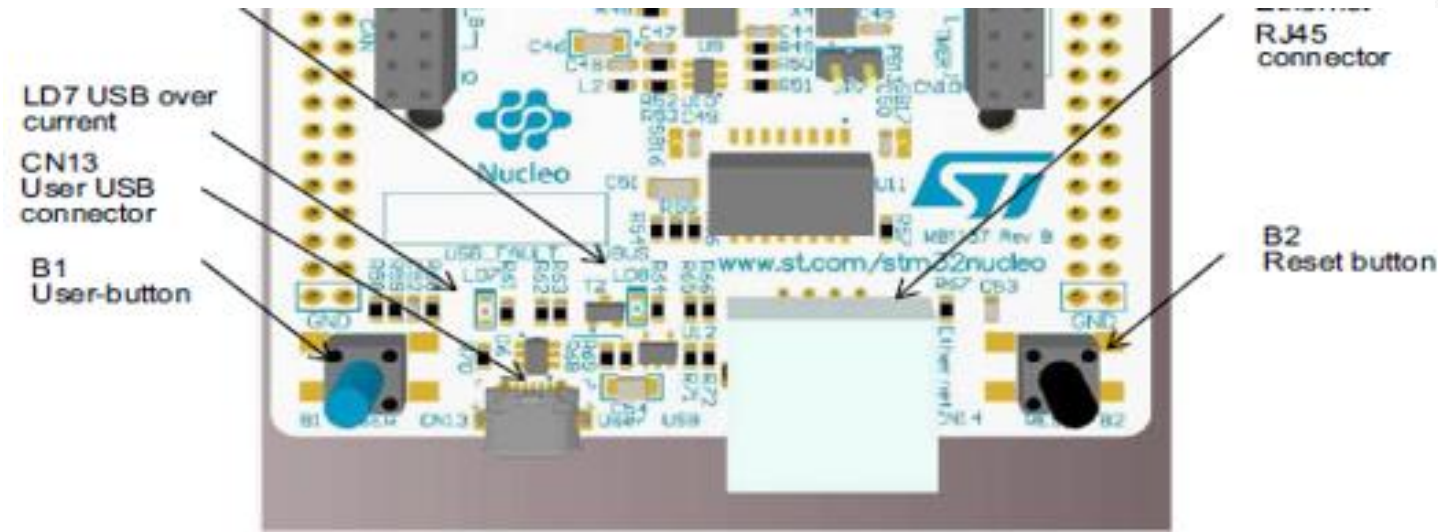


```
main.cpp
1  #include <stdio.h>
2  #include "STM32F7xx.h"
3
4  //BASICO LED 1 EJEMPLO INTRO SALIDA DIGITAL.
5
6  int main(void)
7  {
8      //*****
9      //CONFIGURACION "CLOCK"
10     RCC->AHB1ENR |= (1UL << 1);    //PRENDER EL CLOCK DEL PTB
11
12     //*****
13     //CONFIGURACION DE PINES
14     GPIOB->MODER = 0X555555;    //PTB -> OUTPUT
15
16     //*****
17
18     while(true){                //bucle infinito
19         GPIOB->ODR |= (1UL << 0);    //enciende LED
20     } //cierra while
21
22 } //cierra main
23
```


Push-buttons

B1 USER: the user button is connected to the I/O PC13 by default (Tamper support, SB173 ON and SB180 OFF) or PA0 (Wakeup support, SB180 ON and SB173 OFF) of the STM32 microcontroller.

B2 RESET: this push-button is connected to NRST and is used to RESET the STM32 microcontroller.



```
#include "STM32F7xx.h"

//BASICO LED 1 EJEMPLO ENTRADA SALIDA DIGITAL.

int main(void)
{
    //*****
    //CONFIGURACION "CLOCK"
    RCC->AHB1ENR |= 6;    //PRENDER EL CLOCK DEL PTB

    //*****
    //CONFIGURACION DE PINES
    GPIOB->MODER =1; //PTB -> OUTPUT
    GPIOC->MODER =0;    //PTC -> INPUT
    //*****

    while(true){        //bucle infinito
        //Lee pulsador enciende LED
        while((GPIOC->IDR & 0X2000)==0X2000 ){ GPIOB->ODR |= 1; }
        GPIOB->ODR = 0;
    } //cierra while

} //cierra main
```



GPIO functional description

Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability

Each I/O port bit is freely programmable, however the I/O port registers have to be accessed as 32-bit words, half-words or bytes. The purpose of the GPIOx_BSRR and GPIOx_BRR registers is to allow atomic read/modify accesses to any of the GPIOx_ODR registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.



General-purpose I/O (GPIO)

During and just after reset, the alternate functions are not active and most of the I/O ports are configured in input floating mode.

The debug pins are in AF pull-up/pull-down after reset:

- PA15: JTDI in pull-up
- PA14: JTCK/SWCLK in pull-down
- PA13: JTMS/SWDAT in pull-up
- PB4: NJTRST in pull-up
- PB3: JTDO in floating state

When the pin is configured as output, the value written to the output data register (GPIOx_ODR) is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the low level is driven, high level is HI-Z).

The input data register (GPIOx_IDR) captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors, which can be activated or not depending on the value in the GPIOx_PUPDR register.



Clocks

Three different clock sources can be used to drive the system clock (SYSCLK):

- HSI oscillator clock
- HSE oscillator clock
- Main PLL (PLL) clock

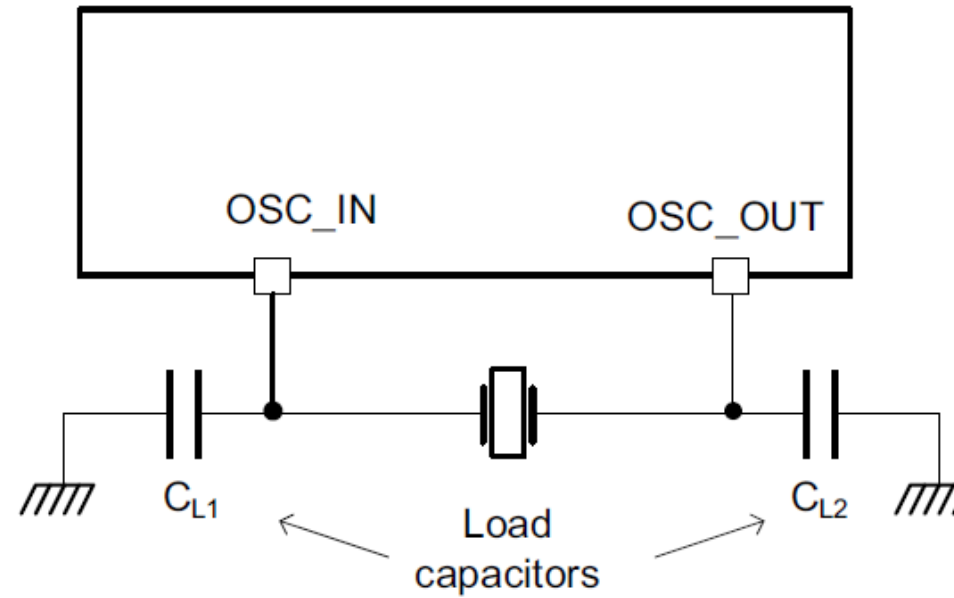
The devices have the two following secondary clock sources:

- 32 kHz low-speed internal RC (LSI RC) which drives the independent watchdog and, optionally, the RTC used for Auto-wakeup from the Stop/Standby mode.
- 32.768 kHz low-speed external crystal (LSE crystal) which optionally drives the RTC clock (RTCCLK)

Each clock source can be switched on or off independently when it is not used, to optimize power consumption.



Crystal/ceramic
resonators



PLL

The devices feature three PLLs:

- A main PLL (PLL) clocked by the HSE or HSI oscillator and featuring two different output clocks:
 - The first output is used to generate the high speed system clock (up to 216 MHz)
 - The second output is used to generate 48MHz clock for the USB OTG FS, SDMMC and RNG.

HSI clock

The HSI clock signal is generated from an internal 16 MHz RC oscillator and can be used directly as a system clock, or used as PLL input.



I/O port control registers

Each of the GPIO ports has four 32-bit memory-mapped control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR) to configure up to 16 I/Os. The GPIOx_MODER register is used to select the I/O mode (input, output, AF, analog). The GPIOx_OTYPER and GPIOx_OSPEEDR registers are used to select the output type (push-pull or open-drain) and speed. The GPIOx_PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.

I/O port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (GPIOx_IDR and GPIOx_ODR). GPIOx_ODR stores the data to be output, it is read/write accessible. The data input through the I/O are stored into the input data register (GPIOx_IDR), a read-only register.



General-purpose I/Os (GPIO)

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_BSRR) for bitwise write access to GPIOx_ODR
- Locking mechanism (GPIOx_LCKR) provided to freeze the I/O port configurations
- Analog function
- Alternate function selection registers
- Fast toggle capable of changing every two clock cycles
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions



5.3.10 RCC AHB1 peripheral clock register (RCC_AHB1ENR)

Address offset: 0x30

Reset value: 0x0010 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
Res.	OTGHS ULPIEN	OTGHS EN	ETHM ACPTP EN	ETHM ACRX EN	ETHM ACTX EN	ETHMA CEN	Res.	DMA2D EN	DMA2 EN	DMA1 EN	DTCMRA MEN	Res.	BKPSR AMEN	Res.	Res.
	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
Res.	Res.	Res.	CRC EN	Res.	GPIOK EN	GPIOJ EN	GPIOI EN	GPIOH EN	GPIOG EN	GPIOF EN	GPIOE EN	GPIOD EN	GPIOC EN	GPIO BEN	GPIO AEN
			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 2 **GPIOCEN**: IO port C clock enable

This bit is set and cleared by software.

0: IO port C clock disabled

1: IO port C clock enabled

Bit 1 **GPIOBEN**: IO port B clock enable

This bit is set and cleared by software.

0: IO port B clock disabled

1: IO port B clock enabled

Bit 0 **GPIOAEN**: IO port A clock enable

This bit is set and cleared by software.

0: IO port A clock disabled

1: IO port A clock enabled



6.4.1 GPIO port mode register (GPIOx_MODER) (x = A..K)

Address offset: 0x00

Reset values:

- 0xA800 0000 for port A
- 0x0000 0280 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[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
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **MODERy[1:0]**: Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O mode.

00: Input mode (reset state)

01: General purpose output mode

10: Alternate function mode

11: Analog mode



6.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A..K)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
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 **OTy**: Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O output type.

0: Output push-pull (reset state)

1: Output open-drain



6.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A..K)

Address offset: 0x08

Reset value:

- 0x0C00 0000 for port A
- 0x0000 00C0 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15 [1:0]		OSPEEDR14 [1:0]		OSPEEDR13 [1:0]		OSPEEDR12 [1:0]		OSPEEDR11 [1:0]		OSPEEDR10 [1:0]		OSPEEDR9 [1:0]		OSPEEDR8 [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
OSPEEDR7 [1:0]		OSPEEDR6 [1:0]		OSPEEDR5 [1:0]		OSPEEDR4 [1:0]		OSPEEDR3 [1:0]		OSPEEDR2 [1:0]		OSPEEDR1 [1:0]		OSPEEDR0 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **OSPEEDRy[1:0]**: Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O output speed.

- 00: Low speed
- 01: Medium speed
- 10: High speed
- 11: Very high speed

Note: Refer to the product datasheets for the values of OSPEEDRy bits versus V_{DD} range and external load.



6.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A..K)

Address offset: 0x0C

Reset values:

- 0x6400 0000 for port A
- 0x0000 0100 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[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
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **PUPDRy[1:0]**: Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O pull-up or pull-down

- 00: No pull-up, pull-down
- 01: Pull-up
- 10: Pull-down
- 11: Reserved



6.4.5 GPIO port input data register (GPIOx_IDR) (x = A..K)

Address offset: 0x10

Reset value: 0x0000 XXXX (where X means undefined)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 15:0 **IDRy**: Port input data bit (y = 0..15)

These bits are read-only. They contain the input value of the corresponding I/O port.



6.4.6 GPIO port output data register (GPIOx_ODR) (x = A..K)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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 bit (y = 0..15)

These bits can be read and written by software.

Note: For atomic bit set/reset, the ODR bits can be individually set and/or reset by writing to the GPIOx_BSRR or GPIOx_BRR registers (x = A..F).



6.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A..K)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFR7[3:0]				AFR6[3:0]				AFR5[3:0]				AFR4[3:0]			
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
AFR3[3:0]				AFR2[3:0]				AFR1[3:0]				AFR0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFRy[3:0]**: Alternate function selection for port x pin y (y = 0..7)

These bits are written by software to configure alternate function I/Os

AFSELY selection:

0000: AF0

0001: AF1

0010: AF2

0011: AF3

0100: AF4

0101: AF5

0110: AF6

0111: AF7

1000: AF8

1001: AF9

1010: AF10

1011: AF11

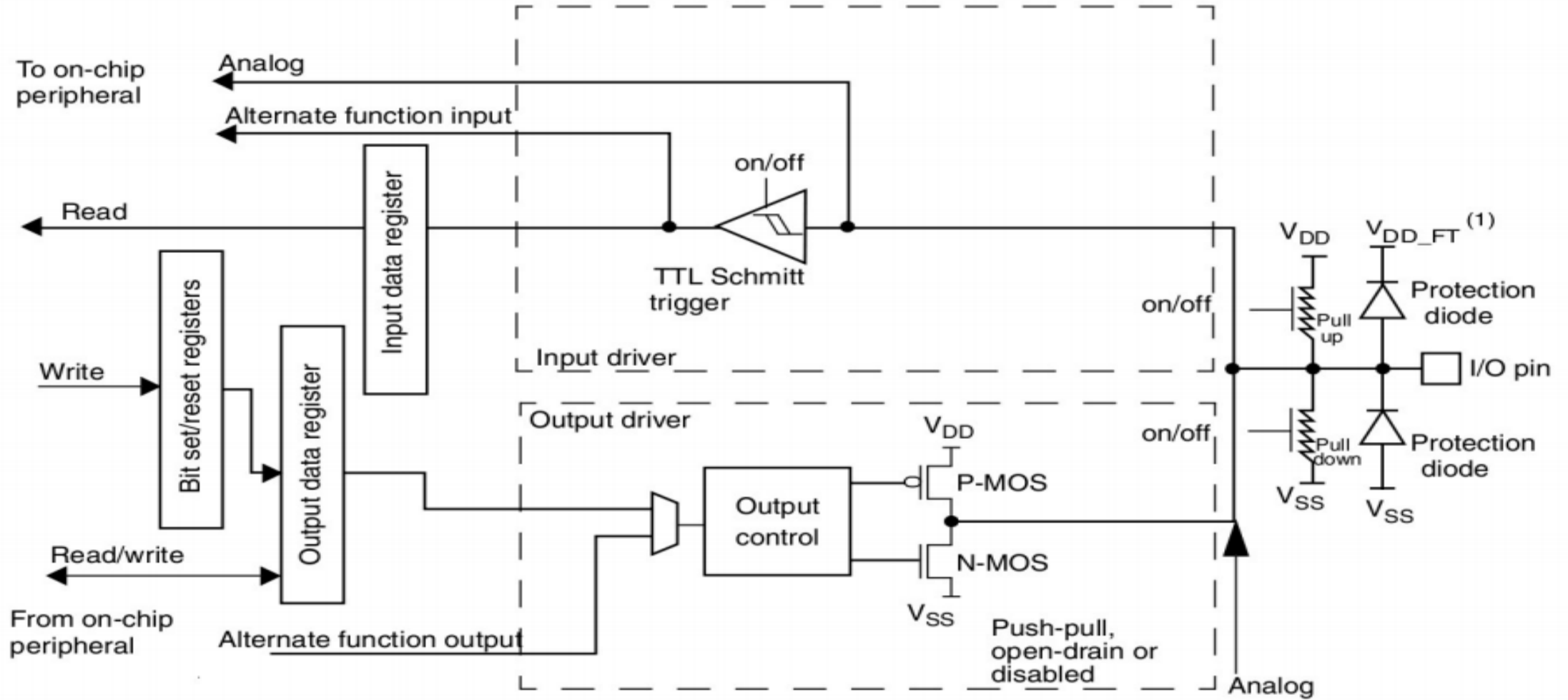
1100: AF12

1101: AF13

1110: AF14

1111: AF15





```
main.cpp
1  #include <stdio.h>
2  #include "stm32f7xx.h"
3
4  int main(void) {
5      RCC -> AHB1ENR = 0X2; //LEDS AHORA EN EL PUERTO B
6      GPIOB -> MODER = 0X10004001; //COLOCAR EN SALIDA PARA ENCENDER LOS LEDS
7      GPIOB -> OTYPER = 0X0; //PUSH PULL
8      GPIOB -> OSPEEDR = 0X10004001; //VELOCIDAD MEDIA
9      GPIOB -> PUPDR = 0X10004001; //PULL UP
10     while(1) {
11         GPIOB -> ODR = 0X4081; //ENCENDER LOS 3 LEDS
12     }
13 }
14
```

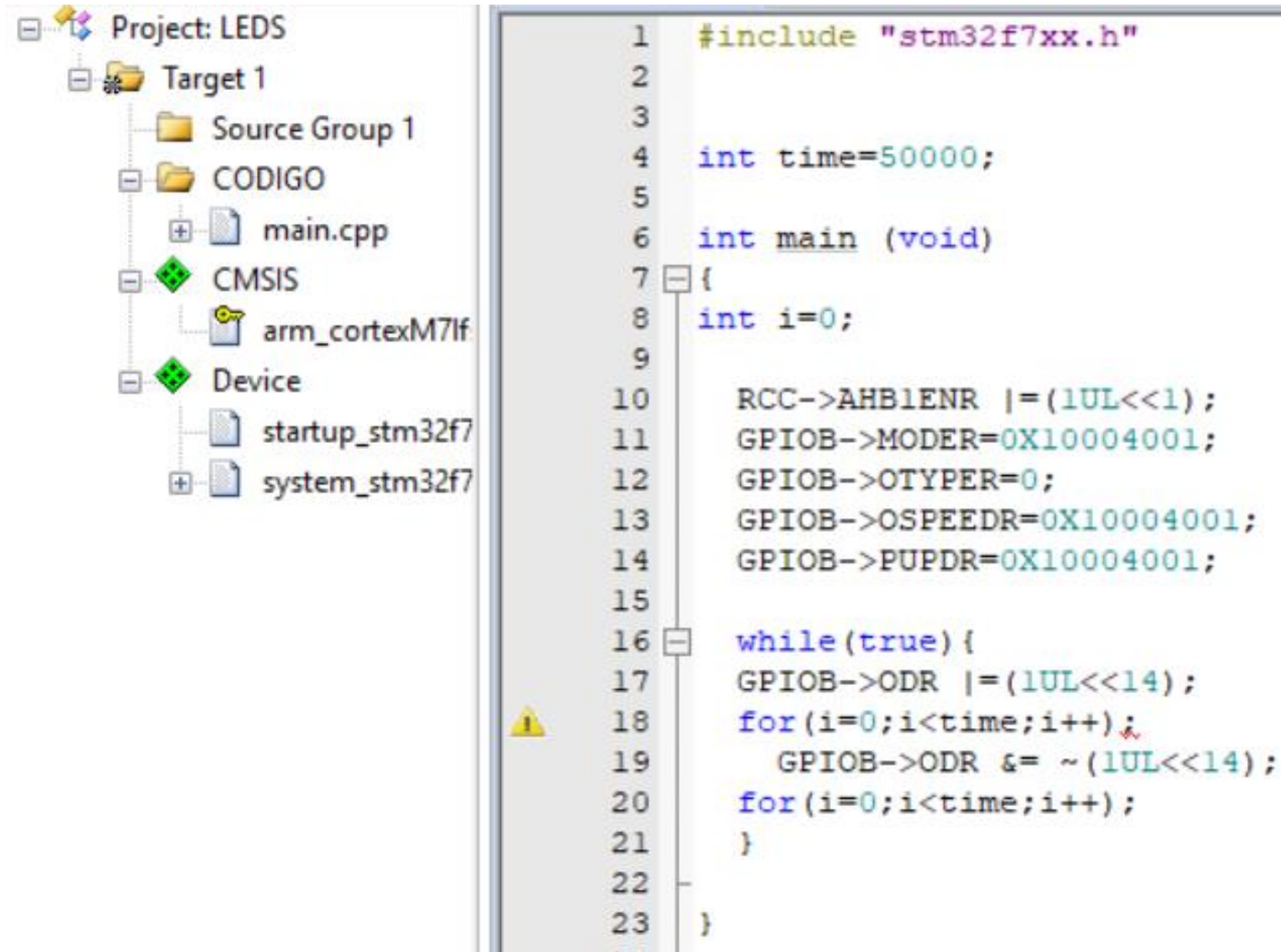


```
1  #include <stdio.h>
2  #include "stm32f7xx.h"
3
4  int main(void){
5      RCC -> AHB1ENR = 0X2; //LEDS AHORA EN EL PUERTO B
6      GPIOB -> MODER = 0X10004001; //COLOCAR EN SALIDA PARA ENCENDER
7      GPIOB -> OYPER = 0X0; //PUSH PULL
8      GPIOB -> OSPEEDR = 0X10004001; //VELOCIDAD MEDIA
9      GPIOB -> PUPDR = 0X10004001; //PULL UP
10     while(1){
11         GPIOB -> ODR = 0X4081; //ENCENDER LOS 3 LEDS
12         GPIOB -> ODR = (0UL << 0); //ENCENDER LOS 3 LEDS
13     }
14 }
15
```




```
2  #include<stm32f7xx.h>
3  int x,y,z;
4  int main(void)
5
6  {
7      RCC->AHB1ENR=9;
8      GPIOD->MODER=0X55550000;
9
10     while (1)
11     {
12
13         if(      ((GPIOA->IDR) & 0X01) ==1  &  y==0)
14         {
15             y=1;
16             GPIOD->ODR=~GPIOD->ODR;
17         }
18
19         if(      ((GPIOA->IDR) & 0X01) ==0  )
20         {
21             y=0;
22             for(z=0;  z<10000;  z++) ;
23
24         }
25
26     }
27 }
```





The screenshot shows an IDE with a project named 'LEDS'. The left sidebar displays the project structure:

- Project: LEDS
 - Target 1
 - Source Group 1
 - CODIGO
 - main.cpp
 - CMSIS
 - arm_cortexM7lf
 - Device
 - startup_stm32f7
 - system_stm32f7

The main editor displays the following C code:

```
1  #include "stm32f7xx.h"
2
3
4  int time=50000;
5
6  int main (void)
7  {
8      int i=0;
9
10     RCC->AHB1ENR |= (1UL<<1);
11     GPIOB->MODER=0X10004001;
12     GPIOB->OTYPER=0;
13     GPIOB->OSPEEDR=0X10004001;
14     GPIOB->PUPDR=0X10004001;
15
16     while (true) {
17         GPIOB->ODR |= (1UL<<14);
18         for(i=0;i<time;i++);
19         GPIOB->ODR &= ~(1UL<<14);
20         for(i=0;i<time;i++);
21     }
22
23 }
```

```

//////////SECUENCIA LEDS 2//////////

#include <stdio.h>
#include "stm32f7xx.h"

int time=0;
int sec[16]={1,2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,16384,32768};

int main(void){
    RCC -> AHB1ENR = 0xff; //LEDS AHORA EN EL PUERTO B
    GPIOG -> MODER = 0X55555555; //COLOCAR EN SALIDA PARA ENCENDER LOS LEDS
    GPIOG -> OSPEEDR = 1; //VELOCIDAD MEDIA
    GPIOG -> PUPDR = 0; // NO PULL UP
    while(1){

        for(int a=0;a<17;a++){
            GPIOG -> ODR = sec[a]; //
            for(time=0;time<50000;time++);}
        }
    }
}

```

TAREA:

Reproducir la secuencia del video agregando al menos dos Secuencias y dos tiempos de variación.

